# Package 'eyeprocess'

October 22, 2026

**Type** Package

**Title** Harmonize Eye-Tracking, Pupillometry, Biometrics, and Psychometric Process Data

**Version** 0.12.1

**Description** Provides an extensible, vendor-neutral framework for importing, validating, harmonizing, transforming, visualizing, and modelling eye-tracking, pupillometry, behavioural, and biometric process data. The package uses explicit timebase and coordinate-space registries, preserves native fields and provenance, and offers first-class adapters for Gazepoint Analysis and Gazepoint Biometrics exports alongside generic and vendor-specific importers.
Downstream tools support trial and area of interest reconstruction, signal-quality auditing, feature derivation, scanpath analysis, response-time and item-response workflows, and optional psychometric modelling engines. An integrated Gazepoint workflow produces quality-control evidence, media-trial reconstruction, plots, analysis-ready process tables, item response theory (IRT)-ready response structures, and reproducible reports. Brain Imaging Data Structure (BIDS) interoperability for eye-tracking and validation-release infrastructure support disk-backed storage, independent multi-vendor evidence, grouped validation, simulation calibration, model-equivalence audits, and explicitly experimental advanced psychometric process models. Research-scale infrastructure adds deterministic resumable Monte Carlo execution, atomic validation checkpoints, explicit advanced-model promotion gates, independent multi-vendor evidence registries, stable object contracts, partitioned disk-backed storage, optional probabilistic engines, and a fully synthetic multimodal benchmark for reproducibility testing. The measurement-intelligence programme adds probabilistic and compositional area of interest (AOI) analysis, measurement-uncertainty propagation, calibration and device-transportability audits, process reliability, phase-amplitude pupil registration,

informative-missingness sensitivity, right-censored gaze-latency survival analysis, temporal and spatial process models, item-bank decision optimization, fairness monitoring, conditional process reference distributions, and evidence-provenance graphs.

**License** MIT + file LICENSE

**URL** https://stefanosbalaskas.github.io/eyeprocess/, https://github.com/stefanosbalaskas/eyeprocess

**BugReports** https://github.com/stefanosbalaskas/eyeprocess/issues

**Encoding** UTF-8

**Depends** R (>= 4.1.0)

**Imports** graphics, grDevices, methods, splines, stats, utils, withr

**Suggests** brms, ggplot2, jsonlite, knitr, lme4, LNIRT, mirt, rmarkdown, TAM, arrow, diffIRT, GDINA, OpenMx, TraMineR, testthat (>= 3.0.0), callr, cmdstanr, eyetrackingR, future, future.apply, openssl, PupillometryR, rtdists, seqHMM, LSMjml, MASS, mgcv, nnet, survival, coxme, eRm, FactoMineR, mice, missForest, plm, psychotree, ranger, robfilter, tidyLPA, loo, posterior, targets, equateIRT, catR, mirtCAT

**Additional_repositories** https://stan-dev.r-universe.dev

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Config/roxygen2/version** 8.0.0

**NeedsCompilation** no

**Author** Stefanos Balaskas [aut, cre] (ORCID: <https://orcid.org/0000-0003-2444-9796>)

**Maintainer** Stefanos Balaskas <s.balaskas@ac.upatras.gr>

**Repository** CRAN

**Date/Publication** 2026-10-22 11:40:02 UTC

# Contents

eyeprocess-package . . . . . 23
ablate_multimodal_channels . . . . . 23
addm_glam_proxy_features . . . . . 24
adjust_pupil_confounds . . . . . 25
advanced_model_evidence_spec . . . . . 25
advanced_validation_grid . . . . . 26
algorithm_facet_effects . . . . . 27
analysis_decision_entropy . . . . . 27
analysis_environment_snapshot . . . . . 28
analysis_resolution_guard . . . . . 28

aoi_membership_probability . . . 29
aoi_perturbation_uncertainty . . . 30
aoi_trajectory_features . . . 34
api_family_map . . . 35
api_lifecycle_diff . . . 36
api_surface_summary . . . 36
apply_preflight_decision . . . 37
apply_synthetic_corruption . . . 37
assign_aois_probabilistic . . . 38
assign_process_feature_family . . . 40
as_irt_recovery_results . . . 40
audit_3pl_process_signatures . . . 41
audit_advanced_model_evidence . . . 41
audit_benchmark_release . . . 42
audit_bias . . . 42
audit_biometric_imputation . . . 43
audit_biometric_preflight . . . 43
audit_candidate_item_bank . . . 44
audit_channel_incremental_information . . . 45
audit_convergence . . . 46
audit_coverage . . . 47
audit_decision_provenance . . . 47
audit_distractor_attention . . . 48
audit_eye_api . . . 49
audit_eye_pipeline . . . 49
audit_frontier_model_contract . . . 50
audit_identifiability . . . 50
audit_interval_width . . . 51
audit_irf_shape . . . 52
audit_item_reduction_sensitivity . . . 52
audit_latent_distribution . . . 53
audit_measurement_transportability . . . 54
audit_model_promotion . . . 54
audit_multimodal_identifiability . . . 55
audit_multimodal_m2_identifiability . . . 56
audit_multimodal_m3_identifiability . . . 56
audit_multimodal_m4_identifiability . . . 57
audit_multimodal_measurement . . . 58
audit_multivariate_process_quality . . . 59
audit_nonparametric_rasch . . . 59
audit_presentation_accessibility . . . 60
audit_process_adjusted_dif . . . 61
audit_process_anomalies . . . 62
audit_process_drift . . . 62
audit_process_external_validity . . . 63
audit_process_local_dependence . . . 64
audit_process_measurement_invariance . . . 65
audit_process_window_sensitivity . . . 65

audit_pupil_fatigue_drift . . . 66
audit_pupil_frequency_stability . . . 67
audit_pupil_preprocessing_order . . . 68
audit_rmse . . . 68
audit_roundtrip_loss . . . 69
audit_sampling_irregularity . . . 69
audit_sbc . . . 70
audit_signal_filter . . . 71
audit_temporal_leakage . . . 71
audit_validation_completion . . . 72
audit_vendor_field_coverage . . . 72
audit_vendor_validation . . . 73
audit_visual_context_dependence . . . 73
bayesian_process_diagnostics_dashboard . . . 74
bayesian_process_diagnostic_flags . . . 74
benchmark_expected_outputs . . . 75
benchmark_eyeprocess . . . 76
benchmark_eye_storage . . . 76
benchmark_memory_estimate . . . 77
benchmark_scaling_curve . . . 77
bind_process_windows . . . 78
biometric_imputation_sensitivity . . . 78
bootstrap_process_reliability . . . 79
build_compatibility_matrix . . . 80
build_evidence_graph . . . 80
build_gazepoint_media_trials . . . 82
calibration_drift_profile . . . 82
calibration_error_model . . . 83
calibration_sensitivity_grid . . . 84
calibration_transfer_audit . . . 84
canonical_eye_api . . . 85
classify_item_missingness . . . 85
collect_eye_storage . . . 86
collect_validation_evidence . . . 86
collect_validation_jobs . . . 87
compare_aoi_methods . . . 87
compare_aoi_trajectories . . . 88
compare_bayesian_process_models . . . 88
compare_decision_manifests . . . 89
compare_deployment_batches . . . 89
compare_diffusion_accuracy_rt . . . 90
compare_dynamic_transition_models . . . 91
compare_engine_adapters . . . 91
compare_fixation_methods . . . 92
compare_functional_scalar_models . . . 92
compare_hard_probabilistic_aoi . . . 93
compare_irt_models . . . 94
compare_latent_distribution_models . . . 94

compare_model_engines . . . 95
compare_parametric_nonparametric_irf . . . 95
compare_presentation_fairness . . . 96
compare_process_criterion_models . . . 97
compare_process_models . . . 97
compare_process_profile_solutions . . . 98
compare_pupil_kernels . . . 98
compare_pupil_preprocessing . . . 99
compare_raw_adjusted_pupil . . . 99
compare_reproducibility_fingerprints . . . 100
compare_signal_filters . . . 100
compare_strategy_heterogeneity . . . 101
compare_validation_engines . . . 101
compare_vendor_semantics . . . 102
compare_visual_context_irt . . . 102
compatibility_evidence_matrix . . . 103
context_factor_effects . . . 103
coordinate_fidelity_audit . . . 104
coverage_calibration_curve . . . 105
create_public_benchmark . . . 105
crossed_grouped_cv . . . 106
crossed_grouped_folds . . . 107
cross_device_process_equating_audit . . . 107
cross_version_adapter_regression . . . 108
data_quality_reporting_table . . . 109
decision_manifest_diff . . . 109
decision_manifest_hash . . . 110
decision_manifest_table . . . 110
decision_space_coverage . . . 111
decision_stability . . . 111
decode_dynamic_states . . . 112
derive_aoi_composition . . . 112
derive_gazepoint_workflow_features . . . 114
detect_calibration_drift . . . 115
detect_corrupt_partitions . . . 116
detect_irt_changepoints . . . 117
detect_process_changepoint . . . 118
detect_process_changepoints . . . 118
device_facet_effects . . . 120
diffusion_identification_study . . . 120
diffusion_parameter_diagnostics . . . 121
diffusion_posterior_predictive . . . 122
distractor_process_map . . . 122
drift_by_device . . . 123
drift_by_site . . . 123
drift_by_stimulus_version . . . 124
drift_by_vendor . . . 124
dynamic_irtree_recovery . . . 125

dynamic_irtree_spec . . . 125
dynamic_posterior_predictive_check . . . 127
dynamic_transition_design . . . 128
effective_sampling_frequency . . . 128
encode_response_combinations . . . 129
engine_adapter_status . . . 130
equate_irt_scales . . . 130
estimate_calibration_error . . . 131
estimate_visual_exposure_probability . . . 132
evaluate_validation_acceptance . . . 132
event_detector_multiverse . . . 133
event_marker_qc . . . 137
event_roundtrip_audit . . . 138
event_semantics_audit . . . 138
expand_eyeprocess_stress_evidence_plan . . . 139
expand_eyeprocess_validation_plan . . . 140
expand_process_validation_design . . . 140
expected_process_information . . . 141
explain_latent_interaction . . . 141
export_eye_bids . . . 142
export_eye_pipeline . . . 143
export_prov_json . . . 143
export_ro_crate_metadata . . . 144
export_validation_bundle . . . 144
external_eye_adapters . . . 145
external_model_engines . . . 146
external_validate_irt . . . 146
extract_diffusion_parameters . . . 147
extract_functional_pupil_parameters . . . 147
extract_parameter_truth . . . 148
extract_process_windows . . . 148
eyeprocess-adapters . . . 149
eyeprocess-class . . . 150
eyeprocess-coordinates-time . . . 150
eyeprocess-experimental . . . 151
eyeprocess-export-report . . . 152
eyeprocess-features . . . 152
eyeprocess-format-validation . . . 153
eyeprocess-import-gazepoint . . . 154
eyeprocess-import-generic . . . 154
eyeprocess-import-vendors . . . 155
eyeprocess-models . . . 155
eyeprocess-plots . . . 156
eyeprocess-preprocessing . . . 157
eyeprocess-quality . . . 157
eyeprocess-schema . . . 158
eyeprocess-simulation . . . 158
eyeprocess-trials-aoi . . . 159

eyeprocess_api_version . . . 159
eyeprocess_benchmark_study . . . 160
eyeprocess_cdm_attribute_profiles . . . 160
eyeprocess_cdm_classification_uncertainty . . . 161
eyeprocess_cdm_dina_ideal_response . . . 161
eyeprocess_cdm_dina_probability . . . 162
eyeprocess_cdm_qmatrix_audit . . . 162
eyeprocess_deprecation . . . 163
eyeprocess_irt_2pl_probability . . . 163
eyeprocess_irt_3pl_probability . . . 164
eyeprocess_irt_4pl_probability . . . 165
eyeprocess_irt_adaptive_trace . . . 165
eyeprocess_irt_anchor_audit . . . 166
eyeprocess_irt_anchor_purification . . . 167
eyeprocess_irt_apply_link . . . 167
eyeprocess_irt_bank_coverage . . . 168
eyeprocess_irt_category_function_audit . . . 169
eyeprocess_irt_classification_precision . . . 169
eyeprocess_irt_conditional_sem . . . 170
eyeprocess_irt_content_balance_audit . . . 170
eyeprocess_irt_device_drift . . . 171
eyeprocess_irt_dif_effect_curve . . . 171
eyeprocess_irt_dtf_curve . . . 172
eyeprocess_irt_eap_score . . . 173
eyeprocess_irt_engine_evidence_table . . . 173
eyeprocess_irt_engine_registry . . . 174
eyeprocess_irt_engine_status . . . 174
eyeprocess_irt_expected_score . . . 175
eyeprocess_irt_exposure_summary . . . 175
eyeprocess_irt_extreme_score_audit . . . 176
eyeprocess_irt_fit_dashboard . . . 176
eyeprocess_irt_functioning_effect_summary . . . 177
eyeprocess_irt_gpcm_probability . . . 178
eyeprocess_irt_grm_probability . . . 178
eyeprocess_irt_haebara_link . . . 179
eyeprocess_irt_identification_audit . . . 179
eyeprocess_irt_infit_outfit . . . 180
eyeprocess_irt_information_area . . . 181
eyeprocess_irt_information_gain . . . 181
eyeprocess_irt_information_targeting . . . 182
eyeprocess_irt_invariance_evidence . . . 182
eyeprocess_irt_item_bank . . . 183
eyeprocess_irt_item_fit_residuals . . . 184
eyeprocess_irt_item_information . . . 184
eyeprocess_irt_item_selection . . . 185
eyeprocess_irt_latent_regression_design . . . 186
eyeprocess_irt_link_stability . . . 186
eyeprocess_irt_local_dependence_pairs . . . 187

eyeprocess_irt_map_score . . . 188
eyeprocess_irt_marginal_reliability . . . 188
eyeprocess_irt_mean_mean_link . . . 189
eyeprocess_irt_mean_sigma_link . . . 190
eyeprocess_irt_measurement_precision_profile . . . 190
eyeprocess_irt_missing_by_design_audit . . . 191
eyeprocess_irt_misspecification_metrics . . . 192
eyeprocess_irt_misspecification_suite . . . 192
eyeprocess_irt_mle_score . . . 193
eyeprocess_irt_model_card . . . 193
eyeprocess_irt_model_card_audit . . . 194
eyeprocess_irt_model_spec . . . 195
eyeprocess_irt_monotonicity_audit . . . 196
eyeprocess_irt_nominal_probability . . . 196
eyeprocess_irt_parameter_plausibility_audit . . . 197
eyeprocess_irt_person_fit_lz . . . 197
eyeprocess_irt_person_fit_residuals . . . 198
eyeprocess_irt_plausible_values . . . 199
eyeprocess_irt_ppc_discrepancy . . . 199
eyeprocess_irt_precision_evidence_table . . . 200
eyeprocess_irt_prior_sensitivity_grid . . . 200
eyeprocess_irt_prior_sensitivity_summary . . . 201
eyeprocess_irt_prior_spec . . . 202
eyeprocess_irt_process_alignment . . . 203
eyeprocess_irt_process_aware_selection_penalty . . . 203
eyeprocess_irt_process_dif_concordance . . . 204
eyeprocess_irt_q3 . . . 205
eyeprocess_irt_recovery_design . . . 205
eyeprocess_irt_recovery_failures . . . 206
eyeprocess_irt_recovery_summary . . . 207
eyeprocess_irt_sbc_ranks . . . 207
eyeprocess_irt_sbc_summary . . . 208
eyeprocess_irt_score_table . . . 208
eyeprocess_irt_score_uncertainty . . . 209
eyeprocess_irt_session_drift . . . 210
eyeprocess_irt_sparse_design_audit . . . 210
eyeprocess_irt_stocking_lord_link . . . 211
eyeprocess_irt_stopping_rule . . . 212
eyeprocess_irt_targeting_gap . . . 213
eyeprocess_irt_testlet_audit . . . 213
eyeprocess_irt_testlet_spec . . . 214
eyeprocess_irt_test_characteristic_curve . . . 214
eyeprocess_irt_test_information . . . 215
eyeprocess_irt_threshold_order_audit . . . 215
eyeprocess_joint_process_irt_spec . . . 216
eyeprocess_mirt_directional_information . . . 217
eyeprocess_mirt_information_matrix . . . 217
eyeprocess_mirt_loading_audit . . . 218

eyeprocess_mirt_loading_spec . . . 219
eyeprocess_multichannel_measurement_map . . . 219
eyeprocess_negative_control_evidence_plan . . . 220
eyeprocess_negative_control_evidence_table . . . 221
eyeprocess_process_irt_data_bundle . . . 221
eyeprocess_process_item_profile . . . 222
eyeprocess_process_missingness_pattern . . . 223
eyeprocess_process_person_profile . . . 223
eyeprocess_recovery_evidence_table . . . 224
eyeprocess_reliability_evidence_plan . . . 224
eyeprocess_reliability_evidence_table . . . 225
eyeprocess_response_time_profile . . . 225
eyeprocess_sbc_evidence_table . . . 226
eyeprocess_speed_accuracy_profile . . . 226
eyeprocess_stress_evidence_plan . . . 227
eyeprocess_stress_evidence_table . . . 228
eyeprocess_validation_atlas_gaps . . . 228
eyeprocess_validation_claim_matrix . . . 229
eyeprocess_validation_evidence_atlas . . . 229
eyeprocess_validation_evidence_grade . . . 230
eyeprocess_validation_evidence_index . . . 231
eyeprocess_validation_evidence_manifest . . . 231
eyeprocess_validation_plan . . . 232
eyeprocess_validation_readiness . . . 233
eyeprocess_validation_release_gate . . . 234
eyeprocess_validation_seed . . . 234
eye_analysis_pipeline . . . 235
eye_analysis_spec . . . 235
eye_api_inventory . . . 236
eye_api_lifecycle . . . 237
eye_api_recommendation . . . 237
eye_api_status . . . 238
eye_api_superseded . . . 238
eye_benchmark_design . . . 239
eye_decision_manifest . . . 239
eye_pipeline_dot . . . 240
eye_pipeline_graph . . . 241
eye_pipeline_manifest . . . 241
eye_pipeline_mermaid . . . 242
eye_pipeline_step . . . 242
eye_plot_spec . . . 243
eye_prov_graph . . . 244
eye_reproducibility_fingerprint . . . 245
eye_session_manifest . . . 246
eye_storage_spec . . . 246
eye_stream_fidelity_audit . . . 247
eye_targets_manifest . . . 248
facet_effects . . . 249

field_fidelity_report . . . 249
file_hash_manifest . . . 250
filter_eye_signal . . . 251
filter_pupil_signal . . . 251
find_process_measures . . . 252
fingerprint_validation_case . . . 252
fit_aoi_growth_curve . . . 253
fit_brms_adapter . . . 254
fit_censored_normal_process_irt . . . 254
fit_changepoint_multimodal_irt . . . 255
fit_changepoint_rt_irt . . . 256
fit_cognitive_diagnosis_process . . . 256
fit_continuous_time_irt . . . 257
fit_crossclassified_process_irt . . . 258
fit_crossclassified_process_irt_mhrm . . . 259
fit_device_linking . . . 259
fit_diffirt_adapter . . . 261
fit_diffirt_engine_adapter . . . 261
fit_dynamic_gpirt . . . 262
fit_dynamic_irtree . . . 262
fit_dynamic_irtree_stan . . . 263
fit_event_time_irt . . . 263
fit_external_engine . . . 264
fit_eyeprocess_erm . . . 265
fit_eyeprocess_gdina . . . 266
fit_eyeprocess_lnirt . . . 266
fit_eyeprocess_mirt . . . 267
fit_eyeprocess_tam . . . 267
fit_eyetrackingr_adapter . . . 268
fit_fixation_point_process . . . 269
fit_flow_mirt . . . 270
fit_functional_pupil_stan . . . 271
fit_gaze_anchored_3pl_audit . . . 271
fit_gaze_diffusion_irt . . . 272
fit_gaze_diffusion_stan . . . 273
fit_gaze_informed_missingness_irt . . . 273
fit_gdina_adapter . . . 274
fit_gpirt . . . 275
fit_irt_model . . . 276
fit_item_parameter_seed_model . . . 276
fit_joint_functional_pupil_irt . . . 277
fit_joint_gaze_rt_irt . . . 278
fit_joint_graded_rt_process_irt . . . 279
fit_kde_latent_distribution_irt . . . 280
fit_latent_class_process_irt . . . 280
fit_latent_space_irt . . . 281
fit_lnirt_adapter . . . 282
fit_manyfacet_process_irt . . . 283

fit_mirt_adapter . . . 284
fit_mixture_irt_process_classes . . . 284
fit_multiblock_process_map . . . 285
fit_multimodal_m2 . . . 286
fit_multimodal_m3 . . . 287
fit_multimodal_m4 . . . 288
fit_multimodal_trait_irt . . . 290
fit_multinomial_transition . . . 291
fit_multiple_response_process_irt . . . 292
fit_nominal_gaze_irt . . . 293
fit_nonignorable_missing_irt . . . 294
fit_omission_survival_irt . . . 294
fit_openmx_adapter . . . 295
fit_openmx_process_model . . . 296
fit_persistence_gaze_diffusion_irt . . . 297
fit_process_dif . . . 297
fit_process_gstudy . . . 299
fit_process_hmm_irt . . . 300
fit_process_missingness_model . . . 301
fit_process_norms . . . 302
fit_process_observation_model . . . 304
fit_process_profile_mixture . . . 305
fit_process_rasch_tree . . . 306
fit_pupillometryr_adapter . . . 306
fit_pupil_confound_model . . . 307
fit_pupil_event_deconvolution . . . 308
fit_response_process_embedding_irt . . . 309
fit_revisit_process_cdm . . . 310
fit_seqhmm_adapter . . . 311
fit_speed_accuracy_engagement_irt . . . 311
fit_strategy_mixture_em . . . 312
fit_strategy_mixture_stan . . . 312
fit_tam_adapter . . . 313
fit_theory_strategy_irt . . . 314
fit_traminer_adapter . . . 314
fit_validation_replicate . . . 315
fit_variational_irt . . . 316
fit_visual_context_irt . . . 316
fixation_boundary_uncertainty . . . 317
freeze_eyeprocess_irt_reference . . . 318
freeze_eyeprocess_validation_atlas . . . 318
freeze_eyeprocess_validation_evidence . . . 319
freeze_software_paper_evidence . . . 320
freeze_validation_reference . . . 320
functional_pupil_basis . . . 321
functional_pupil_diagnostics . . . 322
functional_pupil_irt_spec . . . 322
gaze-survival . . . 324

gazepoint_analysis_tables . . . 327
gazepoint_irt_tables . . . 328
gazepoint_workflow_spec . . . 328
gaze_anchored_3pl_alignment . . . 330
gaze_data_quality_profile . . . 330
gaze_diffusion_spec . . . 331
gaze_precision_rms_s2s . . . 332
gaze_recurrence . . . 333
gaze_uncertainty_ellipse . . . 334
generalizability_process_study . . . 335
get_irt_model . . . 335
grade_model_evidence . . . 336
grouped_cv . . . 336
grouped_folds . . . 337
import_benchmark_study . . . 338
import_eye_bids . . . 338
incremental_process_validity . . . 339
init_vendor_corpus . . . 339
inject_aoi_label_noise . . . 340
inject_calibration_offset . . . 340
inject_device_shift . . . 341
inject_eye_missingness . . . 342
inject_pupil_dropout . . . 342
inject_sampling_jitter . . . 343
inject_trial_imbalance . . . 343
irt_compositional_channel . . . 344
irt_continuous_channel . . . 344
irt_count_channel . . . 345
irt_functional_channel . . . 346
irt_model_spec . . . 346
irt_nominal_channel . . . 347
irt_response_channel . . . 348
irt_rt_channel . . . 349
irt_sequence_channel . . . 349
irt_survival_channel . . . 350
irt_validation_spec . . . 351
item_objective_spec . . . 352
latent_distribution_stress_test . . . 354
latent_trait_trajectory . . . 354
leave_device_out_validation . . . 355
leave_item_out_validation . . . 355
leave_session_out_validation . . . 356
leave_site_out_validation . . . 356
list_irt_models . . . 357
lock_decision_manifest . . . 357
map_latent_classes_to_process_profiles . . . 358
measurement_error_budget . . . 359
migrate_eye_storage_schema . . . 359

model_promotion_spec . . . 360
model_validation_spec . . . 361
model_validation_summary . . . 362
multiblock_contributions . . . 362
multiblock_person_coordinates . . . 363
multiblock_variable_coordinates . . . 363
multilevel-mediation . . . 364
multimodal_backend_status . . . 366
multimodal_irt_spec . . . 367
multimodal_m2_ablation . . . 368
multimodal_m2_negative_controls . . . 368
multimodal_m2_ppc . . . 369
multimodal_m2_process_information . . . 369
multimodal_m2_recovery . . . 370
multimodal_m2_spec . . . 371
multimodal_m3_ablation . . . 371
multimodal_m3_functional_bridge . . . 372
multimodal_m3_negative_controls . . . 373
multimodal_m3_ppc . . . 373
multimodal_m3_process_information . . . 374
multimodal_m3_recovery . . . 374
multimodal_m3_spec . . . 375
multimodal_m4_ablation . . . 376
multimodal_m4_negative_controls . . . 377
multimodal_m4_ppc . . . 377
multimodal_m4_process_information . . . 378
multimodal_m4_recovery . . . 379
multimodal_m4_sensitivity . . . 380
multimodal_m4_spec . . . 380
multimodal_m4_state_diagnostics . . . 381
multimodal_ppc . . . 382
negative_control_concordance . . . 382
negative_control_process_test . . . 383
object_hash . . . 384
object_schema . . . 384
open_eye_storage . . . 385
open_partitioned_eye_storage . . . 385
option_process_information . . . 386
outcome_blind_feature_audit . . . 386
outcome_blind_snapshot . . . 387
package_reproducibility_manifest . . . 387
paper_reproducibility_manifest . . . 388
partition_eye_storage . . . 388
pipeline_failures . . . 389
pipeline_result . . . 389
pipeline_step_status . . . 390
placebo_window_audit . . . 390
plot.eye_adapter_regression_audit . . . 391

plot.eye_aoi_growth_curve . . . 391
plot.eye_aoi_trajectory . . . 392
plot.eye_bayesian_process_dashboard . . . 392
plot.eye_bids_roundtrip . . . 393
plot.eye_biometric_imputation_sensitivity . . . 393
plot.eye_biometric_preflight . . . 394
plot.eye_candidate_item_bank_audit . . . 394
plot.eye_compatibility_evidence_matrix . . . 395
plot.eye_decision_process_proxy . . . 395
plot.eye_dynamic_irtree . . . 396
plot.eye_dynamic_ppc . . . 396
plot.eye_event_roundtrip_audit . . . 397
plot.eye_event_time_irt . . . 397
plot.eye_functional_pupil_diagnostics . . . 398
plot.eye_functional_pupil_irt . . . 398
plot.eye_functional_pupil_sensitivity . . . 399
plot.eye_gated_process_model . . . 399
plot.eye_gaze_anchored_3pl_audit . . . 400
plot.eye_gaze_informed_missingness_irt . . . 400
plot.eye_gpirt . . . 401
plot.eye_incremental_information_audit . . . 401
plot.eye_irt_changepoints . . . 402
plot.eye_irt_equating . . . 402
plot.eye_irt_ppc . . . 403
plot.eye_irt_recovery_summary . . . 403
plot.eye_irt_sbc . . . 404
plot.eye_item_parameter_seed . . . 405
plot.eye_item_reduction_sensitivity . . . 405
plot.eye_joint_gaze_rt_irt . . . 406
plot.eye_joint_graded_rt_process_irt . . . 406
plot.eye_latent_distribution_comparison . . . 407
plot.eye_latent_process_alignment . . . 407
plot.eye_latent_space_irt . . . 408
plot.eye_manyfacet_process_irt . . . 408
plot.eye_mixture_irt_process . . . 409
plot.eye_model_promotion_audit . . . 409
plot.eye_multiblock_process_map . . . 410
plot.eye_nominal_gaze_irt . . . 410
plot.eye_nonparametric_rasch_audit . . . 411
plot.eye_omission_survival_irt . . . 411
plot.eye_preaction_process_features . . . 412
plot.eye_presentation_accessibility . . . 412
plot.eye_presentation_fairness_comparison . . . 413
plot.eye_process_anomaly_audit . . . 413
plot.eye_process_cat_simulation . . . 414
plot.eye_process_channel_ablation . . . 414
plot.eye_process_dependent_discrimination . . . 415
plot.eye_process_drift_audit . . . 415

plot.eye_process_external_validity . . . 416
plot.eye_process_facet_effects . . . 417
plot.eye_process_feature_blocks . . . 417
plot.eye_process_g_study . . . 418
plot.eye_process_hmm_irt . . . 418
plot.eye_process_local_dependence_audit . . . 419
plot.eye_process_negative_control . . . 419
plot.eye_process_person_fit . . . 420
plot.eye_process_profile_mixture . . . 420
plot.eye_process_rasch_tree . . . 421
plot.eye_process_windows . . . 421
plot.eye_process_window_sensitivity . . . 422
plot.eye_pupil_confound_model . . . 422
plot.eye_pupil_deconvolution . . . 423
plot.eye_pupil_fatigue_drift . . . 423
plot.eye_pupil_frequency_features . . . 424
plot.eye_pupil_frequency_stability . . . 424
plot.eye_roundtrip_loss_audit . . . 425
plot.eye_sbc_audit . . . 425
plot.eye_semantic_roundtrip . . . 426
plot.eye_signal_filter_audit . . . 426
plot.eye_storage_benchmark . . . 427
plot.eye_strategy_aoi_sensitivity . . . 427
plot.eye_streaming_score . . . 428
plot.eye_transition_diagnostics . . . 428
plot.eye_validation_bundle . . . 429
plot.eye_vendor_compatibility_matrix . . . 429
plot.eye_vendor_semantic_validation . . . 430
plot.eye_visual_context_irt . . . 430
plot_aoi_transition_matrix . . . 431
plot_aoi_transition_rank . . . 431
plot_distractor_information . . . 432
plot_gazepoint_workflow . . . 433
plot_interval_coverage . . . 433
plot_irf_uncertainty . . . 434
plot_parameter_recovery . . . 434
plot_person_item_space . . . 435
plot_process_changepoint . . . 436
plot_process_channel_ablation_delta . . . 436
plot_process_feature_stability . . . 437
plot_process_window_sensitivity . . . 438
plot_pupil_activity_sensitivity . . . 438
plot_pupil_activity_windows . . . 439
plot_pupil_band_power . . . 439
plot_pupil_components . . . 440
plot_pupil_preprocessing_audit . . . 440
plot_pupil_spectrum . . . 441
plot_sbc_rank . . . 442

plot_validation_failures . . . 442
plot_validation_runtime . . . 443
posterior_predictive_discrepancies . . . 443
posterior_sbc_contract . . . 444
preaction_process_features . . . 444
predict.eye_censored_normal_process_irt . . . 445
predict.eye_multinomial_transition . . . 446
predict_aoi_trajectory . . . 446
predict_item_parameter_priors . . . 447
predict_theta_at_time . . . 447
preflight_decisions . . . 448
preflight_exclusion_manifest . . . 448
preflight_failures . . . 449
preflight_passed . . . 449
prepare_dynamic_irtree_data . . . 450
prepare_functional_pupil_data . . . 450
prepare_gaze_diffusion_data . . . 451
prepare_multimodal_irt_data . . . 452
prepare_strategy_mixture_data . . . 453
prepare_structured_unstructured_process_features . . . 453
preprocessing_multiverse . . . 454
print.eye_benchmark_reproduction . . . 455
print.eye_benchmark_study . . . 455
print.eye_benchmark_validation . . . 456
print.eye_diffusion_diagnostics . . . 456
print.eye_diffusion_identification_study . . . 457
print.eye_dynamic_irtree . . . 457
print.eye_dynamic_ppc . . . 458
print.eye_dynamic_recovery . . . 458
print.eye_engine_adapter_result . . . 459
print.eye_functional_pupil_data . . . 459
print.eye_functional_pupil_diagnostics . . . 460
print.eye_functional_pupil_irt . . . 460
print.eye_functional_pupil_sensitivity . . . 461
print.eye_functional_scalar_comparison . . . 461
print.eye_gated_process_model . . . 462
print.eye_gaze_diffusion_data . . . 462
print.eye_gaze_diffusion_irt . . . 463
print.eye_gaze_diffusion_spec . . . 463
print.eye_irt_evidence_grade . . . 464
print.eye_irt_model_spec . . . 464
print.eye_irt_validation_spec . . . 465
print.eye_model_contract_validation . . . 465
print.eye_model_promotion_audit . . . 466
print.eye_multinomial_transition . . . 466
print.eye_partitioned_storage . . . 467
print.eye_partition_spec . . . 467
print.eye_process_drift_spec . . . 468

print.eye_process_negative_control . . . 468
print.eye_process_preflight_spec . . . 469
print.eye_process_window_spec . . . 469
print.eye_redaction_result . . . 470
print.eye_roundtrip_loss_audit . . . 470
print.eye_strategy_data . . . 471
print.eye_strategy_manipulation_validation . . . 471
print.eye_theory_strategy_irt . . . 472
print.eye_theory_strategy_spec . . . 472
print.eye_transition_design . . . 473
print.eye_transition_diagnostics . . . 473
print.eye_validation_bundle . . . 474
print.eye_validation_case_fingerprint . . . 474
print.eye_validation_collection . . . 475
print.eye_validation_completion_audit . . . 475
print.eye_validation_job_plan . . . 476
print.eye_validation_run . . . 476
print.eye_vendor_case . . . 477
print.eye_vendor_compatibility_matrix . . . 477
print.eye_vendor_schema_contract . . . 478
print.eye_vendor_semantic_comparison . . . 478
print.eye_visual_context_registry . . . 479
probabilistic_aoi_assignment . . . 479
process_anomaly_distance . . . 480
process_bland_altman . . . 481
process_channel_ablation . . . 481
process_criterion_associations . . . 482
process_dependent_discrimination_audit . . . 483
process_dif_nuisance_surrogate . . . 484
process_drift_alerts . . . 484
process_drift_spec . . . 485
process_feature_blocks . . . 486
process_feature_family_registry . . . 486
process_feature_stability . . . 487
process_feature_time_provenance . . . 487
process_icc . . . 488
process_information . . . 489
process_item_information . . . 489
process_measure_card . . . 490
process_measure_coverage . . . 491
process_measure_guardrails . . . 491
process_measure_lineage . . . 492
process_measure_registry . . . 492
process_measure_units . . . 493
process_negative_control_permute . . . 493
process_negative_control_shift . . . 494
process_ngram_features . . . 495
process_null_benchmark . . . 495

process_person_fit . . . 496
process_preflight_spec . . . 496
process_profile_probabilities . . . 498
process_profile_summary . . . 498
process_reliability_profile . . . 499
process_residual_map . . . 499
process_sensitivity_grid . . . 500
process_sequence_embedding . . . 500
process_state_occupancy . . . 501
process_state_transition_summary . . . 501
process_temporal_stability . . . 502
process_uncertainty_spec . . . 503
process_validation_design . . . 504
process_window_spec . . . 505
promote_irt_model . . . 506
promote_vendor_support . . . 507
propagate_calibration_uncertainty . . . 507
provenance_edge_table . . . 508
provenance_lineage_table . . . 509
prune_validation_checkpoints . . . 509
public_validation_corpus . . . 510
pupil_activity_index . . . 510
pupil_band_power . . . 511
pupil_baseline_sensitivity . . . 512
pupil_confound_effects . . . 512
pupil_event_effects . . . 513
pupil_event_regressor . . . 513
pupil_frequency_features . . . 514
pupil_latency_sensitivity . . . 515
pupil_preprocessing_grid . . . 516
pupil_preprocessing_sensitivity . . . 516
pupil_response_kernel . . . 517
pupil_unit_fidelity_audit . . . 518
pupil_velocity_activity . . . 519
quantify_process_leakage . . . 519
query_eye_storage . . . 520
raven_reproduction_spec . . . 520
read_api_lifecycle_registry . . . 521
read_benchmark_table . . . 522
read_decision_manifest . . . 522
read_eyeprocess_validation_evidence . . . 523
read_reproducibility_fingerprint . . . 523
read_validation_job_manifest . . . 524
read_validation_scenario_manifest . . . 524
read_vendor_registry . . . 525
recalibrate_after_changepoint . . . 525
recommended_validation_replications . . . 526
redact_validation_case . . . 527

register_eye_api_status . . . 527
register_irt_model . . . 528
register_process_measure . . . 529
register_pupil_curves . . . 529
register_validation_case . . . 531
register_vendor_semantics . . . 532
reporting_guideline_audit . . . 533
representative_scanpath . . . 534
resume_eye_pipeline . . . 535
resume_validation_jobs . . . 536
roundtrip_eye_bids . . . 536
run_benchmark_reproduction . . . 537
run_eyeprocess_equateirt . . . 538
run_eyeprocess_irt_ability_sbc . . . 538
run_eyeprocess_irt_recovery . . . 539
run_eyeprocess_mirtcat . . . 540
run_eyeprocess_stress_evidence . . . 540
run_eyeprocess_validation_program . . . 541
run_eye_benchmark . . . 542
run_eye_pipeline . . . 543
run_gazepoint_workflow . . . 544
run_model_validation . . . 545
run_posterior_sbc . . . 546
run_process_negative_controls . . . 546
run_process_sensitivity . . . 547
run_process_validation . . . 548
run_raven_reproduction . . . 549
run_sbc . . . 549
run_validation_jobs . . . 550
sbc_ecdf_deviation . . . 551
sbc_rank_diagnostics . . . 552
sbc_summary . . . 552
score_partial_response_pattern . . . 553
score_response_stream . . . 553
select_next_item_process . . . 554
semantic_fidelity_spec . . . 555
semantic_loss_map . . . 556
semantic_roundtrip_audit . . . 556
sensitivity_branch_fingerprint . . . 557
sensitivity_decision_leverage . . . 558
sensitivity_fragility_index . . . 558
sensitivity_multiverse_manifest . . . 559
sensitivity_rank_stability . . . 559
sensitivity_significance_stability . . . 560
sensitivity_sign_stability . . . 560
sensitivity_threshold_stability . . . 561
sequence_interoperability . . . 561
session_facet_effects . . . 562

simulate_advanced_process_data . . . 563
simulate_dynamic_irtree_data . . . 564
simulate_eyeprocess_catr . . . 565
simulate_eyeprocess_irt_binary . . . 565
simulate_from_model . . . 566
simulate_gaze_diffusion_data . . . 567
simulate_irt_model . . . 568
simulate_multimodal_irt . . . 568
simulate_multimodal_m2 . . . 569
simulate_multimodal_m3 . . . 570
simulate_multimodal_m4 . . . 572
simulate_presentation_variants . . . 573
simulate_process_cat . . . 573
simulate_process_validation_data . . . 574
simulate_strategy_mixture_data . . . 575
simulation_based_calibration . . . 575
simulation_rank_statistic . . . 576
software_paper_claim_matrix . . . 577
software_paper_coverage . . . 578
software_paper_evidence_bundle . . . 578
software_paper_gap_analysis . . . 579
software_paper_readiness . . . 580
software_paper_validation_table . . . 581
specification_coverage . . . 581
specification_curve_data . . . 582
split_half_process_reliability . . . 582
split_validation_plan . . . 583
standardized_spatial_quality . . . 584
storage_transaction_manifest . . . 588
strategy_aoi_sensitivity . . . 588
strategy_classification_uncertainty . . . 589
strategy_label_switching_diagnostics . . . 589
strategy_posterior_probabilities . . . 590
streaming_score_history . . . 590
stress_test_latent_distribution . . . 591
stress_test_local_dependence . . . 591
stress_test_missingness . . . 592
stress_test_misspecification . . . 593
stress_test_preprocessing . . . 593
stress_test_process_pipeline . . . 594
stress_test_speededness . . . 595
stress_test_summary . . . 595
stress_tolerance_frontier . . . 596
structural_transition_mask . . . 596
summarise_eyeprocess_stress_evidence . . . 597
summarise_eye_benchmark . . . 598
summarise_process_negative_controls . . . 598
summarise_process_sensitivity . . . 599

summarise_process_validation . . . 599
summarise_validation_acceptance . . . 600
summarize_parameter_recovery . . . 600
summarize_process_windows . . . 601
summary.eye_validation_bundle . . . 601
summary.eye_validation_job_plan . . . 602
synthetic_corruption_plan . . . 602
theory_strategy_spec . . . 603
timestamp_fidelity_audit . . . 605
transition_residual_diagnostics . . . 606
update_person_score . . . 606
upgrade_eyeprocess_model . . . 607
upgrade_eye_dataset . . . 608
validate_against_reference . . . 608
validate_benchmark_study . . . 609
validate_bids_eye_semantics . . . 609
validate_decision_manifest . . . 610
validate_engine_adapter . . . 610
validate_eyeprocess_external_irt_fit . . . 611
validate_eyeprocess_irt_item_bank . . . 611
validate_eyeprocess_irt_model_spec . . . 612
validate_eyeprocess_joint_process_irt_spec . . . 612
validate_eyeprocess_validation_plan . . . 613
validate_eye_pipeline . . . 613
validate_eye_prov_graph . . . 614
validate_eye_storage_metadata . . . 614
validate_feature_availability . . . 615
validate_gazepoint_workflow . . . 615
validate_hed_event_semantics . . . 616
validate_irt_model . . . 616
validate_latent_space_process_similarity . . . 617
validate_model_object . . . 617
validate_multimodal_irt . . . 618
validate_multimodal_m2 . . . 618
validate_multimodal_m3 . . . 619
validate_multimodal_m4 . . . 619
validate_process_measure_registry . . . 620
validate_process_validation_design . . . 621
validate_process_windows . . . 621
validate_strategy_manipulation . . . 622
validate_vendor_semantics . . . 622
validate_vendor_timestamp_semantics . . . 623
validation_acceptance_matrix . . . 624
validation_acceptance_rule . . . 624
validation_bundle_manifest . . . 625
validation_calibration_summary . . . 626
validation_condition_id . . . 626
validation_condition_ranking . . . 627

validation_coverage_table . . . 627
validation_evidence_levels . . . 628
validation_evidence_matrix . . . 628
validation_failure_profile . . . 629
validation_failure_summary . . . 629
validation_failure_taxonomy . . . 630
validation_job_plan . . . 630
validation_ladder . . . 631
validation_mcse . . . 631
validation_mcse_profile . . . 632
validation_recovery_summary . . . 632
validation_recovery_table . . . 633
validation_replication_budget . . . 633
validation_report . . . 634
validation_robustness_score . . . 634
validation_runtime_summary . . . 635
validation_sbc_summary . . . 635
validation_scenario_manifest . . . 636
validation_seed . . . 637
validation_summary_mcse . . . 637
validation_thresholds . . . 638
vendor_schema_contract . . . 639
vendor_validation_spec . . . 640
verify_decision_manifest_lock . . . 641
verify_eyeprocess_validation_atlas . . . 641
verify_eyeprocess_validation_evidence . . . 642
verify_outcome_blind_snapshot . . . 642
verify_reproducibility_fingerprint . . . 643
verify_reproducibility_manifest . . . 643
visual_context_registry . . . 644
write_advanced_model_evidence_report . . . 644
write_api_lifecycle_registry . . . 645
write_benchmark_data_dictionary . . . 646
write_decision_manifest . . . 646
write_eyeprocess_validation_evidence . . . 647
write_eyeprocess_validation_report . . . 647
write_eye_pipeline_report . . . 648
write_eye_storage . . . 648
write_eye_targets_template . . . 649
write_gazepoint_workflow_report . . . 650
write_model_promotion_report . . . 650
write_partitioned_eye_storage . . . 651
write_prov_dot . . . 651
write_reporting_guideline_report . . . 652
write_reproducibility_fingerprint . . . 652
write_software_paper_evidence . . . 653
write_software_paper_reproduction . . . 653
write_software_paper_scaffold . . . 654

write_validation_job_manifest . . . . . 654
write_validation_release_report . . . . . 655
write_validation_report . . . . . 655
write_validation_scenario_manifest . . . . . 656
write_vendor_case_report . . . . . 656
write_vendor_registry . . . . . 657
write_vendor_validation_report . . . . . 657

**Index** **659**

---

`eyeprocess-package` *eyeprocess: Harmonize Eye-Tracking and Psychometric Process Data*

---

## Description

An extensible, vendor-neutral framework for importing, validating, harmonizing, transforming, visualizing, and modelling eye-tracking, pupillometry, behavioural, and biometric process data, with first-class Gazepoint support.

## Details

The package represents heterogeneous exports as linked canonical tables with explicit timebase, coordinate-space, processing-level, quality, and provenance information. It supports generic mappings and dedicated adapters, then provides common downstream workflows for preprocessing, AOIs, trials, features, plots, and optional psychometric modelling.

## Author(s)

Stefanos Balaskas

## References

Package documentation and the methodological references cited in the vignettes.

---

`ablate_multimodal_channels`
*Create formal channel-ablation datasets*

---

## Description

Create formal channel-ablation datasets

## Usage

```
ablate_multimodal_channels(
  x,
  include = c("response", "rt", "gaze", "pupil"),
  include_response = TRUE
)
```

## Arguments

| | |
|---|---|
| x | An 'eye_multimodal_measurement'. |
| include | Character vector of channels eligible for ablation. |
| include_response | Whether response must remain in every scenario. |

## Value

An 'eye_multimodal_ablation' list.

---

# addm_glam_proxy_features

*Compute aDDM/GLAM-inspired gaze-evidence proxy features*

---

## Description

Compute aDDM/GLAM-inspired gaze-evidence proxy features

## Usage

```
addm_glam_proxy_features(
  data,
  by = c("person_id", "trial_id"),
  time = "time_ms",
  aoi = "aoi",
  target_aoi = "target",
  distractor_aoi = "distractor",
  action_aoi = "button"
)
```

## Arguments

| | |
|---|---|
| data | Sample-level data. |
| by | Grouping columns. |
| time, aoi | Columns. |
| target_aoi, distractor_aoi, action_aoi | AOI labels. |

## Value

An object of class "eye_decision_process_proxy", stored as a named list, with components "features", "by", "status", "caveat". It contains aDDM/GLAM-inspired gaze-evidence proxy features and associated metadata or diagnostics needed to interpret the result.

---

`adjust_pupil_confounds` *Extract confound-adjusted pupil values*

---

## Description

Extract confound-adjusted pupil values

## Usage

```
adjust_pupil_confounds(x)
```

## Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |

## Value

A data frame containing the fitted model data, including ‘pupil_confound_residual‘ and ‘pupil_confound_adjusted‘ columns.

---

`advanced_model_evidence_spec` *Specify evidence required to promote advanced model interfaces*

---

## Description

Specify evidence required to promote advanced model interfaces

## Usage

```
advanced_model_evidence_spec(
  models = c("fit_joint_process_model", "fit_shared_process_factor",
    "fit_strategy_mixture", "fit_process_irt", "fit_pupil_informed_irt",
    "fit_multimodal_irt", "fit_dynamic_aoi_model", "fit_gaze_weighted_choice",
   "fit_dynamic_irtree", "fit_joint_functional_pupil_irt", "fit_theory_strategy_irt",
    "fit_gaze_diffusion_irt"),
  require_recovery = TRUE,
  require_calibration = TRUE,
  require_misspecification = TRUE,
```

```
  require_grouped_validation = TRUE,
  require_engine_equivalence = TRUE,
  require_empirical_reproduction = TRUE,
  require_sensitivity = TRUE
)
```

## Arguments

| | |
|---|---|
| models | Advanced model function names. |
| require_recovery | Require passing parameter-recovery evidence. |
| require_calibration | Require simulation-based calibration evidence. |
| require_misspecification | Require evidence that prespecified failure scenarios are detected. |
| require_grouped_validation | Require grouped out-of-sample validation. |
| require_engine_equivalence | Require comparison with a benchmark engine. |
| require_empirical_reproduction | Require a licensed empirical reproduction. |
| require_sensitivity | Require a multi-specification preprocessing/AOI sensitivity analysis. |

## Value

An ‘eye_advanced_evidence_spec‘.

---

# advanced_validation_grid

*Construct the advanced-model validation grid*

---

## Description

By default this returns a one-factor-at-a-time screening design around a declared reference scenario. This preserves every factor and level from the research programme without accidentally launching hundreds of thousands of Monte Carlo scenarios. Set ‘full_factorial = TRUE‘ only when the computing plan explicitly supports the complete Cartesian design.

## Usage

```
advanced_validation_grid(quick = FALSE, full_factorial = FALSE)
```

## Arguments

| | |
|---|---|
| quick | Whether to return a compact smoke-test design. |
| full_factorial | Whether to return the complete Cartesian design. |

### Value

A scenario data frame for ‘run_model_validation()‘ or custom Monte Carlo programmes.

---

`algorithm_facet_effects` *Extract algorithm facet effects*

---

### Description

Extract algorithm facet effects

### Usage

```
algorithm_facet_effects(object, channel = c("response", "process"))
```

### Arguments

| | |
|---|---|
| `object` | A fitted eyeprocess model or audit object. |
| `channel` | Measurement channel to inspect. |

### Value

An object of class "eye_process_facet_effects", stored as a named list, with components "facet", "column", "channel", "random_effects", "variance_component". It contains algorithm facet effects and associated metadata or diagnostics needed to interpret the result.

---

`analysis_decision_entropy` *Entropy of an explicitly enumerated analysis-decision space*

---

### Description

Entropy of an explicitly enumerated analysis-decision space

### Usage

```
analysis_decision_entropy(..., base = 2)
```

### Arguments

| | |
|---|---|
| `...` | Named option vectors. |
| `base` | Logarithm base. |

### Value

Data frame with option counts and maximum entropy under equal weighting.

## analysis_environment_snapshot

*Snapshot an eyeprocess analysis environment*

### Description

Snapshot an eyeprocess analysis environment

### Usage

```
analysis_environment_snapshot(packages = loadedNamespaces())
```

### Arguments

| | |
|---|---|
| packages | Optional package names; defaults to loaded namespaces. |

### Value

A named list with components "r_version", "platform", "os", "locale", "timezone", "packages", containing snapshot an eyeprocess analysis environment and associated metadata or diagnostics.

## analysis_resolution_guard

*Audit compatibility between measurement resolution and an analysis target*

### Description

Audit compatibility between measurement resolution and an analysis target

### Usage

```
analysis_resolution_guard(
  event_duration_ms,
  effective_hz,
  spatial_feature_size = NA_real_,
  radial_error = NA_real_,
  min_samples = 3,
  max_error_fraction = 0.5
)
```

### Arguments

| | |
|---|---|
| `event_duration_ms` | Smallest event duration the analysis intends to resolve. |
| `effective_hz` | Empirical sampling frequency. |
| `spatial_feature_size` | Optional smallest spatial feature/AOI dimension in coordinate units. |
| `radial_error` | Optional empirical radial error in the same spatial units. |
| `min_samples` | User-declared minimum samples per temporal feature. |
| `max_error_fraction` | User-declared maximum spatial-error / feature-size ratio. |

### Value

A named list with components "expected_samples", "temporal_ok", "spatial_error_fraction", "spatial_ok", "min_samples", "max_error_fraction", "overall", "caveat", containing compatibility between measurement resolution and an analysis target and associated metadata or diagnostics.

---

## `aoi_membership_probability` *AOI membership probabilities from uncertainty draws*

---

### Description

AOI membership probabilities from uncertainty draws

### Usage

```
aoi_membership_probability(draws, aois)
```

### Arguments

| | |
|---|---|
| `draws` | Output of ‘propagate_calibration_uncertainty()‘ or compatible table. |
| `aois` | Rectangular AOI table with aoi/x_min/x_max/y_min/y_max. |

### Value

A tabular R object containing aOI membership probabilities from uncertainty draws; rows represent analysis units and columns contain the returned quantities.

aoi_perturbation_uncertainty

*AOI Perturbation and Uncertainty Analysis*

## Description

Vendor-neutral tools for treating area-of-interest (AOI) geometry as an explicit analytical assumption. The workflow validates rectangle and polygon geometry, constructs reproducible perturbations, remaps gaze or fixation coordinates, recomputes AOI features, optionally reruns a user-supplied statistical model, and reports assignment and inference stability without interpreting robustness frequencies as probabilities that a scientific conclusion is true.

## Usage

```
validate_aoi_geometry(aois, allow_overlap = TRUE)

convert_aoi_margin_to_degrees(
  margin_pixels, screen_width_px, screen_height_px,
  viewing_distance, physical_screen_size
)

convert_aoi_margin_to_pixels(
  margin_degrees, screen_width_px, screen_height_px,
  viewing_distance, physical_screen_size
)

aoi_perturbation_spec(
  perturbation_id, operation = "baseline", margin_x = 0,
  margin_y = margin_x, translation_x = 0, translation_y = 0,
  unit = "px", screen_width_px = NULL, screen_height_px = NULL,
  viewing_distance = NULL, physical_screen_size = NULL,
  degrees_per_pixel = NULL, seed = NULL, boundary_policy = "warn"
)

dilate_aoi(aois, margin, ...)
erode_aoi(aois, margin, ...)
translate_aoi(aois, x = 0, y = 0, ...)
jitter_aoi(aois, x, y = x, seed = 20260918L, ...)
perturb_aoi_geometry(aois, spec)

create_aoi_perturbation_grid(
  dilations = NULL, erosions = NULL, translations_x = NULL,
  translations_y = NULL, translations_xy = NULL,
  jitters = NULL, anisotropic = NULL,
  unit = "px", include_baseline = TRUE, seed = 20260918L, ...
)
```

```
apply_aoi_perturbation_grid(aois, grid)
compare_aoi_assignments(baseline, perturbed, ids = seq_along(baseline))

estimate_aoi_assignment_stability(
  comparisons, metadata = NULL, group_cols = NULL
)

estimate_fixation_assignment_probability(
  assignments, ids = NULL, include_baseline = TRUE
)

recompute_aoi_features(
  data, assignments, participant_col = NULL, trial_col = NULL,
  duration_col = NULL, time_col = NULL, perturbation_id = NULL,
  aoi_levels = NULL, observation_level = c("fixation", "sample")
)

run_aoi_sensitivity_analysis(
  data, aois, grid, x_col, y_col, observation_id_col = NULL,
  participant_col = NULL, trial_col = NULL, duration_col = NULL,
  time_col = NULL, observation_level = c("fixation", "sample"),
  overlap_policy = "ambiguous", model_callback = NULL,
  preprocessing_specification = NULL, event_detector = NULL,
  quality_rules = NULL, model_specification = NULL
)

assess_aoi_inference_stability(x, term = NULL)
summarise_aoi_sensitivity(x)
report_aoi_sensitivity(x)

plot_aoi_perturbations(
  x, perturbation_id = NULL, data = NULL,
  x_col = NULL, y_col = NULL, ...
)

plot_aoi_assignment_stability(x, ...)
plot_aoi_coefficient_stability(x, term, ...)

plot_aoi_robustness_surface(
  x, value_col = "proportion_unchanged",
  x_col = "margin_x", y_col = "margin_y", ...
)
```

## Arguments

| | |
|---|---|
| aois | A data frame containing rectangular or polygonal AOI geometry. |
| allow_overlap | Whether overlapping AOIs are allowed during validation. |

`margin_pixels`, `margin_degrees`, `margin`, `margin_x`, `margin_y`
: Perturbation margins in the declared unit.

`screen_width_px`, `screen_height_px`
: Declared display resolution in pixels.

`viewing_distance`
: Viewing distance in the same physical unit used for screen size.

`physical_screen_size`
: Length-two physical screen width and height.

`perturbation_id`
: Stable identifier for a perturbation branch.

`operation`
: One of baseline, dilation, erosion, translate, jitter, or anisotropic expansion.

`translation_x`, `translation_y`, `y`
: Horizontal and vertical translation values.

`unit`
: Either pixels (px) or degrees of visual angle (deg).

`degrees_per_pixel`
: Optional explicit horizontal and vertical degrees-per-pixel values.

`seed`
: Reproducible random seed used for jitter.

`boundary_policy`
: One of warn, clip, error, or allow for geometry outside the declared screen.

`spec`
: An AOI perturbation specification.

`dilations`, `erosions`, `translations_x`, `translations_y`, `translations_xy`, `jitters`, `anisotropic`
: Vectors or pairs defining a deterministic perturbation grid. 'translations_xy' accepts a length-two vector, two-column matrix/data frame, or list of x/y pairs.

`include_baseline`
: Whether to include the unchanged nominal branch.

`grid`
: An AOI perturbation grid.

`baseline`, `perturbed`
: Nominal and perturbed AOI assignment vectors.

`ids`
: Optional observation identifiers.

`comparisons`
: Named AOI assignment comparison objects.

`metadata`
: Optional one-row-per-observation metadata.

`group_cols`
: Optional participant, trial, or other grouping columns.

`assignments`
: AOI assignment vector or named branch list, depending on the function.

`data`
: Sample- or fixation-level data.

`participant_col`, `trial_col`
: Optional participant and trial grouping columns.

`duration_col`, `time_col`
: Optional duration and time columns used for AOI feature recomputation.

`aoi_levels`
: Optional complete AOI identifier universe. When supplied, observed participant/trial groups retain explicit zero-count AOI cells; groups with no valid assignment opportunity retain missing values rather than zeros.

`observation_level`
: Explicit input level, either fixation or sample. Universal observation counts/timing are retained without labeling sample rows as fixations.

`x_col, y_col`
: Coordinate columns in data or robustness-surface dimensions.

`observation_id_col`
: Optional unique observation identifier column.

`overlap_policy`
: How observed overlapping membership is represented; defaults to explicit ambiguity.

`model_callback`
: Optional function receiving recomputed features, assigned row data, and the current perturbation specification.

`preprocessing_specification, event_detector, quality_rules, model_specification`
: Provenance fields retained in the result object.

`x`
: Horizontal translation or jitter value for geometry helpers, or an AOI sensitivity, perturbation, or stability object for plotting and summary methods, depending on the function.

`term`
: Model term to summarize or plot.

`value_col`
: Robustness quantity used in the surface plot.

`...`
: Additional perturbation-specification or plotting arguments.

## Details

Overlapping AOIs are not resolved silently. The default row-level assignment for overlap is the explicit label `__ambiguous__`. Missing coordinates remain missing. Feature recomputation distinguishes structural zero AOI cells from groups with no valid assignment opportunity; the latter remain missing. Degree-based perturbations require declared display geometry and viewing distance, or explicit degrees-per-pixel values.

A model callback must explicitly choose and fit the desired estimator and return at least term, estimate, SE, CI_low, CI_high, p_value, model_converged, and N. Convergence must be explicit; converged rows require finite estimates, standard errors, confidence limits, and positive integer N. Callback errors and non-converged fits remain visible in the audit trail, and inference summaries retain the model-N range across converged perturbations.

Same-sign proportions and assignment frequencies are descriptive sensitivity summaries conditional on the declared perturbation set. They must not be reported as probabilities that the scientific conclusion is true.

## Value

Functions return validated AOI geometry, structured perturbation specifications and grids, perturbed geometry, assignment-comparison and stability objects, recomputed feature tables, AOI sensitivity objects, descriptive summaries, Markdown reporting text, or plots as documented by each function.

## Examples

```
aois <- data.frame(
  aoi_id = c("claim", "disclosure"),
```

```
  xmin = c(100, 100), xmax = c(400, 400),
  ymin = c(100, 240), ymax = c(220, 320)
)

grid <- create_aoi_perturbation_grid(
  dilations = c(5, 10),
  erosions = 5,
  translations_x = 5,
  translations_xy = c(5, -5)
)

gaze <- data.frame(
  id = 1:5,
  x = c(150, 390, 405, 180, NA),
  y = c(150, 215, 215, 270, NA),
  duration = c(.1, .1, .1, .1, NA)
)

result <- run_aoi_sensitivity_analysis(
  gaze, aois, grid,
  x_col = "x", y_col = "y",
  observation_id_col = "id",
  duration_col = "duration",
  observation_level = "fixation"
)

summarise_aoi_sensitivity(result)
```

---

`aoi_trajectory_features` *Extract AOI growth-curve/trajectory features*

---

## Description

Converts AOI occupancy over binned time into orthogonal-polynomial trajectory coefficients. Coefficients summarize temporal shape and are not latent psychological traits by themselves.

## Usage

```
aoi_trajectory_features(
  data,
  person = "person_id",
  trial = "trial_id",
  time = "time_ms",
  aoi = "aoi",
  bin_ms = 100,
  degree = 3L,
  aois = NULL
)
```

### Arguments

| | |
|---|---|
| `data` | Sample-level data. |
| `person`, `trial`, `time`, `aoi` | Column names. |
| `bin_ms` | Temporal bin width. |
| `degree` | Polynomial degree. |
| `aois` | Optional AOIs to encode; defaults to observed AOIs. |

### Value

An object of class "eye_aoi_trajectory", stored as a named list, with components "features", "aois", "degree", "bin_ms", "person", "trial", "caveat". It contains aOI growth-curve/trajectory features and associated metadata or diagnostics needed to interpret the result.

---

## `api_family_map` *Map exported APIs to conceptual families*

---

### Description

Map exported APIs to conceptual families

### Usage

```
api_family_map(inventory)
```

### Arguments

| | |
|---|---|
| `inventory` | API inventory or character names. |

### Value

A data frame containing exported APIs to conceptual families. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## api_lifecycle_diff *Compare two API lifecycle registries*

### Description

Compare two API lifecycle registries

### Usage

```
api_lifecycle_diff(old, new)
```

### Arguments

| | |
|---|---|
| old | Old registry. |
| new | New registry. |

### Value

A tabular R object containing two API lifecycle registries; rows represent analysis units and columns contain the returned quantities.

## api_surface_summary *Summarise API surface by family and lifecycle status*

### Description

Summarise API surface by family and lifecycle status

### Usage

```
api_surface_summary(inventory)
```

### Arguments

| | |
|---|---|
| inventory | Output of ‘eye_api_inventory()‘ or compatible table. |

### Value

A data frame containing aPI surface by family and lifecycle status. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

apply_preflight_decision *Apply a pre-flight decision to data explicitly*

## Description

Apply a pre-flight decision to data explicitly

## Usage

```
apply_preflight_decision(
  data,
  audit,
  keep_decisions = c("pass_preflight", "use_with_caution")
)
```

## Arguments

| | |
|---|---|
| data | Original data. |
| audit | Pre-flight audit. |
| keep_decisions | Decisions to retain. |

## Value

Filtered data with an attached ‘preflight_application‘ attribute.

apply_synthetic_corruption *Apply a synthetic corruption plan*

## Description

Apply a synthetic corruption plan

## Usage

```
apply_synthetic_corruption(
  data,
  plan,
  gaze_columns = c("gaze_x", "gaze_y"),
  pupil = "pupil",
  time = "timestamp_ms",
  aoi = NULL,
  device_column = NULL
)
```

### Arguments

| | |
|---|---|
| data | Data frame. |
| plan | Corruption plan. |
| gaze_columns | Gaze columns receiving generic missingness/offset. |
| pupil | Pupil column. |
| time | Timestamp column. |
| aoi | Optional AOI column. |
| device_column | Optional numeric column receiving device shift. |

### Value

An R object containing a synthetic corruption plan. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

## assign_aois_probabilistic

*Probabilistic AOI assignment and uncertainty propagation*

---

### Description

Probabilistic AOI assignment and uncertainty propagation. These functions form the eyeprocess 0.6.0.9000 measurement-intelligence programme and use dependency-free reference implementations with explicit evidence limits.

### Usage

```
assign_aois_probabilistic(x, aois, error_model = c("empirical", "gaussian",
  "ellipse"), accuracy = NULL, precision = NULL, x_col = NULL, y_col = NULL,
  id_cols = NULL)
audit_aoi_separation(x = NULL, aois = NULL)
summarise_aoi_membership(x, by = NULL)
propagate_aoi_uncertainty(x, metrics = c("dwell", "ttff", "transitions", "entropy"),
  draws = 500, time_col = NULL, duration_col = NULL, seed = 20260807)
plot_aoi_probability_map(x, ...)
plot_aoi_boundary_risk(x, ...)
plot_probabilistic_scanpath(x, ...)
plot_fuzzy_transition_matrix(x, ...)
plot_aoi_metric_uncertainty(x, ...)
```

### Arguments

| | |
|---|---|
| x | Input object or data structure appropriate for the selected analysis. |
| aois | Argument controlling ‘aois‘; see the function usage and returned audit metadata. |
| error_model | Argument controlling ‘error_model‘; see the function usage and returned audit metadata. |

| | |
|---|---|
| accuracy | Argument controlling ‘accuracy‘; see the function usage and returned audit metadata. |
| precision | Argument controlling ‘precision‘; see the function usage and returned audit metadata. |
| x_col | Argument controlling ‘x_col‘; see the function usage and returned audit metadata. |
| y_col | Argument controlling ‘y_col‘; see the function usage and returned audit metadata. |
| id_cols | Argument controlling ‘id_cols‘; see the function usage and returned audit metadata. |
| by | Argument controlling ‘by‘; see the function usage and returned audit metadata. |
| metrics | Argument controlling ‘metrics‘; see the function usage and returned audit metadata. |
| draws | Argument controlling ‘draws‘; see the function usage and returned audit metadata. |
| time_col | Argument controlling ‘time_col‘; see the function usage and returned audit metadata. |
| duration_col | Argument controlling ‘duration_col‘; see the function usage and returned audit metadata. |
| seed | Argument controlling ‘seed‘; see the function usage and returned audit metadata. |
| ... | Additional arguments passed to the underlying method or plotting function. |

## Details

The APIs return auditable S3 objects. Plot wrappers call registered base-graphics methods. Experimental or approximate engines are labelled in object status fields and should be validated before confirmatory or operational use.

## Value

An eyeprocess result object, data frame, model object, plot, or audit table as documented by the individual function.

## See Also

`plot_diagnostics()`, `plot_evidence()`, and `plot_sensitivity()`.

## assign_process_feature_family

*Assign features to conservative process-feature families*

### Description

Assign features to conservative process-feature families

### Usage

```
assign_process_feature_family(
  feature_names,
  registry = process_feature_family_registry()
)
```

### Arguments

| | |
|---|---|
| feature_names | Feature names. |
| registry | Feature-family registry. |

### Value

A vector or matrix containing assign features to conservative process-feature families, with shape determined by the supplied analysis units.

## as_irt_recovery_results

*Canonicalise parameter-recovery results*

### Description

Canonicalise parameter-recovery results

### Usage

```
as_irt_recovery_results(results)
```

### Arguments

| | |
|---|---|
| results | Data frame with at least ‘replicate‘, ‘parameter‘, ‘truth‘, and ‘estimate‘; optional ‘lower‘, ‘upper‘, ‘converged‘, ‘scenario‘, ‘engine‘, and ‘failure_type‘ columns are retained. |

### Value

A data frame containing canonicalise parameter-recovery results. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## audit_3pl_process_signatures *Identify items for descriptive 3PL/process review*

### Description

Identify items for descriptive 3PL/process review

### Usage

```
audit_3pl_process_signatures(
  x,
  lower_asymptote_quantile = 0.8,
  fast_rt_quantile = 0.2,
  fast_ttff_quantile = 0.2
)
```

### Arguments

| | |
|---|---|
| x | An 'eye_gaze_anchored_3pl_audit'. |
| lower_asymptote_quantile | Quantile used to flag relatively large lower asymptotes. |
| fast_rt_quantile | Optional lower quantile for RT review. |
| fast_ttff_quantile | Optional lower quantile for TTFF review. |

### Value

Item-level review table. These flags are not behavioral classifications.

## audit_advanced_model_evidence *Audit advanced-model scientific evidence*

### Description

Audit advanced-model scientific evidence

### Usage

```
audit_advanced_model_evidence(evidence, spec = advanced_model_evidence_spec())
```

### Arguments

| | |
|---|---|
| evidence | Named list keyed by model function. Each model may contain ‘recovery‘, ‘calibration‘, ‘misspecification‘, ‘grouped_validation‘, ‘engine_equivalence‘, ‘empirical_reproduction‘, and ‘sensitivity‘ objects. |
| spec | Evidence specification. |

### Value

An ‘eye_advanced_evidence_audit‘ data frame.

---

## audit_benchmark_release

*Audit whether benchmark assets are ready for public release*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
audit_benchmark_release(study = eyeprocess_benchmark_study())
```

### Arguments

| | |
|---|---|
| study | Benchmark study. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

## audit_bias *Audit bias*

---

### Description

Audit bias

### Usage

```
audit_bias(results, threshold = 0.1, by = c("scenario", "engine", "parameter"))
```

### Arguments

| | |
|---|---|
| results | Validation or model results. |
| threshold | Decision or diagnostic threshold. |
| by | Grouping variables used when summarizing results. |

### Value

An R object containing bias. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

## audit_biometric_imputation

*Alias emphasizing sensitivity rather than automatic replacement*

---

### Description

Alias emphasizing sensitivity rather than automatic replacement

### Usage

```
audit_biometric_imputation(...)
```

### Arguments

| | |
|---|---|
| ... | Additional arguments passed to the underlying method or helper. |

### Value

An R object containing alias emphasizing sensitivity rather than automatic replacement. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

## audit_biometric_preflight

*Audit incoming biometric/process data before modelling*

---

### Description

Aggregates trial-level signal-quality indicators by person/recording (or other supplied grouping columns) and returns review-oriented flags. No rows are automatically deleted.

## Usage

```
audit_biometric_preflight(
  data,
  by = c("person_id"),
  spec = process_preflight_spec(),
  valid_gaze_prop = "valid_gaze_prop",
  valid_pupil_prop = "valid_pupil_prop",
  missing_gaze = "missing_gaze",
  missing_pupil = "missing_pupil",
  rt_ms = "rt_ms",
  blink_cluster_count = "blink_cluster_count",
  sampling_rate_hz = "sampling_rate_hz"
)
```

## Arguments

| | |
|---|---|
| `data` | Trial- or recording-level data. |
| `by` | Grouping columns, usually participant and optionally recording/session. |
| `spec` | A ‘process_preflight_spec()‘ object. |
| `valid_gaze_prop`, `valid_pupil_prop` | Column names for validity proportions. |
| `missing_gaze`, `missing_pupil` | Column names for missingness indicators/proportions. |
| `rt_ms` | Response-time column. |
| `blink_cluster_count` | Blink-cluster count column. |
| `sampling_rate_hz` | Sampling-rate column. |

## Value

An ‘eye_biometric_preflight‘ object.

---

# audit_candidate_item_bank

*Audit a candidate item bank against a seed model*

---

## Description

Audit a candidate item bank against a seed model

### Usage

```
audit_candidate_item_bank(
  object,
  candidate_data,
  difficulty_range = c(-3, 3),
  discrimination_min = 0.3
)
```

### Arguments

| | |
|---|---|
| `object` | Seed model. |
| `candidate_data` | Candidate item feature data. |
| `difficulty_range` | Plausible screening range for predicted difficulty. |
| `discrimination_min` | Minimum screening discrimination. |

### Value

An object of class "eye_candidate_item_bank_audit", stored as a named list, with components "table", "seed_model", "status", "caveat". It contains a candidate item bank against a seed model and associated metadata or diagnostics needed to interpret the result.

---

## `audit_channel_incremental_information` *Audit out-of-sample incremental information from a process channel*

---

### Description

Audit out-of-sample incremental information from a process channel

### Usage

```
audit_channel_incremental_information(
  data,
  fold,
  baseline_fitter,
  process_fitter,
  predictor,
  scorer,
  higher_is_better = TRUE
)
```

### Arguments

| | |
|---|---|
| `data` | Input data. |
| `fold` | Group/fold column. Every unique value is held out once. |
| `baseline_fitter` | Function fitted without the process channel. |
| `process_fitter` | Function fitted with the process channel. |
| `predictor` | Function ‘(fit, test)‘ returning predictions. |
| `scorer` | Function ‘(test, prediction)‘ returning a scalar score. |
| `higher_is_better` | Direction of the score. |

### Value

A tabular R object containing out-of-sample incremental information from a process channel; rows represent analysis units and columns contain the returned quantities.

---

## `audit_convergence` *Audit convergence and classified failures*

---

### Description

Audit convergence and classified failures

### Usage

```
audit_convergence(results, minimum = 0.95, by = c("scenario", "engine"))
```

### Arguments

| | |
|---|---|
| `results` | Validation or model results. |
| `minimum` | Minimum acceptable value or threshold. |
| `by` | Grouping variables used when summarizing results. |

### Value

An object of class "eye_irt_convergence_audit", "data.frame", stored as a data frame, containing convergence and classified failures and associated metadata needed to interpret the result.

## audit_coverage *Audit coverage*

### Description

Audit coverage

### Usage

```
audit_coverage(
  results,
  minimum = 0.9,
  maximum = 1,
  by = c("scenario", "engine", "parameter")
)
```

### Arguments

| | |
|---|---|
| results | Validation or model results. |
| minimum | Minimum acceptable value or threshold. |
| maximum | Maximum acceptable value or threshold. |
| by | Grouping variables used when summarizing results. |

### Value

An R object containing coverage. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## audit_decision_provenance *Audit decision provenance and completeness*

### Description

Audit decision provenance and completeness

### Usage

```
audit_decision_provenance(
  x,
 required_domains = c("sampling", "validity", "fixation", "pupil", "aoi", "model",
    "sensitivity", "exclusions"),
  required_provenance = c("data_source", "software_version", "analysis_commit")
)
```

### Arguments

| | |
|---|---|
| x | Manifest. |
| required_domains | Required decision domains. |
| required_provenance | Provenance keys expected under ‘provenance‘. |

### Value

An object of class "eye_decision_provenance_audit", stored as a named list, with components "missing_domains", "empty_domains", "missing_provenance", "complete", "manifest_hash". It contains decision provenance and completeness and associated metadata or diagnostics needed to interpret the result.

---

## audit_distractor_attention

*Audit distractor attention patterns*

---

### Description

Audit distractor attention patterns

### Usage

```
audit_distractor_attention(
  data,
  response_option = "response_option",
  option_gaze,
  chosen_suffix = NULL
)
```

### Arguments

| | |
|---|---|
| data | Input data frame or compatible tabular object. |
| response_option | Column identifying the selected response option. |
| option_gaze | Option-level gaze variables. |
| chosen_suffix | Suffix identifying the selected option indicator. |

### Value

A data frame containing distractor attention patterns. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## audit_eye_api *Audit API lifecycle completeness and replacement contracts*

### Description

Audit API lifecycle completeness and replacement contracts

### Usage

```
audit_eye_api(inventory = eye_api_inventory(), registry = eye_api_lifecycle())
```

### Arguments

| | |
|---|---|
| inventory | API inventory. |
| registry | Lifecycle registry. |

### Value

An object of class "eye_api_audit", stored as a named list, with components "table", "unreviewed", "invalid_replacements", "invalid_canonical", "reviewed_fraction", "valid". It contains aPI lifecycle completeness and replacement contracts and associated metadata or diagnostics needed to interpret the result.

## audit_eye_pipeline *Audit a pipeline definition or completed run*

### Description

Audit a pipeline definition or completed run

### Usage

```
audit_eye_pipeline(x)
```

### Arguments

| | |
|---|---|
| x | Pipeline or pipeline run. |

### Value

An object of class "eye_pipeline_audit", stored as a named list, with components "table", "undeclared_decisions", "valid", "pipeline_hash". It contains a pipeline definition or completed run and associated metadata or diagnostics needed to interpret the result.

## audit_frontier_model_contract *Audit whether a gated frontier model has a minimum evidence contract*

### Description

Audit whether a gated frontier model has a minimum evidence contract

### Usage

```
audit_frontier_model_contract(x, evidence = list())
```

### Arguments

| | |
|---|---|
| x | Gated model object. |
| evidence | Named evidence objects. |

### Value

A data frame containing whether a gated frontier model has a minimum evidence contract. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## audit_identifiability *Audit empirical identifiability from replicate estimates*

### Description

Flags parameters with near-zero estimate variance, explosive dispersion, excessive missingness, or strongly correlated estimates when a covariance matrix is supplied.

### Usage

```
audit_identifiability(
  results,
  max_missing = 0.05,
  max_sd_ratio = 10,
  correlation_matrix = NULL,
  max_abs_correlation = 0.995
)
```

### Arguments

| | |
|---|---|
| `results` | Validation or model results. |
| `max_missing` | Maximum acceptable missingness. |
| `max_sd_ratio` | Maximum acceptable standard-deviation ratio. |
| `correlation_matrix` | Optional parameter-correlation matrix. |
| `max_abs_correlation` | Maximum acceptable absolute parameter correlation. |

### Value

An object of class "eye_irt_identifiability_audit", "data.frame", stored as a data frame, containing empirical identifiability from replicate estimates and associated metadata needed to interpret the result.

---

## `audit_interval_width` *Audit interval width*

---

### Description

Audit interval width

### Usage

```
audit_interval_width(
  results,
  maximum = Inf,
  by = c("scenario", "engine", "parameter")
)
```

### Arguments

| | |
|---|---|
| `results` | Validation or model results. |
| `maximum` | Maximum acceptable value or threshold. |
| `by` | Grouping variables used when summarizing results. |

### Value

An R object containing interval width. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

`audit_irf_shape` *Audit item response-function shape departures*

## Description

Audit item response-function shape departures

## Usage

```
audit_irf_shape(
  comparison,
  mean_absolute_threshold = 0.05,
  max_absolute_threshold = 0.15
)
```

## Arguments

| | |
|---|---|
| `comparison` | Value supplied to 'comparison'; see Details for its model-specific role. |
| `mean_absolute_threshold` | Threshold for mean absolute IRF departure. |
| `max_absolute_threshold` | Threshold for maximum absolute IRF departure. |

## Value

A tabular R object containing item response-function shape departures; rows represent analysis units and columns contain the returned quantities.

`audit_item_reduction_sensitivity` *Run stepwise Rasch item-reduction as a sensitivity analysis*

## Description

Run stepwise Rasch item-reduction as a sensitivity analysis

## Usage

```
audit_item_reduction_sensitivity(
  erm_model,
  criterion = list("itemfit"),
  alpha = 0.05,
  maxstep = 5L
)
```

### Arguments

| | |
|---|---|
| erm_model | Fitted eRm model. |
| criterion | Criterion list passed to ‘eRm::stepwiseIt()‘. |
| alpha | Significance threshold. |
| maxstep | Maximum elimination steps. |

### Value

An object of class "eye_item_reduction_sensitivity", stored as a named list, with components "model", "eliminated_items", "alpha", "maxstep", "status", "caveat". It contains stepwise Rasch item-reduction as a sensitivity analysis and associated metadata or diagnostics needed to interpret the result.

---

## audit_latent_distribution

*Audit the empirical latent-trait distribution*

---

### Description

Audit the empirical latent-trait distribution

### Usage

```
audit_latent_distribution(theta, tail_z = 3)
```

### Arguments

| | |
|---|---|
| theta | Numeric latent-trait draws/estimates. |
| tail_z | Absolute standardized threshold used for tail-rate diagnostics. |

### Value

A data frame containing the empirical latent-trait distribution. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## audit_measurement_transportability *Summarise measurement transportability across held-out groups*

### Description

Summarise measurement transportability across held-out groups

### Usage

```
audit_measurement_transportability(
  validation,
  metric,
  higher_is_better = TRUE,
  max_range = NULL,
  minimum = NULL,
  maximum = NULL
)
```

### Arguments

| | |
|---|---|
| validation | Validation results or validation specification. |
| metric | Metric to calculate or audit. |
| higher_is_better | Whether larger metric values indicate better performance. |
| max_range | Maximum allowed range across held-out groups. |
| minimum | Minimum acceptable value or threshold. |
| maximum | Maximum acceptable value or threshold. |

### Value

An object of class "eye_measurement_transportability_audit", "data.frame", stored as a data frame, containing measurement transportability across held-out groups and associated metadata needed to interpret the result.

## audit_model_promotion *Audit promotion readiness for advanced model families*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
audit_model_promotion(evidence, spec = model_promotion_spec())
```

### Arguments

| | |
|---|---|
| evidence | Named list by model family. Each family may contain |
| spec | Promotion specification. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

## audit_multimodal_identifiability
*Audit basic multimodal design identifiability*

---

### Description

This is a structural pre-flight screen, not a proof of statistical identifiability.

### Usage

```
audit_multimodal_identifiability(x, min_person = 30L, min_item = 5L)
```

### Arguments

| | |
|---|---|
| x | An ‘eye_multimodal_measurement‘. |
| min_person, min_item | Minimum structural counts. |

### Value

An ‘eye_multimodal_identifiability_audit‘.

## audit_multimodal_m2_identifiability

*Audit structural and data identifiability for M0-M2*

### Description

This is a conservative pre-fit audit. It does not prove global identifiability. It verifies support, design connectivity, channel coverage, response variation, and the explicit scale constraints used by the reference likelihoods.

### Usage

```
audit_multimodal_m2_identifiability(
  x,
  person = "person_id",
  item = "item_id",
  response = "response",
  rt = "rt",
  gaze = "gaze",
  model = c("M2", "M1", "M0"),
  min_persons = 20L,
  min_items = 5L
)
```

### Arguments

| | |
|---|---|
| x | Data frame or compatible M2 object. |
| person, item, response, rt, gaze | Column names. |
| model | '"M0"', '"M1"', or '"M2"'. |
| min_persons, min_items | Conservative design thresholds. |

### Value

An 'eye_multimodal_m2_identifiability' object.

## audit_multimodal_m3_identifiability

*Audit structural and measurement support for M3*

## Description

Performs a conservative pre-fit audit for the four-channel response, RT, fixation-count and pupil reference model. The audit checks design connectivity, channel coverage and variation, pupil nuisance availability, blink/interpolation burden, device/session representation and explicit M3 identification constraints. It is a support screen, not proof of global identifiability or construct validity.

## Usage

```
audit_multimodal_m3_identifiability(
  x,
  pupil_scale = c("z", "raw"),
  min_persons = 20L,
  min_items = 5L,
  max_pupil_missing = 0.5,
  max_blink_rate = 0.3,
  max_interpolation_rate = 0.3
)
```

## Arguments

| | |
|---|---|
| `x` | M3-compatible data, simulation, fit, or measurement object. |
| `pupil_scale` | Pupil transformation used by the reference likelihood. |
| `min_persons`, `min_items` | Conservative design thresholds. |
| `max_pupil_missing`, `max_blink_rate`, `max_interpolation_rate` | Warning thresholds. |

## Value

An ‘eye_multimodal_m3_identifiability‘ object.

---

`audit_multimodal_m4_identifiability`
*Audit M4 structural and posterior identifiability*

---

## Description

Evaluates sequence information, channel support, state occupancy, assignment uncertainty, state separation, transition degeneracy, and HMC diagnostics. The result is deliberately multi-criterion; M4 does not collapse validity to a single undocumented boolean.

### Usage

```
audit_multimodal_m4_identifiability(
  x,
  spec = NULL,
  include_posterior = TRUE,
  rhat_max = 1.05,
  ess_min = 100,
  ebfmi_min = 0.3,
  occupancy_min = 0.03,
  entropy_fraction_review = 0.8
)
```

### Arguments

| | |
|---|---|
| x | Data, M4 simulation, M4 fit, or internal prepared M4 data. |
| spec | Optional M4 specification when ‘x‘ is not a fit. |
| include_posterior | Whether to evaluate posterior criteria for a fit. |
| rhat_max, ess_min, ebfmi_min | Sampler thresholds. |
| occupancy_min | Minimum mean posterior state occupancy for non-null K. |
| entropy_fraction_review | Review threshold relative to maximum entropy. |

### Value

An ‘eye_multimodal_m4_identifiability‘ with machine-readable checks.

---

## audit_multimodal_measurement *Audit a multimodal measurement object*

---

### Description

Audit a multimodal measurement object

### Usage

```
audit_multimodal_measurement(x)
```

### Arguments

| | |
|---|---|
| x | An ‘eye_multimodal_measurement‘. |

### Value

An ‘eye_multimodal_audit‘.

`audit_multivariate_process_quality` *Alias emphasizing data-quality interpretation of process anomaly auditing*

## Description

Alias emphasizing data-quality interpretation of process anomaly auditing

## Usage

```
audit_multivariate_process_quality(...)
```

## Arguments

| | |
|---|---|
| `...` | Additional arguments passed to the underlying method or helper. |

## Value

An R object containing alias emphasizing data-quality interpretation of process anomaly auditing. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

`audit_nonparametric_rasch` *Run nonparametric Rasch diagnostics with eRm*

## Description

Run nonparametric Rasch diagnostics with eRm

## Usage

```
audit_nonparametric_rasch(
  response_matrix,
  methods = c("T1", "T10"),
  n = 100L,
  splitcr = "median",
  seed = 321
)
```

### Arguments

| | |
|---|---|
| response_matrix | Dichotomous response matrix. |
| methods | eRm NPtest methods, e.g. T1 and T10. |
| n | Number of sampled matrices. |
| splitcr | Split criterion for tests that use one. |
| seed | Seed. |

### Value

An object of class "eye_nonparametric_rasch_audit", stored as a named list, with components "tests", "status", "n", "splitcr", "caveat". It contains nonparametric Rasch diagnostics with eRm and associated metadata or diagnostics needed to interpret the result.

---

## audit_presentation_accessibility

*Audit presentation/accessibility sensitivity without clinical inference*

---

### Description

Creates a transparent presentation-review score from response time/dwell, revisit/entropy, pupil-effort proxies, and gaze quality. It is an experimental design/fairness audit only.

### Usage

```
audit_presentation_accessibility(
  data,
  person = "person_id",
  rt = "rt_ms",
  dwell = "dwell_ms",
  revisits = "revisits",
  entropy = "aoi_entropy",
  pupil = "pupil_peak",
  gaze_validity = "valid_gaze_prop",
  review_quantile = 0.9
)
```

### Arguments

| | |
|---|---|
| data | Process data. |
| person | Person identifier. |
| rt, dwell, revisits, entropy, pupil, gaze_validity | Optional column names. |
| review_quantile | Quantile for a presentation-review flag. |

## Value

An object of class "eye_presentation_accessibility", stored as a named list, with components "table", "threshold", "review_quantile", "status", "caveat". It contains presentation/accessibility sensitivity without clinical inference and associated metadata or diagnostics needed to interpret the result.

---

`audit_process_adjusted_dif` *Audit DIF before and after process-data adjustment*

---

## Description

Audit DIF before and after process-data adjustment

## Usage

```
audit_process_adjusted_dif(
  data,
  response = "response",
  ability,
  group,
  item = "item_id",
  process_features,
  person = "participant_id"
)
```

## Arguments

| | |
|---|---|
| `data` | Input data frame or compatible tabular object. |
| `response` | Response variable or response-column name. |
| `ability` | Value supplied to ‘ability‘; see Details for its model-specific role. |
| `group` | Value supplied to ‘group‘; see Details for its model-specific role. |
| `item` | Item identifier, name, or item column. |
| `process_features` | Names of process-derived features. |
| `person` | Person or participant identifier column. |

## Value

An object of class "eye_process_adjusted_dif", stored as a named list, with components "unadjusted_model", "adjusted_model", "coefficients", "surrogate", "note". It contains dIF before and after process-data adjustment and associated metadata or diagnostics needed to interpret the result.

## audit_process_anomalies *Audit multivariate process/data-quality anomalies*

### Description

Computes a regularized Mahalanobis distance over selected person-level process metrics. Flags indicate review needs only; they are not cheating, identity, diagnosis, or intent classifications.

### Usage

```
audit_process_anomalies(
  data,
  person = "person_id",
  metrics = NULL,
  alpha = 0.975,
  aggregate = TRUE,
  ridge = 1e-06
)
```

### Arguments

| | |
|---|---|
| data | Data frame. |
| person | Person identifier column. |
| metrics | Numeric process metrics. If omitted, usable numeric columns are selected. |
| alpha | Chi-square review quantile. |
| aggregate | If TRUE, aggregate metrics to person level before auditing. |
| ridge | Diagonal covariance regularization. |

### Value

An object of class "eye_process_anomaly_audit", stored as a named list, with components "table", "metrics", "alpha", "threshold", "center", "covariance", "caveat". It contains multivariate process/data-quality anomalies and associated metadata or diagnostics needed to interpret the result.

## audit_process_drift *Audit post-deployment psychometric and biometric drift*

### Description

Audit post-deployment psychometric and biometric drift

### Usage

```
audit_process_drift(
  data,
  item = "item_id",
  batch = "deployment_batch",
 metrics = c("irt_difficulty", "irt_discrimination", "rt_ms", "dwell_ms", "pupil_bc",
    "valid_gaze_prop", "screen_luminance"),
  spec = process_drift_spec(),
  reference_batch = NULL,
  aggregate_fun = .ep08_mean
)
```

### Arguments

| | |
|---|---|
| data | Item-by-batch or trial-level deployment data. |
| item | Item identifier column. |
| batch | Ordered deployment batch/date column. |
| metrics | Numeric metrics to monitor. |
| spec | Drift specification. |
| reference_batch | Optional reference batch value(s). |
| aggregate_fun | Aggregation function used when multiple rows occur within item x batch. |

### Value

An object of class "eye_process_drift_audit", stored as a named list, with components "table", "trajectories", "item", "batch", "metrics", "spec", "reference_batch", "flag_columns", "caveat". It contains post-deployment psychometric and biometric drift and associated metadata or diagnostics needed to interpret the result.

---

## audit_process_external_validity

*Audit external/structural validity of process traits*

---

### Description

Audit external/structural validity of process traits

### Usage

```
audit_process_external_validity(
  data,
  criterion,
  predictors,
  baseline_predictors = NULL
)
```

### Arguments

| | |
|---|---|
| data | Person-level data containing a criterion and process predictors. |
| criterion | External criterion column. |
| predictors | Process predictors. |
| baseline_predictors | Optional baseline predictors for incremental validity. |

### Value

An object of class "eye_process_external_validity", stored as a named list, with components "full_model", "baseline_model", "comparison", "associations", "criterion", "predictors", "baseline_predictors", "data", "incremental_r2", "status", "caveat". It contains external/structural validity of process traits and associated metadata or diagnostics needed to interpret the result.

---

## audit_process_local_dependence

*Audit inter-option/process local dependence*

---

### Description

Computes pairwise residual correlations (a Q3-style diagnostic) across item or option columns and, when supplied, analogous process-residual correlations. This is a diagnostic for local dependence, not a formal test with universal cutoffs.

### Usage

```
audit_process_local_dependence(
  response_residuals,
  process_residuals = NULL,
  threshold = 0.2
)
```

### Arguments

| | |
|---|---|
| response_residuals | Person-by-item/option residual matrix. |
| process_residuals | Optional aligned process-residual matrix. |
| threshold | Absolute correlation threshold used only for flagging. |

### Value

An object of class "eye_process_local_dependence_audit", stored as a named list, with components "pairs", "threshold", "max_absolute_response", "note". It contains inter-option/process local dependence and associated metadata or diagnostics needed to interpret the result.

## audit_process_measurement_invariance *Audit process measurement invariance across facets*

### Description

Audit process measurement invariance across facets

### Usage

```
audit_process_measurement_invariance(
  object,
  channel = c("process", "response"),
  relative_sd_threshold = 0.25
)
```

### Arguments

| | |
|---|---|
| object | A fitted eyeprocess model or audit object. |
| channel | Measurement channel to inspect. |
| relative_sd_threshold | Value supplied to 'relative_sd_threshold'; see Details for its model-specific role. |

### Value

An object of class "eye_process_measurement_invariance", stored as a named list, with components "pass", "threshold", "components", "channel", "note". It contains process measurement invariance across facets and associated metadata or diagnostics needed to interpret the result.

## audit_process_window_sensitivity *Audit sensitivity of process summaries to temporal window choices*

### Description

Audit sensitivity of process summaries to temporal window choices

### Usage

```
audit_process_window_sensitivity(
  data,
  widths_ms = c(250, 500, 1000, 1500),
  steps_ms = c(100, 250, 500),
  metric = "pupil_mean",
  grid = TRUE,
  ...
)
```

### Arguments

| | |
|---|---|
| data | Sample-level data. |
| widths_ms | Window widths. |
| steps_ms | Step widths; recycled or crossed depending on ‘grid‘. |
| metric | Extracted process metric to compare. |
| grid | If TRUE, evaluate all width-step combinations. |
| ... | Passed to ‘extract_process_windows()‘. |

### Value

An object of class "eye_process_window_sensitivity", stored as a named list, with components "table", "metric", "settings", "caveat". It contains sensitivity of process summaries to temporal window choices and associated metadata or diagnostics needed to interpret the result.

