Package 'surveyframe'

August 21, 2026

Title Survey Instrument Workflows

Version 0.4.0

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. This release adds three capability themes: multi-criteria decision analysis (AHP, ANP, DEMATEL, TOPSIS, VIKOR, MOORA, SMART, WASPAS, PROMETHEE II, ELECTRE I), small-sample survey helpers, and text and open-ended response analysis (term and n-gram frequency, keyword in context, co-occurrence and co-occurrence networks, sentiment, document-feature matrices, and topic modelling via LDA or structural topic models).

License MIT + file LICENSE

URL https://mohammedalisharafuddin.github.io/surveyframe/, 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 ggplot2 (>= 3.4.0), googlesheets4 (>= 1.1.0), shiny (>= 1.7.0), psych (>= 2.3.0), MASS, nnet, logistf (>= 1.24.0), digest (>= 0.6.0), lavaan (>= 0.6-0), testthat (>= 3.0.0), knitr, rmarkdown, naniar (>= 1.0.0), pagedown (>= 0.20), RMCDA (>= 0.3.1), rstudioapi (>= 0.13), jmv (>= 2.4.0), tidytext (>= 0.4.0), quanteda (>= 3.0.0), stm (>= 1.3.0), topicmodels, igraph (>= 1.5.0)

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

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 2026-08-20 23:20:02 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.

`amendment_log` *Read an instrument's amendment log*

Description

Returns the disclosed-amendment history recorded by `amend_sframe()` as a data frame, one row per amendment in the order they were recorded.

Usage

```
amendment_log(instrument)
```

Arguments

instrument	An sframe object.

Value

A data frame with columns timestamp, reason_code, reason_text, tier, author, deviation_report, signoff, previous_hash, new_hash, and changed_fields (a comma-joined string). Zero rows if the instrument has no recorded amendments. Export with write.csv() for an external audit trail.

See Also

amend_sframe()

Examples

```
item  <- sf_item("q1", "How satisfied are you?", type = "text")
instr <- sf_instrument("Demo", components = list(item))
amendment_log(instr)
```

---

amend_sframe *Record a disclosed amendment to an instrument*

Description

Appends a structured, disclosed-revision entry to an instrument's amendment log, comparing previous against instrument to record what changed and why. This is the path around read_sframe()'s hash check for *legitimate* revision: a data-entry correction, bot-response removal, or a documented model respecification. It does not weaken that check – read_sframe() still hard-aborts on any edit that never went through amend_sframe(). What it adds is a place for a disclosed change to be recorded inside the file itself, alongside the content it explains, rather than only in an email or a lab notebook.

Usage

```
amend_sframe(
  previous,
  instrument,
  reason_code,
  reason_text,
  tier = NULL,
  author = NULL,
  deviation_report = NULL,
  second_signoff = NULL
)
```

Arguments

previous	An sframe object: the instrument's state before this amendment.
instrument	An sframe object: the instrument's state after the change this call discloses.
reason_code	One of "data_correction", "bot_removal", "model_respecification", "instrument_revision", "other".
reason_text	Character. A free-text explanation. Required and must be non-empty regardless of reason_code.
tier	"pipeline" or "design". When NULL (the default), inferred from reason_code: data_correction/bot_removal default to "pipeline"; everything else defaults to "design". Pass explicitly to override the default in either direction.
author	Character or NULL. Who made the change.
deviation_report	Character or NULL. Required when tier is "design": what changed in the research question, method, or model, and why. Ignored (may be NULL) for "pipeline" amendments.
second_signoff	Character or NULL. A second reviewer's name or identifier (an ethics board reference, a co-author). When omitted, the entry records signoff = "none".

Details

Amendments come in two tiers. A "pipeline" amendment (data corrections, bot removal) is expected researcher behaviour and needs only a reason. A "design" amendment (anything touching the analysis plan or a measurement or structural model) is exactly the kind of post-hoc change the design-time analysis plan exists to guard against, so it additionally requires a deviation_report describing what changed in the research question, method, or model and why. second_signoff is optional at either tier; when omitted, the log entry records signoff = "none" rather than leaving the field blank, so the absence of independent review is visible to anyone auditing the log later.

previous_hash and new_hash on each entry are a **content** fingerprint (a SHA-256 over the instrument's substantive fields – items, choices, scales, branching, checks, analysis plan, models, designs – with the hash and amendments fields themselves excluded), not the .sframe file's own integrity hash from write_sframe(). The two serve different purposes: the file hash (via read_sframe()) proves the file on disk is byte-identical to what was written; an amendment's content hash proves what the instrument's substance was immediately before and after this specific, disclosed change.

Value

The amended sframe object, with the new entry appended to its amendment log. Call write_sframe() to persist it.

See Also

amendment_log(), write_sframe(), read_sframe()

Examples

```
item  <- sf_item("q1", "How satisfied are you?", type = "text")
instr <- sf_instrument("Demo", components = list(item))
item2 <- sf_item("q1", "How satisfied are you overall?", type = "text")
revised <- sf_instrument("Demo", components = list(item2))
amended <- amend_sframe(
  instr, revised,
  reason_code = "instrument_revision",
  reason_text = "Clarified item wording after a pilot round.",
  deviation_report = "Wording only; no change to the construct measured."
)
nrow(amendment_log(amended))
```

---

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`.

`as_sframe` *Coerce to an instrument*

Description

Recovers the `sframe` instrument from a validation diagnostic. This is the migration path for code that used the `strict = TRUE` return of `validate_sframe()` as an instrument, which it no longer is.

Usage

```
as_sframe(x, ...)

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

## S3 method for class 'sframe_validation'
as_sframe(x, ...)
```

Arguments

`x`	An sframe_validation object or an `sframe`.
`...`	Passed to methods.

Value

An `sframe` object. When the validation passed, its `meta$validated` is `TRUE`.

See Also

`validate_sframe()`, sframe_validation

Examples

```
item  <- sf_item("q1", "How satisfied are you?", type = "text")
instr <- sf_instrument("Demo", components = list(item))

validated <- as_sframe(validate_sframe(instr, strict = TRUE))
isTRUE(sf_meta(validated)$validated)
```

`bootstrap_ci` *Percentile bootstrap confidence interval for a statistic*

Description

Resamples `x` with replacement `R` times, applies `FUN` to each resample, and returns the percentile interval of the resampled statistics together with the observed value.

Usage

```
bootstrap_ci(x, FUN = stats::median, R = 2000, conf.level = 0.95, seed = NULL)
```

Arguments

`x`	A numeric vector.
`FUN`	A function of one vector returning a single number. Defaults to `stats::median()`.
`R`	Integer. Number of bootstrap resamples. Defaults to 2000.
`conf.level`	Confidence level. Defaults to 0.95.
`seed`	Integer or NULL. When supplied, sets the random seed so the interval is reproducible.

Value

A named numeric vector: `estimate`, `lower`, `upper`. The bounds are `NA` when `x` has fewer than 3 finite values.

See Also

`cohens_d_ci()`, `cramers_v_ci()`, `eta_sq_ci()`

Examples

```
bootstrap_ci(mtcars$mpg, seed = 42)
bootstrap_ci(mtcars$mpg, FUN = mean, conf.level = 0.90, seed = 42)
```

`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)
```

`clean_text_responses` *Clean open-ended text responses for analysis*

Description

Extracts one text/textarea item's responses from a response data frame and applies light, configurable cleaning: lower-casing, punctuation removal, and optional number stripping. Blank and missing responses are dropped rather than kept as empty strings, since they carry no term-frequency signal and would otherwise inflate downstream response counts.

Usage

```
clean_text_responses(
  data,
  item_id,
  lowercase = TRUE,
  remove_punct = TRUE,
  strip_numbers = FALSE,
  instrument = NULL
)
```

Arguments

`data`	A data.frame of responses.
`item_id`	Character. The text/textarea item's column name.
`lowercase`	Logical. Lower-case the text. Default `TRUE`.
`remove_punct`	Logical. Strip punctuation. Default `TRUE`.
`strip_numbers`	Logical. Strip digits. Default `FALSE`.
`instrument`	Optional `sframe` instrument. When supplied, `item_id` is validated as a `"text"` or `"textarea"` item before cleaning.

Value

A character vector of cleaned responses, with an integer `"respondent"` attribute giving each entry's original row index in `data`, so quotes extracted later can cite a respondent.

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(sf_items(cb))
nrow(sf_scales(cb))
```

cohens_d_ci *Bootstrap confidence interval for Cohen's d*

Description

Percentile bootstrap for the standardised mean difference between two independent groups. Each resample draws within each group, preserving the group sizes.

Usage

```
cohens_d_ci(x, y, R = 2000, conf.level = 0.95, seed = NULL)
```

Arguments

x, y	Numeric vectors, one per group.
R	Integer. Number of bootstrap resamples. Defaults to 2000.
conf.level	Confidence level. Defaults to 0.95.
seed	Integer or NULL. When supplied, sets the random seed.

Value

A named numeric vector: estimate, lower, upper. The bounds are NA when either group has fewer than 3 finite values.

