Package 'VCMoE'

November 23, 2026

Title Varying-Coefficient Mixture-of-Experts Models

Version 0.3.0

Description Fits Gaussian, Binomial, and Negative-Binomial varying-coefficient mixture-of-experts models with local-linear estimation, explicit label alignment, bandwidth selection, diagnostics, bootstrap inference, confidence bands, coefficient-specific generalized likelihood-ratio tests (GLRT) with analytic or bootstrap calibration, and local-grid or joint-path expectation-maximization fitting engines. For clustered observations it provides cluster-robust (CR2) intervals with wild cluster bootstrap bands and path tests, and a cluster-robust screen of candidate covariates for expert effects.

License MIT + file LICENSE

URL https://qc-zhao.github.io/VCMoE/, https://github.com/qc-zhao/VCMoE

BugReports https://github.com/qc-zhao/VCMoE/issues

Encoding UTF-8

RoxygenNote 7.3.3

Imports ggplot2, stats, utils

Suggests knitr, MASS, rmarkdown, sandwich, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Qicheng Zhao [aut, cre],
Celia M.T. Greenwood [aut],
Qihuang Zhang [aut]

Maintainer Qicheng Zhao <790471666@qq.com>

Repository CRAN

Date/Publication 2026-11-22 23:20:02 UTC

Contents

coef.vcmoe . . . 2
confint.vcmoe_bootstrap . . . 3
plot_coefficients . . . 4
plot_diagnostics . . . 4
plot_inference . . . 5
plot_posterior . . . 5
predict.vcmoe . . . 6
simulate_vcmoe_binomial . . . 7
simulate_vcmoe_gaussian . . . 7
simulate_vcmoe_negbin . . . 8
vcmoe_bootstrap . . . 9
vcmoe_confband . . . 10
vcmoe_diagnostics . . . 11
vcmoe_fit . . . 12
vcmoe_fit_reduced . . . 14
vcmoe_gating_contrasts . . . 15
vcmoe_gene_screen . . . 16
vcmoe_glrt . . . 17
vcmoe_grid_neighbor_inits . . . 19
vcmoe_merge_grid_fits . . . 20
vcmoe_parameterization . . . 20
vcmoe_refine . . . 21
vcmoe_scaled_slopes . . . 22
vcmoe_select_bandwidth . . . 22

Index **24**

`coef.vcmoe` *Extract VCMoE coefficients*

Description

Extracts expert coefficients, gating coefficients, Gaussian variance intercepts, Gaussian log-sigma local-linear slopes, Negative-Binomial theta, or all fitted coefficient blocks from a VCMoE fit.

Usage

```
## S3 method for class 'vcmoe'
coef(object, type = c("all", "expert", "gating", "sigma", "sigma_slope", "theta"), ...)
```

Arguments

`object`	A `vcmoe` object.
`type`	Coefficient block to return.
`...`	Unused.

Details

For Gaussian fits, `coef(fit, "sigma")` returns the component-specific standard deviation function at each `u_grid` point, and `coef(fit, "sigma_slope")` returns the scaled local-linear slope of `log(sigma)` on the `(u - u0) / h` basis.

For Binomial fits, expert coefficients are on the logit success-probability scale and `coef(fit, "sigma")` returns `NULL`. For Negative-Binomial fits, expert coefficients are on the log mean count scale, `coef(fit, "theta")` returns the component-specific size parameter, and `coef(fit, "sigma")` returns `NULL`.

Value

A list or array of fitted coefficient functions.

`confint.vcmoe_bootstrap` *Bootstrap confidence intervals for VCMoE coefficients*

Description

Summarizes pointwise or simultaneous bootstrap intervals for expert or gating coefficient functions.

Usage

```
## S3 method for class 'vcmoe_bootstrap'
confint(
  object,
  parm = c("expert", "gating"),
  level = 0.95,
  type = c("pointwise", "simultaneous"),
  ...
)
```

Arguments

`object`	A `vcmoe_bootstrap` object.
`parm`	Coefficient set to summarize: `"expert"`, `"gating"`, or both.
`level`	Confidence level.
`type`	Interval type. `"pointwise"` uses percentile bootstrap intervals. `"simultaneous"` uses a max standardized bootstrap-deviation band over `u` for each component and term.
`...`	Unused.

Details

Pointwise intervals are percentile intervals at each grid point. Simultaneous bands compute bootstrap standard errors and use the empirical quantile of the maximum standardized absolute deviation over the `u_grid`. Near-zero standard errors are floored internally to avoid division by zero.

Value

A tidy data frame with columns `coefficient_set`, `term`, `component`, `u`, `estimate`, `se`, `lower`, `upper`, `type`, `level`, and `n_successful`.