---

## audit_pupil_fatigue_drift

*Audit within-person pupil fatigue/trial-order drift*

---

### Description

Audit within-person pupil fatigue/trial-order drift

### Usage

```
audit_pupil_fatigue_drift(
  data,
  pupil = "pupil_peak",
  trial_order = "trial_sequence",
  person = "person_id",
  luminance = NULL,
  difficulty = NULL,
  engine = c("auto", "plm", "lm_fixed_effects")
)
```

### Arguments

| | |
|---|---|
| data | Trial-level data. |
| pupil, trial_order, person | Required columns. |
| luminance, difficulty | Optional covariates. |
| engine | ‘auto‘, ‘plm‘, or ‘lm_fixed_effects‘. |

## Value

An object of class "eye_pupil_fatigue_drift", stored as a named list, with components "model", "coefficients", "data", "engine", "status", "caveat". It contains within-person pupil fatigue/trial-order drift and associated metadata or diagnostics needed to interpret the result.

---

`audit_pupil_frequency_stability` *Audit stability of pupil frequency features across window lengths*

---

## Description

Audit stability of pupil frequency features across window lengths

## Usage

```
audit_pupil_frequency_stability(
  data,
  windows_ms = c(500, 1000, 2000),
  by = c("person_id", "trial_id"),
  time = "time_ms",
  pupil = "pupil_bc",
  sampling_rate_hz = 60
)
```

## Arguments

| | |
|---|---|
| `data` | Sample-level data. |
| `windows_ms` | Window lengths to evaluate. |
| `by`, `time`, `pupil`, `sampling_rate_hz` | Passed through to feature construction. |

## Value

An object of class "eye_pupil_frequency_stability", stored as a named list, with components "table", "windows_ms", "caveat". It contains stability of pupil frequency features across window lengths and associated metadata or diagnostics needed to interpret the result.

## audit_pupil_preprocessing_order *Audit declared order of pupil preprocessing steps*

### Description

Audit declared order of pupil preprocessing steps

### Usage

```
audit_pupil_preprocessing_order(
  steps,
 cleaning_patterns = c("blink", "missing", "interpol", "artifact", "smooth", "filter"),
  baseline_pattern = "baseline"
)
```

### Arguments

| | |
|---|---|
| steps | Character vector in execution order. |
| cleaning_patterns | Patterns considered cleaning/preprocessing. |
| baseline_pattern | Pattern identifying baseline correction. |

### Value

A named list with components "steps", "baseline_positions", "cleaning_positions", "cleaning_after_baseline", "status", "caveat", containing declared order of pupil preprocessing steps and associated metadata or diagnostics.

## audit_rmse *Audit rmse*

### Description

Audit rmse

### Usage

```
audit_rmse(results, threshold = 0.3, by = c("scenario", "engine", "parameter"))
```

### Arguments

| | |
|---|---|
| results | Validation or model results. |
| threshold | Decision or diagnostic threshold. |
| by | Grouping variables used when summarizing results. |

### Value

An R object containing rmse. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

`audit_roundtrip_loss` *Audit semantic and numerical loss after a round trip*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
audit_roundtrip_loss(source, roundtrip, tables = canonical_table_names(),
  tolerance = 1e-8)
```

### Arguments

| | |
|---|---|
| `source` | Source and re-imported ‘eye_dataset‘ objects. |
| `roundtrip` | Source and re-imported ‘eye_dataset‘ objects. |
| `tables` | Canonical tables to compare. |
| `tolerance` | Numeric tolerance. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`audit_sampling_irregularity`
*Audit sampling irregularity*

---

### Description

Audit sampling irregularity

### Usage

```
audit_sampling_irregularity(
  data,
  time = "timestamp_ms",
  unit = c("ms", "s", "us"),
  by = NULL,
  cv_threshold = 0.05
)
```

### Arguments

| | |
|---|---|
| data | Sample data. |
| time | Timestamp column. |
| unit | Timestamp unit. |
| by | Optional grouping columns. |
| cv_threshold | Review threshold for interval coefficient of variation. |

### Value

An object of class "eye_sampling_irregularity_audit", stored as a named list, with components "table", "cv_threshold", "caveat". It contains sampling irregularity and associated metadata or diagnostics needed to interpret the result.

---

## audit_sbc *Audit SBC rank uniformity*

---

### Description

Uses binned chi-square diagnostics as a coarse screening diagnostic and also reports the mean/variance of normalized ranks. It is not a replacement for rank-histogram inspection.

### Usage

```
audit_sbc(x, bins = 10L, alpha = 0.01)
```

### Arguments

| | |
|---|---|
| x | Object to print, plot, summarize, or audit. |
| bins | Number of bins used by the diagnostic. |
| alpha | Significance or tail-probability level. |

### Value

An object of class "eye_sbc_audit", "data.frame", stored as a data frame, containing sBC rank uniformity and associated metadata needed to interpret the result.

## audit_signal_filter *Summarize a signal-filter audit*

### Description

Summarize a signal-filter audit

### Usage

```
audit_signal_filter(x)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |

### Value

A data frame containing a signal-filter audit. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## audit_temporal_leakage *Audit temporal leakage in a feature provenance table*

### Description

Leakage here means information becoming available after the declared outcome boundary. It is a data/analysis property and is not an allegation of misconduct.

### Usage

```
audit_temporal_leakage(provenance, allow_equal = TRUE, tolerance = 0)
```

### Arguments

| | |
|---|---|
| provenance | Output of [process_feature_time_provenance()] or compatible table. |
| allow_equal | Whether features available exactly at outcome time are allowed. |
| tolerance | Numeric tolerance in the provenance time unit. |

### Value

A named list with components "status", "n_features", "n_flagged", "flagged_fraction", "detail", "interpretation", containing temporal leakage in a feature provenance table and associated metadata or diagnostics.

## audit_validation_completion

*Audit whether a validation programme is complete*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
audit_validation_completion(x, thresholds = validation_thresholds(),
  empirical_reproduction = NULL)
```

### Arguments

| | |
|---|---|
| x | Validation collection. |
| thresholds | Validation thresholds. |
| empirical_reproduction | Optional empirical-reproduction evidence required when configured in 'thresholds'. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## audit_vendor_field_coverage

*Audit vendor field coverage against canonical semantics*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
audit_vendor_field_coverage(semantics, required_fields)
```

### Arguments

| | |
|---|---|
| semantics | Semantics registry or corpus path. |
| required_fields | Data frame with canonical table/field pairs. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`audit_vendor_validation` *Audit a multi-vendor validation corpus*

---

### Description

Audit a multi-vendor validation corpus

### Usage

```
audit_vendor_validation(x, spec = vendor_validation_spec())
```

### Arguments

| | |
|---|---|
| `x` | An ‘eye_corpus_validation‘ object or its summary data frame. |
| `spec` | Validation specification. |

### Value

An ‘eye_vendor_validation‘ data frame.

---

`audit_visual_context_dependence` *Audit visual-context dependence*

---

### Description

Audit visual-context dependence

### Usage

```
audit_visual_context_dependence(x)
```

### Arguments

| | |
|---|---|
| `x` | Object to process, inspect, compare, or plot. |

### Value

A data frame containing visual-context dependence. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## bayesian_process_diagnostics_dashboard

*Summarize Bayesian process-model diagnostics*

### Description

Collects model availability, optional approximate leave-one-out summaries, posterior diagnostics, and (only when explicitly requested) a Bayes-factor comparison. The function does not treat any one diagnostic as proof of the substantive process interpretation.

### Usage

```
bayesian_process_diagnostics_dashboard(
  ...,
  model_names = NULL,
  compute_loo = TRUE,
  compute_bayes_factor = FALSE,
  posterior_summary = TRUE
)
```

### Arguments

| | |
|---|---|
| `...` | Fitted ‘brmsfit‘ objects. |
| `model_names` | Optional model labels. |
| `compute_loo` | Whether to compute approximate leave-one-out diagnostics. |
| `compute_bayes_factor` | Whether to attempt a Bayes factor. This requires exactly two suitable brms models and typically models fitted with ‘save_pars = save_pars(all = TRUE)‘. |
| `posterior_summary` | Whether to collect posterior convergence summaries when package ‘posterior‘ is available. |

### Value

An ‘eye_bayesian_process_dashboard‘ object.

## bayesian_process_diagnostic_flags

*Extract compact Bayesian process-model diagnostic flags*

### Description

Extract compact Bayesian process-model diagnostic flags

### Usage

```
bayesian_process_diagnostic_flags(
  x,
  rhat_threshold = 1.01,
  ess_threshold = 400
)
```

### Arguments

| | |
|---|---|
| x | An ‘eye_bayesian_process_dashboard‘. |
| rhat_threshold | Review threshold for R-hat. |
| ess_threshold | Review threshold for bulk/tail effective sample size. |

### Value

An R object containing compact Bayesian process-model diagnostic flags. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

## benchmark_expected_outputs

*Return expected benchmark outputs*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
benchmark_expected_outputs(study = eyeprocess_benchmark_study())
```

### Arguments

| | |
|---|---|
| study | Benchmark object or path. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## benchmark_eyeprocess *Benchmark an eyeprocess operation*

### Description

Benchmark an eyeprocess operation

### Usage

```
benchmark_eyeprocess(expr, iterations = 5L, label = "operation")
```

### Arguments

| | |
|---|---|
| expr | Function with no arguments to benchmark. |
| iterations | Number of iterations. |
| label | Benchmark label. |

### Value

An 'eye_benchmark' data frame.

## benchmark_eye_storage *Benchmark storage formats and query operations*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
benchmark_eye_storage(x, formats = c("rds", "csv", "parquet"),
  partition_by = c("participant_id", "recording_id"), repetitions = 3L,
  directory = tempdir())
```

### Arguments

| | |
|---|---|
| x | Named list of tables or eye dataset. |
| formats | Formats to benchmark. |
| partition_by | Partition columns. |
| repetitions | Repetitions. |
| directory | Parent temporary directory. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## benchmark_memory_estimate

*Memory estimate for an R object or generated problem size*

### Description

Memory estimate for an R object or generated problem size

### Usage

```
benchmark_memory_estimate(x, generator = NULL)
```

### Arguments

| | |
|---|---|
| x | Object, or numeric n when ‘generator‘ is supplied. |
| generator | Optional function taking n. |

### Value

A data frame containing memory estimate for an R object or generated problem size. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## benchmark_scaling_curve

*Estimate scaling exponent from benchmark results*

### Description

Estimate scaling exponent from benchmark results

### Usage

```
benchmark_scaling_curve(x)
```

### Arguments

| | |
|---|---|
| x | Benchmark result. |

### Value

A data frame containing scaling exponent from benchmark results. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## bind_process_windows *Bind compatible process-window objects*

### Description

Bind compatible process-window objects

### Usage

```
bind_process_windows(...)
```

### Arguments

| | |
|---|---|
| ... | Additional arguments passed to the underlying method or helper. |

### Value

An object of class "eye_process_windows", stored as a named list, with components "data", "spec", "source_n", "status". It contains bind compatible process-window objects and associated metadata or diagnostics needed to interpret the result.

## biometric_imputation_sensitivity *Run biometric-feature imputation as a sensitivity analysis*

### Description

Run biometric-feature imputation as a sensitivity analysis

### Usage

```
biometric_imputation_sensitivity(
  data,
  variables,
  methods = c("mice", "missForest"),
  m = 3L,
  maxit = 3L,
  seed = 521
)
```

### Arguments

| | |
|---|---|
| `data` | Data containing biometric/process features. |
| `variables` | Variables to impute. |
| `methods` | Any of 'mice' and 'missForest'. |
| `m` | Number of MICE imputations. |
| `maxit` | Iteration count. |
| `seed` | Seed. |

### Value

An 'eye_biometric_imputation_sensitivity' object. Complete-case analysis is not replaced automatically.

---

## `bootstrap_process_reliability` *Bootstrap ICC reliability by resampling participants*

---

### Description

Bootstrap ICC reliability by resampling participants

### Usage

```
bootstrap_process_reliability(
  data,
  person,
  session,
  measure,
  replications = 500L,
  seed = 1L
)
```

### Arguments

| | |
|---|---|
| `data` | Long data. |
| `person`, `session`, `measure` | Column names. |
| `replications` | Bootstrap replications. |
| `seed` | Seed. |

### Value

A data frame containing bootstrap ICC reliability by resampling participants. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

`build_compatibility_matrix` *Build the declared/fixture/empirical compatibility matrix*

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
build_compatibility_matrix(x, required_vendors = c("gazepoint", "tobii", "pupillabs",
  "eyelink", "smi"), min_empirical_cases = 2L)
```

## Arguments

| | |
|---|---|
| `x` | Corpus path or registry data frame. |
| `required_vendors` | Required vendors. |
| `min_empirical_cases` | Minimum independent empirical cases per vendor. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

`build_evidence_graph` *Evidence and decision provenance graphs*

## Description

Evidence and decision provenance graphs. These functions form the eyeprocess 0.6.0.9000 measurement-intelligence programme and use dependency-free reference implementations with explicit evidence limits.

## Usage

```
build_evidence_graph(raw_data, transformations = NULL, metrics = NULL,
  models = NULL, diagnostics = NULL, decisions = NULL, edges = NULL)
trace_item_decision(graph, item_id)
compare_decision_provenance(graph_a, graph_b)
audit_evidence_dependencies(graph)
plot_evidence_graph(x, ...)
plot_item_decision_path(x, ...)
plot_metric_dependency_graph(x, ...)
plot_model_decision_impact(x, ...)
```

## Arguments

| | |
|---|---|
| raw_data | Argument controlling ‘raw_data‘; see the function usage and returned audit metadata. |
| transformations | Argument controlling ‘transformations‘; see the function usage and returned audit metadata. |
| metrics | Argument controlling ‘metrics‘; see the function usage and returned audit metadata. |
| models | Argument controlling ‘models‘; see the function usage and returned audit metadata. |
| diagnostics | Argument controlling ‘diagnostics‘; see the function usage and returned audit metadata. |
| decisions | Argument controlling ‘decisions‘; see the function usage and returned audit metadata. |
| edges | Argument controlling ‘edges‘; see the function usage and returned audit metadata. |
| graph | Argument controlling ‘graph‘; see the function usage and returned audit metadata. |
| item_id | Argument controlling ‘item_id‘; see the function usage and returned audit metadata. |
| graph_a | Argument controlling ‘graph_a‘; see the function usage and returned audit metadata. |
| graph_b | Argument controlling ‘graph_b‘; see the function usage and returned audit metadata. |
| x | Input object or data structure appropriate for the selected analysis. |
| ... | Additional arguments passed to the underlying method or plotting function. |

## Details

The APIs return auditable S3 objects. Plot wrappers call registered base-graphics methods. Experimental or approximate engines are labelled in object status fields and should be validated before confirmatory or operational use.

## Value

An eyeprocess result object, data frame, model object, plot, or audit table as documented by the individual function.

## See Also

`plot_diagnostics()`, `plot_evidence()`, and `plot_sensitivity()`.

## build_gazepoint_media_trials

*Reconstruct media presentations as analysis trials*

### Description

Converts contiguous Gazepoint media runs into explicit trial intervals. This provides stable person-by-item-by-trial keys even when the original export contains no behavioural response file.

### Usage

```
build_gazepoint_media_trials(x, item_map = NULL, overwrite = TRUE)
```

### Arguments

| | |
|---|---|
| x | An ‘eye_dataset‘ imported from Gazepoint. |
| item_map | Optional data frame or CSV path containing ‘stimulus_id‘, ‘item_id‘, and optionally ‘condition_id‘. |
| overwrite | Replace existing trial intervals. |

### Value

The updated ‘eye_dataset‘.

## calibration_drift_profile

*Calibration drift profile across sessions/batches*

### Description

Calibration drift profile across sessions/batches

### Usage

```
calibration_drift_profile(
  data,
  by,
  gaze_x = "gaze_x",
  gaze_y = "gaze_y",
  target_x = "target_x",
  target_y = "target_y"
)
```

### Arguments

| | |
|---|---|
| `data` | Validation-target data. |
| `by` | Ordered batch/session column. |
| `gaze_x, gaze_y` | Recorded gaze-coordinate columns. |
| `target_x, target_y` | Known target-coordinate columns. |

### Value

An object of class "eye_calibration_drift_profile", stored as a named list, with components "table", "by", "caveat". It contains calibration drift profile across sessions/batches and associated metadata or diagnostics needed to interpret the result.

---

## `calibration_error_model` *Build an empirical bivariate calibration-error model*

---

### Description

Build an empirical bivariate calibration-error model

### Usage

```
calibration_error_model(
  data,
  gaze_x = "gaze_x",
  gaze_y = "gaze_y",
  target_x = "target_x",
  target_y = "target_y"
)
```

### Arguments

| | |
|---|---|
| `data` | Validation-target data. |
| `gaze_x, gaze_y` | Recorded gaze-coordinate columns. |
| `target_x, target_y` | Known target-coordinate columns. |

### Value

An object of class "eye_calibration_error_model", stored as a named list, with components "mean_error", "covariance", "errors", "n", "metrics", "coordinate_units", "status", "caveat". It contains an empirical bivariate calibration-error model and associated metadata or diagnostics needed to interpret the result.

## calibration_sensitivity_grid

*Sensitivity grid for deterministic calibration offsets*

### Description

Sensitivity grid for deterministic calibration offsets

### Usage

```
calibration_sensitivity_grid(
  offset_x = c(-0.02, 0, 0.02),
  offset_y = c(-0.02, 0, 0.02)
)
```

### Arguments

| | |
|---|---|
| `offset_x`, `offset_y` | Candidate offsets in coordinate units. |

### Value

A tabular R object containing sensitivity grid for deterministic calibration offsets; rows represent analysis units and columns contain the returned quantities.

## calibration_transfer_audit

*Audit transfer of calibration across devices/sessions/sites*

### Description

Audit transfer of calibration across devices/sessions/sites

### Usage

```
calibration_transfer_audit(data, group, observed, predicted)
```

### Arguments

| | |
|---|---|
| `data` | Data containing group, observed and predicted values. |
| `group` | Grouping column. |
| `observed` | Observed binary/numeric outcome column. |
| `predicted` | Predicted probability/numeric score column. |

### Value

An object of class "eye_calibration_transfer_audit", "data.frame", stored as a data frame, containing transfer of calibration across devices/sessions/sites and associated metadata needed to interpret the result.

---

## canonical_eye_api *Canonical API mapping*

---

### Description

Canonical API mapping

### Usage

```
canonical_eye_api(registry = eye_api_lifecycle())
```

### Arguments

| | |
|---|---|
| registry | Lifecycle registry. |

### Value

A tabular R object containing canonical API mapping; rows represent analysis units and columns contain the returned quantities.

---

## classify_item_missingness *Classify item missingness using exposure and response evidence*

---

### Description

Classify item missingness using exposure and response evidence

### Usage

```
classify_item_missingness(
  data,
  response = "response",
  reached = "reached",
  inspected = NULL,
  started = NULL
)
```

### Arguments

| | |
|---|---|
| data | Input data frame or compatible tabular object. |
| response | Response variable or response-column name. |
| reached | Indicator that the item was reached. |
| inspected | Indicator that the item or response area was inspected. |
| started | Indicator that responding was initiated. |

### Value

An object of class "factor", stored as an R object, containing classify item missingness using exposure and response evidence and associated metadata needed to interpret the result.

---

## collect_eye_storage *Collect a disk-backed eye dataset*

---

### Description

Collect a disk-backed eye dataset

### Usage

```
collect_eye_storage(x, tables = NULL)
```

### Arguments

| | |
|---|---|
| x | An ‘eye_storage‘ handle or storage path. |
| tables | Optional subset of canonical tables. |

### Value

An ‘eye_dataset‘.

---

## collect_validation_evidence *Collect validation evidence into a common bundle*

---

### Description

Collect validation evidence into a common bundle

### Usage

```
collect_validation_evidence(..., model_name = NULL, notes = NULL)
```

### Arguments

| | |
|---|---|
| ... | Named validation objects. |
| model_name | Optional model/workflow label. |
| notes | Optional free-text notes. |

### Value

An ‘eye_validation_bundle‘ object.

---

## collect_validation_jobs *Collect validation checkpoints from one or more directories*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
collect_validation_jobs(path, plan = NULL, strict = TRUE)
```

### Arguments

| | |
|---|---|
| path | Manifest/output directory or character vector of directories. |
| plan | Optional validation plan. |
| strict | Fail when duplicate job identifiers disagree. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

## compare_aoi_methods *Compare explicit AOI assignment methods*

---

### Description

Compare explicit AOI assignment methods

### Usage

```
compare_aoi_methods(
  data,
  methods,
  analysis_fun,
  extract_fun = .ep09_default_sensitivity_extract
)
```

### Arguments

| | |
|---|---|
| data | Data. |
| methods | Named methods/specifications. |
| analysis_fun | Function ‘(data, method, specification)‘. |
| extract_fun | Result extractor. |

### Value

An object of class "eye_process_sensitivity", stored as a named list, with components "grid", "results", "failures", "warnings", "grid_hash", "created_at", "status", "caveat". It contains explicit AOI assignment methods and associated metadata or diagnostics needed to interpret the result.

---

## compare_aoi_trajectories

*Compare AOI trajectory feature objects*

---

### Description

Compare AOI trajectory feature objects

### Usage

```
compare_aoi_trajectories(...)
```

### Arguments

| | |
|---|---|
| ... | Additional arguments passed to the underlying method or helper. |

### Value

A data frame containing aOI trajectory feature objects. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

---

## compare_bayesian_process_models

*Compare Bayesian process models by LOO or Bayes factor*

---

### Description

Compare Bayesian process models by LOO or Bayes factor

### Usage

```
compare_bayesian_process_models(..., method = c("loo", "bayes_factor"))
```

### Arguments

| | |
|---|---|
| ... | Fitted brms models. |
| method | ‘loo‘ or ‘bayes_factor‘. |

### Value

An R object containing bayesian process models by LOO or Bayes factor. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

## compare_decision_manifests

*Compare two research decision manifests*

---

### Description

Compare two research decision manifests

### Usage

```
compare_decision_manifests(old, new)
```

### Arguments

| | |
|---|---|
| old | Earlier manifest. |
| new | Later manifest. |

### Value

An object of class "eye_decision_manifest_diff", "data.frame", stored as a data frame, containing two research decision manifests and associated metadata needed to interpret the result.

---

## compare_deployment_batches

*Compare two deployment batches descriptively*

---

### Description

Compare two deployment batches descriptively

### Usage

```
compare_deployment_batches(
  data,
  batch = "deployment_batch",
  batch_a,
  batch_b,
  metrics = NULL,
  item = "item_id"
)
```

### Arguments

| | |
|---|---|
| data | Deployment data. |
| batch | Batch column. |
| batch_a, batch_b | Values to compare. |
| metrics | Metrics to compare. |
| item | Optional item identifier for item-matched differences. |

### Value

A data frame containing two deployment batches descriptively. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

---

## compare_diffusion_accuracy_rt

*Compare diffusion and conventional accuracy-RT models*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
compare_diffusion_accuracy_rt(object)
```

### Arguments

| | |
|---|---|
| object | Diffusion fit. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

`compare_dynamic_transition_models` *Compare dynamic transition models*

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
compare_dynamic_transition_models(..., criterion = c("AIC", "BIC", "log_score",
  "accuracy"))
```

## Arguments

| | |
|---|---|
| `...` | Named fitted dynamic IRTree models. |
| `criterion` | AIC, BIC, log score, or classification accuracy. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

`compare_engine_adapters` *Compare multiple external-engine adapter results*

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
compare_engine_adapters(...)
```

## Arguments

| | |
|---|---|
| `...` | Adapter results or a list. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

compare_fixation_methods

*Compare explicit fixation-detection methods*

## Description

Compare explicit fixation-detection methods

## Usage

```
compare_fixation_methods(
  data,
  methods,
  analysis_fun,
  extract_fun = .ep09_default_sensitivity_extract
)
```

## Arguments

| | |
|---|---|
| data | Data. |
| methods | Named methods/specifications. |
| analysis_fun | Function '(data, method, specification)'. |
| extract_fun | Result extractor. |

## Value

An object of class "eye_process_sensitivity", stored as a named list, with components "grid", "results", "failures", "warnings", "grid_hash", "created_at", "status", "caveat". It contains explicit fixation-detection methods and associated metadata or diagnostics needed to interpret the result.

compare_functional_scalar_models

*Compare functional and scalar pupil summaries*

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
compare_functional_scalar_models(x, scalar_features = c("pupil_peak", "pupil_auc",
  "pupil_mean"), criterion = c("AIC", "log_loss"), folds = 5L, seed = 1L)
```

### Arguments

| | |
|---|---|
| x | Functional pupil fit or prepared data. |
| scalar_features | Scalar feature names. |
| criterion | AIC or cross-validated log loss. |
| folds | Grouped folds for cross-validation. |
| seed | Random seed. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

## compare_hard_probabilistic_aoi

*Compare hard and probabilistic AOI assignments*

---

### Description

Compare hard and probabilistic AOI assignments

### Usage

```
compare_hard_probabilistic_aoi(
  data,
  aois,
  probabilistic,
  x = "gaze_x",
  y = "gaze_y"
)
```

### Arguments

| | |
|---|---|
| data | Gaze samples. |
| aois | Rectangular AOIs. |
| probabilistic | Probabilistic assignment object. |
| x, y | Gaze-coordinate columns. |

### Value

An R object containing hard and probabilistic AOI assignments. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## compare_irt_models *Compare multimodal IRT model objects*

### Description

Uses supplied scoring functions when models expose different engines. The default extracts AIC/BIC/logLik when available and never treats in-sample fit as sufficient evidence for model promotion.

### Usage

```
compare_irt_models(..., names = NULL)
```

### Arguments

| | |
|---|---|
| ... | Additional arguments passed to the selected model, engine, or method. |
| names | Value supplied to 'names'; see Details for its model-specific role. |

### Value

An R object containing multimodal IRT model objects. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## compare_latent_distribution_models *Compare simple latent-distribution reference models*

### Description

Compares Gaussian, location/scale Student-t, and two-normal-mixture reference densities by AIC/BIC. This is a stress-test diagnostic; it does not change the latent distribution inside an already fitted IRT model.

### Usage

```
compare_latent_distribution_models(theta)
```

### Arguments

| | |
|---|---|
| theta | Numeric latent-trait draws/estimates. |

### Value

An object of class "eye_latent_distribution_comparison", stored as a named list, with components "comparison", "audit", "student_t", "mixture", "status". It contains simple latent-distribution reference models and associated metadata or diagnostics needed to interpret the result.

## compare_model_engines *Compare equivalent model engines*

### Description

Compare equivalent model engines

### Usage

```
compare_model_engines(
  data,
  engines,
  extractors,
  reference = names(engines)[1L],
  tolerance = 0.05
)
```

### Arguments

| | |
|---|---|
| data | Model-ready data. |
| engines | Named list of fitting functions receiving ‘data‘. |
| extractors | Named list of extractor functions or one shared extractor. |
| reference | Optional reference engine name. |
| tolerance | Maximum absolute estimate difference for equivalence. |

### Value

An ‘eye_engine_comparison‘ object.

## compare_parametric_nonparametric_irf *Compare conventional logistic and flexible IRF shapes*

### Description

Compare conventional logistic and flexible IRF shapes

### Usage

```
compare_parametric_nonparametric_irf(
  response_matrix,
  gpirt_object = NULL,
  theta_grid = seq(-4, 4, length.out = 101)
)
```

### Arguments

| | |
|---|---|
| response_matrix | Person-by-item response matrix. |
| gpirt_object | Value supplied to ‘gpirt_object‘; see Details for its model-specific role. |
| theta_grid | Grid of latent-trait values used for evaluation. |

### Value

An object of class "eye_irf_comparison", "data.frame", stored as a data frame, containing conventional logistic and flexible IRF shapes and associated metadata needed to interpret the result.

---

## compare_presentation_fairness

*Compare outcomes across presentation variants*

---

### Description

Compare outcomes across presentation variants

### Usage

```
compare_presentation_fairness(data, variant, outcome, person = NULL)
```

### Arguments

| | |
|---|---|
| data | Data containing presentation version and an outcome. |
| variant | Presentation-version column. |
| outcome | Numeric outcome. |
| person | Optional participant column for descriptive aggregation. |

### Value

An object of class "eye_presentation_fairness_comparison", stored as a named list, with components "model", "summary", "status", "caveat". It contains outcomes across presentation variants and associated metadata or diagnostics needed to interpret the result.

## compare_process_criterion_models

*Compare process external-validity models*

### Description

Compare process external-validity models

### Usage

```
compare_process_criterion_models(x)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |

### Value

A data frame containing process external-validity models. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## compare_process_models

*Compare explicit process-model specifications*

### Description

Compare explicit process-model specifications

### Usage

```
compare_process_models(
  data,
  methods,
  analysis_fun,
  extract_fun = .ep09_default_sensitivity_extract
)
```

### Arguments

| | |
|---|---|
| data | Data. |
| methods | Named methods/specifications. |
| analysis_fun | Function ‘(data, method, specification)‘. |
| extract_fun | Result extractor. |

### Value

An object of class "eye_process_sensitivity", stored as a named list, with components "grid", "results", "failures", "warnings", "grid_hash", "created_at", "status", "caveat". It contains explicit process-model specifications and associated metadata or diagnostics needed to interpret the result.

---

## compare_process_profile_solutions

*Compare candidate process-profile solutions*

---

### Description

Compare candidate process-profile solutions

### Usage

```
compare_process_profile_solutions(data, variables, k_values = 2:6, seed = 777)
```

### Arguments

| | |
|---|---|
| data | Person-level process data. |
| variables | Variables used for profiling. |
| k_values | Candidate numbers of profiles. |
| seed | Seed. |

### Value

A tabular R object containing candidate process-profile solutions; rows represent analysis units and columns contain the returned quantities.

---

## compare_pupil_kernels *Compare pupil deconvolution kernels*

---

### Description

Compare pupil deconvolution kernels

### Usage

```
compare_pupil_kernels(data, tmax_values = c(512, 930), ...)
```

### Arguments

| | |
|---|---|
| data | Same input used for fitting. |
| tmax_values | Candidate peak times. |
| ... | Passed to ‘fit_pupil_event_deconvolution()‘. |

## Value

A tabular R object containing pupil deconvolution kernels; rows represent analysis units and columns contain the returned quantities.

---

`compare_pupil_preprocessing` *Compare explicit pupil-preprocessing methods*

---

## Description

Compare explicit pupil-preprocessing methods

## Usage

```
compare_pupil_preprocessing(
  data,
  methods,
  analysis_fun,
  extract_fun = .ep09_default_sensitivity_extract
)
```

## Arguments

| | |
|---|---|
| `data` | Data. |
| `methods` | Named methods/specifications. |
| `analysis_fun` | Function ‘(data, method, specification)‘. |
| `extract_fun` | Result extractor. |

## Value

An object of class "eye_process_sensitivity", stored as a named list, with components "grid", "results", "failures", "warnings", "grid_hash", "created_at", "status", "caveat". It contains explicit pupil-preprocessing methods and associated metadata or diagnostics needed to interpret the result.

---

`compare_raw_adjusted_pupil` *Compare raw and confound-adjusted pupil values*

---

## Description

Compare raw and confound-adjusted pupil values

## Usage

```
compare_raw_adjusted_pupil(x)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |

### Value

A data frame containing raw and confound-adjusted pupil values. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

---

## compare_reproducibility_fingerprints

*Compare two reproducibility fingerprints*

---

### Description

Compare two reproducibility fingerprints

### Usage

```
compare_reproducibility_fingerprints(old, new)
```

### Arguments

| | |
|---|---|
| old, new | Fingerprints. |

### Value

A named list with components "detail", "identical", "old_hash", "new_hash", containing two reproducibility fingerprints and associated metadata or diagnostics.

---

## compare_signal_filters

*Compare multiple signal filters*

---

### Description

Compare multiple signal filters

### Usage

```
compare_signal_filters(
  signal,
  widths = c(5L, 9L, 15L),
  methods = c("runmed", "robfilter")
)
```

### Arguments

| | |
|---|---|
| signal | Numeric signal. |
| widths | Widths to compare. |
| methods | Methods to compare. |

### Value

A tabular R object containing multiple signal filters; rows represent analysis units and columns contain the returned quantities.

---

## compare_strategy_heterogeneity

*Compare mixture and continuous heterogeneity descriptions*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
compare_strategy_heterogeneity(object)
```

### Arguments

| | |
|---|---|
| object | Strategy fit. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

## compare_validation_engines

*Compare validation engines on common recovery output*

---

### Description

Compare validation engines on common recovery output

### Usage

```
compare_validation_engines(results)
```

### Arguments

| | |
|---|---|
| results | Validation or model results. |

### Value

An object of class "eye_validation_engine_comparison", "data.frame", stored as a data frame, containing validation engines on common recovery output and associated metadata needed to interpret the result.

---

`compare_vendor_semantics`

*Compare semantic mappings between vendors*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
compare_vendor_semantics(x, vendors = NULL)
```

### Arguments

| | |
|---|---|
| `x` | Corpus path or semantics data frame. |
| `vendors` | Optional vendor subset. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`compare_visual_context_irt`

*Compare base and visual-context IRT models*

---

### Description

Compare base and visual-context IRT models

### Usage

```
compare_visual_context_irt(x)
```

### Arguments

| | |
|---|---|
| `x` | Object to process, inspect, compare, or plot. |

### Value

A data frame containing base and visual-context IRT models. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## compatibility_evidence_matrix

*Build a detailed compatibility evidence matrix*

### Description

Build a detailed compatibility evidence matrix

### Usage

```
compatibility_evidence_matrix(compatibility, evidence = NULL)
```

### Arguments

| | |
|---|---|
| compatibility | Existing output from 'build_compatibility_matrix()' or a compatible data frame. |
| evidence | Case-level evidence with columns 'ecosystem', 'device', 'evidence_level', and optionally 'semantic_roundtrip_pass'. |

### Value

An 'eye_compatibility_evidence_matrix' object.

## context_factor_effects

*Extract visual-context factor effects/loadings*

### Description

Extract visual-context factor effects/loadings

### Usage

```
context_factor_effects(x, IRTpars = FALSE)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |
| IRTpars | Passed to the underlying IRT coefficient extractor to request IRT parameterization when supported. |

### Value

An R object containing visual-context factor effects/loadings. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

`coordinate_fidelity_audit` *Coordinate semantic-fidelity audit*

## Description

Detects lossless preservation and stable affine coordinate transformations.

## Usage

```
coordinate_fidelity_audit(
  source,
  roundtrip,
  source_x = "x",
  source_y = "y",
  roundtrip_x = source_x,
  roundtrip_y = source_y,
  key = NULL,
  tolerance = 1e-06,
  correlation_floor = 0.999
)
```

## Arguments

| | |
|---|---|
| `source` | Original/source representation. |
| `roundtrip` | Round-tripped or comparison representation. |
| `source_x` | Source horizontal coordinate column. |
| `source_y` | Source vertical coordinate column. |
| `roundtrip_x` | Round-tripped horizontal coordinate column. |
| `roundtrip_y` | Round-tripped vertical coordinate column. |
| `key` | Column or columns used to align records. |
| `tolerance` | Numerical tolerance used by the comparison. |
| `correlation_floor` | Minimum correlation treated as compatible. |

## Value

An object of class "eye_coordinate_fidelity", stored as a named list, with components "status", "x", "y", "matched_n", "tolerance", "correlation_floor". It contains coordinate semantic-fidelity audit and associated metadata or diagnostics needed to interpret the result.

coverage_calibration_curve *Interval coverage calibration curve*

## Description

Interval coverage calibration curve

## Usage

```
coverage_calibration_curve(truth, lower, upper, nominal = NULL)
```

## Arguments

| | |
|---|---|
| truth | True values. |
| lower | Matrix/data.frame of lower limits or numeric vector. |
| upper | Matrix/data.frame of upper limits or numeric vector. |
| nominal | Nominal coverage labels, one per interval column. |

## Value

A data frame containing interval coverage calibration curve. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

create_public_benchmark *Create a public, de-identified benchmark bundle*

## Description

Create a public, de-identified benchmark bundle

## Usage

```
create_public_benchmark(
  x,
  path,
  max_participants = 50L,
  include_samples = FALSE,
  overwrite = FALSE
)
```

### Arguments

| | |
|---|---|
| x | An ‘eye_dataset‘. |
| path | Output directory. |
| max_participants | Optional participant cap. |
| include_samples | Whether to include sample-level tables. |
| overwrite | Whether to replace the output directory. |

### Value

Output directory.

---

## crossed_grouped_cv *Cross-classified grouped cross-validation*

---

### Description

Cross-classified grouped cross-validation

### Usage

```
crossed_grouped_cv(
  data,
  formula,
  family = stats::binomial(),
  groups = c("participant_id", "item_id"),
  v = 5L,
  metric = c("log_loss", "brier", "accuracy"),
  seed = 1L
)
```

### Arguments

| | |
|---|---|
| data | Data frame. |
| formula | Model formula. |
| family | GLM family. |
| groups | Crossed grouping columns. |
| v | Number of folds. |
| metric | Metric: log loss, Brier score, or accuracy. |
| seed | Random seed. |

### Value

An ‘eye_crossed_grouped_cv‘ object.

## crossed_grouped_folds *Create cross-classified grouped folds*

### Description

Holds out levels from every declared grouping dimension simultaneously. The assessment set is the intersection of held-out levels; the analysis set excludes every held-out level. Rows combining held-out and retained levels form a buffer and are deliberately used in neither set.

### Usage

```
crossed_grouped_folds(
  data,
  groups = c("participant_id", "item_id"),
  v = 5L,
  seed = 1L
)
```

### Arguments

| | |
|---|---|
| data | Data frame. |
| groups | Two or more crossed grouping columns. |
| v | Number of folds. |
| seed | Random seed. |

### Value

An ‘eye_crossed_grouped_folds‘ object.

## cross_device_process_equating_audit *Cross-device process-scale equating audit*

### Description

Fits a simple affine linking map on anchor observations and reports residual bias/RMSE by device. It complements, rather than replaces, IRT anchor linking.

### Usage

```
cross_device_process_equating_audit(
  data,
  value,
  reference_value,
  device,
  anchor = NULL
)
```

### Arguments

| | |
|---|---|
| data | Input data frame or compatible tabular object. |
| value | Process-value column or values. |
| reference_value | Value supplied to 'reference_value'; see Details for its model-specific role. |
| device | Device identifier or device facet. |
| anchor | Anchor or reference group used for linking. |

### Value

An object of class "eye_cross_device_equating_audit", "data.frame", stored as a data frame, containing cross-device process-scale equating audit and associated metadata needed to interpret the result.

---

## cross_version_adapter_regression

*Compare adapter output across software/format versions*

---

### Description

Compare adapter output across software/format versions

### Usage

```
cross_version_adapter_regression(
  input,
  baseline_adapter,
  candidate_adapter,
  baseline_version = "baseline",
  candidate_version = "candidate",
  extract_samples = .ep07_adapter_extract_samples,
  audit_args = list()
)
```

### Arguments

| | |
|---|---|
| input | Shared raw fixture/input. |
| baseline_adapter, candidate_adapter | Functions that parse 'input'. |
| baseline_version, candidate_version | Version labels. |
| extract_samples | Function extracting comparable sample tables. |
| audit_args | Arguments passed to 'field_fidelity_report()'. |

### Value

An object of class "eye_adapter_regression_audit", stored as a named list, with components "status", "baseline_version", "candidate_version", "fidelity", "baseline", "candidate". It contains adapter output across software/format versions and associated metadata or diagnostics needed to interpret the result.

---

## data_quality_reporting_table

*Compact reporting table for eye-tracking data quality*

---

### Description

Compact reporting table for eye-tracking data quality

### Usage

```
data_quality_reporting_table(x)
```

### Arguments

| | |
|---|---|
| x | Data-quality profile. |

### Value

An R object containing compact reporting table for eye-tracking data quality. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

## decision_manifest_diff

*Alias for manifest comparison emphasizing changed decision paths*

---

### Description

Alias for manifest comparison emphasizing changed decision paths

### Usage

```
decision_manifest_diff(old, new)
```

### Arguments

| | |
|---|---|
| old | Earlier manifest. |
| new | Later manifest. |

## Value

An R object containing alias for manifest comparison emphasizing changed decision paths. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

`decision_manifest_hash`

*Stable hash of decision content*

---

## Description

Stable hash of decision content

## Usage

```
decision_manifest_hash(x)
```

## Arguments

| | |
|---|---|
| `x` | Manifest or decision object. |

## Value

An R object containing stable hash of decision content. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

`decision_manifest_table`

*Flatten a decision manifest to a table*

---

## Description

Flatten a decision manifest to a table

## Usage

```
decision_manifest_table(x)
```

## Arguments

| | |
|---|---|
| `x` | Manifest. |

## Value

A logical value or vector indicating flatten a decision manifest to a table.

## decision_space_coverage *Coverage of a declared decision space by evaluated specifications*

### Description

Coverage of a declared decision space by evaluated specifications

### Usage

```
decision_space_coverage(grid, evaluated)
```

### Arguments

| | |
|---|---|
| grid | Sensitivity grid. |
| evaluated | Sensitivity result or vector of evaluated specification IDs. |

### Value

A data frame containing coverage of a declared decision space by evaluated specifications. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## decision_stability *Overall decision-stability summary*

### Description

Overall decision-stability summary

### Usage

```
decision_stability(
  x,
  effect = "effect",
  p_value = NULL,
  alpha = 0.05,
  threshold = 0
)
```

### Arguments

| | |
|---|---|
| x | Sensitivity result. |
| effect | Effect column. |
| p_value | Optional p-value column. |
| alpha | Significance threshold. |
| threshold | Substantive threshold. |

## Value

An object of class "eye_decision_stability", stored as a named list, with components "summary", "stable_sign", "stable_threshold", "stable_significance", "thresholds", "caveat". It contains overall decision-stability summary and associated metadata or diagnostics needed to interpret the result.

---

`decode_dynamic_states` *Decode latent or fitted transition states*

---

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
decode_dynamic_states(object, method = c("mode", "probability", "draw"))
```

## Arguments

| | |
|---|---|
| `object` | Dynamic IRTree fit. |
| `method` | Marginal probabilities, posterior mode, or draw. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`derive_aoi_composition`

*Compositional analysis of AOI attention*

---

## Description

Compositional analysis of AOI attention. These functions form the eyeprocess 0.6.0.9000 measurement-intelligence programme and use dependency-free reference implementations with explicit evidence limits.

## Usage

```
derive_aoi_composition(x, aois, denominator = c("total_aoi_dwell",
  "trial_duration"), zero_method = c("multiplicative", "bayesian"), id_cols = NULL,
   aoi_col = "aoi", value_col = "dwell_ms", trial_duration_col = NULL)
transform_aoi_composition(x, method = c("ilr", "clr", "alr"), reference = NULL)
fit_aoi_compositional_model(composition, formula, random = NULL, data = NULL,
  method = "ilr")
compare_aoi_compositions(x, group, method = c("permanova", "compositional_manova"),
  permutations = 499, seed = 20260807)
aoi_balance_coordinates(x, balances)
plot_aoi_ternary(x, ...)
plot_aoi_balance_biplot(x, ...)
plot_aoi_variation_matrix(x, ...)
plot_compositional_group_difference(x, ...)
plot_aoi_composition_trajectory(x, ...)
```

## Arguments

| | |
|---|---|
| x | Input object or data structure appropriate for the selected analysis. |
| aois | Argument controlling 'aois'; see the function usage and returned audit metadata. |
| denominator | Argument controlling 'denominator'; see the function usage and returned audit metadata. |
| zero_method | Argument controlling 'zero_method'; see the function usage and returned audit metadata. |
| id_cols | Argument controlling 'id_cols'; see the function usage and returned audit metadata. |
| aoi_col | Argument controlling 'aoi_col'; see the function usage and returned audit metadata. |
| value_col | Argument controlling 'value_col'; see the function usage and returned audit metadata. |
| trial_duration_col | Argument controlling 'trial_duration_col'; see the function usage and returned audit metadata. |
| method | Argument controlling 'method'; see the function usage and returned audit metadata. |
| reference | Argument controlling 'reference'; see the function usage and returned audit metadata. |
| composition | Argument controlling 'composition'; see the function usage and returned audit metadata. |
| formula | Argument controlling 'formula'; see the function usage and returned audit metadata. |
| random | Argument controlling 'random'; see the function usage and returned audit metadata. |
| data | Argument controlling 'data'; see the function usage and returned audit metadata. |

| | |
|---|---|
| group | Argument controlling 'group'; see the function usage and returned audit metadata. |
| permutations | Argument controlling 'permutations'; see the function usage and returned audit metadata. |
| seed | Argument controlling 'seed'; see the function usage and returned audit metadata. |
| balances | Argument controlling 'balances'; see the function usage and returned audit metadata. |
| ... | Additional arguments passed to the underlying method or plotting function. |

## Details

The APIs return auditable S3 objects. Plot wrappers call registered base-graphics methods. Experimental or approximate engines are labelled in object status fields and should be validated before confirmatory or operational use.

## Value

An eyeprocess result object, data frame, model object, plot, or audit table as documented by the individual function.

## See Also

`plot_diagnostics()`, `plot_evidence()`, and `plot_sensitivity()`.

---

# derive_gazepoint_workflow_features

*Derive the complete Gazepoint workflow feature set*

---

## Description

Derive the complete Gazepoint workflow feature set

## Usage

```
derive_gazepoint_workflow_features(x, reset_workflow_features = TRUE)
```

## Arguments

| | |
|---|---|
| x | An 'eye_dataset' with reconstructed media trials. |
| reset_workflow_features | Remove features previously generated by this workflow while preserving native Gazepoint Data Summary features. |

## Value

The updated 'eye_dataset'.

`detect_calibration_drift` *Calibration drift and offline recalibration*

## Description

Calibration drift and offline recalibration. These functions form the eyeprocess 0.6.0.9000 measurement-intelligence programme and use dependency-free reference implementations with explicit evidence limits.

## Usage

```
detect_calibration_drift(x, references = NULL, window = "30 sec",
  method = c("targets", "known_aois", "fixation_density"), x_col = NULL,
  y_col = NULL, time_col = NULL)
fit_offline_recalibration(x, method = c("translation", "affine", "polynomial"),
  robust = TRUE, x_col = NULL, y_col = NULL, reference_x_col = NULL,
  reference_y_col = NULL)
apply_offline_recalibration(x, model, x_col = NULL, y_col = NULL,
  suffix = "_recalibrated")
audit_recalibration(before, after, minimum_improvement = NULL)
plot_calibration_vector_field(x, ...)
plot_calibration_error_ellipses(x, ...)
plot_drift_over_time(x, ...)
plot_recalibration_before_after(x, ...)
plot_screen_coverage(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Input object or data structure appropriate for the selected analysis. |
| `references` | Argument controlling 'references'; see the function usage and returned audit metadata. |
| `window` | Argument controlling 'window'; see the function usage and returned audit metadata. |
| `method` | Argument controlling 'method'; see the function usage and returned audit metadata. |
| `x_col` | Argument controlling 'x_col'; see the function usage and returned audit metadata. |
| `y_col` | Argument controlling 'y_col'; see the function usage and returned audit metadata. |
| `time_col` | Argument controlling 'time_col'; see the function usage and returned audit metadata. |
| `robust` | Argument controlling 'robust'; see the function usage and returned audit metadata. |

`reference_x_col`
: Argument controlling 'reference_x_col'; see the function usage and returned audit metadata.

`reference_y_col`
: Argument controlling 'reference_y_col'; see the function usage and returned audit metadata.

`model`
: Argument controlling 'model'; see the function usage and returned audit metadata.

`suffix`
: Argument controlling 'suffix'; see the function usage and returned audit metadata.

`before`
: Argument controlling 'before'; see the function usage and returned audit metadata.

`after`
: Argument controlling 'after'; see the function usage and returned audit metadata.

`minimum_improvement`
: Argument controlling 'minimum_improvement'; see the function usage and returned audit metadata.

`...`
: Additional arguments passed to the underlying method or plotting function.

## Details

The APIs return auditable S3 objects. Plot wrappers call registered base-graphics methods. Experimental or approximate engines are labelled in object status fields and should be validated before confirmatory or operational use.

## Value

An eyeprocess result object, data frame, model object, plot, or audit table as documented by the individual function.

## See Also

`plot_diagnostics()`, `plot_evidence()`, and `plot_sensitivity()`.

---

# `detect_corrupt_partitions` *Detect missing, truncated, or modified partitions*

---

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
detect_corrupt_partitions(storage)
```

### Arguments

| | |
|---|---|
| storage | Storage object or path. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

## detect_irt_changepoints

*Detect IRT/process change points using an SIC-inspired multichannel score*

---

### Description

This implementation is a transparent package reference inspired by the 2026 SIC-CPA literature. It combines Bernoulli response likelihood with normal log-RT and standardized gaze likelihoods. It is not a line-for-line reproduction of the article's estimator.

### Usage

```
detect_irt_changepoints(
  data,
  person = "participant_id",
  order = "item_order",
  response = "response",
  rt = "rt",
  gaze = NULL,
  min_segment = 5L,
  min_delta_sic = 2,
  max_changes = 2L
)
```

### Arguments

| | |
|---|---|
| data | Input data frame or compatible tabular object. |
| person | Person or participant identifier column. |
| order | Within-sequence ordering variable. |
| response | Response variable or response-column name. |
| rt | Response-time variable or column name. |
| gaze | Gaze/process variable or column name. |
| min_segment | Minimum segment length. |
| min_delta_sic | Minimum information-criterion improvement. |
| max_changes | Maximum number of change points. |

## Value

An object of class "eye_irt_changepoints", stored as a named list, with components "results", "channels", "method", "min_segment", "min_delta_sic", "max_changes". It contains iRT/process change points using an SIC-inspired multichannel score and associated metadata or diagnostics needed to interpret the result.

---

`detect_process_changepoint`

*Detect a response-process change point*

---

## Description

Public roadmap alias for ‘detect_irt_changepoints()‘.

## Usage

```
detect_process_changepoint(...)
```

## Arguments

| | |
|---|---|
| ... | Additional arguments passed to the selected model, engine, or method. |

## Value

An R object containing a response-process change point. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

`detect_process_changepoints`

*Cognitive-episode change-point detection*

---

## Description

Cognitive-episode change-point detection. These functions form the eyeprocess 0.6.0.9000 measurement-intelligence programme and use dependency-free reference implementations with explicit evidence limits.

## Usage

```
detect_process_changepoints(x, channels = c("gaze_velocity", "aoi", "pupil", "eda"),
  time_col = NULL, window = 10, threshold_quantile = 0.9, min_segment = 5)
segment_process_episodes(x, ...)
label_process_episodes(x, rules = NULL, model = NULL)
compare_episode_structure(x, group)
plot_process_episodes(x, ...)
plot_changepoint_ribbons(x, ...)
plot_episode_waterfall(x, ...)
plot_episode_transition_graph(x, ...)
plot_episode_duration_distribution(x, ...)
```

## Arguments

| | |
|---|---|
| x | Input object or data structure appropriate for the selected analysis. |
| channels | Argument controlling 'channels'; see the function usage and returned audit metadata. |
| time_col | Argument controlling 'time_col'; see the function usage and returned audit metadata. |
| window | Argument controlling 'window'; see the function usage and returned audit metadata. |
| threshold_quantile | Argument controlling 'threshold_quantile'; see the function usage and returned audit metadata. |
| min_segment | Argument controlling 'min_segment'; see the function usage and returned audit metadata. |
| ... | Additional arguments passed to the underlying method or plotting function. |
| rules | Argument controlling 'rules'; see the function usage and returned audit metadata. |
| model | Argument controlling 'model'; see the function usage and returned audit metadata. |
| group | Argument controlling 'group'; see the function usage and returned audit metadata. |

## Details

The APIs return auditable S3 objects. Plot wrappers call registered base-graphics methods. Experimental or approximate engines are labelled in object status fields and should be validated before confirmatory or operational use.

## Value

An eyeprocess result object, data frame, model object, plot, or audit table as documented by the individual function.

## See Also

plot_diagnostics(), plot_evidence(), and plot_sensitivity().

---

device_facet_effects *Extract device facet effects*

---

## Description

Extract device facet effects

## Usage

```
device_facet_effects(object, channel = c("response", "process"))
```

## Arguments

| | |
|---|---|
| object | A fitted eyeprocess model or audit object. |
| channel | Measurement channel to inspect. |

## Value

An object of class "eye_process_facet_effects", stored as a named list, with components "facet", "column", "channel", "random_effects", "variance_component". It contains device facet effects and associated metadata or diagnostics needed to interpret the result.

---

diffusion_identification_study
*Construct a simulation-based identification study*

---

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
diffusion_identification_study(conditions = list(n_person = c(50L, 150L),
 n_item = c(10L, 30L), gaze_effect = c(0, 0.35), contaminant_fraction = c(0, 0.05)),
  replications = 20L, base_seed = 20260805L,
  spec = gaze_diffusion_spec(drift_features = "gaze_balance", engine = "stan"))
```

### Arguments

| | |
|---|---|
| conditions | Named list or data frame of design conditions. |
| replications | Replications per condition. |
| base_seed | Base seed. |
| spec | Confirmatory diffusion specification used for each fit. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

## diffusion_parameter_diagnostics

*Diagnose diffusion-parameter trade-offs and sampling*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
diffusion_parameter_diagnostics(object, correlation_threshold = 0.85)
```

### Arguments

| | |
|---|---|
| object | Diffusion fit. |
| correlation_threshold | Correlation threshold. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

`diffusion_posterior_predictive` *Posterior predictive summaries for accuracy and RT*

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
diffusion_posterior_predictive(object, draws = 200L, method = c("rtdists",
  "stan_proxy"), seed = 1L)
```

## Arguments

| | |
|---|---|
| `object` | Diffusion fit. |
| `draws` | Number of generated-quantity draws to retain. |
| `method` | Value for 'method'. See the function description and relevant article for constraints. |
| `seed` | Value for 'seed'. See the function description and relevant article for constraints. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

`distractor_process_map` *Build a distractor process map*

## Description

Build a distractor process map

## Usage

```
distractor_process_map(object)
```

## Arguments

| | |
|---|---|
| `object` | A fitted eyeprocess model or audit object. |

## Value

A data frame containing a distractor process map. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## drift_by_device *Drift audit stratified by device*

### Description

Drift audit stratified by device

### Usage

```
drift_by_device(data, device = "device_id", ...)
```

### Arguments

| | |
|---|---|
| data | Data frame containing the required process variables. |
| device | Name of the column identifying device. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

An R object containing drift audit stratified by device. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## drift_by_site *Drift audit stratified by study site*

### Description

Drift audit stratified by study site

### Usage

```
drift_by_site(data, site = "site_id", ...)
```

### Arguments

| | |
|---|---|
| data | Data frame containing the required process variables. |
| site | Name of the column identifying site. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

An R object containing drift audit stratified by study site. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## drift_by_stimulus_version *Drift audit stratified by stimulus version*

### Description

Drift audit stratified by stimulus version

### Usage

```
drift_by_stimulus_version(data, stimulus_version = "stimulus_version", ...)
```

### Arguments

| | |
|---|---|
| data | Data frame containing the required process variables. |
| stimulus_version | Name of the column identifying stimulus version. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

An R object containing drift audit stratified by stimulus version. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## drift_by_vendor *Drift audit stratified by vendor*

### Description

Drift audit stratified by vendor

### Usage

```
drift_by_vendor(data, vendor = "vendor", ...)
```

### Arguments

| | |
|---|---|
| data | Data frame containing the required process variables. |
| vendor | Name of the column identifying vendor. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

An R object containing drift audit stratified by vendor. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## dynamic_irtree_recovery

*Evaluate dynamic-state recovery under misclassification*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
dynamic_irtree_recovery(grid = expand.grid(state_misclassification = c(0, 0.05, 0.15),
  missing_state = c(0, 0.10), stringsAsFactors = FALSE), replications = 20L,
  spec = dynamic_irtree_spec(engine = "multinomial"), base_seed = 1L)
```

### Arguments

| | |
|---|---|
| grid | Scenario grid. |
| replications | Replications per scenario. |
| spec | Dynamic IRTree specification. |
| base_seed | Base seed. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## dynamic_irtree_spec *Specify a hardened dynamic gaze-state IRTree*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
dynamic_irtree_spec(source = c("samples", "visits", "fixations"),
  collapse_consecutive = TRUE, engine = c("baseline", "multinomial", "stan"),
  hidden_states = 0L, include_response = TRUE, include_person = FALSE,
  include_item = TRUE, condition_columns = character(),
  transition_predictors = character(), interactions = character(),
  include_time_gap = TRUE, person_effect = c("none", "fixed", "random"),
  item_effect = c("none", "fixed", "random"), structural_zeros = NULL,
  allowed_transitions = NULL, hidden_structural_zeros = NULL,
  hidden_allowed_transitions = NULL, missing_state = c("drop", "unknown",
```

```
"marginalize"), uncertain_state_probability = NULL, misclassification_matrix = NULL,
 ridge = 1e-4, standardize = TRUE, reference_state = NULL, chains = 4L,
 parallel_chains = chains, iter_warmup = 1000L, iter_sampling = 1000L,
 adapt_delta = 0.95, max_treedepth = 12L)
```

## Arguments

| | |
|---|---|
| source | Sequence source for an 'eye_dataset'. |
| collapse_consecutive | Collapse consecutive identical states. |
| engine | Estimation engine: auditable binary baseline, penalized |
| hidden_states | Number of latent states. Zero fits observed-state |
| include_response | Include item response as a predictor. |
| include_person | Include person effects. |
| include_item | Include item effects. |
| condition_columns | Condition-level predictors. |
| transition_predictors | Additional transition-level predictors. |
| interactions | Optional interaction terms supplied as formula strings. |
| include_time_gap | Include log time gap for irregular observations. |
| person_effect | Person effect type for Stan. |
| item_effect | Item effect type for Stan. |
| structural_zeros | Optional forbidden observed-state transition pairs. |
| allowed_transitions | Optional allowed observed-state transition pairs. |
| hidden_structural_zeros | Optional forbidden hidden-state pairs named 'state1', 'state2', and so forth. |
| hidden_allowed_transitions | Optional explicitly allowed hidden-state pairs. |
| missing_state | Treatment of missing observed states. |
| uncertain_state_probability | Optional column containing probability of |
| misclassification_matrix | Optional observed-state misclassification matrix. |
| ridge | Penalization for the multinomial baseline. |
| standardize | Standardize numeric design columns. |
| reference_state | Optional reference destination state. |
| chains | CmdStan controls. |

| | |
|---|---|
| parallel_chains | CmdStan controls. |
| iter_warmup | CmdStan controls. |
| iter_sampling | CmdStan controls. |
| adapt_delta | CmdStan sampler controls. |
| max_treedepth | CmdStan sampler controls. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

# dynamic_posterior_predictive_check

*Posterior predictive checks for dynamic state models*

---

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
dynamic_posterior_predictive_check(object, draws = 200L, seed = 1L)
```

## Arguments

| | |
|---|---|
| object | Dynamic IRTree fit using the Stan engine. |
| draws | Maximum posterior predictive draws to summarize. |
| seed | Seed used when subsampling posterior draws. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## dynamic_transition_design

*Build an explicit dynamic-transition design matrix*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
dynamic_transition_design(data, spec = dynamic_irtree_spec(), formula = NULL)
```

### Arguments

| | |
|---|---|
| data | Prepared transition data. |
| spec | Dynamic IRTree specification. |
| formula | Optional explicit right-hand-side formula. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## effective_sampling_frequency

*Estimate effective sampling frequency from timestamps*

### Description

Estimate effective sampling frequency from timestamps

### Usage

```
effective_sampling_frequency(
  data,
  time = "timestamp_ms",
  unit = c("ms", "s", "us"),
  by = NULL
)
```

### Arguments

| | |
|---|---|
| data | Sample data. |
| time | Timestamp column. |
| unit | Timestamp unit. |
| by | Optional grouping columns. |

## Value

A logical value or vector indicating effective sampling frequency from timestamps.

---

`encode_response_combinations` *Encode multiple-response item response combinations*

---

## Description

Encode multiple-response item response combinations

## Usage

```
encode_response_combinations(
  data,
  person = "participant_id",
  item = "item_id",
  option = "option_id",
  selected = "selected",
  sort_options = TRUE,
  empty_code = "<none>"
)
```

## Arguments

| | |
|---|---|
| `data` | Long person-item-option table. |
| `person`, `item`, `option`, `selected` | Column names. |
| `sort_options` | Sort selected option labels before combining. |
| `empty_code` | Code for no selected options. |

## Value

A tabular R object containing encode multiple-response item response combinations; rows represent analysis units and columns contain the returned quantities.

## engine_adapter_status *Report an adapter's availability and contract*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
engine_adapter_status(engine)
```

### Arguments

| | |
|---|---|
| engine | Engine name. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## equate_irt_scales *Equate IRT scales using anchor item parameters*

### Description

Implements mean-sigma, mean-mean, Stocking-Lord, and Haebara linking for dichotomous 2PL-style item parameters. New-form parameters are transformed onto the reference scale using theta_ref = A * theta_new + B.

### Usage

```
equate_irt_scales(
  reference,
  new,
  method = c("stocking-lord", "haebara", "mean-sigma", "mean-mean"),
  theta_grid = seq(-4, 4, length.out = 81)
)
```

### Arguments

| | |
|---|---|
| reference | Reference-scale parameters or data. |
| new | New-scale parameters or data. |
| method | Method used for estimation, linking, or comparison. |
| theta_grid | Grid of latent-trait values used for evaluation. |

### Value

An object of class "eye_irt_equating", stored as a named list, with components "A", "B", "method", "transformed", "reference", "new", "equation". It contains equate IRT scales using anchor item parameters and associated metadata or diagnostics needed to interpret the result.

---

`estimate_calibration_error` *Estimate empirical calibration/validation error*

---

### Description

Estimate empirical calibration/validation error

### Usage

```
estimate_calibration_error(
  data,
  gaze_x = "gaze_x",
  gaze_y = "gaze_y",
  target_x = "target_x",
  target_y = "target_y",
  by = NULL
)
```

### Arguments

| | |
|---|---|
| `data` | Validation-target data. |
| `gaze_x, gaze_y` | Recorded gaze-coordinate columns. |
| `target_x, target_y` | Known target-coordinate columns. |
| `by` | Optional grouping columns such as participant/session. |

### Value

A tabular R object containing empirical calibration/validation error; rows represent analysis units and columns contain the returned quantities.

`estimate_visual_exposure_probability` *Estimate visual exposure probability*

## Description

Estimate visual exposure probability

## Usage

```
estimate_visual_exposure_probability(
  data,
  exposed = "reached",
  predictors,
  family = stats::binomial()
)
```

## Arguments

| | |
|---|---|
| `data` | Input data frame or compatible tabular object. |
| `exposed` | Value supplied to ‘exposed‘; see Details for its model-specific role. |
| `predictors` | Predictor variables used by the model. |
| `family` | Statistical family used by the channel or model. |

## Value

An object of class "eye_visual_exposure_model", stored as a named list, with components "model", "fitted_probability", "exposed", "predictors". It contains visual exposure probability and associated metadata or diagnostics needed to interpret the result.

`evaluate_validation_acceptance` *Evaluate a validation acceptance rule*

## Description

Evaluate a validation acceptance rule

## Usage

```
evaluate_validation_acceptance(value, rule)
```

## Arguments

| | |
|---|---|
| `value` | Observed metric value to evaluate. |
| `rule` | Validation acceptance rule to apply. |

## Value

A logical value or vector indicating a validation acceptance rule.

---

`event_detector_multiverse`

*Event-detector multiverse and inference robustness*

---

## Description

Build explicit event-detector specifications, run detector branches independently, compare event catalogues, propagate detector choice through AOI features and an identical statistical model, and report scientific inference stability.

## Usage

```
define_event_detector_spec(
  detector_id, algorithm, velocity_threshold = NULL,
  dispersion_threshold = NULL, minimum_duration_ms = NULL,
  maximum_gap_ms = NULL, merge_rule = "none", sampling_rate = NULL,
  smoothing = NULL, filter = NULL, coordinate_unit = "degrees",
  implementation = NULL, implementation_version = NULL,
  parameters = list(), callback = NULL
)
validate_event_detector_spec(spec)
create_detector_multiverse(
  specs = NULL, base_spec = NULL, parameter_grid = NULL,
  id_template = "%s_%03d", label = "event_detector_multiverse"
)
detect_events_with_spec(x, spec)
import_external_detector_events(events, spec, dataset = NULL)
run_detector_multiverse(x, multiverse, continue_on_error = TRUE)
match_detected_events(
  reference, candidate, event_type = "fixation",
  onset_tolerance_ms = 75, minimum_overlap = 0.10
)
compare_event_catalogues(
  reference, candidate, event_type = "fixation",
  onset_tolerance_ms = 75, minimum_overlap = 0.10
)
estimate_detector_agreement(
  x, event_type = "fixation",
  onset_tolerance_ms = 75, minimum_overlap = 0.10
)
summarise_detector_events(x)
summarise_detector_disagreement(x, event_type = "fixation")
propagate_detector_to_aoi(x, overlap = "error", continue_on_error = TRUE)
```

```
propagate_detector_to_features(x, continue_on_error = TRUE)
run_detector_inference_multiverse(
  x, model_spec, model_callback = NULL, minimum_valid_fraction = NULL
)
assess_detector_inference_stability(
  x, term, substantive_threshold = NULL,
  direction = c("above", "below", "absolute")
)
summarise_detector_robustness(
  x, inference = NULL, term = NULL,
  substantive_threshold = NULL, direction = "above"
)
plot_detector_event_timeline(x, trial_id = NULL, event_type = "fixation", ...)
plot_detector_agreement(x, metric = "mean_event_overlap", ...)
plot_detector_feature_distributions(
  x, feature = "dwell_time_ms", aoi_id = NULL, ...
)
plot_detector_coefficient_stability(x, term, ...)
plot_detector_multiverse(
  x, inference = NULL, term = NULL,
  feature = "dwell_time_ms", aoi_id = NULL
)
report_detector_multiverse(
  x, inference = NULL, term = NULL,
  substantive_threshold = NULL, path = NULL
)
simulate_detector_multiverse_data(
  n_participants = 12L, sampling_rate = 60,
  trial_duration_s = 2, seed = 20260918L
)
```

## Arguments

| | |
|---|---|
| `detector_id` | Unique detector-branch identifier. |
| `algorithm` | One of `"ivt"`, `"idt"`, `"adaptive_velocity"`, `"remodnav"`, `"external"`, or `"vendor"`. |
| `velocity_threshold` | Explicit velocity threshold in the declared coordinate unit per second. |
| `dispersion_threshold` | Explicit dispersion threshold in the declared coordinate unit. |
| `minimum_duration_ms` | Minimum event duration in milliseconds. |
| `maximum_gap_ms` | Maximum gap used by algorithms that support gap handling. |
| `merge_rule` | Recorded merge rule. No undeclared merging is applied. |
| `sampling_rate` | Declared sample rate in Hz. |
| `smoothing`, `filter` | Recorded preprocessing choices. |

`coordinate_unit`
: `"degrees"`, `"pixels"`, or `"normalized"`.

`implementation`, `implementation_version`
: Detector implementation identity and version.

`parameters`
: Named algorithm-specific parameter list.

`callback`
: Explicit external detector or model callback.

`spec`
: An event-detector specification.

`specs`
: A list of explicit detector specifications.

`base_spec`
: Base specification for an explicit parameter grid.

`parameter_grid`
: Named list of parameter values. Values are study-specific alternatives, not universal defaults.

`id_template`
: `sprintf()` template for generated branch identifiers.

`label`
: Multiverse label.

`x`
: An `eye_dataset`, detector multiverse result, or detector inference result as required.

`multiverse`
: An event-detector multiverse created by `create_detector_multiverse()`.

`events`, `reference`, `candidate`
: Canonical or canonicalizable event data frames.

`dataset`
: Optional source `eye_dataset` used to attach provenance to imported events.

`continue_on_error`
: Whether to retain a failed branch and continue other specifications.

`event_type`
: Event type to compare, usually `"fixation"`.

`onset_tolerance_ms`
: Allowed onset difference for near-boundary matches.

`minimum_overlap`
: Minimum temporal intersection-over-union for event matching.

`overlap`
: AOI-overlap rule. The default `"error"` prevents silent ambiguity resolution.

`model_spec`
: Named list declaring the model engine, formula, outcome, and optional AOI/model settings.

`model_callback`
: Specialist model callback returning the documented tidy coefficient contract.

`minimum_valid_fraction`
: Explicit minimum gaze-validity fraction before modelling.

`term`
: Model coefficient term to assess.

`substantive_threshold`
: Optional analyst-declared effect threshold.

`direction`
: How the substantive threshold is interpreted.

`inference`
: Optional detector inference result.

`trial_id`
: Optional trial filter.

`metric`
: Agreement metric to plot.

`feature`
: Propagated feature to plot.

`aoi_id`
: Optional AOI filter.

| | |
|---|---|
| path | Optional Markdown report path. |
| n_participants | Synthetic-example participant count. |
| trial_duration_s | Synthetic trial duration in seconds. |
| seed | Synthetic-data random seed. |
| ... | Additional base-graphics arguments. |

## Details

The multiverse is intended to answer whether event detection materially changes the quantities and scientific inference used by a study. It is not a mechanism for selecting the branch with a preferred p-value.

Zero-event trials remain represented during feature propagation. Trials with no valid gaze remain missing rather than being converted to zero. AOI ambiguity is an error unless an analyst explicitly selects another overlap rule. Model engines are explicit; formula engines use `na.fail`; and non-converged model branches are retained diagnostically but excluded from coefficient-stability summaries.

`algorithm = "remodnav"` calls an installed REMoDNaV executable and never substitutes an internal approximation. Pixel coordinates require an explicit `parameters$px2deg`. The built-in adaptive-velocity implementation is reported as a robust-MAD reference detector and must not be described as a named external algorithm.

## Value

Specification functions return detector specification or multiverse objects. Execution/propagation functions return governed multiverse result objects. `run_detector_inference_multiverse()` returns an inference result containing coefficients, failures, warnings, model metadata, a feature fingerprint, and `input_audit`. The audit has one row per planned detector and records `input_rows`, `aoi_selected_rows`, `quality_excluded_rows`, `outcome_missing_rows`, `model_rows_used`, and branch `status`. Comparison and stability functions return data frames or named summary lists. For `assess_detector_inference_stability()`, `specifications` is the number of planned detector specifications, `term_available_specifications` is the number that returned the requested coefficient, and `model_failure_specifications` is the number with recorded model-stage failures. The convergence-rate denominator is the full declared multiverse, so failed, missing-term, and non-converged branches cannot disappear from robustness accounting. Plotting functions draw base-R diagnostics and invisibly return plotted data. `report_detector_multiverse()` returns Markdown text. `simulate_detector_multiverse_data()` returns a synthetic `eye_dataset`.

## See Also

`detect_fixations_ivt`, `detect_fixations_idt`, `assign_aois`, `preprocessing_multiverse`

## Examples

```
d <- simulate_detector_multiverse_data(n_participants = 4, seed = 1)

s1 <- define_event_detector_spec(
  "ivt25", "ivt",
```

```
  velocity_threshold = 25,
  minimum_duration_ms = 60,
  maximum_gap_ms = 75,
  sampling_rate = 60
)
s2 <- define_event_detector_spec(
  "ivt35", "ivt",
  velocity_threshold = 35,
  minimum_duration_ms = 60,
  maximum_gap_ms = 75,
  sampling_rate = 60
)

mv <- create_detector_multiverse(list(s1, s2))
out <- run_detector_multiverse(d, mv)
out <- propagate_detector_to_aoi(out)
out <- propagate_detector_to_features(out)
summarise_detector_events(out)
```

---

`event_marker_qc` *Event-marker plausibility audit*

---

## Description

Evaluates whether independent channel offsets corroborate a nominal event. This is annotation/event plausibility QC, not clock synchronization, and it never modifies timestamps.

## Usage

```
event_marker_qc(offsets, tolerance, min_corroborating = 2L)
```

## Arguments

| | |
|---|---|
| `offsets` | Numeric corroborating-channel offsets in seconds. |
| `tolerance` | Allowed absolute offset in seconds. |
| `min_corroborating` | Minimum corroborating channels for ‘confirmed‘. |

## Value

A list containing status, consensus offset, uncertainty, and counts.

## event_roundtrip_audit *Audit event survival across an interchange round trip*

### Description

Audit event survival across an interchange round trip

### Usage

```
event_roundtrip_audit(source_events, roundtrip_events, hed_column = NULL, ...)
```

### Arguments

| | |
|---|---|
| source_events, roundtrip_events | Event tables. |
| hed_column | Optional HED annotation column to compare structurally. |
| ... | Arguments forwarded to 'event_semantics_audit()'. |

### Value

An object of class "eye_event_roundtrip_audit", stored as a named list, with components "status", "event_semantics", "hed". It contains event survival across an interchange round trip and associated metadata or diagnostics needed to interpret the result.

## event_semantics_audit *Audit event semantic preservation*

### Description

Audit event semantic preservation

### Usage

```
event_semantics_audit(
  source_events,
  roundtrip_events,
  label = "event",
  time = "timestamp",
  key = NULL,
  tolerance = 1e-06
)
```

### Arguments

| | |
|---|---|
| source_events, roundtrip_events | Event tables. |
| label | Event-label field. |
| time | Event-time field; set 'NULL' to compare labels only. |
| key | Optional event identity key. |
| tolerance | Timestamp tolerance. |

### Value

An object of class "eye_event_semantics", stored as a named list, with components "status", "source_n", "roundtrip_n", "matched_n", "exact_label_fraction", "max_time_error". It contains event semantic preservation and associated metadata or diagnostics needed to interpret the result.

---

## expand_eyeprocess_stress_evidence_plan *Expand a stress evidence plan into one-factor-at-a-time scenarios*

---

### Description

Expand a stress evidence plan into one-factor-at-a-time scenarios

### Usage

```
expand_eyeprocess_stress_evidence_plan(plan)
```

### Arguments

| | |
|---|---|
| plan | Validation or stress-evidence plan object. |

### Value

A tabular R object containing expand a stress evidence plan into one-factor-at-a-time scenarios; rows represent analysis units and columns contain the returned quantities.

## expand_eyeprocess_validation_plan

*Expand a validation-evidence plan to a scenario table*

### Description

Expand a validation-evidence plan to a scenario table

### Usage

```
expand_eyeprocess_validation_plan(x)
```

### Arguments

| | |
|---|---|
| x | Object to validate, summarize, verify, or otherwise process. |

### Value

An R object containing expand a validation-evidence plan to a scenario table. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## expand_process_validation_design

*Expand a process-validation design into explicit conditions*

### Description

Expand a process-validation design into explicit conditions

### Usage

```
expand_process_validation_design(x, max_conditions = 250000L)
```

### Arguments

| | |
|---|---|
| x | Validation design. |
| max_conditions | Optional hard cap for accidental combinatorial explosion. |

### Value

A tabular R object containing expand a process-validation design into explicit conditions; rows represent analysis units and columns contain the returned quantities.

## expected_process_information

*Expected process-aware item utility under a theta distribution*

### Description

Expected process-aware item utility under a theta distribution

### Usage

```
expected_process_information(info, theta_weights = NULL)
```

### Arguments

| | |
|---|---|
| info | Value supplied to 'info'; see Details for its model-specific role. |
| theta_weights | Weights over the theta distribution. |

### Value

An R object containing expected process-aware item utility under a theta distribution. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## explain_latent_interaction

*Explain local person-item latent-space interactions*

### Description

Returns the closest person-item pairs in the fitted residual latent space. Closeness is descriptive residual structure, not a causal explanation.

### Usage

```
explain_latent_interaction(object, person = NULL, item = NULL, top = 10L)
```

### Arguments

| | |
|---|---|
| object | Fitted 'eye_latent_space_irt' object. |
| person | Optional person row/index/name to restrict. |
| item | Optional item row/index/name to restrict. |
| top | Number of closest pairs to return. |

## Value

An R object containing explain local person-item latent-space interactions. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

`export_eye_bids` *Export Eye-Tracking-BIDS physiological recordings*

---

## Description

Writes BIDS-compatible '_physio.tsv.gz' and JSON pairs with 'PhysioType = "eyetrack"'. Continuous files contain no header; 'Columns' is stored in the sidecar. Each recorded eye is written separately.

## Usage

```
export_eye_bids(
  x,
  path,
  task = "task",
  dataset_name = "eyeprocess eye-tracking dataset",
  overwrite = FALSE,
  screen_distance_m = NULL,
  screen_size_m = NULL
)
```

## Arguments

| | |
|---|---|
| `x` | An 'eye_dataset'. |
| `path` | BIDS dataset root. |
| `task` | Task label. |
| `dataset_name` | Dataset name for 'dataset_description.json'. |
| `overwrite` | Whether to replace existing files. |
| `screen_distance_m` | Optional screen distance in metres. |
| `screen_size_m` | Optional two-element screen size in metres. |

## Value

A manifest of written files.

## export_eye_pipeline *Export a pipeline manifest and optional run status*

### Description

Export a pipeline manifest and optional run status

### Usage

```
export_eye_pipeline(x, path)
```

### Arguments

| | |
|---|---|
| x | Pipeline or run. |
| path | CSV path. |

### Value

An R object containing a pipeline manifest and optional run status. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## export_prov_json *Export lightweight PROV-oriented JSON*

### Description

This is a compact interoperability representation inspired by W3C PROV. It is not asserted to be a complete PROV-O serialization; consumers requiring full conformance should validate/transform it externally.

### Usage

```
export_prov_json(x, path)
```

### Arguments

| | |
|---|---|
| x | Provenance graph. |
| path | Output JSON path. |

### Value

An R object containing lightweight PROV-oriented JSON. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

`export_ro_crate_metadata` *Export minimal RO-Crate 1.3 metadata*

## Description

Export minimal RO-Crate 1.3 metadata

## Usage

```
export_ro_crate_metadata(
  path = "ro-crate-metadata.json",
  name = "eyeprocess analysis",
  description = "Reproducible eyeprocess analysis crate",
  files = NULL,
  creator = NULL,
  license = NULL,
  doi = NULL
)
```

## Arguments

| | |
|---|---|
| `path` | Output ‘ro-crate-metadata.json‘ path. |
| `name` | Crate/dataset name. |
| `description` | Description. |
| `files` | Optional files to include as File entities. |
| `creator` | Optional creator name. |
| `license` | Optional license URL or identifier. |
| `doi` | Optional DOI for the software/data product. |

## Value

An R object containing minimal RO-Crate 1.3 metadata. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

`export_validation_bundle` *Export a validation evidence bundle*

## Description

Export a validation evidence bundle

### Usage

```
export_validation_bundle(x, directory, overwrite = FALSE, include_rds = TRUE)
```

### Arguments

| | |
|---|---|
| x | Validation bundle. |
| directory | Output directory. |
| overwrite | Allow writing into a non-empty target directory. |
| include_rds | Save full R objects as RDS. |

### Value

An object of class "eye_validation_export", stored as a named list, with components "directory", "files", "manifest". It contains a validation evidence bundle and associated metadata or diagnostics needed to interpret the result.

---

## external_eye_adapters *Convert common external eye-tracking objects*

---

### Description

Convert common external eye-tracking objects

### Usage

```
as_eyeprocess_eyetools(x, mapping = NULL, ...)

as_eyeprocess_eyetrackingr(x, mapping = NULL, ...)

as_eyeprocess_gazer(x, mapping = NULL, ...)

as_eyeprocess_eyeris(x, mapping = NULL, ...)

as_eyeprocess_pupillometryr(x, mapping = NULL, ...)
```

### Arguments

| | |
|---|---|
| x | External object coercible to a data frame. |
| mapping | Optional ‘eye_mapping‘. |
| ... | Passed to ‘read_eye_generic()‘. |

### Value

An ‘eye_dataset‘.

## external_model_engines *List external engine adapters*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
external_model_engines()
```

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## external_validate_irt *External validation on a completely held-out dataset*

### Description

External validation on a completely held-out dataset

### Usage

```
external_validate_irt(
  train_data,
  external_data,
  fitter,
  predictor,
  scorer,
  label = "external"
)
```

### Arguments

| | |
|---|---|
| train_data | Value supplied to ‘train_data‘; see Details for its model-specific role. |
| external_data | Value supplied to ‘external_data‘; see Details for its model-specific role. |
| fitter | Model-fitting function. |
| predictor | Prediction function. |
| scorer | Function that scores predictions. |
| label | Value supplied to ‘label‘; see Details for its model-specific role. |

### Value

A logical value or vector indicating external validation on a completely held-out dataset.

---

`extract_diffusion_parameters`
*Extract diffusion parameter summaries*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
extract_diffusion_parameters(object, variables = c("beta_drift", "beta_boundary",
  "beta_nondecision", "beta_starting", "person_drift", "item_difficulty", "boundary",
   "nondecision", "starting", "contaminant_probability"))
```

### Arguments

| | |
|---|---|
| `object` | Diffusion fit. |
| `variables` | Optional variable prefixes. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`extract_functional_pupil_parameters`
*Extract functional pupil parameters for validation*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
extract_functional_pupil_parameters(x, pattern = NULL, confidence = 0.95)
```

### Arguments

| | |
|---|---|
| `x` | Functional pupil fit. |
| `pattern` | Optional parameter regex. |
| `confidence` | Credible/confidence interval level. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`extract_parameter_truth` *Extract canonical parameter truth from simulated data*

---

## Description

Extract canonical parameter truth from simulated data

## Usage

```
extract_parameter_truth(simulation)
```

## Arguments

| | |
|---|---|
| `simulation` | Simulation object containing ‘truth‘, ‘parameters‘, or a ‘truth‘ attribute. |

## Value

Data frame with ‘parameter‘ and ‘truth‘.

---

`extract_process_windows` *Extract standardized sliding-window process features*

---

## Description

Extract standardized sliding-window process features

## Usage

```
extract_process_windows(
  data,
  person = "person_id",
  trial = "trial_id",
  time = "time_ms",
  spec = process_window_spec(),
  align_time = NULL,
  pupil = "pupil_bc",
  pupil_tonic = "pupil_tonic",
  pupil_phasic = "pupil_phasic",
  gaze_x = "x",
  gaze_y = "y",
```

```
  aoi = "aoi",
  valid_gaze = "valid_gaze_prop",
  valid_pupil = "valid_pupil_prop",
  blink = "blink",
  trackloss = "trackloss"
)
```

## Arguments

| | |
|---|---|
| `data` | Sample-level eye-tracking/pupil data. |
| `person`, `trial`, `time` | Identifier/time columns. |
| `spec` | Window specification. |
| `align_time` | Optional column containing the alignment event time in the same units as ‘time‘. Required for response/custom alignment unless ‘time‘ is already relative to the desired origin. |
| `pupil` | Optional pupil signal column. |
| `pupil_tonic`, `pupil_phasic` | Optional decomposed pupil columns. |
| `gaze_x`, `gaze_y` | Optional gaze coordinates. |
| `aoi` | Optional AOI-state column. |
| `valid_gaze`, `valid_pupil` | Optional validity columns/indicators. |
| `blink`, `trackloss` | Optional blink/trackloss indicators. |

## Value

An ‘eye_process_windows‘ object.

---

# `eyeprocess-adapters` *Mappings and adapter registry*

---

## Description

Map heterogeneous exports, register import adapters, detect formats, and combine canonical datasets.

## Details

These functions operate on the canonical relational representation used by **eyeprocess**. Native fields and timestamps are retained whenever possible, and transformations should be recorded in the provenance table. Optional modelling engines are used only when their packages are installed.

## Value

The returned value depends on the function. Import and transformation functions generally return an `eye_dataset`; audit and feature functions return data frames or enriched datasets; plotting functions return their plotted data invisibly; modelling functions return engine-specific or `eyeprocess_model` objects.

## See Also

`new_eye_dataset()`, `read_eye_export()`, `validate_eye_dataset()`, `provenance_manifest()`

---

`eyeprocess-class` *Create and manage eyeprocess datasets*

---

## Description

Construct, inspect, validate, modify, and summarize canonical eye-tracking datasets.

## Details

These functions operate on the canonical relational representation used by **eyeprocess**. Native fields and timestamps are retained whenever possible, and transformations should be recorded in the provenance table. Optional modelling engines are used only when their packages are installed.

## Value

The returned value depends on the function. Import and transformation functions generally return an `eye_dataset`; audit and feature functions return data frames or enriched datasets; plotting functions return their plotted data invisibly; modelling functions return engine-specific or `eyeprocess_model` objects.

## See Also

`new_eye_dataset()`, `read_eye_export()`, `validate_eye_dataset()`, `provenance_manifest()`

---

`eyeprocess-coordinates-time`
*Coordinate and timebase management*

---

## Description

Register and convert coordinate spaces, normalize native clocks, estimate sampling rates, and synchronize streams.

## Details

These functions operate on the canonical relational representation used by **eyeprocess**. Native fields and timestamps are retained whenever possible, and transformations should be recorded in the provenance table. Optional modelling engines are used only when their packages are installed.

## Value

The returned value depends on the function. Import and transformation functions generally return an `eye_dataset`; audit and feature functions return data frames or enriched datasets; plotting functions return their plotted data invisibly; modelling functions return engine-specific or `eyeprocess_model` objects.

## See Also

`new_eye_dataset()`, `read_eye_export()`, `validate_eye_dataset()`, `provenance_manifest()`

---

`eyeprocess-experimental`

*Experimental psychometric process models*

---

## Description

Experimental interfaces for multimodal IRT, functional pupil summaries, strategy mixtures, diffusion summaries, and missing-process sensitivity.

## Details

These functions operate on the canonical relational representation used by **eyeprocess**. Native fields and timestamps are retained whenever possible, and transformations should be recorded in the provenance table. Optional modelling engines are used only when their packages are installed.

## Value

The returned value depends on the function. Import and transformation functions generally return an `eye_dataset`; audit and feature functions return data frames or enriched datasets; plotting functions return their plotted data invisibly; modelling functions return engine-specific or `eyeprocess_model` objects.

## See Also

`new_eye_dataset()`, `read_eye_export()`, `validate_eye_dataset()`, `provenance_manifest()`

## eyeprocess-export-report

*Export, reporting, and package bridges*

### Description

Persist canonical datasets, produce provenance-aware reports, and bridge gp3tools or gpbiometrics objects.

### Details

These functions operate on the canonical relational representation used by **eyeprocess**. Native fields and timestamps are retained whenever possible, and transformations should be recorded in the provenance table. Optional modelling engines are used only when their packages are installed.

### Value

The returned value depends on the function. Import and transformation functions generally return an `eye_dataset`; audit and feature functions return data frames or enriched datasets; plotting functions return their plotted data invisibly; modelling functions return engine-specific or `eyeprocess_model` objects.

### See Also

`new_eye_dataset()`, `read_eye_export()`, `validate_eye_dataset()`, `provenance_manifest()`

## eyeprocess-features

*Derive gaze, pupil, response-time, and biometric features*

### Description

Generate declared person-item-trial features, scanpaths, transitions, entropy measures, and feature dictionaries.

### Details

These functions operate on the canonical relational representation used by **eyeprocess**. Native fields and timestamps are retained whenever possible, and transformations should be recorded in the provenance table. Optional modelling engines are used only when their packages are installed.

### Value

The returned value depends on the function. Import and transformation functions generally return an `eye_dataset`; audit and feature functions return data frames or enriched datasets; plotting functions return their plotted data invisibly; modelling functions return engine-specific or `eyeprocess_model` objects.

## See Also

new_eye_dataset(), read_eye_export(), validate_eye_dataset(), provenance_manifest()

---

# eyeprocess-format-validation

*Validate real eye-tracking exports and compatibility corpora*

---

## Description

Build reproducible evidence for source-format compatibility. The validation framework distinguishes declared support, synthetic-fixture testing, and empirical validation with real exports. It audits source structure, adapter detection, canonical import, field coverage, native timestamp preservation, coordinate semantics, provenance, and canonical round-trip fidelity.

## Details

validate_eye_source() validates one file or folder. Use validation_manifest() and validate_eye_corpus() to evaluate a versioned collection of exports from multiple vendors, devices, and software versions. init_validation_corpus() is idempotent and preserves an existing manifest and case files unless overwrite = TRUE is supplied. A source can be retained temporarily in the result for creation of an anonymized validation bundle; raw vendor data are never included in bundles by default.

anonymize_eye_dataset() replaces participant, recording, and session identifiers, removes raw tables and source paths by default, and records the operation in provenance. Automated anonymization cannot guarantee removal of all study-specific free text, so exported bundles still require human review.

## Value

Depending on the function, an eye_format_validation_spec, source-file manifest, schema-coverage table, source-preservation audit, canonical round-trip result, eye_format_validation, eye_corpus_validation, anonymized eye_dataset, report path, or validation-bundle path.

## See Also

read_eye_export(), validate_eye_dataset(), supported_eye_formats(), write_eye_dataset(), provenance_manifest()

## eyeprocess-import-gazepoint

*Import Gazepoint and Gazepoint Biometrics exports*

### Description

Detect, profile, import, pair, validate, and reconstruct Gazepoint Analysis and Biometrics exports, including Gazepoint Analysis 7.x `*_all_gaze.csv`, `*_fixations.csv`, and multi-section `Data_Summary_export_*.csv` files.

### Details

These functions operate on the canonical relational representation used by **eyeprocess**. For Gazepoint Analysis 7.x exports, `TIMETICK(f=...)` is retained as the native clock and normalized to seconds from recording start, while media-relative `TIME(...)` values are retained as source fields. Fixation identifiers that restart for each media item are namespaced by stimulus. Multi-section Data Summary reports are parsed into AOI definitions, user-AOI features, raw report tables, and vendor metadata. Native fields and transformations remain represented in provenance and vendor-specific metadata.

### Value

The returned value depends on the function. Import and transformation functions generally return an `eye_dataset`; audit and feature functions return data frames or enriched datasets; plotting functions return their plotted data invisibly; modelling functions return engine-specific or `eyeprocess_model` objects.

### See Also

`new_eye_dataset()`, `read_eye_export()`, `validate_eye_dataset()`, `provenance_manifest()`

## eyeprocess-import-generic

*Import generic delimited eye-tracking data*

### Description

Transform explicitly mapped data frames, CSV files, and TSV files into canonical eyeprocess datasets.

### Details

These functions operate on the canonical relational representation used by **eyeprocess**. Native fields and timestamps are retained whenever possible, and transformations should be recorded in the provenance table. Optional modelling engines are used only when their packages are installed.

## Value

The returned value depends on the function. Import and transformation functions generally return an `eye_dataset`; audit and feature functions return data frames or enriched datasets; plotting functions return their plotted data invisibly; modelling functions return engine-specific or `eyeprocess_model` objects.

## See Also

`new_eye_dataset()`, `read_eye_export()`, `validate_eye_dataset()`, `provenance_manifest()`

---

`eyeprocess-import-vendors`
*Import Tobii, Pupil Labs, EyeLink, and SMI exports*

---

## Description

Dedicated import adapters for common and legacy eye-tracking export ecosystems.

## Details

These functions operate on the canonical relational representation used by **eyeprocess**. Native fields and timestamps are retained whenever possible, and transformations should be recorded in the provenance table. Optional modelling engines are used only when their packages are installed.

## Value

The returned value depends on the function. Import and transformation functions generally return an `eye_dataset`; audit and feature functions return data frames or enriched datasets; plotting functions return their plotted data invisibly; modelling functions return engine-specific or `eyeprocess_model` objects.

## See Also

`new_eye_dataset()`, `read_eye_export()`, `validate_eye_dataset()`, `provenance_manifest()`

---

`eyeprocess-models` *Psychometric and process-data models*

---

## Description

Prepare response matrices and fit optional IRT, explanatory, response-time, DIF, shared-process, and dynamic models.

## Details

These functions operate on the canonical relational representation used by **eyeprocess**. Native fields and timestamps are retained whenever possible, and transformations should be recorded in the provenance table. Optional modelling engines are used only when their packages are installed.

## Value

The returned value depends on the function. Import and transformation functions generally return an `eye_dataset`; audit and feature functions return data frames or enriched datasets; plotting functions return their plotted data invisibly; modelling functions return engine-specific or `eyeprocess_model` objects.

## See Also

`new_eye_dataset()`, `read_eye_export()`, `validate_eye_dataset()`, `provenance_manifest()`

---

`eyeprocess-plots` *Visualize eye-tracking and multimodal process data*

---

## Description

Base-R plots for traces, fixations, scanpaths, heatmaps, AOIs, pupil, biometrics, quality, timing, and models.

## Details

These functions operate on the canonical relational representation used by **eyeprocess**. Native fields and timestamps are retained whenever possible, and transformations should be recorded in the provenance table. Optional modelling engines are used only when their packages are installed.

## Value

The returned value depends on the function. Import and transformation functions generally return an `eye_dataset`; audit and feature functions return data frames or enriched datasets; plotting functions return their plotted data invisibly; modelling functions return engine-specific or `eyeprocess_model` objects.

## See Also

`new_eye_dataset()`, `read_eye_export()`, `validate_eye_dataset()`, `provenance_manifest()`

`eyeprocess-preprocessing` *Gaze and pupil preprocessing*

## Description

Declare preprocessing, filter signals, interpolate pupil gaps, detect ocular events, and retain a transformation record.

## Details

These functions operate on the canonical relational representation used by **eyeprocess**. Native fields and timestamps are retained whenever possible, and transformations should be recorded in the provenance table. Optional modelling engines are used only when their packages are installed.

## Value

The returned value depends on the function. Import and transformation functions generally return an `eye_dataset`; audit and feature functions return data frames or enriched datasets; plotting functions return their plotted data invisibly; modelling functions return engine-specific or `eyeprocess_model` objects.

## See Also

`new_eye_dataset()`, `read_eye_export()`, `validate_eye_dataset()`, `provenance_manifest()`

`eyeprocess-quality` *Quality control, sensitivity, and governance*

## Description

Audit data quality, flag process leakage, assess readiness, and compare preprocessing or AOI specifications.

## Details

These functions operate on the canonical relational representation used by **eyeprocess**. Native fields and timestamps are retained whenever possible, and transformations should be recorded in the provenance table. Optional modelling engines are used only when their packages are installed.

## Value

The returned value depends on the function. Import and transformation functions generally return an `eye_dataset`; audit and feature functions return data frames or enriched datasets; plotting functions return their plotted data invisibly; modelling functions return engine-specific or `eyeprocess_model` objects.

## See Also

new_eye_dataset(), read_eye_export(), validate_eye_dataset(), provenance_manifest()

---

eyeprocess-schema *Canonical schemas and coordinate spaces*

---

## Description

Define, inspect, standardize, and validate the relational tables used by eyeprocess.

## Details

These functions operate on the canonical relational representation used by **eyeprocess**. Native fields and timestamps are retained whenever possible, and transformations should be recorded in the provenance table. Optional modelling engines are used only when their packages are installed.

## Value

The returned value depends on the function. Import and transformation functions generally return an eye_dataset; audit and feature functions return data frames or enriched datasets; plotting functions return their plotted data invisibly; modelling functions return engine-specific or eyeprocess_model objects.

## See Also

new_eye_dataset(), read_eye_export(), validate_eye_dataset(), provenance_manifest()

---

eyeprocess-simulation *Simulate and validate eyeprocess workflows*

---

## Description

Simulate canonical multimodal datasets and conduct parameter-recovery or power experiments.

## Details

These functions operate on the canonical relational representation used by **eyeprocess**. Native fields and timestamps are retained whenever possible, and transformations should be recorded in the provenance table. Optional modelling engines are used only when their packages are installed.

## Value

The returned value depends on the function. Import and transformation functions generally return an eye_dataset; audit and feature functions return data frames or enriched datasets; plotting functions return their plotted data invisibly; modelling functions return engine-specific or eyeprocess_model objects.

### See Also

new_eye_dataset(), read_eye_export(), validate_eye_dataset(), provenance_manifest()

---

## eyeprocess-trials-aoi *Trials, responses, stimuli, and areas of interest*

---

### Description

Reconstruct trial intervals, attach responses, register static or dynamic AOIs, and derive AOI visits.

### Details

These functions operate on the canonical relational representation used by **eyeprocess**. Native fields and timestamps are retained whenever possible, and transformations should be recorded in the provenance table. Optional modelling engines are used only when their packages are installed.

### Value

The returned value depends on the function. Import and transformation functions generally return an eye_dataset; audit and feature functions return data frames or enriched datasets; plotting functions return their plotted data invisibly; modelling functions return engine-specific or eyeprocess_model objects.

### See Also

new_eye_dataset(), read_eye_export(), validate_eye_dataset(), provenance_manifest()

---

## eyeprocess_api_version

*Return the public eyeprocess API version*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
eyeprocess_api_version()
```

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## eyeprocess_benchmark_study

*Locate the bundled public benchmark study*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
eyeprocess_benchmark_study(path = NULL)
```

### Arguments

| | |
|---|---|
| path | Optional alternative benchmark root. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## eyeprocess_cdm_attribute_profiles

*Enumerate latent attribute profiles*

### Description

Enumerate latent attribute profiles

### Usage

```
eyeprocess_cdm_attribute_profiles(
  n_attributes,
  attribute_names = paste0("A", seq_len(n_attributes))
)
```

### Arguments

| | |
|---|---|
| n_attributes | Number of cognitive-diagnosis attributes. |
| attribute_names | Optional names for the cognitive-diagnosis attributes. |

### Value

An R object containing enumerate latent attribute profiles. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

`eyeprocess_cdm_classification_uncertainty` *Summarise CDM classification uncertainty from profile probabilities*

## Description

Summarise CDM classification uncertainty from profile probabilities

## Usage

```
eyeprocess_cdm_classification_uncertainty(profile_probabilities)
```

## Arguments

| | |
|---|---|
| `profile_probabilities` | Posterior attribute-profile probabilities. |

## Value

A data frame containing cDM classification uncertainty from profile probabilities. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

`eyeprocess_cdm_dina_ideal_response` *Compute deterministic DINA ideal responses from a Q-matrix*

## Description

Compute deterministic DINA ideal responses from a Q-matrix

## Usage

```
eyeprocess_cdm_dina_ideal_response(Q, profiles)
```

## Arguments

| | |
|---|---|
| `Q` | Binary item-by-attribute Q-matrix. |
| `profiles` | Attribute mastery profiles, with rows representing profiles. |

## Value

A vector or matrix containing deterministic DINA ideal responses from a Q-matrix, with shape determined by the supplied analysis units.

## eyeprocess_cdm_dina_probability

*DINA response probabilities from slip and guess parameters*

### Description

DINA response probabilities from slip and guess parameters

### Usage

```
eyeprocess_cdm_dina_probability(ideal_response, slip = 0.1, guess = 0.2)
```

### Arguments

| | |
|---|---|
| ideal_response | Ideal-response indicator or matrix implied by the cognitive-diagnosis model. |
| slip | DINA slip parameter or vector of slip parameters. |
| guess | DINA guessing parameter or vector of guessing parameters. |

### Value

An object of class "matrix", stored as an R object, containing dINA response probabilities from slip and guess parameters and associated metadata needed to interpret the result.

## eyeprocess_cdm_qmatrix_audit

*Audit a cognitive-diagnosis Q-matrix*

### Description

Audit a cognitive-diagnosis Q-matrix

### Usage

```
eyeprocess_cdm_qmatrix_audit(
  Q,
  item_ids = rownames(Q),
  attribute_names = colnames(Q)
)
```

### Arguments

| | |
|---|---|
| Q | Binary item-by-attribute Q-matrix. |
| item_ids | Optional item identifiers. |
| attribute_names | Optional names for the cognitive-diagnosis attributes. |

### Value

An object of class "eye_cdm_qmatrix_audit", stored as a named list, with components "item", "attribute", "duplicate_rows", "complete_identity_block". It contains a cognitive-diagnosis Q-matrix and associated metadata or diagnostics needed to interpret the result.

---

## eyeprocess_deprecation

*Declare a deprecation in a structured form*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
eyeprocess_deprecation(old, replacement, since, remove_after, reason = "")
```

### Arguments

| | |
|---|---|
| old | Deprecated symbol. |
| replacement | Replacement symbol. |
| since | Version where deprecation started. |
| remove_after | Earliest removal version. |
| reason | Reason. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

## eyeprocess_irt_2pl_probability

*2PL item-response probability*

---

### Description

2PL item-response probability

### Usage

```
eyeprocess_irt_2pl_probability(theta, a = 1, b = 0, D = 1)
```

### Arguments

| | |
|---|---|
| theta | Latent-trait value or vector of latent-trait values. |
| a | Item discrimination parameter or vector. |
| b | Item difficulty or location parameter or vector. |
| D | Logistic scaling constant. |

### Value

A numeric value or vector containing 2PL item-response probability.

