Package 'surveycore'

November 3, 2026

Title Core Survey Analysis Infrastructure

Version 1.2.0

Description A modern, 'S7'-based foundation for survey analysis spanning both probability and non-probability samples. Probability sample designs include Taylor series linearization, replicate weights (BRR, Fay, jackknife, bootstrap), and two-phase estimation, following 'Lumley' (2004) <doi:10.18637/jss.v009.i08>. Non-probability sample designs support bootstrap and jackknife variance estimation for opt-in panels and convenience samples. Provides a unified estimator interface for means, frequencies, totals, quantiles, ratios, correlations, regression, and t-tests, with weighted 'polychoric' and 'polyserial' correlation following 'Mannan' (2025) <doi:10.2139/ssrn.6580480>. A metadata system preserves 'haven'-style variable labels, value labels, and question-preface attributes through all operations, alongside dataset-level metadata that records the survey name, the fielding vendor, and the field period. Uses a 'tidyselect' interface throughout.

License GPL (>= 3)

Encoding UTF-8

Depends R (>= 4.3.0)

Imports S7 (>= 0.1.0), rlang (>= 1.0.0), tidyselect (>= 1.2.0), cli (>= 3.6.0), tibble (>= 3.1.0), dplyr (>= 1.1.0), marginaleffects (>= 0.24.0), pbivnorm (>= 0.6.0), stats, graphics, utils

Suggests testthat (>= 3.0.0), withr (>= 2.5.0), surveytidy (>= 0.5.0), survey (>= 4.0), survival, srvyr (>= 1.0), haven (>= 2.5.0), lifecycle (>= 1.0.0), broom (>= 1.0.0), polycor (>= 0.8.0), jtools (>= 2.2.0), labelled (>= 2.12.0), covr, knitr, rmarkdown

Additional_repositories `https://jdenn0514.r-universe.dev`

VignetteBuilder knitr

Config/testthat/edition 3

URL https://github.com/JDenn0514/surveycore, https://jdenn0514.github.io/surveycore/

BugReports https://github.com/JDenn0514/surveycore/issues

LazyData true

LazyDataCompression xz

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Jacob Dennen [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-3006-7364>), Thomas Lumley [ctb, cph] (Author of variance estimation code vendored from the 'survey' package)

Maintainer Jacob Dennen <jdenn0514@gmail.com>

Repository CRAN

Date/Publication 2026-11-03 09:00:02 UTC

Contents

.get_design_vars_flat 4
acs_pums_wy 5
add_survey 6
anes_2024 8
anova.survey_glm_fit 10
as_caldata 11
as_survey 13
as_survey_collection 16
as_survey_nonprob 18
as_survey_replicate 22
as_survey_twophase 27
as_svydesign 31
as_tbl_svy 33
ca_api_2000 34
classify_question_type 37
clean 38
coef.survey_result 40
confint.survey_result 41
extract_dataset_metadata 42
extract_data_name 44
extract_field_dates 45
extract_field_period 46
extract_higher_is 47
extract_metadata 48
extract_missing_codes 49
extract_question_preface 50
extract_reverse_coded 52

extract_sata . . . 53
extract_survey_name . . . 54
extract_universe . . . 55
extract_val_labels . . . 56
extract_var_extra . . . 57
extract_var_label . . . 58
extract_var_note . . . 60
extract_vendor . . . 61
from_svydesign . . . 62
from_tbl_svy . . . 64
get_anova . . . 65
get_corr . . . 67
get_covariance . . . 72
get_diffs . . . 75
get_effective_n . . . 78
get_freqs . . . 80
get_means . . . 83
get_pairwise . . . 86
get_quantiles . . . 88
get_ratios . . . 91
get_totals . . . 93
get_t_test . . . 96
get_variance . . . 98
gss_2024 . . . 101
infer_question_prefaces . . . 103
meta . . . 106
nhanes_2017 . . . 107
ns_wave1 . . . 109
pew_jewish_2020 . . . 116
pew_npors_2025 . . . 122
print.survey_anova . . . 125
print.survey_diffs . . . 126
print.survey_pairwise . . . 126
print.survey_result . . . 127
print.survey_t_test . . . 128
remove_survey . . . 128
SE . . . 129
SE.default . . . 130
SE.survey_result . . . 131
set_collection_id . . . 132
set_collection_if_missing_var . . . 133
set_dataset_metadata . . . 134
set_data_name . . . 137
set_field_dates . . . 138
set_field_period . . . 139
set_higher_is . . . 140
set_missing_codes . . . 141
set_question_preface . . . 143

set_reverse_coded . . . 144
set_sata . . . 145
set_survey_name . . . 147
set_universe . . . 148
set_val_labels . . . 149
set_var_extra . . . 150
set_var_label . . . 151
set_var_note . . . 153
set_vendor . . . 154
SURVEYCORE_DOMAIN_COL . . . 155
survey_collection . . . 156
survey_data . . . 157
survey_glm . . . 159
survey_glm_fit . . . 162
survey_metadata . . . 164
survey_nonprob . . . 166
survey_replicate . . . 169
survey_taylor . . . 171
survey_twophase . . . 173
survey_weighting_history . . . 174
update_design . . . 175
vcov.survey_result . . . 177

Index **179**

`.get_design_vars_flat` *Get design variable column names*

Description

Returns a flat character vector of all design-variable column names (ids, weights, strata, fpc) for any survey design class. `NULL` entries are dropped; names are unique. Exported for use by extension packages (e.g., `surveytidy`); not intended for end users.

Usage

```
.get_design_vars_flat(design)
```

Arguments

`design`	A survey design object (`survey_base` subclass).

Value

A character vector of column names.

acs_pums_wy *ACS PUMS 2022 1-Year: Wyoming Persons*

Description

All person records from the 2022 American Community Survey (ACS) 1-Year Public Use Microdata Sample (PUMS) for Wyoming (state FIPS 56). Wyoming is the least-populous U.S. state, making this the smallest state-level PUMS file — ideal for fast tests and examples.

Usage

```
acs_pums_wy
```

Format

A data frame with 5,962 rows and 96 variables. Columns `pwgtp1` through `pwgtp80` are the 80 successive difference replicate weights for variance estimation; the remaining 16 variables are:

- `puma`: Public Use Microdata Area code. Use as the cluster ID (PSU) for variance estimation.
- `st`: State FIPS code (all 56 = Wyoming).
- `pwgtp`: Person weight. Represents the number of people in the Wyoming population that this record represents.
- `agep`: Age (0–99 years).
- `sex`: Sex (1 = male, 2 = female).
- `rac1p`: Recoded detailed race (1 = White alone, 2 = Black or African American alone, 3 = American Indian alone, 6 = Asian alone, 9 = Two or more races).
- `hisp`: Recoded Hispanic origin (01 = Not Spanish/Hispanic/Latino; 02–24 = specific Hispanic origin).
- `schl`: Educational attainment (24 categories: 01 = no schooling, 16 = regular high school diploma, 21 = bachelor's degree, 24 = doctorate degree).
- `esr`: Employment status recode (1 = civilian employed at work, 2 = civilian employed with job but not at work, 3 = unemployed, 4 = Armed Forces at work, 5 = Armed Forces not at work, 6 = Not in labor force).
- `pincp`: Total person income in the past 12 months (dollars, signed; negative values indicate a net loss). Multiply by `adjinc / 1e6` to adjust to constant dollars.
- `wagp`: Wages or salary income in the past 12 months (dollars). `NA` if not applicable.
- `hicov`: Health insurance coverage (1 = with health insurance, 2 = without health insurance).
- `dis`: Disability recode (1 = with a disability, 2 = without a disability).
- `povpip`: Income-to-poverty ratio (0–501; 501 means 501% or more).
- `wkhp`: Usual hours worked per week in the past 12 months. `NA` if not in the labor force.
- `adjinc`: Adjustment factor for income and earnings. Divide by 1,000,000 and multiply income variables to convert to 2022 constant dollars.

Details

Survey design: Successive difference replication (SDR). Use `as_survey_replicate()` with all 80 replicate weights:

```
svy <- as_survey_replicate(
  acs_pums_wy,
  weights    = pwgtp,
  repweights = pwgtp1:pwgtp80,
  type       = "successive-difference"
)
```

Income adjustment: Income variables (`pincp`, `wagp`) are in survey-year dollars. Multiply by `adjinc / 1e6` to convert to 2022 inflation-adjusted dollars before comparing across ACS years.

Metadata: The ACS PUMS source is a plain CSV with no embedded labels. Columns in `acs_pums_wy` carry no `"label"`, `"labels"`, or `"question_preface"` attributes. Variable descriptions are documented here in `?acs_pums_wy` and in `data-raw/README.md`. Use `set_var_label()` and `set_val_labels()` to attach labels manually before analysis if needed.

Source

U.S. Census Bureau. 2022 ACS 1-Year PUMS. https://www.census.gov/programs-surveys/acs/microdata/access.html

Examples

```
# Wyoming population represented
sum(acs_pums_wy$pwgtp)

# Age distribution
hist(acs_pums_wy$agep, main = "Age distribution, Wyoming 2022", xlab = "Age")

# Confirm 80 replicate weights are present
sum(grepl("^pwgtp[0-9]", names(acs_pums_wy)))
```

---

add_survey *Add Surveys to a* `survey_collection`

Description

Appends one or more surveys to an existing collection and returns a new `survey_collection`. The original collection is unchanged. Surveys may be passed with explicit names or as bare symbols (auto-named, like `as_survey_collection()`). Duplicate names are repaired by appending _1, _2, ... Existing names are never modified during repair.

Usage

```
add_survey(.collection, ...)
```

Arguments

.collection	A survey_collection. Named with a leading dot so it cannot collide with user-supplied names in ... (e.g., a survey named "x").
...	One or more surveys to append. Accepts named arguments ("wave3" = d3) or bare symbols (d3, auto-named to "d3"). If a new name collides with an existing one (or with another new one), it is repaired by appending _1, _2, ... and a surveycore_warning_collection_duplicate_name_repaired warning is emitted with the mapping.

Details

Calling add_survey(x) with no additional surveys returns x unchanged; no error is raised.

Value

A new survey_collection with the appended surveys.

See Also

as_survey_collection(), remove_survey()

Other collections: as_survey_collection(), remove_survey(), set_collection_id(), set_collection_if_missing_survey_collection()

Examples

```
d1 <- as_survey(
  gss_2024,
  ids = vpsu,
  weights = wtssps,
  strata = vstrat,
  nest = TRUE
)
d2 <- as_survey(
  gss_2024,
  ids = vpsu,
  weights = wtssps,
  strata = vstrat,
  nest = TRUE
)
coll <- as_survey_collection(a = d1)
coll2 <- add_survey(coll, b = d2)
names(coll2)
```

`anes_2024` *ANES 2024: American National Election Studies Time Series*

Description

A 19-variable extract from the 2024 American National Election Studies (ANES) Time Series Study, a landmark biennial pre- and post-election survey of the American electorate. Fielded via face-to-face interview and web (n = 5,521). This extract uses the FTF + Web combined design variables (`v240103a`–`v240103d`), the recommended set for most analyses.

Usage

```
anes_2024
```

Format

A data frame with 5,521 rows and 19 variables:

v240103a Pre-election weight (FTF+Web combined). Use for variables asked before November 5, 2024.

v240103b Post-election weight (FTF+Web combined). Use for variables asked after November 5, 2024.

v240103c PSU (FTF+Web combined). Use as the cluster ID for variance estimation.

v240103d Stratum (FTF+Web combined). Use as the stratification variable.

v240001 2024 Time Series Case ID. Unique respondent identifier.

v240003 Sample type: `1` = Panel, `2` = Fresh Web, `3` = Fresh FTF, `4` = GSS.

v240002c Pre/Post interview completion: `1` = Pre-election only, `2` = Pre- and post-election.

v243002 State FIPS code.

v243007 Census region: `1` = Northeast, `2` = Midwest, `3` = South, `4` = West.

v241458x Age on Election Day (summary). Top-coded at 80. `-2` = missing.

v241550 Sex: `1` = male, `2` = female.

v241501x Race/ethnicity (5-category summary): White non-Hispanic, Black non-Hispanic, Hispanic, Asian/NHPI non-Hispanic, Other/Multiracial non-Hispanic.

v241465x Education (5-category summary): `1` = less than HS, `2` = HS diploma, `3` = some college, `4` = bachelor's degree, `5` = graduate degree.

v241566x Household income (28 categories from < $5,000 to $250,000+).

v241177 Liberal-conservative self-placement (7-point scale): `1` = extremely liberal, `7` = extremely conservative. `99` = haven't thought about this.

v241222 Party identification strength: `1` = strong, `2` = not very strong.

v241223 Party identification lean (Independents): `1` = closer to Republican, `2` = neither, `3` = closer to Democrat.

v242066 Did respondent vote for President (POST): `1` = yes, `2` = no.

v242067 Presidential vote choice (POST): `1` = Harris, `2` = Trump, `3` = RFK Jr., `4` = West, `5` = Stein, `6` = Other.

Details

Survey design: Stratified cluster — use Taylor series linearization. Two weights are available depending on whether the analysis uses pre- or post-election variables:

```
# Pre-election analysis (party ID, ideology, candidate preference)
svy_pre <- as_survey(anes_2024,
  ids     = v240103c,
  strata  = v240103d,
  weights = v240103a,
  nest    = TRUE
)

# Post-election analysis (validated vote choice)
svy_post <- as_survey(anes_2024,
  ids     = v240103c,
  strata  = v240103d,
  weights = v240103b,
  nest    = TRUE
)
```

Missing value codes: The ANES uses negative integer codes for missing data throughout: `-9` = Refused, `-8` = Don't know, `-4` = Technical error, `-1` = Inapplicable, and others. These must be recoded to `NA` before analysis. Check `attr(anes_2024$v241177, "labels")` for the full set of codes for a given variable.

Metadata: All columns carry variable labels and value labels as R attributes from the original Stata file, automatically extracted into surveycore's metadata system when you call `as_survey()`.

- **Variable labels** (`"label"` attribute): A human-readable description of each column. Example: `attr(anes_2024$v241550, "label")` returns `"PRE: What is your sex?"` (or similar ANES phrasing).
- **Value labels** (`"labels"` attribute): A named numeric vector mapping each code to its meaning, including all missing-value codes. Example: `attr(anes_2024$v241550, "labels")` returns a vector with entries for `Male`, `Female`, and the applicable negative missing codes.

Source

American National Election Studies. 2024 Time Series Study. Available at electionstudies.org (free account required to download raw data; the processed `.rda` is included in the package). Prepared by `data-raw/prepare-anes-2024.R`.

Examples

```
# Variables in the dataset
names(anes_2024)

# Create pre-election design
svy <- as_survey(
  anes_2024,
  ids = v240103c,
```

```
  strata = v240103d,
  weights = v240103a,
  nest = TRUE
)

# Inspect variable label (ANES uses opaque V-codes; labels give context)
attr(anes_2024$v241177, "label")

# Inspect value labels, including missing-value codes
attr(anes_2024$v241177, "labels")
```

`anova.survey_glm_fit` *ANOVA Method for Survey GLM Fits*

Description

S3 method that dispatches to `get_anova()`. Pass one or two survey_glm_fit objects; the single-model or pairwise path is chosen automatically.

Usage

```
## S3 method for class 'survey_glm_fit'
anova(object, ..., method = "LRT", test = "F", null = NULL)
```

Arguments

`object`	A survey_glm_fit object.
`...`	An optional second survey_glm_fit for pairwise comparison; anything else errors.
`method`	Character(1). `"LRT"` (default) or `"Wald"`.
`test`	Character(1). `"F"` (default) or `"Chisq"`.
`null`	Numeric or `NULL`. Hypothesized coefficient value (Wald only).

Value

A `survey_anova` tibble; see `get_anova()` for column details.

as_caldata *Build a Calibration Data Element*

Description

Constructs a calibration data object from base sampling weights, g-weights (calibration factors), and a model matrix of calibration covariates. The returned list is suitable for assignment to the `@calibration` slot of a survey_taylor or survey_replicate object.

Usage

```
as_caldata(base_weights, g_weights, model_matrix)
```

Arguments

`base_weights`	A numeric vector of positive, finite base sampling weights (length n). These are the original sampling weights before calibration is applied. Must not contain `NA`, `NaN`, or `Inf`.
`g_weights`	A numeric vector of positive, finite g-factors (length n). The calibrated weights are `base_weights * g_weights`. g-factors equal to 1.0 represent no adjustment. Must not contain `NA`, `NaN`, or `Inf`. Must have the same length as `base_weights`. The internal quantity `g_weights * sqrt(base_weights)` must not have any near-zero values (below `sqrt(.Machine$double.eps)`); if it does, a `surveycore_error_caldata_weig` warning is issued.
`model_matrix`	A numeric matrix with n rows and at least 1 column, representing the calibration covariates used during post-stratification or raking. Must not contain `NA`, `NaN`, or `Inf`.

Details

The resulting calibration object is used by the variance estimation routines to apply a Deville-Sarndal (1992) calibration correction to Taylor-series and replicate-weight variance estimates.

GLM limitation: Using a calibrated `survey_taylor` object with `survey_glm()` produces correct but conservative standard errors until GREG-GLM variance is implemented in a future release. The calibration correction is not applied in the GLM variance path.

Value

A named list with four elements:

`qr` A QR decomposition (class `"qr"`) of `sqrt(base_weights) * model_matrix`. Used for calibration projection in variance estimation.

`w` A numeric vector of length n equal to `g_weights * sqrt(base_weights)`. This intermediate quantity (the square root of the calibrated weights scaled by g) is used directly in the GREG variance projection formula.

`stage` Integer scalar `0L`. Currently only between-PSU calibration (stage 0) is supported.

`index` `NULL`. Reserved for future within-PSU calibration support.

Inter-package contract

GREG calibration (single auxiliary variable or multiple uncorrelated variables): pass the model matrix from `model.matrix(formula, data)` directly – one column per calibration variable. The intercept column (`(Intercept)`) is included by default from `model.matrix()`; it contributes one degree of freedom to the calibration adjustment.

Raking (multiple calibration margins, Architecture A): combine all margin indicator matrices into a single matrix before calling `as_caldata()`. Column-bind the matrices and drop one reference column per margin to avoid rank deficiency (e.g., drop the last column of each per-margin block). Pass this single combined matrix. Do **not** call `as_caldata()` once per margin; that uses Architecture B (sequential), which requires a separate `as_caldata()` element per calibration pass and stores them all in the `@calibration` list.

q_k = 1 assumption: `as_caldata()` always assumes $q_k = 1$ (uniform calibration weights). If your calibration uses a non-unity q_k (variance-function weights from `survey::calibrate( calfun = "linear", variance = ...)`) you must absorb those weights into `model_matrix` before calling `as_caldata()`.

For `survey_replicate` designs

`@calibration` on a `survey_replicate` object is **provenance-only**: it documents that the replicate weights were derived from a calibrated design, but the variance estimator does not apply any GREG projection. Calibration is already encoded in the replicate weights themselves. Do not expect `get_means()` SE to differ between a `survey_replicate` with and without `@calibration` set.

References

Deville, J.-C., and Sarndal, C.-E. (1992). Calibration estimators in survey sampling. *Journal of the American Statistical Association*, 87, 376–382.

See Also

`as_survey()` for Taylor series designs, `as_survey_replicate()` for replicate-weight designs, `as_survey_twophase()` for two-phase designs

Other constructors: `as_survey()`, `as_survey_nonprob()`, `as_survey_replicate()`, `as_survey_twophase()`, `survey_glm()`, `survey_glm_fit()`, `survey_nonprob()`, `survey_replicate()`, `survey_taylor()`, `survey_twophase()`

Examples

```
# Minimal example: 3-unit design, intercept-only calibration
base_weights <- c(2.5, 3.0, 4.0)
g_weights <- c(1.02, 0.98, 1.01)
model_matrix <- matrix(1, nrow = 3, ncol = 1)

cd <- as_caldata(base_weights, g_weights, model_matrix)
names(cd) # "qr", "w", "stage", "index"

# Assign to a survey_taylor design
df <- data.frame(y = c(1.2, 2.3, 3.4), wt = base_weights)
design <- as_survey(df, weights = wt)
```

```
design@calibration <- list(cd)
is.null(design@calibration) # FALSE
```

---

as_survey *Create a Taylor Series Linearization Survey Design*

Description

Creates a survey design object using Taylor series (linearization) for variance estimation. Supports simple random samples, stratified designs, single- and multi-stage cluster designs, and designs with finite population correction. Uses a tidy-select interface for all design variable arguments.

Usage

```
as_survey(
  data,
  ids = NULL,
  probs = NULL,
  weights = NULL,
  strata = NULL,
  fpc = NULL,
  nest = FALSE,
  calibration = NULL
)
```

Arguments

data	A `data.frame` containing the survey responses. Must have at least one row and unique column names.
ids	<tidy-select> Cluster (PSU) ID column(s). For single-stage: `ids = psu`. For multi-stage: `ids = c(psu, ssu)`. Omit entirely for simple random sampling.
probs	<tidy-select> Sampling probability column (a single column, values in (0, 1]). Converted to weights = `1/probs` and stored internally. Cannot be used together with `weights` unless the values are consistent (`weights == 1/probs`).
weights	<tidy-select> Sampling weight column (a single column, values strictly > 0).
strata	<tidy-select> Stratification variable column (a single column).
fpc	<tidy-select> Finite population correction column(s). For single-stage designs, supply one column. For multi-stage designs, supply one column per stage: `fpc = c(fpc_stage1, fpc_stage2)`. Each column accepts either total population size (integer, all > 1) or sampling fraction (numeric, all in (0, 1]). Cannot contain `NA`. Cannot have more columns than `ids` stages; fewer is allowed (later stages assume infinite population).
nest	Logical. If `TRUE`, PSU IDs are treated as nested within strata — i.e., the same ID value in two different strata refers to two distinct PSUs. Set `nest = TRUE` when PSU IDs are not globally unique (e.g., NHANES, where PSU IDs restart from 1 in each stratum). Requires `strata` to be specified. Default `FALSE`.

`calibration` A list of calibration data elements, each produced by `as_caldata()`, or `NULL` (default) for no calibration adjustment. When non-`NULL`, variance estimation applies a Deville-Sarndal GREG projection that reduces standard errors proportional to the correlation between the auxiliary variables and the outcome. Equivalent to assigning `design@calibration <- list(cd)` after construction.

Known limitations (not validated at construction time):

- *Weight consistency*: surveycore cannot verify that `cd$w` encodes the same base weights as the design weight column. Mismatched base weights produce incorrect variance estimates.
- *Stale calibration after* `update_design()`: changing the weight column on a calibrated design with `update_design()` makes `@calibration` stale. Clear `@calibration` manually after any weight column change.

Value

A `survey_taylor` object. Columns imported with haven-style value labels are stored as their underlying type: the `label` and `labels` attributes are kept, and the value labels are also recorded in the metadata system, but the `haven_labelled` class is not stored. Use `survey_data(x, haven_class = TRUE)` to read the data back with that class rebuilt.

Tidy-select

All design variable arguments (`ids`, `probs`, `weights`, `strata`, `fpc`) support tidy-select syntax: bare column names, `c()` to combine multiple columns (multi-stage `ids = c(psu, ssu)`, multi-stage `fpc`), and tidyselect helpers like `starts_with()`. See the Examples section below for runnable demonstrations.

Simple random sample

When no `ids` or `strata` are specified, the result is a `survey_taylor` object with `NULL` ids and strata — i.e., a simple random sample (SRS). The Taylor variance machinery produces the same estimates as the classical SRS formula `(1 - f) * s^2 / n`. If `weights` and `probs` are also both omitted, uniform weights are assigned and a warning is issued.

Known limitations

`as_survey()` does not support probability-proportional-to-size (PPS) variance estimation. Taylor series linearization treats all designs as with-replacement, which overestimates (is conservative for) variance in PPS-without-replacement designs. The Yates-Grundy and Brewer/Overton estimators available in `survey::svydesign()` via its `pps` and `variance` arguments are not supported.

If your design requires PPS-specific variance estimation, create the design with `survey::svydesign()` and convert it with `from_svydesign()`:

```
d_survey <- survey::svydesign(
  ids = ~psu, weights = ~wt, strata = ~stratum,
  pps = "brewer", data = mydata
)
d <- from_svydesign(d_survey)
```

References

Deville, J.-C. and Sarndal, C.-E. (1992) Calibration estimators in survey sampling. *Journal of the American Statistical Association* **87**(418), 376–382.

Deville, J.-C., Sarndal, C.-E. and Sautory, O. (1993) Generalized raking procedures in survey sampling. *Journal of the American Statistical Association* **88**(423), 1013–1020.

Lumley, T. (2004) Analysis of complex survey samples. *Journal of Statistical Software* **9**(1), 1–19.

Lumley, T. (2010) *Complex Surveys: A Guide to Analysis Using R*. John Wiley and Sons.

Rao, J.N.K., Yung, W. and Hidiroglou, M.A. (2002) Estimating equations for the analysis of survey data using poststratification information. *Sankhya* **64-A**, 22–36.

Sarndal, C-E., Swensson, B. and Wretman, J. (1992) *Model Assisted Survey Sampling*. Springer.

See Also

`as_survey_replicate()` for replicate-weight designs, `as_survey_twophase()` for two-phase designs, `set_var_label()` to add variable labels

Other constructors: `as_caldata()`, `as_survey_nonprob()`, `as_survey_replicate()`, `as_survey_twophase()`, `survey_glm()`, `survey_glm_fit()`, `survey_nonprob()`, `survey_replicate()`, `survey_taylor()`, `survey_twophase()`

Examples

```
# Full NHANES design: stratified cluster with PSU IDs nested within strata
d <- as_survey(
  nhanes_2017,
  ids = sdmvpsu,
  weights = wtint2yr,
  strata = sdmvstra,
  nest = TRUE
)

# Stratified design without PSU cluster IDs
d_strat <- as_survey(nhanes_2017, weights = wtint2yr, strata = sdmvstra)

# Blood pressure analysis: filter to exam participants, use MEC weight
exam <- nhanes_2017[nhanes_2017$ridstatr == 2, ]
d_bp <- as_survey(
  exam,
  ids = sdmvpsu,
  weights = wtmec2yr,
  strata = sdmvstra,
  nest = TRUE
)

# c() to combine multiple columns — sketched on a synthetic two-stage frame
df <- data.frame(
  psu = rep(1:5, each = 4),
  ssu = 1:20,
  wt = runif(20, 0.5, 2)
)
```

```
d_ms <- as_survey(df, ids = c(psu, ssu), weights = wt)

# Tidy-select helpers like starts_with() also work
d_h <- as_survey(
  gss_2024,
  ids = vpsu,
  strata = vstrat,
  weights = starts_with("wtssn"),
  nest = TRUE
)
```

`as_survey_collection` *Create a Collection of Survey Designs*

Description

Builds a survey_collection from one or more survey design objects for comparative analysis across waves, cross-sections, or sub-populations. Each element is stored independently — designs are never combined, and variance estimation is never re-specified.

Usage

```
as_survey_collection(..., group, .id = ".survey", .if_missing_var = "error")
```

Arguments

`...`	One or more `survey_base` objects, passed with explicit names or as bare symbols. At least one argument is required.
`group`	<tidy-select> Grouping variable(s) to apply uniformly across every member survey. Accepts bare names (`region`, `c(region, stratum)`), `all_of()`, etc. When supplied and resolving to a non-empty character vector, the named columns must exist in every member's `@data`; they are propagated onto each member's `@groups` and set as `coll@groups`. If a member already carries a non-empty `@groups` that differs from the resolved target, the target takes precedence and a `surveycore_warning_collection_group_overridden` warning is emitted (one per divergent member). When missing or resolving to an empty vector (`NULL`, `character(0)`, `c()`, `all_of(character(0))`), the collection adopts the members' uniform `@groups` if they are all identical, or errors `surveycore_error_collection_group` if they differ. Default: missing (adopt-from-members).
`.id`	Character(1). Identifier column name used when dispatching analysis functions across the collection. Default `".survey"`. Stored on the returned collection's `@id` property and used as the default by `.dispatch_over_collection()` when a per-call `.id` is not supplied (i.e., when an analysis function is called with `.id = NULL`). Mutate via `set_collection_id()`.
`.if_missing_var`	Character(1), one of `c("error", "skip")`. Default `"error"`. Stored on the returned collection's `@if_missing_var` property and used as the default by

.dispatch_over_collection() when a per-call .if_missing_var is not supplied (i.e., when an analysis function is called with .if_missing_var = NULL). When "skip", member surveys missing a requested variable are dropped from the dispatched result; when "error", the dispatcher aborts. Mutate via set_collection_if_missing_va

Details

Arguments may be passed with explicit names ("wave1" = d1) or as bare symbols (d1, auto-named to "d1"). An unnamed argument that is not a bare symbol (e.g., an inline as_survey(...) call) raises surveycore_error_collection_unnamed_expr — name such arguments explicitly.

Duplicate names are repaired by appending _1, _2, ... to subsequent occurrences (first occurrence preserved). When any rename occurs, a surveycore_warning_collection_duplicate_name_repaired warning is emitted showing the original -> repaired mapping.

Value

A survey_collection object containing the supplied surveys.

See Also

survey_collection, add_survey(), remove_survey()

Other collections: add_survey(), remove_survey(), set_collection_id(), set_collection_if_missing_var(), survey_collection()

Examples

```
d1 <- as_survey(
  gss_2024,
  ids = vpsu,
  weights = wtssps,
  strata = vstrat,
  nest = TRUE
)
d2 <- as_survey(
  gss_2024,
  ids = vpsu,
  weights = wtssps,
  strata = vstrat,
  nest = TRUE
)

# Explicit names
coll <- as_survey_collection("2020" = d1, "2024" = d2)
names(coll)

# Bare-symbol auto-naming
coll2 <- as_survey_collection(d1, d2)
names(coll2)

# Uniform grouping across members
coll3 <- as_survey_collection(d1, d2, group = vstrat)
```

```
names(survey_data(coll3[[1L]]))
```

---

as_survey_nonprob *Create a Non-probability Survey Design*

Description

Creates a survey design object for non-probability samples (e.g., online panels, quota samples, volunteer panels). Accepts pre-computed calibration weights (including raking and post-stratification) or inverse probability weighting (IPW) pseudo-weights.

Usage

```
as_survey_nonprob(
  data,
  weights,
  repweights = NULL,
  type = "bootstrap",
  scale = NULL,
  rscales = NULL,
  mse = TRUE,
  reference_sample = NULL,
  calibration = NULL
)
```

Arguments

data	A `data.frame` containing the survey responses with pre-computed calibration weights. Must have at least two rows and unique column names.
weights	<tidy-select> Calibration weight column (a single column, values strictly > 0). Typically produced by an external raking function (e.g., `anesrake::anesrake()`) or a **surveywts** calibration function. Zero weights are refused here but may appear later, after a **surveywts** adjustment such as nonresponse sets some weights to 0.
repweights	<tidy-select> Replicate weight columns (bootstrap or jackknife; at least 2). Each column must be numeric and represents one set of calibrated weights re-estimated on one replicate draw (calibration already applied within each replicate). Supply `NULL` (the default) to use SRS-based variance approximation. See `type` for supported replicate schemes.
type	Character scalar. Replicate variance type. When `repweights = NULL`, this argument is ignored. Case-sensitive. Valid values: `"bootstrap"` Bootstrap variance. Default scale: `1/R`. Default value for `type`. `"JK1"` Delete-one jackknife for unclustered nonprob designs. Default scale: `(R-1)/R`. Appropriate when each unit is its own replication unit. For clustered designs, use `"JK2"` or `"JKn"` with explicit `rscales`.

"jackknife" Alias for "JK1". Normalized to "JK1" before storage — the stored value is always "JK1", never "jackknife".

"JK2" Stratified jackknife. Default scale: 1. Requires explicit rscales (stratum-specific scale factors of the form (n_h - 1) / n_h). as_survey_replicate() accepts "JK2" with no rscales, as survey::svrepdesign() does; as_survey_nonprob() keeps the refusal, because survey has no non-probability design class and so is not the reference for this constructor.

"JKn" Equivalent to "JK2" for stratified nonprob designs. Default scale: 1. Requires explicit rscales.

scale Numeric scalar. Scaling factor for the replicate variance formula. Default NULL, which sets scale = 1 / R (where R is the number of replicate columns). Note: this default differs from as_survey_replicate(), which uses type-specific defaults.

rscales Numeric vector of length R. Per-replicate scale factors. All values must be non-negative and non-NA. Default NULL, which sets rscales = rep(1, R), except for "JK2" and "JKn", which require rscales: rscales = NULL is refused with surveycore_error_stratified_jk_rscales_unset.

mse Logical. If TRUE (the default), the mean-squared-error form of the variance estimator is used: (1/R) * sum((theta_r - theta)^2). If FALSE, the centered form is used instead. Default TRUE. Note: this default differs from as_survey_replicate() — mse = TRUE is the appropriate default for bootstrap replicates from calibrated non-probability samples (Wu 2022).

reference_sample

Optional. A survey_taylor object representing the probability-based reference sample used to estimate propensity scores or calibration targets. Stored in @reference_sample for reproducibility. Supply NULL (the default) when no reference sample is available.

calibration Optional. A calibration provenance object returned by a **surveywts** weighting function. Stored in @calibration for reproducibility only — it is not used in variance estimation (unlike as_survey() where @calibration drives GREG variance correction). When repweights is also supplied, two consistency checks are applied: for type = "bootstrap", calibration$bootstrap must be TRUE; for all types, calibration$R must equal the number of replicate columns when calibration$R is non-NULL. Supply NULL (the default) when no provenance metadata is available.

Details

Unlike probability samples, non-probability samples have no design weights derived from known selection probabilities, which means estimates carry additional uncertainty not captured by standard design-based variance formulas. Per Elliott and Valliant (2017), Valliant, Dever, and Kreuter (2018), and Brick (2015), bootstrap or jackknife replicate weights are the recommended approach for variance estimation — they propagate calibration uncertainty into standard errors. Note, however, that replicate variance addresses calibration uncertainty only; it does not resolve uncertainty about the selection mechanism itself, which requires untestable modeling assumptions about the relationship between sample membership and the survey variables of interest. Without replicate weights, standard errors use a model-assisted SRS approximation that systematically underestimates variance for non-probability samples.

When repweights is supplied, the variance estimator uses the replicate formula: V = scale * sum(rscales * (theta_r - theta)^2). For bootstrap replicates (type = "bootstrap"), the default scale = 1/R follows Wu (2022) and Chen et al. (2021). survey::svrepdesign() uses 1/(R - 1) for the bootstrap, and as_survey_replicate() matches that value. This constructor keeps 1/R because survey has no non-probability design class and so is not an oracle for one. On the same frame the bootstrap standard error from as_survey_nonprob() is therefore smaller than the one from as_survey_replicate() by a factor of sqrt((R - 1)/R), a fall of 2.5% at R = 20. For jackknife replicates (type = "JK1", "JK2", or "JKn"), scale and rscales follow the standard jackknife variance conventions; see type for defaults.

When repweights = NULL, standard errors use an SRS approximation (treating each observation as its own PSU). This understates calibration uncertainty; see vignette("creating-survey-objects") for details.

Value

A survey_nonprob object. Columns imported with haven-style value labels are stored as their underlying type: the label and labels attributes are kept, and the value labels are also recorded in the metadata system, but the haven_labelled class is not stored. Use survey_data(x, haven_class = TRUE) to read the data back with that class rebuilt.

When to use

Use as_survey_nonprob() instead of as_survey() when:

- Your data comes from a non-probability sample (online panel, quota sample, MTurk/Prolific, etc.)
- You have calibration or raking weights but no probability sampling design structure (no PSU IDs, strata, etc.)
- You want to explicitly record the provenance of your calibration weights for reproducibility

If your data comes from a probability sample with known design structure, use as_survey(), as_survey_replicate(), or as_survey_twophase() instead.

Variance estimation

Two modes are available, depending on whether repweights is supplied:

SRS approximation (repweights = NULL, the default) Standard errors treat the calibrated weights as fixed and assume simple random sampling. This is a model-assisted approximation that understates calibration uncertainty. Use this mode only when replicate weights are unavailable; interpret standard errors with caution (Valliant 2020; Elliott and Valliant 2017).

Bootstrap variance (repweights supplied) Each replicate weight column must contain calibrated weights re-estimated on one bootstrap draw (i.e., raking or post-stratification was re-applied within each replicate). This propagates calibration uncertainty into the variance estimate and is the recommended approach (Chrostowski et al. 2025; Kolenikov 2014).

See vignette("creating-survey-objects") for guidance on choosing between these modes and on the limitations of SRS-based variance for calibrated non-probability samples.

References

Valliant, R. (2020). Comparing alternatives for estimation from nonprobability samples. *Journal of Survey Statistics and Methodology* **8**(2), 231–263. doi:10.1093/jssam/smz003

Elliott, M.R. and Valliant, R. (2017). Inference for nonprobability samples. *Statistical Science* **32**(2), 249–264.

Chrostowski, M.J., Guzman, C.A. and Malm, L. (2025). Variance estimation for non-probability surveys. *Journal of Survey Statistics and Methodology* (forthcoming).

Brick, J.M. (2015). Compositional model inference. In *Proceedings of the Section on Survey Research Methods*, pp. 299–307. American Statistical Association, Alexandria, VA.

Valliant, R., Dever, J.A. and Kreuter, F. (2018). *Practical Tools for Designing and Weighting Survey Samples*, 2nd ed. Springer, New York.

Kolenikov, S. (2014). Calibrating variance estimation with proxy variables. *Survey Methodology* **40**(1), 21–38.

Wu, C. (2022). Statistical inference with non-probability survey samples. *Survey Methodology* **48**(2), 283–311.

Chen, Y., Li, P. and Wu, C. (2021). Doubly robust inference with non-probability survey samples. *Journal of the American Statistical Association* **115**(532), 2011–2021.

See Also

`as_survey()` for probability designs with Taylor variance, `as_survey_replicate()` for replicate-weight designs

Other constructors: `as_caldata()`, `as_survey()`, `as_survey_replicate()`, `as_survey_twophase()`, `survey_glm()`, `survey_glm_fit()`, `survey_nonprob()`, `survey_replicate()`, `survey_taylor()`, `survey_twophase()`

Examples

```
# Minimal: pre-computed calibration weights, SRS-based variance
df <- data.frame(
  y = rnorm(200),
  age = sample(c("18-34", "35-54", "55+"), 200, replace = TRUE),
  cal_wt = runif(200, 0.5, 2.5)
)
d <- as_survey_nonprob(df, weights = cal_wt)

# Bootstrap variance: replicate weights with calibration re-applied in each
set.seed(1)
R <- 50
rep_cols <- setNames(
  as.data.frame(
    matrix(runif(200 * R, 0.5, 2.5), nrow = 200)
  ),
  paste0("rep_", seq_len(R))
)
df_rep <- cbind(df, rep_cols)
d_boot <- as_survey_nonprob(
```

```
  df_rep,
  weights = cal_wt,
  repweights = starts_with("rep_"),
  type = "bootstrap"
)

# Jackknife variance (JK1): delete-one replicate weights
d_jk <- as_survey_nonprob(
  df_rep,
  weights = cal_wt,
  repweights = starts_with("rep_"),
  type = "JK1"
)
```

`as_survey_replicate` *Create a Replicate Weights Survey Design*

Description

Creates a survey design object using replicate weights for variance estimation. Supports all common replicate methods: jackknife (JK1, JK2, JKn), balanced repeated replication (BRR, Fay), bootstrap, ACS, successive-difference, and user-defined types. Uses a tidy-select interface for weight and replicate-weight columns.

Usage

```
as_survey_replicate(
  data,
  weights,
  repweights,
  type = c("JK1", "JK2", "JKn", "BRR", "Fay", "bootstrap", "ACS",
    "successive-difference", "other"),
  rho = NULL,
  scale = NULL,
  rscales = NULL,
  fpc = NULL,
  fpctype = NULL,
  mse = TRUE,
  calibration = NULL
)
```

Arguments

`data`	A `data.frame` containing the survey responses. Must have at least one row and unique column names.
`weights`	<tidy-select> Sampling weight column (a single column, values strictly > 0). Required.

repweights <tidy-select> Replicate weight columns. Must select at least one column. Supports tidy-select helpers (e.g., `starts_with("repwt")`). Required.

type Character. Replicate weight method. One of `"JK1"` (delete-1 jackknife), `"JK2"` (paired jackknife, two PSUs per stratum), `"JKn"` (delete-1 jackknife with varying replication counts), `"BRR"` (balanced repeated replication), `"Fay"` (Fay's method, a modified BRR; needs `rho`), `"bootstrap"`, `"ACS"` (used in American Community Survey), `"successive-difference"`, or `"other"` (user-specified scale). Case-sensitive.

rho Numeric scalar or `NULL` (default). The Fay shrinkage factor: the factor the replicate weights were built with. Each Fay replicate multiplies the weights of one half sample by `2 - rho` and the weights of the other half sample by `rho`. Required when `type = "Fay"`. The valid range is `[0, 1)`, and `rho = 0` gives BRR. An integer is accepted and stored as a double; a value with a `dim` attribute is refused. Find the value in the survey's technical documentation, where it is often called the Fay coefficient or epsilon. surveycore cannot check that the value matches the replicate columns.

The Fay scale `1 / (R * (1 - rho)^2)` assumes a two-PSU-per-stratum layout with Hadamard-balanced half samples, which surveycore does not check; the scale is exact for totals and first-order for means and ratios. `type` stays `"Fay"` at `rho = 0`, unlike `survey::as.svrepdesign()`, which relabels it `"BRR"`.

For every other `type`, a supplied `rho` is ignored with a warning and the design stores no `rho`. The warning has the class `surveycore_warning_rho_ignored` and also the class `surveycore_warning_replicate_arg_ignored`. For `"JKn"` with no `rscales`, the refusal fires and no `rho` warning is raised, where `survey::svrepdesign()` raises both.

scale Numeric. Scaling factor applied to the replicate variance formula. If `NULL` (default), computed automatically from `type` and the number of replicates `R`: `(R-1)/R` for `"JK1"`; `1/R` for `"BRR"`; `1 / (R * (1 - rho)^2)` for `"Fay"`; `1/(R-1)` for `"bootstrap"`; `4/R` for `"ACS"` and `"successive-difference"` (per Ash 2014 / Fay & Train 1995); `1` for `"JKn"`, `"JK2"` and `"other"`.

For `"JK1"`, `"JKn"`, `"bootstrap"` and `"other"`, a supplied `scale` is stored as given.