---

plot_coefficients *Plot fitted coefficient functions*

Description

Plots fitted expert or gating coefficient functions over the VCMoE grid.

Usage

```
plot_coefficients(object, type = c("expert", "gating"))
```

Arguments

`object`	A `vcmoe` object.
`type`	`"expert"` or `"gating"`.

Value

A `ggplot` object.

---

plot_diagnostics *Plot VCMoE fit diagnostics*

Description

Plots convergence, posterior entropy, component proportions, effective local sample size, and label ambiguity flags over the coefficient grid.

Usage

```
plot_diagnostics(object)
```

Arguments

`object`	A `vcmoe` object.

Details

This plot is intended as a first real-data sanity check before interpreting coefficient functions. Ambiguity or non-convergence at many grid points should be treated as evidence that the fitted component labels or coefficient paths need closer review.

Value

A ggplot object.

---

plot_inference *Plot bootstrap inference intervals*

Description

Plots fitted coefficient functions with bootstrap pointwise intervals or simultaneous bands.

Usage

```
plot_inference(
  object,
  coefficient_set = "expert",
  type = c("pointwise", "simultaneous"),
  level = 0.95
)
```

Arguments

object	A vcmoe_bootstrap object.
coefficient_set	Coefficient set to plot: "expert" or "gating".
type	Interval type passed to confint().
level	Confidence level.

Value

A ggplot object.

---

plot_posterior *Plot fitted posterior summaries*

Description

Plots mean posterior probabilities for each component over the coefficient grid.

Usage

```
plot_posterior(object)
```

Arguments

object	A vcmoe object.

Value

A ggplot object.

`predict.vcmoe` *Predict from a VCMoE fit*

Description

Returns fitted means, component-specific means, posterior probabilities, or gating probabilities.

Usage

```
## S3 method for class 'vcmoe'
predict(object, newdata = NULL, u = NULL,
  type = c("mean", "posterior", "component", "prior"), ...)
```

Arguments

`object`	A vcmoe object.
`newdata`	Optional data frame.
`u`	Optional index values for newdata.
`type`	Prediction type.
`...`	Unused.

Details

For Gaussian fits, `type = "component"` returns component-specific means and `type = "mean"` returns the posterior-weighted fitted mean. For Binomial fits, `type = "component"` returns component-specific success probabilities and `type = "mean"` returns the marginal success probability. For Negative-Binomial fits, `type = "component"` returns component-specific mean counts and `type = "mean"` returns the marginal mean count.

Value

A vector or matrix depending on type.

simulate_vcmoe_binomial

Simulate Binomial VCMoE data

Description

Generates Binomial VCMoE simulations for Bernoulli and grouped-count examples and tests.

Usage

```
simulate_vcmoe_binomial(n = 300L, k = 2L, seed = NULL,
  separation = 1, u = NULL, scenario = "well_separated", trials = 1L)
```

Arguments

n	Number of observations.
k	Number of components. Values 2 through 10 are supported.
seed	Optional random seed.
separation	Controls expert separation.
u	Optional numeric vector of index values.
scenario	Simulation scenario: "well_separated", "moderate", "near_overlap", "crossing", or "imbalanced_gating".
trials	Binomial trial counts. Use 1 for Bernoulli data, or a positive integer scalar/vector for grouped Binomial data.

Value

A list with data and truth. Expert truth is on the logit scale. The truth entry includes component coefficients, gating logits, component probabilities, component-specific success probabilities, sampled class labels, and success/failure counts.

simulate_vcmoe_gaussian

Simulate Gaussian VCMoE data

Description

Generates a small Gaussian no-offset VCMoE simulation for tutorials and tests.

Usage

```
simulate_vcmoe_gaussian(n = 300L, k = 2L, seed = NULL,
  separation = 1, u = NULL, scenario = "well_separated")
```

Arguments

n	Number of observations.
k	Number of components. Values 2 through 10 are supported.
seed	Optional random seed.
separation	Controls expert separation.
u	Optional numeric vector of index values.
scenario	Simulation scenario: "well_separated", "moderate", "near_overlap", "crossing", or "imbalanced_gating".

Value

A list with data and truth. The truth entry includes component coefficients, gating logits, probabilities, means, standard deviations, and sampled class labels.

---

simulate_vcmoe_negbin *Simulate Negative-Binomial VCMoE count data*

Description

Generates Negative-Binomial VCMoE simulations for gene-expression count examples and tests.