---

## eyeprocess_irt_3pl_probability
*3PL item-response probability*

---

### Description

3PL item-response probability

### Usage

```
eyeprocess_irt_3pl_probability(theta, a = 1, b = 0, c = 0.2, D = 1)
```

### Arguments

| | |
|---|---|
| theta | Latent-trait value or vector of latent-trait values. |
| a | Item discrimination parameter or vector. |
| b | Item difficulty or location parameter or vector. |
| c | Lower-asymptote parameter or vector. |
| D | Logistic scaling constant. |

### Value

A numeric value or vector containing 3PL item-response probability.

## eyeprocess_irt_4pl_probability *4PL item-response probability*

### Description

4PL item-response probability

### Usage

```
eyeprocess_irt_4pl_probability(theta, a = 1, b = 0, c = 0, d = 1, D = 1)
```

### Arguments

| | |
|---|---|
| theta | Latent-trait value or vector of latent-trait values. |
| a | Item discrimination parameter or vector. |
| b | Item difficulty or location parameter or vector. |
| c | Lower-asymptote parameter or vector. |
| d | Upper-asymptote parameter or vector. |
| D | Logistic scaling constant. |

### Value

A numeric value or vector containing 4PL item-response probability.

## eyeprocess_irt_adaptive_trace *Create an auditable adaptive-testing trace*

### Description

Create an auditable adaptive-testing trace

### Usage

```
eyeprocess_irt_adaptive_trace(
  item_id,
  theta_before,
  theta_after,
  se_after,
  information,
  response = NA_real_
)
```

### Arguments

| | |
|---|---|
| item_id | Item identifier or vector of item identifiers. |
| theta_before | Latent-trait estimate before item administration. |
| theta_after | Latent-trait estimate after item administration. |
| se_after | Conditional standard error after item administration. |
| information | Item or test information value or vector. |
| response | Observed item response or response variable. |

### Value

An object of class "eye_irt_adaptive_trace", "data.frame", stored as a data frame, containing an auditable adaptive-testing trace and associated metadata needed to interpret the result.

---

## eyeprocess_irt_anchor_audit

*Audit candidate anchor items using supplied DIF evidence*

---

### Description

Audit candidate anchor items using supplied DIF evidence

### Usage

```
eyeprocess_irt_anchor_audit(
  items,
  dif = NULL,
  max_abs_effect = 0.1,
  min_information = NULL
)
```

### Arguments

| | |
|---|---|
| items | Item-parameter data frame or item collection. |
| dif | Differential-item-functioning evidence or summary. |
| max_abs_effect | Maximum permitted absolute effect for an anchor candidate. |
| min_information | Minimum required item information. |

### Value

A data frame containing candidate anchor items using supplied DIF evidence. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## eyeprocess_irt_anchor_purification

*Iteratively remove anchors exceeding a supplied effect threshold*

### Description

Iteratively remove anchors exceeding a supplied effect threshold

### Usage

```
eyeprocess_irt_anchor_purification(
  items,
  effect_fun,
  initial = items$item_id,
  threshold = 0.1,
  max_iter = 10L
)
```

### Arguments

| | |
|---|---|
| items | Item-parameter data frame or item collection. |
| effect_fun | Function that computes the anchor-screening effect. |
| initial | Initial value, state, or anchor set. |
| threshold | Decision or diagnostic threshold. |
| max_iter | Maximum number of iterations. |

### Value

An object of class "eye_irt_anchor_purification", stored as a named list, with components "anchors", "history", "threshold". It contains iteratively remove anchors exceeding a supplied effect threshold and associated metadata or diagnostics needed to interpret the result.

## eyeprocess_irt_apply_link

*Apply linear IRT scale-linking coefficients*

### Description

Apply linear IRT scale-linking coefficients

### Usage

```
eyeprocess_irt_apply_link(items, link)
```

### Arguments

| | |
|---|---|
| items | Item-parameter data frame or item collection. |
| link | IRT scale-linking coefficients or linking object. |

### Value

An R object containing linear IRT scale-linking coefficients. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

## eyeprocess_irt_bank_coverage

*Audit item-bank information coverage across a theta region*

---

### Description

Audit item-bank information coverage across a theta region

### Usage

```
eyeprocess_irt_bank_coverage(
  items,
  theta = seq(-4, 4, length.out = 161),
  target_information = 5,
  target = c(-2, 2)
)
```

### Arguments

| | |
|---|---|
| items | Item-parameter data frame or item collection. |
| theta | Latent-trait value or vector of latent-trait values. |
| target_information | Target test-information level. |
| target | Target level, distribution, or criterion. |

### Value

An object of class "eye_irt_bank_coverage", stored as a named list, with components "curve", "target", "target_information", "fraction_target_met", "minimum_information", "maximum_sem", "gaps". It contains item-bank information coverage across a theta region and associated metadata or diagnostics needed to interpret the result.

## eyeprocess_irt_category_function_audit

*Audit category probability functions*

### Description

Audit category probability functions

### Usage

```
eyeprocess_irt_category_function_audit(probabilities, tolerance = 1e-08)
```

### Arguments

| | |
|---|---|
| probabilities | Probability matrix or vector, with dimensions appropriate to the model. |
| tolerance | Numerical or decision tolerance. |

### Value

An object of class "eye_irt_category_audit", stored as a named list, with components "valid_bounds", "rows_sum_to_one", "max_sum_error", "min_probability", "max_probability". It contains category probability functions and associated metadata or diagnostics needed to interpret the result.

## eyeprocess_irt_classification_precision

*Summarise decision precision at one or more theta cut scores*

### Description

Summarise decision precision at one or more theta cut scores

### Usage

```
eyeprocess_irt_classification_precision(
  theta_estimate,
  standard_error,
  cut_score = 0,
  confidence = 0.95
)
```

### Arguments

| | |
|---|---|
| theta_estimate | Estimated latent-trait values. |
| standard_error | Standard errors corresponding to the estimates. |
| cut_score | Latent-scale classification cut score. |
| confidence | Requested confidence level. |

## Value

A tabular R object containing decision precision at one or more theta cut scores; rows represent analysis units and columns contain the returned quantities.

---

`eyeprocess_irt_conditional_sem`
*Conditional standard error from information*

---

## Description

Conditional standard error from information

## Usage

```
eyeprocess_irt_conditional_sem(information)
```

## Arguments

| | |
|---|---|
| `information` | Item or test information value or vector. |

## Value

A logical value or vector indicating conditional standard error from information.

---

`eyeprocess_irt_content_balance_audit`
*Audit content balance in an administered adaptive form*

---

## Description

Audit content balance in an administered adaptive form

## Usage

```
eyeprocess_irt_content_balance_audit(administered, item_bank, target = NULL)
```

## Arguments

| | |
|---|---|
| `administered` | Identifiers or records for administered items. |
| `item_bank` | Item bank or item-bank data frame. |
| `target` | Target level, distribution, or criterion. |

## Value

A data frame containing content balance in an administered adaptive form. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## eyeprocess_irt_device_drift

*Summarise parameter drift across acquisition devices*

### Description

Summarise parameter drift across acquisition devices

### Usage

```
eyeprocess_irt_device_drift(
  parameters,
  item_id = "item_id",
  device = "device",
  parameter = "b"
)
```

### Arguments

| | |
|---|---|
| `parameters` | Item-parameter data frame or parameter estimates. |
| `item_id` | Item identifier or vector of item identifiers. |
| `device` | Device identifier or grouping variable. |
| `parameter` | Name of the parameter to compare. |

### Value

A tabular R object containing parameter drift across acquisition devices; rows represent analysis units and columns contain the returned quantities.

## eyeprocess_irt_dif_effect_curve

*Differential item functioning effect curve from two parameter sets*

### Description

Differential item functioning effect curve from two parameter sets

### Usage

```
eyeprocess_irt_dif_effect_curve(
  reference_item,
  focal_item,
  theta = seq(-4, 4, length.out = 81)
)
```

### Arguments

| | |
|---|---|
| reference_item | Reference-group item parameters. |
| focal_item | Focal-group item parameters. |
| theta | Latent-trait value or vector of latent-trait values. |

### Value

An object of class "eye_irt_dif_curve", "data.frame", stored as a data frame, containing differential item functioning effect curve from two parameter sets and associated metadata needed to interpret the result.

---

## eyeprocess_irt_dtf_curve

*Differential test functioning effect curve*

---

### Description

Differential test functioning effect curve

### Usage

```
eyeprocess_irt_dtf_curve(reference, focal, theta = seq(-4, 4, length.out = 81))
```

### Arguments

| | |
|---|---|
| reference | Reference-form or reference-group item parameters. |
| focal | Focal-form or focal-group item parameters. |
| theta | Latent-trait value or vector of latent-trait values. |

### Value

An object of class "eye_irt_dtf_curve", "data.frame", stored as a data frame, containing differential test functioning effect curve and associated metadata needed to interpret the result.

## eyeprocess_irt_eap_score

*EAP score for dichotomous IRT item parameters*

### Description

EAP score for dichotomous IRT item parameters

### Usage

```
eyeprocess_irt_eap_score(
  response,
  items,
  theta_grid = seq(-4, 4, length.out = 81),
  prior_mean = 0,
  prior_sd = 1,
  D = 1
)
```

### Arguments

| | |
|---|---|
| response | Observed item response or response variable. |
| items | Item-parameter data frame or item collection. |
| theta_grid | Grid of latent-trait values used for numerical scoring or integration. |
| prior_mean | Mean of the normal latent-trait prior. |
| prior_sd | Standard deviation of the normal latent-trait prior. |
| D | Logistic scaling constant. |

### Value

An object of class "eye_irt_score", stored as a named list, with components "estimate", "se", "theta", "posterior", "method". It contains eAP score for dichotomous IRT item parameters and associated metadata or diagnostics needed to interpret the result.

## eyeprocess_irt_engine_evidence_table

*Build an external-engine capability and availability table*

### Description

Build an external-engine capability and availability table

### Usage

```
eyeprocess_irt_engine_evidence_table()
```

## Value

A data frame containing an external-engine capability and availability table. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

---

`eyeprocess_irt_engine_registry`
*Registry of specialized external IRT engines*

---

## Description

Registry of specialized external IRT engines

## Usage

```
eyeprocess_irt_engine_registry()
```

## Value

An object of class "eye_irt_engine_registry", "data.frame", stored as a data frame, containing registry of specialized external IRT engines and associated metadata needed to interpret the result.

---

`eyeprocess_irt_engine_status`
*Query an external IRT engine*

---

## Description

Query an external IRT engine

## Usage

```
eyeprocess_irt_engine_status(engine)
```

## Arguments

| | |
|---|---|
| `engine` | Requested estimation or analysis engine. |

## Value

An object of class "eye_irt_engine_registry", "data.frame", stored as a data frame, containing query an external IRT engine and associated metadata needed to interpret the result.

## eyeprocess_irt_expected_score *Expected item score*

### Description

Expected item score

### Usage

```
eyeprocess_irt_expected_score(
  theta,
  family = c("2pl", "3pl", "4pl", "grm", "gpcm", "nominal"),
  ...
)
```

### Arguments

| | |
|---|---|
| theta | Latent-trait value or vector of latent-trait values. |
| family | IRT response family or model family. |
| ... | Additional arguments passed to the selected method or external engine. |

### Value

A numeric value or vector containing expected item score.

## eyeprocess_irt_exposure_summary *Summarise item exposure rates*

### Description

Summarise item exposure rates

### Usage

```
eyeprocess_irt_exposure_summary(
  administered,
  item_bank_ids = unique(administered)
)
```

### Arguments

| | |
|---|---|
| administered | Identifiers or records for administered items. |
| item_bank_ids | Complete set of item identifiers in the bank. |

## Value

A data frame containing item exposure rates. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

---

`eyeprocess_irt_extreme_score_audit`
*Audit extreme response scores without assigning behavioral labels*

---

## Description

Audit extreme response scores without assigning behavioral labels

## Usage

```
eyeprocess_irt_extreme_score_audit(
  responses,
  lower_fraction = 0.02,
  upper_fraction = 0.98
)
```

## Arguments

| | |
|---|---|
| `responses` | Response matrix or response data. |
| `lower_fraction` | Lower extreme-score fraction. |
| `upper_fraction` | Upper extreme-score fraction. |

## Value

A data frame containing extreme response scores without assigning behavioral labels. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

---

`eyeprocess_irt_fit_dashboard`
*Build an integrated IRT diagnostic dashboard object*

---

## Description

Build an integrated IRT diagnostic dashboard object

### Usage

```
eyeprocess_irt_fit_dashboard(
  item_fit = NULL,
  person_fit = NULL,
  q3 = NULL,
  parameter_audit = NULL,
  identification = NULL
)
```

### Arguments

| | |
|---|---|
| item_fit | Item-fit diagnostic object or table. |
| person_fit | Person-fit diagnostic object or table. |
| q3 | Q3 residual-correlation matrix or summary. |
| parameter_audit | Item-parameter plausibility audit. |
| identification | Identification specification or identification audit. |

### Value

An object of class "eye_irt_fit_dashboard", stored as a named list, with components "components", "present", "n_components", "interpretation". It contains an integrated IRT diagnostic dashboard object and associated metadata or diagnostics needed to interpret the result.

---

## eyeprocess_irt_functioning_effect_summary

*Summarise DIF/DTF curve magnitude*

---

### Description

Summarise DIF/DTF curve magnitude

### Usage

```
eyeprocess_irt_functioning_effect_summary(curve)
```

### Arguments

| | |
|---|---|
| curve | Curve data to summarize or inspect. |

### Value

A named list with components "max_abs", "mean_abs", "signed_area", containing dIF/DTF curve magnitude and associated metadata or diagnostics.

---

`eyeprocess_irt_gpcm_probability`
*Generalized partial-credit category probabilities*

---

## Description

Generalized partial-credit category probabilities

## Usage

```
eyeprocess_irt_gpcm_probability(theta, a = 1, steps, D = 1)
```

## Arguments

| | |
|---|---|
| `theta` | Latent-trait value or vector of latent-trait values. |
| `a` | Item discrimination parameter or vector. |
| `steps` | Step parameters for a generalized partial-credit model. |
| `D` | Logistic scaling constant. |

## Value

A logical value or vector indicating generalized partial-credit category probabilities.

---

`eyeprocess_irt_grm_probability`
*Graded-response category probabilities*

---

## Description

Graded-response category probabilities

## Usage

```
eyeprocess_irt_grm_probability(theta, a = 1, thresholds, D = 1)
```

## Arguments

| | |
|---|---|
| `theta` | Latent-trait value or vector of latent-trait values. |
| `a` | Item discrimination parameter or vector. |
| `thresholds` | Ordered response-category thresholds. |
| `D` | Logistic scaling constant. |

## Value

A logical value or vector indicating graded-response category probabilities.

## eyeprocess_irt_haebara_link

*Haebara item-characteristic-curve linking*

### Description

Haebara item-characteristic-curve linking

### Usage

```
eyeprocess_irt_haebara_link(
  reference,
  focal,
  anchors = NULL,
  theta = seq(-4, 4, length.out = 81),
  weights = NULL,
  start = c(A = 1, B = 0)
)
```

### Arguments

| | |
|---|---|
| reference | Reference-form or reference-group item parameters. |
| focal | Focal-form or focal-group item parameters. |
| anchors | Anchor-item identifiers. |
| theta | Latent-trait value or vector of latent-trait values. |
| weights | Optional numerical weights. |
| start | Starting values for numerical optimization. |

### Value

An object of class "eye_irt_link", stored as a named list, with components "A", "B", "method", "anchors", "objective", "convergence". It contains haebara item-characteristic-curve linking and associated metadata or diagnostics needed to interpret the result.

## eyeprocess_irt_identification_audit

*Audit IRT scale/location identification*

### Description

Audit IRT scale/location identification

### Usage

```
eyeprocess_irt_identification_audit(
  spec,
  constraints = list(),
  n_items = NULL,
  n_persons = NULL
)
```

### Arguments

| | |
|---|---|
| spec | Model, validation, or analysis specification object. |
| constraints | Identification or model constraints. |
| n_items | Number of items. |
| n_persons | Number of persons. |

### Value

An object of class "eye_irt_identification_audit", stored as a named list, with components "spec", "location_identified", "scale_identified", "anchors", "warnings", "valid", "n_items", "n_persons". It contains iRT scale/location identification and associated metadata or diagnostics needed to interpret the result.

---

## eyeprocess_irt_infit_outfit

*Compute residual-based Infit and Outfit summaries*

---

### Description

These statistics summarize response-model residual behavior. They are not labels for motivation, misconduct, diagnosis, or respondent intent.

### Usage

```
eyeprocess_irt_infit_outfit(
  observed,
  expected,
  by = c("item", "person"),
  min_variance = 1e-08
)
```

### Arguments

| | |
|---|---|
| observed | Observed responses or observed values. |
| expected | Model-expected probabilities or expected values. |
| by | Grouping variables or aggregation level. |
| min_variance | Minimum variance used to stabilize residual calculations. |

### Value

A data frame containing residual-based Infit and Outfit summaries. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

---

## eyeprocess_irt_information_area

*Area under an information curve*

---

### Description

Area under an information curve

### Usage

```
eyeprocess_irt_information_area(theta, information)
```

### Arguments

| | |
|---|---|
| theta | Latent-trait value or vector of latent-trait values. |
| information | Item or test information value or vector. |

### Value

A numeric value or vector containing area under an information curve.

---

## eyeprocess_irt_information_gain

*Information gain between two conditional standard errors*

---

### Description

Information gain between two conditional standard errors

### Usage

```
eyeprocess_irt_information_gain(se_before, se_after)
```

### Arguments

| | |
|---|---|
| se_before | Conditional standard error before an item is administered. |
| se_after | Conditional standard error after item administration. |

### Value

A numeric value or vector containing information gain between two conditional standard errors.

## eyeprocess_irt_information_targeting *Audit how well item information targets a theta distribution*

### Description

Audit how well item information targets a theta distribution

### Usage

```
eyeprocess_irt_information_targeting(items, theta, weights = NULL, D = 1)
```

### Arguments

| | |
|---|---|
| items | Item-parameter data frame or item collection. |
| theta | Latent-trait value or vector of latent-trait values. |
| weights | Optional numerical weights. |
| D | Logistic scaling constant. |

### Value

An object of class "eye_irt_information_targeting", stored as a named list, with components "weighted_information", "weighted_sem", "curve", "weights". It contains how well item information targets a theta distribution and associated metadata or diagnostics needed to interpret the result.

## eyeprocess_irt_invariance_evidence *Combine invariance evidence without converting it to a binary validity claim*

### Description

Combine invariance evidence without converting it to a binary validity claim

### Usage

```
eyeprocess_irt_invariance_evidence(
  anchor_audit = NULL,
  dif = NULL,
  dtf = NULL,
  linking = NULL,
  process_concordance = NULL
)
```

### Arguments

| | |
|---|---|
| anchor_audit | Anchor-item audit result. |
| dif | Differential-item-functioning evidence or summary. |
| dtf | Differential-test-functioning evidence or curve. |
| linking | Scale-linking evidence or result. |
| process_concordance | Process-DIF concordance evidence. |

### Value

An object of class "eye_irt_invariance_evidence", stored as a named list, with components "components", "present", "completeness", "interpretation". It contains combine invariance evidence without converting it to a binary validity claim and associated metadata or diagnostics needed to interpret the result.

---

## eyeprocess_irt_item_bank

*Item bank object for adaptive design*

---

### Description

Item bank object for adaptive design

### Usage

```
eyeprocess_irt_item_bank(items, content = NULL, exposure_limit = 1)
```

### Arguments

| | |
|---|---|
| items | Item-parameter data frame or item collection. |
| content | Item content/category metadata. |
| exposure_limit | Maximum permitted item exposure. |

### Value

An object of class "eye_irt_item_bank", stored as a named list, with components "items", "exposure_limit". It contains item bank object for adaptive design and associated metadata or diagnostics needed to interpret the result.

## eyeprocess_irt_item_fit_residuals

*Compute item residual fit summaries from observed and predicted probabilities*

### Description

Compute item residual fit summaries from observed and predicted probabilities

### Usage

```
eyeprocess_irt_item_fit_residuals(
  responses,
  probabilities,
  item_ids = colnames(responses)
)
```

### Arguments

| | |
|---|---|
| `responses` | Response matrix or response data. |
| `probabilities` | Probability matrix or vector, with dimensions appropriate to the model. |
| `item_ids` | Optional item identifiers. |

### Value

An object of class "eye_irt_item_fit", "data.frame", stored as a data frame, containing item residual fit summaries from observed and predicted probabilities and associated metadata needed to interpret the result.

## eyeprocess_irt_item_information

*Compute item information for transparent IRT families*

### Description

Compute item information for transparent IRT families

### Usage

```
eyeprocess_irt_item_information(
  theta,
  family = c("2pl", "3pl", "4pl", "grm", "gpcm", "nominal"),
  ...,
  D = 1
)
```

### Arguments

| | |
|---|---|
| theta | Latent-trait value or vector of latent-trait values. |
| family | IRT response family or model family. |
| ... | Additional arguments passed to the selected method or external engine. |
| D | Logistic scaling constant. |

### Value

A numeric value or vector containing item information for transparent IRT families.

---

## eyeprocess_irt_item_selection

*Select the most informative eligible item at a theta estimate*

---

### Description

Select the most informative eligible item at a theta estimate

### Usage

```
eyeprocess_irt_item_selection(
  bank,
  theta,
  administered = character(),
  exposure = NULL,
  content_required = NULL,
  D = 1
)
```

### Arguments

| | |
|---|---|
| bank | Validated item-bank object. |
| theta | Latent-trait value or vector of latent-trait values. |
| administered | Identifiers or records for administered items. |
| exposure | Item exposure information used by the adaptive-selection rule. |
| content_required | Content constraints required for item selection. |
| D | Logistic scaling constant. |

### Value

An object of class "eye_irt_item_selection", stored as a named list, with components "selected", "information", "theta", "reason", "candidate_count". It contains the most informative eligible item at a theta estimate and associated metadata or diagnostics needed to interpret the result.

`eyeprocess_irt_latent_regression_design` *Build a latent-regression design matrix with explicit centering metadata*

## Description

Build a latent-regression design matrix with explicit centering metadata

## Usage

```
eyeprocess_irt_latent_regression_design(data, formula, center_numeric = TRUE)
```

## Arguments

| | |
|---|---|
| `data` | Input data frame, matrix, or compatible analysis object. |
| `formula` | Model formula. |
| `center_numeric` | Whether numeric predictors are centered. |

## Value

An object of class "eye_irt_latent_regression_design", stored as a named list, with components "matrix", "formula", "centers", "complete". It contains a latent-regression design matrix with explicit centering metadata and associated metadata or diagnostics needed to interpret the result.

`eyeprocess_irt_link_stability` *Compare linking estimates across anchor subsets*

## Description

Compare linking estimates across anchor subsets

## Usage

```
eyeprocess_irt_link_stability(
  reference,
  focal,
  anchor_sets,
  method = c("mean-sigma", "mean-mean", "Stocking-Lord", "Haebara")
)
```

### Arguments

| | |
|---|---|
| reference | Reference-form or reference-group item parameters. |
| focal | Focal-form or focal-group item parameters. |
| anchor_sets | Value supplied for the anchor sets argument. |
| method | Scoring, linking, or analysis method. |

### Value

An object of class "eye_irt_link_stability", stored as a named list, with components "table", "sd_A", "sd_B", "method". It contains linking estimates across anchor subsets and associated metadata or diagnostics needed to interpret the result.

---

## eyeprocess_irt_local_dependence_pairs
*Extract high residual-dependence item pairs*

---

### Description

Extract high residual-dependence item pairs

### Usage

```
eyeprocess_irt_local_dependence_pairs(q3, threshold = 0.2, absolute = TRUE)
```

### Arguments

| | |
|---|---|
| q3 | Q3 residual-correlation matrix or summary. |
| threshold | Decision or diagnostic threshold. |
| absolute | Whether diagnostic thresholds apply to absolute values. |

### Value

A data frame containing high residual-dependence item pairs. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## eyeprocess_irt_map_score

*MAP score for dichotomous IRT item parameters*

### Description

MAP score for dichotomous IRT item parameters

### Usage

```
eyeprocess_irt_map_score(
  response,
  items,
  bounds = c(-6, 6),
  prior_mean = 0,
  prior_sd = 1,
  D = 1
)
```

### Arguments

| | |
|---|---|
| response | Observed item response or response variable. |
| items | Item-parameter data frame or item collection. |
| bounds | Numerical lower and upper optimization bounds. |
| prior_mean | Mean of the normal latent-trait prior. |
| prior_sd | Standard deviation of the normal latent-trait prior. |
| D | Logistic scaling constant. |

### Value

An object of class "eye_irt_score", stored as a named list, with components "estimate", "objective", "method", "bounds". It contains mAP score for dichotomous IRT item parameters and associated metadata or diagnostics needed to interpret the result.

## eyeprocess_irt_marginal_reliability

*Marginal reliability from latent-score variance and conditional error variance*

### Description

Marginal reliability from latent-score variance and conditional error variance

### Usage

```
eyeprocess_irt_marginal_reliability(theta_estimate, se)
```

### Arguments

| | |
|---|---|
| theta_estimate | Estimated latent-trait values. |
| se | Standard-error values. |

### Value

A numeric value or vector containing marginal reliability from latent-score variance and conditional error variance.

---

## eyeprocess_irt_mean_mean_link

*Mean-mean IRT linking coefficients*

---

### Description

Mean-mean IRT linking coefficients

### Usage

```
eyeprocess_irt_mean_mean_link(reference, focal, anchors = NULL)
```

### Arguments

| | |
|---|---|
| reference | Reference-form or reference-group item parameters. |
| focal | Focal-form or focal-group item parameters. |
| anchors | Anchor-item identifiers. |

### Value

An object of class "eye_irt_link", stored as a named list, with components "A", "B", "method", "anchors", "objective". It contains mean-mean IRT linking coefficients and associated metadata or diagnostics needed to interpret the result.

## eyeprocess_irt_mean_sigma_link

*Mean-sigma IRT linking coefficients*

### Description

Mean-sigma IRT linking coefficients

### Usage

```
eyeprocess_irt_mean_sigma_link(reference, focal, anchors = NULL)
```

### Arguments

| | |
|---|---|
| reference | Reference-form or reference-group item parameters. |
| focal | Focal-form or focal-group item parameters. |
| anchors | Anchor-item identifiers. |

### Value

An object of class "eye_irt_link", stored as a named list, with components "A", "B", "method", "anchors", "objective". It contains mean-sigma IRT linking coefficients and associated metadata or diagnostics needed to interpret the result.

## eyeprocess_irt_measurement_precision_profile

*Summarise measurement precision across a theta region*

### Description

Summarise measurement precision across a theta region

### Usage

```
eyeprocess_irt_measurement_precision_profile(
  theta,
  items,
  target = c(-2, 2),
  D = 1
)
```

### Arguments

| | |
|---|---|
| theta | Latent-trait value or vector of latent-trait values. |
| items | Item-parameter data frame or item collection. |
| target | Target level, distribution, or criterion. |
| D | Logistic scaling constant. |

### Value

An object of class "eye_irt_precision_profile", stored as a named list, with components "curve", "target", "area", "min_information", "max_sem". It contains measurement precision across a theta region and associated metadata or diagnostics needed to interpret the result.

---

## eyeprocess_irt_missing_by_design_audit

*Audit missing-by-design structure in an IRT response matrix*

---

### Description

Audit missing-by-design structure in an IRT response matrix

### Usage

```
eyeprocess_irt_missing_by_design_audit(
  responses,
  design = NULL,
  min_administered = 1L
)
```

### Arguments

| | |
|---|---|
| responses | Response matrix or response data. |
| design | Validation or simulation design object. |
| min_administered | Minimum number of administered items required for a record. |

### Value

An object of class "eye_irt_missing_design_audit", stored as a named list, with components "n_persons", "n_items", "observed_fraction", "administered_per_person", "administered_per_item", "sparse_persons", "structural_missing", "unexpected_missing", "has_declared_design". It contains missing-by-design structure in an IRT response matrix and associated metadata or diagnostics needed to interpret the result.

## eyeprocess_irt_misspecification_metrics

*Compare recovery under reference and misspecified scenarios*

### Description

Compare recovery under reference and misspecified scenarios

### Usage

```
eyeprocess_irt_misspecification_metrics(
  reference_summary,
  misspecified_summary
)
```

### Arguments

`reference_summary`
: Reference-model validation summary.

`misspecified_summary`
: Misspecified-model validation summary.

### Value

An R object containing recovery under reference and misspecified scenarios. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## eyeprocess_irt_misspecification_suite

*Create a model-misspecification suite*

### Description

Create a model-misspecification suite

### Usage

```
eyeprocess_irt_misspecification_suite()
```

### Value

A data frame containing a model-misspecification suite. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

`eyeprocess_irt_mle_score` *Bounded ML score for dichotomous IRT item parameters*

## Description

Bounded ML score for dichotomous IRT item parameters

## Usage

```
eyeprocess_irt_mle_score(response, items, bounds = c(-6, 6), D = 1)
```

## Arguments

| | |
|---|---|
| `response` | Observed item response or response variable. |
| `items` | Item-parameter data frame or item collection. |
| `bounds` | Numerical lower and upper optimization bounds. |
| `D` | Logistic scaling constant. |

## Value

An object of class "eye_irt_score", stored as a named list, with components "estimate", "objective", "method", "bounds", "boundary". It contains bounded ML score for dichotomous IRT item parameters and associated metadata or diagnostics needed to interpret the result.

`eyeprocess_irt_model_card` *Create a governed IRT model card*

## Description

Create a governed IRT model card

## Usage

```
eyeprocess_irt_model_card(
  spec,
  engine_status = NULL,
  identification = NULL,
  fit_evidence = NULL,
  invariance = NULL,
  validation = NULL,
  intended_use = NULL,
  excluded_interpretations = c("diagnosis", "cheating inference",
    "mental-state inference")
)
```

### Arguments

| | |
|---|---|
| spec | Model, validation, or analysis specification object. |
| engine_status | Availability/status record for the selected estimation engine. |
| identification | Identification specification or identification audit. |
| fit_evidence | Model-fit evidence or diagnostics. |
| invariance | Measurement-invariance evidence or audit. |
| validation | Value supplied for the validation argument. |
| intended_use | Statement of the intended analytical use. |
| excluded_interpretations | Interpretations explicitly excluded by the model card. |

### Value

An object of class "eye_irt_model_card", stored as a named list, with components "specification", "engine_status", "identification", "fit_evidence", "invariance", "validation", "intended_use", "excluded_interpretations", "created", "hash". It contains a governed IRT model card and associated metadata or diagnostics needed to interpret the result.

---

## eyeprocess_irt_model_card_audit *Audit completeness of an IRT model card*

---

### Description

Audit completeness of an IRT model card

### Usage

```
eyeprocess_irt_model_card_audit(card)
```

### Arguments

| | |
|---|---|
| card | Value supplied for the card argument. |

### Value

A data frame containing completeness of an IRT model card. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

`eyeprocess_irt_model_spec` *Declare an eyeprocess IRT model specification*

## Description

This specification records a psychometric model contract. Estimation is delegated to explicit engines where required; the specification itself does not fit a model.

## Usage

```
eyeprocess_irt_model_spec(
 family = c("rasch", "2pl", "3pl", "4pl", "grm", "gpcm", "nominal", "multidimensional",
    "testlet", "latent_regression", "cdm", "joint_rt"),
  dimensions = 1L,
  identification = c("theta_standard", "item_sum_zero", "anchor"),
  engine = c("native_math", "mirt", "TAM", "GDINA", "LNIRT", "eRm", "custom"),
  process_channels = character(),
  status = c("reference", "experimental", "gated"),
  notes = NULL
)
```

## Arguments

| | |
|---|---|
| `family` | IRT response family or model family. |
| `dimensions` | Value supplied for the dimensions argument. |
| `identification` | Identification specification or identification audit. |
| `engine` | Requested estimation or analysis engine. |
| `process_channels` | Declared process-measure channels. |
| `status` | Evidence, model, or governance status. |
| `notes` | Value supplied for the notes argument. |

## Value

An object of class "eyeprocess_irt_model_spec", stored as a named list, with components "family", "dimensions", "identification", "engine", "process_channels", "status", "notes". It contains declare an eyeprocess IRT model specification and associated metadata or diagnostics needed to interpret the result.

`eyeprocess_irt_monotonicity_audit` *Audit monotonicity of an item response curve*

## Description

Audit monotonicity of an item response curve

## Usage

```
eyeprocess_irt_monotonicity_audit(theta, probability, tolerance = 1e-08)
```

## Arguments

| | |
|---|---|
| `theta` | Latent-trait value or vector of latent-trait values. |
| `probability` | Model-implied probability vector. |
| `tolerance` | Numerical or decision tolerance. |

## Value

An object of class "eye_irt_monotonicity_audit", stored as a named list, with components "monotone_non_decreasing", "n_decreases", "largest_decrease", "theta", "probability". It contains monotonicity of an item response curve and associated metadata or diagnostics needed to interpret the result.

---

`eyeprocess_irt_nominal_probability` *Nominal-response category probabilities*

## Description

Nominal-response category probabilities

## Usage

```
eyeprocess_irt_nominal_probability(theta, slopes, intercepts)
```

## Arguments

| | |
|---|---|
| `theta` | Latent-trait value or vector of latent-trait values. |
| `slopes` | Nominal-category slope parameters. |
| `intercepts` | Nominal-category intercept parameters. |

## Value

A numeric value or vector containing nominal-response category probabilities.

## eyeprocess_irt_parameter_plausibility_audit

*Audit basic plausibility of dichotomous item parameters*

### Description

Audit basic plausibility of dichotomous item parameters

### Usage

```
eyeprocess_irt_parameter_plausibility_audit(
  items,
  discrimination = c(0.2, 4),
  difficulty = c(-6, 6),
  lower_asymptote = c(0, 0.5),
  upper_asymptote = c(0.5, 1)
)
```

### Arguments

| | |
|---|---|
| `items` | Item-parameter data frame or item collection. |
| `discrimination` | Discrimination vector or matrix. |
| `difficulty` | Item difficulty or location parameter. |
| `lower_asymptote` | Lower-asymptote parameter values. |
| `upper_asymptote` | Upper-asymptote parameter values. |

### Value

A data frame containing basic plausibility of dichotomous item parameters. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## eyeprocess_irt_person_fit_lz

*Standardized log-likelihood person-fit diagnostic*

### Description

Computes a Bernoulli response-pattern log-likelihood standardized against its model-implied mean and variance. Extreme values are model diagnostics, not evidence of cheating, disengagement, or a psychological state.

## Usage

```
eyeprocess_irt_person_fit_lz(observed, expected, min_probability = 1e-08)
```

## Arguments

| | |
|---|---|
| observed | Observed responses or observed values. |
| expected | Model-expected probabilities or expected values. |
| min_probability | Lower probability bound used for numerical stabilization. |

## Value

A data frame containing standardized log-likelihood person-fit diagnostic. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

---

# eyeprocess_irt_person_fit_residuals

*Compute person residual fit summaries*

---

## Description

Compute person residual fit summaries

## Usage

```
eyeprocess_irt_person_fit_residuals(
  responses,
  probabilities,
  person_ids = rownames(responses)
)
```

## Arguments

| | |
|---|---|
| responses | Response matrix or response data. |
| probabilities | Probability matrix or vector, with dimensions appropriate to the model. |
| person_ids | Optional person identifiers. |

## Value

An object of class "eye_irt_person_fit", "data.frame", stored as a data frame, containing person residual fit summaries and associated metadata needed to interpret the result.

`eyeprocess_irt_plausible_values` *Draw plausible values from a discrete posterior grid*

## Description

Draw plausible values from a discrete posterior grid

## Usage

```
eyeprocess_irt_plausible_values(score, n = 5L, seed = 1L)
```

## Arguments

| | |
|---|---|
| `score` | Score object containing posterior or uncertainty information. |
| `n` | Number of values, draws, or plausible values to generate. |
| `seed` | Random-number seed for reproducible execution. |

## Value

An R object containing plausible values from a discrete posterior grid. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

`eyeprocess_irt_ppc_discrepancy` *Compare observed and replicated IRT discrepancy statistics*

## Description

Compare observed and replicated IRT discrepancy statistics

## Usage

```
eyeprocess_irt_ppc_discrepancy(
  observed,
  replicated,
  statistic = c("mean_score", "score_sd", "item_means", "max_item_residual")
)
```

## Arguments

| | |
|---|---|
| `observed` | Observed responses or observed values. |
| `replicated` | Replicated data or replicated statistic values. |
| `statistic` | Discrepancy statistic or statistic function. |

## Value

An object of class "eye_irt_ppc_discrepancy", stored as a named list, with components "statistic", "observed", "replicated", "posterior_predictive_p", "interval". It contains observed and replicated IRT discrepancy statistics and associated metadata or diagnostics needed to interpret the result.

---

`eyeprocess_irt_precision_evidence_table`

*Build a paper-ready IRT information/precision table*

---

## Description

Build a paper-ready IRT information/precision table

## Usage

```
eyeprocess_irt_precision_evidence_table(
  items,
  theta = seq(-3, 3, by = 0.5),
  digits = 4L
)
```

## Arguments

| | |
|---|---|
| `items` | Item-parameter data frame or item collection. |
| `theta` | Latent-trait value or vector of latent-trait values. |
| `digits` | Number of decimal digits used for presentation. |

## Value

An R object containing a paper-ready IRT information/precision table. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

`eyeprocess_irt_prior_sensitivity_grid`

*Construct a prior-sensitivity grid for Bayesian IRT analyses*

---

## Description

Construct a prior-sensitivity grid for Bayesian IRT analyses

### Usage

```
eyeprocess_irt_prior_sensitivity_grid(
  discrimination_scale = c(0.5, 1, 1.5),
  difficulty_scale = c(1, 2),
  guessing_mean = c(0.1, 0.2)
)
```

### Arguments

| | |
|---|---|
| discrimination_scale | Prior scale for item discrimination. |
| difficulty_scale | Prior scale for item difficulty or location. |
| guessing_mean | Prior mean for the lower-asymptote or guessing parameter. |

### Value

An R object containing a prior-sensitivity grid for Bayesian IRT analyses. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

## eyeprocess_irt_prior_sensitivity_summary

*Summarise sensitivity of an estimand across declared prior specifications*

---

### Description

Summarise sensitivity of an estimand across declared prior specifications

### Usage

```
eyeprocess_irt_prior_sensitivity_summary(
  results,
  prior_id = "prior_id",
  estimate = "estimate"
)
```

### Arguments

| | |
|---|---|
| results | Results table or analysis results. |
| prior_id | Identifier for the prior specification. |
| estimate | Estimate column or numerical estimates to summarize. |

## Value

An object of class "eye_irt_prior_sensitivity", stored as a named list, with components "n_specifications", "n_finite", "median", "range", "sd", "table", "guardrail". It contains sensitivity of an estimand across declared prior specifications and associated metadata or diagnostics needed to interpret the result.

---

`eyeprocess_irt_prior_spec` *Declare prior families for Bayesian IRT engine adapters*

---

## Description

Declare prior families for Bayesian IRT engine adapters

## Usage

```
eyeprocess_irt_prior_spec(
  discrimination = c("lognormal", "normal"),
  difficulty = "normal",
  guessing = c("beta", "logit-normal"),
  location = 0,
  scale = 1,
  guessing_shape = c(5, 17),
  label = "default"
)
```

## Arguments

| | |
|---|---|
| `discrimination` | Discrimination vector or matrix. |
| `difficulty` | Item difficulty or location parameter. |
| `guessing` | Prior distribution family for the guessing parameter. |
| `location` | Prior location hyperparameter. |
| `scale` | Prior scale hyperparameter. |
| `guessing_shape` | Shape parameters for the guessing prior. |
| `label` | Human-readable label. |

## Value

An object of class "eye_irt_prior_spec", stored as a named list, with components "discrimination", "difficulty", "guessing", "location", "scale", "guessing_shape", "label". It contains declare prior families for Bayesian IRT engine adapters and associated metadata or diagnostics needed to interpret the result.

## eyeprocess_irt_process_alignment

*Align item parameters with process-channel summaries*

### Description

Align item parameters with process-channel summaries

### Usage

```
eyeprocess_irt_process_alignment(
  item_parameters,
  process_profile,
  process_columns = NULL
)
```

### Arguments

| | |
|---|---|
| item_parameters | Item-parameter data frame. |
| process_profile | Process-measure profile or summary. |
| process_columns | Names of process-measure columns to use. |

### Value

An object of class "eye_irt_process_alignment", stored as a named list, with components "table", "correlations", "guardrail". It contains align item parameters with process-channel summaries and associated metadata or diagnostics needed to interpret the result.

## eyeprocess_irt_process_aware_selection_penalty

*Process-aware selection penalty without mental-state inference*

### Description

Process-aware selection penalty without mental-state inference

### Usage

```
eyeprocess_irt_process_aware_selection_penalty(
  information,
  burden,
  burden_weight = 0,
  quality_risk = 0,
  quality_weight = 0
)
```

### Arguments

| | |
|---|---|
| information | Item or test information value or vector. |
| burden | Item-level process burden or cost measure. |
| burden_weight | Weight applied to the burden penalty. |
| quality_risk | Item-level measurement-quality risk. |
| quality_weight | Weight applied to the quality-risk penalty. |

### Value

A numeric value or vector containing process-aware selection penalty without mental-state inference.

---

## eyeprocess_irt_process_dif_concordance

*Compare psychometric DIF effect sizes with process-channel contrasts*

---

### Description

Compare psychometric DIF effect sizes with process-channel contrasts

### Usage

```
eyeprocess_irt_process_dif_concordance(
  dif,
  process,
  item_id = "item_id",
  dif_effect = "effect",
  process_effect = "effect"
)
```

### Arguments

| | |
|---|---|
| dif | Differential-item-functioning evidence or summary. |
| process | Process-measure columns or process object. |
| item_id | Item identifier or vector of item identifiers. |
| dif_effect | DIF effect used for concordance. |
| process_effect | Process-side item effect used for concordance. |

### Value

An object of class "eye_irt_process_dif_concordance", stored as a named list, with components "n", "correlation", "table", "guardrail". It contains psychometric DIF effect sizes with process-channel contrasts and associated metadata or diagnostics needed to interpret the result.

---

## eyeprocess_irt_q3 *Compute Yen-style Q3 residual correlations*

---

### Description

Compute Yen-style Q3 residual correlations

### Usage

```
eyeprocess_irt_q3(responses, probabilities, use = "pairwise.complete.obs")
```

### Arguments

| | |
|---|---|
| responses | Response matrix or response data. |
| probabilities | Probability matrix or vector, with dimensions appropriate to the model. |
| use | Missing-data handling mode passed to the residual correlation calculation. |

### Value

An R object containing yen-style Q3 residual correlations. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

## eyeprocess_irt_recovery_design *Create an IRT recovery design*

---

### Description

Create an IRT recovery design

### Usage

```
eyeprocess_irt_recovery_design(
  sample_size = c(250L, 750L),
  n_items = c(12L, 24L),
  missing_rate = c(0, 0.15),
  testlet_sd = c(0, 0.35),
  replications = 10L,
  seed = 20260811L
)
```

### Arguments

| | |
|---|---|
| sample_size | Validation sample size or vector of sample sizes. |
| n_items | Number of items. |
| missing_rate | Proportion of responses or observations set missing. |
| testlet_sd | Standard deviation of simulated testlet effects. |
| replications | Number of simulation or validation replications. |
| seed | Random-number seed for reproducible execution. |

### Value

An object of class "eye_irt_recovery_design", "data.frame", stored as a data frame, containing an IRT recovery design and associated metadata needed to interpret the result.

---

## eyeprocess_irt_recovery_failures
*Summarise recovery failure rates*

---

### Description

Summarise recovery failure rates

### Usage

```
eyeprocess_irt_recovery_failures(x)
```

### Arguments

| | |
|---|---|
| x | Object to validate, summarize, verify, or otherwise process. |

### Value

A tabular R object containing recovery failure rates; rows represent analysis units and columns contain the returned quantities.

## eyeprocess_irt_recovery_summary *Summarise IRT parameter recovery*

### Description

Summarise IRT parameter recovery

### Usage

```
eyeprocess_irt_recovery_summary(x)
```

### Arguments

| | |
|---|---|
| x | Object to validate, summarize, verify, or otherwise process. |

### Value

A tabular R object containing iRT parameter recovery; rows represent analysis units and columns contain the returned quantities.

## eyeprocess_irt_sbc_ranks *Construct SBC ranks from scalar truths and posterior draws*

### Description

Construct SBC ranks from scalar truths and posterior draws

### Usage

```
eyeprocess_irt_sbc_ranks(truth, draws, randomize_ties = TRUE, seed = 1L)
```

### Arguments

| | |
|---|---|
| truth | Known simulated parameter value or vector of true values. |
| draws | Posterior draws, with draws arranged by simulation case as required. |
| randomize_ties | Whether ties in SBC ranks are randomized. |
| seed | Random-number seed for reproducible execution. |

### Value

A vector or matrix containing sBC ranks from scalar truths and posterior draws, with shape determined by the supplied analysis units.

`eyeprocess_irt_sbc_summary` *Summarise IRT SBC ranks with the package SBC diagnostics*

## Description

Summarise IRT SBC ranks with the package SBC diagnostics

## Usage

```
eyeprocess_irt_sbc_summary(ranks, n_draws, bins = NULL)
```

## Arguments

| | |
|---|---|
| `ranks` | Simulation-based-calibration rank values. |
| `n_draws` | Number of posterior draws underlying each rank. |
| `bins` | Number of bins used for rank-distribution summaries. |

## Value

An object of class "eye_irt_sbc_evidence", stored as a named list, with components "diagnostics", "ecdf_deviation", "n", "n_draws". It contains iRT SBC ranks with the package SBC diagnostics and associated metadata or diagnostics needed to interpret the result.

`eyeprocess_irt_score_table` *Score a response matrix with EAP, MAP, or ML*

## Description

Score a response matrix with EAP, MAP, or ML

## Usage

```
eyeprocess_irt_score_table(
  responses,
  items,
  method = c("EAP", "MAP", "ML"),
  person_ids = rownames(responses),
  ...
)
```

### Arguments

| | |
|---|---|
| `responses` | Response matrix or response data. |
| `items` | Item-parameter data frame or item collection. |
| `method` | Scoring, linking, or analysis method. |
| `person_ids` | Optional person identifiers. |
| `...` | Additional arguments passed to the selected method or external engine. |

### Value

A tabular R object containing a response matrix with EAP, MAP, or ML; rows represent analysis units and columns contain the returned quantities.

---

## `eyeprocess_irt_score_uncertainty` *Summarise score uncertainty*

---

### Description

Summarise score uncertainty

### Usage

```
eyeprocess_irt_score_uncertainty(scores)
```

### Arguments

| | |
|---|---|
| `scores` | Score object or score table. |

### Value

An object of class "eye_irt_score_uncertainty", stored as a named list, with components "n", "mean_se", "median_se", "p95_se", "marginal_reliability". It contains score uncertainty and associated metadata or diagnostics needed to interpret the result.

## eyeprocess_irt_session_drift *Summarise item-parameter drift over sessions*

### Description

Summarise item-parameter drift over sessions

### Usage

```
eyeprocess_irt_session_drift(
  parameters,
  item_id = "item_id",
  session = "session",
  parameter = "b"
)
```

### Arguments

| | |
|---|---|
| parameters | Item-parameter data frame or parameter estimates. |
| item_id | Item identifier or vector of item identifiers. |
| session | Session identifier or grouping variable. |
| parameter | Name of the parameter to compare. |

### Value

A tabular R object containing item-parameter drift over sessions; rows represent analysis units and columns contain the returned quantities.

## eyeprocess_irt_sparse_design_audit *Audit sparse person-item response coverage*

### Description

Audit sparse person-item response coverage

### Usage

```
eyeprocess_irt_sparse_design_audit(
  data,
  person,
  item,
  response = NULL,
  min_person_items = 3L,
  min_item_persons = 10L
)
```

### Arguments

| | |
|---|---|
| data | Input data frame, matrix, or compatible analysis object. |
| person | Name of the person identifier column. |
| item | Name of the item identifier column. |
| response | Observed item response or response variable. |
| min_person_items | Minimum number of observed items required per person. |
| min_item_persons | Minimum number of observed persons required per item. |

### Value

An object of class "eye_irt_sparse_design_audit", stored as a named list, with components "n_persons", "n_items", "n_observed", "density", "person_counts", "item_counts", "sparse_persons", "sparse_items", "min_person_items", "min_item_persons". It contains sparse person-item response coverage and associated metadata or diagnostics needed to interpret the result.

---

## eyeprocess_irt_stocking_lord_link

*Stocking-Lord characteristic-curve linking*

---

### Description

Stocking-Lord characteristic-curve linking

### Usage

```
eyeprocess_irt_stocking_lord_link(
  reference,
  focal,
  anchors = NULL,
  theta = seq(-4, 4, length.out = 81),
  weights = NULL,
  start = c(A = 1, B = 0)
)
```

### Arguments

| | |
|---|---|
| reference | Reference-form or reference-group item parameters. |
| focal | Focal-form or focal-group item parameters. |
| anchors | Anchor-item identifiers. |
| theta | Latent-trait value or vector of latent-trait values. |
| weights | Optional numerical weights. |
| start | Starting values for numerical optimization. |

## Value

An object of class "eye_irt_link", stored as a named list, with components "A", "B", "method", "anchors", "objective", "convergence". It contains stocking-Lord characteristic-curve linking and associated metadata or diagnostics needed to interpret the result.

---

`eyeprocess_irt_stopping_rule`

*Evaluate a simple adaptive stopping rule*

---

## Description

Evaluate a simple adaptive stopping rule

## Usage

```
eyeprocess_irt_stopping_rule(
  n_administered,
  se = NA_real_,
  min_items = 5L,
  max_items = 30L,
  target_se = 0.3
)
```

## Arguments

| | |
|---|---|
| `n_administered` | Number of items already administered. |
| `se` | Standard-error values. |
| `min_items` | Minimum number of items required. |
| `max_items` | Maximum permitted test length. |
| `target_se` | Target conditional standard error for stopping. |

## Value

A named list with components "stop", "reason", "n_administered", "se", containing a simple adaptive stopping rule and associated metadata or diagnostics.

## eyeprocess_irt_targeting_gap

*Compare an examinee distribution with item-bank targeting*

### Description

Compare an examinee distribution with item-bank targeting

### Usage

```
eyeprocess_irt_targeting_gap(theta, items, breaks = seq(-4, 4, by = 0.5))
```

### Arguments

| | |
|---|---|
| theta | Latent-trait value or vector of latent-trait values. |
| items | Item-parameter data frame or item collection. |
| breaks | Break points used to summarize latent-scale targeting. |

### Value

An object of class "eye_irt_targeting_gap", stored as a named list, with components "table", "absolute_gap", "interpretation". It contains an examinee distribution with item-bank targeting and associated metadata or diagnostics needed to interpret the result.

## eyeprocess_irt_testlet_audit

*Audit testlet sizes and singleton structures*

### Description

Audit testlet sizes and singleton structures

### Usage

```
eyeprocess_irt_testlet_audit(spec, min_items = 2L)
```

### Arguments

| | |
|---|---|
| spec | Model, validation, or analysis specification object. |
| min_items | Minimum number of items required. |

### Value

A data frame containing testlet sizes and singleton structures. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## eyeprocess_irt_testlet_spec

*Declare a testlet structure for bifactor/two-tier IRT engines*

### Description

Declare a testlet structure for bifactor/two-tier IRT engines

### Usage

```
eyeprocess_irt_testlet_spec(item_id, testlet, general_dimension = "general")
```

### Arguments

| | |
|---|---|
| item_id | Item identifier or vector of item identifiers. |
| testlet | Testlet membership identifier. |
| general_dimension | General dimension name or index. |

### Value

An object of class "eye_irt_testlet_spec", "data.frame", stored as a data frame, containing declare a testlet structure for bifactor/two-tier IRT engines and associated metadata needed to interpret the result.

## eyeprocess_irt_test_characteristic_curve

*Test characteristic curve for dichotomous item parameters*

### Description

Test characteristic curve for dichotomous item parameters

### Usage

```
eyeprocess_irt_test_characteristic_curve(theta, items, D = 1)
```

### Arguments

| | |
|---|---|
| theta | Latent-trait value or vector of latent-trait values. |
| items | Item-parameter data frame or item collection. |
| D | Logistic scaling constant. |

### Value

An object of class "eye_irt_test_characteristic_curve", "data.frame", stored as a data frame, containing characteristic curve for dichotomous item parameters and associated metadata needed to interpret the result.

---

`eyeprocess_irt_test_information`

*Compute a test information curve from item parameters*

---

### Description

Compute a test information curve from item parameters

### Usage

```
eyeprocess_irt_test_information(theta, items, D = 1)
```

### Arguments

| | |
|---|---|
| `theta` | Latent-trait value or vector of latent-trait values. |
| `items` | Item-parameter data frame or item collection. |
| `D` | Logistic scaling constant. |

### Value

An object of class "eye_irt_information_profile", "data.frame", stored as a data frame, containing a test information curve from item parameters and associated metadata needed to interpret the result.

---

`eyeprocess_irt_threshold_order_audit`

*Audit ordered category thresholds*

---

### Description

Audit ordered category thresholds

### Usage

```
eyeprocess_irt_threshold_order_audit(item_id, thresholds)
```

### Arguments

| | |
|---|---|
| `item_id` | Item identifier or vector of item identifiers. |
| `thresholds` | Ordered response-category thresholds. |

## Value

An object of class "eye_irt_threshold_audit", stored as a named list, with components "item_id", "thresholds", "ordered", "minimum_gap", "reversals". It contains ordered category thresholds and associated metadata or diagnostics needed to interpret the result.

---

`eyeprocess_joint_process_irt_spec` *Declare a response/process joint IRT specification*

---

## Description

Declare a response/process joint IRT specification

## Usage

```
eyeprocess_joint_process_irt_spec(
  response_family = c("2pl", "rasch", "grm", "gpcm"),
  time_model = c("none", "lognormal", "custom"),
  process_channels = c("dwell", "pupil", "transitions"),
  person_covariates = character(),
  item_covariates = character(),
  missingness = c("ignorable", "modeled", "gated"),
  status = c("experimental", "reference", "gated")
)
```

## Arguments

| | |
|---|---|
| `response_family` | Response-model family. |
| `time_model` | Response-time model specification. |
| `process_channels` | Declared process-measure channels. |
| `person_covariates` | Person-level covariates. |
| `item_covariates` | Item-level covariates. |
| `missingness` | Missing-data handling or missingness specification. |
| `status` | Evidence, model, or governance status. |

## Value

An object of class "eye_joint_process_irt_spec", stored as a named list, with components "response_family", "time_model", "process_channels", "person_covariates", "item_covariates", "missingness", "status". It contains declare a response/process joint IRT specification and associated metadata or diagnostics needed to interpret the result.

## eyeprocess_mirt_directional_information
*Directional multidimensional 2PL information*

### Description

Directional multidimensional 2PL information

### Usage

```
eyeprocess_mirt_directional_information(
  theta,
  discrimination,
  difficulty = 0,
  direction = NULL,
  D = 1
)
```

### Arguments

| | |
|---|---|
| theta | Latent-trait value or vector of latent-trait values. |
| discrimination | Discrimination vector or matrix. |
| difficulty | Item difficulty or location parameter. |
| direction | Direction vector used to project multidimensional information. |
| D | Logistic scaling constant. |

### Value

A numeric value or vector containing directional multidimensional 2PL information.

## eyeprocess_mirt_information_matrix
*Multidimensional 2PL item information matrix*

### Description

Multidimensional 2PL item information matrix

### Usage

```
eyeprocess_mirt_information_matrix(
  theta,
  discrimination,
  difficulty = 0,
  D = 1
)
```

### Arguments

| | |
|---|---|
| theta | Latent-trait value or vector of latent-trait values. |
| discrimination | Discrimination vector or matrix. |
| difficulty | Item difficulty or location parameter. |
| D | Logistic scaling constant. |

### Value

A numeric value or vector containing multidimensional 2PL item information matrix.

---

## eyeprocess_mirt_loading_audit
*Audit multidimensional IRT loading coverage*

---

### Description

Audit multidimensional IRT loading coverage

### Usage

```
eyeprocess_mirt_loading_audit(spec, min_items_per_dimension = 3L)
```

### Arguments

| | |
|---|---|
| spec | Model, validation, or analysis specification object. |
| min_items_per_dimension | Minimum number of items required per dimension. |

### Value

A data frame containing multidimensional IRT loading coverage. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## eyeprocess_mirt_loading_spec *Declare a multidimensional IRT loading structure*

### Description

Declare a multidimensional IRT loading structure

### Usage

```
eyeprocess_mirt_loading_spec(
  items,
  loadings,
  dimension_names = colnames(loadings),
  simple_structure = FALSE
)
```

### Arguments

| | |
|---|---|
| items | Item-parameter data frame or item collection. |
| loadings | Item-by-dimension loading matrix. |
| dimension_names | Optional names for latent dimensions. |
| simple_structure | Whether a simple-structure loading pattern is required. |

### Value

An object of class "eye_mirt_loading_spec", stored as a named list, with components "items", "loadings", "dimensions", "simple_structure", "violations". It contains declare a multidimensional IRT loading structure and associated metadata or diagnostics needed to interpret the result.

## eyeprocess_multichannel_measurement_map *Construct a multichannel measurement map*

### Description

Construct a multichannel measurement map

### Usage

```
eyeprocess_multichannel_measurement_map(
  response = "accuracy",
  channels = c("response_time", "dwell", "pupil", "transitions"),
  role = NULL
)
```

### Arguments

| | |
|---|---|
| response | Observed item response or response variable. |
| channels | Names or definitions of measurement channels. |
| role | Declared role of each measurement channel. |

### Value

A data frame containing a multichannel measurement map. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

---

## eyeprocess_negative_control_evidence_plan

*Declare negative-control evidence targets*

---

### Description

Declare negative-control evidence targets

### Usage

```
eyeprocess_negative_control_evidence_plan(
 controls = c("permutation", "temporal_shift", "placebo_window", "known_leakage"),
  replications = 100L,
  seed = 20260811L
)
```

### Arguments

| | |
|---|---|
| controls | Negative-control methods requested by the evidence plan. |
| replications | Number of simulation or validation replications. |
| seed | Random-number seed for reproducible execution. |

### Value

An object of class "eye_negative_control_evidence_plan", stored as a named list, with components "controls", "replications", "seed", "guardrail". It contains declare negative-control evidence targets and associated metadata or diagnostics needed to interpret the result.

## eyeprocess_negative_control_evidence_table

*Build a paper-ready negative-control table*

### Description

Build a paper-ready negative-control table

### Usage

```
eyeprocess_negative_control_evidence_table(x, digits = 4L)
```

### Arguments

| | |
|---|---|
| x | Object to validate, summarize, verify, or otherwise process. |
| digits | Number of decimal digits used for presentation. |

### Value

An R object containing a paper-ready negative-control table. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## eyeprocess_process_irt_data_bundle

*Prepare a sparse response/process bundle for external joint engines*

### Description

Prepare a sparse response/process bundle for external joint engines

### Usage

```
eyeprocess_process_irt_data_bundle(
  data,
  person,
  item,
  response,
  response_time = NULL,
  process = character(),
  covariates = character()
)
```

### Arguments

| | |
|---|---|
| data | Input data frame, matrix, or compatible analysis object. |
| person | Name of the person identifier column. |
| item | Name of the item identifier column. |
| response | Observed item response or response variable. |
| response_time | Response-time variable or values. |
| process | Process-measure columns or process object. |
| covariates | Optional covariate columns or covariate data. |

### Value

An object of class "eye_process_irt_data_bundle", stored as a named list, with components "data", "person", "item", "response", "response_time", "process", "covariates", "n_persons", "n_items". It contains a sparse response/process bundle for external joint engines and associated metadata or diagnostics needed to interpret the result.

---

## eyeprocess_process_item_profile

*Aggregate process channels by item*

---

### Description

Aggregate process channels by item

### Usage

```
eyeprocess_process_item_profile(data, item, channels)
```

### Arguments

| | |
|---|---|
| data | Input data frame, matrix, or compatible analysis object. |
| item | Name of the item identifier column. |
| channels | Names or definitions of measurement channels. |

### Value

A tabular R object containing aggregate process channels by item; rows represent analysis units and columns contain the returned quantities.

## eyeprocess_process_missingness_pattern

*Summarise missingness patterns across response and process channels*

### Description

Summarise missingness patterns across response and process channels

### Usage

```
eyeprocess_process_missingness_pattern(data, response, channels)
```

### Arguments

| | |
|---|---|
| data | Input data frame, matrix, or compatible analysis object. |
| response | Observed item response or response variable. |
| channels | Names or definitions of measurement channels. |

### Value

A data frame containing missingness patterns across response and process channels. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## eyeprocess_process_person_profile

*Aggregate process channels by person*

### Description

Aggregate process channels by person

### Usage

```
eyeprocess_process_person_profile(data, person, channels)
```

### Arguments

| | |
|---|---|
| data | Input data frame, matrix, or compatible analysis object. |
| person | Name of the person identifier column. |
| channels | Names or definitions of measurement channels. |

### Value

A tabular R object containing aggregate process channels by person; rows represent analysis units and columns contain the returned quantities.

## eyeprocess_recovery_evidence_table

*Build a paper-ready parameter-recovery table*

### Description

Build a paper-ready parameter-recovery table

### Usage

```
eyeprocess_recovery_evidence_table(x, digits = 4L)
```

### Arguments

| | |
|---|---|
| x | Object to validate, summarize, verify, or otherwise process. |
| digits | Number of decimal digits used for presentation. |

### Value

An R object containing a paper-ready parameter-recovery table. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## eyeprocess_reliability_evidence_plan

*Declare reliability evidence targets*

### Description

Declare reliability evidence targets

### Usage

```
eyeprocess_reliability_evidence_plan(
  metrics = c("split_half", "icc", "temporal_stability", "bland_altman"),
  bootstrap = 200L,
  seed = 20260811L
)
```

### Arguments

| | |
|---|---|
| metrics | Reliability metrics requested by the evidence plan. |
| bootstrap | Number of bootstrap replicates or bootstrap configuration. |
| seed | Random-number seed for reproducible execution. |

### Value

An object of class "eye_reliability_evidence_plan", stored as a named list, with components "metrics", "bootstrap", "seed", "guardrail". It contains declare reliability evidence targets and associated metadata or diagnostics needed to interpret the result.

---

## eyeprocess_reliability_evidence_table

*Build a paper-ready reliability table*

---

### Description

Build a paper-ready reliability table

### Usage

```
eyeprocess_reliability_evidence_table(x, digits = 4L)
```

### Arguments

| | |
|---|---|
| x | Object to validate, summarize, verify, or otherwise process. |
| digits | Number of decimal digits used for presentation. |

### Value

An R object containing a paper-ready reliability table. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

## eyeprocess_response_time_profile

*Summarise response-time structure for joint IRT work*

---

### Description

Summarise response-time structure for joint IRT work

### Usage

```
eyeprocess_response_time_profile(data, person, item, response_time)
```

### Arguments

| | |
|---|---|
| data | Input data frame, matrix, or compatible analysis object. |
| person | Name of the person identifier column. |
| item | Name of the item identifier column. |
| response_time | Response-time variable or values. |

### Value

An object of class "eye_response_time_profile", stored as a named list, with components "item", "person", "n". It contains response-time structure for joint IRT work and associated metadata or diagnostics needed to interpret the result.

---

## eyeprocess_sbc_evidence_table

*Build a paper-ready SBC table*

---

### Description

Build a paper-ready SBC table

### Usage

```
eyeprocess_sbc_evidence_table(x, digits = 4L)
```

### Arguments

| | |
|---|---|
| x | Object to validate, summarize, verify, or otherwise process. |
| digits | Number of decimal digits used for presentation. |

### Value

An R object containing a paper-ready SBC table. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

## eyeprocess_speed_accuracy_profile

*Describe speed-accuracy association without causal interpretation*

---

### Description

Describe speed-accuracy association without causal interpretation

### Usage

```
eyeprocess_speed_accuracy_profile(data, person, response, response_time)
```

### Arguments

| | |
|---|---|
| data | Input data frame, matrix, or compatible analysis object. |
| person | Name of the person identifier column. |
| response | Observed item response or response variable. |
| response_time | Response-time variable or values. |

## Value

An object of class "eye_speed_accuracy_profile", stored as a named list, with components "person", "pooled_correlation", "guardrail". It contains describe speed-accuracy association without causal interpretation and associated metadata or diagnostics needed to interpret the result.

---

`eyeprocess_stress_evidence_plan` *Declare the Milestone #2 measurement-quality stress evidence plan*

---

## Description

Declare the Milestone #2 measurement-quality stress evidence plan

## Usage

```
eyeprocess_stress_evidence_plan(
  missing_gaze = c(0, 0.05, 0.15, 0.3),
  pupil_dropout = c(0, 0.05, 0.15),
  calibration_offset = c(0, 0.01, 0.03, 0.06),
  sampling_jitter = c(0, 0.05, 0.15),
  aoi_label_noise = c(0, 0.02, 0.1),
  device_shift = c(0, 0.02, 0.05),
  trial_imbalance = c(0, 0.1, 0.25),
  seed = 20260811L
)
```

## Arguments

| | |
|---|---|
| `missing_gaze` | Severity level for synthetic gaze missingness. |
| `pupil_dropout` | Severity level for synthetic pupil dropout. |
| `calibration_offset` | Magnitude of synthetic calibration offset. |
| `sampling_jitter` | Magnitude of synthetic sampling-time jitter. |
| `aoi_label_noise` | Rate of synthetic AOI-label corruption. |
| `device_shift` | Magnitude of synthetic device shift. |
| `trial_imbalance` | Severity of synthetic trial imbalance. |
| `seed` | Random-number seed for reproducible execution. |

## Value

An object of class "eye_stress_evidence_plan", stored as a named list, with components "seed". It contains declare the Milestone #2 measurement-quality stress evidence plan and associated metadata or diagnostics needed to interpret the result.

---

`eyeprocess_stress_evidence_table`
*Build a paper-ready stress-test table*

---

## Description

Build a paper-ready stress-test table

## Usage

```
eyeprocess_stress_evidence_table(x, digits = 4L)
```

## Arguments

| | |
|---|---|
| `x` | Object to validate, summarize, verify, or otherwise process. |
| `digits` | Number of decimal digits used for presentation. |

## Value

An R object containing a paper-ready stress-test table. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

`eyeprocess_validation_atlas_gaps`
*Summarise gaps in a validation evidence atlas*

---

## Description

Summarise gaps in a validation evidence atlas

## Usage

```
eyeprocess_validation_atlas_gaps(atlas)
```

## Arguments

| | |
|---|---|
| `atlas` | Validation-evidence atlas object. |

## Value

A named list with components "missing_components", "unresolved_claims", "complete", containing gaps in a validation evidence atlas and associated metadata or diagnostics.

`eyeprocess_validation_claim_matrix` *Build a machine-readable validation claim/evidence matrix*

## Description

Build a machine-readable validation claim/evidence matrix

## Usage

```
eyeprocess_validation_claim_matrix(
  claim_id,
  claim,
  evidence_id,
  evidence_type,
  status = "qualified",
  boundary = NA_character_
)
```

## Arguments

| | |
|---|---|
| `claim_id` | Unique identifier for the claim. |
| `claim` | Text of the software-validation claim. |
| `evidence_id` | Identifier of evidence supporting or testing the claim. |
| `evidence_type` | Type or class of evidence. |
| `status` | Evidence, model, or governance status. |
| `boundary` | Explicit interpretation or scope boundary for the claim. |

## Value

A data frame containing a machine-readable validation claim/evidence matrix. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

`eyeprocess_validation_evidence_atlas` *Assemble a validation evidence atlas*

## Description

The atlas is an organizational object. It preserves links between claims, software-validation results, figures, tables, and provenance; it does not upgrade software-validation evidence into construct-validity evidence.

## Usage

```
eyeprocess_validation_evidence_atlas(
  claims,
  recovery = NULL,
  sbc = NULL,
  stress = NULL,
  reliability = NULL,
  negative_controls = NULL,
  irt = NULL,
  provenance = NULL,
  artifacts = NULL
)
```

## Arguments

| | |
|---|---|
| claims | Claim-evidence mapping table. |
| recovery | Parameter-recovery evidence object or table. |
| sbc | Simulation-based-calibration evidence object or table. |
| stress | Measurement-stress evidence object or table. |
| reliability | Reliability or repeatability evidence object or table. |
| negative_controls | Negative-control evidence object or table. |
| irt | IRT-specific evidence object or table. |
| provenance | Provenance metadata or provenance object. |
| artifacts | Artifact table or file-index information. |

## Value

An object of class "eye_validation_evidence_atlas", stored as a named list, with components "claims", "components", "component_status", "coverage", "hash", "guardrail". It contains assemble a validation evidence atlas and associated metadata or diagnostics needed to interpret the result.

---

# eyeprocess_validation_evidence_grade

*Grade the completeness of validation evidence*

---

## Description

Grade the completeness of validation evidence

## Usage

```
eyeprocess_validation_evidence_grade(
  components,
  required = c("design", "execution", "summary", "provenance", "hash")
)
```

### Arguments

| | |
|---|---|
| components | Named evidence components. |
| required | Required evidence components or requirements. |

### Value

An object of class "eye_validation_evidence_grade", stored as a named list, with components "grade", "required", "present", "coverage". It contains grade the completeness of validation evidence and associated metadata or diagnostics needed to interpret the result.

---

## eyeprocess_validation_evidence_index

*Create an index over frozen validation evidence artifacts*

---

### Description

Create an index over frozen validation evidence artifacts

### Usage

```
eyeprocess_validation_evidence_index(root, recursive = TRUE)
```

### Arguments

| | |
|---|---|
| root | Root directory for evidence indexing. |
| recursive | Whether evidence files are indexed recursively. |

### Value

A data frame containing an index over frozen validation evidence artifacts. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

---

## eyeprocess_validation_evidence_manifest

*Create an evidence manifest from files and in-memory objects*

---

### Description

Create an evidence manifest from files and in-memory objects

### Usage

```
eyeprocess_validation_evidence_manifest(
  files = character(),
  objects = list(),
  source_commit = NA_character_,
  label = "eyeprocess-0.9-m2"
)
```

### Arguments

| | |
|---|---|
| files | File paths to include in the evidence manifest. |
| objects | Objects to include in the evidence manifest. |
| source_commit | Source-control commit associated with the evidence. |
| label | Human-readable label. |

### Value

An object of class "eye_validation_evidence_manifest", stored as a named list, with components "label", "source_commit", "files", "objects", "generated_at". It contains an evidence manifest from files and in-memory objects and associated metadata or diagnostics needed to interpret the result.

---

## eyeprocess_validation_plan

*Declare an eyeprocess validation-evidence plan*

---

### Description

Creates a deterministic validation plan. The plan describes software-validation scenarios and does not constitute evidence for the construct validity of any gaze, pupil, response-time, or psychometric measure.

### Usage

```
eyeprocess_validation_plan(
  families = c("recovery", "sbc", "stress", "reliability", "negative_control"),
  sample_size = c(250L, 750L),
  n_items = c(12L, 24L),
  missing_rate = c(0, 0.15),
  noise_level = c("reference", "elevated"),
  specification = c("correct", "misspecified"),
  replications = 20L,
  seed = 20260811L,
  label = "eyeprocess-0.9-m2"
)
```

### Arguments

| | |
|---|---|
| families | Validation-evidence families to include. |
| sample_size | Validation sample size or vector of sample sizes. |
| n_items | Number of items. |
| missing_rate | Proportion of responses or observations set missing. |
| noise_level | Declared simulation noise regime. |
| specification | Whether the validation scenario is correctly specified or deliberately misspecified. |
| replications | Number of simulation or validation replications. |
| seed | Random-number seed for reproducible execution. |
| label | Human-readable label. |

### Value

An object of class "eye_validation_evidence_plan", stored as a named list, with components "families", "sample_size", "n_items", "missing_rate", "noise_level", "specification", "replications", "seed", "label". It contains declare an eyeprocess validation-evidence plan and associated metadata or diagnostics needed to interpret the result.

---

## eyeprocess_validation_readiness

*Evaluate readiness of a Milestone #2 validation evidence bundle*

---

### Description

Evaluate readiness of a Milestone #2 validation evidence bundle

### Usage

```
eyeprocess_validation_readiness(
  x,
 required = c("design", "recovery", "stress", "reliability", "negative_controls",
    "claims", "provenance")
)
```

### Arguments

| | |
|---|---|
| x | Object to validate, summarize, verify, or otherwise process. |
| required | Required evidence components or requirements. |

### Value

An object of class "eye_validation_readiness", stored as a named list, with components "ready", "table", "hash_valid", "source_commit". It contains readiness of a Milestone #2 validation evidence bundle and associated metadata or diagnostics needed to interpret the result.

## eyeprocess_validation_release_gate

*Apply a conservative software-release evidence gate*

### Description

Apply a conservative software-release evidence gate

### Usage

```
eyeprocess_validation_release_gate(
  readiness,
  acceptance = NULL,
  require_hash = TRUE
)
```

### Arguments

| | |
|---|---|
| readiness | Validation-readiness result. |
| acceptance | Acceptance-rule results. |
| require_hash | Whether a verified integrity hash is required. |

### Value

An object of class "eye_validation_release_gate", stored as a named list, with components "pass", "readiness", "acceptance", "hash", "interpretation". It contains a conservative software-release evidence gate and associated metadata or diagnostics needed to interpret the result.

## eyeprocess_validation_seed

*Derive a deterministic bounded validation seed*

### Description

Derive a deterministic bounded validation seed

### Usage

```
eyeprocess_validation_seed(master_seed, index, stream = 0L)
```

### Arguments

| | |
|---|---|
| master_seed | Master random-number seed. |
| index | Deterministic substream index. |
| stream | Named random-number stream. |

## Value

A numeric value or vector containing a deterministic bounded validation seed.

---

`eye_analysis_pipeline` *Construct a governed analysis pipeline*

---

## Description

Construct a governed analysis pipeline

## Usage

```
eye_analysis_pipeline(
  ...,
  spec = eye_analysis_spec(),
  name = "eye_analysis",
  strict = TRUE
)
```

## Arguments

| | |
|---|---|
| `...` | ‘eye_pipeline_step‘ objects. |
| `spec` | Optional ‘eye_analysis_spec‘. |
| `name` | Pipeline label. |
| `strict` | If ‘TRUE‘, undeclared dependencies are errors. |

## Value

An object of class "eye_analysis_pipeline", stored as a named list, with components "name", "steps", "spec", "strict", "created_at", "status". It contains a governed analysis pipeline and associated metadata or diagnostics needed to interpret the result.

---

`eye_analysis_spec` *Define explicit analysis decisions for an eyeprocess workflow*

---

## Description

Define explicit analysis decisions for an eyeprocess workflow

### Usage

```
eye_analysis_spec(
  blink_correction = NULL,
  pupil_baseline = NULL,
  fixation_algorithm = NULL,
  aoi_rule = NULL,
  exclusions = NULL,
  model = NULL,
  sensitivity = NULL,
  ...
)
```

### Arguments

| | |
|---|---|
| blink_correction | Named blink-correction rule or 'NULL'. |
| pupil_baseline | Numeric baseline window or 'NULL'. |
| fixation_algorithm | Named fixation algorithm or 'NULL'. |
| aoi_rule | Named AOI assignment rule or 'NULL'. |
| exclusions | Explicit exclusion specification or 'NULL'. |
| model | Explicit model specification or 'NULL'. |
| sensitivity | Sensitivity specification or 'NULL'. |
| ... | Additional named decisions. |

### Value

An 'eye_analysis_spec' object.

---

## eye_api_inventory *Inventory the public eyeprocess API*

---

### Description

Inventory the public eyeprocess API

### Usage

```
eye_api_inventory(package = "eyeprocess", lifecycle = eye_api_lifecycle())
```

### Arguments

| | |
|---|---|
| package | Package name or namespace environment. |
| lifecycle | Lifecycle registry. Defaults to the packaged 0.9 registry from 'eye_api_lifecycle()'. |

### Value

Data frame of exported symbols and lifecycle metadata.

---

## eye_api_lifecycle *Normalize or create an API lifecycle registry*

---

### Description

Normalize or create an API lifecycle registry

### Usage

```
eye_api_lifecycle(registry = NULL)
```

### Arguments

| | |
|---|---|
| registry | Optional data frame with at least ‘name‘ and ‘status‘. ‘NULL‘ loads the packaged 0.9 lifecycle registry. |

### Value

‘eye_api_lifecycle‘ data frame.

---

## eye_api_recommendation *Lifecycle recommendation for API review*

---

### Description

Lifecycle recommendation for API review

### Usage

```
eye_api_recommendation(audit)
```

### Arguments

| | |
|---|---|
| audit | ‘eye_api_audit‘ object. |

### Value

A data frame containing lifecycle recommendation for API review. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## eye_api_status *Lookup lifecycle status for one or more APIs*

### Description

Lookup lifecycle status for one or more APIs

### Usage

```
eye_api_status(name, registry = eye_api_lifecycle())
```

### Arguments

| | |
|---|---|
| name | API names. |
| registry | Lifecycle registry. |

### Value

A data frame containing lookup lifecycle status for one or more APIs. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## eye_api_superseded *Return superseded/deprecated compatibility interfaces*

### Description

Return superseded/deprecated compatibility interfaces

### Usage

```
eye_api_superseded(registry = eye_api_lifecycle())
```

### Arguments

| | |
|---|---|
| registry | Lifecycle registry. |

### Value

A tabular R object containing return superseded/deprecated compatibility interfaces; rows represent analysis units and columns contain the returned quantities.

## eye_benchmark_design *Define a computational benchmark design*

### Description

Define a computational benchmark design

### Usage

```
eye_benchmark_design(
  n_obs = c(10000, 1e+05, 1e+06),
  repetitions = 3L,
  label = "eyeprocess_scaling"
)
```

### Arguments

| | |
|---|---|
| n_obs | Observation counts. |
| repetitions | Repetitions per size. |
| label | Benchmark label. |

### Value

A tabular R object containing define a computational benchmark design; rows represent analysis units and columns contain the returned quantities.

