

Package: cardx (via r-universe)

October 8, 2026

Title Extra Analysis Results Data Utilities

Version 0.3.4.9003

Description Create extra Analysis Results Data (ARD) summary objects.

The package supplements the simple ARD functions from the 'cards' package, exporting functions to put statistical results in the ARD format. These objects are used and re-used to construct summary tables, visualizations, and written reports.

License Apache License 2.0

URL <https://github.com/pharmaverse/cardx>,
<https://pharmaverse.github.io/cardx/>

BugReports <https://github.com/pharmaverse/cardx/issues>

Depends R (>= 4.2)

Imports cards (>= 0.8.0.9000), cli (>= 3.6.1), dplyr (>= 1.2.0), glue (>= 1.6.2), lifecycle (>= 1.0.3), rlang (>= 1.1.1), tidyr (>= 1.3.0)

Suggests aod (>= 1.3.3), broom (>= 1.0.8), broom.helpers (>= 1.17.0), broom.mixed (>= 0.2.9), car (>= 3.1-2), effectsize (>= 0.8.8), emmeans (>= 1.7.3), geepack (>= 1.3.2), ggsvrfit (>= 1.1.0), lme4 (>= 1.1-37), parameters (>= 0.20.2), smd (>= 0.6.6), survey (>= 4.2), survival (>= 3.6-4), testthat (>= 3.2.0), withr (>= 2.5.0)

Remotes pharmaverse/cards

Config/Needs/website insightsengineering/nesttemplate

Config/roxygen2/version 8.1.0

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE)

Config/pak/sysreqs libicu-dev

Repository <https://ddsjoberg.r-universe.dev>

Date/Publication 2026-09-07 16:05:35 UTC

RemoteUrl <https://github.com/pharmaverse/cardx>

RemoteRef HEAD

RemoteSha 96457da8cd94a4c74188cc165c0bac05689d1223

Contents

ard_aod_wald_test	3
ard_attributes.survey.design	4
ard_car_anova	5
ard_car_vif	5
ard_categorical_ci	6
ard_categorical_ci.survey.design	7
ard_continuous_ci	9
ard_continuous_ci.survey.design	10
ard_effectsize_cohens_d	11
ard_effectsize_hedges_g	12
ard_emmeans_contrast	13
ard_incidence_rate	16
ard_missing.survey.design	18
ard_regression	19
ard_regression_basic	21
ard_smd_smd	22
ard_stats_anova	23
ard_stats_aov	25
ard_stats_chisq_test	26
ard_stats_fisher_test	27
ard_stats_kruskal_test	27
ard_stats_mantelhaen_test	28
ard_stats_mcnemar_test	29
ard_stats_mood_test	30
ard_stats_oneway_test	31
ard_stats_poisson_test	32
ard_stats_prop_test	33
ard_stats_t_test	34
ard_stats_t_test_onesample	35
ard_stats_wilcox_test	36
ard_stats_wilcox_test_onesample	37
ard_summary.survey.design	38
ard_survey_svychisq	39
ard_survey_svyranktest	40
ard_survey_svytttest	41
ard_survival_survdiff	42
ard_survival_survfit	42

<i>ard_aod_wald_test</i>	3
ard_survival_survfit_diff	45
ard_tabulate.survey.design	46
ard_tabulate_abnormal	47
ard_tabulate_max	49
ard_tabulate_value.survey.design	51
ard_total_n.survey.design	52
construction_helpers	53
proportion_ci	55
Index	59

<i>ard_aod_wald_test</i>	<i>ARD Wald Test</i>
--------------------------	----------------------

Description

Function takes a regression model object and calculates Wald statistical test using `aod::wald.test()`.

Usage

```
ard_aod_wald_test(  
  x,  
  tidy_fun = broom.helpers::tidy_with_broom_or_parameters,  
  ...  
)
```

Arguments

- x regression model object
- tidy_fun (function)
a tidier. Default is `broom.helpers::tidy_with_broom_or_parameters`
- ... arguments passed to `aod::wald.test(...)`

Value

data frame

Examples

```
lm(AGE ~ ARM, data = cards::ADSL) |>  
  ard_aod_wald_test()
```

```
ard_attributes.survey.design
```

ARD Attributes

Description

Add variable attributes to an ARD data frame.

- The label attribute will be added for all columns, and when no label is specified and no label has been set for a column using the `label=` argument, the column name will be placed in the label statistic.
- The class attribute will also be returned for all columns.
- Any other attribute returned by `attributes()` will also be added, e.g. factor levels.

Usage

```
## S3 method for class 'survey.design'
ard_attributes(data, variables = everything(), label = NULL, ...)
```

Arguments

<code>data</code>	(<code>survey.design</code>) a design object often created with <code>survey::svydesign()</code> .
<code>variables</code>	(<code>tidy-select</code>) variables to include
<code>label</code>	(named list) named list of variable labels, e.g. <code>list(cyl = "No. Cylinders")</code> . Default is <code>NULL</code>
<code>...</code>	These dots are for future extensions and must be empty.

Value

an ARD data frame of class 'card'

Examples

```
data(api, package = "survey")
dclus1 <- survey::svydesign(id = ~dnum, weights = ~pw, data = apiclus1, fpc = ~fpc)

ard_attributes(
  data = dclus1,
  variables = c(sname, dname),
  label = list(sname = "School Name", dname = "District Name")
)
```

ard_car_anova	<i>ARD ANOVA from car Package</i>
---------------	-----------------------------------

Description

Function takes a regression model object and calculated ANOVA using `car::Anova()`.