See Also

bootstrap_ci()

Examples

```
cohens_d_ci(mtcars$mpg[mtcars$am == 1], mtcars$mpg[mtcars$am == 0],
            seed = 42)
```

cramers_v_ci *Bootstrap confidence interval for Cramer's V*

Description

Percentile bootstrap for the association strength in a contingency table. The table is expanded back to individual observations, which are resampled jointly. For a 2 by 2 table the statistic equals phi.

Usage

```
cramers_v_ci(tab, R = 2000, conf.level = 0.95, seed = NULL)
```

Arguments

tab	A contingency table (from table()) or a matrix of counts.
R	Integer. Number of bootstrap resamples. Defaults to 2000.
conf.level	Confidence level. Defaults to 0.95.
seed	Integer or NULL. When supplied, sets the random seed.

Value

A named numeric vector: estimate, lower, upper. The bounds are NA when the table holds fewer than 3 observations.

See Also

bootstrap_ci()

Examples

```
cramers_v_ci(table(mtcars$am, mtcars$cyl), seed = 42)
```

---

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 suggested number of factors from parallel analysis is returned in `$suggested_nfactors`. 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`. Alongside the psych objects it carries three tidy data frames ready for plotting and reporting: `loadings_long` (item_id, factor, loading), `communalities_table` (item_id, communality, uniqueness), and `variance_table` (factor, ss_loadings, proportion_var, cumulative_var).

`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.

eta_sq_ci *Bootstrap confidence interval for eta squared*

Description

Percentile bootstrap for the proportion of variance in `outcome` explained by `group`, resampling observations jointly so the group structure travels with each resample.

Usage

```
eta_sq_ci(outcome, group, R = 2000, conf.level = 0.95, seed = NULL)
```

Arguments

`outcome`	A numeric vector.
`group`	A grouping vector of the same length.
`R`	Integer. Number of bootstrap resamples. Defaults to 2000.
`conf.level`	Confidence level. Defaults to 0.95.
`seed`	Integer or NULL. When supplied, sets the random seed.

Value

A named numeric vector: `estimate`, `lower`, `upper`. The bounds are `NA` with fewer than 3 complete observations or fewer than 2 groups.

See Also

`bootstrap_ci()`

Examples

```
eta_sq_ci(mtcars$mpg, mtcars$cyl, seed = 42)
```

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
)
```

`extract_quotes` *Extract representative quotes for each STM topic*

Description

Takes the result of `sframe_run_stm_topics()` (not a raw `stm` model object) and, for each topic, pulls the top `n_quotes` documents by topic-document probability using `stm::findThoughts()`, then maps each one back to its original respondent row via the mapping `sframe_run_stm_topics()` stored on `result$fit`.

Usage

```
extract_quotes(model, text, n_quotes = 3L)
```

Arguments

`model`	A `stm_topics` result list from `sframe_run_stm_topics()` (i.e. `result`, not `result$fit` and not the raw `stm` object).
`text`	The response vector the topic model was fit on: either the raw vector (one entry per original data row, indexed 1:1 by row number) or the `clean_text_responses()`-cleaned vector, which drops blank/missing rows and is therefore *shorter*, its positions no longer equal original row numbers once any earlier row was dropped. Both forms work correctly: when `text` carries the `respondent` attribute `clean_text_responses()` sets, that mapping is used to find each quote's real position; otherwise `text` is assumed to be the raw, 1:1-indexed vector.
`n_quotes`	Integer. Quotes to return per topic. Default `3L`.

Value

A data.frame with columns `topic`, `rank`, `respondent` (the original row index in the data the model's `text` argument came from, not a document-matrix or corpus row index), and `quote`.

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.

format.sf_branch *Format an sf_branch object as a string*

Description

Format an sf_branch object as a string

Usage

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

Arguments

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

Value

A single character string.

format.sf_check *Format an sf_check object as a string*

Description

Format an sf_check object as a string

Usage

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

Arguments

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

Value

A single character string.

format.sf_choices *Format an sf_choices object as a string*

Description

Format an sf_choices object as a string

Usage

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

Arguments

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

Value

A single character string.

format.sf_item *Format an sf_item object as a string*

Description

Format an sf_item object as a string

Usage

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

Arguments

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

Value

A single character string.

format.sf_model *Format an sf_model object as a string*

Description

Format an sf_model object as a string

Usage

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

Arguments

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

Value

A single character string.

format.sf_scale *Format an sf_scale object as a string*

Description

Format an sf_scale object as a string

Usage

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

Arguments

x	An object of class sf_scale.
...	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, with no manual file-load step 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. Required. Calling `launch_dashboard()` with no instrument errors with guidance. Use `launch_dashboard_demo()` for the bundled demo or `launch_studio()` to upload interactively.
`responses`	A `data.frame` or `tibble` of survey responses, as produced by `read_responses()` or `read_sheet_responses()`. When NULL 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 and takes its data from R. It has no upload screen, so pass `instrument` and `responses` directly. To open and upload data interactively, use `launch_studio()`, which includes this same dashboard as its Dashboard tab. For a quick look at bundled demo data, use `launch_dashboard_demo()`.

Value

Called for its side effect. Returns nothing.

See Also

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

Examples

```
## Not run:
# For the bundled demo, use launch_dashboard_demo().
# To upload data interactively, use launch_studio().

# 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)

## End(Not run)
```

`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

```
## Not run:
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"
)

## End(Not run)
```

---

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.

link_git_commit *Link an instrument to its current Git commit*

Description

Records the current Git commit SHA and subject line for `repo_path` alongside the instrument. This does not replace Git history – it is a pointer into it. The SHA-256 hash `write_sframe()` embeds in the `.sframe` file confirms the file on disk matches what this specific, already-explained commit produced; a reviewer reads the commit itself for the "what changed and why."

Usage

```
link_git_commit(instrument, repo_path = ".")
```

Arguments

`instrument`	An `sframe` object.
`repo_path`	Character. Path to check for a Git repository. Defaults to the current working directory.

Details

Git is entirely optional. When `repo_path` is not inside a Git repository, or the `git` executable is not on the `PATH`, this returns a clear, non-error result with `linked = FALSE` rather than aborting – the rest of surveyframe never requires Git.

Value

A list with `linked` (logical), and when `linked` is `TRUE`, `commit` (the full commit SHA) and `message` (the commit's subject line); when `linked` is `FALSE`, `reason` (a human-readable explanation: `"git not found"` or `"not a git repository"`).

See Also

`amend_sframe()`, `write_sframe()`

Examples

```
item  <- sf_item("q1", "How satisfied are you?", type = "text")
instr <- sf_instrument("Demo", components = list(item))
link_git_commit(instr, repo_path = tempdir())
```

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.

ngram_frequency *N-gram frequency for open-ended text*

Description

Tokenises text via the same tokeniser as term_frequency() (whitespace splitting, lower-casing, punctuation stripping, and stop-word removal), then slides a window of n tokens across each response's token vector and counts how often each resulting n-gram occurs. n = 2 (the default) gives bigrams; n = 3 gives trigrams. Because stop words are already removed by the shared tokeniser, an n-gram never straddles a dropped word; it is built only from tokens that survive filtering, in their original order within each response.

Usage

```
ngram_frequency(text, n = 2L, stop_words = NULL, top_n = 30L)
```

Arguments

text	Character vector of responses (raw or already cleaned by clean_text_responses()).
n	Integer. N-gram size. Default 2 (bigrams).
stop_words	Character vector of words to exclude, or NULL to use the built-in English list, or character(0) for no filtering.
top_n	Integer. Maximum number of n-grams to return, most frequent first. Default 30.

Value

A data.frame with columns term (the space-joined n-gram), n, and pct.

`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, and `"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
```

plot.sframe_analysis_results *Plot analysis-plan results*

Description

Draws the charts that run_analysis_plan() attaches when called with plots = TRUE. With which supplied, returns that single chart. With which omitted, prints every attached chart in queue order and returns the list invisibly. Regression diagnostic panels stay on the result's diagnostic_plots element and are not drawn here.

Usage

```
## S3 method for class 'sframe_analysis_results'
plot(x, ..., which = NULL)
```

Arguments

x	An sframe_analysis_results object from run_analysis_plan().
...	Ignored.
which	A research-question number or a plan block id selecting one chart, or NULL for all.

Value

A ggplot2 object when which is supplied, otherwise an invisible named list of ggplot2 objects keyed by plan block id.

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)
```

print.sf_branch *Print an sf_branch object*

Description

Print an sf_branch object

Usage

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

Arguments

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

Value

x, invisibly.

Examples

```
br <- sf_branch("q2", depends_on = "q1", operator = "==",
                value = "yes", action = "show")
print(br)
```

print.sf_check *Print an sf_check object*

Description

Print an sf_check object

Usage

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

Arguments

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

Value

x, invisibly.

Examples

```
ck <- sf_check("attn1", item_id = "q5", type = "attention",
               pass_values = 3)
print(ck)
```

`print.sf_choices` *Print an sf_choices object*

Description

Print an sf_choices object

Usage

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

Arguments

`x`	An object of class `sf_choices`.
`...`	Ignored. Present for S3 consistency.

Value

x, invisibly.

Examples

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

print.sf_item *Print an sf_item object*

Description

Print an sf_item object

Usage

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

Arguments

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

Value

x, invisibly.

Examples

```
it <- sf_item("q1", "How satisfied are you?", type = "likert",
              choice_set = "agree5")
print(it)
```

print.sf_model *Print an sf_model object*

Description

Print an sf_model object

Usage

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

Arguments

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

Value

x, invisibly.

print.sf_scale *Print an sf_scale object*

Description

Print an sf_scale object

Usage

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

Arguments

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

Value

x, invisibly.

Examples

```
sc <- sf_scale("sat", "Satisfaction", items = c("q1", "q2", "q3"))
print(sc)
```

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, outside the item IDs, to retain (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",
  meta_cols = NULL
)
```

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"`.
`meta_cols`	Character vector or NULL. Additional sheet columns to accept as metadata without a warning, for example bridge fields a host application appends to each submission. `"started_at"` is always included.