## eye_decision_manifest *Create a machine-readable research decision manifest*

### Description

Create a machine-readable research decision manifest

### Usage

```
eye_decision_manifest(
  sampling = list(),
  validity = list(),
  fixation = list(),
  pupil = list(),
  aoi = list(),
  model = list(),
  sensitivity = list(),
  exclusions = list(),
  provenance = list(),
```

```
  notes = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| sampling | Sampling decisions. |
| validity | Validity/missingness decisions. |
| fixation | Fixation construction decisions. |
| pupil | Pupil preprocessing decisions. |
| aoi | AOI assignment decisions. |
| model | Statistical/psychometric model decisions. |
| sensitivity | Sensitivity-analysis decisions. |
| exclusions | Exclusion rules. |
| provenance | Optional provenance fields. |
| notes | Notes. |
| ... | Additional named decision domains. |

## Value

An object of class "eye_decision_manifest", stored as a named list, with components "domains", "notes", "created_at", "schema_version", "status", "caveat". It contains a machine-readable research decision manifest and associated metadata or diagnostics needed to interpret the result.

---

# eye_pipeline_dot *Render pipeline dependencies as Graphviz DOT*

---

## Description

Render pipeline dependencies as Graphviz DOT

## Usage

```
eye_pipeline_dot(x)
```

## Arguments

| | |
|---|---|
| x | Pipeline. |

## Value

A character value or vector containing render pipeline dependencies as Graphviz DOT.

## eye_pipeline_graph *Return pipeline vertices and dependency edges*

### Description

Return pipeline vertices and dependency edges

### Usage

```
eye_pipeline_graph(x)
```

### Arguments

| | |
|---|---|
| x | Pipeline. |

### Value

A named list with components "vertices", "edges", containing return pipeline vertices and dependency edges and associated metadata or diagnostics.

## eye_pipeline_manifest *Machine-readable pipeline manifest*

### Description

Machine-readable pipeline manifest

### Usage

```
eye_pipeline_manifest(x)
```

### Arguments

| | |
|---|---|
| x | Pipeline. |

### Value

A data frame containing machine-readable pipeline manifest. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## eye_pipeline_mermaid *Render pipeline dependencies as Mermaid flowchart text*

### Description

Render pipeline dependencies as Mermaid flowchart text

### Usage

```
eye_pipeline_mermaid(x)
```

### Arguments

| | |
|---|---|
| x | Pipeline. |

### Value

A character value or vector containing render pipeline dependencies as Mermaid flowchart text.

## eye_pipeline_step *Define a governed pipeline step*

### Description

Define a governed pipeline step

### Usage

```
eye_pipeline_step(
  name,
  fun,
  requires = character(),
  optional = FALSE,
  description = NULL,
  decision = NULL
)
```

### Arguments

| | |
|---|---|
| name | Unique step name. |
| fun | Function executed by the step. |
| requires | Names of upstream steps. |
| optional | Whether an error may be retained without stopping the pipeline. |
| description | Human-readable description. |
| decision | Optional decision label linking the step to an analysis specification. |

## Value

An ‘eye_pipeline_step‘ object.

---

`eye_plot_spec` *Measurement-intelligence plotting and result infrastructure*

---

## Description

Measurement-intelligence plotting and result infrastructure. These functions form the eyeprocess 0.6.0.9000 measurement-intelligence programme and use dependency-free reference implementations with explicit evidence limits.

## Usage

```
eye_plot_spec(type = "default", title = NULL, xlab = NULL, ylab = NULL,
  caption = NULL, show_uncertainty = TRUE, show_raw = TRUE, facet_by = NULL,
  label_items = FALSE, interactive = FALSE)
plot_diagnostics(x, ...)
plot_evidence(x, ...)
plot_sensitivity(x, ...)
autoplot_eyeprocess(object, ...)
```

## Arguments

| | |
|---|---|
| `type` | Argument controlling ‘type‘; see the function usage and returned audit metadata. |
| `title` | Argument controlling ‘title‘; see the function usage and returned audit metadata. |
| `xlab` | Argument controlling ‘xlab‘; see the function usage and returned audit metadata. |
| `ylab` | Argument controlling ‘ylab‘; see the function usage and returned audit metadata. |
| `caption` | Argument controlling ‘caption‘; see the function usage and returned audit metadata. |
| `show_uncertainty` | Argument controlling ‘show_uncertainty‘; see the function usage and returned audit metadata. |
| `show_raw` | Argument controlling ‘show_raw‘; see the function usage and returned audit metadata. |
| `facet_by` | Argument controlling ‘facet_by‘; see the function usage and returned audit metadata. |
| `label_items` | Argument controlling ‘label_items‘; see the function usage and returned audit metadata. |
| `interactive` | Argument controlling ‘interactive‘; see the function usage and returned audit metadata. |
| `x` | Input object or data structure appropriate for the selected analysis. |
| `...` | Additional arguments passed to the underlying method or plotting function. |
| `object` | Input object or data structure appropriate for the selected analysis. |

## Details

The APIs return auditable S3 objects. Plot wrappers call registered base-graphics methods. Experimental or approximate engines are labelled in object status fields and should be validated before confirmatory or operational use.

## Value

An eyeprocess result object, data frame, model object, plot, or audit table as documented by the individual function.

## See Also

`plot_diagnostics()`, `plot_evidence()`, and `plot_sensitivity()`.

---

# eye_prov_graph *Construct a lightweight provenance graph*

---

## Description

Construct a lightweight provenance graph

## Usage

```
eye_prov_graph(
  nodes,
 edges = data.frame(from = character(), to = character(), relation = character()),
  metadata = list()
)
```

## Arguments

| | |
|---|---|
| `nodes` | Node table. |
| `edges` | Edge table. |
| `metadata` | Optional metadata. |

## Value

A named list with components "nodes", "edges", "metadata", containing a lightweight provenance graph and associated metadata or diagnostics.

`eye_reproducibility_fingerprint` *Construct a reproducibility fingerprint*

## Description

Construct a reproducibility fingerprint

## Usage

```
eye_reproducibility_fingerprint(
  data = NULL,
  analysis_spec = NULL,
  model_spec = NULL,
  decisions = NULL,
  result = NULL,
  files = NULL,
  seeds = NULL,
  label = "eyeprocess_analysis"
)
```

## Arguments

| | |
|---|---|
| `data` | Input data or hashable object. |
| `analysis_spec` | Analysis specification. |
| `model_spec` | Model specification. |
| `decisions` | Decision manifest. |
| `result` | Optional result object. |
| `files` | Optional input file paths. |
| `seeds` | Optional seeds. |
| `label` | Fingerprint label. |

## Value

A named list with components "schema_version", "label", "eyeprocess_version", "data_hash", "analysis_spec_hash", "model_spec_hash", "decisions_hash", "result_hash", "file_manifest", "seeds", "environment", containing a reproducibility fingerprint and associated metadata or diagnostics.

## eye_session_manifest *Create a session-level provenance manifest*

### Description

Create a session-level provenance manifest

### Usage

```
eye_session_manifest(
  data = NULL,
  files = NULL,
  adapter = NA_character_,
  decisions = NULL,
  pipeline = NULL,
  seeds = NULL,
  notes = NULL
)
```

### Arguments

| | |
|---|---|
| data | Optional data object. |
| files | Optional input file paths. |
| adapter | Adapter/importer identifier. |
| decisions | Optional decision manifest. |
| pipeline | Optional pipeline or run object. |
| seeds | Optional named random seeds. |
| notes | Optional notes. |

### Value

A named list with components "created_utc", "eyeprocess_version", "data_hash", "files", "adapter", "decisions_hash", "pipeline_hash", "seeds", "environment", "notes", containing a session-level provenance manifest and associated metadata or diagnostics.

## eye_storage_spec *Specify disk-backed eyeprocess storage*

### Description

Specify disk-backed eyeprocess storage

## Usage

```
eye_storage_spec(
  path,
  format = c("rds", "parquet", "arrow_dataset"),
  tables = canonical_table_names(),
  partitioning = NULL,
  compression = "zstd"
)
```

## Arguments

| | |
|---|---|
| path | Storage directory or RDS file. |
| format | One of ‘"rds"‘, ‘"parquet"‘, or ‘"arrow_dataset"‘. |
| tables | Canonical tables to store. |
| partitioning | Optional Arrow partition columns. |
| compression | Parquet compression codec. For writes, the default ‘"zstd"‘ is preferred; when the argument is omitted and that codec is unavailable, storage falls back to ‘"snappy"‘ and then ‘"uncompressed"‘. An explicitly requested unavailable codec errors. |

## Value

An ‘eye_storage_spec‘ object.

---

# eye_stream_fidelity_audit

*Audit preservation of monocular/binocular stream semantics*

---

## Description

Audit preservation of monocular/binocular stream semantics

## Usage

```
eye_stream_fidelity_audit(
  source,
  roundtrip,
  source_eye = "eye",
  roundtrip_eye = source_eye,
  key = NULL
)
```

### Arguments

| | |
|---|---|
| source | Original/source representation. |
| roundtrip | Round-tripped or comparison representation. |
| source_eye | Source recorded-eye column. |
| roundtrip_eye | Round-tripped recorded-eye column. |
| key | Column or columns used to align records. |

### Value

An object of class "eye_stream_fidelity", stored as a named list, with components "status", "matched_n", "source_streams", "roundtrip_streams", "confusion". It contains preservation of monocular/binocular stream semantics and associated metadata or diagnostics needed to interpret the result.

---

## eye_targets_manifest *Create a targets-compatible dependency manifest*

---

### Description

This function does not execute or silently translate arbitrary closures into a targets pipeline. It provides the dependency contract required to build an explicit '_targets.R' file.

### Usage

```
eye_targets_manifest(x)
```

### Arguments

| | |
|---|---|
| x | Pipeline. |

### Value

A data frame containing a targets-compatible dependency manifest. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## facet_effects *Extract facet effects from a many-facet process model*

### Description

Extract facet effects from a many-facet process model

### Usage

```
facet_effects(object, channel = c("response", "process"))
```

### Arguments

| | |
|---|---|
| object | A fitted eyeprocess model or audit object. |
| channel | Measurement channel to inspect. |

### Value

A named list with components "random_effects", "variance_components", containing facet effects from a many-facet process model and associated metadata or diagnostics.

## field_fidelity_report *Field-level semantic fidelity report*

### Description

Classifies canonical fields after a semantic round trip. Character fields are compared exactly after NA preservation; numeric fields are compared directly and also tested for a stable affine transform.

### Usage

```
field_fidelity_report(
  source,
  roundtrip,
  fields = NULL,
  mapping = NULL,
  key = NULL,
  tolerance = 1e-08,
  spec = semantic_fidelity_spec()
)
```

### Arguments

| | |
|---|---|
| source | Original canonical data. |
| roundtrip | Canonical data reconstructed after an interchange round trip. |
| fields | Fields to compare. Defaults to common fields. |
| mapping | Optional named character vector mapping source field names to round-trip field names. |
| key | Optional unique row-alignment fields. |
| tolerance | Numeric equality tolerance. |
| spec | A ‘semantic_fidelity_spec()‘ object. |

### Value

An object of class ‘eye_field_fidelity_report‘.

---

## file_hash_manifest *Build a file hash manifest*

---

### Description

Build a file hash manifest

### Usage

```
file_hash_manifest(paths, algorithm = c("md5", "sha256"))
```

### Arguments

| | |
|---|---|
| paths | File paths. |
| algorithm | Hash algorithm; currently ‘md5‘ uses base R, ‘sha256‘ uses openssl when available. |

### Value

A data frame containing a file hash manifest. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## filter_eye_signal *Filter a one-dimensional eye signal robustly*

### Description

Filter a one-dimensional eye signal robustly

### Usage

```
filter_eye_signal(
  signal,
  width = 9L,
  method = c("auto", "robfilter", "runmed"),
  online = TRUE
)
```

### Arguments

| | |
|---|---|
| signal | Numeric signal. |
| width | Median-filter width. |
| method | 'auto', 'robfilter', or 'runmed'. |
| online | Passed to 'robfilter::med.filter()' when used. |

### Value

An 'eye_signal_filter_audit' object.

## filter_pupil_signal *Filter pupil signal robustly*

### Description

Filter pupil signal robustly

### Usage

```
filter_pupil_signal(...)
```

### Arguments

| | |
|---|---|
| ... | Additional arguments passed to the underlying method or helper. |

### Value

An R object containing filter pupil signal robustly. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## find_process_measures *Find process measures by channel, level, status, or text*

### Description

Find process measures by channel, level, status, or text

### Usage

```
find_process_measures(
  registry = process_measure_registry(),
  channel = NULL,
  level = NULL,
  status = NULL,
  query = NULL
)
```

### Arguments

| | |
|---|---|
| registry | Registry. |
| channel | Optional channel filter. |
| level | Optional level pattern. |
| status | Optional status filter. |
| query | Optional text query. |

### Value

A tabular R object containing find process measures by channel, level, status, or text; rows represent analysis units and columns contain the returned quantities.

## fingerprint_validation_case *Fingerprint every file in a validation case*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
fingerprint_validation_case(path, algorithms = c("md5", "sha256"),
  include_hidden = FALSE)
```

### Arguments

| | |
|---|---|
| path | File or directory. |
| algorithms | Hash algorithms. Base R always supplies MD5; SHA-256 is |
| include_hidden | Include hidden files. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

## fit_aoi_growth_curve *Fit a single AOI growth curve*

---

### Description

Fit a single AOI growth curve

### Usage

```
fit_aoi_growth_curve(data, time, outcome, degree = 3L)
```

### Arguments

| | |
|---|---|
| data | Data frame. |
| time | Time column. |
| outcome | Numeric AOI proportion/indicator column. |
| degree | Polynomial degree. |

### Value

An object of class "eye_aoi_growth_curve", stored as a named list, with components "model", "time", "outcome", "degree", "poly", "range", "status". It contains a single AOI growth curve and associated metadata or diagnostics needed to interpret the result.

## fit_brms_adapter *fit brms adapter*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
fit_brms_adapter(formula, data, purpose, ...)
```

### Arguments

| | |
|---|---|
| formula | Value for 'formula'. See the function description and relevant article for constraints. |
| data | Value for 'data'. See the function description and relevant article for constraints. |
| purpose | Value for 'purpose'. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## fit_censored_normal_process_irt *Conditional censored-normal calibration for bounded process measurements*

### Description

Fits the 2026 censored-normal response form item-by-item conditional on a supplied latent score. This is a useful calibration/diagnostic engine for bounded continuous process variables such as AOI proportions. It is NOT the paper's full marginal EM estimator and should therefore remain experimental.

### Usage

```
fit_censored_normal_process_irt(
  response_matrix,
  theta,
  lower = 0,
  upper = 1,
  control = list(maxit = 1000)
)
```

### Arguments

| | |
|---|---|
| response_matrix | Person x item bounded continuous matrix. |
| theta | Supplied person latent scores on the calibration scale. |
| lower, upper | Observable bounds. |
| control | 'optim()' control list. |

### Value

An object of class "eye_censored_normal_process_irt", stored as a named list, with components "coefficients", "fits", "theta", "lower", "upper", "engine", "status", "citation", "caveat". It contains conditional censored-normal calibration for bounded process measurements and associated metadata or diagnostics needed to interpret the result.

---

## fit_changepoint_multimodal_irt

*Fit a multimodal change-point IRT workflow*

---

### Description

Fit a multimodal change-point IRT workflow

### Usage

```
fit_changepoint_multimodal_irt(
  data,
  ...,
  gaze = "fixation_count",
  refit = TRUE
)
```

### Arguments

| | |
|---|---|
| data | Input data frame or compatible tabular object. |
| ... | Additional arguments passed to the selected model, engine, or method. |
| gaze | Gaze/process variable or column name. |
| refit | Whether the model is refitted after segmentation. |

### Value

An object of class "eye_changepoint_multimodal_irt", stored as a named list, with components "changepoints", "refit_requested", "status". It contains a multimodal change-point IRT workflow and associated metadata or diagnostics needed to interpret the result.

## fit_changepoint_rt_irt

*Fit a change-point RT IRT workflow*

### Description

Fit a change-point RT IRT workflow

### Usage

```
fit_changepoint_rt_irt(data, ..., refit = TRUE)
```

### Arguments

| | |
|---|---|
| data | Input data frame or compatible tabular object. |
| ... | Additional arguments passed to the selected model, engine, or method. |
| refit | Whether the model is refitted after segmentation. |

### Value

An object of class "eye_changepoint_rt_irt", stored as a named list, with components "changepoints", "refit_requested", "status". It contains a change-point RT IRT workflow and associated metadata or diagnostics needed to interpret the result.

## fit_cognitive_diagnosis_process

*Cognitive-diagnosis model with process indicators*

### Description

Uses GDINA for the response layer when available and retains process features as a parallel diagnostic channel. Process/mastery associations are reported descriptively and do not redefine the Q-matrix or skill labels.

### Usage

```
fit_cognitive_diagnosis_process(
  response_matrix,
  q_matrix,
  process_data = NULL,
  process_features = NULL,
  person_id = NULL,
  engine = c("GDINA", "external"),
  external_engine = NULL,
  ...
)
```

### Arguments

| | |
|---|---|
| response_matrix | Person-by-item response matrix. |
| q_matrix | Q-matrix for cognitive-diagnosis modeling. |
| process_data | Process-data input used by the model. |
| process_features | Names of process-derived features. |
| person_id | Person or participant identifier. |
| engine | Estimation engine. |
| external_engine | Validated external fitting function. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

An object of class "eye_cognitive_diagnosis_process", stored as a named list, with components "response_model", "process_summary", "process_mastery_correlation", "q_matrix", "status". It contains cognitive-diagnosis model with process indicators and associated metadata or diagnostics needed to interpret the result.

---

## fit_continuous_time_irt

*Continuous-time IRT external-engine gate*

---

### Description

Continuous-time IRT external-engine gate

### Usage

```
fit_continuous_time_irt(data, external_engine = NULL, ...)
```

### Arguments

| | |
|---|---|
| data | Input data frame or compatible tabular object. |
| external_engine | Validated external fitting function. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

An object of class "eye_continuous_time_irt", stored as a named list, with components "model", "status". It contains continuous-time IRT external-engine gate and associated metadata or diagnostics needed to interpret the result.

`fit_crossclassified_process_irt` *Cross-classified process IRT reference model*

## Description

Treats the process outcome as repeated evidence crossed by person and item, with optional contextual grouping factors. This is useful when process events themselves, not only item summaries, are the observations.

## Usage

```
fit_crossclassified_process_irt(
  data,
  outcome,
  person = "participant_id",
  item = "item_id",
  context = NULL,
  family = c("gaussian", "binomial", "poisson", "negative_binomial"),
  fixed = NULL
)
```

## Arguments

| | |
|---|---|
| `data` | Input data frame or compatible tabular object. |
| `outcome` | Outcome variable. |
| `person` | Person or participant identifier column. |
| `item` | Item identifier, name, or item column. |
| `context` | Context or grouping variable. |
| `family` | Statistical family used by the channel or model. |
| `fixed` | Fixed-effects specification. |

## Value

An object of class "eye_crossclassified_process_irt", stored as a named list, with components "model", "family", "person", "item", "context", "status". It contains cross-classified process IRT reference model and associated metadata or diagnostics needed to interpret the result.

## fit_crossclassified_process_irt_mhrm *Create a gated scalable cross-classified MH-RM process IRT interface*

### Description

Create a gated scalable cross-classified MH-RM process IRT interface

### Usage

```
fit_crossclassified_process_irt_mhrm(data, engine = NULL, ...)
```

### Arguments

| | |
|---|---|
| data | Data supplied to an optional external engine. |
| engine | Optional estimator implementing the intended scalable MH-RM model. |
| ... | Passed to engine. |

### Value

An object of class "eye_gated_process_model", stored as a named list, with components "id", "purpose", "required_evidence", "engine", "fit", "status", "notes", "caveat". It contains a gated scalable cross-classified MH-RM process IRT interface and associated metadata or diagnostics needed to interpret the result.

## fit_device_linking *Cross-device and cross-vendor metric linking*

### Description

Cross-device and cross-vendor metric linking. These functions form the eyeprocess 0.6.0.9000 measurement-intelligence programme and use dependency-free reference implementations with explicit evidence limits.

### Usage

```
fit_device_linking(x, metric, reference_device, method = c("mixed_bland_altman",
  "hierarchical", "equipercentile"), device_col = "device", id_cols = c("person_id",
  "item_id"))
apply_device_linking(x, linking_model, metric = NULL, device_col = NULL,
  output_col = NULL)
audit_device_equivalence(x, equivalence_margin, by = c("metric", "task", "aoi"))
estimate_device_specific_error(x)
plot_device_agreement(x, ...)
plot_device_bias_by_magnitude(x, ...)
```

```
plot_device_transfer_curve(x, ...)
plot_device_equivalence_intervals(x, ...)
plot_cross_vendor_metric_matrix(x, ...)
```

## Arguments

| | |
|---|---|
| x | Input object or data structure appropriate for the selected analysis. |
| metric | Argument controlling 'metric'; see the function usage and returned audit metadata. |
| reference_device | Argument controlling 'reference_device'; see the function usage and returned audit metadata. |
| method | Argument controlling 'method'; see the function usage and returned audit metadata. |
| device_col | Argument controlling 'device_col'; see the function usage and returned audit metadata. |
| id_cols | Argument controlling 'id_cols'; see the function usage and returned audit metadata. |
| linking_model | Argument controlling 'linking_model'; see the function usage and returned audit metadata. |
| output_col | Argument controlling 'output_col'; see the function usage and returned audit metadata. |
| equivalence_margin | Argument controlling 'equivalence_margin'; see the function usage and returned audit metadata. |
| by | Argument controlling 'by'; see the function usage and returned audit metadata. |
| ... | Additional arguments passed to the underlying method or plotting function. |

## Details

The APIs return auditable S3 objects. Plot wrappers call registered base-graphics methods. Experimental or approximate engines are labelled in object status fields and should be validated before confirmatory or operational use.

## Value

An eyeprocess result object, data frame, model object, plot, or audit table as documented by the individual function.

## See Also

plot_diagnostics(), plot_evidence(), and plot_sensitivity().

## fit_diffirt_adapter *Fit a diffusion IRT adapter*

### Description

Fit a diffusion IRT adapter

### Usage

```
fit_diffirt_adapter(x, model = c("D", "Q"), ...)
```

### Arguments

| | |
|---|---|
| x | An ‘eye_dataset‘. |
| model | Diffusion IRT model, ‘"D"‘ or ‘"Q"‘. |
| ... | Passed to ‘diffIRT::diffIRT()‘. |

### Value

An ‘eyeprocess_model‘.

## fit_diffirt_engine_adapter *fit diffirt engine adapter*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
fit_diffirt_engine_adapter(data, purpose, ...)
```

### Arguments

| | |
|---|---|
| data | Value for ‘data‘. See the function description and relevant article for constraints. |
| purpose | Value for ‘purpose‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## fit_dynamic_gpirt *Dynamic GPIRT external-engine gate*

### Description

Dynamic GPIRT external-engine gate

### Usage

```
fit_dynamic_gpirt(data, external_engine = NULL, ...)
```

### Arguments

| | |
|---|---|
| data | Input data frame or compatible tabular object. |
| external_engine | Validated external fitting function. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

An object of class "eye_dynamic_gpirt", stored as a named list, with components "model", "engine", "status". It contains dynamic GPIRT external-engine gate and associated metadata or diagnostics needed to interpret the result.

## fit_dynamic_irtree *Fit a dynamic gaze-state response-tree model*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
fit_dynamic_irtree(x, spec = dynamic_irtree_spec(), min_transitions = 10L, seed = 1L,
  ...)
```

### Arguments

| | |
|---|---|
| x | An ‘eye_dataset‘ or transition/long-state data frame. |
| spec | Dynamic IRTree specification. |
| min_transitions | Minimum support per destination for baseline logits. |
| seed | Random seed for probabilistic engines. |
| ... | Engine-specific arguments. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`fit_dynamic_irtree_stan` *Fit an optional CmdStan dynamic-transition model*

---

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
fit_dynamic_irtree_stan(design, spec, seed = 1L, refresh = 0L, output_dir = NULL, ...)
```

## Arguments

| | |
|---|---|
| `design` | Transition design. |
| `spec` | Dynamic IRTree specification. |
| `seed` | Random seed. |
| `refresh` | CmdStan refresh interval. |
| `output_dir` | Optional CmdStan output directory. |
| `...` | Additional arguments to 'CmdStanModel$sample()'. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`fit_event_time_irt` *Fit an event-time IRT reference workflow*

---

## Description

The built-in Cox reference conditions on an already supplied theta and is therefore an event-time *measurement diagnostic*, not a full continuous-time latent-trait estimator. A validated exact implementation can be supplied via 'external_engine'.

## Usage

```
fit_event_time_irt(
  data,
  event_time = "event_time",
  event = "event",
  theta = "theta",
  person = "participant_id",
  item = "item_id",
  engine = c("cox_reference", "external"),
  external_engine = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| data | Long event/item data. |
| event_time | Time-to-event column. |
| event | Event indicator (1 event, 0 censored). |
| theta | Supplied latent-trait column for the reference engine. |
| person, item | Person and item identifiers. |
| engine | 'cox_reference' or 'external'. |
| external_engine | Function implementing a study-specific event-time IRT. |
| ... | Additional arguments passed to the external engine. |

## Value

An object of class "eye_event_time_irt", stored as a named list, with components "model", "engine", "theta_conditioned", "status", "note". It contains an event-time IRT reference workflow and associated metadata or diagnostics needed to interpret the result.

---

# fit_external_engine *Fit an external model engine through a stable adapter*

---

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
fit_external_engine(engine, data, specification = NULL, purpose, ...)
```

### Arguments

| | |
|---|---|
| engine | Engine name. |
| data | Engine-ready data. |
| specification | Engine-specific specification. |
| purpose | Declared scientific purpose. |
| ... | Engine arguments. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

## fit_eyeprocess_erm *Fit an eRm Rasch-family model without fallback substitution*

---

### Description

Fit an eRm Rasch-family model without fallback substitution

### Usage

```
fit_eyeprocess_erm(data, model = c("RM", "PCM"), ..., engine = "eRm")
```

### Arguments

| | |
|---|---|
| data | Input data frame, matrix, or compatible analysis object. |
| model | Model specification passed to the selected external engine. |
| ... | Additional arguments passed to the selected method or external engine. |
| engine | Requested estimation or analysis engine. |

### Value

An object of class "eye_external_irt_fit", stored as a named list, with components "status", "engine", "fit", "call". It contains an eRm Rasch-family model without fallback substitution and associated metadata or diagnostics needed to interpret the result.

## fit_eyeprocess_gdina *Fit a G-DINA cognitive-diagnosis model without fallback substitution*

### Description

Fit a G-DINA cognitive-diagnosis model without fallback substitution

### Usage

```
fit_eyeprocess_gdina(dat, Q, model = "GDINA", ..., engine = "GDINA")
```

### Arguments

| | |
|---|---|
| dat | Response data supplied to the GDINA engine. |
| Q | Binary item-by-attribute Q-matrix. |
| model | Model specification passed to the selected external engine. |
| ... | Additional arguments passed to the selected method or external engine. |
| engine | Requested estimation or analysis engine. |

### Value

An object of class "eye_external_irt_fit", stored as a named list, with components "status", "engine", "fit", "call". It contains a G-DINA cognitive-diagnosis model without fallback substitution and associated metadata or diagnostics needed to interpret the result.

## fit_eyeprocess_lnirt *Fit a joint response/response-time LNIRT model without fallback substitution*

### Description

Fit a joint response/response-time LNIRT model without fallback substitution

### Usage

```
fit_eyeprocess_lnirt(Y, RT, quadratic = FALSE, ..., engine = "LNIRT")
```

### Arguments

| | |
|---|---|
| Y | Item-response matrix supplied to LNIRT. |
| RT | Response-time matrix supplied to LNIRT. |
| quadratic | Whether the LNIRT quadratic option is requested. |
| ... | Additional arguments passed to the selected method or external engine. |
| engine | Requested estimation or analysis engine. |

### Value

An object of class "eye_external_irt_fit", stored as a named list, with components "status", "engine", "fit", "call", "rt_scale". It contains a joint response/response-time LNIRT model without fallback substitution and associated metadata or diagnostics needed to interpret the result.

---

## fit_eyeprocess_mirt *Fit a model with mirt without substituting another estimator*

---

### Description

Fit a model with mirt without substituting another estimator

### Usage

```
fit_eyeprocess_mirt(data, model = 1, itemtype = "2PL", ..., engine = "mirt")
```

### Arguments

| | |
|---|---|
| data | Input data frame, matrix, or compatible analysis object. |
| model | Model specification passed to the selected external engine. |
| itemtype | Item type specification for mirt. |
| ... | Additional arguments passed to the selected method or external engine. |
| engine | Requested estimation or analysis engine. |

### Value

An object of class "eye_external_irt_fit", stored as a named list, with components "status", "engine", "fit", "call". It contains a model with mirt without substituting another estimator and associated metadata or diagnostics needed to interpret the result.

---

## fit_eyeprocess_tam *Fit a TAM model without substituting another estimator*

---

### Description

Fit a TAM model without substituting another estimator

### Usage

```
fit_eyeprocess_tam(
  resp,
  model = c("rasch", "2pl", "gpcm"),
  ...,
  engine = "TAM"
)
```

### Arguments

| | |
|---|---|
| resp | Response matrix supplied to TAM. |
| model | Model specification passed to the selected external engine. |
| ... | Additional arguments passed to the selected method or external engine. |
| engine | Requested estimation or analysis engine. |

### Value

An object of class "eye_external_irt_fit", stored as a named list, with components "status", "engine", "fit", "call". It contains a TAM model without substituting another estimator and associated metadata or diagnostics needed to interpret the result.

---

## fit_eyetrackingr_adapter

*fit eyetrackingr adapter*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
fit_eyetrackingr_adapter(data, purpose, ...)
```

### Arguments

| | |
|---|---|
| data | Value for ‘data‘. See the function description and relevant article for constraints. |
| purpose | Value for ‘purpose‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

`fit_fixation_point_process` *Spatio-temporal fixation point-process models*

## Description

Spatio-temporal fixation point-process models. These functions form the eyeprocess 0.6.0.9000 measurement-intelligence programme and use dependency-free reference implementations with explicit evidence limits.

## Usage

```
fit_fixation_point_process(x, spatial_covariates = NULL, temporal_covariates = NULL,
  interaction = c("none", "self_exciting"), x_col = "x", y_col = "y",
  time_col = "time", grid_size = 20)
fit_marked_gaze_process(x, marks = c("duration", "pupil", "saccade_amplitude"),
  x_col = "x", y_col = "y", time_col = "time")
predict_fixation_intensity(model, new_stimulus = NULL)
diagnose_gaze_point_process(model)
plot_fixation_intensity(x, ...)
plot_spatial_residuals(x, ...)
plot_temporal_excitation_kernel(x, ...)
plot_covariate_effect_surface(x, ...)
plot_observed_expected_fixations(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Input object or data structure appropriate for the selected analysis. |
| `spatial_covariates` | Argument controlling ‘spatial_covariates‘; see the function usage and returned audit metadata. |
| `temporal_covariates` | Argument controlling ‘temporal_covariates‘; see the function usage and returned audit metadata. |
| `interaction` | Argument controlling ‘interaction‘; see the function usage and returned audit metadata. |
| `x_col` | Argument controlling ‘x_col‘; see the function usage and returned audit metadata. |
| `y_col` | Argument controlling ‘y_col‘; see the function usage and returned audit metadata. |
| `time_col` | Argument controlling ‘time_col‘; see the function usage and returned audit metadata. |
| `grid_size` | Argument controlling ‘grid_size‘; see the function usage and returned audit metadata. |

| | |
|---|---|
| marks | Argument controlling ‘marks‘; see the function usage and returned audit metadata. |
| model | Argument controlling ‘model‘; see the function usage and returned audit metadata. |
| new_stimulus | Argument controlling ‘new_stimulus‘; see the function usage and returned audit metadata. |
| ... | Additional arguments passed to the underlying method or plotting function. |

## Details

The APIs return auditable S3 objects. Plot wrappers call registered base-graphics methods. Experimental or approximate engines are labelled in object status fields and should be validated before confirmatory or operational use.

## Value

An eyeprocess result object, data frame, model object, plot, or audit table as documented by the individual function.

## See Also

`plot_diagnostics()`, `plot_evidence()`, and `plot_sensitivity()`.

---

# fit_flow_mirt *Flow-MIRT external-engine gate*

---

## Description

Flow-MIRT external-engine gate

## Usage

```
fit_flow_mirt(response_matrix, external_engine = NULL, ...)
```

## Arguments

| | |
|---|---|
| response_matrix | Person-by-item response matrix. |
| external_engine | Validated external fitting function. |
| ... | Additional arguments passed to the selected model, engine, or method. |

## Value

An object of class "eye_flow_mirt", stored as a named list, with components "model", "status", "engine". It contains flow-MIRT external-engine gate and associated metadata or diagnostics needed to interpret the result.

## fit_functional_pupil_stan

*Fit the bundled joint functional pupil-IRT Stan model*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
fit_functional_pupil_stan(prepared, basis_matrix = NULL, seed = 1L, refresh = 0L,
  output_dir = NULL, ...)
```

### Arguments

| | |
|---|---|
| prepared | Prepared functional pupil data. |
| basis_matrix | Functional basis matrix. |
| seed | Random seed. |
| refresh | CmdStan refresh interval. |
| output_dir | Optional CmdStan output directory. |
| ... | Additional sampling arguments. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## fit_gaze_anchored_3pl_audit

*Fit a standard 3PL and audit lower-asymptote alignment with process evidence*

### Description

Fits a conventional 3PL response model with ‘mirt‘, then aligns the fitted lower-asymptote parameter with item-level gaze/pupil/RT summaries. The alignment is descriptive and diagnostic. It must not be interpreted as a confirmatory detector of guessing, rapid responding, disengagement, or any other latent behavior without independent validation.

### Usage

```
fit_gaze_anchored_3pl_audit(
  response_matrix,
  process_data = NULL,
  item = "item_id",
  process_features = c("ttff_ms", "dwell_ms", "pupil_bc", "pupil_peak", "rt_ms",
    "accuracy"),
  model = 1,
  SE = FALSE
)
```

### Arguments

| | |
|---|---|
| response_matrix | Person x item dichotomous response matrix. |
| process_data | Optional trial/person-item process table. |
| item | Item identifier in 'process_data'. |
| process_features | Candidate numeric process features. |
| model | mirt model specification. |
| SE | Request standard errors from 'mirt'. |

### Value

An 'eye_gaze_anchored_3pl_audit' object.

---

## fit_gaze_diffusion_irt

*Fit a gaze-informed diffusion model*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
fit_gaze_diffusion_irt(data, spec = gaze_diffusion_spec(), seed = 1L, ...)
```

### Arguments

| | |
|---|---|
| data | Trial-level data. |
| spec | Diffusion specification. |
| seed | Seed. |
| ... | Engine arguments. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`fit_gaze_diffusion_stan`

*Fit the CmdStan Wiener diffusion model*

---

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
fit_gaze_diffusion_stan(prepared, seed = 1L, refresh = 0L, output_dir = NULL, ...)
```

## Arguments

| | |
|---|---|
| `prepared` | Prepared data. |
| `seed` | Seed. |
| `refresh` | CmdStan refresh interval. |
| `output_dir` | Optional output directory. |
| `...` | Additional sampling arguments. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`fit_gaze_informed_missingness_irt`

*Fit a gaze-informed missingness IRT diagnostic*

---

## Description

Fits a transparent two-part reference model: (1) whether an item response is missing and (2) the observed response, both conditional on a supplied latent trait (or a clearly labelled person-score proxy), item, and visual exposure. This is a diagnostic bridge to joint MNAR/process IRT, not a substitute for a fully joint latent missingness model.

### Usage

```
fit_gaze_informed_missingness_irt(
  data,
  response = "response",
  person = "participant_id",
  item = "item_id",
  gaze_exposure = "gaze_exposure",
  theta = NULL,
  reached = NULL
)
```

### Arguments

| | |
|---|---|
| data | Long person-item data. |
| response | Response column; missing values identify omissions. |
| person, item | Person and item identifiers. |
| gaze_exposure | Non-negative visual-exposure measure. |
| theta | Optional latent-trait column. If 'NULL', a smoothed person proportion-correct logit is used as an explicit proxy. |
| reached | Optional reached/not-reached indicator. |

### Value

An 'eye_gaze_informed_missingness_irt' object.

---

## fit_gdina_adapter *fit gdina adapter*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
fit_gdina_adapter(data, Q, model = "GDINA", purpose = "cognitive diagnosis", ...)
```

### Arguments

| | |
|---|---|
| data | Value for 'data'. See the function description and relevant article for constraints. |
| Q | Q-matrix with one row per item. |
| model | GDINA model specification used for eye-dataset inputs. |
| purpose | Declared scientific purpose for the external-engine contract. |
| ... | Additional arguments passed to the selected engine or method. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`fit_gpirt` *GPIRT model-criticism interface*

---

## Description

'external' is the only exact GPIRT path. 'spline_reference' fits flexible logistic spline IRFs solely as a nonparametric stress test for conventional logistic IRF shape; it is deliberately not described as a Gaussian process.

## Usage

```
fit_gpirt(
  response_matrix,
  engine = c("spline_reference", "external"),
  external_engine = NULL,
  spline_df = 5L,
  ...
)
```

## Arguments

| | |
|---|---|
| `response_matrix` | Person-by-item response matrix. |
| `engine` | Estimation engine. |
| `external_engine` | Validated external fitting function. |
| `spline_df` | Degrees of freedom for the spline reference model. |
| `...` | Additional arguments passed to the selected model, engine, or method. |

## Value

An object of class "eye_gpirt", stored as a named list, with components "response_matrix", "models", "theta_proxy", "engine", "exact_gpirt", "status", "note". It contains gPIRT model-criticism interface and associated metadata or diagnostics needed to interpret the result.

## fit_irt_model *Fit a registered multimodal IRT model*

### Description

Fit a registered multimodal IRT model

### Usage

```
fit_irt_model(spec, data, ..., allow_experimental = FALSE)
```

### Arguments

| | |
|---|---|
| spec | IRT model or validation specification. |
| data | Input data frame or compatible tabular object. |
| ... | Additional arguments passed to the selected model, engine, or method. |
| allow_experimental | Whether experimental models are permitted. |

### Value

An R object containing a registered multimodal IRT model. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## fit_item_parameter_seed_model *Fit an experimental pre-pilot item-parameter seeding model*

### Description

Estimates screening predictions for item difficulty/discrimination from item design/process features. Predictions are not calibrated operational parameters.

### Usage

```
fit_item_parameter_seed_model(
  item_data,
  difficulty = "irt_difficulty",
  discrimination = "irt_discrimination",
  predictors,
  engine = c("auto", "ranger", "lm"),
  seed = 2221
)
```

### Arguments

| | |
|---|---|
| item_data | Calibrated item-level training data. |
| difficulty, discrimination | Target columns. |
| predictors | Design/process predictors. |
| engine | 'auto', 'ranger', or 'lm'. |
| seed | Seed. |

### Value

An object of class "eye_item_parameter_seed", stored as a named list, with components "difficulty_model", "discrimination_model", "difficulty", "discrimination", "predictors", "engine", "training_data", "status", "caveat". It contains an experimental pre-pilot item-parameter seeding model and associated metadata or diagnostics needed to interpret the result.

---

## fit_joint_functional_pupil_irt

*Fit a functional pupil-informed IRT workflow*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
fit_joint_functional_pupil_irt(x, spec = functional_pupil_irt_spec(), seed = 1L, ...)
```

### Arguments

| | |
|---|---|
| x | Eye dataset or long pupil data. |
| spec | Functional pupil specification. |
| seed | Random seed. |
| ... | Engine-specific arguments. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

`fit_joint_gaze_rt_irt` *Joint response, response-time, and gaze-process IRT*

## Description

The reference engine fits crossed person/item submodels for accuracy, log-response-time, and a gaze process, then returns person- and item-side latent-score covariance summaries. The ‘brms‘ engine uses multivariate formulas with shared group-level IDs so person/item random effects can be correlated across channels.

## Usage

```
fit_joint_gaze_rt_irt(
  data,
  response = "response",
  rt = "rt",
  gaze = "fixation_count",
  person = "participant_id",
  item = "item_id",
  gaze_family = c("negative_binomial", "poisson"),
  engine = c("reference", "brms"),
  iter = 2000,
  chains = 4,
  cores = 1,
  seed = 1,
  ...
)
```

## Arguments

| | |
|---|---|
| `data` | Long person-by-item data. |
| `response` | Binary response variable. |
| `rt` | Positive response-time variable. |
| `gaze` | Gaze process variable, usually fixation count or dwell count. |
| `person`, `item` | Person and item identifiers. |
| `gaze_family` | Poisson or negative-binomial reference channel. |
| `engine` | ‘reference‘ or ‘brms‘. |
| `iter`, `chains`, `cores` | Passed to brms. |
| `seed` | Random seed. |
| `...` | Additional arguments to the selected engine. |

## Value

An ‘eye_joint_gaze_rt_irt‘ object.

`fit_joint_graded_rt_process_irt` *Joint graded-response, RT, and process reference model*

## Description

Extends the 2026 graded-response/RT direction with an optional gaze/process channel. The bundled reference engine uses proportional-odds plus crossed RT and process submodels; it is explicitly experimental rather than a claim to reproduce the published SAEM estimator.

## Usage

```
fit_joint_graded_rt_process_irt(
  data,
  response = "response",
  rt = "rt",
  process = "fixation_count",
  person = "participant_id",
  item = "item_id",
  engine = c("reference", "brms"),
  process_family = c("negative_binomial", "poisson", "gaussian"),
  iter = 2000,
  chains = 4,
  cores = 1,
  seed = 1,
  ...
)
```

## Arguments

| | |
|---|---|
| `data` | Input data frame or compatible tabular object. |
| `response` | Response variable or response-column name. |
| `rt` | Response-time variable or column name. |
| `process` | Process variable or column name. |
| `person` | Person or participant identifier column. |
| `item` | Item identifier, name, or item column. |
| `engine` | Estimation engine. |
| `process_family` | Distributional family for the process channel. |
| `iter` | Number of estimation iterations. |
| `chains` | Number of Bayesian chains. |
| `cores` | Number of processor cores. |
| `seed` | Random-number seed. |
| `...` | Additional arguments passed to the selected model, engine, or method. |

## Value

An object of class "eye_joint_graded_rt_process_irt", stored as a named list, with components "engine", "response_model", "rt_model", "process_model", "data_n", "status", "note". It contains joint graded-response, RT, and process reference model and associated metadata or diagnostics needed to interpret the result.

---

`fit_kde_latent_distribution_irt`

*Create a gated KDE latent-distribution IRT interface*

---

## Description

Create a gated KDE latent-distribution IRT interface

## Usage

```
fit_kde_latent_distribution_irt(response_matrix, engine = NULL, ...)
```

## Arguments

| | |
|---|---|
| `response_matrix` | Response data supplied to an optional external engine. |
| `engine` | Optional function implementing the exact/nonparametric marginal likelihood estimator. If omitted, a gated specification is returned. |
| `...` | Passed to ‘engine‘ when supplied. |

## Value

An object of class "eye_gated_process_model", stored as a named list, with components "id", "purpose", "required_evidence", "engine", "fit", "status", "notes", "caveat". It contains a gated KDE latent-distribution IRT interface and associated metadata or diagnostics needed to interpret the result.

---

`fit_latent_class_process_irt`

*Latent process-class IRT reference model*

---

## Description

Latent process-class IRT reference model

## Usage

```
fit_latent_class_process_irt(
  data,
  response = "response",
  process_features,
  person = "participant_id",
  item = "item_id",
  n_classes = 2L,
  seed = 1
)
```

## Arguments

| | |
|---|---|
| data | Input data frame or compatible tabular object. |
| response | Response variable or response-column name. |
| process_features | Names of process-derived features. |
| person | Person or participant identifier column. |
| item | Item identifier, name, or item column. |
| n_classes | Number of latent classes. |
| seed | Random-number seed. |

## Value

An object of class "eye_latent_class_process_irt", stored as a named list, with components "response_model", "class", "centers", "data", "process_features", "status". It contains latent process-class IRT reference model and associated metadata or diagnostics needed to interpret the result.

---

# fit_latent_space_irt *Fit a latent-space IRT model using LSMjml*

---

## Description

Fit a latent-space IRT model using LSMjml

## Usage

```
fit_latent_space_irt(
  response_matrix,
  dimensions = 2L,
  penalty = NULL,
  constraint = NULL,
  starts = NULL,
  tol = 0.001,
  silent = TRUE
)
```

### Arguments

| | |
|---|---|
| response_matrix | Person-by-item matrix with lowest score coded zero. |
| dimensions | Latent-space dimensionality. |
| penalty | Optional L2 penalty passed to 'LSMjml::LSMfit()'. |
| constraint | Optional norm constraint 'C' passed to 'LSMfit()'. |
| starts | Starting-value strategy. |
| tol | Numerical convergence tolerance. |
| silent | Whether engine messages are suppressed. |

### Value

An object of class "eye_latent_space_irt", stored as a named list, with components "model", "person_coordinates", "item_coordinates", "person_intercept", "item_intercept", "dimensions", "engine", "status". It contains a latent-space IRT model using LSMjml and associated metadata or diagnostics needed to interpret the result.

---

## fit_lnirt_adapter *fit lnirt adapter*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
fit_lnirt_adapter(data, purpose, ...)
```

### Arguments

| | |
|---|---|
| data | Value for 'data'. See the function description and relevant article for constraints. |
| purpose | Value for 'purpose'. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

`fit_manyfacet_process_irt` *Many-facet process IRT reference model*

## Description

Fits crossed random effects for available person, item, device, session, site, algorithm, and AOI-definition facets. For a binary response this is a generalized many-facet reference model; it is not marketed as a FACETS software replica.

## Usage

```
fit_manyfacet_process_irt(
  data,
  response = "response",
  process = NULL,
  person = "participant_id",
  item = "item_id",
  device = NULL,
  session = NULL,
  site = NULL,
  algorithm = NULL,
  aoi_definition = NULL,
  process_family = c("gaussian", "poisson", "negative_binomial")
)
```

## Arguments

| | |
|---|---|
| `data` | Input data frame or compatible tabular object. |
| `response` | Response variable or response-column name. |
| `process` | Process variable or column name. |
| `person` | Person or participant identifier column. |
| `item` | Item identifier, name, or item column. |
| `device` | Device identifier or device facet. |
| `session` | Session identifier or session facet. |
| `site` | Site identifier or site facet. |
| `algorithm` | Algorithm identifier or algorithm facet. |
| `aoi_definition` | Value supplied to ‘aoi_definition‘; see Details for its model-specific role. |
| `process_family` | Distributional family for the process channel. |

## Value

An object of class "eye_manyfacet_process_irt", stored as a named list, with components "response_model", "process_model", "facets", "process_family", "status". It contains many-facet process IRT reference model and associated metadata or diagnostics needed to interpret the result.

## fit_mirt_adapter *fit mirt adapter*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
fit_mirt_adapter(data, model = 1L, purpose, ...)
```

### Arguments

| | |
|---|---|
| data | Value for 'data'. See the function description and relevant article for constraints. |
| model | Value for 'model'. See the function description and relevant article for constraints. |
| purpose | Value for 'purpose'. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## fit_mixture_irt_process_classes

*Fit a true mirt mixture-IRT response model*

### Description

This is distinct from the existing two-stage process-class reference model: the response distribution itself is calibrated using mirt's mixture density. Process features may then be compared with the fitted response classes only when a defensible class-membership extraction is available.

### Usage

```
fit_mixture_irt_process_classes(
  response_matrix,
  n_classes = 2L,
  model = 1,
  itemtype = "2PL",
  SE = FALSE
)
```

### Arguments

| | |
|---|---|
| response_matrix | Person x item response matrix. |
| n_classes | Number of mixture classes. The currently verified internal route supports two classes (‘mixture-2‘). |
| model | mirt model specification, default one dimension. |
| itemtype | Item type. |
| SE | Request standard errors. |

### Value

An object of class "eye_mixture_irt_process", stored as a named list, with components "model", "coefficients", "n_classes", "itemtype", "status", "caveat". It contains a true mirt mixture-IRT response model and associated metadata or diagnostics needed to interpret the result.

---

## fit_multiblock_process_map

*Fit a multiblock psychometric/gaze/pupil/quality structure map*

---

### Description

Uses FactoMineR MFA when available/requested. A block-standardized PCA fallback is available as a transparent exploratory reference and is explicitly labeled.

### Usage

```
fit_multiblock_process_map(
  x,
  blocks = NULL,
  id = NULL,
  engine = c("auto", "FactoMineR", "pca_block_scaled"),
  ncp = 5L
)
```

### Arguments

| | |
|---|---|
| x | A ‘process_feature_blocks()‘ object or data frame. |
| blocks | Required if ‘x‘ is a data frame. |
| id | Optional identifier. |
| engine | ‘auto‘, ‘FactoMineR‘, or ‘pca_block_scaled‘. |
| ncp | Number of components retained where supported. |

## Value

An object of class "eye_multiblock_process_map", stored as a named list, with components "model", "person_coordinates", "variable_coordinates", "block_coordinates", "blocks", "engine", "status", "caveat". It contains a multiblock psychometric/gaze/pupil/quality structure map and associated metadata or diagnostics needed to interpret the result.

---

`fit_multimodal_m2` *Fit the M2 response + RT + gaze reference model*

---

## Description

Fits the likelihood-faithful M2 reference model with CmdStanR. There is no silent fallback. Missing observations are omitted from their channel-specific likelihood under an explicit ignorable missingness assumption; the person and item structural layers remain joint across the observed channels.

## Usage

```
fit_multimodal_m2(
  x,
  person = "person_id",
  item = "item_id",
  response = "response",
  rt = "rt",
  gaze = "gaze",
  prior_profile = c("regularized", "paper_centered"),
  chains = 4L,
  parallel_chains = chains,
  iter_warmup = 1000L,
  iter_sampling = 1000L,
  seed = 20260814L,
  adapt_delta = 0.95,
  max_treedepth = 12L,
  refresh = 100L,
  quiet_compile = TRUE
)
```

## Arguments

| | |
|---|---|
| `x` | Data frame, 'eye_multimodal_m2_simulation', or compatible eyeprocess multimodal measurement object. |
| `person`, `item`, `response`, `rt`, `gaze` | Column names. |
| `prior_profile` | Prior profile. |
| `chains`, `parallel_chains`, `iter_warmup`, `iter_sampling` | CmdStan sampling controls. |
| `seed` | Reproducibility seed. |

adapt_delta, max_treedepth, refresh
: CmdStan controls.

quiet_compile
: Suppress CmdStan compilation messages.

## Value

An 'eye_multimodal_m2_fit'.

---

`fit_multimodal_m3` *Fit the M3 response + RT + gaze + pupil reference model*

---

## Description

Fits the four-channel reference likelihood using CmdStanR. Pupil summaries are standardized by default for a scale-stable reference parameterization; the transformation is retained in the returned data object. No missing nuisance values are silently imputed when the corresponding nuisance column is supplied.

## Usage

```
fit_multimodal_m3(
  x,
  person = "person_id",
  item = "item_id",
  response = "response",
  rt = "rt",
  gaze = "gaze",
  pupil = "pupil",
  baseline = "pupil_baseline",
  luminance = "luminance",
  gaze_x = "gaze_x",
  gaze_y = "gaze_y",
  quality = "pupil_quality",
  time_on_task = "time_on_task",
  blink = "pupil_blink",
  interpolated = "pupil_interpolated",
  device = "device",
  session = "session",
  sampling_rate = "sampling_rate_hz",
  pupil_scale = c("z", "raw"),
  prior_profile = c("regularized", "paper_centered"),
  nuisance = stats::setNames(rep(TRUE, 8L), .ep10_m3_nuisance_names),
  chains = 4L,
  parallel_chains = chains,
  iter_warmup = 1000L,
  iter_sampling = 1000L,
  seed = 20260815L,
```

```
  adapt_delta = 0.95,
  max_treedepth = 12L,
  refresh = 100L,
  quiet_compile = TRUE,
  init = 0
)
```

## Arguments

| | |
|---|---|
| x | Data frame, M3 simulation, or compatible measurement object. |
| person, item, response, rt, gaze, pupil | Column names. |
| baseline, luminance, gaze_x, gaze_y, quality, time_on_task | Optional nuisance columns. |
| blink, interpolated | Optional pupil nuisance/audit indicators. |
| device, session, sampling_rate | Optional measurement-context audit columns. |
| pupil_scale | '"z"' or '"raw"'. |
| prior_profile | Prior profile. |
| nuisance | Named logical vector selecting the eight explicit pupil measurement nuisance terms. |
| chains, parallel_chains, iter_warmup, iter_sampling | CmdStan sampling controls. |
| seed, adapt_delta, max_treedepth, refresh, quiet_compile | CmdStan controls. |
| init | CmdStan initialization; default zero initializes Cholesky factors at an interior identity point. |

## Value

An 'eye_multimodal_m3_fit'.

---

# fit_multimodal_m4 *Fit the M4 latent response-process state model*

---

## Description

Fits the marginalized M4 state model using CmdStanR. Sequence order is validated but never silently changed. The latent state contributes only to selected process channels in the reference implementation; the scored-response Rasch equation remains exactly state-independent.

## Usage

```
fit_multimodal_m4(
  x,
  spec = NULL,
  person = "person_id",
  item = "item_id",
  response = "response",
  rt = "rt",
  gaze = "gaze",
  pupil = "pupil",
  sequence = "sequence_id",
  order = "trial_index",
  baseline = "pupil_baseline",
  luminance = "luminance",
  gaze_x = "gaze_x",
  gaze_y = "gaze_y",
  quality = "pupil_quality",
  time_on_task = "time_on_task",
  blink = "pupil_blink",
  interpolated = "pupil_interpolated",
  device = "device",
  session = "session",
  sampling_rate = "sampling_rate_hz",
  pupil_scale = c("z", "raw"),
  n_states = 2L,
  state_channels = c("rt", "gaze", "pupil"),
  transition_structure = c("markov", "iid"),
  trait_conditioning = c("theta", "tau"),
  initial_trait_conditioning = TRUE,
  min_sequence_length = 2L,
  prior_profile = c("regularized", "paper_centered"),
  nuisance = stats::setNames(rep(TRUE, 8L), .ep10_m3_nuisance_names),
  chains = 4L,
  parallel_chains = chains,
  iter_warmup = 1000L,
  iter_sampling = 1000L,
  seed = 20260820L,
  adapt_delta = 0.97,
  max_treedepth = 13L,
  refresh = 100L,
  quiet_compile = TRUE,
  init = 0
)
```

## Arguments

| | |
|---|---|
| x | Data frame, M4 simulation, or compatible eyeprocess object. |
| spec | Optional ‘multimodal_m4_spec()‘. When omitted, a conservative two-state spec- |

ification is created from the explicit arguments.

`person`, `item`, `response`, `rt`, `gaze`, `pupil`
: Column names.

`sequence`, `order`
: Sequence identifier and within-sequence order columns.

`baseline`, `luminance`, `gaze_x`, `gaze_y`, `quality`, `time_on_task`, `blink`, `interpolated`
: Optional M3 pupil nuisance columns.

`device`, `session`, `sampling_rate`
: Optional measurement-context columns.

`pupil_scale`
: '"z"' or '"raw"'.

`n_states`, `state_channels`, `transition_structure`, `trait_conditioning`
: Used only when 'spec' is null.

`initial_trait_conditioning`, `min_sequence_length`
: Used only when 'spec' is null.

`prior_profile`, `nuisance`
: Used only when 'spec' is null.

`chains`, `parallel_chains`, `iter_warmup`, `iter_sampling`
: Sampling controls.

`seed`, `adapt_delta`, `max_treedepth`, `refresh`, `quiet_compile`, `init`
: CmdStan controls.

## Value

An 'eye_multimodal_m4_fit' retaining data, sequence audit, specification, Stan data, sampling controls, and provenance.

---

`fit_multimodal_trait_irt` *Multimodal trait-model convenience wrapper*

---

## Description

Extends the gaze/RT architecture to noncognitive or other latent traits by allowing caller-defined semantic labels. The statistical engine is delegated to 'fit_joint_gaze_rt_irt()'; interpretation remains the researcher's job.

## Usage

```
fit_multimodal_trait_irt(
  data,
  response,
  rt,
  gaze,
  person,
  item,
```

```
  trait_label = "trait",
  process_label = "process",
  ...
)
```

## Arguments

| | |
|---|---|
| data | Input data frame or compatible tabular object. |
| response | Response variable or response-column name. |
| rt | Response-time variable or column name. |
| gaze | Gaze/process variable or column name. |
| person | Person or participant identifier column. |
| item | Item identifier, name, or item column. |
| trait_label | Value supplied to 'trait_label'; see Details for its model-specific role. |
| process_label | Value supplied to 'process_label'; see Details for its model-specific role. |
| ... | Additional arguments passed to the selected model, engine, or method. |

## Value

An object of class "eye_joint_gaze_rt_irt", stored as a named list, with components "engine", "response_model", "rt_model", "gaze_model", "person_scores", "item_scores", "person_covariance", "item_covariance", "data_n", "gaze_family", "columns", "status", and additional components. It contains multimodal trait-model convenience wrapper and associated metadata or diagnostics needed to interpret the result.

---

# fit_multinomial_transition

*Fit a penalized multinomial transition model*

---

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
fit_multinomial_transition(design, ridge = 1e-4, reference_state = NULL,
  control = list(maxit = 1000L, reltol = 1e-9))
```

## Arguments

| | |
|---|---|
| design | Transition design. |
| ridge | Ridge penalty. |
| reference_state | Reference destination state. |
| control | Passed to 'optim()'. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`fit_multiple_response_process_irt` *Fit a multiple-response process-IRT reference model*

---

## Description

The bundled reference treats option selections as repeated binary outcomes with option-specific random difficulty/discrimination and optional gaze. This preserves option-level observations but does **not** reproduce the 2026 MRM/MRM-LD likelihood. Use ‘engine = "external"‘ for a validated exact implementation of a multiple-response model with inter-option dependence.

## Usage

```
fit_multiple_response_process_irt(
  data,
  selected = "selected",
  theta = "theta",
  person = "participant_id",
  item = "item_id",
  option = "option_id",
  gaze = NULL,
  engine = c("reference", "external"),
  external_engine = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| `data` | Long person-item-option table. |
| `selected` | Binary option-selection indicator. |
| `theta` | Supplied latent-trait estimate/score. |
| `person`, `item`, `option` | Identifiers. |
| `gaze` | Optional option-level process measure. |
| `engine` | ‘reference‘ or ‘external‘. |
| `external_engine` | Validated external fitter. |
| `...` | Arguments passed to the external engine. |

## Value

An object of class "eye_multiple_response_process_irt", stored as a named list, with components "model", "data", "gaze", "engine", "exact_multiple_response", "status", "note". It contains a multiple-response process-IRT reference model and associated metadata or diagnostics needed to interpret the result.

---

`fit_nominal_gaze_irt` *Nominal/distractor IRT with option-level gaze*

---

## Description

The bundled estimator is a transparent two-stage process-augmented nominal model: participant ability may be supplied, or a shrinkage logit accuracy proxy is estimated; option-level gaze proportions then enter a multinomial response model. This is intended for validation and exploratory distractor research, not as a replacement for a fully latent nominal-response model.

## Usage

```
fit_nominal_gaze_irt(
  data,
  response_option = "response_option",
  option_gaze,
  person = "participant_id",
  item = "item_id",
  ability = NULL,
  correct_option = NULL,
  add_item_effects = TRUE,
  ...
)
```

## Arguments

| | |
|---|---|
| `data` | Input data frame or compatible tabular object. |
| `response_option` | Column identifying the selected response option. |
| `option_gaze` | Character vector naming one gaze column per response option. Names should correspond to option labels when possible. |
| `person` | Person or participant identifier column. |
| `item` | Item identifier, name, or item column. |
| `ability` | Optional existing ability score column. |
| `correct_option` | Optional scalar or column name identifying correct option. |
| `add_item_effects` | Whether item effects are included. |
| `...` | Additional arguments passed to the selected model, engine, or method. |

## Value

An object of class "eye_nominal_gaze_irt", stored as a named list, with components "model", "baseline_model", "data", "option_gaze", "gaze_proportion_columns", "ability", "person", "item", "logLik_gain", "status", "note". It contains nominal/distractor IRT with option-level gaze and associated metadata or diagnostics needed to interpret the result.

---

`fit_nonignorable_missing_irt`
*Create a gated Bayesian nonignorable-missing IRT interface*

---

## Description

Create a gated Bayesian nonignorable-missing IRT interface

## Usage

```
fit_nonignorable_missing_irt(data, engine = NULL, ...)
```

## Arguments

| | |
|---|---|
| `data` | Response/missingness data. |
| `engine` | Optional externally validated Bayesian estimator. |
| `...` | Passed to ‘engine‘. |

## Value

An object of class "eye_gated_process_model", stored as a named list, with components "id", "purpose", "required_evidence", "engine", "fit", "status", "notes", "caveat". It contains a gated Bayesian nonignorable-missing IRT interface and associated metadata or diagnostics needed to interpret the result.

---

`fit_omission_survival_irt`
*Response/RT/omission survival IRT reference model*

---

## Description

Fits a response model among reached/answered observations and cause-specific survival models for omission and not-reached processes. The function keeps the missingness mechanisms distinct by construction.

## Usage

```
fit_omission_survival_irt(
  data,
  response = "response",
  response_time = "response_time",
  omission_time = NULL,
  reached = "reached",
  person = "participant_id",
  item = "item_id",
  gaze_exposure = NULL,
  first_fixation_latency = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| data | Input data frame or compatible tabular object. |
| response | Response variable or response-column name. |
| response_time | Response-time variable or column name. |
| omission_time | Time associated with an omitted response. |
| reached | Indicator that the item was reached. |
| person | Person or participant identifier column. |
| item | Item identifier, name, or item column. |
| gaze_exposure | Gaze-based exposure measure. |
| first_fixation_latency | Latency to first fixation. |
| ... | Additional arguments passed to the selected model, engine, or method. |

## Value

An object of class "eye_omission_survival_irt", stored as a named list, with components "response_model", "omission_model", "not_reached_model", "classified_data", "state_counts", "status", "note". It contains response/RT/omission survival IRT reference model and associated metadata or diagnostics needed to interpret the result.

---

# fit_openmx_adapter *fit openmx adapter*

---

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
fit_openmx_adapter(model, purpose, ...)
```

## Arguments

| | |
|---|---|
| model | Value for 'model'. See the function description and relevant article for constraints. |
| purpose | Value for 'purpose'. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

# fit_openmx_process_model

*Fit a user-defined OpenMx process model*

---

## Description

Fit a user-defined OpenMx process model

## Usage

```
fit_openmx_process_model(x, model_builder, include_features = TRUE, ...)
```

## Arguments

| | |
|---|---|
| x | An 'eye_dataset'. |
| model_builder | Function receiving model data and returning an OpenMx model. |
| include_features | Whether to merge process features. |
| ... | Passed to 'OpenMx::mxRun()'. |

## Value

An 'eyeprocess_model'.

## fit_persistence_gaze_diffusion_irt *Create a gated persistence-augmented gaze-diffusion IRT interface*

### Description

Create a gated persistence-augmented gaze-diffusion IRT interface

### Usage

```
fit_persistence_gaze_diffusion_irt(data, engine = NULL, ...)
```

### Arguments

| | |
|---|---|
| data | Response/RT/process data. |
| engine | Optional externally validated estimator function. |
| ... | Passed to 'engine'. |

### Value

An object of class "eye_gated_process_model", stored as a named list, with components "id", "purpose", "required_evidence", "engine", "fit", "status", "notes", "caveat". It contains a gated persistence-augmented gaze-diffusion IRT interface and associated metadata or diagnostics needed to interpret the result.

## fit_process_dif *Dynamic process-DIF and fairness drift*

### Description

Dynamic process-DIF and fairness drift. These functions form the eyeprocess 0.6.0.9000 measurement-intelligence programme and use dependency-free reference implementations with explicit evidence limits.

### Usage

```
fit_process_dif(x, response, process, group, item, ability = NULL)
monitor_dif_drift(x, time, group, metrics, item = NULL)
decompose_dif_evidence(psychometric, process = NULL, design_features = NULL)
audit_fairness_transportability(x, context = "device", effect = "process_dif",
  item = "item_id")
plot_group_icc_process_overlay(x, ...)
plot_process_dif_forest(x, ...)
plot_dif_drift_heatmap(x, ...)
plot_fairness_transport_matrix(x, ...)
plot_item_group_process_curves(x, ...)
```

## Arguments

| | |
|---|---|
| x | Input object or data structure appropriate for the selected analysis. |
| response | Argument controlling 'response'; see the function usage and returned audit metadata. |
| process | Argument controlling 'process'; see the function usage and returned audit metadata. |
| group | Argument controlling 'group'; see the function usage and returned audit metadata. |
| item | Argument controlling 'item'; see the function usage and returned audit metadata. |
| ability | Argument controlling 'ability'; see the function usage and returned audit metadata. |
| time | Argument controlling 'time'; see the function usage and returned audit metadata. |
| metrics | Argument controlling 'metrics'; see the function usage and returned audit metadata. |
| psychometric | Argument controlling 'psychometric'; see the function usage and returned audit metadata. |
| design_features | Argument controlling 'design_features'; see the function usage and returned audit metadata. |
| context | Argument controlling 'context'; see the function usage and returned audit metadata. |
| effect | Argument controlling 'effect'; see the function usage and returned audit metadata. |
| ... | Additional arguments passed to the underlying method or plotting function. |

## Details

The APIs return auditable S3 objects. Plot wrappers call registered base-graphics methods. Experimental or approximate engines are labelled in object status fields and should be validated before confirmatory or operational use.

## Value

An eyeprocess result object, data frame, model object, plot, or audit table as documented by the individual function.

## See Also

`plot_diagnostics()`, `plot_evidence()`, and `plot_sensitivity()`.

`fit_process_gstudy` *Process reliability and Generalizability Theory*

## Description

Process reliability and Generalizability Theory. These functions form the eyeprocess 0.6.0.9000 measurement-intelligence programme and use dependency-free reference implementations with explicit evidence limits.

## Usage

```
fit_process_gstudy(x, metric, facets = c("person", "item", "session", "device"),
  design = c("crossed", "nested"))
process_variance_components(x)
design_process_dstudy(gstudy, persons = NULL, items = seq(5, 50, 5), sessions = 1:5,
  devices = 1)
audit_process_reliability(x, metrics, method = c("icc", "gtheory", "split_half",
  "bootstrap"), person_col = "person_id", item_col = "item_id", draws = 250)
plot_variance_components(x, ...)
plot_dependability_surface(x, ...)
plot_reliability_by_metric(x, ...)
plot_session_stability(x, ...)
plot_item_sampling_reliability(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Input object or data structure appropriate for the selected analysis. |
| `metric` | Argument controlling ‘metric‘; see the function usage and returned audit metadata. |
| `facets` | Argument controlling ‘facets‘; see the function usage and returned audit metadata. |
| `design` | Argument controlling ‘design‘; see the function usage and returned audit metadata. |
| `gstudy` | Argument controlling ‘gstudy‘; see the function usage and returned audit metadata. |
| `persons` | Argument controlling ‘persons‘; see the function usage and returned audit metadata. |
| `items` | Argument controlling ‘items‘; see the function usage and returned audit metadata. |
| `sessions` | Argument controlling ‘sessions‘; see the function usage and returned audit metadata. |
| `devices` | Argument controlling ‘devices‘; see the function usage and returned audit metadata. |
| `metrics` | Argument controlling ‘metrics‘; see the function usage and returned audit metadata. |

| | |
|---|---|
| method | Argument controlling 'method'; see the function usage and returned audit metadata. |
| person_col | Argument controlling 'person_col'; see the function usage and returned audit metadata. |
| item_col | Argument controlling 'item_col'; see the function usage and returned audit metadata. |
| draws | Argument controlling 'draws'; see the function usage and returned audit metadata. |
| ... | Additional arguments passed to the underlying method or plotting function. |

## Details

The APIs return auditable S3 objects. Plot wrappers call registered base-graphics methods. Experimental or approximate engines are labelled in object status fields and should be validated before confirmatory or operational use.

## Value

An eyeprocess result object, data frame, model object, plot, or audit table as documented by the individual function.

## See Also

`plot_diagnostics()`, `plot_evidence()`, and `plot_sensitivity()`.

---

# fit_process_hmm_irt *Process-state HMM with an IRT response layer*

---

## Description

Fits a diagonal-Gaussian HMM to standardized process features within each sequence, then uses state occupancy as explicit process evidence in a response model. This two-stage reference engine is deliberately interpretable and should be distinguished from a fully joint HMM-IRT likelihood.

## Usage

```
fit_process_hmm_irt(
  data,
  sequence_id = "trial_id",
  order = "timestamp",
  process_features = c("x", "y"),
  response = "response",
  person = "participant_id",
  item = "item_id",
  n_states = 3L,
  max_iter = 100L,
```

```
  tol = 1e-05,
  seed = 1
)
```

## Arguments

| | |
|---|---|
| data | Input data frame or compatible tabular object. |
| sequence_id | Sequence identifier. |
| order | Within-sequence ordering variable. |
| process_features | Names of process-derived features. |
| response | Response variable or response-column name. |
| person | Person or participant identifier column. |
| item | Item identifier, name, or item column. |
| n_states | Number of latent process states. |
| max_iter | Maximum number of iterations. |
| tol | Numerical convergence tolerance. |
| seed | Random-number seed. |

## Value

An object of class "eye_process_hmm_irt", stored as a named list, with components "pi", "transition", "means", "sds", "posterior_state", "state", "row_data", "occupancy", "summary_data", "response_model", "logLik", "logLik_history", and additional components. It contains process-state HMM with an IRT response layer and associated metadata or diagnostics needed to interpret the result.

---

# fit_process_missingness_model

*Measurement-intelligence compatibility adapters*

---

## Description

Measurement-intelligence compatibility adapters. These functions form the eyeprocess 0.6.0.9000 measurement-intelligence programme and use dependency-free reference implementations with explicit evidence limits.

## Usage

```
fit_process_missingness_model(x, observed, predictors, random = c("person", "item"))
crossmodal_recurrence_model(x, y, outcome = NULL, channels = c("gaze_pupil",
  "gaze_eda", "pupil_eda"), radius = NULL, covariates = NULL)
```

## Arguments

| | |
|---|---|
| x | Input object or data structure appropriate for the selected analysis. |
| observed | Argument controlling 'observed'; see the function usage and returned audit metadata. |
| predictors | Argument controlling 'predictors'; see the function usage and returned audit metadata. |
| random | Argument controlling 'random'; see the function usage and returned audit metadata. |
| y | Argument controlling 'y'; see the function usage and returned audit metadata. |
| outcome | Argument controlling 'outcome'; see the function usage and returned audit metadata. |
| channels | Argument controlling 'channels'; see the function usage and returned audit metadata. |
| radius | Argument controlling 'radius'; see the function usage and returned audit metadata. |
| covariates | Argument controlling 'covariates'; see the function usage and returned audit metadata. |

## Details

The APIs return auditable S3 objects. Plot wrappers call registered base-graphics methods. Experimental or approximate engines are labelled in object status fields and should be validated before confirmatory or operational use.

## Value

An eyeprocess result object, data frame, model object, plot, or audit table as documented by the individual function.

## See Also

`plot_diagnostics()`, `plot_evidence()`, and `plot_sensitivity()`.

---

# fit_process_norms *Conditional process reference centiles*

---

## Description

Conditional process reference centiles. These functions form the eyeprocess 0.6.0.9000 measurement-intelligence programme and use dependency-free reference implementations with explicit evidence limits.

## Usage

```
fit_process_norms(x, metric, covariates, family = c("auto", "gaussian",
  "lognormal"))
predict_process_centiles(model, newdata, centiles = c(2.5, 10, 25, 50, 75, 90,
  97.5))
score_process_deviation(model, newdata, type = c("z", "centile",
  "tail_probability"))
audit_norm_transportability(model, new_sample)
plot_process_centiles(x, ...)
plot_normative_fan(x, ...)
plot_person_normative_profile(x, ...)
plot_item_normative_deviation(x, ...)
```

## Arguments

| | |
|---|---|
| x | Input object or data structure appropriate for the selected analysis. |
| metric | Argument controlling 'metric'; see the function usage and returned audit metadata. |
| covariates | Argument controlling 'covariates'; see the function usage and returned audit metadata. |
| family | Argument controlling 'family'; see the function usage and returned audit metadata. |
| model | Argument controlling 'model'; see the function usage and returned audit metadata. |
| newdata | Argument controlling 'newdata'; see the function usage and returned audit metadata. |
| centiles | Argument controlling 'centiles'; see the function usage and returned audit metadata. |
| type | Argument controlling 'type'; see the function usage and returned audit metadata. |
| new_sample | Argument controlling 'new_sample'; see the function usage and returned audit metadata. |
| ... | Additional arguments passed to the underlying method or plotting function. |

## Details

The APIs return auditable S3 objects. Plot wrappers call registered base-graphics methods. Experimental or approximate engines are labelled in object status fields and should be validated before confirmatory or operational use.

## Value

An eyeprocess result object, data frame, model object, plot, or audit table as documented by the individual function.

## See Also

`plot_diagnostics()`, `plot_evidence()`, and `plot_sensitivity()`.

`fit_process_observation_model` *Informative missingness and MNAR sensitivity*

## Description

Informative missingness and MNAR sensitivity. These functions form the eyeprocess 0.6.0.9000 measurement-intelligence programme and use dependency-free reference implementations with explicit evidence limits.

## Usage

```
fit_process_observation_model(x, observed, predictors, random = c("person", "item"))
fit_joint_signal_missingness(outcome, observation, method = c("selection",
  "shared_parameter"), x = NULL, predictors = NULL)
process_pattern_mixture(x, delta = seq(-1, 1, 0.1), metric = NULL, estimand = mean)
sensitivity_mnar_process(x, estimand = mean, null = 0, ...)
plot_observation_probability(x, ...)
plot_missingness_by_time(x, ...)
plot_missingness_by_aoi(x, ...)
plot_mnar_tipping_point(x, ...)
plot_complete_case_sensitivity(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Input object or data structure appropriate for the selected analysis. |
| `observed` | Argument controlling 'observed'; see the function usage and returned audit metadata. |
| `predictors` | Argument controlling 'predictors'; see the function usage and returned audit metadata. |
| `random` | Argument controlling 'random'; see the function usage and returned audit metadata. |
| `outcome` | Argument controlling 'outcome'; see the function usage and returned audit metadata. |
| `observation` | Argument controlling 'observation'; see the function usage and returned audit metadata. |
| `method` | Argument controlling 'method'; see the function usage and returned audit metadata. |
| `delta` | Argument controlling 'delta'; see the function usage and returned audit metadata. |
| `metric` | Argument controlling 'metric'; see the function usage and returned audit metadata. |
| `estimand` | Argument controlling 'estimand'; see the function usage and returned audit metadata. |
| `null` | Argument controlling 'null'; see the function usage and returned audit metadata. |
| `...` | Additional arguments passed to the underlying method or plotting function. |

## Details

The APIs return auditable S3 objects. Plot wrappers call registered base-graphics methods. Experimental or approximate engines are labelled in object status fields and should be validated before confirmatory or operational use.

## Value

An eyeprocess result object, data frame, model object, plot, or audit table as documented by the individual function.

## See Also

`plot_diagnostics()`, `plot_evidence()`, and `plot_sensitivity()`.

---

`fit_process_profile_mixture` *Fit exploratory process profiles*

---

## Description

Fit exploratory process profiles

## Usage

```
fit_process_profile_mixture(
  data,
  variables,
  k = 3L,
  id = "person_id",
  engine = c("auto", "tidyLPA", "kmeans_reference"),
  seed = 777
)
```

## Arguments

| | |
|---|---|
| `data` | Person-level process data. |
| `variables` | Continuous process variables. |
| `k` | Number of profiles. |
| `id` | Optional person identifier. |
| `engine` | 'auto', 'tidyLPA', or 'kmeans_reference'. |
| `seed` | Random seed. |

## Value

An object of class "eye_process_profile_mixture", stored as a named list, with components "model", "assignment", "summary", "variables", "k", "engine", "scaled_data", "status", "caveat". It contains exploratory process profiles and associated metadata or diagnostics needed to interpret the result.

## fit_process_rasch_tree

*Fit a process-informed Rasch tree*

### Description

Fit a process-informed Rasch tree

### Usage

```
fit_process_rasch_tree(
  response_matrix,
  covariates,
  formula = NULL,
  maxit = 60L
)
```

### Arguments

| | |
|---|---|
| response_matrix | Person x item dichotomous response matrix. |
| covariates | Person-level response-process covariates. |
| formula | Optional splitting formula. If omitted, all covariates are used. |
| maxit | Maximum model iterations. |

### Value

An object of class "eye_process_rasch_tree", stored as a named list, with components "model", "covariates", "formula", "status", "caveat". It contains a process-informed Rasch tree and associated metadata or diagnostics needed to interpret the result.

## fit_pupillometryr_adapter

*fit pupillometryr adapter*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
fit_pupillometryr_adapter(data, purpose, ...)
```

### Arguments

| | |
|---|---|
| `data` | Value for ‘data‘. See the function description and relevant article for constraints. |
| `purpose` | Value for ‘purpose‘. See the function description and relevant article for constraints. |
| `...` | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

## `fit_pupil_confound_model` *Fit a luminance/fatigue/process confound model for pupil response*

---

### Description

Fit a luminance/fatigue/process confound model for pupil response

### Usage

```
fit_pupil_confound_model(
  data,
  pupil = "pupil_peak",
  luminance = "screen_luminance",
  trial_order = "trial_sequence",
  theta = NULL,
  person = "person_id",
  item = "item_id",
  engine = c("auto", "mgcv", "lm")
)
```

### Arguments

| | |
|---|---|
| `data` | Trial-level data. |
| `pupil`, `luminance`, `trial_order` | Column names. |
| `theta` | Optional latent-score column. |
| `person`, `item` | Optional identifiers. |
| `engine` | ‘auto‘, ‘mgcv‘, or ‘lm‘. |

### Value

An ‘eye_pupil_confound_model‘ object containing raw and adjusted values.

`fit_pupil_event_deconvolution` *Fit transparent event-related pupil deconvolution models*

## Description

Fits a linear superposition model of overlapping event kernels. This is a transparent reference deconvolution layer, not a universal physiological model.

## Usage

```
fit_pupil_event_deconvolution(
  data,
  by = c("person_id", "trial_id"),
  time = "time_ms",
  pupil = "pupil_bc",
  events = list(stimulus = 0),
  tmax_ms = 930,
  shape = 10.1,
  min_samples = 20L
)
```

## Arguments

| | |
|---|---|
| `data` | Sample-level pupil data. |
| `by` | Grouping columns, typically person and trial. |
| `time`, `pupil` | Column names. |
| `events` | Named list mapping event labels to scalar event times or columns. |
| `tmax_ms`, `shape` | Kernel parameters. |
| `min_samples` | Minimum usable samples per group. |

## Value

An object of class "eye_pupil_deconvolution", stored as a named list, with components "fits", "effects", "fitted", "events", "tmax_ms", "shape", "by", "status", "caveat". It contains transparent event-related pupil deconvolution models and associated metadata or diagnostics needed to interpret the result.

`fit_response_process_embedding_irt` *Fit an IRT response model augmented by sequence embeddings*

## Description

Fit an IRT response model augmented by sequence embeddings

## Usage

```
fit_response_process_embedding_irt(
  data,
  sequences,
  response = "response",
  person = "participant_id",
  item = "item_id",
  dimensions = 5L,
  n = c(1L, 2L, 3L)
)
```

## Arguments

| | |
|---|---|
| `data` | Input data frame or compatible tabular object. |
| `sequences` | Sequence inputs. |
| `response` | Response variable or response-column name. |
| `person` | Person or participant identifier column. |
| `item` | Item identifier, name, or item column. |
| `dimensions` | Number of embedding dimensions. |
| `n` | Requested count or n-gram order, depending on context. |

## Value

An object of class "eye_response_process_embedding_irt", stored as a named list, with components "model", "embedding", "data", "status". It contains an IRT response model augmented by sequence embeddings and associated metadata or diagnostics needed to interpret the result.

`fit_revisit_process_cdm` *Fit a revisiting-aware cognitive-diagnosis process workflow*

## Description

Convenience adapter motivated by current work combining response time and item revisiting with cognitive diagnosis. The response layer is delegated to 'fit_cognitive_diagnosis_process()'; revisiting/RT/gaze remain process evidence and do not redefine attributes or the Q-matrix.

## Usage

```
fit_revisit_process_cdm(
  response_matrix,
  q_matrix,
  process_data,
  person_id = "participant_id",
  revisited = "revisited",
  rt = "response_time",
  gaze = NULL,
  ...
)
```

## Arguments

| | |
|---|---|
| `response_matrix` | Person-by-item responses. |
| `q_matrix` | Item-by-attribute Q-matrix. |
| `process_data` | Long process data. |
| `person_id` | Person identifier in 'process_data'. |
| `revisited` | Revisit indicator/count column. |
| `rt` | Response-time column. |
| `gaze` | Optional gaze process columns. |
| `...` | Passed to 'fit_cognitive_diagnosis_process()'. |

## Value

An object of class "eye_cognitive_diagnosis_process", stored as a named list, with components "response_model", "process_summary", "process_mastery_correlation", "q_matrix", "status". It contains a revisiting-aware cognitive-diagnosis process workflow and associated metadata or diagnostics needed to interpret the result.

## fit_seqhmm_adapter *fit seqhmm adapter*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
fit_seqhmm_adapter(data, purpose, ...)
```

### Arguments

| | |
|---|---|
| data | Value for ‘data‘. See the function description and relevant article for constraints. |
| purpose | Value for ‘purpose‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## fit_speed_accuracy_engagement_irt *Speed-accuracy-engagement IRT convenience wrapper*

### Description

Speed-accuracy-engagement IRT convenience wrapper

### Usage

```
fit_speed_accuracy_engagement_irt(data, ..., engine = c("reference", "brms"))
```

### Arguments

| | |
|---|---|
| data | Input data frame or compatible tabular object. |
| ... | Additional arguments passed to the selected model, engine, or method. |
| engine | Estimation engine. |

## Value

An object of class "eye_joint_gaze_rt_irt", stored as a named list, with components "engine", "response_model", "rt_model", "gaze_model", "person_scores", "item_scores", "person_covariance", "item_covariance", "data_n", "gaze_family", "columns", "status", and additional components. It contains speed-accuracy-engagement IRT convenience wrapper and associated metadata or diagnostics needed to interpret the result.

---

`fit_strategy_mixture_em`

*Fit the deterministic multi-start EM baseline*

---

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
fit_strategy_mixture_em(prepared, starts = prepared$spec$multiple_starts,
  max_iter = 300L, tolerance = 1e-7, seed = 1L)
```

## Arguments

| | |
|---|---|
| `prepared` | Prepared strategy data. |
| `starts` | Number of starts. |
| `max_iter` | Maximum EM iterations. |
| `tolerance` | Relative log-likelihood tolerance. |
| `seed` | Seed. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`fit_strategy_mixture_stan`

*Fit the probabilistic strategy-mixture engine*

---

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
fit_strategy_mixture_stan(prepared, seed = 1L, refresh = 0L, output_dir = NULL, ...)
```

### Arguments

| | |
|---|---|
| prepared | Prepared strategy data. |
| seed | Seed. |
| refresh | CmdStan refresh interval. |
| output_dir | Optional output directory. |
| ... | Additional CmdStan sampling arguments. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

## fit_tam_adapter *fit tam adapter*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
fit_tam_adapter(data, purpose, ...)
```

### Arguments

| | |
|---|---|
| data | Value for ‘data‘. See the function description and relevant article for constraints. |
| purpose | Value for ‘purpose‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## fit_theory_strategy_irt *Fit a theory-constrained strategy mixture*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
fit_theory_strategy_irt(data, spec, seed = 1L, response = NULL, participant = NULL,
  item = NULL, ...)
```

### Arguments

| | |
|---|---|
| data | Trial-level data. |
| spec | Strategy specification. |
| seed | Seed. |
| response | Value for 'response'. See the function description and relevant article for constraints. |
| participant | Value for 'participant'. See the function description and relevant article for constraints. |
| item | Value for 'item'. See the function description and relevant article for constraints. |
| ... | Engine arguments. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## fit_traminer_adapter *fit traminer adapter*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
fit_traminer_adapter(data, purpose, ...)
```

### Arguments

| | |
|---|---|
| data | Value for ‘data‘. See the function description and relevant article for constraints. |
| purpose | Value for ‘purpose‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

## fit_validation_replicate

*Fit one model-validation replicate*

---

### Description

Fit one model-validation replicate

### Usage

```
fit_validation_replicate(
  replicate,
  generator,
  fitter,
  extractor,
  scenario = "baseline",
  engine = "unspecified"
)
```

### Arguments

| | |
|---|---|
| replicate | Replicate id. |
| generator | Function ‘(replicate, scenario)‘ returning simulated data; the simulation should expose truth via ‘extract_parameter_truth()‘. |
| fitter | Function accepting the simulated data (or its ‘$data‘ member). |
| extractor | Function ‘(fit, simulation)‘ returning estimates with at least ‘parameter‘ and ‘estimate‘; optional ‘lower‘/‘upper‘ are retained. |
| scenario | Scenario label/object passed to the generator. |
| engine | Engine label. |

### Value

A logical value or vector indicating one model-validation replicate.

## fit_variational_irt *Variational IRT external-engine gate*

### Description

Variational IRT external-engine gate

### Usage

```
fit_variational_irt(response_matrix, external_engine = NULL, ...)
```

### Arguments

| | |
|---|---|
| response_matrix | Person-by-item response matrix. |
| external_engine | Validated external fitting function. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

An object of class "eye_variational_irt", stored as a named list, with components "model", "status", "engine". It contains variational IRT external-engine gate and associated metadata or diagnostics needed to interpret the result.

## fit_visual_context_irt *Fit an explicit visual-context/testlet IRT model*

### Description

Fit an explicit visual-context/testlet IRT model

### Usage

```
fit_visual_context_irt(
  response_matrix,
  registry,
  context = NULL,
  itemtype = "2PL",
  model_dimension = "Ability",
  context_dimension = "VisualContextFactor",
  SE = FALSE
)
```

### Arguments

| | |
|---|---|
| response_matrix | Person x item response matrix. |
| registry | 'visual_context_registry()' object. |
| context | Optional context identifier to model; defaults to the first shared context. |
| itemtype | mirt item type. |
| model_dimension | Name for the primary latent dimension. |
| context_dimension | Name for the context/testlet dimension. |
| SE | Request standard errors from mirt. |

### Value

An object of class "eye_visual_context_irt", stored as a named list, with components "base_model", "context_model", "comparison", "registry", "context", "positions", "itemtype", "model_string", "status", "caveat". It contains an explicit visual-context/testlet IRT model and associated metadata or diagnostics needed to interpret the result.

---

## fixation_boundary_uncertainty

*Distance to nearest rectangular AOI boundary*

---

### Description

Distance to nearest rectangular AOI boundary

### Usage

```
fixation_boundary_uncertainty(data, aois, x = "gaze_x", y = "gaze_y")
```

### Arguments

| | |
|---|---|
| data | Gaze samples. |
| aois | Rectangular AOIs. |
| x, y | Coordinates. |

### Value

A tabular R object containing distance to nearest rectangular AOI boundary; rows represent analysis units and columns contain the returned quantities.

## freeze_eyeprocess_irt_reference

*Freeze IRT validation reference summaries*

### Description

Freeze IRT validation reference summaries

### Usage

```
freeze_eyeprocess_irt_reference(
  recovery_summary = NULL,
  sbc = NULL,
  failures = NULL,
  metadata = list()
)
```

### Arguments

| | |
|---|---|
| recovery_summary | Parameter-recovery summary object or table. |
| sbc | Simulation-based-calibration evidence object or table. |
| failures | Failure records or failure summary. |
| metadata | Named metadata to store with the frozen object. |

### Value

A named list with components "recovery_summary", "sbc", "failures", "metadata", "scientific_scope", containing freeze IRT validation reference summaries and associated metadata or diagnostics.

## freeze_eyeprocess_validation_atlas

*Freeze a validation atlas with a reproducibility fingerprint*

### Description

Freeze a validation atlas with a reproducibility fingerprint

### Usage

```
freeze_eyeprocess_validation_atlas(atlas, metadata = list())
```

### Arguments

| | |
|---|---|
| atlas | Validation-evidence atlas object. |
| metadata | Named metadata to store with the frozen object. |

## Value

An object of class "eye_validation_atlas_freeze", stored as a named list, with components "payload", "hash", "frozen". It contains freeze a validation atlas with a reproducibility fingerprint and associated metadata or diagnostics needed to interpret the result.

---

`freeze_eyeprocess_validation_evidence` *Freeze a complete Milestone #2 evidence bundle*

---

## Description

Freeze a complete Milestone #2 evidence bundle

## Usage

```
freeze_eyeprocess_validation_evidence(
  design,
  recovery = NULL,
  sbc = NULL,
  stress = NULL,
  reliability = NULL,
  negative_controls = NULL,
  irt = NULL,
  claims = NULL,
  provenance = NULL,
  source_commit = NA_character_
)
```

## Arguments

| | |
|---|---|
| `design` | Validation or simulation design object. |
| `recovery` | Parameter-recovery evidence object or table. |
| `sbc` | Simulation-based-calibration evidence object or table. |
| `stress` | Measurement-stress evidence object or table. |
| `reliability` | Reliability or repeatability evidence object or table. |
| `negative_controls` | Negative-control evidence object or table. |
| `irt` | IRT-specific evidence object or table. |
| `claims` | Claim-evidence mapping table. |
| `provenance` | Provenance metadata or provenance object. |
| `source_commit` | Source-control commit associated with the evidence. |

### Value

An object of class "eye_validation_evidence_freeze", stored as a named list, with components "components", "presence", "source_commit", "frozen_at", "scientific_scope". It contains freeze a complete Milestone #2 evidence bundle and associated metadata or diagnostics needed to interpret the result.

---

## freeze_software_paper_evidence

*Freeze software-paper evidence to an RDS with a hash*

---

### Description

Freeze software-paper evidence to an RDS with a hash

### Usage

```
freeze_software_paper_evidence(x, path)
```

### Arguments

| | |
|---|---|
| x | Evidence bundle. |
| path | Output path. |

### Value

A data frame containing freeze software-paper evidence to an RDS with a hash. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

---

## freeze_validation_reference

*Freeze a compact validation reference for regression testing*

---

### Description

Freeze a compact validation reference for regression testing

### Usage

```
freeze_validation_reference(x, path = NULL, digits = 8L)
```

### Arguments

| | |
|---|---|
| x | Validation result. |
| path | Optional RDS path. |
| digits | Numeric rounding applied before hashing. |

## Value

An object of class "eye_validation_reference", stored as a named list, with components "summary", "failure_profile", "design_hash", "summary_hash", "created_at", "status". It contains freeze a compact validation reference for regression testing and associated metadata or diagnostics needed to interpret the result.