Usage

```
ard_car_anova(x, ...)
```

Arguments

x	regression model object
...	arguments passed to <code>car::Anova(...)</code>

Value

data frame

Examples

```
lm(AGE ~ ARM, data = cards::ADSL) |>
  ard_car_anova()

glm(vs ~ factor(cyl) + factor(am), data = mtcars, family = binomial) |>
  ard_car_anova(test.statistic = "Wald")
```

ard_car_vif	<i>Regression VIF ARD</i>
-------------	---------------------------

Description

Function takes a regression model object and returns the variance inflation factor (VIF) using `car::vif()` and converts it to a ARD structure

Usage

```
ard_car_vif(x, ...)
```

Arguments

x	regression model object See <code>car::vif()</code> for details
...	arguments passed to <code>car::vif(...)</code>

Value

data frame

Examples

```
lm(AGE ~ ARM + SEX, data = cards::ADSL) |>
  ard_car_vif()
```

ard_categorical_ci	<i>ARD Proportion Confidence Intervals</i>
--------------------	--

Description

Calculate confidence intervals for proportions.

Usage

```
ard_categorical_ci(data, ...)

## S3 method for class 'data.frame'
ard_categorical_ci(
  data,
  variables,
  by = dplyr::group_vars(data),
  method = c("waldcc", "wald", "clopper-pearson", "wilson", "wilsoncc", "strat_wilson",
    "strat_wilsoncc", "agresti-coull", "jeffreys"),
  denominator = c("column", "row", "cell"),
  conf.level = 0.95,
  value = list(where(is_binary) ~ 1L, where(is_logical) ~ TRUE),
  strata = NULL,
  weights = NULL,
  max.iterations = 10,
  ...
)
```

Arguments

data	(data.frame) a data frame
...	Arguments passed to methods.
variables	(tidy-select) columns to include in summaries. Columns must be class <logical> or <numeric> values coded as c(0,1).
by	(tidy-select) columns to stratify calculations by.

method	(string) string indicating the type of confidence interval to calculate. Must be one of . See ?proportion_ci for details.
denominator	(string) Must be one of 'column' (default), 'row', and 'cell', which specifies the di- rection of the calculation/denominator. Argument is similar to cards::ard_tabulate(denominator).
conf.level	(scalar numeric) a scalar in (0,1) indicating the confidence level. Default is 0.95
value	(formula-list-selector) function will calculate the CIs for all levels of the variables specified. Use this argument to instead request only a single level by summarized. Default is list(where(is_binary) ~ 1L, where(is.logical) ~ TRUE), where columns coded as 0/1 and TRUE/FALSE will summarize the 1 and TRUE levels.
strata, weights, max.iterations	arguments passed to proportion_ci_strat_wilson(), when method='strat_wilson'

Value

an ARD data frame

Examples

```
# compute CI for binary variables
ard_categorical_ci(mtcars, variables = c(vs, am), method = "wilson")

# compute CIs for each level of a categorical variable
ard_categorical_ci(mtcars, variables = cyl, method = "jeffreys")
```

ard_categorical_ci.survey.design
ARD survey categorical CIs

Description

Confidence intervals for categorical variables calculated via [survey::svyciprop\(\)](#).

Usage

```
## S3 method for class 'survey.design'
ard_categorical_ci(
  data,
  variables,
  by = NULL,
  method = c("logit", "likelihood", "asin", "beta", "mean", "xlogit"),
  conf.level = 0.95,
  value = list(where(is_binary) ~ 1L, where(is.logical) ~ TRUE),
```

```

    df = survey::degf(data),
    ...
  )

```

Arguments

<code>data</code>	(<code>survey.design</code>) a design object often created with <code>survey::svydesign()</code> .
<code>variables</code>	(<code>tidy-select</code>) columns to include in summaries.
<code>by</code>	(<code>tidy-select</code>) results are calculated for all combinations of the columns specified, including unobserved combinations and unobserved factor levels.
<code>method</code>	(<code>string</code>) Method passed to <code>survey::svyciprop(method)</code>
<code>conf.level</code>	(<code>scalar numeric</code>) a scalar in $(0, 1)$ indicating the confidence level. Default is 0.95
<code>value</code>	(<code>formula-list-selector</code>) function will calculate the CIs for all levels of the variables specified. Use this argument to instead request only a single level by summarized. Default is <code>list(where(is_binary) ~ 1L, where(is_logical) ~ TRUE)</code> , where columns coded as 0/1 and TRUE/FALSE will summarize the 1 and TRUE levels.
<code>df</code>	(<code>numeric</code>) denominator degrees of freedom, passed to <code>survey::svyciprop(df)</code> . Default is <code>survey::degf(data)</code> .
<code>...</code>	arguments passed to <code>survey::svyciprop()</code>

Value

ARD data frame

Examples

```

data(api, package = "survey")
dclus1 <- survey::svydesign(id = ~dnum, weights = ~pw, data = apiclus1, fpc = ~fpc)

ard_categorical_ci(dclus1, variables = sch.wide)
ard_categorical_ci(dclus1, variables = sch.wide, value = sch.wide ~ "Yes", method = "xlogit")

```

ard_continuous_ci *ARD continuous CIs*

Description

One-sample confidence intervals for continuous variable means and medians.

Usage

```
ard_continuous_ci(data, ...)

## S3 method for class 'data.frame'
ard_continuous_ci(
  data,
  variables,
  by = dplyr::group_vars(data),
  conf.level = 0.95,
  method = c("t.test", "wilcox.test"),
  ...
)
```

Arguments

data	(data.frame) a data frame. See below for details.
...	arguments passed to <code>t.test()</code> or <code>wilcox.test()</code>
variables	(tidy-select) column names to be compared. Independent t-tests will be computed for each variable.
by	(tidy-select) optional column name to compare by.
conf.level	(scalar numeric) confidence level for confidence interval. Default is 0.95.
method	(string) a string indicating the method to use for the confidence interval calculation. Must be one of "t.test" or "wilcox.test"

Value

ARD data frame

Examples

```
ard_continuous_ci(mtcars, variables = c(mpg, hp), method = "wilcox.test")
ard_continuous_ci(mtcars, variables = mpg, by = am, method = "t.test")
```

```
ard_continuous_ci.survey.design
```

ARD survey continuous CIs

Description

One-sample confidence intervals for continuous variables' means and medians. Confidence limits are calculated with `survey::svymean()` and `survey::svyquantile()`.

