Package 'LISTC'

July 28, 2026

Type Package

Title Pivot-Style Statistical Tables for Large-Scale Assessment Data

Version 1.0.0

Description Turns assessment and survey sample data (demographics, scores, sampling weights, ability estimates with individual item response theory standard errors, replicate weights and plausible values) into fully customizable pivot-style statistical tables in which every cell carries a design-appropriate standard error. Provides weighted means, proportions above cut scores, proficiency-level percentages and quantiles; sampling variance via linearization, Woodruff (1952) <doi:10.1080/01621459.1952.10483443> intervals for quantiles, or balanced repeated replication and jackknife replicate weights including Fay's method; measurement variance via delta-method propagation of individual standard errors or Rubin (1987) <doi:10.1002/9780470316696> combination across plausible values. Imports data from 'CSV', 'Excel', 'SPSS', 'SAS', 'Stata' and item response theory software person files ('Winsteps', 'ConQuest'); a configuration-file interface serves non-programmers and automation; results export to formatted 'Excel', 'JSON' and standalone 'HTML' reports with rule-based plain-language interpretation.

License GPL (>= 2)

Copyright See file inst/COPYRIGHTS.

Encoding UTF-8

Language zh-CN

Depends R (>= 4.1)

Imports data.table, haven, jsonlite, openxlsx, readxl, rlang, stats, tibble, utils, yaml

Suggests covr, knitr, readr, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/weiandata/LISTC

BugReports https://github.com/weiandata/LISTC/issues

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Kunxiang Ma [aut, cre],
WEIAN DATA TECH (Beijing) Co., Ltd. [cph, fnd]

Maintainer Kunxiang Ma <makunxiang@weiandata.com>

Repository CRAN

Date/Publication 2026-07-28 16:10:17 UTC

Contents

as_long — *Extract the tidy long form of a listc_table*

Description

One row per group combination x statistic x category, with columns estimate, se_sampling, se_measurement, se_total, n, sum_w.

Usage

```
as_long(tab)
```

Arguments

tab	A listc_table.

Value

A tibble.

Examples

```
d <- data.frame(id = 1:100, region = rep(c("north", "south"), 50),
                w = runif(100, 0.5, 2), theta = rnorm(100),
                se = runif(100, 0.2, 0.4))
x <- lst_data(d, id = id, group = region, weight = w,
              theta = c(math = theta), theta_se = c(math = se))
tab <- lst_table(x, rows = region, values = list(mean = st_mean(math)))
as_long(tab)
```

as_wide — *Extract the wide (row x column layout) form of a listc_table*

Description

Cells are formatted according to the table's format and digits; proportion-type statistics are shown as percentages.

Usage

```
as_wide(tab)
```

Arguments

tab	A listc_table.

Value

A tibble laid out as declared in `lst_table()`.

Examples

```
d <- data.frame(id = 1:100, region = rep(c("north", "south"), 50),
                w = runif(100, 0.5, 2), theta = rnorm(100),
                se = runif(100, 0.2, 0.4))
x <- lst_data(d, id = id, group = region, weight = w,
              theta = c(math = theta), theta_se = c(math = se))
tab <- lst_table(x, rows = region, values = list(mean = st_mean(math)))
as_wide(tab)
```

`lst_above` *Add an above-cutoff indicator variable*

Description

Uses `x >= cutoff` (reaching the cut score counts as above).

Usage

```
lst_above(x, var, cutoff, name = NULL)
```

Arguments

`x`	A `listc_data` object.
`var`	Measure: theta/score dimension name or numeric column.
`cutoff`	Numeric threshold.
`name`	Name of the new column (default `<var>_above`).

Value

`x` with a 0/1 indicator column added.

