

# Package ‘surveycore’

May 9, 2026

**Title** Core Survey Analysis Infrastructure

**Version** 0.8.3

**Description** Provides 'S7'-based infrastructure for survey analysis.

Supports Taylor series, replicate weight, and two-phase designs following the methods in 'Lumley' (2004) <[doi:10.18637/jss.v009.i08](https://doi.org/10.18637/jss.v009.i08)>. Includes design-based estimators such as means, frequencies, and regression models, with weighted 'polychoric' and 'polyserial' correlation following 'Mannan' (2025) <[doi:10.2139/ssrn.6580480](https://doi.org/10.2139/ssrn.6580480)>. A metadata system automatically preserves 'haven'-style variable labels, value labels, and question-preface attributes through all operations. Uses a 'tidyselect' interface for design specification.

**License** GPL (>= 3)

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**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), margineffects (>= 0.18.0), pbivnorm (>= 0.6.0), stats, graphics

**Suggests** testthat (>= 3.0.0), withr (>= 2.5.0), 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), covr, knitr, rmarkdown

**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

**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** 2026-05-05 15:12:03 UTC

## Contents

acs_pums_wy . . . . .	3
add_survey . . . . .	5
anes_2024 . . . . .	6
as_survey . . . . .	8
as_survey_collection . . . . .	11
as_survey_nonprob . . . . .	13
as_survey_replicate . . . . .	14
as_survey_twophase . . . . .	17
as_svydesign . . . . .	19
as_tbl_svy . . . . .	20
classify_question_type . . . . .	21
clean . . . . .	22
extract_metadata . . . . .	24
extract_missing_codes . . . . .	25
extract_question_preface . . . . .	26
extract_sata . . . . .	27
extract_universe . . . . .	28
extract_val_labels . . . . .	29
extract_var_label . . . . .	30
extract_var_note . . . . .	31
from_svydesign . . . . .	32
from_tbl_svy . . . . .	33
get_anova . . . . .	34
get_corr . . . . .	36
get_covariance . . . . .	40
get_diffs . . . . .	43
get_freqs . . . . .	46
get_means . . . . .	49
get_pairwise . . . . .	51
get_quantiles . . . . .	53
get_ratios . . . . .	56
get_totals . . . . .	58
get_t_test . . . . .	60
get_variance . . . . .	62
gss_2024 . . . . .	65

infer_question_prefaces . . . . .	67
meta . . . . .	69
nhanes_2017 . . . . .	71
ns_wave1 . . . . .	73
pew_jewish_2020 . . . . .	80
pew_npors_2025 . . . . .	86
remove_survey . . . . .	89
set_collection_id . . . . .	90
set_collection_if_missing_var . . . . .	91
set_missing_codes . . . . .	92
set_question_preface . . . . .	93
set_sata . . . . .	94
set_universe . . . . .	95
set_val_labels . . . . .	96
set_var_label . . . . .	98
set_var_note . . . . .	99
survey_collection . . . . .	100
survey_data . . . . .	101
survey_glm . . . . .	102
survey_glm_fit . . . . .	105
survey_metadata . . . . .	107
survey_replicate . . . . .	108
survey_taylor . . . . .	110
survey_twophase . . . . .	111
survey_weighting_history . . . . .	112
update_design . . . . .	113

**Index****115**

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).
- `povpn`: 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

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

---

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
)

```

## 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.

## Value

A survey\_taylor object.

## 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

- Sarndal, C-E., Swensson, B. and Wretman, J. (1991) *Model Assisted Survey Sampling*. Springer.
- 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.

### 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_survey_nonprob()`, `as_survey_replicate()`, `as_survey_twophase()`, `survey_data()`, `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$ridstr == 2, ]
d_bp <- as_survey(exam, ids = sdmvpsu, weights = wtmecl2yr,
  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 <code>survey_base</code> objects, passed with explicit names or as bare symbols. At least one argument is required.
group	<code>&lt;tidy-select&gt;</code> Grouping variable(s) to apply uniformly across every member survey. Accepts bare names ( <code>region</code> , <code>c(region, stratum)</code> ), <code>all_of()</code> , etc. When supplied and resolving to a non-empty character vector, the named columns must exist in every member's <code>@data</code> ; they are propagated onto each member's <code>@groups</code> and set as <code>coll@groups</code> . If a member already carries a non-empty <code>@groups</code> that differs from the resolved target, the target takes precedence and a <code>surveycore_warning_collection_group_overridden</code> warning is emitted (one per divergent member). When missing or resolving to an empty vector ( <code>NULL</code> , <code>character(0)</code> , <code>c()</code> , <code>all_of(character(0))</code> ), the collection adopts the members' uniform <code>@groups</code> if they are all identical, or errors <code>surveycore_error_collection_group</code> if they differ. Default: missing (adopt-from-members).
.id	Character(1). Identifier column name used when dispatching analysis functions across the collection. Default <code>".survey"</code> . Stored on the returned collection's <code>@id</code> property and used as the default by <code>.dispatch_over_collection()</code> when a per-call <code>.id</code> is not supplied (i.e., when an analysis function is called with <code>.id = NULL</code> ). Mutate via <code>set_collection_id()</code> .

`.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\\_var\(\)](#)

### 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)
coll3@groups
```

---

as\_survey\_nonprob      *Create a Calibrated / Non-Probability Survey Design*

---

## Description

**[Experimental]**

## Usage

```
as_survey_nonprob(data, weights, calibration = NULL)
```

## Arguments

data	A <code>data.frame</code> containing the survey responses with pre-computed calibration weights. Must have at least one row and unique column names.
weights	<code>&lt;tidy-select&gt;</code> Calibration weight column (a single column, values strictly > 0). Typically produced by an external raking function (e.g., <code>anesrake::anesrake()</code> ) or a <b>surveywts</b> calibration function.
calibration	Optional. The calibration provenance object returned by a <b>surveywts</b> calibration function (e.g., <code>surveywts::rake()</code> ). Stored in <code>@calibration</code> for reproducibility. Supply <code>NULL</code> (the default) when calibration was performed externally and provenance metadata is not available. The object's structure is defined by <b>surveywts</b> and will be formally specified in Phase 2.5.

## Details

Creates a survey design object for non-probability samples and post-hoc calibrated designs (e.g., raked online panels, post-stratified samples). Accepts pre-computed calibration weights and optionally stores calibration provenance from **surveywts** output for reproducibility.

## Value

A `survey_nonprob` object.

## Phase 2.5 skeleton

This constructor is a **skeleton**. The resulting `survey_nonprob` object supports estimation via a model-assisted SRS variance assumption — the same as calling `as_survey()` with weights only. Full bootstrap re-calibration variance (which re-applies the raking procedure on each replicate) will be implemented in Phase 2.5 alongside the **surveywts** package.

## 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 note

Standard errors from a `survey_nonprob` object assume simple random sampling within the calibrated weights. This is consistent with common applied practice for raked non-probability samples, but is technically a model-assisted approximation rather than design-based variance. See `vignette("creating-survey-objects")` for details and limitations.

### See Also

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

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

### Examples

```
# Minimal: pre-computed calibration weights from an external tool
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)
```

---

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, JK<sub>n</sub>), 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"),
  scale = NULL,
  rscales = NULL,
  fpc = NULL,
  fpctype = c("fraction", "correction"),
  mse = TRUE
)
```

**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" (delete-1 jackknife, stratified), "JKn" (delete-1 jackknife with varying replication counts), "BRR" (balanced repeated replication), "Fay" (Fay's method, a modified BRR), "bootstrap", "ACS" (used in American Community Survey), "successive-difference", or "other" (user-specified scale). Case-sensitive.
scale	Numeric. Scaling factor applied to the replicate variance formula. If NULL (default), computed automatically from type and the number of replicates: $(R-1)/R$ for jackknife methods, $1/4$ for BRR/Fay, $1/R$ for bootstrap/ACS, $2/R$ for successive-difference, 1 for other.
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.
fpc	<tidy-select> Finite population correction column (a single column). Used by some replicate methods to adjust the variance estimator. NULL means no FPC correction.
fpctype	Character. How fpc is interpreted: "fraction" (sampling fraction, 0–1) or "correction" (multiplier for the replicate variance). Default "fraction". Case-sensitive.
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.

**Value**

A survey\_replicate object.

### 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  $\times$  80 replicates), this is roughly `nrow * n_replicates * 8 bytes per call` (~363 MB for ACS Wyoming  $\times$  80). If you are estimating many variables, this is repeated for each call. This behaviour matches the survey package reference implementation.

### References

Judkins, D.R. (1990) Fay's method for variance estimation. *Journal of the American Statistical Association* **85**(410), 895–904.

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

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\\_survey\(\)](#), [as\\_survey\\_nonprob\(\)](#), [as\\_survey\\_twophase\(\)](#), [survey\\_data\(\)](#), [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"
)

```

---

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 <code>survey_base</code> ) representing the Phase 1 design. Accepts <code>survey_taylor</code> or <code>survey_replicate</code> objects. Its <code>@data</code> must contain ALL rows from both phases, plus a logical indicator column for Phase 2 membership. Create with <code>as_survey()</code> or <code>as_survey_replicate()</code> .
ids2	<code>&lt;tidy-select&gt;</code> Phase 2 cluster ID column(s). For single-stage Phase 2: <code>ids2 = psu2</code> . For multi-stage: <code>ids2 = c(psu2, ssu2)</code> . Omit if Phase 2 has no within-stratum clustering.
strata2	<code>&lt;tidy-select&gt;</code> Phase 2 stratification column (a single column). Optional.
probs2	<code>&lt;tidy-select&gt;</code> 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). 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.