---

`functional_pupil_basis` *Construct a functional basis for pupil trajectories*

---

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
functional_pupil_basis(x, df = 6L, basis = c("natural_spline", "bspline"),
  degree = 3L, boundary_knots = NULL, knots = NULL)
```

## Arguments

| | |
|---|---|
| `x` | Prepared functional pupil data or numeric time vector. |
| `df` | Degrees of freedom. |
| `basis` | Natural spline or B-spline. |
| `degree` | B-spline polynomial degree; ignored for natural splines. |
| `boundary_knots` | Optional boundary knots. |
| `knots` | Optional internal knots. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## functional_pupil_diagnostics

*Diagnose functional pupil model and preprocessing quality*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
functional_pupil_diagnostics(x)
```

### Arguments

| | |
|---|---|
| x | Functional pupil fit. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## functional_pupil_irt_spec

*Specify a functional pupil-IRT model*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
functional_pupil_irt_spec(df = 6L, basis = c("natural_spline", "bspline"),
 response = "score", engine = c("two_stage_glm", "two_stage_lme4", "brms", "stan"),
  alignment = c("trial", "event"), event_time_column = NULL, latency_ms = 200,
 baseline_window = c(-200, 0), baseline_method = c("subtract", "percent", "zscore"),
  min_baseline_samples = 3L, drop_invalid_baseline = TRUE, time_window = NULL,
 pupil_column = NULL, time_column = NULL, participant_column = "participant_id",
  item_column = "item_id", trial_column = "trial_id", luminance_column = NULL,
  gaze_x_column = NULL, gaze_y_column = NULL, blink_column = NULL,
  interpolated_column = NULL, max_interpolated_fraction = 0.20,
  nuisance_by_participant = FALSE, include_response_time = TRUE, ar1 = TRUE,
  participant_effect = TRUE, item_effect = TRUE, chains = 4L,
  parallel_chains = chains, iter_warmup = 1000L, iter_sampling = 1000L,
  adapt_delta = 0.95, max_treedepth = 12L)
```

## Arguments

| | |
|---|---|
| df | Basis degrees of freedom. |
| basis | Natural spline or B-spline basis. |
| response | Response field. |
| engine | Two-stage baseline, multilevel baseline, brms bridge, or |
| alignment | Trial- or event-aligned time. |
| event_time_column | Event timestamp column required for event alignment. |
| latency_ms | Physiological latency shift applied before basis creation. |
| baseline_window | Numeric time window used for baseline correction. |
| baseline_method | Subtract, percent change, or z score. |
| min_baseline_samples | Minimum finite baseline samples per trial. |
| drop_invalid_baseline | Whether to exclude trials with invalid baselines. |
| time_window | Optional analysis time window. |
| pupil_column | Column names. |
| time_column | Column names. |
| participant_column | Column names. |
| item_column | Column names. |
| trial_column | Column names. |
| luminance_column | Optional nuisance columns. |
| gaze_x_column | Optional nuisance columns. |
| gaze_y_column | Optional nuisance columns. |
| blink_column | Optional quality columns. |
| interpolated_column | Optional quality columns. |
| max_interpolated_fraction | Maximum interpolated fraction per trial. |
| nuisance_by_participant | Whether nuisance residualization includes participant fixed effects. |
| include_response_time | Include response time in supported joint bridges. |
| ar1 | Include trial-wise AR(1) residual structure in Stan. |
| participant_effect | Include participant/item pupil effects. |
| item_effect | Include participant/item pupil effects. |

| | |
|---|---|
| chains | CmdStan controls. |
| parallel_chains | CmdStan controls. |
| iter_warmup | CmdStan controls. |
| iter_sampling | CmdStan controls. |
| adapt_delta | CmdStan sampler controls. |
| max_treedepth | CmdStan sampler controls. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

# gaze-survival *Censored Gaze-Latency Survival Analysis*

---

## Description

Vendor-neutral preparation, validation, descriptive estimation, Cox and accelerated failure-time modelling, diagnostics, sensitivity analysis, plotting, prediction, and reporting for eye-tracking latency outcomes with explicit right censoring.

## Usage

```
prepare_gaze_survival_data(
  trials, events = NULL, target_aoi = NULL, event_type = "first_fixation",
  participant_col = "participant_id", trial_col = "trial_id",
  recording_col = "recording_id", stimulus_col = "stimulus_id",
  condition_col = "condition", trial_start_col = NULL, trial_end_col = NULL,
  time_origin_col = NULL, observation_end_reason_col = NULL,
  event_time_col = "start_time", event_aoi_col = "aoi_id",
  event_trial_col = "trial_id", event_participant_col = "participant_id",
  event_recording_col = "recording_id", episode_type_col = "episode_type",
  event_observed_col = "event_observed", supplied_event_time_col = "event_time",
  supplied_censor_time_col = "censor_time", n_valid_samples_col = "n_valid_samples",
  valid_fraction_col = "valid_data_fraction", valid_observation_col = NULL,
  min_valid_fraction = NULL, time_origin = "trial_start",
  time_unit = c("seconds", "milliseconds"), source_data = NULL,
  preprocessing_specification = NULL, event_detector = NULL,
  aoi_specification = NULL, quality_rules = list()
)
validate_gaze_survival_data(data, raise_on_error = TRUE)
summarise_gaze_censoring(data, by = NULL)
estimate_gaze_survival(data, group = NULL, conf_level = 0.95)
fit_gaze_cox_model(data, formula, ties = c("breslow", "efron", "exact"), cluster = NULL)
fit_gaze_mixed_cox_model(
  data, formula, participant_col = "participant_id",
```

```
  structure = NULL, ties = c("breslow", "efron", "exact")
)
fit_gaze_aft_model(data, formula, distribution = NULL, ...)
tidy_gaze_survival_model(model, conf_level = 0.95)
check_gaze_proportional_hazards(model, transform = "km", alpha = 0.05)
compare_gaze_survival_models(...)
predict_gaze_survival(model, newdata, times)
estimate_gaze_latency_quantiles(
  object, probs = c(0.25, 0.5, 0.75), group = NULL,
  newdata = NULL
)
plot_gaze_survival_curve(data, group = NULL, ...)
plot_gaze_cumulative_incidence(data, group = NULL, ...)
plot_gaze_hazard(data, group = NULL, ...)
plot_gaze_cox_diagnostics(model, ...)
compare_gaze_survival_specifications(
  specifications, formula, model_families,
  participant_col = "participant_id", ties = "breslow"
)
report_gaze_survival_model(model, conf_level = 0.95)
simulate_gaze_survival_inputs(
  kind = c("disclosure", "verification"), seed = 20260918,
  n_participants = 36L, trials_per_participant = 3L
)
simulate_gaze_survival_example(
  kind = c("disclosure", "verification"), seed = 20260918,
  n_participants = 36L, trials_per_participant = 3L
)
```

## Arguments

| | |
|---|---|
| `trials` | Trial-level observation-window table. Each participant/trial key must be unique. |
| `events` | Optional fixation or AOI-visit event table. |
| `target_aoi` | Target AOI used to define the event of interest. |
| `event_type` | Event definition, including first fixation, first AOI entry, first evidence inspection, first revisit, first transition, or disengagement. |
| `participant_col`, `trial_col`, `recording_col`, `stimulus_col`, `condition_col` | Column names defining trial identity and design fields. |
| `trial_start_col`, `trial_end_col`, `time_origin_col`, `observation_end_reason_col` | Observation-window and time-origin columns. |
| `event_time_col`, `event_aoi_col`, `event_trial_col`, `event_participant_col`, `event_recording_col`, `episode_type_col` | Event-table mappings. |
| `event_observed_col`, `supplied_event_time_col`, `supplied_censor_time_col` | Columns used when event/censoring values are supplied directly. |

`n_valid_samples_col`, `valid_fraction_col`, `valid_observation_col`, `min_valid_fraction`
: Quality fields and explicit minimum-validity rule.

`time_origin`
: Named time origin recorded in the canonical table.

`time_unit`
: Input time unit. Units are never changed silently.

`source_data`, `preprocessing_specification`, `event_detector`, `aoi_specification`, `quality_rules`
: Provenance metadata retained on derived data.

`data`
: Canonical gaze-survival data.

`raise_on_error`
: Whether validation errors should stop execution.

`by`, `group`
: Optional grouping variable for descriptive summaries or curves.

`conf_level`
: Confidence level for intervals.

`formula`
: Right-hand-side model formula or formula string.

`ties`
: Cox tie-handling method.

`cluster`
: Optional cluster column for robust Cox uncertainty.

`structure`
: Required repeated-participant Cox structure: `"cluster_robust"` or `"frailty"`. No estimator is selected implicitly.

`distribution`
: Required AFT distribution: `"weibull"` or `"lognormal"`. No AFT family is selected implicitly.

`model`, `object`
: A fitted gaze-survival model or descriptive survival estimate.

`transform`, `alpha`
: Transformation and significance threshold used by the proportional-hazards diagnostic.

`newdata`, `times`, `probs`
: Prediction and latency-quantile inputs.

`specifications`
: Named collection of explicitly prepared alternative survival datasets.

`model_families`
: Explicit model families to fit across sensitivity branches.

`kind`
: Synthetic example family: disclosure inspection or evidence verification.

`seed`, `n_participants`, `trials_per_participant`
: Deterministic synthetic-example controls.

`...`
: Additional arguments passed to the selected backend or plotting method.

## Details

A missing target event is not automatically censoring. A row is right-censored only when the observation window is known and the trial is otherwise analyzable. Rows with incomplete or unusable gaze remain explicit review states. Cox models estimate hazard ratios; AFT models estimate multiplicative time effects. Participant-clustered Cox uncertainty and Gaussian frailty Cox models are distinct estimators and are never silently substituted. Information criteria from ordinary Cox partial likelihood, `coxme` penalized frailty likelihood, and AFT full likelihood are not treated as rank-comparable across likelihood bases.

## Value

Preparation returns a canonical trial-level data frame. Validation and summaries return data frames. Estimation/model helpers return survival/model objects that retain model specification and provenance. Plot helpers return their plotting objects. Synthetic helpers return deterministic raw trial/event inputs or prepared canonical data.

## See Also

survival::Surv, survival::coxph, survival::survreg

## Examples

```
raw <- simulate_gaze_survival_inputs(seed = 1, n_participants = 12, trials_per_participant = 3)
surv <- prepare_gaze_survival_data(
  raw$trials, raw$events,
  target_aoi = "disclosure",
  source_data = "synthetic disclosure example",
  event_detector = "synthetic fixation events",
  aoi_specification = "disclosure rectangle"
)
summarise_gaze_censoring(surv, by = "condition")
validate_gaze_survival_data(surv, raise_on_error = FALSE)

if (requireNamespace("survival", quietly = TRUE)) {
  km <- estimate_gaze_survival(surv, group = "condition")
  cox <- fit_gaze_mixed_cox_model(
    surv, ~ condition,
    participant_col = "participant_id",
    structure = "cluster_robust"
  )
  tidy_gaze_survival_model(cox)
}
```

---

# gazepoint_analysis_tables

*Build person-by-item-by-trial analysis tables*

---

## Description

Build person-by-item-by-trial analysis tables

## Usage

```
gazepoint_analysis_tables(x)
```

## Arguments

| | |
|---|---|
| x | A processed ‘eye_dataset‘ with workflow features. |

### Value

A named list containing trial, AOI, fixation, pupil, biometric, process, and feature-dictionary tables.

---

## gazepoint_irt_tables *Create IRT-ready response and process tables*

---

### Description

This function does not manufacture responses. When no responses have been supplied, it returns a complete response template and marks the outcome as process-ready but response-pending.

### Usage

```
gazepoint_irt_tables(x, process_table = NULL)
```

### Arguments

| | |
|---|---|
| x | A processed ‘eye_dataset‘. |
| process_table | Optional process table from ‘gazepoint_analysis_tables()‘. |

### Value

A list containing a response template, process covariates, long IRT table, optional matrices, and a readiness assessment.

---

## gazepoint_workflow_spec *Specify an integrated Gazepoint downstream workflow*

---

### Description

Creates a declarative specification for the end-to-end Gazepoint workflow. The defaults preserve vendor fixations, interpolate only short pupil gaps, apply a small median pupil filter, and avoid automatic pupil baseline correction when no pre-stimulus baseline is available.

### Usage

```
gazepoint_workflow_spec(
  expected_sampling_rate = 60,
  sampling_tolerance_hz = 5,
  minimum_valid_gaze = 0.8,
  minimum_valid_pupil = 0.7,
  pupil_interpolation = "linear",
  pupil_max_gap_ms = 150,
  pupil_filter = "median",
```

```
  pupil_window = 5L,
  pupil_baseline = "none",
  pupil_baseline_window = c(0, 0.5),
  detect_blinks = TRUE,
 biometric_channels = c("eda", "skin_conductance_level", "skin_conductance_response",
    "heart_rate", "interbeat_interval", "engagement_dial"),
  create_plots = TRUE,
  create_html_report = TRUE,
  retain_raw = TRUE
)
```

## Arguments

| | |
|---|---|
| expected_sampling_rate | Expected gaze sampling rate in hertz. |
| sampling_tolerance_hz | Allowed absolute sampling-rate deviation. |
| minimum_valid_gaze | Minimum acceptable valid-gaze fraction. |
| minimum_valid_pupil | Minimum acceptable valid-pupil fraction. |
| pupil_interpolation | Pupil interpolation method. |
| pupil_max_gap_ms | Maximum pupil gap eligible for interpolation. |
| pupil_filter | Pupil smoothing method. |
| pupil_window | Smoothing window in samples. |
| pupil_baseline | Baseline correction method. The default is ‘"none"‘. |
| pupil_baseline_window | Baseline window relative to media/trial onset. |
| detect_blinks | Whether to derive blink episodes from missing pupil data. |
| biometric_channels | Channels to retain in workflow plots and tables. |
| create_plots | Whether to create the complete plot suite. |
| create_html_report | Whether to render an HTML copy of the report when ‘rmarkdown‘ and Pandoc are available. |
| retain_raw | Whether imported native exports are retained in the object. |

## Value

An ‘eye_gazepoint_workflow_spec‘ object.

## gaze_anchored_3pl_alignment

*Return the process-alignment table from a gaze-anchored 3PL audit*

### Description

Return the process-alignment table from a gaze-anchored 3PL audit

### Usage

```
gaze_anchored_3pl_alignment(x)
```

### Arguments

| | |
|---|---|
| x | An ‘eye_gaze_anchored_3pl_audit‘. |

### Value

An R object containing return the process-alignment table from a gaze-anchored 3PL audit. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## gaze_data_quality_profile

*Empirical gaze data-quality profile*

### Description

Empirical gaze data-quality profile

### Usage

```
gaze_data_quality_profile(
  data,
  x = "gaze_x",
  y = "gaze_y",
  time = "timestamp_ms",
  target_x = NULL,
  target_y = NULL,
  valid = NULL,
  by = NULL,
  time_unit = c("ms", "s", "us")
)
```

### Arguments

| | |
|---|---|
| data | Gaze samples. |
| x, y | Gaze coordinates. |
| time | Timestamp column. |
| target_x, target_y | Optional known target coordinates. |
| valid | Optional validity indicator column. |
| by | Optional grouping columns. |
| time_unit | Timestamp unit. |

### Value

An object of class "eye_data_quality_profile", stored as a named list, with components "table", "coordinate_units", "caveat". It contains empirical gaze data-quality profile and associated metadata or diagnostics needed to interpret the result.

---

## gaze_diffusion_spec *Define a gaze-informed diffusion model*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
gaze_diffusion_spec(response = "score", response_time = "response_time",
  participant = "participant_id", item = "item_id", drift_features = character(),
  boundary_features = character(), nondecision_features = character(),
  starting_features = character(), censor_column = NULL, contaminant = TRUE,
  engine = c("baseline", "stan", "ez_regression", "diffIRT", "brms"),
  gaze_features = NULL, chains = 4L, parallel_chains = min(4L, chains),
 iter_warmup = 1000L, iter_sampling = 1000L, adapt_delta = 0.97, max_treedepth = 13L)
```

### Arguments

| | |
|---|---|
| response | Binary response column. |
| response_time | Response-time column in seconds. |
| participant | Participant identifier. |
| item | Item identifier. |
| drift_features | Features assigned a priori to drift rate. |
| boundary_features | Features assigned a priori to boundary separation. |

| | |
|---|---|
| nondecision_features | Features assigned a priori to non-decision time. |
| starting_features | Features assigned a priori to starting-point bias. |
| censor_column | Optional censoring column with ‘observed‘, ‘left‘, or ‘right‘. |
| contaminant | Whether to estimate a uniform contaminant mixture. |
| engine | Baseline approximation or Stan Wiener model. |
| gaze_features | Value for ‘gaze_features‘. See the function description and relevant article for constraints. |
| chains parallel_chains | Stan controls. |
| iter_warmup | Stan controls. |
| iter_sampling | Stan controls. |
| adapt_delta | Stan controls. |
| max_treedepth | Stan controls. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

# gaze_precision_rms_s2s

*Estimate RMS successive-sample gaze imprecision*

---

## Description

Estimate RMS successive-sample gaze imprecision

## Usage

```
gaze_precision_rms_s2s(
  data,
  x = "gaze_x",
  y = "gaze_y",
  time = NULL,
  by = NULL
)
```

## Arguments

| | |
|---|---|
| data | Gaze samples. |
| x, y | Gaze-coordinate columns. |
| time | Optional timestamp column used to order samples. |
| by | Optional grouping columns. |

### Value

A logical value or vector indicating rMS successive-sample gaze imprecision.

---

`gaze_recurrence` *Recurrence and cross-recurrence analysis*

---

### Description

Recurrence and cross-recurrence analysis. These functions form the eyeprocess 0.6.0.9000 measurement-intelligence programme and use dependency-free reference implementations with explicit evidence limits.

### Usage

```
gaze_recurrence(x, representation = c("coordinates", "aoi", "velocity"),
  x_col = "x", y_col = "y", aoi_col = "aoi", radius = NULL)
cross_recurrence(x, y, channels = c("gaze_pupil", "gaze_eda", "pupil_eda"),
  radius = NULL)
windowed_recurrence(x, window, step)
recurrence_features(x, minimum_line = 2L)
plot_recurrence_matrix(x, ...)
plot_windowed_recurrence(x, ...)
plot_diagonal_recurrence_profile(x, ...)
plot_crossmodal_recurrence(x, ...)
plot_recurrence_network(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | Input object or data structure appropriate for the selected analysis. |
| `representation` | Argument controlling 'representation'; see the function usage and returned audit metadata. |
| `x_col` | Argument controlling 'x_col'; see the function usage and returned audit metadata. |
| `y_col` | Argument controlling 'y_col'; see the function usage and returned audit metadata. |
| `aoi_col` | Argument controlling 'aoi_col'; see the function usage and returned audit metadata. |
| `radius` | Argument controlling 'radius'; see the function usage and returned audit metadata. |
| `y` | Argument controlling 'y'; see the function usage and returned audit metadata. |
| `channels` | Argument controlling 'channels'; see the function usage and returned audit metadata. |
| `window` | Argument controlling 'window'; see the function usage and returned audit metadata. |

| | |
|---|---|
| step | Argument controlling 'step'; see the function usage and returned audit metadata. |
| minimum_line | Argument controlling 'minimum_line'; see the function usage and returned audit metadata. |
| ... | Additional arguments passed to the underlying method or plotting function. |

## Details

The APIs return auditable S3 objects. Plot wrappers call registered base-graphics methods. Experimental or approximate engines are labelled in object status fields and should be validated before confirmatory or operational use.

## Value

An eyeprocess result object, data frame, model object, plot, or audit table as documented by the individual function.

## See Also

plot_diagnostics(), plot_evidence(), and plot_sensitivity().

---

# gaze_uncertainty_ellipse

*Uncertainty ellipse implied by an empirical calibration-error model*

---

## Description

Uncertainty ellipse implied by an empirical calibration-error model

## Usage

```
gaze_uncertainty_ellipse(model, level = 0.95, center = NULL)
```

## Arguments

| | |
|---|---|
| model | Calibration error model. |
| level | Probability level. |
| center | Optional center; defaults to model mean error. |

## Value

A data frame containing uncertainty ellipse implied by an empirical calibration-error model. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## generalizability_process_study *Generalizability-style variance decomposition for a process measure*

### Description

Useful before many-facet IRT: quantifies how much variance comes from person, item, device, session, algorithm, and residual sources.

### Usage

```
generalizability_process_study(data, outcome, facets, REML = TRUE)
```

### Arguments

| | |
|---|---|
| data | Input data frame or compatible tabular object. |
| outcome | Outcome variable. |
| facets | Facet variables included in the analysis. |
| REML | Whether restricted maximum likelihood is used. |

### Value

An object of class "eye_process_g_study", stored as a named list, with components "model", "variance_components", "facets", "outcome". It contains generalizability-style variance decomposition for a process measure and associated metadata or diagnostics needed to interpret the result.

## get_irt_model *Retrieve a registered multimodal IRT model*

### Description

Retrieve a registered multimodal IRT model

### Usage

```
get_irt_model(id)
```

### Arguments

| | |
|---|---|
| id | Stable identifier. |

### Value

An R object containing retrieve a registered multimodal IRT model. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## grade_model_evidence *Grade model evidence against an explicit validation contract*

### Description

This is intentionally conservative: failure of any required criterion caps the evidence grade. It does not convert exploratory evidence into a claim of substantive validity.

### Usage

```
grade_model_evidence(
  recovery,
  spec = irt_validation_spec("unspecified"),
  external_validation = NULL,
  sbc = NULL,
  ppc = NULL,
  semantic_roundtrip = NULL
)
```

### Arguments

| | |
|---|---|
| recovery | Value supplied to ‘recovery‘; see Details for its model-specific role. |
| spec | IRT model or validation specification. |
| external_validation | Value supplied to ‘external_validation‘; see Details for its model-specific role. |
| sbc | Value supplied to ‘sbc‘; see Details for its model-specific role. |
| ppc | Value supplied to ‘ppc‘; see Details for its model-specific role. |
| semantic_roundtrip | Value supplied to ‘semantic_roundtrip‘; see Details for its model-specific role. |

### Value

An object of class "eye_irt_evidence_grade", stored as a named list, with components "model_id", "grade", "checks", "recovery", "contract", "warning". It contains grade model evidence against an explicit validation contract and associated metadata or diagnostics needed to interpret the result.

## grouped_cv *Evaluate a model with grouped cross-validation*

### Description

Evaluate a model with grouped cross-validation

### Usage

```
grouped_cv(
  data,
  formula,
  family = stats::binomial(),
  group = "participant_id",
  v = 5L,
  metric = c("log_loss", "brier", "accuracy"),
  seed = 1L
)
```

### Arguments

| | |
|---|---|
| data | Data frame. |
| formula | Model formula. |
| family | GLM family. |
| group | Grouping columns. |
| v | Number of folds. |
| metric | Metric: log loss, Brier score, or accuracy. |
| seed | Random seed. |

### Value

An ‘eye_grouped_cv‘ object.

---

## grouped_folds *Create grouped cross-validation folds*

---

### Description

Create grouped cross-validation folds

### Usage

```
grouped_folds(data, group = c("participant_id"), v = 5L, seed = 1L)
```

### Arguments

| | |
|---|---|
| data | Data frame. |
| group | Grouping columns that must not cross folds. |
| v | Number of folds. |
| seed | Random seed. |

### Value

An ‘eye_grouped_folds‘ object.

## import_benchmark_study

*Build an eye dataset from the public benchmark*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
import_benchmark_study(study = eyeprocess_benchmark_study())
```

### Arguments

| | |
|---|---|
| study | Benchmark object or path. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## import_eye_bids

*Import Eye-Tracking-BIDS physiological recordings*

### Description

Import Eye-Tracking-BIDS physiological recordings

### Usage

```
import_eye_bids(path, validate = TRUE)
```

### Arguments

| | |
|---|---|
| path | BIDS dataset root. |
| validate | Whether to validate the resulting canonical dataset. |

### Value

An ‘eye_dataset‘.

## incremental_process_validity *Extract incremental process validity*

### Description

Extract incremental process validity

### Usage

```
incremental_process_validity(x)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |

### Value

A data frame containing incremental process validity. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## init_vendor_corpus *Create or validate a multi-vendor corpus directory*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
init_vendor_corpus(path, overwrite = FALSE)
```

### Arguments

| | |
|---|---|
| path | Corpus directory. |
| overwrite | Reinitialize an existing empty corpus. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## inject_aoi_label_noise

*Inject AOI label noise*

### Description

Inject AOI label noise

### Usage

```
inject_aoi_label_noise(data, aoi = "aoi", proportion, seed = 1L)
```

### Arguments

| | |
|---|---|
| data | Data. |
| aoi | AOI label column. |
| proportion | Proportion reassigned to another observed label. |
| seed | Seed. |

### Value

An R object containing inject AOI label noise. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## inject_calibration_offset

*Inject additive gaze calibration offset in coordinate units*

### Description

Inject additive gaze calibration offset in coordinate units

### Usage

```
inject_calibration_offset(
  data,
  x = "gaze_x",
  y = "gaze_y",
  offset_x = 0,
  offset_y = 0
)
```

### Arguments

| | |
|---|---|
| `data` | Data. |
| `x, y` | Gaze coordinate columns. |
| `offset_x, offset_y` | Additive offsets in the same units as x/y. |

### Value

An R object containing inject additive gaze calibration offset in coordinate units. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

## `inject_device_shift` *Inject an additive device/site shift in a numeric feature*

---

### Description

Inject an additive device/site shift in a numeric feature

### Usage

```
inject_device_shift(data, column, shift, rows = NULL)
```

### Arguments

| | |
|---|---|
| `data` | Data. |
| `column` | Numeric column. |
| `shift` | Additive shift. |
| `rows` | Optional logical/index rows affected. |

### Value

An R object containing inject an additive device/site shift in a numeric feature. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## inject_eye_missingness *Inject generic missingness into selected columns*

### Description

Inject generic missingness into selected columns

### Usage

```
inject_eye_missingness(data, columns, proportion, seed = 1L)
```

### Arguments

| | |
|---|---|
| data | Data. |
| columns | Columns. |
| proportion | Missingness proportion. |
| seed | Seed. |

### Value

An R object containing inject generic missingness into selected columns. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## inject_pupil_dropout *Inject pupil dropout*

### Description

Inject pupil dropout

### Usage

```
inject_pupil_dropout(data, pupil = "pupil", proportion, seed = 1L)
```

### Arguments

| | |
|---|---|
| data | Data. |
| pupil | Pupil column. |
| proportion | Dropout proportion. |
| seed | Seed. |

### Value

An R object containing inject pupil dropout. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

`inject_sampling_jitter` *Inject timestamp jitter*

## Description

Inject timestamp jitter

## Usage

```
inject_sampling_jitter(data, time = "timestamp_ms", sd, seed = 1L)
```

## Arguments

| | |
|---|---|
| `data` | Data. |
| `time` | Timestamp column. |
| `sd` | Jitter standard deviation in timestamp units. |
| `seed` | Seed. |

## Value

An R object containing inject timestamp jitter. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

`inject_trial_imbalance` *Inject trial/row imbalance by dropping observations*

## Description

Inject trial/row imbalance by dropping observations

## Usage

```
inject_trial_imbalance(data, proportion, seed = 1L)
```

## Arguments

| | |
|---|---|
| `data` | Data. |
| `proportion` | Proportion dropped. |
| `seed` | Seed. |

## Value

A tabular R object containing inject trial/row imbalance by dropping observations; rows represent analysis units and columns contain the returned quantities.

## irt_compositional_channel

*Compositional AOI channel*

### Description

Compositional AOI channel

### Usage

```
irt_compositional_channel(
  parts,
  family = c("logratio_gaussian", "dirichlet"),
  latent = "process",
  options = list()
)
```

### Arguments

| | |
|---|---|
| parts | Compositional parts. |
| family | Statistical family used by the channel or model. |
| latent | Latent variable or latent-variable labels. |
| options | Additional channel/model options. |

### Value

A named list with components "type", "family", "role", "link", "variables", "latent", "options", containing compositional AOI channel and associated metadata or diagnostics.

## irt_continuous_channel

*Continuous/bounded process channel for multimodal IRT*

### Description

Continuous/bounded process channel for multimodal IRT

### Usage

```
irt_continuous_channel(
  family = c("censored_normal", "beta", "gaussian"),
  value = "process_value",
  lower = 0,
  upper = 1,
  latent = "process",
  options = list()
)
```

### Arguments

| | |
|---|---|
| family | Continuous response family. |
| value | Variable name. |
| lower, upper | Bounds where applicable. |
| latent | Latent dimension. |
| options | Additional channel metadata. |

### Value

A named list with components "type", "family", "role", "link", "variables", "latent", "options", containing continuous/bounded process channel for multimodal IRT and associated metadata or diagnostics.

---

## irt_count_channel *Count-valued process channel for multimodal IRT*

---

### Description

Count-valued process channel for multimodal IRT

### Usage

```
irt_count_channel(
  family = c("negative_binomial", "poisson"),
  value = "fixation_count",
  latent = "engagement",
  options = list()
)
```

### Arguments

| | |
|---|---|
| family | Statistical family used by the channel or model. |
| value | Process-value column or values. |
| latent | Latent variable or latent-variable labels. |
| options | Additional channel/model options. |

### Value

A named list with components "type", "family", "role", "link", "variables", "latent", "options", containing count-valued process channel for multimodal IRT and associated metadata or diagnostics.

## irt_functional_channel *Functional trajectory channel*

### Description

Functional trajectory channel

### Usage

```
irt_functional_channel(
  value = "pupil",
  time = "time",
  family = c("basis_gaussian", "functional_factor"),
  latent = "process",
  options = list()
)
```

### Arguments

| | |
|---|---|
| value | Process-value column or values. |
| time | Time values. |
| family | Statistical family used by the channel or model. |
| latent | Latent variable or latent-variable labels. |
| options | Additional channel/model options. |

### Value

A named list with components "type", "family", "role", "link", "variables", "latent", "options", containing functional trajectory channel and associated metadata or diagnostics.

## irt_model_spec *Define a multimodal IRT model specification*

### Description

Define a multimodal IRT model specification

## Usage

```
irt_model_spec(
  id,
  latent,
  channels,
  status = c("experimental", "reference", "gated"),
  fit_fun = NULL,
  simulate_fun = NULL,
  validate_fun = NULL,
  citation = character(),
  description = NULL,
  requirements = character(),
  metadata = list()
)
```

## Arguments

| | |
|---|---|
| id | Stable model identifier. |
| latent | Named or unnamed latent dimensions. |
| channels | Named list of channel objects. |
| status | One of reference, experimental, or gated. |
| fit_fun | Optional fitting function. |
| simulate_fun | Optional simulator. |
| validate_fun | Optional model-specific validation function. |
| citation | Character vector of citations/DOIs. |
| description | Human-readable model description. |
| requirements | Optional packages/engines. |
| metadata | Additional metadata. |

## Value

An ‘eye_irt_model_spec‘.

---

# irt_nominal_channel *Nominal response/process channel*

---

## Description

Nominal response/process channel

### Usage

```
irt_nominal_channel(
  choice = "response_option",
  categories = NULL,
  latent = "ability",
  options = list()
)
```

### Arguments

| | |
|---|---|
| choice | Observed nominal choice. |
| categories | Nominal response categories. |
| latent | Latent variable or latent-variable labels. |
| options | Additional channel/model options. |

### Value

A named list with components "type", "family", "role", "link", "variables", "latent", "options", containing nominal response/process channel and associated metadata or diagnostics.

---

## irt_response_channel *Binary/ordinal response channel for multimodal IRT*

---

### Description

Binary/ordinal response channel for multimodal IRT

### Usage

```
irt_response_channel(
  family = c("2pl", "rasch", "graded", "partial_credit"),
  response = "response",
  latent = "ability",
  options = list()
)
```

### Arguments

| | |
|---|---|
| family | Statistical family used by the channel or model. |
| response | Response variable or response-column name. |
| latent | Latent variable or latent-variable labels. |
| options | Additional channel/model options. |

### Value

A named list with components "type", "family", "role", "link", "variables", "latent", "options", containing binary/ordinal response channel for multimodal IRT and associated metadata or diagnostics.

## irt_rt_channel *Response-time channel for multimodal IRT*

### Description

Response-time channel for multimodal IRT

### Usage

```
irt_rt_channel(
  family = c("lognormal", "gaussian_log", "shifted_lognormal"),
  rt = "rt",
  latent = "speed",
  options = list()
)
```

### Arguments

| | |
|---|---|
| family | Statistical family used by the channel or model. |
| rt | Response-time variable or column name. |
| latent | Latent variable or latent-variable labels. |
| options | Additional channel/model options. |

### Value

A named list with components "type", "family", "role", "link", "variables", "latent", "options", containing response-time channel for multimodal IRT and associated metadata or diagnostics.

## irt_sequence_channel *Sequence/process-state channel*

### Description

Sequence/process-state channel

### Usage

```
irt_sequence_channel(
  sequence = "sequence",
  family = c("ngram", "hmm", "embedding"),
  latent = "strategy",
  options = list()
)
```

### Arguments

| | |
|---|---|
| sequence | Sequence input. |
| family | Statistical family used by the channel or model. |
| latent | Latent variable or latent-variable labels. |
| options | Additional channel/model options. |

### Value

A named list with components "type", "family", "role", "link", "variables", "latent", "options", containing sequence/process-state channel and associated metadata or diagnostics.

---

## irt_survival_channel *Survival/event-time channel for multimodal IRT*

---

### Description

Survival/event-time channel for multimodal IRT

### Usage

```
irt_survival_channel(
  family = c("cox", "weibull", "exponential"),
  time = "time",
  event = "event",
  latent = NULL,
  options = list()
)
```

### Arguments

| | |
|---|---|
| family | Statistical family used by the channel or model. |
| time | Time values. |
| event | Event indicator or event column. |
| latent | Latent variable or latent-variable labels. |
| options | Additional channel/model options. |

### Value

A named list with components "type", "family", "role", "link", "variables", "latent", "options", containing survival/event-time channel for multimodal IRT and associated metadata or diagnostics.

`irt_validation_spec` *Specify a validation programme for a process-IRT model*

## Description

Creates an evidence contract rather than executing a particular estimator. The object can be passed to stress-test and evidence-grading helpers.

## Usage

```
irt_validation_spec(
  model_id,
  replications = 250L,
  parameters = NULL,
  metrics = c("bias", "rmse", "coverage", "interval_width", "convergence"),
  grouped_validation = c("device", "session", "site"),
  preprocessing_variants = NULL,
  misspecification_scenarios = NULL,
  thresholds = list(max_abs_bias = 0.1, max_rmse = 0.3, min_coverage = 0.9,
    max_failure_rate = 0.05, min_external_folds = 2L),
  seed = 20260808L,
  notes = NULL
)
```

## Arguments

| | |
|---|---|
| `model_id` | Stable model identifier. |
| `replications` | Number of simulation replications planned. |
| `parameters` | Parameter families expected to be recovered. |
| `metrics` | Recovery metrics to require. |
| `grouped_validation` | Grouping variables for transport validation. |
| `preprocessing_variants` | Optional named preprocessing variants. |
| `misspecification_scenarios` | Optional named scenarios. |
| `thresholds` | Named evidence thresholds. |
| `seed` | Reproducibility seed. |
| `notes` | Free-text scientific notes. |

## Value

An object of class "eye_irt_validation_spec", stored as a named list, with components "model_id", "replications", "parameters", "metrics", "grouped_validation", "preprocessing_variants", "misspecification_scenarios", "thresholds", "seed", "notes", "contract_version", "created_with". It contains specify a validation programme for a process-IRT model and associated metadata or diagnostics needed to interpret the result.

---

`item_objective_spec` *Multi-objective item-bank optimization*

---

## Description

Multi-objective item-bank optimization. These functions form the eyeprocess 0.6.0.9000 measurement-intelligence programme and use dependency-free reference implementations with explicit evidence limits.

## Usage

```
item_objective_spec(information, process_burden, fairness, exposure,
  content_constraints = NULL, weights = c(information = 1, process_burden = 1,
  fairness = 1, exposure = 1), directions = c(information = "max",
  process_burden = "min", fairness = "min", exposure = "min"))
item_pareto_front(x, objectives)
optimize_item_bank(x, n_items, objectives, constraints = NULL, method = c("integer",
  "evolutionary"), iterations = 500, seed = 20260807)
audit_bank_decision_stability(x, draws = 1000, noise_sd = 0.1, seed = 20260807)
plot_item_pareto(x, ...)
plot_objective_tradeoffs(x, ...)
plot_bank_information_coverage(x, ...)
plot_decision_stability(x, ...)
plot_selected_bank_profile(x, ...)
```

## Arguments

| | |
|---|---|
| `information` | Argument controlling ‘information‘; see the function usage and returned audit metadata. |
| `process_burden` | Argument controlling ‘process_burden‘; see the function usage and returned audit metadata. |
| `fairness` | Argument controlling ‘fairness‘; see the function usage and returned audit metadata. |
| `exposure` | Argument controlling ‘exposure‘; see the function usage and returned audit metadata. |
| `content_constraints` | Argument controlling ‘content_constraints‘; see the function usage and returned audit metadata. |

| | |
|---|---|
| weights | Argument controlling ‘weights‘; see the function usage and returned audit metadata. |
| directions | Argument controlling ‘directions‘; see the function usage and returned audit metadata. |
| x | Input object or data structure appropriate for the selected analysis. |
| objectives | Argument controlling ‘objectives‘; see the function usage and returned audit metadata. |
| n_items | Argument controlling ‘n_items‘; see the function usage and returned audit metadata. |
| constraints | Argument controlling ‘constraints‘; see the function usage and returned audit metadata. |
| method | Argument controlling ‘method‘; see the function usage and returned audit metadata. |
| iterations | Argument controlling ‘iterations‘; see the function usage and returned audit metadata. |
| seed | Argument controlling ‘seed‘; see the function usage and returned audit metadata. |
| draws | Argument controlling ‘draws‘; see the function usage and returned audit metadata. |
| noise_sd | Argument controlling ‘noise_sd‘; see the function usage and returned audit metadata. |
| ... | Additional arguments passed to the underlying method or plotting function. |

## Details

The APIs return auditable S3 objects. Plot wrappers call registered base-graphics methods. Experimental or approximate engines are labelled in object status fields and should be validated before confirmatory or operational use.

## Value

An eyeprocess result object, data frame, model object, plot, or audit table as documented by the individual function.

## See Also

`plot_diagnostics()`, `plot_evidence()`, and `plot_sensitivity()`.

## latent_distribution_stress_test

*Stress-test IRT estimators across latent distributions*

### Description

Public roadmap alias for ‘stress_test_latent_distribution()‘.

### Usage

```
latent_distribution_stress_test(...)
```

### Arguments

| | |
|---|---|
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

An R object containing stress-test IRT estimators across latent distributions. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## latent_trait_trajectory

*Estimate a descriptive continuous-time latent trajectory*

### Description

Estimate a descriptive continuous-time latent trajectory

### Usage

```
latent_trait_trajectory(time, theta, spar = NULL)
```

### Arguments

| | |
|---|---|
| time | Time values. |
| theta | Latent-trait values. |
| spar | Value supplied to ‘spar‘; see Details for its model-specific role. |

### Value

An object of class "eye_latent_trait_trajectory", stored as a named list, with components "model", "time", "theta". It contains a descriptive continuous-time latent trajectory and associated metadata or diagnostics needed to interpret the result.

## leave_device_out_validation

*Leave device out validation*

### Description

Leave device out validation

### Usage

```
leave_device_out_validation(data, device, fitter, predictor, scorer)
```

### Arguments

| | |
|---|---|
| data | Input data frame or compatible tabular object. |
| device | Device identifier or device facet. |
| fitter | Model-fitting function. |
| predictor | Prediction function. |
| scorer | Function that scores predictions. |

### Value

An object of class "eye_leave_device_out_validation", "data.frame", stored as a data frame, containing leave device out validation and associated metadata needed to interpret the result.

## leave_item_out_validation

*Leave item out validation*

### Description

Leave item out validation

### Usage

```
leave_item_out_validation(data, item, fitter, predictor, scorer)
```

### Arguments

| | |
|---|---|
| data | Input data frame or compatible tabular object. |
| item | Item identifier, name, or item column. |
| fitter | Model-fitting function. |
| predictor | Prediction function. |
| scorer | Function that scores predictions. |

## Value

An object of class "eye_leave_item_out_validation", "data.frame", stored as a data frame, containing leave item out validation and associated metadata needed to interpret the result.

---

`leave_session_out_validation`

*Leave session out validation*

---

## Description

Leave session out validation

## Usage

```
leave_session_out_validation(data, session, fitter, predictor, scorer)
```

## Arguments

| | |
|---|---|
| `data` | Input data frame or compatible tabular object. |
| `session` | Session identifier or session facet. |
| `fitter` | Model-fitting function. |
| `predictor` | Prediction function. |
| `scorer` | Function that scores predictions. |

## Value

An object of class "eye_leave_session_out_validation", "data.frame", stored as a data frame, containing leave session out validation and associated metadata needed to interpret the result.

---

`leave_site_out_validation`

*Leave site out validation*

---

## Description

Leave site out validation

## Usage

```
leave_site_out_validation(data, site, fitter, predictor, scorer)
```

### Arguments

| | |
|---|---|
| data | Input data frame or compatible tabular object. |
| site | Site identifier or site facet. |
| fitter | Model-fitting function. |
| predictor | Prediction function. |
| scorer | Function that scores predictions. |

### Value

An object of class "eye_leave_site_out_validation", "data.frame", stored as a data frame, containing leave site out validation and associated metadata needed to interpret the result.

---

## list_irt_models *List registered multimodal IRT models*

---

### Description

List registered multimodal IRT models

### Usage

```
list_irt_models()
```

### Value

A tabular R object containing list registered multimodal IRT models; rows represent analysis units and columns contain the returned quantities.

---

## lock_decision_manifest *Lock a decision manifest by content hash*

---

### Description

Lock a decision manifest by content hash

### Usage

```
lock_decision_manifest(x, label = "analysis_decisions")
```

### Arguments

| | |
|---|---|
| x | Manifest. |
| label | Optional lock label. |

## Value

An object of class "eye_decision_manifest_lock", stored as a named list, with components "manifest", "manifest_hash", "label", "locked_at", "status". It contains lock a decision manifest by content hash and associated metadata or diagnostics needed to interpret the result.

---

`map_latent_classes_to_process_profiles`
*Map supplied latent-class memberships to process summaries*

---

## Description

Map supplied latent-class memberships to process summaries

## Usage

```
map_latent_classes_to_process_profiles(
  class_membership,
  process_data,
  person = "person_id",
  class_col = "class",
  process_features
)
```

## Arguments

| | |
|---|---|
| `class_membership` | Data containing person/class assignments or probabilities. |
| `process_data` | Person-level or trial-level process data. |
| `person` | Person identifier present in both objects. |
| `class_col` | Class-assignment column. |
| `process_features` | Numeric process features. |

## Value

An object of class "eye_latent_process_alignment", stored as a named list, with components "data", "summary", "process_features", "class_col", "caveat". It contains supplied latent-class memberships to process summaries and associated metadata or diagnostics needed to interpret the result.

`measurement_error_budget` *Build a non-collapsed measurement-error budget*

## Description

Build a non-collapsed measurement-error budget

## Usage

```
measurement_error_budget(
  accuracy = NA_real_,
  precision = NA_real_,
  data_loss = NA_real_,
  effective_hz = NA_real_,
  calibration_drift = NA_real_,
  units = NULL
)
```

## Arguments

| | |
|---|---|
| `accuracy` | Measurement inaccuracy/offset metric. |
| `precision` | Measurement imprecision metric. |
| `data_loss` | Data-loss proportion. |
| `effective_hz` | Effective sampling frequency. |
| `calibration_drift` | Optional drift metric. |
| `units` | Optional named units. |

## Value

A data frame containing a non-collapsed measurement-error budget. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

`migrate_eye_storage_schema` *Migrate a storage schema through an atomic rewrite*

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
migrate_eye_storage_schema(storage, target_path, target_version = "2.0.0",
  format = NULL, overwrite = FALSE)
```

### Arguments

| | |
|---|---|
| storage | Storage object or path. |
| target_path | Destination. |
| target_version | Target schema version. |
| format | Target format. |
| overwrite | Whether to replace target. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

## model_promotion_spec *Specify evidence gates for model promotion*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
model_promotion_spec(model_families = c("dynamic_irtree", "functional_pupil_irt",
  "theory_strategy_irt", "gaze_diffusion_irt"), require_completion = TRUE,
  require_sbc = TRUE, require_misspecification = TRUE,
  require_grouped_validation = TRUE, require_engine_equivalence = TRUE,
 require_empirical_reproduction = TRUE, require_preprocessing_sensitivity = TRUE,
  require_multi_vendor = FALSE)
```

### Arguments

| | |
|---|---|
| model_families | Model families to audit. |
| require_completion | Require complete Monte Carlo evidence. |
| require_sbc | Require simulation-based calibration. |
| require_misspecification | Require declared misspecification studies. |
| require_grouped_validation | Require grouped out-of-sample validation. |
| require_engine_equivalence | Require external-engine comparison. |

require_empirical_reproduction
: Require an empirical reproduction.

require_preprocessing_sensitivity
: Require preprocessing/AOI sensitivity.

require_multi_vendor
: Require independent multi-vendor evidence.

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

# model_validation_spec *Specify a model-validation programme*

---

## Description

Specify a model-validation programme

## Usage

```
model_validation_spec(
  replications = 100L,
  confidence = 0.95,
  max_abs_bias = 0.1,
  min_coverage = 0.9,
  max_failure_rate = 0.05
)
```

## Arguments

| | |
|---|---|
| replications | Number of Monte Carlo replications. |
| confidence | Confidence level for interval coverage. |
| max_abs_bias | Maximum acceptable absolute bias. |
| min_coverage | Minimum acceptable interval coverage. |
| max_failure_rate | Maximum acceptable estimation failure rate. |

## Value

An ‘eye_model_validation_spec‘.

`model_validation_summary` *Summarize model validation*

## Description

Summarize model validation

## Usage

```
model_validation_summary(x)
```

## Arguments

| | |
|---|---|
| `x` | An ‘eye_model_validation‘ object. |

## Value

Scenario-by-parameter validation metrics.

`multiblock_contributions` *Extract multiblock block contributions/coordinates*

## Description

Extract multiblock block contributions/coordinates

## Usage

```
multiblock_contributions(x)
```

## Arguments

| | |
|---|---|
| `x` | Object to process, inspect, compare, or plot. |

## Value

An R object containing multiblock block contributions/coordinates. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

`multiblock_person_coordinates` *Extract multiblock person coordinates*

## Description

Extract multiblock person coordinates

## Usage

```
multiblock_person_coordinates(x)
```

## Arguments

| | |
|---|---|
| `x` | Object to process, inspect, compare, or plot. |

## Value

An R object containing multiblock person coordinates. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

`multiblock_variable_coordinates` *Extract multiblock variable coordinates*

## Description

Extract multiblock variable coordinates

## Usage

```
multiblock_variable_coordinates(x)
```

## Arguments

| | |
|---|---|
| `x` | Object to process, inspect, compare, or plot. |

## Value

An R object containing multiblock variable coordinates. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

`multilevel-mediation` *Trial-level multilevel mediation preparation*

## Description

Vendor-neutral preparation and auditing utilities for repeated-measures mediation. The functions preserve trial-level rows, separate within- and between-participant variation, distinguish observed zero mediators from unobserved mediators, and retain provenance needed by downstream specialist estimators.

## Usage

```
center_within_participant(
  data, value_col, participant_col = "participant_id", output_col = NULL
)

decompose_within_between(
  data, columns, participant_col = "participant_id",
  grand_mean_center_between = FALSE
)

summarise_within_between_variance(
  data, columns, participant_col = "participant_id"
)

identify_mediation_levels(
  data, columns, participant_col = "participant_id", tolerance = 1e-12
)

audit_mediation_missingness(
  data, x_col, mediator_col, outcome_col,
  participant_col = "participant_id", trial_col = "trial_id",
  quality_col = NULL, minimum_quality = NULL,
  mediator_observed_col = NULL, response_observed_col = NULL
)

check_mediation_trial_counts(
  data, participant_col = "participant_id", trial_col = "trial_id",
  minimum_trials = 2L
)

validate_multilevel_mediation_data(
  data, x_col, mediator_col, outcome_col,
  participant_col = "participant_id", trial_col = "trial_id",
  require_within_x = TRUE
)
```

```
prepare_multilevel_mediation_data(
  data, x_col, mediator_col, outcome_col,
  participant_col = "participant_id", trial_col = "trial_id",
  quality_col = NULL, minimum_quality = NULL,
  mediator_observed_col = NULL, response_observed_col = NULL,
  quality_action = c("flag", "mask_mediator"),
  grand_mean_center_between = FALSE, require_within_x = TRUE,
  source_id = NULL, preprocessing_spec = NULL, event_detector = NULL,
  aoi_specification = NULL, warn = TRUE
)

add_multilevel_mediation_component(
  prepared, value_col, semantic, within_col, between_col,
  grand_mean_center_between = NULL
)
```

## Arguments

| | |
|---|---|
| `data` | Repeated-measures trial-level data frame. Rows are preserved. |
| `value_col` | Numeric variable to decompose or add as a mediation component. |
| `columns` | One or more numeric variables to inspect or decompose. |
| `participant_col` | Participant identifier column. |
| `trial_col` | Trial identifier column. |
| `output_col` | Optional name for a within-participant centered variable. |
| `grand_mean_center_between` | Whether participant means are centered around the observed grand mean. |
| `tolerance` | Non-negative tolerance for classifying within/between variation. |
| `x_col` | Exposure or manipulation column. |
| `mediator_col` | Trial-level mediator column. |
| `outcome_col` | Trial-level outcome column. |
| `quality_col` | Optional numeric trial-quality column. |
| `minimum_quality` | Required finite threshold when `quality_col` is supplied. |
| `mediator_observed_col` | Optional explicit marker that the mediator was observed. |
| `response_observed_col` | Optional explicit marker that the outcome was observed. |
| `minimum_trials` | Minimum trial count used for the descriptive participant audit. |
| `require_within_x` | Whether preparation requires within-participant exposure variation. |
| `quality_action` | Either `"flag"` or `"mask_mediator"`; no quality action is selected silently. |
| `source_id` | Optional source-data identifier retained in provenance. |

preprocessing_spec, event_detector, aoi_specification
: Optional preprocessing, event-detector, and AOI provenance.

warn
: Whether preparation emits informative warnings stored in the returned object.

prepared
: An eye_multilevel_mediation_data object.

semantic
: Semantic name used to register an additional mediation component.

within_col, between_col
: Explicit output columns for an added within/between component.

## Details

The preparation layer does not fit a mediation estimator and does not convert missing mediators to zero. Quality rules are explicit. When quality_action = "mask_mediator", masking occurs before decomposition and is recorded in provenance. Within- and between-participant components are kept separate so a downstream multilevel estimator can model the intended level rather than conflating trial and person variation.

## Value

Centering and decomposition functions return data frames. Audit functions return data frames or validation objects. prepare_multilevel_mediation_data() returns an eye_multilevel_mediation_data object containing the preserved trial table, audits, column mappings, and provenance.

## Examples

```
d <- data.frame(
  participant_id = rep(c("P1", "P2"), each = 3),
  trial_id = paste0("T", 1:6),
  condition = c(0, 1, 0, 1, 0, 1),
  dwell = c(1.0, 1.4, 1.2, 0.8, 1.1, 1.3),
  outcome = c(3, 4, 3, 2, 3, 4)
)

x <- prepare_multilevel_mediation_data(
  d,
  x_col = "condition",
  mediator_col = "dwell",
  outcome_col = "outcome"
)

head(x$data[c("X_within", "X_between", "M_within", "M_between")])
```

---

# multimodal_backend_status

*Report multimodal backend availability*

---

## Description

Report multimodal backend availability

## Usage

```
multimodal_backend_status()
```

## Value

A data.frame describing optional backend availability.

---

`multimodal_irt_spec` *Consolidated multimodal IRT specification*

---

## Description

This is a thin multimodal adapter over the established ‘irt_model_spec()‘ / ‘eye_irt_model_spec‘ architecture. It composes existing ‘eye_irt_channel‘ objects and does not define a parallel channel or model-specification ecosystem.

## Usage

```
multimodal_irt_spec(
  response = NULL,
  rt = NULL,
  gaze = NULL,
  pupil = NULL,
  model = c("M0", "M1", "M2", "M3"),
  backend = c("cmdstanr", "existing"),
  identification = list(),
  priors = list()
)
```

## Arguments

| | |
|---|---|
| `response`, `rt`, `gaze`, `pupil` | Existing ‘eye_irt_channel‘ objects or NULL. |
| `model` | Development model identifier. |
| `backend` | Requested backend. |
| `identification` | Named identification settings retained as explicit multimodal metadata. |
| `priors` | Named prior settings retained as explicit multimodal metadata. |

## Value

An ‘eye_multimodal_irt_spec‘ convenience subclass of the established ‘eye_irt_model_spec‘.

## multimodal_m2_ablation

*Fit M0, M1, and M2 as a response-target ablation sequence*

### Description

Fits response-only (M0), response+RT (M1), and response+RT+gaze (M2) with compatible hierarchical Stan implementations. The sequence is designed for response-target comparison rather than for asserting that information from distinct channels is algebraically additive.

### Usage

```
multimodal_m2_ablation(x, ...)
```

### Arguments

| | |
|---|---|
| x | Data accepted by [fit_multimodal_m2()]. |
| ... | Sampling arguments forwarded to the internal reference fitters. |

### Value

An 'eye_multimodal_m2_ablation'.

## multimodal_m2_negative_controls

*Generate M2 alignment negative controls*

### Description

Generates deterministic within-item permutations that preserve each item's marginal channel distribution while breaking person-level alignment for gaze, RT, or response. These are falsification controls, not causal interventions and not misconduct detectors.

### Usage

```
multimodal_m2_negative_controls(x, seed = 20260814L)
```

### Arguments

| | |
|---|---|
| x | M2-compatible data. |
| seed | Seed for deterministic permutations. |

### Value

An 'eye_multimodal_m2_negative_controls'.

## multimodal_m2_ppc *Posterior predictive checks for the M2 three-way model*

### Description

Reproduces the channel-specific discrepancy logic used in the published three-way framework: response W, response-time L, and fixation-count M residual statistics, aggregated by item.

### Usage

```
multimodal_m2_ppc(x)
```

### Arguments

| | |
|---|---|
| x | An ‘eye_multimodal_m2_fit‘ with ‘model == "M2"‘. |

### Value

An ‘eye_multimodal_m2_ppc‘.

## multimodal_m2_process_information *Quantify response-target process information in the M0-M2 sequence*

### Description

Computes two complementary quantities on a common response target: response-target PSIS-LOO ELPD and posterior variance of person ability. This avoids simply summing channel Fisher information under a joint correlated model.

### Usage

```
multimodal_m2_process_information(x)
```

### Arguments

| | |
|---|---|
| x | An ‘eye_multimodal_m2_ablation‘. |

### Value

An ‘eye_multimodal_m2_information‘.

`multimodal_m2_recovery` *Run repeated M2 estimator recovery*

## Description

Repeatedly simulates from the M2 generating model, fits the M2 CmdStan reference estimator, and computes bias, RMSE, posterior standard deviation, and 95 dispersion, correlation, and hyperparameter families.

## Usage

```
multimodal_m2_recovery(
  n_rep = 10L,
  n_person = 100L,
  n_item = 10L,
  dropout = c(response = 0, rt = 0, gaze = 0),
  base_seed = 20260814L,
  chains = 4L,
  parallel_chains = chains,
  iter_warmup = 1000L,
  iter_sampling = 1000L,
  prior_profile = c("regularized", "paper_centered"),
  adapt_delta = 0.95,
  max_treedepth = 12L,
  refresh = 0L
)
```

## Arguments

| | |
|---|---|
| `n_rep` | Number of simulation/fit replications. |
| `n_person`, `n_item` | Simulation size. |
| `dropout` | Channel dropout probabilities. |
| `base_seed` | Base seed. |
| `chains`, `parallel_chains`, `iter_warmup`, `iter_sampling` | CmdStan controls. |
| `prior_profile` | Prior profile. |
| `adapt_delta`, `max_treedepth`, `refresh` | CmdStan controls. |

## Details

This is intentionally computationally expensive and is not executed during ordinary package tests.

### Value

An ‘eye_multimodal_m2_recovery‘.

---

`multimodal_m2_spec` *M2 response + RT + gaze reference specification*

---

### Description

Creates a thin M2 specialization of the established [multimodal_irt_spec()] / ‘irt_model_spec()‘ architecture. The measurement likelihood follows Man, Harring, and Zhan (2022): Rasch response, lognormal response time, and negative-binomial gaze-fixation counts. Priors are implemented in Stan using either a regularized profile or a paper-centered profile; the latter is not claimed to reproduce every published hyperprior exactly.

### Usage

```
multimodal_m2_spec(
  backend = "cmdstanr",
  prior_profile = c("regularized", "paper_centered"),
  missingness = "ignorable"
)
```

### Arguments

| | |
|---|---|
| `backend` | Currently ‘"cmdstanr"‘ only. |
| `prior_profile` | ‘"regularized"‘ or ‘"paper_centered"‘. |
| `missingness` | Currently ‘"ignorable"‘ only. Channel-specific missing observations contribute no level-1 likelihood term. |

### Value

An ‘eye_multimodal_m2_spec‘, inheriting the established ‘eye_multimodal_irt_spec‘ and ‘eye_irt_model_spec‘ classes.

---

`multimodal_m3_ablation`

*Fit the complete M3 response-anchored channel-ablation lattice*

---

### Description

Fits all eight response-anchored combinations of RT, gaze and pupil. Models use one common subset-likelihood implementation to make the target and prior family explicit. This is inferential ablation, not a causal intervention on sensors.

## Usage

```
multimodal_m3_ablation(
  x,
  models = NULL,
  nuisance = stats::setNames(rep(TRUE, 8L), .ep10_m3_nuisance_names),
  ...
)
```

## Arguments

| | |
|---|---|
| x | M3-compatible data. |
| models | Optional subset of the eight model identifiers. |
| nuisance | Named logical vector selecting pupil nuisance terms. The same selection is applied to every ablation model containing pupil. |
| ... | Sampling controls forwarded to the subset fitter. |

## Value

An 'eye_multimodal_m3_ablation'.

---

# multimodal_m3_functional_bridge

*Bridge existing functional pupil outputs into the scalar M3 reference layer*

---

## Description

This bridge deliberately requires an analyst-supplied, already-derived trial-level functional score. It does not silently reduce a raw time series. Existing 'functional_pupil_irt_spec()', deconvolution, and confound tools remain the authoritative trajectory-level machinery.

## Usage

```
multimodal_m3_functional_bridge(
  data,
  score,
  pupil = "pupil",
  provenance = NULL
)
```

## Arguments

| | |
|---|---|
| data | Trial-level M3 data. |
| score | Trial-level functional/trajectory score or its column name. |
| pupil | Name of the output pupil column. |
| provenance | Free-text derivation/provenance note. |

### Value

An 'eye_multimodal_m3_functional_bridge' data object.

---

## multimodal_m3_negative_controls *Generate M3 multimodal falsification controls*

---

### Description

Includes within-item RT/gaze/pupil permutations, within-person pupil permutation across items, pupil phase randomization, luminance-only pseudo-pupil, and an irrelevant synthetic channel. Controls preserve selected marginals while deliberately breaking alignment; they are not causal interventions or misconduct detectors.

### Usage

```
multimodal_m3_negative_controls(x, seed = 20260815L)
```

### Arguments

| | |
|---|---|
| x | M3-compatible data. |
| seed | Deterministic seed. |

### Value

An 'eye_multimodal_m3_negative_controls'.

---

## multimodal_m3_ppc *Posterior predictive checks for the M3 four-channel model*

---

### Description

Extends the M2 W/L/M discrepancy family with a standardized pupil residual discrepancy P. Tail flags are diagnostics, not construct validation.

### Usage

```
multimodal_m3_ppc(x)
```

### Arguments

| | |
|---|---|
| x | An M3 fit. |

### Value

An 'eye_multimodal_m3_ppc'.

## multimodal_m3_process_information

*Quantify M3 process information, pupil increment, redundancy and sensor value*

### Description

Uses response-target PSIS-LOO and posterior ability variance across the eight-channel ablation lattice. It additionally reports paired pupil gains, a non-additivity/redundancy contrast, a channel-conflict screen, and optional information per usable pupil observation or sensor cost. Positive values do not establish construct validity or causal sensor value.

### Usage

```
multimodal_m3_process_information(x, pupil_cost = 1, decisive_z = 2)
```

### Arguments

| | |
|---|---|
| x | An M3 ablation object. |
| pupil_cost | Optional positive relative pupil-sensor cost. |
| decisive_z | Absolute delta/SE ratio used only as a descriptive evidence flag. |

### Value

An ‘eye_multimodal_m3_information‘ object.

## multimodal_m3_recovery

*Run M3 parameter-recovery and stress evidence*

### Description

Repeatedly simulates and fits M3 across pupil-signal and pupil-missingness scenarios, summarizing bias, RMSE, posterior SD, interval coverage and MCMC diagnostics. Small ‘reps‘ values are smoke tests; scientific promotion requires a predeclared larger grid.

### Usage

```
multimodal_m3_recovery(
  reps = 3L,
  pupil_signal = c("informative", "weak", "null", "redundant", "confounded"),
  pupil_missingness = c("mcar", "quality", "device"),
  n_person = 80L,
  n_item = 10L,
  seed = 20260815L,
```

```
  fit_args = list(chains = 2L, parallel_chains = 2L, iter_warmup = 500L, iter_sampling =
    300L, refresh = 0L, init = 0)
)
```

## Arguments

| | |
|---|---|
| reps | Replications per design cell. |
| pupil_signal, pupil_missingness | Scenario vectors. |
| n_person, n_item | Design size. |
| seed | Base seed. |
| fit_args | Named sampling arguments for 'fit_multimodal_m3()'. |

## Value

An 'eye_multimodal_m3_recovery' object.

---

# multimodal_m3_spec *M3 response + RT + gaze + pupil specification*

---

## Description

Extends the established eyeprocess multimodal IRT specification with a continuous pupil-responsivity channel. The summary representation is fitted by the bundled four-channel Stan model. 'functional_score' records that the pupil input was derived from a trajectory/functional workflow; it does not silently turn the scalar reference likelihood into a functional likelihood.

## Usage

```
multimodal_m3_spec(
  backend = "cmdstanr",
  prior_profile = c("regularized", "paper_centered"),
  missingness = "ignorable",
  pupil_representation = c("summary", "functional_score"),
  nuisance = stats::setNames(rep(TRUE, 8L), .ep10_m3_nuisance_names)
)
```

## Arguments

| | |
|---|---|
| backend | Currently '"cmdstanr"' only. |
| prior_profile | Prior profile. |
| missingness | Currently '"ignorable"' only. |
| pupil_representation | '"summary"' or '"functional_score"'. |
| nuisance | Named logical vector selecting baseline, luminance, gaze X/Y, quality, blink, interpolation and time-on-task nuisance terms. Availability is also audited from data. |

## Value

An 'eye_multimodal_m3_spec' inheriting the established multimodal and IRT spec classes.

---

`multimodal_m4_ablation` *Plan or fit the focused M3-to-M4 ablation set*

---

## Description

The default is plan-only. The essential lattice compares M3, formal K=1, reference K=2, K=2 without trait conditioning, and K=2 with iid states. Optional RT-anchored channel ablations are available but are not part of the default development burden. A no-RT K>1 model is intentionally excluded because the reference label-identification policy orders RT state deviations.

## Usage

```
multimodal_m4_ablation(
  x,
  run = FALSE,
  include_channel_ablations = FALSE,
  m3_fit = NULL,
  m4_fit = NULL,
  fit_args = list()
)
```

## Arguments

| | |
|---|---|
| `x` | Data or M4 simulation. |
| `run` | Whether to fit models. Default 'FALSE'. |
| `include_channel_ablations` | Add RT-only, no-gaze, and no-pupil state models. |
| `m3_fit, m4_fit` | Optional already fitted baseline/reference objects to reuse. |
| `fit_args` | Sampling arguments shared across new fits. |

## Value

An 'eye_multimodal_m4_ablation'.

## multimodal_m4_negative_controls

*Construct M4 temporal, nuisance, device, and overfitting negative controls*

### Description

Controls are transformations/stress designs by default and therefore do not incur backend fitting. Set ‘run = TRUE‘ only when explicit fitted comparisons are needed.

### Usage

```
multimodal_m4_negative_controls(
  x,
  controls = c("order_shuffle", "process_shuffle", "state_independent",
   "nuisance_pseudostate", "device_session_pseudostate", "overfit_state_count"),
  seed = 20260820L,
  run = FALSE,
  fit_args = list()
)
```

### Arguments

| | |
|---|---|
| x | Data or M4 simulation. |
| controls | Control names. |
| seed | Deterministic seed. |
| run | Whether to fit the generated controls. |
| fit_args | Arguments for optional M4 fits. |

### Value

An ‘eye_multimodal_m4_negative_controls‘.

## multimodal_m4_ppc

*Posterior predictive checks for M4 measurement and sequential behavior*

### Description

Compares observed process summaries with posterior-replicated summaries and reports replicated latent-state dynamics separately. Because states are latent, inferred state labels are never treated as observed ground truth.

## Usage

```
multimodal_m4_ppc(x)
```

## Arguments

| | |
|---|---|
| x | An M4 fit. |

## Value

An 'eye_multimodal_m4_ppc'.

---

`multimodal_m4_process_information` *Quantify incremental response-target information supplied by M4 state structure*

---

## Description

Uses commensurate response-target PSIS-LOO and ability-posterior uncertainty to compare M3 with focused M4 variants. It explicitly allows no benefit, redundancy, or destabilization and does not sum channel Fisher information.

## Usage

```
multimodal_m4_process_information(x, decisive_z = 2)
```

## Arguments

| | |
|---|---|
| x | Executed M4 ablation object. |
| decisive_z | Descriptive absolute delta/SE threshold. |

## Value

An 'eye_multimodal_m4_information'.

multimodal_m4_recovery *Evaluate deterministic M4 parameter and state recovery*

## Description

Recovery aligns latent-state labels before scoring multichannel state effects, and emphasizes posterior-probability calibration, occupancy, and transition recovery rather than treating MAP classification accuracy as the primary criterion. By default the function returns a small five-scenario design and does not launch expensive fitting.

## Usage

```
multimodal_m4_recovery(
  simulation = NULL,
  fit = NULL,
 scenarios = c("clear", "weak", "null", "trait_conditioned", "nuisance_confounded"),
  run = FALSE,
  simulation_args = list(n_person = 60L, n_item = 10L),
  fit_args = list()
)
```

## Arguments

| | |
|---|---|
| simulation | Optional single M4 simulation. |
| fit | Optional already-fitted M4 model corresponding to ‘simulation‘. |
| scenarios | Deterministic development battery used when ‘run = TRUE‘. |
| run | Whether to execute the small recovery battery. Default ‘FALSE‘. |
| simulation_args | Named arguments forwarded to simulation. |
| fit_args | Named arguments forwarded to ‘fit_multimodal_m4()‘. |

## Value

An ‘eye_multimodal_m4_recovery‘.

## multimodal_m4_sensitivity

*Plan or run M4 state-count and modelling sensitivity analyses*

### Description

The default returns a transparent sensitivity design without fitting. It covers K, priors, trait conditioning, transition structure, state channels, nuisance adjustment, and sequence-length threshold. No automatic 'best_K' is declared.

### Usage

```
multimodal_m4_sensitivity(x, n_states = 1:4, run = FALSE, fit_args = list())
```

### Arguments

| | |
|---|---|
| x | Data or simulation. |
| n_states | Candidate state counts, default 1:4. |
| run | Whether to execute K-sensitivity fits. Other sensitivity dimensions remain explicit design rows rather than an automatic combinatorial grid. |
| fit_args | Optional fitting arguments. |

### Value

An 'eye_multimodal_m4_sensitivity'.

## multimodal_m4_spec

*Specify M4 trait-conditioned latent response-process states*

### Description

Defines the M4 sequential extension of the validated M3 response + RT + gaze + pupil model. The reference implementation keeps the scored-response Rasch equation state-independent and lets latent states shift selected RT, gaze, and pupil process channels. State labels are statistical identification labels only and do not imply psychological constructs.

### Usage

```
multimodal_m4_spec(
  n_states = 2L,
  state_channels = c("rt", "gaze", "pupil"),
  transition_structure = c("markov", "iid"),
  trait_conditioning = c("theta", "tau"),
  initial_trait_conditioning = TRUE,
  min_sequence_length = 2L,
```

```
  identification = c("ordered_rt_effect"),
  prior_profile = c("regularized", "paper_centered"),
  missingness = "ignorable",
  nuisance = stats::setNames(rep(TRUE, 8L), .ep10_m3_nuisance_names),
  backend = "cmdstanr"
)
```

## Arguments

| | |
|---|---|
| n_states | Number of latent states, from 1 through 4. ‘1‘ is the formal null state model and should be treated as scientifically meaningful. |
| state_channels | Subset of ‘"rt"‘, ‘"gaze"‘, and ‘"pupil"‘ receiving state-dependent deviations. |
| transition_structure | ‘"markov"‘ for first-order transitions or ‘"iid"‘ for independent state membership over ordered trials. |
| trait_conditioning | Subset of ‘theta‘, ‘tau‘, ‘omega‘, and ‘rho‘ used to condition transition logits. The conservative default is ‘theta + tau‘. |
| initial_trait_conditioning | Whether the same selected traits condition initial-state probabilities. |
| min_sequence_length | Minimum sequence length required by the structural audit for fitting. No rows are silently removed when sequences are shorter. |
| identification | State-label identification policy. The reference policy orders centered RT state deviations; this is a label convention only. |
| prior_profile | Prior profile inherited from M3. |
| missingness | Currently ‘"ignorable"‘ only. |
| nuisance | Named pupil-nuisance selection vector inherited from M3. |
| backend | Currently ‘"cmdstanr"‘ only. |

## Value

An ‘eye_multimodal_m4_spec‘ inheriting the canonical eyeprocess multimodal/IRT specification classes.

---

# multimodal_m4_state_diagnostics

*Summarize M4 latent-state uncertainty and dynamics*

---

## Description

Returns posterior state probabilities, entropy, occupancy, MAP-run summaries, transition probabilities, and posterior-weighted process-channel profiles. MAP states are explicitly secondary summaries of posterior probabilities.

## Usage

```
multimodal_m4_state_diagnostics(x)
```

## Arguments

| | |
|---|---|
| x | M4 fit or M4 simulation. Simulation diagnostics use known synthetic truth and are labelled accordingly. |

## Value

An ‘eye_multimodal_m4_states‘ object.

---

`multimodal_ppc` *Posterior predictive checks for multimodal development fits*

---

## Description

Posterior predictive checks for multimodal development fits

## Usage

```
multimodal_ppc(object, variables = NULL)
```

## Arguments

| | |
|---|---|
| object | An ‘eye_multimodal_irt_fit‘. |
| variables | Optional generated-quantity variable names. |

## Value

An ‘eye_multimodal_ppc‘.

---

`negative_control_concordance`
*Concordance of multiple negative-control families*

---

## Description

Concordance of multiple negative-control families

## Usage

```
negative_control_concordance(x, effect = "effect", tolerance = 0.05)
```

### Arguments

| | |
|---|---|
| x | Negative-control result. |
| effect | Effect column. |
| tolerance | Absolute mean-effect tolerance. |

### Value

A named list with components "summary", "all_within_tolerance", "tolerance", containing concordance of multiple negative-control families and associated metadata or diagnostics.

---

## negative_control_process_test

*Negative-control test for an allegedly informative process channel*

---

### Description

The user supplies a complete evaluation callback. eyeprocess permutes the named process columns, optionally within grouping strata, and compares the observed score with the permutation distribution.

### Usage

```
negative_control_process_test(
  data,
  process_columns,
  evaluator,
  within = NULL,
  permutations = 100L,
  higher_is_better = TRUE,
  seed = 20260808L
)
```

### Arguments

| | |
|---|---|
| data | Input data. |
| process_columns | Columns to permute. |
| evaluator | Function returning one scalar out-of-sample score. |
| within | Optional grouping columns within which permutation occurs. |
| permutations | Number of negative-control permutations. |
| higher_is_better | Score direction. |
| seed | Random-number seed. |

### Value

An object of class "eye_process_negative_control", stored as a named list, with components "observed", "null", "p_value", "permutations", "process_columns", "within", "higher_is_better", "seed". It contains negative-control test for an allegedly informative process channel and associated metadata or diagnostics needed to interpret the result.

---

## object_hash *Hash an R object reproducibly within an R serialization version*

---

### Description

Hash an R object reproducibly within an R serialization version

### Usage

```
object_hash(x)
```

### Arguments

| | |
|---|---|
| x | R object. |

### Value

An R object containing hash an R object reproducibly within an R serialization version. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

## object_schema *Describe a stable object schema*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
object_schema(object = c("eye_dataset", "eyeprocess_model", "validation_plan",
  "validation_collection", "vendor_corpus", "eye_storage"))
```

### Arguments

| | |
|---|---|
| object | Object or schema name. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## open_eye_storage *Open an eyeprocess storage handle*

### Description

Open an eyeprocess storage handle

### Usage

```
open_eye_storage(path, format = NULL)
```

### Arguments

| | |
|---|---|
| path | Storage path. |
| format | Optional explicit format. |

### Value

An 'eye_storage' handle without collecting all tables.

## open_partitioned_eye_storage *Open partitioned eye storage*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
open_partitioned_eye_storage(path)
```

### Arguments

| | |
|---|---|
| path | Storage directory. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## option_process_information

*Quantify option-process information from a nominal gaze model*

### Description

Quantify option-process information from a nominal gaze model

### Usage

```
option_process_information(object)
```

### Arguments

| | |
|---|---|
| object | A fitted eyeprocess model or audit object. |

### Value

An object of class "eye_option_process_information", "data.frame", stored as a data frame, containing quantify option-process information from a nominal gaze model and associated metadata needed to interpret the result.

## outcome_blind_feature_audit

*Audit whether candidate predictors can be constructed without an outcome column*

### Description

Audit whether candidate predictors can be constructed without an outcome column

### Usage

```
outcome_blind_feature_audit(data, outcome, feature_fun)
```

### Arguments

| | |
|---|---|
| data | Data frame. |
| outcome | Outcome column name. |
| feature_fun | Function receiving outcome-hidden data and returning features. |

### Value

A named list with components "status", "outcome", "feature_result", "error", "warnings", "input_columns", "interpretation", containing whether candidate predictors can be constructed without an outcome column and associated metadata or diagnostics.

`outcome_blind_snapshot` *Create an outcome-blind data snapshot*

## Description

Create an outcome-blind data snapshot

## Usage

```
outcome_blind_snapshot(data, outcome, id = NULL)
```

## Arguments

| | |
|---|---|
| `data` | Data frame. |
| `outcome` | Outcome column(s) to remove from the analysis snapshot. |
| `id` | Optional identifier columns retained in the snapshot. |

## Value

An object of class "eye_outcome_blind_snapshot", stored as a named list, with components "data", "removed_outcomes", "id", "source_columns", "blinded_columns", "blinded_hash", "created_at", "caveat". It contains an outcome-blind data snapshot and associated metadata or diagnostics needed to interpret the result.

`package_reproducibility_manifest` *Create a reproducibility manifest for files and software*

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
package_reproducibility_manifest(paths, include_session = TRUE)
```

## Arguments

| | |
|---|---|
| `paths` | Files/directories. |
| `include_session` | Whether to capture session information. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## paper_reproducibility_manifest *Create a compact paper reproducibility manifest*

### Description

Create a compact paper reproducibility manifest

### Usage

```
paper_reproducibility_manifest(
  evidence,
  manuscript = NULL,
  figures = NULL,
  tables = NULL
)
```

### Arguments

| | |
|---|---|
| evidence | Evidence bundle. |
| manuscript | Optional manuscript path. |
| figures | Optional figure paths. |
| tables | Optional table paths. |

### Value

A named list with components "evidence_hash", "manuscript", "files", "reproducibility", "generated_utc", containing a compact paper reproducibility manifest and associated metadata or diagnostics.

## partition_eye_storage *Create a partition specification*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
partition_eye_storage(by = c("participant_id", "session_id", "recording_id"),
  format = c("parquet", "csv", "rds"), compression = "zstd", max_rows = 1000000L)
```

### Arguments

| | |
|---|---|
| by | Partition columns. |
| format | Storage format. |
| compression | Compression codec for Parquet. |
| max_rows | Maximum rows per physical file. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

## pipeline_failures *Return failed pipeline steps*

---

### Description

Return failed pipeline steps

### Usage

```
pipeline_failures(x)
```

### Arguments

| | |
|---|---|
| x | Pipeline run. |

### Value

A data frame containing return failed pipeline steps. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

---

## pipeline_result *Extract a pipeline result by step name*

---

### Description

Extract a pipeline result by step name

### Usage

```
pipeline_result(x, step)
```

### Arguments

| | |
|---|---|
| x | Pipeline run. |
| step | Step name. |

## Value

An R object containing a pipeline result by step name. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

`pipeline_step_status` *Pipeline step status table*

---

## Description

Pipeline step status table

## Usage

```
pipeline_step_status(x)
```

## Arguments

| | |
|---|---|
| `x` | Pipeline run. |

## Value

A data frame containing pipeline step status table. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

---

`placebo_window_audit` *Audit a placebo/pre-event window*

---

## Description

Audit a placebo/pre-event window

## Usage

```
placebo_window_audit(data, time, value, window, expected = 0, by = NULL)
```

## Arguments

| | |
|---|---|
| `data` | Data frame. |
| `time` | Time column. |
| `value` | Value column. |
| `window` | Two-element placebo window. |
| `expected` | Optional expected mean, typically zero. |
| `by` | Optional grouping columns. |

## Value

An R object containing a placebo/pre-event window. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

`plot.eye_adapter_regression_audit`

*Plot eye adapter regression audit*

---

## Description

Plot eye adapter regression audit

## Usage

```
## S3 method for class 'eye_adapter_regression_audit'
plot(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Object to print, plot, summarize, or audit. |
| `...` | Additional arguments passed to the selected model, engine, or method. |

## Value

Invisibly returns the plotting result when available; the primary effect is drawing plot eye adapter regression audit.

---

`plot.eye_aoi_growth_curve`

*Plot aoi growth curve diagnostics*

---

## Description

Plot aoi growth curve diagnostics

## Usage

```
## S3 method for class 'eye_aoi_growth_curve'
plot(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Object to process, inspect, compare, or plot. |
| `...` | Additional arguments passed to the underlying method or helper. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot aoi growth curve diagnostics.

---

`plot.eye_aoi_trajectory`
*Plot aoi trajectory diagnostics*

---

### Description

Plot aoi trajectory diagnostics

### Usage

```
## S3 method for class 'eye_aoi_trajectory'
plot(x, type = c("coefficients", "profiles"), ...)
```

### Arguments

| | |
|---|---|
| `x` | Object to process, inspect, compare, or plot. |
| `type` | Type of summary or visual representation to produce. |
| `...` | Additional arguments passed to the underlying method or helper. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot aoi trajectory diagnostics.

---

`plot.eye_bayesian_process_dashboard`
*Plot bayesian process dashboard diagnostics*

---

### Description

Plot bayesian process dashboard diagnostics

### Usage

```
## S3 method for class 'eye_bayesian_process_dashboard'
plot(x, type = c("loo", "rhat", "ess"), ...)
```

### Arguments

| | |
|---|---|
| `x` | Object to process, inspect, compare, or plot. |
| `type` | Type of summary or visual representation to produce. |
| `...` | Additional arguments passed to the underlying method or helper. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot bayesian process dashboard diagnostics.

---

`plot.eye_bids_roundtrip`

*Plot eye bids roundtrip*

---

### Description

Plot eye bids roundtrip

### Usage

```
## S3 method for class 'eye_bids_roundtrip'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | Object to print, plot, summarize, or audit. |
| `...` | Additional arguments passed to the selected model, engine, or method. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot eye bids roundtrip.

---

`plot.eye_biometric_imputation_sensitivity`

*Plot biometric imputation sensitivity diagnostics*

---

### Description

Plot biometric imputation sensitivity diagnostics

### Usage

```
## S3 method for class 'eye_biometric_imputation_sensitivity'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | Object to process, inspect, compare, or plot. |
| `...` | Additional arguments passed to the underlying method or helper. |

## Value

Invisibly returns the plotting result when available; the primary effect is drawing plot biometric imputation sensitivity diagnostics.

---

`plot.eye_biometric_preflight`
*Plot biometric preflight diagnostics*

---

## Description

Plot biometric preflight diagnostics

## Usage

```
## S3 method for class 'eye_biometric_preflight'
plot(x, type = c("heatmap", "decision_counts"), ...)
```

## Arguments

| | |
|---|---|
| `x` | Object to process, inspect, compare, or plot. |
| `type` | Type of summary or visual representation to produce. |
| `...` | Additional arguments passed to the underlying method or helper. |

## Value

Invisibly returns the plotting result when available; the primary effect is drawing plot biometric preflight diagnostics.

---

`plot.eye_candidate_item_bank_audit`
*Plot candidate item bank audit diagnostics*

---

## Description

Plot candidate item bank audit diagnostics

## Usage

```
## S3 method for class 'eye_candidate_item_bank_audit'
plot(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Object to process, inspect, compare, or plot. |
| `...` | Additional arguments passed to the underlying method or helper. |

## Value

Invisibly returns the plotting result when available; the primary effect is drawing plot candidate item bank audit diagnostics.

---

`plot.eye_compatibility_evidence_matrix`
*Plot detailed compatibility evidence*

---

## Description

Plot detailed compatibility evidence

## Usage

```
## S3 method for class 'eye_compatibility_evidence_matrix'
plot(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Object to print, plot, summarize, or audit. |
| `...` | Additional arguments passed to the selected model, engine, or method. |

## Value

Invisibly returns the plotting result when available; the primary effect is drawing plot detailed compatibility evidence.

---

`plot.eye_decision_process_proxy`
*Plot decision process proxy diagnostics*

---

## Description

Plot decision process proxy diagnostics

## Usage

```
## S3 method for class 'eye_decision_process_proxy'
plot(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Object to process, inspect, compare, or plot. |
| `...` | Additional arguments passed to the underlying method or helper. |

## Value

Invisibly returns the plotting result when available; the primary effect is drawing plot decision process proxy diagnostics.

---

`plot.eye_dynamic_irtree` *plot eye dynamic irtree*

---

## Description

S3 method supporting a research-scale eyeprocess object.

## Usage

```
## S3 method for class 'eye_dynamic_irtree'
plot(x, type = c("observed", "fitted", "residual"), ...)
```

## Arguments

| | |
|---|---|
| `x` | Value for ‘x‘. See the function description and relevant article for constraints. |
| `type` | Value for ‘type‘. See the function description and relevant article for constraints. |
| `...` | Additional arguments passed to the selected engine or method. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`plot.eye_dynamic_ppc` *plot eye dynamic ppc*

---

## Description

S3 method supporting a research-scale eyeprocess object.

## Usage

```
## S3 method for class 'eye_dynamic_ppc'
plot(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Value for ‘x‘. See the function description and relevant article for constraints. |
| `...` | Additional arguments passed to the selected engine or method. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## plot.eye_event_roundtrip_audit *Plot eye event roundtrip audit*

### Description

Plot eye event roundtrip audit

### Usage

```
## S3 method for class 'eye_event_roundtrip_audit'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to print, plot, summarize, or audit. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot eye event roundtrip audit.

## plot.eye_event_time_irt *Plot eye event time irt*

### Description

Plot eye event time irt

### Usage

```
## S3 method for class 'eye_event_time_irt'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to print, plot, summarize, or audit. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot eye event time irt.

## plot.eye_functional_pupil_diagnostics

*plot eye functional pupil diagnostics*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_functional_pupil_diagnostics'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## plot.eye_functional_pupil_irt

*plot eye functional pupil irt*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_functional_pupil_irt'
plot(x, type = c("trajectories", "coefficients",
  "recovery"), ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| type | Value for ‘type‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## plot.eye_functional_pupil_sensitivity

*plot eye functional pupil sensitivity*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_functional_pupil_sensitivity'
plot(x, parameter = NULL, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| parameter | Value for ‘parameter‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## plot.eye_gated_process_model

*Plot gated process model diagnostics*

### Description

Plot gated process model diagnostics

### Usage

```
## S3 method for class 'eye_gated_process_model'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot gated process model diagnostics.

## plot.eye_gaze_anchored_3pl_audit

*Plot gaze anchored 3pl audit diagnostics*

### Description

Plot gaze anchored 3pl audit diagnostics

### Usage

```
## S3 method for class 'eye_gaze_anchored_3pl_audit'
plot(
  x,
  type = c("lower_asymptote", "process_alignment", "difficulty_discrimination"),
  feature = NULL,
  ...
)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |
| type | Type of summary or visual representation to produce. |
| feature | Process feature to evaluate or display. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot gaze anchored 3pl audit diagnostics.

## plot.eye_gaze_informed_missingness_irt

*Plot eye gaze informed missingness irt*

### Description

Plot eye gaze informed missingness irt

### Usage

```
## S3 method for class 'eye_gaze_informed_missingness_irt'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to print, plot, summarize, or audit. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot eye gaze informed missingness irt.

---

## plot.eye_gpirt *Plot flexible IRF shape diagnostics*

---

### Description

Plot flexible IRF shape diagnostics

### Usage

```
## S3 method for class 'eye_gpirt'
plot(x, item = NULL, ...)
```

### Arguments

| | |
|---|---|
| x | Object to print, plot, summarize, or audit. |
| item | Item identifier, name, or item column. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot flexible IRF shape diagnostics.

---

## plot.eye_incremental_information_audit *Plot incremental process-channel information by fold*

---

### Description

Plot incremental process-channel information by fold

### Usage

```
## S3 method for class 'eye_incremental_information_audit'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to print, plot, summarize, or audit. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot incremental process-channel information by fold.

---

## plot.eye_irt_changepoints *Plot detected process changepoints*

---

### Description

Plot detected process changepoints

### Usage

```
## S3 method for class 'eye_irt_changepoints'
plot(x, person = NULL, ...)
```

### Arguments

| | |
|---|---|
| x | Object to print, plot, summarize, or audit. |
| person | Person or participant identifier column. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot detected process changepoints.

---

## plot.eye_irt_equating *Plot an IRT linking/equating transformation*

---

### Description

Plot an IRT linking/equating transformation

### Usage

```
## S3 method for class 'eye_irt_equating'
plot(x, theta = seq(-4, 4, length.out = 201), ...)
```

### Arguments

| | |
|---|---|
| x | Object to print, plot, summarize, or audit. |
| theta | Latent-trait values. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot an IRT linking/equating transformation.

---

## plot.eye_irt_ppc *Plot posterior predictive discrepancy tail probabilities*

---

### Description

Plot posterior predictive discrepancy tail probabilities

### Usage

```
## S3 method for class 'eye_irt_ppc'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to print, plot, summarize, or audit. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot posterior predictive discrepancy tail probabilities.

---

## plot.eye_irt_recovery_summary *Plot parameter-recovery bias or RMSE*

---

### Description

Plot parameter-recovery bias or RMSE

### Usage

```
## S3 method for class 'eye_irt_recovery_summary'
plot(x, metric = c("rmse", "absolute_bias", "coverage"), ...)
```

### Arguments

| | |
|---|---|
| x | Object to print, plot, summarize, or audit. |
| metric | Metric to calculate or audit. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot parameter-recovery bias or RMSE.

---

## plot.eye_irt_sbc *Plot SBC rank histograms by parameter*

---

### Description

Plot SBC rank histograms by parameter

### Usage

```
## S3 method for class 'eye_irt_sbc'
plot(x, parameter = NULL, breaks = 10L, ...)
```

### Arguments

| | |
|---|---|
| x | Object to print, plot, summarize, or audit. |
| parameter | Value supplied to ‘parameter‘; see Details for its model-specific role. |
| breaks | Histogram or discretization breaks. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot SBC rank histograms by parameter.

## plot.eye_item_parameter_seed

*Plot item parameter seed diagnostics*

### Description

Plot item parameter seed diagnostics

### Usage

```
## S3 method for class 'eye_item_parameter_seed'
plot(x, candidate_data = NULL, ...)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |
| candidate_data | Optional candidate-item data used for the requested display. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot item parameter seed diagnostics.

## plot.eye_item_reduction_sensitivity

*Plot item reduction sensitivity diagnostics*

### Description

Plot item reduction sensitivity diagnostics

### Usage

```
## S3 method for class 'eye_item_reduction_sensitivity'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot item reduction sensitivity diagnostics.

## plot.eye_joint_gaze_rt_irt

*Plot a joint gaze-response-time IRT fit*

### Description

Plot a joint gaze-response-time IRT fit

### Usage

```
## S3 method for class 'eye_joint_gaze_rt_irt'
plot(x, type = c("latent", "components"), ...)
```

### Arguments

| | |
|---|---|
| x | Object to print, plot, summarize, or audit. |
| type | Plot or result type. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot a joint gaze-response-time IRT fit.

## plot.eye_joint_graded_rt_process_irt

*Plot a graded response + RT/process fit*

### Description

Plot a graded response + RT/process fit

### Usage

```
## S3 method for class 'eye_joint_graded_rt_process_irt'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to print, plot, summarize, or audit. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot a graded response + RT/process fit.

## plot.eye_latent_distribution_comparison

*Plot eye latent distribution comparison*

### Description

Plot eye latent distribution comparison

### Usage

```
## S3 method for class 'eye_latent_distribution_comparison'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to print, plot, summarize, or audit. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot eye latent distribution comparison.

## plot.eye_latent_process_alignment

*Plot latent process alignment diagnostics*

### Description

Plot latent process alignment diagnostics

### Usage

```
## S3 method for class 'eye_latent_process_alignment'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot latent process alignment diagnostics.

## plot.eye_latent_space_irt *Plot a latent-space IRT adapter fit*

### Description

Plot a latent-space IRT adapter fit

### Usage

```
## S3 method for class 'eye_latent_space_irt'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to print, plot, summarize, or audit. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot a latent-space IRT adapter fit.

## plot.eye_manyfacet_process_irt *Plot many-facet process IRT effects*

### Description

Plot many-facet process IRT effects

### Usage

```
## S3 method for class 'eye_manyfacet_process_irt'
plot(x, facet = NULL, ...)
```

### Arguments

| | |
|---|---|
| x | Object to print, plot, summarize, or audit. |
| facet | Facet to display or extract. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot many-facet process IRT effects.

## plot.eye_mixture_irt_process

*Plot mixture irt process diagnostics*

### Description

Plot mixture irt process diagnostics

### Usage

```
## S3 method for class 'eye_mixture_irt_process'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot mixture irt process diagnostics.

## plot.eye_model_promotion_audit

*plot eye model promotion audit*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_model_promotion_audit'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`plot.eye_multiblock_process_map`
*Plot multiblock process map diagnostics*

---

## Description

Plot multiblock process map diagnostics

## Usage

```
## S3 method for class 'eye_multiblock_process_map'
plot(x, type = c("individuals", "variables", "blocks", "contributions"), ...)
```

## Arguments

| | |
|---|---|
| `x` | Object to process, inspect, compare, or plot. |
| `type` | Type of summary or visual representation to produce. |
| `...` | Additional arguments passed to the underlying method or helper. |

## Value

Invisibly returns the plotting result when available; the primary effect is drawing plot multiblock process map diagnostics.

---

`plot.eye_nominal_gaze_irt`
*Plot nominal-response gaze results*

---

## Description

Plot nominal-response gaze results

## Usage

```
## S3 method for class 'eye_nominal_gaze_irt'
plot(x, type = c("distractor_map", "coefficients"), ...)
```

## Arguments

| | |
|---|---|
| `x` | Object to print, plot, summarize, or audit. |
| `type` | Plot or result type. |
| `...` | Additional arguments passed to the selected model, engine, or method. |

## Value

Invisibly returns the plotting result when available; the primary effect is drawing plot nominal-response gaze results.

## plot.eye_nonparametric_rasch_audit *Plot nonparametric rasch audit diagnostics*

### Description

Plot nonparametric rasch audit diagnostics

### Usage

```
## S3 method for class 'eye_nonparametric_rasch_audit'
plot(x, method = names(x$tests)[1L], ...)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |
| method | Method used for the requested diagnostic or summary. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot nonparametric rasch audit diagnostics.

## plot.eye_omission_survival_irt *Plot omission/not-reached survival IRT diagnostics*

### Description

Plot omission/not-reached survival IRT diagnostics

### Usage

```
## S3 method for class 'eye_omission_survival_irt'
plot(x, type = c("survival", "missingness"), ...)
```

### Arguments

| | |
|---|---|
| x | Object to print, plot, summarize, or audit. |
| type | Plot or result type. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot omission/not-reached survival IRT diagnostics.

`plot.eye_preaction_process_features`
*Plot preaction process features diagnostics*

## Description

Plot preaction process features diagnostics

## Usage

```
## S3 method for class 'eye_preaction_process_features'
plot(x, feature = "pupil_mean", ...)
```

## Arguments

| | |
|---|---|
| `x` | Object to process, inspect, compare, or plot. |
| `feature` | Process feature to evaluate or display. |
| `...` | Additional arguments passed to the underlying method or helper. |

## Value

Invisibly returns the plotting result when available; the primary effect is drawing plot preaction process features diagnostics.

`plot.eye_presentation_accessibility`
*Plot presentation accessibility diagnostics*

## Description

Plot presentation accessibility diagnostics

## Usage

```
## S3 method for class 'eye_presentation_accessibility'
plot(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Object to process, inspect, compare, or plot. |
| `...` | Additional arguments passed to the underlying method or helper. |

## Value

Invisibly returns the plotting result when available; the primary effect is drawing plot presentation accessibility diagnostics.

## plot.eye_presentation_fairness_comparison

*Plot presentation fairness comparison diagnostics*

### Description

Plot presentation fairness comparison diagnostics

### Usage

```
## S3 method for class 'eye_presentation_fairness_comparison'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot presentation fairness comparison diagnostics.

## plot.eye_process_anomaly_audit

*Plot process anomaly audit diagnostics*

### Description

Plot process anomaly audit diagnostics

### Usage

```
## S3 method for class 'eye_process_anomaly_audit'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot process anomaly audit diagnostics.

## plot.eye_process_cat_simulation

*Plot CAT simulation information accumulation*

### Description

Plot CAT simulation information accumulation

### Usage

```
## S3 method for class 'eye_process_cat_simulation'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to print, plot, summarize, or audit. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot CAT simulation information accumulation.

## plot.eye_process_channel_ablation

*Plot process-channel ablation*

### Description

Plot process-channel ablation

### Usage

```
## S3 method for class 'eye_process_channel_ablation'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to print, plot, summarize, or audit. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot process-channel ablation.

## plot.eye_process_dependent_discrimination

*Plot process-dependent discrimination*

### Description

Plot process-dependent discrimination

### Usage

```
## S3 method for class 'eye_process_dependent_discrimination'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to print, plot, summarize, or audit. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot process-dependent discrimination.

## plot.eye_process_drift_audit

*Plot process drift audit diagnostics*

### Description

Plot process drift audit diagnostics

### Usage

```
## S3 method for class 'eye_process_drift_audit'
plot(
  x,
  type = c("trajectory", "delta", "heatmap", "control"),
  metric = NULL,
  item = NULL,
  ...
)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |
| type | Type of summary or visual representation to produce. |
| metric | Metric to evaluate or display. |
| item | Optional item identifier used to restrict or highlight results. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot process drift audit diagnostics.

---

## plot.eye_process_external_validity

*Plot process external validity diagnostics*

---

### Description

Plot process external validity diagnostics

### Usage

```
## S3 method for class 'eye_process_external_validity'
plot(
  x,
  type = c("associations", "incremental", "observed_fitted", "residuals"),
  ...
)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |
| type | Type of summary or visual representation to produce. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot process external validity diagnostics.

## plot.eye_process_facet_effects

*Plot eye process facet effects*

### Description

Plot eye process facet effects

### Usage

```
## S3 method for class 'eye_process_facet_effects'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to print, plot, summarize, or audit. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot eye process facet effects.

## plot.eye_process_feature_blocks

*Plot process feature blocks diagnostics*

### Description

Plot process feature blocks diagnostics

### Usage

```
## S3 method for class 'eye_process_feature_blocks'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot process feature blocks diagnostics.

---

`plot.eye_process_g_study` *Plot process-measure variance components*

---

## Description

Plot process-measure variance components

## Usage

```
## S3 method for class 'eye_process_g_study'
plot(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Object to print, plot, summarize, or audit. |
| `...` | Additional arguments passed to the selected model, engine, or method. |

## Value

Invisibly returns the plotting result when available; the primary effect is drawing plot process-measure variance components.

---

`plot.eye_process_hmm_irt` *Plot a process-HMM IRT fit*

---

## Description

Plot a process-HMM IRT fit

## Usage

```
## S3 method for class 'eye_process_hmm_irt'
plot(x, type = c("occupancy", "transition"), ...)
```

## Arguments

| | |
|---|---|
| `x` | Object to print, plot, summarize, or audit. |
| `type` | Plot or result type. |
| `...` | Additional arguments passed to the selected model, engine, or method. |

## Value

Invisibly returns the plotting result when available; the primary effect is drawing plot a process-HMM IRT fit.

## plot.eye_process_local_dependence_audit

*Plot process/local-dependence diagnostics*

### Description

Plot process/local-dependence diagnostics

### Usage

```
## S3 method for class 'eye_process_local_dependence_audit'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to print, plot, summarize, or audit. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot process/local-dependence diagnostics.

## plot.eye_process_negative_control

*Plot a process-channel negative-control distribution*

### Description

Plot a process-channel negative-control distribution

### Usage

```
## S3 method for class 'eye_process_negative_control'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to print, plot, summarize, or audit. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot a process-channel negative-control distribution.

## plot.eye_process_person_fit

*Plot process person-fit discrepancies*

### Description

Plot process person-fit discrepancies

### Usage

```
## S3 method for class 'eye_process_person_fit'
plot(x, top = 25L, ...)
```

### Arguments

| | |
|---|---|
| x | Object to print, plot, summarize, or audit. |
| top | Number of highest-ranked cases to display. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot process person-fit discrepancies.

## plot.eye_process_profile_mixture

*Plot process profile mixture diagnostics*

### Description

Plot process profile mixture diagnostics

### Usage

```
## S3 method for class 'eye_process_profile_mixture'
plot(x, type = c("profiles", "posterior", "scatter", "parallel"), ...)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |
| type | Type of summary or visual representation to produce. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot process profile mixture diagnostics.

## plot.eye_process_rasch_tree

*Plot process rasch tree diagnostics*

### Description

Plot process rasch tree diagnostics

### Usage

```
## S3 method for class 'eye_process_rasch_tree'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot process rasch tree diagnostics.

