Package 'surveyframe'

May 27, 2026

Title Survey Instrument Workflows

Version 0.3.0

Description Supports survey research workflows built around a typed instrument object (the sframe). Features include visual instrument design via a browser-based builder or 'Shiny' studio, export to a self-contained static HTML survey, an embeddable 'Shiny' module, SHA-256 integrity-checked serialisation to the '.sframe' format, multi-page survey rendering, branching logic, response quality checking, scale scoring, psychometric diagnostics, analysis-plan execution, model syntax planning, an interactive response dashboard, codebook generation, and reproducible HTML reporting.

License MIT + file LICENSE

URL https://github.com/MohammedAliSharafuddin/surveyframe

BugReports https://github.com/MohammedAliSharafuddin/surveyframe/issues

Encoding UTF-8

Language en-GB

Depends R (>= 4.1.0)

Imports jsonlite (>= 1.8.0), rlang (>= 1.1.0), openssl (>= 2.1.0)

Suggests googlesheets4 (>= 1.1.0), shiny (>= 1.7.0), psych (>= 2.3.0), MASS, nnet, digest (>= 0.6.0), testthat (>= 3.0.0), knitr, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

RoxygenNote 8.0.0

NeedsCompilation no

Author Mohammed Ali Sharafuddin [aut, cre] (ORCID: <https://orcid.org/0000-0001-5247-2964>)

Maintainer Mohammed Ali Sharafuddin <mohammedali.page@gmail.com>

Repository CRAN

Date/Publication 2026-05-27 20:10:09 UTC

Contents

`add_model` *Add a model specification to an instrument*

Description

Add a model specification to an instrument

Usage

```
add_model(instrument, model, validate = TRUE, replace = TRUE)
```

Arguments

`instrument`	An `sframe` object.
`model`	An `sf_model()` object.
`validate`	Logical. Whether to validate the model against the instrument before adding it.
`replace`	Logical. Whether to replace an existing model with the same ID. Defaults to `TRUE`.

Value

The updated `sframe` object.

assumption_report *Assumption-check report*

Description

Performs common assumption checks for survey analyses using base R where possible: Shapiro-Wilk tests, skewness/kurtosis screening, Levene and Brown-Forsythe tests, regression residual checks, VIF, Cook's distance, expected-count checks, and sparse-cell warnings.

Usage

```
assumption_report(
  data,
  variables = NULL,
  group = NULL,
  outcome = NULL,
  predictors = NULL,
  table_vars = NULL
)
```

Arguments

data	A data.frame.
variables	Numeric variables for normality screening.
group	Optional grouping variable for Levene/Brown-Forsythe tests.
outcome	Optional regression outcome.
predictors	Optional regression predictors.
table_vars	Optional two categorical variables for expected-count checks.

Value

An object of class `sframe_assumption_report`.

cfa_lavaan_syntax *Generate lavaan CFA syntax*

Description

Generate lavaan CFA syntax

Usage

```
cfa_lavaan_syntax(
  instrument = NULL,
  model = NULL,
  scales = NULL,
  ordered = FALSE,
  std_lv = TRUE,
  residual_covariances = NULL,
  latent_covariances = TRUE
)
```

Arguments

`instrument`	Optional `sframe` object used to derive constructs from scales when `model` is not supplied.
`model`	Optional `sf_model()` object.
`scales`	Optional scale IDs when deriving a model from an instrument.
`ordered`	Logical. Whether to add an ordered-item note.
`std_lv`	Logical. Whether to add a `std.lv = TRUE` note.
`residual_covariances`	Optional list of `sf_covariance()` objects for correlated residuals.
`latent_covariances`	Logical. Whether to include model-level latent covariances supplied in `model`.

Value

A lavaan syntax string.

`cfa_syntax` *Generate lavaan CFA syntax from an instrument object*

Description

Produces a character string of `lavaan` model syntax derived from the scale structure in the instrument. The syntax can be passed directly to `lavaan::cfa()`. Reverse-coded items are noted in a comment but are not transformed in the syntax; recoding should be applied to the data before fitting the model.

Usage

```
cfa_syntax(instrument, scales = NULL, std_lv = TRUE)
```

Arguments

instrument	An sframe object.
scales	Character vector or NULL. A subset of scale IDs to include. When NULL, all scales are included.
std_lv	Logical. Whether to include the std.lv = TRUE argument note in the output comment header. Defaults to TRUE.

Value

A character string of lavaan CFA model syntax.

See Also

efa_report(), reliability_report()

Examples

```
cs    <- sf_choices("ag5", 1:5,
           c("Strongly disagree", "Disagree", "Neutral",
             "Agree", "Strongly agree"))
i1    <- sf_item("sat_1", "Item 1", type = "likert",
                 choice_set = "ag5", scale_id = "sat")
i2    <- sf_item("sat_2", "Item 2", type = "likert",
                 choice_set = "ag5", scale_id = "sat")
i3    <- sf_item("sat_3", "Item 3 (reverse)", type = "likert",
                 choice_set = "ag5", scale_id = "sat", reverse = TRUE)
scale <- sf_scale("sat", "Satisfaction",
                  items = c("sat_1", "sat_2", "sat_3"))
instr <- sf_instrument("Demo Survey", components = list(cs, i1, i2, i3, scale))

syntax <- cfa_syntax(instr)
cat(syntax)

## Not run:
# lavaan is not installed by default; install it before fitting.
demo   <- sframe_demo_data()
scored <- score_scales(demo$responses, demo$instrument)
fit    <- lavaan::cfa(syntax, data = scored, std.lv = TRUE)
summary(fit, fit.measures = TRUE)

## End(Not run)
```

---

codebook_report *Generate a survey codebook from an instrument object*

Description

Produces a structured codebook listing all items, their types, choice sets, scale membership, and reverse-coding status. The codebook can be rendered as HTML or Markdown.

Usage

```
codebook_report(instrument, format = c("html", "md"))
```

Arguments

instrument	An sframe object.
format	Character. Output format. Either "html" or "md".

Value

An object of class sframe_codebook, a list with elements instrument_meta, items_table, choices_table, and scales_table. Call print() to display a compact summary or use render_report() to include the codebook in a full report.

See Also

render_report()

Examples

```
cs    <- sf_choices("ag5", 1:5,
           c("Strongly disagree", "Disagree", "Neutral",
             "Agree", "Strongly agree"))
i1    <- sf_item("sat_1", "Item 1", type = "likert",
                 choice_set = "ag5", scale_id = "sat")
i2    <- sf_item("sat_2", "Item 2", type = "likert",
                 choice_set = "ag5", scale_id = "sat")
scale <- sf_scale("sat", "Satisfaction", items = c("sat_1", "sat_2"))
instr <- sf_instrument("Demo Survey", components = list(cs, i1, i2, scale))

cb <- codebook_report(instr)
print(cb)
nrow(cb$items_table)
nrow(cb$scales_table)
```

---

descriptives_report *Descriptive statistics report*

Description

Computes survey descriptives for numeric, Likert, and scale-score columns, including missingness, mean, standard deviation, median, IQR, range, skewness, kurtosis, standard error, and confidence intervals.

Usage

```
descriptives_report(
  data,
  variables = NULL,
  split_by = NULL,
  conf_level = 0.95,
  weights = NULL
)
```

Arguments

data	A data.frame of responses.
variables	Character vector of variables. When NULL, numeric-like columns are used.
split_by	Optional grouping variable.
conf_level	Confidence level for the mean interval.
weights	Optional case-weight column.

Value

An object of class sframe_descriptives_report.

---

efa_report *Prepare a survey instrument for exploratory factor analysis*

Description

Reports KMO sampling adequacy, Bartlett's test of sphericity, and a parallel analysis scree plot to inform factor number selection. The report prepares the researcher to estimate an EFA solution with a separate package such as psych or lavaan.

Usage

```
efa_report(
  data,
  instrument,
  scales = NULL,
  nfactors = NULL,
  rotation = "oblimin"
)
```

Arguments

data	A tibble or data.frame of responses.
instrument	An sframe object.
scales	Character vector or NULL. Scale IDs whose items to include. When NULL, all scale items are pooled.
nfactors	Integer or NULL. Suggested number of factors to highlight on the scree plot. When NULL, the parallel analysis recommendation is used.
rotation	Character. The rotation method to display in the diagnostic notes. Does not affect the diagnostics themselves. Defaults to "oblimin".