Usage

```
simulate_vcmoe_negbin(n = 300L, k = 2L, seed = NULL,
  separation = 1, u = NULL, scenario = "well_separated",
  size_factor = NULL, mean_count = 5)
```

Arguments

n	Number of observations.
k	Number of components. Values 2 through 10 are supported.
seed	Optional random seed.
separation	Controls expert separation.
u	Optional numeric vector of index values.
scenario	Simulation scenario: "well_separated", "moderate", "near_overlap", "crossing", or "imbalanced_gating".
size_factor	Optional positive size factors. If NULL, log-normal size factors are generated.
mean_count	Baseline count scale.

Value

A list with data and truth. Expert truth is on the log mean count scale. The data include size_factor and log_size_factor for use with offset(log_size_factor).

vcmoe_bootstrap *Parametric bootstrap inference for a VCMoE fit*

Description

Parametric bootstrap inference for a VCMoE fit

Usage

```
vcmoe_bootstrap(
  fit,
  data,
  u = NULL,
  B = 200L,
  coefficient_set = c("expert", "gating"),
  seed = NULL,
  control = list(),
  min_successful = max(20L, ceiling(0.5 * B)),
  keep_fits = FALSE,
  verbose = FALSE
)
```

Arguments

fit	A vcmoe fit with k = 2:10.
data	Original data frame used to fit fit.
u	Optional original u values or column name.
B	Number of bootstrap replicates.
coefficient_set	Coefficient sets to store.
seed	Optional random seed.
control	Control overrides for bootstrap refits.
min_successful	Minimum successful replicates for reliable inference.
keep_fits	Whether to store successful bootstrap fit objects.
verbose	Whether to message progress.

Details

Bootstrap refits preserve the reference fitting engine. A joint-path reference is therefore refitted with vcmoe_fit(..., engine = "joint_path_em") rather than silently falling back to local-grid EM.

Value

An object of class vcmoe_bootstrap.

vcmoe_confband *Confidence bands for a VCMoE fit*

Description

Pointwise intervals and finite-grid simultaneous bands for the coefficient functions of a fitted VC-MoE model, from the local sandwich covariance.

Usage

```
vcmoe_confband(
  fit,
  data = NULL,
  level = 0.95,
  type = c("pointwise", "simultaneous"),
  coefficient_set = c("expert", "gating", "sigma", "theta"),
  strict = TRUE,
  control = list(),
  cluster = NULL
)
```

Arguments

fit	A vcmoe fit with k = 2:10.
data	Optional original data frame. The current implementation uses the data stored in fit$fitted; refit with keep_data = TRUE if needed.
level	Confidence level.
type	Interval columns to expose as lower and upper.
coefficient_set	Coefficient blocks to return.
strict	Whether weak local fits should return blocked intervals.
control	Optional inference controls. Without cluster, the covariance is HC0 and the simultaneous band uses the analytic Epanechnikov path critical value; simultaneous_method = "multiplier_coefficient_path" calibrates each coefficient over the complete fitted grid by a Gaussian multiplier bootstrap instead. With cluster, covariance_adjustment may be "CR0", "CR1", "CR2" (default) or "CR3", and simultaneous_method "wild_cluster_path" (default) or "multiplier_coefficient_path". multiplier_B (default 199) sets the number of bootstrap draws and seed their random seed. path_tests = TRUE adds the zero and constancy path tests described below; path_sets, a named list of grid-index vectors, adds bands and tests over sub-paths (for example the interior grid points).
cluster	Optional atomic vector of independent-cluster identifiers (for example donors), one per retained fitted row, in fitted-row order (not the original unfiltered order). If named, names must match the retained design-matrix row names exactly. Missing IDs are rejected.

Details

Independent observations. HC0 sandwich intervals with normal critical values. For `engine = "joint_path_em"`, the covariance is the observed local-likelihood sandwich evaluated at the joint-path estimate; it does not include shared-path, label-selection, or finite-grid cross-grid responsibility uncertainty, so joint-path convergence and the returned score-imbalance diagnostics should be inspected.

Clustered observations. With `cluster`, kernel-weighted scores are summed within clusters. Cluster inference supports unrestricted local-grid fits that retain their data. Coefficients and the penalized bread are unchanged. CR0 uses normal critical values. CR1 scales by `G / (G - 1)` with `t(G - 1)`, for `G` clusters with positive kernel weight at the grid point. CR2 (bias-reduced linearization in score space) and CR3 (one-step cluster jackknife) use Bell-McCaffrey degrees of freedom under a working model of independent observations within clusters; both use cluster Hessian blocks. The default simultaneous band studentizes Webb-weighted wild cluster score draws with the same adjustment at every grid point; one weight per cluster is shared by all grid points and coefficients. When the number of distinct Webb weight vectors, `6^G`, is at most `multiplier_B`, all of them are used. Clusters must be independent; smoothing bias and bandwidth selection are not corrected. With fewer than 20 clusters, or fewer than 10 effective clusters, at a grid point a warning flags the intervals as sensitivity analyses.