## Details

### Variance methods:

- "full" — Full two-phase variance formula. Accounts for variability in both phases. Requires Phase 2 design information (probs2, ids2, strata2) when Phase 2 is not a simple random subsample. If none of these are provided, a warning is issued and Phase 2 selection is treated as SRS within Phase 1 strata.
- "approx" — Approximation that ignores Phase 1 sampling variability. Faster but less accurate than "full" when the Phase 1 sampling fraction is non-negligible.
- "simple" — Treats Phase 2 as a single-phase design, ignoring Phase 1. Only valid when Phase 1 is a census (no sampling). Issues a warning when Phase 1 has PSU cluster variables, because this understates variance for clustered designs.

## Value

A survey\_twophase object.

## 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, [as\\_survey\\_replicate\(\)](#) for replicate-weight designs

Other constructors: [as\\_survey\(\)](#), [as\\_survey\\_nonprob\(\)](#), [as\\_survey\\_replicate\(\)](#), [survey\\_data\(\)](#), [survey\\_glm\(\)](#), [survey\\_glm\\_fit\(\)](#), [survey\\_nonprob\(\)](#), [survey\\_replicate\(\)](#), [survey\\_taylor\(\)](#), [survey\\_twophase\(\)](#)

## Examples

```
# Minimal two-phase design: Phase 1 = full cohort, Phase 2 = random subset
df <- data.frame(
  id      = 1:20,
  wt      = rep(2, 20),
```

```

    in_phase2 = c(rep(TRUE, 10), rep(FALSE, 10)),
    y         = rnorm(20)
  )
phase1 <- as_survey(df, ids = id, weights = wt)
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),
  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)
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`, or `survey_twophase` 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`, or `survey_twophase` 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.

**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`, or `survey_twophase` object.

**Details**

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

**Value**

A `srvyr::tbl_svy` object.

**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)
}
```

---

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**

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

**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\\_metadata\(\)](#), [extract\\_missing\\_codes\(\)](#), [extract\\_question\\_preface\(\)](#), [extract\\_sata\(\)](#), [extract\\_universe\(\)](#), [extract\\_val\\_labels\(\)](#), [extract\\_var\\_label\(\)](#), [extract\\_var\\_note\(\)](#), [infer\\_question\\_prefaces\(\)](#), [set\\_missing\\_codes\(\)](#), [set\\_question\\_preface\(\)](#), [set\\_sata\(\)](#), [set\\_universe\(\)](#), [set\\_val\\_labels\(\)](#), [set\\_var\\_label\(\)](#), [set\\_var\\_note\(\)](#), [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, bpxsyl)
```

---

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 <code>survey_glm_fit</code> object from <code>survey_glm()</code> .
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 <code>estimate = NA</code> and <code>reference_row = TRUE</code> . Default TRUE.
n	Logical. If TRUE, adds an <code>n_obs</code> column with the unweighted observation count per term. Default FALSE.
statistic	Logical. If TRUE (default), includes the <code>statistic</code> (t-statistic) column. Set to FALSE to drop it.
exponentiate	Logical. If TRUE, exponentiates <code>estimate</code> , <code>conf_low</code> , and <code>conf_high</code> . <code>std_error</code> is left on the log scale (matching broom convention). Fires <code>surveycore_warning_exponentiate_nonlog</code> when the model link is not log-based. Default FALSE.
interaction_sep	Character scalar. Separator for interaction term labels. Default <code>" * "</code> .
...	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: `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(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)
```

---

extract_metadata	<i>Extract All Metadata for Variables</i>
------------------	---

---

## 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: <code>tidyselect::starts_with()</code> , <code>tidyselect::all_of()</code> , <code>tidyselect::any_of()</code> , <code>tidyselect::matches()</code> , etc. If empty, returns metadata for all variables. Use <code>tidyselect::any_of()</code> 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.

## 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_missing_codes()`, `extract_question_preface()`, `extract_sata()`, `extract_universe()`, `extract_val_labels()`, `extract_var_label()`, `extract_var_note()`, `infer_question_prefaces()`, `set_missing_codes()`, `set_question_preface()`, `set_sata()`, `set_universe()`, `set_val_labels()`, `set_var_label()`, `set_var_note()`, `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: <code>tidyselect::starts_with()</code> , <code>tidyselect::all_of()</code> , <code>tidyselect::any_of()</code> , <code>tidyselect::matches()</code> , etc. If empty, returns metadata for all variables. Use <code>tidyselect::any_of()</code> 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.

### 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\\_metadata\(\)](#), [extract\\_question\\_preface\(\)](#), [extract\\_sata\(\)](#), [extract\\_universe\(\)](#), [extract\\_val\\_labels\(\)](#), [extract\\_var\\_label\(\)](#), [extract\\_var\\_note\(\)](#), [infer\\_question\\_prefaces\(\)](#), [set\\_missing\\_codes\(\)](#), [set\\_question\\_preface\(\)](#), [set\\_sata\(\)](#), [set\\_universe\(\)](#), [set\\_val\\_labels\(\)](#), [set\\_var\\_label\(\)](#), [set\\_var\\_note\(\)](#), [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: <code>tidyselect::starts_with()</code> , <code>tidyselect::all_of()</code> , <code>tidyselect::any_of()</code> , <code>tidyselect::matches()</code> , etc. If empty, returns metadata for all variables. Use <code>tidyselect::any_of()</code> 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_metadata()`, `extract_missing_codes()`, `extract_sata()`, `extract_universe()`, `extract_val_labels()`, `extract_var_label()`, `extract_var_note()`, `infer_question_prefaces()`, `set_missing_codes()`, `set_question_preface()`, `set_sata()`, `set_universe()`, `set_val_labels()`, `set_var_label()`, `set_var_note()`, `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_sata	<i>Extract SATA (Select-All-That-Apply) Flags</i>
--------------	---

---

### 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: <a href="#">tidyselect::starts_with()</a> , <a href="#">tidyselect::all_of()</a> , <a href="#">tidyselect::any_of()</a> , 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\\_metadata\(\)](#), [extract\\_missing\\_codes\(\)](#), [extract\\_question\\_preface\(\)](#), [extract\\_universe\(\)](#), [extract\\_val\\_labels\(\)](#), [extract\\_var\\_label\(\)](#), [extract\\_var\\_note\(\)](#), [infer\\_question\\_prefaces\(\)](#), [set\\_missing\\_codes\(\)](#), [set\\_question\\_preface\(\)](#), [set\\_sata\(\)](#), [set\\_universe\(\)](#), [set\\_val\\_labels\(\)](#), [set\\_var\\_label\(\)](#), [set\\_var\\_note\(\)](#), [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_universe	<i>Extract Universe Descriptions</i>
------------------	--------------------------------------

---

### 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: <a href="#">tidyselect::starts_with()</a> , <a href="#">tidyselect::all_of()</a> , <a href="#">tidyselect::any_of()</a> , <a href="#">tidyselect::matches()</a> , etc. If empty, returns metadata for all variables. Use <a href="#">tidyselect::any_of()</a> 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\\_metadata\(\)](#), [extract\\_missing\\_codes\(\)](#), [extract\\_question\\_preface\(\)](#), [extract\\_sata\(\)](#), [extract\\_val\\_labels\(\)](#), [extract\\_var\\_label\(\)](#), [extract\\_var\\_note\(\)](#), [infer\\_question\\_prefaces\(\)](#), [set\\_missing\\_codes\(\)](#), [set\\_question\\_preface\(\)](#), [set\\_sata\(\)](#), [set\\_universe\(\)](#), [set\\_val\\_labels\(\)](#), [set\\_var\\_label\(\)](#), [set\\_var\\_note\(\)](#), [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	<i>Extract Value Labels</i>
--------------------	-----------------------------

---

### 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: <code>tidyselect::starts_with()</code> , <code>tidyselect::all_of()</code> , <code>tidyselect::any_of()</code> , <code>tidyselect::matches()</code> , etc. If empty, returns metadata for all variables. Use <code>tidyselect::any_of()</code> 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.