Value

An object of class sframe_efa_report with elements kmo, bartlett, parallel, and suggested_nfactors.

See Also

reliability_report(), cfa_syntax()

Examples

```
if (requireNamespace("psych", quietly = TRUE)) {
  demo <- sframe_demo_data()
  er <- efa_report(demo$responses, demo$instrument)
  print(er)
}
```

---

efa_solution *Estimate an exploratory factor solution*

Description

Runs psych::fa() on selected item columns and returns loadings, communalities, uniqueness, variance summaries, and simple item retention flags. The psych package is optional and is only required when this function is called.

Usage

```
efa_solution(
  data,
  instrument,
  items = NULL,
  scales = NULL,
  nfactors = 1L,
  extraction = c("minres", "pa", "ml"),
  rotation = c("oblimin", "promax", "varimax"),
```

```
  min_loading = 0.3,
  cross_loading = 0.3
)
```

Arguments

data	A data.frame of responses.
instrument	An sframe object.
items	Character vector of item IDs. When NULL, scale items are used.
scales	Optional scale IDs used to select item columns.
nfactors	Number of factors.
extraction	Extraction method passed to psych::fa().
rotation	Rotation method passed to psych::fa().
min_loading	Minimum salient loading.
cross_loading	Maximum secondary loading before a warning is raised.

Value

An object of class sframe_efa_solution.

---

efa_syntax *Generate EFA planning syntax*

Description

Generate EFA planning syntax

Usage

```
efa_syntax(
  items,
  nfactors = 1L,
  extraction = c("minres", "pa", "ml"),
  rotation = c("oblimin", "promax", "varimax"),
  data_name = "data"
)
```

Arguments

items	Character vector of item IDs.
nfactors	Number of factors.
extraction	Extraction method.
rotation	Rotation method.
data_name	Name of the data object in generated R code.

Value

A character string with R syntax.

`export_google_sheet` *Export a survey instrument to Google Sheets collection format*

Description

Generates a Google Apps Script file that, when run in a Google Sheet, creates a response collection endpoint for a survey instrument. The builder can store the deployed Apps Script URL in survey metadata, and the same sheet can be read back with `read_sheet_responses()`.

Usage

```
export_google_sheet(instrument, sheet_url, output_dir = ".")
```

Arguments

`instrument`	An `sframe` object.
`sheet_url`	Character. The URL of an existing Google Sheet. The sheet must be shared so that anyone with the link can edit, or use service account credentials via `googlesheets4`.
`output_dir`	Character. Directory to write the Apps Script file. Defaults to the current working directory.

Value

The path to the generated `.gs` Apps Script file, invisibly.

See Also

`read_sheet_responses()`, `read_responses()`, `write_sframe()`

Examples

```
instr <- read_sframe(
  system.file("extdata", "tourism_services_demo.sframe",
              package = "surveyframe")
)
script <- export_google_sheet(
  instr,
  sheet_url = "https://docs.google.com/spreadsheets/d/demo",
  output_dir = tempdir()
)
file.exists(script)
```

`export_static_survey` *Export a self-contained static HTML survey*

Description

Generates a single HTML file that presents the survey instrument in a browser without requiring a Shiny server or any internet connection. All thirteen item types, branching logic, required-field validation, and multi-page navigation are handled entirely in client-side JavaScript.

Usage

```
export_static_survey(
  instrument,
  output_path = NULL,
  open = interactive(),
  endpoint_url = NULL,
  overwrite = FALSE
)
```

Arguments

`instrument`	An `sframe` object.
`output_path`	Character. File path for the output HTML. When `NULL`, a `<survey_title>.html` file is written in `tempdir()`.
`open`	Logical. If `TRUE` (default) and the session is interactive, the file is opened in the default browser after writing.
`endpoint_url`	Character or NULL. A URL to which responses are POSTed as JSON on submission. When NULL, CSV download is the only collection mechanism.
`overwrite`	Logical. Whether to overwrite an existing file at `output_path`. Defaults to `FALSE`.

Details

When `output_path` is `NULL`, the file is written to `tempdir()`. Supply an explicit `output_path` for any production export that should be kept.

When a respondent clicks the submit button, the browser downloads a one-row CSV file named `<survey_title>_response_<id>.csv`. If `endpoint_url` is supplied, the same payload is also sent as a JSON POST request to that URL (for example a Google Apps Script web app or a serverless function). The two mechanisms are independent: the download happens regardless, so responses are never lost if the POST fails.

The exported file works offline. It can be hosted on GitHub Pages, Netlify, any static file server, or e-mailed as an attachment for opening directly from disk.

Value

The output path, invisibly.

See Also

launch_studio(), launch_builder(), render_survey()

Examples

```
cs    <- sf_choices("ag5", 1:5,
           c("Strongly disagree", "Disagree", "Neutral",
             "Agree", "Strongly agree"))
i1    <- sf_item("sat_1", "Overall I am satisfied with the service.",
                 type = "likert", choice_set = "ag5", required = TRUE)
i2    <- sf_item("comments", "Any additional comments?", type = "textarea")
instr <- sf_instrument("Customer Satisfaction Survey",
                       components = list(cs, i1, i2))

# Write to a temp file without opening the browser
out <- export_static_survey(instr,
                            output_path = file.path(tempdir(), "sat.html"),
                            open = FALSE)
file.exists(out)


# Write to a temp file and open in the default browser
export_static_survey(instr,
                     output_path = file.path(tempdir(), "sat_browser.html"),
                     overwrite = TRUE)

# Write with a Google Apps Script endpoint for server-side collection
export_static_survey(
  instr,
  output_path  = file.path(tempdir(), "sat_endpoint.html"),
  endpoint_url = "https://script.google.com/macros/s/XXXXX/exec",
  open         = FALSE,
  overwrite    = TRUE
)
```

---

format.sframe *Format an sframe instrument object as a string*

Description

Format an sframe instrument object as a string

Usage

```
## S3 method for class 'sframe'
format(x, ...)
```

Arguments

x	An object of class sframe.
...	Ignored. Present for S3 consistency.

Value

A single character string.

---

item_report *Generate item-level diagnostics*

Description

Produces item-total correlations, floor and ceiling effect proportions, and item means and standard deviations for each item within each scale.

Usage

```
item_report(data, instrument, scales = NULL)
```

Arguments

data	A tibble or data.frame of responses.
instrument	An sframe object.
scales	Character vector or NULL. A subset of scale IDs to analyse. When NULL (default), all scales are included.

Value

An object of class sframe_item_report, a list with one data.frame per scale.

See Also

reliability_report(), sf_scale()

Examples

```
demo <- sframe_demo_data()
ir <- item_report(demo$responses, demo$instrument)
print(ir)
```

launch_builder *Launch the surveyframe visual survey builder*

Description

Opens the SurveyBuilder, a self-contained HTML application for visual survey design. The builder runs client-side without an R session or Shiny server. Save instruments as `.sframe` files from the browser and load them into R with `read_sframe()`.

Usage

```
launch_builder(open = TRUE)
```

Arguments

`open`	Logical. When `TRUE` (the default), the builder HTML file is opened in the system's default web browser with `utils::browseURL()`. Set to `FALSE` to return the file path without opening it, which is useful for automated testing.

Details

The builder includes a three-mode interface.

Build An item editor with a persistent inspector panel, drag-to-reorder, undo/redo, and autosave to browser localStorage.

Preview A full live render of the survey showing welcome, body, and thank-you pages.

Analyse A role-based analysis planner with method-specific options, planned outputs, reporting references, and decision rules.

The builder includes a pure-JavaScript SHA-256 fallback for browsers or security policies where `crypto.subtle` is unavailable on `file://` origins. Saved `.sframe` files can be loaded and validated with `read_sframe()`.

Value

The path to the bundled builder HTML file, invisibly.