For `"BRR"`, `"JK2"`, `"ACS"` and `"successive-difference"`, a supplied `scale` is ignored with a warning of class `surveycore_warning_replicate_arg_ignored`, and the design stores the type's own value from the list above. `survey::svrepdesign()` also ignores a supplied `scale` for these four types. Any value other than `NULL` counts as supplied, even one equal to the type's own value, and surveycore does not check it.

surveycore warns only when the caller supplies `scale` or `rscales`. `survey::svrepdesign()` warns on every `"JK2"` call, even when the caller supplies neither. surveycore does not copy that warning: a warning that fires when the caller passed nothing tells the caller nothing, and warnings that no one asserts hide new ones.

For `"Fay"`, a supplied `scale` is discarded with no warning, as `survey::svrepdesign()` does, and the Fay scale always comes from `rho`.

`"JK2"` is the paired jackknife. Each replicate is a half sample, so the per-stratum factor is already inside the replicate weights: `rscales` stays at `rep(1, R)` and

the overall scale stays at 1. `survey::svrepdesign()` forces both. `as_survey_nonprob()` uses the same scale of 1 but requires `rscales` for `"JK2"`.

`"JKn"` is the stratified delete-one jackknife. Its per-stratum factor belongs in `rscales` and not in the overall scale. The factor is `(n_h - 1) / n_h` when PSUs are selected with replacement, or when the sampling fraction is negligible. `"JKn"` requires `rscales`, and `rscales = NULL` is refused, as `survey::svrepdesign()` refuses it for combined weights.

A caller who samples PSUs without replacement at a sampling fraction that is not negligible folds the correction into each `rscales` entry by hand, which makes the entry `(n_h - 1) * (1 - n_h / N_h) / n_h`. Here `n_h` is the number of sample PSUs in stratum `h`, and `N_h` is the number of population PSUs in stratum `h`. The caller supplies `N_h` from their own sampling frame. The `fpc` argument applies the same correction: `fpc = n_h / N_h` per replicate, with `fpctype = "fraction"`.

`"bootstrap"` divides by `R - 1`, which is the value `survey::svrepdesign()` computes at `bootstrap.average = 1`. surveycore has no `bootstrap.average` argument, so a caller cannot build a design with one and this default is always `1/(R-1)`; a design imported from `survey` keeps its effective scale exactly, in both directions. A bootstrap design of one replicate column gives `scale = Inf`, and `survey::svrepdesign()` stores `Inf` for the same input.

`as_survey_nonprob()` keeps `1/R` for `"bootstrap"`, so the same `type` string means a different divisor in the two constructors. The difference is a decision and not a defect; see `as_survey_nonprob()` for the reason and the size of the gap.

Pass `scale` explicitly to reproduce numbers published before these two defaults moved: `scale = (R - 1) / R` for `"JKn"` and `scale = 1 / R` for `"bootstrap"`. An old `"JKn"` call that passed no `rscales` also needs `rscales = rep(1, R)`, because `"JKn"` refuses `rscales = NULL`.

`rscales` Numeric vector of replicate-specific scaling factors, or `NULL`. If provided, must have the same length as the number of replicate weight columns selected by `repweights`, with no `NA` and no negative value. These length and value checks run on a supplied `rscales` for every `type`, including the three that ignore it.

Required for `"JKn"`: `rscales = NULL` is refused with `surveycore_error_stratified_jk_rscales_uns`

For `"JK2"`, `"ACS"` and `"successive-difference"`, a supplied `rscales` is ignored with the same warning as a supplied `scale`, `surveycore_warning_replicate_arg_ignored`. The design stores `NULL`, which the variance treats as `rep(1, R)`. `survey::svrepdesign()` forces `rep(1, R)` for these three types.

For `"JKn"` and the other five types, `"JK1"`, `"BRR"`, `"Fay"`, `"bootstrap"` and `"other"`, a supplied `rscales` is stored as given.

For a stratified jackknife whose PSUs are selected without replacement at a sampling fraction that is not negligible, the `scale` argument gives the entry to build here.

`fpc` A finite population correction, as a number or as one value per replicate column, or `NULL` (default, no correction). A number is recycled to every replicate. Accepted for `"JK1"`, `"JKn"` and `"other"` only, as in `survey::svrepdesign()`; every other type raises an error of class `surveycore_error_replicate_fpc_type_unsupported`. For `"bootstrap"` the correction belongs in the replicate weights themselves.

The correction is applied the way `survey::svrepdesign()` applies its own `fpc`: each `rscales` entry is multiplied by `1 - f`, where `f` is the sampling fraction for that replicate. A `NULL` `rscales` starts from `rep(1, R)`. For `"JKn"`, `f` is the sampling fraction of the stratum the replicate drops a PSU from, and `rscales` is still required. The design stores the corrected `rscales` and records the fraction, so `print()` shows it.

A column is not accepted. A bare name, or a single string, that names a column of `data` raises an error of class `surveycore_error_replicate_fpc_unsupported`, even when a variable of the same name exists in the calling environment. No replicate variance formula reads a per-row FPC column. To use such a variable, write `.env$name`. Any other expression is evaluated in the calling environment. A value that is not numeric, has the wrong length, holds `NA` or lies out of range raises an error of class `surveycore_error_replicate_fpc_invalid`.

`fpctype` How to read `fpc`: `"fraction"` (the sampling fraction `f`, with `0 <= f < 1`) or `"correction"` (`1 - f`, used as given, with `0 < 1 - f <= 1`). Required whenever `fpc` is supplied, as in `survey::svrepdesign()`, because the same number reads either way and the two give different standard errors. Must match exactly. Supplying `fpctype` without `fpc` raises an error of class `surveycore_error_replicate_fpc_unsupported`.

`mse` Logical. If `TRUE` (default), use mean-squared-error estimates (subtract the full-sample estimate rather than the mean replicate estimate when computing variance). Recommended for most designs.

`calibration` A list of calibration data elements, each produced by `as_caldata()`, or `NULL` (default). Stored at `@calibration` for provenance and reproducibility. **Not used in variance estimation**: the replicate variance estimator ignores `@calibration` entirely — calibration is already encoded in the replicate weights.

Known limitations (not validated at construction time):

- *Weight consistency*: surveycore cannot verify that `cd$w` encodes the same base weights as the design weight column.
- *Stale calibration after* `update_design()`: changing the weight column makes `@calibration` stale; clear it manually.

Value

A `survey_replicate` object. Columns imported with haven-style value labels are stored as their underlying type: the `label` and `labels` attributes are kept, and the value labels are also recorded in the metadata system, but the `haven_labelled` class is not stored. Use `survey_data(x, haven_class = TRUE)` to read the data back with that class rebuilt.

Tidy-select

Both `weights` and `repweights` support tidy-select syntax:

```
# Bare name for weights
as_survey_replicate(
  df, weights = wt, repweights = starts_with("repwt"), type = "BRR"
)
# c() for explicit replicate columns
```

```
as_survey_replicate(
  df, weights = wt, repweights = c(rep1, rep2, rep3), type = "JK1"
)
```

Replicate weight matrix

The replicate weight matrix is **not stored** in the object. Only the column names are stored in `@variables$repweights`. Variance estimation computes the matrix on demand: `as.matrix(design@data[, design@variables$repweights])`.

Memory usage

Each call to an estimation function (e.g., `get_means()`, `get_totals()`) materialises the full replicate weight matrix from the data frame. For large designs (e.g., ACS PUMS with 500k+ rows × 80 replicates), this is roughly `nrow * n_replicates * 8` bytes per call (~363 MB for ACS Wyoming × 80). If you are estimating many variables, this is repeated for each call. This behaviour matches the `survey` package reference implementation.

References

Canty, A.J. and Davison, A.C. (1999) Resampling-based variance estimation for labour force surveys. *The Statistician* **48**(3), 379–391.

Deville, J.-C. and Sarndal, C.-E. (1992) Calibration estimators in survey sampling. *Journal of the American Statistical Association* **87**(418), 376–382.

Deville, J.-C., Sarndal, C.-E. and Sautory, O. (1993) Generalized raking procedures in survey sampling. *Journal of the American Statistical Association* **88**(423), 1013–1020.

Judkins, D.R. (1990) Fay's method for variance estimation. *Journal of Official Statistics* **6**(3), 223–239.

Rao, J.N.K., Wu, C.F.J. and Yue, K. (1992) Some recent work on resampling methods for complex surveys. *Survey Methodology* **18**(2), 209–217.

Shao, J. and Tu, D. (1995) *The Jackknife and Bootstrap*. Springer.

See Also

`as_survey()` for Taylor series designs, `as_survey_twophase()` for two-phase designs, `set_var_label()` to add variable labels

Other constructors: `as_caldata()`, `as_survey()`, `as_survey_nonprob()`, `as_survey_twophase()`, `survey_glm()`, `survey_glm_fit()`, `survey_nonprob()`, `survey_replicate()`, `survey_taylor()`, `survey_twophase()`

Examples

```
# ACS PUMS Wyoming: 80 successive-difference replicate weights
d_acs <- as_survey_replicate(
  acs_pums_wy,
  weights = pwgtp,
  repweights = pwgtp1:pwgtp80,
  type = "successive-difference"
```

```
)

# Explicit replicate columns using c()
d_sub <- as_survey_replicate(
  acs_pums_wy,
  weights = pwgtp,
  repweights = c(pwgtp1, pwgtp2, pwgtp3, pwgtp4),
  type = "JK1"
)

# A finite population correction: a 5% sampling fraction, folded into
# the replicate scales
d_fpc <- as_survey_replicate(
  acs_pums_wy,
  weights = pwgtp,
  repweights = c(pwgtp1, pwgtp2, pwgtp3, pwgtp4),
  type = "JK1",
  fpc = 0.05,
  fpctype = "fraction"
)

# Fay's method: 4 replicate columns built with rho = 0.5
set.seed(1)
df_fay <- data.frame(
  y = rnorm(20),
  wt = runif(20, 1, 3)
)
for (r in 1:4) {
  factor_r <- sample(c(0.5, 1.5), 20, replace = TRUE)
  df_fay[[paste0("rep", r)]] <- df_fay$wt * factor_r
}
d_fay <- as_survey_replicate(
  df_fay,
  weights = wt,
  repweights = starts_with("rep"),
  type = "Fay",
  rho = 0.5
)
d_fay
```

`as_survey_twophase` *Create a Two-Phase Survey Design*

Description

Creates a two-phase (double) sampling design from an existing `survey_taylor` Phase 1 object. Phase 1 covers all rows; Phase 2 is a strict subset indicated by a logical column. Uses a tidy-select interface for all Phase 2 design variable arguments.

Usage

```
as_survey_twophase(
  phase1,
  ids2 = NULL,
  strata2 = NULL,
  probs2 = NULL,
  fpc2 = NULL,
  subset,
  method = c("full", "approx", "simple")
)
```

Arguments

phase1	A survey design object (inheriting from survey_base) representing the Phase 1 design. Accepts a survey_taylor object. A survey_replicate object is refused: the phase-1 variance for a replicate design is not implemented. Its @data must contain ALL rows from both phases, plus a logical indicator column for Phase 2 membership. Create with as_survey().
ids2	<tidy-select> Phase 2 cluster ID column: a single column, such as ids2 = psu2. Omit if Phase 2 has no within-stratum clustering. Without probs2 or fpc2, the Phase 2 inclusion probability is the fraction of clusters selected within each Phase 2 stratum, not the fraction of rows. Multi-stage Phase 2 is not supported. The two-phase variance uses one Phase 2 sampling stage, so selecting more than one column, such as ids2 = c(psu2, ssu2), is an error (surveycore_error_twophase_multistage_phase2) rather than a standard error that silently ignores the later stages.
strata2	<tidy-select> Phase 2 stratification column (a single column; more than one is an error, as Phase 2 has one sampling stage). Optional.
probs2	<tidy-select> Phase 2 inclusion probability column (a single column, values in (0, 1]). Optional.
fpc2	<tidy-select> Phase 2 finite population correction column (a single column; more than one is an error, as Phase 2 has one sampling stage). Optional.
subset	<tidy-select> Single logical column in phase1@data. TRUE = row selected into Phase 2; FALSE = Phase 1 only. Required. Must contain both TRUE and FALSE values (non-degenerate).
method	Character. Variance estimation method for combining Phase 1 and Phase 2 variability. One of "full" (default), "approx", or "simple". Case-sensitive. See Details. "full" and "approx" both match survey::twophase(), and both can give standard errors that are too large (conservative) for some designs: • "full" without an fpc on the Phase 1 design treats the Phase 1 sampling units (clusters, or rows when Phase 1 is unclustered) as Poisson-sampled. For a fixed-size Phase 1 sample that is very conservative for totals: in a simulation of a stratified cluster sample, about 13 times the true variance, or 3.7 times the standard error, and an unclustered, unstratified Phase 1 was overstated by a similar amount. Means are much less affected when Phase

1 is unstratified, but are still overstated when it is stratified. Supply `fpc` to `as_survey()` for Phase 1; a weighted Phase 1 with no `fpc` raises a warning; see Details.

- `"approx"` (and so `"simple"`) is conservative when Phase 1 is clustered and Phase 2 subsamples rows within the Phase 1 clusters, that is, when `ids2` is absent or finer than the Phase 1 cluster: in a simulation, about 2.4 times the true variance, or 1.55 times the standard error. It is close to unbiased when Phase 2 draws whole Phase 1 clusters or Phase 1 is unclustered. For the first kind of design, `method = "full"` with a Phase 1 `fpc` is the alternative.

Details

Variance methods:

- `"full"` — Full two-phase variance formula: the double-expansion Horvitz-Thompson estimator of `survey::twophase(method = "full")` (Breslow and Chatterjee, 1999). Accounts for variability in both phases, using the Phase 1 weights as inclusion probabilities, and treats a Phase 1 with no weights, strata or `fpc` as an iid sample. Requires Phase 2 design information (`probs2`, `ids2`, `strata2`) when Phase 2 is not a simple random subsample. If none of these are provided, an error is raised.
- `"approx"` — The two-phase variance formula from survey 3.14 and earlier: a Phase 1 term plus a Phase 2 term. Needs less memory than `"full"`, and is exact for some special cases, most importantly Phase 1 simple random sampling with stratified random sampling at Phase 2.
- `"simple"` — The same method as `"approx"`, as in `survey::twophase()`. The design records `"simple"`, and every estimate and standard error equals the `"approx"` one.

Phase 1 fpc and `method = "full"`:

The `"full"` variance needs the Phase 1 sampling probabilities and, for a fixed-size sample, the Phase 1 population sizes. Without a Phase 1 `fpc` it treats each Phase 1 sampling unit as drawn independently with probability `1 / weight` (Poisson sampling), so it never centres the unit totals. For a fixed-size Phase 1 sample the resulting standard error of a total can be several times too large, whether or not Phase 1 is clustered or stratified. With an `fpc`, the variance is that of a fixed-size sample in each Phase 1 stratum, and is close to unbiased.

`as_survey_twophase()` raises a warning of class `surveycore_warning_twophase_full_no_fpc` when all three hold: `method = "full"`, the Phase 1 design has no `fpc`, and the Phase 1 weights are not all 1. Phase 1 weights of all 1 mean the design was built with no weights, which `"full"` treats as an iid sample, or is a census, which contributes no Phase 1 variance; neither is overstated. The warning fires once, when the design is built. `from_svydesign()` raises the same warning for a `twophase2` object under the same rule.

Value

A `survey_twophase` object. Columns imported with haven-style value labels are stored as their underlying type: the `label` and `labels` attributes are kept, and the value labels are also recorded in the metadata system, but the `haven_labelled` class is not stored. Use `survey_data(x, haven_class = TRUE)` to read the data back with that class rebuilt.

References

Sarndal, C-E., Swensson, B. and Wretman, J. (1992) *Model Assisted Survey Sampling*. Springer.

Breslow, N.E. and Chatterjee, N. (1999) Design and analysis of two-phase studies with binary outcome applied to Wilms tumour prognosis. *Applied Statistics* **48**, 457–468.

Breslow, N., Lumley, T., Ballantyne, C.M., Chambless, L.E. and Kulick, M. (2009) Improved Horvitz-Thompson estimation of model parameters from two-phase stratified samples: applications in epidemiology. *Statistics in Biosciences*. doi:10.1007/s1256100990016

See Also

`as_survey()` for Taylor series designs

Other constructors: `as_caldata()`, `as_survey()`, `as_survey_nonprob()`, `as_survey_replicate()`, `survey_glm()`, `survey_glm_fit()`, `survey_nonprob()`, `survey_replicate()`, `survey_taylor()`, `survey_twophase()`

Examples

```
# Minimal two-phase design: Phase 1 = a sample of 20 from 40, Phase 2 =
# random subset. The Phase 1 fpc gives method = "full" the population size.
df <- data.frame(
  id = 1:20,
  wt = rep(2, 20),
  pop = rep(40, 20),
  in_phase2 = c(rep(TRUE, 10), rep(FALSE, 10)),
  y = rnorm(20)
)
phase1 <- as_survey(df, ids = id, weights = wt, fpc = pop)
d2 <- as_survey_twophase(phase1, subset = in_phase2)

# With Phase 2 stratification and inclusion probabilities
df2 <- data.frame(
  id = 1:30,
  wt = rep(3, 30),
  pop = rep(90, 30),
  in_phase2 = c(rep(TRUE, 15), rep(FALSE, 15)),
  arm = rep(c("A", "B", "C"), 10),
  subsamprate = rep(c(0.5, 0.7, 0.3), 10),
  y = rnorm(30)
)
phase1b <- as_survey(df2, ids = id, weights = wt, fpc = pop)
d2b <- as_survey_twophase(
  phase1b,
  strata2 = arm,
  probs2 = subsamprate,
  subset = in_phase2,
  method = "full"
)
```

as_svydesign *Convert a surveycore Design Object to a survey Package Design*

Description

Converts a survey_taylor, survey_replicate, survey_twophase, or survey_nonprob object to the corresponding survey package object: svydesign, svrepdesign, or twophase. Useful for accessing survey package estimation functions or for round-trip testing.

Usage

```
as_svydesign(x)
```

Arguments

x	A survey_taylor, survey_replicate, survey_twophase, or survey_nonprob object.

Details

Metadata (variable labels, value labels) is NOT carried over — the survey package has no metadata system.

Value

A survey::svydesign, survey::svrepdesign, or survey::twophase object. The returned class follows the input class, and for a survey_nonprob design it follows the design's shape: a design that names replicate weights returns a svrepdesign, and a design that names none returns a svydesign. A filtered input returns an object restricted to the active domain; the A filtered design's domain section below states what that means on each route. Value labels are not carried into the returned object — the survey package has no metadata system. To read the data back with haven-style classes rebuilt, use survey_data(x, haven_class = TRUE) on the surveycore design instead.

A non-probability design

A survey_nonprob design converts on the shape of its weights. A design that names replicate weights becomes a svrepdesign and keeps every replicate column, so the converted object computes the replicate variance surveycore computes. A design that names none becomes a svydesign with ids = ~1, and the call warns: the standard errors then use a simple random sample approximation that charges nothing for calibration uncertainty, and the returned object records nothing about it. See as_survey_nonprob() for the two variance modes.

The conversion drops the design's calibration provenance and its reference sample, and the round trip does not return a non-probability design. from_svydesign() on a converted design returns a survey_taylor or a survey_replicate object, because a survey object records nothing that marks a sample as non-probability. The rebuilt design reports design-based standard errors and no longer warns, on data that has not changed. Keep the original object when you need any of that.

Degrees of freedom on a converted design

surveycore reports `Inf` degrees of freedom for a `survey_nonprob` design, and the `survey` package computes a finite number from the converted object. A 40-row design with no replicate weights converts to 39, and the same design with 8 replicate weight columns converts to 7. Both counts carry a qualifier. Without replicate weights the `survey` package counts the rows whose weight is not zero and subtracts 1, so the same design with one zero-weight row converts to 38. With replicate weights it takes the numerical rank of the matrix of replicate weights and subtracts 1, so the figure is `R - 1` at full column rank and lower otherwise.

The difference reaches a result only when the caller asks for it. A default `confint()` call on a `svystat` or a `svrepstat` uses `df = Inf`, so the default interval from `survey::svymean()` matches surveycore's. The gap appears when the caller passes `degf(design)`, or calls a `survey` function that reads `degf()` itself, such as `svyglm()` or `svyttest()`. See `vignette("surveycore-vs-survey")` for the two counting rules in the source.

A filtered design's domain

The converted object represents the active domain and not the full stored sample. `survey::svymean()` on it answers the domain estimate.

A domain arrives from `filter()` in surveytidy. That verb keeps every row and marks domain membership in a logical column named by `SURVEYCORE_DOMAIN_COL`, which holds `"..surveycore_domain.."`. The restriction applies whenever that column is present. A design that was never filtered carries no such column and converts with no restriction.

The marker column stays in the converted object's data. It is an internal marker and not survey data. On the routes that remove rows, every value left in it is `TRUE`. On the two-phase route, which removes no row, the column arrives unchanged and still marks the zero-weighted rows `FALSE`. A row whose marker is `NA` counts as outside the domain.

A filter that matches no row still converts, and the conversion raises no surveycore condition on any route. What `survey` reports afterwards depends on the route:

- Taylor, and a non-probability design that names no replicate weights: an estimate of 0 with a standard error of 0.
- Replicate, and a non-probability design that names replicate weights: an error from `survey` at estimation time.
- Two-phase: `NaN`.

The two-phase route applies the domain by weighting the excluded rows out rather than by removing them. Each excluded row takes an infinite probability, so the converted object's row count does not change while its estimates answer for the domain.

A two-phase design with `method = "full"` converts to a `twophase2` object. `survey::twophase()` refuses a `weights` argument for that method, so the phase-1 weights are passed as inclusion probabilities, `probs = ~I(1 / w)`. A phase 1 with no weights, strata or `fpc` passes none, and `survey` then treats it as an iid sample, as surveycore does.

A round trip through `from_svydesign()` recovers the restricted rows and not the original sample. On the four routes that remove rows, the rebuilt design's row count is its new total. The original row count is unrecoverable from the object, because a `survey` object records a subset and not a marker. The all-`TRUE` marker column travels back, so the rebuilt design prints `Domain: n of n rows`, which

looks identical to a design that was never filtered. Printing a filtered design shows the domain count and the stored count; after a round trip the two counts are the same number twice. Keep the original design when you need the original row count. The two-phase route is the exception: it removes no row, so its round trip recovers the marker at both levels and the rebuilt design prints two different counts.

`as_tbl_svy()` hands a `tbl_svy` that is already restricted to `srvyr`. `srvyr`'s own `filter()` removes rows where surveycore's `filter()` marks a domain. The two verbs share a name and differ in semantics, so a caller who chains surveycore's `filter()` into `srvyr::filter()` filters an object that is already restricted.

See Also

`from_svydesign()` to convert back from a `survey` design

Other conversion: `as_tbl_svy()`, `from_svydesign()`, `from_tbl_svy()`

Examples

```
d <- as_survey(
  nhanes_2017,
  ids = sdmvpsu,
  weights = wtint2yr,
  strata = sdmvstra,
  nest = TRUE
)
if (requireNamespace("survey", quietly = TRUE)) {
  sv <- as_svydesign(d)
  survey::svymean(~ridageyr, sv, na.rm = TRUE)
}
```

---

as_tbl_svy *Convert a surveycore Design Object to an srvyr tbl_svy*

Description

Converts a surveycore design object to an `srvyr` `tbl_svy` by first converting to a `survey` design via `as_svydesign()` and then wrapping with `srvyr::as_survey()`. Requires both `survey` and `srvyr`.

Usage

```
as_tbl_svy(x)
```

Arguments

`x`	A `survey_taylor`, `survey_replicate`, `survey_twophase`, or `survey_nonprob` object.

Details

Metadata (variable labels, value labels) is NOT carried over.

Value

A `srvyr::tbl_svy` object. Value labels are not carried into the returned object — the `survey` package has no metadata system. To read the data back with haven-style classes rebuilt, use `survey_data(x, haven_class = TRUE)` on the surveycore design instead.

See Also

`from_tbl_svy()` to convert back from a `tbl_svy` object

Other conversion: `as_svydesign()`, `from_svydesign()`, `from_tbl_svy()`

Examples

```
d <- as_survey(
  nhanes_2017,
  ids = sdmvpsu,
  weights = wtint2yr,
  strata = sdmvstra,
  nest = TRUE
)
if (
  requireNamespace("survey", quietly = TRUE) &&
    requireNamespace("srvyr", quietly = TRUE)
) {
  ts <- as_tbl_svy(d)
}
```

---

ca_api_2000 *California Academic Performance Index 2000: Simple Random Sample*

Description

A simple random sample from the 2000 California Academic Performance Index (API) study. 200 schools were randomly sampled. This is the same underlying data as `apisrs` in the `survey` package, reformatted to surveycore conventions.

Usage

```
ca_api_2000
```

Format

A data frame with 200 rows and 38 variables:

pw Sampling weight (inverse probability of selection).
fpc FPC (number of schools in the California API system).
cds County/district/school code (character, 14-digit).
snum School number (integer).
dnum District number (integer).
name Short school name (character).
sname Full school name (character).
dname District name (character).
cname County name (character).
cnum County number (integer).
api00 API score 2000 (integer).
api99 API score 1999 (integer).
target API growth target (integer).
growth API score change, `api00 - api99` (integer).
pcttest Percent of students tested (integer).
sch_wide Met school-wide growth target (integer, 0 = No, 1 = Yes).
comp_imp Met comparable improvement target (integer, 0 = No, 1 = Yes).
both Met both targets (integer, 0 = No, 1 = Yes).
awards Eligible for awards program (integer, 0 = No, 1 = Yes).
stype School type (integer): 1 = Elementary, 2 = High, 3 = Middle.
yr_rnd Year-round school (integer, 0 = No, 1 = Yes).
meals Percent of students receiving free meals (integer).
ell Number of English language learners (integer).
mobility Percent of students in first year at school (integer).
enroll Total number of students (integer).
api_stu Number of students included in API 2000 (integer).
acs_k3 Average class size, grades K–3 (integer; `NA` for high and middle schools).
acs_46 Average class size, grades 4–6 (integer; `NA` for high schools and some others).
acs_core Average class size, core academic courses (integer; `NA` for most elementary schools).
not_hsg Percent of parents who did not complete high school (integer).
hsg Percent of parents who are high school graduates (integer).
some_col Percent of parents with some college (integer).
col_grad Percent of parents who are college graduates (integer).
grad_sch Percent of parents with graduate school education (integer).
avg_ed Average parent education level (numeric).
pct_resp Percent of parents who responded to the survey (integer).
full Percent of teachers fully credentialed (integer).
emer Percent of teachers on emergency credentials (integer).

Details

Survey design: Simple random sample. Use `as_survey()` with `weights = pw` and `fpc = fpc`:

```
svy <- as_survey(
  ca_api_2000,
  weights = pw,
  fpc = fpc
)
```

Missing values: Several columns have `NA` for schools where the value is inapplicable: `acs_k3` (grades K–3) is `NA` for high schools and middle schools, where those grade spans do not exist; `acs_46` (grades 4–6) is `NA` for all high schools and some elementary and middle schools; `acs_core` is `NA` for most elementary schools.

Metadata: All 38 columns carry `"label"` attributes (human-readable variable descriptions). The six categorical columns (`stype`, `sch_wide`, `comp_imp`, `both`, `awards`, `yr_rnd`) additionally carry `"labels"` attributes mapping integer codes to category names, compatible with surveycore's metadata system.

Relationship to `apisrs`: This dataset contains the same observations as `survey::apisrs`, with three differences: (1) the all-`NA` `flag` column is dropped; (2) factor columns are stored as plain integers with `labels` attributes; (3) column names are in snake_case.

Source

Lumley T (2004). *Analysis of complex survey samples.* Journal of Statistical Software, 9(1):1–19. Data distributed with the `survey` R package.

California Department of Education, Academic Performance Index 2000.

Examples

```
head(ca_api_2000[, c("pw", "fpc", "api00", "enroll")])

# Create an SRS design
svy <- as_survey(ca_api_2000, weights = pw, fpc = fpc)
svy

# Inspect variable label
attr(ca_api_2000$api00, "label")

# Inspect value labels for school type
attr(ca_api_2000$stype, "labels")
```

`classify_question_type` *Classify Variable Question Types*

Description

Groups variables by their shared `question_preface` metadata and classifies each group as one of `"single"`, `"sata"`, or `"battery"`. This is the single source of truth used by downstream export functions to decide how to render each question.

Usage

```
classify_question_type(x, ..., variable = NULL)
```

Arguments

`x`	A survey design object or `data.frame`.
`...`	<tidy-select> Variables to classify. Supports selection helpers: `tidyselect::starts_with()`, `tidyselect::all_of()`, `tidyselect::any_of()`, etc. Cannot be combined with `variable`.
`variable`	`character`. Alternative programmatic interface: character vector of variable names. Cannot be combined with `...`.

Details

The classification rules, applied per requested variable:

1. If the variable has no `question_preface`, or is the only requested variable sharing its preface, `type = "single"`.
2. If a `question_preface` is shared by 2+ requested variables and at least one is flagged via `set_sata()`, all variables in that group get `type = "sata"`.
3. Otherwise (shared preface, no SATA flag), all variables in the group get `type = "battery"`.

Group numbers are assigned sequentially by first appearance in the input.

Value

A tibble with columns:

- `variable` (character) — variable name
- `question_preface` (character) — the preface, or `NA` if none
- `type` (character) — one of `"single"`, `"sata"`, or `"battery"`
- `group` (integer) — group id; variables with the same non-NA preface share a group

See Also

set_sata(), extract_sata(), set_question_preface()

Other metadata: extract_data_name(), extract_dataset_metadata(), extract_field_dates(), extract_field_period(), extract_higher_is(), extract_metadata(), extract_missing_codes(), extract_question_preface(), extract_reverse_coded(), extract_sata(), extract_survey_name(), extract_universe(), extract_val_labels(), extract_var_extra(), extract_var_label(), extract_var_note(), extract_vendor(), infer_question_prefaces(), set_data_name(), set_dataset_metadata(), set_field_dates(), set_field_period(), set_higher_is(), set_missing_codes(), set_question_preface(), set_reverse_coded(), set_sata(), set_survey_name(), set_universe(), set_val_labels(), set_var_extra(), set_var_label(), set_var_note(), set_vendor(), survey_metadata(), survey_weighting_history()

Examples

```
d <- as_survey(
  nhanes_2017,
  ids = sdmvpsu,
  weights = wtint2yr,
  strata = sdmvstra,
  nest = TRUE
)
d <- set_question_preface(
  d,
  riagendr = "Demographics",
  ridageyr = "Demographics"
)
d <- set_sata(d, riagendr, ridageyr)
classify_question_type(d, riagendr, ridageyr, bpxsy1)
```

---

clean *Tidy a Survey GLM Fit*

Description

Converts a survey_glm_fit object into a survey_glm_tidy result tibble with one row per model coefficient (plus optional reference rows for factor predictors), design-based standard errors, confidence intervals, and structured metadata.

Usage

```
clean(
  model,
  conf_level = 0.95,
  include_reference = TRUE,
  n = FALSE,
  statistic = TRUE,
  exponentiate = FALSE,
```

```
  interaction_sep = " * ",
  ...
)
```

Arguments

model	A survey_glm_fit object from survey_glm().
conf_level	Numeric scalar in (0, 1). Confidence level for confidence intervals. Default 0.95.
include_reference	Logical. If TRUE, reference levels for unordered factor predictors appear as rows with estimate = NA and reference_row = TRUE. Default TRUE.
n	Logical. If TRUE, adds an n_obs column with the unweighted observation count per term. Default FALSE.
statistic	Logical. If TRUE (default), includes the statistic (t-statistic) column. Set to FALSE to drop it.
exponentiate	Logical. If TRUE, exponentiates estimate, conf_low, and conf_high. std_error is left on the log scale (matching broom convention). Fires surveycore_warning_exponentiate_nonlog when the model link is not log-based. Default FALSE.
interaction_sep	Character scalar. Separator for interaction term labels. Default " * ".
...	Currently unused.

Value

A survey_glm_tidy object: a tibble with S3 class c("survey_glm_tidy", "survey_result", "tbl_df", "tbl", "data.frame"). Metadata is accessed via meta().

See Also

survey_glm() to fit the model, meta() to access metadata.

Other analysis: SE(), get_anova(), get_corr(), get_covariance(), get_diffs(), get_effective_n(), get_freqs(), get_means(), get_pairwise(), get_quantiles(), get_ratios(), get_t_test(), get_totals(), get_variance(), meta()

Examples

```
d <- as_survey(
  gss_2024,
  ids = vpsu,
  weights = wtssps,
  strata = vstrat,
  nest = TRUE
)
fit <- survey_glm(d, age ~ sex)
clean(fit)
clean(fit, conf_level = 0.99, exponentiate = FALSE)
```

`coef.survey_result` *Extract coefficients from a survey result object*

Description

Returns a named numeric vector of point estimates from a `survey_result` object (produced by `get_means()`, `get_totals()`, `get_freqs()`, `get_ratios()`, `get_quantiles()`, `get_corr()`, `get_covariance()`, or `get_diffs()`).

Usage

```
## S3 method for class 'survey_result'
coef(object, ...)
```

Arguments

`object`	A `survey_result` object from any supported `get_*()` function.
`...`	Unused. Reserved for future extensions.

Details

Not all `survey_result` subclasses support `coef()`. `survey_t_test` and `survey_pairwise` objects throw `surveycore_error_result_method_unsupported`. Wide-format `survey_corr` objects (from `get_corr(format = "wide")`) also throw this error. For those objects, access columns directly (e.g., `result$estimate`, `result$se`).

Value

A named numeric vector. Names follow variable-major, group-secondary ordering matching `survey::coef.svyby()`. For grouped results, names are `"group_value:variable_name"` (e.g., `"A:y1"`, `"B:y1"`). For ungrouped results, names are the variable name (e.g., `"y1"`). See the spec for class-specific naming rules.

Examples

```
d <- as_survey(
  nhanes_2017,
  ids = sdmvpsu,
  weights = wtint2yr,
  strata = sdmvstra,
  nest = TRUE
)
result <- get_means(d, ridageyr, variance = "se")
coef(result)
```

confint.survey_result *Confidence intervals for survey result objects*

Description

Computes confidence intervals for the parameters of a survey_result object using the t-distribution with per-parameter degrees of freedom stored in attr(object, ".survey_result")$df.

Usage

```
## S3 method for class 'survey_result'
confint(object, parm, level = 0.95, ...)
```

Arguments

object	A survey_result object from any supported get_*() function.
parm	A numeric, integer, or character vector selecting a subset of parameters, or missing (all parameters). Character parm is matched against names(coef(object)).
level	Confidence level, strictly between 0 and 1. Default 0.95.
...	Unused.

Details

For non-calibrated Taylor, replicate, and nonprob designs, df = Inf is stored, so qt(1 - (1 - level)/2, df = Inf) equals qnorm(...) — a normal approximation. For calibrated Taylor designs, per-cell finite df values are used. Either way the stored df is the one behind the result's ci_low and ci_high columns, so for get_means(), get_totals(), get_freqs(), get_ratios() and get_covariance(), confint(object, level = l) returns the same interval as those columns when the result was computed at conf_level = l. This holds on a filtered (domain) design too.

When SE(object)[i] is NA, the corresponding CI row is c(NA, NA).

Value

A numeric matrix with two columns. Column names are "2.5 %" and "97.5 %" for level = 0.95 (matching stats::confint.default() format). Rownames are names(coef(object))[parm]. Dimensions: length(parm) × 2.

Examples

```
d <- as_survey(
  nhanes_2017,
  ids = sdmvpsu,
  weights = wtint2yr,
  strata = sdmvstra,
  nest = TRUE
)
result <- get_means(d, ridageyr, variance = "se")
```

```
confint(result)
confint(result, level = 0.90)
```

`extract_dataset_metadata` *Extract Dataset-Level Metadata*

Description

Returns whole-dataset metadata — the survey name, the display name, the fielding vendor, and the field dates — from a survey design object or a data frame.

Usage

```
extract_dataset_metadata(x, ..., key = NULL, format = "list", fill = NULL)
```

Arguments

`x`	A survey design object or `data.frame`.
`...`	Keys to return, as bare names (`vendor`) or strings (`"vendor"`). Supports `!!!` splicing of a character vector of names. Matching is by plain name and is **not** tidy-select: a key is not a column, so selection helpers such as `all_of()` are rejected. Repeated requests are deduplicated, keeping the first position. If empty, returns every key that is set.
`key`	`character` vector of key names, or `NULL` (default). A programmatic alternative to `...`; supplying both is an error. Must be supplied by name, because it follows `....`
`format`	`character(1)`. Output shape: `"list"` (default) or `"data_frame"`. Must be supplied by name, because it follows `....`
`fill`	`NULL` (default), `NA`, or `NA_character_`. Has two effects. For keys named in `...` or `key` that are not set: `NULL` omits them, while `NA` and `NA_character_` include them with a missing value. With an empty request, a non-`NULL` `fill` returns the full six-key schema with every absent key filled — the codebook idiom. Must be supplied by name, because it follows `....`

Details

The key vocabulary is closed. Exactly six keys are valid, and this is their canonical order:

`survey_name character(1)`. Full formal survey name.

`data_name character(1)`. Display label for this dataset.

`vendor character(1)`. Fielding vendor.

`field_start Date(1)`. First day in the field.

`field_end Date(1)`. Last day in the field.

`field_period character(1)`. Prose field period, for display.

survey_name and data_name are independent. This function never reads one to compute the other, and it never composes a label from the field dates.

On a survey design object the stored metadata is the single source of truth; attributes on the underlying data frame are not consulted. On a data frame the six attribute names above are read directly, plus the pre-1.2.0 name dates, which is read as field_period when no field_period attribute is present. Any other attribute is ignored. An attribute whose value fails the type rules above is dropped silently — this function never warns.

A design object restored from a file written by surveycore 1.1.0 or earlier reports no dataset metadata rather than failing.

extract_metadata() is deliberately separate: it reports per-variable fields only, and a dataset-level key is not a variable.

Each key also has a convenience extractor that returns one scalar: extract_survey_name(), extract_data_name(), extract_vendor(), extract_field_dates(), and extract_field_period().

Value

- "list" (default): named list of the stored values, in request order, or canonical key order for an empty request. Date keys are Date(1). A filled absent key is type-matched: as.Date(NA) for field_start and field_end, NA_character_ for the four character keys. Empty result: list().
- "data_frame": tibble with columns key and value, both character. A Date renders through format() as YYYY-MM-DD, and a filled absent key is always NA_character_ regardless of its type. Empty result: zero-row tibble with both columns.

See Also

Other metadata: classify_question_type(), extract_data_name(), extract_field_dates(), extract_field_period(), extract_higher_is(), extract_metadata(), extract_missing_codes(), extract_question_preface(), extract_reverse_coded(), extract_sata(), extract_survey_name(), extract_universe(), extract_val_labels(), extract_var_extra(), extract_var_label(), extract_var_note(), extract_vendor(), infer_question_prefaces(), set_data_name(), set_dataset_metadata(), set_field_dates(), set_field_period(), set_higher_is(), set_missing_codes(), set_question_preface(), set_reverse_coded(), set_sata(), set_survey_name(), set_universe(), set_val_labels(), set_var_extra(), set_var_label(), set_var_note(), set_vendor(), survey_metadata(), survey_weighting_history()

Examples

```
# On a data frame, dataset metadata lives in whole-object attributes.
df <- data.frame(id = 1:5, y = c(2, 4, 3, 5, 1), w = rep(1, 5))
attr(df, "survey_name") <- "Example Attitudes Survey 2026"
attr(df, "vendor") <- "Ipsos KnowledgePanel"
attr(df, "field_start") <- as.Date("2026-02-10")
attr(df, "field_end") <- as.Date("2026-03-04")

extract_dataset_metadata(df)
extract_dataset_metadata(df, vendor)
extract_dataset_metadata(df, key = c("field_start", "field_end"))
```

```
extract_dataset_metadata(df, format = "data_frame")

# An unset key is omitted by default, or filled on request.
extract_dataset_metadata(df, data_name)
extract_dataset_metadata(df, data_name, fill = NA)

# A non-NULL fill plus an empty request gives the whole six-key schema.
extract_dataset_metadata(df, fill = NA, format = "data_frame")
```

`extract_data_name` *Extract the Dataset Display Name*

Description

Returns the `data_name` dataset-level metadata key from a survey design object or a data frame.

Usage

```
extract_data_name(x)
```

Arguments

`x`	A survey design object or `data.frame`.

Details

Returns the stored `data_name` only. When the key is not set this function returns `NA_character_` even when `survey_name`, `field_start`, `field_end`, and `field_period` are all set. It never composes a label from the other keys. A design object restored from a file written by surveycore 1.1.0 or earlier reports `NA_character_` rather than failing.

Value

`character(1)`: the stored display name, or `NA_character_` when the key is not set.

See Also

Other metadata: `classify_question_type()`, `extract_dataset_metadata()`, `extract_field_dates()`, `extract_field_period()`, `extract_higher_is()`, `extract_metadata()`, `extract_missing_codes()`, `extract_question_preface()`, `extract_reverse_coded()`, `extract_sata()`, `extract_survey_name()`, `extract_universe()`, `extract_val_labels()`, `extract_var_extra()`, `extract_var_label()`, `extract_var_note()`, `extract_vendor()`, `infer_question_prefaces()`, `set_data_name()`, `set_dataset_metadata()`, `set_field_dates()`, `set_field_period()`, `set_higher_is()`, `set_missing_codes()`, `set_question_preface()`, `set_reverse_coded()`, `set_sata()`, `set_survey_name()`, `set_universe()`, `set_val_labels()`, `set_var_extra()`, `set_var_label()`, `set_var_note()`, `set_vendor()`, `survey_metadata()`, `survey_weighting_history()`

Examples

```
df <- data.frame(id = 1:5, y = c(2, 4, 3, 5, 1), w = rep(1, 5))
df <- set_survey_name(df, "Example Attitudes Survey 2026")

# A set survey_name does not fill data_name.
extract_data_name(df)

df <- set_data_name(df, "Example (February-March 2026)")
extract_data_name(df)
```

`extract_field_dates` *Extract the Field Dates*

Description

Returns the `field_start` and `field_end` dataset-level metadata keys from a survey design object or a data frame.

Usage

```
extract_field_dates(x)
```

Arguments

`x`	A survey design object or `data.frame`.

Details

Returns the two structured dates only. The prose field period has its own extractor, `extract_field_period()`, and this function never derives one from the other. An unset key is a normal state, reported as `as.Date(NA)`; this function never warns. A design object restored from a file written by surveycore 1.1.0 or earlier reports both entries as `as.Date(NA)` rather than failing.

Value

A named list of length 2, `field_start` then `field_end`, each a `Date` of length 1. An unset key is `as.Date(NA)`.