## plot.eye_process_windows

*Plot process windows diagnostics*

### Description

Plot process windows diagnostics

### Usage

```
## S3 method for class 'eye_process_windows'
plot(x, feature = "pupil_mean", group = NULL, ...)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |
| feature | Process feature to evaluate or display. |
| group | Optional grouping variable. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot process windows diagnostics.

---

`plot.eye_process_window_sensitivity`

*Plot process window sensitivity diagnostics*

---

## Description

Plot process window sensitivity diagnostics

## Usage

```
## S3 method for class 'eye_process_window_sensitivity'
plot(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Object to process, inspect, compare, or plot. |
| `...` | Additional arguments passed to the underlying method or helper. |

## Value

Invisibly returns the plotting result when available; the primary effect is drawing plot process window sensitivity diagnostics.

---

`plot.eye_pupil_confound_model`

*Plot pupil confound model diagnostics*

---

## Description

Plot pupil confound model diagnostics

## Usage

```
## S3 method for class 'eye_pupil_confound_model'
plot(
  x,
 type = c("luminance", "trial_order", "raw_adjusted", "theta_luminance_surface"),
  ...
)
```

## Arguments

| | |
|---|---|
| `x` | Object to process, inspect, compare, or plot. |
| `type` | Type of summary or visual representation to produce. |
| `...` | Additional arguments passed to the underlying method or helper. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot pupil confound model diagnostics.

---

`plot.eye_pupil_deconvolution`

*Plot pupil deconvolution diagnostics*

---

### Description

Plot pupil deconvolution diagnostics

### Usage

```
## S3 method for class 'eye_pupil_deconvolution'
plot(x, type = c("observed_fitted", "effects", "residuals", "kernels"), ...)
```

### Arguments

| | |
|---|---|
| `x` | Object to process, inspect, compare, or plot. |
| `type` | Type of summary or visual representation to produce. |
| `...` | Additional arguments passed to the underlying method or helper. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot pupil deconvolution diagnostics.

---

`plot.eye_pupil_fatigue_drift`

*Plot pupil fatigue drift diagnostics*

---

### Description

Plot pupil fatigue drift diagnostics

### Usage

```
## S3 method for class 'eye_pupil_fatigue_drift'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| `x` | Object to process, inspect, compare, or plot. |
| `...` | Additional arguments passed to the underlying method or helper. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot pupil fatigue drift diagnostics.

---

`plot.eye_pupil_frequency_features`
*Plot pupil frequency features diagnostics*

---

### Description

Plot pupil frequency features diagnostics

### Usage

```
## S3 method for class 'eye_pupil_frequency_features'
plot(x, type = c("features", "power_relationship"), ...)
```

### Arguments

| | |
|---|---|
| `x` | Object to process, inspect, compare, or plot. |
| `type` | Type of summary or visual representation to produce. |
| `...` | Additional arguments passed to the underlying method or helper. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot pupil frequency features diagnostics.

---

`plot.eye_pupil_frequency_stability`
*Plot pupil frequency stability diagnostics*

---

### Description

Plot pupil frequency stability diagnostics

### Usage

```
## S3 method for class 'eye_pupil_frequency_stability'
plot(x, feature = "pupil_frequency_contrast", ...)
```

### Arguments

| | |
|---|---|
| `x` | Object to process, inspect, compare, or plot. |
| `feature` | Process feature to evaluate or display. |
| `...` | Additional arguments passed to the underlying method or helper. |

## Value

Invisibly returns the plotting result when available; the primary effect is drawing plot pupil frequency stability diagnostics.

---

`plot.eye_roundtrip_loss_audit` *plot eye roundtrip loss audit*

---

## Description

S3 method supporting a research-scale eyeprocess object.

## Usage

```
## S3 method for class 'eye_roundtrip_loss_audit'
plot(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Value for ‘x‘. See the function description and relevant article for constraints. |
| `...` | Additional arguments passed to the selected engine or method. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`plot.eye_sbc_audit` *Plot SBC audit summaries*

---

## Description

Plot SBC audit summaries

## Usage

```
## S3 method for class 'eye_sbc_audit'
plot(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Object to print, plot, summarize, or audit. |
| `...` | Additional arguments passed to the selected model, engine, or method. |

## Value

Invisibly returns the plotting result when available; the primary effect is drawing plot SBC audit summaries.

---

`plot.eye_semantic_roundtrip`
*Plot semantic round-trip fidelity*

---

## Description

Plot semantic round-trip fidelity

## Usage

```
## S3 method for class 'eye_semantic_roundtrip'
plot(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Object to print, plot, summarize, or audit. |
| `...` | Additional arguments passed to the selected model, engine, or method. |

## Value

Invisibly returns the plotting result when available; the primary effect is drawing plot semantic round-trip fidelity.

---

`plot.eye_signal_filter_audit`
*Plot signal filter audit diagnostics*

---

## Description

Plot signal filter audit diagnostics

## Usage

```
## S3 method for class 'eye_signal_filter_audit'
plot(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Object to process, inspect, compare, or plot. |
| `...` | Additional arguments passed to the underlying method or helper. |

## Value

Invisibly returns the plotting result when available; the primary effect is drawing plot signal filter audit diagnostics.

## plot.eye_storage_benchmark

*plot eye storage benchmark*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_storage_benchmark'
plot(x, metric = c("write_seconds", "read_seconds",
  "bytes"), ...)
```

### Arguments

| | |
|---|---|
| x | Value for 'x'. See the function description and relevant article for constraints. |
| metric | Value for 'metric'. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## plot.eye_strategy_aoi_sensitivity

*plot eye strategy aoi sensitivity*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_strategy_aoi_sensitivity'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for 'x'. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

`plot.eye_streaming_score` *Plot streaming score diagnostics*

## Description

Plot streaming score diagnostics

## Usage

```
## S3 method for class 'eye_streaming_score'
plot(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Object to process, inspect, compare, or plot. |
| `...` | Additional arguments passed to the underlying method or helper. |

## Value

Invisibly returns the plotting result when available; the primary effect is drawing plot streaming score diagnostics.

`plot.eye_transition_diagnostics` *plot eye transition diagnostics*

## Description

S3 method supporting a research-scale eyeprocess object.

## Usage

```
## S3 method for class 'eye_transition_diagnostics'
plot(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Value for ‘x‘. See the function description and relevant article for constraints. |
| `...` | Additional arguments passed to the selected engine or method. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## plot.eye_validation_bundle

*Plot validation bundle diagnostics*

### Description

Plot validation bundle diagnostics

### Usage

```
## S3 method for class 'eye_validation_bundle'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

Invisibly returns the plotting result when available; the primary effect is drawing plot validation bundle diagnostics.

## plot.eye_vendor_compatibility_matrix

*plot eye vendor compatibility matrix*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_vendor_compatibility_matrix'
plot(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`plot.eye_vendor_semantic_validation`
*Plot eye vendor semantic validation*

---

## Description

Plot eye vendor semantic validation

## Usage

```
## S3 method for class 'eye_vendor_semantic_validation'
plot(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Object to print, plot, summarize, or audit. |
| `...` | Additional arguments passed to the selected model, engine, or method. |

## Value

Invisibly returns the plotting result when available; the primary effect is drawing plot eye vendor semantic validation.

---

`plot.eye_visual_context_irt`
*Plot visual context irt diagnostics*

---

## Description

Plot visual context irt diagnostics

## Usage

```
## S3 method for class 'eye_visual_context_irt'
plot(x, type = c("loadings", "difficulty_change", "context_registry"), ...)
```

## Arguments

| | |
|---|---|
| `x` | Object to process, inspect, compare, or plot. |
| `type` | Type of summary or visual representation to produce. |
| `...` | Additional arguments passed to the underlying method or helper. |

## Value

Invisibly returns the plotting result when available; the primary effect is drawing plot visual context irt diagnostics.

`plot_aoi_transition_matrix` *Plot an AOI transition matrix*

## Description

Plot an AOI transition matrix

## Usage

```
plot_aoi_transition_matrix(
  data,
  from = "from",
  to = "to",
  normalize = c("from", "all", "none"),
  ...
)
```

## Arguments

| | |
|---|---|
| `data` | Transition-pair data. |
| `from`, `to` | Column names. |
| `normalize` | Normalize within from-AOI, globally, or not at all. |
| `...` | Additional arguments passed to the underlying method or helper. |

## Value

An R object containing plot an AOI transition matrix. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

`plot_aoi_transition_rank` *Plot top AOI transitions by probability/count*

## Description

Plot top AOI transitions by probability/count

### Usage

```
plot_aoi_transition_rank(
  data,
  from = "from",
  to = "to",
  normalize = c("from", "all", "none"),
  top_n = 20L,
  ...
)
```

### Arguments

| | |
|---|---|
| data | Data frame containing the required process variables. |
| from | Name of the column identifying the transition origin. |
| to | Name of the column identifying the transition destination. |
| normalize | Normalization rule applied to transition counts or weights. |
| top_n | Maximum number of highest-ranked entries to display. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

An R object containing plot top AOI transitions by probability/count. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

## plot_distractor_information

*Plot option-level distractor information*

---

### Description

Plot option-level distractor information

### Usage

```
plot_distractor_information(object, ...)
```

### Arguments

| | |
|---|---|
| object | An ‘eye_nominal_gaze_irt‘ object or a data frame returned by ‘distractor_process_map()‘. |
| ... | Graphical arguments passed to base graphics. |

### Value

The plotted distractor map, invisibly.

## plot_gazepoint_workflow

*Generate the complete Gazepoint workflow plot suite*

### Description

Generate the complete Gazepoint workflow plot suite

### Usage

```
plot_gazepoint_workflow(x, directory, channels = NULL, expected_hz = 60)
```

### Arguments

| | |
|---|---|
| x | A processed 'eye_dataset'. |
| directory | Destination directory. |
| channels | Biometric channels to plot. |
| expected_hz | Expected gaze sampling rate shown in diagnostics. |

### Value

A plot manifest data frame.

## plot_interval_coverage

*Plot interval coverage*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
plot_interval_coverage(x, target = 0.95, engine = c("auto", "ggplot2", "base"), ...)
```

### Arguments

| | |
|---|---|
| x | Validation collection or recovery summary. |
| target | Nominal coverage target. |
| engine | Plot engine. |
| ... | Additional plotting arguments. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## plot_irf_uncertainty *Plot uncertainty for flexible item response functions*

### Description

For the bundled spline-reference engine, standard errors are derived on the logit scale from the fitted GLM and transformed to response probabilities. Exact GPIRT engines should supply their own posterior uncertainty summaries.

### Usage

```
plot_irf_uncertainty(
  object,
  item = 1L,
  theta_grid = seq(-4, 4, length.out = 101),
  level = 0.95,
  ...
)
```

### Arguments

| | |
|---|---|
| object | An ‘eye_gpirt‘ object. |
| item | Item name or index. |
| theta_grid | Trait grid. |
| level | Pointwise confidence level for the spline-reference diagnostic. |
| ... | Graphical arguments. |

### Value

A data frame containing plot uncertainty for flexible item response functions. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## plot_parameter_recovery *Plot parameter recovery*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
plot_parameter_recovery(x, parameter = NULL, engine = c("auto", "ggplot2", "base"),
  ...)
```

### Arguments

| | |
|---|---|
| x | Validation collection or estimates data frame. |
| parameter | Optional parameter subset. |
| engine | Plot engine. |
| ... | Additional plotting arguments. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

## plot_person_item_space

*Plot person/item latent-space coordinates*

---

### Description

Plot person/item latent-space coordinates

### Usage

```
plot_person_item_space(object, dimensions = c(1L, 2L), labels = FALSE, ...)
```

### Arguments

| | |
|---|---|
| object | Fitted ‘eye_latent_space_irt‘ object. |
| dimensions | Two coordinate dimensions to display. |
| labels | Whether to add entity labels. |
| ... | Graphical arguments. |

### Value

A named list with components "person", "item", "dimensions", containing plot person/item latent-space coordinates and associated metadata or diagnostics.

## plot_process_changepoint

*Plot detected process change points*

### Description

Plot detected process change points

### Usage

```
plot_process_changepoint(object, ...)
```

### Arguments

| | |
|---|---|
| object | An ‘eye_irt_changepoints‘, ‘eye_changepoint_rt_irt‘, or ‘eye_changepoint_multimodal_irt‘ object. |
| ... | Graphical arguments. |

### Value

An R object containing plot detected process change points. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## plot_process_channel_ablation_delta

*Plot channel-ablation delta from a full/reference model*

### Description

Plot channel-ablation delta from a full/reference model

### Usage

```
plot_process_channel_ablation_delta(
  x = NULL,
  table = NULL,
  channel_col = "channel",
  metric_col = "metric",
  value_col = "value",
  full_label = "full",
  metric = NULL,
  ...
)
```

### Arguments

| | |
|---|---|
| x | 'eye_process_channel_ablation' object or compatible table. |
| table | Optional explicit table. |
| channel_col, metric_col, value_col | Column names. |
| full_label | Full/reference channel label. |
| metric | Metric to evaluate or display. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

A data frame containing plot channel-ablation delta from a full/reference model. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

---

## plot_process_feature_stability

*Plot process-feature stability across resamples/splits*

---

### Description

Plot process-feature stability across resamples/splits

### Usage

```
plot_process_feature_stability(
  data,
  feature = "feature",
  stability = "selection_rate",
  top_n = 20L,
  ...
)
```

### Arguments

| | |
|---|---|
| data | Table with feature and stability/rank information. |
| feature | Feature column. |
| stability | Stability/selection-rate column. |
| top_n | Maximum features shown. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

An R object containing plot process-feature stability across resamples/splits. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## plot_process_window_sensitivity

*Explicit wrapper for process-window sensitivity plotting*

### Description

Explicit wrapper for process-window sensitivity plotting

### Usage

```
plot_process_window_sensitivity(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

An R object containing explicit wrapper for process-window sensitivity plotting. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## plot_pupil_activity_sensitivity

*Plot pupil activity sensitivity to window length*

### Description

Plot pupil activity sensitivity to window length

### Usage

```
plot_pupil_activity_sensitivity(x, feature = "pupil_frequency_contrast", ...)
```

### Arguments

| | |
|---|---|
| x | ‘eye_pupil_frequency_stability‘ object. |
| feature | Feature name. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

An R object containing plot pupil activity sensitivity to window length. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## plot_pupil_activity_windows *Plot pupil activity features across windows/groups*

### Description

Plot pupil activity features across windows/groups

### Usage

```
plot_pupil_activity_windows(x, feature = "pupil_frequency_contrast", ...)
```

### Arguments

| | |
|---|---|
| x | 'eye_pupil_frequency_features' or 'eye_pupil_frequency_stability' object. |
| feature | Activity feature. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

An R object containing plot pupil activity features across windows/groups. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## plot_pupil_band_power *Plot pupil low/high-band power summaries*

### Description

Plot pupil low/high-band power summaries

### Usage

```
plot_pupil_band_power(x, ...)
```

### Arguments

| | |
|---|---|
| x | 'eye_pupil_frequency_features' object. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

An R object containing plot pupil low/high-band power summaries. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## plot_pupil_components *Plot tonic/phasic pupil components*

### Description

Plot tonic/phasic pupil components

### Usage

```
plot_pupil_components(
  data,
  time = "time_ms",
  smoothed = "pupil_smoothed",
  tonic = "pupil_tonic",
  phasic = "pupil_phasic",
  ...
)
```

### Arguments

| | |
|---|---|
| data | Data frame containing the required process variables. |
| time | Time values or name of the time variable. |
| smoothed | Name of the smoothed pupil-signal column. |
| tonic | Name of the tonic pupil-component column. |
| phasic | Name of the phasic pupil-component column. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

A tabular R object containing plot tonic/phasic pupil components; rows represent analysis units and columns contain the returned quantities.

## plot_pupil_preprocessing_audit *Plot raw-to-processed pupil preprocessing stages*

### Description

Plot raw-to-processed pupil preprocessing stages

### Usage

```
plot_pupil_preprocessing_audit(
  data,
  time = "time_ms",
  signals = c("pupil_raw", "pupil_interpolated", "pupil_smoothed", "pupil_bc"),
  ...
)
```

### Arguments

| | |
|---|---|
| data | Sample-level data. |
| time | Time column. |
| signals | Signal columns to overlay. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

A tabular R object containing plot raw-to-processed pupil preprocessing stages; rows represent analysis units and columns contain the returned quantities.

---

## plot_pupil_spectrum *Plot a pupil-signal power spectrum*

---

### Description

Plot a pupil-signal power spectrum

### Usage

```
plot_pupil_spectrum(signal, sampling_rate_hz, max_hz = sampling_rate_hz/2, ...)
```

### Arguments

| | |
|---|---|
| signal | Numeric pupil signal. |
| sampling_rate_hz | Sampling rate. |
| max_hz | Maximum frequency shown; defaults to Nyquist. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

A data frame containing plot a pupil-signal power spectrum. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

`plot_sbc_rank` *Plot SBC rank histograms*

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
plot_sbc_rank(x, parameter = NULL, bins = 10L, engine = c("auto", "ggplot2", "base"),
  ...)
```

## Arguments

| | |
|---|---|
| `x` | Validation collection or draw data frame. |
| `parameter` | Optional parameter. |
| `bins` | Number of bins. |
| `engine` | Plot engine. |
| `...` | Additional plotting arguments. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`plot_validation_failures`
*Plot validation failure rates*

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
plot_validation_failures(x, engine = c("auto", "ggplot2", "base"), ...)
```

## Arguments

| | |
|---|---|
| `x` | Validation collection or failure summary. |
| `engine` | Plot engine. |
| `...` | Additional plotting arguments. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## plot_validation_runtime *Plot validation runtime*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
plot_validation_runtime(x, engine = c("auto", "ggplot2", "base"), ...)
```

### Arguments

| | |
|---|---|
| x | Validation collection or job table. |
| engine | Plot engine. |
| ... | Additional plotting arguments. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## posterior_predictive_discrepancies *Posterior predictive discrepancy table*

### Description

Posterior predictive discrepancy table

### Usage

```
posterior_predictive_discrepancies(
  observed,
  replicated,
  discrepancies = list(mean = function(x) mean(x, na.rm = TRUE), sd = function(x)
   stats::sd(x, na.rm = TRUE), zero_rate = function(x) mean(x == 0, na.rm = TRUE))
)
```

### Arguments

| | |
|---|---|
| observed | Observed vector/data object. |
| replicated | List of replicated datasets, or matrix with one replicate per row. |
| discrepancies | Named list of functions mapping a dataset to one number. |

## Value

An object of class "eye_irt_ppc", "data.frame", stored as a data frame, containing posterior predictive discrepancy table and associated metadata needed to interpret the result.

---

`posterior_sbc_contract`

*Define a posterior-SBC replication contract*

---

## Description

Posterior SBC is deliberately callback-driven: conditioning/augmentation differs by model, and eyeprocess does not pretend that re-running ordinary prior SBC on an observed-data neighbourhood is posterior SBC.

## Usage

```
posterior_sbc_contract(replication)
```

## Arguments

| | |
|---|---|
| `replication` | Function '(replicate, observed_data)' returning 'list(truth=named_numeric, draws=matrix_or_data_frame The callback is responsible for the conditional posterior-SBC construction appropriate to the model, including fitting to the observed data and the required self-consistency experiment. |

## Value

An object of class "eye_posterior_sbc_contract", stored as a named list, with components "replication", "requirement". It contains define a posterior-SBC replication contract and associated metadata or diagnostics needed to interpret the result.

---

`preaction_process_features`

*Build pre-action process features*

---

## Description

Build pre-action process features

### Usage

```
preaction_process_features(
  data,
  by = c("person_id", "trial_id"),
  time = "time_ms",
  response_time = "response_time_ms",
  windows_ms = c(500, 1000, 2000),
  aoi = "aoi",
  pupil = "pupil_bc",
  blink = "blink"
)
```

### Arguments

| | |
|---|---|
| `data` | Sample-level data. |
| `by` | Grouping columns. |
| `time`, `response_time` | Time and response-event columns. |
| `windows_ms` | Positive look-back windows before action. |
| `aoi`, `pupil`, `blink` | Optional feature columns. |

### Value

An object of class "eye_preaction_process_features", stored as a named list, with components "data", "windows_ms", "by", "status", "caveat". It contains pre-action process features and associated metadata or diagnostics needed to interpret the result.

---

## `predict.eye_censored_normal_process_irt` *Predict expected bounded response from a censored-normal process IRT fit*

---

### Description

Predict expected bounded response from a censored-normal process IRT fit

### Usage

```
## S3 method for class 'eye_censored_normal_process_irt'
predict(object, theta = object$theta, items = NULL, ...)
```

### Arguments

| | |
|---|---|
| `object` | A fitted eyeprocess model or audit object. |
| `theta` | Latent-trait values. |
| `items` | Items to include. |
| `...` | Additional arguments passed to the selected model, engine, or method. |

### Value

A vector or matrix containing expected bounded response from a censored-normal process IRT fit, with shape determined by the supplied analysis units.

---

## predict.eye_multinomial_transition

*Predict destination-state probabilities*

---

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_multinomial_transition'
predict(object, newdata = NULL,
  type = c("probability", "class", "link"), ...)
```

### Arguments

| | |
|---|---|
| object | Multinomial transition model. |
| newdata | Optional prepared transition data. |
| type | Probability, class, or linear predictor. |
| ... | Unused. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

## predict_aoi_trajectory

*Predict from an AOI growth curve*

---

### Description

Predict from an AOI growth curve

### Usage

```
predict_aoi_trajectory(object, time = NULL)
```

### Arguments

| | |
|---|---|
| object | Object supplied to the S3 method. |
| time | Time values or name of the time variable. |

### Value

A data frame containing from an AOI growth curve. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

---

## predict_item_parameter_priors *Predict pre-pilot item-parameter priors*

---

### Description

Predict pre-pilot item-parameter priors

### Usage

```
predict_item_parameter_priors(object, newdata)
```

### Arguments

| | |
|---|---|
| object | Seed model. |
| newdata | Candidate item feature data. |

### Value

An R object containing pre-pilot item-parameter priors. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

## predict_theta_at_time *Predict a latent trait at arbitrary times*

---

### Description

Predict a latent trait at arbitrary times

### Usage

```
predict_theta_at_time(object, time)
```

### Arguments

| | |
|---|---|
| object | A fitted eyeprocess model or audit object. |
| time | Time values. |

### Value

An R object containing a latent trait at arbitrary times. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## preflight_decisions *Extract pre-flight decisions*

### Description

Extract pre-flight decisions

### Usage

```
preflight_decisions(x)
```

### Arguments

| | |
|---|---|
| x | An ‘eye_biometric_preflight‘ object. |

### Value

An R object containing pre-flight decisions. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## preflight_exclusion_manifest *Create an explicit pre-flight exclusion/review manifest*

### Description

The manifest records recommendations only. It does not remove observations.

### Usage

```
preflight_exclusion_manifest(x)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |

### Value

A tabular R object containing an explicit pre-flight exclusion/review manifest; rows represent analysis units and columns contain the returned quantities.

## preflight_failures *Extract pre-flight failures/review cases*

### Description

Extract pre-flight failures/review cases

### Usage

```
preflight_failures(x)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |

### Value

A tabular R object containing pre-flight failures/review cases; rows represent analysis units and columns contain the returned quantities.

## preflight_passed *Extract pre-flight passes*

### Description

Extract pre-flight passes

### Usage

```
preflight_passed(x)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |

### Value

A tabular R object containing pre-flight passes; rows represent analysis units and columns contain the returned quantities.

## prepare_dynamic_irtree_data

*Prepare ordered transition data for dynamic IRTree models*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
prepare_dynamic_irtree_data(x, spec = dynamic_irtree_spec(),
  person = "participant_id", item = "item_id", trial = "trial_id", state = "state",
   time = NULL, from = "from_state", to = "to_state", states = NULL)
```

### Arguments

| | |
|---|---|
| x | An 'eye_dataset' or data frame. |
| spec | Dynamic IRTree specification. |
| person | Column names for long state data. |
| item | Column names for long state data. |
| trial | Column names for long state data. |
| state | Column names for long state data. |
| time | Column names for long state data. |
| from | Existing transition columns for transition-format data. |
| to | Existing transition columns for transition-format data. |
| states | Optional complete state vocabulary. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## prepare_functional_pupil_data

*Prepare aligned, corrected functional pupil data*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
prepare_functional_pupil_data(x, spec = functional_pupil_irt_spec())
```

## Arguments

| | |
|---|---|
| x | Eye dataset or long pupil data. |
| spec | Functional pupil specification. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`prepare_gaze_diffusion_data` *Prepare joint accuracy-response-time data*

---

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
prepare_gaze_diffusion_data(data, spec, minimum_rt = 0.05)
```

## Arguments

| | |
|---|---|
| data | Trial-level data. |
| spec | Diffusion specification. |
| minimum_rt | Minimum admissible RT in seconds. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

`prepare_multimodal_irt_data` *Prepare a canonical multimodal person-item-trial measurement object*

## Description

Creates a loss-aware person-item-trial table for response, response time, gaze, and pupil channels while retaining references to optional sample-level or sequence payloads. No time series are silently aggregated.

## Usage

```
prepare_multimodal_irt_data(
  data,
  person,
  item,
  trial = NULL,
  response = NULL,
  rt = NULL,
  gaze = NULL,
  pupil = NULL,
  quality = character(),
  device = NULL,
  payloads = list(),
  provenance = list()
)
```

## Arguments

| | |
|---|---|
| `data` | A data.frame containing one row per intended person-item-trial. |
| `person`, `item`, `trial` | Column names identifying the measurement keys. |
| `response`, `rt`, `gaze`, `pupil` | Optional column names for channel summaries. |
| `quality` | Optional character vector of quality-field names. |
| `device` | Optional list or data.frame of device metadata. |
| `payloads` | Named list of sample-level/sequence payloads. |
| `provenance` | Optional provenance list. |

## Value

An ‘eye_multimodal_measurement‘ object.

## prepare_strategy_mixture_data

*Prepare data for a strategy-mixture model*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
prepare_strategy_mixture_data(data, spec, standardize = TRUE)
```

### Arguments

| | |
|---|---|
| data | Trial-level data. |
| spec | Strategy specification. |
| standardize | Whether to standardize process features. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## prepare_structured_unstructured_process_features

*Prepare leakage-safe structured/unstructured process representations*

### Description

Prepare leakage-safe structured/unstructured process representations

### Usage

```
prepare_structured_unstructured_process_features(
  structured,
  unstructured = NULL,
  fold = NULL,
  builder = NULL,
  id = c("person_id", "item_id"),
  ...
)
```

### Arguments

| | |
|---|---|
| structured | Structured person/item/window features. |
| unstructured | Optional sequence/sample object. |
| fold | Fold identifier. If supplied, representation builders are applied fold-locally using ‘builder‘. |
| builder | Optional function ‘(train_structured, train_unstructured, test_structured, test_unstructured, fold_value, ...)‘ returning a fold result. |
| id | Optional identifier columns retained in the representation contract. |
| ... | Passed to ‘builder‘. |

### Value

An object of class "eye_structured_unstructured_process_features", stored as a named list, with components "contract", "folds", "status". It contains leakage-safe structured/unstructured process representations and associated metadata or diagnostics needed to interpret the result.

---

## preprocessing_multiverse

*Run a preprocessing or AOI multiverse*

---

### Description

Run a preprocessing or AOI multiverse

### Usage

```
preprocessing_multiverse(
  x,
  specifications,
  transform,
  analyse,
  extract = function(z) as.data.frame(z)
)
```

### Arguments

| | |
|---|---|
| x | Input object. |
| specifications | Named list of specification objects. |
| transform | Function receiving ‘x‘ and one specification. |
| analyse | Analysis function receiving the transformed object. |
| extract | Function converting an analysis result to a data frame. |

### Value

An ‘eye_multiverse‘ object.

## print.eye_benchmark_reproduction

*print eye benchmark reproduction*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_benchmark_reproduction'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_benchmark_study

*print eye benchmark study*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_benchmark_study'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_benchmark_validation

*print eye benchmark validation*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_benchmark_validation'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_diffusion_diagnostics

*print eye diffusion diagnostics*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_diffusion_diagnostics'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_diffusion_identification_study

*print eye diffusion identification study*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_diffusion_identification_study'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_dynamic_irtree

*print eye dynamic irtree*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_dynamic_irtree'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_dynamic_ppc *print eye dynamic ppc*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_dynamic_ppc'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_dynamic_recovery *print eye dynamic recovery*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_dynamic_recovery'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_engine_adapter_result

*print eye engine adapter result*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_engine_adapter_result'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_functional_pupil_data

*print eye functional pupil data*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_functional_pupil_data'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_functional_pupil_diagnostics

*print eye functional pupil diagnostics*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_functional_pupil_diagnostics'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_functional_pupil_irt

*print eye functional pupil irt*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_functional_pupil_irt'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_functional_pupil_sensitivity
*print eye functional pupil sensitivity*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_functional_pupil_sensitivity'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_functional_scalar_comparison
*print eye functional scalar comparison*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_functional_scalar_comparison'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_gated_process_model

*Print a gated process model object*

### Description

Print a gated process model object

### Usage

```
## S3 method for class 'eye_gated_process_model'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

Invisibly returns the input object after printing its summary; the object's class and contents are unchanged.

## print.eye_gaze_diffusion_data

*print eye gaze diffusion data*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_gaze_diffusion_data'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_gaze_diffusion_irt

*print eye gaze diffusion irt*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_gaze_diffusion_irt'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_gaze_diffusion_spec

*print eye gaze diffusion spec*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_gaze_diffusion_spec'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_irt_evidence_grade

*Print eye irt evidence grade*

### Description

Print eye irt evidence grade

### Usage

```
## S3 method for class 'eye_irt_evidence_grade'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to print, plot, summarize, or audit. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

Invisibly returns the input object after printing its summary; the object's class and contents are unchanged.

## print.eye_irt_model_spec

*Print a multimodal IRT model specification*

### Description

Print a multimodal IRT model specification

### Usage

```
## S3 method for class 'eye_irt_model_spec'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to print, plot, summarize, or audit. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

Invisibly returns the input object after printing its summary; the object's class and contents are unchanged.

## print.eye_irt_validation_spec

*Print eye irt validation spec*

### Description

Print eye irt validation spec

### Usage

```
## S3 method for class 'eye_irt_validation_spec'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to print, plot, summarize, or audit. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

Invisibly returns the input object after printing its summary; the object's class and contents are unchanged.

## print.eye_model_contract_validation

*print eye model contract validation*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_model_contract_validation'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_model_promotion_audit

*print eye model promotion audit*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_model_promotion_audit'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_multinomial_transition

*print eye multinomial transition*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_multinomial_transition'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_partitioned_storage

*print eye partitioned storage*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_partitioned_storage'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_partition_spec

*print eye partition spec*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_partition_spec'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`print.eye_process_drift_spec`
*Print a process drift spec object*

---

## Description

Print a process drift spec object

## Usage

```
## S3 method for class 'eye_process_drift_spec'
print(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Object to process, inspect, compare, or plot. |
| `...` | Additional arguments passed to the underlying method or helper. |

## Value

Invisibly returns the input object after printing its summary; the object's class and contents are unchanged.

---

`print.eye_process_negative_control`
*Print eye process negative control*

---

## Description

Print eye process negative control

## Usage

```
## S3 method for class 'eye_process_negative_control'
print(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Object to print, plot, summarize, or audit. |
| `...` | Additional arguments passed to the selected model, engine, or method. |

## Value

Invisibly returns the input object after printing its summary; the object's class and contents are unchanged.

## print.eye_process_preflight_spec

*Print a process preflight spec object*

### Description

Print a process preflight spec object

### Usage

```
## S3 method for class 'eye_process_preflight_spec'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

Invisibly returns the input object after printing its summary; the object's class and contents are unchanged.

## print.eye_process_window_spec

*Print a process window spec object*

### Description

Print a process window spec object

### Usage

```
## S3 method for class 'eye_process_window_spec'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

Invisibly returns the input object after printing its summary; the object's class and contents are unchanged.

---

`print.eye_redaction_result` *print eye redaction result*

---

## Description

S3 method supporting a research-scale eyeprocess object.

## Usage

```
## S3 method for class 'eye_redaction_result'
print(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Value for ‘x‘. See the function description and relevant article for constraints. |
| `...` | Additional arguments passed to the selected engine or method. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`print.eye_roundtrip_loss_audit` *print eye roundtrip loss audit*

---

## Description

S3 method supporting a research-scale eyeprocess object.

## Usage

```
## S3 method for class 'eye_roundtrip_loss_audit'
print(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Value for ‘x‘. See the function description and relevant article for constraints. |
| `...` | Additional arguments passed to the selected engine or method. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_strategy_data

*print eye strategy data*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_strategy_data'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_strategy_manipulation_validation

*print eye strategy manipulation validation*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_strategy_manipulation_validation'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

`print.eye_theory_strategy_irt` *print eye theory strategy irt*

## Description

S3 method supporting a research-scale eyeprocess object.

## Usage

```
## S3 method for class 'eye_theory_strategy_irt'
print(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Value for ‘x‘. See the function description and relevant article for constraints. |
| `...` | Additional arguments passed to the selected engine or method. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

`print.eye_theory_strategy_spec` *print eye theory strategy spec*

## Description

S3 method supporting a research-scale eyeprocess object.

## Usage

```
## S3 method for class 'eye_theory_strategy_spec'
print(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Value for ‘x‘. See the function description and relevant article for constraints. |
| `...` | Additional arguments passed to the selected engine or method. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_transition_design

*print eye transition design*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_transition_design'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_transition_diagnostics

*print eye transition diagnostics*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_transition_diagnostics'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_validation_bundle

*Print a validation bundle object*

### Description

Print a validation bundle object

### Usage

```
## S3 method for class 'eye_validation_bundle'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

Invisibly returns the input object after printing its summary; the object's class and contents are unchanged.

## print.eye_validation_case_fingerprint

*print eye validation case fingerprint*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_validation_case_fingerprint'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_validation_collection

*print eye validation collection*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_validation_collection'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_validation_completion_audit

*print eye validation completion audit*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_validation_completion_audit'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_validation_job_plan

*print eye validation job plan*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_validation_job_plan'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_validation_run

*print eye validation run*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_validation_run'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_vendor_case *print eye vendor case*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_vendor_case'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_vendor_compatibility_matrix *print eye vendor compatibility matrix*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_vendor_compatibility_matrix'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_vendor_schema_contract

*Print eye vendor schema contract*

### Description

Print eye vendor schema contract

### Usage

```
## S3 method for class 'eye_vendor_schema_contract'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to print, plot, summarize, or audit. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

Invisibly returns the input object after printing its summary; the object's class and contents are unchanged.

## print.eye_vendor_semantic_comparison

*print eye vendor semantic comparison*

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_vendor_semantic_comparison'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Value for ‘x‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## print.eye_visual_context_registry *Print a visual context registry object*

### Description

Print a visual context registry object

### Usage

```
## S3 method for class 'eye_visual_context_registry'
print(x, ...)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

Invisibly returns the input object after printing its summary; the object's class and contents are unchanged.

## probabilistic_aoi_assignment *Probabilistic AOI assignment under empirical calibration uncertainty*

### Description

Probabilistic AOI assignment under empirical calibration uncertainty

### Usage

```
probabilistic_aoi_assignment(
  data,
  aois,
  model,
  x = "gaze_x",
  y = "gaze_y",
  draws = 500L,
  seed = 1L,
  min_probability = 0.5
)
```

### Arguments

| | |
|---|---|
| `data` | Gaze samples. |
| `aois` | Rectangular AOI table. |
| `model` | Calibration error model. |
| `x, y` | Gaze-coordinate columns. |
| `draws` | Monte Carlo draws. |
| `seed` | Seed. |
| `min_probability` | Minimum probability for assignment; lower maxima become 'NA'. |

### Value

An object of class "eye_probabilistic_aoi_assignment", stored as a named list, with components "assignments", "probabilities", "aois", "model", "min_probability", "caveat". It contains probabilistic AOI assignment under empirical calibration uncertainty and associated metadata or diagnostics needed to interpret the result.

---

## `process_anomaly_distance` *Extract multivariate process anomaly distances*

---

### Description

Extract multivariate process anomaly distances

### Usage

```
process_anomaly_distance(x)
```

### Arguments

| | |
|---|---|
| `x` | Object to process, inspect, compare, or plot. |

### Value

An R object containing multivariate process anomaly distances. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## process_bland_altman *Bland-Altman repeatability summary for two sessions*

### Description

Bland-Altman repeatability summary for two sessions

### Usage

```
process_bland_altman(data, person, session, measure, sessions = NULL)
```

### Arguments

| | |
|---|---|
| data | Long data. |
| person | Participant column. |
| session | Session column containing exactly two selected sessions. |
| measure | Measure column. |
| sessions | Optional two session labels. |

### Value

An object of class "eye_process_bland_altman", stored as a named list, with components "pairs", "summary", "sessions". It contains bland-Altman repeatability summary for two sessions and associated metadata or diagnostics needed to interpret the result.

## process_channel_ablation *Ablate process channels under a common out-of-sample evaluator*

### Description

Ablate process channels under a common out-of-sample evaluator

### Usage

```
process_channel_ablation(
  data,
  channels,
  evaluator,
  baseline = character(),
  higher_is_better = TRUE
)
```

### Arguments

| | |
|---|---|
| data | Input data. |
| channels | Named list whose elements are character vectors of columns. |
| evaluator | Function ‘(data, active_columns, channel_name)‘ returning a scalar out-of-sample score. The evaluator owns all fitting/splitting logic. |
| baseline | Character vector of always-active columns. |
| higher_is_better | Direction of the score. |

### Value

An object of class "eye_process_channel_ablation", "data.frame", stored as a data frame, containing ablate process channels under a common out-of-sample evaluator and associated metadata needed to interpret the result.

---

## process_criterion_associations

*Extract process-criterion associations*

---

### Description

Extract process-criterion associations

### Usage

```
process_criterion_associations(x)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |

### Value

An R object containing process-criterion associations. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

process_dependent_discrimination_audit
*Audit process-dependent item discrimination*

## Description

Residualises a process variable against person/item baselines, then tests whether the theta-response slope changes with that residual process value. This is a transparent diagnostic inspired by 2026 evidence on conditional response-time/discrimination dependencies, not an exact reproduction of the published meta-analytic model.

## Usage

```
process_dependent_discrimination_audit(
  data,
  response,
  theta,
  process,
  person,
  item,
  nonlinear = TRUE
)
```

## Arguments

| | |
|---|---|
| data | Long-format response data. |
| response | Binary response column. |
| theta | Person latent-score column. |
| process | RT/gaze/process measure. |
| person, item | Identifier columns. |
| nonlinear | If TRUE and mgcv is installed, additionally estimate a smooth theta-by-process diagnostic surface. |

## Value

An object of class "eye_process_dependent_discrimination", stored as a named list, with components "process_model", "response_model", "interaction", "smooth_model", "residual_process", "status", "caveat". It contains process-dependent item discrimination and associated metadata or diagnostics needed to interpret the result.

## process_dif_nuisance_surrogate *Construct a process-data nuisance surrogate for DIF analysis*

### Description

Construct a process-data nuisance surrogate for DIF analysis

### Usage

```
process_dif_nuisance_surrogate(
  data,
  process_features,
  person = "participant_id",
  aggregate = TRUE
)
```

### Arguments

| | |
|---|---|
| data | Input data frame or compatible tabular object. |
| process_features | Names of process-derived features. |
| person | Person or participant identifier column. |
| aggregate | Aggregation rule for process features. |

### Value

A data frame containing a process-data nuisance surrogate for DIF analysis. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## process_drift_alerts *Extract drift alerts*

### Description

Extract drift alerts

### Usage

```
process_drift_alerts(x)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |

### Value

A logical value or vector indicating drift alerts.

---

`process_drift_spec` *Specify a process-deployment drift audit*

---

### Description

Specify a process-deployment drift audit

### Usage

```
process_drift_spec(
  baseline = c("first_batch", "reference_batch"),
  difficulty_limit = 0.4,
  discrimination_limit = 0.35,
  gaze_validity_drop = 0.1,
  luminance_limit = 25,
  relative_metric_quantile = 0.9,
  min_batches = 2L
)
```

### Arguments

| | |
|---|---|
| `baseline` | Baseline rule: first observed batch or an explicitly supplied reference batch. |
| `difficulty_limit` | Absolute item-difficulty change triggering review. |
| `discrimination_limit` | Absolute discrimination change triggering review. |
| `gaze_validity_drop` | Maximum tolerated decrease in gaze validity. |
| `luminance_limit` | Absolute luminance change triggering review. |
| `relative_metric_quantile` | Quantile of absolute deltas used for process metrics without an externally meaningful absolute threshold. |
| `min_batches` | Minimum number of batches per item for drift assessment. |

### Value

An ‘eye_process_drift_spec‘ object.

process_feature_blocks *Define conceptual process-feature blocks*

## Description

Define conceptual process-feature blocks

## Usage

```
process_feature_blocks(data, blocks, id = NULL, drop_constant = TRUE)
```

## Arguments

| | |
|---|---|
| data | Data frame. |
| blocks | Named list of column names for conceptual blocks. |
| id | Optional identifier column. |
| drop_constant | Remove non-varying columns. |

## Value

An object of class "eye_process_feature_blocks", stored as a named list, with components "data", "blocks", "id", "block_sizes", "status". It contains define conceptual process-feature blocks and associated metadata or diagnostics needed to interpret the result.

process_feature_family_registry *Registry of process-feature families and interpretation guardrails*

## Description

Registry of process-feature families and interpretation guardrails

## Usage

```
process_feature_family_registry()
```

## Value

A data frame containing registry of process-feature families and interpretation guardrails. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

process_feature_stability *Summarize process-feature stability across repeated analyses*

## Description

Summarize process-feature stability across repeated analyses

## Usage

```
process_feature_stability(
  data,
  feature = "feature",
  split = "split",
  importance = "importance",
  top_n = 20L
)
```

## Arguments

| | |
|---|---|
| data | Long table containing feature names and ranks/importance values. |
| feature | Feature column. |
| split | Split/resample column. |
| importance | Importance column, where larger is better. |
| top_n | Number of top features counted per split. |

## Value

A data frame containing process-feature stability across repeated analyses. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

process_feature_time_provenance *Declare temporal provenance for process features*

## Description

Declare temporal provenance for process features

### Usage

```
process_feature_time_provenance(
  feature,
  available_at,
  outcome_at,
  source = NA_character_,
  transformation = NA_character_,
  unit = "ms"
)
```

### Arguments

| | |
|---|---|
| feature | Feature names. |
| available_at | Earliest time at which each feature is available. |
| outcome_at | Time at which the modeled outcome becomes available. |
| source | Optional source labels. |
| transformation | Optional transformation descriptions. |
| unit | Time unit label. |

### Value

A provenance table.

---

## process_icc *Absolute-agreement ICC(A,1) for repeated process measures*

---

### Description

Absolute-agreement ICC(A,1) for repeated process measures

### Usage

```
process_icc(data, person, session, measure)
```

### Arguments

| | |
|---|---|
| data | Long data. |
| person | Participant column. |
| session | Session/repetition column. |
| measure | Measure column. |

### Value

A data frame containing absolute-agreement ICC(A,1) for repeated process measures. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

process_information *Quantify incremental process information*

## Description

Compares posterior draws for the same latent target under a baseline and an augmented model. This implementation intentionally uses posterior uncertainty metrics rather than assuming Fisher information is additively decomposable across heterogeneous channels.

## Usage

```
process_information(
  baseline,
  augmented,
  metric = c("variance_reduction", "precision_gain", "entropy_reduction")
)
```

## Arguments

| | |
|---|---|
| baseline, augmented | Numeric posterior draws. Rows are draws and columns are matched latent targets. |
| metric | 'variance_reduction', 'precision_gain', or 'entropy_reduction'. |

## Value

An 'eye_process_information' data.frame.

process_item_information *2PL response item information*

## Description

2PL response item information

## Usage

```
process_item_information(
  theta,
  a,
  b,
  process_information = 0,
  rt_information = 0,
  weights = c(response = 1, rt = 0, process = 0),
  expected_time = 0,
  burden_weight = 0
)
```

### Arguments

| | |
|---|---|
| theta | Latent-trait values. |
| a | Item discrimination parameter or parameters. |
| b | Item difficulty/location parameter or parameters. |
| process_information | Information supplied by the process channel. |
| rt_information | Information supplied by response time. |
| weights | Weights used to combine information components. |
| expected_time | Expected response time or burden. |
| burden_weight | Penalty applied to expected burden. |

### Value

An object of class "eye_process_item_information", stored as a named list, with components "theta", "response_information", "utility". It contains 2PL response item information and associated metadata or diagnostics needed to interpret the result.

---

## process_measure_card *Return a one-measure process card*

---

### Description

Return a one-measure process card

### Usage

```
process_measure_card(name, registry = process_measure_registry())
```

### Arguments

| | |
|---|---|
| name | Measure name. |
| registry | Registry. |

### Value

An object of class "eye_process_measure_card", stored as a named list, containing return a one-measure process card and associated metadata needed to interpret the result.

process_measure_coverage *Process-measure coverage for an observed dataset*

## Description

Process-measure coverage for an observed dataset

## Usage

```
process_measure_coverage(data, registry = process_measure_registry())
```

## Arguments

| | |
|---|---|
| data | Data frame. |
| registry | Registry. |

## Value

A data frame containing process-measure coverage for an observed dataset. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

process_measure_guardrails *Process-measure guardrail table*

## Description

Process-measure guardrail table

## Usage

```
process_measure_guardrails(registry = process_measure_registry())
```

## Arguments

| | |
|---|---|
| registry | Registry. |

## Value

An R object containing process-measure guardrail table. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

process_measure_lineage *Process-measure lineage table*

## Description

Process-measure lineage table

## Usage

```
process_measure_lineage(
  measure,
  inputs,
  transformations = character(),
  output_level = NA_character_
)
```

## Arguments

| | |
|---|---|
| measure | Measure name. |
| inputs | Required input variable names. |
| transformations | Ordered transformation labels. |
| output_level | Output aggregation level. |

## Value

An object of class "eye_process_measure_lineage", stored as a named list, with components "measure", "inputs", "transformations", "output_level", "lineage_hash". It contains process-measure lineage table and associated metadata or diagnostics needed to interpret the result.

process_measure_registry *Unified process-measure registry*

## Description

Unified process-measure registry

## Usage

```
process_measure_registry(include_experimental = TRUE)
```

### Arguments

include_experimental
: Include experimental registry entries.

### Value

A data frame containing unified process-measure registry. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

---

## process_measure_units *List units used by registered process measures*

---

### Description

List units used by registered process measures

### Usage

```
process_measure_units(registry = process_measure_registry())
```

### Arguments

registry
: Registry.

### Value

An R object containing list units used by registered process measures. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

## process_negative_control_permute *Permutation negative control*

---

### Description

Permutation negative control

### Usage

```
process_negative_control_permute(data, outcome, seed = 1L, within = NULL)
```

### Arguments

| | |
|---|---|
| data | Data frame. |
| outcome | Outcome column. |
| seed | Random seed. |
| within | Optional grouping columns within which to permute. |

### Value

An R object containing permutation negative control. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

## process_negative_control_shift

*Temporal-shift negative control*

---

### Description

Temporal-shift negative control

### Usage

```
process_negative_control_shift(data, column, lag = 1L, by = NULL)
```

### Arguments

| | |
|---|---|
| data | Data frame. |
| column | Column to shift. |
| lag | Number of rows to shift within each group. |
| by | Optional grouping columns. |

### Value

An R object containing temporal-shift negative control. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

process_ngram_features *N-gram features from process sequences*

## Description

N-gram features from process sequences

## Usage

```
process_ngram_features(sequence, n = c(1L, 2L, 3L), separator = ">")
```

## Arguments

| | |
|---|---|
| sequence | Sequence input. |
| n | Requested count or n-gram order, depending on context. |
| separator | Sequence-token separator. |

## Value

An object of class "matrix", stored as an R object, containing n-gram features from process sequences and associated metadata needed to interpret the result.

process_null_benchmark *Compare an observed effect against a negative-control null distribution*

## Description

Compare an observed effect against a negative-control null distribution

## Usage

```
process_null_benchmark(observed, controls, effect = "effect")
```

## Arguments

| | |
|---|---|
| observed | Observed scalar effect. |
| controls | Negative-control result or numeric vector. |
| effect | Effect column when controls is an object. |

## Value

A named list with components "observed", "n_null", "null_mean", "null_sd", "percentile", "two_sided_tail", "standardized_distance", containing an observed effect against a negative-control null distribution and associated metadata or diagnostics.

## process_person_fit *Joint response-process person-fit diagnostic*

### Description

Produces model-discrepancy evidence; it never labels a participant as dishonest, impaired, disengaged, or otherwise psychologically categorized.

### Usage

```
process_person_fit(
  object,
  data = NULL,
  person = NULL,
  response_weight = 1,
  rt_weight = 1,
  process_weight = 1
)
```

### Arguments

| | |
|---|---|
| object | A fitted eyeprocess model or audit object. |
| data | Input data frame or compatible tabular object. |
| person | Person or participant identifier column. |
| response_weight | Weight assigned to response discrepancy. |
| rt_weight | Weight assigned to response-time discrepancy. |
| process_weight | Weight assigned to process discrepancy. |

### Value

An object of class "eye_process_person_fit", "data.frame", stored as a data frame, containing joint response-process person-fit diagnostic and associated metadata needed to interpret the result.

## process_preflight_spec *Specify a biometric process pre-flight gate*

### Description

Defines transparent, review-oriented thresholds for incoming gaze/pupil data. The specification is a data-quality governance object, not a behavioral or clinical classifier.

## Usage

```
process_preflight_spec(
  min_gaze_validity = 0.8,
  min_pupil_validity = 0.7,
  max_gaze_missingness = 0.25,
  max_pupil_missingness = 0.3,
  min_valid_trial_fraction = 0.7,
  trial_gaze_validity_threshold = 0.75,
  min_rt_ms = 200,
  max_rt_ms = 10000,
  sampling_rate_tolerance = 0.2,
  blink_quantile = 0.95,
  caution_flags = 1L,
  review_flags = 2L
)
```

## Arguments

| | |
|---|---|
| `min_gaze_validity` | Minimum mean gaze-validity proportion. |
| `min_pupil_validity` | Minimum mean pupil-validity proportion. |
| `max_gaze_missingness` | Maximum mean gaze-missingness proportion. |
| `max_pupil_missingness` | Maximum mean pupil-missingness proportion. |
| `min_valid_trial_fraction` | Minimum fraction of trials meeting the trial-level gaze-validity threshold. |
| `trial_gaze_validity_threshold` | Gaze-validity threshold used to count an acceptable trial. |
| `min_rt_ms`, `max_rt_ms` | Plausible mean response-time bounds in milliseconds. |
| `sampling_rate_tolerance` | Fractional deviation from the cohort median sampling rate that triggers review. |
| `blink_quantile` | Cohort quantile used for an extreme blink-cluster flag. |
| `caution_flags` | Number of flags yielding ‘use_with_caution‘. |
| `review_flags` | Number of flags yielding ‘review_or_exclude_from_biometric_models‘. |

## Value

An ‘eye_process_preflight_spec‘ object.

process_profile_probabilities
*Extract process-profile probabilities*

## Description

Extract process-profile probabilities

## Usage

```
process_profile_probabilities(x)
```

## Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |

## Value

An R object containing process-profile probabilities. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

process_profile_summary
*Summarize process profiles*

## Description

Summarize process profiles

## Usage

```
process_profile_summary(x)
```

## Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |

## Value

An R object containing process profiles. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## process_reliability_profile *Test-retest process reliability profile*

### Description

Test-retest process reliability profile

### Usage

```
process_reliability_profile(data, person, session, measure)
```

### Arguments

| | |
|---|---|
| data | Long data. |
| person, session, measure | Column names. |

### Value

An object of class "eye_process_reliability_profile", stored as a named list, with components "measure", "icc", "bland_altman", "caveat". It contains test-retest process reliability profile and associated metadata or diagnostics needed to interpret the result.

## process_residual_map *Return person/item latent-space coordinates*

### Description

Return person/item latent-space coordinates

### Usage

```
process_residual_map(object, entity = c("both", "person", "item"))
```

### Arguments

| | |
|---|---|
| object | A fitted eyeprocess model or audit object. |
| entity | Entity type to map or validate. |

### Value

A tabular R object containing return person/item latent-space coordinates; rows represent analysis units and columns contain the returned quantities.

## process_sensitivity_grid

*Construct an explicit process-analysis sensitivity grid*

### Description

Construct an explicit process-analysis sensitivity grid

### Usage

```
process_sensitivity_grid(
  ...,
  label = "process_sensitivity",
  max_specifications = 100000L
)
```

### Arguments

| | |
|---|---|
| `...` | Named vectors of defensible analysis options. |
| `label` | Grid label. |
| `max_specifications` | Safety cap. |

### Value

A tabular R object containing an explicit process-analysis sensitivity grid; rows represent analysis units and columns contain the returned quantities.

## process_sequence_embedding

*Low-dimensional embedding of response-process sequences*

### Description

Uses TF-IDF-weighted n-gram features and truncated SVD. This is a transparent classical embedding; sequence autoencoders can be supplied later through an external engine without changing the downstream contract.

### Usage

```
process_sequence_embedding(sequence, n = c(1L, 2L, 3L), dimensions = 5L)
```

### Arguments

| | |
|---|---|
| sequence | Sequence input. |
| n | Requested count or n-gram order, depending on context. |
| dimensions | Number of embedding dimensions. |

### Value

An R object containing low-dimensional embedding of response-process sequences. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

## process_state_occupancy

*Summarize HMM state occupancy*

---

### Description

Summarize HMM state occupancy

### Usage

```
process_state_occupancy(object)
```

### Arguments

| | |
|---|---|
| object | A fitted eyeprocess model or audit object. |

### Value

An R object containing hMM state occupancy. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

## process_state_transition_summary

*Summarize HMM process-state transitions*

---

### Description

Summarize HMM process-state transitions

### Usage

```
process_state_transition_summary(object)
```

### Arguments

| | |
|---|---|
| `object` | A fitted eyeprocess model or audit object. |

### Value

A data frame containing hMM process-state transitions. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

---

## `process_temporal_stability` *Pairwise temporal stability across sessions*

---

### Description

Pairwise temporal stability across sessions

### Usage

```
process_temporal_stability(
  data,
  person,
  session,
  measure,
  method = c("pearson", "spearman")
)
```

### Arguments

| | |
|---|---|
| `data` | Long data. |
| `person`, `session`, `measure` | Column names. |
| `method` | Correlation method. |

### Value

A logical value or vector indicating pairwise temporal stability across sessions.

process_uncertainty_spec

*Process measurement-uncertainty budgets*

## Description

Process measurement-uncertainty budgets. These functions form the eyeprocess 0.6.0.9000 measurement-intelligence programme and use dependency-free reference implementations with explicit evidence limits.

## Usage

```
process_uncertainty_spec(calibration = TRUE, aoi_assignment = TRUE,
  preprocessing = TRUE, sampling = TRUE, model = TRUE, source_sd = NULL,
  draws = 1000, seed = 20260807)
estimate_process_uncertainty(x, spec = process_uncertainty_spec(), metrics = NULL,
  cluster = NULL)
propagate_process_uncertainty(x, estimand = function(data) mean(data, na.rm = TRUE),
  method = c("bootstrap", "simulation", "posterior"), draws = NULL, seed = NULL)
uncertainty_budget(x)
compare_uncertainty_budgets(...)
plot_uncertainty_waterfall(x, ...)
plot_uncertainty_tornado(x, ...)
plot_uncertainty_by_item(x, ...)
plot_uncertainty_by_stage(x, ...)
```

## Arguments

| | |
|---|---|
| calibration | Argument controlling ‘calibration‘; see the function usage and returned audit metadata. |
| aoi_assignment | Argument controlling ‘aoi_assignment‘; see the function usage and returned audit metadata. |
| preprocessing | Argument controlling ‘preprocessing‘; see the function usage and returned audit metadata. |
| sampling | Argument controlling ‘sampling‘; see the function usage and returned audit metadata. |
| model | Argument controlling ‘model‘; see the function usage and returned audit metadata. |
| source_sd | Argument controlling ‘source_sd‘; see the function usage and returned audit metadata. |
| draws | Argument controlling ‘draws‘; see the function usage and returned audit metadata. |
| seed | Argument controlling ‘seed‘; see the function usage and returned audit metadata. |

| | |
|---|---|
| x | Input object or data structure appropriate for the selected analysis. |
| spec | Argument controlling 'spec'; see the function usage and returned audit metadata. |
| metrics | Argument controlling 'metrics'; see the function usage and returned audit metadata. |
| cluster | Argument controlling 'cluster'; see the function usage and returned audit metadata. |
| estimand | Argument controlling 'estimand'; see the function usage and returned audit metadata. |
| method | Argument controlling 'method'; see the function usage and returned audit metadata. |
| ... | Additional arguments passed to the underlying method or plotting function. |

## Details

The APIs return auditable S3 objects. Plot wrappers call registered base-graphics methods. Experimental or approximate engines are labelled in object status fields and should be validated before confirmatory or operational use.

## Value

An eyeprocess result object, data frame, model object, plot, or audit table as documented by the individual function.

## See Also

`plot_diagnostics()`, `plot_evidence()`, and `plot_sensitivity()`.

---

`process_validation_design`

*Define an empirical process-validation design*

---

## Description

The design is intentionally explicit. It records measurement conditions under which recovery, uncertainty, convergence, and failure behavior will be evaluated. It does not imply that every combination is appropriate for every estimator.

## Usage

```
process_validation_design(
  n_persons = c(50L, 150L, 500L),
  n_trials = c(10L, 30L, 80L),
  missingness = c(0, 0.05, 0.15, 0.3),
  sampling_rate_hz = c(60, 120, 300, 1000),
  aoi_error = c("low", "moderate", "severe"),
```

```
  calibration_error = c(0, 0.5, 1),
  pupil_dropout = c(0, 0.1, 0.3),
  heterogeneity = c("low", "moderate"),
  model_misspecification = c(FALSE, TRUE),
  replications = 100L,
  seed = 1L,
  label = "process_validation"
)
```

## Arguments

| | |
|---|---|
| n_persons | Participant counts. |
| n_trials | Trial/item counts per participant. |
| missingness | Proportion of generic process observations made missing. |
| sampling_rate_hz | Nominal sampling rates. |
| aoi_error | AOI uncertainty regimes. |
| calibration_error | Calibration-error regimes in user-defined units. |
| pupil_dropout | Pupil-specific dropout proportions. |
| heterogeneity | Participant-heterogeneity regimes. |
| model_misspecification | Logical regimes indicating deliberate mismatch. |
| replications | Monte Carlo replications per condition. |
| seed | Master simulation seed. |
| label | Optional design label. |

## Value

An ‘eye_process_validation_design‘ object.

---

# process_window_spec *Specify temporal process windows*

---

## Description

Specify temporal process windows

### Usage

```
process_window_spec(
  width_ms = 1000,
  step_ms = 500,
  start_ms = 0,
  end_ms = 3000,
  align = c("stimulus", "response", "custom"),
  min_samples = 5L
)
```

### Arguments

| | |
|---|---|
| width_ms | Window width in milliseconds. |
| step_ms | Step between successive windows. |
| start_ms, end_ms | Analysis range relative to the chosen alignment origin. |
| align | Alignment label, e.g. stimulus, response, or custom. |
| min_samples | Minimum samples required within a window. |

### Value

An ‘eye_process_window_spec‘ object.

---

## promote_irt_model *Promote an IRT model after evidence gates are met*

---

### Description

Promotion is an evidence record, not a mutable global status change unless ‘update_registry = TRUE‘ is requested.

### Usage

```
promote_irt_model(
  spec,
  evidence,
  target = c("experimental", "reference"),
  update_registry = FALSE
)
```

### Arguments

| | |
|---|---|
| spec | IRT model or validation specification. |
| evidence | Validation evidence used for promotion. |
| target | Target evidence/status level. |
| update_registry | Whether the in-memory registry is updated. |

### Value

An object of class "eye_irt_promotion", stored as a named list, with components "model", "from", "to", "evidence_grade", "evidence_pass", "timestamp". It contains promote an IRT model after evidence gates are met and associated metadata or diagnostics needed to interpret the result.

---

## promote_vendor_support

*Promote a case support level only when evidence is supplied*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
promote_vendor_support(corpus_path, case_id, level = c("fixture-tested",
  "empirically-validated"), validation, reviewer, notes = NA_character_)
```

### Arguments

| | |
|---|---|
| corpus_path | Corpus directory. |
| case_id | Case identifier. |
| level | New support level. |
| validation | Validation result/audit. |
| reviewer | Reviewer identifier. |
| notes | Notes. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

## propagate_calibration_uncertainty

*Propagate empirical calibration uncertainty around gaze samples*

---

### Description

Propagate empirical calibration uncertainty around gaze samples

### Usage

```
propagate_calibration_uncertainty(
  data,
  model,
  x = "gaze_x",
  y = "gaze_y",
  draws = 500L,
  seed = 1L
)
```

### Arguments

| | |
|---|---|
| data | Gaze samples. |
| model | Calibration error model. |
| x, y | Gaze-coordinate columns. |
| draws | Monte Carlo draws per sample. |
| seed | Seed. |

### Value

An R object containing propagate empirical calibration uncertainty around gaze samples. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

## provenance_edge_table *Build a provenance edge table*

---

### Description

Build a provenance edge table

### Usage

```
provenance_edge_table(from, to, relation = "wasDerivedFrom")
```

### Arguments

| | |
|---|---|
| from, to | Node identifiers. |
| relation | PROV-like relation labels. |

### Value

A data frame containing a provenance edge table. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## provenance_lineage_table

*Build a provenance lineage node table*

### Description

Build a provenance lineage node table

### Usage

```
provenance_lineage_table(
  id,
  type = "entity",
  label = id,
  value = NA_character_
)
```

### Arguments

| | |
|---|---|
| id | Node identifiers. |
| type | Node types such as entity, activity, agent. |
| label | Human-readable labels. |
| value | Optional values/locations. |

### Value

A data frame containing a provenance lineage node table. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## prune_validation_checkpoints

*Remove obsolete or corrupt validation checkpoints*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
prune_validation_checkpoints(path, statuses = c("corrupt", "locked"), dry_run = TRUE)
```

### Arguments

| | |
|---|---|
| path | Validation output directory. |
| statuses | Statuses to remove. |
| dry_run | Report without deleting. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

## public_validation_corpus

*Public validation-corpus registry*

---

### Description

Returns conservative metadata only. The package deliberately does not auto-download third-party human-participant data; users must review the source licence/terms and obtain data from the authoritative repository.

### Usage

```
public_validation_corpus()
```

### Value

A data frame containing public validation-corpus registry. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

---

## pupil_activity_index *Compute a transparent pupil activity index*

---

### Description

Compute a transparent pupil activity index

### Usage

```
pupil_activity_index(
  y,
  time_ms = seq_along(y),
  sampling_rate_hz = NULL,
  method = c("velocity", "frequency_contrast", "ripa_proxy"),
  low_band = c(0.05, 0.5),
  high_band = c(0.5, 4),
  fast_window_ms = 250,
  slow_window_ms = 750
)
```

### Arguments

| | |
|---|---|
| y | Pupil signal. |
| time_ms | Time vector. |
| sampling_rate_hz | Sampling rate for frequency methods. |
| method | 'velocity', 'frequency_contrast', or 'ripa_proxy'. |
| low_band, high_band | Frequency bands for frequency contrast. |
| fast_window_ms, slow_window_ms | Smoothing windows for the RIPA-style proxy. |

### Value

A numeric value or vector containing a transparent pupil activity index.

---

## pupil_band_power *Compute pupil signal power in a frequency band*

---

### Description

Compute pupil signal power in a frequency band

### Usage

```
pupil_band_power(y, sampling_rate_hz, lower_hz, upper_hz, detrend = TRUE)
```

### Arguments

| | |
|---|---|
| y | Pupil signal. |
| sampling_rate_hz | Sampling rate in Hz. |
| lower_hz, upper_hz | Frequency-band limits. |
| detrend | Remove the mean before FFT. |

### Value

A numeric value or vector containing pupil signal power in a frequency band.

## pupil_baseline_sensitivity

*Evaluate pupil baseline-window sensitivity*

### Description

Evaluate pupil baseline-window sensitivity

### Usage

```
pupil_baseline_sensitivity(
  data,
  time = "time_ms",
  pupil = "pupil",
  windows,
  by = NULL,
  correction = c("subtractive", "divisive")
)
```

### Arguments

| | |
|---|---|
| data | Pupil data. |
| time | Time column. |
| pupil | Pupil column. |
| windows | Named list of two-element baseline windows. |
| by | Optional grouping columns. |
| correction | 'subtractive' or 'divisive'. |

### Value

An R object containing pupil baseline-window sensitivity. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## pupil_confound_effects

*Extract pupil confound-model effects*

### Description

Extract pupil confound-model effects

### Usage

```
pupil_confound_effects(x)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |

### Value

A numeric coefficient table: the smooth-term table for ‘mgcv::gam()‘ fits or the coefficient matrix for ‘stats::lm()‘ fits.

---

## pupil_event_effects *Extract event effects from pupil deconvolution*

---

### Description

Extract event effects from pupil deconvolution

### Usage

```
pupil_event_effects(x)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |

### Value

An R object containing event effects from pupil deconvolution. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

## pupil_event_regressor *Build an event-locked pupil regressor*

---

### Description

Build an event-locked pupil regressor

### Usage

```
pupil_event_regressor(time_ms, event_time_ms, tmax_ms = 930, shape = 10.1)
```

### Arguments

| | |
|---|---|
| time_ms | Sample times. |
| event_time_ms | Event onset. |
| tmax_ms, shape | Kernel parameters. |

## Value

A numeric value or vector containing an event-locked pupil regressor.

---

`pupil_frequency_features` *Extract pupil frequency-domain and activity features by group*

---

## Description

Extract pupil frequency-domain and activity features by group

## Usage

```
pupil_frequency_features(
  data,
  by = c("person_id", "trial_id"),
  time = "time_ms",
  pupil = "pupil_bc",
  sampling_rate_hz = 60,
  low_band = c(0.05, 0.5),
  high_band = c(0.5, 4)
)
```

## Arguments

| | |
|---|---|
| `data` | Sample-level data. |
| `by` | Grouping columns, e.g. person and trial/window. |
| `time`, `pupil` | Column names. |
| `sampling_rate_hz` | Either a scalar or a column name. |
| `low_band`, `high_band` | Frequency bands. |

## Value

An object of class "eye_pupil_frequency_features", stored as a named list, with components "features", "low_band", "high_band", "by", "pupil", "time", "caveat". It contains pupil frequency-domain and activity features by group and associated metadata or diagnostics needed to interpret the result.

pupil_latency_sensitivity *Pupil latency estimator sensitivity and resolvability audit*

## Description

Estimates pupil-response latency using a sustained threshold, maximum-slope tangent intersection, and piecewise breakpoint. The result reports estimator spread and a sampling/noise-aware resolvability label rather than treating a single latency estimate as algorithm- or hardware-independent.

## Usage

```
pupil_latency_sensitivity(
  time,
  pupil,
  event_time = 0,
  baseline_window = c(-0.5, 0),
  search_window = c(0, 2),
  direction = c("constriction", "dilation"),
  threshold_sigma = 3,
  sustain_ms = 40
)
```

## Arguments

| | |
|---|---|
| time | Numeric sample times in seconds. |
| pupil | Numeric pupil values. |
| event_time | Nominal event time in seconds. |
| baseline_window | Two-element window relative to ‘event_time‘. |
| search_window | Two-element search window relative to ‘event_time‘. |
| direction | ‘"constriction"‘ or ‘"dilation"‘. |
| threshold_sigma | Robust-noise multiples used by the sustained threshold. |
| sustain_ms | Required duration above threshold in milliseconds. |

## Value

A list of estimator-specific latencies, spread, signal diagnostics, resolvability, and provenance.

## pupil_preprocessing_grid

*Create a preprocessing sensitivity grid for pupil analysis*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
pupil_preprocessing_grid(baseline_windows = list(c(-200, 0), c(-500, 0)),
  latency_ms = c(100, 200, 300), basis_df = c(4L, 6L, 8L),
 baseline_methods = c("subtract", "percent"), max_interpolated_fraction = c(0.10,
  0.20))
```

### Arguments

| | |
|---|---|
| baseline_windows | List of baseline windows. |
| latency_ms | Latency shifts. |
| basis_df | Basis degrees of freedom. |
| baseline_methods | Baseline corrections. |
| max_interpolated_fraction | Interpolation thresholds. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## pupil_preprocessing_sensitivity

*Run functional pupil preprocessing sensitivity analysis*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
pupil_preprocessing_sensitivity(x, grid = pupil_preprocessing_grid(),
  base_spec = functional_pupil_irt_spec(engine = "two_stage_glm"), fit = TRUE,
 extractor = extract_functional_pupil_parameters, continue_on_error = TRUE, ...)
```

### Arguments

| | |
|---|---|
| x | Eye dataset or long pupil data. |
| grid | Sensitivity grid. |
| base_spec | Base functional pupil specification. |
| fit | Whether to fit each specification. |
| extractor | Optional result extractor. |
| continue_on_error | Record errors rather than stopping. |
| ... | Passed to model fitting. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

## pupil_response_kernel *Canonical gamma-shaped pupil response kernel*

---

### Description

Canonical gamma-shaped pupil response kernel

### Usage

```
pupil_response_kernel(
  time_since_event_ms,
  tmax_ms = 930,
  shape = 10.1,
  normalize = TRUE
)
```

### Arguments

| | |
|---|---|
| time_since_event_ms | Time relative to event onset. |
| tmax_ms | Approximate response peak time. |
| shape | Shape parameter. |
| normalize | Normalize peak to one. |

### Value

A numeric value or vector containing canonical gamma-shaped pupil response kernel.

pupil_unit_fidelity_audit *Pupil-unit semantic-fidelity audit*

## Description

Pupil-unit semantic-fidelity audit

## Usage

```
pupil_unit_fidelity_audit(
  source,
  roundtrip,
  source_pupil = "pupil_size",
  roundtrip_pupil = source_pupil,
  key = NULL,
  tolerance = 1e-06,
  correlation_floor = 0.995
)
```

## Arguments

| | |
|---|---|
| source | Original/source representation. |
| roundtrip | Round-tripped or comparison representation. |
| source_pupil | Source pupil-measure column. |
| roundtrip_pupil | Round-tripped pupil-measure column. |
| key | Column or columns used to align records. |
| tolerance | Numerical tolerance used by the comparison. |
| correlation_floor | Minimum correlation treated as compatible. |

## Value

An object of class "eye_pupil_fidelity", stored as a named list, with components "status", "matched_n", "correlation", "estimated_scale_ratio", "scaled_max_error", "tolerance". It contains pupil-unit semantic-fidelity audit and associated metadata or diagnostics needed to interpret the result.

## pupil_velocity_activity

*Derivative-based pupil activity magnitude*

### Description

Derivative-based pupil activity magnitude

### Usage

```
pupil_velocity_activity(y, time_ms)
```

### Arguments

| | |
|---|---|
| y | Pupil signal. |
| time_ms | Time in milliseconds. |

### Value

A numeric value or vector containing derivative-based pupil activity magnitude.

## quantify_process_leakage

*Quantify leakage from row-wise rather than grouped validation*

### Description

Quantify leakage from row-wise rather than grouped validation

### Usage

```
quantify_process_leakage(
  data,
  formula,
  group = c("participant_id", "item_id"),
  v = 5L,
  seed = 1L
)
```

### Arguments

| | |
|---|---|
| data | Data frame. |
| formula | Binary-outcome formula. |
| group | Grouping columns. |
| v | Number of folds. |
| seed | Random seed. |

## Value

Comparison table.

---

`query_eye_storage` *Query partitioned eye storage lazily where possible*

---

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
query_eye_storage(storage, table, filters = list(), columns = NULL, collect = TRUE)
```

## Arguments

| | |
|---|---|
| `storage` | Storage object or path. |
| `table` | Canonical table. |
| `filters` | Named list of equality filters. |
| `columns` | Optional selected columns. |
| `collect` | Whether to collect an Arrow query. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`raven_reproduction_spec` *Specify a published Raven strategy-model reproduction*

---

## Description

Specify a published Raven strategy-model reproduction

## Usage

```
raven_reproduction_spec(
  data_path,
  response,
  strategy_features,
  published_targets = NULL,
  licence_reviewed = FALSE,
  citation = "10.1016/j.intell.2023.101782"
)
```

### Arguments

| | |
|---|---|
| data_path | Path to the exact public data/materials. |
| response | Response field. |
| strategy_features | Theory-defined eye-tracking strategy indicators. |
| published_targets | Optional named target estimates. |
| licence_reviewed | Whether data and code reuse has been reviewed. |
| citation | Citation or DOI for the reproduced analysis. |

### Value

An ‘eye_raven_reproduction_spec‘.

---

## read_api_lifecycle_registry *Read API lifecycle registry from CSV*

---

### Description

Read API lifecycle registry from CSV

### Usage

```
read_api_lifecycle_registry(path)
```

### Arguments

| | |
|---|---|
| path | CSV path. |

### Value

An R object containing aPI lifecycle registry from CSV. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## read_benchmark_table *Read a benchmark table*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
read_benchmark_table(study = eyeprocess_benchmark_study(), table)
```

### Arguments

| | |
|---|---|
| study | Benchmark object or path. |
| table | Table name. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## read_decision_manifest *Read a decision manifest written by eyeprocess*

### Description

Read a decision manifest written by eyeprocess

### Usage

```
read_decision_manifest(path, format = NULL)
```

### Arguments

| | |
|---|---|
| path | Input path. |
| format | Optional format; inferred from extension when omitted. |

### Value

A logical value or vector indicating a decision manifest written by eyeprocess.

## read_eyeprocess_validation_evidence

*Read and verify a frozen evidence bundle*

### Description

Read and verify a frozen evidence bundle

### Usage

```
read_eyeprocess_validation_evidence(path, verify = TRUE)
```

### Arguments

| | |
|---|---|
| path | File path for reading or writing. |
| verify | Whether integrity verification is performed when reading. |

### Value

An R object containing and verify a frozen evidence bundle. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## read_reproducibility_fingerprint

*Read a reproducibility fingerprint*

### Description

Read a reproducibility fingerprint

### Usage

```
read_reproducibility_fingerprint(path, format = NULL)
```

### Arguments

| | |
|---|---|
| path | Input path. |
| format | Optional format. |

### Value

An R object containing a reproducibility fingerprint. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

`read_validation_job_manifest` *Read a validation manifest*

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
read_validation_job_manifest(path)
```

## Arguments

| | |
|---|---|
| `path` | Manifest directory. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

`read_validation_scenario_manifest` *Read a validation scenario manifest*

## Description

Read a validation scenario manifest

## Usage

```
read_validation_scenario_manifest(path)
```

## Arguments

| | |
|---|---|
| `path` | File path for reading or writing. |

## Value

An R object containing a validation scenario manifest. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## read_vendor_registry *Read the multi-vendor case registry*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
read_vendor_registry(corpus_path)
```

### Arguments

| | |
|---|---|
| corpus_path | Corpus directory. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## recalibrate_after_changepoint *Iteratively detect, clean, and recalibrate after process change points*

### Description

Iteratively detect, clean, and recalibrate after process change points

### Usage

```
recalibrate_after_changepoint(
  data,
  fitter,
  person = "participant_id",
  order = "item_order",
  policy = c("flag", "exclude_post_change", "add_regime"),
  ...
)
```

### Arguments

| | |
|---|---|
| data | Input data frame or compatible tabular object. |
| fitter | Function accepting a data frame and returning a calibration fit. |
| person | Person or participant identifier column. |
| order | Within-sequence ordering variable. |
| policy | 'flag', 'exclude_post_change', or 'add_regime'. |
| ... | Additional arguments passed to the selected model, engine, or method. |

## Value

An object of class "eye_changepoint_recalibration", stored as a named list, with components "changepoints", "data", "fit", "policy". It contains iteratively detect, clean, and recalibrate after process change points and associated metadata or diagnostics needed to interpret the result.

---

`recommended_validation_replications` *Approximate simulation replications needed for a target Monte Carlo error*

---

## Description

Approximate simulation replications needed for a target Monte Carlo error

## Usage

```
recommended_validation_replications(
  target_mcse = 0.01,
  metric = c("coverage", "mean"),
  anticipated_sd = 1,
  anticipated_probability = 0.95,
  minimum = 100L
)
```

## Arguments

| | |
|---|---|
| `target_mcse` | Target Monte Carlo standard error. |
| `metric` | Metric to calculate or audit. |
| `anticipated_sd` | Anticipated standard deviation. |
| `anticipated_probability` | Anticipated probability for a binary metric. |
| `minimum` | Minimum acceptable value or threshold. |

## Value

A numeric value or vector containing approximate simulation replications needed for a target Monte Carlo error.

## redact_validation_case

*Redact a validation case without inventing replacement data*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
redact_validation_case(source_path, output_path, id_columns = c("participant_id",
  "subject", "participant", "recording_id", "session_id"), remove_columns = c("name",
  "email", "address", "birthdate", "date_of_birth"), text_redactor = NULL, salt,
  copy_non_tabular = FALSE, overwrite = FALSE)
```

### Arguments

| | |
|---|---|
| source_path | Source file/directory. |
| output_path | Redacted output directory. |
| id_columns | Identifier columns to pseudonymize. |
| remove_columns | Columns to remove. |
| text_redactor | Optional function applied to character columns. |
| salt | Required project-specific salt. |
| copy_non_tabular | Copy unsupported files unchanged. |
| overwrite | Replace output. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## register_eye_api_status

*Add or update API lifecycle metadata without global mutation*

### Description

Add or update API lifecycle metadata without global mutation

## Usage

```
register_eye_api_status(
  registry = eye_api_lifecycle(),
  name,
  status,
  canonical = NA_character_,
  replacement = NA_character_,
  since = "0.9.0.9000",
  notes = NA_character_
)
```

## Arguments

| | |
|---|---|
| registry | Existing lifecycle registry. |
| name | API name. |
| status | Lifecycle status. |
| canonical | Canonical API for the same concept, if applicable. |
| replacement | Replacement for deprecated/superseded API. |
| since | Version in which status applies. |
| notes | Notes. |

## Value

An R object containing add or update API lifecycle metadata without global mutation. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

# register_irt_model *Register a multimodal IRT model*

---

## Description

Register a multimodal IRT model

## Usage

```
register_irt_model(spec, overwrite = FALSE)
```

## Arguments

| | |
|---|---|
| spec | An ‘irt_model_spec()‘. |
| overwrite | Whether to replace an existing model with the same id. |

## Value

An R object containing a multimodal IRT model. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

`register_process_measure` *Add a process measure to a registry without global mutation*

## Description

Add a process measure to a registry without global mutation

## Usage

```
register_process_measure(
  registry = process_measure_registry(),
  name,
  channel,
  unit,
  level,
  interpretation,
  guardrail,
  status = "user_defined"
)
```

## Arguments

| | |
|---|---|
| `registry` | Registry. |
| `name`, `channel`, `unit`, `level`, `interpretation`, `guardrail`, `status` | Measure metadata. |

## Value

A tabular R object containing add a process measure to a registry without global mutation; rows represent analysis units and columns contain the returned quantities.

`register_pupil_curves` *Pupil phase-amplitude registration*

## Description

Pupil phase-amplitude registration. These functions form the eyeprocess 0.6.0.9000 measurement-intelligence programme and use dependency-free reference implementations with explicit evidence limits.

## Usage

```
register_pupil_curves(x, time, pupil, anchor = c("stimulus", "response", "event"),
  method = c("elastic", "landmark"), id_col = "person_id", grid_size = 101)
decompose_pupil_phase_amplitude(x, components = 3)
fit_phase_amplitude_irt(responses, phase_scores, amplitude_scores = NULL,
  person_id = NULL, family = c("gaussian", "binomial"), ...)
audit_pupil_registration(x)
plot_pupil_registration(x, ...)
plot_warping_functions(x, ...)
plot_phase_amplitude_scores(x, ...)
plot_item_phase_delay(x, ...)
plot_registered_pupil_effects(x, ...)
```

## Arguments

| | |
|---|---|
| x | Input object or data structure appropriate for the selected analysis. |
| time | Argument controlling 'time'; see the function usage and returned audit metadata. |
| pupil | Argument controlling 'pupil'; see the function usage and returned audit metadata. |
| anchor | Argument controlling 'anchor'; see the function usage and returned audit metadata. |
| method | Argument controlling 'method'; see the function usage and returned audit metadata. |
| id_col | Argument controlling 'id_col'; see the function usage and returned audit metadata. |
| grid_size | Argument controlling 'grid_size'; see the function usage and returned audit metadata. |
| components | Argument controlling 'components'; see the function usage and returned audit metadata. |
| responses | Argument controlling 'responses'; see the function usage and returned audit metadata. |
| phase_scores | Argument controlling 'phase_scores'; see the function usage and returned audit metadata. |
| amplitude_scores | Argument controlling 'amplitude_scores'; see the function usage and returned audit metadata. |
| person_id | Argument controlling 'person_id'; see the function usage and returned audit metadata. |
| family | Argument controlling 'family'; see the function usage and returned audit metadata. |
| ... | Additional arguments passed to the underlying method or plotting function. |

## Details

The APIs return auditable S3 objects. Plot wrappers call registered base-graphics methods. Experimental or approximate engines are labelled in object status fields and should be validated before confirmatory or operational use.

## Value

An eyeprocess result object, data frame, model object, plot, or audit table as documented by the individual function.

## See Also

`plot_diagnostics()`, `plot_evidence()`, and `plot_sensitivity()`.

---

# register_validation_case

*Register an independent validation case*

---

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
register_validation_case(corpus_path, source_path, vendor, device_model,
  software_name, software_version, hardware_version = NA_character_,
  export_profile = NA_character_, sampling_rate_hz = NA_real_,
  coordinate_system = NA_character_, timebase = NA_character_,
  event_semantics = NA_character_, ocular_structure = NA_character_,
  missingness_convention = NA_character_, vendor_fixations = NA_character_,
  package_transformations = NA_character_, unsupported_fields = NA_character_,
 independent_source = TRUE, licence_reviewed = FALSE, redistribution_allowed = FALSE,
  support_level = c("declared", "fixture-tested", "empirically-validated"),
  mode = c("reference", "copy"), case_id = NULL, notes = NA_character_)
```

## Arguments

| | |
|---|---|
| `corpus_path` | Corpus directory. |
| `source_path` | Real export file or directory. |
| `vendor` | Vendor name. |
| `device_model` | Hardware model. |
| `software_name` | Export software and version. |
| `software_version` | Export software and version. |

`hardware_version`
: Optional hardware/firmware version.

`export_profile`
: Export options/profile.

`sampling_rate_hz`
: Nominal or observed rate.

`coordinate_system`
: Semantics metadata.

`timebase`
: Semantics metadata.

`event_semantics`
: Semantics metadata.

`ocular_structure`
: Monocular/binocular structure.

`missingness_convention`
: Vendor missing-value convention.

`vendor_fixations`
: Description of vendor-derived fixation fields.

`package_transformations`
: Declared package transformations.

`unsupported_fields`
: Known unsupported fields.

`independent_source`
: Whether independently obtained.

`licence_reviewed`
: Whether licensing review is complete.

`redistribution_allowed`
: Whether redacted material may be redistributed.

`support_level`
: Declared support level.

`mode`
: Reference source in place or copy it into the private corpus.

`case_id`
: Optional explicit identifier.

`notes`
: Notes.

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`register_vendor_semantics` *Register vendor-field semantics*

---

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
register_vendor_semantics(corpus_path, vendor, native_field, native_meaning,
 canonical_table, canonical_field, unit = NA_character_, transformation = "identity",
  loss_risk = c("none", "low", "moderate", "high", "unsupported"),
  evidence_case_id = NA_character_)
```

### Arguments

| | |
|---|---|
| corpus_path | Corpus directory. |
| vendor | Vendor. |
| native_field | Native field and meaning. |
| native_meaning | Native field and meaning. |
| canonical_table | Canonical destination. |
| canonical_field | Canonical destination. |
| unit | Unit. |
| transformation | Transformation description. |
| loss_risk | None, low, moderate, high, or unsupported. |
| evidence_case_id | Supporting case. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

## reporting_guideline_audit

*Audit reporting-guideline coverage*

---

### Description

Audit reporting-guideline coverage

### Usage

```
reporting_guideline_audit(x, model = NULL, sensitivity = NULL)
```

### Arguments

| | |
|---|---|
| x | An ‘eye_dataset‘. |
| model | Optional ‘eyeprocess_model‘. |
| sensitivity | Optional ‘eye_multiverse‘ sensitivity result. |

### Value

An ‘eye_reporting_audit‘ data frame.

`representative_scanpath` *Representative scanpaths and scanpath distributions*

## Description

Representative scanpaths and scanpath distributions. These functions form the eyeprocess 0.6.0.9000 measurement-intelligence programme and use dependency-free reference implementations with explicit evidence limits.

## Usage

```
representative_scanpath(x, method = c("medoid", "barycenter", "consensus"),
  id_col = "person_id", aoi_col = "aoi", x_col = "x", y_col = "y",
  distance = c("multimatch", "edit", "transport"))
scanpath_dispersion(x)
compare_scanpath_distributions(x, group, distance = c("multimatch", "edit",
  "transport"), permutations = 499)
bootstrap_representative_scanpath(x, draws = 250, seed = 20260807)
plot_scanpath_atlas(x, ...)
plot_representative_scanpath(x, ...)
plot_scanpath_dispersion(x, ...)
plot_group_scanpath_transport(x, ...)
plot_scanpath_similarity_matrix(x, ...)
```

## Arguments

| | |
|---|---|
| `x` | Input object or data structure appropriate for the selected analysis. |
| `method` | Argument controlling ‘method‘; see the function usage and returned audit metadata. |
| `id_col` | Argument controlling ‘id_col‘; see the function usage and returned audit metadata. |
| `aoi_col` | Argument controlling ‘aoi_col‘; see the function usage and returned audit metadata. |
| `x_col` | Argument controlling ‘x_col‘; see the function usage and returned audit metadata. |
| `y_col` | Argument controlling ‘y_col‘; see the function usage and returned audit metadata. |
| `distance` | Argument controlling ‘distance‘; see the function usage and returned audit metadata. |
| `group` | Argument controlling ‘group‘; see the function usage and returned audit metadata. |
| `permutations` | Argument controlling ‘permutations‘; see the function usage and returned audit metadata. |

| | |
|---|---|
| draws | Argument controlling 'draws'; see the function usage and returned audit metadata. |
| seed | Argument controlling 'seed'; see the function usage and returned audit metadata. |
| ... | Additional arguments passed to the underlying method or plotting function. |

## Details

The APIs return auditable S3 objects. Plot wrappers call registered base-graphics methods. Experimental or approximate engines are labelled in object status fields and should be validated before confirmatory or operational use.

## Value

An eyeprocess result object, data frame, model object, plot, or audit table as documented by the individual function.

## See Also

`plot_diagnostics()`, `plot_evidence()`, and `plot_sensitivity()`.

---

# resume_eye_pipeline *Resume a governed pipeline from a prior run*

---

## Description

Resume a governed pipeline from a prior run

## Usage

```
resume_eye_pipeline(x, previous, context = list(), stop_on_error = TRUE)
```

## Arguments

| | |
|---|---|
| x | Pipeline. |
| previous | Prior pipeline run. |
| context | Context used for new steps. |
| stop_on_error | Stop on non-optional error. |

## Value

An object of class "eye_pipeline_run", stored as a named list, with components "pipeline", "pipeline_hash", "outputs", "records", "errors", "warnings", "context_hash", "completed", "created_at", "status". It contains resume a governed pipeline from a prior run and associated metadata or diagnostics needed to interpret the result.

## resume_validation_jobs *Resume incomplete or failed validation jobs*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
resume_validation_jobs(plan, output_dir, retry = c("missing", "failed",
  "nonconverged", "locked", "corrupt"), ...)
```

### Arguments

| | |
|---|---|
| plan | Validation plan or manifest path. |
| output_dir | Validation output directory. |
| retry | Which statuses to re-run. |
| ... | Passed to 'run_validation_jobs()'. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## roundtrip_eye_bids *Execute and audit an Eye-Tracking-BIDS round trip*

### Description

This is a callback harness so it remains stable even if the package's BIDS writer/reader signatures evolve. 'exporter' receives the source object plus 'export_args'; 'importer' receives the exporter result plus 'import_args'.

### Usage

```
roundtrip_eye_bids(
  source,
  exporter,
  importer,
  export_args = list(),
  import_args = list(),
  extract_samples = .ep07_roundtrip_extract_samples,
  audit_args = list()
)
```

### Arguments

| | |
|---|---|
| source | Source eyeprocess object/table. |
| exporter | Function that writes/exports BIDS and returns a locator or object consumable by 'importer'. |
| importer | Function that reconstructs an eyeprocess object/table. |
| export_args, import_args | Named argument lists. |
| extract_samples | Function extracting the canonical sample table from source and reconstructed objects. |
| audit_args | Arguments forwarded to 'semantic_roundtrip_audit()'. |

### Value

An object of class "eye_bids_roundtrip", stored as a named list, with components "exported", "reconstructed", "audit", "status". It contains execute and audit an Eye-Tracking-BIDS round trip and associated metadata or diagnostics needed to interpret the result.

---

## run_benchmark_reproduction

*Derive reproducible benchmark summaries*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
run_benchmark_reproduction(study = eyeprocess_benchmark_study())
```

### Arguments

| | |
|---|---|
| study | Benchmark object or path. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## run_eyeprocess_equateirt

*Run a named equateIRT linking/equating function without fallback substitution*

### Description

The caller supplies an exported equateIRT function name and its arguments. eyeprocess does not replace the requested equating estimator when the engine is unavailable.

### Usage

```
run_eyeprocess_equateirt(function_name, ..., engine = "equateIRT")
```

### Arguments

| | |
|---|---|
| function_name | Name of the external equateIRT function to call. |
| ... | Additional arguments passed to the selected method or external engine. |
| engine | Requested estimation or analysis engine. |

### Value

An object of class "eye_external_irt_fit", stored as a named list, with components "status", "engine", "function_name", "fit", "call". It contains a named equateIRT linking/equating function without fallback substitution and associated metadata or diagnostics needed to interpret the result.

## run_eyeprocess_irt_ability_sbc

*Run simulation-based calibration for known-item IRT ability scoring*

### Description

Simulates abilities from the declared normal prior, responses from the known item-response model, and posterior draws from the same grid-based scoring algorithm used by ‘eyeprocess_irt_eap_score()‘. This validates computational calibration of the scoring workflow under the declared generative model; it does not establish empirical adequacy or construct validity.

### Usage

```
run_eyeprocess_irt_ability_sbc(
  items,
  replications = 200L,
  posterior_draws = 99L,
  theta_grid = seq(-5, 5, length.out = 401),
  prior_mean = 0,
```

```
  prior_sd = 1,
  interval = 0.95,
  seed = 20260811L,
  D = 1
)
```

## Arguments

| | |
|---|---|
| items | Item-parameter data frame or item collection. |
| replications | Number of simulation or validation replications. |
| posterior_draws | Number of posterior draws generated per SBC replication. |
| theta_grid | Grid of latent-trait values used for numerical scoring or integration. |
| prior_mean | Mean of the normal latent-trait prior. |
| prior_sd | Standard deviation of the normal latent-trait prior. |
| interval | Central posterior interval probability used for coverage assessment. |
| seed | Random-number seed for reproducible execution. |
| D | Logistic scaling constant. |

## Value

An object of class "eye_irt_sbc_evidence", stored as a named list, with components "diagnostics", "ecdf_deviation", "n", "n_draws". It contains simulation-based calibration for known-item IRT ability scoring and associated metadata or diagnostics needed to interpret the result.

---

# run_eyeprocess_irt_recovery

*Run IRT parameter recovery with the exact mirt engine*

---

## Description

When mirt is unavailable the function returns a gated result rather than a substitute estimator.

## Usage

```
run_eyeprocess_irt_recovery(design, engine = "mirt", verbose = TRUE)
```

## Arguments

| | |
|---|---|
| design | Validation or simulation design object. |
| engine | Requested estimation or analysis engine. |
| verbose | Value supplied for the verbose argument. |

## Value

An object of class "eye_irt_recovery_result", stored as a named list, with components "design", "estimates", "failures", "engine". It contains iRT parameter recovery with the exact mirt engine and associated metadata or diagnostics needed to interpret the result.