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", "pdf"),
  include_quality = TRUE,
  include_reliability = TRUE,
  include_codebook = TRUE,
  include_missing = TRUE,
  include_descriptives = TRUE,
  include_analysis = TRUE,
  include_models = TRUE,
  plot_palette = c("web", "print"),
  interpretations = NULL
)
```

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: "html" (default) or "pdf". PDF output renders the HTML report and prints it through pagedown::chrome_print(), which requires the pagedown package (in Suggests) and a local Chrome or Chromium installation. The HTML path is unchanged.
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`.

plot_palette
: One of `"web"` (brand colours, for on-screen reading) or `"print"` (black, grey, and white, for a journal-ready or print-friendly report). Applied to every chart the report embeds. See `sframe_brand()`.

interpretations
: Named list or NULL. Written interpretations keyed by analysis-plan block id, added after the results are known. When a block has an entry, its report section shows the pre-declared decision rule under a "Planned decision rule" label followed by the written text under an "Interpretation" label. Blocks without an entry render exactly as they do when this argument is NULL. Interpretations are report content only and are never written into the instrument.

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,
  interpretations = 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.

`interpretations` Named list or NULL. Written interpretations keyed by analysis-plan block id, added after the results are known. A block with an entry shows that text in its Interpretation section in place of the pre-declared prompt fallback. Blocks without an entry render exactly as they do when this argument is NULL. Interpretations are report content only and are never written into the instrument.

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

```
cs    <- sf_choices("ag5", 1:5,
           c("Strongly disagree", "Disagree", "Neutral",
             "Agree", "Strongly agree"))
item  <- sf_item("sat_1", "How satisfied are you?",
                 type = "likert", choice_set = "ag5")
instr <- sf_instrument("My Survey", components = list(cs, item))
app <- render_survey(instr)
app <- render_survey(instr, save_responses = "csv", output_path = tempfile(fileext = ".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,
  plots = FALSE,
  plot_palette = c("web", "print")
)
```

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`.
`plots`	Logical. When `TRUE` and ggplot2 is installed, supported blocks gain a `$plot` element holding a brand-styled ggplot object: bar charts for frequency and chi-square blocks, scatter plots with a regression overlay for correlation and linear-regression blocks. Defaults to `FALSE`.
`plot_palette`	One of `"web"` (brand colours, for on-screen use) or `"print"` (black, grey, and white, for journal-ready print figures). Applied to every plot attached when `plots = TRUE`. See `sframe_brand()`.

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. Inferential blocks also carry a `$table` data frame suitable for `knitr::kable()`. 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.

sensitivity_analysis *Test how far a decision ranking moves when the weights are perturbed*

Description

Perturbs each criterion weight up and down by delta, renormalises the weight vector to sum to 1, reruns the same ranking method, and compares the perturbed ranking against the base ranking. A ranking that survives this unchanged is one a reviewer can be told is robust to the weights. A ranking whose leader changes under a 5 percent nudge is not.

Usage

```
sensitivity_analysis(
  x,
  weights,
  criteria_types,
  method = "topsis",
  delta = 0.05,
  alternatives = NULL,
  criteria = NULL,
  ...
)
```

Arguments

x	Numeric performance matrix, alternatives in rows and criteria in columns.
weights	Numeric weight vector, one per criterion. Renormalised to sum to 1 before use.
criteria_types	Character vector of "benefit" or "cost", one per criterion.
method	Ranking method. One of "topsis", "vikor", "moora", "smart", "waspas", "promethee", or "electre".
delta	Perturbation size as a proportion of the weight, default 0.05. A weight of 0.40 with delta = 0.05 is tested at 0.42 and 0.38 before renormalisation.
alternatives	Optional labels for the rows of x.
criteria	Optional labels for the columns of x.
...	Passed to the underlying method, for example v for VIKOR or lambda for WASPAS.

Value

An object of class sframe_sensitivity, a list with $table (one row per criterion and direction, carrying criterion, direction, rho, rank_changed, and top_changed), $base_ranks, $method, $delta, and $stable, a single logical that is TRUE when no perturbation changed the ranking.

See Also

run_analysis_plan(), sframe_decision_options()

Examples

```
x <- matrix(c(4.1, 3.0, 210, 3.6, 4.5, 180, 4.8, 2.5, 260),
            nrow = 3, byrow = TRUE)
sa <- sensitivity_analysis(
  x,
  weights        = c(0.4, 0.3, 0.3),
  criteria_types = c("benefit", "benefit", "cost"),
  method         = "topsis",
  alternatives   = c("Alpha", "Basilica", "Coral"),
  criteria       = c("service", "location", "price")
)
sa$stable
as.data.frame(sa)
```

---

sframe_aggregate_judgements

Aggregate individual judgement matrices

Description

Aggregation of individual judgements (AIJ) across respondents. The element-wise geometric mean is the standard choice for reciprocal AHP matrices, because it is the only mean that preserves reciprocity: the arithmetic mean of `m[a, b]` and the arithmetic mean of `m[b, a]` are not reciprocals of each other. DEMATEL matrices are not reciprocal, so they aggregate arithmetically.

Usage

```
sframe_aggregate_judgements(
  matrices,
  method = c("geometric", "arithmetic"),
  cr_filter = FALSE
)
```

Arguments

`matrices`	Either a list of square numeric matrices or the object returned by `sframe_assemble_pairwise()`.
`method`	`"geometric"` (the AHP default) or `"arithmetic"` (DEMATEL).
`cr_filter`	Logical. When `TRUE`, individual matrices with a consistency ratio at or above 0.10 are dropped before aggregation. Applies to reciprocal matrices only. Default `FALSE`.

Value

A list with `matrix`, `method`, `n_respondents`, `n_dropped`, `n_dropped_consistency`, and `consistency` (the per-respondent CR distribution, or `NULL` for a non-reciprocal set).

See Also

`sframe_assemble_pairwise()`

Examples

```
m1 <- matrix(c(1, 3, 1 / 3, 1), 2, 2)
m2 <- matrix(c(1, 5, 1 / 5, 1), 2, 2)
sframe_aggregate_judgements(list(m1, m2))$matrix
```

`sframe_assemble_pairwise` *Assemble per-respondent comparison matrices*

Description

Builds one square judgement matrix per respondent from the pair columns of a `"pairwise_comparison"` item. A `"saaty"` item stores one signed integer per unordered pair, so the reciprocal half of each matrix is reconstructed here rather than stored (a collected value of +5 becomes `m[a, b] = 5` and `m[b, a] = 1/5`, and the diagonal is 1). An `"influence"` item stores one unsigned integer per ordered pair, fills the directed cell only, and has a zero diagonal, because the influence of a on b says nothing about the influence of b on a.

Usage

```
sframe_assemble_pairwise(data, instrument, item_id)
```

Arguments

data	A data frame of responses, as produced by read_responses().
instrument	An sframe instrument declaring item_id.
item_id	Character. The id of a "pairwise_comparison" item.

Details

Respondents who left any pair blank, or whose answer falls outside the declared scale, are dropped whole and counted. Completing partial matrices (the Harker approach) is deliberately out of scope.

Value

A list with matrices (a list of square numeric matrices with dimnames), n_respondents, n_dropped, dropped (a data frame of row number and reason), items, and scale.

See Also

sframe_aggregate_judgements(), sframe_collected_weights()

Examples

```
crits <- c("price", "speed")
study <- sf_instrument(
  title = "Demo", components = list(
    sf_item("pairs", "Compare the criteria", type = "pairwise_comparison",
            comparison_items = crits)
  )
)
responses <- data.frame(pairs__price__vs__speed = c(3, -5))
sframe_assemble_pairwise(responses, study, "pairs")$matrices[[1]]
```

---

sframe_as_data_frame *Coerce a surveyframe object to a data frame*

Description

Every surveyframe class returns its primary table. For an instrument that is the item table, for a validation result the problems, for a report the table it is mainly about. Where an object holds more than one table, the others are reachable through the named accessors in sf_accessors or, for a full tabular record of an instrument, through codebook_report().

Usage

```
## S3 method for class 'sframe'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'sf_choices'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'sframe_codebook'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'sframe_reliability_report'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'sframe_item_report'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'sframe_efa_report'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'sframe_efa_solution'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'sframe_descriptives_report'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'sframe_missing_data_report'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'sframe_validity_report'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'sframe_assumption_report'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'sframe_sample_size_plan'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'sframe_quality_report'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'sframe_sensitivity'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'sframe_analysis_results'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x	A surveyframe object.
row.names	Passed to base::as.data.frame().
optional	Passed to base::as.data.frame().
...	Ignored. Present for S3 consistency.