See Also

Other metadata: `classify_question_type()`, `extract_data_name()`, `extract_dataset_metadata()`, `extract_field_period()`, `extract_higher_is()`, `extract_metadata()`, `extract_missing_codes()`, `extract_question_preface()`, `extract_reverse_coded()`, `extract_sata()`, `extract_survey_name()`, `extract_universe()`, `extract_val_labels()`, `extract_var_extra()`, `extract_var_label()`, `extract_var_note()`, `extract_vendor()`, `infer_question_prefaces()`, `set_data_name()`, `set_dataset_metadata()`, `set_field_dates()`, `set_field_period()`, `set_higher_is()`, `set_missing_codes()`, `set_question_preface()`, `set_reverse_coded()`, `set_sata()`, `set_survey_name()`, `set_universe()`, `set_val_labels()`, `set_var_extra()`, `set_var_label()`, `set_var_note()`, `set_vendor()`, `survey_metadata()`, `survey_weighting_history()`

Examples

```
df <- data.frame(id = 1:5, y = c(2, 4, 3, 5, 1), w = rep(1, 5))
extract_field_dates(df)

df <- set_field_dates(df, "2026-02-10", "2026-03-04")
extract_field_dates(df)
```

`extract_field_period` *Extract the Prose Field Period*

Description

Returns the `field_period` dataset-level metadata key from a survey design object or a data frame.

Usage

```
extract_field_period(x)
```

Arguments

`x`	A survey design object or `data.frame`.

Details

Returns the stored `field_period` only. It never composes a period from `field_start` and `field_end` — use `extract_field_dates()` for those. On a data frame the pre-1.2.0 attribute name `dates` is read as `field_period` when no `field_period` attribute is present. A design object restored from a file written by surveycore 1.1.0 or earlier reports `NA_character_` rather than failing.

Value

`character(1)`: the stored field period, or `NA_character_` when the key is not set.

See Also

Other metadata: `classify_question_type()`, `extract_data_name()`, `extract_dataset_metadata()`, `extract_field_dates()`, `extract_higher_is()`, `extract_metadata()`, `extract_missing_codes()`, `extract_question_preface()`, `extract_reverse_coded()`, `extract_sata()`, `extract_survey_name()`, `extract_universe()`, `extract_val_labels()`, `extract_var_extra()`, `extract_var_label()`, `extract_var_note()`, `extract_vendor()`, `infer_question_prefaces()`, `set_data_name()`, `set_dataset_metadata()`, `set_field_dates()`, `set_field_period()`, `set_higher_is()`, `set_missing_codes()`, `set_question_preface()`, `set_reverse_coded()`, `set_sata()`, `set_survey_name()`, `set_universe()`, `set_val_labels()`, `set_var_extra()`, `set_var_label()`, `set_var_note()`, `set_vendor()`, `survey_metadata()`, `survey_weighting_history()`

Examples

```
df <- data.frame(id = 1:5, y = c(2, 4, 3, 5, 1), w = rep(1, 5))
extract_field_period(df)

df <- set_field_period(df, "February-March 2026")
extract_field_period(df)
```

---

extract_higher_is *Extract Direction-of-Improvement Attributes*

Description

Returns the direction-of-improvement (`"better"` or `"worse"`) for one or more variables in a survey design object or data frame. Variables with no direction set return `NA_character_`.

Usage

```
extract_higher_is(x, ..., variable = NULL)
```

Arguments

`x`	A survey design object or `data.frame`.
`...`	<tidy-select> Variables to query. If empty, returns direction for all columns of `x`. Mutually exclusive with `variable`.
`variable`	`character`. Variable name(s) — alternative to `...`. Mutually exclusive with `...`.

Value

A named character vector. Unset variables return `NA_character_`. Returns `character(0)` (named, zero-length) when all specified variables are missing from `x`.

See Also

`set_higher_is()` to set direction attributes

Other metadata: `classify_question_type()`, `extract_data_name()`, `extract_dataset_metadata()`, `extract_field_dates()`, `extract_field_period()`, `extract_metadata()`, `extract_missing_codes()`, `extract_question_preface()`, `extract_reverse_coded()`, `extract_sata()`, `extract_survey_name()`, `extract_universe()`, `extract_val_labels()`, `extract_var_extra()`, `extract_var_label()`, `extract_var_note()`, `extract_vendor()`, `infer_question_prefaces()`, `set_data_name()`, `set_dataset_metadata()`, `set_field_dates()`, `set_field_period()`, `set_higher_is()`, `set_missing_codes()`, `set_question_preface()`, `set_reverse_coded()`, `set_sata()`, `set_survey_name()`, `set_universe()`, `set_val_labels()`, `set_var_extra()`, `set_var_label()`, `set_var_note()`, `set_vendor()`, `survey_metadata()`, `survey_weighting_history()`

Examples

```
d <- as_survey(
  nhanes_2017,
  ids = sdmvpsu,
  weights = wtint2yr,
  strata = sdmvstra,
  nest = TRUE
)
d <- set_higher_is(d, bpxsy1 = "worse")
extract_higher_is(d, bpxsy1)
extract_higher_is(d)
```

`extract_metadata` *Extract All Metadata for Variables*

Description

Returns a summary of all metadata fields for one or more variables in a survey design object or data frame. Useful for auditing metadata state or building codebooks.

Usage

```
extract_metadata(x, ..., fill = NULL)
```

Arguments

`x`	A survey design object or `data.frame`.
`...`	<tidy-select> Variables to query. Supports selection helpers: `tidyselect::starts_with()`, `tidyselect::all_of()`, `tidyselect::any_of()`, `tidyselect::matches()`, etc. If empty, returns metadata for all variables. Use `tidyselect::any_of()` to silently skip missing variable names.
`fill`	`NULL` (default) or `"include"`. `NULL` omits variables that have no metadata in any field; `"include"` returns all variables regardless.

Details

This function reports per-variable fields only; dataset-level keys such as `vendor` and `field_start` are not variables, so they are excluded — read them with `extract_dataset_metadata()`.

Value

A named list. Each entry is a named list with keys: `variable_label`, `value_labels`, `question_preface`, `note`, `universe`, `missing_codes`, `transformations`.

See Also

Other metadata: classify_question_type(), extract_data_name(), extract_dataset_metadata(), extract_field_dates(), extract_field_period(), extract_higher_is(), extract_missing_codes(), extract_question_preface(), extract_reverse_coded(), extract_sata(), extract_survey_name(), extract_universe(), extract_val_labels(), extract_var_extra(), extract_var_label(), extract_var_note(), extract_vendor(), infer_question_prefaces(), set_data_name(), set_dataset_metadata(), set_field_dates(), set_field_period(), set_higher_is(), set_missing_codes(), set_question_preface(), set_reverse_coded(), set_sata(), set_survey_name(), set_universe(), set_val_labels(), set_var_extra(), set_var_label(), set_var_note(), set_vendor(), survey_metadata(), survey_weighting_history()

Examples

```
d <- as_survey(
  nhanes_2017,
  ids = sdmvpsu,
  weights = wtint2yr,
  strata = sdmvstra,
  nest = TRUE
)
d <- set_universe(d, ridageyr = "All participants 0+")
extract_metadata(d, ridageyr)
extract_metadata(d, fill = "include")
```

---

extract_missing_codes *Extract Missing Value Codes*

Description

Returns missing value sentinel codes for one or more variables in a survey design object or data frame.

Usage

```
extract_missing_codes(x, ..., format = "list", fill = NULL)
```

Arguments

x	A survey design object or data.frame.
...	<tidy-select> Variables to query. Supports selection helpers: tidyselect::starts_with(), tidyselect::all_of(), tidyselect::any_of(), tidyselect::matches(), etc. If empty, returns metadata for all variables. Use tidyselect::any_of() to silently skip missing variable names.
format	character(1). Output format: "list" (default) or "data_frame". "named_vector" is not valid for this function.
fill	Scalar or NULL. How to handle variables with no codes: NULL (default) omits them; NA_character_ includes them as NULL entries in "list" format. In "data_frame" format, variables with no codes are always excluded regardless of fill.

Value

- "list" (default): named list of atomic vectors. Empty: list().
- "data_frame": long-format tibble with columns variable, description (NA if codes vector is unnamed), code (coerced to character). Empty: zero-row tibble.

See Also

set_missing_codes() to set missing value codes

Other metadata: classify_question_type(), extract_data_name(), extract_dataset_metadata(), extract_field_dates(), extract_field_period(), extract_higher_is(), extract_metadata(), extract_question_preface(), extract_reverse_coded(), extract_sata(), extract_survey_name(), extract_universe(), extract_val_labels(), extract_var_extra(), extract_var_label(), extract_var_note(), extract_vendor(), infer_question_prefaces(), set_data_name(), set_dataset_metadata(), set_field_dates(), set_field_period(), set_higher_is(), set_missing_codes(), set_question_preface(), set_reverse_coded(), set_sata(), set_survey_name(), set_universe(), set_val_labels(), set_var_extra(), set_var_label(), set_var_note(), set_vendor(), survey_metadata(), survey_weighting_history()

Examples

```
d <- as_survey(
  nhanes_2017,
  ids = sdmvpsu,
  weights = wtint2yr,
  strata = sdmvstra,
  nest = TRUE
)
d <- set_missing_codes(d, ridageyr = c("Not applicable" = 999L))
extract_missing_codes(d, ridageyr)
extract_missing_codes(d, ridageyr, format = "data_frame")
```

---

extract_question_preface

Extract Question Prefaces

Description

Returns question preface text for one or more variables in a survey design object or data frame.

Usage

```
extract_question_preface(x, ..., format = "named_vector", fill = NULL)
```

Arguments

x	A survey design object or `data.frame`.
...	<tidy-select> Variables to query. Supports selection helpers: `tidyselect::starts_with()`, `tidyselect::all_of()`, `tidyselect::any_of()`, `tidyselect::matches()`, etc. If empty, returns metadata for all variables. Use `tidyselect::any_of()` to silently skip missing variable names.
format	`character(1)`. Output format: `"named_vector"` (default), `"list"`, or `"data_frame"`.
fill	Scalar or `NULL`. How to handle variables with no preface: `NULL` (default) omits them; `NA_character_` includes them with `NA`.

Value

- `"named_vector"` (default): named character vector. Empty: `character(0)`.
- `"list"`: named list of character scalars. Empty: `list()`.
- `"data_frame"`: tibble with columns `variable` and `preface`. Empty: zero-row tibble.

See Also

`set_question_preface()` to set a question preface

Other metadata: `classify_question_type()`, `extract_data_name()`, `extract_dataset_metadata()`, `extract_field_dates()`, `extract_field_period()`, `extract_higher_is()`, `extract_metadata()`, `extract_missing_codes()`, `extract_reverse_coded()`, `extract_sata()`, `extract_survey_name()`, `extract_universe()`, `extract_val_labels()`, `extract_var_extra()`, `extract_var_label()`, `extract_var_note()`, `extract_vendor()`, `infer_question_prefaces()`, `set_data_name()`, `set_dataset_metadata()`, `set_field_dates()`, `set_field_period()`, `set_higher_is()`, `set_missing_codes()`, `set_question_preface()`, `set_reverse_coded()`, `set_sata()`, `set_survey_name()`, `set_universe()`, `set_val_labels()`, `set_var_extra()`, `set_var_label()`, `set_var_note()`, `set_vendor()`, `survey_metadata()`, `survey_weighting_history()`

Examples

```
d <- as_survey(
  gss_2024,
  ids = vpsu,
  weights = wtssps,
  strata = vstrat,
  nest = TRUE
)
d <- set_question_preface(d, happy = "Taken all together...")
extract_question_preface(d, happy)
```

`extract_reverse_coded` *Extract Reverse-Coded Flags*

Description

Returns the reverse-coded status for one or more variables in a survey design object or data frame.

Usage

```
extract_reverse_coded(x, ..., variable = NULL)
```

Arguments

`x`	A survey design object or `data.frame`.
`...`	<tidy-select> Variables to query. If empty, returns reverse-coded status for all columns of `x`. Cannot be combined with `variable`.
`variable`	`character`. Alternative programmatic interface: character vector of variable names. Cannot be combined with `...`.

Value

A named logical vector. Variables not marked as reverse-coded return `FALSE`. When all specified variables are missing, returns `logical(0)`.

See Also

`set_reverse_coded()` to set reverse-coded flags

Other metadata: `classify_question_type()`, `extract_data_name()`, `extract_dataset_metadata()`, `extract_field_dates()`, `extract_field_period()`, `extract_higher_is()`, `extract_metadata()`, `extract_missing_codes()`, `extract_question_preface()`, `extract_sata()`, `extract_survey_name()`, `extract_universe()`, `extract_val_labels()`, `extract_var_extra()`, `extract_var_label()`, `extract_var_note()`, `extract_vendor()`, `infer_question_prefaces()`, `set_data_name()`, `set_dataset_metadata()`, `set_field_dates()`, `set_field_period()`, `set_higher_is()`, `set_missing_codes()`, `set_question_preface()`, `set_reverse_coded()`, `set_sata()`, `set_survey_name()`, `set_universe()`, `set_val_labels()`, `set_var_extra()`, `set_var_label()`, `set_var_note()`, `set_vendor()`, `survey_metadata()`, `survey_weighting_history()`

Examples

```
d <- as_survey(
  nhanes_2017,
  ids = sdmvpsu,
  weights = wtint2yr,
  strata = sdmvstra,
  nest = TRUE
)
d <- set_reverse_coded(d, bpxsy1)
```

```
extract_reverse_coded(d, bpxsy1)
extract_reverse_coded(d)
```

`extract_sata` *Extract SATA (Select-All-That-Apply) Flags*

Description

Returns the SATA status for one or more variables in a survey design object or a data frame.

Usage

```
extract_sata(x, ..., format = "named_vector", fill = FALSE)
```

Arguments

`x`	A survey design object or `data.frame`.
`...`	<tidy-select> Variables to query. Supports selection helpers: `tidyselect::starts_with()`, `tidyselect::all_of()`, `tidyselect::any_of()`, etc. If empty, returns SATA status for all columns of `x`.
`format`	`character(1)`. Output format: `"named_vector"` (default), `"list"`, or `"data_frame"`.
`fill`	`FALSE` (default) or `NULL`. Controls how unmarked variables are reported. `FALSE` includes them in the result with value `FALSE` (dense view); `NULL` omits them (sparse view). `TRUE` and other values are rejected.

Value

- `"named_vector"` (default): named logical vector. Empty: `logical(0)`.
- `"list"`: named list of logical scalars. Empty: `list()`.
- `"data_frame"`: tibble with columns `variable` (character) and `sata` (logical). Empty: zero-row tibble.

See Also

`set_sata()` to set SATA flags

Other metadata: `classify_question_type()`, `extract_data_name()`, `extract_dataset_metadata()`, `extract_field_dates()`, `extract_field_period()`, `extract_higher_is()`, `extract_metadata()`, `extract_missing_codes()`, `extract_question_preface()`, `extract_reverse_coded()`, `extract_survey_name()`, `extract_universe()`, `extract_val_labels()`, `extract_var_extra()`, `extract_var_label()`, `extract_var_note()`, `extract_vendor()`, `infer_question_prefaces()`, `set_data_name()`, `set_dataset_metadata()`, `set_field_dates()`, `set_field_period()`, `set_higher_is()`, `set_missing_codes()`, `set_question_preface()`, `set_reverse_coded()`, `set_sata()`, `set_survey_name()`, `set_universe()`, `set_val_labels()`, `set_var_extra()`, `set_var_label()`, `set_var_note()`, `set_vendor()`, `survey_metadata()`, `survey_weighting_history()`

Examples

```
d <- as_survey(
  nhanes_2017,
  ids = sdmvpsu,
  weights = wtint2yr,
  strata = sdmvstra,
  nest = TRUE
)
d <- set_sata(d, riagendr)
extract_sata(d, riagendr)
extract_sata(d, fill = NULL)
```

---

extract_survey_name *Extract the Formal Survey Name*

Description

Returns the survey_name dataset-level metadata key from a survey design object or a data frame.

Usage

```
extract_survey_name(x)
```

Arguments

x	A survey design object or data.frame.

Details

Returns the stored survey_name only. When the key is not set this function returns NA_character_ even if data_name is set: the two keys are independent, and no extractor substitutes one for the other. A design object restored from a file written by surveycore 1.1.0 or earlier reports NA_character_ rather than failing.

Value

character(1): the stored survey name, or NA_character_ when the key is not set.

See Also

Other metadata: classify_question_type(), extract_data_name(), extract_dataset_metadata(), extract_field_dates(), extract_field_period(), extract_higher_is(), extract_metadata(), extract_missing_codes(), extract_question_preface(), extract_reverse_coded(), extract_sata(), extract_universe(), extract_val_labels(), extract_var_extra(), extract_var_label(), extract_var_note(), extract_vendor(), infer_question_prefaces(), set_data_name(), set_dataset_metadata(), set_field_dates(), set_field_period(), set_higher_is(), set_missing_codes(), set_question_preface(), set_reverse_coded(), set_sata(), set_survey_name(), set_universe(), set_val_labels(), set_var_extra(), set_var_label(), set_var_note(), set_vendor(), survey_metadata(), survey_weighting_history()

Examples

```
df <- data.frame(id = 1:5, y = c(2, 4, 3, 5, 1), w = rep(1, 5))
extract_survey_name(df)

df <- set_survey_name(df, "Example Attitudes Survey 2026")
extract_survey_name(df)
```

`extract_universe` *Extract Universe Descriptions*

Description

Returns universe (eligibility) descriptions for one or more variables in a survey design object or data frame.

Usage

```
extract_universe(x, ..., format = "named_vector", fill = NULL)
```

Arguments

`x`	A survey design object or `data.frame`.
`...`	<tidy-select> Variables to query. Supports selection helpers: `tidyselect::starts_with()`, `tidyselect::all_of()`, `tidyselect::any_of()`, `tidyselect::matches()`, etc. If empty, returns metadata for all variables. Use `tidyselect::any_of()` to silently skip missing variable names.
`format`	`character(1)`. Output format: `"named_vector"` (default), `"list"`, or `"data_frame"`.
`fill`	Scalar or `NULL`. How to handle variables with no universe: `NULL` (default) omits them; `NA_character_` includes them with `NA`.

Value

- `"named_vector"` (default): named character vector. Empty: `character(0)`.
- `"list"`: named list of character scalars. Empty: `list()`.
- `"data_frame"`: tibble with columns `variable` and `universe`. Empty: zero-row tibble.

See Also

`set_universe()` to set a universe description

Other metadata: `classify_question_type()`, `extract_data_name()`, `extract_dataset_metadata()`, `extract_field_dates()`, `extract_field_period()`, `extract_higher_is()`, `extract_metadata()`, `extract_missing_codes()`, `extract_question_preface()`, `extract_reverse_coded()`, `extract_sata()`, `extract_survey_name()`, `extract_val_labels()`, `extract_var_extra()`, `extract_var_label()`, `extract_var_note()`, `extract_vendor()`, `infer_question_prefaces()`, `set_data_name()`, `set_dataset_metadata()`, `set_field_dates()`, `set_field_period()`, `set_higher_is()`, `set_missing_codes()`, `set_question_preface()`, `set_reverse_coded()`, `set_sata()`, `set_survey_name()`, `set_universe()`, `set_val_labels()`, `set_var_extra()`, `set_var_label()`, `set_var_note()`, `set_vendor()`, `survey_metadata()`, `survey_weighting_history()`

Examples

```
d <- as_survey(
  nhanes_2017,
  ids = sdmvpsu,
  weights = wtint2yr,
  strata = sdmvstra,
  nest = TRUE
)
d <- set_universe(d, ridageyr = "All participants 0+")
extract_universe(d)
extract_universe(d, ridageyr, format = "data_frame")
```

---

extract_val_labels *Extract Value Labels*

Description

Returns value labels for one or more variables in a survey design object or data frame.

Usage

```
extract_val_labels(x, ..., format = "list", fill = NULL)
```

Arguments

x	A survey design object or data.frame.
...	<tidy-select> Variables to query. Supports selection helpers: tidyselect::starts_with(), tidyselect::all_of(), tidyselect::any_of(), tidyselect::matches(), etc. If empty, returns metadata for all variables. Use tidyselect::any_of() to silently skip missing variable names.
format	character(1). Output format: "list" (default) or "data_frame". "named_vector" is not valid for this function.
fill	Scalar or NULL. How to handle variables with no labels: NULL (default) omits them; NA_character_ includes them as NULL entries in "list" format. In "data_frame" format, variables with no labels are always excluded regardless of fill.

Value

- "list" (default): named list of named vectors. Empty: list().
- "data_frame": long-format tibble with columns variable, label, value (codes coerced to character). Empty: zero-row tibble.

See Also

set_val_labels() to set value labels

Other metadata: classify_question_type(), extract_data_name(), extract_dataset_metadata(), extract_field_dates(), extract_field_period(), extract_higher_is(), extract_metadata(), extract_missing_codes(), extract_question_preface(), extract_reverse_coded(), extract_sata(), extract_survey_name(), extract_universe(), extract_var_extra(), extract_var_label(), extract_var_note(), extract_vendor(), infer_question_prefaces(), set_data_name(), set_dataset_metadata(), set_field_dates(), set_field_period(), set_higher_is(), set_missing_codes(), set_question_preface(), set_reverse_coded(), set_sata(), set_survey_name(), set_universe(), set_val_labels(), set_var_extra(), set_var_label(), set_var_note(), set_vendor(), survey_metadata(), survey_weighting_history()

Examples

```
d <- as_survey(
  nhanes_2017,
  ids = sdmvpsu,
  weights = wtint2yr,
  strata = sdmvstra,
  nest = TRUE
)
extract_val_labels(d, riagendr)
extract_val_labels(d, riagendr, format = "data_frame")
```

`extract_var_extra` *Extract Extension Slot Payload(s)*

Description

Returns the extension-slot payload for one or more variables in a survey design object or data frame. Read-only; performs no validation of payload contents — values are returned exactly as stored, never coerced.

Usage

```
extract_var_extra(x, ..., format = "list", fill = NULL)
```

Arguments

`x`	A survey design object or `data.frame`.
`...`	<tidy-select> Variables to query. Supports selection helpers: tidyselect::starts_with(), tidyselect::all_of(), tidyselect::any_of(), tidyselect::matches(), etc. If empty, returns the payload for every column in `x`. Use tidyselect::any_of() to silently skip missing variable names.
`format`	`character(1)`. Output format: `"list"` (default) or `"data_frame"`. `"named_vector"` is not valid for this function (payloads are not scalar).

`fill` Scalar or `NULL`. How to handle variables with no payload: `NULL` (default) omits them; `NA_character_` includes them as `NULL` entries in `"list"` format. In `"data_frame"` format, variables with no payload are always excluded regardless of `fill`.

Value

- `"list"` (default): named list. Keys are variable names; values are each variable's stored payload, or `NULL` for variables included via `fill = NA_character_` with no payload set. Empty: `list()`.
- `"data_frame"`: a tibble with columns `variable` (`character`), `key` (`character`), and `value` (a list-column, one raw payload value per row, unvalidated and untouched). One row per `(variable, key)` pair; a variable whose payload is `list()` or unset contributes zero rows. Empty: a zero-row tibble typed as above.

See Also

`set_var_extra()` to set an extension-slot payload

Other metadata: `classify_question_type()`, `extract_data_name()`, `extract_dataset_metadata()`, `extract_field_dates()`, `extract_field_period()`, `extract_higher_is()`, `extract_metadata()`, `extract_missing_codes()`, `extract_question_preface()`, `extract_reverse_coded()`, `extract_sata()`, `extract_survey_name()`, `extract_universe()`, `extract_val_labels()`, `extract_var_label()`, `extract_var_note()`, `extract_vendor()`, `infer_question_prefaces()`, `set_data_name()`, `set_dataset_metadata()`, `set_field_dates()`, `set_field_period()`, `set_higher_is()`, `set_missing_codes()`, `set_question_preface()`, `set_reverse_coded()`, `set_sata()`, `set_survey_name()`, `set_universe()`, `set_val_labels()`, `set_var_extra()`, `set_var_label()`, `set_var_note()`, `set_vendor()`, `survey_metadata()`, `survey_weighting_history()`

Examples

```
d <- as_survey(
  gss_2024,
  ids = vpsu,
  weights = wtssps,
  strata = vstrat,
  nest = TRUE
)
d <- set_var_extra(d, age = list(role = "demographic"))
extract_var_extra(d, age)
extract_var_extra(d, age, format = "data_frame")
```

---

extract_var_label *Extract Variable Labels*

Description

Returns variable labels for one or more variables in a survey design object or data frame.

Usage

```
extract_var_label(x, ..., format = "named_vector", fill = NULL)
```

Arguments

x	A survey design object or `data.frame`.
...	<tidy-select> Variables to query. Supports selection helpers: `tidyselect::starts_with()`, `tidyselect::all_of()`, `tidyselect::any_of()`, `tidyselect::matches()`, etc. If empty, returns metadata for all variables. Use `tidyselect::any_of()` to silently skip missing variable names.
format	`character(1)`. Output format: `"named_vector"` (default), `"list"`, or `"data_frame"`.
fill	Scalar or `NULL`. How to handle variables with no label: `NULL` (default) omits them; `NA_character_` includes them with `NA`.

Value

- `"named_vector"` (default): named character vector. Empty: `character(0)`.
- `"list"`: named list of character scalars. Empty: `list()`.
- `"data_frame"`: tibble with columns `variable` and `label`. Empty: zero-row tibble.

See Also

`set_var_label()` to set a variable label

Other metadata: `classify_question_type()`, `extract_data_name()`, `extract_dataset_metadata()`, `extract_field_dates()`, `extract_field_period()`, `extract_higher_is()`, `extract_metadata()`, `extract_missing_codes()`, `extract_question_preface()`, `extract_reverse_coded()`, `extract_sata()`, `extract_survey_name()`, `extract_universe()`, `extract_val_labels()`, `extract_var_extra()`, `extract_var_note()`, `extract_vendor()`, `infer_question_prefaces()`, `set_data_name()`, `set_dataset_metadata()`, `set_field_dates()`, `set_field_period()`, `set_higher_is()`, `set_missing_codes()`, `set_question_preface()`, `set_reverse_coded()`, `set_sata()`, `set_survey_name()`, `set_universe()`, `set_val_labels()`, `set_var_extra()`, `set_var_label()`, `set_var_note()`, `set_vendor()`, `survey_metadata()`, `survey_weighting_history()`

Examples

```
d <- as_survey(
  nhanes_2017,
  ids = sdmvpsu,
  weights = wtint2yr,
  strata = sdmvstra,
  nest = TRUE
)
extract_var_label(d)
extract_var_label(d, riagendr, ridageyr)
extract_var_label(d, format = "data_frame")
extract_var_label(d, fill = NA_character_)
```

`extract_var_note` *Extract Analyst Notes*

Description

Returns analyst notes for one or more variables in a survey design object or data frame.

Usage

```
extract_var_note(x, ..., format = "named_vector", fill = NULL)
```

Arguments

`x`	A survey design object or `data.frame`.
`...`	<tidy-select> Variables to query. Supports selection helpers: `tidyselect::starts_with()`, `tidyselect::all_of()`, `tidyselect::any_of()`, `tidyselect::matches()`, etc. If empty, returns metadata for all variables. Use `tidyselect::any_of()` to silently skip missing variable names.
`format`	`character(1)`. Output format: `"named_vector"` (default), `"list"`, or `"data_frame"`.
`fill`	Scalar or `NULL`. How to handle variables with no note: `NULL` (default) omits them; `NA_character_` includes them with `NA`.

Value

- `"named_vector"` (default): named character vector. Empty: `character(0)`.
- `"list"`: named list of character scalars. Empty: `list()`.
- `"data_frame"`: tibble with columns `variable` and `note`. Empty: zero-row tibble.

See Also

`set_var_note()` to set a note

Other metadata: `classify_question_type()`, `extract_data_name()`, `extract_dataset_metadata()`, `extract_field_dates()`, `extract_field_period()`, `extract_higher_is()`, `extract_metadata()`, `extract_missing_codes()`, `extract_question_preface()`, `extract_reverse_coded()`, `extract_sata()`, `extract_survey_name()`, `extract_universe()`, `extract_val_labels()`, `extract_var_extra()`, `extract_var_label()`, `extract_vendor()`, `infer_question_prefaces()`, `set_data_name()`, `set_dataset_metadata()`, `set_field_dates()`, `set_field_period()`, `set_higher_is()`, `set_missing_codes()`, `set_question_preface()`, `set_reverse_coded()`, `set_sata()`, `set_survey_name()`, `set_universe()`, `set_val_labels()`, `set_var_extra()`, `set_var_label()`, `set_var_note()`, `set_vendor()`, `survey_metadata()`, `survey_weighting_history()`

Examples

```
d <- as_survey(
  gss_2024,
  ids = vpsu,
  weights = wtssps,
  strata = vstrat,
  nest = TRUE
)
d <- set_var_note(d, age = "Top-coded at 89")
extract_var_note(d, age)
```

`extract_vendor` *Extract the Fielding Vendor*

Description

Returns the `vendor` dataset-level metadata key from a survey design object or a data frame.

Usage

```
extract_vendor(x)
```

Arguments

`x`	A survey design object or `data.frame`.

Details

Returns the stored value only. This function never composes a value from another key, and it never warns: an unset key is a normal state, reported as `NA_character_`. A design object restored from a file written by surveycore 1.1.0 or earlier reports `NA_character_` rather than failing.

Value

`character(1)`: the stored vendor, or `NA_character_` when the key is not set.

See Also

Other metadata: `classify_question_type()`, `extract_data_name()`, `extract_dataset_metadata()`, `extract_field_dates()`, `extract_field_period()`, `extract_higher_is()`, `extract_metadata()`, `extract_missing_codes()`, `extract_question_preface()`, `extract_reverse_coded()`, `extract_sata()`, `extract_survey_name()`, `extract_universe()`, `extract_val_labels()`, `extract_var_extra()`, `extract_var_label()`, `extract_var_note()`, `infer_question_prefaces()`, `set_data_name()`, `set_dataset_metadata()`, `set_field_dates()`, `set_field_period()`, `set_higher_is()`, `set_missing_codes()`, `set_question_preface()`, `set_reverse_coded()`, `set_sata()`, `set_survey_name()`, `set_universe()`, `set_val_labels()`, `set_var_extra()`, `set_var_label()`, `set_var_note()`, `set_vendor()`, `survey_metadata()`, `survey_weighting_history()`

Examples

```
df <- data.frame(id = 1:5, y = c(2, 4, 3, 5, 1), w = rep(1, 5))
extract_vendor(df)

df <- set_vendor(df, "Ipsos KnowledgePanel")
extract_vendor(df)
```

`from_svydesign` *Convert a survey Package Design to a surveycore Design Object*

Description

Converts a `survey` package design object (`svydesign`, `svrepdesign`, or `twophase`) to the corresponding surveycore S7 object. The data and the design variables are preserved; metadata (variable labels, value labels) is not — the `survey` package has no metadata system.

Usage

```
from_svydesign(x)
```

Arguments

`x`	A `survey::svydesign`, `survey::svrepdesign`, `survey::twophase`, `survey::twophase2`, or `srvyr::tbl_svy` object. Both `"twophase"` and `"twophase2"` classes from the `survey` package are dispatched to the two-phase conversion path.

Details

Weight column names are recovered from the design call when available. When the call does not contain a formula (e.g., weights were passed as a vector), the weight column is identified by matching the stored weight values against columns in the data. If no match is found, a `..surveycore_wt..` column is added.

Value

A `survey_taylor`, `survey_replicate`, or `survey_twophase` object.

Replicate weights

A `svrepdesign` object stores either finished replicate weights or replication factors, and reports which through its `combined.weights` field. On a design that reports replication factors the conversion transforms the replicate weights rather than preserving them: it multiplies each replicate column by the base weight, so the returned design always carries finished weights.

When the source design does not name its replicate columns — the case for every design that `survey::as.svrepdesign()` builds — the conversion writes a generated block into the data, one column per replicate, named on the same `..surveycore_wt..` pattern as the manufactured weight column above: `..surveycore_repwt_1..`, `..surveycore_repwt_2..`, and so on, with the index

zero-padded to the width of the replicate count. When the source does name its replicate columns, those names pass through unchanged.

Either way, the `repweights` design variable of the returned object names those columns, in replicate order.

Two-phase designs

The variance method comes from the object's class: a `twophase2` object, which `survey::twophase()` returns for `method = "full"`, converts to `method = "full"`, and a `twophase` object converts to `method = "approx"`. `survey` builds `method = "simple"` as a `twophase` object too, and surveycore treats `"simple"` as `"approx"`. A `twophase2` object whose phase 1 has weights other than 1 and no `fpc` raises the warning `surveycore_warning_twophase_full_no_fpc`, as `as_survey_twophase()` does; see its section on the Phase 1 fpc.

The phase 2 design, `ids2`, `strata2`, `probs2` and `fpc2`, is read from the formulas the `survey` object stores for it. A formula that names data columns directly, such as `~psu2` or `~strata`, keeps those names. Any other formula, such as `~I(1/w2)`, is evaluated on the phase 1 data as `survey` evaluates it, and the values go into a generated column named `..surveycore_ids2..`, `..surveycore_strata2..`, `..surveycore_probs2..` or `..surveycore_fpc2...` A phase 2 `weights` formula, which `method = "approx"` accepts, goes in as `..surveycore_probs2..`, because surveycore records phase 2 probabilities and not weights. The phase 1 weights of a `twophase2` object, which `survey` stores as `probs = ~I(1/w)`, keep the column name `w`.

Multi-stage phase 2 is not supported. A two-phase object whose phase 2 `id`, `strata` or `fpc` has more than one stage, such as `id = list(~psu, ~psu + ssu)`, raises the error `surveycore_error_twophase_multistage_phase2`, because surveycore's two-phase variance uses one phase 2 stage and would otherwise drop the later ones from the standard error. `survey` itself refuses a multi-stage phase 2 `strata` or `fpc` unless the phase 2 `id` also has more than one stage.

See Also

`as_svydesign()` to convert in the other direction

Other conversion: `as_svydesign()`, `as_tbl_svy()`, `from_tbl_svy()`

Examples

```
if (requireNamespace("survey", quietly = TRUE)) {
  sv <- survey::svydesign(
    ids = ~sdmvpsu,
    weights = ~wtint2yr,
    strata = ~sdmvstra,
    data = nhanes_2017,
    nest = TRUE
  )
  d <- from_svydesign(sv)
  survey_data(d)
}
```

from_tbl_svy *Convert an srvyr tbl_svy to a surveycore Design Object*

Description

Converts an srvyr tbl_svy to a surveycore design object by delegating to from_svydesign(). A tbl_svy IS a survey.design, so the conversion is structurally identical. Requires both survey and srvyr.

Usage

```
from_tbl_svy(x)
```

Arguments

x	A srvyr::tbl_svy object.

Value

A survey_taylor, survey_replicate, or survey_twophase object.

See Also

as_tbl_svy() to convert in the other direction

Other conversion: as_svydesign(), as_tbl_svy(), from_svydesign()

Examples

```
if (
  requireNamespace("survey", quietly = TRUE) &&
    requireNamespace("srvyr", quietly = TRUE)
) {
  ts <- srvyr::as_survey(
    survey::svydesign(
      ids = ~sdmvpsu,
      weights = ~wtint2yr,
      strata = ~sdmvstra,
      data = nhanes_2017,
      nest = TRUE
    )
  )
  d <- from_tbl_svy(ts)
}
```

get_anova *Design-Based Analysis of Variance and Wald Tests for Survey GLM Fits*

Description

Rao-Scott design-based ANOVA and design-based Wald tests for survey_glm() fits. Accepts three input shapes on object:

Usage

```
get_anova(
  object,
  formula = NULL,
  response = NULL,
  predictors = NULL,
  ...,
  method = c("LRT", "Wald"),
  test = c("F", "Chisq"),
  null = NULL,
  tolerance = sqrt(.Machine$double.eps),
  decimals = NULL,
  label_vars = TRUE,
  name_style = "surveycore"
)
```

Arguments

object	A survey_glm_fit, a list of survey_glm_fit objects, or a survey design (survey_base subclass).
formula	A model formula (e.g. y ~ x1 + x2). Only used when object is a survey design. Passed through to survey_glm(); supplying formula alongside response / predictors is rejected by survey_glm()'s validator.
response	Character string naming the outcome variable. Only used when object is a survey design. Forwarded to survey_glm().
predictors	Character vector of predictor variable names. Only used when object is a survey design. Forwarded to survey_glm().
...	Additional arguments forwarded to survey_glm() when object is a survey design (e.g. family, na.action, quiet). For fit or list inputs, ... must be empty — any extras error via rlang::check_dots_empty() with fuzzy typo detection.
method	Character(1). "LRT" (default) or "Wald".
test	Character(1). "F" (default) or "Chisq" reference distribution.

null	Numeric or NULL. Hypothesized value for the tested coefficients (Wald only). Only used when object is a single survey_glm_fit or a survey design (reducing to single-model mode); ignored with warning surveycore_warning_anova_null_ignored when object is a list of fits.
tolerance	Numeric(1). Reciprocal-condition-number threshold for the naive-covariance near-singular gate in the Rao-Scott LRT. Default sqrt(.Machine$double.eps).
decimals	Integer(1) or NULL. Round double output columns.
label_vars	Logical(1). When TRUE, compose term-row labels from @metadata@variable_labels for the term column. Default TRUE.
name_style	Character(1). "surveycore" (default) or "broom".

Details

- A single survey_glm_fit — sequential mode, one row per term.
- A list of survey_glm_fit objects — chained pairwise comparison, producing length(object) - 1 rows.
- A survey design (any survey_base subclass) — fits the model internally via survey_glm() using formula (or response + predictors), then runs sequential anova on the fit.

Supports the four method x test combinations shared with survey::anova.svyglm(): Rao-Scott working-LRT with F or Chisq reference, and design-based Wald with F or Chisq reference.

Value

A survey_anova tibble with columns term, statistic, df, ddf, deff, p_value, stars and a .meta attribute. When name_style = "broom", p_value is renamed to p.value and ddf is renamed to df.residual.

See Also

Other analysis: SE(), clean(), get_corr(), get_covariance(), get_diffs(), get_effective_n(), get_freqs(), get_means(), get_pairwise(), get_quantiles(), get_ratios(), get_t_test(), get_totals(), get_variance(), meta()

Examples

```
gss_cc <- gss_2024[stats::complete.cases(gss_2024[, c("age", "sex", "educ")]), ]
gss_design <- as_survey(
  gss_cc,
  ids = vpsu,
  weights = wtssps,
  strata = vstrat,
  nest = TRUE
)

# Single fit
fit <- survey_glm(gss_design, age ~ sex + educ)
get_anova(fit)
```

```
# Design + formula (fits internally)
get_anova(gss_design, age ~ sex + educ)

# List of fits (chained pairwise comparison)
fit_s <- survey_glm(gss_design, age ~ sex)
fit_b <- survey_glm(gss_design, age ~ sex + educ)
get_anova(list(fit_s, fit_b))
```

`get_corr` *Survey-Weighted Correlation (Pearson, Polychoric, Polyserial)*

Description

Compute pairwise correlations between two or more variables in a survey design, with design-based standard errors and confidence intervals. Returns results in long or wide format. The estimator is selected by `method`: `"pearson"` (default) for two numeric variables, `"polychoric"` for two ordinal variables under a bivariate-normal latent model (Olsson 1979), or `"polyserial"` for one ordinal + one continuous variable (Cox 1974). The survey-weighted polychoric and polyserial estimators (point estimates and design-based variance) are implemented from scratch following Mannan (2025); they are not derived from the `survey` package, which does not provide these estimators.

Usage

```
get_corr(
  design,
  x,
  group = NULL,
  format = c("long", "wide"),
  redundant = FALSE,
  diagonal = FALSE,
  variance = "ci",
  conf_level = 0.95,
  n_weighted = FALSE,
  decimals = NULL,
  min_cell_n = 30L,
  na.rm = TRUE,
  label_values = TRUE,
  label_vars = TRUE,
  name_style = "surveycore",
  method = "pearson",
  ...,
  .id = NULL,
  .if_missing_var = NULL
)
```

Arguments

design	A survey design object: survey_taylor, survey_replicate, survey_twophase, or survey_nonprob. method values "polychoric" and "polyserial" are supported on survey_taylor, survey_replicate, and survey_nonprob designs that supply replicate weights (repweights argument in as_survey_nonprob()). survey_nonprob without replicate weights and survey_twophase designs raise surveycore_error_polychoric_design_unsupported for latent methods.
x	<tidy-select> Two or more unquoted variable names. For method = "pearson", non-numeric columns are dropped with a warning. For method = "polychoric", every selected column must classify as ordinal — an ordered factor, an unordered factor, or a numeric column whose values are all whole numbers, none of them infinite, with 1 to 10 distinct values. Non-ordinal columns raise surveycore_error_polychoric_requires_ordinal. Missing values do not count: NA and NaN are removed before the distinct values are counted. A numeric column with no non-missing value at all is not ordinal, and it raises the same error. A factor or ordered column declares its levels, so it stays ordinal even when every value is missing; a bare numeric column has only its values to go on, and an empty one shows no scale. An all-NA factor pair returns an r of NA and an n of 0. A numeric column with few distinct whole values is treated as a scale. This is what makes the method work on a value-labelled column read from an SPSS, Stata or SAS file, where every coded scale arrives as a numeric column. It also means a coarse measurement in whole units is accepted as ordinal: three distinct values of 1000, 2000 and 3000 are read as a three-point scale, not as income in whole dollars. Two consequences to check before you call it: • Recode in-band missing codes to NA first. A four-point scale that still carries 8 for "Don't know" and 9 for "Refused" is read as a six-category scale, and the two codes become the top two categories. • Use method = "pearson" for a genuine count with few values, such as the number of people in a household. For method = "polyserial", each pair is canonicalized by type: one ordinal column and one continuous column. A pair of ordinal columns — including two numeric columns that both meet the rule above — raises surveycore_error_polyserial_requires_mi use method = "polychoric" for that pair. Logical, character and high-cardinality integer columns raise surveycore_error_polyserial_canonicalization_ambiguous.
group	<tidy-select> Optional grouping variable(s). Combined with any grouping set by group_by(). Default NULL.
format	"long" (default) or "wide". Long format returns one row per variable pair with inference statistics. Wide format returns the correlation matrix (r values only — no variance or inference columns). When group is active, group columns are prepended in both formats. Case-sensitive.
redundant	Logical. If FALSE (default), each pair appears once (lower triangle: pairs where var1 precedes var2 in input order). If TRUE, both (A, B) and (B, A) are included (full directed pairs). Only affects long format; wide format always shows the full symmetric matrix.

`diagonal` Logical. If `FALSE` (default), self-correlations are excluded (diagonal is `NA` in wide format). If `TRUE`, self-correlations (r equals 1) are included.