### 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\\_metadata\(\)](#), [extract\\_missing\\_codes\(\)](#), [extract\\_question\\_preface\(\)](#), [extract\\_sata\(\)](#), [extract\\_universe\(\)](#), [extract\\_var\\_label\(\)](#), [extract\\_var\\_note\(\)](#), [infer\\_question\\_prefaces\(\)](#), [set\\_missing\\_codes\(\)](#), [set\\_question\\_preface\(\)](#), [set\\_sata\(\)](#), [set\\_universe\(\)](#), [set\\_val\\_labels\(\)](#), [set\\_var\\_label\(\)](#), [set\\_var\\_note\(\)](#), [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\_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: <code>tidyselect::starts_with()</code> , <code>tidyselect::all_of()</code> , <code>tidyselect::any_of()</code> , <code>tidyselect::matches()</code> , etc. If empty, returns metadata for all variables. Use <code>tidyselect::any_of()</code> 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\\_metadata\(\)](#), [extract\\_missing\\_codes\(\)](#), [extract\\_question\\_preface\(\)](#), [extract\\_sata\(\)](#), [extract\\_universe\(\)](#), [extract\\_val\\_labels\(\)](#), [extract\\_var\\_note\(\)](#), [infer\\_question\\_prefaces\(\)](#), [set\\_missing\\_codes\(\)](#), [set\\_question\\_preface\(\)](#), [set\\_sata\(\)](#), [set\\_universe\(\)](#), [set\\_val\\_labels\(\)](#), [set\\_var\\_label\(\)](#), [set\\_var\\_note\(\)](#), [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	<i>Extract Analyst Notes</i>
------------------	------------------------------

---

## 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: <a href="#">tidyselect::starts_with()</a> , <a href="#">tidyselect::all_of()</a> , <a href="#">tidyselect::any_of()</a> , <a href="#">tidyselect::matches()</a> , etc. If empty, returns metadata for all variables. Use <a href="#">tidyselect::any_of()</a> 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\\_metadata\(\)](#), [extract\\_missing\\_codes\(\)](#), [extract\\_question\\_preface\(\)](#), [extract\\_sata\(\)](#), [extract\\_universe\(\)](#), [extract\\_val\\_labels\(\)](#), [extract\\_var\\_label\(\)](#), [infer\\_question\\_prefaces\(\)](#), [set\\_missing\\_codes\(\)](#), [set\\_question\\_preface\(\)](#), [set\\_sata\(\)](#), [set\\_universe\(\)](#), [set\\_val\\_labels\(\)](#), [set\\_var\\_label\(\)](#), [set\\_var\\_note\(\)](#), [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)
```

---

from_svydesign	<i>Convert a survey Package Design to a surveycore Design Object</i>
----------------	--

---

### Description

Converts a survey package design object (svydesign, svrepdesign, or twophase) to the corresponding surveycore S7 object. The data, design variables, and replicate weights 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`, or `srvyr::tbl_svy` object.

### 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.

### 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 = ~sdmvpstu, weights = ~wtint2yr, strata = ~sdmvsra,
    data = nhanes_2017, nest = TRUE
  )
  d <- from_svydesign(sv)
  survey_data(d)
}
```

---

from_tbl_svy	<i>Convert an srvyr tbl_svy to a surveycore Design Object</i>
--------------	---

---

### 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 for Survey GLM Fits***Description**

Rao-Scott design-based ANOVA 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 <a href="#">survey_glm_fit</a> , a list of <a href="#">survey_glm_fit</a> objects, or a survey design ( <a href="#">survey_base</a> subclass).
formula	A model formula (e.g. $y \sim x1 + x2$ ). Only used when object is a survey design. Passed through to <a href="#">survey_glm()</a> ; supplying formula alongside response / predictors is rejected by <a href="#">survey_glm()</a> 's validator.
response	Character string naming the outcome variable. Only used when object is a survey design. Forwarded to <a href="#">survey_glm()</a> .
predictors	Character vector of predictor variable names. Only used when object is a survey design. Forwarded to <a href="#">survey_glm()</a> .
...	Additional arguments forwarded to <a href="#">survey_glm()</a> when object is a survey design (e.g. <code>family</code> , <code>na.action</code> , <code>quiet</code> ). For fit or list inputs, ... must be empty — any extras error via <code>rlang::check_dots_empty()</code> 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 <a href="#">survey_glm_fit</a> or a survey design (reducing to single-model mode); ignored with warning <code>surveycore_warning_anova_null_ignored</code> 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 <code>sqrt(.Machine\$double.eps)</code> .
decimals	Integer(1) or NULL. Round double output columns.
label_vars	Logical(1). When TRUE, compose term-row labels from <code>@metadata@variable_labels</code> 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.

### See Also

Other analysis: [clean\(\)](#), [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

```
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 and survey_replicate only; other design classes raise surveycore_error_polychoric_design_unsupported.
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 (ordered factor, unordered factor,

or integer with  $\leq 10$  distinct values) — non-ordinal columns raise `surveycore_error_polychoric_req`. For `method = "polyserial"`, each pair is canonicalized by type (one ordinal

- one continuous); logical / character / high-cardinality integer columns raise `surveycore_error_polyserial_canonicalization_ambiguous`.

<code>group</code>	<code>&lt;tidy-select&gt;</code> Optional grouping variable(s). Combined with any grouping set by <code>group_by()</code> . Default <code>NULL</code> .
<code>format</code>	"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 <code>group</code> is active, group columns are prepended in both formats. Case-sensitive.
<code>redundant</code>	Logical. If <code>FALSE</code> (default), each pair appears once (lower triangle: pairs where <code>var1</code> precedes <code>var2</code> in input order). If <code>TRUE</code> , both (A, B) and (B, A) are included (full directed pairs). Only affects long format; wide format always shows the full symmetric matrix.
<code>diagonal</code>	Logical. If <code>FALSE</code> (default), self-correlations are excluded (diagonal is <code>NA</code> in wide format). If <code>TRUE</code> , self-correlations (r equals 1) are included.
<code>variance</code>	<code>NULL</code> 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.
<code>conf_level</code>	Numeric scalar in (0, 1). Default 0.95.
<code>n_weighted</code>	Logical. If <code>TRUE</code> , add an <code>n_weighted</code> column with the pairwise sum of weights (both variables non-NA). Default <code>FALSE</code> .
<code>decimals</code>	Integer or <code>NULL</code> . If an integer, rounds all numeric output columns (e.g., r, se, ci_low, ci_high) to this many decimal places. Default <code>NULL</code> (no rounding).
<code>min_cell_n</code>	Integer. Minimum pairwise unweighted count before <code>surveycore_warning_small_cell</code> fires. Default 30L (AAPOR guidance).
<code>na.rm</code>	Logical. If <code>TRUE</code> (default), pairs use complete cases for each variable pair separately (pairwise deletion), and observations where any group variable is <code>NA</code> are excluded from the output. If <code>FALSE</code> , pairwise complete cases are still used for each variable pair, and observations where a group variable is <code>NA</code> are collected into their own group row in the output (appearing after all non-NA group rows).
<code>label_values</code>	Logical. If <code>TRUE</code> (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.
<code>label_vars</code>	Logical. If <code>TRUE</code> (default) and variable labels are set in metadata, <code>var1/var2</code> columns (long) and <code>variable</code> column (wide) show labels instead of raw names. Falls back to raw names if labels are unset.
<code>name_style</code>	"surveycore" (default) or "broom". When "broom", renames <code>r</code> → <code>estimate</code> , <code>se</code> → <code>std.error</code> , etc. Only affects long format.
<code>method</code>	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 <code>base::match.arg()</code> signal.

...	Unused. Reserved so that <code>.id</code> and <code>.if_missing_var</code> remain named-only when a <code>survey_collection</code> is passed as <code>design</code> .
<code>.id</code>	Character(1) or NULL. Column name used to identify each survey when <code>design</code> is a <code>survey_collection</code> . For collection inputs, NULL (the default) resolves to the collection's stored <code>@id</code> property. Pass a non-NULL value to override. Ignored when <code>design</code> is a single survey.
<code>.if_missing_var</code>	"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 <code>@if_missing_var</code> property. Pass a non-NULL value to override. Ignored when <code>design</code> 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` — Pearson correlation coefficient.
- Variance columns (`se, var, cv, ci_low, ci_high, moe, deff`) — only those requested via `variance`.

- `p_value` — two-tailed p-value.
- `statistic` — t-statistic.
- `df` — degrees of freedom for the t-test (n minus 2).
- `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](https://ssrn.com/abstract=6580480)
- Olsson, U. (1979). Maximum likelihood estimation of the polychoric correlation coefficient. *Psychometrika*, 44(4), 443-460.

## See Also

Other analysis: `clean()`, `get_anova()`, `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)
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

<code>design</code>	A survey design object: <code>survey_taylor</code> , <code>survey_replicate</code> , <code>survey_twophase</code> , or <code>survey_nonprob</code> . Also accepts a <a href="#">survey_collection</a> .
<code>x</code>	<code>&lt;tidy-select&gt;</code> 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 $\text{covariance} = \widehat{\text{Var}}(x) - \text{Var}_{\text{hat}}(x)$ (the design-based variance – <b>not</b> 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 <a href="#">survey_collection</a> . 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:  $\text{ci\_low} = \text{covariance} - z * \text{se}$ ,  $\text{ci\_high} = \text{covariance} + z * \text{se}$ , where  $z = \text{qnorm}((1 + \text{conf\_level}) / 2)$ . The bounds are **not clamped**. Covariance is unbounded —  $\text{ci\_low}$  and  $\text{ci\_high}$  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  $SE\_SRS(cov) = \sqrt{(\text{Var}(x) * \text{Var}(y) + cov^2) / (n - 1)}$ . 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: `clean()`, `get_anova()`, `get_corr()`, `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
  )
  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,
  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: <code>survey_taylor</code> , <code>survey_replicate</code> , <code>survey_twophase</code> , or <code>survey_nonprob</code> .
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 <code>group_by()</code> . Default NULL.
covariates	Character vector of additional model terms as strings. Supports interactions (" <code>age * gender</code> "), polynomials (" <code>poly(edu, 2)</code> "), and transformations (" <code>log(income)</code> "). When provided, forces the <code>marginaleffects</code> estimation path. Default NULL.
ref_level	Character(1). Reference level of <code>treats</code> 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 <code>stats::p.adjust()</code> . Options: " <code>holm</code> ", " <code>hochberg</code> ", " <code>hommel</code> ", " <code>bonferroni</code> ", " <code>BH</code> ", " <code>BY</code> ", " <code>fdr</code> ", " <code>none</code> ". NULL = no adjustment. When <code>group</code> is active, adjustment is applied independently within each group.
show_means	Logical. If TRUE (default), includes a mean column and a reference row with <code>estimate = 0</code> . Subject to link-scale suppression (see Details).
show_pct_change	Logical. If TRUE, includes a <code>pct_change</code> column: <code>estimate / reference_mean</code> . Subject to link-scale suppression (see Details). Default FALSE.
scale	Character(1). " <code>ame</code> " (default): average marginal effects on the response scale. " <code>link</code> ": 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 " <code>se</code> ", " <code>ci</code> ". Controls which uncertainty columns appear. Default " <code>ci</code> ".
conf_level	Numeric(1) in (0, 1). Confidence level. Default 0.95.
min_cell_n	Integer(1). Minimum unweighted cell size before <code>surveycore_warning_small_cell</code> fires. Default 30L.
n_weighted	Logical. If TRUE, includes an <code>n_weighted</code> column with sum of weights per treatment level. Default FALSE.
decimals	Integer(1) or NULL. If non-NULL, rounds numeric output columns. <code>pct_change</code> is rounded to <code>decimals + 2</code> . Default NULL.
na.rm	Logical. If TRUE (default), rows with NA in <code>x</code> , <code>treats</code> , or <code>group</code> are dropped before fitting. If FALSE, NA values cause an error.
label_values	Logical. If TRUE (default), the <code>treats</code> and <code>group</code> 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 <code>survey_glm()</code> . Common uses: <code>family = quasibinomial()</code> .
.id	Character(1) or NULL. Column name used to identify each survey when design is a <code>survey_collection</code> . 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, no group): extracts coefficients directly from `clean()`. The intercept is the reference group mean; treatment coefficients are differences from reference.
- **Marginal effects 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`. Use `meta()` to access design type, family, reference level, and other metadata.

**See Also**

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

**Examples**