Value

A data frame.

See Also

sf_accessors, codebook_report()

Examples

```
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))

as.data.frame(instr)
as.data.frame(cs)
```

---

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(),
  render = 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.
`render`	List of rendering settings (welcome, header/logo, thankyou, theme) carried from the loaded instrument so previews and exports match.

Value

A list with `valid`, `problems`, and `instrument`.

`sframe_codebook_items_display` *Enrich a codebook's items table for display*

Description

Replaces `items_table`'s `choice_set` id with the choice set's actual response options ("1 = Strongly disagree; 2 = Disagree; ...") and its `scale_id` with the scale's label, so each row of the printed codebook is self-contained. `codebook_report()` itself keeps the raw ids, for joining `items_table` to `choices_table`/`scales_table` programmatically. This enrichment is for the rendered document, where a reader should not need to cross-reference a separate table just to see what "1" means on a scale shared by many items.

Usage

```
sframe_codebook_items_display(cb)
```

Arguments

`cb`	An `sframe_codebook` object from `codebook_report()`.

Value

A data.frame, `cb$items_table` with `choice_set` and `scale_id` replaced by display text.

See Also

`codebook_report()`

sframe_collected_weights *Criterion weights collected from respondents*

Description

Resolves a weight vector from either kind of collected weight item, so that a ranking runner never has to know which one the researcher used. A `"criteria_weight"` item is renormalised to sum 1 per respondent before the arithmetic mean is taken, so a respondent who allocated 90 points in total carries the same influence as one who allocated exactly 100. A `"pairwise_comparison"` item is assembled, aggregated geometrically, and reduced to its principal eigenvector.

Usage

```
sframe_collected_weights(data, instrument, item_id, cr_filter = FALSE)
```

Arguments

`data`	A data frame of responses.
`instrument`	An `sframe` instrument declaring `item_id`.
`item_id`	Character. A `"criteria_weight"` or `"pairwise_comparison"` item id.
`cr_filter`	Logical. Passed to `sframe_aggregate_judgements()` for a pairwise item. Ignored otherwise.

Value

A list with `weights` (a named numeric vector summing to 1), `criteria`, `source`, `item_id`, `n_respondents`, `n_dropped`, and `consistency` (pairwise items only).

See Also

`sframe_assemble_pairwise()`

Examples

```
study <- sf_instrument(
  title = "Demo", components = list(
    sf_item("pts", "Divide 100 points", type = "criteria_weight",
            comparison_items = c("price", "speed"))
  )
)
sframe_collected_weights(
  data.frame(pts__price = c(60, 40), pts__speed = c(40, 60)), study, "pts"
)$weights
```

`sframe_decision_options` *Normalise the decision options of an analysis-plan block*

Description

A researcher-supplied performance matrix round-trips through JSON as a list of numeric row vectors with its dimnames dropped, so it is stored as `options$matrix` plus the `options$alternatives` and `options$criteria` label vectors and rebuilt here. Every length agreement between the matrix, its labels, the weights, and the criterion types is checked once, in one place, with the exact mismatch named.

Usage

```
sframe_decision_options(options)
```

Arguments

`options`	The `options` list of a decision analysis block.

Value

The same list with `matrix` rebuilt as a numeric matrix carrying dimnames, and `weights` and `criteria_types` coerced and checked.

PROMETHEE preference functions

A PROMETHEE block takes `options$preference_function`, one of `"usual"`, `"linear"`, or `"level"`, and `options$thresholds` for the 2 that need them. The default is `"usual"`, Brans and Vincke's type I step function, which needs no thresholds and so adds no researcher degrees of freedom.

This default differs from several other MCDM implementations, which default to the linear (V-shape) function and derive its thresholds from the range of the supplied data. Deriving thresholds that way makes the result depend on choices the researcher never declared, which is what this package exists to prevent, so surveyframe requires a threshold-bearing preference function to be asked for explicitly.

The choice changes the answer. Net flows always differ between the 2 functions, and the ranking itself changed in 226 of 400 randomly drawn 4-alternative by 3-criterion matrices. So a ranking cross-checked against an implementation that defaults to `"linear"` will often disagree unless `preference_function = "linear"` is set here and the same thresholds are supplied on both sides. The preference function actually used is always named in the block's APA sentence. Note also that `"usual"` produces tied ranks readily, because a step function scores every non-zero difference identically.

See Also

`run_analysis_plan()`

Examples

```
sframe_decision_options(list(
  matrix = list(c(4, 210), c(3, 180)),
  alternatives = c("Alpha", "Basilica"),
  criteria = c("service", "price"),
  criteria_types = c("benefit", "cost")
))$matrix
```

`sframe_dematel_compute` *The DEMATEL total-relation classification*

Description

Normalises the direct-influence matrix `X` by the larger of its greatest row sum and its greatest column sum (the standard DEMATEL normalisation, which keeps the Neumann series `N + N^2 + N^3 + ...` convergent), then solves the total-relation matrix `T = N(I - N)^-1` in closed form rather than by truncating the series. `D` is each criterion's row sum of `T` (how much it influences the others, direct and indirect combined) and `R` its column sum (how much it is influenced). Prominence `D + R` is overall involvement in the system, while relation `D - R` is net direction, positive for a net cause and negative or zero for a net effect. The threshold is the arithmetic mean of every entry of `T`: relations at or above it are considered significant enough to draw in an influence diagram.

Usage

```
sframe_dematel_compute(x)
```

Arguments

`x`	A square numeric matrix of direct influence, zero diagonal.

Details

`solve()` fails outright if `I - N` is exactly singular, which does not arise for a matrix normalised this way in ordinary use. No fallback series truncation is implemented, unlike the harvested source, because a singular `I - N` here would signal a malformed matrix rather than a case to work around silently.

Value

A list with `normalised` (N), `total_relation` (T), `D`, `R`, `prominence` (D + R), `relation` (D - R), `threshold` (mean of T), and `role` (a character vector, `"cause"` where relation > 0, else `"effect"`).

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_draw_mosaic *Mosaic plot for a two-way categorical result*

Description

Base-graphics mosaic plot (via `graphics::mosaicplot()`), matching the existing base-graphics precedent in this file (`sframe_draw_likert_diverging()`) so it renders without ggplot2. An alternative view of the same crosstab data `sframe_plot_crosstab()` renders as a grouped bar. Use whichever reads better for the table's shape (mosaic scales better to unbalanced group sizes).

Usage

```
sframe_draw_mosaic(result, palette = c("web", "print"))
```

Arguments

`result`	A `crosstab`/`chi_square` result list with a contingency `table`.
`palette`	One of `"web"` or `"print"`. See `sframe_brand()`.

Value

Invisibly `NULL`, called for its plotting side effect on the current graphics device.

See Also

`sframe_plot_crosstab()`

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`.

sframe_likert_scale_groups *Group a scale's Likert items for a combined diverging chart*

Description

Identifies which of an instrument's scales are eligible for one grouped diverging chart across their member items (`sframe_plot_likert_scale()`), the same way a `"matrix"` question's rows are grouped (`sframe_plot_likert_matrix()`): every member item is `"likert"` type and all share one choice set. Scales that mix response scales, that resolve to fewer than 2 qualifying items, or whose choice set cannot be found are left out and fall back to one chart per item in the report's Response distributions section.

Usage

```
sframe_likert_scale_groups(instrument)
```

Arguments

`instrument`	An `sframe` object.

Value

A named list, one entry per eligible scale (named by scale id), each a list with `scale_id`, `title` (the scale's label), `items` (the member item objects, in scale order), and `choice_set` (the shared choice set object). Empty list if no scale qualifies.

See Also

`sframe_plot_likert_scale()`, `sf_scale()`

sframe_plot_cooccurrence

Term co-occurrence heatmap

Description

Tile heatmap of pairwise within-response term co-occurrence counts for a `co_occurrence` result. The result's edge list (`term_a`, `term_b`, `n`) is pivoted into a full symmetric term-by-term grid before plotting, so each pair's tile appears twice, once on either side of the diagonal, the way the other tile heatmaps in this file (`sframe_plot_correlation_matrix()`, `sframe_plot_efa_loadings()`) read as a full grid rather than a triangle.

Usage

```
sframe_plot_cooccurrence(result, palette = c("web", "print"))
```

Arguments

`result`	A `co_occurrence` result list from `run_analysis_plan()`.
`palette`	One of `"web"` or `"print"`. See `sframe_brand()`.

Value

A ggplot2 object, or `NULL` when the result carries no table.

See Also

`run_analysis_plan()`, `term_frequency()`

sframe_plot_cooccurrence_network

Term co-occurrence network plot

Description

Plots a `co_occurrence_network` result's node table (`term`, `frequency`, `cluster`, `x`, `y`) as a network diagram: edges (from `result$edges`) as line segments underneath, nodes as points sized by term frequency and coloured by Louvain cluster, with term labels on the larger points only. Labelling every point on a dense network risks overlap chaos, so only the top 15 nodes by frequency are labelled; the full term list stays available in `result$table`.

Usage

```
sframe_plot_cooccurrence_network(result, palette = c("web", "print"))
```

Arguments

result	A co_occurrence_network result list from run_analysis_plan(), carrying table and edges.
palette	One of "web" or "print". See sframe_brand().