`variance` `NULL` or a character vector of one or more of `"se"`, `"ci"`, `"var"`, `"cv"`, `"moe"`, `"deff"`. Default `"ci"`. CI bounds use the Fisher Z transform (guaranteeing bounds in (-1, 1)). Only applies to long format.

`conf_level` Numeric scalar in (0, 1). Default `0.95`.

`n_weighted` Logical. If `TRUE`, add an `n_weighted` column with the pairwise sum of weights (both variables non-NA). Default `FALSE`.

`decimals` Integer or `NULL`. If an integer, rounds all numeric output columns (e.g., `r`, `se`, `ci_low`, `ci_high`) to this many decimal places. Default `NULL` (no rounding). Silently ignored when `format = "wide"` (the wide matrix contains only `r` values, which are not rounded by this argument).

`min_cell_n` Integer. Minimum pairwise unweighted count before `surveycore_warning_small_cell` fires. Default `30L` (AAPOR guidance).

`na.rm` Logical. Controls `NA` handling for group variables and the computation domain. Pairwise complete-case deletion is always applied for the correlation variables themselves regardless of this flag. If `TRUE` (default), observations where any group variable is `NA` are excluded from the output. If `FALSE`, observations where a group variable is `NA` are collected into their own group row in the output (appearing after all non-`NA` group rows).

`label_values` Logical. If `TRUE` (default) and the grouping variable has value labels, the group column is converted to a labelled factor. Has no visible effect when no groups are active.

`label_vars` Logical. If `TRUE` (default) and variable labels are set in metadata, `var1`/`var2` columns (long) and `variable` column (wide) show labels instead of raw names. Falls back to raw names if labels are unset.

`name_style` `"surveycore"` (default) or `"broom"`. When `"broom"`, renames `r` → `estimate`, `se` → `std.error`, etc. Only affects long format.

`method` Character(1). Estimator applied to every pair. One of `"pearson"` (default, sample-based product-moment correlation), `"polychoric"` (MLE under a bivariate-normal latent model for two ordinal variables), or `"polyserial"` (MLE for one ordinal + one continuous variable). The same `method` applies to every pair; it cannot be vectorised. Non-matching values raise the standard `base::match.arg()` signal.

`...` Unused. Reserved so that `.id` and `.if_missing_var` remain named-only when a `survey_collection` is passed as `design`.

`.id` Character(1) or `NULL`. Column name used to identify each survey when `design` is a `survey_collection`. For collection inputs, `NULL` (the default) resolves to the collection's stored `@id` property. Pass a non-`NULL` value to override. Ignored when `design` is a single survey.

`.if_missing_var` `"error"`, `"skip"`, or `NULL`. How to handle surveys in a collection that lack one of the requested NSE variables. For collection inputs, `NULL` (the default) resolves to the collection's stored `@if_missing_var` property. Pass a non-`NULL` value to override. Ignored when `design` is a single survey.

Details

Polychoric / polyserial semantics. For `method != "pearson"`, each pair is fit by a two-step MLE: weighted marginal thresholds (and, for polyserial, a weighted standardization of the continuous side) are estimated first, then `rho` is maximised over the weighted log-likelihood via `stats::optimize()` on `(-1 + 1e-6, 1 - 1e-6)`. Confidence intervals are constructed on the Fisher-z scale (`atanh(rho)`) and back-transformed via `tanh` with truncation to `[-1, 1]`. The Wald statistic `zeta.hat / SE(zeta.hat)` is referred to a standard normal distribution, so `df = NA_integer_` — distinct from the Pearson case where `df = n - 2` and the t-distribution is used. Column label attributes are method-neutral (e.g. `"statistic"`, not `"t-statistic"` / `"z-statistic"`); check `meta(result)$method` to interpret the values.

Bivariate-normal assumption. The polychoric / polyserial MLEs assume the underlying latent variables are jointly bivariate-normal. This is an unverified assumption; no runtime diagnostic is performed.

Taylor-path cost. On a `survey_taylor` design, the variance path for `method != "pearson"` is `O(n)` re-optimisations per variable pair (a perturbation-based influence function). For large `n` and many pairs, passing a `survey_replicate` design (one re-fit per replicate, not per respondent) is substantially faster.

Replicate-type caveat. Mannan (2025) verifies the replicate-weight variance formula for jackknife and bootstrap replicates. BRR and Fay replicates are admitted mechanically via the design's stored `scale` / `rscales` coefficients, but the paper does not validate their behaviour for this non-linear pseudo-likelihood estimator.

Value

A `survey_corr` tibble (also inheriting `survey_result`).

When `group` is active, group variable columns are prepended before all other columns in both long and wide formats.

Long format columns:

- `[group_cols...]` — group variable columns (when active), first.
- `var1`, `var2` — variable names (or labels when `label_vars = TRUE`).
- `r` — correlation coefficient. For `method = "pearson"`, the weighted product-moment correlation; for `"polychoric"` / `"polyserial"`, the MLE of `rho` under a bivariate-normal latent model.
- Variance columns (`se`, `var`, `cv`, `ci_low`, `ci_high`, `moe`, `deff`) — only those requested via `variance`.
- `p_value` — two-tailed p-value.
- `statistic` — test statistic. A t-statistic for `method = "pearson"`; a Wald z-statistic for latent methods.
- `df` — degrees of freedom. `n - 2` for `method = "pearson"`; `NA_integer_` for `"polychoric"` and `"polyserial"` (asymptotic normal distribution is used).
- `n` — pairwise unweighted count.
- `n_weighted` — pairwise sum of weights (only when requested).

Wide format columns:

- `[group_cols...]` — group variable columns (when active), first.
- `variable` — row variable names (or labels).
- One column per focal variable, containing `r` values.

Use `meta(result)` to access design type, variable labels, and `method` (`"pearson"`, `"polychoric"`, or `"polyserial"`). For `method != "pearson"`, `meta(result)$bivariate_normal_cdf` is `"pbivnorm"` (the bivariate-normal CDF used internally). When the replicate variance path observed one or more non-converged replicates, `meta(result)$n_failed_replicates_total` carries the scalar total.

References

Cox, N. R. (1974). Estimation of the correlation between a continuous and a discrete variable. *Biometrics*, 30(1), 171-178.

Mannan, H. (2025). SAS programs for estimation of weighted polychoric and weighted polyserial correlations in a complex survey. SSRN. doi:10.2139/ssrn.6580480

Olsson, U. (1979). Maximum likelihood estimation of the polychoric correlation coefficient. *Psychometrika*, 44(4), 443-460.

See Also

Other analysis: `SE()`, `clean()`, `get_anova()`, `get_covariance()`, `get_diffs()`, `get_effective_n()`, `get_freqs()`, `get_means()`, `get_pairwise()`, `get_quantiles()`, `get_ratios()`, `get_t_test()`, `get_totals()`, `get_variance()`, `meta()`

Examples

```
d <- as_survey(
  nhanes_2017,
  ids = sdmvpsu,
  weights = wtint2yr,
  strata = sdmvstra,
  nest = TRUE
)
get_corr(d, x = c(ridageyr, bpxsy1))

# Wide correlation matrix
get_corr(d, x = c(ridageyr, bpxsy1), format = "wide")

# AAPOR-compliant
get_corr(
  d,
  x = c(ridageyr, bpxsy1),
  variance = c("ci", "moe"),
  n_weighted = TRUE
)

# Polychoric correlation between two ordinal variables
df <- data.frame(
  id = 1:200,
  wt = runif(200, 0.5, 2),
```

```
  o1 = factor(sample(1:4, 200, replace = TRUE), ordered = TRUE),
  o2 = factor(sample(1:4, 200, replace = TRUE), ordered = TRUE)
)
d_ord <- as_survey(df, weights = wt)
get_corr(d_ord, x = c(o1, o2), method = "polychoric")
```

`get_covariance` *Design-Based Population Covariance for a Survey Design*

Description

Compute the design-based estimate of the finite-population Pearson covariance for every (unordered, by default) pair of numeric variables selected from `x`, with optional grouping, uncertainty quantification, and metadata-driven labelling. Matches the off-diagonal entries of `survey::svyvar()` (Kish `n/(n-1)` correction) on Taylor, replicate, twophase, and nonprob designs at numerical parity.

Usage

```
get_covariance(
  design,
  x,
  group = NULL,
  redundant = FALSE,
  diagonal = FALSE,
  variance = "ci",
  conf_level = 0.95,
  n_weighted = FALSE,
  decimals = NULL,
  min_cell_n = 30L,
  na.rm = TRUE,
  label_values = TRUE,
  label_vars = TRUE,
  name_style = "surveycore",
  ...,
  .id = NULL,
  .if_missing_var = NULL
)
```

Arguments

`design`	A survey design object: `survey_taylor`, `survey_replicate`, `survey_twophase`, or `survey_nonprob`. Also accepts a `survey_collection`.
`x`	<tidy-select> Two or more unquoted variable names. Must resolve to at least two columns. Non-numeric columns are dropped with a warning; if fewer than 2 numeric variables remain, an error is raised.
`group`	<tidy-select> Optional grouping variable(s). Combined with any grouping set by `group_by()`. Default `NULL`. Covariances are estimated separately within each group using that group's own weighted means for centring.

redundant	Logical. If FALSE (default), each unordered pair appears once in supply order (lower-triangle). If TRUE, both (A, B) and (B, A) are emitted.
diagonal	Logical. If FALSE (default), self-pairs (x, x) are excluded. If TRUE, one self-pair per variable is emitted with covariance = \eqn{\widehat{\mathrm{Var}}(x)}{Var_hat(x)} (the design-based variance – **not** 1).
variance	NULL or a character vector of one or more of "se", "ci", "var", "cv", "moe", "deff". Default "ci".
conf_level	Numeric scalar in (0, 1). Default 0.95.
n_weighted	Logical. If TRUE, append an n_weighted column with the pair's pairwise-complete sum of weights. Default FALSE.
decimals	Integer or NULL. If integer, rounds all numeric output columns to this many places. Default NULL (no rounding).
min_cell_n	Integer. Minimum pairwise unweighted count before surveycore_warning_small_cell fires. Default 30L (AAPOR).
na.rm	Logical. If TRUE (default), pairwise-complete deletion per pair, and rows with NA in any group variable are excluded from the output. If FALSE, NAs propagate to produce NaN estimates; NA group values are retained as their own group row.
label_values	Logical. If TRUE (default) and the grouping variable has value labels, the group column is converted to a labelled factor.
label_vars	Logical. If TRUE (default) and variable labels are set in metadata, var1 and var2 show labels instead of raw names.
name_style	"surveycore" (default) or "broom". Under "broom", renames covariance -> estimate, se -> std.error, ci_low -> conf.low, ci_high -> conf.high.
...	Unused. Reserved so that .id and .if_missing_var remain named-only when a survey_collection is passed as design.
.id	Character(1) or NULL. Column name used to identify each survey when design is a survey_collection. For collection inputs, NULL (the default) resolves to the collection's stored @id property. Pass a non-NULL value to override. Ignored when design is a single survey.
.if_missing_var	"error", "skip", or NULL. How to handle surveys in a collection that lack one of the requested NSE variables. For collection inputs, NULL (the default) resolves to the collection's stored @if_missing_var property. Pass a non-NULL value to override. Ignored when design is a single survey.

Details

Confidence intervals use the normal-Wald approximation on the SE of the covariance estimate: ci_low = covariance - z * se, ci_high = covariance + z * se, where z = qnorm((1 + conf_level) / 2). The bounds are **not clamped**. Covariance is unbounded — ci_low and ci_high may have opposite signs and may cross zero. Users who want clamped intervals can post-process. This behaviour matches survey::svyvar().

NA handling is **pairwise-complete per pair**: each ordered pair drops rows where either variable is NA. There is no na_handling argument; pairwise is the only policy. This matches survey::svyvar()

off-diagonal pair-at-a-time semantics, **not** `svyvar()`'s default listwise deletion across a multi-variable formula. Numerical parity therefore only holds when oracle calls are made pair-at-a-time (`survey::svyvar(~x + y, design)` per pair).

Under `diagonal = TRUE`, the self-pair `(x, x)` returns the design-based Kish-corrected **variance** of `x` on the active domain — not 1 as in `get_corr()`. The covariance matrix diagonal is the variance vector, not the identity. The diagonal-parity gate guarantees that `get_covariance(d, c(x, x), diagonal = TRUE)$covariance` and `$se` equal `get_variance(d, x)$variance` and `$se` numerically (point at `1e-10`, SE at `1e-8`) when the active domains match.

Design effect (`deff`) uses the Goodnight / Mood-Graybill SRS reference `SE_SRS(cov) = sqrt((Var(x) * Var(y) + cov^2) / (n - 1))`. When both the design SE and SRS SE are zero (constant-variable pairs), `deff` is set to exactly `0` (0 / 0 guard).

Value

A `survey_covariance` tibble (also inheriting `survey_result`). Columns, in order:

- `[group_cols...]` — group variable columns (when active), first.
- `var1`, `var2` — factor columns identifying the pair (levels in `x`-supply order).
- `covariance` — design-based Pearson covariance estimate (Kish-corrected). `NaN` for degenerate cells; `0` for pairs where at least one variable is constant on the active domain.
- Uncertainty columns (`se`, `var`, `cv`, `ci_low`, `ci_high`, `moe`, `deff`) — only those requested via `variance`.
- `n` — pairwise unweighted count.
- `n_weighted` — pair's sum of weights (only when requested).

References

Mood, A. M., Graybill, F. A., & Boes, D. C. (1974). *Introduction to the Theory of Statistics* (3rd ed.). McGraw-Hill.

Lumley, T. (2010). *Complex Surveys: A Guide to Analysis Using R*. Wiley.

Cochran, W. G. (1977). *Sampling Techniques* (3rd ed.). Wiley.

Demnati, A., & Rao, J. N. K. (2004). Linearization variance estimators for survey data. *Survey Methodology*, 30, 17–26.

See Also

Other analysis: `SE()`, `clean()`, `get_anova()`, `get_corr()`, `get_diffs()`, `get_effective_n()`, `get_freqs()`, `get_means()`, `get_pairwise()`, `get_quantiles()`, `get_ratios()`, `get_t_test()`, `get_totals()`, `get_variance()`, `meta()`

Examples

```
d <- as_survey(
  nhanes_2017,
  ids = sdmvpsu,
  weights = wtint2yr,
  strata = sdmvstra,
```

```
  nest = TRUE
)
get_covariance(d, x = c(ridageyr, bpxsy1))

# Include the diagonal (self-pairs return Var(x), not 1)
get_covariance(d, x = c(ridageyr, bpxsy1), diagonal = TRUE)

# With grouping
get_covariance(d, x = c(ridageyr, bpxsy1), group = riagendr)
```

`get_diffs` *Treatment Effect Estimation for Survey Designs*

Description

Estimates treatment effects (differences from a reference group) via survey-weighted regression. Supports bivariate and multivariate models, Gaussian and non-Gaussian families, and optional subgroup analysis.

Usage

```
get_diffs(
  design,
  x,
  treats,
  group = NULL,
  covariates = NULL,
  ref_level = NULL,
  pval_adj = NULL,
  show_means = TRUE,
  show_pct_change = FALSE,
  scale = c("ame", "link"),
  variance = "ci",
  conf_level = 0.95,
  alpha = 0.05,
  show_favorability = FALSE,
  min_cell_n = 30L,
  n_weighted = FALSE,
  decimals = NULL,
  na.rm = TRUE,
  label_values = TRUE,
  name_style = "surveycore",
  ...,
  .id = NULL,
  .if_missing_var = NULL
)
```

Arguments

design	A survey design object: survey_taylor, survey_replicate, survey_twophase, or survey_nonprob.
x	<tidy-select> A single unquoted numeric variable name for the dependent variable. Must resolve to exactly one numeric column (continuous or 0/1 binary).
treats	<tidy-select> A single unquoted variable name for the treatment/group variable. Must resolve to exactly one column with at least 2 unique levels. Coerced to factor if not already.
group	<tidy-select> Optional subgroup variable(s) for interaction analysis. When provided, treatment effects are reported separately within each subgroup. Combined with any grouping set by group_by(). Default NULL.
covariates	Character vector of additional model terms as strings. Supports interactions ("age * gender"), polynomials ("poly(edu, 2)"), and transformations ("log(income)"). When provided, forces the marginaleffects estimation path. Default NULL.
ref_level	Character(1). Reference level of treats for comparisons. If NULL (default), the first factor level is used. Must match an existing level.
pval_adj	Character(1) or NULL. P-value adjustment method passed to stats::p.adjust(). Options: "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none". NULL = no adjustment. When group is active, adjustment is applied independently within each group.
show_means	Logical. If TRUE (default), includes a mean column and a reference row with estimate = 0. Subject to link-scale suppression (see Details).
show_pct_change	Logical. If TRUE, includes a pct_change column: estimate / reference_mean. Subject to link-scale suppression (see Details). Default FALSE.
scale	Character(1). "ame" (default): average marginal effects on the response scale. "link": coefficients on the link scale. For Gaussian/identity models, both are identical. Case-sensitive.
variance	NULL or a character vector of one or more of "se", "ci". Controls which uncertainty columns appear. Default "ci".
conf_level	Numeric(1) in (0, 1). Confidence level. Default 0.95.
alpha	Numeric(1) strictly between 0 and 1. Significance threshold used when show_favorability = TRUE to classify whether a difference is statistically significant. Uses strict < (p < alpha). Default 0.05.
show_favorability	Logical. If TRUE, appends two logical columns to the result: favorable (the difference is statistically significant and in the direction indicated by higher_is metadata on x) and backlash (significant but in the opposite direction). Requires higher_is metadata set on x via set_higher_is(); if not set, both columns are all FALSE. Default FALSE.
min_cell_n	Integer(1). Minimum unweighted cell size before surveycore_warning_small_cell fires. Default 30L.

n_weighted	Logical. If TRUE, includes an n_weighted column with sum of weights per treatment level. Default FALSE.
decimals	Integer(1) or NULL. If non-NULL, rounds numeric output columns. pct_change is rounded to decimals + 2. Default NULL.
na.rm	Logical. If TRUE (default), rows with NA in x, treats, or group are dropped before fitting. If FALSE, NA values cause an error.
label_values	Logical. If TRUE (default), the treats and group columns display value labels from metadata instead of raw codes. Output type is factor when labels are applied.
name_style	"surveycore" (default) or "broom". When "broom", renames se to std.error, ci_low to conf.low, etc. The mean column is excluded from renaming.
...	Passed to survey_glm(). Common uses: family = quasibinomial().
.id	Character(1) or NULL. Column name used to identify each survey when design is a survey_collection. For collection inputs, NULL (the default) resolves to the collection's stored @id property. Pass a non-NULL value to override. Ignored when design is a single survey.
.if_missing_var	"error", "skip", or NULL. How to handle surveys in a collection that lack one of the requested NSE variables. For collection inputs, NULL (the default) resolves to the collection's stored @if_missing_var property. Pass a non-NULL value to override. Ignored when design is a single survey.

Details

Estimation Paths:

get_diffs() uses two estimation paths:

- **Clean path** (bivariate Gaussian with no covariates and no group, OR any family with scale = "link"): extracts coefficients directly from clean(). The intercept is the reference group mean; treatment coefficients are differences from reference. When scale = "link" and the family is non-Gaussian, mean and pct_change are suppressed.
- **Marginaleffects path** (covariates, non-Gaussian with scale = "ame", or group): uses avg_slopes() for estimates and avg_predictions() for means.

Link-Scale Suppression:

When scale = "link" and the family is non-Gaussian, the mean and pct_change columns are suppressed (omitted entirely). Link-scale means are not substantively meaningful.

P-Value Adjustment:

When group is active, p-value adjustment is applied independently within each group. For global adjustment across all comparisons, apply stats::p.adjust() to the result manually. Confidence intervals reflect the specified conf_level and are not affected by p-value adjustment.

Degrees of Freedom:

All p-values and confidence intervals use the t-distribution with design-based residual degrees of freedom, regardless of estimation path.

Non-Gaussian Models:

By default, non-Gaussian models report average marginal effects on the response scale. Set scale = "link" for coefficients on the link scale (e.g., log-odds for logistic regression).

Value

A survey_diffs tibble (also inheriting survey_result). Columns (in order): group columns (when active), treatment variable, estimate, pct_change (optional), mean (optional), n, n_weighted (optional), se (optional), ci_low (optional), ci_high (optional), p_value, stars, favorable (optional), backlash (optional). Use meta() to access design type, family, reference level, and other metadata.

See Also

Other analysis: SE(), clean(), get_anova(), get_corr(), get_covariance(), get_effective_n(), get_freqs(), get_means(), get_pairwise(), get_quantiles(), get_ratios(), get_t_test(), get_totals(), get_variance(), meta()

Examples

```
# Create survey design with treatment groups
set.seed(42)
df <- data.frame(
  id = 1:200,
  wt = runif(200, 0.5, 2),
  dv = rnorm(200, 50, 10),
  arm = factor(sample(c("Control", "A", "B"), 200, TRUE))
)
d <- as_survey(df, weights = wt)

# Basic treatment effect
get_diffs(d, dv, arm)

# With percentage change and p-value adjustment
get_diffs(d, dv, arm, show_pct_change = TRUE, pval_adj = "BH")
```

---

get_effective_n *Effective Sample Size for a Survey Design*

Description

Computes the effective sample size of a survey design using either the Kish (1965) weight-only approximation (method = "kish") or the full design-effect-based formula for a specified variable (method = "deff").

Usage

```
get_effective_n(
  design,
  x = NULL,
  group = NULL,
  method = c("kish", "deff"),
  na.rm = TRUE,
```

```
  decimals = NULL,
  min_cell_n = 30L,
  ...,
  .id = NULL,
  .if_missing_var = NULL
)
```

Arguments

design	A survey design object: survey_taylor, survey_replicate, survey_twophase, survey_nonprob, or a survey_collection.
x	<tidy-select> A single unquoted numeric variable name. Required when method = "deff"; ignored (with a message) when method = "kish". Default NULL.
group	<tidy-select> Optional grouping variable(s). Combined with any grouping set by group_by(). Default NULL.
method	Character(1). "kish" (default) or "deff". Controls the effective-N formula. Matched via match.arg().
na.rm	Logical. If TRUE (default), exclude observations with NA weights or group variables from the Kish computation; passed to get_means() for the DEFF computation.
decimals	Integer or NULL. Rounds n_eff and deff columns to this many decimal places. n is always integer and is never rounded. Default NULL.
min_cell_n	Integer. Minimum unweighted cell count before surveycore_warning_small_cell fires (Kish method only). Default 30L.
...	Unused. Reserved so that .id and .if_missing_var remain named-only when a survey_collection is passed.
.id	Character(1) or NULL. Column name identifying each survey in a survey_collection. Default NULL (uses the collection's stored @id).
.if_missing_var	"error", "skip", or NULL. Handling for surveys in a collection that lack x. Default NULL.

Details

The **Kish method** (method = "kish") computes effective N from survey weights alone: n_eff = sum(w)^2 / sum(w^2). It captures only weight variation. For clustered designs with equal weights, deff_kish = 1.0 even when the true design effect is substantially greater due to clustering. Use method = "deff" to capture the full design effect for a specific analysis variable.

The **DEFF method** (method = "deff") computes effective N as n_eff = n / DEFF, where DEFF = Var_design / Var_SRS for variable x. It captures clustering, stratification, and weight variation jointly.

Value

A survey_effective_n tibble (also inheriting survey_result). Columns, in order:

- `[.id]` — survey identifier column (when `design` is a collection).
- `[group_cols...]` — group variable columns (when grouping is active).
- `n` — integer. Unweighted count of observations.
- `n_eff` — numeric. Effective sample size.
- `deff_kish` — numeric. Weight-based design effect (`n / n_eff`). Present when `method = "kish"` only.
- `deff` — numeric. Full design effect (`Var_design / Var_SRS`). Present when `method = "deff"` only.

Use `meta(result)$method` to retrieve the formula used. For DEFF, `meta(result)$x` is a named list with variable metadata.

See Also

Other analysis: `SE()`, `clean()`, `get_anova()`, `get_corr()`, `get_covariance()`, `get_diffs()`, `get_freqs()`, `get_means()`, `get_pairwise()`, `get_quantiles()`, `get_ratios()`, `get_t_test()`, `get_totals()`, `get_variance()`, `meta()`

Examples

```
d <- as_survey(
  nhanes_2017,
  ids = sdmvpsu,
  weights = wtint2yr,
  strata = sdmvstra,
  nest = TRUE
)

# Kish effective N (weight-only approximation)
get_effective_n(d)

# Kish effective N by group (one row per level of riagendr)
get_effective_n(d, group = riagendr)

# Full DEFF effective N for a specific variable
get_effective_n(d, ridageyr, method = "deff")
```

`get_freqs` *Weighted Frequency Tables for Categorical Survey Variables*

Description

Compute weighted proportions (percentages) for one or more categorical variables in a survey design, with optional grouping, uncertainty quantification, and metadata-driven labelling.

Usage

```
get_freqs(
  design,
  x,
  ...,
  group = NULL,
  names_to = "name",
  values_to = "value",
  variance = NULL,
  conf_level = 0.95,
  n_weighted = FALSE,
  decimals = NULL,
  min_cell_n = 30L,
  na.rm = TRUE,
  label_values = TRUE,
  label_vars = TRUE,
  name_style = "surveycore",
  .id = NULL,
  .if_missing_var = NULL
)
```

Arguments

design	A survey design object: survey_taylor, survey_replicate, survey_twophase, or survey_nonprob.
x	<tidy-select> One or more categorical variables. Bare names or tidy-select helpers (e.g., c(q1, q2, q3)). When two or more variables are selected, multi-variable stacking mode is activated (see Details).
...	Additional arguments forwarded to .dispatch_over_collection() when design is a survey_collection. For single-survey inputs these arguments are ignored.
group	<tidy-select> Optional grouping variable(s). Combined with any grouping set by group_by(). Default NULL.
names_to	Character(1). Column name for the variable identifier in multi-variable mode. Default "name".
values_to	Character(1). Column name for the response value in multi-variable mode. Default "value".
variance	NULL or a character vector of one or more of "se", "ci", "var", "cv", "moe", "deff". Controls which uncertainty columns appear in the output. Default NULL (no uncertainty columns).
conf_level	Numeric scalar in (0, 1). Confidence level for intervals. Default 0.95.
n_weighted	Logical. If TRUE, add an n_weighted column with the sum of weights (estimated population count) per cell. Default FALSE.
decimals	Integer or NULL. If an integer, rounds all numeric output columns (e.g., pct, se, ci_low, ci_high) to this many decimal places. Default NULL (no rounding).
min_cell_n	Integer. Minimum unweighted cell count before surveycore_warning_small_cell fires. Default 30L (AAPOR guidance).

na.rm	Logical. If TRUE (default), NA values are excluded from analysis: observations where the focal variable is NA are dropped from frequency counts, and observations where any group variable is NA are excluded from the output. If FALSE, NA values in the focal variable appear as a dedicated frequency row in the output (not merely counted), and observations where a group variable is NA are collected into their own group row (appearing after all non-NA group rows).
label_values	Logical. If TRUE (default), convert raw variable values to labels using metadata or haven attributes. Falls back to raw values when no labels exist.
label_vars	Logical. If TRUE (default), use variable labels from metadata in the names_to column (multi-variable mode only). Falls back to the raw variable name when no label is set.
name_style	"surveycore" (default) or "broom". When "broom", renames pct → estimate, se → std.error, etc.
.id	Character(1) or NULL. Column name used to identify each survey when design is a survey_collection. For collection inputs, NULL (the default) resolves to the collection's stored @id property. Pass a non-NULL value to override. Ignored when design is a single survey.
.if_missing_var	"error", "skip", or NULL. How to handle surveys in a collection that lack one of the requested NSE variables. For collection inputs, NULL (the default) resolves to the collection's stored @if_missing_var property. Pass a non-NULL value to override. Ignored when design is a single survey.

Details

Single-variable mode (when x resolves to exactly one variable): The focal variable name becomes the first column. Rows follow the factor level order (if the variable is a factor) or ascending sort order otherwise.

Multi-variable mode (when x resolves to two or more variables): Results are stacked in long format. The names_to column contains the variable label (when label_vars = TRUE) or the raw variable name as fallback. The values_to column contains the response values.

Domain estimation: Proportions use the ratio linearization approach, equivalent to survey::svymean() on a binary indicator within the active domain. The full design structure is used for variance estimation — rows are not physically removed for domain/group subsets.

na.rm = FALSE**:** NA is appended as the last level. All proportions (including non-NA levels) have their denominator inflated to include NA rows, so the pct column sums to 1.

Value

A survey_freqs tibble (also inheriting survey_result). Columns:

- [group_cols...] — group variable columns (when active), first.
- [variable_name] (single) or [names_to] + [values_to] (multi).
- pct — weighted proportion (0–1).
- Variance columns (se, var, cv, ci_low, ci_high, moe, deff) — only those requested via variance.

- n — unweighted cell count (sample basis of each estimate).
- n_weighted — estimated population count (only when requested).

Use meta(result) to access design type, variable labels, value labels, and other metadata.

On a Taylor or replicate design (including a survey_nonprob design with replicate weights), the result also stores the joint covariance matrix of its estimates, covering every pair of rows. Read it with vcov(result); see vcov.survey_result().

See Also

Other analysis: SE(), clean(), get_anova(), get_corr(), get_covariance(), get_diffs(), get_effective_n(), get_means(), get_pairwise(), get_quantiles(), get_ratios(), get_t_test(), get_totals(), get_variance(), meta()

Examples

```
# NHANES exam weights are 0 for non-examined participants; filter first
nhanes_sub <- nhanes_2017[nhanes_2017$wtmec2yr > 0, ]
d <- as_survey(
  nhanes_sub,
  ids = sdmvpsu,
  weights = wtmec2yr,
  strata = sdmvstra,
  nest = TRUE
)

# Single variable
get_freqs(d, riagendr)

# With confidence intervals
get_freqs(d, riagendr, variance = "ci")

# Grouped
get_freqs(d, riagendr, group = sdmvstra)

# Multi-variable (stacked)
get_freqs(d, c(riagendr, ridreth3), names_to = "item", values_to = "value")
```

---

get_means *Weighted Mean for a Survey Design*

Description

Compute the weighted mean of a single numeric variable in a survey design, with optional grouping, uncertainty quantification, and metadata-driven labelling.

Usage

```
get_means(
  design,
  x,
  group = NULL,
  variance = "ci",
  conf_level = 0.95,
  n_weighted = FALSE,
  decimals = NULL,
  min_cell_n = 30L,
  na.rm = TRUE,
  label_values = TRUE,
  label_vars = TRUE,
  name_style = "surveycore",
  ...,
  .id = NULL,
  .if_missing_var = NULL
)
```

Arguments

design	A survey design object: survey_taylor, survey_replicate, survey_twophase, or survey_nonprob.
x	<tidy-select> A single unquoted numeric variable name. Must resolve to exactly one numeric column.
group	<tidy-select> Optional grouping variable(s). Combined with any grouping set by group_by(). Default NULL.
variance	NULL or a character vector of one or more of "se", "ci", "var", "cv", "moe", "deff". Controls which uncertainty columns appear in the output. Default "ci".
conf_level	Numeric scalar in (0, 1). Confidence level for intervals. Default 0.95.
n_weighted	Logical. If TRUE, add an n_weighted column with the sum of weights for non-NA observations in each group. Default FALSE.
decimals	Integer or NULL. If an integer, rounds all numeric output columns (e.g., mean, se, ci_low, ci_high) to this many decimal places. Default NULL (no rounding).
min_cell_n	Integer. Minimum unweighted cell count before surveycore_warning_small_cell fires. Default 30L (AAPOR guidance).
na.rm	Logical. If TRUE (default), NA values are excluded from analysis: observations where the analysis variable is NA are dropped from calculations, and observations where any group variable is NA are excluded from the output. If FALSE, NA observations in the analysis variable are included in calculations, and observations where a group variable is NA are collected into their own group row in the output (appearing after all non-NA group rows).
label_values	Logical. Accepted for API consistency across get_*() functions. For get_means(), no value-level cells appear in the output, so this parameter has no effect. Default TRUE.

label_vars — Logical. Accepted for API uniformity; has no visible effect since get_means() output contains no variable-name value cells. Default TRUE.

name_style — "surveycore" (default) or "broom". When "broom", renames mean → estimate, se → std.error, etc.

... — Unused. Reserved so that .id and .if_missing_var remain named-only when a survey_collection is passed as design.

.id — Character(1) or NULL. Column name used to identify each survey when design is a survey_collection. For collection inputs, NULL (the default) resolves to the collection's stored @id property. Pass a non-NULL value to override. Ignored when design is a single survey.

.if_missing_var — "error", "skip", or NULL. How to handle surveys in a collection that lack one of the requested NSE variables. For collection inputs, NULL (the default) resolves to the collection's stored @if_missing_var property. Pass a non-NULL value to override. Ignored when design is a single survey.

Value

A survey_means tibble (also inheriting survey_result). Columns:

- [group_cols...] — group variable columns (when active), first.
- mean — weighted mean estimate.
- Variance columns (se, var, cv, ci_low, ci_high, moe, deff) — only those requested via variance.
- df — degrees of freedom used for CI calculation. Present only for survey_taylor designs with an active @calibration object (GREG-corrected SE); counts only the PSUs and strata holding a row in the cell. For all other designs the normal approximation (Inf) is used and df is not included. confint() on the result uses the same df, so it reproduces ci_low and ci_high.
- n — unweighted count of non-NA observations used in the estimate.
- n_weighted — sum of weights (only when requested).

The variable name is stored in meta(result)$x, not as a column. Use meta(result) to access design type, variable labels, and other metadata.

On a Taylor or replicate design (including a survey_nonprob design with replicate weights), the result also stores the joint covariance matrix of its estimates, covering every pair of rows. Read it with vcov(result); see vcov.survey_result().

See Also

Other analysis: SE(), clean(), get_anova(), get_corr(), get_covariance(), get_diffs(), get_effective_n(), get_freqs(), get_pairwise(), get_quantiles(), get_ratios(), get_t_test(), get_totals(), get_variance(), meta()

Examples

```
d <- as_survey(
  nhanes_2017,
  ids = sdmvpsu,
  weights = wtint2yr,
  strata = sdmvstra,
  nest = TRUE
)
get_means(d, ridageyr)

# With grouped estimate
get_means(d, ridageyr, group = riagendr)

# AAPOR-compliant
get_means(d, ridageyr, variance = c("ci", "moe"), n_weighted = TRUE)
```

---

get_pairwise *All-Pairs Pairwise T-Tests for Survey Designs*

Description

Runs all k(k-1)/2 pairwise two-sample t-tests for a grouping variable with k levels and applies multiple-comparison p-value adjustment. Delegates pair-level computations to `get_t_test()`.

Usage

```
get_pairwise(
  design,
  x,
  by,
  group = NULL,
  pval_adj = "holm",
  conf_level = 0.95,
  variance = "ci",
  na.rm = TRUE,
  min_cell_n = 30L,
  decimals = NULL,
  label_values = TRUE,
  label_vars = TRUE,
  name_style = "surveycore",
  ...,
  .id = NULL,
  .if_missing_var = NULL
)
```

Arguments

design	A survey design object: survey_taylor, survey_replicate, survey_twophase, or survey_nonprob.
x	<tidy-select> A single unquoted numeric variable name for the outcome variable.
by	<tidy-select> A single unquoted variable name for the grouping variable. Must have at least 2 active levels.
group	<tidy-select> Optional subgroup variable(s). When supplied, pairwise comparisons are run within each group stratum. P-value adjustment is applied separately per stratum. Default NULL.
pval_adj	Character(1). P-value adjustment method passed to stats::p.adjust(). Default "holm". Use "none" for unadjusted p-values. Error: surveycore_error_invalid_pval_adj.
conf_level	Numeric(1). Confidence level strictly in (0, 1). Default 0.95.
variance	Character. Which uncertainty columns to include. Valid values: "se", "ci". Default "ci".
na.rm	Logical(1). Accepted for API uniformity. Default TRUE.
min_cell_n	Integer(1). Warn for small cells. Default 30L.
decimals	Integer(1) or NULL. Round all double output columns. Default NULL.
label_values	Logical(1). Convert by/group codes to value labels. Default TRUE.
label_vars	Logical(1). Accepted for API uniformity; no visible effect. Default TRUE.
name_style	Character(1). "surveycore" (default) or "broom".
...	Additional arguments forwarded to .dispatch_over_collection() when design is a survey_collection. For single-survey inputs these arguments are ignored.
.id	Character(1) or NULL. Column name used to identify each survey when design is a survey_collection. For collection inputs, NULL (the default) resolves to the collection's stored @id property. Pass a non-NULL value to override. Ignored when design is a single survey.
.if_missing_var	"error", "skip", or NULL. How to handle surveys in a collection that lack one of the requested NSE variables. For collection inputs, NULL (the default) resolves to the collection's stored @if_missing_var property. Pass a non-NULL value to override. Ignored when design is a single survey.

Value

A survey_pairwise tibble (also inheriting survey_result). Columns: group columns (when active), level_a, level_b, estimate, mean_a, mean_b, n_a, n_b, se (optional), ci_low (optional), ci_high (optional), t_stat, df, p_value (adjusted), stars. Use meta() to access the adjustment method and other metadata.

See Also

Other analysis: SE(), clean(), get_anova(), get_corr(), get_covariance(), get_diffs(), get_effective_n(), get_freqs(), get_means(), get_quantiles(), get_ratios(), get_t_test(), get_totals(), get_variance(), meta()

Examples

```
gss_sub <- gss_2024[gss_2024$sex %in% c(1L, 2L) & !is.na(gss_2024$age), ]
gss_sub$sex <- factor(
  gss_sub$sex,
  levels = c(1, 2),
  labels = c("Male", "Female")
)
gss_design <- as_survey(
  gss_sub,
  ids = vpsu,
  weights = wtssps,
  strata = vstrat,
  nest = TRUE
)
get_pairwise(gss_design, age, by = sex)
```

---

get_quantiles *Survey-Weighted Quantiles*

Description

Compute survey-weighted quantiles (including the median) for a single numeric variable using the Woodruff (1952) confidence interval method. Supports optional grouping, domain estimation, and all five survey design classes.

Usage

```
get_quantiles(
  design,
  x,
  probs = c(0.25, 0.5, 0.75),
  group = NULL,
  variance = "ci",
  conf_level = 0.95,
  n_weighted = FALSE,
  decimals = NULL,
  min_cell_n = 30L,
  na.rm = TRUE,
  label_values = TRUE,
  label_vars = TRUE,
  name_style = "surveycore",
  ...,
  .id = NULL,
  .if_missing_var = NULL
)
```

Arguments

design	A survey design object: survey_taylor, survey_replicate, survey_twophase, or survey_nonprob.
x	<tidy-select> A single unquoted numeric variable name. Must resolve to exactly one numeric column.
probs	Numeric vector of probabilities in (0, 1). Default c(0.25, 0.5, 0.75) (IQR + median).
group	<tidy-select> Optional grouping variable(s). Combined with any grouping set by group_by(). Default NULL.
variance	NULL or a character vector from "se", "ci", "var", "cv", "moe", "deff". Controls which uncertainty columns appear in the output. CIs use the Woodruff (1952) back-transformation method and are **not** symmetric around the estimate. The t critical value takes its degrees of freedom from the rows each estimate uses: only PSUs and strata holding a row in the filtered domain, the group, and (with na.rm = TRUE) the non-missing values count, as survey::degf() does on the design survey::svyquantile() receives. "deff" is always NA for quantiles (no closed-form SRS SE). Default "ci".
conf_level	Numeric scalar in (0, 1). Confidence level for Woodruff intervals. Default 0.95.
n_weighted	Logical. If TRUE, add an n_weighted column with the sum of weights for non-NA observations in each group. Default FALSE.
decimals	Integer or NULL. If an integer, rounds all numeric output columns (e.g., estimate, se, ci_low, ci_high) to this many decimal places. Default NULL (no rounding).
min_cell_n	Integer. Minimum unweighted cell count before surveycore_warning_small_cell fires. Default 30L (AAPOR guidance).
na.rm	Logical. If TRUE (default), NA values in the analysis variable are excluded from calculations. If FALSE, any NA values in the analysis variable cause all quantile estimates for that cell to be NA_real_. Observations where any group variable is NA are always excluded from the output when na.rm = TRUE; when na.rm = FALSE they are collected into their own group row (appearing after all non-NA rows).
label_values	Logical. Accepted for API consistency across get_*() functions. For get_quantiles(), no value-level cells appear in the output, so this parameter has no effect. Default TRUE.
label_vars	Logical. Accepted for API uniformity; has no visible effect on get_quantiles() output. Default TRUE.
name_style	"surveycore" (default) or "broom". When "broom", renames se → std.error, ci_low → conf.low, ci_high → conf.high. The estimate column is unchanged.
...	Unused. Reserved so that .id and .if_missing_var remain named-only when a survey_collection is passed as design.
.id	Character(1) or NULL. Column name used to identify each survey when design is a survey_collection. For collection inputs, NULL (the default) resolves to the collection's stored @id property. Pass a non-NULL value to override. Ignored when design is a single survey.

`.if_missing_var`
: `"error"`, `"skip"`, or `NULL`. How to handle surveys in a collection that lack one of the requested NSE variables. For collection inputs, `NULL` (the default) resolves to the collection's stored `@if_missing_var` property. Pass a non-`NULL` value to override. Ignored when `design` is a single survey.

Value

A `survey_quantiles` tibble (also inheriting `survey_result`).

- `[group_cols...]` — group variable columns (when active), first.
- `quantile` — probability label: `"p25"`, `"p50"`, etc.
- `estimate` — weighted quantile estimate.
- Variance columns (`se`, `var`, `cv`, `ci_low`, `ci_high`, `moe`, `deff`) — only those requested via `variance`. CIs are Woodruff intervals and are generally asymmetric around `estimate`. `deff` is always `NA` for quantile estimates: computing it requires a kernel density estimate at the quantile point (the Woodruff SRS approximation used by `survey::svyquantile(deff = TRUE)`), which is not implemented.
- `n` — unweighted count of observations in the active domain used in the estimate. When `na.rm = TRUE`, counts only non-NA observations; when `na.rm = FALSE`, counts all active-domain rows (including NAs, though the estimate will be `NA_real_`).
- `n_weighted` — sum of weights (only when requested).

One row per (group combination × quantile probability). The variable name and `probs` vector are stored in `meta(result)`.

References

Woodruff, R. S. (1952). Confidence intervals for medians and other position measures. *Journal of the American Statistical Association*, **47**(260), 635–646.