Examples

```
d <- data.frame(id = 1:100, region = rep(c("north", "south"), 50),
                w = runif(100, 0.5, 2), theta = rnorm(100),
                se = runif(100, 0.2, 0.4))
x <- lst_data(d, id = id, group = region, weight = w,
              theta = c(math = theta), theta_se = c(math = se))
x <- lst_above(x, math, cutoff = 1)
mean(x$data$math_above)
```

lst_classify *Classify a variable into proficiency levels by cut scores*

Description

Boundary convention: reaching a cut score places the person in the higher level (`x >= lower & x < upper`).

Usage

```
lst_classify(x, var, breaks, labels = NULL, name = NULL)
```

Arguments

`x`	A `listc_data` object.
`var`	Measure: theta/score dimension name or numeric column.
`breaks`	Named numeric vector: level name -> lower bound, e.g. `c(fail = -Inf, pass = 0.5, excellent = 1.2)`.
`labels`	Optional level labels (defaults to names of `breaks`).
`name`	Name of the new column (default `<var>_level`).

Value

`x` with an ordered-factor level column added.

Examples

```
d <- data.frame(id = 1:100, region = rep(c("north", "south"), 50),
                w = runif(100, 0.5, 2), theta = rnorm(100),
                se = runif(100, 0.2, 0.4))
x <- lst_data(d, id = id, group = region, weight = w,
              theta = c(math = theta), theta_se = c(math = se))
x <- lst_classify(x, math, c(low = -Inf, mid = -0.5, high = 0.8))
table(x$data$math_level)
```

lst_config *Read and validate a LISTC run configuration*

Description

Accepts a YAML/JSON file path, a YAML/JSON string, or a named list. Validation errors are reported in plain Chinese pointing to the offending field. (Excel configuration workbooks arrive in v0.2.)

Usage

```
lst_config(config)
```

Arguments

config	Path to .yml/.yaml/.json, a YAML/JSON string, or a list.

Value

A validated listc_config object.

Examples

```
cfg <- lst_config(list(
  data = "students.csv",
  roles = list(id = "id", weight = "w"),
  tables = list(list(name = "t1",
                     values = list(n = list(stat = "st_count"))))
))
class(cfg)
```

---

lst_config_template *Copy the Excel configuration template to a writable location*

Description

Copy the Excel configuration template to a writable location

Usage

```
lst_config_template(path = NULL, overwrite = FALSE)
```

Arguments

path	Target path for the template workbook; NULL uses a built-in default file name (Chinese, ending in ".xlsx").
overwrite	Overwrite an existing file.

Value

path, invisibly.

Examples

```
f <- file.path(tempdir(), "listc-config.xlsx")
lst_config_template(f, overwrite = TRUE)
file.exists(f)
```

`lst_data` *Create a listc_data object*

Description

Attaches variable roles (id, group, weight, score, theta, theta_se, resp) to a data frame. Columns can be given as bare names or character vectors; theta/theta_se accept named vectors defining dimensions, e.g. `theta = c(math = th_math)`.

Usage

```
lst_data(
  data,
  id = NULL,
  group = NULL,
  weight = NULL,
  score = NULL,
  theta = NULL,
  theta_se = NULL,
  resp = NULL,
  key = NULL,
  rep_weights = NULL,
  rep_method = NULL,
  fay_k = 0.5,
  pv = NULL,
  pv_sampling = c("first", "average")
)
```

Arguments

`data`	A data.frame.
`id`	Sample identifier column.
`group`	Demographic/grouping columns.
`weight`	Sampling weight column (defaults to 1 for all rows).
`score`	Observed score column(s).
`theta`	Ability estimate column(s), named by dimension.
`theta_se`	IRT standard error column(s), paired with `theta`.
`resp`	Item response columns.
`key`	Optional scoring key for `resp` (named vector: item -> correct answer), required for `st_pvalue()`.
`rep_weights`	Replicate weight columns (v0.3): a character vector of column names, bare names, or a single prefix string such as `"W_FSTR"` which expands to `W_FSTR1..R` (PISA style). When set, sampling variance switches to the replicate-weights engine.

rep_method	Replicate method: "fay" (PISA BRR-Fay), "brr", "jk1", or "jk2" (TIMSS). Required with rep_weights.
fay_k	Fay factor for rep_method = "fay" (default 0.5).
pv	Plausible values (v0.4): named list, dimension -> PV columns or a template with # for the PV number, e.g. pv = list(math = "PV#MATH") expands to PV1MATH..PV10MATH. PV dimensions are analyzed with Rubin combination.
pv_sampling	Sampling-variance convention for PV dimensions: "first" (PISA manual practice: first PV only) or "average" (average across all PVs).