Paths and tests. A band covers one coefficient's complete fitted grid, not a continuous domain or all coefficients jointly. A blocked grid point blocks that coefficient's complete-path band, not its other pointwise intervals; a band over a sub-path in `path_sets` remains available. With `path_tests = TRUE` each complete path also gets two sup-t tests from the same bootstrap draws as its band: `beta(u) = 0` (the band-inversion test) and `beta(u)` constant, which contrasts each grid estimate with the path mean and studentizes the contrast with the same small-cluster adjustment. These tests are the recommended constancy tests for clustered data; the likelihood-ratio test of `vcmoe_glrt()` assumes independent observations.

Value

A `vcmoe_confband` object with interval and diagnostic data frames; with `cluster`, `cluster_support` reports per-component cluster support, and with `path_tests = TRUE`, `tests` holds the path-test statistics and p-values.

`vcmoe_diagnostics` *Summarize VCMoE fit diagnostics*

Description

Returns a compact diagnostic table for reviewing whether a fitted VCMoE model is reliable enough to interpret.

Usage

```
vcmoe_diagnostics(object)
```

Arguments

object	A vcmoe object.

Details

The table includes convergence status, iterations, local log-likelihood, local-weighted posterior entropy, label ambiguity flags, alignment margin, effective local sample size, local-weighted component posterior proportions, and expert optimizer diagnostics when available.

Posterior entropy and component proportions use the same local kernel weights as the fitted grid point when the fit retains training data. If the fit was created with `control$keep_data = FALSE`, component proportions fall back to unweighted posterior means and effective local sample size is `NA`.

Value

A data frame with one row per coefficient grid point. Besides the fit summaries it reports, per grid point, the EM stop reason (`"tolerance"` or `"maxit"`), the last change of the penalized local objective, the selected start, whether an unconverged start reached a higher objective (`better_unconverged_start`), whether global label alignment overrode a confident neighbour match (`alignment_override`), and whether the grid point was refined by `vcmoe_refine()`. Negative- Binomial eta-bound counts use only observations with positive kernel weight at the grid point.

---

vcmoe_fit *Fit a varying-coefficient mixture-of-experts model*

Description

Fit a varying-coefficient mixture-of-experts model

Usage

```
vcmoe_fit(
  formula,
  data,
  u,
  k = 2L,
  family = "gaussian",
  bandwidth = NULL,
  u_grid = NULL,
  control = list(),
  label = "align",
  parameterization = "a1_epanechnikov_scaled",
  u_scale = c("unit", "none"),
  engine = c("local_grid_em", "joint_path_em"),
  progress = NULL
)
```

Arguments

formula	A formula of the form `y ~ expert_terms | gating_terms`. For grouped binomial data, use `cbind(success, failure) ~ expert_terms | gating_terms`. For Negative-Binomial count data, expert-side `offset(log_size_factor)` terms are supported.
data	A data frame.
u	Continuous index column name or numeric vector.
k	Number of mixture components. `k = 2` is the primary v0 target; `k = 3:10` are high-k candidate support and require diagnostics before being treated as stable.
family	Model family. `"gaussian"`, `"binomial"`, and `"negative-binomial"` are implemented.
bandwidth	Kernel bandwidth. If `NULL`, a Silverman-style default is used for `u`.
u_grid	Grid where coefficient functions are estimated.
control	Named list overriding EM and label-alignment settings. Common entries: `maxit` (default 100), `tol` (default `1e-5`), `n_starts` (default 3) and `seed`. The local EM stops when the relative change of the penalized local objective, `|delta F| / (|F| + 1)`, falls below `tol`; with large samples this can stop well before the local maximum, so check `vcmoe_diagnostics()` and use `vcmoe_refine()` before inference. Negative-Binomial: `negbin_theta_ridge` and `negbin_theta_target` add a ridge on `log(theta)` towards the target, scaled by the kernel mass of the grid point (`negbin_theta_penalty_mass = "kernel"`, the default; `"component"` is the rule of VCMoE 0.2.0). Grid-parallel fitting (local-grid engine): `grid_subset` fits only the given grid indices and returns a `vcmoe_grid_part` to merge with `vcmoe_merge_grid_fits()`; `grid_seed = "per_grid"` reseeds each grid point with `seed + index`; `grid_warm_start = FALSE` drops the warm start from the previous grid point; `grid_init_locals` replaces the starts at given grid points (see `vcmoe_grid_neighbor_inits()`). The defaults reproduce the serial fit. Known labels (local-grid engine): `fixed_membership` gives each row of `data` a component, as indices in `1..k`, a factor (levels in component order) or an `nrow(data) x k` 0/1 matrix. Each local fit is then the kernel-weighted complete-data fit (component-wise experts and a multinomial-logit gate on the labels), with one start and no label alignment, and inference uses the complete-data local likelihood. Joint path: `joint_path_global_start` (default `TRUE`) makes the first start the posterior of a mixture of experts without `u` dependence, which keeps component labels consistent along `u`.
label	Label strategy. `"align"` uses exact global alignment for `k <= 6` and sequential assignment with ambiguity margins for `k >= 7`; `"global"` requests exact global alignment when feasible and falls back to the same sequential assignment path for `k >= 7`; `"greedy"` keeps the older one-step alignment.
parameterization	Estimator convention. The public package uses `"a1_epanechnikov_scaled"`: Epanechnikov density weights `K((u-u0)/h)/h` and the scaled local-linear basis `(u-u0)/h`.
u_scale	How to transform `u` before fitting. The default `"unit"` maps complete-row `u` values to `[0, 1]`; `"none"` leaves `u` on the supplied scale. `bandwidth` and `u_grid` are interpreted on the transformed analysis scale.