Usage

```
## S3 method for class 'survey.design'
ard_continuous_ci(
  data,
  variables,
  by = NULL,
  method = c("svymean", "svymedian.mean", "svymedian.beta", "svymedian.xlogit",
    "svymedian.asin", "svymedian.score"),
  conf.level = 0.95,
  df = survey::degf(data),
  ...
)
```

Arguments

<code>data</code>	(<code>survey.design</code>) a design object often created with <code>survey::svydesign()</code> .
<code>variables</code>	(<code>tidy-select</code>) columns to include in summaries.
<code>by</code>	(<code>tidy-select</code>) results are calculated for all combinations of the columns specified, including unobserved combinations and unobserved factor levels.
<code>method</code>	(string) Method for confidence interval calculation. When "svymean", the calculation is computed via <code>survey::svymean()</code> . Otherwise, it is calculated via <code>survey::svyquantile(interval.type)</code> .
<code>conf.level</code>	(scalar numeric) confidence level for confidence interval. Default is 0.95.
<code>df</code>	(numeric) denominator degrees of freedom, passed to <code>survey::confint(df)</code> . Default is <code>survey::degf(data)</code> .
<code>...</code>	arguments passed to <code>survey::confint()</code>

Value

ARD data frame

Examples

```
data(api, package = "survey")
dclus1 <- survey::svydesign(id = ~dnum, weights = ~pw, data = apiclus1, fpc = ~fpc)

ard_continuous_ci(dclus1, variables = api00)
ard_continuous_ci(dclus1, variables = api00, method = "svymedian.xlogit")
```

ard_effectsize_cohens_d

ARD Cohen's D Test

Description

Analysis results data for paired and non-paired Cohen's D Effect Size Test using `effectsize::cohens_d()`.

Usage

```
ard_effectsize_cohens_d(data, by, variables, conf.level = 0.95, ...)

ard_effectsize_paired_cohens_d(data, by, variables, id, conf.level = 0.95, ...)
```

Arguments

data	(data.frame) a data frame. See below for details.
by	(tidy-select) column name to compare by. Must be a categorical variable with exactly two levels.
variables	(tidy-select) column names to be compared. Must be a continuous variables. Independent tests will be run for each variable.
conf.level	(scalar numeric) confidence level for confidence interval. Default is 0.95.
...	arguments passed to <code>effectsize::cohens_d(...)</code>
id	(tidy-select) column name of the subject or participant ID

Details

For the `ard_effectsize_cohens_d()` function, the data is expected to be one row per subject. The data is passed as `effectsize::cohens_d(data[[variable]]~data[[by]], data, paired = FALSE, ...)`.

For the `ard_effectsize_paired_cohens_d()` function, the data is expected to be one row per subject per by level. Before the effect size is calculated, the data are reshaped to a wide format to be one row per subject. The data are then passed as `effectsize::cohens_d(x = data_wide[<by level 1>], y = data_wide[<by level 2>])`.

Value

ARD data frame

Examples

```
cards::ADSL |>
  dplyr::filter(ARM %in% c("Placebo", "Xanomeline High Dose")) |>
  ard_effectsize_cohens_d(by = ARM, variables = AGE)

# constructing a paired data set,
# where patients receive both treatments
cards::ADSL[c("ARM", "AGE")] |>
  dplyr::filter(ARM %in% c("Placebo", "Xanomeline High Dose")) |>
  dplyr::mutate(.by = ARM, USUBJID = dplyr::row_number()) |>
  dplyr::arrange(USUBJID, ARM) |>
  dplyr::group_by(USUBJID) |>
  dplyr::filter(dplyr::n() > 1) |>
  ard_effectsize_paired_cohens_d(by = ARM, variables = AGE, id = USUBJID)
```

ard_effectsize_hedges_g

ARD Hedge's G Test

Description

Analysis results data for paired and non-paired Hedge's G Effect Size Test using `effectsize::hedges_g()`.

Usage

```
ard_effectsize_hedges_g(data, by, variables, conf.level = 0.95, ...)
```

```
ard_effectsize_paired_hedges_g(data, by, variables, id, conf.level = 0.95, ...)
```

Arguments

data	(data.frame) a data frame. See below for details.
by	(tidy-select) column name to compare by. Must be a categorical variable with exactly two levels.
variables	(tidy-select) column names to be compared. Must be a continuous variable. Independent tests will be run for each variable
conf.level	(scalar numeric) confidence level for confidence interval. Default is 0.95.
...	arguments passed to <code>effectsize::hedges_g(...)</code>

id (tidy-select)
column name of the subject or participant ID

Details

For the `ard_effectsize_hedges_g()` function, the data is expected to be one row per subject. The data is passed as `effectsize::hedges_g(data[[variable]]~data[[by]], data, paired = FALSE, ...)`.

For the `ard_effectsize_paired_hedges_g()` function, the data is expected to be one row per subject per by level. Before the effect size is calculated, the data are reshaped to a wide format to be one row per subject. The data are then passed as `effectsize::hedges_g(x = data_wide[[<by level 1>]], y = data_wide[[<by level 2>]])`.

Value

ARD data frame

Examples

```
cards::ADSL |>
  dplyr::filter(ARM %in% c("Placebo", "Xanomeline High Dose")) |>
  ard_effectsize_hedges_g(by = ARM, variables = AGE)

# constructing a paired data set,
# where patients receive both treatments
cards::ADSL[c("ARM", "AGE")] |>
  dplyr::filter(ARM %in% c("Placebo", "Xanomeline High Dose")) |>
  dplyr::mutate(.by = ARM, USUBJID = dplyr::row_number()) |>
  dplyr::arrange(USUBJID, ARM) |>
  dplyr::group_by(USUBJID) |>
  dplyr::filter(dplyr::n() > 1) |>
  ard_effectsize_paired_hedges_g(by = ARM, variables = AGE, id = USUBJID)
```

ard_emmeans_contrast *ARDs for LS Mean Difference and LS Means*

Description

The `ard_emmeans_contrast()` function calculates least-squares mean differences using the 'emmeans' package using the following

```
emmeans::emmeans(object = <regression model>, specs = ~ <primary covariate>) |>
  emmeans::contrast(method = "pairwise") |>
  summary(infer = TRUE, level = <confidence level>)
```

The `ard_emmeans_emmeans()` function calculates least-squares means using the 'emmeans' package using the following

```
emmeans::emmeans(object = <regression model>, specs = ~ <primary covariate>) |>
  summary(emmeans, calc = c(n = ".wgt."))
```

The arguments `data`, `formula`, `method`, `method.args`, `package` are used to construct the regression model via `cardx::construct_model()`.