Value

A listc_data object.

Examples

```
d <- data.frame(id = 1:100, region = rep(c("north", "south"), 50),
                w = runif(100, 0.5, 2), theta = rnorm(100),
                se = runif(100, 0.2, 0.4))
x <- lst_data(d, id = id, group = region, weight = w,
              theta = c(math = theta), theta_se = c(math = se))
x
```

---

lst_derive *Add arbitrary derived variables (mutate-style)*

Description

Add arbitrary derived variables (mutate-style)

Usage

```
lst_derive(x, ...)
```

Arguments

x	A listc_data object.
...	Name-value expressions evaluated in the data.

Value

x with derived columns added.

Examples

```
d <- data.frame(id = 1:5, part1 = 1:5, part2 = 6:10)
x <- lst_data(d, id = id)
x <- lst_derive(x, total = part1 + part2)
x$data$total
```

lst_interpret *Rule-based plain-language interpretation of a listc_table*

Description

Generates descriptive conclusions from templated rules (no LLM): highest/lowest groups, significant differences (difference > 2 x combined SE), and small-sample warnings (n < 30). v0.1 covers scalar statistics (mean, sd, prop_above); level/item statistics gain rules in v0.2.

Usage

```
lst_interpret(tab, lang = c("zh", "en"))
```

Arguments

tab	A listc_table.
lang	Output language; v0.1 supports "zh".

Value

Character vector of interpretation sentences.

Examples

```
d <- data.frame(id = 1:100, region = rep(c("north", "south"), 50),
                w = runif(100, 0.5, 2), theta = rnorm(100),
                se = runif(100, 0.2, 0.4))
x <- lst_data(d, id = id, group = region, weight = w,
              theta = c(math = theta), theta_se = c(math = se))
tab <- lst_table(x, rows = region, values = list(mean = st_mean(math)))
lst_interpret(tab)
```

lst_join_person *Join person parameters onto a listc_data by id*

Description

Merges a person-parameter table (columns id, theta, theta_se) onto the data by the declared id role and registers the theta/theta_se roles.

Usage

```
lst_join_person(x, person, dim = "theta")
```

Arguments

x	A listc_data object.
person	A data.frame with columns id, theta, theta_se.
dim	Dimension name for the merged theta (default "theta").

Value

x with theta/theta_se roles filled.

Examples

```
d <- data.frame(id = c("S001", "S002"), grade = c("G4", "G4"))
x <- lst_data(d, id = id, group = grade)
person <- data.frame(id = c("S001", "S002"),
                     theta = c(0.5, -1.0), theta_se = c(0.4, 0.45))
x <- lst_join_person(x, person, dim = "math")
x$roles$theta
```

---

lst_run *One-shot entry point: run a full LISTC analysis from a configuration*

Description

Reads the config, imports only the needed columns, applies roles, computes all requested tables and writes all requested outputs (xlsx and/or json). Primary interface for non-R users and AI agents.

Usage

```
lst_run(config, quiet = FALSE)
```

Arguments

config	Anything accepted by lst_config().
quiet	Suppress progress messages.

Value

Invisibly, list(tables = <named listc_table list>, log = <list>).