engine	Fitting engine. The default "local_grid_em" preserves the original independent local-grid EM path. "joint_path_em" updates a shared observation-level responsibility path across grid points.
progress	Joint-path progress reporting. The default NULL is silent; use TRUE for messages or a single CSV file path for structured logging. Iteration frequency is controlled by control$progress_every.

Details

At each grid point, starts that met the stopping rule are preferred and the one with the highest penalized local objective is kept; when no start met it, the best finite start is kept with converged = FALSE. fit$diagnostics records, per grid point, the stop reason, the last objective change, the selected start and whether an unconverged start reached a higher objective. With global label alignment, a grid point relabelled against a confident match with its neighbour is flagged in alignment_override with a warning; compare with label = "greedy".

Rows with missing or non-finite response, covariates, or u are removed consistently before fitting, with a warning. For single-trial Bernoulli responses, the default gating ridge is strengthened to control$ridge = 1 unless the user explicitly supplies control$ridge; grouped Binomial and other families keep the global default. Joint-path traces record the sample-level nearest-grid log-likelihood as a diagnostic criterion. The label-consistent updates do not guarantee that this diagnostic is monotone; convergence is based on posterior and parameter deltas instead.

Value

An object of class vcmoe, or a vcmoe_grid_part when control$grid_subset is set.

---

vcmoe_fit_reduced *Fit a block-constant reduced VCMoE model*

Description

Refits a VCMoE object under a block-constant coefficient constraint. A local-grid reference uses the established constrained BFGS optimizer; a joint-path reference uses a grid projection (equal weights by default; see null_projection in vcmoe_glrt()) and applies the selected constraint after every M-step. This projection is not a generic constrained optimizer, and its diagnostic likelihood need not be monotone.

Usage

```
vcmoe_fit_reduced(
  fit,
  constrain = c("gating_constant", "expert_constant", "all_constant"),
  control = list()
)
```

Arguments

fit	A vcmoe fit.
constrain	Constraint to impose. "gating_constant" freezes the gating contrast functions in u; "expert_constant" freezes expert mean functions and family dispersion paths; "all_constant" freezes all fitted coefficient functions.
control	Controls for constrained null fitting. With the default strict = TRUE, a nonconverged reduced fit is rejected.

Value

A reduced object of class vcmoe with recomputed posterior and likelihood caches.

---

vcmoe_gating_contrasts

Report Identifiable VCMoE Gating Contrasts

Description

Report identifiable gating contrasts.

Usage

```
vcmoe_gating_contrasts(object, baseline = NULL, scaled = FALSE)
```

Arguments

object	A vcmoe object.
baseline	Component used as the contrast baseline. By default, k = 2 uses component 2 and k > 2 uses component 1.
scaled	If TRUE, slope contrasts are converted to the scaled local-linear basis (u - u0) / h.

Details

VCMoE stores gating coefficients as centered logits, so the absolute level of all component logits is not identifiable. Interpretable gating effects are component contrasts such as component 1 versus component 2 for k = 2, or component 1 and component 2 versus component 3 for k = 3 comparisons when baseline = 3.

Value

A data frame with one row per grid point, contrast, term, and block.

vcmoe_gene_screen *Cluster-robust gene screen for Negative-Binomial expert effects*

Description

Screens candidate covariates (typically genes) for an association with the response inside the latent components, while the fitted VCMoE (the null model) is held fixed. For each candidate, every grid point of the null fit is augmented by the covariate, entering each expert's local linear predictor (the tested coefficients) and each gating contrast (nuisance), with coefficients starting at zero. A one-step estimate of the expert coefficients is studentized with the CR2 cluster-robust covariance and calibrated over the grid by a studentized wild cluster bootstrap, as in the path tests of `vcmoe_confband()`. The null model is fitted once, so all candidates share its moments and the screen scales to thousands of genes.