Usage

```
ard_emmeans_contrast(
  data,
  formula,
  method,
  method.args = list(),
  package = "base",
  response_type = c("continuous", "dichotomous"),
  conf.level = 0.95,
  primary_covariate = getElement(attr(stats::terms(formula), "term.labels"), 1L)
)

ard_emmeans_emmeans(
  data,
  formula,
  method,
  method.args = list(),
  package = "base",
  response_type = c("continuous", "dichotomous"),
  conf.level = 0.95,
  primary_covariate = getElement(attr(stats::terms(formula), "term.labels"), 1L)
)
```

Arguments

<code>data</code>	(data.frame/survey.design) a data frame or survey design object
<code>formula</code>	(formula) a formula
<code>method</code>	(string) string of function naming the function to be called, e.g. "glm". If function belongs to a library that is not attached, the package name must be specified in the package argument.
<code>method.args</code>	(named list) named list of arguments that will be passed to method. Note that this list may contain non-standard evaluation components. If you are wrapping this function in other functions, the argument must be passed in a way that does not evaluate the list, e.g. using rlang's embrace operator <code>{{ . }}</code> .
<code>package</code>	(string) a package name that will be temporarily loaded when function specified in method is executed.

response_type (string) string indicating whether the model outcome is 'continuous' or 'dichotomous'. When 'dichotomous', the call to `emmeans::emmeans()` is supplemented with argument `regrid="response"`.

conf.level (scalar numeric)
confidence level for confidence interval. Default is 0.95.

primary_covariate
(string)
string indicating the primary covariate (typically the dichotomous treatment variable). Default is the first covariate listed in the formula.

Value

ARD data frame

Examples

```
# LS Mean Difference
ard_emmeans_contrast(
  data = mtcars,
  formula = mpg ~ am + cyl,
  method = "lm"
)

ard_emmeans_contrast(
  data = mtcars,
  formula = vs ~ am + mpg,
  method = "glm",
  method.args = list(family = binomial),
  response_type = "dichotomous"
)

# LS Means
ard_emmeans_emmeans(
  data = mtcars,
  formula = mpg ~ am + cyl,
  method = "lm"
)

ard_emmeans_emmeans(
  data = mtcars,
  formula = vs ~ am + mpg,
  method = "glm",
  method.args = list(family = binomial),
  response_type = "dichotomous"
)
```

ard_incidence_rate	<i>ARD Incidence Rate</i>
--------------------	---------------------------

Description

Function takes a time at risk variable (time) and event count variable (count) and calculates the incidence rate in person-years.

Incidence rate is calculated as: Total number of events that occurred / Total person-time at risk

Usage

```
ard_incidence_rate(
  data,
  time,
  count = NULL,
  id = NULL,
  by = NULL,
  strata = NULL,
  n_person_time = 100,
  unit_label = "time",
  conf.level = 0.95,
  conf.type = c("normal", "normal-log", "exact", "byar")
)
```

Arguments

data	(data.frame) a data frame.
time	(tidy-select) column name of time at risk variable.
count	(tidy-select) column name of variable indicating count of events that occurred. If NULL, each row in data is assumed to correspond to a single event occurrence.
id	(tidy-select) column name used to identify unique subjects in data. If NULL, each row in data is assumed to correspond to a unique subject.
by, strata	(tidy-select) columns to tabulate by/stratify by for summary statistic calculation. Arguments are similar, but with an important distinction: by: results are calculated for all combinations of the columns specified, including unobserved combinations and unobserved factor levels. strata: results are calculated for all <i>observed</i> combinations of the columns specified. Arguments may be used in conjunction with one another.

n_person_time	(numeric) amount of person-time to estimate incidence rate for. Defaults to 100.
unit_label	(string) label for the unit of values in time and estimated person-time output (e.g. "years" for person-years, "days" for person-days, etc.). If the desired person-time estimate unit does not match the current time unit, values of time should be converted to the correct unit during pre-processing. Defaults to "time" (person-time).
conf.level	(numeric) confidence level for the estimated incidence rate.
conf.type	(string) confidence interval type for the estimated incidence rate. One of: normal (default), normal-log, exact, or byar.

Details

The formulas used to calculate the confidence interval for each CI type are as follows, where x_i and t_i represent the number of events and follow-up time for subject i , respectively.

- byar: Byar's approximation of a Poisson CI. A continuity correction of 0.5 is included in the calculation.

$$CI = (\sum x_i + 0.5)(1 - 1/(9 \times (\sum x_i + 0.5))) \pm Z_{1-\alpha/2}/(3\sqrt{\sum x_i + 0.5})^3 / \sum t_i$$

- normal: Normal CI.

$$CI = \sum x_i / \sum t_i \pm Z_{1-\alpha/2} \sqrt{\sum x_i / \sum t_i}$$

- normal-log: Normal-Log CI.

$$CI = \exp(\log(\sum x_i / \sum t_i) \pm Z_{1-\alpha/2} / \sqrt{\sum x_i})$$

- exact: Exact CI for a Poisson mean.

$$CI_{lower} = \chi^2_{\alpha/2, 2\sum x_i + 2} / 2 \sum t_i$$

$$CI_{upper} = \chi^2_{1-\alpha/2, 2\sum x_i + 2} / 2 \sum t_i$$

Value

an ARD data frame of class 'card'