See Also

`launch_studio()`, `read_sframe()`, `run_analysis_plan()`

Examples

```
# Retrieve the builder path for inspection without opening the browser
path <- launch_builder(open = FALSE)
file.exists(path)
```

launch_builder_demo *Launch SurveyBuilder with the bundled input-types demo preloaded*

Description

Opens a temporary copy of the SurveyBuilder with the bundled input-types instrument already injected into the JavaScript state. The demo questions, scales, and analysis plan are visible immediately — no manual file-load step is required.

Usage

```
launch_builder_demo(open = TRUE)
```

Arguments

open	Logical. When TRUE (the default), the pre-populated builder HTML is opened in the system's default web browser.

Value

Invisibly returns a list with builder_path, demo_file, and responses_path.

launch_dashboard *Launch the interactive response dashboard*

Description

Opens a Shiny dashboard to explore collected response data alongside the instrument definition. Use this interface after response collection for analysis and quality control. Use launch_builder() to design new questionnaires. The dashboard includes five panels:

Usage

```
launch_dashboard(
  instrument = NULL,
  responses = NULL,
  port = NULL,
  host = "127.0.0.1",
  launch.browser = interactive()
)
```

Arguments

`instrument`	An `sframe` object. When `NULL`, the bundled tourism services demo instrument is loaded.
`responses`	A `data.frame` or `tibble` of survey responses, as produced by `read_responses()` or `read_sheet_responses()`. When NULL and `instrument` is also NULL, the bundled simulated demo responses are loaded. When NULL with a user-supplied instrument, the dashboard opens with instrument metadata and no response summaries.
`port`	Integer or NULL. TCP port for the Shiny server. When NULL, Shiny selects an available port automatically.
`host`	Character. Host address passed to `shiny::runApp()`. Defaults to `"127.0.0.1"`.
`launch.browser`	Logical. Whether to open the dashboard in the default browser automatically. Defaults to `TRUE` in interactive sessions.

Details

Overview Response count, date range, and instrument metadata.

Items Per-item frequency bar charts, histograms, and tabulated frequency counts for choice-type questions.

Scales Scale score distributions with mean overlay, and a summary table of scale definitions.

Quality Attention check pass rates for each check defined in the instrument.

Raw data Scrollable response table with a CSV download button.

The dashboard is read-only. Use it for descriptive exploration after collecting responses and before running formal analysis with `run_analysis_plan()`.

Value

Called for its side effect. Returns nothing.

See Also

`run_analysis_plan()`, `quality_report()`, `score_scales()`

Examples

```
# Open the bundled tourism-services response dashboard.
# To build a questionnaire instead, use launch_builder().
launch_dashboard()

# Open the dashboard with your own instrument and responses
instr <- read_sframe(
  system.file("extdata", "tourism_services_demo.sframe",
              package = "surveyframe")
)
responses <- read_responses(
  system.file("extdata", "tourism_services_responses.csv",
              package = "surveyframe"),
```

```
  instr,
  respondent_id = "respondent_id",
  submitted_at = "submitted_at",
  meta_cols = "started_at"
)
launch_dashboard(instr, responses)
```

`launch_dashboard_demo` *Launch the response dashboard with the bundled input-types demo*

Description

Opens the dashboard with the bundled input-types questionnaire and 120 simulated responses already loaded. The browser is opened automatically by default.

Usage

```
launch_dashboard_demo(port = NULL, host = "127.0.0.1", launch.browser = TRUE)
```

Arguments

`port`	TCP port for the Shiny server.
`host`	Host address for the Shiny server.
`launch.browser`	Whether to open the browser automatically. Defaults to `TRUE` for this demo helper.

Value

Called for its side effect.

`launch_studio` *Launch the SurveyStudio interface*

Description

Opens the SurveyStudio Shiny application, a visual interface for the complete surveyframe workflow. The studio includes screens to build a survey draft, open an existing instrument, preview the survey, upload responses, review data quality, inspect reliability, plan analyses, and export outputs.

Usage

```
launch_studio(
  instrument = NULL,
  responses = NULL,
  respondent_id = NULL,
  submitted_at = NULL,
  meta_cols = NULL,
  strict = TRUE,
  screen = c("auto", "build", "preview", "data", "quality", "analysis", "dashboard"),
  port = NULL,
  host = "127.0.0.1",
  launch.browser = interactive()
)
```

Arguments

instrument	An sframe object or NULL.
responses	A data.frame, tibble, CSV file path, or NULL.
respondent_id	Character or NULL. Response ID column when responses is a CSV path.
submitted_at	Character or NULL. Submission time column when responses is a CSV path.
meta_cols	Character vector or NULL. Metadata columns when responses is a CSV path.
strict	Logical. Passed to read_responses() when responses is a CSV path.
screen	Initial studio screen. One of "auto", "build", "preview", "data", "quality", "analysis", or "dashboard".
port	TCP port for the Shiny server.
host	Host address passed to shiny::runApp().
launch.browser	Whether to open the browser automatically.

Value

Called for its side effect.

See Also

launch_builder(), launch_dashboard(), read_sframe(), read_responses()

Examples

```
launch_studio()

demo <- sframe_demo_data()
launch_studio(instrument = demo$instrument, launch.browser = FALSE)

launch_studio(
  instrument    = demo$instrument,
  responses     = demo$responses,
  respondent_id = "respondent_id",
```

```
  submitted_at  = "submitted_at"
)
```

`launch_studio_demo` *Launch SurveyStudio with the bundled input-types demo*

Description

Opens SurveyStudio with the bundled input-types questionnaire and simulated response data already loaded. The browser is opened automatically by default.

Usage

```
launch_studio_demo(
  screen = "preview",
  port = NULL,
  host = "127.0.0.1",
  launch.browser = TRUE
)
```

Arguments

`screen`	Initial studio screen. Defaults to `"preview"` so the demo content is immediately visible.
`port`	TCP port for the Shiny server.
`host`	Host address for the Shiny server.
`launch.browser`	Whether to open the browser automatically. Defaults to `TRUE` for this demo helper.

Value

Called for its side effect.

`missing_data_report` *Missing-data report*

Description

Reports item-wise missingness, respondent-wise missingness, missing-data patterns, listwise and pairwise deletion counts, and scale scoring missing rules. No imputation is performed.

Usage

```
missing_data_report(data, instrument = NULL, variables = NULL)
```

Arguments

`data`	A data.frame of responses.
`instrument`	Optional `sframe` object.
`variables`	Optional response columns. Defaults to instrument item IDs when an instrument is supplied, otherwise all columns.

Value

An object of class `sframe_missing_data_report`.

`model_json` *Serialise a model specification to JSON*

Description

Serialise a model specification to JSON

Usage

```
model_json(model, pretty = TRUE)
```

Arguments

`model`	An `sf_model()` object.
`pretty`	Logical. Whether to pretty-print the JSON.

Value

A JSON string.

`model_report_template` *Create a model reporting template*

Description

Create a model reporting template

Usage

```
model_report_template(model, include_json = TRUE)
```

Arguments

`model`	An `sf_model()` object.
`include_json`	Logical. Whether to include the JSON schema block.

Value

A character string.

`outlier_report` *Flag univariate and multivariate outliers*

Description

Uses transparent screening rules for numeric survey response variables. The report supports data review before modelling, not automatic deletion.

Usage

```
outlier_report(
  data,
  variables = NULL,
  method = c("zscore", "iqr", "mahalanobis"),
  z_cut = 3,
  iqr_multiplier = 1.5,
  p_cut = 0.975
)
```

Arguments

`data`	A data.frame.
`variables`	Character vector of numeric variables to screen. When `NULL`, all numeric columns are used.
`method`	Outlier rule. `"zscore"` flags absolute z scores above `z_cut`; `"iqr"` flags values outside Tukey fences; `"mahalanobis"` flags rows above the chi-square cutoff for the selected variables.
`z_cut`	Numeric cutoff for `"zscore"`. Defaults to `3`.
`iqr_multiplier`	Numeric multiplier for `"iqr"` fences. Defaults to `1.5`.
`p_cut`	Probability cutoff for `"mahalanobis"`. Defaults to `0.975`.

Value

An object of class `sframe_outlier_report` with the method, screened variables, a result table, flagged row numbers, and a reporting prompt.