Details

Clusters beyond the first 8 (ranked largest first) are folded into a single "Other" bucket rather than cycling or interpolating a new hue, per the dataviz skill's categorical-colour guidance; see .sframe_cluster_palette().

Value

A ggplot2 object, or NULL when the result carries no table.

See Also

run_analysis_plan()

---

sframe_plot_correlation_matrix

Correlation matrix heatmap

Description

Computes and plots a full pairwise correlation matrix, independent of run_analysis_plan()'s pairwise correlation_pearson/_spearman/ _kendall runners (which plot one variable pair at a time via sframe_plot_correlation()). Useful directly, and as the visual companion to validity_report()'s discriminant-validity checks.

Usage

```
sframe_plot_correlation_matrix(
  data,
  vars,
  method = "pearson",
  palette = c("web", "print")
)
```

Arguments

data	A data frame of survey responses.
vars	Character vector of column names to correlate.
method	One of "pearson", "spearman", "kendall".
palette	One of "web" (diverging red/teal gradient) or "print" (white-to-black gradient by magnitude, signed label). See sframe_brand().

Value

A ggplot2 object.

See Also

validity_report()

`sframe_plot_decision_ranking` *Ranked-score bar chart for a decision-family result*

Description

The shared chart for every MCDM ranking method: one horizontal bar per alternative, ordered best first, with the leading alternative picked out. It is generic over the method rather than tied to one, so AHP criterion weights and any ranking method's scores all draw through it. The score column is whatever the method reports as its headline quantity (a closeness coefficient, a net flow, a priority weight), so the axis is labelled from the result rather than hard-coded.

Usage

```
sframe_plot_decision_ranking(result, palette = c("web", "print"))
```

Arguments

`result`	A decision-family result list from `run_analysis_plan()`, carrying either `scores` and `alternatives` or a ranking `table`.
`palette`	One of `"web"` or `"print"`. See `sframe_brand()`.

Value

A ggplot2 object, or `NULL` when the result carries no ranking.

See Also

run_analysis_plan()

sframe_plot_dematel_influence

Prominence-relation scatter for a DEMATEL result

Description

The dedicated DEMATEL chart: one point per criterion, prominence (D + R) on x and relation (D - R) on y, with a horizontal quadrant line at relation = 0 separating causes (above) from effects (below). This is deliberately a different shape from `sframe_plot_decision_ranking()`: DEMATEL classifies criteria on two axes rather than producing a single ranked score, so a ranking bar chart would misrepresent it.

Usage

```
sframe_plot_dematel_influence(result, palette = c("web", "print"))
```

Arguments

`result`	A `dematel`-family result list from `run_analysis_plan()` or `sframe_run_dematel()`, carrying `prominence`, `relation`, and `criteria`.
`palette`	One of `"web"` or `"print"`. See `sframe_brand()`.

Value

A ggplot2 object, or `NULL` when the result carries no DEMATEL fields to plot.

See Also

`sframe_plot_decision_ranking()`

sframe_plot_descriptives

Distribution shape by variable, standardised

Description

One violin per variable in a `descriptives_report()` table, built from the underlying response data rather than from the summary skewness and kurtosis numbers, so the reader sees the actual shape (asymmetry, multimodality, tails) instead of reading it off a bar height. Each variable is standardised (z-scored) before plotting so variables on different original scales (a 5-point Likert item next to a 0-100 slider) share one comparable y-axis. Standardising is a linear transform and does not change skewness. Each violin's subtitle-free panel keeps the variable's skewness value in its axis label. Grouped `descriptives_report()` output (one row per variable per `split_by` group) is faceted by group.

Usage

```
sframe_plot_descriptives(x, data, palette = c("web", "print"))
```

Arguments

x	An sframe_descriptives_report object from descriptives_report().
data	The same data.frame passed to descriptives_report(). Required: x only carries the summary table, not the raw values the violins need.
palette	One of "web" or "print". See sframe_brand().

Value

A ggplot2 object, or NULL if none of the report's variables have enough data to draw.

See Also

descriptives_report()

---

sframe_plot_efa_loadings

Loadings heatmap from a fitted EFA solution

Description

Loadings heatmap from a fitted EFA solution

Usage

```
sframe_plot_efa_loadings(x, palette = c("web", "print"))
```

Arguments

x	An sframe_efa_solution object from efa_solution().
palette	One of "web" (diverging red/teal gradient) or "print" (white-to-black gradient by magnitude, with sign conveyed by the printed label rather than colour, so it stays legible in monochrome). See sframe_brand().

Value

A ggplot2 object.

See Also

efa_solution()

sframe_plot_efa_scree *Scree plot from an EFA readiness report*

Description

Plots the parallel-analysis eigenvalues from `efa_report()` (both the observed factor-analysis eigenvalues and the simulated comparison line), with the suggested factor count marked.

Usage

```
sframe_plot_efa_scree(x, palette = c("web", "print"))
```

Arguments

x	An `sframe_efa_report` object from `efa_report()`.
palette	One of `"web"` or `"print"`. See `sframe_brand()`.

Value

A ggplot2 object.

See Also

`efa_report()`

sframe_plot_group_comparison *Group-comparison boxplot*

Description

Boxplot with jittered points, shared across every runner whose result carries `vars = c(group_column, outcome_column)`: `t_test_ind`, `mann_whitney`, `kruskal_wallis`, and `anova_one`. One function instead of four, since the underlying comparison (an outcome split by a grouping factor) and the data shape needed to plot it are identical across all four tests, and only the inferential statistic differs.

Usage

```
sframe_plot_group_comparison(result, data, palette = c("web", "print"))
```

Arguments

result	A result list from one of the four runners above, with `vars = c(group_column, outcome_column)`.
data	The response data frame the result was computed from.
palette	One of `"web"` or `"print"`. See `sframe_brand()`.

Value

A ggplot2 object, or NULL if the columns are missing, fewer than two groups remain after removing missing values, or ggplot2 is unavailable.

See Also

run_analysis_plan()

---

sframe_plot_likert_matrix

Grouped diverging chart for a Likert matrix question

Description

A matrix question asks several rows against one shared response scale (a "grid" of Likert items). Plotting each row as its own separate sframe_draw_likert_diverging() chart loses the grouping the question was designed with, so this draws every row as one diverging bar inside a single chart, sharing one x scale and one legend, the standard way a Likert matrix is reported (compare a typical multi-item satisfaction grid). Same diverging-stack convention as the single-item chart: the middle category of an odd-length scale is neutral and split evenly across the zero line, and colour saturation increases toward each pole.

Usage

```
sframe_plot_likert_matrix(item, data, choice_set, palette = c("web", "print"))
```

Arguments

item	A "matrix" sframe item, with matrix_items (the row labels) and a choice_set naming the shared response scale.
data	The response data.frame, with one expanded <item id>__<row label> column per matrix row, as produced by read_responses().
choice_set	The item's choice set object (values, labels), typically looked up from instrument$choices by item$choice_set.
palette	One of "web" or "print". See sframe_brand().

Value

A ggplot2 object, or NULL if no row has response data.

See Also

sframe_draw_likert_diverging()

`sframe_plot_likert_scale` *Grouped diverging chart for a scale's Likert items*

Description

Several *separate* Likert items that make up one `sf_scale()` (unlike a `"matrix"` item's rows, which are one question) are, by default, each reported as their own single-item diverging chart. That scatters a related batch of items (a satisfaction scale's 2-3 items, say) across several charts instead of showing them the way a Likert matrix or a typical multi-item satisfaction grid is reported: one grouped chart, one diverging bar per item, sharing an x scale and a legend. Applies only when every item in the scale shares the same choice set. Scales that mix response scales fall back to one chart per item.

Usage

```
sframe_plot_likert_scale(
  items,
  data,
  choice_set,
  title,
  palette = c("web", "print")
)
```

Arguments

`items`	A list of `"likert"` sframe items belonging to one scale, in display order.
`data`	The response data.frame, with one column per item id.
`choice_set`	The shared choice set object (`values`, `labels`).
`title`	Chart title, typically the scale's label.
`palette`	One of `"web"` or `"print"`. See `sframe_brand()`.

Value

A ggplot2 object, or `NULL` if no item has response data.

See Also

`sframe_plot_likert_matrix()`, `sf_scale()`

sframe_plot_missingness

Missing-data report plot: missingness rate by item

Description

Missing-data report plot: missingness rate by item

Usage

```
sframe_plot_missingness(x, palette = c("web", "print"))
```

Arguments

x	An sframe_missing_data_report object from missing_data_report().
palette	One of "web" or "print". See sframe_brand().

Value

A ggplot2 object. When no item has missing values, this is a short "no missing responses" message rather than an empty bar chart.

See Also

missing_data_report()

sframe_plot_ngram_frequency

N-gram-frequency plot: horizontal bar

Description

Top 20 n-grams from an ngram_freq result as a horizontal bar chart. Shares its bar-building logic with sframe_plot_term_frequency()'s bar path via the internal .sframe_plot_term_bar() helper; unlike that function, there is no word-cloud mode and no group faceting for this id.