Examples

```

set.seed(1)
data <- data.frame(
  USUBJID = 1:100,
  TRTA = sample(LETTERS[1:3], 100, replace = TRUE),
  AETTE1 = abs(rnorm(100, mean = 0.5)),
  AETOT1 = sample(0:20, 100, replace = TRUE)
)

data |>
  ard_incidence_rate(time = AETTE1, count = AETOT1, id = USUBJID, by = TRTA, unit_label = "years")

```

ard_missing.survey.design

ARD Missing Survey Statistics

Description

Compute Analysis Results Data (ARD) for statistics related to data missingness for survey objects

Usage

```

## S3 method for class 'survey.design'
ard_missing(
  data,
  variables,
  by = NULL,
  statistic = everything() ~ c("N_obs", "N_miss", "N_nonmiss", "p_miss", "p_nonmiss",
    "N_obs_unweighted", "N_miss_unweighted", "N_nonmiss_unweighted", "p_miss_unweighted",
    "p_nonmiss_unweighted"),
  fmt_fun = NULL,
  stat_label = everything() ~ list(N_obs = "Total N", N_miss = "N Missing", N_nonmiss =
    "N not Missing", p_miss = "% Missing", p_nonmiss = "% not Missing",
    N_obs_unweighted = "Total N (unweighted)", N_miss_unweighted =
    "N Missing (unweighted)", N_nonmiss_unweighted = "N not Missing (unweighted)",
    p_miss_unweighted = "% Missing (unweighted)", p_nonmiss_unweighted =
    "% not Missing (unweighted)"),
  fmt_fn = deprecated(),
  ...
)

```

Arguments

data	(survey.design) a design object often created with <code>survey::svydesign()</code> .
variables	(tidy-select) columns to include in summaries.

by	(tidy-select) results are calculated for all combinations of the column specified and the variables. A single column may be specified.
statistic	(formula-list-selector) a named list, a list of formulas, or a single formula where the list element is a character vector of statistic names to include. See default value for options.
fmt_fun	(formula-list-selector) a named list, a list of formulas, or a single formula where the list element is a named list of functions (or the RHS of a formula), e.g. <code>list(mpg = list(mean = \(x) round(x, digits</code>
stat_label	(formula-list-selector) a named list, a list of formulas, or a single formula where the list element is either a named list or a list of formulas defining the statistic labels, e.g. <code>everything() ~ list(mean = "Mean", sd = "SD")</code> or <code>everything() ~ list(mean ~ "Mean", sd ~ "SD")</code> .
fmt_fn	[Deprecated]
...	These dots are for future extensions and must be empty.

Value

an ARD data frame of class 'card'

Examples

```
svy_titanic <- survey::svydesign(~1, data = as.data.frame(Titanic), weights = ~Freq)

ard_missing(svy_titanic, variables = c(Class, Age), by = Survived)
```

ard_regression	<i>Regression ARD</i>
----------------	-----------------------

Description

Function takes a regression model object and converts it to a ARD structure using the `broom.helpers` package.

Usage

```
ard_regression(x, ...)

## Default S3 method:
ard_regression(x, tidy_fun = broom.helpers::tidy_with_broom_or_parameters, ...)

## S3 method for class 'data.frame'
ard_regression(
  x,
```

```

    formula,
    method,
    method.args = list(),
    package = "base",
    tidy_fun = broom.helpers::tidy_with_broom_or_parameters,
    ...
  )

```

Arguments

<code>x</code>	(regression model/data.frame) regression model object or a data frame
<code>...</code>	Arguments passed to <code>broom.helpers::tidy_plus_plus()</code>
<code>tidy_fun</code>	(function) a tidier. Default is <code>broom.helpers::tidy_with_broom_or_parameters</code>
<code>formula</code>	(formula) a formula
<code>method</code>	(string) string of function naming the function to be called, e.g. "glm". If function belongs to a library that is not attached, the package name must be specified in the package argument.
<code>method.args</code>	(named list) named list of arguments that will be passed to method. Note that this list may contain non-standard evaluation components. If you are wrapping this function in other functions, the argument must be passed in a way that does not evaluate the list, e.g. using rlang's embrace operator <code>{{ . }}</code> .
<code>package</code>	(string) a package name that will be temporarily loaded when function specified in method is executed.

Value

data frame

Examples

```

lm(AGE ~ ARM, data = cards::ADSL) |>
  ard_regression(add_estimate_to_reference_rows = TRUE)

ard_regression(
  x = cards::ADSL,
  formula = AGE ~ ARM,
  method = "lm"
)

```

ard_regression_basic *Basic Regression ARD*

Description

A function that takes a regression model and provides basic statistics in an ARD structure. The default output is simpler than `ard_regression()`. The function primarily matches regression terms to underlying variable names and levels. The default arguments used are

```
broom.helpers::tidy_plus_plus(
  add_reference_rows = FALSE,
  add_estimate_to_reference_rows = FALSE,
  add_n = FALSE,
  intercept = FALSE
)
```

Usage

```
ard_regression_basic(x, ...)

## Default S3 method:
ard_regression_basic(
  x,
  tidy_fun = broom.helpers::tidy_with_broom_or_parameters,
  stats_to_remove = c("term", "var_type", "var_label", "var_class", "label",
    "contrasts_type", "contrasts", "var_nlevels"),
  ...
)

## S3 method for class 'data.frame'
ard_regression_basic(
  x,
  formula,
  method,
  method.args = list(),
  package = "base",
  tidy_fun = broom.helpers::tidy_with_broom_or_parameters,
  stats_to_remove = c("term", "var_type", "var_label", "var_class", "label",
    "contrasts_type", "contrasts", "var_nlevels"),
  ...
)
```

Arguments

x	(regression model/data.frame)
	regression model object or a data frame
...	Arguments passed to <code>broom.helpers::tidy_plus_plus()</code>

tidy_fun	(function) a tidier. Default is <code>broom.helpers::tidy_with_broom_or_parameters</code>
stats_to_remove	(character) character vector of statistic names to remove. Default is <code>c("term", "var_type", "var_label", "var_class", "label", "contrasts_type", "contrasts", "var_nlevels")</code> .
formula	(formula) a formula
method	(string) string of function naming the function to be called, e.g. "glm". If function belongs to a library that is not attached, the package name must be specified in the package argument.
method.args	(named list) named list of arguments that will be passed to method. Note that this list may contain non-standard evaluation components. If you are wrapping this function in other functions, the argument must be passed in a way that does not evaluate the list, e.g. using <code>rlang</code> 's embrace operator <code>{{ . }}</code> .
package	(string) a package name that will be temporarily loaded when function specified in method is executed.