See Also

Other analysis: `SE()`, `clean()`, `get_anova()`, `get_corr()`, `get_covariance()`, `get_diffs()`, `get_effective_n()`, `get_freqs()`, `get_means()`, `get_pairwise()`, `get_ratios()`, `get_t_test()`, `get_totals()`, `get_variance()`, `meta()`

Examples

```
d <- as_survey(
  nhanes_2017,
  ids = sdmvpsu,
  weights = wtint2yr,
  strata = sdmvstra,
  nest = TRUE
)

# IQR + median (default)
get_quantiles(d, ridageyr)
```

```
# Median only with SE
get_quantiles(d, ridageyr, probs = 0.5, variance = c("ci", "se"))

# Grouped quartiles
get_quantiles(d, ridageyr, group = riagendr)
```

`get_ratios` *Survey-Weighted Ratio Estimation*

Description

Estimate the ratio of two survey-weighted totals (numerator / denominator) for a survey design object. Uses the delta method (linearization) for variance estimation for Taylor, SRS, calibrated, and two-phase designs, and direct per-replicate computation for replicate-weight designs. Both approaches are equivalent to `survey::svyratio()` for their respective design types. Supports optional grouping, domain estimation, and all five survey design classes.

Usage

```
get_ratios(
  design,
  numerator,
  denominator,
  group = NULL,
  variance = "ci",
  conf_level = 0.95,
  n_weighted = FALSE,
  decimals = NULL,
  min_cell_n = 30L,
  na.rm = TRUE,
  label_values = TRUE,
  label_vars = TRUE,
  name_style = "surveycore",
  ...,
  .id = NULL,
  .if_missing_var = NULL
)
```

Arguments

`design`	A survey design object: `survey_taylor`, `survey_replicate`, `survey_twophase`, or `survey_nonprob`.
`numerator`	<tidy-select> A single unquoted numeric variable name for the numerator. Must resolve to exactly one numeric column.
`denominator`	<tidy-select> A single unquoted numeric variable name for the denominator. Must resolve to exactly one numeric column. All in-domain values must not sum to zero.

group — <tidy-select> Optional grouping variable(s). Combined with any grouping set by group_by(). Rows where the grouping variable is NA are excluded from all groups and do not appear in the output. This matches dplyr::group_by() semantics. Default NULL.

variance — NULL or a character vector from "se", "ci", "var", "cv", "moe", "deff". Controls which uncertainty columns appear in the output. Default "ci".

conf_level — Numeric scalar in (0, 1). Confidence level for confidence intervals. Default 0.95.

n_weighted — Logical. If TRUE, add an n_weighted column with the sum of weights for rows where both numerator and denominator are non-NA in each group. Default FALSE.

decimals — Integer or NULL. If an integer, rounds all numeric output columns (e.g., ratio, se, ci_low, ci_high) to this many decimal places. Default NULL (no rounding).

min_cell_n — Integer. Minimum unweighted cell count before surveycore_warning_small_cell fires. Default 30L (AAPOR guidance).

na.rm — Logical. If TRUE (default), NA values are excluded from analysis: observations where the analysis variable is NA are dropped from calculations, and observations where any group variable is NA are excluded from the output. If FALSE, NA observations in the analysis variable are included in calculations, and observations where a group variable is NA are collected into their own group row in the output (appearing after all non-NA group rows).

label_values — Logical. Accepted for API consistency across get_*() functions. For get_ratios(), no value-level cells appear in the output, so this parameter has no effect. Default TRUE.

label_vars — Logical. Accepted for API uniformity; has no visible effect on get_ratios() output. Default TRUE.

name_style — "surveycore" (default) or "broom". When "broom", renames ratio → estimate, se → std.error, ci_low → conf.low, ci_high → conf.high.

... — Unused. Reserved so that .id and .if_missing_var remain named-only when a survey_collection is passed as design.

.id — Character(1) or NULL. Column name used to identify each survey when design is a survey_collection. For collection inputs, NULL (the default) resolves to the collection's stored @id property. Pass a non-NULL value to override. Ignored when design is a single survey.

.if_missing_var — "error", "skip", or NULL. How to handle surveys in a collection that lack one of the requested NSE variables. For collection inputs, NULL (the default) resolves to the collection's stored @if_missing_var property. Pass a non-NULL value to override. Ignored when design is a single survey.

Value

A survey_ratios tibble (also inheriting survey_result).

- [group_cols...] — group variable columns (when active), first.

- `ratio` — estimated ratio (weighted total of numerator / weighted total of denominator).
- Variance columns (`se`, `var`, `cv`, `ci_low`, `ci_high`, `moe`, `deff`) — only those requested via `variance`.
- `n` — unweighted count of rows where both numerator and denominator are non-NA.
- `n_weighted` — sum of weights (only when requested).

Numerator and denominator variable names are stored in `meta(result)`, not as output columns. Use `meta(result)$numerator` and `meta(result)$denominator` to access them.

On a Taylor or replicate design (including a `survey_nonprob` design with replicate weights), the result also stores the joint covariance matrix of its estimates, covering every pair of rows. Read it with `vcov(result)`; see `vcov.survey_result()`.

See Also

Other analysis: `SE()`, `clean()`, `get_anova()`, `get_corr()`, `get_covariance()`, `get_diffs()`, `get_effective_n()`, `get_freqs()`, `get_means()`, `get_pairwise()`, `get_quantiles()`, `get_t_test()`, `get_totals()`, `get_variance()`, `meta()`

Examples

```
d <- as_survey(pew_npors_2025, weights = weight, strata = stratum)

# Ratio of prayer frequency to in-person attendance frequency
get_ratios(d, numerator = pray, denominator = attendper)

# With grouped estimates
get_ratios(d, pray, attendper, group = gender)

# AAPOR-compliant output
get_ratios(d, pray, attendper, variance = c("ci", "moe"), n_weighted = TRUE)
```

---

get_totals *Weighted Total for a Survey Design*

Description

Compute the estimated population total of a numeric variable in a survey design, one total per level of a categorical variable, or the estimated population size when no variable is supplied. Supports optional grouping, uncertainty quantification, and metadata-driven labelling.

Usage

```
get_totals(
  design,
  x = NULL,
  group = NULL,
  variance = "ci",
```

```
  conf_level = 0.95,
  n_weighted = FALSE,
  decimals = NULL,
  min_cell_n = 30L,
  na.rm = TRUE,
  label_values = TRUE,
  label_vars = TRUE,
  name_style = "surveycore",
  ...,
  .id = NULL,
  .if_missing_var = NULL
)
```

Arguments

design	A survey design object: survey_taylor, survey_replicate, survey_twophase, or survey_nonprob.
x	<tidy-select> Optional single unquoted variable name. When NULL (default), estimates the population size (sum of weights). When numeric, estimates the weighted sum (sum of w_i * x_i). When a factor or character vector, returns one row per level: the level's total is the weighted sum of its 0/1 indicator, the same estimate and standard error as survey::svytotal() on a factor. Factor levels keep their declared order and character levels sort ascending. A level with no observed rows in the active domain is dropped, as in get_freqs(); survey::svytotal() returns it as a zero row. Logical and other column types are rejected.
group	<tidy-select> Optional grouping variable(s). Default NULL.
variance	NULL or a character vector from "se", "ci", "var", "cv", "moe", "deff". Default "ci".
conf_level	Numeric scalar in (0, 1). Default 0.95.
n_weighted	Logical. For get_totals(d) (no variable), equals the total column and is included for API uniformity. For a numeric x, adds the sum of weights for non-NA observations. For a categorical x, adds the sum of weights at each level, which equals total. Default FALSE.
decimals	Integer or NULL. If an integer, rounds all numeric output columns (e.g., total, se, ci_low, ci_high) to this many decimal places. Default NULL (no rounding).
min_cell_n	Integer. Default 30L.
na.rm	Logical. If TRUE (default), NA values are excluded from analysis: observations where the analysis variable is NA are dropped from calculations, and observations where any group variable is NA are excluded from the output. If FALSE, NA observations in the analysis variable are included in calculations, and observations where a group variable is NA are collected into their own group row in the output (appearing after all non-NA group rows). For a categorical x, FALSE also adds an NA level row after the other levels, holding the total of the rows where x is NA.

label_values — Logical. If TRUE (default), group columns and the level column of a categorical x show value labels where they exist. For a numeric x, no value-level cells appear in the output.

label_vars — Logical. Accepted for API uniformity. Default TRUE.

name_style — "surveycore" (default) or "broom".

... — Additional arguments forwarded to .dispatch_over_collection() when design is a survey_collection. For single-survey inputs these arguments are ignored.

.id — Character(1) or NULL. Column name used to identify each survey when design is a survey_collection. For collection inputs, NULL (the default) resolves to the collection's stored @id property. Pass a non-NULL value to override. Ignored when design is a single survey.

.if_missing_var — "error", "skip", or NULL. How to handle surveys in a collection that lack one of the requested NSE variables. For collection inputs, NULL (the default) resolves to the collection's stored @if_missing_var property. Pass a non-NULL value to override. Ignored when design is a single survey.

Value

A survey_totals tibble (also inheriting survey_result). Columns:

- [group_cols...] — group variable columns (when active), first.
- [x] — the level of a categorical x, named after the variable (categorical x only).
- total — the weighted sum estimate.
- Variance columns — only those requested via variance.
- n — unweighted count (omitted in no-variable mode). For a categorical x, the count of rows at the level.
- n_weighted — sum of weights (only when requested).

The variable name (or NULL for no-variable mode) is in meta(result)$x. Use meta(result) for additional metadata.

On a Taylor or replicate design (including a survey_nonprob design with replicate weights), the result also stores the joint covariance matrix of its estimates, covering every pair of rows. Read it with vcov(result); see vcov.survey_result().

See Also

Other analysis: SE(), clean(), get_anova(), get_corr(), get_covariance(), get_diffs(), get_effective_n(), get_freqs(), get_means(), get_pairwise(), get_quantiles(), get_ratios(), get_t_test(), get_variance(), meta()

Examples

```
d <- as_survey_replicate(
  acs_pums_wy,
  weights = pwgtp,
  repweights = pwgtp1:pwgtp80,
```

```
  type = "successive-difference"
)

# Population size
get_totals(d)

# Total for a variable
get_totals(d, agep)

# Grouped
get_totals(d, agep, group = sex)

# One total per level of a factor
acs <- acs_pums_wy
acs$sex_f <- factor(acs$sex, levels = 1:2, labels = c("Male", "Female"))
d_f <- as_survey_replicate(
  acs,
  weights = pwgtp,
  repweights = pwgtp1:pwgtp80,
  type = "successive-difference"
)
get_totals(d_f, sex_f)
```

`get_t_test` *Design-Based Two-Sample T-Test for Survey Designs*

Description

Compares the weighted means of two groups using a design-based t-test. Follows the mathematical model of `survey::svyttest()` but uses surveycore's own variance machinery (`survey_glm()`). Supports all four survey design classes and optional subgroup analysis via `group`.

Usage

```
get_t_test(
  design,
  x,
  by,
  group = NULL,
  conf_level = 0.95,
  variance = "ci",
  na.rm = TRUE,
  min_cell_n = 30L,
  decimals = NULL,
  label_values = TRUE,
  label_vars = TRUE,
  name_style = "surveycore",
  ...,
  .id = NULL,
```

```
  .if_missing_var = NULL
)
```

Arguments

design	A survey design object: survey_taylor, survey_replicate, survey_twophase, or survey_nonprob.
x	<tidy-select> A single unquoted numeric variable name for the outcome variable. Must resolve to exactly one numeric column.
by	<tidy-select> A single unquoted variable name for the grouping variable. Must produce a model matrix with exactly 2 columns after fitting (intercept + one binary indicator). Character, integer, and logical columns are coerced to factor with a warning. Ordered factors are accepted as-is.
group	<tidy-select> Optional subgroup variable(s). When supplied, the t-test is run separately within each unique combination of group values. Combined with any grouping set by group_by(). Default NULL.
conf_level	Numeric(1). Confidence level strictly in (0, 1). Default 0.95.
variance	Character. Which uncertainty columns to include. Valid values: "se", "ci". Default "ci". Both may be requested: c("se", "ci").
na.rm	Logical(1). Accepted for API uniformity with other get_*() functions. NA rows in x or by are always excluded (the GLM requires complete cases). Default TRUE.
min_cell_n	Integer(1). Warn when either group has fewer than this many unweighted observations. Default 30L. Use 0L to suppress.
decimals	Integer(1) or NULL. Round all double output columns to this many decimal places. NULL = no rounding. Default NULL.
label_values	Logical(1). When TRUE (default), convert by and group factor codes to their value labels in the output.
label_vars	Logical(1). Accepted for API uniformity; has no visible effect because column names are fixed. Default TRUE.
name_style	Character(1). Output column naming style. "surveycore" (default) or "broom" (renames se to std.error, ci_low to conf.low, ci_high to conf.high, p_value to p.value, df to parameter). t_stat is not renamed.
...	Additional arguments forwarded to .dispatch_over_collection() when design is a survey_collection. For single-survey inputs these arguments are ignored.
.id	Character(1) or NULL. Column name used to identify each survey when design is a survey_collection. For collection inputs, NULL (the default) resolves to the collection's stored @id property. Pass a non-NULL value to override. Ignored when design is a single survey.
.if_missing_var	"error", "skip", or NULL. How to handle surveys in a collection that lack one of the requested NSE variables. For collection inputs, NULL (the default) resolves to the collection's stored @if_missing_var property. Pass a non-NULL value to override. Ignored when design is a single survey.

Value

A survey_t_test tibble (also inheriting survey_result). Columns: group columns (when active), level_a, level_b, estimate, mean_a, mean_b, n_a, n_b, se (optional), ci_low (optional), ci_high (optional), t_stat, df, p_value, stars. Use meta() to access design type, conf_level, and variable metadata.

See Also

Other analysis: SE(), clean(), get_anova(), get_corr(), get_covariance(), get_diffs(), get_effective_n(), get_freqs(), get_means(), get_pairwise(), get_quantiles(), get_ratios(), get_totals(), get_variance(), meta()

Examples

```
gss_sub <- gss_2024[gss_2024$sex %in% c(1L, 2L) & !is.na(gss_2024$age), ]
gss_sub$sex <- factor(
  gss_sub$sex,
  levels = c(1, 2),
  labels = c("Male", "Female")
)
gss_design <- as_survey(
  gss_sub,
  ids = vpsu,
  weights = wtssps,
  strata = vstrat,
  nest = TRUE
)
get_t_test(gss_design, age, by = sex)
```

---

get_variance *Design-Based Population Variance for a Survey Design*

Description

Compute the design-based estimate of the finite-population variance for one or more numeric variables in a survey design, with optional grouping, uncertainty quantification, and metadata-driven labelling. Matches survey::svyvar() numerically (Kish n/(n-1) correction) on Taylor, replicate, twophase, and nonprob designs.

Usage

```
get_variance(
  design,
  x,
  group = NULL,
  variance = "ci",
  conf_level = 0.95,
  n_weighted = FALSE,
```

```
  decimals = NULL,
  min_cell_n = 30L,
  na.rm = TRUE,
  na_handling = c("pairwise", "listwise"),
  label_values = TRUE,
  label_vars = TRUE,
  name_style = "surveycore",
  ...,
  .id = NULL,
  .if_missing_var = NULL
)
```

Arguments

design	A survey design object: survey_taylor, survey_replicate, survey_twophase, or survey_nonprob. Also accepts a survey_collection.
x	<tidy-select> One or more unquoted numeric variable names. Must resolve to at least one numeric column; non-numeric columns are rejected (no silent drop).
group	<tidy-select> Optional grouping variable(s). Combined with any grouping set by group_by(). Default NULL.
variance	NULL or a character vector of one or more of "se", "ci", "var", "cv", "moe", "deff". Controls which uncertainty columns appear in the output. Default "ci".
conf_level	Numeric scalar in (0, 1). Confidence level for intervals. Default 0.95.
n_weighted	Logical. If TRUE, add an n_weighted column with the sum of weights for non-NA, positive-weight observations in each row's estimate. Default FALSE.
decimals	Integer or NULL. If an integer, rounds all numeric output columns to this many decimal places. Default NULL (no rounding).
min_cell_n	Integer. Minimum unweighted cell count before surveycore_warning_small_cell fires. Default 30L (AAPOR guidance).
na.rm	Logical. If TRUE (default), NA values in the focal variable are excluded from the estimate and rows with NA in any grouping variable are excluded from the output. If FALSE, NA propagates to produce NaN estimates.
na_handling	"pairwise" (default) or "listwise". In multi-variable mode controls whether each focal variable uses its own complete-case set ("pairwise") or the intersection across all focal variables ("listwise"). Ignored when na.rm = FALSE.
label_values	Logical. Accepted for API consistency across get_*() functions. Used to convert grouping-variable codes to value labels. Default TRUE.
label_vars	Logical. If TRUE (default), the name column shows variable labels when available (falling back to raw names).
name_style	"surveycore" (default) or "broom". Under "broom", renames variance → estimate, se → std.error, ci_low → conf.low, ci_high → conf.high.
...	Unused. Reserved so that .id and .if_missing_var remain named-only when a survey_collection is passed as design.

`.id` Character(1) or `NULL`. Column name used to identify each survey when `design` is a `survey_collection`. For collection inputs, `NULL` (the default) resolves to the collection's stored `@id` property. Pass a non-`NULL` value to override. Ignored when `design` is a single survey.

`.if_missing_var` `"error"`, `"skip"`, or `NULL`. How to handle surveys in a collection that lack one of the requested NSE variables. For collection inputs, `NULL` (the default) resolves to the collection's stored `@if_missing_var` property. Pass a non-`NULL` value to override. Ignored when `design` is a single survey.

Details

Confidence intervals use the normal-Wald approximation on the SE of the variance estimate: `ci_low = variance - z * se`, `ci_high = variance + z * se`, where `z = qnorm((1 + conf_level) / 2)`. The bounds are **not clamped**. When the true variance is near zero with wide SE, `ci_low` may be negative. Users who want non-negative lower bounds can clamp at `0` post-hoc. This behaviour matches `survey::svyvar()`.

Under `na_handling = "pairwise"` (the default), each focal variable contributes its own per-variable complete-case count to `n`. Under `na_handling = "listwise"`, every output row shares the intersection complete-case count — rows with `NA` in any selected variable are excluded from every variable's calculation.

Value

A `survey_variance` tibble (also inheriting `survey_result`). Columns, in order:

- `[.id]` — survey identifier column, only when `design` is a `survey_collection`.
- `[group_cols...]` — group variable columns (when active), first.
- `name` — focal variable name (or its label when `label_vars = TRUE`).
- `variance` — design-based point estimate of the finite-population variance. Note: the column is always named `variance` regardless of the `variance` parameter (which controls uncertainty columns, not this column). `NaN` for degenerate cells; exact `0` for constant-in-domain variables.
- Uncertainty columns (`se`, `var`, `cv`, `ci_low`, `ci_high`, `moe`, `deff`) — only those requested via the `variance` parameter. The `var` uncertainty column is the variance of the estimated variance, distinct from the `variance` point estimate column.
- `n` — unweighted count of non-NA observations used.
- `n_weighted` — sum of weights (only when `n_weighted = TRUE`).

See Also

Other analysis: `SE()`, `clean()`, `get_anova()`, `get_corr()`, `get_covariance()`, `get_diffs()`, `get_effective_n()`, `get_freqs()`, `get_means()`, `get_pairwise()`, `get_quantiles()`, `get_ratios()`, `get_t_test()`, `get_totals()`, `meta()`

Examples

```
d <- as_survey(
  nhanes_2017,
  ids = sdmvpsu,
  weights = wtint2yr,
  strata = sdmvstra,
  nest = TRUE
)
get_variance(d, ridageyr)

# Multiple variables
get_variance(d, c(ridageyr, bpxsy1))

# With grouping
get_variance(d, ridageyr, group = riagendr)
```

`gss_2024` *GSS 2024: General Social Survey*

Description

A 27-variable extract from the 2024 General Social Survey (GSS), one of the longest-running sociological surveys in the United States (fielded annually or biennially since 1972). All 3,309 respondents from the 2024 cross-section are included.

Usage

```
gss_2024
```

Format

A data frame with 3,309 rows and 27 variables:

vpsu Variance primary sampling unit. Use as the cluster ID for variance estimation.

vstrat Variance stratum. Use as the stratification variable.

wtssps Person post-stratification weight. Standard analysis weight.

wtssnrps Person post-stratification weight adjusted for differential non-response. Preferred when non-response bias is a concern.

id Respondent ID. Unique case identifier.

year Survey year (all `2024` in this extract).

ballot Ballot form (`A`, `B`, `C`, or `D`). The GSS uses a split-ballot design; not all questions appear on every ballot. Inapplicable items are coded `-100`.

age Age in years (`89` = 89 or older).

sex Sex: `1` = male, `2` = female.

race Race: `1` = white, `2` = black, `3` = other.

hispanic Hispanic origin: 1 = not Hispanic; 2–50 = specific Hispanic origin.

educ Highest year of school completed (0–20 years).

degree Highest degree: 0 = less than HS, 1 = high school, 2 = associate, 3 = bachelor's, 4 = graduate.

income16 Total family income (26 categories from < $1,000 to $170,000+).

marital Marital status: 1 = married, 2 = widowed, 3 = divorced, 4 = separated, 5 = never married.

wrkstat Labor force status: 1 = full time, 2 = part time, 3 = temporarily not working, 4 = unemployed, 5 = retired, 6 = in school, 7 = keeping house, 8 = other.

hrs1 Hours worked last week (for employed respondents only).

adults Number of adults in household (8 = 8 or more).

partyid Party identification: 0 = strong Democrat, 3 = Independent, 6 = strong Republican, 7 = other party.

polviews Political views: 1 = extremely liberal, 7 = extremely conservative.

happy General happiness: 1 = very happy, 2 = pretty happy, 3 = not too happy.

health Self-rated health: 1 = excellent, 2 = good, 3 = fair, 4 = poor.

trust Social trust: 1 = most people can be trusted, 2 = can't be too careful, 3 = depends.

natfare Government spending on welfare: 1 = too little, 2 = about right, 3 = too much.

abany Abortion for any reason: 1 = yes, 2 = no.

attend Religious service attendance: 0 = never, 8 = several times a week.

relig Religious preference: 1 = Protestant, 2 = Catholic, 3 = Jewish, 4 = none, and others.

Details

Survey design: Stratified multi-stage cluster — use Taylor series linearization:

```
svy <- as_survey(gss_2024,
  ids     = vpsu,
  strata  = vstrat,
  weights = wtssps,      # or wtssnrps for non-response-adjusted weight
  nest    = TRUE
)
```

Missing value codes: The GSS uses a consistent system of negative integer codes for missing data across all variables:

Code	Meaning
-100	Inapplicable (question not asked of this respondent)
-99	No answer
-98	Don't know
-97	Skipped on web
-90	Refused

These codes are stored as value labels on every column (check `attr(gss_2024$happy, "labels")`). Recode them to `NA` before analysis.

Split-ballot design: The `ballot` variable indicates which question module a respondent received. Variables asked only on some ballots will have `-100` (Inapplicable) for respondents on other ballots.

Metadata: All columns carry variable labels and value labels as R attributes from the original SPSS file, automatically extracted into surveycore's metadata system when you call `as_survey()`.

- **Variable labels** (`"label"` attribute): A human-readable description of each column. Example: `attr(gss_2024$happy, "label")` returns `"GENERAL HAPPINESS"`.
- **Value labels** (`"labels"` attribute): A named numeric vector mapping each code to its meaning, including all missing-value codes. Example: `attr(gss_2024$happy, "labels")` returns entries for `Very happy`, `Pretty happy`, `Not too happy`, and the negative missing codes.

Source

NORC at the University of Chicago. General Social Survey 2024. https://gss.norc.org (free account required to download raw data; the processed `.rda` is included in the package). Prepared by `data-raw/prepare-gss-2024.R`.

Examples

```
# Variables in the dataset
names(gss_2024)

# Create survey design
svy <- as_survey(
  gss_2024,
  ids = vpsu,
  strata = vstrat,
  weights = wtssps,
  nest = TRUE
)

# Inspect variable label
attr(gss_2024$happy, "label")

# Inspect value labels (includes GSS missing-value codes)
attr(gss_2024$happy, "labels")

# Split-ballot: how many respondents per ballot form?
table(gss_2024$ballot)
```

`infer_question_prefaces`

Infer Question Prefaces from Variable Labels

Description

Scans variable labels in a survey design object or labelled data frame for groups of variables sharing a common preface (via separator or longest common prefix). For survey design objects, detected prefaces are written to `@metadata@question_prefaces`. For data frames, prefaces are written to `attr(col, "question_preface")` on each column (no metadata object exists until `as_survey()` is called). The shared text is trimmed from each variable label, leaving only the unique suffix.

Usage

```
infer_question_prefaces(
  x,
  sep = c(" - ", "- ", " – ", ": ", " | "),
  min_vars = 2L,
  lcp_min = 20L,
  overwrite = FALSE,
  verbose = TRUE
)
```

Arguments

`x`	A survey design object (`survey_taylor`, `survey_replicate`, etc.) **or** a data frame with haven-style `"label"` attributes.
`sep`	Character vector of literal separator strings to try, in priority order. Default: `c(" - ", "- ", " \u2013 ", ": ", " | ")`.
`min_vars`	Minimum number of variables that must share a candidate preface to trigger extraction. Default `2L`.
`lcp_min`	Minimum character length (after trimming to a word boundary) for an LCP-derived preface to be accepted. Default `20L`.
`overwrite`	If `FALSE` (default), variables that already have a `question_preface` are skipped and a warning is emitted. Set `TRUE` to replace existing prefaces without warning.
`verbose`	If `TRUE` (default), emits a cli summary for each detected group.

Details

Detection algorithm (two passes):

1. **Separator pass** — for each separator in `sep` (tried in order):
   - Variables whose label contains the separator are grouped by their candidate preface (text before the *first* occurrence of the separator, trimmed).
   - Any group with $\geq$ `min_vars` members is recorded; those variables are excluded from all subsequent passes.
2. **LCP pass** — for remaining labelled variables (≥ 2):
   - The character-level longest common prefix (LCP) of all remaining labels is computed and trimmed to the last word boundary.
   - If the trimmed LCP is $\geq$ `lcp_min` characters, the group is recorded.

Apply step:

- Variables with an existing `question_preface` are skipped when `overwrite = FALSE` (default); a warning is emitted listing the count of skipped variables.
- Variables whose unique suffix would be empty after trimming are always skipped with a per-variable warning.

Data frame integration: When called on a data frame, the detected preface is written to `attr(col, "question_preface")`. Passing the result to `as_survey()` automatically picks up both the trimmed label and the preface via the internal haven metadata extraction step.

Value

The modified `x`, invisibly.

See Also

Other metadata: `classify_question_type()`, `extract_data_name()`, `extract_dataset_metadata()`, `extract_field_dates()`, `extract_field_period()`, `extract_higher_is()`, `extract_metadata()`, `extract_missing_codes()`, `extract_question_preface()`, `extract_reverse_coded()`, `extract_sata()`, `extract_survey_name()`, `extract_universe()`, `extract_val_labels()`, `extract_var_extra()`, `extract_var_label()`, `extract_var_note()`, `extract_vendor()`, `set_data_name()`, `set_dataset_metadata()`, `set_field_dates()`, `set_field_period()`, `set_higher_is()`, `set_missing_codes()`, `set_question_preface()`, `set_reverse_coded()`, `set_sata()`, `set_survey_name()`, `set_universe()`, `set_val_labels()`, `set_var_extra()`, `set_var_label()`, `set_var_note()`, `set_vendor()`, `survey_metadata()`, `survey_weighting_history()`

Examples

```
# Data frame with haven-style labels (Qualtrics / SPSS export pattern)
df <- data.frame(
  discrim_a = 1:5,
  discrim_b = 2:6,
  discrim_c = 3:7
)
attr(df$discrim_a, "label") <-
  "Please rate discrimination - Evangelical Christians"
attr(df$discrim_b, "label") <-
  "Please rate discrimination - Muslims"
attr(df$discrim_c, "label") <-
  "Please rate discrimination - Jews"

df <- infer_question_prefaces(df, verbose = FALSE)
attr(df$discrim_a, "label") # "Evangelical Christians"
attr(df$discrim_a, "question_preface") # "Please rate discrimination"
```

meta *Extract Metadata from a Survey Result*

Description

Retrieves the structured metadata list attached to a survey result object returned by any `get_*()` analysis function.

Usage

```
meta(x, ...)

## S3 method for class 'survey_result'
meta(x, ...)
```

Arguments

`x`	A `survey_result` object returned by any `get_*()` function.
`...`	Currently unused. Reserved for future extensions.

Details

This is the **only** supported way to access result metadata — do not use `attr(result, ".meta")` directly.

Value

A named list. Common fields present on every result:

`design_type` Character(1). Design class: `"taylor"`, `"replicate"`, `"twophase"`, or `"nonprob"`. SRS designs are represented as `survey_taylor` (no IDs/strata) and report `"taylor"`.

`conf_level` Numeric(1). Confidence level used (e.g. `0.95`).

`call` Language. Matched call to the `get_*()` function.

`n_respondents` Integer(1). Total rows in the design, regardless of groups, domain status, or weights.

`group` Named list. One entry per grouping variable; empty list (`list()`) when no groups are active. Each entry is a named list with: `variable_label` (character or `NULL`), `question_preface` (character or `NULL`), `value_labels` (named vector or `NULL`).

`x` Named list. One entry per focal variable. Length 1 for single-x functions (`get_means`, `get_totals`, `get_quantiles`); length N for multi-x functions (`get_freqs`, `get_corr`). Each entry has the same sub-structure as `group` entries. `NULL` for `get_totals()` when called without an `x` argument.

Function-specific additional fields:

`probs` (`get_quantiles` only) Numeric vector of quantile probabilities.

method (get_corr only) Character(1) correlation method.

numerator, denominator (get_ratios only) Flat named lists with keys name, variable_label, question_preface, value_labels.

See Also

Other analysis: SE(), clean(), get_anova(), get_corr(), get_covariance(), get_diffs(), get_effective_n(), get_freqs(), get_means(), get_pairwise(), get_quantiles(), get_ratios(), get_t_test(), get_totals(), get_variance()

Examples

```
# Construct a minimal survey_result to illustrate meta():
result <- structure(
  tibble::tibble(mean = 42.0, se = 1.5, n = 100L),
  .meta = list(
    design_type = "taylor",
    conf_level = 0.95,
    call = quote(get_means(d, x)),
    n_respondents = 100L,
    group = list(),
    x = list(
      x = list(
        variable_label = NULL,
        question_preface = NULL,
        value_labels = NULL
      )
    )
  ),
  class = c("survey_means", "survey_result", "tbl_df", "tbl", "data.frame")
)
meta(result)$design_type # "taylor"
meta(result)$n_respondents # 100L
meta(result)$conf_level # 0.95
```

---

nhanes_2017 *NHANES 2017-2018: Demographics and Blood Pressure*

Description

A merged dataset from the National Health and Nutrition Examination Survey (NHANES) 2017-2018 cycle, combining demographic characteristics with blood pressure measurements. Covers all 9,254 sampled participants; blood pressure variables are NA for the 550 interview-only participants (ridstatr == 1).

Usage

```
nhanes_2017
```

Format

A data frame with 9,254 rows and 14 variables:

seqn Respondent sequence number (unique identifier, join key).

sdmvpsu Masked variance pseudo-PSU. Use as the cluster ID for variance estimation. See Details.

sdmvstra Masked variance pseudo-stratum. Use as the stratification variable for variance estimation. See Details.

wtmec2yr Full-sample 2-year MEC examination weight. Use for any analysis involving examination measurements (e.g., blood pressure).

wtint2yr Full-sample 2-year interview weight. Use for analyses based on interview data only.

ridstatr Interview/examination status: `1` = interview only, `2` = both interview and MEC examination.

riagendr Gender: `1` = male, `2` = female.

ridageyr Age in years at screening, top-coded at 80.

ridreth3 Race/Hispanic origin (6 categories): `1` = Mexican American, `2` = Other Hispanic, `3` = Non-Hispanic White, `4` = Non-Hispanic Black, `6` = Non-Hispanic Asian, `7` = Other/Multiracial.

indfmpir Ratio of family income to the federal poverty level (continuous, 0–5; values >5 are top-coded at 5).

dmdeduc2 Education level for adults 20+: `1` = Less than 9th grade, `2` = 9th–11th grade, `3` = High school graduate/GED, `4` = Some college/AA, `5` = College graduate or above.

bpxsy1 Systolic blood pressure, 1st reading (mm Hg). `NA` if not examined.

bpxdi1 Diastolic blood pressure, 1st reading (mm Hg). `NA` if not examined.

bpxpls 60-second pulse rate (beats per minute). `NA` if not examined.

Details

Survey design: Taylor series linearization. When creating a survey design object, use `sdmvpsu` as the cluster ID, `sdmvstra` as the stratum, and `wtmec2yr` as the weight for examination-based analyses:

```
svy <- as_survey(nhanes_2017,
  ids     = sdmvpsu,
  strata  = sdmvstra,
  weights = wtmec2yr
)
```

Use `wtint2yr` instead of `wtmec2yr` for interview-only variables (e.g., income, education).

Metadata: All columns carry a variable label as an R attribute, automatically extracted into surveycore's metadata system when you call `as_survey()`. No column carries value labels.

- **Variable labels** (`"label"` attribute): A human-readable description of each column. Example: `attr(nhanes_2017$riagendr, "label")` returns `"Gender"`.

- **Value labels** (`"labels"` attribute): None. NHANES publishes the data as SAS XPORT files. That format stores a label for each variable, but it does not store labels for the codes inside a variable. The code meanings are in the Format section above and in the NHANES codebooks. Use `set_val_labels()` to attach them if you need them.

Source files: DEMO_J.xpt (demographics) merged with BPX_J.xpt (blood pressure) on `seqn`. Prepared by `data-raw/download-nhanes.R`.

Source

National Center for Health Statistics, CDC. NHANES 2017-2018 Continuous Survey. https://www.cdc.gov/nchs/nhanes/

Examples

```
# All 9,254 participants (interview + exam)
head(nhanes_2017)

# Restrict to exam participants for blood pressure analysis
exam_only <- nhanes_2017[nhanes_2017$ridstatr == 2, ]

# Inspect variable label
attr(nhanes_2017$riagendr, "label")

# Inspect variable label for race/ethnicity
attr(nhanes_2017$ridreth3, "label")
```

---

ns_wave1 *Nationscape Wave 1: July 18, 2019*

Description

The first weekly wave of the Democracy Fund + UCLA Nationscape survey, fielded July 18–24, 2019. Approximately 6,250 completed online interviews drawn from the Lucid respondent exchange platform using a non-probability quota design, with raking weights calibrated to ACS demographic targets and 2016 presidential vote choice.

Usage

```
ns_wave1
```

Format

A data frame with approximately 6,250 rows and 171 variables (170 survey variables plus `wave_id` added by the prepare script).

response_id Unique respondent ID (integer).

start_date Interview date (character, `"YYYY-MM-DD"` format).

wave_id Wave identifier: `"ns20190718"` for all rows in this dataset.

weight Raking weight calibrated to ACS demographic targets and 2016 presidential vote choice. Use for all population-level estimates.

right_track Country direction: 1 = Right direction, 2 = Wrong track, 3 = Not sure.

economy_better Economy outlook: 1 = Better, 2 = Worse, 3 = Same, 4 = Not sure.

interest Political interest (4-pt): 1 = Very interested, 4 = Not at all interested.

registration Voter registration: 1 = Registered, 2 = Not registered, 3 = Not eligible.

pres_approval Trump presidential approval: 1 = Strongly approve, 2 = Somewhat approve, 3 = Somewhat disapprove, 4 = Strongly disapprove.

vote_intention 2020 vote intention: 1 = Trump, 2 = Democratic candidate, 3 = Other, 4 = Don't plan to vote, 5 = Not sure.

vote_2016 2016 presidential vote. See labels.

vote_2016_other_text Write-in for `vote_2016` "other" choice.

consider_trump Would consider voting for Trump: 1 = Yes, 2 = No.

not_trump Reason for not considering Trump (open text).

primary_party Primary vote party: 1 = Democratic, 2 = Republican, 3 = Other.

dem_vote_intent Democratic primary vote intention. See labels.

dem_vote_intent_TEXT Write-in for `dem_vote_intent` "other".

rank_dems_1 Top-ranked Democratic presidential candidate. See labels.

rank_dems_2 Second-ranked Democratic candidate. See labels.

rank_dems_3 Third-ranked Democratic candidate. See labels.

replace_trump Wants non-Trump Republican nominee: 1 = Yes, 2 = No, 3 = Not sure.

house_intent U.S. House vote intention: 1 = Democrat, 2 = Republican, 3 = Other, 4 = Won't vote, 5 = Not sure.

senate_intent U.S. Senate vote intention. Same codes as `house_intent`.

governor_intent Governor vote intention. Same codes as `house_intent`.

news_sources_facebook Used social media for political news in past week: 1 = Selected, 2 = Not selected. See `"question_preface"` attribute for shared question stem. Same coding for all `news_sources_*` variables.

news_sources_cnn Used CNN for political news.

news_sources_msnbc Used MSNBC for political news.

news_sources_fox Used Fox News for political news.

news_sources_network Used network news (ABC/CBS/NBC/PBS).

news_sources_localtv Used local TV news.

news_sources_telemundo Used Telemundo or Univision.

news_sources_npr Used NPR.

news_sources_amtalk Used AM talk radio.

news_sources_new_york_times Used a national newspaper.

news_sources_local_newspaper Used a local newspaper.

news_sources_other Used another news source: 1 = Selected, 2 = Not selected.

news_sources_other_TEXT Write-in for news_sources_other.

group_favorability_whites Favorability toward Whites: 1 = Very favorable, 2 = Somewhat favorable, 3 = Somewhat unfavorable, 4 = Very unfavorable, 5 = Not sure. Same coding for all group_favorability_* variables.

group_favorability_blacks Favorability toward Blacks.

group_favorability_latinos Favorability toward Latinos.

group_favorability_asians Favorability toward Asians.

group_favorability_christians Favorability toward Christians.

group_favorability_socialists Favorability toward Socialists.

group_favorability_muslims Favorability toward Muslims.

group_favorability_labor_unions Favorability toward labor unions.

group_favorability_the_police Favorability toward the police.

group_favorability_undocumented Favorability toward undocumented immigrants.

group_favorability_lgbt Favorability toward gays and lesbians.

group_favorability_republicans Favorability toward Republicans.

group_favorability_democrats Favorability toward Democrats.

cand_favorability_trump Favorability toward Donald Trump. Same 5-point scale as group_favorability_* variables.

cand_favorability_obama Favorability toward Barack Obama.

cand_favorability_cortez Favorability toward Alexandria Ocasio-Cortez.

cand_favorability_biden Favorability toward Joe Biden.

cand_favorability_harris Favorability toward Kamala Harris.

cand_favorability_buttigieg Favorability toward Pete Buttigieg.

cand_favorability_warren Favorability toward Elizabeth Warren.

cand_favorability_sanders Favorability toward Bernie Sanders.

cand_favorability_pence Favorability toward Mike Pence.

trump_biden Trump vs. Biden head-to-head: 1 = Trump, 2 = Biden, 3 = Not sure. Same coding for all trump_* matchup variables.

trump_sanders Trump vs. Sanders.

trump_harris Trump vs. Harris.

trump_warren Trump vs. Warren.

trump_buttigieg Trump vs. Buttigieg.

trump_booker Trump vs. Cory Booker.

trump_castro Trump vs. Julian Castro.

trump_gabbard Trump vs. Tulsi Gabbard.

trump_gillibrand Trump vs. Kirsten Gillibrand.

trump_orourke Trump vs. Beto O'Rourke.

pence_biden Pence vs. Biden head-to-head: 1 = Pence, 2 = Biden, 3 = Not sure. Same coding for all pence_* matchup variables.

pence_buttigieg Pence vs. Buttigieg.

pence_harris Pence vs. Harris.

pence_sanders Pence vs. Sanders.

pence_warren Pence vs. Warren.

cand_truth_donald_trump Whether Donald Trump cares about telling the truth: 1 = Yes, 2 = No, 3 = Not sure. Same coding for all `cand_truth_*` variables.

cand_truth_elizabeth_warren Whether Elizabeth Warren cares about the truth.

cand_truth_joe_biden Whether Joe Biden cares about the truth.

cand_truth_bernie_sanders Whether Bernie Sanders cares about the truth.

cand_truth_pete_buttigieg Whether Pete Buttigieg cares about the truth.

cand_truth_kamala_harris Whether Kamala Harris cares about the truth.

cand_facts_donald_trump Whether Donald Trump relies on facts vs. hunches: 1 = Facts and evidence, 2 = Hunches, 3 = Not sure. Same coding for all `cand_facts_*` variables.

cand_facts_elizabeth_warren Whether Elizabeth Warren relies on facts.

cand_facts_joe_biden Whether Joe Biden relies on facts.

cand_facts_bernie_sanders Whether Bernie Sanders relies on facts.

cand_facts_pete_buttigieg Whether Pete Buttigieg relies on facts.

cand_facts_kamala_harris Whether Kamala Harris relies on facts.

racial_attitudes_tryhard Agree/disagree: minorities should work their way up without special favors. 1 = Strongly agree, 2 = Agree, 3 = Neither, 4 = Disagree, 5 = Strongly disagree. Same scale for all `racial_attitudes_*` and `gender_attitudes_*` variables.

racial_attitudes_generations Agree/disagree: generations of slavery make it difficult for Blacks to work out of the lower class.

racial_attitudes_marry Agree/disagree: I prefer close relatives marry someone from the same race.

racial_attitudes_date Agree/disagree: it's alright for Blacks and Whites to date.

gender_attitudes_maleboss Agree/disagree: more comfortable with a male boss than female boss.

gender_attitudes_logical Agree/disagree: women are just as capable of thinking logically as men.

gender_attitudes_opportunity Agree/disagree: increased opportunities for women have improved quality of life.

gender_attitudes_complain Agree/disagree: women who complain about harassment cause more problems than they solve.

discrimination_blacks Perceived discrimination against Blacks: 1 = A great deal, 2 = A lot, 3 = A little, 4 = None at all, 5 = Not sure. Same scale for all `discrimination_*` variables.

discrimination_whites Perceived discrimination against Whites.

discrimination_muslims Perceived discrimination against Muslims.

discrimination_christians Perceived discrimination against Christians.

discrimination_women Perceived discrimination against Women.

discrimination_men Perceived discrimination against Men.

sen_knowledge U.S. Senate knowledge question. See labels.

sc_knowledge U.S. Supreme Court knowledge question. See labels.

pid3 3-category party ID: 1 = Democrat, 2 = Republican, 3 = Independent, 4 = Something else.

pid7_legacy 7-point party ID (legacy coding). See labels.

strength_democrat Strength of Democratic ID (conditional on `pid3 == 1`). See labels.

strength_republican Strength of Republican ID (conditional on `pid3 == 2`). See labels.

lean_independent Partisan lean of Independents (conditional on `pid3 == 3`). See labels.

ideo5 5-point ideological self-placement: 1 = Very liberal, 5 = Very conservative.

employment Employment status (selected choice). See labels.

employment_other_text Write-in for `employment` "other".

foreign_born Born outside the U.S.: 1 = Yes, 2 = No.

language Primary language at home. See labels.

religion Religious affiliation (selected choice). See labels.

religion_other_text Write-in for `religion` "other".

is_evangelical Born-again or evangelical Christian: 1 = Yes, 2 = No.

orientation_group Sexual orientation. See labels.

in_union Labor union membership: 1 = Yes, 2 = No, 3 = Non-union household, 4 = Not sure.

household_gun_owner Household gun ownership: 1 = Yes, 2 = No, 3 = Not sure.

wall Support building a wall on the southern U.S. border: 1 = Strongly support, 2 = Somewhat support, 3 = Somewhat oppose, 4 = Strongly oppose, 5 = Not sure. Same scale for all policy items through `limit_magazines`. See `"question_preface"` attribute on each variable for the exact shared question stem.

cap_carbon Support capping carbon emissions.

environment Support large-scale government investment in environmental technology.

guns_bg Support requiring background checks for all gun purchases.

mctaxes Support cutting taxes for families making < $100K/year.

estate_tax Support eliminating the estate tax.

raise_upper_tax Support raising taxes on families making > $600K.

college Support ensuring all students can graduate from state colleges debt-free.

abortion_waiting Support requiring a waiting period and ultrasound before an abortion.

abortion_never Support never permitting abortion.

abortion_conditions Support permitting abortion in cases other than rape/incest/life at risk.

late_term_abortion Support permitting late-term abortion.

abortion_insurance Support allowing employers to decline abortion coverage.

guaranteed_jobs Support guaranteeing jobs for all Americans.

green_new_deal Support enacting a Green New Deal.

gun_registry Support creating a public registry of gun ownership.

immigration_separation Support separating children from parents prosecuted for illegal border crossing.

immigration_system Support shifting to a merit-based immigration system.

immigration_wire Support requiring proof of citizenship to wire money internationally.

impeach_trump Support impeaching President Trump.

israel Support withdrawing military support for Israel.

marijuana Support legalizing marijuana.

maternityleave Support requiring 12 weeks of paid maternity leave.

medicare_for_all Support Medicare-for-All.

military_size Support reducing the size of the U.S. military.

minwage Support raising the minimum wage to $15/hour.

muslimban Support banning people from predominantly Muslim countries.

oil_and_gas Support removing barriers to domestic oil and gas drilling.

reparations Support granting reparations to descendants of slaves.

right_to_work Support allowing people to work in unionized workplaces without paying union dues.

ten_commandments Support displaying the Ten Commandments in public schools and courthouses.

trade Support limiting trade with other countries.

trans_military Support allowing transgender people to serve in the military.

uctaxes2 Support raising taxes on families making > $250K.

vouchers Support providing tax-funded vouchers for private or religious schools.

gov_insurance Support providing government-run health insurance to all Americans.

public_option Support providing the option to purchase government-run insurance.

health_subsidies Support subsidizing health insurance for lower income people not on Medicaid.

path_to_citizenship Support creating a path to citizenship for all undocumented immigrants.

dreamers Support a path to citizenship for DREAMers.

deportation Support deporting all undocumented immigrants.

ban_guns Support banning all guns.

ban_assault_rifles Support banning assault rifles.

limit_magazines Support limiting gun magazines to 10 bullets.

age Respondent age in years.

gender Gender: 1 = Male, 2 = Female, 3 = Other.

census_region Census region: 1 = Northeast, 2 = Midwest, 3 = South, 4 = West.

hispanic Hispanic or Latino origin: 1 = Yes, 2 = No.

race_ethnicity Race/ethnicity (6 categories). See labels.

household_income Household income (7 brackets). See labels.

education Educational attainment (6 categories). See labels.

state U.S. state of residence (2-letter abbreviation).

congress_district Congressional district.