Usage

```
vcmoe_gene_screen(
  fit,
  X,
  cluster,
  profile_genes = NULL,
  grids = NULL,
  B = 9999L,
  seed = NULL,
  adjust = c("BH", "minP", "both"),
  chunk_size = 200L,
  control = list(),
  profile_control = list()
)
```

Arguments

`fit`	The null model: a single-response Negative-Binomial local-grid `vcmoe` fit that retains its data.
`X`	Numeric matrix of candidate covariates, one row per retained fitted row (in fitted-row order) and one uniquely named column per candidate. Columns are centred internally.
`cluster`	Independent-cluster identifiers (for example donors), one per retained fitted row. Required.
`profile_genes`	Optional column names of `X` to screen after profiling their gating effect: the candidate first enters the gating of the null model and every screened grid point is refitted by local EM, giving the restricted local fit under the null (no expert effect, gating effect free). Use this for strong markers of the latent components, whose gating effect is far from zero; the one-step screen holds it at zero.
`grids`	Grid indices that form the tested path. Defaults to every grid point. A candidate is reported with an `NA` p-value when any of these grid points is blocked for it.

B	Number of wild bootstrap draws (Webb weights). One set of weights is generated per call and shared by every candidate, chunk and profiled refit, including when seed = NULL. When 6^G <= B for G clusters, all 6^G weight vectors are used.
seed	Optional seed for the bootstrap weights.
adjust	Multiplicity adjustment over the full candidate set: "BH" (Benjamini-Hochberg, default), "minP" (single-step Westfall-Young min-p from the shared draws) or "both". Candidates with an NA p-value stay in the family size of the BH adjustment.
chunk_size	Number of candidates processed together; smaller values use less memory and do not change the results.
control	Optional inference controls passed to the local moment computations (for example strict, max_hessian_dim).
profile_control	List with maxit (default 300) and tol (default 1e-9) for the profiled refits.

Details

The screen tests expert effects, so the null model should explain component membership well: include covariates that distinguish the latent components (for example known subtype markers) in its gating formula. Otherwise strong markers can leak into the expert test. Results describe associations on the log-mean scale of each expert; effect_per_sd gives the one-step level coefficient at the maximising grid point per standard deviation of the candidate.

Value

A list of class vcmoe_gene_screen with genes (one row per candidate: sup-t statistic, raw and adjusted p-values, maximising component and grid, per-component statistics and directions, per-SD effects, grid usage and status), pointwise (per grid and coefficient one-step estimates, standard errors and Bell-McCaffrey degrees of freedom), grids (grid-level status), profile (profiled refits), draws (shared bootstrap maxima) and settings.

---

vcmoe_glrt *Generalized likelihood-ratio test for VCMoE coefficient variation*

Description

Generalized likelihood-ratio test for VCMoE coefficient variation

Usage

```
vcmoe_glrt(
  fit,
  data,
  test = c("coefficient", "constant_block", "constant_all"),
  coefficient_set = c("expert", "gating", "sigma", "theta"),
```

```
  component = NULL,
  term = NULL,
  calibration = c("none", "bootstrap", "analytic_epanechnikov", "both",
    "parametric_bootstrap"),
  B = 200L,
  seed = NULL,
  control = list(),
  refit_control = list(),
  verbose = FALSE
)
```

Arguments

fit	A vcmoe fit.
data	Original data frame used to fit fit.
test	Test type. "coefficient" tests one coefficient function; "constant_block" tests all expert or gating functions jointly; "constant_all" tests all fitted coefficient functions jointly.
coefficient_set	Coefficient block for coefficient-specific or block-constant tests.
component	Component label or index for coefficient-specific tests.
term	Term name for coefficient-specific tests.
calibration	Calibration method. The default "none" returns the statistic without attaching a reference distribution. "analytic_epanechnikov" uses the Epanechnikov modified chi-square approximation; "bootstrap" uses parametric bootstrap calibration; "both" reports both.
B	Number of bootstrap calibration replicates.
seed	Optional random seed.
control	Controls for constrained null optimization and diagnostics: lambda_type ("observation", the default; "fit_objective", the scale used by VCMoE 0.2.0; or "grid_sum"), strict (default FALSE), C_const (default 1), null_projection for joint-path nulls ("uniform", the default, or "sample_weighted", the rule of VCMoE 0.2.0), maxit and reltol for the null optimizer, and lambda_tol.
refit_control	Controls overriding bootstrap full-model refits.
verbose	Whether to message bootstrap progress.