Usage

```
sframe_plot_ngram_frequency(result, palette = c("web", "print"))
```

Arguments

result	An ngram_freq result list from run_analysis_plan().
palette	One of "web" or "print". See sframe_brand().

Value

A ggplot2 object, or NULL when the result carries no table.

See Also

run_analysis_plan(), ngram_frequency()

---

sframe_plot_paired_comparison

Paired-comparison slope plot

Description

One line per respondent connecting their two paired values, shared by t_test_pair and wilcoxon_pair (both carry vars = c(x_column, y_column) on the same respondents). The standard visual for a paired design: it shows the direction and consistency of individual change, which a plain bar-of-means would hide.

Usage

```
sframe_plot_paired_comparison(result, data, palette = c("web", "print"))
```

Arguments

result	A result list from t_test_pair/wilcoxon_pair, with vars = c(x_column, y_column).
data	The response data frame the result was computed from.
palette	One of "web" or "print". See sframe_brand().

Value

A ggplot2 object, or NULL if fewer than two complete pairs remain, or ggplot2 is unavailable.

See Also

run_analysis_plan()

sframe_plot_quality *Quality report plot: straight-lining flag rate by scale*

Description

Quality report plot: straight-lining flag rate by scale

Usage

```
sframe_plot_quality(x, palette = c("web", "print"))
```

Arguments

x	An sframe_quality_report object from quality_report().
palette	One of "web" or "print". See sframe_brand().

Value

A ggplot2 object.

See Also

quality_report()

sframe_plot_regression_diagnostics *Regression diagnostic plots for a regression_linear result*

Description

The four standard diagnostic panels (residuals vs fitted, normal Q-Q, scale-location, residuals vs leverage), built from the plain data frame run_analysis_plan() attaches to a regression_linear result rather than the lm object itself, so the result stays JSON-serialisable.

Usage

```
sframe_plot_regression_diagnostics(result, palette = c("web", "print"))
```

Arguments

result	A regression_linear result list containing a diagnostics data frame (as produced internally by run_analysis_plan()).
palette	One of "web" or "print". See sframe_brand().

Value

A named list of four ggplot2 objects (`residuals_fitted`, `qq`, `scale_location`, `leverage`), or `NULL` if diagnostics are unavailable.

See Also

`run_analysis_plan()`

`sframe_plot_reliability` *Reliability plot: alpha and omega by scale*

Description

Reliability plot: alpha and omega by scale

Usage

```
sframe_plot_reliability(x, palette = c("web", "print"))
```

Arguments

`x`	An `sframe_reliability_report` object from `reliability_report()`.
`palette`	One of `"web"` or `"print"`. See `sframe_brand()`.

Value

A ggplot2 object.

See Also

`reliability_report()`

`sframe_plot_sentiment` *Sentiment plot: diverging bar, or a positive/negative comparison cloud*

Description

A ggplot2 diverging bar for a `tidy_sentiment` result by default: positive counts extend one direction, negative counts the other, so bar position (not colour alone) carries the primary polarity signal, the same convention `sframe_draw_likert_diverging()` uses for Likert agreement (dark ramp toward the pole) rebuilt here in ggplot2 rather than called directly, since that helper is base-graphics and Likert-scale-specific. Facets by group when `result$table` carries a `group` column, mirroring `sframe_plot_term_frequency()`'s grouped branch.

Usage

```
sframe_plot_sentiment(result, palette = c("web", "print"))
```

Arguments

result	A tidy_sentiment result list from run_analysis_plan().
palette	One of "web" or "print". See sframe_brand().

Details

When result$options$wordcloud is TRUE (opt-in, default FALSE, matching sframe_plot_term_frequency()'s own word-cloud toggle), draws a comparison cloud instead: negative-sentiment words above the centre line, positive-sentiment words below it, each word sized by how often it occurred, using the internal tidy_sentiment runner's $word_sentiment word-by-sentiment counts. Answers a different question from the diverging bar: not "how many responses leaned positive," but "which *words* drove that."

Value

A ggplot2 object, or NULL when the result carries no table.

See Also

run_analysis_plan(), sframe_draw_likert_diverging()

---

sframe_plot_term_frequency

Term-frequency plot: horizontal bar or word cloud

Description

Top terms from a term_freq result as a horizontal bar chart, or a word cloud when result$options$wordcloud is TRUE (opt-in, default FALSE). Facets by group when the result carries a group role (todo_0.5.md section 1a).

Usage

```
sframe_plot_term_frequency(result, palette = c("web", "print"))
```

Arguments

result	A term_freq result list from run_analysis_plan().
palette	One of "web" or "print". See sframe_brand().

Value

A ggplot2 object, or NULL when the result carries no table.

See Also

run_analysis_plan(), term_frequency()

---

sframe_plot_topics *Topic-model top-terms plot: faceted bars, one facet per topic*

Description

Serves both sframe_run_topic_model_lda() and sframe_run_stm_topics() results with no dispatch on result$test: both runners emit a $table with the same topic/term/beta columns (LDA's beta from tidytext::tidy(), STM's from its fitted word-topic distribution), so this function reads that shared shape directly.

Usage

```
sframe_plot_topics(result, palette = c("web", "print"))
```

Arguments

result	A topic_model_lda or stm_topics result list from run_analysis_plan().
palette	One of "web" or "print". See sframe_brand().

Value

A ggplot2 object, or NULL when the result carries no usable table.

See Also

sframe_run_topic_model_lda(), sframe_run_stm_topics()

---

sframe_plot_validity *Validity report plot: composite reliability and AVE by construct*

Description

Validity report plot: composite reliability and AVE by construct

Usage

```
sframe_plot_validity(x, palette = c("web", "print"))
```

Arguments

x	An sframe_validity_report object from validity_report().
palette	One of "web" or "print". See sframe_brand().

Value

A ggplot2 object.

See Also

validity_report()

---

sframe_plot_variable_distribution

Raw-variable distribution panels: histogram, boxplot, and Q-Q

Description

Unlike sframe_plot_descriptives(), which summarises skewness and kurtosis *across* the variables in a descriptives_report() table, this operates on one variable's raw values directly (the report table only stores summary statistics, not the underlying vector), matching the pattern sframe_plot_correlation_matri already uses for report-independent, data-driven plots.

Usage

```
sframe_plot_variable_distribution(data, variable, palette = c("web", "print"))
```

Arguments

data	A data frame of survey responses.
variable	Character. Column name of the variable to plot.
palette	One of "web" or "print". See sframe_brand().

Value

A named list of three ggplot2 objects (histogram, boxplot, qq), or NULL if fewer than two complete values remain.

See Also

descriptives_report(), sframe_plot_descriptives()

sframe_rated_matrix *Build a performance matrix from rated matrix items*

Description

The third collection path: respondents rate every alternative on every criterion with ordinary matrix items, one item per criterion with the alternatives as its rows, and the decision matrix is the per-cell aggregate. No new item type is needed. Per-cell counts and standard deviations are kept so the report can show how firm each cell is.

Usage

```
sframe_rated_matrix(data, instrument, items, statistic = c("mean", "median"))
```

Arguments

data	A data frame of responses.
instrument	An sframe instrument declaring every id in items.
items	Character vector of "matrix" item ids, one per criterion, in the intended criterion order. Every item must declare the same matrix_items (the alternatives) in the same order.
statistic	"mean" or "median".

Value

A list with matrix (alternatives x criteria, with dimnames), n, sd, alternatives, criteria, and statistic.

See Also

sframe_collected_weights()

sframe_subset *Subset a surveyframe report*

Description

Keeps the report class, so a subset still prints as a report and still answers as.data.frame().

Usage

```
## S3 method for class 'sframe_analysis_results'
x[i, ...]

## S3 method for class 'sframe_reliability_report'
x[i, ...]

## S3 method for class 'sframe_item_report'
x[i, ...]
```

Arguments

x	An sframe_analysis_results, sframe_reliability_report, or sframe_item_report object.
i	Index, name, or logical vector.
...	Ignored. Present for S3 consistency.

Value

An object of the same class as x.

---

sframe_validation *Report on a validation result*

Description

sframe_validation is the diagnostic object returned by validate_sframe() and validate_model(). It records whether the object passed, every problem found, and every check that ran, including the checks that found nothing.

Usage

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

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

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

## S3 method for class 'sframe_validation'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

x, object	An sframe_validation object.
...	Ignored. Present for S3 consistency.
row.names	Passed to base::as.data.frame().
optional	Passed to base::as.data.frame().

Details

Use sf_is_valid() for the pass or fail flag, sf_problems() for the messages, as.data.frame() for a problem-per-row table, summary() for the full check roster, and as_sframe() to recover the validated instrument.

Value

print() returns x invisibly. format() returns a single character string. summary() returns the check table as a data frame. as.data.frame() returns one row per problem.

See Also

validate_sframe(), validate_model(), sf_problems(), sf_is_valid(), as_sframe()

Examples

```
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))

v <- validate_sframe(instr, strict = FALSE)
v
sf_is_valid(v)
sf_problems(v)
as.data.frame(v)
summary(v)
```

---

sf_accessors *Explore a surveyframe object*

Description

Accessors for the parts of an instrument, a codebook, or a report. They replace reaching into the object with $, which ties user code to the internal layout.