Value

data frame

Examples

```
lm(AGE ~ ARM, data = cards::ADSL) |>
  ard_regression_basic()

ard_regression_basic(
  x = cards::ADSL,
  formula = AGE ~ ARM,
  method = "lm"
)
```

ard_smd_smd

ARD Standardized Mean Difference

Description

Standardized mean difference calculated via `smd::smd()` with `na.rm = TRUE`. Additionally, this function add a confidence interval to the SMD when `std.error=TRUE`, which the original `smd::smd()` does not include.

Usage

```
ard_smd_smd(data, by, variables, std.error = TRUE, conf.level = 0.95, ...)
```

Arguments

- data (data.frame/survey.design)
a data frame or object of class 'survey.design' (typically created with `survey::svydesign()`).
- by (`tidy-select`)
column name to compare by.
- variables (`tidy-select`)
column names to be compared. Independent tests will be computed for each variable.
- std.error (scalar logical)
Logical indicator for computing standard errors using `smd::compute_smd_var()`. Default is TRUE.
- conf.level (scalar numeric)
confidence level for confidence interval. Default is 0.95.
- ... arguments passed to `smd::smd()`

Value

ARD data frame

Examples

```
ard_smd_smd(cards::ADSL, by = SEX, variables = AGE)
ard_smd_smd(cards::ADSL, by = SEX, variables = AGEGR1)
```

ard_stats_anova	ARD ANOVA
-----------------	-----------

Description

Prepare ANOVA results from the `stats::anova()` function. Users may pass a pre-calculated `stats::anova()` object or a list of formulas. In the latter case, the models will be constructed using the information passed and models will be passed to `stats::anova()`.

Usage

```
ard_stats_anova(x, ...)

## S3 method for class 'anova'
ard_stats_anova(x, method_text = "ANOVA results from `stats::anova()`", ...)

## S3 method for class 'data.frame'
```

```
ard_stats_anova(
  x,
  formulas,
  method,
  method.args = list(),
  package = "base",
  method_text = "ANOVA results from `stats::anova()`",
  ...
)
```

Arguments

<code>x</code>	(anova or data.frame) an object of class 'anova' created with <code>stats::anova()</code> or a data frame
<code>...</code>	These dots are for future extensions and must be empty.
<code>method_text</code>	(string) string of the method used. Default is "ANOVA results from stats::anova()". We provide the option to change this as <code>stats::anova()</code> can produce results from many types of models that may warrant a more precise description.
<code>formulas</code>	(list) a list of formulas
<code>method</code>	(string) string of function naming the function to be called, e.g. "glm". If function belongs to a library that is not attached, the package name must be specified in the package argument.
<code>method.args</code>	(named list) named list of arguments that will be passed to method. Note that this list may contain non-standard evaluation components. If you are wrapping this function in other functions, the argument must be passed in a way that does not evaluate the list, e.g. using <code>rlang</code> 's embrace operator <code>{{ . }}</code> .
<code>package</code>	(string) a package name that will be temporarily loaded when function specified in method is executed.

Details

When a list of formulas is supplied to `ard_stats_anova()`, these formulas along with information from other arguments, are used to construct models and pass those models to `stats::anova()`.

The models are constructed using `rlang::exec()`, which is similar to `do.call()`.

```
rlang::exec(.fn = method, formula = formula, data = data, !!!method.args)
```

The above function is executed in `withr::with_namespace(package)`, which allows for the use of `ard_stats_anova(method)` from packages, e.g. `package = 'lme4'` must be specified when `method = 'glmer'`. See example below.

Value

ARD data frame

Examples

```
anova(
  lm(mpg ~ am, mtcars),
  lm(mpg ~ am + hp, mtcars)
) |>
  ard_stats_anova()

ard_stats_anova(
  x = mtcars,
  formulas = list(am ~ mpg, am ~ mpg + hp),
  method = "glm",
  method.args = list(family = binomial)
)

ard_stats_anova(
  x = mtcars,
  formulas = list(am ~ 1 + (1 | vs), am ~ mpg + (1 | vs)),
  method = "glmer",
  method.args = list(family = binomial),
  package = "lme4"
)
```

ard_stats_aov	ARD ANOVA
---------------	-----------

Description

Analysis results data for Analysis of Variance. Calculated with stats::aov()

Usage

```
ard_stats_aov(formula, data, ...)
```

Arguments

- formula A formula specifying the model.
- data A data frame in which the variables specified in the formula will be found. If missing, the variables are searched for in the standard way.
- ... arguments passed to stats::aov(...)

Value

ARD data frame

Examples

```
ard_stats_aov(AGE ~ ARM, data = cards::ADSL)
```

```
ard_stats_chisq_test
```

ARD Chi-squared Test

Description

Analysis results data for Pearson's Chi-squared Test. Calculated with `chisq.test(x = data[[variable]], y = data[[by]], ...)`

Usage

```
ard_stats_chisq_test(data, by, variables, ...)
```

Arguments

<code>data</code>	(<code>data.frame</code>) a data frame.
<code>by</code>	(<code>tidy-select</code>) column name to compare by.
<code>variables</code>	(<code>tidy-select</code>) column names to be compared. Independent tests will be computed for each variable.
<code>...</code>	additional arguments passed to <code>chisq.test(...)</code>

Value

ARD data frame

Examples

```
cards::ADSL |>
  ard_stats_chisq_test(by = "ARM", variables = "AGEGR1")
```

ard_stats_fisher_test *ARD Fisher's Exact Test*

Description

Analysis results data for Fisher's Exact Test. Calculated with `fisher.test(x = data[[variable]], y = data[[by]], ...)`