Examples

```
csv <- tempfile(fileext = ".csv")
write.csv(data.frame(id = 1:60, g = rep(c("a", "b"), 30),
                     score = rnorm(60, 50, 10)), csv,
          row.names = FALSE)
out <- tempfile(fileext = ".json")
res <- lst_run(list(
  data = csv,
```

```
  roles = list(id = "id", group = list("g")),
  tables = list(list(name = "t1", rows = list("g"),
                     values = list(mean = list(stat = "st_mean",
                                               var = "score")))),
  output = list(json = out)
), quiet = TRUE)
res$log$tables
```

`lst_table` *Build a pivot-style statistical table*

Description

Excel-pivot-like interface: declare row variables, column variables and cell statistics; every cell carries estimate and SE components (sampling + measurement, design doc 6).

Usage

```
lst_table(
  x,
  rows = NULL,
  cols = NULL,
  values,
  format = "est_se",
  digits = 2,
  margins = FALSE
)
```

Arguments

`x`	A `listc_data` object.
`rows`	Row grouping variables (bare names or character vector).
`cols`	Column grouping variables.
`values`	Named list of statistic specs (`st_*` functions).
`format`	Cell display: `"est"`, `"est_se"`, `"est_ci"`, `"percent"`.
`digits`	Rounding digits (single value, or named per-statistic).
`margins`	Add "Total" row/column margins.

Value

A `listc_table` object; see `as_long()` and `as_wide()`.

Examples

```
d <- data.frame(id = 1:100, region = rep(c("north", "south"), 50),
                w = runif(100, 0.5, 2), theta = rnorm(100),
                se = runif(100, 0.2, 0.4))
x <- lst_data(d, id = id, group = region, weight = w,
              theta = c(math = theta), theta_se = c(math = se))
tab <- lst_table(x, rows = region, values = list(
  mean = st_mean(math),
  above = st_prop_above(math, cutoff = 1, method = "prob"),
  n = st_count()
), margins = TRUE)
tab
```

`lst_to_excel` *Export listc_table(s) to a formatted Excel workbook*

Description

Chinese-friendly defaults: DengXian base font and column widths estimated from full-width character counts; override via `style`. A final interpretation sheet (titled "conclusions" in Chinese) carries the rule-based interpretation.

Usage

```
lst_to_excel(tab, path, style = NULL, overwrite = FALSE, interpret = TRUE)
```

Arguments

`tab`	A `listc_table` or named list of them (names become sheets).
`path`	Output .xlsx path.
`style`	Optional list of overrides: `font`, `font_size`, `header_fill`.
`overwrite`	Overwrite existing file.
`interpret`	Include the interpretation sheet.

Value

`path`, invisibly.

Examples

```
d <- data.frame(id = 1:100, region = rep(c("north", "south"), 50),
                w = runif(100, 0.5, 2), theta = rnorm(100),
                se = runif(100, 0.2, 0.4))
x <- lst_data(d, id = id, group = region, weight = w,
              theta = c(math = theta), theta_se = c(math = se))
tab <- lst_table(x, rows = region, values = list(mean = st_mean(math)))
f <- tempfile(fileext = ".xlsx")
lst_to_excel(tab, f, overwrite = TRUE)
file.exists(f)
```

lst_to_html *Export listc_table(s) as a standalone styled HTML report*

Description

Dependency-free HTML rendering (v0.4): Chinese-friendly fonts, zebra-striped pivot tables, an interpretation section per table and a methods footnote describing the variance engine in use. Suitable for emailing or embedding in Rmd/Quarto via `htmltools::HTML`.

Usage

```
lst_to_html(tab, path = NULL, title = NULL, interpret = TRUE)
```

Arguments

`tab`	A `listc_table` or named list of them.
`path`	Optional output .html path; when `NULL`, returns the HTML string invisibly.
`title`	Report title; `NULL` uses a built-in default title (Chinese for "statistical results").
`interpret`	Include the rule-based interpretation section.

Value

HTML string, invisibly.