Details

This dataset is the first of 77 weekly waves collected from July 2019 through January 2021. The full survey ran in three phases:

Phase	Weeks	Dates	Approx. N
Phase 1	1–24	Jul 18, 2019 – Dec 26, 2019	150,000
Phase 2	25–50	Jan 2, 2020 – Jun 25, 2020	162,500
Phase 3	51–77	Jul 2, 2020 – Jan 12, 2021	168,750

Only Wave 1 is bundled in the package because 77 waves × ~6,250 rows would be prohibitively large. To obtain the full dataset by phase, use the prepare scripts in `data-raw/` (see the Source section).

Survey design: The Nationscape is a calibrated non-probability sample (quota design with raking weights). Use `as_survey_nonprob()` — it is designed specifically for this use case and will gain bootstrap re-calibration variance in Phase 2.5:

```
svy <- as_survey_nonprob(ns_wave1, weights = weight)
```

Metadata: All substantive columns carry variable labels (`"label"` attribute) set during data preparation. Battery items additionally carry a `"question_preface"` attribute with the shared question stem. Value labels (`"labels"` attribute) are present for all coded response items.

Battery structure: Most multi-item question groups follow a `{battery}_{item}` naming convention. All items within a battery share an identical `"question_preface"` attribute:

Battery prefix	Preface summary	N items
`news_sources_*`	News sources used in past week	13
`group_favorability_*`	Favorability toward named groups	13
`cand_favorability_*`	Favorability toward named candidates	9
`trump_*`	Trump head-to-head matchups	10
`pence_*`	Pence head-to-head matchups	5
`cand_truth_*`	Whether each candidate tells the truth	6
`cand_facts_*`	Whether each candidate relies on facts	6
`racial_attitudes_*`	Agree/disagree racial attitude items	4
`gender_attitudes_*`	Agree/disagree gender attitude items	4
`discrimination_*`	Perceived discrimination by group	6

Three policy batteries share the same Agree/Disagree/Neither scale: `wall`, `cap_carbon`, `environment`, `guns_bg`, `mctaxes`, `estate_tax`, `raise_upper_tax`, `college`, `abortion_waiting`, `abortion_never`, `abortion_conditions`, `late_term_abortion`, `abortion_insurance`, `guaranteed_jobs`, `green_new_deal`, `gun_registry`, `immigration_separation`, `immigration_system`, `immigration_wire`, `impeach_trump`, `israel`, `marijuana`, `maternityleave`, `medicare_for_all`, `military_size`, `minwage`, `muslimban`, `oil_and_gas`, `reparations`, `right_to_work`, `ten_commandments`, `trade`, `trans_military`, `uctaxes2`, `vouchers`, `gov_insurance`, `public_option`, `health_subsidies`, `path_to_citizenship`, `dreamers`, `deportation`, `ban_guns`, `ban_assault_rifles`, `limit_magazines`.

Source

Democracy Fund Voter Study Group / UCLA. Nationscape Data Set, version December 2021. https://www.voterstudygroup.org/data/nationscape (free download; academic research use). Prepared by `data-raw/prepare-nationscape-phase1.R`.

For full methodology, see the Nationscape User Guide and the Representative Assessment report in `data-raw/nationscape/Nationscape-User-Guide-2021Dec.pdf`.

References

Tausanovitch, Chris and Lynn Vavreck. 2021. Democracy Fund + UCLA Nationscape, July 18–24, 2019 — Wave 1 (version 20210301). Retrieved from voterstudygroup.org/data/nationscape.

Rivers, Douglas and Delia Bailey. 2009. "Inference from matched samples in the 2008 U.S. national elections." Proceedings of the Joint Statistical Meetings, Social Statistics Section.

Examples

```
# Design variables
head(ns_wave1[, c("response_id", "weight", "age", "gender")])

# Inspect a battery item's metadata
attr(ns_wave1$group_favorability_blacks, "label")
attr(ns_wave1$group_favorability_blacks, "question_preface")
attr(ns_wave1$news_sources_cnn, "labels")

# Create a calibrated survey design (correct approach for raked
# non-prob samples)
svy <- as_survey_nonprob(ns_wave1, weights = weight)
get_freqs(svy, pres_approval)

# Party identification distribution
table(ns_wave1$pid3)
```

---

pew_jewish_2020 *Pew Jewish Americans 2020*

Description

The extended survey dataset from Pew Research Center's 2019-2020 Survey of U.S. Jews, fielded November 19, 2019 – June 3, 2020 (n = 5,881). Respondents were drawn from a national, stratified random sample of residential mailing addresses with oversampling of households likely to contain Jewish respondents. The dataset carries 100 jackknife replicate weights alongside the main weight.

Usage

```
pew_jewish_2020
```

Format

A data frame with 5,881 rows and 130 variables. Variables `extweight1`–`extweight100` are jackknife replicate weights; the remaining 30 variables are:

extweight Full-sample base weight. Use for all estimates.

extweight1 Jackknife replicate weight 1 of 100.

extweight2 Jackknife replicate weight 2 of 100.

extweight3 Jackknife replicate weight 3 of 100.

extweight4 Jackknife replicate weight 4 of 100.

extweight5 Jackknife replicate weight 5 of 100.

extweight6 Jackknife replicate weight 6 of 100.

extweight7 Jackknife replicate weight 7 of 100.

extweight8 Jackknife replicate weight 8 of 100.

extweight9 Jackknife replicate weight 9 of 100.

extweight10 Jackknife replicate weight 10 of 100.

extweight11 Jackknife replicate weight 11 of 100.

extweight12 Jackknife replicate weight 12 of 100.

extweight13 Jackknife replicate weight 13 of 100.

extweight14 Jackknife replicate weight 14 of 100.

extweight15 Jackknife replicate weight 15 of 100.

extweight16 Jackknife replicate weight 16 of 100.

extweight17 Jackknife replicate weight 17 of 100.

extweight18 Jackknife replicate weight 18 of 100.

extweight19 Jackknife replicate weight 19 of 100.

extweight20 Jackknife replicate weight 20 of 100.

extweight21 Jackknife replicate weight 21 of 100.

extweight22 Jackknife replicate weight 22 of 100.

extweight23 Jackknife replicate weight 23 of 100.

extweight24 Jackknife replicate weight 24 of 100.

extweight25 Jackknife replicate weight 25 of 100.

extweight26 Jackknife replicate weight 26 of 100.

extweight27 Jackknife replicate weight 27 of 100.

extweight28 Jackknife replicate weight 28 of 100.

extweight29 Jackknife replicate weight 29 of 100.

extweight30 Jackknife replicate weight 30 of 100.

extweight31 Jackknife replicate weight 31 of 100.

extweight32 Jackknife replicate weight 32 of 100.

extweight33 Jackknife replicate weight 33 of 100.

extweight34 Jackknife replicate weight 34 of 100.
extweight35 Jackknife replicate weight 35 of 100.
extweight36 Jackknife replicate weight 36 of 100.
extweight37 Jackknife replicate weight 37 of 100.
extweight38 Jackknife replicate weight 38 of 100.
extweight39 Jackknife replicate weight 39 of 100.
extweight40 Jackknife replicate weight 40 of 100.
extweight41 Jackknife replicate weight 41 of 100.
extweight42 Jackknife replicate weight 42 of 100.
extweight43 Jackknife replicate weight 43 of 100.
extweight44 Jackknife replicate weight 44 of 100.
extweight45 Jackknife replicate weight 45 of 100.
extweight46 Jackknife replicate weight 46 of 100.
extweight47 Jackknife replicate weight 47 of 100.
extweight48 Jackknife replicate weight 48 of 100.
extweight49 Jackknife replicate weight 49 of 100.
extweight50 Jackknife replicate weight 50 of 100.
extweight51 Jackknife replicate weight 51 of 100.
extweight52 Jackknife replicate weight 52 of 100.
extweight53 Jackknife replicate weight 53 of 100.
extweight54 Jackknife replicate weight 54 of 100.
extweight55 Jackknife replicate weight 55 of 100.
extweight56 Jackknife replicate weight 56 of 100.
extweight57 Jackknife replicate weight 57 of 100.
extweight58 Jackknife replicate weight 58 of 100.
extweight59 Jackknife replicate weight 59 of 100.
extweight60 Jackknife replicate weight 60 of 100.
extweight61 Jackknife replicate weight 61 of 100.
extweight62 Jackknife replicate weight 62 of 100.
extweight63 Jackknife replicate weight 63 of 100.
extweight64 Jackknife replicate weight 64 of 100.
extweight65 Jackknife replicate weight 65 of 100.
extweight66 Jackknife replicate weight 66 of 100.
extweight67 Jackknife replicate weight 67 of 100.
extweight68 Jackknife replicate weight 68 of 100.
extweight69 Jackknife replicate weight 69 of 100.
extweight70 Jackknife replicate weight 70 of 100.

extweight71 Jackknife replicate weight 71 of 100.

extweight72 Jackknife replicate weight 72 of 100.

extweight73 Jackknife replicate weight 73 of 100.

extweight74 Jackknife replicate weight 74 of 100.

extweight75 Jackknife replicate weight 75 of 100.

extweight76 Jackknife replicate weight 76 of 100.

extweight77 Jackknife replicate weight 77 of 100.

extweight78 Jackknife replicate weight 78 of 100.

extweight79 Jackknife replicate weight 79 of 100.

extweight80 Jackknife replicate weight 80 of 100.

extweight81 Jackknife replicate weight 81 of 100.

extweight82 Jackknife replicate weight 82 of 100.

extweight83 Jackknife replicate weight 83 of 100.

extweight84 Jackknife replicate weight 84 of 100.

extweight85 Jackknife replicate weight 85 of 100.

extweight86 Jackknife replicate weight 86 of 100.

extweight87 Jackknife replicate weight 87 of 100.

extweight88 Jackknife replicate weight 88 of 100.

extweight89 Jackknife replicate weight 89 of 100.

extweight90 Jackknife replicate weight 90 of 100.

extweight91 Jackknife replicate weight 91 of 100.

extweight92 Jackknife replicate weight 92 of 100.

extweight93 Jackknife replicate weight 93 of 100.

extweight94 Jackknife replicate weight 94 of 100.

extweight95 Jackknife replicate weight 95 of 100.

extweight96 Jackknife replicate weight 96 of 100.

extweight97 Jackknife replicate weight 97 of 100.

extweight98 Jackknife replicate weight 98 of 100.

extweight99 Jackknife replicate weight 99 of 100.

extweight100 Jackknife replicate weight 100 of 100.

qkey Unique respondent identifier.

jewishcat Jewish identity category: 1 = Jews By Religion, 2 = Jews Of No Religion, 3 = Jewish Background, 4 = Jewish Affinity, 5 = Respondent Not Jewish In Any Way.

finalmode Collection mode: 1 = Screener And Extended Survey Via Cawi, 2 = Screener And Extended Survey Via Teleform, 3 = Screener Via Cawi, Extended Survey Via Teleform.

region Census region: 1 = Northeast, 2 = Midwest, 3 = South, 4 = West.

sexask Sex: 1 = Male, 2 = Female, 99 = Not Answered.

age4cat Age: 1 = 18-29, 2 = 30-49, 3 = 50-64, 4 = 65+; 999 = No Answer.

educ4cat Education: 1 = High School Or Less, 2 = Some College, 3 = College Graduate, 4 = Postgrad Degree; 99 = No Answer.

religmod Current religion (24 categories including Jewish subgroups and combinations).

hisp Hispanic origin: 1 = Yes, 2 = No, 99 = Not Answered.

racecmb Race (5 categories).

racethn Race-ethnicity (4 categories).

presapp Presidential approval (Trump): 1 = Strongly Approve, 2 = Somewhat Approve, 3 = Somewhat Disapprove, 4 = Strongly Disapprove, 99 = Not Answered.

track Right track/wrong track: 1 = Generally Headed In The Right Direction, 2 = Off On The Wrong Track, 99 = Not Answered.

satisfpersmod Personal life satisfaction: 1 = Excellent, 2 = Good, 3 = Only Fair, 4 = Poor, 99 = Not Answered.

localrating Community as a place to live: 1 = Excellent, 2 = Good, 3 = Only Fair, 4 = Poor, 99 = Not Answered.

relconsider_a Jewish. Battery 1: religious identity (select-all-that-apply). See Details for question text.

relconsider_b Catholic. Battery 1: religious identity.

relconsider_c Mormon. Battery 1: religious identity.

relconsider_d Muslim. Battery 1: religious identity.

relraised_a Jewish. Battery 2: religious background (select-all-that-apply). See Details for question text.

relraised_b Catholic. Battery 2: religious background.

relraised_c Mormon. Battery 2: religious background.

relraised_d Muslim. Battery 2: religious background.

discrim_a Evangelical Christians. Battery 3: discrimination perceptions (rating scale). See Details for question text.

discrim_b Muslims. Battery 3: discrimination perceptions.

discrim_c Jews. Battery 3: discrimination perceptions.

discrim_d Blacks. Battery 3: discrimination perceptions.

discrim_e Hispanics. Battery 3: discrimination perceptions.

discrim_f Gays and lesbians. Battery 3: discrimination perceptions.

Details

Survey design: Jackknife replication — use `as_survey_replicate()` with all 100 replicate weights:

```
svy <- as_survey_replicate(
  pew_jewish_2020,
  weights    = extweight,
  repweights = extweight1:extweight100,
  type       = "JK1"
)
```

Jewish identity classification: The `jewishcat` variable classifies respondents into five mutually exclusive categories used in the published Pew report. Use `jewishcat` rather than constructing your own classification from the raw religion variables.

Battery question stems:

- **Battery 1** (`relconsider_a`–`relconsider_d`): `"ASIDE from religion, do you consider yourself to be any of the following in any way (for example ethnically, culturally or because of your family's background)?"` Values: `1` = Yes, Consider Myself This, `2` = No, Do Not Consider Myself This, `99` = Refused.
- **Battery 2** (`relraised_a`–`relraised_d`): `"Please indicate whether you were raised in any of the following traditions or had a parent from any of the following backgrounds."` Values: `1` = Yes, Was Raised In This Tradition Or Had A Parent From This Background, `2` = No, Was Not Raised In This Tradition And Did Not Have A Parent From This Background, `99` = Refused.
- **Battery 3** (`discrim_a`–`discrim_f`): `"Please tell us how much discrimination there is against each of these groups in our society today."` Values: `1` = A Lot, `2` = Some, `3` = Not Much, `4` = None At All, `99` = Not Answered.

Metadata: All columns carry variable labels and value labels as R attributes from the original Stata file. The three battery variable groups additionally carry a `"question_preface"` attribute with the shared question stem. All three attribute types are automatically extracted into surveycore's metadata system when you call `as_survey_replicate()`.

- **Variable labels** (`"label"` attribute): A human-readable description of each column — for battery items this is the unique item text (e.g., `"Jewish"`). Example: `attr(pew_jewish_2020$relconsider_a, "label")` returns `"Jewish"`.
- **Value labels** (`"labels"` attribute): A named numeric vector mapping each code to its meaning. Example: `attr(pew_jewish_2020$relconsider_a, "labels")` returns `c("Yes, Consider Myself This" = 1, "No, Do Not Consider Myself This" = 2, Refused = 99)`.
- **Question preface** (`"question_preface"` attribute): The shared question stem for each battery group. Example: `attr(pew_jewish_2020$discrim_a, "question_preface")` returns `"Please tell us how much discrimination there is against each of these groups in our society today."`.

Source

Pew Research Center. Jewish Americans in 2020 (Extended Dataset). https://www.pewresearch.org/datasets/ (free account required to download raw data; the processed `.rda` is included in the package). Prepared by `data-raw/prepare-pew-jewish-2020.R`.

Examples

```
# Design variables
head(pew_jewish_2020[, c("qkey", "extweight", "jewishcat")])

# Confirm 100 replicate weights are present
sum(grepl("^extweight[0-9]", names(pew_jewish_2020)))

# Inspect variable label (unique item text for battery variable)
```

```
attr(pew_jewish_2020$discrim_a, "label")

# Inspect value labels
attr(pew_jewish_2020$discrim_a, "labels")

# Inspect question preface (shared stem across the battery)
attr(pew_jewish_2020$discrim_a, "question_preface")

# Jewish identity distribution (use jewishcat, not raw religion vars)
table(pew_jewish_2020$jewishcat)
```

`pew_npors_2025` *Pew NPORS 2025: National Public Opinion Reference Survey*

Description

The 2025 National Public Opinion Reference Survey (NPORS), conducted February 5 – June 18, 2025, by Pew Research Center (n = 5,022). An address-based sample (ABS) drawn from the USPS Computerized Delivery Sequence File, with respondents completing the survey online, by paper, or by telephone in English or Spanish. All 65 columns from the public release file are retained.

Usage

```
pew_npors_2025
```

Format

A data frame with 5,022 rows and 65 variables. The 11 `smuse_*` variables form a battery asking about social media platform use and share a `"question_preface"` attribute. All other variables are documented individually below:

respid Case ID. Unique respondent identifier.

stratum Sampling stratum (10 levels, defined by census block group demographics).

basewt Base weight — inverse probability of selection, with adaptive mode adjustment.

weight Final weight — `basewt` after raking to Census population targets. Use for all population-level estimates.

mode Data collection mode: `1` = Online, `2` = Paper, `3` = Phone.

language Language interview completed in: `1` = English, `2` = Spanish.

languageinitial Language interview started in.

interview_start Interview start timestamp.

interview_end Interview end timestamp.

econ1mod Economic conditions in your community today (Excellent / Good / Fair / Poor).

econ1bmod Economic conditions one year from now (Better / Worse / Same).

comtype2 Community type: Urban / Suburban / Rural.

unity Americans united vs. divided on values.

crimesafe Area safety in terms of crime (Extremely safe – Not at all safe).

govprotct Government's role in protecting people from themselves.

moregunimpact Impact of more gun ownership on crime.

fin_sit Household financial situation (Comfortable – Can't meet basics).

vet1 Military service in household.

vol12_cps Volunteered for any organization in past 12 months.

eminuse Uses internet or email at least occasionally.

intmob Accesses internet on a mobile device.

intfreq Internet use frequency (6 categories).

intfreq_collapsed Internet use frequency (4 categories, derived).

home4nw2 Subscribes to home internet service.

bbhome Home internet type (dial-up, broadband, etc.).

smuse_fb Facebook. Part of social media use battery (see Details).

smuse_yt YouTube. Part of social media use battery (see Details).

smuse_x X (formerly Twitter). Part of social media use battery.

smuse_ig Instagram. Part of social media use battery.

smuse_sc Snapchat. Part of social media use battery.

smuse_wa WhatsApp. Part of social media use battery.

smuse_tt TikTok. Part of social media use battery.

smuse_rd Reddit. Part of social media use battery.

smuse_bsk Bluesky. Part of social media use battery.

smuse_th Threads. Part of social media use battery.

smuse_ts Truth Social. Part of social media use battery.

radio Listens to radio.

device1a Has a cell phone.

smart2 Cell phone is a smartphone.

nhisll Has a working landline telephone at home.

relig Current religion (12 categories).

religcat1 Religion (4 categories: Protestant, Catholic, Unaffiliated, Other).

born Born-again or evangelical Christian.

attendper In-person religious service attendance (6 categories).

attendonline2 Online/TV religious service participation (6 categories).

relimp Importance of religion in life (Very – Not at all).

pray Prayer frequency outside of services (7 categories).

educcat Education level (categorical).

hisp Hispanic origin.

racecmb Race (5 categories).

racethn Race-ethnicity (5 categories including Asian non-Hispanic).

agegrp Age in 13 five-year groups.

agecat Age (4 categories: 18-29, 30-49, 50-64, 65+).

birthplace U.S. born vs. foreign born.

gender Gender (man / woman / other).

adults Number of adults in household.

inc_sdt1 Total family income (8 categories from < $30,000 to $150,000+).

cregion Census region (NE / MW / S / W).

metro Metropolitan area indicator.

registration Registered to vote at current address.

party Party affiliation (Rep / Dem / Ind / Other).

partyln Party lean for Independents (Rep / Dem).

partysum Party summary (Rep+Lean Rep / Dem+Lean Dem / No lean).

voted2024 Voted in the 2024 presidential election.

votegen_post 2024 presidential vote choice (Trump / Harris / Other).

Details

Survey design: Stratified address-based sample with raking post-stratification — use Taylor series linearization. NPORS has no PSU (each address is its own unit, effectively a stratified SRS):

```
svy <- as_survey(pew_npors_2025,
  strata  = stratum,
  weights = weight
)
```

Use `basewt` instead of `weight` for sensitivity analyses comparing pre- and post-raking estimates.

Social media battery: All 11 `smuse_*` variables share the question stem `"Please indicate whether or not you ever use the following websites or apps."` Values: `1` = Selected, `2` = Not selected, `99` = Refused. Each variable additionally carries a `"question_preface"` attribute with this shared stem.

Metadata: All columns carry variable labels and value labels as R attributes from the original SPSS file. The 11 `smuse_*` battery variables additionally carry a `"question_preface"` attribute with the shared question stem. All three attribute types are automatically extracted into surveycore's metadata system when you call `as_survey()`.

- **Variable labels** (`"label"` attribute): A human-readable description of each column — for `smuse_*` variables this is just the platform name (e.g., `"Facebook"`). Example: `attr(pew_npors_2025$smuse_fb, "label")` returns `"Facebook"`.
- **Value labels** (`"labels"` attribute): A named numeric vector mapping each code to its meaning. Example: `attr(pew_npors_2025$smuse_fb, "labels")` returns `c(Selected = 1, "Not selected" = 2, Refused = 99)`.
- **Question preface** (`"question_preface"` attribute): The shared question stem for battery items, set on all `smuse_*` columns. Example: `attr(pew_npors_2025$smuse_fb, "question_preface")` returns `"Please indicate whether or not you ever use the following websites or apps."`.

Source

Pew Research Center. 2025 National Public Opinion Reference Survey. https://www.pewresearch.org/datasets/ (free account required to download raw data; the processed .rda is included in the package). Prepared by data-raw/prepare-pew-npors-2025.R.

Examples

```
# Variables in the dataset
names(pew_npors_2025)

# Create survey design (no PSU for ABS design)
svy <- as_survey(
  pew_npors_2025,
  strata = stratum,
  weights = weight
)

# Inspect variable label
attr(pew_npors_2025$smuse_fb, "label")

# Inspect value labels
attr(pew_npors_2025$smuse_fb, "labels")

# Inspect question preface (shared stem for all smuse_* battery items)
attr(pew_npors_2025$smuse_fb, "question_preface")
```

---

print.survey_anova *Print Method for survey_anova Objects*

Description

Print Method for survey_anova Objects

Usage

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

Arguments

x	A survey_anova tibble produced by get_anova().
...	Additional arguments (currently unused).

Value

x, invisibly.

print.survey_diffs *Print a Survey Diffs Result*

Description

Prints a structured header showing design type, family, dependent variable, treatment variable with reference level, and estimation method, then delegates to the tibble print method for the body.

Usage

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

Arguments

x	A survey_diffs object.
...	Passed to the tibble print method.

Value

x, invisibly.

print.survey_pairwise *Print method for survey_pairwise objects.*

Description

Print method for survey_pairwise objects.

Usage

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

Arguments

x	A survey_pairwise object.
...	Additional arguments (unused).

Value

x, invisibly.

`print.survey_result` *Print a Survey Result Object*

Description

Prints a labelled header showing the specific result class and dimensions, then delegates to the tibble print method for the tabular content.

Usage

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

Arguments

`x`	A `survey_result` object.
`...`	Passed to the tibble print method.

Value

`x`, invisibly.

Examples

```
result <- structure(
  tibble::tibble(mean = 42.0, se = 1.5, n = 100L),
  .meta = list(
    design_type = "taylor",
    conf_level = 0.95,
    call = quote(get_means(d, x)),
    n_respondents = 100L,
    group = list(),
    x = list(
      x = list(
        variable_label = NULL,
        question_preface = NULL,
        value_labels = NULL
      )
    )
  ),
  class = c("survey_means", "survey_result", "tbl_df", "tbl", "data.frame")
)
print(result)
```

print.survey_t_test *Print method for survey_t_test objects.*

Description

Print method for survey_t_test objects.

Usage

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

Arguments

x	A survey_t_test object.
...	Additional arguments (unused).

Value

x, invisibly.

remove_survey *Remove Surveys from a* survey_collection

Description

Drops one or more named surveys from a collection and returns a new survey_collection. Errors if any requested name is not present.

Usage

```
remove_survey(x, name)
```

Arguments

x	A survey_collection.
name	Character vector of survey names to drop. All names must be present in names(x).

Value

A new survey_collection without the dropped surveys. Errors surveycore_error_collection_empty if removing would leave the collection empty. This error is raised by the S7 class validator, not by remove_survey() itself.

See Also

as_survey_collection(), add_survey()

Other collections: add_survey(), as_survey_collection(), set_collection_id(), set_collection_if_missing_var
survey_collection()

Examples

```
d1 <- as_survey(
  gss_2024,
  ids = vpsu,
  weights = wtssps,
  strata = vstrat,
  nest = TRUE
)
d2 <- as_survey(
  gss_2024,
  ids = vpsu,
  weights = wtssps,
  strata = vstrat,
  nest = TRUE
)
coll <- as_survey_collection(a = d1, b = d2)
coll2 <- remove_survey(coll, "a")
names(coll2)
```

SE *Extract standard errors from a model or result object*

Description

A generic function that extracts standard errors. For survey_result objects (produced by get_means(), get_totals(), and other get_*() functions), returns a named numeric vector matching names(coef(object)).

Usage

```
SE(object, ...)
```

Arguments

object	A model or result object.
...	Additional arguments passed to methods.

Value

A named numeric vector of standard errors. The exact structure depends on the method.

Note

When both surveycore and survey are loaded, surveycore::SE masks survey::SE. The SE.default() method handles most survey package objects (svystat, svyby) via sqrt(diag(vcov(object, ...))). Exception: SE.default is NOT equivalent to survey::SE.svyby when vartype = "cvpct" was requested — in that case, qualify the call with survey::SE().

See Also

Other analysis: clean(), get_anova(), get_corr(), get_covariance(), get_diffs(), get_effective_n(), get_freqs(), get_means(), get_pairwise(), get_quantiles(), get_ratios(), get_t_test(), get_totals(), get_variance(), meta()

SE.default *Default SE method: extracts from vcov diagonal*

Description

Computes sqrt(diag(vcov(object, ...))). Works for any object with a vcov() method, including lm, glm, and survey::svystat objects.

Usage

```
## Default S3 method:
SE(object, ...)
```

Arguments

object	An object with a vcov() method.
...	Forwarded to vcov().

Value

A named numeric vector of standard errors.

Note

When surveycore::SE masks survey::SE, this default method covers survey::svystat objects but NOT survey::svyby with vartype = "cvpct". Use survey::SE() explicitly in that case.

`SE.survey_result` *Extract standard errors from a survey result object*

Description

Returns a named numeric vector of standard errors from a `survey_result` object. Values are `sqrt(diag(vcov(object)))`, ensuring consistency between `SE()` and `vcov()`.

Usage

```
## S3 method for class 'survey_result'
SE(object, ...)
```

Arguments

`object`	A `survey_result` object from any supported `get_*()` function.
`...`	Unused.

Value

A named numeric vector. Names are identical to `names(coef(object))`. `NA` entries propagate from missing `se` column or `NA` in `se`.

Note

`vcov.survey_result()` emits `surveycore_warning_vcov_incomplete` when it has no joint covariance for a result with `p > 1`. This warning is suppressed inside `SE.survey_result()` because the off-diagonal structure is irrelevant when only standard errors are requested. Call `vcov(result)` directly to observe this warning.

Examples

```
d <- as_survey(
  nhanes_2017,
  ids = sdmvpsu,
  weights = wtint2yr,
  strata = sdmvstra,
  nest = TRUE
)
result <- get_means(d, ridageyr, variance = "se")
SE(result)
```

set_collection_id *Set the Identifier Column on a* survey_collection

Description

Updates the @id property of a survey_collection. The new value is the column name .dispatch_over_collection() injects when an analysis function (get_means(), get_freqs(), etc.) is dispatched across the collection without an explicit per-call .id.

Usage

```
set_collection_id(x, id)
```

Arguments

x	A survey_collection.
id	Character(1). The new identifier column name. Must be non-NA and non-empty.

Details

Setting the same value as the existing @id returns the collection unchanged (no error, no warning). All other invariants on the collection (@surveys, @groups, @if_missing_var) are preserved.

Pipes naturally with the rest of the collection API:

```
coll |> set_collection_id("wave") |> get_means(y1)
```

Value

The modified survey_collection, invisibly.

See Also

Other collections: add_survey(), as_survey_collection(), remove_survey(), set_collection_if_missing_var(), survey_collection()

Examples

```
d1 <- as_survey(
  gss_2024,
  ids = vpsu,
  weights = wtssps,
  strata = vstrat,
  nest = TRUE
)
coll <- as_survey_collection(a = d1)
coll <- set_collection_id(coll, "wave")
coll@id
```

`set_collection_if_missing_var` *Set the Missing-Variable Behaviour on a* `survey_collection`

Description

Updates the `@if_missing_var` property of a `survey_collection`. The new value is the per-call default `.dispatch_over_collection()` uses when an analysis function (`get_means()`, `get_freqs()`, etc.) is dispatched across the collection without an explicit per-call `.if_missing_var`.

Usage

```
set_collection_if_missing_var(x, if_missing_var)
```

Arguments

`x`	A survey_collection.
`if_missing_var`	Character(1), one of `c("error", "skip")`. When `"skip"`, member surveys missing a requested variable are dropped from the dispatched result; when `"error"`, the dispatcher aborts.

Details

Setting the same value as the existing `@if_missing_var` returns the collection unchanged (no error, no warning). All other invariants on the collection (`@surveys`, `@groups`, `@id`) are preserved.

Pipes naturally with the rest of the collection API:

```
coll |> set_collection_if_missing_var("skip") |> get_means(y1)
```

Value

The modified `survey_collection`, invisibly.

See Also

Other collections: `add_survey()`, `as_survey_collection()`, `remove_survey()`, `set_collection_id()`, `survey_collection()`

Examples

```
d1 <- as_survey(
  gss_2024,
  ids = vpsu,
  weights = wtssps,
  strata = vstrat,
  nest = TRUE
)
coll <- as_survey_collection(a = d1)
```

```
coll <- set_collection_if_missing_var(coll, "skip")
coll@if_missing_var
```

`set_dataset_metadata` *Set Dataset-Level Metadata*

Description

Sets whole-dataset metadata — the survey name, the display name, the fielding vendor, and the field dates — on a survey design object or a data frame, using one of three conventions.

Usage

```
set_dataset_metadata(x, ..., key = NULL, value = NULL)
```

Arguments

`x`	A survey design object or `data.frame`.
`...`	Named arguments where the name is a dataset metadata key and the value is the value to store. A single named list is also accepted. Supports `!!!` list splicing. A `NULL` value deletes the key.
`key`	`character` vector of key names, or `NULL` (default). Use with `value`. Must be supplied by name, because it follows `....`
`value`	A list with one element per element of `key`, or `NULL` (default). An atomic vector is coerced with `as.list()`. Names on `value` are ignored — `key` supplies the names. A `NULL` element deletes that one key; `value = NULL` deletes every key in `key` and skips the length check. Must be supplied by name, because it follows `....`

Details

Convention 1 (named ...) — recommended:

```
set_dataset_metadata(x, vendor = "Ipsos", field_start = "2026-02-10")
```

Convention 2 (single named list in ...):

```
set_dataset_metadata(x, list(vendor = "Ipsos"))
```

Convention 3 (`key` + `value`):

```
set_dataset_metadata(x, key = "vendor", value = "Ipsos")
```

The key vocabulary is closed. Exactly six keys are valid, and this is their canonical order:

`survey_name` `character(1)`. Full formal survey name.

`data_name` `character(1)`. Display label for this dataset.

`vendor` `character(1)`. Fielding vendor.

`field_start` `Date(1)`. First day in the field.

`field_end` `Date(1)`. Last day in the field.

`field_period` `character(1)`. Prose field period, for display.

Any other key is an error. The four character keys take one non-`NA` string. The two date keys take a `Date` scalar or an ISO 8601 string (`"YYYY-MM-DD"`), which is stored as a `Date`; a looser string such as `"2026/02/10"` or `"2026-2-1"` is rejected. Stored keys are always kept in the canonical order above.

`survey_name` and `data_name` are independent. This function never reads one to fill or check the other.

Each key also has a convenience setter that takes one value: `set_survey_name()`, `set_data_name()`, `set_vendor()`, `set_field_dates()`, and `set_field_period()`. Use `extract_dataset_metadata()` to read the keys back.

A `NULL` value deletes a key. Deleting a key that is not set does nothing and raises no condition. A zero-length value such as `character(0)` is not a deletion — it is an invalid value.

`field_start` must not be after `field_end`. The comparison uses the *effective* value of each date: the new value when the call supplies one, no value when the call supplies `NULL`, and the stored value otherwise. So one call can repair a reversed stored pair by deleting one date and setting the other. Every check runs before any write, so a rejected call changes nothing.

On a survey design object the values go into the design's metadata, which is the single source of truth on every read path; attributes on the underlying data frame are neither read nor written. On a data frame the values go into whole-object attributes, one per key, and the columns are left untouched — even a column whose name matches a key.

`dates` is the pre-1.2.0 attribute name for the prose field period. Setting it is an error that points at `field_period` and `set_field_period()`. Deleting it is allowed: `dates = NULL` is an alias for `field_period = NULL`, and either spelling removes both the `field_period` and the legacy `dates` attribute from a data frame.

A design object restored from a file written by surveycore 1.1.0 or earlier cannot store dataset metadata. Reading such an object works, but a write raises an error naming the remedy: rebuild the object with a constructor.

Value

The modified object, invisibly.

Attribute persistence

On a data frame, dataset metadata lives in whole-object attributes, and those are fragile. Common operations return a new frame without them:

- base `[` when it selects columns, as in `df[, cols]`
- `merge()` and `subset()`
- `tibble::as_tibble()`, which has dropped them in some versions and keeps them in others, so neither behavior is safe to rely on

So set dataset metadata **last**, directly before `usethis::use_data()`. Or set it on the survey design instead: there it lives in the design's metadata, where no data-frame operation can drop it.

A constructor **copies** these attributes into the design; it never strips them. The data frame inside a design therefore still carries the values it was built with. A later `set_vendor()` call, or a deletion, changes the design's metadata only. Rebuilding a design from the data frame that `survey_data()` returns re-promotes the original attributes, so an edited or deleted key comes back. Avoid that round trip and keep working with the design object.

See Also

Other metadata: `classify_question_type()`, `extract_data_name()`, `extract_dataset_metadata()`, `extract_field_dates()`, `extract_field_period()`, `extract_higher_is()`, `extract_metadata()`, `extract_missing_codes()`, `extract_question_preface()`, `extract_reverse_coded()`, `extract_sata()`, `extract_survey_name()`, `extract_universe()`, `extract_val_labels()`, `extract_var_extra()`, `extract_var_label()`, `extract_var_note()`, `extract_vendor()`, `infer_question_prefaces()`, `set_data_name()`, `set_field_dates()`, `set_field_period()`, `set_higher_is()`, `set_missing_codes()`, `set_question_preface()`, `set_reverse_coded()`, `set_sata()`, `set_survey_name()`, `set_universe()`, `set_val_labels()`, `set_var_extra()`, `set_var_label()`, `set_var_note()`, `set_vendor()`, `survey_metadata()`, `survey_weighting_history()`

Examples

```
# On a data frame, dataset metadata lives in whole-object attributes.
df <- data.frame(id = 1:5, y = c(2, 4, 3, 5, 1), w = rep(1, 5))
df <- set_dataset_metadata(
  df,
  survey_name = "Example Attitudes Survey 2026",
  vendor = "Ipsos KnowledgePanel"
)
extract_dataset_metadata(df)

# An ISO 8601 string is stored as a Date.
df <- set_dataset_metadata(df, field_start = "2026-02-10")
extract_dataset_metadata(df, field_start)

# Convention 2: one named list.
df <- set_dataset_metadata(df, list(field_end = "2026-03-04"))

# Convention 3: key plus value.
df <- set_dataset_metadata(
  df,
  key = c("data_name", "field_period"),
  value = list("Example (February-March 2026)", "February-March 2026")
)
extract_dataset_metadata(df, format = "data_frame")

# A NULL value deletes a key.
df <- set_dataset_metadata(df, vendor = NULL)
extract_dataset_metadata(df, vendor, fill = NA)

# The same API works on a survey design object.
```

```
d <- as_survey(df, weights = w)
d <- set_dataset_metadata(d, vendor = "Cint")
extract_dataset_metadata(d, vendor)
```

`set_data_name` *Set the Dataset Display Name*

Description

Sets the `data_name` dataset-level metadata key — the display label for this particular dataset — on a survey design object or a data frame.

Usage

```
set_data_name(x, name)
```

Arguments

`x`	A survey design object or `data.frame`.
`name`	`character(1)`, non-`NA`, or `NULL` to delete the key. Omitting it is an error, not a deletion.

Details

A convenience wrapper for `set_dataset_metadata(x, data_name = name)`.

`data_name` is usually a short name plus the field period, for example `"AAA Ipsos (February-March 2026)"`. It is independent of `survey_name`: this function writes `data_name` and nothing else, and it never reads, fills, or checks `survey_name`. Use `set_survey_name()` for the formal name.

Value

The modified object, invisibly.

See Also

Other metadata: `classify_question_type()`, `extract_data_name()`, `extract_dataset_metadata()`, `extract_field_dates()`, `extract_field_period()`, `extract_higher_is()`, `extract_metadata()`, `extract_missing_codes()`, `extract_question_preface()`, `extract_reverse_coded()`, `extract_sata()`, `extract_survey_name()`, `extract_universe()`, `extract_val_labels()`, `extract_var_extra()`, `extract_var_label()`, `extract_var_note()`, `extract_vendor()`, `infer_question_prefaces()`, `set_dataset_metadata()`, `set_field_dates()`, `set_field_period()`, `set_higher_is()`, `set_missing_codes()`, `set_question_preface()`, `set_reverse_coded()`, `set_sata()`, `set_survey_name()`, `set_universe()`, `set_val_labels()`, `set_var_extra()`, `set_var_label()`, `set_var_note()`, `set_vendor()`, `survey_metadata()`, `survey_weighting_history()`

Examples

```
df <- data.frame(id = 1:5, y = c(2, 4, 3, 5, 1), w = rep(1, 5))
df <- set_data_name(df, "Example (February-March 2026)")
extract_data_name(df)

# The formal survey name is a separate key and stays unset.
extract_survey_name(df)

d <- as_survey(df, weights = w)
d <- set_data_name(d, "Example (March 2026)")
extract_data_name(d)
```

---

set_field_dates *Set the Field Dates*

Description

Sets the `field_start` and `field_end` dataset-level metadata keys — the first and last day in the field — on a survey design object or a data frame.