```
library(marginaleffects)

# 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\_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: <code>survey_taylor</code> , <code>survey_replicate</code> , <code>survey_twophase</code> , or <code>survey_nonprob</code> .
x	<code>&lt;tidy-select&gt;</code> One or more categorical variables. Bare names or tidy-select helpers (e.g., <code>c(q1, q2, q3)</code> ). When two or more variables are selected, multi-variable stacking mode is activated (see Details).
...	Additional arguments passed to tidy-select (future-proof; currently unused).
group	<code>&lt;tidy-select&gt;</code> Optional grouping variable(s). Combined with any grouping set by <code>group_by()</code> . Default <code>NULL</code> .
names_to	Character(1). Column name for the variable identifier in multi-variable mode. Default <code>"name"</code> .
values_to	Character(1). Column name for the response value in multi-variable mode. Default <code>"value"</code> .
variance	<code>NULL</code> or a character vector of one or more of <code>"se"</code> , <code>"ci"</code> , <code>"var"</code> , <code>"cv"</code> , <code>"moe"</code> , <code>"deff"</code> . Controls which uncertainty columns appear in the output. Default <code>NULL</code> (no uncertainty columns).
conf_level	Numeric scalar in (0, 1). Confidence level for intervals. Default <code>0.95</code> .
n_weighted	Logical. If <code>TRUE</code> , add an <code>n_weighted</code> column with the sum of weights (estimated population count) per cell. Default <code>FALSE</code> .
decimals	Integer or <code>NULL</code> . If an integer, rounds all numeric output columns (e.g., <code>pct</code> , <code>se</code> , <code>ci_low</code> , <code>ci_high</code> ) to this many decimal places. Default <code>NULL</code> (no rounding).
min_cell_n	Integer. Minimum unweighted cell count before <code>surveycore_warning_small_cell</code> fires. Default <code>30L</code> (AAPOR guidance).
na.rm	Logical. If <code>TRUE</code> (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 <code>FALSE</code> , 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 <code>TRUE</code> (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 <code>TRUE</code> (default), use variable labels from metadata in the <code>names_to</code> column (multi-variable mode only). Falls back to the raw variable name when no label is set.
name_style	<code>"surveycore"</code> (default) or <code>"broom"</code> . When <code>"broom"</code> , renames <code>pct</code> → <code>estimate</code> , <code>se</code> → <code>std.error</code> , etc.

<code>.id</code>	Character(1) or NULL. Column name used to identify each survey when design is a <a href="#">survey_collection</a> . For collection inputs, NULL (the default) resolves to the collection's stored <code>@id</code> property. Pass a non-NULL value to override. Ignored when design is a single survey.
<code>.if_missing_var</code>	"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 <code>@if_missing_var</code> 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.

## See Also

Other analysis: [clean\(\)](#), [get\\_anova\(\)](#), [get\\_corr\(\)](#), [get\\_covariance\(\)](#), [get\\_diffs\(\)](#), [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: <code>survey_taylor</code> , <code>survey_replicate</code> , <code>survey_twophase</code> , or <code>survey_nonprob</code> .
x	<code>&lt;tidy-select&gt;</code> A single unquoted numeric variable name. Must resolve to exactly one numeric column.
group	<code>&lt;tidy-select&gt;</code> Optional grouping variable(s). Combined with any grouping set by <code>group_by()</code> . Default <code>NULL</code> .
variance	<code>NULL</code> or a character vector of one or more of <code>"se"</code> , <code>"ci"</code> , <code>"var"</code> , <code>"cv"</code> , <code>"moe"</code> , <code>"deff"</code> . Controls which uncertainty columns appear in the output. Default <code>"ci"</code> .
conf_level	Numeric scalar in (0, 1). Confidence level for intervals. Default <code>0.95</code> .
n_weighted	Logical. If <code>TRUE</code> , add an <code>n_weighted</code> column with the sum of weights for non-NA observations in each group. Default <code>FALSE</code> .
decimals	Integer or <code>NULL</code> . If an integer, rounds all numeric output columns (e.g., <code>mean</code> , <code>se</code> , <code>ci_low</code> , <code>ci_high</code> ) to this many decimal places. Default <code>NULL</code> (no rounding).
min_cell_n	Integer. Minimum unweighted cell count before <code>surveycore_warning_small_cell</code> fires. Default <code>30L</code> (AAPOR guidance).
na.rm	Logical. If <code>TRUE</code> (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 <code>FALSE</code> , 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 uniformity; has no visible effect since <code>get_means()</code> output contains no categorical value cells. Default <code>TRUE</code> .
label_vars	Logical. Accepted for API uniformity; has no visible effect since <code>get_means()</code> output contains no variable-name value cells. Default <code>TRUE</code> .
name_style	<code>"surveycore"</code> (default) or <code>"broom"</code> . When <code>"broom"</code> , renames <code>mean</code> → <code>estimate</code> , <code>se</code> → <code>std.error</code> , etc.
...	Unused. Reserved so that <code>.id</code> and <code>.if_missing_var</code> remain named-only when a <code>survey_collection</code> is passed as <code>design</code> .
.id	Character(1) or <code>NULL</code> . Column name used to identify each survey when <code>design</code> is a <code>survey_collection</code> . For collection inputs, <code>NULL</code> (the default) resolves to the collection's stored <code>@id</code> property. Pass a non- <code>NULL</code> value to override. Ignored when <code>design</code> is a single survey.
.if_missing_var	<code>"error"</code> , <code>"skip"</code> , or <code>NULL</code> . How to handle surveys in a collection that lack one of the requested NSE variables. For collection inputs, <code>NULL</code> (the default) resolves to the collection's stored <code>@if_missing_var</code> property. Pass a non- <code>NULL</code> value to override. Ignored when <code>design</code> 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.
- 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)$variable`, not as a column. Use `meta(result)` to access design type, variable labels, and other metadata.