Details

Statistic. lambda is the difference between the full and null observation-level mixture log-likelihoods, sum_i log sum_k pi_k(u_i) f_k(y_i; u_i), with each coefficient path evaluated at the observation's own u_i by the linear interpolation used by predict.vcmoe(). This is the likelihood of the Wilks-type result for local likelihood ratio tests. lambda_type = "fit_objective" instead uses the objective the engine maximises (the sum of grid-point local likelihoods for local-grid fits, the nearest-grid observation likelihood for joint-path fits). A sum over grid points counts each observation about 1 / (grid spacing) times, so analytic p-values on that scale are far too small; it is kept only to reproduce earlier results. The fitted-objective ratio is always reported as

lambda_fit_objective, and a negative value blocks the test as a null-optimizer failure. A negative observation-level ratio is a valid, non-significant outcome.

Analytic calibration. The statistic rK * max(lambda, 0) is compared with a chi-square distribution with delta = rK * p_test * C_const * |U| * (K(0) - int K^2 / 2) / h degrees of freedom, where p_test is the number of coefficient functions constrained under the null: 1 for one coefficient function, k - 1 contrasts for a gating term, and the whole block for block tests. The approximation is intended for independent observations and joint-path fits; for local-grid fits it has not been validated and settings$theory_status says so. With clustered observations (for example cells from the same donor), use the cluster-robust path tests of vcmoe_confband() instead.

Null fits. Local-grid fits use a constrained BFGS null optimizer. Joint-path fits use a grid-projected null: after every M-step each constrained coefficient path is replaced by its average over the grid points (equal weights by default; with null_projection = "sample_weighted", weights proportional to the number of observations assigned to each nearest grid point), and constrained local slopes are set to zero. An optimizer error, or a null fit that beats the full fit on the fitted objective, blocks the test and returns an NA p-value. A null fit that stops at its iteration limit gives a p-value with the warning null_optimizer_not_fully_converged and null_converged = FALSE; strict = TRUE blocks it, together with every other warning.

Bootstrap. Bootstrap refits preserve the engine of fit. Each refit numbers its components independently, so a component-specific test follows the matched reference component into the refit; for a gating contrast with k >= 3 both components of the contrast are followed. Draws whose null optimizer failed, or whose null beats the full refit, are excluded; draws whose null fit only reached its iteration limit are kept. bootstrap_counts reports the requested, successful and excluded draws.

Value

A vcmoe_glrt object.

---

vcmoe_grid_neighbor_inits

Neighbour warm starts for a second grid-parallel pass

Description

For each requested grid point, returns the best local fits across parts (ranked as in vcmoe_merge_grid_fits()) at the neighbouring grid points, ready to pass as control$grid_init_locals in a second vcmoe_fit() call with control$grid_subset. This restores the serial fit's warm start in both directions without the serial dependency.

Usage

```
vcmoe_grid_neighbor_inits(parts, grid_ids = NULL)
```

Arguments

parts	A list of vcmoe_grid_part objects.
grid_ids	Grid indices to build starts for; default all covered ones.

Value

A named list (by grid index) of lists of local fits.

`vcmoe_merge_grid_fits` *Merge grid-subset VCMoE fits*

Description

Combines `vcmoe_grid_part` objects returned by `vcmoe_fit()` with `control$grid_subset` (for example one per cluster job) into an ordinary `vcmoe` fit. Every grid point must be covered. When several parts cover the same grid point (for example a cold-start pass and a neighbour warm-start pass), a converged local fit is preferred, then the one with the higher penalized local objective. Label alignment is then run over the whole grid as in a serial fit.

Usage

```
vcmoe_merge_grid_fits(parts, label = NULL)
```

Arguments

`parts`	A list of `vcmoe_grid_part` objects from the same data and settings.
`label`	Optional label-alignment override: `"align"`, `"global"` or `"greedy"`. The default keeps the alignment of the original call.

Value

A `vcmoe` object; `diagnostics$grid_source` records which part supplied each grid point.

`vcmoe_parameterization`

Inspect VCMoE Parameterization Metadata

Description

Inspect VCMoE parameterization metadata.

Usage

```
vcmoe_parameterization(object)
```

Arguments

`object`	A `vcmoe` object.

Details

The package default is `"a1_epanechnikov_scaled"`: Epanechnikov density weights `0.75 * (1 - t^2)_+ / h` with `t = (u - u0) / h`, scaled local-linear slope storage, and centered gating logits. This helper reports those conventions for reproducible model summaries.

Value

A named list describing the estimator convention used by the fit, including kernel weights, local-linear basis scale, gating-logit storage, dispersion block, label-alignment method, and optimization controls.