Usage

```
set_field_dates(x, field_start, field_end)
```

Arguments

`x`	A survey design object or `data.frame`.
`field_start`	`Date(1)`, an ISO 8601 `character(1)`, or `NULL` to delete the key. Omit the argument to leave the stored value unchanged.
`field_end`	`Date(1)`, an ISO 8601 `character(1)`, or `NULL` to delete the key. Omit the argument to leave the stored value unchanged.

Details

A convenience wrapper for `set_dataset_metadata()` that forwards **only the arguments the call supplies**. The three states of each argument are distinct:

not supplied The stored key is left unchanged.

`NULL` The stored key is deleted.

a value The key is set, after validation and coercion.

A value is either a `Date` scalar or an ISO 8601 string (`"YYYY-MM-DD"`), which is stored as a `Date`. A looser string such as `"2026/02/10"` or `"2026-2-1"` is rejected, and the error names this function's own argument.

`field_start` must not be after `field_end`. The comparison uses the *effective* value of each key: the new value when the call supplies one, no value when the call supplies `NULL`, and the stored value when the call does not mention the key. So one call can repair a reversed stored pair by deleting one date and setting the other, and deleting both always succeeds.

This function never touches `field_period` — the prose period has its own setter, `set_field_period()`.

Value

The modified object, invisibly.

See Also

Other metadata: classify_question_type(), extract_data_name(), extract_dataset_metadata(), extract_field_dates(), extract_field_period(), extract_higher_is(), extract_metadata(), extract_missing_codes(), extract_question_preface(), extract_reverse_coded(), extract_sata(), extract_survey_name(), extract_universe(), extract_val_labels(), extract_var_extra(), extract_var_label(), extract_var_note(), extract_vendor(), infer_question_prefaces(), set_data_name(), set_dataset_metadata(), set_field_period(), set_higher_is(), set_missing_codes(), set_question_preface(), set_reverse_coded(), set_sata(), set_survey_name(), set_universe(), set_val_labels(), set_var_extra(), set_var_label(), set_var_note(), set_vendor(), survey_metadata(), survey_weighting_history()

Examples

```
df <- data.frame(id = 1:5, y = c(2, 4, 3, 5, 1), w = rep(1, 5))
df <- set_field_dates(df, "2026-02-10", "2026-03-04")
extract_field_dates(df)

# Supplying one argument leaves the other key unchanged.
df <- set_field_dates(df, field_end = "2026-03-10")
extract_field_dates(df)

# NULL deletes a key.
df <- set_field_dates(df, field_start = NULL)
extract_field_dates(df)

d <- as_survey(df, weights = w)
d <- set_field_dates(d, field_start = as.Date("2026-02-11"))
extract_field_dates(d)
```

---

set_field_period *Set the Prose Field Period*

Description

Sets the `field_period` dataset-level metadata key — the field period written for display, such as `"February-March 2026"` — on a survey design object or a data frame.

Usage

```
set_field_period(x, period)
```

Arguments

`x`	A survey design object or `data.frame`.
`period`	`character(1)`, non-`NA`, or `NULL` to delete the key. Omitting it is an error, not a deletion.

Details

A convenience wrapper for `set_dataset_metadata(x, field_period = period)`.

The argument is named `period` because the function name already carries the `field_` context; the stored key is `field_period`. This function never touches the two structured dates — use `set_field_dates()` for those.

On a data frame, deleting the key also removes the pre-1.2.0 `dates` attribute, so the deletion is idempotent and a later `as_survey()` call cannot restore the old value.

Value

The modified object, invisibly.

See Also

Other metadata: `classify_question_type()`, `extract_data_name()`, `extract_dataset_metadata()`, `extract_field_dates()`, `extract_field_period()`, `extract_higher_is()`, `extract_metadata()`, `extract_missing_codes()`, `extract_question_preface()`, `extract_reverse_coded()`, `extract_sata()`, `extract_survey_name()`, `extract_universe()`, `extract_val_labels()`, `extract_var_extra()`, `extract_var_label()`, `extract_var_note()`, `extract_vendor()`, `infer_question_prefaces()`, `set_data_name()`, `set_dataset_metadata()`, `set_field_dates()`, `set_higher_is()`, `set_missing_codes()`, `set_question_preface()`, `set_reverse_coded()`, `set_sata()`, `set_survey_name()`, `set_universe()`, `set_val_labels()`, `set_var_extra()`, `set_var_label()`, `set_var_note()`, `set_vendor()`, `survey_metadata()`, `survey_weighting_history()`

Examples

```
df <- data.frame(id = 1:5, y = c(2, 4, 3, 5, 1), w = rep(1, 5))
df <- set_field_period(df, "February-March 2026")
extract_field_period(df)

d <- as_survey(df, weights = w)
d <- set_field_period(d, "Spring 2026")
extract_field_period(d)
```

---

set_higher_is *Set Direction-of-Improvement Attribute*

Description

Records whether a higher value is better (`"better"`) or worse (`"worse"`) for one or more variables in a survey design object or data frame. This metadata is used by `get_diffs()` when `show_favorability = TRUE`.

Usage

```
set_higher_is(x, ..., variable = NULL, direction = NULL)
```

Arguments

x	A survey design object or data.frame.
...	Named arguments where the name is the variable and the value is the direction ("better" or "worse"). Supports Convention 1 (named args: bpxsy1 = "worse") and Convention 2 (named character vector: c(bpxsy1 = "worse", lbxtc = "better")). Mutually exclusive with variable.
variable	character. Variable name(s) — Convention 3 alternative to .... Mutually exclusive with ....
direction	character. Direction value(s) for Convention 3. Must be "better", "worse", or NULL (to remove the attribute). Same length as variable, or NULL to remove.

Value

The modified object, invisibly.

See Also

extract_higher_is() to retrieve direction attributes

Other metadata: classify_question_type(), extract_data_name(), extract_dataset_metadata(), extract_field_dates(), extract_field_period(), extract_higher_is(), extract_metadata(), extract_missing_codes(), extract_question_preface(), extract_reverse_coded(), extract_sata(), extract_survey_name(), extract_universe(), extract_val_labels(), extract_var_extra(), extract_var_label(), extract_var_note(), extract_vendor(), infer_question_prefaces(), set_data_name(), set_dataset_metadata(), set_field_dates(), set_field_period(), set_missing_codes(), set_question_preface(), set_reverse_coded(), set_sata(), set_survey_name(), set_universe(), set_val_labels(), set_var_extra(), set_var_label(), set_var_note(), set_vendor(), survey_metadata(), survey_weighting_history()

Examples

```
d <- as_survey(
  nhanes_2017,
  ids = sdmvpsu,
  weights = wtint2yr,
  strata = sdmvstra,
  nest = TRUE
)
d <- set_higher_is(d, bpxsy1 = "worse")
extract_higher_is(d, bpxsy1)
```

---

set_missing_codes *Set Missing Code(s)*

Description

Sets missing-value codes for one or more variables. Missing codes are atomic vectors documenting which data values represent missing data (e.g., c(Refused = -2L, DontKnow = -1L)).

Usage

```
set_missing_codes(x, ..., variable = NULL, codes = NULL)
```

Arguments

x	A survey design object or a data frame.
...	Named arguments where the name is the variable and the value is a named atomic vector of missing codes. Supports !!! list splicing.
variable	A character vector of variable names. Use with codes.
codes	A list of named atomic vectors, one per element of variable. When variable has length 1, a bare named atomic vector is also accepted.

Details

Supports Conventions 1, 2, and 3 — see set_var_label() for details on the calling conventions. For Convention 3 with a single variable, a bare named atomic vector is accepted in addition to a list.

Value

The modified object, invisibly.

See Also

extract_missing_codes() to retrieve missing value codes

Other metadata: classify_question_type(), extract_data_name(), extract_dataset_metadata(), extract_field_dates(), extract_field_period(), extract_higher_is(), extract_metadata(), extract_missing_codes(), extract_question_preface(), extract_reverse_coded(), extract_sata(), extract_survey_name(), extract_universe(), extract_val_labels(), extract_var_extra(), extract_var_label(), extract_var_note(), extract_vendor(), infer_question_prefaces(), set_data_name(), set_dataset_metadata(), set_field_dates(), set_field_period(), set_higher_is(), set_question_preface(), set_reverse_coded(), set_sata(), set_survey_name(), set_universe(), set_val_labels(), set_var_extra(), set_var_label(), set_var_note(), set_vendor(), survey_metadata(), survey_weighting_history()

Examples

```
d <- as_survey(
  gss_2024,
  ids = vpsu,
  weights = wtssps,
  strata = vstrat,
  nest = TRUE
)
d <- set_missing_codes(d, happy = c(Refused = -1L, DK = -2L))
extract_missing_codes(d, happy)
```

`set_question_preface` *Set Question Preface(s)*

Description

Sets the question preface string for one or more variables. Question prefaces are the shared introductory text for a battery of related questions.

Usage

```
set_question_preface(x, ..., variable = NULL, preface = NULL)
```

Arguments

`x`	A survey design object or a data frame.
`...`	Named arguments where the name is the variable and the value is the preface string. Supports `!!!` list splicing.
`variable`	A character vector of variable names. Use with `preface`.
`preface`	A character vector of preface strings, one per element of `variable`.

Details

Supports Conventions 1, 2, and 3 — see `set_var_label()` for details.

Value

The modified object, invisibly.

See Also

`extract_question_preface()` to retrieve a preface

Other metadata: `classify_question_type()`, `extract_data_name()`, `extract_dataset_metadata()`, `extract_field_dates()`, `extract_field_period()`, `extract_higher_is()`, `extract_metadata()`, `extract_missing_codes()`, `extract_question_preface()`, `extract_reverse_coded()`, `extract_sata()`, `extract_survey_name()`, `extract_universe()`, `extract_val_labels()`, `extract_var_extra()`, `extract_var_label()`, `extract_var_note()`, `extract_vendor()`, `infer_question_prefaces()`, `set_data_name()`, `set_dataset_metadata()`, `set_field_dates()`, `set_field_period()`, `set_higher_is()`, `set_missing_codes()`, `set_reverse_coded()`, `set_sata()`, `set_survey_name()`, `set_universe()`, `set_val_labels()`, `set_var_extra()`, `set_var_label()`, `set_var_note()`, `set_vendor()`, `survey_metadata()`, `survey_weighting_history()`

Examples

```
d <- as_survey(
  gss_2024,
  ids = vpsu,
  weights = wtssps,
  strata = vstrat,
  nest = TRUE
)
d <- set_question_preface(d, happy = "Taken all together...")
extract_question_preface(d, happy)
```

`set_reverse_coded` *Set Reverse-Coded Flag*

Description

Marks one or more variables as reverse-coded in a survey design object or data frame. Uses the same two-convention pattern as `set_sata()`.

Usage

```
set_reverse_coded(x, ..., variable = NULL, reverse_coded = TRUE)
```

Arguments

`x`	A survey design object or `data.frame`.
`...`	<`tidy-select`> Variables to mark. Cannot be combined with `variable`.
`variable`	`character`. Alternative programmatic interface: character vector of variable names. Cannot be combined with `...`.
`reverse_coded`	`logical(1)`. `TRUE` (default) marks variables as reverse-coded; `FALSE` removes the flag. `NA` is not accepted.

Details

Convention A (tidy-select `...`) — recommended:

```
design |> set_reverse_coded(anxiety, worry)
```

Convention B (`variable` = character vector) — programmatic:

```
vars <- c("anxiety", "worry")
design |> set_reverse_coded(variable = vars)
```

Setting `reverse_coded = FALSE` removes the flag.

Value

The modified object, invisibly.

See Also

`extract_reverse_coded()` to retrieve reverse-coded flags

Other metadata: `classify_question_type()`, `extract_data_name()`, `extract_dataset_metadata()`, `extract_field_dates()`, `extract_field_period()`, `extract_higher_is()`, `extract_metadata()`, `extract_missing_codes()`, `extract_question_preface()`, `extract_reverse_coded()`, `extract_sata()`, `extract_survey_name()`, `extract_universe()`, `extract_val_labels()`, `extract_var_extra()`, `extract_var_label()`, `extract_var_note()`, `extract_vendor()`, `infer_question_prefaces()`, `set_data_name()`, `set_dataset_metadata()`, `set_field_dates()`, `set_field_period()`, `set_higher_is()`, `set_missing_codes()`, `set_question_preface()`, `set_sata()`, `set_survey_name()`, `set_universe()`, `set_val_labels()`, `set_var_extra()`, `set_var_label()`, `set_var_note()`, `set_vendor()`, `survey_metadata()`, `survey_weighting_history()`

Examples

```
d <- as_survey(
  nhanes_2017,
  ids = sdmvpsu,
  weights = wtint2yr,
  strata = sdmvstra,
  nest = TRUE
)
d <- set_reverse_coded(d, bpxsy1, ridageyr)
d <- set_reverse_coded(d, bpxsy1, reverse_coded = FALSE)
```

---

set_sata *Set SATA (Select-All-That-Apply) Flag*

Description

Marks one or more variables as select-all-that-apply (SATA) in a survey design object or a data frame. Unlike the other unified setters (which map variable names to heterogeneous content), `set_sata()` applies a single logical flag to all listed variables, so it uses a simplified two-convention pattern.

Usage

```
set_sata(x, ..., variable = NULL, sata = TRUE)
```

Arguments

`x`	A survey design object or `data.frame`.
`...`	<`tidy-select`> Variables to mark. Supports selection helpers: `tidyselect::starts_with()`, `tidyselect::all_of()`, `tidyselect::any_of()`, etc. Cannot be combined with `variable`.

variable	character. Alternative programmatic interface: character vector of variable names. Cannot be combined with ....
sata	logical(1). TRUE (default) marks variables as SATA; FALSE removes the SATA flag. NA is not accepted.

Details

Convention A (tidy-select ...) — recommended:

```
design |> set_sata(news_tv, news_online, news_radio)
design |> set_sata(starts_with("news_"))
```

Convention B (variable = character vector) — programmatic:

```
sata_vars <- c("news_tv", "news_online", "news_radio")
design |> set_sata(variable = sata_vars)
```

Setting sata = FALSE unmarks the listed variables.

Value

The modified object, invisibly.

See Also

extract_sata() to retrieve SATA flags

Other metadata: classify_question_type(), extract_data_name(), extract_dataset_metadata(), extract_field_dates(), extract_field_period(), extract_higher_is(), extract_metadata(), extract_missing_codes(), extract_question_preface(), extract_reverse_coded(), extract_sata(), extract_survey_name(), extract_universe(), extract_val_labels(), extract_var_extra(), extract_var_label(), extract_var_note(), extract_vendor(), infer_question_prefaces(), set_data_name(), set_dataset_metadata(), set_field_dates(), set_field_period(), set_higher_is(), set_missing_codes(), set_question_preface(), set_reverse_coded(), set_survey_name(), set_universe(), set_val_labels(), set_var_extra(), set_var_label(), set_var_note(), set_vendor(), survey_metadata(), survey_weighting_history()

Examples

```
d <- as_survey(
  nhanes_2017,
  ids = sdmvpsu,
  weights = wtint2yr,
  strata = sdmvstra,
  nest = TRUE
)
d <- set_sata(d, riagendr, ridageyr)
d <- set_sata(d, riagendr, sata = FALSE)
```

set_survey_name *Set the Formal Survey Name*

Description

Sets the survey_name dataset-level metadata key — the full formal name of the survey — on a survey design object or a data frame.

Usage

```
set_survey_name(x, name)
```

Arguments

x	A survey design object or data.frame.
name	character(1), non-NA, or NULL to delete the key. Omitting it is an error, not a deletion.

Details

A convenience wrapper for set_dataset_metadata(x, survey_name = name).

survey_name and data_name are independent keys. This function writes survey_name and nothing else: it never reads, fills, or checks data_name. Use set_data_name() for the display label. The two values may drift, and no function in the package reconciles them.

Value

The modified object, invisibly.

See Also

Other metadata: classify_question_type(), extract_data_name(), extract_dataset_metadata(), extract_field_dates(), extract_field_period(), extract_higher_is(), extract_metadata(), extract_missing_codes(), extract_question_preface(), extract_reverse_coded(), extract_sata(), extract_survey_name(), extract_universe(), extract_val_labels(), extract_var_extra(), extract_var_label(), extract_var_note(), extract_vendor(), infer_question_prefaces(), set_data_name(), set_dataset_metadata(), set_field_dates(), set_field_period(), set_higher_is(), set_missing_codes(), set_question_preface(), set_reverse_coded(), set_sata(), set_universe(), set_val_labels(), set_var_extra(), set_var_label(), set_var_note(), set_vendor(), survey_metadata(), survey_weighting_history()

Examples

```
df <- data.frame(id = 1:5, y = c(2, 4, 3, 5, 1), w = rep(1, 5))
df <- set_survey_name(df, "Example Attitudes Survey 2026")
extract_survey_name(df)
```

```
# The display label is a separate key and stays unset.
extract_data_name(df)

d <- as_survey(df, weights = w)
d <- set_survey_name(d, "Example Attitudes Survey, Wave 2")
extract_survey_name(d)
```

`set_universe` *Set Universe Description(s)*

Description

Sets the universe description for one or more variables. The universe describes the population to which a variable applies (e.g., `"Adults 18+"`).

Usage

```
set_universe(x, ..., variable = NULL, universe = NULL)
```

Arguments

`x`	A survey design object or a data frame.
`...`	Named arguments where the name is the variable and the value is the universe description string. Supports `!!!` list splicing.
`variable`	A character vector of variable names. Use with `universe`.
`universe`	A character vector of universe description strings, one per element of `variable`.

Details

Supports Conventions 1, 2, and 3 — see `set_var_label()` for details.

Value

The modified object, invisibly.

See Also

`extract_universe()` to retrieve universe descriptions

Other metadata: `classify_question_type()`, `extract_data_name()`, `extract_dataset_metadata()`, `extract_field_dates()`, `extract_field_period()`, `extract_higher_is()`, `extract_metadata()`, `extract_missing_codes()`, `extract_question_preface()`, `extract_reverse_coded()`, `extract_sata()`, `extract_survey_name()`, `extract_universe()`, `extract_val_labels()`, `extract_var_extra()`, `extract_var_label()`, `extract_var_note()`, `extract_vendor()`, `infer_question_prefaces()`, `set_data_name()`, `set_dataset_metadata()`, `set_field_dates()`, `set_field_period()`, `set_higher_is()`, `set_missing_codes()`, `set_question_preface()`, `set_reverse_coded()`, `set_sata()`, `set_survey_name()`, `set_val_labels()`, `set_var_extra()`, `set_var_label()`, `set_var_note()`, `set_vendor()`, `survey_metadata()`, `survey_weighting_history()`

Examples

```
d <- as_survey(
  gss_2024,
  ids = vpsu,
  weights = wtssps,
  strata = vstrat,
  nest = TRUE
)
d <- set_universe(d, age = "All respondents 18+")
extract_metadata(d, age)
```

---

set_val_labels *Set Value Labels*

Description

Sets value labels for one or more variables using one of three conventions.

Usage

```
set_val_labels(x, ..., variable = NULL, labels = NULL)
```

Arguments

x	A survey design object or a data frame.
...	Named arguments where the name is the variable and the value is a fully named vector of value labels. Supports !!! list splicing.
variable	A character vector of variable names.
labels	A list of named vectors, one per element of variable. When variable has length 1, a bare named vector is also accepted.

Details

Convention 1 (named ...) — recommended:

```
set_val_labels(x, sex = c(Male = 1L, Female = 2L))
```

Convention 2 (single named list in ...):

```
set_val_labels(x, list(sex = c(Male = 1L, Female = 2L)))
```

Convention 3 (variable + labels):

```
set_val_labels(x, variable = "sex", labels = c(Male = 1L, Female = 2L))
```

Value

The modified object, invisibly.

See Also

`extract_val_labels()` to retrieve value labels

Other metadata: `classify_question_type()`, `extract_data_name()`, `extract_dataset_metadata()`, `extract_field_dates()`, `extract_field_period()`, `extract_higher_is()`, `extract_metadata()`, `extract_missing_codes()`, `extract_question_preface()`, `extract_reverse_coded()`, `extract_sata()`, `extract_survey_name()`, `extract_universe()`, `extract_val_labels()`, `extract_var_extra()`, `extract_var_label()`, `extract_var_note()`, `extract_vendor()`, `infer_question_prefaces()`, `set_data_name()`, `set_dataset_metadata()`, `set_field_dates()`, `set_field_period()`, `set_higher_is()`, `set_missing_codes()`, `set_question_preface()`, `set_reverse_coded()`, `set_sata()`, `set_survey_name()`, `set_universe()`, `set_var_extra()`, `set_var_label()`, `set_var_note()`, `set_vendor()`, `survey_metadata()`, `survey_weighting_history()`

Examples

```
d <- as_survey(
  nhanes_2017,
  ids = sdmvpsu,
  weights = wtint2yr,
  strata = sdmvstra,
  nest = TRUE
)
d <- set_val_labels(d, riagendr = c(Male = 1L, Female = 2L))
```

---

set_var_extra *Set Variable Extension Slot Payload(s)*

Description

Sets a per-variable extension-slot payload — a flat, pass-through named list that surveycore stores and returns unchanged. surveycore never reads, validates, or acts on the payload's *values*; it confirms only that the top-level value is a list or `NULL`, and that a non-empty list's top-level names are present, non-empty, and unique.

Usage

```
set_var_extra(x, ..., variable = NULL, extra = NULL)
```

Arguments

`x`	A survey design object or a data frame.
`...`	Named arguments where the name is the variable and the value is the extension payload (a list, or `NULL` to clear it). Supports `!!!` list splicing.
`variable`	A character vector of variable names. Use with `extra`.

extra — A list of payloads, one element per entry of variable, in the same order. Each element must itself be a list or NULL.

Details

Supports Conventions 1, 2, and 3 — see set_var_label() for details on the calling conventions. Unlike set_missing_codes(), there is no bare-payload shorthand for Convention 3 with a single variable: a bare named list cannot be distinguished from a one-element outer list, so the payload must be wrapped in an outer list (extra = list(list(role = "demographic"))). Passing an unwrapped named list of length 1 raises surveycore_error_var_extra_ambiguous_wrap.

Value

The modified object, invisibly.

See Also

extract_var_extra() to retrieve extension-slot payloads

Other metadata: classify_question_type(), extract_data_name(), extract_dataset_metadata(), extract_field_dates(), extract_field_period(), extract_higher_is(), extract_metadata(), extract_missing_codes(), extract_question_preface(), extract_reverse_coded(), extract_sata(), extract_survey_name(), extract_universe(), extract_val_labels(), extract_var_extra(), extract_var_label(), extract_var_note(), extract_vendor(), infer_question_prefaces(), set_data_name(), set_dataset_metadata(), set_field_dates(), set_field_period(), set_higher_is(), set_missing_codes(), set_question_preface(), set_reverse_coded(), set_sata(), set_survey_name(), set_universe(), set_val_labels(), set_var_label(), set_var_note(), set_vendor(), survey_metadata(), survey_weighting_history()

Examples

```
d <- as_survey(
  gss_2024,
  ids = vpsu,
  weights = wtssps,
  strata = vstrat,
  nest = TRUE
)
d <- set_var_extra(d, age = list(role = "demographic"))
extract_var_extra(d, age)
```

---

set_var_label *Set Variable Label(s)*

Description

Sets variable labels using one of three conventions.

Usage

```
set_var_label(x, ..., variable = NULL, label = NULL)
```

Arguments

x	A survey design object or a data frame.
...	Named arguments where the name is the variable and the value is the label string. Supports !!! list splicing.
variable	A character vector of variable names. Use with label.
label	A character vector of label strings, one per element of variable.

Details

Convention 1 (named ...) — recommended for interactive use:

```
set_var_label(x, age = "Age in years", income = "Annual income")
set_var_label(x, !!!labels_list)   # list splicing
```

Convention 2 (named vector in ...) — useful for programmatic use:

```
set_var_label(x, c(age = "Age in years", income = "Annual income"))
```

Convention 3 (variable + label arguments) — for vector input:

```
vars <- c("age", "income")
lbls <- c("Age in years", "Annual income")
set_var_label(x, variable = vars, label = lbls)
```

Value

The modified object, invisibly.

See Also

extract_var_label() to retrieve a label

Other metadata: classify_question_type(), extract_data_name(), extract_dataset_metadata(), extract_field_dates(), extract_field_period(), extract_higher_is(), extract_metadata(), extract_missing_codes(), extract_question_preface(), extract_reverse_coded(), extract_sata(), extract_survey_name(), extract_universe(), extract_val_labels(), extract_var_extra(), extract_var_label(), extract_var_note(), extract_vendor(), infer_question_prefaces(), set_data_name(), set_dataset_metadata(), set_field_dates(), set_field_period(), set_higher_is(), set_missing_codes(), set_question_preface(), set_reverse_coded(), set_sata(), set_survey_name(), set_universe(), set_val_labels(), set_var_extra(), set_var_note(), set_vendor(), survey_metadata(), survey_weighting_history()

Examples

```
d <- as_survey(
  nhanes_2017,
  ids = sdmvpsu,
  weights = wtint2yr,
  strata = sdmvstra,
  nest = TRUE
)
d <- set_var_label(d, indfmpir = "Income-to-poverty ratio")

# Multiple variables
d <- set_var_label(
  d,
  bpxsy1 = "Systolic BP (1st reading)",
  bpxdi1 = "Diastolic BP (1st reading)"
)
```

`set_var_note` *Set Analyst Note(s)*

Description

Sets an analyst note for one or more variables. Notes are free-text annotations for documenting processing decisions, data quality concerns, or other context.

Usage

```
set_var_note(x, ..., variable = NULL, note = NULL)
```

Arguments

`x`	A survey design object or a data frame.
`...`	Named arguments where the name is the variable and the value is the note string. Supports `!!!` list splicing.
`variable`	A character vector of variable names. Use with `note`.
`note`	A character vector of note strings, one per element of `variable`.

Details

Supports Conventions 1, 2, and 3 — see `set_var_label()` for details.

Value

The modified object, invisibly.

See Also

`extract_var_note()` to retrieve a note

Other metadata: `classify_question_type()`, `extract_data_name()`, `extract_dataset_metadata()`, `extract_field_dates()`, `extract_field_period()`, `extract_higher_is()`, `extract_metadata()`, `extract_missing_codes()`, `extract_question_preface()`, `extract_reverse_coded()`, `extract_sata()`, `extract_survey_name()`, `extract_universe()`, `extract_val_labels()`, `extract_var_extra()`, `extract_var_label()`, `extract_var_note()`, `extract_vendor()`, `infer_question_prefaces()`, `set_data_name()`, `set_dataset_metadata()`, `set_field_dates()`, `set_field_period()`, `set_higher_is()`, `set_missing_codes()`, `set_question_preface()`, `set_reverse_coded()`, `set_sata()`, `set_survey_name()`, `set_universe()`, `set_val_labels()`, `set_var_extra()`, `set_var_label()`, `set_vendor()`, `survey_metadata()`, `survey_weighting_history()`

Examples

```
d <- as_survey(
  gss_2024,
  ids = vpsu,
  weights = wtssps,
  strata = vstrat,
  nest = TRUE
)
d <- set_var_note(d, age = "Top-coded at 89")
extract_var_note(d, age)
```

---

set_vendor *Set the Fielding Vendor*

Description

Sets the `vendor` dataset-level metadata key — the company that fielded the survey — on a survey design object or a data frame.

Usage

```
set_vendor(x, vendor)
```

Arguments

`x`	A survey design object or `data.frame`.
`vendor`	`character(1)`, non-`NA`, or `NULL` to delete the key. Omitting it is an error, not a deletion.

Details

A convenience wrapper for `set_dataset_metadata(x, vendor = vendor)`. It writes only the `vendor` key and leaves every other key untouched.

Value

The modified object, invisibly.

See Also

Other metadata: classify_question_type(), extract_data_name(), extract_dataset_metadata(), extract_field_dates(), extract_field_period(), extract_higher_is(), extract_metadata(), extract_missing_codes(), extract_question_preface(), extract_reverse_coded(), extract_sata(), extract_survey_name(), extract_universe(), extract_val_labels(), extract_var_extra(), extract_var_label(), extract_var_note(), extract_vendor(), infer_question_prefaces(), set_data_name(), set_dataset_metadata(), set_field_dates(), set_field_period(), set_higher_is(), set_missing_codes(), set_question_preface(), set_reverse_coded(), set_sata(), set_survey_name(), set_universe(), set_val_labels(), set_var_extra(), set_var_label(), set_var_note(), survey_metadata(), survey_weighting_history()

Examples

```
df <- data.frame(id = 1:5, y = c(2, 4, 3, 5, 1), w = rep(1, 5))
df <- set_vendor(df, "Ipsos KnowledgePanel")
extract_vendor(df)

# NULL deletes the key.
df <- set_vendor(df, NULL)
extract_vendor(df)

# The same call works on a survey design object.
d <- as_survey(df, weights = w)
d <- set_vendor(d, "Cint")
extract_vendor(d)
```

SURVEYCORE_DOMAIN_COL *Internal Domain Column Name Constant*

Description

The name of the logical column added to `@data` by `filter()` (from `surveytidy`) to mark domain membership. Exposed here so that sibling packages (`surveytidy`, `surveywts`) can reference it without using `:::`.

Usage

```
SURVEYCORE_DOMAIN_COL
```

survey_collection *Multi-Survey Container*

Description

An S7 container that holds multiple independent survey_base objects (e.g., multiple waves of a panel or cross-sectional series) for comparative analysis. Create with as_survey_collection().

Usage

```
survey_collection(
  surveys = list(),
  groups = character(0),
  id = ".survey",
  if_missing_var = "error"
)
```

Arguments

surveys	A named list of survey_base objects.
groups	Character vector of grouping variable names. Every member's @groups must be identical() to this value. Default character(0).
id	Character(1). Identifier column name used when dispatching analysis functions across the collection. Default ".survey".
if_missing_var	Character(1), one of c("error", "skip"). Default "error". Controls how dispatched get_*() functions behave when a member survey is missing a requested variable.

Details

survey_collection deliberately does **not** inherit from survey_base. This prevents collection-of-collections nesting: a survey_collection passed as an element of another collection fails the element-type check automatically.

Each element of @surveys is an independent survey_base subclass object (e.g., survey_taylor, survey_replicate, survey_twophase, survey_nonprob). Mixed-type collections are allowed — the collection never combines designs, so heterogeneous classes cannot produce an invalid state.

Value

A survey_collection object.

Properties

surveys A fully named list of survey_base objects. Length ≥ 1. Names are unique, non-NA, and non-empty.

groups A character vector of grouping variable names applied uniformly across every member survey. Default `character(0)` (ungrouped). When non-empty, every member's `@groups` is asserted `identical()` to this value.

id Character(1). Identifier column name injected by `.dispatch_over_collection()` when a `get_*()` is called on the collection. Default `".survey"`. Stored on the collection and consumed as the per-call default; a non-`NULL` `.id` at the analysis-function call site overrides this stored value. Mutate via `set_collection_id()`.

if_missing_var Character(1), one of `c("error", "skip")`. Default `"error"`. Controls how dispatched `get_*()` functions behave when a member is missing a requested variable. Stored on the collection and consumed as the per-call default; a non-`NULL` `.if_missing_var` at the analysis-function call site overrides this stored value. Mutate via `set_collection_if_missing_var()`.

See Also

`as_survey_collection()` to build a collection from survey objects; `add_survey()` / `remove_survey()` to mutate an existing collection.

Other collections: `add_survey()`, `as_survey_collection()`, `remove_survey()`, `set_collection_id()`, `set_collection_if_missing_var()`

Examples

```
d1 <- as_survey(
  gss_2024,
  ids = vpsu,
  weights = wtssps,
  strata = vstrat,
  nest = TRUE
)
coll <- survey_collection(surveys = list(gss = d1))
length(coll)
names(coll)
```

---

survey_data *Access the Data Component of a Survey Design Object*

Description

Returns the underlying data frame stored in a survey design object. This is a thin accessor for `x@data` that provides a stable public name independent of the S7 property structure.

Usage

```
survey_data(x, haven_class = FALSE)
```

Arguments

x	A survey_taylor, survey_replicate, survey_twophase, or survey_nonprob object.
haven_class	Rebuild the haven_labelled class on every column that has value labels. The design's metadata is read first, and the column's labels attribute is the fallback, so labels set by set_val_labels() on the design reach the returned column. The variable label follows the same order, so set_var_label() on the design reaches the returned column's label attribute. Value labels are the whole class test, not a record of which columns arrived as haven_labelled: a column that never carried the class is promoted too, so a column labelled by sjlabelled comes back with the class on it. A column that carries a class of its own is left alone: a factor keeps its levels and a Date keeps its calendar, whatever attributes they hold. FALSE, the default, returns base types, which is what every arithmetic and modelling operation needs. TRUE returns columns that haven and labelled recognise, so that haven::as_factor() and labelled::to_factor() give the label strings instead of the codes, and so that haven::write_sav() writes the labels the design holds. haven does not need to be installed for either value. Two known limits: haven::write_sav() raises on a column holding a tagged NA — that is a .sav format limit inside haven, and a column built by haven itself fails the same way. And set_val_labels(x, v = NULL) clears the metadata entry without touching the column, so the fallback rebuilds the class from the column's own attribute.

Details

nrow() and dim() report the stored row count, not the in-domain count. A domain filter keeps every row of the design and marks membership in a flag column, so the two counts differ after a filter. Printing a filtered design shows both, on the Domain: n of N rows line.

To get the in-domain count, call get_effective_n() and read its n column. That function honours the domain flag, takes a group argument, and accepts a survey_collection.

ncol() counts the full stored width. The count includes the weight column, the cluster and stratum ids, fpc, the domain flag when present, and every replicate-weight column. It is not a count of analysis variables, and it is not comparable across design classes.

nrow() is the unweighted count of sample rows. It is not a population estimate.

Value

A data.frame with all variables, including design variables. It is a tibble when the design stores a tibble.

With haven_class = FALSE, the default, no column carries the haven_labelled or haven_labelled_spss class. A survey design object stores base types. Every attribute the import set — the label string, the labels value-label vector, and the SPSS na_values and na_range vectors — stays on the column, and the value labels are also available through extract_val_labels(). In a printed tibble a previously labelled column shows the type token for its own type — <dbl>, <int> or <chr> — in place of the labelled token.

With haven_class = TRUE, every column that has value labels and no class of its own is returned with its haven_labelled class rebuilt, and a column that also carries na_values or na_range is

returned as haven_labelled_spss. A factor, a Date, and any other classed column keep the class they have. The labels and label attributes on the returned column hold what the design's metadata holds, which is what extract_val_labels() and extract_var_label() report. Values are unchanged, and the printed type token reads as the labelled token again — <dbl+lbl> for a double-backed column when the haven namespace is loaded, since haven supplies the method that renders it.

Examples

```
d <- as_survey(
  nhanes_2017,
  ids = sdmvpsu,
  weights = wtint2yr,
  strata = sdmvstra,
  nest = TRUE
)
head(survey_data(d))

# Read the data back with the labelled class rebuilt. gss_2024 carries
# value labels, so the class comes back on the columns that had one.
g <- as_survey(gss_2024, ids = vpsu, weights = wtssps, strata = vstrat)
class(survey_data(g)$happy)
class(survey_data(g, haven_class = TRUE)$happy)

# Value labels set on the design reach the rebuilt column, so an export
# carries the labels the analyst set rather than the labels the file had.
g <- set_val_labels(g, happy = c(Happy = 1, `So-so` = 2, Unhappy = 3))
attr(survey_data(g, haven_class = TRUE)$happy, "labels")
```

---

survey_glm *Fit a Survey-Weighted Generalised Linear Model*

Description

Fits a GLM to survey data, producing design-based coefficient estimates and variance-covariance matrix via the Binder (1983) sandwich estimator. All four concrete surveycore design classes are supported (survey_taylor, survey_replicate, survey_twophase, survey_nonprob). survey_collection inputs are rejected; call survey_glm() on each element individually.

Usage

```
survey_glm(
  design,
  formula = NULL,
  response = NULL,
  predictors = NULL,
  family = stats::gaussian(),
  na.action = stats::na.omit,
  start = NULL,
```

```
  etastart = NULL,
  mustart = NULL,
  control = list(),
  quiet = FALSE
)
```

Arguments

design	A survey design object created by as_survey(), as_survey_replicate(), as_survey_twophase(), or as_survey_nonprob().
formula	A model formula in standard R notation (e.g. y ~ x1 + x2). Mutually exclusive with response/predictors. If NULL and response is also NULL, errors with surveycore_error_formula_missing.
response	Character string naming the outcome variable. Programmatic alternative to formula. Mutually exclusive with formula. Use with predictors to build a model formula via reformulate(predictors, response). Suitable for lapply() iteration.
predictors	Character vector of predictor variable names. Used with response to build the model formula. If response is supplied and predictors is NULL, an intercept-only model is fitted.
family	A GLM family object specifying the error distribution and link function. Default gaussian(). Any family accepted by stats::glm() is supported. For binomial() and quasibinomial() families, the "non-integer #successes" warning is suppressed because survey weights are non-integer by design.
na.action	How to handle NA values in the model frame. Default na.omit (silently drops rows with any NA in model variables). na.fail errors with surveycore_error_na_in_data listing the offending columns and NA counts. Note: na.action applies only to model frame variables; survey weights are validated separately.
start	Starting values for the coefficient vector.
etastart	Starting values for the linear predictor.
mustart	Starting values for the mean.
control	A list of GLM control parameters passed to stats::glm.control().
quiet	Logical. If TRUE, suppresses convergence warnings emitted by survey_glm() and its internal replicate-weight refitting loop. Convergence status is always stored in fit@converged regardless of this setting, so non-convergence can still be detected programmatically. Default FALSE.

Details

Variance estimation: Uses the Binder (1983) sandwich estimator, which decomposes into per-observation score vectors passed to the Phase 0 variance machinery. The bread (X'WX)^(-1) accounts for IRLS working weights and is correct for all GLM families including binomial and Poisson.

binomial() **family:** Wraps the stats::glm() call in suppressWarnings() to suppress the "non-integer #successes" warning that fires for every survey-weighted binomial model.

Domain estimation: Use `surveytidy::filter()` before calling `survey_glm()`. The GLM is fit on in-domain rows only; variance estimation uses the full design for correct design-based SEs. The design degrees of freedom count only the PSUs and strata holding a row used in the fit (in the domain and not dropped by `na.action`), matching `survey::svyglm()` on a `subset()` design.

Multinomial response: `cbind()` on the LHS of `formula` is not supported. Multinomial logistic regression is deferred to a later phase.

Formula to model matrix: `survey_glm()` passes the formula to `stats::model.matrix()` via `stats::glm()`. Factor and character predictors are dummy-coded using `model.matrix()` default contrasts (treatment coding: first level as reference). Numeric predictors enter as-is. Interaction terms (`:`, `*`) and inline transformations (`log()`, `I()`) are supported as in any standard R formula. The resulting model matrix is `n x p` where `p` is the number of coefficients including the intercept.

Predictor variable types: Predictors may be numeric, integer, logical, factor, or character. Character predictors are coerced to factor by `stats::model.matrix()`. Ordered factors use polynomial contrasts by default. All other R types (list columns, complex, raw) will produce an error from `stats::model.matrix()`.

Input assumptions: surveycore assumes (1) each row of `design@data` represents one sampled unit; (2) survey weights are positive and finite for all rows (validated at construction time); (3) the model formula variables are columns of `design@data`; (4) the design is correctly specified before calling `survey_glm()`. No centering, scaling, or other pre-processing is applied to predictor variables beyond what the formula specifies.

Data transformations: No automatic transformation is applied to predictor or response variables. Factor encoding is handled by `stats::model.matrix()` using the active contrasts. Link function transformations (e.g. `log` link in `poisson()`) are applied by the family object, not by surveycore. To apply custom transformations, use `I()` or `log()` etc. inside the formula.

Row and column names: The coefficient vector returned in `fit@coefficients` carries the names produced by `stats::model.matrix()` (e.g. `"(Intercept)"`, `"sexFemale"`, `"age"`). `fit@vcov` carries the same names on rows and columns. `model.frame.survey_glm_fit()` returns the model frame with row names matching the rows used in fitting (i.e. the row names of `design@data` after applying `na.action`). Rows excluded by `na.action = na.omit` do not appear in the model frame.

Missing values: `na.action` controls handling of `NA` in model frame variables (predictors and response). `na.omit` (default) silently drops rows with any `NA`; the variance estimator uses the full design for correct sandwich SEs. `na.fail` stops with an informative error listing all variables containing `NA` and the row count for each. Survey weights are validated separately at construction time and must not contain `NA`.

Performance: Runtime scales as $O(n \cdot p^2)$ for the score matrix computation and $O(p^3)$ for the bread matrix (solve). For Taylor designs, variance estimation adds $O(n \cdot H \cdot p^2)$ where H is the number of strata. For replicate designs it adds $O(R \cdot n \cdot p)$ where R is the number of replicates. The dominant cost for large n is typically the `stats::glm()` IRLS fit ($O(n \cdot p^2 \cdot I)$ per IRLS iteration).

Value

A survey_glm_fit S7 object.

References

Binder, D.A. (1983) On the variances of asymptotically normal estimators from complex surveys. *International Statistical Review* **51**(3), 279–292.

Binder, D.A. (1991) Use of estimating functions for interval estimation from complex surveys. *Proceedings of the American Statistical Association, Section on Survey Research Methods*, 34–42.

Lumley, T. and Scott, A. (2014) Tests in surveys with complex sampling. *Journal of the Royal Statistical Society: Series B* **76**(2), 431–452.

See Also

Other constructors: as_caldata(), as_survey(), as_survey_nonprob(), as_survey_replicate(), as_survey_twophase(), survey_glm_fit(), survey_nonprob(), survey_replicate(), survey_taylor(), survey_twophase()

Examples

```
d <- as_survey(
  gss_2024,
  ids = vpsu,
  weights = wtssps,
  strata = vstrat,
  nest = TRUE
)

# Linear model: respondent age predicted by education and sex
fit <- survey_glm(d, age ~ educ + sex)
fit@coefficients
fit@vcov

# Programmatic interface — suitable for lapply()
results <- lapply(c("age", "educ"), function(v) {
  survey_glm(d, response = v, predictors = "sex")
})
```

---

survey_glm_fit *Survey-Weighted GLM Fit Object*

Description

S7 class produced by survey_glm(). Holds all regression output from a survey-weighted generalised linear model: design-based coefficient estimates, variance-covariance matrix, fitted values, residuals, and model metadata.