---

`run_eyeprocess_mirtcat` *Run a mirtCAT adaptive-testing workflow without fallback substitution*

---

## Description

Run a mirtCAT adaptive-testing workflow without fallback substitution

## Usage

```
run_eyeprocess_mirtcat(..., engine = "mirtCAT")
```

## Arguments

| | |
|---|---|
| `...` | Additional arguments passed to the selected method or external engine. |
| `engine` | Requested estimation or analysis engine. |

## Value

An object of class "eye_external_irt_fit", stored as a named list, with components "status", "engine", "fit", "call". It contains a mirtCAT adaptive-testing workflow without fallback substitution and associated metadata or diagnostics needed to interpret the result.

---

`run_eyeprocess_stress_evidence` *Execute a declared measurement-stress evidence plan*

---

## Description

'corruptors' is a named list of functions accepting '(data, severity, seed)'. 'metric_fun' must return a named finite/numeric vector (NA is allowed for metrics that are undefined in a scenario). The executor records software behavior under declared corruptions and does not define universal data-quality thresholds.

## Usage

```
run_eyeprocess_stress_evidence(data, plan, corruptors, metric_fun)
```

### Arguments

| | |
|---|---|
| data | Input data frame, matrix, or compatible analysis object. |
| plan | Validation or stress-evidence plan object. |
| corruptors | Named list of corruption functions used by the stress programme. |
| metric_fun | Function used to compute the stress-programme evaluation metric. |

### Value

An object of class "eye_stress_evidence_result", stored as a named list, with components "plan", "scenarios", "baseline", "results", "failures", "guardrail". It contains execute a declared measurement-stress evidence plan and associated metadata or diagnostics needed to interpret the result.

---

## run_eyeprocess_validation_program

*Run the complete validation-release programme*

---

### Description

Run the complete validation-release programme

### Usage

```
run_eyeprocess_validation_program(
  corpus,
  output_dir,
  model_jobs = list(),
  sbc_jobs = list(),
  engine_jobs = list(),
  reproduction_jobs = list(),
  grouped_jobs = list(),
  leakage_jobs = list(),
  multiverse_jobs = list(),
  benchmark_jobs = list(),
  reporting_dataset = NULL,
  public_benchmark_dataset = NULL,
  public_benchmark_include_samples = FALSE,
  advanced_evidence = list(),
  evidence_spec = advanced_model_evidence_spec(),
  overwrite = FALSE
)
```

## Arguments

| | |
|---|---|
| corpus | Validation corpus or manifest. |
| output_dir | Output directory. |
| model_jobs | Named list of model-validation job specifications. Each job supplies ‘simulator‘, ‘fitter‘, ‘extractor‘, ‘truth_extractor‘, and optional ‘grid‘ and ‘spec‘. |
| sbc_jobs | Named list of simulation-based-calibration job specifications. |
| engine_jobs | Named list of equivalent-engine comparison jobs. |
| reproduction_jobs | Named list of licensed empirical-reproduction jobs. |
| grouped_jobs | Named list of grouped-validation jobs. Set ‘crossed = TRUE‘ in a job to call ‘crossed_grouped_cv()‘. |
| leakage_jobs | Named list of leakage-quantification jobs. |
| multiverse_jobs | Named list of preprocessing-multiverse jobs. |
| benchmark_jobs | Named list of zero-argument functions or benchmark argument lists. |
| reporting_dataset | Optional ‘eye_dataset‘ for reporting-guideline coverage. |
| public_benchmark_dataset | Optional ‘eye_dataset‘ from which to write a de-identified public benchmark bundle. |
| public_benchmark_include_samples | Whether the public benchmark retains sample-level tables. |
| advanced_evidence | Optional named evidence records keyed by model function. |
| evidence_spec | Advanced-model promotion-gate specification. |
| overwrite | Whether to replace the output directory. |

## Value

An ‘eye_validation_program‘ object.

---

# run_eye_benchmark *Run a computational scaling benchmark*

---

## Description

Run a computational scaling benchmark

### Usage

```
run_eye_benchmark(
  design = eye_benchmark_design(),
  generator = .ep09_default_benchmark_generator,
  operation = .ep09_default_benchmark_operation,
  gc_before = TRUE,
  progress = interactive()
)
```

### Arguments

| | |
|---|---|
| design | Benchmark design. |
| generator | Function ‘(n, row)‘ returning benchmark input. |
| operation | Function ‘(data, row)‘ representing the operation under test. |
| gc_before | Run garbage collection before timing. |
| progress | Print progress. |

### Value

An object of class "eye_benchmark_result", stored as a named list, with components "design", "results", "created_at", "status", "caveat". It contains a computational scaling benchmark and associated metadata or diagnostics needed to interpret the result.

---

## run_eye_pipeline *Run a governed eyeprocess pipeline*

---

### Description

Run a governed eyeprocess pipeline

### Usage

```
run_eye_pipeline(x, context = list(), stop_on_error = TRUE, previous = NULL)
```

### Arguments

| | |
|---|---|
| x | Pipeline. |
| context | Initial named context available as ‘.context‘. |
| stop_on_error | Stop on a non-optional step error. |
| previous | Optional prior ‘eye_pipeline_run‘ used for resumption. |

### Value

An object of class "eye_pipeline_run", stored as a named list, with components "pipeline", "pipeline_hash", "outputs", "records", "errors", "warnings", "context_hash", "completed", "created_at", "status". It contains a governed eyeprocess pipeline and associated metadata or diagnostics needed to interpret the result.

run_gazepoint_workflow *Run the complete Gazepoint downstream workflow*

## Description

Imports a real Gazepoint folder, constructs a canonical ‘eye_dataset‘, runs QC, reconstructs media trials, processes pupil and biometric streams, derives gaze/AOI/pupil/biometric features, creates analysis and IRT tables, generates plots, exports every stage, and writes a reproducible report.

## Usage

```
run_gazepoint_workflow(
  path,
  output_dir = file.path(getwd(), "eyeprocess-gazepoint-workflow"),
  responses = NULL,
  score_key = NULL,
  item_map = NULL,
  spec = gazepoint_workflow_spec(),
  overwrite = FALSE,
  quiet = FALSE
)
```

## Arguments

| | |
|---|---|
| path | Gazepoint export folder. |
| output_dir | Destination directory. |
| responses | Optional response data frame or CSV path. |
| score_key | Optional named vector of correct responses by item id. |
| item_map | Optional stimulus-to-item map. |
| spec | Workflow specification from ‘gazepoint_workflow_spec()‘. |
| overwrite | Replace an existing output directory. |
| quiet | Suppress progress messages. |

## Value

An ‘eye_gazepoint_workflow‘ object.

`run_model_validation` *Run parameter-recovery, coverage, and misspecification validation*

## Description

The fitter may return a model or throw an error. The extractor must return a data frame with ‘parameter‘, ‘estimate‘, and optionally ‘std_error‘, ‘lower‘, and ‘upper‘. The truth extractor must return a named numeric vector.

## Usage

```
run_model_validation(
  simulator,
  fitter,
  extractor,
  truth_extractor,
  grid = NULL,
  spec = model_validation_spec(),
  seed = 1L,
  continue_on_error = TRUE
)
```

## Arguments

| | |
|---|---|
| `simulator` | Simulation function. |
| `fitter` | Estimation function receiving the simulation result. |
| `extractor` | Parameter extraction function. |
| `truth_extractor` | Truth extraction function. |
| `grid` | Scenario grid. |
| `spec` | Validation specification. |
| `seed` | Random seed. |
| `continue_on_error` | Record rather than stop on estimation errors. |

## Value

An ‘eye_model_validation‘ object.

## run_posterior_sbc *Run posterior simulation-based calibration from an explicit contract*

### Description

Run posterior simulation-based calibration from an explicit contract

### Usage

```
run_posterior_sbc(
  observed_data,
  contract,
  replications = 100L,
  seed = 20260808L
)
```

### Arguments

| | |
|---|---|
| observed_data | Observed dataset used by the calibration procedure. |
| contract | Posterior-SBC or validation contract. |
| replications | Number of simulation or validation replications. |
| seed | Random-number seed. |

### Value

An object of class "eye_posterior_sbc", "eye_irt_sbc", stored as a named list, with components "ranks", "failures", "replications", "seed", "method", "requirement". It contains posterior simulation-based calibration from an explicit contract and associated metadata or diagnostics needed to interpret the result.

## run_process_negative_controls *Run repeated process negative controls*

### Description

Run repeated process negative controls

## Usage

```
run_process_negative_controls(
  data,
  outcome,
  analysis_fun,
  controls = c("permutation", "shift"),
  replications = 100L,
  seed = 1L,
  extract_fun = .ep09_default_control_extract,
  shift_lags = c(-3L, -2L, -1L, 1L, 2L, 3L),
  within = NULL
)
```

## Arguments

| | |
|---|---|
| data | Data frame. |
| outcome | Outcome column. |
| analysis_fun | Function applied to each negative-control dataset. |
| controls | Character vector among ‘permutation‘ and ‘shift‘. |
| replications | Number of controls per type. |
| seed | Seed. |
| extract_fun | Function converting analysis result to a data.frame. |
| shift_lags | Lags sampled for shift controls. |
| within | Optional permutation groups. |

## Value

A named list with components "results", "outcome", "controls", "replications", "seed", "interpretation", containing repeated process negative controls and associated metadata or diagnostics.

---

# run_process_sensitivity

*Run an explicit process-analysis multiverse*

---

## Description

Run an explicit process-analysis multiverse

## Usage

```
run_process_sensitivity(
  data,
  grid,
  analysis_fun,
  extract_fun = .ep09_default_sensitivity_extract,
  progress = interactive()
)
```

### Arguments

| | |
|---|---|
| data | Analysis data. |
| grid | Sensitivity grid. |
| analysis_fun | Function ‘(data, specification)‘. |
| extract_fun | Function ‘(fit, specification)‘ returning one or more rows. |
| progress | Print progress. |

### Value

An object of class "eye_process_sensitivity", stored as a named list, with components "grid", "results", "failures", "warnings", "grid_hash", "created_at", "status", "caveat". It contains an explicit process-analysis multiverse and associated metadata or diagnostics needed to interpret the result.

---

## run_process_validation

*Run an empirical process-validation programme*

---

### Description

Run an empirical process-validation programme

### Usage

```
run_process_validation(
  design,
  simulate_fun = simulate_process_validation_data,
  fit_fun = .ep09_default_validation_fit,
  extract_fun = .ep09_default_validation_extract,
  max_conditions = Inf,
  progress = interactive()
)
```

### Arguments

| | |
|---|---|
| design | Validation design or expanded condition table. |
| simulate_fun | Function ‘(condition, replication, seed)‘ returning a simulation object. |
| fit_fun | Function ‘(simulated, condition)‘ returning a fitted object. |
| extract_fun | Function ‘(fit, simulated, condition)‘ returning one or more rows with estimates. |
| max_conditions | Optional cap on conditions actually run. |
| progress | Print compact progress messages. |

### Value

An ‘eye_process_validation_result‘ object.

`run_raven_reproduction` *Execute a licensed published-model reproduction*

## Description

Execute a licensed published-model reproduction

## Usage

```
run_raven_reproduction(spec, importer, fitter, extractor, tolerance = 0.05)
```

## Arguments

| | |
|---|---|
| `spec` | Reproduction specification. |
| `importer` | Function receiving ‘spec$data_path‘. |
| `fitter` | Function receiving imported data and ‘spec‘. |
| `extractor` | Function returning named estimates or a parameter/estimate data frame. |
| `tolerance` | Absolute target tolerance. |

## Value

An ‘eye_empirical_reproduction‘ object.

`run_sbc` *Run generic simulation-based calibration*

## Description

Run generic simulation-based calibration

## Usage

```
run_sbc(
  simulator,
  fitter,
  posterior_draws,
  replications = 100L,
  seed = 20260808L
)
```

### Arguments

| | |
|---|---|
| simulator | Function ‘simulator(replicate)‘ returning ‘list(data, truth)‘; ‘truth‘ must be a named numeric vector. |
| fitter | Function ‘fitter(data)‘ returning a fitted object. |
| posterior_draws | Function returning a draws matrix/data frame whose columns match names in ‘truth‘. |
| replications | Number of SBC replications. |
| seed | RNG seed. |

### Value

An object of class "eye_irt_sbc", stored as a named list, with components "ranks", "failures", "replications", "seed", "method". It contains generic simulation-based calibration and associated metadata or diagnostics needed to interpret the result.

---

## run_validation_jobs *Run validation jobs with checkpointing and deterministic seeds*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
run_validation_jobs(plan, simulator, fitter, extractor, truth_extractor, output_dir,
  workers = 1L, backend = c("auto", "sequential", "future"), isolation = c("auto",
   "in_process", "callr"), timeout_seconds = Inf, memory_limit_mb = Inf,
   stale_lock_seconds = 3600, overwrite = FALSE, fail_fast = FALSE,
  progress = interactive(), job_ids = NULL, chunks = NULL, simulation_args = list(),
   fit_args = list(), diagnostics_extractor = NULL, draws_extractor = NULL,
   predictions_extractor = NULL, confidence = 0.95, run_metadata = list())
```

### Arguments

| | |
|---|---|
| plan | Validation plan or manifest directory. |
| simulator | Simulation function. |
| fitter | Fitting function receiving the simulated object first. |
| extractor | Function extracting parameter estimates. |
| truth_extractor | Function extracting named true parameter values. |
| output_dir | Validation output directory. |
| workers | Number of workers. |

| | |
|---|---|
| backend | Sequential or optional ‘future‘ backend. |
| isolation | In-process execution or optional ‘callr‘ isolation. |
| timeout_seconds | Per-job timeout. Enforced only with ‘callr‘ isolation. |
| memory_limit_mb | Best-effort per-job memory limit. |
| stale_lock_seconds | Age after which an abandoned job lock may be reclaimed. |
| overwrite | Re-run completed checkpoints. |
| fail_fast | Stop after the first failed/nonconverged job. |
| progress | Display progress in sequential mode. |
| job_ids | Optional subset of job identifiers. |
| chunks | Optional subset of chunk identifiers. |
| simulation_args | Additional simulator arguments. |
| fit_args | Additional fitter arguments. |
| diagnostics_extractor | Optional diagnostics extractor. |
| draws_extractor | Optional posterior-draw extractor. |
| predictions_extractor | Optional prediction extractor. |
| confidence | Confidence level used when standard errors are supplied. |
| run_metadata | Named metadata included in the runner fingerprint; use it to record code, prior, or engine variants captured outside function bodies. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`sbc_ecdf_deviation` *ECDF deviation summary for SBC ranks*

---

## Description

ECDF deviation summary for SBC ranks

## Usage

```
sbc_ecdf_deviation(x, n_draws = NULL)
```

### Arguments

| | |
|---|---|
| x | SBC diagnostics or ranks. |
| n_draws | Required if x is ranks. |

### Value

An R object containing eCDF deviation summary for SBC ranks. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

## sbc_rank_diagnostics *Build SBC rank diagnostics*

---

### Description

Build SBC rank diagnostics

### Usage

```
sbc_rank_diagnostics(ranks, n_draws, bins = NULL)
```

### Arguments

| | |
|---|---|
| ranks | Integer rank statistics from 0 through ‘n_draws‘. |
| n_draws | Number of posterior draws used per rank. |
| bins | Histogram bins; defaults to a bounded square-root rule. |

### Value

‘eye_sbc_diagnostics‘ object.

---

## sbc_summary *Summarize simulation-based calibration*

---

### Description

Summarize simulation-based calibration

### Usage

```
sbc_summary(x)
```

### Arguments

| | |
|---|---|
| x | An ‘eye_sbc‘ object. |

## Value

Parameter-level rank and standardized-bias summaries.

---

`score_partial_response_pattern` *Score a partial response pattern from a calibrated mirt model*

---

## Description

Score a partial response pattern from a calibrated mirt model

## Usage

```
score_partial_response_pattern(
  model,
  response_pattern,
  method = c("MAP", "EAP"),
  ...
)
```

## Arguments

| | |
|---|---|
| `model` | Calibrated 'mirt' model. |
| `response_pattern` | Full-length response vector with future/unobserved items as NA. |
| `method` | mirt scoring method, typically MAP or EAP. |
| `...` | Passed to 'mirt::fscores()'. |

## Value

A data frame containing a partial response pattern from a calibrated mirt model. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

---

`score_response_stream` *Score a response stream cumulatively*

---

## Description

Score a response stream cumulatively

### Usage

```
score_response_stream(
  model,
  response_pattern,
  observed_order = NULL,
  method = c("MAP", "EAP"),
  ...
)
```

### Arguments

| | |
|---|---|
| model | Calibrated ‘mirt‘ model. |
| response_pattern | Complete or partial response vector in item order. |
| observed_order | Order in which observed items arrive. Defaults to sequence. |
| method | MAP or EAP. |
| ... | Passed to ‘mirt::fscores()‘. |

### Value

An ‘eye_streaming_score‘ object.

---

## select_next_item_process *Select the next item using response/process utility*

---

### Description

Select the next item using response/process utility

### Usage

```
select_next_item_process(
  theta,
  item_bank,
  used = character(),
  weights = c(response = 1, rt = 0, process = 0),
  burden_weight = 0
)
```

### Arguments

| | |
|---|---|
| theta | Latent-trait values. |
| item_bank | Value supplied to ‘item_bank‘; see Details for its model-specific role. |
| used | Items already used or unavailable for selection. |
| weights | Weights used to combine information components. |
| burden_weight | Penalty applied to expected burden. |

## Value

A named list with components "item_id", "utility", "row", "all_utilities", containing the next item using response/process utility and associated metadata or diagnostics.

---

`semantic_fidelity_spec` *Semantic fidelity specification*

---

## Description

Semantic fidelity specification

## Usage

```
semantic_fidelity_spec(
  timestamp_tolerance = 1e-06,
  coordinate_tolerance = 1e-06,
  pupil_tolerance = 1e-06,
  missingness_tolerance = 1e-06,
  correlation_floor = 0.999,
  allow_row_reorder = TRUE
)
```

## Arguments

`timestamp_tolerance`
: Absolute tolerance after time-unit normalization.

`coordinate_tolerance`
: Absolute tolerance after coordinate normalization.

`pupil_tolerance`
: Absolute tolerance after pupil-unit normalization.

`missingness_tolerance`
: Maximum tolerated absolute change in missingness.

`correlation_floor`
: Correlation floor used when deciding whether a numeric transformation remains semantically equivalent.

`allow_row_reorder`
: Whether row reordering is permitted when a key is supplied.

## Value

An object of class ‘eye_semantic_fidelity_spec‘.

## semantic_loss_map *Convert a semantic round-trip audit into a loss map*

### Description

Convert a semantic round-trip audit into a loss map

### Usage

```
semantic_loss_map(x)
```

### Arguments

| | |
|---|---|
| x | Object to print, plot, summarize, or audit. |

### Value

A tabular R object containing a semantic round-trip audit into a loss map; rows represent analysis units and columns contain the returned quantities.

## semantic_roundtrip_audit *Audit a complete semantic round trip*

### Description

Audit a complete semantic round trip

### Usage

```
semantic_roundtrip_audit(
  source,
  roundtrip,
  key = NULL,
  fields = NULL,
  timestamp = list(),
  coordinates = list(),
  pupil = NULL,
  eye = NULL,
  source_events = NULL,
  roundtrip_events = NULL,
  event_args = list()
)
```

### Arguments

| | |
|---|---|
| source | Original canonical samples. |
| roundtrip | Canonical samples reconstructed after interchange. |
| key | Optional row identity key. |
| fields | Fields for field-level comparison. |
| timestamp | Optional list of arguments forwarded to ‘timestamp_fidelity_audit()‘. |
| coordinates | Optional list of arguments forwarded to ‘coordinate_fidelity_audit()‘. |
| pupil | Optional list of arguments forwarded to ‘pupil_unit_fidelity_audit()‘. |
| eye | Optional list of arguments forwarded to ‘eye_stream_fidelity_audit()‘. |
| source_events, roundtrip_events | Optional event tables. |
| event_args | Optional event-audit arguments. |

### Value

An ‘eye_semantic_roundtrip‘ object.

---

## sensitivity_branch_fingerprint *Stable fingerprint of a sensitivity branch*

---

### Description

Stable fingerprint of a sensitivity branch

### Usage

```
sensitivity_branch_fingerprint(specification)
```

### Arguments

| | |
|---|---|
| specification | One-row specification table or named list. |

### Value

An R object containing stable fingerprint of a sensitivity branch. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## sensitivity_decision_leverage

*Decision leverage of each analytical choice*

### Description

Leverage is descriptive variation in mean effect across option levels; it is not causal attribution of researcher decisions.

### Usage

```
sensitivity_decision_leverage(x, effect = "effect")
```

### Arguments

| | |
|---|---|
| x | Sensitivity result. |
| effect | Effect column. |

### Value

A tabular R object containing decision leverage of each analytical choice; rows represent analysis units and columns contain the returned quantities.

## sensitivity_fragility_index

*Fragility index across analysis specifications*

### Description

Fragility index across analysis specifications

### Usage

```
sensitivity_fragility_index(x, effect = "effect", threshold = 0)
```

### Arguments

| | |
|---|---|
| x | Sensitivity result. |
| effect | Effect column. |
| threshold | Decision threshold. |

### Value

A numeric value or vector containing fragility index across analysis specifications.

## sensitivity_multiverse_manifest

*Machine-readable multiverse manifest*

### Description

Machine-readable multiverse manifest

### Usage

```
sensitivity_multiverse_manifest(x)
```

### Arguments

| | |
|---|---|
| x | Sensitivity result. |

### Value

A data frame containing machine-readable multiverse manifest. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## sensitivity_rank_stability

*Rank stability across specifications*

### Description

Rank stability across specifications

### Usage

```
sensitivity_rank_stability(x, id = NULL, rank = NULL, specification = NULL)
```

### Arguments

| | |
|---|---|
| x | Data frame or list of ranking vectors. |
| id | Optional item identifier when x is a long data frame. |
| rank | Optional rank/value column when x is a long data frame. |
| specification | Optional specification column. |

### Value

An R object containing rank stability across specifications. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## sensitivity_significance_stability

*Significance-decision stability across specifications*

### Description

Significance-decision stability across specifications

### Usage

```
sensitivity_significance_stability(x, p_value = "p_value", alpha = 0.05)
```

### Arguments

| | |
|---|---|
| x | Sensitivity result. |
| p_value | P-value column. |
| alpha | Decision threshold. |

### Value

A numeric value or vector containing significance-decision stability across specifications.

## sensitivity_sign_stability

*Effect-sign stability across specifications*

### Description

Effect-sign stability across specifications

### Usage

```
sensitivity_sign_stability(x, effect = "effect")
```

### Arguments

| | |
|---|---|
| x | Sensitivity result. |
| effect | Effect column. |

### Value

A numeric value or vector containing effect-sign stability across specifications.

## sensitivity_threshold_stability

*Substantive-threshold stability across specifications*

### Description

Substantive-threshold stability across specifications

### Usage

```
sensitivity_threshold_stability(
  x,
  effect = "effect",
  threshold = 0,
  direction = c("above", "below", "absolute")
)
```

### Arguments

| | |
|---|---|
| x | Sensitivity result. |
| effect | Effect column. |
| threshold | Threshold. |
| direction | 'above', 'below', or 'absolute'. |

### Value

A numeric value or vector containing substantive-threshold stability across specifications.

## sequence_interoperability

*Convert scanpaths to process/sequence package contracts*

### Description

Convert scanpaths to process/sequence package contracts

### Usage

```
as_procdata_sequence(
  x,
  source = c("visits", "fixations", "samples"),
  collapse_consecutive = TRUE
)
```

```
as_traminer_sequence(
  x,
  source = c("visits", "fixations", "samples"),
  collapse_consecutive = TRUE,
  create_object = FALSE
)

as_seqhmm_data(
  x,
  source = c("visits", "fixations", "samples"),
  collapse_consecutive = TRUE
)
```

## Arguments

| | |
|---|---|
| x | An ‘eye_dataset‘. |
| source | AOI sequence source. |
| collapse_consecutive | Whether to collapse repeated adjacent states. |
| create_object | For ‘as_traminer_sequence()‘, whether to return a native ‘TraMineR‘ sequence object instead of the package-neutral wide table. |

## Value

A package-compatible representation.

---

`session_facet_effects` *Extract session facet effects*

---

## Description

Extract session facet effects

## Usage

```
session_facet_effects(object, channel = c("response", "process"))
```

## Arguments

| | |
|---|---|
| object | A fitted eyeprocess model or audit object. |
| channel | Measurement channel to inspect. |

## Value

An object of class "eye_process_facet_effects", stored as a named list, with components "facet", "column", "channel", "random_effects", "variance_component". It contains session facet effects and associated metadata or diagnostics needed to interpret the result.

`simulate_advanced_process_data` *Simulate advanced response-process data*

## Description

Generates accuracy, response time, process features, theory-defined strategy, dynamic AOI states, pupil trajectories, measurement error, missing process data, DIF, and local dependence for validation studies.

## Usage

```
simulate_advanced_process_data(
  n_person = 100L,
  n_item = 20L,
  n_time = 30L,
  n_states = 3L,
  ability_speed_correlation = -0.3,
  gaze_effect = 0.35,
  feature_reliability = 0.7,
  missing_process = 0,
  state_misclassification = 0,
  pupil_ar1 = 0.6,
  luminance_effect = 0,
  dif_effect = 0,
  local_dependence = 0,
  seed = 1L
)
```

## Arguments

| | |
|---|---|
| `n_person` | Number of persons. |
| `n_item` | Number of items. |
| `n_time` | Pupil time bins. |
| `n_states` | Number of AOI states. |
| `ability_speed_correlation` | Correlation between ability and speed. |
| `gaze_effect` | Process-feature coefficient in the response model. |
| `feature_reliability` | Reliability of observed gaze features. |
| `missing_process` | Fraction of process observations set missing. |
| `state_misclassification` | Probability of AOI-state misclassification. |
| `pupil_ar1` | AR(1) coefficient for pupil noise. |

`luminance_effect`
: Effect of simulated luminance on pupil size.

`dif_effect`
: Logit-scale DIF effect for the focal group on flagged items.

`local_dependence`
: Shared testlet-effect standard deviation.

`seed`
: Random seed.

## Value

A list with trial data, state data, pupil data, and truth.

---

`simulate_dynamic_irtree_data` *Simulate observed dynamic-state transitions*

---

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
simulate_dynamic_irtree_data(n_person = 100L, n_item = 20L,
  transitions_per_trial = 8L, states = c("prompt", "evidence", "options"),
  beta_response = 0.5, person_sd = 0.4, item_sd = 0.3, irregular_time = TRUE,
 state_misclassification = 0, missing_state = 0, structural_zeros = NULL, seed = 1L)
```

## Arguments

`n_person`
: Number of persons and items.

`n_item`
: Number of persons and items.

`transitions_per_trial`
: Number of transitions per person-item trial.

`states`
: State labels.

`beta_response`
: Response effect on transitions.

`person_sd`
: Person/item heterogeneity.

`item_sd`
: Person/item heterogeneity.

`irregular_time`
: Whether to generate irregular time gaps.

`state_misclassification`
: Destination-state error probability.

`missing_state`
: Missing destination-state probability.

`structural_zeros`
: Optional forbidden transitions.

`seed`
: Random seed.

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`simulate_eyeprocess_catr`

*Run a catR adaptive-testing simulation without fallback substitution*

---

## Description

Run a catR adaptive-testing simulation without fallback substitution

## Usage

```
simulate_eyeprocess_catr(itemBank, trueTheta = 0, ..., engine = "catR")
```

## Arguments

| | |
|---|---|
| `itemBank` | Item bank supplied to catR. |
| `trueTheta` | Known true latent-trait value used for CAT simulation. |
| `...` | Additional arguments passed to the selected method or external engine. |
| `engine` | Requested estimation or analysis engine. |

## Value

An object of class "eye_external_irt_fit", stored as a named list, with components "status", "engine", "fit", "call". It contains a catR adaptive-testing simulation without fallback substitution and associated metadata or diagnostics needed to interpret the result.

---

`simulate_eyeprocess_irt_binary`

*Simulate dichotomous IRT responses with optional local dependence and missingness*

---

## Description

Simulate dichotomous IRT responses with optional local dependence and missingness

### Usage

```
simulate_eyeprocess_irt_binary(
  n_persons = 500L,
  items,
  theta = NULL,
  missing_rate = 0,
  testlet_sd = 0,
  seed = 1L,
  D = 1
)
```

### Arguments

| | |
|---|---|
| n_persons | Number of persons. |
| items | Item-parameter data frame or item collection. |
| theta | Latent-trait value or vector of latent-trait values. |
| missing_rate | Proportion of responses or observations set missing. |
| testlet_sd | Standard deviation of simulated testlet effects. |
| seed | Random-number seed for reproducible execution. |
| D | Logistic scaling constant. |

### Value

An object of class "eye_irt_simulation", stored as a named list, with components "responses", "probabilities", "theta", "items", "missing_rate", "testlet_sd", "seed". It contains dichotomous IRT responses with optional local dependence and missingness and associated metadata or diagnostics needed to interpret the result.

---

## simulate_from_model *Simulate data from a model or registered model specification*

---

### Description

Simulate data from a model or registered model specification

### Usage

```
simulate_from_model(model, ...)
```

### Arguments

| | |
|---|---|
| model | Registered model id/specification, simulation function, or an object exposing a ‘simulate_fun‘ function. |
| ... | Arguments passed to the simulator. |

## Value

An R object containing data from a model or registered model specification. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

`simulate_gaze_diffusion_data` *Simulate a hierarchical gaze-diffusion study*

---

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
simulate_gaze_diffusion_data(n_person = 80L, n_item = 20L, trials_per_item = 1L,
  gaze_effect = 0.35, contaminant_fraction = 0.02, time_step = 0.002,
  max_decision_time = 10, seed = 1L)
```

## Arguments

| | |
|---|---|
| `n_person` | Number of participants/items. |
| `n_item` | Number of participants/items. |
| `trials_per_item` | Replications per person-item. |
| `gaze_effect` | Drift effect of gaze feature. |
| `contaminant_fraction` | Contaminant fraction. |
| `time_step` | Time step for the built-in Wiener discretization fallback. |
| `max_decision_time` | Maximum fallback decision time in seconds. |
| `seed` | Seed. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## simulate_irt_model *Simulate from a registered multimodal IRT model*

### Description

Simulate from a registered multimodal IRT model

### Usage

```
simulate_irt_model(spec, ..., allow_experimental = TRUE)
```

### Arguments

| | |
|---|---|
| spec | IRT model or validation specification. |
| ... | Additional arguments passed to the selected model, engine, or method. |
| allow_experimental | Whether experimental models are permitted. |

### Value

An R object containing from a registered multimodal IRT model. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## simulate_multimodal_irt *Simulate multimodal IRT process data*

### Description

Generates deterministic synthetic person-item observations for staged M0-M3 development. Gaze is a non-negative count process and pupil is a neutral pupil-responsivity channel with explicit nuisance effects.

### Usage

```
simulate_multimodal_irt(
  n_person = 120L,
  n_item = 20L,
  seed = 42L,
  latent_cor = diag(4L),
  rt_sd = 0.3,
  gaze_size = 8,
  pupil_sd = 0.2,
  pupil_luminance = -0.2,
```

```
  gaze_x_effect = 0.08,
  gaze_y_effect = -0.05,
  missing_fraction = 0
)
```

## Arguments

| | |
|---|---|
| n_person, n_item | Positive integers. |
| seed | Random seed. |
| latent_cor | 4x4 correlation matrix for ability, speed, gaze process, and pupil responsivity. |
| rt_sd | Residual log-RT SD. |
| gaze_size | Negative-binomial size. |
| pupil_sd | Residual pupil-channel SD. |
| pupil_luminance, gaze_x_effect, gaze_y_effect | Nuisance coefficients. |
| missing_fraction | Independent channel dropout fraction used for a baseline stress condition. |

## Value

An 'eye_multimodal_simulation'.

---

# simulate_multimodal_m2

*Simulate from the M2 response + RT + gaze generative model*

---

## Description

Simulates the same level-1 likelihood used by [fit_multimodal_m2()] and retains all person/item hyperparameters and realized latent parameters as truth. Channel-specific dropout is applied only after the complete generative data have been created.

## Usage

```
simulate_multimodal_m2(
  n_person = 100L,
  n_item = 10L,
  mu_item = c(difficulty = 0, time_intensity = 4, gaze_intensity = 3.5),
  sd_person = c(ability = 1, speed = 0.5, gaze_process = 0.5),
 cor_person = matrix(c(1, 0.3, -0.3, 0.3, 1, -0.25, -0.3, -0.25, 1), 3L, 3L, byrow =
    TRUE),
  sd_item = c(difficulty = 0.75, time_intensity = 0.35, gaze_intensity = 0.6),
 cor_item = matrix(c(1, 0.25, 0.2, 0.25, 1, 0.3, 0.2, 0.3, 1), 3L, 3L, byrow = TRUE),
  nu_range = c(0.5, 0.8),
  gaze_shape = c(shape = 2, scale = 6),
```

```
  dropout = c(response = 0, rt = 0, gaze = 0),
  seed = 20260814L
)
```

## Arguments

| | |
|---|---|
| n_person | Number of persons. |
| n_item | Number of items. |
| mu_item | Means for item difficulty, log-time intensity, and log-gaze intensity. |
| sd_person | Person-side standard deviations for ability, speed, and gaze-process propensity. |
| cor_person | Person-side correlation matrix. |
| sd_item | Item-side standard deviations. |
| cor_item | Item-side correlation matrix. |
| nu_range | Uniform range for item time-discrimination parameters. |
| gaze_shape | Shape/scale parameters for inverse-gamma generation of negative-binomial shape parameters. |
| dropout | Named probabilities for response, RT, and gaze missingness. |
| seed | Reproducibility seed. |

## Value

An ‘eye_multimodal_m2_simulation‘.

---

# simulate_multimodal_m3

*Simulate the M3 response + RT + gaze + pupil generative model*

---

## Description

Generates a four-dimensional correlated person process and four-dimensional correlated item process, explicit pupil nuisance variables, complete-data truth, device/session metadata, blink/interpolation indicators and pupil dropout. Pupil scenarios include informative, weak, null, redundant and confound-only conditions so that validation includes cases where pupil should add no defensible psychometric information.

## Usage

```
simulate_multimodal_m3(
  n_person = 120L,
  n_item = 12L,
  pupil_signal = c("informative", "weak", "null", "redundant", "confounded"),
  pupil_missingness = c("mcar", "quality", "gaze", "ability", "device", "none"),
 mu_item = c(difficulty = 0, time_intensity = 4, gaze_intensity = 3.5, pupil_intensity =
    0),
```

```
  sd_person = c(ability = 1, speed = 0.5, gaze_process = 0.5, pupil_responsivity = 0.55),
  cor_person = matrix(c(1, 0.3, -0.3, 0.2, 0.3, 1, -0.25, -0.15, -0.3, -0.25, 1, 0.25,
    0.2, -0.15, 0.25, 1), 4L, 4L, byrow = TRUE),
  sd_item = c(difficulty = 0.75, time_intensity = 0.35, gaze_intensity = 0.6,
    pupil_intensity = 0.4),
  cor_item = matrix(c(1, 0.25, 0.2, 0.1, 0.25, 1, 0.3, 0.15, 0.2, 0.3, 1, 0.2, 0.1, 0.15,
    0.2, 1), 4L, 4L, byrow = TRUE),
  nu_range = c(0.5, 0.8),
  gaze_shape = c(shape = 2, scale = 6),
  pupil_noise = 0.65,
  confound_strength = c(baseline = 0.25, luminance = -0.35, gaze_x = 0.12, gaze_y = -0.1,
    quality = 0.2, blink = -0.18, interpolated = -0.12, time_on_task = 0.15),
  dropout = c(response = 0, rt = 0, gaze = 0.05, pupil = 0.12),
  device_effect = 0,
  session_effect = 0,
  seed = 20260815L
)
```

## Arguments

| | |
|---|---|
| n_person, n_item | Design size. |
| pupil_signal | Pupil signal scenario. |
| pupil_missingness | Missingness stress mechanism. |
| mu_item | Named numeric vector of population means for item difficulty, response-time intensity, gaze intensity, and pupil intensity. |
| sd_person | Named positive numeric vector of population standard deviations for person ability, speed, gaze-process propensity, and pupil responsivity. |
| cor_person | A 4 x 4 correlation matrix for person ability, speed, gaze-process, and pupil-responsivity effects, in that order. |
| sd_item | Named positive numeric vector of population standard deviations for item difficulty, response-time intensity, gaze intensity, and pupil intensity. |
| cor_item | A 4 x 4 correlation matrix for item difficulty, response-time intensity, gaze intensity, and pupil intensity, in that order. |
| nu_range | Length-two positive increasing numeric vector giving the lower and upper bounds for the item-specific response-time inverse-scale parameter ‘nu‘; the log-response-time residual standard deviation is ‘1 / nu‘. |
| gaze_shape | Named positive numeric vector with elements ‘shape‘ and ‘scale‘ defining the inverse-gamma generator for item-specific negative-binomial gaze dispersion. |
| pupil_noise | Residual SD for the pupil channel. |
| confound_strength | Named standardized nuisance coefficients. |
| dropout | Named base dropout probabilities for response, RT, gaze, pupil. |
| device_effect, session_effect | Additive pupil measurement shifts. |
| seed | Reproducibility seed. |

### Value

An ‘eye_multimodal_m3_simulation‘ retaining complete truth.

---

`simulate_multimodal_m4` *Simulate M4 multimodal sequential measurement data*

---

### Description

Generates deterministic synthetic response, RT, gaze, pupil, nuisance, and ordered latent-state data for software validation and methodological stress testing. Synthetic state labels are known truth for recovery only and are not psychological constructs.

### Usage

```
simulate_multimodal_m4(
  n_person = 80L,
  n_item = 12L,
  n_session = 1L,
  n_states = 2L,
  scenario = c("clear", "weak", "null", "persistent", "rapid_switch",
    "trait_conditioned", "rt_redundant", "gaze_redundant", "pupil_redundant",
    "nuisance_confounded", "device_confounded"),
  missingness = c("none", "mcar", "quality", "gaze", "pupil_quality", "device",
    "state_dependent"),
  missing_rate = 0.08,
  seed = 20260820L
)
```

### Arguments

| | |
|---|---|
| `n_person`, `n_item` | Number of persons and total item trials per person. |
| `n_session` | Number of non-overlapping ordered sessions per person. |
| `n_states` | True latent-state count. May be 1 through 4. |
| `scenario` | State/data-generating scenario: ‘clear‘, ‘weak‘, ‘null‘, ‘persistent‘, ‘rapid_switch‘, ‘trait_conditioned‘, ‘rt_redundant‘, ‘gaze_redundant‘, ‘pupil_redundant‘, ‘nuisance_confounded‘, or ‘device_confounded‘. |
| `missingness` | Missingness stress mechanism. |
| `missing_rate` | Base channel-missingness probability. |
| `seed` | Deterministic random seed. |

### Value

An ‘eye_multimodal_m4_simulation‘ containing ‘data‘, full generating ‘truth‘, scenario metadata, and interpretation boundary.

## simulate_presentation_variants *Simulate pre-registered presentation variants for review*

### Description

Simulate pre-registered presentation variants for review

### Usage

```
simulate_presentation_variants(
  audit,
  line_spacing_multiplier = 1.25,
  key_term_highlighting = TRUE
)
```

### Arguments

| | |
|---|---|
| audit | An accessibility audit. |
| line_spacing_multiplier | Example line-spacing multiplier for flagged rows. |
| key_term_highlighting | Whether the simulated review variant highlights key terms. |

### Value

An R object containing pre-registered presentation variants for review. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## simulate_process_cat *Simulate a simple process-aware CAT policy*

### Description

This is a design simulator, not a production testing engine.

### Usage

```
simulate_process_cat(
  item_bank,
  true_theta = 0,
  n_items = 10L,
  weights = c(response = 1, rt = 0, process = 0),
  burden_weight = 0,
  seed = 1
)
```

### Arguments

| | |
|---|---|
| item_bank | Value supplied to 'item_bank'; see Details for its model-specific role. |
| true_theta | Simulated true latent-trait value or values. |
| n_items | Number of items. |
| weights | Weights used to combine information components. |
| burden_weight | Penalty applied to expected burden. |
| seed | Random-number seed. |

### Value

An object of class "eye_process_cat_simulation", "data.frame", stored as a data frame, containing a simple process-aware CAT policy and associated metadata needed to interpret the result.

---

## simulate_process_validation_data

*Simulate a generic multimodal validation dataset with known truth*

---

### Description

This simulator is a neutral software-validation fixture, not a substantive psychological data-generating model. The generated process channels should not be interpreted as mental-state measurements.

### Usage

```
simulate_process_validation_data(
  condition,
  replication = 1L,
  seed = NULL,
  beta = 0.35
)
```

### Arguments

| | |
|---|---|
| condition | One-row validation condition. |
| replication | Replication index. |
| seed | Optional seed override. |
| beta | Known effect of 'x' on the generic process outcome. |

### Value

List with 'data' and 'truth'.

## simulate_strategy_mixture_data

*Simulate a theory-defined strategy-mixture study*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
simulate_strategy_mixture_data(n_person = 100L, n_item = 20L, signatures,
  trials_per_item = 1L, strategy_prevalence = NULL, feature_sd = 0.6, seed = 1L)
```

### Arguments

| | |
|---|---|
| n_person | Number of participants. |
| n_item | Number of items. |
| signatures | Strategy signature matrix. |
| trials_per_item | Trials per person-item combination. |
| strategy_prevalence | Strategy prevalence. |
| feature_sd | Feature residual SD. |
| seed | Seed. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## simulation_based_calibration

*Run simulation-based calibration*

### Description

Run simulation-based calibration

## Usage

```
simulation_based_calibration(
  simulator,
  fitter,
  posterior_draws,
  truth_extractor,
  replications = 100L,
  seed = 1L,
  ...
)
```

## Arguments

| | |
|---|---|
| simulator | Function returning simulated data and named truth. |
| fitter | Function receiving one simulation result. |
| posterior_draws | Function returning a numeric matrix/data frame whose columns are named parameters. |
| truth_extractor | Function returning a named numeric truth vector. |
| replications | Number of simulated data sets. |
| seed | Random seed. |
| ... | Passed to ‘simulator()‘. |

## Value

An ‘eye_sbc‘ object with parameter ranks and calibration summaries.

---

# simulation_rank_statistic

*Compute a simulation-based calibration rank statistic*

---

## Description

Compute a simulation-based calibration rank statistic

## Usage

```
simulation_rank_statistic(truth, draws, seed = NULL)
```

## Arguments

| | |
|---|---|
| truth | Scalar simulated truth. |
| draws | Posterior draws for the same parameter. |
| seed | Seed used only to randomize ties. |

## Value

A numeric value or vector containing a simulation-based calibration rank statistic.

---

`software_paper_claim_matrix` *Create or normalize a software-paper claim matrix*

---

## Description

Create or normalize a software-paper claim matrix

## Usage

```
software_paper_claim_matrix(
  claim,
  evidence_id = NA_character_,
  evidence_type = NA_character_,
  status = "pending",
  scope = NA_character_,
  source = NA_character_
)
```

## Arguments

| | |
|---|---|
| `claim` | Claim text. |
| `evidence_id` | Evidence identifiers. |
| `evidence_type` | Evidence type. |
| `status` | Status such as supported, qualified, pending, or unsupported. |
| `scope` | Explicit scope/qualification. |
| `source` | Optional source/location. |

## Value

A data frame containing or normalize a software-paper claim matrix. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## software_paper_coverage

*Compute descriptive evidence coverage*

### Description

Compute descriptive evidence coverage

### Usage

```
software_paper_coverage(x, supported = c("supported", "qualified"))
```

### Arguments

| | |
|---|---|
| x | Evidence bundle or claim matrix. |
| supported | Status labels counted as covered. |

### Value

A data frame containing descriptive evidence coverage. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## software_paper_evidence_bundle

*Construct a software-paper evidence bundle*

### Description

Construct a software-paper evidence bundle

### Usage

```
software_paper_evidence_bundle(
  claims = NULL,
  validation = NULL,
  examples = NULL,
  articles = NULL,
  benchmarks = NULL,
  reproducibility = NULL,
  metadata = list()
)
```

### Arguments

| | |
|---|---|
| claims | Claim table or list. |
| validation | Validation evidence table/list. |
| examples | Optional example inventory. |
| articles | Optional article inventory. |
| benchmarks | Optional benchmark evidence. |
| reproducibility | Optional reproducibility fingerprint. |
| metadata | Optional metadata. |

### Value

A named list with components "schema_version", "claims", "validation", "examples", "articles", "benchmarks", "reproducibility", "metadata", "created_utc", containing a software-paper evidence bundle and associated metadata or diagnostics.

---

## software_paper_gap_analysis

*Identify gaps in a software-paper evidence bundle*

---

### Description

Identify gaps in a software-paper evidence bundle

### Usage

```
software_paper_gap_analysis(x)
```

### Arguments

| | |
|---|---|
| x | Evidence bundle. |

### Value

A named list with components "requirement_gaps", "claim_gaps", containing gaps in a software-paper evidence bundle and associated metadata or diagnostics.

`software_paper_readiness` *Descriptive software-paper readiness audit*

## Description

Readiness is defined only against caller-specified requirements; it is not a journal acceptance prediction.

## Usage

```
software_paper_readiness(
  x,
  required_statuses = c("supported", "qualified"),
  require_validation = TRUE,
  require_reproducibility = TRUE,
  require_examples = TRUE,
  require_articles = TRUE
)
```

## Arguments

| | |
|---|---|
| `x` | Evidence bundle. |
| `required_statuses` | Statuses allowed for claims. |
| `require_validation` | Require non-empty validation evidence. |
| `require_reproducibility` | Require a reproducibility fingerprint. |
| `require_examples` | Require examples. |
| `require_articles` | Require articles. |

## Value

A named list with components "ready", "checks", "interpretation", containing descriptive software-paper readiness audit and associated metadata or diagnostics.

## software_paper_validation_table

*Summarise validation evidence for a software paper*

### Description

Summarise validation evidence for a software paper

### Usage

```
software_paper_validation_table(x)
```

### Arguments

| | |
|---|---|
| x | Validation result/evidence matrix/table. |

### Value

An R object containing validation evidence for a software paper. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## specification_coverage

*Fraction of planned specifications successfully evaluated*

### Description

Fraction of planned specifications successfully evaluated

### Usage

```
specification_coverage(x)
```

### Arguments

| | |
|---|---|
| x | Sensitivity result. |

### Value

A numeric value or vector containing fraction of planned specifications successfully evaluated.

`specification_curve_data` *Prepare ordered specification-curve data*

## Description

Prepare ordered specification-curve data

## Usage

```
specification_curve_data(x, effect = "effect", lower = NULL, upper = NULL)
```

## Arguments

| | |
|---|---|
| `x` | Sensitivity result. |
| `effect` | Effect column. |
| `lower` | Optional lower interval column. |
| `upper` | Optional upper interval column. |

## Value

A tabular R object containing ordered specification-curve data; rows represent analysis units and columns contain the returned quantities.

`split_half_process_reliability` *Split-half reliability for a trial-level process measure*

## Description

Split-half reliability for a trial-level process measure

## Usage

```
split_half_process_reliability(
  data,
  person,
  trial,
  measure,
  split = c("odd_even", "random"),
  repetitions = 100L,
  seed = 1L,
  aggregate_fun = mean
)
```

### Arguments

| | |
|---|---|
| data | Long trial-level data. |
| person | Participant column. |
| trial | Trial column. |
| measure | Measure column. |
| split | Odd/even or repeated random split. |
| repetitions | Number of random splits. |
| seed | Seed. |
| aggregate_fun | Within-half aggregation function. |

### Value

A tabular R object containing split-half reliability for a trial-level process measure; rows represent analysis units and columns contain the returned quantities.

---

## split_validation_plan *Split a validation plan into independent chunks*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
split_validation_plan(plan, chunks = NULL)
```

### Arguments

| | |
|---|---|
| plan | Validation plan. |
| chunks | Optional chunk identifiers. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

`standardized_spatial_quality` *Standardized vendor-neutral gaze data-quality metrics*

## Description

Compute and report separable eye-tracking quality dimensions: target-referenced accuracy, RMS sample-to-sample precision, spatial standard-deviation precision, BCEA, effective sampling behavior, sampling jitter, valid-data fraction, and data loss. User thresholds create review flags only and never trigger automatic exclusion.

## Usage

```
validate_gaze_quality_inputs(data, x = "gaze_x", y = "gaze_y", time = NULL,
  target_x = NULL, target_y = NULL, by = NULL, unit = "degrees",
  time_unit = "ms", unit_column = NULL)

compute_gaze_accuracy(data, x = "gaze_x", y = "gaze_y",
  target_x = "target_x", target_y = "target_y", by = NULL,
  unit = "degrees", output_unit = NULL, geometry = NULL)

compute_rms_s2s(data, x = "gaze_x", y = "gaze_y", time = NULL, by = NULL,
  unit = "degrees", output_unit = NULL, geometry = NULL,
  dimension = c("2d", "horizontal", "vertical"), time_unit = "ms",
  max_gap_ms = NULL)

compute_gaze_sd_precision(data, x = "gaze_x", y = "gaze_y", by = NULL,
  unit = "degrees", output_unit = NULL, geometry = NULL)

compute_bcea(data, x = "gaze_x", y = "gaze_y", by = NULL,
  probability = 0.68, unit = "degrees", output_unit = NULL,
  geometry = NULL)

compute_gaze_precision(data, x = "gaze_x", y = "gaze_y", time = NULL,
  by = NULL, probability = 0.68, unit = "degrees", output_unit = NULL,
  geometry = NULL, dimension = "2d", time_unit = "ms", max_gap_ms = NULL)

estimate_sampling_interval(data, time = "timestamp_ms", by = NULL,
  time_unit = "ms")

estimate_sampling_jitter(data, time = "timestamp_ms", by = NULL,
  time_unit = "ms")

estimate_effective_sampling_rate(data, time = "timestamp_ms", by = NULL,
  time_unit = "ms", nominal_sampling_hz = NULL,
  dropped_interval_factor = 1.5, x = NULL, y = NULL, valid = NULL)
```

```
compute_valid_sample_fraction(data, x = "gaze_x", y = "gaze_y",
  valid = NULL, by = NULL)

compute_gaze_data_loss(data, x = "gaze_x", y = "gaze_y",
  time = "timestamp_ms", valid = NULL, missing_reason = NULL,
  by = NULL, time_unit = "ms")

summarise_spatial_quality(data, x = "gaze_x", y = "gaze_y",
  target_x = "target_x", target_y = "target_y", time = NULL, by = NULL,
  unit = "degrees", output_unit = NULL, geometry = NULL,
  dimension = "2d", time_unit = "ms", max_gap_ms = NULL,
  probability = 0.68)

summarise_sampling_quality(data, time = "timestamp_ms", by = NULL,
  time_unit = "ms", nominal_sampling_hz = NULL,
  dropped_interval_factor = 1.5, x = NULL, y = NULL, valid = NULL)

create_gaze_quality_report(data, x = "gaze_x", y = "gaze_y",
  time = "timestamp_ms", target_x = "target_x", target_y = "target_y",
  valid = NULL, missing_reason = NULL, by = NULL, unit = "degrees",
  output_unit = NULL, geometry = NULL, time_unit = "ms",
  nominal_sampling_hz = NULL, bcea_probability = 0.68,
  max_gap_ms = NULL, thresholds = NULL, preprocessing_spec = NULL,
  event_detector = NULL, aoi_specification = NULL, quality_rules = NULL,
  model_specification = NULL, software_version = NULL,
  unit_column = NULL)

compare_gaze_quality_sessions(data, session = "session_id", by = NULL, ...)

compare_gaze_quality_conditions(data, condition = "condition", by = NULL, ...)

plot_gaze_accuracy(report, metric = "accuracy_mean", ...)
plot_gaze_precision(report, metric = "precision_rms_s2s", ...)
plot_bcea(report, ...)
plot_sampling_intervals(data, time = "timestamp_ms", time_unit = "ms", ...)
plot_gaze_quality_dashboard(report, ...)
report_gaze_quality(report, digits = 3L)

simulate_gaze_quality_calibration(seed = 20260918L,
  samples_per_target = 18L, nominal_sampling_hz = 60)
```

## Arguments

| | |
|---|---|
| `data` | A data frame or object coercible to a data frame. |
| `x, y` | Columns containing horizontal and vertical gaze coordinates. |
| `time` | Timestamp column. |
| `target_x, target_y` | Known validation-target coordinate columns. |

| | |
|---|---|
| by | Optional grouping columns defining the quality-analysis unit. |
| unit | Input coordinate unit: "degrees", "pixels", or "normalized". |
| output_unit | Optional explicit output coordinate unit. No conversion occurs when omitted. |
| geometry | Named screen/viewing geometry used only for explicit coordinate conversion. |
| time_unit | Timestamp unit: seconds, milliseconds, microseconds, or nanoseconds. |
| unit_column | Optional column used to reject mixed or conflicting coordinate units. For the canonical report, an existing coordinate_unit column is detected automatically when this argument is omitted. |
| dimension | RMS-S2S dimension: two-dimensional, horizontal, or vertical. |
| max_gap_ms | Optional explicit maximum interval permitted for an adjacent-sample RMS pair. |
| probability, bcea_probability | Probability region used for BCEA. |
| nominal_sampling_hz | Optional nominal hardware rate used for comparison and dropped-interval diagnostics. |
| dropped_interval_factor | Multiple of the nominal interval used to count long intervals. |
| valid | Optional explicit validity column. Missing gaze coordinates remain unavailable regardless of this flag. |
| missing_reason | Optional loss-reason column such as blink, tracker invalidity, or off-screen. |
| thresholds | Optional named study-specific review thresholds; these never exclude data automatically. |
| preprocessing_spec, event_detector, aoi_specification, quality_rules, model_specification, software_version | Optional provenance fields preserved on the canonical report. |
| session, condition | Grouping column used for convenience comparisons. |
| report | A quality-report-like data frame. |
| metric | Metric column plotted by a focused quality plot. |
| digits | Number of digits in the compact reporting string. |
| seed | Deterministic synthetic-data seed. |
| samples_per_target | Synthetic samples generated at each of nine validation targets. |
| ... | Additional arguments forwarded to the canonical report or base plotting function. |

## Details

Accuracy is target-referenced error. RMS-S2S precision quantifies adjacent-sample fluctuation and never bridges a missing sample. Spatial SD and BCEA quantify point-cloud spread around a stable target and use population standard deviations (denominator $n$). BCEA always records its probability level.

Effective sampling frequency is estimated from the effective sample count divided by an estimated recording duration (timestamp span plus one median positive inter-sample interval). When gaze coordinates and validity are supplied, the effective sample count contains valid gaze observations with finite timestamps; otherwise the metric describes the timestamp stream. With a nominal rate, `long_interval_count` counts intervals exceeding the declared nominal-gap multiple, whereas `dropped_interval_count` estimates the number of missing nominal samples represented by those long intervals.

The package deliberately does not impose universal acceptability cutoffs. A `gaze_quality_report` stores review flags and provenance while preserving all analysis units for downstream filtering, weighting, stratification, sensitivity analysis, and transparent reporting.

## Value

Metric functions return data frames with explicit units and diagnostics. `create_gaze_quality_report()` returns a data frame of class `gaze_quality_report` with a `gaze_quality_provenance` attribute. Plot functions draw base-R diagnostics and invisibly return the plotted data. `report_gaze_quality()` returns a compact character summary. `simulate_gaze_quality_calibration()` returns a deterministic synthetic nine-point validation data set spanning six quality profiles.

## References

Dunn, M. J., et al. (2024). A minimal reporting guideline for eye-tracking studies. *Behavior Research Methods*.

Niehorster, D. C., et al. (2026). How to determine the quality of eye-tracking data: A tutorial. *Behavior Research Methods*. doi:10.3758/s13428-026-03039-4.

## Examples

```
d <- simulate_gaze_quality_calibration(samples_per_target = 4)

q <- create_gaze_quality_report(
  d,
  by = c("profile", "target_id"),
  valid = "valid",
  missing_reason = "missing_reason",
  nominal_sampling_hz = 60
)

head(q[c(
  "profile", "target_id", "accuracy_mean", "precision_rms_s2s",
  "bcea", "effective_sampling_hz", "data_loss_fraction"
)])

report_gaze_quality(q)
```

`storage_transaction_manifest` *Return the transaction manifest*

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
storage_transaction_manifest(storage)
```

## Arguments

| | |
|---|---|
| `storage` | Storage object or path. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

`strategy_aoi_sensitivity` *Assess sensitivity to alternative AOI feature definitions*

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
strategy_aoi_sensitivity(datasets, spec, seed = 1L, ...)
```

## Arguments

| | |
|---|---|
| `datasets` | Named list of alternative trial-level datasets. |
| `spec` | Strategy specification. |
| `seed` | Seed. |
| `...` | Fit arguments. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## strategy_classification_uncertainty

*Quantify strategy-classification uncertainty*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
strategy_classification_uncertainty(object, threshold = 0.70)
```

### Arguments

| | |
|---|---|
| object | Strategy fit. |
| threshold | Minimum modal probability. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## strategy_label_switching_diagnostics

*Diagnose label stability across multiple starts*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
strategy_label_switching_diagnostics(object, tolerance = 1e-4)
```

### Arguments

| | |
|---|---|
| object | Strategy fit. |
| tolerance | Log-likelihood tolerance for equivalent starts. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## strategy_posterior_probabilities

*Posterior strategy probabilities*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
strategy_posterior_probabilities(object)
```

### Arguments

| | |
|---|---|
| object | Strategy fit. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## streaming_score_history

*Extract streaming score history*

### Description

Extract streaming score history

### Usage

```
streaming_score_history(x)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |

### Value

An R object containing streaming score history. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## stress_test_latent_distribution
*Stress test latent distribution*

### Description

Stress test latent distribution

### Usage

```
stress_test_latent_distribution(runner, replications = 50L, seed = 20260808L)
```

### Arguments

| | |
|---|---|
| runner | Function that executes one stress-test scenario. |
| replications | Number of simulation or validation replications. |
| seed | Random-number seed. |

### Value

An object of class "eye_irt_stress_test", "data.frame", stored as a data frame, containing stress test latent distribution and associated metadata needed to interpret the result.

## stress_test_local_dependence
*Stress test local dependence*

### Description

Stress test local dependence

### Usage

```
stress_test_local_dependence(
  runner,
  strengths = c(0, 0.2, 0.5, 0.8),
  replications = 50L,
  seed = 20260808L
)
```

### Arguments

| | |
|---|---|
| runner | Function that executes one stress-test scenario. |
| strengths | Local-dependence strengths to evaluate. |
| replications | Number of simulation or validation replications. |
| seed | Random-number seed. |

## Value

An object of class "eye_irt_stress_test", "data.frame", stored as a data frame, containing stress test local dependence and associated metadata needed to interpret the result.

---

`stress_test_missingness` *Stress test missingness*

---

## Description

Stress test missingness

## Usage

```
stress_test_missingness(
  runner,
  mechanisms = c("MCAR", "MAR", "MNAR_omission", "not_reached"),
  rates = c(0.05, 0.15, 0.3),
  replications = 50L,
  seed = 20260808L
)
```

## Arguments

| | |
|---|---|
| `runner` | Function that executes one stress-test scenario. |
| `mechanisms` | Missingness mechanisms to evaluate. |
| `rates` | Missingness rates to evaluate. |
| `replications` | Number of simulation or validation replications. |
| `seed` | Random-number seed. |

## Value

An object of class "eye_irt_stress_test", "data.frame", stored as a data frame, containing stress test missingness and associated metadata needed to interpret the result.

## stress_test_misspecification

*Run a generic misspecification stress-test grid*

### Description

Run a generic misspecification stress-test grid

### Usage

```
stress_test_misspecification(
  scenarios,
  runner,
  replications = 50L,
  seed = 20260808L
)
```

### Arguments

| | |
|---|---|
| scenarios | Data frame or named list describing scenarios. |
| runner | Function ‘runner(scenario, replicate)‘ returning a one-row or tidy data frame. Errors are retained as classified failures. |
| replications | Number of simulation or validation replications. |
| seed | Random-number seed. |

### Value

An object of class "eye_irt_stress_test", "data.frame", stored as a data frame, containing a generic misspecification stress-test grid and associated metadata needed to interpret the result.

## stress_test_preprocessing

*Stress test preprocessing*

### Description

Stress test preprocessing

### Usage

```
stress_test_preprocessing(
  runner,
  variants,
  replications = 25L,
  seed = 20260808L
)
```

### Arguments

| | |
|---|---|
| runner | Function that executes one stress-test scenario. |
| variants | Preprocessing variants to evaluate. |
| replications | Number of simulation or validation replications. |
| seed | Random-number seed. |

### Value

An object of class "eye_irt_stress_test", "data.frame", stored as a data frame, containing stress test preprocessing and associated metadata needed to interpret the result.

---

## stress_test_process_pipeline

*Stress-test an analysis under explicit synthetic corruptions*

---

### Description

Stress-test an analysis under explicit synthetic corruptions

### Usage

```
stress_test_process_pipeline(
  data,
  plans,
  analysis_fun,
  metric_fun = .ep09_default_sensitivity_extract,
  ...
)
```

### Arguments

| | |
|---|---|
| data | Baseline data. |
| plans | List of corruption plans. |
| analysis_fun | Function ‘(data, plan)‘. |
| metric_fun | Function ‘(analysis_result, plan)‘ returning scalar/list/data.frame metrics. |
| ... | Passed to ‘apply_synthetic_corruption()‘. |

### Value

An object of class "eye_process_stress_test", stored as a named list, with components "plans", "results", "baseline_hash", "created_at", "caveat". It contains stress-test an analysis under explicit synthetic corruptions and associated metadata or diagnostics needed to interpret the result.

## stress_test_speededness *Stress test speededness*

### Description

Stress test speededness

### Usage

```
stress_test_speededness(
  runner,
  proportions = c(0, 0.1, 0.25, 0.4),
  replications = 50L,
  seed = 20260808L
)
```

### Arguments

| | |
|---|---|
| runner | Function that executes one stress-test scenario. |
| proportions | Speededness proportions to evaluate. |
| replications | Number of simulation or validation replications. |
| seed | Random-number seed. |

### Value

An object of class "eye_irt_stress_test", "data.frame", stored as a data frame, containing stress test speededness and associated metadata needed to interpret the result.

## stress_test_summary *Summarise stress-test metrics*

### Description

Summarise stress-test metrics

### Usage

```
stress_test_summary(x, metric = "effect")
```

### Arguments

| | |
|---|---|
| x | Stress-test result. |
| metric | Numeric metric column. |

## Value

A data frame containing stress-test metrics. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

---

`stress_tolerance_frontier`
*Identify the empirical stress frontier for a metric*

---

## Description

Identify the empirical stress frontier for a metric

## Usage

```
stress_tolerance_frontier(x, severity, metric, acceptable)
```

## Arguments

| | |
|---|---|
| `x` | Stress-test result. |
| `severity` | Numeric corruption/severity column. |
| `metric` | Metric column. |
| `acceptable` | Function returning TRUE/FALSE for metric values. |

## Value

A data frame containing the empirical stress frontier for a metric. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

---

`structural_transition_mask`
*Define structural-zero and allowed transition masks*

---

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
structural_transition_mask(states, forbidden = NULL, allowed = NULL,
  allow_self = TRUE, structural_zeros = NULL, allowed_transitions = NULL)
```

### Arguments

| | |
|---|---|
| `states` | State labels. |
| `forbidden` | Two-column data frame/matrix of forbidden from-to pairs. |
| `allowed` | Two-column data frame/matrix of explicitly allowed pairs. |
| `allow_self` | Whether self transitions are allowed by default. |
| `structural_zeros` | Alias for ‘forbidden‘. |
| `allowed_transitions` | Alias for ‘allowed‘. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

## `summarise_eyeprocess_stress_evidence` *Summarise executed measurement-stress evidence*

---

### Description

Summarise executed measurement-stress evidence

### Usage

```
summarise_eyeprocess_stress_evidence(x)
```

### Arguments

| | |
|---|---|
| `x` | Object to validate, summarize, verify, or otherwise process. |

### Value

A tabular R object containing executed measurement-stress evidence; rows represent analysis units and columns contain the returned quantities.

## summarise_eye_benchmark

*Summarise benchmark timing and memory by problem size*

### Description

Summarise benchmark timing and memory by problem size

### Usage

```
summarise_eye_benchmark(x)
```

### Arguments

| | |
|---|---|
| x | Benchmark result. |

### Value

A logical value or vector indicating benchmark timing and memory by problem size.

## summarise_process_negative_controls

*Summarise process negative controls*

### Description

Summarise process negative controls

### Usage

```
summarise_process_negative_controls(x, effect = "effect", threshold = 0)
```

### Arguments

| | |
|---|---|
| x | Negative-control result. |
| effect | Effect column. |
| threshold | Optional absolute effect threshold. |

### Value

A logical value or vector indicating process negative controls.

summarise_process_sensitivity *Summarise process sensitivity results*

## Description

Summarise process sensitivity results

## Usage

```
summarise_process_sensitivity(
  x,
  effect = "effect",
  p_value = NULL,
  threshold = 0,
  alpha = 0.05
)
```

## Arguments

| | |
|---|---|
| x | Sensitivity result. |
| effect | Effect column. |
| p_value | Optional p-value column. |
| threshold | Optional substantive effect threshold. |
| alpha | Significance threshold used only when ‘p_value‘ is supplied. |

## Value

A data frame containing process sensitivity results. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

summarise_process_validation *Summarise a process-validation result*

## Description

Summarise a process-validation result

## Usage

```
summarise_process_validation(x, by = NULL)
```

### Arguments

| | |
|---|---|
| x | Validation result. |
| by | Optional grouping variables in addition to parameter. |

### Value

A tabular R object containing a process-validation result; rows represent analysis units and columns contain the returned quantities.

---

## summarise_validation_acceptance

*Summarise an acceptance matrix*

---

### Description

Summarise an acceptance matrix

### Usage

```
summarise_validation_acceptance(x, by = character())
```

### Arguments

| | |
|---|---|
| x | Object to validate, summarize, verify, or otherwise process. |
| by | Grouping variables or aggregation level. |

### Value

A tabular R object containing an acceptance matrix; rows represent analysis units and columns contain the returned quantities.

---

## summarize_parameter_recovery

*Summarise parameter recovery*

---

### Description

Summarise parameter recovery

### Usage

```
summarize_parameter_recovery(
  results,
  by = c("scenario", "engine", "parameter"),
  interval_level = 0.95
)
```

### Arguments

| | |
|---|---|
| `results` | Canonical or raw recovery results. |
| `by` | Grouping columns. |
| `interval_level` | Nominal interval level, used only for labelling. |

### Value

A tabular R object containing parameter recovery; rows represent analysis units and columns contain the returned quantities.

---

## `summarize_process_windows` *Summarize extracted process windows*

---

### Description

Summarize extracted process windows

### Usage

```
summarize_process_windows(x, by = NULL)
```

### Arguments

| | |
|---|---|
| `x` | ‘eye_process_windows‘ object. |
| `by` | Optional grouping columns present in the extracted table. |

### Value

An R object containing extracted process windows. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

---

## `summary.eye_validation_bundle` *Summarize a validation bundle object*

---

### Description

Summarize a validation bundle object

### Usage

```
## S3 method for class 'eye_validation_bundle'
summary(object, ...)
```

### Arguments

| | |
|---|---|
| object | Object supplied to the S3 method. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

A named list with components "model_name", "manifest", "report", containing a validation bundle object and associated metadata or diagnostics.

---

## summary.eye_validation_job_plan

*summary eye validation job plan*

---

### Description

S3 method supporting a research-scale eyeprocess object.

### Usage

```
## S3 method for class 'eye_validation_job_plan'
summary(object, ...)
```

### Arguments

| | |
|---|---|
| object | Value for ‘object‘. See the function description and relevant article for constraints. |
| ... | Additional arguments passed to the selected engine or method. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

## synthetic_corruption_plan

*Define synthetic measurement corruptions for stress testing*

---

### Description

Define synthetic measurement corruptions for stress testing

### Usage

```
synthetic_corruption_plan(
  missingness = 0,
  pupil_dropout = 0,
  gaze_offset_x = 0,
  gaze_offset_y = 0,
  sampling_jitter_sd = 0,
  aoi_label_noise = 0,
  device_shift = 0,
  trial_drop = 0,
  seed = 1L
)
```

### Arguments

| | |
|---|---|
| missingness | Generic missingness proportion. |
| pupil_dropout | Pupil dropout proportion. |
| gaze_offset_x, gaze_offset_y | Additive gaze-coordinate offsets. |
| sampling_jitter_sd | Timestamp jitter SD in timestamp units. |
| aoi_label_noise | Proportion of AOI labels randomly reassigned. |
| device_shift | Additive shift for a declared device-sensitive numeric column. |
| trial_drop | Proportion of rows/trials removed. |
| seed | Seed. |

### Value

An object of class "eye_synthetic_corruption_plan", stored as a named list, with components "missingness", "pupil_dropout", "gaze_offset_x", "gaze_offset_y", "sampling_jitter_sd", "aoi_label_noise", "device_shift", "trial_drop", "seed", "status". It contains define synthetic measurement corruptions for stress testing and associated metadata or diagnostics needed to interpret the result.

---

## theory_strategy_spec *Define a theory-constrained strategy-mixture model*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
theory_strategy_spec(strategies = NULL, feature_columns = NULL, response = "score",
  participant = "participant_id", item = "item_id", condition = NULL,
  item_availability = NULL, engine = c("em", "stan"), multiple_starts = 10L,
  anchor_strength = 3, chains = 4L, parallel_chains = min(4L, chains),
 iter_warmup = 1000L, iter_sampling = 1000L, adapt_delta = 0.95, max_treedepth = 12L,
  prototypes = NULL, feature_sd = NULL, prior = NULL)
```

## Arguments

| | |
|---|---|
| strategies | Named list of strategy signatures. Each signature is a named |
| feature_columns | Process-feature columns. |
| response | Binary response column. |
| participant | Participant identifier. |
| item | Item identifier. |
| condition | Optional condition column. |
| item_availability | Optional item-by-strategy availability matrix or data frame. |
| engine | Estimation engine. |
| multiple_starts | Number of starts for the EM baseline. |
| anchor_strength | Prior/penalty strength anchoring classes to signatures. |
| chains | Stan controls. |
| parallel_chains | Stan controls. |
| iter_warmup | Stan controls. |
| iter_sampling | Stan controls. |
| adapt_delta | Stan controls. |
| max_treedepth | Stan controls. |
| prototypes | Value for 'prototypes'. See the function description and relevant article for constraints. |
| feature_sd | Value for 'feature_sd'. See the function description and relevant article for constraints. |
| prior | Value for 'prior'. See the function description and relevant article for constraints. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

`timestamp_fidelity_audit` *Timestamp semantic-fidelity audit*

## Description

Timestamp semantic-fidelity audit

## Usage

```
timestamp_fidelity_audit(
  source,
  roundtrip,
  source_time = "timestamp",
  roundtrip_time = source_time,
  source_unit = "seconds",
  roundtrip_unit = source_unit,
  key = NULL,
  tolerance = 1e-06
)
```

## Arguments

| | |
|---|---|
| `source` | Original data. |
| `roundtrip` | Round-tripped data. |
| `source_time` | Source timestamp column. |
| `roundtrip_time` | Round-trip timestamp column. |
| `source_unit`, `roundtrip_unit` | One of seconds, milliseconds, microseconds, or nanoseconds. |
| `key` | Optional alignment key. |
| `tolerance` | Seconds-scale tolerance after normalization. |

## Value

An ‘eye_timestamp_fidelity‘ object.

## transition_residual_diagnostics *Compute transition residual diagnostics*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
transition_residual_diagnostics(object, type = c("pearson", "deviance", "randomized"))
```

### Arguments

| | |
|---|---|
| object | Dynamic IRTree fit. |
| type | Pearson, deviance, or randomized quantile residuals. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## update_person_score *Update a partial person score with one new response*

### Description

Update a partial person score with one new response

### Usage

```
update_person_score(
  model,
  current_pattern,
  item_position,
  response,
  method = c("MAP", "EAP"),
  ...
)
```

### Arguments

| | |
|---|---|
| model | Calibrated model. |
| current_pattern | Existing full-length partial pattern. |
| item_position | Item receiving the new response. |
| response | New response. |
| method | Scoring method. |
| ... | Additional arguments passed to the underlying method or helper. |

### Value

A named list with components "pattern", "score", containing update a partial person score with one new response and associated metadata or diagnostics.

---

## upgrade_eyeprocess_model

*Upgrade a legacy eyeprocess model*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
upgrade_eyeprocess_model(x, target_version = "1.0.0")
```

### Arguments

| | |
|---|---|
| x | Model object. |
| target_version | Target model-contract version. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## upgrade_eye_dataset *Upgrade a legacy eye dataset*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
upgrade_eye_dataset(x, target_version = "2.0.0", copy = TRUE)
```

### Arguments

| | |
|---|---|
| x | Dataset. |
| target_version | Target schema version. |
| copy | Whether to copy before migration. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## validate_against_reference *Compare a validation result with a frozen reference*

### Description

Compare a validation result with a frozen reference

### Usage

```
validate_against_reference(x, reference, tolerance = 1e-06)
```

### Arguments

| | |
|---|---|
| x | Validation result. |
| reference | Frozen reference object or RDS path. |
| tolerance | Numeric tolerance for matched summary values. |

### Value

An object of class "eye_validation_reference_comparison", stored as a named list, with components "table", "tolerance", "pass", "reference_hash", "current_hash". It contains a validation result with a frozen reference and associated metadata or diagnostics needed to interpret the result.

## validate_benchmark_study

*Validate benchmark integrity and relational constraints*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
validate_benchmark_study(study = eyeprocess_benchmark_study(), verify_hashes = TRUE)
```

### Arguments

| | |
|---|---|
| study | Benchmark object or path. |
| verify_hashes | Whether to verify MD5 hashes. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## validate_bids_eye_semantics

*Validate BIDS eye-tracking semantics*

### Description

Implements a lightweight contract for BIDS 1.11.1 eye-tracking physiology data. It does not replace the BIDS Validator.

### Usage

```
validate_bids_eye_semantics(data, metadata, events_metadata = NULL)
```

### Arguments

| | |
|---|---|
| data | Eye-tracking physiology table. |
| metadata | Parsed JSON sidecar as a named list. |
| events_metadata | Optional event-sidecar metadata containing 'StimulusPresentation' information for gaze-on-screen recordings. |

### Value

An 'eye_bids_semantic_audit' object.

validate_decision_manifest *Validate a research decision manifest*

## Description

Validate a research decision manifest

## Usage

```
validate_decision_manifest(
  x,
 required_domains = c("sampling", "validity", "fixation", "pupil", "aoi", "model",
    "sensitivity", "exclusions"),
  require_nonempty = FALSE
)
```

## Arguments

| | |
|---|---|
| x | Manifest. |
| required_domains | Domains that must exist. |
| require_nonempty | If TRUE, required domains must contain at least one decision. |

## Value

A logical value or vector indicating a research decision manifest.

validate_engine_adapter *Validate an external-engine adapter contract*

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
validate_engine_adapter(result, require_fit = FALSE)
```

## Arguments

| | |
|---|---|
| result | Adapter result. |
| require_fit | Require fitted status. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`validate_eyeprocess_external_irt_fit`
*Validate that an external IRT fit used the requested engine*

---

### Description

Validate that an external IRT fit used the requested engine

### Usage

```
validate_eyeprocess_external_irt_fit(x, engine = NULL)
```

### Arguments

| | |
|---|---|
| `x` | Object to validate, summarize, verify, or otherwise process. |
| `engine` | Requested estimation or analysis engine. |

### Value

A logical value or vector indicating that an external IRT fit used the requested engine.

---

`validate_eyeprocess_irt_item_bank`
*Validate an adaptive IRT item bank*

---

### Description

Validate an adaptive IRT item bank

### Usage

```
validate_eyeprocess_irt_item_bank(x)
```

### Arguments

| | |
|---|---|
| `x` | Object to validate, summarize, verify, or otherwise process. |

### Value

A logical value or vector indicating an adaptive IRT item bank.

## validate_eyeprocess_irt_model_spec

*Validate an eyeprocess IRT model specification*

### Description

Validate an eyeprocess IRT model specification

### Usage

```
validate_eyeprocess_irt_model_spec(x)
```

### Arguments

| | |
|---|---|
| x | Object to validate, summarize, verify, or otherwise process. |

### Value

A logical value or vector indicating an eyeprocess IRT model specification.

## validate_eyeprocess_joint_process_irt_spec

*Validate a joint process IRT specification*

### Description

Validate a joint process IRT specification

### Usage

```
validate_eyeprocess_joint_process_irt_spec(x)
```

### Arguments

| | |
|---|---|
| x | Object to validate, summarize, verify, or otherwise process. |

### Value

A logical value or vector indicating a joint process IRT specification.

## validate_eyeprocess_validation_plan *Validate a validation-evidence plan*

### Description

Validate a validation-evidence plan

### Usage

```
validate_eyeprocess_validation_plan(x)
```

### Arguments

| | |
|---|---|
| x | Object to validate, summarize, verify, or otherwise process. |

### Value

A logical value or vector indicating a validation-evidence plan.

## validate_eye_pipeline *Validate a governed eyeprocess pipeline*

### Description

Validate a governed eyeprocess pipeline

### Usage

```
validate_eye_pipeline(x)
```

### Arguments

| | |
|---|---|
| x | Pipeline. |

### Value

A logical value or vector indicating a governed eyeprocess pipeline.

## validate_eye_prov_graph *Validate a provenance graph*

### Description

Validate a provenance graph

### Usage

```
validate_eye_prov_graph(x)
```

### Arguments

| | |
|---|---|
| x | Provenance graph. |

### Value

A logical value or vector indicating a provenance graph.

## validate_eye_storage_metadata *Validate storage metadata and partition fingerprints*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
validate_eye_storage_metadata(storage, verify_hashes = TRUE)
```

### Arguments

| | |
|---|---|
| storage | Storage object or path. |
| verify_hashes | Whether to recompute all fingerprints. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

validate_feature_availability *Validate feature availability against an analysis cutoff*

## Description

Validate feature availability against an analysis cutoff

## Usage

```
validate_feature_availability(provenance, cutoff)
```

## Arguments

| | |
|---|---|
| provenance | Feature provenance table. |
| cutoff | Scalar cutoff or named vector by feature. |

## Value

A data frame containing feature availability against an analysis cutoff. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

validate_gazepoint_workflow *Validate an integrated Gazepoint workflow result*

## Description

Validate an integrated Gazepoint workflow result

## Usage

```
validate_gazepoint_workflow(x)
```

## Arguments

| | |
|---|---|
| x | An ‘eye_gazepoint_workflow‘ object. |

## Value

A data frame of workflow checks.

## validate_hed_event_semantics *Minimal HED annotation audit for event tables*

### Description

This function checks presence, non-empty annotations and balanced grouping. It is deliberately a structural audit rather than a full HED validator; use the official HED validation tooling when formal schema validation is needed.

### Usage

```
validate_hed_event_semantics(events, hed_column = "HED")
```

### Arguments

| | |
|---|---|
| events | Value supplied to 'events'; see Details for its model-specific role. |
| hed_column | Column containing HED annotations. |

### Value

A data frame containing minimal HED annotation audit for event tables. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## validate_irt_model *Validate a registered multimodal IRT model*

### Description

Validate a registered multimodal IRT model

### Usage

```
validate_irt_model(spec, validation = NULL, ...)
```

### Arguments

| | |
|---|---|
| spec | IRT model or validation specification. |
| validation | Validation results or validation specification. |
| ... | Additional arguments passed to the selected model, engine, or method. |

### Value

An object of class "eye_irt_evidence_grade", stored as a named list, with components "model_id", "grade", "checks", "recovery", "contract", "warning". It contains a registered multimodal IRT model and associated metadata or diagnostics needed to interpret the result.

## validate_latent_space_process_similarity *Validate latent-space proximity against process similarity*

### Description

Validate latent-space proximity against process similarity

### Usage

```
validate_latent_space_process_similarity(
  object,
  process_matrix,
  entity = c("person", "item")
)
```

### Arguments

| | |
|---|---|
| object | A fitted eyeprocess model or audit object. |
| process_matrix | Rows correspond to persons or items in the same order as the fitted latent coordinates. |
| entity | Entity type to map or validate. |

### Value

An object of class "eye_latent_space_process_validation", stored as a named list, with components "entity", "spearman_distance_correlation", "latent_distance", "process_distance", "interpretation". It contains latent-space proximity against process similarity and associated metadata or diagnostics needed to interpret the result.

## validate_model_object *Validate a fitted model against the stable model contract*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
validate_model_object(object, strict = FALSE)
```

### Arguments

| | |
|---|---|
| object | Model object. |
| strict | Whether warnings become errors. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`validate_multimodal_irt`

*Validate a multimodal IRT development object*

---

## Description

Validate a multimodal IRT development object

## Usage

```
validate_multimodal_irt(x)
```

## Arguments

| | |
|---|---|
| `x` | Measurement, fit, simulation, or process-information object. |

## Value

An 'eye_multimodal_validation'.

---

`validate_multimodal_m2`

*Validate an M2 fit or simulation*

---

## Description

For simulations, performs data/support and identifiability checks. For fitted models, additionally audits MCMC diagnostics and optionally channel-specific posterior predictive checks.

## Usage

```
validate_multimodal_m2(x, include_ppc = TRUE, rhat_max = 1.01, ess_min = 200)
```

## Arguments

| | |
|---|---|
| `x` | M2 simulation or fit. |
| `include_ppc` | Include posterior predictive checks for fitted M2 models. |
| `rhat_max` | Maximum acceptable R-hat. |
| `ess_min` | Minimum bulk/tail ESS threshold. |

## Value

An 'eye_multimodal_m2_validation'.

## validate_multimodal_m3

*Validate an M3 simulation or fitted four-channel model*

### Description

Combines structural support, sampler diagnostics, PPC, pupil-confound availability, device/missingness summaries and explicit interpretive boundaries. Validation of synthetic or computational behavior is not empirical construct validation.

### Usage

```
validate_multimodal_m3(
  x,
  include_ppc = TRUE,
  rhat_max = 1.05,
  ess_min = 50,
  ebfmi_min = 0.3
)
```

### Arguments

| | |
|---|---|
| x | M3 simulation or fit. |
| include_ppc | Include M3 PPC for fits. |
| rhat_max, ess_min, ebfmi_min | Diagnostic thresholds. |

### Value

An 'eye_multimodal_m3_validation'.

## validate_multimodal_m4

*Validate M4 data, computation, state behavior, and evidence*

### Description

Combines structural identifiability, sampler diagnostics, state uncertainty, PPC, and optionally supplied information/negative-control/sensitivity/recovery evidence into domain-specific statuses. 'PASS' means the declared checks pass; it never means that state labels have substantive psychological validity.

### Usage

```
validate_multimodal_m4(
  x,
  information = NULL,
  negative_controls = NULL,
  sensitivity = NULL,
  recovery = NULL,
  include_ppc = TRUE
)
```

### Arguments

| | |
|---|---|
| x | M4 fit. |
| information | Optional M4 information object. |
| negative_controls | Optional M4 negative-controls object. |
| sensitivity | Optional M4 sensitivity object. |
| recovery | Optional M4 recovery object. |
| include_ppc | Whether to compute PPC summaries. |

### Value

An ‘eye_multimodal_m4_validation‘.

---

## validate_process_measure_registry *Validate a process-measure registry*

---

### Description

Validate a process-measure registry

### Usage

```
validate_process_measure_registry(registry)
```

### Arguments

| | |
|---|---|
| registry | Registry data frame. |

### Value

A logical value or vector indicating a process-measure registry.

## validate_process_validation_design

*Validate a process-validation design*

### Description

Validate a process-validation design

### Usage

```
validate_process_validation_design(x)
```

### Arguments

| | |
|---|---|
| x | Validation design. |

### Value

A logical value or vector indicating a process-validation design.

## validate_process_windows

*Validate a process-window representation*

### Description

Validate a process-window representation

### Usage

```
validate_process_windows(x)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |

### Value

A data frame containing a process-window representation. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

`validate_strategy_manipulation` *Validate strategy posteriors against an experimental manipulation*

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
validate_strategy_manipulation(object, condition, expected_strategy,
  minimum_contrast = 0)
```

## Arguments

| | |
|---|---|
| `object` | Fitted strategy-mixture object. |
| `condition` | Condition column in the original trial data. |
| `expected_strategy` | Named character vector mapping condition values to prespecified strategy labels. |
| `minimum_contrast` | Minimum mean posterior-probability contrast over alternative strategies. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`validate_vendor_semantics` *Validate imported data against a vendor semantic contract*

## Description

Validate imported data against a vendor semantic contract

## Usage

```
validate_vendor_semantics(data, contract, metadata = list())
```

## Arguments

| | |
|---|---|
| `data` | Imported/canonical table. |
| `contract` | ‘eye_vendor_schema_contract‘. |
| `metadata` | Optional named metadata list. |

## Value

An object of class "eye_vendor_semantic_validation", stored as a named list, with components "pass", "vendor", "version", "fields", "aliases", "timestamp", "units", "contract". It contains imported data against a vendor semantic contract and associated metadata or diagnostics needed to interpret the result.

---

`validate_vendor_timestamp_semantics`
*Validate vendor-specific timestamp semantics*

---

## Description

The audit records expected semantics rather than silently coercing clocks. Tobii data can carry device/system timing; Pupil Labs Neon timestamps are high-resolution UTC nanoseconds in native recordings; Gazepoint may expose native monotonic and media-relative clocks depending on export type.

## Usage

```
validate_vendor_timestamp_semantics(
  data,
  vendor,
  device_time = NULL,
  system_time = NULL,
  media_time = NULL
)
```

## Arguments

| | |
|---|---|
| `data` | Input data frame or compatible tabular object. |
| `vendor` | Vendor identifier. |
| `device_time` | Device timestamp column. |
| `system_time` | System timestamp column. |
| `media_time` | Media/stimulus timestamp column. |

## Value

An object of class "eye_vendor_timestamp_semantics", stored as a named list, with components "vendor", "pass", "clocks". It contains vendor-specific timestamp semantics and associated metadata or diagnostics needed to interpret the result.

## validation_acceptance_matrix

*Evaluate a table against named validation rules*

### Description

Evaluate a table against named validation rules

### Usage

```
validation_acceptance_matrix(summary, rules, id_cols = character())
```

### Arguments

| | |
|---|---|
| summary | Validation summary table. |
| rules | Collection of validation acceptance rules. |
| id_cols | Columns identifying validation scenarios. |

### Value

A tabular R object containing a table against named validation rules; rows represent analysis units and columns contain the returned quantities.

## validation_acceptance_rule

*Define a validation acceptance rule*

### Description

Define a validation acceptance rule

### Usage

```
validation_acceptance_rule(
  metric,
  direction = c("max", "min", "between", "equals"),
  threshold,
  upper = NULL,
  tolerance = 0
)
```

### Arguments

| | |
|---|---|
| metric | Metric name or metric column. |
| direction | Direction vector used to project multidimensional information. |
| threshold | Decision or diagnostic threshold. |
| upper | Optional upper threshold for interval-style acceptance rules. |
| tolerance | Numerical or decision tolerance. |

### Value

An object of class "eye_validation_acceptance_rule", stored as a named list, with components "metric", "direction", "threshold", "upper", "tolerance". It contains define a validation acceptance rule and associated metadata or diagnostics needed to interpret the result.

---

## validation_bundle_manifest

*Create a machine-readable validation manifest*

---

### Description

Create a machine-readable validation manifest

### Usage

```
validation_bundle_manifest(x)
```

### Arguments

| | |
|---|---|
| x | Object to process, inspect, compare, or plot. |

### Value

A data frame containing a machine-readable validation manifest. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## validation_calibration_summary

*Summarize prediction calibration*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
validation_calibration_summary(x, by = c("model_family", "scenario_id"), bins = 10L)
```

### Arguments

| | |
|---|---|
| x | Validation collection or prediction data frame. |
| by | Grouping columns. |
| bins | Number of reliability bins. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## validation_condition_id

*Return stable validation condition identifiers*

### Description

Return stable validation condition identifiers

### Usage

```
validation_condition_id(x)
```

### Arguments

| | |
|---|---|
| x | Validation design or expanded condition table. |

### Value

A character value or vector containing return stable validation condition identifiers.

`validation_condition_ranking` *Rank validation conditions by a transparent robustness score*

## Description

Rank validation conditions by a transparent robustness score

## Usage

```
validation_condition_ranking(
  x,
  weights = c(rmse = 1, abs_bias = 1, coverage_error = 1, failure_rate = 1)
)
```

## Arguments

| | |
|---|---|
| `x` | Validation result. |
| `weights` | Named weights for rmse, absolute bias, coverage error, and failure rate. |

## Value

A tabular R object containing rank validation conditions by a transparent robustness score; rows represent analysis units and columns contain the returned quantities.

---

`validation_coverage_table` *Interval-coverage table*

## Description

Interval-coverage table

## Usage

```
validation_coverage_table(x, nominal = 0.95, by = NULL)
```

## Arguments

| | |
|---|---|
| `x` | Validation result. |
| `nominal` | Nominal coverage used for deviation reporting. |
| `by` | Optional grouping variables. |

## Value

A tabular R object containing interval-coverage table; rows represent analysis units and columns contain the returned quantities.

## validation_evidence_levels

*Detailed validation evidence levels*

### Description

Returns the fine-grained evidence ladder used by the 0.7 validation programme. This is deliberately orthogonal to the existing public support levels (declared / fixture-tested / empirically-validated), so existing compatibility claims remain stable.

### Usage

```
validation_evidence_levels()
```

### Value

A data frame ordered from weakest to strongest evidence.

## validation_evidence_matrix

*Create a model-by-evidence validation matrix*

### Description

Create a model-by-evidence validation matrix

### Usage

```
validation_evidence_matrix(...)
```

### Arguments

| | |
|---|---|
| ... | Named validation results, bundles, or arbitrary evidence objects. |

### Value

A tabular R object containing a model-by-evidence validation matrix; rows represent analysis units and columns contain the returned quantities.

## validation_failure_profile

*Failure profile for a validation programme*

### Description

Failure profile for a validation programme

### Usage

```
validation_failure_profile(x)
```

### Arguments

| | |
|---|---|
| x | Validation result. |

### Value

A data frame containing failure profile for a validation programme. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## validation_failure_summary

*Summarize convergence and execution failures*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
validation_failure_summary(x, by = c("model_family", "scenario_id"))
```

### Arguments

| | |
|---|---|
| x | Validation collection or job table. |
| by | Grouping columns. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## validation_failure_taxonomy *Classify common estimator failures without hiding the original message*

### Description

Classify common estimator failures without hiding the original message

### Usage

```
validation_failure_taxonomy(x)
```

### Arguments

| | |
|---|---|
| x | Error/condition/message vector. |

### Value

A data frame containing classify common estimator failures without hiding the original message. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

## validation_job_plan *Create a deterministic validation job plan*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
validation_job_plan(grid = NULL, replications = 100L, base_seed = 1L,
  model_family = "unspecified", plan_id = NULL, chunk_size = 1L, metadata = list())
```

### Arguments

| | |
|---|---|
| grid | Scenario grid or named list of factor levels. |
| replications | Replications per design cell. |
| base_seed | Base seed used for deterministic seed allocation. |
| model_family | Model-family label. |
| plan_id | Optional explicit plan identifier. |
| chunk_size | Number of jobs assigned to each chunk. |
| metadata | Arbitrary plan metadata. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`validation_ladder` *Build a measurement-to-generalization validation ladder*

---

## Description

Build a measurement-to-generalization validation ladder

## Usage

```
validation_ladder(
  acquisition_qc = "not_assessed",
  analytical_qc = "not_assessed",
  construct_check = "not_assessed",
  within_person = "not_assessed",
  held_out_person = "not_assessed",
  claim = "descriptive"
)
```

## Arguments

| | |
|---|---|
| `acquisition_qc, analytical_qc, construct_check, within_person, held_out_person` | Stage statuses: ‘pass‘, ‘warning‘, ‘fail‘, or ‘not_assessed‘. |
| `claim` | Claim type; use ‘generalizable‘ for out-of-person claims. |

## Value

A structured validation-ladder result.

---

`validation_mcse` *Monte Carlo standard errors for validation metrics*

---

## Description

Monte Carlo standard errors for validation metrics

## Usage

```
validation_mcse(results, metric = c("bias", "rmse", "coverage"))
```

### Arguments

| | |
|---|---|
| `results` | Validation or model results. |
| `metric` | Metric to calculate or audit. |

### Value

A data frame containing monte Carlo standard errors for validation metrics. Rows represent the analysis units and columns contain the identifiers, estimates, or diagnostics defined by the function.

---

## `validation_mcse_profile` *Estimate Monte Carlo uncertainty for validation summaries*

---

### Description

Estimate Monte Carlo uncertainty for validation summaries

### Usage

```
validation_mcse_profile(x, metric, by = character())
```

### Arguments

| | |
|---|---|
| `x` | Object to validate, summarize, verify, or otherwise process. |
| `metric` | Metric name or metric column. |
| `by` | Grouping variables or aggregation level. |

### Value

A tabular R object containing monte Carlo uncertainty for validation summaries; rows represent analysis units and columns contain the returned quantities.

---

## `validation_recovery_summary` *Summarize parameter recovery*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
validation_recovery_summary(x, by = character())
```

### Arguments

| | |
|---|---|
| x | Validation collection or estimates data frame. |
| by | Additional grouping columns. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

## validation_recovery_table

*Parameter-recovery table*

---

### Description

Parameter-recovery table

### Usage

```
validation_recovery_table(x, by = NULL)
```

### Arguments

| | |
|---|---|
| x | Validation result. |
| by | Optional grouping variables. |

### Value

A tabular R object containing parameter-recovery table; rows represent analysis units and columns contain the returned quantities.