Examples

```
demo <- sframe_demo_data()
outliers <- outlier_report(
  demo$responses,
  variables = c("dm_1", "dm_2", "sat_1"),
  method = "zscore"
)
outliers$flagged_rows
```

posthoc_report *Post-hoc and pairwise comparison report*

Description

Post-hoc and pairwise comparison report

Usage

```
posthoc_report(
  data,
  method = c("anova", "kruskal_wallis", "chi_square", "cochran_q"),
  outcome = NULL,
  group = NULL,
  table_vars = NULL,
  measures = NULL,
  correction = c("holm", "bonferroni", "BH")
)
```

Arguments

data	A data.frame.
method	Comparison family. Supports "anova", "kruskal_wallis", "chi_square", and "cochran_q".
outcome	Outcome variable for group comparisons.
group	Grouping variable for group comparisons.
table_vars	Two categorical variables for chi-square residuals and pairwise proportion tests.
measures	Repeated binary measures for pairwise McNemar tests.
correction	Multiple-comparison correction.

Value

An object of class sframe_posthoc_report.

print.sframe *Print an sframe instrument object*

Description

Displays a compact summary of an sframe instrument object, showing the title, version, item count, scale count, and validation status.

Usage

```
## S3 method for class 'sframe'
print(x, ...)
```

Arguments

x	An object of class sframe.
...	Ignored. Present for S3 consistency.

Value

x, invisibly.

Examples

```
item <- sf_item("q1", "How satisfied are you?", type = "likert",
                choice_set = "agree5")
instr <- sf_instrument("My Survey", components = list(item))
print(instr)
```

---

quality_report *Generate a data quality report for survey responses*

Description

Evaluates collected response data against the instrument specification and produces a structured quality report. The report covers attention check performance, completion time, straight-lining within scale blocks, item-level missingness, respondent-level missingness, and duplicate respondent IDs where supplied.

Usage

```
quality_report(
  data,
  instrument,
  respondent_id = NULL,
  submitted_at = NULL,
  started_at = NULL,
  time_min = NULL,
  straightline_scales = TRUE,
  missing_threshold = 0.2
)
```

Arguments

data	A tibble or data.frame of responses, typically produced by read_responses().
instrument	An sframe object created by sf_instrument().
respondent_id	Character or NULL. The column name holding unique respondent identifiers. Used for duplicate detection.
submitted_at	Character or NULL. The column name holding submission timestamps. Used for completion time analysis.
started_at	Character or NULL. The column name holding survey start timestamps. When NULL, quality_report() looks for a recognised start-time column automatically.
time_min	Numeric or NULL. Minimum acceptable completion time in seconds. Respondents with a submission time below this threshold are flagged as speeders when timing data are available.
straightline_scales	Logical. Whether to check for straight-lining within each defined scale block. Defaults to TRUE.
missing_threshold	Numeric. The proportion of missing item responses above which a respondent is flagged. Defaults to 0.2.

Details

Timing analysis is available when the data contain a submission timestamp column and either an explicit started_at column or one of the recognised defaults: started_at, start_time, started, or .started_at.

Value

An object of class sframe_quality_report, a named list with elements: summary, attention, timing, straightline, missing, and duplicates. Use print() for a formatted summary.

See Also

sf_check(), read_responses(), score_scales()

Examples

```
instr <- read_sframe(
  system.file("extdata", "tourism_services_demo.sframe",
              package = "surveyframe")
)
responses <- read_responses(
  system.file("extdata", "tourism_services_responses.csv",
              package = "surveyframe"),
  instr,
  respondent_id = "respondent_id",
  submitted_at = "submitted_at",
  meta_cols = "started_at"
```

```
)
qr <- quality_report(
  responses,
  instr,
  respondent_id = "respondent_id",
  submitted_at = "submitted_at",
  started_at = "started_at",
  straightline_scales = FALSE
)
print(qr)
```

`read_responses` *Read and validate survey responses*

Description

Loads survey response data and checks that it conforms to the instrument specification. Column names in the response file must match item IDs defined in the instrument. Non-item columns are allowed only when declared through `respondent_id`, `submitted_at`, or `meta_cols`.

Usage

```
read_responses(
  x,
  instrument,
  respondent_id = NULL,
  submitted_at = NULL,
  meta_cols = NULL,
  strict = TRUE
)
```

Arguments

`x`	A file path to a CSV file, a `data.frame`, or a `tibble`.
`instrument`	An `sframe` object created by `sf_instrument()`.
`respondent_id`	Character or NULL. The name of the column containing unique respondent identifiers. If NULL, no respondent ID column is expected.
`submitted_at`	Character or NULL. The name of the column containing submission timestamps.
`meta_cols`	Character vector or NULL. Additional column names that are not item IDs but should be retained (for example, condition assignment or source URL).
`strict`	Logical. When `TRUE` (default), columns in the response data outside the declared item IDs and metadata columns raise an error. When `FALSE`, undeclared columns are retained with a warning.

Value

A `data.frame` with columns ordered as: metadata columns first, then item columns in instrument order. Unrecognised columns are dropped when `strict = TRUE` or appended with a warning when `strict = FALSE`.

See Also

`quality_report()`, `score_scales()`

Examples

```
responses <- read_responses(
  x = system.file("extdata", "tourism_services_responses.csv",
                  package = "surveyframe"),
  instrument = read_sframe(
    system.file("extdata", "tourism_services_demo.sframe",
                package = "surveyframe")
  ),
  respondent_id = "respondent_id",
  submitted_at = "submitted_at",
  meta_cols = "started_at"
)
head(responses[, c("respondent_id", "visit_type", "dm_1")])
```

`read_sframe` *Read an instrument from a .sframe file*

Description

Reads a `.sframe` JSON file and reconstructs an `sframe` instrument object. The SHA-256 integrity hash is verified on load unless `validate = FALSE`.

Usage

```
read_sframe(path, validate = TRUE)
```

Arguments

`path`	Character. The path to a `.sframe` file.
`validate`	Logical. Whether to validate the loaded instrument with `validate_sframe()`. Defaults to `TRUE`.

Value

An `sframe` object.

See Also

`write_sframe()`, `validate_sframe()`

Examples

```
instr <- read_sframe(
  system.file("extdata", "tourism_services_demo.sframe",
              package = "surveyframe")
)
print(instr)
```

`read_sheet_responses` *Read survey responses from a Google Sheet*

Description

Reads response data collected by the surveyframe Google Apps Script endpoint and returns a validated data frame ready for the surveyframe analysis pipeline.

Usage

```
read_sheet_responses(
  sheet_id,
  instrument,
  sheet_name = "Responses",
  respondent_id = "respondent_id",
  submitted_at = "submitted_at"
)
```

Arguments

`sheet_id`	Character. The Google Sheet ID or full URL.
`instrument`	An `sframe` object.
`sheet_name`	Character. The name of the sheet tab holding responses. Defaults to `"Responses"`.
`respondent_id`	Character or NULL. Column holding respondent IDs. Defaults to `"respondent_id"`.
`submitted_at`	Character or NULL. Column holding submission timestamps. Defaults to `"submitted_at"`.

Value

A `data.frame` validated against the instrument, ready for `quality_report()`, `score_scales()`, and `reliability_report()`.

See Also

`export_google_sheet()`, `read_responses()`, `quality_report()`

Examples

```
## Not run:
responses <- read_sheet_responses(
  sheet_id   = "your-sheet-id",
  instrument = instr
)
qr <- quality_report(responses, instr, respondent_id = "respondent_id")

## End(Not run)
```

`reliability_report` *Compute reliability statistics for scored scales*

Description

Produces Cronbach's alpha and McDonald's omega for each scale defined in the instrument, along with the number of items and sample size.

Usage

```
reliability_report(data, instrument, scales = NULL, alpha = TRUE, omega = TRUE)
```

Arguments

`data`	A `tibble` or `data.frame` of responses. Item columns must be present.
`instrument`	An `sframe` object.
`scales`	Character vector or NULL. A subset of scale IDs to analyse. When NULL (default), all scales in the instrument are included.
`alpha`	Logical. Whether to compute Cronbach's alpha. Defaults to `TRUE`.
`omega`	Logical. Whether to compute McDonald's omega. Defaults to `TRUE`.