Usage

```
ard_stats_fisher_test(data, by, variables, conf.level = 0.95, ...)
```

Arguments

<code>data</code>	(data.frame) a data frame.
<code>by</code>	(tidy-select) column name to compare by
<code>variables</code>	(tidy-select) column names to be compared. Independent tests will be computed for each variable.
<code>conf.level</code>	(scalar numeric) confidence level for confidence interval. Default is 0.95.
<code>...</code>	additional arguments passed to <code>fisher.test(...)</code>

Value

ARD data frame

Examples

```
cards::ADSL[1:30, ] |>  
  ard_stats_fisher_test(by = "ARM", variables = "AGEGR1")
```

ard_stats_kruskal_test *ARD Kruskal-Wallis Test*

Description

Analysis results data for Kruskal-Wallis Rank Sum Test.
Calculated with `kruskal.test(data[[variable]], data[[by]], ...)`

Usage

```
ard_stats_kruskal_test(data, by, variables)
```

Arguments

data	(data.frame) a data frame.
by	(tidy-select) column name to compare by.
variables	(tidy-select) column names to be compared. Independent tests will be computed for each variable.

Value

ARD data frame

Examples

```
cards::ADSL |>
  ard_stats_kruskal_test(by = "ARM", variables = "AGE")
```

```
ard_stats_mantelhaen_test
```

ARD Cochran-Mantel-Haenszel Chi-Squared Test

Description

Analysis results data for Cochran-Mantel-Haenszel Chi-Squared Test for count data. Calculated with `mantelhaen.test(x = data[[variables]], y = data[[by]], z = data[[strata]], ...)`.

Usage

```
ard_stats_mantelhaen_test(data, by, variables, strata, ...)
```

Arguments

data	(data.frame) a data frame.
by	(tidy-select) column name to compare by.
variables	(tidy-select) column names to be compared. Independent tests will be computed for each variable.
strata	(tidy-select) column name to stratify by.
...	additional arguments passed to <code>stats::mantelhaen.test(...)</code>

Value

ARD data frame

Examples

```
cards::ADSL |>
  ard_stats_mantelhaen_test(by = "ARM", variables = "AGEGR1", strata = "SEX")
```

ard_stats_mcnemar_test

ARD McNemar's Test

Description

Analysis results data for McNemar's statistical test. We have two functions depending on the structure of the data.

- `ard_stats_mcnemar_test()` is the structure expected by `stats::mcnemar.test()`
- `ard_stats_mcnemar_test_long()` is one row per ID per group

Usage

```
ard_stats_mcnemar_test(data, by, variables, ...)
```

```
ard_stats_mcnemar_test_long(data, by, variables, id, ...)
```

Arguments

<code>data</code>	(<code>data.frame</code>) a data frame. See below for details.
<code>by</code>	(tidy-select) column name to compare by.
<code>variables</code>	(tidy-select) column names to be compared. Independent tests will be computed for each variable.
<code>...</code>	arguments passed to <code>stats::mcnemar.test(...)</code>
<code>id</code>	(tidy-select) column name of the subject or participant ID

Details

For the `ard_stats_mcnemar_test()` function, the data is expected to be one row per subject. The data is passed as `stats::mcnemar.test(x = data[[variable]], y = data[[by]], ...)`. Please use `table(x = data[[variable]], y = data[[by]])` to check the contingency table.

Value

ARD data frame

Examples

```
cards::ADSL |>
  ard_stats_mcnemar_test(by = "SEX", variables = "EFFFL")

set.seed(1234)
cards::ADSL[c("USUBJID", "TRT01P")] |>
  dplyr::mutate(TYPE = "PLANNED") |>
  dplyr::rename(TRT01 = TRT01P) %>%
  dplyr::bind_rows(dplyr::mutate(., TYPE = "ACTUAL", TRT01 = sample(TRT01))) |>
  ard_stats_mcnemar_test_long(
    by = TYPE,
    variable = TRT01,
    id = USUBJID
  )
```

ard_stats_mood_test	<i>ARD Mood Test</i>
---------------------	----------------------

Description

Analysis results data for Mood two sample test of scale. Note this not to be confused with the Brown-Mood test of medians.

Usage

```
ard_stats_mood_test(data, by, variables, ...)
```

Arguments

data	(data.frame) a data frame. See below for details.
by	(tidy-select) column name to compare by.
variables	(tidy-select) column name to be compared. Independent tests will be run for each variable.
...	arguments passed to <code>mood.test(...)</code>

Details

For the `ard_stats_mood_test()` function, the data is expected to be one row per subject. The data is passed as `mood.test(data[[variable]] ~ data[[by]], ...)`.

Value

ARD data frame

Examples

```
cards::ADSL |>
  ard_stats_mood_test(by = "SEX", variables = "AGE")
```

ard_stats_oneway_test *ARD One-way Test*

Description

Analysis results data for Testing Equal Means in a One-Way Layout. calculated with `oneway.test()`

Usage

```
ard_stats_oneway_test(formula, data, ...)
```

Arguments

formula	a formula of the form <code>lhs ~ rhs</code> where <code>lhs</code> gives the sample values and <code>rhs</code> the corresponding groups.
data	an optional matrix or data frame (or similar: see model.frame) containing the variables in the formula <code>formula</code> . By default the variables are taken from <code>environment(formula)</code> .
...	additional arguments passed to <code>oneway.test(...)</code>

Value

ARD data frame

Examples

```
ard_stats_oneway_test(AGE ~ ARM, data = cards::ADSL)
```

ard_stats_poisson_test

ARD Poisson Test

Description

Analysis results data for exact tests of a simple null hypothesis about the rate parameter in Poisson distribution, or the comparison of two rate parameters.

Usage

```
ard_stats_poisson_test(
  data,
  variables,
  na.rm = TRUE,
  by = NULL,
  conf.level = 0.95,
  ...
)
```

Arguments

data	(data.frame) a data frame. See below for details.
variables	(tidy-select) names of the event and time variables (in that order) to be used in computations. Must be of length 2.
na.rm	(scalar logical) whether missing values should be removed before computations. Default is TRUE.
by	(tidy-select) optional column name to compare by.
conf.level	(scalar numeric) confidence level for confidence interval. Default is 0.95.
...	arguments passed to poisson.test() .