---

## validation_replication_budget

*Compute a replication budget from a target MCSE*

---

### Description

Compute a replication budget from a target MCSE

### Usage

```
validation_replication_budget(
  pilot_sd,
  target_mcse,
  minimum = 20L,
  maximum = 10000L
)
```

### Arguments

| | |
|---|---|
| pilot_sd | Pilot estimate of the metric standard deviation. |
| target_mcse | Target Monte Carlo standard error. |
| minimum | Minimum permitted replication count. |
| maximum | Maximum permitted replication count. |

### Value

A numeric value or vector containing a replication budget from a target MCSE.

---

## validation_report *Render a conservative validation report*

---

### Description

Render a conservative validation report

### Usage

```
validation_report(x, include_session = TRUE)
```

### Arguments

| | |
|---|---|
| x | Validation bundle. |
| include_session | Include abbreviated session provenance. |

### Value

Character vector of report lines.

---

## validation_robustness_score *Overall validation robustness score*

---

### Description

Overall validation robustness score

### Usage

```
validation_robustness_score(x)
```

### Arguments

| | |
|---|---|
| x | Validation result. |

### Value

A single numeric robustness score: the mean finite condition-level robustness score, or 'NA_real_' when no finite score is available.

---

## validation_runtime_summary

*Summarize validation runtime and checkpoint scale*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
validation_runtime_summary(x, by = c("model_family", "scenario_id"))
```

### Arguments

| | |
|---|---|
| x | Validation collection or job table. |
| by | Grouping columns. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

## validation_sbc_summary

*Summarize simulation-based calibration ranks*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
validation_sbc_summary(x, by = c("model_family", "scenario_id"), bins = 10L)
```

### Arguments

| | |
|---|---|
| x | Validation collection or posterior draws data frame. |
| by | Grouping columns. |
| bins | Rank-histogram bins. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

## validation_scenario_manifest

*Create a scenario manifest for frozen validation work*

---

### Description

Create a scenario manifest for frozen validation work

### Usage

```
validation_scenario_manifest(
  plan,
  source_commit = NA_character_,
  generated_at = Sys.time()
)
```

### Arguments

| | |
|---|---|
| plan | Validation or stress-evidence plan object. |
| source_commit | Source-control commit associated with the evidence. |
| generated_at | Generation timestamp stored in the manifest. |

### Value

An object of class "eye_validation_scenario_manifest", stored as a named list, with components "label", "plan_hash", "scenarios", "source_commit", "generated_at", "scientific_scope". It contains a scenario manifest for frozen validation work and associated metadata or diagnostics needed to interpret the result.

## validation_seed *Allocate a deterministic validation seed*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
validation_seed(design, replication, base_seed = 1L, stream = 1L)
```

### Arguments

| | |
|---|---|
| design | A list or one-row data frame describing a design cell. |
| replication | Positive replication number. |
| base_seed | Base integer seed. |
| stream | Optional independent stream number. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## validation_summary_mcse *Monte Carlo standard-error diagnostics for validation summaries*

### Description

Monte Carlo standard-error diagnostics for validation summaries

### Usage

```
validation_summary_mcse(x, by = NULL)
```

### Arguments

| | |
|---|---|
| x | Validation result. |
| by | Optional grouping variables. |

### Value

A tabular R object containing monte Carlo standard-error diagnostics for validation summaries; rows represent analysis units and columns contain the returned quantities.

validation_thresholds *Specify completion and scientific-promotion thresholds*

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
validation_thresholds(required_replications = 100L, max_failure_rate = 0.05,
 max_absolute_bias = 0.10, max_rmse = Inf, min_coverage = 0.90, max_coverage = 0.99,
 max_rhat = 1.01, min_ess_bulk = 400, max_divergence_rate = 0.01, require_sbc = TRUE,
  require_empirical_reproduction = TRUE)
```

## Arguments

| | |
|---|---|
| required_replications | Required completed replications per scenario. |
| max_failure_rate | Maximum tolerated job failure rate. |
| max_absolute_bias | Maximum absolute mean bias. |
| max_rmse | Maximum RMSE. |
| min_coverage | Minimum interval coverage. |
| max_coverage | Maximum interval coverage. |
| max_rhat | Maximum acceptable R-hat. |
| min_ess_bulk | Minimum bulk effective sample size. |
| max_divergence_rate | Maximum divergence rate. |
| require_sbc | Require passing SBC uniformity evidence. |
| require_empirical_reproduction | Require empirical-reproduction evidence. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

vendor_schema_contract *Declare a vendor semantic schema contract*

## Description

Declare a vendor semantic schema contract

## Usage

```
vendor_schema_contract(
  vendor,
  version = NA_character_,
  required_fields = character(),
  optional_fields = character(),
  aliases = list(),
  timestamp = list(),
  coordinate = list(),
  units = list(),
  eye_streams = character(),
  event_fields = character()
)
```

## Arguments

| | |
|---|---|
| vendor | Vendor/ecosystem label. |
| version | Optional format/software version. |
| required_fields | Fields that must survive import. |
| optional_fields | Fields that may be present. |
| aliases | Named list mapping canonical fields to accepted vendor names. |
| timestamp | Named list describing device/system/media time columns. |
| coordinate | Named list describing x/y columns and coordinate semantics. |
| units | Named list of expected units for canonical fields. |
| eye_streams | Expected eye streams (‘left‘, ‘right‘, ‘cyclopean‘, etc.). |
| event_fields | Event/annotation fields expected to survive. |

## Value

An object of class "eye_vendor_schema_contract", stored as a named list, with components "vendor", "version", "required_fields", "optional_fields", "aliases", "timestamp", "coordinate", "units", "eye_streams", "event_fields", "contract_version". It contains declare a vendor semantic schema contract and associated metadata or diagnostics needed to interpret the result.

`vendor_validation_spec` *Specify multi-vendor empirical validation requirements*

## Description

Specify multi-vendor empirical validation requirements

## Usage

```
vendor_validation_spec(
  required_vendors = c("gazepoint", "tobii", "pupillabs", "eyelink", "smi"),
  min_cases_per_vendor = 2L,
  min_pass_rate = 0.95,
  require_versions = TRUE,
  require_devices = TRUE,
  require_independent_sources = TRUE,
  require_licence_reviewed = TRUE
)
```

## Arguments

| | |
|---|---|
| `required_vendors` | Vendors that must be represented. |
| `min_cases_per_vendor` | Minimum independent cases per vendor. |
| `min_pass_rate` | Minimum acceptable pass rate per vendor. |
| `require_versions` | Require non-missing software versions. |
| `require_devices` | Require non-missing device models. |
| `require_independent_sources` | Require each case to be marked as an independently obtained source rather than a duplicated fixture. |
| `require_licence_reviewed` | Require each corpus case to have a completed data/code licence and redistribution review. |

## Value

An ‘eye_vendor_validation_spec‘.

## verify_decision_manifest_lock

*Verify that a locked manifest has not changed*

### Description

Verify that a locked manifest has not changed

### Usage

```
verify_decision_manifest_lock(x)
```

### Arguments

| | |
|---|---|
| x | Manifest lock. |

### Value

A logical value or vector indicating verify that a locked manifest has not changed.

## verify_eyeprocess_validation_atlas

*Verify a frozen validation atlas*

### Description

Verify a frozen validation atlas

### Usage

```
verify_eyeprocess_validation_atlas(x)
```

### Arguments

| | |
|---|---|
| x | Object to validate, summarize, verify, or otherwise process. |

### Value

A logical value or vector indicating verify a frozen validation atlas.

`verify_eyeprocess_validation_evidence` *Verify the integrity hash of a frozen evidence bundle*

## Description

Verify the integrity hash of a frozen evidence bundle

## Usage

```
verify_eyeprocess_validation_evidence(x)
```

## Arguments

| | |
|---|---|
| `x` | Object to validate, summarize, verify, or otherwise process. |

## Value

A logical value or vector indicating verify the integrity hash of a frozen evidence bundle.

`verify_outcome_blind_snapshot` *Verify an outcome-blind snapshot has not changed*

## Description

Verify an outcome-blind snapshot has not changed

## Usage

```
verify_outcome_blind_snapshot(x)
```

## Arguments

| | |
|---|---|
| `x` | Snapshot. |

## Value

A logical value or vector indicating verify an outcome-blind snapshot has not changed.

## verify_reproducibility_fingerprint

*Verify an internally stored fingerprint hash*

### Description

Verify an internally stored fingerprint hash

### Usage

```
verify_reproducibility_fingerprint(x)
```

### Arguments

| | |
|---|---|
| x | Fingerprint. |

### Value

A logical value or vector indicating verify an internally stored fingerprint hash.

## verify_reproducibility_manifest

*Verify a reproducibility manifest*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
verify_reproducibility_manifest(manifest)
```

### Arguments

| | |
|---|---|
| manifest | Manifest. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## visual_context_registry

*Build an item-to-visual-context registry*

### Description

Build an item-to-visual-context registry

### Usage

```
visual_context_registry(
  item_metadata,
  item = "item_id",
  context = NULL,
 context_candidates = c("visual_anchor_id", "stimulus_id", "stimulus_page", "page_id",
    "layout_id", "screen_id", "diagram_id"),
  min_items_per_context = 3L
)
```

### Arguments

| | |
|---|---|
| item_metadata | Item-level metadata. |
| item | Item identifier column. |
| context | Optional explicit context column. |
| context_candidates | Candidate metadata columns searched in order. |
| min_items_per_context | Minimum items required for a shared context. |

### Value

An object of class "eye_visual_context_registry", stored as a named list, with components "mapping", "source_item_column", "source_context_column", "min_items_per_context", "caveat". It contains an item-to-visual-context registry and associated metadata or diagnostics needed to interpret the result.

## write_advanced_model_evidence_report

*Write an advanced-model evidence report*

### Description

Write an advanced-model evidence report

## Usage

```
write_advanced_model_evidence_report(x, path)
```

## Arguments

| | |
|---|---|
| x | An ‘eye_advanced_evidence_audit‘ object. |
| path | Markdown output file. |

## Value

Normalized report path.

---

`write_api_lifecycle_registry` *Write API lifecycle registry to CSV*

---

## Description

Write API lifecycle registry to CSV

## Usage

```
write_api_lifecycle_registry(registry, path)
```

## Arguments

| | |
|---|---|
| registry | Lifecycle registry. |
| path | Output path. |

## Value

An R object containing aPI lifecycle registry to CSV. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## write_benchmark_data_dictionary

*Write the benchmark data dictionary*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
write_benchmark_data_dictionary(study = eyeprocess_benchmark_study(),
  path = "benchmark-data-dictionary.md")
```

### Arguments

| | |
|---|---|
| study | Benchmark object or path. |
| path | Output Markdown file. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## write_decision_manifest

*Write a decision manifest*

### Description

Write a decision manifest

### Usage

```
write_decision_manifest(x, path, format = c("rds", "dput", "json"))
```

### Arguments

| | |
|---|---|
| x | Manifest. |
| path | Output path. |
| format | ‘rds‘, ‘dput‘, or ‘json‘. |

### Value

An R object containing a decision manifest. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## write_eyeprocess_validation_evidence *Write a frozen evidence bundle*

### Description

Write a frozen evidence bundle

### Usage

```
write_eyeprocess_validation_evidence(x, path)
```

### Arguments

| | |
|---|---|
| x | Object to validate, summarize, verify, or otherwise process. |
| path | File path for reading or writing. |

### Value

A character string or vector giving the path or identifier for a frozen evidence bundle.

## write_eyeprocess_validation_report *Write a compact Markdown validation report*

### Description

Write a compact Markdown validation report

### Usage

```
write_eyeprocess_validation_report(
  atlas,
  path,
  title = "eyeprocess validation evidence report"
)
```

### Arguments

| | |
|---|---|
| atlas | Validation-evidence atlas object. |
| path | File path for reading or writing. |
| title | Report title. |

### Value

A character string or vector giving the path or identifier for a compact Markdown validation report.

## write_eye_pipeline_report *Write a conservative pipeline report*

### Description

Write a conservative pipeline report

### Usage

```
write_eye_pipeline_report(x, path)
```

### Arguments

| | |
|---|---|
| x | Pipeline or run. |
| path | Output text/markdown path. |

### Value

A character string or vector giving the path or identifier for a conservative pipeline report.

## write_eye_storage *Write an eye dataset to RDS or Arrow/Parquet storage*

### Description

Write an eye dataset to RDS or Arrow/Parquet storage

### Usage

```
write_eye_storage(
  x,
  path,
  format = c("rds", "parquet", "arrow_dataset"),
  tables = canonical_table_names(),
  partitioning = NULL,
  compression = "zstd",
  overwrite = FALSE,
  retain_metadata = TRUE
)
```

### Arguments

| | |
|---|---|
| x | An ‘eye_dataset‘. |
| path | Output path. |
| format | Storage format. |
| tables | Canonical tables to write. |
| partitioning | Optional partition columns for Arrow datasets. |
| compression | Parquet compression codec. For writes, the default ‘"zstd"‘ is preferred; when the argument is omitted and that codec is unavailable, storage falls back to ‘"snappy"‘ and then ‘"uncompressed"‘. An explicitly requested unavailable codec errors. |
| overwrite | Whether to replace an existing target. |
| retain_metadata | Whether to retain raw/vendor metadata in a sidecar RDS. |

### Value

An ‘eye_storage‘ handle.

---

## write_eye_targets_template *Write an explicit ‘_targets.R‘ template from a governed pipeline*

---

### Description

Write an explicit ‘_targets.R‘ template from a governed pipeline

### Usage

```
write_eye_targets_template(x, path = "_targets.R")
```

### Arguments

| | |
|---|---|
| x | Pipeline. |
| path | Output path. |

### Value

An R object containing an explicit ‘_targets.R‘ template from a governed pipeline. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## write_gazepoint_workflow_report

*Write a reproducible Gazepoint workflow report*

### Description

Write a reproducible Gazepoint workflow report

### Usage

```
write_gazepoint_workflow_report(
  workflow,
  path = file.path(workflow$output_dir, "gazepoint-workflow-report.md"),
  render_html = workflow$spec$create_html_report
)
```

### Arguments

| | |
|---|---|
| workflow | An 'eye_gazepoint_workflow' result. |
| path | Markdown report destination. |
| render_html | Render an HTML copy when possible. |

### Value

The normalized report path.

## write_model_promotion_report

*Write a model-promotion report*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
write_model_promotion_report(x, path)
```

### Arguments

| | |
|---|---|
| x | Model-promotion audit. |
| path | Markdown output file. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## write_partitioned_eye_storage *Write an eye dataset as atomic partitioned storage*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
write_partitioned_eye_storage(x, path,
  spec = partition_eye_storage(format = if (requireNamespace("arrow",
  quietly = TRUE)) "parquet" else "rds"), overwrite = FALSE, tables = NULL)
```

### Arguments

| | |
|---|---|
| x | Eye dataset or named list of data frames. |
| path | Destination directory. |
| spec | Partition specification. |
| overwrite | Whether to replace an existing store. |
| tables | Tables to write. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

## write_prov_dot *Return Graphviz DOT for a provenance graph*

### Description

Return Graphviz DOT for a provenance graph

### Usage

```
write_prov_dot(x)
```

### Arguments

| | |
|---|---|
| x | Provenance graph. |

### Value

A character value or vector containing return Graphviz DOT for a provenance graph.

---

`write_reporting_guideline_report`
*Write a reporting-guideline audit report*

---

## Description

Write a reporting-guideline audit report

## Usage

```
write_reporting_guideline_report(x, path)
```

## Arguments

| | |
|---|---|
| `x` | An ‘eye_reporting_audit‘ object. |
| `path` | Markdown output file. |

## Value

Normalized report path.

---

`write_reproducibility_fingerprint`
*Write a reproducibility fingerprint*

---

## Description

Write a reproducibility fingerprint

## Usage

```
write_reproducibility_fingerprint(x, path, format = c("rds", "dput", "json"))
```

## Arguments

| | |
|---|---|
| `x` | Fingerprint. |
| `path` | Output path. |
| `format` | ‘rds‘, ‘dput‘, or ‘json‘. |

## Value

A character string or vector giving the path or identifier for a reproducibility fingerprint.

## write_software_paper_evidence

*Write a human-readable software-paper evidence report*

### Description

Write a human-readable software-paper evidence report

### Usage

```
write_software_paper_evidence(x, path)
```

### Arguments

| | |
|---|---|
| x | Evidence bundle. |
| path | Markdown output path. |

### Value

An R object containing a human-readable software-paper evidence report. The concrete class and structure follow the selected method, engine, or input object and are preserved as documented by that workflow.

## write_software_paper_reproduction

*Write a complete software-paper reproduction scaffold*

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
write_software_paper_reproduction(directory, study = eyeprocess_benchmark_study(),
  overwrite = FALSE)
```

### Arguments

| | |
|---|---|
| directory | Output directory. |
| study | Benchmark study. |
| overwrite | Whether to replace existing files. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

`write_software_paper_scaffold` *Write a methodological software-paper scaffold*

## Description

Write a methodological software-paper scaffold

## Usage

```
write_software_paper_scaffold(
  path,
  title = "eyeprocess: Reproducible Psychometric Process Modelling in R",
  author = "Stefanos Balaskas"
)
```

## Arguments

| | |
|---|---|
| `path` | Output R Markdown file. |
| `title` | Paper title. |
| `author` | Author string. |

## Value

The normalized path.

`write_validation_job_manifest` *Write a machine-readable validation manifest*

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
write_validation_job_manifest(plan, path, overwrite = FALSE)
```

## Arguments

| | |
|---|---|
| `plan` | An ‘eye_validation_job_plan‘. |
| `path` | Manifest directory. |
| `overwrite` | Replace an existing manifest directory. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`write_validation_release_report`

*Write a validation release report*

---

## Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

## Usage

```
write_validation_release_report(x, path, completion = NULL, promotion = NULL,
  title = "eyeprocess validation release report", include_session = TRUE)
```

## Arguments

| | |
|---|---|
| `x` | Validation collection. |
| `path` | Markdown output file. |
| `completion` | Optional completion audit. |
| `promotion` | Optional model-promotion audit. |
| `title` | Report title. |
| `include_session` | Include session information. |

## Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

`write_validation_report`

*Write a validation report to disk*

---

## Description

Write a validation report to disk

## Usage

```
write_validation_report(x, path, ...)
```

### Arguments

| | |
|---|---|
| x | Validation bundle. |
| path | Output text-file path. |
| ... | Passed to 'validation_report()'. |

### Value

A character string or vector giving the path or identifier for a validation report to disk.

---

## write_validation_scenario_manifest

*Write a validation scenario manifest*

---

### Description

Write a validation scenario manifest

### Usage

```
write_validation_scenario_manifest(x, path)
```

### Arguments

| | |
|---|---|
| x | Object to validate, summarize, verify, or otherwise process. |
| path | File path for reading or writing. |

### Value

A character string or vector giving the path or identifier for a validation scenario manifest.

---

## write_vendor_case_report

*Write a vendor case evidence report*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
write_vendor_case_report(corpus_path, case_id, path, validation = NULL,
  roundtrip = NULL)
```

### Arguments

| | |
|---|---|
| corpus_path | Corpus directory. |
| case_id | Case identifier. |
| path | Markdown output path. |
| validation | Optional validation result. |
| roundtrip | Optional round-trip audit. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

## write_vendor_registry *Write the multi-vendor case registry*

---

### Description

Part of the research-scale validation, advanced-model, interoperability, storage, adapter, or reproducibility programme. Experimental model functions remain subject to declared evidence gates.

### Usage

```
write_vendor_registry(x, corpus_path)
```

### Arguments

| | |
|---|---|
| x | Registry data frame. |
| corpus_path | Corpus directory. |

### Value

The documented eyeprocess object, data frame, plot, report path, or adapter result.

---

## write_vendor_validation_report *Write a multi-vendor validation report*

---

### Description

Write a multi-vendor validation report

### Usage

```
write_vendor_validation_report(x, path)
```

## Arguments

| | |
|---|---|
| x | An ‘eye_vendor_validation‘ object. |
| path | Markdown output file. |

## Value

The normalized report path.

# Index


∗ **data**
eyeprocess-format-validation, 153
∗ **methods**
assign_aois_probabilistic, 38
build_evidence_graph, 80
derive_aoi_composition, 112
detect_calibration_drift, 115
detect_process_changepoints, 118
eye_plot_spec, 243
eyeprocess-adapters, 149
eyeprocess-class, 150
eyeprocess-coordinates-time, 150
eyeprocess-experimental, 151
eyeprocess-export-report, 152
eyeprocess-features, 152
eyeprocess-format-validation, 153
eyeprocess-import-gazepoint, 154
eyeprocess-import-generic, 154
eyeprocess-import-vendors, 155
eyeprocess-models, 155
eyeprocess-plots, 156
eyeprocess-preprocessing, 157
eyeprocess-quality, 157
eyeprocess-schema, 158
eyeprocess-simulation, 158
eyeprocess-trials-aoi, 159
fit_device_linking, 259
fit_fixation_point_process, 269
fit_process_dif, 297
fit_process_gstudy, 299
fit_process_missingness_model, 301
fit_process_norms, 302
fit_process_observation_model, 304
gaze_recurrence, 333
item_objective_spec, 352
process_uncertainty_spec, 503
register_pupil_curves, 529
representative_scanpath, 534
∗ **package**
eyeprocess-package, 23

ablate_multimodal_channels, 23
add_multilevel_mediation_component *(multilevel-mediation)*, 364
add_provenance *(eyeprocess-class)*, 150
add_responses *(eyeprocess-trials-aoi)*, 159
addm_glam_proxy_features, 24
adjust_pupil_confounds, 25
advanced_model_evidence_spec, 25
advanced_validation_grid, 26
algorithm_facet_effects, 27
align_clock *(eyeprocess-coordinates-time)*, 150
align_response_matrices *(eyeprocess-models)*, 155
analysis_decision_entropy, 27
analysis_environment_snapshot, 28
analysis_readiness *(eyeprocess-quality)*, 157
analysis_resolution_guard, 28
anonymize_eye_dataset *(eyeprocess-format-validation)*, 153
aoi_balance_coordinates *(derive_aoi_composition)*, 112
aoi_membership_probability, 29
aoi_perturbation_spec *(aoi_perturbation_uncertainty)*, 30
aoi_perturbation_uncertainty, 30
aoi_trajectory_features, 34
api_family_map, 35
api_lifecycle_diff, 36
api_surface_summary, 36
append_eye_table *(eyeprocess-class)*, 150
apply_aoi_perturbation_grid *(aoi_perturbation_uncertainty)*,

30
apply_clock_transform (eyeprocess-coordinates-time), 150
apply_device_linking (fit_device_linking), 259
apply_offline_recalibration (detect_calibration_drift), 115
apply_preflight_decision, 37
apply_synthetic_corruption, 37
as_eye_biometrics (eyeprocess-export-report), 152
as_eye_dataset (eyeprocess-class), 150
as_eye_dataset.gp3_analysis (eyeprocess-export-report), 152
as_eye_dataset.gp3_data (eyeprocess-export-report), 152
as_eye_dataset.gp3_recording (eyeprocess-export-report), 152
as_eye_dataset.gpbiometrics_data (eyeprocess-export-report), 152
as_eyeprocess_eyeris (external_eye_adapters), 145
as_eyeprocess_eyetools (external_eye_adapters), 145
as_eyeprocess_eyetrackingr (external_eye_adapters), 145
as_eyeprocess_gazer (external_eye_adapters), 145
as_eyeprocess_pupillometryr (external_eye_adapters), 145
as_irt_recovery_results, 40
as_procdata_sequence (sequence_interoperability), 561
as_seqhmm_data (sequence_interoperability), 561
as_traminer_sequence (sequence_interoperability), 561
assess_aoi_inference_stability (aoi_perturbation_uncertainty), 30
assess_detector_inference_stability (event_detector_multiverse), 133
assign_aois, *136*
assign_aois (eyeprocess-trials-aoi), 159
assign_aois_probabilistic, 38
assign_process_feature_family, 40
assign_trials (eyeprocess-trials-aoi), 159
audit_3pl_process_signatures, 41
audit_advanced_model_evidence, 41
audit_aoi_separation (assign_aois_probabilistic), 38
audit_aois (eyeprocess-quality), 157
audit_bank_decision_stability (item_objective_spec), 352
audit_benchmark_release, 42
audit_bias, 42
audit_biometric_imputation, 43
audit_biometric_preflight, 43
audit_candidate_item_bank, 44
audit_channel_incremental_information, 45
audit_clock_sync (eyeprocess-coordinates-time), 150
audit_convergence, 46
audit_coordinate_spaces (eyeprocess-coordinates-time), 150
audit_coverage, 47
audit_decision_provenance, 47
audit_device_equivalence (fit_device_linking), 259
audit_distractor_attention, 48
audit_episodes (eyeprocess-quality), 157
audit_event_order (eyeprocess-quality), 157
audit_evidence_dependencies (build_evidence_graph), 80
audit_eye_api, 49
audit_eye_pipeline, 49
audit_fairness_transportability (fit_process_dif), 297
audit_frontier_model_contract, 50
audit_identifiability, 50
audit_interval_width, 51
audit_irf_shape, 52
audit_item_reduction_sensitivity, 52
audit_latent_distribution, 53
audit_measurement_transportability, 54
audit_mediation_missingness

(multilevel-mediation), 364
audit_missingness (eyeprocess-quality), 157
audit_model_promotion, 54
audit_multimodal_identifiability, 55
audit_multimodal_m2_identifiability, 56
audit_multimodal_m3_identifiability, 56
audit_multimodal_m4_identifiability, 57
audit_multimodal_measurement, 58
audit_multivariate_process_quality, 59
audit_nonparametric_rasch, 59
audit_norm_transportability (fit_process_norms), 302
audit_presentation_accessibility, 60
audit_process_adjusted_dif, 61
audit_process_anomalies, 62
audit_process_drift, 62
audit_process_external_validity, 63
audit_process_local_dependence, 64
audit_process_measurement_invariance, 65
audit_process_reliability (fit_process_gstudy), 299
audit_process_window_sensitivity, 65
audit_pupil_fatigue_drift, 66
audit_pupil_frequency_stability, 67
audit_pupil_preprocessing_order, 68
audit_pupil_quality (eyeprocess-quality), 157
audit_pupil_registration (register_pupil_curves), 529
audit_recalibration (detect_calibration_drift), 115
audit_rmse, 68
audit_roundtrip_loss, 69
audit_sampling_irregularity, 69
audit_sampling_rate (eyeprocess-quality), 157
audit_sbc, 70
audit_signal_filter, 71
audit_signal_quality (eyeprocess-quality), 157
audit_temporal_leakage, 71
audit_timebase (eyeprocess-coordinates-time), 150
audit_trial_coverage (eyeprocess-quality), 157
audit_validation_completion, 72
audit_vendor_field_coverage, 72
audit_vendor_validation, 73
audit_visual_context_dependence, 73
autoplot_eyeprocess (eye_plot_spec), 243

baseline_pupil (eyeprocess-preprocessing), 157
bayesian_process_diagnostic_flags, 74
bayesian_process_diagnostics_dashboard, 74
benchmark_expected_outputs, 75
benchmark_eye_storage, 76
benchmark_eyeprocess, 76
benchmark_memory_estimate, 77
benchmark_scaling_curve, 77
bind_process_windows, 78
biometric_imputation_sensitivity, 78
bootstrap_process_reliability, 79
bootstrap_representative_scanpath (representative_scanpath), 534
build_aoi_visits (eyeprocess-trials-aoi), 159
build_compatibility_matrix, 80
build_evidence_graph, 80
build_gazepoint_media_trials, 82
build_item_responses (eyeprocess-trials-aoi), 159
build_stimulus_intervals (eyeprocess-trials-aoi), 159
build_trials (eyeprocess-trials-aoi), 159

calibration_drift_profile, 82
calibration_error_model, 83
calibration_sensitivity_grid, 84
calibration_transfer_audit, 84
canonical_eye_api, 85
canonical_table_names (eyeprocess-schema), 158
center_within_participant (multilevel-mediation), 364
check_feature_level (eyeprocess-quality), 157
check_gaze_proportional_hazards (gaze-survival), 324

check_local_dependence
(eyeprocess-models), 155
check_mediation_trial_counts
(multilevel-mediation), 364
check_process_leakage
(eyeprocess-quality), 157
classify_item_missingness, 85
collect_eye_storage, 86
collect_validation_evidence, 86
collect_validation_jobs, 87
combine_eye_datasets
(eyeprocess-adapters), 149
compact_eye_dataset (eyeprocess-class),
150
compare_aoi_assignments
(aoi_perturbation_uncertainty),
30
compare_aoi_compositions
(derive_aoi_composition), 112
compare_aoi_definitions
(eyeprocess-quality), 157
compare_aoi_methods, 87
compare_aoi_trajectories, 88
compare_bayesian_process_models, 88
compare_decision_manifests, 89
compare_decision_provenance
(build_evidence_graph), 80
compare_deployment_batches, 89
compare_diffusion_accuracy_rt, 90
compare_dynamic_transition_models, 91
compare_engine_adapters, 91
compare_episode_structure
(detect_process_changepoints),
118
compare_event_catalogues
(event_detector_multiverse),
133
compare_eye_datasets
(eyeprocess-format-validation),
153
compare_fixation_methods, 92
compare_functional_scalar_models, 92
compare_gaze_quality_conditions
(standardized_spatial_quality),
584
compare_gaze_quality_sessions
(standardized_spatial_quality),
584
compare_gaze_survival_models
(gaze-survival), 324
compare_gaze_survival_specifications
(gaze-survival), 324
compare_hard_probabilistic_aoi, 93
compare_irt_models, 94
compare_latent_distribution_models, 94
compare_model_engines, 95
compare_parametric_nonparametric_irf,
95
compare_preprocessing
(eyeprocess-quality), 157
compare_presentation_fairness, 96
compare_process_criterion_models, 97
compare_process_models, 97
compare_process_profile_solutions, 98
compare_pupil_kernels, 98
compare_pupil_preprocessing, 99
compare_raw_adjusted_pupil, 99
compare_reproducibility_fingerprints,
100
compare_scanpath_distributions
(representative_scanpath), 534
compare_signal_filters, 100
compare_strategy_heterogeneity, 101
compare_uncertainty_budgets
(process_uncertainty_spec), 503
compare_validation_engines, 101
compare_vendor_semantics, 102
compare_visual_context_irt, 102
compatibility_evidence_matrix, 103
compute_bcea
(standardized_spatial_quality),
584
compute_gaze_accuracy
(standardized_spatial_quality),
584
compute_gaze_data_loss
(standardized_spatial_quality),
584
compute_gaze_precision
(standardized_spatial_quality),
584
compute_gaze_sd_precision
(standardized_spatial_quality),
584
compute_rms_s2s
(standardized_spatial_quality),

584
compute_valid_sample_fraction (standardized_spatial_quality), 584
context_factor_effects, 103
convert_aoi_margin_to_degrees (aoi_perturbation_uncertainty), 30
convert_aoi_margin_to_pixels (aoi_perturbation_uncertainty), 30
convert_coordinates (eyeprocess-coordinates-time), 150
convert_xy (eyeprocess-coordinates-time), 150
coordinate_fidelity_audit, 104
coordinate_space (eyeprocess-coordinates-time), 150
coverage_calibration_curve, 105
create_aoi_perturbation_grid (aoi_perturbation_uncertainty), 30
create_detector_multiverse (event_detector_multiverse), 133
create_gaze_quality_report (standardized_spatial_quality), 584
create_public_benchmark, 105
create_validation_bundle (eyeprocess-format-validation), 153
cross_device_process_equating_audit, 107
cross_recurrence (gaze_recurrence), 333
cross_version_adapter_regression, 108
crossed_grouped_cv, 106
crossed_grouped_folds, 107
crossmodal_recurrence_model (fit_process_missingness_model), 301

data_quality_reporting_table, 109
decision_manifest_diff, 109
decision_manifest_hash, 110
decision_manifest_table, 110
decision_space_coverage, 111
decision_stability, 111
decode_dynamic_states, 112
decompose_dif_evidence (fit_process_dif), 297
decompose_pupil_phase_amplitude (register_pupil_curves), 529
decompose_within_between (multilevel-mediation), 364
define_event_detector_spec (event_detector_multiverse), 133
derive_all_features (eyeprocess-features), 152
derive_aoi_composition, 112
derive_biometric_features (eyeprocess-features), 152
derive_gaze_features (eyeprocess-features), 152
derive_gazepoint_workflow_features, 114
derive_pupil_features (eyeprocess-features), 152
derive_rt_features (eyeprocess-features), 152
design_process_dstudy (fit_process_gstudy), 299
detect_blinks (eyeprocess-preprocessing), 157
detect_calibration_drift, 115
detect_corrupt_partitions, 116
detect_events_with_spec (event_detector_multiverse), 133
detect_eye_format (eyeprocess-adapters), 149
detect_fixations_idt, *136*
detect_fixations_idt (eyeprocess-preprocessing), 157
detect_fixations_ivt, *136*
detect_fixations_ivt (eyeprocess-preprocessing), 157
detect_irt_changepoints, 117
detect_process_changepoint, 118
detect_process_changepoints, 118
detect_saccades (eyeprocess-preprocessing), 157
device_facet_effects, 120

diagnose_gaze_point_process
(fit_fixation_point_process), 269
diffusion_identification_study, 120
diffusion_parameter_diagnostics, 121
diffusion_posterior_predictive, 122
dilate_aoi
(aoi_perturbation_uncertainty), 30
discover_validation_cases
(eyeprocess-format-validation), 153
distractor_process_map, 122
drift_by_device, 123
drift_by_site, 123
drift_by_stimulus_version, 124
drift_by_vendor, 124
dynamic_irtree_recovery, 125
dynamic_irtree_spec, 125
dynamic_posterior_predictive_check, 127
dynamic_transition_design, 128

effective_sampling_frequency, 128
empty_eye_table (eyeprocess-schema), 158
encode_response_combinations, 129
engine_adapter_status, 130
equate_irt_scales, 130
erode_aoi
(aoi_perturbation_uncertainty), 30
estimate_aoi_assignment_stability
(aoi_perturbation_uncertainty), 30
estimate_calibration_error, 131
estimate_clock_transform
(eyeprocess-coordinates-time), 150
estimate_detector_agreement
(event_detector_multiverse), 133
estimate_device_specific_error
(fit_device_linking), 259
estimate_effective_sampling_rate
(standardized_spatial_quality), 584
estimate_ez_diffusion
(eyeprocess-experimental), 151
estimate_fixation_assignment_probability
(aoi_perturbation_uncertainty), 30
estimate_gaze_latency_quantiles
(gaze-survival), 324
estimate_gaze_survival (gaze-survival), 324
estimate_process_uncertainty
(process_uncertainty_spec), 503
estimate_sampling_interval
(standardized_spatial_quality), 584
estimate_sampling_jitter
(standardized_spatial_quality), 584
estimate_sampling_rate
(eyeprocess-coordinates-time), 150
estimate_visual_exposure_probability, 132
evaluate_validation_acceptance, 132
event_detector_multiverse, 133
event_marker_qc, 137
event_roundtrip_audit, 138
event_semantics_audit, 138
expand_eyeprocess_stress_evidence_plan, 139
expand_eyeprocess_validation_plan, 140
expand_process_validation_design, 140
expected_process_information, 141
explain_latent_interaction, 141
export_canonical
(eyeprocess-export-report), 152
export_eye_bids, 142
export_eye_pipeline, 143
export_prov_json, 143
export_ro_crate_metadata, 144
export_validation_bundle, 144
external_eye_adapters, 145
external_model_engines, 146
external_validate_irt, 146
extract_diffusion_parameters, 147
extract_functional_pupil_parameters, 147
extract_parameter_truth, 148
extract_process_windows, 148
eye_analysis_pipeline, 235
eye_analysis_spec, 235

eye_api_inventory, 236
eye_api_lifecycle, 237
eye_api_recommendation, 237
eye_api_status, 238
eye_api_superseded, 238
eye_benchmark_design, 239
eye_decision_manifest, 239
eye_format_profiles *(*eyeprocess-format-validation*)*, 153
eye_mapping *(*eyeprocess-adapters*)*, 149
eye_pipeline_dot, 240
eye_pipeline_graph, 241
eye_pipeline_manifest, 241
eye_pipeline_mermaid, 242
eye_pipeline_step, 242
eye_plot_spec, 243
eye_prov_graph, 244
eye_reproducibility_fingerprint, 245
eye_schema *(*eyeprocess-schema*)*, 158
eye_session_manifest, 246
eye_storage_spec, 246
eye_stream_fidelity_audit, 247
eye_targets_manifest, 248
eyeprocess *(*eyeprocess-package*)*, 23
eyeprocess-adapters, 149
eyeprocess-class, 150
eyeprocess-coordinates-time, 150
eyeprocess-experimental, 151
eyeprocess-export-report, 152
eyeprocess-features, 152
eyeprocess-format-validation, 153
eyeprocess-import-gazepoint, 154
eyeprocess-import-generic, 154
eyeprocess-import-vendors, 155
eyeprocess-models, 155
eyeprocess-package, 23
eyeprocess-plots, 156
eyeprocess-preprocessing, 157
eyeprocess-quality, 157
eyeprocess-schema, 158
eyeprocess-simulation, 158
eyeprocess-trials-aoi, 159
eyeprocess_api_version, 159
eyeprocess_benchmark_study, 160
eyeprocess_cdm_attribute_profiles, 160
eyeprocess_cdm_classification_uncertainty, 161
eyeprocess_cdm_dina_ideal_response, 161
eyeprocess_cdm_dina_probability, 162
eyeprocess_cdm_qmatrix_audit, 162
eyeprocess_deprecation, 163
eyeprocess_irt_2pl_probability, 163
eyeprocess_irt_3pl_probability, 164
eyeprocess_irt_4pl_probability, 165
eyeprocess_irt_adaptive_trace, 165
eyeprocess_irt_anchor_audit, 166
eyeprocess_irt_anchor_purification, 167
eyeprocess_irt_apply_link, 167
eyeprocess_irt_bank_coverage, 168
eyeprocess_irt_category_function_audit, 169
eyeprocess_irt_classification_precision, 169
eyeprocess_irt_conditional_sem, 170
eyeprocess_irt_content_balance_audit, 170
eyeprocess_irt_device_drift, 171
eyeprocess_irt_dif_effect_curve, 171
eyeprocess_irt_dtf_curve, 172
eyeprocess_irt_eap_score, 173
eyeprocess_irt_engine_evidence_table, 173
eyeprocess_irt_engine_registry, 174
eyeprocess_irt_engine_status, 174
eyeprocess_irt_expected_score, 175
eyeprocess_irt_exposure_summary, 175
eyeprocess_irt_extreme_score_audit, 176
eyeprocess_irt_fit_dashboard, 176
eyeprocess_irt_functioning_effect_summary, 177
eyeprocess_irt_gpcm_probability, 178
eyeprocess_irt_grm_probability, 178
eyeprocess_irt_haebara_link, 179
eyeprocess_irt_identification_audit, 179
eyeprocess_irt_infit_outfit, 180
eyeprocess_irt_information_area, 181
eyeprocess_irt_information_gain, 181
eyeprocess_irt_information_targeting, 182
eyeprocess_irt_invariance_evidence, 182

eyeprocess_irt_item_bank, 183
eyeprocess_irt_item_fit_residuals, 184
eyeprocess_irt_item_information, 184
eyeprocess_irt_item_selection, 185
eyeprocess_irt_latent_regression_design, 186
eyeprocess_irt_link_stability, 186
eyeprocess_irt_local_dependence_pairs, 187
eyeprocess_irt_map_score, 188
eyeprocess_irt_marginal_reliability, 188
eyeprocess_irt_mean_mean_link, 189
eyeprocess_irt_mean_sigma_link, 190
eyeprocess_irt_measurement_precision_profile, 190
eyeprocess_irt_missing_by_design_audit, 191
eyeprocess_irt_misspecification_metrics, 192
eyeprocess_irt_misspecification_suite, 192
eyeprocess_irt_mle_score, 193
eyeprocess_irt_model_card, 193
eyeprocess_irt_model_card_audit, 194
eyeprocess_irt_model_spec, 195
eyeprocess_irt_monotonicity_audit, 196
eyeprocess_irt_nominal_probability, 196
eyeprocess_irt_parameter_plausibility_audit, 197
eyeprocess_irt_person_fit_lz, 197
eyeprocess_irt_person_fit_residuals, 198
eyeprocess_irt_plausible_values, 199
eyeprocess_irt_ppc_discrepancy, 199
eyeprocess_irt_precision_evidence_table, 200
eyeprocess_irt_prior_sensitivity_grid, 200
eyeprocess_irt_prior_sensitivity_summary, 201
eyeprocess_irt_prior_spec, 202
eyeprocess_irt_process_alignment, 203
eyeprocess_irt_process_aware_selection_penalty, 203
eyeprocess_irt_process_dif_concordance, 204
eyeprocess_irt_q3, 205
eyeprocess_irt_recovery_design, 205
eyeprocess_irt_recovery_failures, 206
eyeprocess_irt_recovery_summary, 207
eyeprocess_irt_sbc_ranks, 207
eyeprocess_irt_sbc_summary, 208
eyeprocess_irt_score_table, 208
eyeprocess_irt_score_uncertainty, 209
eyeprocess_irt_session_drift, 210
eyeprocess_irt_sparse_design_audit, 210
eyeprocess_irt_stocking_lord_link, 211
eyeprocess_irt_stopping_rule, 212
eyeprocess_irt_targeting_gap, 213
eyeprocess_irt_test_characteristic_curve, 214
eyeprocess_irt_test_information, 215
eyeprocess_irt_testlet_audit, 213
eyeprocess_irt_testlet_spec, 214
eyeprocess_irt_threshold_order_audit, 215
eyeprocess_joint_process_irt_spec, 216
eyeprocess_mirt_directional_information, 217
eyeprocess_mirt_information_matrix, 217
eyeprocess_mirt_loading_audit, 218
eyeprocess_mirt_loading_spec, 219
eyeprocess_multichannel_measurement_map, 219
eyeprocess_negative_control_evidence_plan, 220
eyeprocess_negative_control_evidence_table, 221
eyeprocess_process_irt_data_bundle, 221
eyeprocess_process_item_profile, 222
eyeprocess_process_missingness_pattern, 223
eyeprocess_process_person_profile, 223
eyeprocess_recovery_evidence_table, 224
eyeprocess_reliability_evidence_plan, 224
eyeprocess_reliability_evidence_table, 225
eyeprocess_response_time_profile, 225
eyeprocess_sbc_evidence_table, 226

eyeprocess_speed_accuracy_profile, 226
eyeprocess_stress_evidence_plan, 227
eyeprocess_stress_evidence_table, 228
eyeprocess_validation_atlas_gaps, 228
eyeprocess_validation_claim_matrix, 229
eyeprocess_validation_evidence_atlas, 229
eyeprocess_validation_evidence_grade, 230
eyeprocess_validation_evidence_index, 231
eyeprocess_validation_evidence_manifest, 231
eyeprocess_validation_plan, 232
eyeprocess_validation_readiness, 233
eyeprocess_validation_release_gate, 234
eyeprocess_validation_seed, 234

facet_effects, 249
feature_dictionary (eyeprocess-features), 152
feature_spec (eyeprocess-features), 152
features_wide (eyeprocess-features), 152
field_fidelity_report, 249
file_hash_manifest, 250
filter_eye_signal, 251
filter_gaze (eyeprocess-preprocessing), 157
filter_pupil (eyeprocess-preprocessing), 157
filter_pupil_signal, 251
find_process_measures, 252
fingerprint_eye_dataset (eyeprocess-format-validation), 153
fingerprint_validation_case, 252
fit_accuracy_rt (eyeprocess-models), 155
fit_aoi_compositional_model (derive_aoi_composition), 112
fit_aoi_growth_curve, 253
fit_brms_adapter, 254
fit_censored_normal_process_irt, 254
fit_changepoint_multimodal_irt, 255
fit_changepoint_rt_irt, 256
fit_cognitive_diagnosis_process, 256
fit_continuous_time_irt, 257
fit_crossclassified_process_irt, 258
fit_crossclassified_process_irt_mhrm, 259
fit_device_linking, 259
fit_dif (eyeprocess-models), 155
fit_diffirt_adapter, 261
fit_diffirt_engine_adapter, 261
fit_dynamic_aoi_model (eyeprocess-models), 155
fit_dynamic_gpirt, 262
fit_dynamic_irtree, 262
fit_dynamic_irtree_stan, 263
fit_event_time_irt, 263
fit_explanatory_irt (eyeprocess-models), 155
fit_external_engine, 264
fit_eyeprocess_erm, 265
fit_eyeprocess_gdina, 266
fit_eyeprocess_lnirt, 266
fit_eyeprocess_mirt, 267
fit_eyeprocess_tam, 267
fit_eyetrackingr_adapter, 268
fit_fixation_point_process, 269
fit_flow_mirt, 270
fit_functional_pupil_stan, 271
fit_gaze_aft_model (gaze-survival), 324
fit_gaze_anchored_3pl_audit, 271
fit_gaze_cox_model (gaze-survival), 324
fit_gaze_diffusion_irt, 272
fit_gaze_diffusion_stan, 273
fit_gaze_informed_irt (eyeprocess-experimental), 151
fit_gaze_informed_missingness_irt, 273
fit_gaze_mixed_cox_model (gaze-survival), 324
fit_gaze_weighted_choice (eyeprocess-experimental), 151
fit_gdina_adapter, 274
fit_gpirt, 275
fit_irt (eyeprocess-models), 155
fit_irt_model, 276
fit_item_parameter_seed_model, 276
fit_joint_functional_pupil_irt, 277
fit_joint_gaze_rt_irt, 278
fit_joint_graded_rt_process_irt, 279
fit_joint_process_model (eyeprocess-models), 155
fit_joint_signal_missingness (fit_process_observation_model),

304
fit_kde_latent_distribution_irt, 280
fit_latent_class_process_irt, 280
fit_latent_space_irt, 281
fit_lnirt_adapter, 282
fit_manyfacet_process_irt, 283
fit_marked_gaze_process (fit_fixation_point_process), 269
fit_mirt_adapter, 284
fit_mixture_irt_process_classes, 284
fit_multiblock_process_map, 285
fit_multimodal_irt (eyeprocess-experimental), 151
fit_multimodal_m2, 286
fit_multimodal_m3, 287
fit_multimodal_m4, 288
fit_multimodal_trait_irt, 290
fit_multinomial_transition, 291
fit_multiple_response_process_irt, 292
fit_nominal_gaze_irt, 293
fit_nonignorable_missing_irt, 294
fit_offline_recalibration (detect_calibration_drift), 115
fit_omission_survival_irt, 294
fit_openmx_adapter, 295
fit_openmx_process_model, 296
fit_persistence_gaze_diffusion_irt, 297
fit_phase_amplitude_irt (register_pupil_curves), 529
fit_process_dif, 297
fit_process_gstudy, 299
fit_process_hmm_irt, 300
fit_process_irt (eyeprocess-experimental), 151
fit_process_missingness_model, 301
fit_process_norms, 302
fit_process_observation_model, 304
fit_process_profile_mixture, 305
fit_process_rasch_tree, 306
fit_pupil_confound_model, 307
fit_pupil_event_deconvolution, 308
fit_pupil_informed_irt (eyeprocess-experimental), 151
fit_pupillometryr_adapter, 306
fit_response_process_embedding_irt, 309
fit_revisit_process_cdm, 310
fit_seqhmm_adapter, 311
fit_shared_process_factor (eyeprocess-models), 155
fit_speed_accuracy_engagement_irt, 311
fit_strategy_mixture (eyeprocess-experimental), 151
fit_strategy_mixture_em, 312
fit_strategy_mixture_stan, 312
fit_tam_adapter, 313
fit_theory_strategy_irt, 314
fit_traminer_adapter, 314
fit_validation_replicate, 315
fit_variational_irt, 316
fit_visual_context_irt, 316
fixation_boundary_uncertainty, 317
flag_gaze_outliers (eyeprocess-preprocessing), 157
format_compatibility_matrix (eyeprocess-format-validation), 153
format_validation_spec (eyeprocess-format-validation), 153
freeze_eyeprocess_irt_reference, 318
freeze_eyeprocess_validation_atlas, 318
freeze_eyeprocess_validation_evidence, 319
freeze_software_paper_evidence, 320
freeze_validation_reference, 320
functional_pupil_basis, 321
functional_pupil_diagnostics, 322
functional_pupil_features (eyeprocess-experimental), 151
functional_pupil_irt_spec, 322

gaze-survival, 324
gaze_anchored_3pl_alignment, 330
gaze_data_quality_profile, 330
gaze_diffusion_spec, 331
gaze_entropy (eyeprocess-features), 152
gaze_precision_rms_s2s, 332
gaze_recurrence, 333
gaze_uncertainty_ellipse, 334
gaze_velocity (eyeprocess-preprocessing), 157
gazepoint_analysis_tables, 327
gazepoint_irt_tables, 328

gazepoint_workflow_spec, 328
generalizability_process_study, 335
get_eye_table *(eyeprocess-class)*, 150
get_irt_model, 335
gp_align_media_ids *(eyeprocess-import-gazepoint)*, 154
gp_audit_file_pairs *(eyeprocess-import-gazepoint)*, 154
gp_check_biometrics_sync *(eyeprocess-import-gazepoint)*, 154
gp_check_fixation_ids *(eyeprocess-import-gazepoint)*, 154
gp_check_media_timing *(eyeprocess-import-gazepoint)*, 154
gp_check_pupil_channels *(eyeprocess-import-gazepoint)*, 154
gp_check_sampling_rate *(eyeprocess-import-gazepoint)*, 154
gp_check_validity_fields *(eyeprocess-import-gazepoint)*, 154
gp_identify_export_type *(eyeprocess-import-gazepoint)*, 154
gp_list_export_fields *(eyeprocess-import-gazepoint)*, 154
gp_match_biometrics *(eyeprocess-import-gazepoint)*, 154
gp_match_recordings *(eyeprocess-import-gazepoint)*, 154
gp_pair_exports *(eyeprocess-import-gazepoint)*, 154
gp_parse_markers *(eyeprocess-import-gazepoint)*, 154
gp_parse_media_events *(eyeprocess-import-gazepoint)*, 154
gp_parse_user_events *(eyeprocess-import-gazepoint)*, 154
gp_profile_export *(eyeprocess-import-gazepoint)*, 154
gp_reconstruct_stimuli *(eyeprocess-import-gazepoint)*, 154
gp_reconstruct_trials *(eyeprocess-import-gazepoint)*, 154
gp_validate_export *(eyeprocess-import-gazepoint)*, 154
grade_model_evidence, 336
grouped_cv, 336
grouped_folds, 337

identify_mediation_levels *(multilevel-mediation)*, 364
import_benchmark_study, 338
import_canonical *(eyeprocess-export-report)*, 152
import_external_detector_events *(event_detector_multiverse)*, 133
import_eye_bids, 338
incremental_process_validity, 339
infer_eye_mapping *(eyeprocess-adapters)*, 149
init_validation_corpus *(eyeprocess-format-validation)*, 153
init_vendor_corpus, 339
inject_aoi_label_noise, 340
inject_calibration_offset, 340
inject_device_shift, 341
inject_eye_missingness, 342
inject_pupil_dropout, 342
inject_sampling_jitter, 343
inject_trial_imbalance, 343
inspect_eye_source *(eyeprocess-format-validation)*, 153
interpolate_pupil *(eyeprocess-preprocessing)*, 157

interpretive_warnings
(eyeprocess-quality), 157
irt_compositional_channel, 344
irt_continuous_channel, 344
irt_count_channel, 345
irt_functional_channel, 346
irt_model_spec, 346
irt_nominal_channel, 347
irt_response_channel, 348
irt_rt_channel, 349
irt_sequence_channel, 349
irt_survival_channel, 350
irt_validation_spec, 351
is_eye_dataset (eyeprocess-class), 150
is_eyelink_export
(eyeprocess-import-vendors), 155
is_gazepoint_export
(eyeprocess-import-gazepoint), 154
is_pupil_labs_export
(eyeprocess-import-vendors), 155
is_smi_export
(eyeprocess-import-vendors), 155
is_tobii_export
(eyeprocess-import-vendors), 155
item_objective_spec, 352
item_parameters (eyeprocess-models), 155
item_pareto_front
(item_objective_spec), 352

jitter_aoi
(aoi_perturbation_uncertainty), 30

label_process_episodes
(detect_process_changepoints), 118
latent_distribution_stress_test, 354
latent_trait_trajectory, 354
leave_device_out_validation, 355
leave_item_out_validation, 355
leave_session_out_validation, 356
leave_site_out_validation, 356
list_irt_models, 357
lock_decision_manifest, 357

map_latent_classes_to_process_profiles, 358
match_detected_events
(event_detector_multiverse), 133
measurement_error_budget, 359
migrate_eye_storage_schema, 359
model_data (eyeprocess-models), 155
model_fit_statistics
(eyeprocess-models), 155
model_missing_process
(eyeprocess-experimental), 151
model_promotion_spec, 360
model_validation_spec, 361
model_validation_summary, 362
monitor_dif_drift (fit_process_dif), 297
multiblock_contributions, 362
multiblock_person_coordinates, 363
multiblock_variable_coordinates, 363
multilevel-mediation, 364
multimodal_backend_status, 366
multimodal_irt_spec, 367
multimodal_m2_ablation, 368
multimodal_m2_negative_controls, 368
multimodal_m2_ppc, 369
multimodal_m2_process_information, 369
multimodal_m2_recovery, 370
multimodal_m2_spec, 371
multimodal_m3_ablation, 371
multimodal_m3_functional_bridge, 372
multimodal_m3_negative_controls, 373
multimodal_m3_ppc, 373
multimodal_m3_process_information, 374
multimodal_m3_recovery, 374
multimodal_m3_spec, 375
multimodal_m4_ablation, 376
multimodal_m4_negative_controls, 377
multimodal_m4_ppc, 377
multimodal_m4_process_information, 378
multimodal_m4_recovery, 379
multimodal_m4_sensitivity, 380
multimodal_m4_spec, 380
multimodal_m4_state_diagnostics, 381
multimodal_ppc, 382

negative_control_concordance, 382
negative_control_process_test, 383
new_aoi (eyeprocess-trials-aoi), 159

new_coordinate_space
(eyeprocess-schema), 158
new_eye_dataset (eyeprocess-class), 150
normalize_timebase
(eyeprocess-coordinates-time), 150

object_hash, 384
object_schema, 384
open_eye_storage, 385
open_partitioned_eye_storage, 385
optimize_item_bank
(item_objective_spec), 352
option_process_information, 386
outcome_blind_feature_audit, 386
outcome_blind_snapshot, 387

package_reproducibility_manifest, 387
paper_reproducibility_manifest, 388
parameter_recovery
(eyeprocess-simulation), 158
partition_eye_storage, 388
person_scores (eyeprocess-models), 155
perturb_aoi_geometry
(aoi_perturbation_uncertainty), 30
pipeline_failures, 389
pipeline_result, 389
pipeline_step_status, 390
placebo_window_audit, 390
plot.eye_adapter_regression_audit, 391
plot.eye_aoi_composition
(derive_aoi_composition), 112
plot.eye_aoi_composition_comparison
(derive_aoi_composition), 112
plot.eye_aoi_composition_model
(derive_aoi_composition), 112
plot.eye_aoi_growth_curve, 391
plot.eye_aoi_separation_audit
(assign_aois_probabilistic), 38
plot.eye_aoi_trajectory, 392
plot.eye_aoi_uncertainty
(assign_aois_probabilistic), 38
plot.eye_bank_decision_stability
(item_objective_spec), 352
plot.eye_bayesian_process_dashboard, 392
plot.eye_bids_roundtrip, 393
plot.eye_biometric_imputation_sensitivity, 393
plot.eye_biometric_preflight, 394
plot.eye_calibration_drift
(detect_calibration_drift), 115
plot.eye_candidate_item_bank_audit, 394
plot.eye_compatibility_evidence_matrix, 395
plot.eye_corpus_validation
(eyeprocess-format-validation), 153
plot.eye_cross_recurrence
(gaze_recurrence), 333
plot.eye_crossmodal_recurrence_model
(fit_process_missingness_model), 301
plot.eye_dataset (eyeprocess-plots), 156
plot.eye_decision_process_proxy, 395
plot.eye_decision_trace
(build_evidence_graph), 80
plot.eye_device_equivalence
(fit_device_linking), 259
plot.eye_device_linking
(fit_device_linking), 259
plot.eye_dif_drift (fit_process_dif), 297
plot.eye_dynamic_irtree, 396
plot.eye_dynamic_ppc, 396
plot.eye_episode_comparison
(detect_process_changepoints), 118
plot.eye_event_roundtrip_audit, 397
plot.eye_event_time_irt, 397
plot.eye_evidence_graph
(build_evidence_graph), 80
plot.eye_fairness_transportability
(fit_process_dif), 297
plot.eye_fixation_point_process
(fit_fixation_point_process), 269
plot.eye_format_validation
(eyeprocess-format-validation), 153
plot.eye_functional_pupil_diagnostics, 398
plot.eye_functional_pupil_irt, 398
plot.eye_functional_pupil_sensitivity,

399
plot.eye_gated_process_model, 399
plot.eye_gaze_anchored_3pl_audit, 400
plot.eye_gaze_informed_missingness_irt, 400
plot.eye_gaze_point_process_diagnostics (fit_fixation_point_process), 269
plot.eye_gpirt, 401
plot.eye_incremental_information_audit, 401
plot.eye_irt_changepoints, 402
plot.eye_irt_equating, 402
plot.eye_irt_ppc, 403
plot.eye_irt_recovery_summary, 403
plot.eye_irt_sbc, 404
plot.eye_item_bank_optimization (item_objective_spec), 352
plot.eye_item_parameter_seed, 405
plot.eye_item_pareto (item_objective_spec), 352
plot.eye_item_reduction_sensitivity, 405
plot.eye_joint_gaze_rt_irt, 406
plot.eye_joint_graded_rt_process_irt, 406
plot.eye_latent_distribution_comparison, 407
plot.eye_latent_process_alignment, 407
plot.eye_latent_space_irt, 408
plot.eye_manyfacet_process_irt, 408
plot.eye_mixture_irt_process, 409
plot.eye_mnar_sensitivity (fit_process_observation_model), 304
plot.eye_mnar_tipping_point (fit_process_observation_model), 304
plot.eye_model_promotion_audit, 409
plot.eye_multiblock_process_map, 410
plot.eye_nominal_gaze_irt, 410
plot.eye_nonparametric_rasch_audit, 411
plot.eye_norm_transportability (fit_process_norms), 302
plot.eye_omission_survival_irt, 411
plot.eye_parameter_recovery (eyeprocess-simulation), 158
plot.eye_phase_amplitude_irt (register_pupil_curves), 529
plot.eye_preaction_process_features, 412
plot.eye_presentation_accessibility, 412
plot.eye_presentation_fairness_comparison, 413
plot.eye_probabilistic_aoi (assign_aois_probabilistic), 38
plot.eye_process_anomaly_audit, 413
plot.eye_process_cat_simulation, 414
plot.eye_process_changepoints (detect_process_changepoints), 118
plot.eye_process_channel_ablation, 414
plot.eye_process_dependent_discrimination, 415
plot.eye_process_dif (fit_process_dif), 297
plot.eye_process_drift_audit, 415
plot.eye_process_dstudy (fit_process_gstudy), 299
plot.eye_process_episodes (detect_process_changepoints), 118
plot.eye_process_external_validity, 416
plot.eye_process_facet_effects, 417
plot.eye_process_feature_blocks, 417
plot.eye_process_g_study, 418
plot.eye_process_gstudy (fit_process_gstudy), 299
plot.eye_process_hmm_irt, 418
plot.eye_process_local_dependence_audit, 419
plot.eye_process_negative_control, 419
plot.eye_process_norms (fit_process_norms), 302
plot.eye_process_observation_model (fit_process_observation_model), 304
plot.eye_process_person_fit, 420
plot.eye_process_profile_mixture, 420
plot.eye_process_rasch_tree, 421
plot.eye_process_reliability_audit (fit_process_gstudy), 299
plot.eye_process_uncertainty

(process_uncertainty_spec), 503
plot.eye_process_uncertainty_propagation (process_uncertainty_spec), 503
plot.eye_process_window_sensitivity, 422
plot.eye_process_windows, 421
plot.eye_provenance_comparison (build_evidence_graph), 80
plot.eye_pupil_confound_model, 422
plot.eye_pupil_deconvolution, 423
plot.eye_pupil_fatigue_drift, 423
plot.eye_pupil_frequency_features, 424
plot.eye_pupil_frequency_stability, 424
plot.eye_pupil_phase_amplitude (register_pupil_curves), 529
plot.eye_pupil_registration (register_pupil_curves), 529
plot.eye_recalibration_audit (detect_calibration_drift), 115
plot.eye_recurrence (gaze_recurrence), 333
plot.eye_roundtrip_loss_audit, 425
plot.eye_sbc_audit, 425
plot.eye_scanpath_bootstrap (representative_scanpath), 534
plot.eye_scanpath_comparison (representative_scanpath), 534
plot.eye_scanpath_representative (representative_scanpath), 534
plot.eye_semantic_roundtrip, 426
plot.eye_signal_filter_audit, 426
plot.eye_storage_benchmark, 427
plot.eye_strategy_aoi_sensitivity, 427
plot.eye_streaming_score, 428
plot.eye_transition_diagnostics, 428
plot.eye_uncertainty_budget_comparison (process_uncertainty_spec), 503
plot.eye_validation_bundle, 429
plot.eye_vendor_compatibility_matrix, 429
plot.eye_vendor_semantic_validation, 430
plot.eye_visual_context_irt, 430
plot.eye_windowed_recurrence (gaze_recurrence), 333
plot_aoi_assignment_stability (aoi_perturbation_uncertainty), 30
plot_aoi_balance_biplot (derive_aoi_composition), 112
plot_aoi_boundary_risk (assign_aois_probabilistic), 38
plot_aoi_coefficient_stability (aoi_perturbation_uncertainty), 30
plot_aoi_composition_trajectory (derive_aoi_composition), 112
plot_aoi_dwell (eyeprocess-plots), 156
plot_aoi_metric_uncertainty (assign_aois_probabilistic), 38
plot_aoi_perturbations (aoi_perturbation_uncertainty), 30
plot_aoi_probability_map (assign_aois_probabilistic), 38
plot_aoi_robustness_surface (aoi_perturbation_uncertainty), 30
plot_aoi_ternary (derive_aoi_composition), 112
plot_aoi_transition_matrix, 431
plot_aoi_transition_rank, 431
plot_aoi_variation_matrix (derive_aoi_composition), 112
plot_bank_information_coverage (item_objective_spec), 352
plot_bcea (standardized_spatial_quality), 584
plot_biometrics (eyeprocess-plots), 156
plot_calibration_error_ellipses (detect_calibration_drift), 115
plot_calibration_vector_field (detect_calibration_drift), 115
plot_changepoint_ribbons (detect_process_changepoints), 118
plot_clock_alignment (eyeprocess-plots), 156
plot_complete_case_sensitivity (fit_process_observation_model), 304
plot_compositional_group_difference (derive_aoi_composition), 112
plot_coordinate_spaces

(eyeprocess-plots), 156
plot_covariate_effect_surface (fit_fixation_point_process), 269
plot_cross_vendor_metric_matrix (fit_device_linking), 259
plot_crossmodal_recurrence (gaze_recurrence), 333
plot_decision_stability (item_objective_spec), 352
plot_dependability_surface (fit_process_gstudy), 299
plot_detector_agreement (event_detector_multiverse), 133
plot_detector_coefficient_stability (event_detector_multiverse), 133
plot_detector_event_timeline (event_detector_multiverse), 133
plot_detector_feature_distributions (event_detector_multiverse), 133
plot_detector_multiverse (event_detector_multiverse), 133
plot_device_agreement (fit_device_linking), 259
plot_device_bias_by_magnitude (fit_device_linking), 259
plot_device_equivalence_intervals (fit_device_linking), 259
plot_device_transfer_curve (fit_device_linking), 259
plot_diagnostics (eye_plot_spec), 243
plot_diagonal_recurrence_profile (gaze_recurrence), 333
plot_dif_drift_heatmap (fit_process_dif), 297
plot_distractor_information, 432
plot_drift_over_time (detect_calibration_drift), 115
plot_episode_duration_distribution (detect_process_changepoints), 118
plot_episode_transition_graph (detect_process_changepoints), 118
plot_episode_waterfall (detect_process_changepoints), 118
plot_evidence (eye_plot_spec), 243
plot_evidence_graph (build_evidence_graph), 80
plot_eye_overview (eyeprocess-plots), 156
plot_eye_trace (eyeprocess-plots), 156
plot_fairness_transport_matrix (fit_process_dif), 297
plot_feature_correlation (eyeprocess-plots), 156
plot_feature_distribution (eyeprocess-plots), 156
plot_fixation_intensity (fit_fixation_point_process), 269
plot_fixations (eyeprocess-plots), 156
plot_fuzzy_transition_matrix (assign_aois_probabilistic), 38
plot_gaze_accuracy (standardized_spatial_quality), 584
plot_gaze_cox_diagnostics (gaze-survival), 324
plot_gaze_cumulative_incidence (gaze-survival), 324
plot_gaze_hazard (gaze-survival), 324
plot_gaze_heatmap (eyeprocess-plots), 156
plot_gaze_precision (standardized_spatial_quality), 584
plot_gaze_quality_dashboard (standardized_spatial_quality), 584
plot_gaze_survival_curve (gaze-survival), 324
plot_gazepoint_workflow, 433
plot_group_icc_process_overlay (fit_process_dif), 297
plot_group_scanpath_transport (representative_scanpath), 534
plot_interval_coverage, 433
plot_irf_uncertainty, 434
plot_item_decision_path

(build_evidence_graph), 80
plot_item_difficulty
(eyeprocess-plots), 156
plot_item_group_process_curves
(fit_process_dif), 297
plot_item_normative_deviation
(fit_process_norms), 302
plot_item_pareto (item_objective_spec), 352
plot_item_phase_delay
(register_pupil_curves), 529
plot_item_sampling_reliability
(fit_process_gstudy), 299
plot_metric_dependency_graph
(build_evidence_graph), 80
plot_missingness (eyeprocess-plots), 156
plot_missingness_by_aoi
(fit_process_observation_model), 304
plot_missingness_by_time
(fit_process_observation_model), 304
plot_mnar_tipping_point
(fit_process_observation_model), 304
plot_model_decision_impact
(build_evidence_graph), 80
plot_model_diagnostics
(eyeprocess-plots), 156
plot_normative_fan (fit_process_norms), 302
plot_objective_tradeoffs
(item_objective_spec), 352
plot_observation_probability
(fit_process_observation_model), 304
plot_observed_expected_fixations
(fit_fixation_point_process), 269
plot_parameter_recovery, 434
plot_person_item_space, 435
plot_person_normative_profile
(fit_process_norms), 302
plot_phase_amplitude_scores
(register_pupil_curves), 529
plot_probabilistic_scanpath
(assign_aois_probabilistic), 38
plot_process_centiles
(fit_process_norms), 302
plot_process_changepoint, 436
plot_process_channel_ablation_delta, 436
plot_process_dif_forest
(fit_process_dif), 297
plot_process_episodes
(detect_process_changepoints), 118
plot_process_feature_stability, 437
plot_process_window_sensitivity, 438
plot_pupil_activity_sensitivity, 438
plot_pupil_activity_windows, 439
plot_pupil_band_power, 439
plot_pupil_components, 440
plot_pupil_preprocessing_audit, 440
plot_pupil_registration
(register_pupil_curves), 529
plot_pupil_spectrum, 441
plot_pupil_timeseries
(eyeprocess-plots), 156
plot_recalibration_before_after
(detect_calibration_drift), 115
plot_recurrence_matrix
(gaze_recurrence), 333
plot_recurrence_network
(gaze_recurrence), 333
plot_registered_pupil_effects
(register_pupil_curves), 529
plot_reliability_by_metric
(fit_process_gstudy), 299
plot_representative_scanpath
(representative_scanpath), 534
plot_sampling_intervals
(standardized_spatial_quality), 584
plot_sampling_rate (eyeprocess-plots), 156
plot_sbc_rank, 442
plot_scanpath (eyeprocess-plots), 156
plot_scanpath_atlas
(representative_scanpath), 534
plot_scanpath_dispersion
(representative_scanpath), 534
plot_scanpath_similarity_matrix
(representative_scanpath), 534
plot_screen_coverage
(detect_calibration_drift), 115

plot_selected_bank_profile (item_objective_spec), 352
plot_sensitivity (eye_plot_spec), 243
plot_session_stability (fit_process_gstudy), 299
plot_signal_quality (eyeprocess-plots), 156
plot_spatial_residuals (fit_fixation_point_process), 269
plot_temporal_excitation_kernel (fit_fixation_point_process), 269
plot_transition_matrix (eyeprocess-plots), 156
plot_trial_timeline (eyeprocess-plots), 156
plot_uncertainty_by_item (process_uncertainty_spec), 503
plot_uncertainty_by_stage (process_uncertainty_spec), 503
plot_uncertainty_tornado (process_uncertainty_spec), 503
plot_uncertainty_waterfall (process_uncertainty_spec), 503
plot_validation_failures, 442
plot_validation_runtime, 443
plot_variance_components (fit_process_gstudy), 299
plot_warping_functions (register_pupil_curves), 529
plot_windowed_recurrence (gaze_recurrence), 333
posterior_predictive_discrepancies, 443
posterior_sbc_contract, 444
power_process_simulation (eyeprocess-simulation), 158
preaction_process_features, 444
predict.eye_censored_normal_process_irt, 445
predict.eye_multinomial_transition, 446
predict.eyeprocess_model (eyeprocess-models), 155
predict_aoi_trajectory, 446
predict_fixation_intensity (fit_fixation_point_process), 269
predict_gaze_survival (gaze-survival), 324
predict_item_parameter_priors, 447
predict_process_centiles (fit_process_norms), 302
predict_theta_at_time, 447
preflight_decisions, 448
preflight_exclusion_manifest, 448
preflight_failures, 449
preflight_passed, 449
prepare_dynamic_irtree_data, 450
prepare_functional_pupil_data, 450
prepare_gaze_diffusion_data, 451
prepare_gaze_survival_data (gaze-survival), 324
prepare_multilevel_mediation_data (multilevel-mediation), 364
prepare_multimodal_irt_data, 452
prepare_strategy_mixture_data, 453
prepare_structured_unstructured_process_features, 453
preprocess_eye (eyeprocess-preprocessing), 157
preprocess_spec (eyeprocess-preprocessing), 157
preprocessing_multiverse, *136*, 454
print.eye_aoi (eyeprocess-trials-aoi), 159
print.eye_benchmark_reproduction, 455
print.eye_benchmark_study, 455
print.eye_benchmark_validation, 456
print.eye_clock_transform (eyeprocess-coordinates-time), 150
print.eye_corpus_validation (eyeprocess-format-validation), 153
print.eye_dataset (eyeprocess-class), 150
print.eye_diffusion_diagnostics, 456
print.eye_diffusion_identification_study, 457
print.eye_dynamic_aoi (eyeprocess-models), 155
print.eye_dynamic_irtree, 457
print.eye_dynamic_ppc, 458
print.eye_dynamic_recovery, 458

print.eye_engine_adapter_result, 459
print.eye_feature_spec
(eyeprocess-features), 152
print.eye_format_detection
(eyeprocess-adapters), 149
print.eye_format_validation
(eyeprocess-format-validation), 153
print.eye_format_validation_spec
(eyeprocess-format-validation), 153
print.eye_functional_pupil_data, 459
print.eye_functional_pupil_diagnostics, 460
print.eye_functional_pupil_irt, 460
print.eye_functional_pupil_sensitivity, 461
print.eye_functional_scalar_comparison, 461
print.eye_gated_process_model, 462
print.eye_gaze_diffusion_data, 462
print.eye_gaze_diffusion_irt, 463
print.eye_gaze_diffusion_spec, 463
print.eye_irt_evidence_grade, 464
print.eye_irt_model_spec, 464
print.eye_irt_validation_spec, 465
print.eye_mapping
(eyeprocess-adapters), 149
print.eye_mi_result (eye_plot_spec), 243
print.eye_missing_sensitivity
(eyeprocess-experimental), 151
print.eye_model_contract_validation, 465
print.eye_model_promotion_audit, 466
print.eye_multinomial_transition, 466
print.eye_partition_spec, 467
print.eye_partitioned_storage, 467
print.eye_plot_spec (eye_plot_spec), 243
print.eye_preprocess_spec
(eyeprocess-preprocessing), 157
print.eye_process_drift_spec, 468
print.eye_process_negative_control, 468
print.eye_process_preflight_spec, 469
print.eye_process_uncertainty_spec
(process_uncertainty_spec), 503
print.eye_process_window_spec, 469
print.eye_readiness
(eyeprocess-quality), 157
print.eye_redaction_result, 470
print.eye_roundtrip_loss_audit, 470
print.eye_roundtrip_validation
(eyeprocess-format-validation), 153
print.eye_sensitivity
(eyeprocess-quality), 157
print.eye_strategy_data, 471
print.eye_strategy_manipulation_validation, 471
print.eye_strategy_mixture
(eyeprocess-experimental), 151
print.eye_theory_strategy_irt, 472
print.eye_theory_strategy_spec, 472
print.eye_transition_design, 473
print.eye_transition_diagnostics, 473
print.eye_two_stage_rt
(eyeprocess-models), 155
print.eye_validation
(eyeprocess-class), 150
print.eye_validation_bundle, 474
print.eye_validation_case_fingerprint, 474
print.eye_validation_collection, 475
print.eye_validation_completion_audit, 475
print.eye_validation_job_plan, 476
print.eye_validation_run, 476
print.eye_vendor_case, 477
print.eye_vendor_compatibility_matrix, 477
print.eye_vendor_schema_contract, 478
print.eye_vendor_semantic_comparison, 478
print.eye_visual_context_registry, 479
print.eyeprocess_model
(eyeprocess-models), 155
print.process_irt_spec
(eyeprocess-experimental), 151
print.summary.eye_dataset
(eyeprocess-class), 150
print.summary.eyeprocess_model
(eyeprocess-models), 155
probabilistic_aoi_assignment, 479
process_anomaly_distance, 480
process_bland_altman, 481
process_channel_ablation, 481

process_criterion_associations, 482
process_dependent_discrimination_audit, 483
process_dif_nuisance_surrogate, 484
process_drift_alerts, 484
process_drift_spec, 485
process_feature_blocks, 486
process_feature_family_registry, 486
process_feature_stability, 487
process_feature_time_provenance, 487
process_icc, 488
process_information, 489
process_irt_diagnostics (eyeprocess-experimental), 151
process_irt_spec (eyeprocess-experimental), 151
process_item_information, 489
process_measure_card, 490
process_measure_coverage, 491
process_measure_guardrails, 491
process_measure_lineage, 492
process_measure_registry, 492
process_measure_units, 493
process_negative_control_permute, 493
process_negative_control_shift, 494
process_ngram_features, 495
process_null_benchmark, 495
process_pattern_mixture (fit_process_observation_model), 304
process_person_fit, 496
process_preflight_spec, 496
process_profile_probabilities, 498
process_profile_summary, 498
process_reliability_profile, 499
process_residual_map, 499
process_sensitivity_grid, 500
process_sequence_embedding, 500
process_state_occupancy, 501
process_state_transition_summary, 501
process_temporal_stability, 502
process_uncertainty_spec, 503
process_validation_design, 504
process_variance_components (fit_process_gstudy), 299
process_window_spec, 505
promote_irt_model, 506
promote_vendor_support, 507
propagate_aoi_uncertainty (assign_aois_probabilistic), 38
propagate_calibration_uncertainty, 507
propagate_detector_to_aoi (event_detector_multiverse), 133
propagate_detector_to_features (event_detector_multiverse), 133
propagate_process_uncertainty (process_uncertainty_spec), 503
provenance_edge_table, 508
provenance_lineage_table, 509
provenance_manifest (eyeprocess-class), 150
prune_validation_checkpoints, 509
public_validation_corpus, 510
pupil_activity_index, 510
pupil_band_power, 511
pupil_baseline_sensitivity, 512
pupil_confound_effects, 512
pupil_deconvolve (eyeprocess-preprocessing), 157
pupil_event_effects, 513
pupil_event_regressor, 513
pupil_frequency_features, 514
pupil_labs_format (eyeprocess-import-vendors), 155
pupil_latency_sensitivity, 515
pupil_preprocessing_grid, 516
pupil_preprocessing_sensitivity, 516
pupil_response_kernel, 517
pupil_unit_fidelity_audit, 518
pupil_velocity_activity, 519

quantify_process_leakage, 519
query_eye_storage, 520

raven_reproduction_spec, 520
read_api_lifecycle_registry, 521
read_benchmark_table, 522
read_decision_manifest, 522
read_eye_dataset (eyeprocess-export-report), 152
read_eye_export (eyeprocess-adapters), 149
read_eye_folder (eyeprocess-adapters), 149

read_eye_generic
(eyeprocess-import-generic), 154
read_eyelink_asc
(eyeprocess-import-vendors), 155
read_eyelink_edf
(eyeprocess-import-vendors), 155
read_eyelink_report
(eyeprocess-import-vendors), 155
read_eyeprocess_validation_evidence, 523
read_gazepoint
(eyeprocess-import-gazepoint), 154
read_gazepoint_aoi_statistics
(eyeprocess-import-gazepoint), 154
read_gazepoint_biometrics
(eyeprocess-import-gazepoint), 154
read_gazepoint_combined
(eyeprocess-import-gazepoint), 154
read_gazepoint_events
(eyeprocess-import-gazepoint), 154
read_gazepoint_fixations
(eyeprocess-import-gazepoint), 154
read_gazepoint_folder
(eyeprocess-import-gazepoint), 154
read_gazepoint_gaze
(eyeprocess-import-gazepoint), 154
read_gazepoint_summary
(eyeprocess-import-gazepoint), 154
read_pupil_core
(eyeprocess-import-vendors), 155
read_pupil_neon
(eyeprocess-import-vendors), 155
read_pupillabs
(eyeprocess-import-vendors), 155
read_reproducibility_fingerprint, 523
read_smi (eyeprocess-import-vendors), 155
read_smi_aoi_export
(eyeprocess-import-vendors), 155
read_smi_event_export
(eyeprocess-import-vendors), 155
read_smi_raw_export
(eyeprocess-import-vendors), 155
read_tobii (eyeprocess-import-vendors), 155
read_validation_job_manifest, 524
read_validation_manifest
(eyeprocess-format-validation), 153
read_validation_scenario_manifest, 524
read_vendor_registry, 525
recalibrate_after_changepoint, 525
recommended_validation_replications, 526
recompute_aoi_features
(aoi_perturbation_uncertainty), 30
recurrence_features (gaze_recurrence), 333
redact_validation_case, 527
register_aois (eyeprocess-trials-aoi), 159
register_coordinate_space
(eyeprocess-coordinates-time), 150
register_eye_adapter
(eyeprocess-adapters), 149
register_eye_api_status, 527
register_irt_model, 528
register_process_measure, 529
register_pupil_curves, 529
register_validation_case, 531
register_vendor_semantics, 532
remap_recording_ids
(eyeprocess-adapters), 149
report_aoi_sensitivity
(aoi_perturbation_uncertainty),

30
report_detector_multiverse (event_detector_multiverse), 133
report_eye_dataset (eyeprocess-export-report), 152
report_gaze_quality (standardized_spatial_quality), 584
report_gaze_survival_model (gaze-survival), 324
report_processirt (eyeprocess-export-report), 152
reporting_guideline_audit, 533
representative_scanpath, 534
response_matrix (eyeprocess-models), 155
response_time_matrix (eyeprocess-models), 155
resume_eye_pipeline, 535
resume_validation_jobs, 536
rolling_apply (eyeprocess-preprocessing), 157
roundtrip_eye_bids, 536
roundtrip_eye_dataset (eyeprocess-format-validation), 153
run_aoi_sensitivity_analysis (aoi_perturbation_uncertainty), 30
run_benchmark_reproduction, 537
run_detector_inference_multiverse (event_detector_multiverse), 133
run_detector_multiverse (event_detector_multiverse), 133
run_eye_benchmark, 542
run_eye_pipeline, 543
run_eyeprocess_equateirt, 538
run_eyeprocess_irt_ability_sbc, 538
run_eyeprocess_irt_recovery, 539
run_eyeprocess_mirtcat, 540
run_eyeprocess_stress_evidence, 540
run_eyeprocess_validation_program, 541
run_gazepoint_workflow, 544
run_model_validation, 545
run_posterior_sbc, 546
run_process_negative_controls, 546
run_process_sensitivity, 547
run_process_validation, 548
run_raven_reproduction, 549
run_sbc, 549
run_validation_jobs, 550

sbc_ecdf_deviation, 551
sbc_rank_diagnostics, 552
sbc_summary, 552
scanpath_dispersion (representative_scanpath), 534
scanpath_sequence (eyeprocess-features), 152
schema_coverage (eyeprocess-format-validation), 153
schema_coverage_summary (eyeprocess-format-validation), 153
schema_table (eyeprocess-schema), 158
score_partial_response_pattern, 553
score_process_deviation (fit_process_norms), 302
score_response_stream, 553
segment_process_episodes (detect_process_changepoints), 118
select_next_item_process, 554
semantic_fidelity_spec, 555
semantic_loss_map, 556
semantic_roundtrip_audit, 556
sensitivity_branch_fingerprint, 557
sensitivity_decision_leverage, 558
sensitivity_fragility_index, 558
sensitivity_missing_process (eyeprocess-experimental), 151
sensitivity_mnar_process (fit_process_observation_model), 304
sensitivity_multiverse_manifest, 559
sensitivity_process (eyeprocess-quality), 157
sensitivity_rank_stability, 559
sensitivity_sign_stability, 560
sensitivity_significance_stability, 560
sensitivity_threshold_stability, 561
sequence_interoperability, 561
session_facet_effects, 562

set_eye_table *(eyeprocess-class)*, 150
simulate_advanced_process_data, 563
simulate_detector_multiverse_data *(event_detector_multiverse)*, 133
simulate_dynamic_irtree_data, 564
simulate_eye_dataset *(eyeprocess-simulation)*, 158
simulate_eyeprocess_catr, 565
simulate_eyeprocess_irt_binary, 565
simulate_from_model, 566
simulate_gaze_diffusion_data, 567
simulate_gaze_quality_calibration *(standardized_spatial_quality)*, 584
simulate_gaze_survival_example *(gaze-survival)*, 324
simulate_gaze_survival_inputs *(gaze-survival)*, 324
simulate_irt_model, 568
simulate_multimodal_irt, 568
simulate_multimodal_m2, 569
simulate_multimodal_m3, 570
simulate_multimodal_m4, 572
simulate_presentation_variants, 573
simulate_process_cat, 573
simulate_process_irt *(eyeprocess-simulation)*, 158
simulate_process_validation_data, 574
simulate_strategy_mixture_data, 575
simulation_based_calibration, 575
simulation_rank_statistic, 576
software_paper_claim_matrix, 577
software_paper_coverage, 578
software_paper_evidence_bundle, 578
software_paper_gap_analysis, 579
software_paper_readiness, 580
software_paper_validation_table, 581
source_preservation_audit *(eyeprocess-format-validation)*, 153
specification_coverage, 581
specification_curve_data, 582
split_half_process_reliability, 582
split_validation_plan, 583
standardize_eye_table *(eyeprocess-schema)*, 158
standardized_spatial_quality, 584
storage_transaction_manifest, 588
store_quality *(eyeprocess-quality)*, 157
strategy_aoi_sensitivity, 588
strategy_classification_uncertainty, 589
strategy_label_switching_diagnostics, 589
strategy_posterior_probabilities, 590
streaming_score_history, 590
stress_test_latent_distribution, 591
stress_test_local_dependence, 591
stress_test_missingness, 592
stress_test_misspecification, 593
stress_test_preprocessing, 593
stress_test_process_pipeline, 594
stress_test_speededness, 595
stress_test_summary, 595
stress_tolerance_frontier, 596
structural_transition_mask, 596
summarise_aoi_membership *(assign_aois_probabilistic)*, 38
summarise_aoi_sensitivity *(aoi_perturbation_uncertainty)*, 30
summarise_detector_disagreement *(event_detector_multiverse)*, 133
summarise_detector_events *(event_detector_multiverse)*, 133
summarise_detector_robustness *(event_detector_multiverse)*, 133
summarise_eye_benchmark, 598
summarise_eyeprocess_stress_evidence, 597
summarise_gaze_censoring *(gaze-survival)*, 324
summarise_process_negative_controls, 598
summarise_process_sensitivity, 599
summarise_process_validation, 599
summarise_sampling_quality *(standardized_spatial_quality)*, 584
summarise_spatial_quality *(standardized_spatial_quality)*, 584

summarise_validation_acceptance, 600
summarise_within_between_variance (multilevel-mediation), 364
summarize_fixations (eyeprocess-features), 152
summarize_parameter_recovery, 600
summarize_process_windows, 601
summary.eye_dataset (eyeprocess-class), 150
summary.eye_format_validation (eyeprocess-format-validation), 153
summary.eye_parameter_recovery (eyeprocess-simulation), 158
summary.eye_validation_bundle, 601
summary.eye_validation_job_plan, 602
summary.eyeprocess_model (eyeprocess-models), 155
supported_eye_formats (eyeprocess-adapters), 149
survival::coxph, *327*
survival::Surv, *327*
survival::survreg, *327*
synchronize_eye_biometrics (eyeprocess-coordinates-time), 150
synthetic_corruption_plan, 602

theory_strategy_spec, 603
tidy_gaze_survival_model (gaze-survival), 324
timestamp_fidelity_audit, 605
trace_item_decision (build_evidence_graph), 80
transform_aoi_composition (derive_aoi_composition), 112
transition_entropy (eyeprocess-features), 152
transition_matrix (eyeprocess-features), 152
transition_residual_diagnostics, 606
translate_aoi (aoi_perturbation_uncertainty), 30
trial_table (eyeprocess-features), 152

uncertainty_budget (process_uncertainty_spec), 503
unregister_eye_adapter (eyeprocess-adapters), 149
update_person_score, 606
upgrade_eye_dataset, 608
upgrade_eyeprocess_model, 607

validate_against_reference, 608
validate_aoi_geometry (aoi_perturbation_uncertainty), 30
validate_benchmark_study, 609
validate_bids_eye_semantics, 609
validate_decision_manifest, 610
validate_engine_adapter, 610
validate_event_detector_spec (event_detector_multiverse), 133
validate_eye_corpus (eyeprocess-format-validation), 153
validate_eye_dataset (eyeprocess-class), 150
validate_eye_mapping (eyeprocess-adapters), 149
validate_eye_pipeline, 613
validate_eye_prov_graph, 614
validate_eye_source (eyeprocess-format-validation), 153
validate_eye_storage_metadata, 614
validate_eye_table (eyeprocess-schema), 158
validate_eyelink_export (eyeprocess-format-validation), 153
validate_eyeprocess_external_irt_fit, 611
validate_eyeprocess_irt_item_bank, 611
validate_eyeprocess_irt_model_spec, 612
validate_eyeprocess_joint_process_irt_spec, 612
validate_eyeprocess_validation_plan, 613
validate_feature_availability, 615
validate_gaze_quality_inputs (standardized_spatial_quality), 584

validate_gaze_survival_data (gaze-survival), 324
validate_gazepoint_workflow, 615
validate_generic_export (eyeprocess-format-validation), 153
validate_hed_event_semantics, 616
validate_irt_model, 616
validate_latent_space_process_similarity, 617
validate_model_object, 617
validate_multilevel_mediation_data (multilevel-mediation), 364
validate_multimodal_irt, 618
validate_multimodal_m2, 618
validate_multimodal_m3, 619
validate_multimodal_m4, 619
validate_process_measure_registry, 620
validate_process_validation_design, 621
validate_process_windows, 621
validate_pupillabs_export (eyeprocess-format-validation), 153
validate_smi_export (eyeprocess-format-validation), 153
validate_strategy_manipulation, 622
validate_tobii_export (eyeprocess-format-validation), 153
validate_vendor_semantics, 622
validate_vendor_timestamp_semantics, 623
validation_acceptance_matrix, 624
validation_acceptance_rule, 624
validation_bundle_manifest, 625
validation_calibration_summary, 626
validation_condition_id, 626
validation_condition_ranking, 627
validation_coverage_table, 627
validation_evidence_levels, 628
validation_evidence_matrix, 628
validation_failure_profile, 629
validation_failure_summary, 629
validation_failure_taxonomy, 630
validation_job_plan, 630
validation_ladder, 631
validation_manifest (eyeprocess-format-validation), 153
validation_mcse, 631
validation_mcse_profile, 632
validation_recovery_summary, 632
validation_recovery_table, 633
validation_replication_budget, 633
validation_report, 634
validation_robustness_score, 634
validation_runtime_summary, 635
validation_sbc_summary, 635
validation_scenario_manifest, 636
validation_seed, 637
validation_summary_mcse, 637
validation_thresholds, 638
vendor_schema_contract, 639
vendor_validation_spec, 640
verify_decision_manifest_lock, 641
verify_eyeprocess_validation_atlas, 641
verify_eyeprocess_validation_evidence, 642
verify_outcome_blind_snapshot, 642
verify_reproducibility_fingerprint, 643
verify_reproducibility_manifest, 643
visual_context_registry, 644

windowed_recurrence (gaze_recurrence), 333
write_advanced_model_evidence_report, 644
write_api_lifecycle_registry, 645
write_benchmark_data_dictionary, 646
write_decision_manifest, 646
write_eye_dataset (eyeprocess-export-report), 152
write_eye_pipeline_report, 648
write_eye_storage, 648
write_eye_targets_template, 649
write_eyeprocess_validation_evidence, 647
write_eyeprocess_validation_report, 647
write_format_validation_report (eyeprocess-format-validation), 153
write_gazepoint_workflow_report, 650

write_model_promotion_report, 650
write_partitioned_eye_storage, 651
write_prov_dot, 651
write_provenance
(eyeprocess-export-report), 152
write_reporting_guideline_report, 652
write_reproducibility_fingerprint, 652
write_software_paper_evidence, 653
write_software_paper_reproduction, 653
write_software_paper_scaffold, 654
write_validation_job_manifest, 654
write_validation_manifest
(eyeprocess-format-validation),
153
write_validation_release_report, 655
write_validation_report, 655
write_validation_scenario_manifest,
656
write_vendor_case_report, 656
write_vendor_registry, 657
write_vendor_validation_report, 657