Value

An object of class `sframe_reliability_report`, a list with one element per scale. Each element is a list of statistics and a summary tibble.

See Also

`sf_scale()`, `item_report()`

Examples

```
if (requireNamespace("psych", quietly = TRUE)) {
  demo <- sframe_demo_data()
  rr <- reliability_report(demo$responses, demo$instrument, omega = FALSE)
  print(rr)
}
```

`render_report` *Render a reproducible survey report*

Description

Generates an HTML report that includes the instrument codebook, data quality summary, reliability diagnostics, and analysis-plan content. When Quarto and the bundled template are available, the report is rendered through Quarto. Otherwise, surveyframe writes an internal HTML fallback so the reporting workflow still runs on machines without Quarto.

Usage

```
render_report(
  instrument,
  data = NULL,
  output_file = NULL,
  output_path = NULL,
  format = c("html"),
  include_quality = TRUE,
  include_reliability = TRUE,
  include_codebook = TRUE,
  include_missing = TRUE,
  include_descriptives = TRUE,
  include_analysis = TRUE,
  include_models = TRUE
)
```

Arguments

`instrument`	An `sframe` object.
`data`	A `tibble` or `data.frame` of responses, or NULL to generate a codebook-only report.
`output_file`	Character or NULL. The output file path. When NULL, a temporary file is written and its path returned.
`output_path`	Character or NULL. Alias for `output_file`. If both are supplied, `output_file` takes precedence.
`format`	Character. Output format. Currently `"html"`.
`include_quality`	Logical. Whether to include the data quality report. Requires `data`. Defaults to `TRUE`.
`include_reliability`	Logical. Whether to include reliability diagnostics. Requires `data`. Defaults to `TRUE`.
`include_codebook`	Logical. Whether to include the instrument codebook. Defaults to `TRUE`.

include_missing
: Logical. Whether to include the missing-data report. Requires `data`. Defaults to `TRUE`.

include_descriptives
: Logical. Whether to include descriptive statistics. Requires `data`. Defaults to `TRUE`.

include_analysis
: Logical. Whether to include analysis-plan results when `data` are supplied and the instrument has an `analysis_plan`.

include_models
: Logical. Whether to include saved model JSON and generated syntax blocks. Defaults to `TRUE`.

Value

The output file path, invisibly.

See Also

`codebook_report()`, `quality_report()`, `reliability_report()`

Examples

```
instr <- read_sframe(
  system.file("extdata", "tourism_services_demo.sframe",
              package = "surveyframe")
)
responses <- read_responses(
  system.file("extdata", "tourism_services_responses.csv",
              package = "surveyframe"),
  instr,
  respondent_id = "respondent_id",
  submitted_at = "submitted_at",
  meta_cols = "started_at"
)
old <- options(surveyframe.use_quarto = FALSE)
out <- tryCatch(
  render_report(
    instr,
    data = responses,
    output_file = tempfile(fileext = ".html"),
    include_reliability = FALSE,
    include_analysis = FALSE
  ),
  finally = options(old)
)
file.exists(out)
```

`render_results` *Render analysis results to a formatted HTML report*

Description

Generates a self-contained HTML report from the output of `run_analysis_plan()`. Each section corresponds to one research question and includes the APA-formatted statistical result, an interpretation space, and a reference list.

Usage

```
render_results(
  results = NULL,
  instrument,
  output_file = NULL,
  output_path = NULL,
  citation_format = c("apa", "ama", "vancouver"),
  title = NULL
)
```

Arguments

`results`	An `sframe_analysis_results` object from `run_analysis_plan()`.
`instrument`	An `sframe` object.
`output_file`	Character or NULL. Path to the output HTML file. When NULL, a temporary file is written and its path returned.
`output_path`	Character or NULL. Alias for `output_file`.
`citation_format`	Character. Reference format. One of `"apa"`, `"ama"`, or `"vancouver"`. Defaults to `"apa"`.
`title`	Character or NULL. Report title. Defaults to the instrument title with " – Results" appended.

Value

The output file path, invisibly.

See Also

`run_analysis_plan()`, `render_report()`

Examples

```
instr <- read_sframe(
  system.file("extdata", "tourism_services_demo.sframe",
              package = "surveyframe")
)
responses <- read_responses(
  system.file("extdata", "tourism_services_responses.csv",
              package = "surveyframe"),
  instr,
  respondent_id = "respondent_id",
  submitted_at = "submitted_at",
  meta_cols = "started_at"
)
results <- run_analysis_plan(responses, instr)
out <- render_results(results, instr,
                      output_file = tempfile(fileext = ".html"))
file.exists(out)
```

`render_survey` *Render a survey from an instrument object*

Description

Launches a Shiny survey with a welcome page, configurable header, all item types, branching logic, required-field enforcement, progress tracking, standard and conversational (one-question-at-a-time) display modes, and a customisable thank-you page. Responses can be persisted to CSV or passed to a callback.

Usage

```
render_survey(
  instrument,
  mode = c("shiny"),
  title = NULL,
  theme = NULL,
  save_responses = c("none", "csv"),
  output_path = NULL,
  on_submit = NULL
)
```

Arguments

`instrument`	An `sframe` object.
`mode`	Character. Deployment mode. Currently `"shiny"`.
`title`	Character or NULL. Override for the survey title.
`theme`	Character or NULL. Hex colour for the survey theme.
`save_responses`	Character. `"none"` (default) or `"csv"`.

output_path	Character or NULL. CSV path when save_responses = "csv".
on_submit	Function or NULL. Callback receiving the submitted row.

Value

A shiny.appobj.

See Also

launch_studio(), read_responses()

Examples

```
render_survey(instr)
render_survey(instr, save_responses = "csv", output_path = "responses.csv")
```

---

run_analysis_plan *Run a pre-planned analysis from an instrument's analysis plan*

Description

Executes every analysis block defined in the instrument's analysis_plan slot against the supplied response data. Each block corresponds to one research question defined during instrument design in the SurveyBuilder. Results include APA-formatted statistics, effect sizes, interpretation prompts, and reporting references.

Usage

```
run_analysis_plan(data, instrument, scored = TRUE)
```

Arguments

data	A tibble or data.frame of responses, typically produced by read_responses() or read_sheet_responses().
instrument	An sframe object containing an analysis_plan.
scored	Logical. Whether to automatically score scales before running the analysis. Defaults to TRUE.

Value

An object of class sframe_analysis_results, a list with one element per analysis block. Each element contains the test result, APA string, interpretation prompt, and reporting-reference metadata. Pass to render_results() to generate a formatted report.

See Also

render_results(), read_sheet_responses()

Examples

```
instr <- read_sframe(
  system.file("extdata", "tourism_services_demo.sframe",
              package = "surveyframe")
)
responses <- read_responses(
  system.file("extdata", "tourism_services_responses.csv",
              package = "surveyframe"),
  instr,
  respondent_id = "respondent_id",
  submitted_at = "submitted_at",
  meta_cols = "started_at"
)
results <- run_analysis_plan(responses, instr)
print(results)
```

---

sample_size_plan *Sample-size and power planning helper*

Description

Sample-size and power planning helper

Usage

```
sample_size_plan(
 type = c("proportion", "mean", "correlation", "t_test", "anova", "regression", "sem"),
  margin_error = NULL,
  sd = NULL,
  p = 0.5,
  r = NULL,
  alpha = 0.05,
  power = 0.8,
  groups = 2L,
  predictors = NULL
)
```

Arguments

type	Planning target: "proportion", "mean", "correlation", "t_test", "anova", "regression", or "sem".
margin_error	Margin of error for mean/proportion planning.
sd	Standard deviation for mean planning.
p	Expected proportion.
r	Expected correlation.
alpha	Significance level.

power	Desired power.
groups	Number of groups for ANOVA/t-test planning.
predictors	Number of predictors for regression planning.

Value

A list of planning estimates and warnings.

`score_scales` *Score defined scales from survey responses*

Description

Applies scale scoring rules from the instrument to response data. Handles reverse coding, optional weighted composite score computation, and minimum valid item thresholds. Returns a data frame with one scored column per scale.