Examples

```
d <- data.frame(id = 1:100, region = rep(c("north", "south"), 50),
                w = runif(100, 0.5, 2), theta = rnorm(100),
                se = runif(100, 0.2, 0.4))
x <- lst_data(d, id = id, group = region, weight = w,
              theta = c(math = theta), theta_se = c(math = se))
tab <- lst_table(x, rows = region, values = list(mean = st_mean(math)))
f <- tempfile(fileext = ".html")
lst_to_html(tab, f, title = "Report")
file.exists(f)
```

lst_to_json *Export listc_table(s) as machine-readable JSON (for AI agents)*

Description

Emits tidy long results plus metadata for every statistic (type, variable, method, correction, rho) and an `interpretation` field from `lst_interpret()`.

Usage

```
lst_to_json(tab, path = NULL, pretty = TRUE)
```

Arguments

tab	A listc_table or named list of them.
path	Optional output path; when NULL, returns the JSON string.
pretty	Pretty-print JSON.

Value

JSON string (invisibly when written to path).

Examples

```
d <- data.frame(id = 1:100, region = rep(c("north", "south"), 50),
                w = runif(100, 0.5, 2), theta = rnorm(100),
                se = runif(100, 0.2, 0.4))
x <- lst_data(d, id = id, group = region, weight = w,
              theta = c(math = theta), theta_se = c(math = se))
tab <- lst_table(x, rows = region, values = list(mean = st_mean(math)))
substr(lst_to_json(tab, pretty = FALSE), 1, 80)
```

---

lst_validate *Validate a listc_data object*

Description

Checks role pairing (theta/theta_se), non-negative weights, unique ids.

Usage

```
lst_validate(x)
```

Arguments

x	A listc_data object.

Value

x, invisibly on success.

Examples

```
d <- data.frame(id = 1:100, region = rep(c("north", "south"), 50),
                w = runif(100, 0.5, 2), theta = rnorm(100),
                se = runif(100, 0.2, 0.4))
x <- lst_data(d, id = id, group = region, weight = w,
              theta = c(math = theta), theta_se = c(math = se))
lst_validate(x)
```

read_conquest_person *Read ConQuest person estimate output (WLE/EAP tables)*

Description

Reads whitespace-delimited ConQuest person files (e.g. `show cases` output or `.wle` files). ConQuest files rarely carry headers, so the column positions are declared explicitly and default to the common 6-column WLE layout: seq, id, score, max, theta, se.

Usage

```
read_conquest_person(path, cols = c(id = 2, theta = 5, theta_se = 6))
```

Arguments

`path`	ConQuest person file path.
`cols`	Named integer vector giving 1-based column positions for `id`, `theta`, `theta_se`.

Value

Tibble with columns id, theta, theta_se.

Examples

```
f <- tempfile(fileext = ".wle")
writeLines(c("1 S001 12.00 20.00 0.523 0.412",
             "2 S002 6.00 20.00 -1.031 0.437"), f)
read_conquest_person(f)
```

read_listc *Read sample data from common file formats*

Description

Dispatches on file extension: csv/tsv/txt (`data.table::fread`), xlsx/xls (readxl), sav/zsav/dta/sas7bdat (haven, value labels are converted to factors). Use `col_select` to load only needed columns (design doc 9.1).

Usage

```
read_listc(path, col_select = NULL, ...)
```

Arguments

`path`	File path.
`col_select`	Optional character vector of columns to read.
`...`	Passed to the backend reader.

Value

A tibble.

Examples

```
f <- tempfile(fileext = ".csv")
write.csv(data.frame(id = 1:3, score = c(10, 12, 9)), f,
          row.names = FALSE)
read_listc(f)
read_listc(f, col_select = "score")
```

`read_winsteps_pfile` *Read a Winsteps PFILE (person parameter file)*

Description

Handles both fixed/whitespace PFILE output and csv PFILE (`PFILE=xxx.csv`). Comment lines starting with ; are skipped; the header row is located by the presence of a `MEASURE` column. The SE column is taken from `ERROR` (or `MODLSE`), the id from `NAME` (falling back to `ENTRY`).

Usage

```
read_winsteps_pfile(path, id_col = NULL, theta_col = NULL, se_col = NULL)
```

Arguments

`path`	PFILE path.
`id_col`, `theta_col`, `se_col`	Optional column-name overrides.