### See Also

Other analysis: `clean()`, `get_anova()`, `get_corr()`, `get_covariance()`, `get_diffs()`, `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: <code>survey_taylor</code> , <code>survey_replicate</code> , <code>survey_twophase</code> , or <code>survey_nonprob</code> .
x	<code>&lt;tidy-select&gt;</code> A single unquoted numeric variable name for the outcome variable.
by	<code>&lt;tidy-select&gt;</code> A single unquoted variable name for the grouping variable. Must have at least 2 active levels.
group	<code>&lt;tidy-select&gt;</code> Optional subgroup variable(s). When supplied, pairwise comparisons are run within each group stratum. P-value adjustment is applied separately per stratum. Default <code>NULL</code> .
pval_adj	Character(1). P-value adjustment method passed to <code>stats::p.adjust()</code> . Default <code>"holm"</code> . Use <code>"none"</code> for unadjusted p-values. Error: <code>surveycore_error_invalid_pval_adj</code> .
conf_level	Numeric(1). Confidence level strictly in (0, 1). Default <code>0.95</code> .
variance	Character. Which uncertainty columns to include. Valid values: <code>"se"</code> , <code>"ci"</code> . Default <code>"ci"</code> .
na.rm	Logical(1). Accepted for API uniformity. Default <code>TRUE</code> .
min_cell_n	Integer(1). Warn for small cells. Default <code>30L</code> .
decimals	Integer(1) or <code>NULL</code> . Round all double output columns. Default <code>NULL</code> .
label_values	Logical(1). Convert by/group codes to value labels. Default <code>TRUE</code> .
label_vars	Logical(1). Accepted for API uniformity; no visible effect. Default <code>TRUE</code> .
name_style	Character(1). <code>"surveycore"</code> (default) or <code>"broom"</code> .
...	Unused. Reserved so that <code>.id</code> and <code>.if_missing_var</code> remain named-only when a <code>survey_collection</code> is passed as <code>design</code> .
.id	Character(1) or <code>NULL</code> . Column name used to identify each survey when <code>design</code> is a <code>survey_collection</code> . For collection inputs, <code>NULL</code> (the default) resolves to the collection's stored <code>@id</code> property. Pass a non- <code>NULL</code> value to override. Ignored when <code>design</code> is a single survey.
.if_missing_var	<code>"error"</code> , <code>"skip"</code> , or <code>NULL</code> . How to handle surveys in a collection that lack one of the requested NSE variables. For collection inputs, <code>NULL</code> (the default) resolves to the collection's stored <code>@if_missing_var</code> property. Pass a non- <code>NULL</code> value to override. Ignored when <code>design</code> 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: `clean()`, `get_anova()`, `get_corr()`, `get_covariance()`, `get_diffs()`, `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: <code>survey_taylor</code> , <code>survey_replicate</code> , <code>survey_twophase</code> , or <code>survey_nonprob</code> .
x	<code>&lt;tidy-select&gt;</code> A single unquoted numeric variable name. Must resolve to exactly one numeric column.
probs	Numeric vector of probabilities in (0, 1). Default <code>c(0.25, 0.5, 0.75)</code> (IQR + median).
group	<code>&lt;tidy-select&gt;</code> Optional grouping variable(s). Combined with any grouping set by <code>group_by()</code> . Default <code>NULL</code> .
variance	<code>NULL</code> or a character vector from <code>"se"</code> , <code>"ci"</code> , <code>"var"</code> , <code>"cv"</code> , <code>"moe"</code> , <code>"deff"</code> . Controls which uncertainty columns appear in the output. CIs use the Woodruff (1952) back-transformation method and are <b>not</b> symmetric around the estimate. <code>"deff"</code> is always NA for quantiles (no closed-form SRS SE). Default <code>"ci"</code> .
conf_level	Numeric scalar in (0, 1). Confidence level for Woodruff intervals. Default <code>0.95</code> .
n_weighted	Logical. If <code>TRUE</code> , add an <code>n_weighted</code> column with the sum of weights for non-NA observations in each group. Default <code>FALSE</code> .
decimals	Integer or <code>NULL</code> . If an integer, rounds all numeric output columns (e.g., <code>estimate</code> , <code>se</code> , <code>ci_low</code> , <code>ci_high</code> ) to this many decimal places. Default <code>NULL</code> (no rounding).
min_cell_n	Integer. Minimum unweighted cell count before <code>surveycore_warning_small_cell</code> fires. Default <code>30L</code> (AAPOR guidance).
na.rm	Logical. If <code>TRUE</code> (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 <code>FALSE</code> , 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 uniformity; has no visible effect on <code>get_quantiles()</code> output. Default <code>TRUE</code> .
label_vars	Logical. Accepted for API uniformity; has no visible effect on <code>get_quantiles()</code> output. Default <code>TRUE</code> .
name_style	<code>"surveycore"</code> (default) or <code>"broom"</code> . When <code>"broom"</code> , renames <code>se</code> → <code>std.error</code> , <code>ci_low</code> → <code>conf.low</code> , <code>ci_high</code> → <code>conf.high</code> . The estimate column is unchanged.
...	Unused. Reserved so that <code>.id</code> and <code>.if_missing_var</code> remain named-only when a <code>survey_collection</code> is passed as <code>design</code> .
.id	Character(1) or <code>NULL</code> . Column name used to identify each survey when <code>design</code> is a <code>survey_collection</code> . For collection inputs, <code>NULL</code> (the default) resolves to the collection's stored <code>@id</code> property. Pass a non- <code>NULL</code> value to override. Ignored when <code>design</code> 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 non-NA observations used in the estimate.
- `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: `clean()`, `get_anova()`, `get_corr()`, `get_covariance()`, `get_diffs()`, `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: <code>survey_taylor</code> , <code>survey_replicate</code> , <code>survey_twophase</code> , or <code>survey_nonprob</code> .
numerator	<code>&lt;tidy-select&gt;</code> A single unquoted numeric variable name for the numerator. Must resolve to exactly one numeric column.
denominator	<code>&lt;tidy-select&gt;</code> 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	<code>&lt;tidy-select&gt;</code> Optional grouping variable(s). Combined with any grouping set by <code>group_by()</code> . Rows where the grouping variable is NA are excluded from all groups and do not appear in the output. This matches <code>dplyr::group_by()</code> 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 uniformity; has no visible effect on get_ratios() output. 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 <a href="#">survey_collection</a> . 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.

### See Also

Other analysis: `clean()`, `get_anova()`, `get_corr()`, `get_covariance()`, `get_diffs()`, `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, 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: <code>survey_taylor</code> , <code>survey_replicate</code> , <code>survey_twophase</code> , or <code>survey_nonprob</code> .
x	<tidy-select> Optional single unquoted numeric variable name. When NULL (default), estimates the population size (sum of weights). When supplied, estimates the weighted sum (sum of $w_i * x_i$ ).
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 <code>get_totals(d)</code> (no variable), equals the total column and is included for API uniformity. For variable mode, adds the sum of weights for non-NA observations. 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).
label_values	Logical. Accepted for API uniformity. Default TRUE.
label_vars	Logical. Accepted for API uniformity. Default TRUE.
name_style	"surveycore" (default) or "broom".
...	Unused. Reserved so that <code>.id</code> and <code>.if_missing_var</code> remain named-only when a <code>survey_collection</code> is passed as <code>design</code> .
.id	Character(1) or NULL. Column name used to identify each survey when <code>design</code> is a <a href="#">survey_collection</a> . For collection inputs, NULL (the default) resolves to the collection's stored <code>@id</code> property. Pass a non-NULL value to override. Ignored when <code>design</code> 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 <code>@if_missing_var</code> property. Pass a non-NULL value to override. Ignored when <code>design</code> is a single survey.

**Value**

A `survey_totals` tibble (also inheriting `survey_result`). Columns:

- `[group_cols...]` — group variable columns (when active), first.
- `total` — the weighted sum estimate.
- Variance columns — only those requested via `variance`.
- `n` — unweighted count (omitted in no-variable mode).
- `n_weighted` — sum of weights (only when requested).

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

**See Also**

Other analysis: [clean\(\)](#), [get\\_anova\(\)](#), [get\\_corr\(\)](#), [get\\_covariance\(\)](#), [get\\_diffs\(\)](#), [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)
```

---

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::svytest()` 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: <code>survey_taylor</code> , <code>survey_replicate</code> , <code>survey_twophase</code> , or <code>survey_nonprob</code> .
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 <code>group_by()</code> . Default <code>NULL</code> .
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: <code>c("se", "ci")</code> .
na.rm	Logical(1). Accepted for API uniformity with other <code>get_*()</code> functions. NA rows in x or by are always excluded (the GLM requires complete cases). Default <code>TRUE</code> .
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 <code>NULL</code> . Round all double output columns to this many decimal places. <code>NULL</code> = no rounding. Default <code>NULL</code> .
label_values	Logical(1). When <code>TRUE</code> (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 <code>TRUE</code> .

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.
...	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 <a href="#">survey_collection</a> . 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: `clean()`, `get_anova()`, `get_corr()`, `get_covariance()`, `get_diffs()`, `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)
```

### 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: <code>survey_taylor</code> , <code>survey_replicate</code> , <code>survey_twophase</code> , or <code>survey_nonprob</code> . Also accepts a <a href="#">survey_collection</a> .
x	<code>&lt;tidy-select&gt;</code> One or more unquoted numeric variable names. Must resolve to at least one numeric column; non-numeric columns are rejected (no silent drop).
group	<code>&lt;tidy-select&gt;</code> Optional grouping variable(s). Combined with any grouping set by <code>group_by()</code> . Default <code>NULL</code> .
variance	<code>NULL</code> or a character vector of one or more of <code>"se"</code> , <code>"ci"</code> , <code>"var"</code> , <code>"cv"</code> , <code>"moe"</code> , <code>"deff"</code> . Controls which uncertainty columns appear in the output. Default <code>"ci"</code> .
conf_level	Numeric scalar in (0, 1). Confidence level for intervals. Default <code>0.95</code> .
n_weighted	Logical. If <code>TRUE</code> , add an <code>n_weighted</code> column with the sum of weights for non-NA, positive-weight observations in each row's estimate. Default <code>FALSE</code> .
decimals	Integer or <code>NULL</code> . If an integer, rounds all numeric output columns to this many decimal places. Default <code>NULL</code> (no rounding).
min_cell_n	Integer. Minimum unweighted cell count before <code>surveycore_warning_small_cell</code> fires. Default <code>30L</code> (AAPOR guidance).
na.rm	Logical. If <code>TRUE</code> (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 <code>FALSE</code> , NA propagates to produce NaN estimates.
na_handling	<code>"pairwise"</code> (default) or <code>"listwise"</code> . In multi-variable mode controls whether each focal variable uses its own complete-case set ( <code>"pairwise"</code> ) or the intersection across all focal variables ( <code>"listwise"</code> ). Ignored when <code>na.rm = FALSE</code> .

label_values	Logical. Accepted for API uniformity; 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 <a href="#">survey_collection</a> . 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:

- `[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. 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 `variance`.
- `n` — unweighted count of non-NA observations used.
- `n_weighted` — sum of weights (only when `n_weighted = TRUE`).