Usage

```
score_scales(data, instrument, keep_items = TRUE, keep_meta = TRUE)
```

Arguments

data	A `tibble` or `data.frame` of responses.
instrument	An `sframe` object.
keep_items	Logical. Whether to retain individual item columns in the output. Defaults to `TRUE`.
keep_meta	Logical. Whether to retain non-item columns (metadata) in the output. Defaults to `TRUE`.

Value

A `data.frame` with scored scale columns appended. Scale columns are named using the scale `id`.

See Also

`sf_scale()`, `reliability_report()`

Examples

```
cs    <- sf_choices("ag5", 1:5,
           c("Strongly disagree", "Disagree", "Neutral",
             "Agree", "Strongly agree"))
i1    <- sf_item("sat_1", "Item 1", type = "likert",
                 choice_set = "ag5", scale_id = "sat")
i2    <- sf_item("sat_2", "Item 2", type = "likert",
                 choice_set = "ag5", scale_id = "sat")
```

```
i3    <- sf_item("sat_3", "Item 3 (reverse)", type = "likert",
                 choice_set = "ag5", scale_id = "sat", reverse = TRUE)
scale <- sf_scale("sat", "Satisfaction",
                  items = c("sat_1", "sat_2", "sat_3"), min_valid = 2L)
instr <- sf_instrument("Demo", components = list(cs, i1, i2, i3, scale))

responses <- data.frame(
  sat_1 = c(4, 5, 3),
  sat_2 = c(4, 4, 3),
  sat_3 = c(2, 1, 3),
  stringsAsFactors = FALSE
)

scored <- score_scales(responses, instr)
scored$sat
```

`seminr_syntax` *Generate seminr PLS-SEM syntax*

Description

Generate seminr PLS-SEM syntax

Usage

```
seminr_syntax(model, data_name = "data", nboot = NULL, seed = 123)
```

Arguments

`model`	An `sf_model()` object of type `"pls_sem"`.
`data_name`	Name of the data object in generated R code.
`nboot`	Number of bootstrap samples.
`seed`	Random seed for bootstrap syntax.

Value

An R syntax string for `seminr`.

sem_lavaan_syntax *Generate lavaan CB-SEM syntax*

Description

Generate lavaan CB-SEM syntax

Usage

```
sem_lavaan_syntax(model, instrument = NULL, standardised = TRUE)
```

Arguments

model	An sf_model() object of type "cb_sem".
instrument	Optional sframe object for indicator validation.
standardised	Logical. Adds a standardised-estimates fitting note.

Value

A lavaan syntax string.

sframe_builder_empty_state *Create an empty SurveyStudio builder state*

Description

Create an empty SurveyStudio builder state

Usage

```
sframe_builder_empty_state()
```

Value

A list containing empty metadata, choice, item, scale, branching, and check collections suitable for SurveyStudio.

sframe_builder_state_from_instrument

Convert an instrument into a SurveyStudio builder state

Description

Convert an instrument into a SurveyStudio builder state

Usage

```
sframe_builder_state_from_instrument(instrument = NULL)
```

Arguments

instrument	An sframe object or NULL.

Value

A builder state list. Component classes are restored so the state can be edited or validated by SurveyStudio.

sframe_builder_validate_draft

Validate a SurveyStudio draft state

Description

Validate a SurveyStudio draft state

Usage

```
sframe_builder_validate_draft(
  meta,
  choices = list(),
  items = list(),
  scales = list(),
  branching = list(),
  checks = list(),
  analysis_plan = list(),
  models = list()
)
```

Arguments

meta	List of instrument metadata.
choices, items, scales, branching, checks	Lists of draft components.
analysis_plan	List of draft analysis-plan blocks.
models	List of draft model specifications.

Value

A list with valid, problems, and instrument.

---

sframe_demo_data *Load bundled surveyframe demo data*

Description

Loads the bundled tourism-services .sframe instrument and simulated response dataset used in package examples and statistical workflow demos.

Usage

```
sframe_demo_data()
```

Value

A list with instrument, responses, instrument_path, and responses_path.

---

sframe_input_types_demo_data *Load bundled input-types demo data*

Description

Loads the bundled .sframe instrument and simulated response dataset that cover all main survey input types supported by surveyframe.

Usage

```
sframe_input_types_demo_data()
```

Value

A list with instrument, responses, instrument_path, and responses_path.

`sf_branch` *Define a branching rule*

Description

Creates a single-condition branching rule that shows or hides a survey item depending on the value of a preceding item. In v0.1, only single-condition rules are supported. Multi-condition AND/OR logic is planned for a later release.

Usage

```
sf_branch(
  item_id,
  depends_on,
  operator = c("==", "!=", "%in%", ">", ">=", "<", "<="),
  value,
  action = c("show", "hide")
)
```

Arguments

`item_id`	Character. The `id` of the item whose visibility this rule controls.
`depends_on`	Character. The `id` of the item whose response value triggers this rule.
`operator`	Character. The comparison operator. One of `"=="`, `"!="`, `"%in%"`, `">"`, `">="`, `"<"`, or `"<="`.
`value`	The value to compare against the response to `depends_on`. For `"%in%"`, supply a character or numeric vector.
`action`	Character. What to do when the condition is met. Either `"show"` (default) or `"hide"`.

Value

An object of class `sf_branch` (a named list).

See Also

`sf_instrument()`, `validate_sframe()`

Examples

```
# Show an open-text follow-up only when the respondent selects "Other"
rule <- sf_branch(
  item_id    = "gender_other",
  depends_on = "gender",
  operator   = "==",
  value      = "other",
  action     = "show"
)
```

sf_check *Define a design-time survey check*

Description

Specifies an attention, instructional, or trap check item at instrument design time. The check is stored in the instrument object and evaluated against collected response data by `quality_report()`. This function only defines the check. Evaluation happens later in `quality_report()`.

Usage

```
sf_check(
  id,
  item_id,
  type = c("attention", "instructional", "trap"),
  pass_values = NULL,
  fail_action = c("flag", "exclude"),
  label = NULL,
  notes = NULL
)
```

Arguments

`id`	Character. A unique identifier for this check.
`item_id`	Character. The `id` of the item used as the check. The item must be defined separately with `sf_item()` and included in the same instrument.
`type`	Character. The check type. One of: • `"attention"`: the item has a stated correct answer and flags respondents who answer incorrectly. • `"instructional"`: a manipulation check item used to test whether instructions were followed. • `"trap"`: an item designed to be selected only by inattentive respondents (e.g. "Please select Strongly agree for this item.").
`pass_values`	Vector or NULL. The response value or values that constitute a pass. For `"attention"` and `"instructional"` types, at least one value should be supplied. For `"trap"` types, this is the value that should NOT be selected.
`fail_action`	Character. What `quality_report()` does with respondents who fail this check. Either `"flag"` (mark in the report but retain) or `"exclude"` (mark for exclusion).
`label`	Character or NULL. An optional human-readable label for the check, used in the quality report.
`notes`	Character or NULL. Optional free-text notes about the purpose or rationale of this check.

Value

An object of class `sf_check` (a named list).

See Also

sf_item(), sf_instrument(), quality_report()

Examples

```
# An attention check: respondent must select 4
chk <- sf_check(
  id          = "attn_1",
  item_id     = "attention_check_q",
  type        = "attention",
  pass_values = 4,
  fail_action = "flag",
  label       = "Attention check 1"
)
```

---

sf_choices *Define a reusable choice set*

Description

Creates a named set of response options that can be referenced by one or more items. Defining choices once and referencing them by `id` keeps the instrument consistent and reduces the risk of label mismatches across items that share the same response format.

Usage

```
sf_choices(id, values, labels, allow_other = FALSE, randomise = FALSE)
```

Arguments

`id`	Character. A unique identifier for this choice set. Referenced in the `choice_set` argument of `sf_item()`.
`values`	Character or numeric vector. The stored values corresponding to each response option. Must have the same length as `labels`.
`labels`	Character vector. The display labels shown to respondents. Must have the same length as `values`.
`allow_other`	Logical. Whether to append an open-text "Other" option at the end of the choice list. Defaults to `FALSE`.
`randomise`	Logical. Whether to randomise the display order of options at render time. Defaults to `FALSE`.

Value

An object of class `sf_choices` (a named list).