Value

Tibble with columns id, theta, theta_se plus the remaining PFILE columns.

Examples

```
f <- tempfile(fileext = ".txt")
writeLines(c("; PERSON FILE",
             ";ENTRY MEASURE COUNT SCORE ERROR NAME",
             "1 0.52 20 12 0.41 S001",
             "2 -1.03 20 6 0.44 S002"), f)
read_winsteps_pfile(f)
```

st_count *Count and weighted-count statistics*

Description

Count and weighted-count statistics

Usage

```
st_count()

st_wcount()
```

Value

A statistic spec for `lst_table()`.

Examples

```
d <- data.frame(id = 1:100, region = rep(c("north", "south"), 50),
                w = runif(100, 0.5, 2), theta = rnorm(100),
                se = runif(100, 0.2, 0.4))
x <- lst_data(d, id = id, group = region, weight = w,
              theta = c(math = theta), theta_se = c(math = se))
lst_table(x, rows = region, values = list(
  n = st_count(), wn = st_wcount()
))
```

st_level_prop *Proportions in proficiency levels*

Description

Proportions in proficiency levels

Usage

```
st_level_prop(
  var,
  breaks,
  method = c("hard", "prob"),
  correction = c("none", "latent"),
  rho = NULL
)
```

Arguments

var	Measure: a theta/score dimension name or a numeric column.
breaks	Named numeric vector: level name -> lower bound, e.g. `c(L1 = -Inf, L2 = 0.5, L3 = 1.2)`.
method	`"hard"` (0/1 classification) or `"prob"` (probabilistic, uses each person's theta SE; design doc 5.2).
correction	`"none"` (correct for EAP estimates with posterior SDs) or `"latent"` (empirical-Bayes shrinkage for WLE/ML estimates with sampling SEs; design doc 4.1). Only relevant when `method = "prob"`.
rho	Optional reliability for `"latent"`; estimated from the data when `NULL`.

Value

A statistic spec for `lst_table()`.

Examples

```
d <- data.frame(id = 1:100, region = rep(c("north", "south"), 50),
                w = runif(100, 0.5, 2), theta = rnorm(100),
                se = runif(100, 0.2, 0.4))
x <- lst_data(d, id = id, group = region, weight = w,
              theta = c(math = theta), theta_se = c(math = se))
brk <- c(low = -Inf, mid = -0.5, high = 0.8)
lst_table(x, rows = region, values = list(
  levels = st_level_prop(math, breaks = brk, method = "prob")
))
```

---

st_mean *Weighted mean statistic*

Description

Weighted mean statistic

Usage

```
st_mean(var)
```

Arguments

var	Measure: a theta/score dimension name or a numeric column.

Value

A statistic spec for `lst_table()`.

Examples

```
d <- data.frame(id = 1:100, region = rep(c("north", "south"), 50),
                w = runif(100, 0.5, 2), theta = rnorm(100),
                se = runif(100, 0.2, 0.4))
x <- lst_data(d, id = id, group = region, weight = w,
              theta = c(math = theta), theta_se = c(math = se))
lst_table(x, rows = region, values = list(mean = st_mean(math)))
```

`st_option_dist` *Weighted item option distribution (including missing rate)*

Description

Each option share is the weighted mean of a 0/1 indicator for that option, so it carries a sampling standard error from the same engine as the other statistics: linearized by default, or replicate-based when `rep_weights` are declared. Missing responses form their own category, and within an item the shares of one group sum to 1. As for `st_pvalue()`, there is no measurement component for raw item responses, so `se_measurement` is 0 and `se_total` equals `se_sampling`.

Usage

```
st_option_dist(items = NULL, missing_as = NULL)
```

Arguments

`items`	Character vector of item (resp) columns; `NULL` means all declared resp columns.
`missing_as`	Label used for missing responses; `NULL` uses a built-in default label (Chinese for "missing").