Usage

```
sf_meta(x, ...)

sf_items(x, ...)

sf_scales(x, ...)

sf_choice_sets(x, ...)

sf_branches(x, ...)

sf_checks(x, ...)

sf_models(x, ...)

sf_plan(x, ...)

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

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

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

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

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

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

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

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

## S3 method for class 'sframe_codebook'
sf_meta(x, ...)

## S3 method for class 'sframe_codebook'
sf_items(x, ...)

## S3 method for class 'sframe_codebook'
```

```
sf_scales(x, ...)

## S3 method for class 'sframe_codebook'
sf_choice_sets(x, ...)

## S3 method for class 'sframe_codebook'
sf_models(x, ...)

## S3 method for class 'sframe_codebook'
sf_plan(x, ...)
```

Arguments

x	A surveyframe object.
...	Passed to methods.

Details

`sf_items()`, `sf_scales()`, `sf_choice_sets()`, `sf_branches()`, `sf_checks()` and `sf_models()` return the component objects as an sf_component_list. `sf_meta()` returns the metadata as a list and `sf_plan()` returns the pre-declared analysis plan. For a flat table of the same content, call `as.data.frame()` on the object instead.

Value

`sf_items()`, `sf_scales()`, `sf_choice_sets()`, `sf_branches()`, `sf_checks()` and `sf_models()` return an sf_component_list. `sf_meta()` and `sf_plan()` return lists.

See Also

`as_sframe()`, `sf_problems()`, sframe_validation

Examples

```
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))

sf_meta(instr)$title
sf_items(instr)
sf_scales(instr)[["sat"]]
as.data.frame(instr)
```

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. 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_component_list *A list of instrument components*

Description

The value returned by sf_items(), sf_scales(), sf_choice_sets(), sf_branches(), sf_checks() and sf_models(). It is a list of component objects named by their IDs, so a single component is reached with [[.

Usage

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

## S3 method for class 'sf_component_list'
x[i, ...]
```

Arguments

x	An sf_component_list.
...	Ignored. Present for S3 consistency.
i	Index, name, or logical vector selecting components.

Value

print() returns x invisibly. [returns an sf_component_list.

Examples

```
item1 <- sf_item("q1", "First question", type = "text")
item2 <- sf_item("q2", "Second question", type = "text")
instr <- sf_instrument("Demo", components = list(item1, item2))

sf_items(instr)
sf_items(instr)[["q2"]]
```

---

sf_conjoint_design *Declare a choice-experiment (conjoint) design*

Description

Generates and stores a conjoint profile set and task schedule as part of the instrument's pre-declared contract. This is a design generator, not an estimator: it fixes what respondents will be shown, and it does not fit or analyse choice models.

Usage

```
sf_conjoint_design(
  id,
  attributes,
  method = c("full", "balanced", "random"),
  n_profiles = NULL,
  n_alternatives = 2L,
  n_tasks = NULL,
  blocks = 1L,
  seed = NULL,
  profiles = NULL,
  label = NULL
)
```

Arguments

`id`	Design identifier. Must start with a letter and contain only letters, numbers, and _ characters.
`attributes`	Named list of character vectors, one per attribute, giving that attribute's levels. At least 2 attributes, each with at least 2 levels.
`method`	One of `"full"`, `"balanced"`, or `"random"`. Ignored when `profiles` is supplied.
`n_profiles`	Number of profiles to keep. Required for `"balanced"` and `"random"`, ignored for `"full"`.
`n_alternatives`	Alternatives shown per choice task, default 2.
`n_tasks`	Choice tasks per block. Defaults to as many whole tasks as the profile set supports.
`blocks`	Number of blocks the tasks are split across, default 1.

`seed`	Integer seed. Generated and stored when not supplied.
`profiles`	Optional data frame of pre-built profiles, one column per attribute. Supplying this bypasses generation and declares the design as given.
`label`	Human-readable label.

Details

The design is reproducible by construction. `seed` is always recorded, and generated when not supplied, so regenerating from the stored declaration returns the identical profiles and tasks.

Value

An object of class `sf_conjoint_design` with `$profiles`, `$tasks` (long format, one row per block, task, and alternative, ready for a choice model), `$balance`, and the declaration that produced them.

Choosing a method

`"full"` enumerates every combination, which is exact but grows fast: 4 attributes at 3 levels each is 81 profiles. `"random"` takes a seeded random subset of that size. `"balanced"` samples repeatedly and keeps the subset with the most even level spread and the weakest association between attributes.

`"balanced"` is a search, not a construction. It does not produce a catalogued orthogonal fractional factorial and does not claim the guarantees of one. The achieved balance is reported in `$balance` so the design can be inspected rather than trusted. A study needing a specific D-optimal or orthogonal design should generate it elsewhere and pass it in through `profiles`, which keeps the declaration in the contract either way.

See Also

`sf_instrument()`

Examples

```
design <- sf_conjoint_design(
  "hotel_dce",
  attributes = list(
    price    = c("50", "100", "150"),
    board    = c("room only", "breakfast"),
    distance = c("beachfront", "10 min walk")
  ),
  method = "balanced", n_profiles = 6, n_alternatives = 2, seed = 42
)
design$profiles
design$tasks
```

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_identity *The ID and label of an instrument component*

Description

The ID and label of an instrument component

Usage

```
sf_id(x, ...)

sf_label(x, ...)

## S3 method for class 'sf_item'
sf_id(x, ...)

## S3 method for class 'sf_choices'
sf_id(x, ...)

## S3 method for class 'sf_scale'
sf_id(x, ...)

## S3 method for class 'sf_branch'
sf_id(x, ...)

## S3 method for class 'sf_check'
sf_id(x, ...)

## S3 method for class 'sf_model'
sf_id(x, ...)

## S3 method for class 'sf_item'
sf_label(x, ...)

## S3 method for class 'sf_choices'
sf_label(x, ...)
```

```
## S3 method for class 'sf_scale'
sf_label(x, ...)

## S3 method for class 'sf_branch'
sf_label(x, ...)

## S3 method for class 'sf_check'
sf_label(x, ...)

## S3 method for class 'sf_model'
sf_label(x, ...)
```

Arguments

x	An sf_item(), sf_choices(), sf_scale(), sf_branch(), sf_check() or sf_model() object.
...	Passed to methods.

Value

A single character string. sf_label() returns "" when the component carries no label.

Examples

```
item <- sf_item("q1", "How satisfied are you?", type = "likert",
                choice_set = "agree5")
sf_id(item)
sf_label(item)
```

---

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"))

visitor_cs <- sf_choices("visitor", c("new", "returning"),
                         c("New visitor", "Returning visitor"))

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)
item3 <- sf_item("visitor_type", "I am a",
                 type = "single_choice", choice_set = "visitor")

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

# The analysis_plan binds each research question to a statistical method
# and the variable roles it needs. Declare it before any data arrive.
plan <- list(
  list(
    id               = "RQ1",
    research_question = "Do new and returning visitors differ in satisfaction?",
    family           = "group_comparison",
    method           = "mann_whitney",
    roles            = list(group = "visitor_type", outcome = "sat"),
    options          = list(alpha = 0.05)
  )
)

instr <- sf_instrument(
  title         = "Service Quality Survey",
  version       = "1.0.0",
  components    = list(choices, visitor_cs, item1, item2, item3, scale),
  analysis_plan = plan
)
print(instr)
length(sf_plan(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", "pairwise_comparison",
    "criteria_weight", "section_break", "text_block"),
  required = FALSE,
  choice_set = NULL,
  scale_id = NULL,
  reverse = FALSE,
  help = NULL,
  placeholder = NULL,
  matrix_items = NULL,
  comparison_items = NULL,
  comparison_scale = NULL,
  slider_min = NULL,
  slider_max = NULL,
  slider_step = NULL,
  rating_max = NULL,
  rating_icon = NULL,
  date_min = NULL,
  date_max = 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", "pairwise_comparison", "criteria_weight", "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.
comparison_items	Character vector or NULL. The things being compared or weighted, for "pairwise_comparison" and "criteria_weight" types. At least 2 entries, all distinct. A "saaty" pairwise item renders n(n-1)/2 unordered pair rows, an "influence" item renders n(n-1) ordered rows, and a "criteria_weight" item renders one numeric input per entry. An advisory warning is raised above 7 items ("saaty") or 6 ("influence"), and above 10 the item is rejected.
comparison_scale	Character or NULL. For "pairwise_comparison" only. "saaty" (the default) gives the bipolar 1-9 importance scale used by AHP and ANP, while "influence" gives the unipolar 0-4 directed influence scale used by DEMATEL.
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".
date_min	Character or Date or NULL. Earliest selectable date for "date" type, as "YYYY-MM-DD".
date_max	Character or Date or NULL. Latest selectable date for "date" type, as "YYYY-MM-DD".
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_plan<- — *Set the pre-declared analysis plan*

Description

The replacement counterpart to sf_plan(). Declaring the plan is the step the whole workflow turns on, so it has a named function rather than assignment into the object's internals.

Usage

```
sf_plan(x) <- value

## S3 replacement method for class 'sframe'
sf_plan(x) <- value
```

Arguments

x	An sframe object.
value	A list of analysis blocks.

Value

The updated sframe object.

See Also

sf_plan(), validate_sframe(), run_analysis_plan()

Examples

```
item  <- sf_item("q1", "How satisfied are you?", type = "numeric")
instr <- sf_instrument("Demo", components = list(item))

sf_plan(instr) <- list(
  list(id = "RQ1", research_question = "What is the average?",
       family = "descriptive", method = "descriptives",
       roles = list(variables = "q1"))
)
length(sf_plan(instr))
```

---

sf_report_accessors *Read the reportable parts of an analysis or quality result*

Description

sf_apa() returns the APA-formatted sentence for each analysis block. sf_flagged() returns the row numbers a quality report flagged.

Usage

```
sf_apa(x, ...)

sf_flagged(x, ...)