See Also

sf_item(), sf_instrument()

Examples

```
# A five-point agreement scale
agree5 <- sf_choices(
  id     = "agree5",
  values = 1:5,
  labels = c("Strongly disagree", "Disagree", "Neutral",
             "Agree", "Strongly agree")
)

# A yes/no set
yn <- sf_choices(
  id     = "yn",
  values = c("yes", "no"),
  labels = c("Yes", "No")
)
```

`sf_construct` *Define a latent or composite construct*

Description

Define a latent or composite construct

Usage

```
sf_construct(
  id,
  label = NULL,
  items = character(0),
  mode = c("reflective", "composite", "formative", "single_item"),
  weights = NULL
)
```

Arguments

`id`	Construct identifier. Must start with a letter and contain only letters, numbers, and _ characters.
`label`	Human-readable construct label.
`items`	Character vector of indicator item IDs.
`mode`	Measurement mode. One of `"reflective"`, `"composite"`, `"formative"`, or `"single_item"`.
`weights`	Optional indicator weights for later PLS-SEM planning.

Value

An object of class `sf_construct`.

sf_covariance *Define a covariance between constructs*

Description

Define a covariance between constructs

Usage

```
sf_covariance(from, to, label = NULL)
```

Arguments

from	First construct ID.
to	Second construct ID.
label	Optional label.

Value

An object of class sf_covariance.

sf_indirect *Define an indirect effect path*

Description

Define an indirect effect path

Usage

```
sf_indirect(from, through, to, label = NULL)
```

Arguments

from	Source construct ID.
through	Character vector of mediator construct IDs.
to	Target construct ID.
label	Optional effect label.

Value

An object of class sf_indirect.

`sf_instrument` *Create a survey instrument object*

Description

Assembles a survey instrument from its component objects. This is the top-level constructor for the `sframe` class. All other constructors (`sf_item()`, `sf_choices()`, `sf_scale()`, `sf_branch()`, `sf_check()`) produce components that are passed into this function via `components`.

Usage

```
sf_instrument(
  title,
  version = "0.1.0",
  description = NULL,
  authors = NULL,
  languages = "en",
  components = list(),
  render = NULL,
  analysis_plan = list(),
  models = list()
)
```

Arguments

`title`	Character. The title of the survey instrument.
`version`	Character. A semantic version string. Defaults to `"0.1.0"`.
`description`	Character or NULL. A brief description of the instrument and its intended population or purpose.
`authors`	Character vector or NULL. Author names, used in codebooks and reports.
`languages`	Character vector. Language codes for the instrument. Defaults to `"en"`. Multi-language support is planned for a later release.
`components`	List. A list of component objects created by the constructor family: `sf_item()`, `sf_choices()`, `sf_scale()`, `sf_branch()`, and `sf_check()`. Components are sorted by class automatically. Supply components created by the surveyframe constructors.
`render`	List or NULL. Optional rendering hints passed to `render_survey()`, such as theme colour or progress bar visibility.
`analysis_plan`	List. Optional pre-planned analysis blocks created in the HTML SurveyBuilder Analyse mode.
`models`	List. Optional model specifications created with `sf_model()` or imported from a `.sframe` file.

Value

An object of class `sframe` with slots `meta`, `items`, `choices`, `scales`, `branching`, `checks`, `analysis_plan`, `models`, and `render`.

See Also

`sf_item()`, `sf_choices()`, `sf_scale()`, `sf_branch()`, `sf_check()`, `validate_sframe()`, `write_sframe()`

Examples

```
choices <- sf_choices("agree5", 1:5,
  c("Strongly disagree", "Disagree", "Neutral", "Agree", "Strongly agree"))

item1 <- sf_item("sat_1", "The service met my expectations.",
                 type = "likert", choice_set = "agree5",
                 scale_id = "sat", required = TRUE)
item2 <- sf_item("sat_2", "I would recommend this service.",
                 type = "likert", choice_set = "agree5",
                 scale_id = "sat", required = TRUE)

scale <- sf_scale("sat", "Satisfaction", items = c("sat_1", "sat_2"))

instr <- sf_instrument(
  title      = "Service Quality Survey",
  version    = "1.0.0",
  components = list(choices, item1, item2, scale)
)
print(instr)
```

`sf_item` *Define a survey item*

Description

Creates a single survey item object for inclusion in an `sframe` instrument. Items are the atomic units of a survey instrument. Every item must have a unique `id` within the instrument it is added to.

Usage

```
sf_item(
  id,
  label,
 type = c("likert", "single_choice", "multiple_choice", "numeric", "text", "textarea",
  "date", "matrix", "slider", "ranking", "rating", "section_break", "text_block"),
  required = FALSE,
  choice_set = NULL,
  scale_id = NULL,
  reverse = FALSE,
```

```
  help = NULL,
  placeholder = NULL,
  matrix_items = NULL,
  slider_min = NULL,
  slider_max = NULL,
  slider_step = NULL,
  rating_max = NULL,
  rating_icon = NULL,
  section_intro = NULL,
  page = NULL
)
```

Arguments

id	Character. A unique identifier for this item. Used as the column name in response data. Must contain only letters, numbers, and _ characters.
label	Character. The question text or content displayed to the respondent.
type	Character. The response type. One of "likert", "single_choice", "multiple_choice", "numeric", "text", "textarea", "date", "matrix", "slider", "ranking", "rating", "section_break", or "text_block".
required	Logical. Whether the respondent must answer this item.
choice_set	Character or NULL. The id of a choice set defined with sf_choices().
scale_id	Character or NULL. The id of the scale this item belongs to.
reverse	Logical. Whether this item is reverse-coded within its scale.
help	Character or NULL. Help text displayed beneath the question.
placeholder	Character or NULL. Placeholder text for text inputs.
matrix_items	Character vector or NULL. Row labels for "matrix" type.
slider_min	Numeric or NULL. Minimum value for "slider" type.
slider_max	Numeric or NULL. Maximum value for "slider" type.
slider_step	Numeric or NULL. Step size for "slider" type.
rating_max	Integer or NULL. Maximum rating for "rating" type.
rating_icon	Character or NULL. Icon type: "star" or "heart".
section_intro	Character or NULL. Intro text for "section_break" type.
page	Integer or NULL. Page number for multi-page surveys.

Value

An object of class sf_item (a named list).

See Also

sf_instrument(), sf_choices(), sf_scale()

Examples

```
item <- sf_item(
  id = "sat_overall", label = "Overall, how satisfied are you?",
  type = "likert", required = TRUE, choice_set = "agree5",
  scale_id = "satisfaction"
)

sec <- sf_item("sec_1", "Demographic Information", type = "section_break",
               section_intro = "Please answer the following questions.")
```

`sf_model` *Create a surveyframe model specification*

Description

Create a surveyframe model specification

Usage

```
sf_model(
  id,
  label = NULL,
  type = c("efa", "cfa", "cb_sem", "pls_sem"),
  engine = NULL,
  constructs = list(),
  paths = list(),
  covariances = list(),
  indirect = list(),
  options = list()
)
```

Arguments

`id`	Model identifier.
`label`	Human-readable model label.
`type`	Model type. One of `"efa"`, `"cfa"`, `"cb_sem"`, or `"pls_sem"`.
`engine`	Optional engine name. Defaults to `"lavaan"` for CFA/CB-SEM and `"seminr"` for PLS-SEM.
`constructs`	List of `sf_construct()` objects.
`paths`	List of `sf_path()` objects.
`covariances`	List of `sf_covariance()` objects.
`indirect`	List of `sf_indirect()` objects.
`options`	List of model options, such as `estimator`, `missing`, `bootstrap`, or `standardised`.

Value

An object of class `sf_model`.

sf_path *Define a structural path between constructs*

Description

Define a structural path between constructs

Usage

```
sf_path(from, to, label = NULL)
```

Arguments

from	Source construct ID.
to	Target construct ID.
label	Optional lavaan label for the path.

Value

An object of class sf_path.

sf_scale *Define a scored scale*

Description

Creates a scale definition that groups items and specifies how composite scores are computed. The scale carries scoring rules used by score_scales() and measurement structure used by reliability_report(), item_report(), and cfa_syntax().