### See Also

Other analysis: [clean\(\)](#), [get\\_anova\(\)](#), [get\\_corr\(\)](#), [get\\_covariance\(\)](#), [get\\_diffs\(\)](#), [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)
```

**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). Detected prefaces are written to `question_preface` in the metadata and 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**

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

**Details****Detection algorithm (two passes):**

- 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  $\geq \text{min\_vars}$  members is recorded; those variables are excluded from all subsequent passes.
- LCP pass** — for remaining labelled variables ( $\geq 2$ ):
  - 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  $\geq \text{lcp\_min}$  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_metadata()`, `extract_missing_codes()`, `extract_question_preface()`, `extract_sata()`, `extract_universe()`, `extract_val_labels()`, `extract_var_label()`, `extract_var_note()`, `set_missing_codes()`, `set_question_preface()`, `set_sata()`, `set_universe()`, `set_val_labels()`, `set_var_label()`, `set_var_note()`, `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"
```

### 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", "srs", or "nonprob".

`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_preamble` (`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_preamble`, `value_labels`.

**See Also**

Other analysis: [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\(\)](#)

**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).

**sdmvpstu** 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 variable labels and value labels as R attributes, 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(nhanes_2017$riagendr, "label")` returns "Gender".
- **Value labels** ("labels" attribute): A named numeric vector mapping each code to its meaning. Example: `attr(nhanes_2017$riagendr, "labels")` returns `c(Male = 1, Female = 2)`.

**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 value labels
attr(nhanes_2017$riagendr, "labels")

# Inspect value labels for race/ethnicity
attr(nhanes_2017$ridreth3, "labels")
```

---

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\_berniesanders** Whether Bernie Sanders cares about the truth.

**cand\_truth\_pete\_buttigieg** Whether Pete Buttigieg cares about the truth.

**cand\_truth\_kamalaharris** 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\_berniesanders** Whether Bernie Sanders relies on facts.

**cand\_facts\_pete\_buttigieg** Whether Pete Buttigieg relies on facts.

**cand\_facts\_kamalaharris** 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 court-houses.

**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  $\times$  ~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, October 10–17, 2019 (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).

**hispanic** 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).

**educat** 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")
```

---

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.

**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)
```

---

<code>set_collection_id</code>	<i>Set the Identifier Column on a survey_collection</i>
--------------------------------	---

---

**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\_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**

<code>x</code>	A survey design object or a data frame.
<code>...</code>	Named arguments where the name is the variable and the value is a named atomic vector of missing codes. Supports !!! list splicing.
<code>variable</code>	A character vector of variable names. Use with <code>codes</code> .
<code>codes</code>	A list of named atomic vectors, one per element of <code>variable</code> . When <code>variable</code> 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**

Other metadata: [classify\\_question\\_type\(\)](#), [extract\\_metadata\(\)](#), [extract\\_missing\\_codes\(\)](#), [extract\\_question\\_preface\(\)](#), [extract\\_sata\(\)](#), [extract\\_universe\(\)](#), [extract\\_val\\_labels\(\)](#), [extract\\_var\\_label\(\)](#), [extract\\_var\\_note\(\)](#), [infer\\_question\\_prefaces\(\)](#), [set\\_question\\_preface\(\)](#), [set\\_sata\(\)](#), [set\\_universe\(\)](#), [set\\_val\\_labels\(\)](#), [set\\_var\\_label\(\)](#), [set\\_var\\_note\(\)](#), [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_metadata()`, `extract_missing_codes()`, `extract_question_preface()`, `extract_sata()`, `extract_universe()`, `extract_val_labels()`, `extract_var_label()`, `extract_var_note()`, `infer_question_prefaces()`, `set_missing_codes()`, `set_sata()`, `set_universe()`, `set_val_labels()`, `set_var_label()`, `set_var_note()`, `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\_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: <code>tidyselect::starts_with()</code> , <code>tidyselect::all_of()</code> , <code>tidyselect::any_of()</code> , etc. Cannot be combined with <code>variable</code> .
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\\_metadata\(\)](#), [extract\\_missing\\_codes\(\)](#), [extract\\_question\\_preface\(\)](#), [extract\\_sata\(\)](#), [extract\\_universe\(\)](#), [extract\\_val\\_labels\(\)](#), [extract\\_var\\_label\(\)](#), [extract\\_var\\_note\(\)](#), [infer\\_question\\_prefaces\(\)](#), [set\\_missing\\_codes\(\)](#), [set\\_question\\_preface\(\)](#), [set\\_universe\(\)](#), [set\\_val\\_labels\(\)](#), [set\\_var\\_label\(\)](#), [set\\_var\\_note\(\)](#), [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\_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**

Other metadata: [classify\\_question\\_type\(\)](#), [extract\\_metadata\(\)](#), [extract\\_missing\\_codes\(\)](#), [extract\\_question\\_preface\(\)](#), [extract\\_sata\(\)](#), [extract\\_universe\(\)](#), [extract\\_val\\_labels\(\)](#), [extract\\_var\\_label\(\)](#), [extract\\_var\\_note\(\)](#), [infer\\_question\\_prefaces\(\)](#), [set\\_missing\\_codes\(\)](#), [set\\_question\\_preface\(\)](#), [set\\_sata\(\)](#), [set\\_val\\_labels\(\)](#), [set\\_var\\_label\(\)](#), [set\\_var\\_note\(\)](#), [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\\_metadata\(\)](#), [extract\\_missing\\_codes\(\)](#), [extract\\_question\\_preface\(\)](#), [extract\\_sata\(\)](#), [extract\\_universe\(\)](#), [extract\\_val\\_labels\(\)](#), [extract\\_var\\_label\(\)](#), [extract\\_var\\_note\(\)](#), [infer\\_question\\_prefaces\(\)](#), [set\\_missing\\_codes\(\)](#), [set\\_question\\_preface\(\)](#), [set\\_sata\(\)](#), [set\\_universe\(\)](#), [set\\_var\\_label\(\)](#), [set\\_var\\_note\(\)](#), [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_label	<i>Set Variable Label(s)</i>
---------------	------------------------------

---

### 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\\_metadata\(\)](#), [extract\\_missing\\_codes\(\)](#), [extract\\_question\\_preface\(\)](#), [extract\\_sata\(\)](#), [extract\\_universe\(\)](#), [extract\\_val\\_labels\(\)](#), [extract\\_var\\_label\(\)](#), [extract\\_var\\_note\(\)](#), [infer\\_question\\_prefaces\(\)](#), [set\\_missing\\_codes\(\)](#), [set\\_question\\_preface\(\)](#), [set\\_sata\(\)](#), [set\\_universe\(\)](#), [set\\_val\\_labels\(\)](#), [set\\_var\\_note\(\)](#), [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	<i>Set Analyst Note(s)</i>
--------------	----------------------------

---

**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\\_metadata\(\)](#), [extract\\_missing\\_codes\(\)](#), [extract\\_question\\_preface\(\)](#), [extract\\_sata\(\)](#), [extract\\_universe\(\)](#), [extract\\_val\\_labels\(\)](#), [extract\\_var\\_label\(\)](#), [extract\\_var\\_note\(\)](#), [infer\\_question\\_prefaces\(\)](#), [set\\_missing\\_codes\(\)](#), [set\\_question\\_preface\(\)](#), [set\\_sata\(\)](#), [set\\_universe\(\)](#), [set\\_val\\_labels\(\)](#), [set\\_var\\_label\(\)](#), [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)
```

---

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  $\geq 1$ . 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)
```

**Arguments**

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

**Value**

A data.frame with all variables, including design variables.

**See Also**

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**

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

---

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 five surveycore design classes are supported.

**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 <a href="#">as_survey()</a> , <a href="#">as_survey_replicate()</a> , <a href="#">as_survey_twophase()</a> , or <a href="#">as_survey_nonprob()</a> .
formula	A model formula in standard R notation (e.g. $y \sim x1 + x2$ ). Mutually exclusive with response/predictors. If NULL and response is also NULL, errors with <code>surveycore_error_formula_missing</code> .

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.

**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  $n \times p$  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_survey()`, `as_survey_nonprob()`, `as_survey_replicate()`, `as_survey_twophase()`, `survey_data()`, `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

<code>coefficients</code>	Named numeric vector of length <code>p</code> .
<code>vcov</code>	<code>p</code> x <code>p</code> design-based variance-covariance matrix.
<code>fitted_values</code>	Numeric vector of length <code>n</code> (response scale).

residuals	Working residuals from IRLS, length $n$ .
weights	Survey weights used in fitting, length $n$ .
design	The original <a href="#">survey_base</a> survey design object.
degf	Raw design degrees of freedom (positive scalar): number of PSUs minus number of strata for Taylor designs, number of replicates minus one for replicate designs, and $n - 1$ for SRS designs. This is <i>not</i> the residual degrees of freedom used for t-statistics and confidence intervals; those are computed as $\text{degf} - (p - 1)$ where $p$ is the number of model coefficients.
family	GLM family object (e.g. <code>gaussian()</code> , <code>binomial()</code> ).
formula	Model formula.
null_deviance	Null model deviance.
deviance	Residual deviance.
df_null	Classical null df ( <code>fit\$df.null</code> from <code>stats::glm()</code> ).
df_residual	Classical residual df ( <code>fit\$df.residual</code> , i.e. $n - p$ ). Used for the deviance display; <b>not</b> the design-based residual df.
converged	Logical; whether IRLS converged.
call	The <code>survey_glm()</code> call (language object or NULL).
fit_	Internal raw <code>stats::glm()</code> result; NULL after serialisation.
term_assign	Integer vector: <code>attr(model.matrix(fit_), "assign")</code> captured at fit time. Maps design-matrix columns to formula terms (0 = intercept; positive values index <code>attr(terms(formula), "term.labels")</code> ). Required by <code>get_anova()</code> 's serialization-safe Wald path (spec §3.3.1): after <code>@fit_</code> is stripped via <code>saveRDS()</code> , 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\\_survey\(\)](#), [as\\_survey\\_nonprob\(\)](#), [as\\_survey\\_replicate\(\)](#), [as\\_survey\\_twophase\(\)](#), [survey\\_data\(\)](#), [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	<i>Survey Metadata Container</i>
-----------------	----------------------------------

---

## 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(),
  universe = list(),
  missing_codes = list(),
  sata = list(),
  transformations = list(),
  weighting_history = list()
)
```