`vcmoe_refine` *Refine a local-grid VCMoE fit*

Description

Continues the local EM at selected grid points from the fitted solution, with a tighter tolerance and more iterations. The relative stopping rule of `vcmoe_fit()` (`tol = 1e-5` by default) corresponds to a large absolute change of the local objective when the sample is large, so a fit can stop before the local maximum is reached; the bread used by `vcmoe_confband()` is then often not positive definite. Refining fixes this without refitting from scratch. Each refined grid point keeps the labels of the original fit; a refinement that would change them is reported, not applied silently.

Usage

```
vcmoe_refine(fit, grids = NULL, tol = 1e-08, maxit = 1000L)
```

Arguments

`fit`	A local-grid `vcmoe` fit that retains its data (`control$keep_data = TRUE`).
`grids`	Grid indices to refine. The default refines every grid point.
`tol`	Relative tolerance on the change of the penalized local objective.
`maxit`	Maximum number of EM iterations per grid point.

Value

The refined `vcmoe` fit. `diagnostics$refined` marks the refined grid points, `diagnostics$refine_objective_gain` gives the change of the penalized local objective and `diagnostics$refine_relabelled` flags grid points whose refinement moved to a relabelled solution (kept with the original labels).

vcmoe_scaled_slopes *Inspect VCMoE Local-Linear Slopes On The Scaled Basis*

Description

Inspect VCMoE local-linear slopes on the scaled basis.

Usage

```
vcmoe_scaled_slopes(object, type = c("expert", "gating"), bandwidth = NULL)
```

Arguments

object	A vcmoe object.
type	Coefficient block, either "expert" or "gating".
bandwidth	Optional bandwidth recorded in the returned attributes. Defaults to the fitted bandwidth.

Details

VCMoE stores slopes on the scaled local-linear basis (u - u0) / h. This helper returns the stored scaled-basis slope block.

Value

An array with the same dimensions as the stored slope block.

vcmoe_select_bandwidth *Select a VCMoE bandwidth by K-fold cross-validation*

Description

Select a VCMoE bandwidth by K-fold cross-validation

Usage

```
vcmoe_select_bandwidth(
  formula,
  data,
  u,
  k = 2L,
  family = "gaussian",
  bandwidth_grid = NULL,
  folds = 5L,
```

```
  u_grid = NULL,
  control = list(),
  label = "align",
  parameterization = "a1_epanechnikov_scaled",
  u_scale = c("unit", "none"),
  seed = NULL,
  refit = TRUE,
  engine = c("local_grid_em", "joint_path_em")
)
```

Arguments

formula	A formula of the form y ~ expert_terms | gating_terms.
data	A data frame.
u	Continuous index column name or numeric vector.
k	Number of mixture components. Values from 2 through 10 are supported.
family	Model family. "gaussian", "binomial", and "negative-binomial" are supported.
bandwidth_grid	Candidate bandwidth values. If NULL, uses multiples of the default bandwidth.
folds	Number of random cross-validation folds.
u_grid	Grid where coefficient functions are estimated.
control	Named list passed to the selected fitting engine.
label	Label strategy passed to the selected fitting engine.
parameterization	Estimator convention passed to the selected fitting engine.
u_scale	u scaling strategy passed to the selected fitting engine.
seed	Optional random seed for fold assignment and, when control$seed is absent, deterministic CV refits.
refit	Whether to refit the final model on all data using the selected bandwidth.
engine	Fitting engine used for every cross-validation fold and the optional final refit. The default "local_grid_em" preserves the 0.1.0 behavior; "joint_path_em" uses the joint-path engine of vcmoe_fit().

Value

An object of class vcmoe_bandwidth_selection.

Index

coef.vcmoe, 2
confint.vcmoe_bootstrap, 3

plot_coefficients, 4
plot_diagnostics, 4
plot_inference, 5
plot_posterior, 5
predict.vcmoe, 6
predict.vcmoe(), *18*

simulate_vcmoe_binomial, 7
simulate_vcmoe_gaussian, 7
simulate_vcmoe_negbin, 8

vcmoe_bootstrap, 9
vcmoe_confband, 10
vcmoe_confband(), *16*, *19*, *21*
vcmoe_diagnostics, 11
vcmoe_fit, 12
vcmoe_fit(), *19–21*
vcmoe_fit_reduced, 14
vcmoe_gating_contrasts, 15
vcmoe_gene_screen, 16
vcmoe_glrt, 17
vcmoe_glrt(), *11*, *14*
vcmoe_grid_neighbor_inits, 19
vcmoe_grid_neighbor_inits(), *13*
vcmoe_merge_grid_fits, 20
vcmoe_merge_grid_fits(), *13*, *19*
vcmoe_parameterization, 20
vcmoe_refine, 21
vcmoe_refine(), *12, 13*
vcmoe_scaled_slopes, 22
vcmoe_select_bandwidth, 22