Usage

```
survey_glm_fit(
  coefficients = integer(0),
  vcov = NULL,
  fitted_values = integer(0),
  residuals = integer(0),
  weights = integer(0),
```

```
  design = survey_base(),
  degf = integer(0),
  family = list(),
  formula = NULL,
  null_deviance = integer(0),
  deviance = integer(0),
  df_null = integer(0),
  df_residual = integer(0),
  converged = logical(0),
  call = NULL,
  fit_ = NULL,
  term_assign = integer(0)
)
```

Arguments

coefficients	Named numeric vector of length p.
vcov	p x p design-based variance-covariance matrix.
fitted_values	Numeric vector of length n (response scale).
residuals	Working residuals from IRLS, length n.
weights	Survey weights used in fitting, length n.
design	The original survey_base survey design object.
degf	Raw design degrees of freedom (positive scalar). For survey_taylor designs (including SRS, which is absorbed into Taylor): number of PSUs minus number of strata, counting only those holding a row used in the fit. For survey_replicate designs: number of replicates minus one. For survey_twophase: phase-2 clusters minus phase-2 strata over the rows used in the fit, as survey::degf() counts it (each phase-2 row is a cluster when ids2 is not set). For survey_nonprob: Inf (no design-based df). This is *not* the residual degrees of freedom used for t-statistics and confidence intervals; those are computed as degf - (p - 1) where p is the number of model coefficients.
family	GLM family object (e.g. gaussian(), binomial()).
formula	Model formula.
null_deviance	Null model deviance.
deviance	Residual deviance.
df_null	Classical null df (fit$df.null from stats::glm()).
df_residual	Classical residual df (fit$df.residual, i.e. n - p). Used for the deviance display; **not** the design-based residual df.
converged	Logical; whether IRLS converged.
call	The survey_glm() call (language object or NULL).
fit_	Internal raw stats::glm() result; NULL after serialisation.
term_assign	Integer vector: attr(model.matrix(fit_), "assign") captured at fit time. Maps design-matrix columns to formula terms (0 = intercept; positive values index attr(terms(formula), "term.labels")). Required by get_anova()'s serialization-safe Wald path (spec §3.3.1): after @fit_ is stripped via saveRDS(), the term-to-column map survives in this slot. Default integer(0).

Value

A survey_glm_fit object.

See Also

survey_glm() to create a survey_glm_fit.

Other constructors: as_caldata(), as_survey(), as_survey_nonprob(), as_survey_replicate(), as_survey_twophase(), survey_glm(), survey_nonprob(), survey_replicate(), survey_taylor(), survey_twophase()

Examples

```
# survey_glm_fit objects are created by survey_glm(), not directly
d <- as_survey(
  gss_2024,
  ids = vpsu,
  weights = wtssps,
  strata = vstrat,
  nest = TRUE
)
fit <- survey_glm(d, age ~ sex)
fit@coefficients
```

---

survey_metadata *Survey Metadata Container*

Description

Stores variable labels, value labels, question prefaces, notes, and transformation history for variables in a survey design object. Automatically populated from haven-style attributes when as_survey() or related constructors are called.

Usage

```
survey_metadata(
  variable_labels = list(),
  value_labels = list(),
  question_prefaces = list(),
  notes = list(),
  var_extra = list(),
  universe = list(),
  missing_codes = list(),
  sata = list(),
  higher_is = list(),
  reverse_coded = list(),
  transformations = list(),
  weighting_history = list(),
  dataset_metadata = list()
)
```

Arguments

variable_labels
: A named list mapping variable names to character labels (e.g., `list(age = "Age in years")`).

value_labels
: A named list mapping variable names to named vectors of value labels (e.g., `list(sex = c(Male = 1L, Female = 2L))`).

question_prefaces
: A named list mapping variable names to shared question battery preface text.

notes
: A named list mapping variable names to analyst notes.

var_extra
: A named list mapping variable names to a pass-through extension payload — itself a named list, or `NULL`. surveycore never inspects, type-checks, or acts on the payload's values; it stores and returns them as-is. Use `set_var_extra()` and `extract_var_extra()` to access this property. Default `list()`.

universe
: A named list mapping variable names to universe descriptions (e.g., `list(age = "Adults 18+")`). Describes the population to which a variable applies.

missing_codes
: A named list mapping variable names to atomic vectors of missing-value codes (e.g., `list(age = c(Refused = 99L, DK = 98L))`).

sata
: A named list mapping variable names to `TRUE` for variables that are select-all-that-apply (SATA). Only variables explicitly marked as SATA appear in this list — absence means the variable is not SATA.

higher_is
: A named list mapping variable names to `"better"` or `"worse"`, indicating the direction of improvement for that variable. Absent keys mean the direction is unset. Use `set_higher_is()` and `extract_higher_is()` to access this property.

reverse_coded
: A named list mapping variable names to `TRUE` for variables that are reverse-coded. Absent keys mean the variable is not reverse-coded. Use `set_reverse_coded()` and `extract_reverse_coded()` to access this property.

transformations
: A named list tracking variable transformation history (populated automatically during operations).

weighting_history
: A list recording weighting operations applied to the survey object (e.g., raking, trimming). Each entry is written by a **surveywts** function and contains the operation name, parameters, effective sample size before/after, and design effect. Always `list()` until a **surveywts** weighting function is applied. Reserved for Phase 2.5.

dataset_metadata
: A named list of whole-dataset facts. The key vocabulary is **closed** — exactly six keys are valid, each with a declared type:

  - `survey_name` — `character(1)`, the full formal survey name.
  - `data_name` — `character(1)`, the display label for this dataset.
  - `vendor` — `character(1)`, the fielding vendor.
  - `field_start` — `Date` of length 1, the first day in the field.
  - `field_end` — `Date` of length 1, the last day in the field.

- `field_period` — `character(1)`, the prose field period.

No value may be `NA` and no element may be `NULL`. Any other key is rejected. `survey_name` and `data_name` are independent: no function derives one from the other. Use `set_dataset_metadata()` to write keys and `extract_dataset_metadata()` to read them; the default is `list()`. Each key also has a convenience pair: `set_survey_name()`, `set_data_name()`, `set_vendor()`, `set_field_dates()`, `set_field_period()`, and the matching `extract_survey_name()`, `extract_data_name()`, `extract_vendor()`, `extract_field_dates()`, `extract_field_period()`.

Value

A `survey_metadata` object.

See Also

Other metadata: `classify_question_type()`, `extract_data_name()`, `extract_dataset_metadata()`, `extract_field_dates()`, `extract_field_period()`, `extract_higher_is()`, `extract_metadata()`, `extract_missing_codes()`, `extract_question_preface()`, `extract_reverse_coded()`, `extract_sata()`, `extract_survey_name()`, `extract_universe()`, `extract_val_labels()`, `extract_var_extra()`, `extract_var_label()`, `extract_var_note()`, `extract_vendor()`, `infer_question_prefaces()`, `set_data_name()`, `set_dataset_metadata()`, `set_field_dates()`, `set_field_period()`, `set_higher_is()`, `set_missing_codes()`, `set_question_preface()`, `set_reverse_coded()`, `set_sata()`, `set_survey_name()`, `set_universe()`, `set_val_labels()`, `set_var_extra()`, `set_var_label()`, `set_var_note()`, `set_vendor()`, `survey_weighting_history()`

Examples

```
# Empty metadata (default)
m <- survey_metadata()
m@variable_labels

# Pre-populated metadata
m <- survey_metadata(
  variable_labels = list(age = "Respondent age", income = "Annual income"),
  value_labels = list(sex = c(Male = 1L, Female = 2L))
)
m@variable_labels$age
m@value_labels$sex

# Dataset-level metadata (closed six-key vocabulary)
m <- survey_metadata(dataset_metadata = list(vendor = "Ipsos"))
m@dataset_metadata$vendor
```

`survey_nonprob` *Non-probability Samples*

Description

A survey design object for non-probability samples (e.g., online panels, quota samples, volunteer panels) with calibration weights (including raking and post-stratification) or inverse probability weighting (IPW) pseudo-weights. Create with `as_survey_nonprob()`.

Usage

```
survey_nonprob(
  data = data.frame(),
  metadata = survey_metadata(),
  variables = list(),
  groups = character(0),
  call = NULL,
  calibration = NULL,
  reference_sample = NULL
)
```

Arguments

`data`	A `data.frame` containing the survey data. Prefer `as_survey_nonprob()` over calling this constructor directly.
`metadata`	A survey_metadata object. Created automatically by `as_survey_nonprob()`.
`variables`	A named list of design specification (`weights`, `probs_provided`). Set automatically by `as_survey_nonprob()`.
`groups`	Set by surveytidy's `group_by()`. Always `character(0)` in standalone surveycore use.
`call`	Language object capturing the construction call.
`calibration`	The calibration provenance object returned by a **surveywts** calibration function (e.g., `surveywts::rake()`), or `NULL` if calibration was performed externally. Stores the calibration targets, variables, and trimming parameters for reproducibility and future bootstrap re-calibration. Default `NULL`.
`reference_sample`	Optional survey_taylor object representing the probability-based reference sample used to estimate propensity scores or calibration targets. Stored for reproducibility. Default `NULL`.

Value

A `survey_nonprob` object.

Variance estimation

Two modes are available, selected by whether `@variables$repweights` is `NULL`:

SRS approximation (no replicate weights) Standard errors treat the calibrated weights as fixed and assume simple random sampling. This understates calibration uncertainty and should only be used when replicate weights are unavailable.

Replicate variance (repweights supplied) Bootstrap or jackknife replicate weights propagate calibration uncertainty into the variance estimate. Each replicate column must contain calibrated weights re-estimated on one replicate draw. This is the recommended approach.

See `as_survey_nonprob()` for the full parameter interface, including `type`, `scale`, `rscales`, and `mse`.

Non-probability samples

Unlike `as_survey()`, `as_survey_replicate()`, and `as_survey_twophase()`, this class does **not** assume a probability sampling design. When no replicate weights are supplied, standard errors rest on a model-assisted SRS assumption, which is consistent with common practice for calibrated non-probability samples (e.g., raked online panels). When replicate weights are supplied, bootstrap or jackknife variance is used instead. See `vignette("creating-survey-objects")` for guidance on choosing between these modes and the limitations of each.

Design variables (`@variables`)

`weights` Character string naming the (calibrated) weight column.

`repweights` Character vector of bootstrap replicate weight column names, or `NULL` when no replicate weights are present.

`type` Replicate type. Only `"bootstrap"` is supported for non-probability samples (`"JK1"`, `"JK2"`, and `"JKn"` are not accepted); or `NULL` when no replicate weights are present.

`scale` Numeric scale factor for the variance formula, or `NULL`.

`rscales` Per-replicate scale factors, or `NULL`.

`mse` Logical. `TRUE` for MSE form of variance, or `NULL`.

`probs_provided` Always `FALSE` for calibrated designs.

Calibration provenance (`@calibration`)

When calibration is performed via **surveywts**, the returned calibration object is stored here. It contains the calibration targets, variables used, trimming cap, effective sample size before and after, and design effect. `NULL` when calibration was performed externally (e.g., via `anesrake`).

See Also

`as_survey_nonprob()` to create a `survey_nonprob` object.

Other constructors: `as_caldata()`, `as_survey()`, `as_survey_nonprob()`, `as_survey_replicate()`, `as_survey_twophase()`, `survey_glm()`, `survey_glm_fit()`, `survey_replicate()`, `survey_taylor()`, `survey_twophase()`

survey_replicate *Replicate Weights Survey Design*

Description

A survey design object using replicate weights for variance estimation. Create with as_survey_replicate().

Usage

```
survey_replicate(
  data = data.frame(),
  metadata = survey_metadata(),
  variables = list(),
  groups = character(0),
  call = NULL,
  calibration = NULL
)
```

Arguments

data	A data.frame containing the survey data. Prefer as_survey_replicate() over calling this constructor directly.
metadata	A survey_metadata object. Created automatically by as_survey_replicate().
variables	A named list of design specification (weights, repweights, type, scale, rscales, fpc, fpctype, fpc_applied, fpc_fraction, mse, rho). Set automatically by as_survey_replicate().
groups	Set by surveytidy's group_by(). Always character(0) in standalone survey-core use.
call	Language object capturing the construction call.
calibration	A list of calibration data elements produced by as_caldata(), or NULL (default) for no calibration. When non-NULL, variance estimation routines apply a Deville-Sarndal calibration correction.

Value

A survey_replicate object.

Design variables (@variables)

weights Character string naming the weight column.

repweights Character vector of replicate weight column names. The replicate weight matrix is computed on demand from design@data[, design@variables$repweights] — it is not stored as a property.

type Replicate weight method: one of "JK1", "JK2", "JKn", "BRR", "Fay", "bootstrap", "ACS", "successive-difference", or "other".

scale Numeric scaling factor for variance estimation.

`rscales` Numeric vector of replicate-specific scales, or `NULL`.

`fpc` Name of the FPC column the design was built from, or `NULL`. Provenance only: no replicate variance formula reads it.

`fpctype` `"fraction"` or `"correction"`: how to read the `fpc` column.

`fpc_applied` Where the finite population correction reached the variance: `"rscales"` (inside `rscales`), `"repweights"` (inside the replicate weight columns), `"none"` (an FPC was available and was not applied), or `NULL` (no FPC was available, or the design predates this key). With `fpc` set and `fpc_applied` `NULL`, whether the FPC was applied is not recorded.

`fpc_fraction` The sampling fraction `f` per replicate, a numeric vector with one value per replicate column, when `as_survey_replicate()` applied a numeric `fpc`; otherwise `NULL`.

`mse` Logical. Use MSE estimates?

`rho` Fay shrinkage factor for `type = "Fay"`; `NULL` for every other type.

References

Canty, A.J. and Davison, A.C. (1999) Resampling-based variance estimation for labour force surveys. *The Statistician* **48**(3), 379–391.

Deville, J.-C. and Sarndal, C.-E. (1992) Calibration estimators in survey sampling. *Journal of the American Statistical Association* **87**(418), 376–382.

Deville, J.-C., Sarndal, C.-E. and Sautory, O. (1993) Generalized raking procedures in survey sampling. *Journal of the American Statistical Association* **88**(423), 1013–1020.

Judkins, D.R. (1990) Fay's method for variance estimation. *Journal of Official Statistics* **6**(3), 223–239.

Rao, J.N.K., Wu, C.F.J. and Yue, K. (1992) Some recent work on resampling methods for complex surveys. *Survey Methodology* **18**(2), 209–217.

Shao, J. and Tu, D. (1995) *The Jackknife and Bootstrap*. Springer.

See Also

`as_survey_replicate()` to create a `survey_replicate` object.

Other constructors: `as_caldata()`, `as_survey()`, `as_survey_nonprob()`, `as_survey_replicate()`, `as_survey_twophase()`, `survey_glm()`, `survey_glm_fit()`, `survey_nonprob()`, `survey_taylor()`, `survey_twophase()`

Examples

```
# Prefer as_survey_replicate() over calling survey_replicate() directly
set.seed(1)
df <- data.frame(
  y = rnorm(20),
  wt = runif(20, 1, 3),
  rep1 = runif(20, 0.5, 2),
  rep2 = runif(20, 0.5, 2)
)
d <- as_survey_replicate(
  df,
```

```
  weights = wt,
  repweights = starts_with("rep"),
  type = "BRR"
)
class(d)
```

---

survey_taylor *Taylor Series Linearization Survey Design*

Description

A survey design object using Taylor series (linearization) for variance estimation. Create with as_survey().

Usage

```
survey_taylor(
  data = data.frame(),
  metadata = survey_metadata(),
  variables = list(),
  groups = character(0),
  call = NULL,
  calibration = NULL
)
```

Arguments

data	A data.frame containing the survey data. Prefer as_survey() over calling this constructor directly.
metadata	A survey_metadata object. Created automatically by as_survey().
variables	A named list of design specification (ids, weights, strata, fpc, nest, probs_provided). Set automatically by as_survey().
groups	Set by surveytidy's group_by(). Always character(0) in standalone survey-core use.
call	Language object capturing the construction call.
calibration	A list of calibration data elements produced by as_caldata(), or NULL (default) for no calibration. When non-NULL, variance estimation routines apply a Deville-Sarndal calibration correction.

Value

A survey_taylor object.

Design variables (@variables)

`ids` Character vector of cluster ID column names, or `NULL` for simple random sampling.

`weights` Character string naming the weight column.

`strata` Character string naming the strata column, or `NULL`.

`fpc` Character string naming the finite population correction column, or `NULL`.

`nest` Logical. `TRUE` if cluster IDs are nested within strata (i.e., the same ID value in two strata refers to two distinct PSUs).

`probs_provided` Logical. `TRUE` if the user supplied `probs` rather than `weights` to `as_survey()`.

References

Deville, J.-C. and Sarndal, C.-E. (1992) Calibration estimators in survey sampling. *Journal of the American Statistical Association* **87**(418), 376–382.

Deville, J.-C., Sarndal, C.-E. and Sautory, O. (1993) Generalized raking procedures in survey sampling. *Journal of the American Statistical Association* **88**(423), 1013–1020.

Lumley, T. (2010) *Complex Surveys: A Guide to Analysis Using R*. John Wiley and Sons.

Rao, J.N.K., Yung, W. and Hidiroglou, M.A. (2002) Estimating equations for the analysis of survey data using poststratification information. *Sankhya* **64-A**, 22–36.

Sarndal, C-E., Swensson, B. and Wretman, J. (1992) *Model Assisted Survey Sampling*. Springer.

See Also

`as_survey()` to create a `survey_taylor` object.

Other constructors: `as_caldata()`, `as_survey()`, `as_survey_nonprob()`, `as_survey_replicate()`, `as_survey_twophase()`, `survey_glm()`, `survey_glm_fit()`, `survey_nonprob()`, `survey_replicate()`, `survey_twophase()`

Examples

```
# Prefer as_survey() over calling survey_taylor() directly
d <- as_survey(
  gss_2024,
  ids = vpsu,
  weights = wtssps,
  strata = vstrat,
  nest = TRUE
)
class(d)
```

survey_twophase *Two-Phase Survey Design*

Description

A survey design object for two-phase (double) sampling. Create with as_survey_twophase().

Usage

```
survey_twophase(
  data = data.frame(),
  metadata = survey_metadata(),
  variables = list(),
  groups = character(0),
  call = NULL
)
```

Arguments

data	A data.frame containing the survey data (all Phase 1 rows, with a logical indicator for Phase 2 membership). Prefer as_survey_twophase() over calling this constructor directly.
metadata	A survey_metadata object. Inherited from the Phase 1 design when using as_survey_twophase().
variables	A named list of design specification (phase1, phase2, subset, method). Set automatically by as_survey_twophase().
groups	Set by surveytidy's group_by(). Always character(0) in standalone survey-core use.
call	Language object capturing the construction call.

Value

A survey_twophase object.

Design variables (@variables)

phase1 Named list containing the Phase 1 design specification (from a survey_taylor object's @variables).

phase2 Named list with optional Phase 2 design columns: ids, strata, probs, fpc — each NULL or a character vector of column names.

subset Character string naming the logical column that indicates Phase 2 membership (TRUE = selected into Phase 2).

method "full", "approx", or "simple".

See Also

as_survey_twophase() to create a survey_twophase object.

Other constructors: as_caldata(), as_survey(), as_survey_nonprob(), as_survey_replicate(), as_survey_twophase(), survey_glm(), survey_glm_fit(), survey_nonprob(), survey_replicate(), survey_taylor()

Examples

```
# Prefer as_survey_twophase() over calling survey_twophase() directly
set.seed(1)
df <- data.frame(
  id = 1:100,
  y = rnorm(100),
  x = rnorm(100),
  wt = runif(100, 1, 3),
  in_phase2 = c(rep(TRUE, 40), rep(FALSE, 60))
)
phase1 <- as_survey(df, weights = wt)
d <- as_survey_twophase(phase1, subset = in_phase2, method = "approx")
class(d)
```

---

survey_weighting_history

Extract the Weighting History from a Survey Object

Description

Returns the list of weighting operations recorded on a survey design object. Each entry is appended by surveywts after a calibration or nonresponse adjustment step. Returns an empty list when no history has been recorded.

Usage

```
survey_weighting_history(x)
```

Arguments

x	A survey design object (any class inheriting from survey_base).

Value

A list of history entries, or list() if no history is present.

See Also

Other metadata: classify_question_type(), extract_data_name(), extract_dataset_metadata(), extract_field_dates(), extract_field_period(), extract_higher_is(), extract_metadata(), extract_missing_codes(), extract_question_preface(), extract_reverse_coded(), extract_sata(), extract_survey_name(), extract_universe(), extract_val_labels(), extract_var_extra(), extract_var_label(), extract_var_note(), extract_vendor(), infer_question_prefaces(), set_data_name(), set_dataset_metadata(), set_field_dates(), set_field_period(), set_higher_is(), set_missing_codes(), set_question_preface(), set_reverse_coded(), set_sata(), set_survey_name(), set_universe(), set_val_labels(), set_var_extra(), set_var_label(), set_var_note(), set_vendor(), survey_metadata()

Examples

```
d <- as_survey(
  nhanes_2017,
  ids = sdmvpsu,
  weights = wtint2yr,
  strata = sdmvstra,
  nest = TRUE
)
survey_weighting_history(d) # list() — no weighting history
```

---

update_design *Update Design Variables on an Existing Survey Object*

Description

Updates one or more design variables (weights, cluster IDs, strata, FPC, or replicate weights) on an existing survey design object. Use this after modifying the underlying data — for example, after recalibrating weights or adding a stratification variable. Emits an informational message listing changed variables.

Usage

```
update_design(
  x,
  ids = NULL,
  weights = NULL,
  strata = NULL,
  fpc = NULL,
  repweights = NULL,
  validate = TRUE
)
```

Arguments

x	A survey_taylor or survey_replicate object. survey_twophase is not supported; create a new design with as_survey_twophase().
ids	<tidy-select> New cluster (PSU) ID column(s). NULL (default) means no change. Only used for survey_taylor objects.
weights	<tidy-select> New weight column (a single column, values strictly > 0). NULL (default) means no change.
strata	<tidy-select> New stratification column (a single column). NULL (default) means no change. Only used for survey_taylor objects.
fpc	<tidy-select> New finite population correction column (a single column). NULL (default) means no change. Only used for survey_taylor objects.
repweights	<tidy-select> New replicate weight columns (one or more). NULL (default) means no change. Only used for survey_replicate objects. Must select the same number of columns the design already has: the design's scale and rscales were set for that number of replicates and are kept as they are. To change the number of replicates, rebuild the design with as_survey_replicate().
validate	Logical. If FALSE, temporarily marks the object to suppress validation during the variable update. In practice this has no observable effect on the returned object; validate is accepted for interface compatibility. Default TRUE.

Value

The modified survey object, invisibly. As a side effect, a cli_inform() message is printed listing each changed design variable (old name → new name).

See Also

as_survey() to create a survey_taylor object, as_survey_replicate() to create a survey_replicate object, as_survey_twophase() to create a survey_twophase object

Examples

```
# NHANES has two weight columns for different analysis types;
# start with the MEC examination weight for exam participants
exam <- nhanes_2017[nhanes_2017$ridstatr == 2, ]
d <- as_survey(
  exam,
  ids = sdmvpsu,
  weights = wtmec2yr,
  strata = sdmvstra,
  nest = TRUE
)

# Switch to interview weight for interview-based variables
d_updated <- update_design(d, weights = wtint2yr)
```

`vcov.survey_result` *Extract variance-covariance matrix from a survey result object*

Description

Returns a named numeric matrix from a `survey_result` object: the covariance matrix of the estimates in `coef(object)`, with the per-estimate variances on the diagonal.

Usage

```
## S3 method for class 'survey_result'
vcov(object, ...)
```

Arguments

`object`	A `survey_result` object from any supported `get_*()` function.
`...`	Unused.

Details

For `get_freqs()`, `get_totals()`, `get_means()` and `get_ratios()` on a Taylor or replicate design (including a `survey_nonprob` design with replicate weights), the full covariance matrix is computed when the result is built and stored with it. It covers every pair of rows: two levels of one variable, the same level in two groups, and rows on a filtered design. It is the same matrix `survey::vcov()` returns for `survey::svymean()`, `survey::svytotal()`, `survey::svyratio()` or `survey::svyby(covmat = TRUE)`. Two complementary proportions, for example, have covariance `-var`. The stored matrix is read by name, so `vcov()` still returns the right block after the result's rows are filtered or reordered, and it does not need an `se` column.

Every other result holds no joint covariance. `vcov()` then returns the per-estimate variances (`se^2`) on the diagonal and `NA_real_` off it, and emits `surveycore_warning_vcov_incomplete` when `p > 1`. These are results from `get_quantiles()`, `get_corr()`, `get_covariance()` and `get_diffs()`; the four functions above on a two-phase design or on a `survey_nonprob` design without replicate weights; and a stored matrix that no longer matches the result's rows. When the `se` column is absent as well, the matrix is all `NA_real_`, with no warning.

Value

A `p × p` numeric matrix with `rownames` and `colnames` equal to `names(coef(object))`.

Examples

```
d_sex <- as_survey(
  transform(nhanes_2017, sex = factor(riagendr, 1:2, c("Male", "Female"))),
  ids = sdmvpsu,
  weights = wtint2yr,
  strata = sdmvstra,
  nest = TRUE
```

```
)
# Two complementary proportions: the covariance is -var
vcov(get_freqs(d_sex, sex))

# Means by group: the groups share PSUs, so they covary
vcov(get_means(d_sex, ridageyr, group = sex))
```

Index

∗ **accessors**
survey_data, 157
∗ **analysis**
clean, 38
get_anova, 65
get_corr, 67
get_covariance, 72
get_diffs, 75
get_effective_n, 78
get_freqs, 80
get_means, 83
get_pairwise, 86
get_quantiles, 88
get_ratios, 91
get_t_test, 96
get_totals, 93
get_variance, 98
meta, 106
SE, 129
∗ **collections**
add_survey, 6
as_survey_collection, 16
remove_survey, 128
set_collection_id, 132
set_collection_if_missing_var, 133
survey_collection, 156
∗ **constructors**
as_caldata, 11
as_survey, 13
as_survey_nonprob, 18
as_survey_replicate, 22
as_survey_twophase, 27
survey_glm, 159
survey_glm_fit, 162
survey_nonprob, 166
survey_replicate, 169
survey_taylor, 171
survey_twophase, 173
∗ **conversion**
as_svydesign, 31
as_tbl_svy, 33
from_svydesign, 62
from_tbl_svy, 64
∗ **datasets**
acs_pums_wy, 5
anes_2024, 8
ca_api_2000, 34
gss_2024, 101
nhanes_2017, 107
ns_wave1, 109
pew_jewish_2020, 116
pew_npors_2025, 122
∗ **metadata**
classify_question_type, 37
extract_data_name, 44
extract_dataset_metadata, 42
extract_field_dates, 45
extract_field_period, 46
extract_higher_is, 47
extract_metadata, 48
extract_missing_codes, 49
extract_question_preface, 50
extract_reverse_coded, 52
extract_sata, 53
extract_survey_name, 54
extract_universe, 55
extract_val_labels, 56
extract_var_extra, 57
extract_var_label, 58
extract_var_note, 60
extract_vendor, 61
infer_question_prefaces, 103
set_data_name, 137
set_dataset_metadata, 134
set_field_dates, 138
set_field_period, 139
set_higher_is, 140
set_missing_codes, 141

set_question_preface, 143
set_reverse_coded, 144
set_sata, 145
set_survey_name, 147
set_universe, 148
set_val_labels, 149
set_var_extra, 150
set_var_label, 151
set_var_note, 153
set_vendor, 154
survey_metadata, 164
survey_weighting_history, 174
.get_design_vars_flat, 4

acs_pums_wy, 5
add_survey, 6
add_survey(), *17*, *129*, *132, 133*, *157*
anes_2024, 8
anova.survey_glm_fit, 10
as_caldata, 11
as_caldata(), *14, 15*, *21*, *25, 26*, *30*, *162*, *164*, *168–172*, *174*
as_survey, 13
as_survey(), *12*, *19–21*, *26*, *28–30*, *105*, *140*, *160*, *162*, *164*, *168*, *170–172*, *174*, *176*
as_survey_collection, 16
as_survey_collection(), *6, 7*, *129*, *132, 133*, *156, 157*
as_survey_nonprob, 18
as_survey_nonprob(), *12*, *15*, *24*, *26*, *30, 31*, *115*, *160*, *162*, *164*, *167, 168*, *170*, *172*, *174*
as_survey_replicate, 22
as_survey_replicate(), *12*, *15*, *19–21*, *30*, *160*, *162*, *164*, *168–170*, *172*, *174*, *176*
as_survey_twophase, 27
as_survey_twophase(), *12*, *15*, *20, 21*, *26*, *63*, *160*, *162*, *164*, *168*, *170*, *172–174*, *176*
as_svydesign, 31
as_svydesign(), *33, 34*, *63, 64*
as_tbl_svy, 33
as_tbl_svy(), *33*, *63, 64*

base::match.arg(), *69*

ca_api_2000, 34
classify_question_type, 37
classify_question_type(), *43–47*, *49–55*, *57–61*, *105*, *136, 137*, *139–143*, *145–148*, *150–152*, *154, 155*, *166*, *175*
clean, 38
clean(), *66*, *71*, *74*, *78*, *80*, *83*, *85*, *87*, *90*, *93*, *95*, *98*, *100*, *107*, *130*
coef.survey_result, 40
confint.survey_result, 41

extract_data_name, 44
extract_data_name(), *38*, *43*, *45–47*, *49–55*, *57–61*, *105*, *136, 137*, *139–143*, *145–148*, *150–152*, *154, 155*, *166*, *175*
extract_dataset_metadata, 42
extract_dataset_metadata(), *38*, *44–55*, *57–61*, *105*, *135–137*, *139–143*, *145–148*, *150–152*, *154, 155*, *166*, *175*
extract_field_dates, 45
extract_field_dates(), *38*, *43, 44*, *46, 47*, *49–55*, *57–61*, *105*, *136, 137*, *139–143*, *145–148*, *150–152*, *154, 155*, *166*, *175*
extract_field_period, 46
extract_field_period(), *38*, *43–45*, *47*, *49–55*, *57–61*, *105*, *136, 137*, *139–143*, *145–148*, *150–152*, *154, 155*, *166*, *175*
extract_higher_is, 47
extract_higher_is(), *38*, *43–46*, *49–55*, *57–61*, *105*, *136, 137*, *139–143*, *145–148*, *150–152*, *154, 155*, *165, 166*, *175*
extract_metadata, 48
extract_metadata(), *38*, *43–47*, *50–55*, *57–61*, *105*, *136, 137*, *139–143*, *145–148*, *150–152*, *154, 155*, *166*, *175*
extract_missing_codes, 49
extract_missing_codes(), *38*, *43–47*, *49*, *51–55*, *57–61*, *105*, *136, 137*, *139–143*, *145–148*, *150–152*, *154, 155*, *166*, *175*
extract_question_preface, 50
extract_question_preface(), *38*, *43–47*, *49, 50*, *52–55*, *57–61*, *105*, *136, 137*,

139–143, *145–148*, *150–152*, *154*, *155*, *166*, *175*
extract_reverse_coded, 52
extract_reverse_coded(), *38*, *43–47*, *49–51*, *53–55*, *57–61*, *105*, *136*, *137*, *139–143*, *145–148*, *150–152*, *154*, *155*, *165*, *166*, *175*
extract_sata, 53
extract_sata(), *38*, *43–47*, *49–52*, *54*, *55*, *57–61*, *105*, *136*, *137*, *139–143*, *145–148*, *150–152*, *154*, *155*, *166*, *175*
extract_survey_name, 54
extract_survey_name(), *38*, *43–47*, *49–53*, *55*, *57–61*, *105*, *136*, *137*, *139–143*, *145–148*, *150–152*, *154*, *155*, *166*, *175*
extract_universe, 55
extract_universe(), *38*, *43–47*, *49–54*, *57–61*, *105*, *136*, *137*, *139–143*, *145–148*, *150–152*, *154*, *155*, *166*, *175*
extract_val_labels, 56
extract_val_labels(), *38*, *43–47*, *49–55*, *58–61*, *105*, *136*, *137*, *139–143*, *145–148*, *150–152*, *154*, *155*, *158*, *159*, *166*, *175*
extract_var_extra, 57
extract_var_extra(), *38*, *43–47*, *49–55*, *57*, *59–61*, *105*, *136*, *137*, *139–143*, *145–148*, *150–152*, *154*, *155*, *165*, *166*, *175*
extract_var_label, 58
extract_var_label(), *38*, *43–47*, *49–55*, *57*, *58*, *60*, *61*, *105*, *136*, *137*, *139–143*, *145–148*, *150–152*, *154*, *155*, *159*, *166*, *175*
extract_var_note, 60
extract_var_note(), *38*, *43–47*, *49–55*, *57–59*, *61*, *105*, *136*, *137*, *139–143*, *145–148*, *150–152*, *154*, *155*, *166*, *175*
extract_vendor, 61
extract_vendor(), *38*, *43–47*, *49–55*, *57–60*, *105*, *136*, *137*, *139–143*, *145–148*, *150–152*, *154*, *155*, *166*, *175*

from_svydesign, 62
from_svydesign(), *14*, *29*, *33*, *34*, *64*
from_tbl_svy, 64
from_tbl_svy(), *33*, *34*, *63*

get_anova, 65
get_anova(), *10*, *39*, *71*, *74*, *78*, *80*, *83*, *85*, *87*, *90*, *93*, *95*, *98*, *100*, *107*, *125*, *130*
get_corr, 67
get_corr(), *39*, *66*, *74*, *78*, *80*, *83*, *85*, *87*, *90*, *93*, *95*, *98*, *100*, *107*, *130*, *177*
get_covariance, 72
get_covariance(), *39*, *66*, *71*, *78*, *80*, *83*, *85*, *87*, *90*, *93*, *95*, *98*, *100*, *107*, *130*, *177*
get_diffs, 75
get_diffs(), *39*, *66*, *71*, *74*, *80*, *83*, *85*, *87*, *90*, *93*, *95*, *98*, *100*, *107*, *130*, *140*, *177*
get_effective_n, 78
get_effective_n(), *39*, *66*, *71*, *74*, *78*, *83*, *85*, *87*, *90*, *93*, *95*, *98*, *100*, *107*, *130*, *158*
get_freqs, 80
get_freqs(), *39*, *66*, *71*, *74*, *78*, *80*, *85*, *87*, *90*, *93–95*, *98*, *100*, *107*, *130*, *177*
get_means, 83
get_means(), *26*, *39*, *66*, *71*, *74*, *78*, *80*, *83*, *87*, *90*, *93*, *95*, *98*, *100*, *107*, *130*, *177*
get_pairwise, 86
get_pairwise(), *39*, *66*, *71*, *74*, *78*, *80*, *83*, *85*, *90*, *93*, *95*, *98*, *100*, *107*, *130*
get_quantiles, 88
get_quantiles(), *39*, *66*, *71*, *74*, *78*, *80*, *83*, *85*, *87*, *93*, *95*, *98*, *100*, *107*, *130*, *177*
get_ratios, 91
get_ratios(), *39*, *66*, *71*, *74*, *78*, *80*, *83*, *85*, *87*, *90*, *95*, *98*, *100*, *107*, *130*, *177*
get_t_test, 96
get_t_test(), *39*, *66*, *71*, *74*, *78*, *80*, *83*, *85–87*, *90*, *93*, *95*, *100*, *107*, *130*
get_totals, 93
get_totals(), *26*, *39*, *66*, *71*, *74*, *78*, *80*, *83*, *85*, *87*, *90*, *93*, *98*, *100*, *107*, *130*, *177*
get_variance, 98
get_variance(), *39*, *66*, *71*, *74*, *78*, *80*, *83*, *85*, *87*, *90*, *93*, *95*, *98*, *107*, *130*
gss_2024, 101

infer_question_prefaces, 103
infer_question_prefaces(), *38*, *43–47*, *49–55*, *57–61*, *136*, *137*, *139–143*,

145–148, *150–152*, *154*, *155*, *166*, *175*

meta, 106
meta(), *39*, *66*, *71*, *74*, *78*, *80*, *83*, *85*, *87*, *90*, *93*, *95*, *98*, *100*, *130*

nhanes_2017, 107
ns_wave1, 109

pew_jewish_2020, 116
pew_npors_2025, 122
print.survey_anova, 125
print.survey_diffs, 126
print.survey_pairwise, 126
print.survey_result, 127
print.survey_t_test, 128

remove_survey, 128
remove_survey(), *7*, *17*, *132*, *133*, *157*
rlang::check_dots_empty(), *65*

SE, 129
SE(), *39*, *66*, *71*, *74*, *78*, *80*, *83*, *85*, *87*, *90*, *93*, *95*, *98*, *100*, *107*
SE.default, 130
SE.survey_result, 131
set_collection_id, 132
set_collection_id(), *7*, *16*, *17*, *129*, *133*, *157*
set_collection_if_missing_var, 133
set_collection_if_missing_var(), *7*, *17*, *129*, *132*, *157*
set_data_name, 137
set_data_name(), *38*, *43–47*, *49–55*, *57–61*, *105*, *135*, *136*, *139–143*, *145–148*, *150–152*, *154*, *155*, *166*, *175*
set_dataset_metadata, 134
set_dataset_metadata(), *38*, *43–47*, *49–55*, *57–61*, *105*, *137*, *139–143*, *145–148*, *150–152*, *154*, *155*, *166*, *175*
set_field_dates, 138
set_field_dates(), *38*, *43–47*, *49–55*, *57–61*, *105*, *135–137*, *140–143*, *145–148*, *150–152*, *154*, *155*, *166*, *175*
set_field_period, 139
set_field_period(), *38*, *43–47*, *49–55*, *57–61*, *105*, *135–139*, *141–143*, *145–148*, *150–152*, *154*, *155*, *166*, *175*
set_higher_is, 140
set_higher_is(), *38*, *43–47*, *49–55*, *57–61*, *76*, *105*, *136*, *137*, *139*, *140*, *142*, *143*, *145–148*, *150–152*, *154*, *155*, *165*, *166*, *175*
set_missing_codes, 141
set_missing_codes(), *38*, *43–47*, *49–55*, *57–61*, *105*, *136*, *137*, *139–141*, *143*, *145–148*, *150–152*, *154*, *155*, *166*, *175*
set_question_preface, 143
set_question_preface(), *38*, *43–47*, *49–55*, *57–61*, *105*, *136*, *137*, *139–142*, *145–148*, *150–152*, *154*, *155*, *166*, *175*
set_reverse_coded, 144
set_reverse_coded(), *38*, *43–47*, *49–55*, *57–61*, *105*, *136*, *137*, *139–143*, *146–148*, *150–152*, *154*, *155*, *165*, *166*, *175*
set_sata, 145
set_sata(), *37*, *38*, *43–47*, *49–55*, *57–61*, *105*, *136*, *137*, *139–145*, *147*, *148*, *150–152*, *154*, *155*, *166*, *175*
set_survey_name, 147
set_survey_name(), *38*, *43–47*, *49–55*, *57–61*, *105*, *135–137*, *139–143*, *145*, *146*, *148*, *150–152*, *154*, *155*, *166*, *175*
set_universe, 148
set_universe(), *38*, *43–47*, *49–55*, *57–61*, *105*, *136*, *137*, *139–143*, *145–147*, *150–152*, *154*, *155*, *166*, *175*
set_val_labels, 149
set_val_labels(), *38*, *43–47*, *49–55*, *57–61*, *105*, *136*, *137*, *139–143*, *145–148*, *151*, *152*, *154*, *155*, *158*, *166*, *175*
set_var_extra, 150
set_var_extra(), *38*, *43–47*, *49–55*, *57–61*, *105*, *136*, *137*, *139–143*, *145–148*, *150*, *152*, *154*, *155*, *165*, *166*, *175*
set_var_label, 151
set_var_label(), *15*, *26*, *38*, *43–47*, *49–55*, *57–61*, *105*, *136*, *137*, *139–143*, *145–148*, *150*, *151*, *153–155*, *158*, *166*, *175*

set_var_note, 153
set_var_note(), *38*, *43–47*, *49–55*, *57–61*, *105*, *136, 137*, *139–143*, *145–148*, *150–152*, *155*, *166*, *175*
set_vendor, 154
set_vendor(), *38*, *43–47*, *49–55*, *57–61*, *105*, *135–137*, *139–143*, *145–148*, *150–152*, *154*, *166*, *175*
stats::glm(), *160*
stats::glm.control(), *160*
stats::optimize(), *70*
stats::p.adjust(), *76, 77*, *87*
survey::as.svrepdesign(), *62*
survey::svydesign(), *14*
survey::twophase(), *63*
survey_base, *65, 66*, *156*, *163*
survey_collection, *16, 17*, *69*, *72, 73*, *77*, *81, 82*, *85*, *87*, *89*, *92*, *95*, *97*, *99, 100*, *132, 133*, 156
survey_collection(), *7*, *17*, *129*, *132, 133*
survey_data, 157
survey_data(), *136*
survey_glm, 159
survey_glm(), *11, 12*, *15*, *21*, *26*, *30*, *39*, *65, 66*, *77*, *162*, *164*, *168*, *170*, *172*, *174*
survey_glm_fit, *10*, *65, 66*, *161*, 162
survey_glm_fit(), *12*, *15*, *21*, *26*, *30*, *162*, *168*, *170*, *172*, *174*
survey_metadata, 164, *167*, *169*, *171*, *173*
survey_metadata(), *38*, *43–47*, *49–55*, *57–61*, *105*, *136, 137*, *139–143*, *145–148*, *150–152*, *154, 155*, *175*
survey_nonprob, 166
survey_nonprob(), *12*, *15*, *21*, *26*, *30*, *162*, *164*, *170*, *172*, *174*
survey_replicate, *11*, 169
survey_replicate(), *12*, *15*, *21*, *26*, *30*, *162*, *164*, *168*, *172*, *174*
survey_taylor, *11*, *19*, *167*, 171
survey_taylor(), *12*, *15*, *21*, *26*, *30*, *162*, *164*, *168*, *170*, *174*
survey_twophase, 173
survey_twophase(), *12*, *15*, *21*, *26*, *30*, *162*, *164*, *168*, *170*, *172*
survey_weighting_history, 174
survey_weighting_history(), *38*, *43–47*, *49–55*, *57–61*, *105*, *136, 137*, *139–143*, *145–148*, *150–152*, *154*, *155*, *166*
SURVEYCORE_DOMAIN_COL, 155

tidyselect::all_of(), *37*, *48, 49*, *51*, *53*, *55–57*, *59, 60*, *145*
tidyselect::any_of(), *37*, *48, 49*, *51*, *53*, *55–57*, *59, 60*, *145*
tidyselect::matches(), *48, 49*, *51*, *55–57*, *59, 60*
tidyselect::starts_with(), *37*, *48, 49*, *51*, *53*, *55–57*, *59, 60*, *145*

update_design, 175
update_design(), *14*

vcov.survey_result, 177
vcov.survey_result(), *83*, *85*, *93*, *95*