## Arguments

<code>variable_labels</code>	A named list mapping variable names to character labels (e.g., <code>list(age = "Age in years")</code> ).
<code>value_labels</code>	A named list mapping variable names to named vectors of value labels (e.g., <code>list(sex = c(Male = 1L, Female = 2L))</code> ).
<code>question_prefaces</code>	A named list mapping variable names to shared question battery preface text.
<code>notes</code>	A named list mapping variable names to analyst notes.
<code>universe</code>	A named list mapping variable names to universe descriptions (e.g., <code>list(age = "Adults 18+")</code> ). Describes the population to which a variable applies.
<code>missing_codes</code>	A named list mapping variable names to atomic vectors of missing-value codes (e.g., <code>list(age = c(Refused = 99L, DK = 98L))</code> ).
<code>sata</code>	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.
<code>transformations</code>	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.

**Value**

A `survey_metadata` object.

**See Also**

Other metadata: `classify_question_type()`, `extract_metadata()`, `extract_missing_codes()`, `extract_question_preface()`, `extract_sata()`, `extract_universe()`, `extract_val_labels()`, `extract_var_label()`, `extract_var_note()`, `infer_question_prefaces()`, `set_missing_codes()`, `set_question_preface()`, `set_sata()`, `set_universe()`, `set_val_labels()`, `set_var_label()`, `set_var_note()`, `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
```

---

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
)
```

**Arguments**

data	A data.frame containing the survey data. Prefer <code>as_survey_replicate()</code> over calling this constructor directly.
metadata	A <code>survey_metadata</code> object. Created automatically by <code>as_survey_replicate()</code> .
variables	A named list of design specification (weights, repweights, type, scale, rscales, fpc, fpctype, mse). Set automatically by <code>as_survey_replicate()</code> .
groups	Set by <code>surveytidy</code> 's <code>group_by()</code> . Always <code>character(0)</code> in standalone <code>survey-core</code> use.
call	Language object capturing the construction call.

**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 <code>design@data[, design@variables\$repweights]</code> — 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	FPC column name or NULL.
fpctype	"fraction" or "correction".
mse	Logical. Use MSE estimates?

**See Also**

`as_survey_replicate()` to create a `survey_replicate` object.

Other constructors: `as_survey()`, `as_survey_nonprob()`, `as_survey_replicate()`, `as_survey_twophase()`, `survey_data()`, `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
)
```

### Arguments

data	A <code>data.frame</code> containing the survey data. Prefer <a href="#">as_survey()</a> over calling this constructor directly.
metadata	A <a href="#">survey_metadata</a> object. Created automatically by <a href="#">as_survey()</a> .
variables	A named list of design specification ( <code>ids</code> , <code>weights</code> , <code>strata</code> , <code>fpc</code> , <code>nest</code> , <code>probs_provided</code> ). Set automatically by <a href="#">as_survey()</a> .
groups	Set by <code>surveytidy</code> 's <code>group_by()</code> . Always <code>character(0)</code> in standalone <code>survey-core</code> use.
call	Language object capturing the construction call.

### 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\(\)](#).

**See Also**

[as\\_survey\(\)](#) to create a `survey_taylor` object.

Other constructors: [as\\_survey\(\)](#), [as\\_survey\\_nonprob\(\)](#), [as\\_survey\\_replicate\(\)](#), [as\\_survey\\_twophase\(\)](#), [survey\\_data\(\)](#), [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	<i>Two-Phase Survey Design</i>
-----------------	--------------------------------

---

**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 <code>data.frame</code> containing the survey data (all Phase 1 rows, with a logical indicator for Phase 2 membership). Prefer <a href="#">as_survey_twophase()</a> over calling this constructor directly.
metadata	A <a href="#">survey_metadata</a> object. Inherited from the Phase 1 design when using <a href="#">as_survey_twophase()</a> .
variables	A named list of design specification (phase1, phase2, subset, method). Set automatically by <a href="#">as_survey_twophase()</a> .
groups	Set by <code>surveytidy</code> 's <code>group_by()</code> . Always <code>character(0)</code> in standalone <code>survey-core</code> 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\\_survey\(\)](#), [as\\_survey\\_nonprob\(\)](#), [as\\_survey\\_replicate\(\)](#), [as\\_survey\\_twophase\(\)](#), [survey\\_data\(\)](#), [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)
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_metadata()`, `extract_missing_codes()`, `extract_question_preface()`, `extract_sata()`, `extract_universe()`, `extract_val_labels()`, `extract_var_label()`, `extract_var_note()`, `infer_question_prefaces()`, `set_missing_codes()`, `set_question_preface()`, `set_sata()`, `set_universe()`, `set_val_labels()`, `set_var_label()`, `set_var_note()`, `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.
validate	Logical. If TRUE (default), re-runs the S7 class validator after updating, which checks structural invariants (column existence, weight column type and positivity, etc.).

**Value**

The modified survey object, invisibly.

**See Also**

[as\\_survey\(\)](#) to create a survey\_taylor object, [as\\_survey\\_replicate\(\)](#) to create a survey\_replicate 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 = wtmecc2yr,
               strata = sdmvstra, nest = TRUE)

# Switch to interview weight for interview-based variables
d_updated <- update_design(d, weights = wtint2yr)
```

# Index

- \* **analysis**
  - clean, 22
  - get\_anova, 34
  - get\_corr, 36
  - get\_covariance, 40
  - get\_diffs, 43
  - get\_freqs, 46
  - get\_means, 49
  - get\_pairwise, 51
  - get\_quantiles, 53
  - get\_ratios, 56
  - get\_t\_test, 60
  - get\_totals, 58
  - get\_variance, 62
  - meta, 69
- \* **collections**
  - add\_survey, 5
  - as\_survey\_collection, 11
  - remove\_survey, 89
  - set\_collection\_id, 90
  - set\_collection\_if\_missing\_var, 91
  - survey\_collection, 100
- \* **constructors**
  - as\_survey, 8
  - as\_survey\_nonprob, 13
  - as\_survey\_replicate, 14
  - as\_survey\_twophase, 17
  - survey\_data, 101
  - survey\_glm, 102
  - survey\_glm\_fit, 105
  - survey\_replicate, 108
  - survey\_taylor, 110
  - survey\_twophase, 111
- \* **conversion**
  - as\_svydesign, 19
  - as\_tbl\_svy, 20
  - from\_svydesign, 32
  - from\_tbl\_svy, 33
- \* **datasets**
  - acs\_pums\_wy, 3
  - anes\_2024, 6
  - gss\_2024, 65
  - nhanes\_2017, 71
  - ns\_wave1, 73
  - pew\_jewish\_2020, 80
  - pew\_npors\_2025, 86
- \* **metadata**
  - classify\_question\_type, 21
  - extract\_metadata, 24
  - extract\_missing\_codes, 25
  - extract\_question\_preamble, 26
  - extract\_sata, 27
  - extract\_universe, 28
  - extract\_val\_labels, 29
  - extract\_var\_label, 30
  - extract\_var\_note, 31
  - infer\_question\_preambles, 67
  - set\_missing\_codes, 92
  - set\_question\_preamble, 93
  - set\_sata, 94
  - set\_universe, 95
  - set\_val\_labels, 96
  - set\_var\_label, 98
  - set\_var\_note, 99
  - survey\_metadata, 107
  - survey\_weighting\_history, 112
- \* **update**
  - update\_design, 113
- acs\_pums\_wy, 3
- add\_survey, 5, 12, 90–92, 101
- add\_survey(), 12, 90, 101
- anes\_2024, 6
- as\_survey, 8, 14, 16, 18, 102, 104, 106, 109, 111, 112
- as\_survey(), 13, 14, 16–18, 69, 102, 107, 110, 111, 114
- as\_survey\_collection, 6, 11, 90–92, 101
- as\_survey\_collection(), 5, 6, 90, 100, 101