## S3 method for class 'sframe_analysis_results'
sf_apa(x, ...)

## S3 method for class 'sframe_descriptives_report'
sf_apa(x, ...)

## S3 method for class 'sframe_missing_data_report'
sf_apa(x, ...)

## S3 method for class 'sframe_validity_report'
sf_apa(x, ...)

## S3 method for class 'sframe_assumption_report'
sf_apa(x, ...)

## S3 method for class 'sframe_quality_report'
sf_flagged(x, ...)
```

Arguments

x	An sframe_analysis_results object for sf_apa(), or an sframe_quality_report for sf_flagged().
...	Passed to methods.

Value

sf_apa() returns a named character vector, one element per analysis block. sf_flagged() returns an integer vector of row numbers.

Examples

```
demo <- sframe_demo_data()
qr <- quality_report(demo$responses, demo$instrument)
head(sf_flagged(qr))
```

---

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, while 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
)
```

---

sf_validation_accessors

Read a validation diagnostic

Description

sf_is_valid() reports whether the object passed. sf_problems() returns the problem messages. sf_object() returns the object that was validated.

Usage

```
sf_is_valid(x, ...)

sf_problems(x, ...)

sf_object(x, ...)

## S3 method for class 'sframe_validation'
sf_is_valid(x, ...)
```

```
## S3 method for class 'sframe_validation'
sf_problems(x, ...)

## S3 method for class 'sframe_validation'
sf_object(x, ...)
```

Arguments

x	An sframe_validation object.
...	Passed to methods.

Value

`sf_is_valid()` returns a single logical. `sf_problems()` returns a character vector, empty when the object is valid. `sf_object()` returns the validated object.

See Also

`validate_sframe()`, sframe_validation, `as_sframe()`

Examples

```
item  <- sf_item("q1", "How satisfied are you?", type = "text")
instr <- sf_instrument("Demo", components = list(item))
v <- validate_sframe(instr, strict = FALSE)

sf_is_valid(v)
sf_problems(v)
```

---

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)
```

---

summary.sf_branch *Summarise an sf_branch object*

Description

Summarise an sf_branch object

Usage

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

Arguments

object	An object of class sf_branch.
...	Ignored. Present for S3 consistency.

Value

object, invisibly.

---

summary.sf_check *Summarise an sf_check object*

Description

Summarise an sf_check object

Usage

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

Arguments

object	An object of class sf_check.
...	Ignored. Present for S3 consistency.

Value

object, invisibly.

---

summary.sf_choices *Summarise an sf_choices object*

Description

Summarise an sf_choices object

Usage

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

Arguments

object	An object of class sf_choices.
...	Ignored. Present for S3 consistency.

Value

object, invisibly.

---

summary.sf_item *Summarise an sf_item object*

Description

Summarise an sf_item object

Usage

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

Arguments

object	An object of class sf_item.
...	Ignored. Present for S3 consistency.

Value

`object`, invisibly.

`summary.sf_model` *Summarise an sf_model object*

Description

Summarise an sf_model object

Usage

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

Arguments

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

Value

`object`, invisibly.

`summary.sf_scale` *Summarise an sf_scale object*

Description

Summarise an sf_scale object

Usage

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

Arguments

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

Value

`object`, invisibly.

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

```
## Not run:
# 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)

## End(Not run)
```

term_context *Keyword-in-context concordance for open-ended text*

Description

Finds every case-insensitive, whole-word match of `term` in `text` and returns the words immediately before and after each match, so a reader can judge how a term is actually being used rather than reading a bare frequency count. Matching is on whole words only, so searching for "room" does not match "roomy". `text` is expected to already have been run through `clean_text_responses()` (its `respondent` attribute is used to cite the original row index for each match; when absent, positions `seq_along(text)` are used instead). Tokenisation here is a plain whitespace split, not the internal stop-word-stripping tokeniser behind `term_frequency()`: stop words are part of a match's context and stripping them would corrupt the very thing a concordance is for.

Usage

```
term_context(text, term, window = 6L, max_matches = 20L)
```

Arguments

`text`	Character vector of responses, ideally already cleaned by `clean_text_responses()`.
`term`	Character. A single keyword to search for.
`window`	Integer. Maximum number of words of context to keep before and after each match. Default `6`.
`max_matches`	Integer. Maximum number of matches to return, counted across all responses. Default `20`.

Value

A data.frame with columns `respondent`, `before`, `match`, and `after`.

term_frequency *Term frequency for open-ended text*

Description

Tokenises `text` (splitting on whitespace, lower-casing, and stripping punctuation), removes stop words, and counts term frequency. Ships a small built-in English stop-word list so this works with zero optional packages; pass `stop_words = character(0)` to disable filtering, or a custom vector to override it.

Usage

```
term_frequency(text, stop_words = NULL, top_n = 30L)
```

Arguments

text	Character vector of responses (raw or already cleaned by clean_text_responses()).
stop_words	Character vector of words to exclude, or NULL to use the built-in English list, or character(0) for no filtering.
top_n	Integer. Maximum number of terms to return, most frequent first. Default 30.

Value

A data.frame with columns term, n, and pct.

---

theme_surveyframe *surveyframe brand theme for ggplot2*

Description

A theme_classic()-based ggplot2 theme (visible axis lines, no floating panel), verified against WCAG 2.2 contrast minimums: 4.5:1 for text, 3:1 for non-text graphical objects. Apply it to any ggplot object, including the plots returned by run_analysis_plan() when plots = TRUE.

Usage

```
theme_surveyframe(
  base_size = 12,
  base_family = "",
  palette = c("web", "print")
)
```

Arguments

base_size	Numeric. Base font size in points. Defaults to 12.
base_family	Character. Base font family. Defaults to "" (the device default).
palette	One of "web" (brand colours, for on-screen use) or "print" (black/grey/white only, for journal-ready figures). See sframe_brand() for the verified contrast ratios behind each.

Value

A ggplot2 theme object.

See Also

run_analysis_plan()

Examples

```
library(ggplot2)
ggplot(mtcars, aes(wt, mpg)) +
  geom_point(colour = "#0E9694") +
  theme_surveyframe()
```

`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`, problems are reported in the returned diagnostic without stopping.

Value

An sframe_validation object. The model is carried inside it and can be recovered with `sf_object()`.

Changed in 0.4.0

Earlier versions returned the model invisibly when `strict = TRUE` and a bare unclassed list when `strict = FALSE`. Both paths now return an sframe_validation object, visibly, so the diagnostic is readable at the console. Code that read `$valid` and `$problems` keeps working.

See Also

sframe_validation, `sf_problems()`, `sf_is_valid()`

`validate_sframe` *Validate an instrument object*

Description

Checks the internal consistency of an `sframe` instrument object and returns a diagnostic result. 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 reported in the returned diagnostic 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

An sframe_validation object. When the instrument is valid, the instrument carried inside it has `meta$validated` set to `TRUE` and can be recovered with `as_sframe()`.

Changed in 0.4.0

Earlier versions returned two different things depending on `strict`: the instrument itself, invisibly, when `strict = TRUE`, and a bare unclassed list when `strict = FALSE`. A validator should report a diagnostic, so both paths now return an sframe_validation object, and they return it visibly, so `validate_sframe(instrument)` typed at the console shows the result. Code that read `$valid` and `$problems` keeps working. Code that used the `strict = TRUE` return as an instrument should now wrap the call in `as_sframe()`.

See Also

sframe_validation, `as_sframe()`, `sf_problems()`, `sf_is_valid()`, `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))

# The result prints its own diagnostic
validate_sframe(instr, strict = FALSE)

# Explore it with dedicated methods rather than reaching in with `$`
v <- validate_sframe(instr, strict = FALSE)
sf_is_valid(v)
sf_problems(v)
summary(v)

# Recover the validated instrument
validated <- as_sframe(validate_sframe(instr, strict = TRUE))
isTRUE(sf_meta(validated)$validated)
```

---

validity_report *Validity report for construct models*

Description

Validity report for construct models

Usage

```
validity_report(loadings, construct_scores = NULL, items_by_construct = 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 and inter-construct correlations.
items_by_construct	Optional named list, one element per construct, each a data.frame of that construct's item-level responses. When supplied, htmt is the Henseler heterotrait-monotrait ratio: the mean absolute heterotrait-heteromethod correlation over the geometric mean of the two constructs' mean absolute monotrait-heteromethod correlations. Constructs with a single item have no monotrait correlations, so their HTMT entries are NA. Without this argument, htmt falls back to the absolute inter-construct correlation matrix from construct_scores (the pre-0.3.4 behaviour). The htmt_method element records which was computed.

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