Details

Cost scales with the number of options: an item with k distinct responses costs about k times a single `st_pvalue()` pass.

Value

A statistic spec for `lst_table()`.

Examples

```
d <- data.frame(id = 1:50,
                q1 = sample(c("A", "B", "C", NA), 50, TRUE))
x <- lst_data(d, id = id, resp = q1)
lst_table(x, values = list(opts = st_option_dist(items = "q1")))
```

st_prop_above *Proportion above a cutoff*

Description

Proportion above a cutoff

Usage

```
st_prop_above(
  var,
  cutoff,
  method = c("hard", "prob"),
  correction = c("none", "latent"),
  rho = NULL
)
```

Arguments

var	Measure: a theta/score dimension name or a numeric column.
cutoff	Numeric threshold.
method	"hard" (0/1 classification) or "prob" (probabilistic, uses each person's theta SE; design doc 5.2).
correction	"none" (correct for EAP estimates with posterior SDs) or "latent" (empirical-Bayes shrinkage for WLE/ML estimates with sampling SEs; design doc 4.1). Only relevant when method = "prob".
rho	Optional reliability for "latent"; estimated from the data when NULL.

Value

A statistic spec for lst_table().

Examples

```
d <- data.frame(id = 1:100, region = rep(c("north", "south"), 50),
                w = runif(100, 0.5, 2), theta = rnorm(100),
                se = runif(100, 0.2, 0.4))
x <- lst_data(d, id = id, group = region, weight = w,
              theta = c(math = theta), theta_se = c(math = se))
lst_table(x, rows = region, values = list(
  hard = st_prop_above(math, cutoff = 1),
  prob = st_prop_above(math, cutoff = 1, method = "prob")
))
```

st_pvalue *Weighted item p-value (proportion correct / mean score rate)*

Description

Requires a scoring key declared in `lst_data()` (`key =`), a named vector item -> correct answer. Numeric responses already scored 0/1 can use `key = NULL` items by declaring them directly.

Usage

```
st_pvalue(items = NULL)
```

Arguments

`items`	Character vector of item (resp) columns; `NULL` means all declared resp columns.

Value

A statistic spec for `lst_table()`.

Examples

```
d <- data.frame(id = 1:50, grp = rep(c("a", "b"), 25),
                q1 = sample(c("A", "B"), 50, TRUE),
                q2 = rbinom(50, 1, 0.7))
x <- lst_data(d, id = id, group = grp, resp = c(q1, q2),
              key = list(q1 = "A"))
lst_table(x, rows = grp, values = list(pv = st_pvalue()))
```

st_quantile *Weighted quantile statistic*

Description

Point estimates only in v0.1 (SE reported as NA; planned for v0.2).

Usage

```
st_quantile(var, probs = 0.5)
```

Arguments

`var`	Measure: a theta/score dimension name or a numeric column.
`probs`	Quantile probabilities.

Value

A statistic spec for lst_table().

Examples

```
d <- data.frame(id = 1:100, region = rep(c("north", "south"), 50),
                w = runif(100, 0.5, 2), theta = rnorm(100),
                se = runif(100, 0.2, 0.4))
x <- lst_data(d, id = id, group = region, weight = w,
              theta = c(math = theta), theta_se = c(math = se))
lst_table(x, rows = region, values = list(
  q = st_quantile(math, probs = c(0.25, 0.5, 0.75))
))
```

---

st_sd *Weighted standard deviation*

Description

Weighted standard deviation

Usage

```
st_sd(var)
```

Arguments

var	Measure: a theta/score dimension name or a numeric column.

Value

A statistic spec for lst_table().

Examples

```
d <- data.frame(id = 1:100, region = rep(c("north", "south"), 50),
                w = runif(100, 0.5, 2), theta = rnorm(100),
                se = runif(100, 0.2, 0.4))
x <- lst_data(d, id = id, group = region, weight = w,
              theta = c(math = theta), theta_se = c(math = se))
lst_table(x, rows = region, values = list(sd = st_sd(math)))
```

Index