- as\_survey\_nonprob, [10](#), [13](#), [16](#), [18](#), [102](#), [104](#), [106](#), [109](#), [111](#), [112](#)
- as\_survey\_nonprob(), [79](#), [102](#)
- as\_survey\_replicate, [10](#), [14](#), [14](#), [18](#), [102](#), [104](#), [106](#), [109](#), [111](#), [112](#)
- as\_survey\_replicate(), [10](#), [14](#), [17](#), [18](#), [102](#), [108](#), [109](#), [114](#)
- as\_survey\_twophase, [10](#), [14](#), [16](#), [17](#), [102](#), [104](#), [106](#), [109](#), [111](#), [112](#)
- as\_survey\_twophase(), [10](#), [14](#), [16](#), [102](#), [111–113](#)
- as\_svydesign, [19](#), [20](#), [32](#), [33](#)
- as\_svydesign(), [20](#), [32](#)
- as\_tbl\_svy, [20](#), [20](#), [32](#), [33](#)
- as\_tbl\_svy(), [33](#)
- base::match.arg(), [37](#)
- classify\_question\_type, [21](#), [24–31](#), [69](#), [93–99](#), [108](#), [113](#)
- clean, [22](#), [35](#), [39](#), [42](#), [46](#), [48](#), [51](#), [53](#), [55](#), [58](#), [60](#), [62](#), [64](#), [70](#)
- extract\_metadata, [22](#), [24](#), [25–31](#), [69](#), [93–99](#), [108](#), [113](#)
- extract\_missing\_codes, [22](#), [24](#), [25](#), [26–31](#), [69](#), [93–99](#), [108](#), [113](#)
- extract\_question\_preface, [22](#), [24](#), [25](#), [26](#), [27–31](#), [69](#), [93–99](#), [108](#), [113](#)
- extract\_question\_preface(), [94](#)
- extract\_sata, [22](#), [24–26](#), [27](#), [28–31](#), [69](#), [93–99](#), [108](#), [113](#)
- extract\_sata(), [22](#), [95](#)
- extract\_universe, [22](#), [24–27](#), [28](#), [29–31](#), [69](#), [93–99](#), [108](#), [113](#)
- extract\_val\_labels, [22](#), [24–28](#), [29](#), [30](#), [31](#), [69](#), [93–99](#), [108](#), [113](#)
- extract\_val\_labels(), [97](#)
- extract\_var\_label, [22](#), [24–29](#), [30](#), [31](#), [69](#), [93–99](#), [108](#), [113](#)
- extract\_var\_label(), [98](#)
- extract\_var\_note, [22](#), [24–30](#), [31](#), [69](#), [93–99](#), [108](#), [113](#)
- extract\_var\_note(), [99](#)
- from\_svydesign, [20](#), [32](#), [33](#)
- from\_svydesign(), [10](#), [20](#), [33](#)
- from\_tbl\_svy, [20](#), [32](#), [33](#)
- from\_tbl\_svy(), [20](#)
- get\_anova, [23](#), [34](#), [39](#), [42](#), [46](#), [48](#), [51](#), [53](#), [55](#), [58](#), [60](#), [62](#), [64](#), [70](#)
- get\_corr, [23](#), [35](#), [36](#), [42](#), [46](#), [48](#), [51](#), [53](#), [55](#), [58](#), [60](#), [62](#), [64](#), [70](#)
- get\_covariance, [23](#), [35](#), [39](#), [40](#), [46](#), [48](#), [51](#), [53](#), [55](#), [58](#), [60](#), [62](#), [64](#), [70](#)
- get\_diffs, [23](#), [35](#), [39](#), [42](#), [43](#), [48](#), [51](#), [53](#), [55](#), [58](#), [60](#), [62](#), [64](#), [70](#)
- get\_freqs, [23](#), [35](#), [39](#), [42](#), [46](#), [46](#), [51](#), [53](#), [55](#), [58](#), [60](#), [62](#), [64](#), [70](#)
- get\_means, [23](#), [35](#), [39](#), [42](#), [46](#), [48](#), [49](#), [53](#), [55](#), [58](#), [60](#), [62](#), [64](#), [70](#)
- get\_means(), [16](#)
- get\_pairwise, [23](#), [35](#), [39](#), [42](#), [46](#), [48](#), [51](#), [51](#), [55](#), [58](#), [60](#), [62](#), [64](#), [70](#)
- get\_quantiles, [23](#), [35](#), [39](#), [42](#), [46](#), [48](#), [51](#), [53](#), [53](#), [58](#), [60](#), [62](#), [64](#), [70](#)
- get\_ratios, [23](#), [35](#), [39](#), [42](#), [46](#), [48](#), [51](#), [53](#), [55](#), [56](#), [60](#), [62](#), [64](#), [70](#)
- get\_t\_test, [23](#), [35](#), [39](#), [42](#), [46](#), [48](#), [51](#), [53](#), [55](#), [58](#), [60](#), [60](#), [64](#), [70](#)
- get\_t\_test(), [51](#)
- get\_totals, [23](#), [35](#), [39](#), [42](#), [46](#), [48](#), [51](#), [53](#), [55](#), [58](#), [58](#), [62](#), [64](#), [70](#)
- get\_totals(), [16](#)
- get\_variance, [23](#), [35](#), [39](#), [42](#), [46](#), [48](#), [51](#), [53](#), [55](#), [58](#), [60](#), [62](#), [62](#), [70](#)
- gss\_2024, [65](#)
- infer\_question\_prefaces, [22](#), [24–31](#), [67](#), [93–99](#), [108](#), [113](#)
- meta, [23](#), [35](#), [39](#), [42](#), [46](#), [48](#), [51](#), [53](#), [55](#), [58](#), [60](#), [62](#), [64](#), [69](#)
- meta(), [23](#), [45](#), [53](#), [62](#)
- nhanes\_2017, [71](#)
- ns\_wave1, [73](#)
- pew\_jewish\_2020, [80](#)
- pew\_npors\_2025, [86](#)
- remove\_survey, [6](#), [12](#), [89](#), [91](#), [92](#), [101](#)
- remove\_survey(), [6](#), [12](#), [101](#)
- rlang::check\_dots\_empty(), [34](#)
- set\_collection\_id, [6](#), [12](#), [90](#), [90](#), [92](#), [101](#)
- set\_collection\_id(), [11](#), [101](#)
- set\_collection\_if\_missing\_var, [6](#), [12](#), [90](#), [91](#), [91](#), [101](#)

- set\_collection\_if\_missing\_var(), [12](#), [101](#)
- set\_missing\_codes, [22](#), [24–31](#), [69](#), [92](#), [94–99](#), [108](#), [113](#)
- set\_missing\_codes(), [25](#)
- set\_question\_preface, [22](#), [24–31](#), [69](#), [93](#), [93](#), [95–99](#), [108](#), [113](#)
- set\_question\_preface(), [22](#), [26](#)
- set\_sata, [22](#), [24–31](#), [69](#), [93](#), [94](#), [94](#), [96–99](#), [108](#), [113](#)
- set\_sata(), [21](#), [22](#), [27](#)
- set\_universe, [22](#), [24–31](#), [69](#), [93–95](#), [95](#), [97–99](#), [108](#), [113](#)
- set\_universe(), [28](#)
- set\_val\_labels, [22](#), [24–31](#), [69](#), [93–96](#), [96](#), [98](#), [99](#), [108](#), [113](#)
- set\_val\_labels(), [29](#)
- set\_var\_label, [22](#), [24–31](#), [69](#), [93–97](#), [98](#), [99](#), [108](#), [113](#)
- set\_var\_label(), [10](#), [16](#), [30](#), [92](#), [93](#), [96](#), [99](#)
- set\_var\_note, [22](#), [24–31](#), [69](#), [93–98](#), [99](#), [108](#), [113](#)
- set\_var\_note(), [31](#)
- stats::glm(), [103](#)
- stats::glm.control(), [103](#)
- stats::optimize(), [38](#)
- stats::p.adjust(), [44](#), [45](#), [52](#)
- survey::svydesign(), [10](#)
- survey\_base, [34](#), [35](#), [100](#), [106](#)
- survey\_collection, [6](#), [11](#), [12](#), [38](#), [40](#), [41](#), [45](#), [48](#), [50](#), [52](#), [54](#), [57](#), [59](#), [62–64](#), [90–92](#), [100](#)
- survey\_data, [10](#), [14](#), [16](#), [18](#), [101](#), [104](#), [106](#), [109](#), [111](#), [112](#)
- survey\_glm, [10](#), [14](#), [16](#), [18](#), [102](#), [102](#), [106](#), [109](#), [111](#), [112](#)
- survey\_glm(), [23](#), [34](#), [35](#), [45](#), [105](#), [106](#)
- survey\_glm\_fit, [10](#), [14](#), [16](#), [18](#), [34](#), [35](#), [102](#), [104](#), [105](#), [109](#), [111](#), [112](#)
- survey\_metadata, [22](#), [24–31](#), [69](#), [93–99](#), [107](#), [109–111](#), [113](#)
- survey\_nonprob, [10](#), [14](#), [16](#), [18](#), [102](#), [104](#), [106](#), [109](#), [111](#), [112](#)
- survey\_replicate, [10](#), [14](#), [16](#), [18](#), [102](#), [104](#), [106](#), [108](#), [111](#), [112](#)
- survey\_taylor, [10](#), [14](#), [16](#), [18](#), [102](#), [104](#), [106](#), [109](#), [110](#), [112](#)
- survey\_twophase, [10](#), [14](#), [16](#), [18](#), [102](#), [104](#), [106](#), [109](#), [111](#), [111](#)
- survey\_weighting\_history, [22](#), [24–31](#), [69](#), [93–99](#), [108](#), [112](#)
- tidyselect::all\_of(), [21](#), [24–31](#), [94](#)
- tidyselect::any\_of(), [21](#), [24–31](#), [94](#)
- tidyselect::matches(), [24–26](#), [28–31](#)
- tidyselect::starts\_with(), [21](#), [24–31](#), [94](#)
- update\_design, [113](#)