Usage

```
sf_scale(
  id,
  label,
  items,
  method = c("mean", "sum"),
  min_valid = NULL,
  reverse_items = NULL,
  weights = NULL
)
```

Arguments

id	Character. A unique identifier for this scale. Referenced in the scale_id argument of sf_item().
label	Character. A human-readable name for the scale, used in reports and codebooks.
items	Character vector. The id values of items that belong to this scale. Order controls presentation in reports; scoring uses the same item IDs regardless of order.
method	Character. Scoring method. Either "mean" (default) or "sum".
min_valid	Integer or NULL. The minimum number of non-missing items required to compute a score for a respondent. When NULL, all items must be present. Used by score_scales().
reverse_items	Character vector or NULL. A subset of items that are reverse-coded. These can also be flagged at the item level with the reverse argument in sf_item(). Both sources are respected.
weights	Numeric vector or NULL. Item weights for weighted scoring. Must have the same length as items if supplied. score_scales() applies the weights to either method = "mean" or method = "sum".

Value

An object of class sf_scale (a named list).

See Also

sf_item(), score_scales(), reliability_report()

Examples

```
sat_scale <- sf_scale(
  id            = "satisfaction",
  label         = "Customer Satisfaction",
  items         = c("sat_overall", "sat_speed", "sat_quality"),
  method        = "mean",
  min_valid     = 2,
  reverse_items = NULL
)
```

---

summary.sframe *Summarise an sframe instrument object*

Description

Prints a structured summary of an sframe object including metadata, item type counts, scale definitions, branching rules, and check specifications.

Usage

```
## S3 method for class 'sframe'
summary(object, ...)
```

Arguments

`object`	An object of class `sframe`.
`...`	Ignored. Present for S3 consistency.

Value

`object`, invisibly.

Examples

```
item <- sf_item("q1", "How satisfied are you?", type = "likert",
                choice_set = "agree5")
instr <- sf_instrument("My Survey", components = list(item))
summary(instr)
```

---

survey_module_server *Shiny module server for an embedded survey*

Description

Renders the survey instrument and collects the respondent's answers. Returns a `reactive` that holds `NULL` until the form is submitted, then returns the response as a named list (one element per visible item).

Usage

```
survey_module_server(id, instrument, on_submit = NULL)
```

Arguments

`id`	A character string matching the `id` passed to `survey_module_ui()`.
`instrument`	An `sframe` object, or a `reactive` that returns one. Changing the reactive value resets the survey.
`on_submit`	Optional function of one argument. Called immediately after submission with the response list. Useful for writing to a database or sending an email without waiting for an `shiny::observeEvent()` elsewhere in the app.

Value

A `reactive` that returns `NULL` before submission and the response list after.

See Also

survey_module_ui()

Examples

```
# See survey_module_ui() for a complete example.
```

---

survey_module_ui *Shiny module UI for an embedded survey*

Description

Places a survey rendered by surveyframe inside a larger Shiny application. Pair with survey_module_server() in the server function. The module renders the full instrument including welcome page, all item types, branching logic, required-field validation, and a thank-you screen.

Usage

```
survey_module_ui(id, width = "100%")
```

Arguments

id	A character string. The module namespace ID, passed identically to survey_module_server().
width	Character. CSS width for the survey card. Defaults to "100%".

Value

A shiny.tag object.

See Also

survey_module_server(), launch_studio(), export_static_survey()

Examples

```
# Minimal embedding example:
library(shiny)
library(surveyframe)

cs    <- sf_choices("ag5", 1:5, c("SD", "D", "N", "A", "SA"))
item  <- sf_item("q1", "Rate your experience.", type = "likert",
                 choice_set = "ag5", required = TRUE)
instr <- sf_instrument("Quick Survey", components = list(cs, item))

ui <- fluidPage(
  survey_module_ui("demo"),
  verbatimTextOutput("result")
```

```
)

server <- function(input, output, session) {
  resp <- survey_module_server("demo", instrument = instr)
  output$result <- renderPrint({
    req(resp())
    resp()
  })
}

shinyApp(ui, server)
```

`validate_model` *Validate a surveyframe model specification*

Description

Checks model IDs, construct IDs, indicators, structural path endpoints, duplicate paths, indirect paths, and engine/type compatibility.

Usage

```
validate_model(model, instrument = NULL, strict = TRUE)
```

Arguments

`model`	An `sf_model()` object or compatible list.
`instrument`	Optional `sframe` object. When supplied, model indicators must match instrument item IDs.
`strict`	Logical. When `TRUE`, invalid models raise an error. When `FALSE`, a list with `valid` and `problems` is returned.

Value

The model invisibly when valid and `strict = TRUE`, otherwise a validation result list.

validate_sframe *Validate an instrument object*

Description

Checks the internal consistency of an sframe instrument object and reports all detected problems. Validation is performed automatically by write_sframe() and optionally by read_sframe(). It can also be run independently at any point during instrument construction.

Usage

```
validate_sframe(instrument, strict = TRUE)
```

Arguments

instrument	An sframe object created by sf_instrument().
strict	Logical. When TRUE (default), any detected problem raises an error of class sframe_validation_error. When FALSE, problems are returned as a character vector of messages without stopping.

Details

The following checks are performed:

- Duplicate item IDs
- Invalid item IDs
- Duplicate choice-set IDs
- Duplicate scale IDs
- Items with missing labels
- Items referencing a missing choice_set in the instrument
- Items referencing a missing scale_id in the instrument
- Items marked reverse = TRUE without a scale_id
- Choice sets referenced by items but not present in the instrument
- Scale items vectors containing IDs not present in the instrument
- Branching rules referencing item IDs not present in the instrument
- Attention checks referencing item IDs not present in the instrument
- Analysis plan roles referencing missing variables or models
- Model specifications referencing missing indicators or constructs

Value

When strict = TRUE and the instrument is valid, the instrument is returned invisibly with meta$validated set to TRUE. When strict = FALSE, a named list with elements valid (logical) and problems (character vector) is returned.

See Also

sf_instrument(), write_sframe()

Examples

```
# Build a minimal valid instrument and validate it
cs    <- sf_choices("ag5", 1:5,
           c("Strongly disagree", "Disagree", "Neutral",
             "Agree", "Strongly agree"))
item  <- sf_item("sat_1", "The service met my expectations.",
                 type = "likert", choice_set = "ag5", scale_id = "sat")
scale <- sf_scale("sat", "Satisfaction", items = "sat_1")
instr <- sf_instrument("Demo Survey", components = list(cs, item, scale))

# Non-strict: returns a list without stopping
result <- validate_sframe(instr, strict = FALSE)
result$valid
result$problems

# Strict: returns instrument invisibly when valid
validated <- validate_sframe(instr, strict = TRUE)
isTRUE(validated$meta$validated)
```

---

validity_report *Validity report for construct models*

Description

Validity report for construct models

Usage

```
validity_report(loadings, construct_scores = NULL)
```

Arguments

loadings	A data.frame with columns construct, item, and loading, or a named list of loading vectors by construct.
construct_scores	Optional data.frame of construct scores for Fornell-Larcker, HTMT, and inter-construct correlations.

Value

An object of class sframe_validity_report.

write_sframe *Write an instrument to a .sframe file*

Description

Serialises an sframe instrument object to a UTF-8 JSON file with a SHA-256 integrity hash. The instrument is validated before writing unless the object already carries a valid status. The hash is computed over the full serialised content with the hash.value field set to an empty string.

Usage

```
write_sframe(instrument, path, pretty = TRUE, overwrite = FALSE)
```

Arguments

instrument	An sframe object created by sf_instrument().
path	Character. The file path to write to. The .sframe extension is appended automatically if not already present.
pretty	Logical. Whether to write formatted JSON with indentation. Defaults to TRUE. Set to FALSE for compact files.
overwrite	Logical. Whether to overwrite an existing file. Defaults to FALSE.

Value

The file path, invisibly.

See Also

read_sframe(), validate_sframe()

Examples

```
instr <- read_sframe(
  system.file("extdata", "tourism_services_demo.sframe",
              package = "surveyframe")
)
out <- write_sframe(instr, tempfile(fileext = ".sframe"))
file.exists(out)
```

Index