Details

- For the `ard_stats_poisson_test()` function, the data is expected to be one row per subject.
- If `by` is not specified, an exact Poisson test of the rate parameter will be performed. Otherwise, a Poisson comparison of two rate parameters will be performed on the levels of `by`. If `by` has more than 2 levels, an error will occur.

Value

an ARD data frame of class 'card'

Examples

```
# Exact test of rate parameter against null hypothesis
cards::ADTTE |>
  ard_stats_poisson_test(variables = c(CNSR, AVAL))

# Comparison test of ratio of 2 rate parameters against null hypothesis
cards::ADTTE |>
  dplyr::filter(TRTA %in% c("Placebo", "Xanomeline High Dose")) |>
  ard_stats_poisson_test(by = TRTA, variables = c(CNSR, AVAL))
```

ard_stats_prop_test	<i>ARD 2-sample proportion test</i>
---------------------	-------------------------------------

Description

Analysis results data for a 2-sample test or proportions using `stats::prop.test()`.

Usage

```
ard_stats_prop_test(data, by, variables, conf.level = 0.95, ...)
```

Arguments

data	(data.frame) a data frame.
by	(tidy-select) column name to compare by
variables	(tidy-select) column names to be compared. Must be a binary column coded as TRUE/FALSE or 1/0. Independent tests will be computed for each variable.
conf.level	(scalar numeric) confidence level for confidence interval. Default is 0.95.
...	arguments passed to <code>prop.test(...)</code>

Value

ARD data frame

Examples

```
mtcars |>
  ard_stats_prop_test(by = vs, variables = am)
```

ard_stats_t_test	<i>ARD t-test</i>
------------------	-------------------

Description

Analysis results data for paired and non-paired t-tests.

Usage

```
ard_stats_t_test(data, variables, by = NULL, conf.level = 0.95, ...)
```

```
ard_stats_paired_t_test(data, by, variables, id, conf.level = 0.95, ...)
```

Arguments

data	(data.frame) a data frame. See below for details.
variables	(tidy-select) column names to be compared. Independent t-tests will be computed for each variable.
by	(tidy-select) optional column name to compare by.
conf.level	(scalar numeric) confidence level for confidence interval. Default is 0.95.
...	arguments passed to <code>t.test()</code>
id	(tidy-select) column name of the subject or participant ID

Details

For the `ard_stats_t_test()` function, the data is expected to be one row per subject. The data is passed as `t.test(data[[variable]] ~ data[[by]], paired = FALSE, ...)`.

For the `ard_stats_paired_t_test()` function, the data is expected to be one row per subject per by level. Before the t-test is calculated, the data are reshaped to a wide format to be one row per subject. The data are then passed as `t.test(x = data_wide[<by level 1>], y = data_wide[<by level 2>], paired =`

Value

ARD data frame

Examples

```
cards::ADSL |>
  dplyr::filter(ARM %in% c("Placebo", "Xanomeline High Dose")) |>
  ard_stats_t_test(by = ARM, variables = c(AGE, BMIBL))
```

```
# constructing a paired data set,
# where patients receive both treatments
cards::ADSL[c("ARM", "AGE")] |>
  dplyr::filter(ARM %in% c("Placebo", "Xanomeline High Dose")) |>
  dplyr::mutate(.by = ARM, USUBJID = dplyr::row_number()) |>
  dplyr::arrange(USUBJID, ARM) |>
  ard_stats_paired_t_test(by = ARM, variables = AGE, id = USUBJID)
```

ard_stats_t_test_onesample

ARD one-sample t-test

Description

Analysis results data for one-sample t-tests. Result may be stratified by including the `by` argument.

Usage

```
ard_stats_t_test_onesample(
  data,
  variables,
  by = dplyr::group_vars(data),
  conf.level = 0.95,
  ...
)
```

Arguments

<code>data</code>	(data.frame) a data frame. See below for details.
<code>variables</code>	(tidy-select) column names to be analyzed. Independent t-tests will be computed for each variable.
<code>by</code>	(tidy-select) optional column name to stratify results by.
<code>conf.level</code>	(scalar numeric) confidence level for confidence interval. Default is 0.95.
<code>...</code>	arguments passed to <code>t.test()</code>

Value

ARD data frame

Examples

```
cards::ADSL |>
  ard_stats_t_test_onesample(by = ARM, variables = AGE)
```

ard_stats_wilcox_test *ARD Wilcoxon Rank-Sum Test*

Description

Analysis results data for paired and non-paired Wilcoxon Rank-Sum tests.

Usage

```
ard_stats_wilcox_test(data, variables, by = NULL, conf.level = 0.95, ...)
ard_stats_paired_wilcox_test(data, by, variables, id, conf.level = 0.95, ...)
```

Arguments

data	(data.frame) a data frame. See below for details.
variables	(tidy-select) column names to be compared. Independent tests will be computed for each variable.
by	(tidy-select) optional column name to compare by.
conf.level	(scalar numeric) confidence level for confidence interval. Default is 0.95.
...	arguments passed to <code>wilcox.test(...)</code>
id	(tidy-select) column name of the subject or participant ID.

Details

For the `ard_stats_wilcox_test()` function, the data is expected to be one row per subject. The data is passed as `wilcox.test(data[[variable]] ~ data[[by]], paired = FALSE, ...)`.

For the `ard_stats_paired_wilcox_test()` function, the data is expected to be one row per subject per by level. Before the test is calculated, the data are reshaped to a wide format to be one row per subject. The data are then passed as `wilcox.test(x = data_wide[<by level 1>], y = data_wide[<by level 2>])`.

Value

ARD data frame

Examples

```
cards::ADSL |>
  dplyr::filter(ARM %in% c("Placebo", "Xanomeline High Dose")) |>
  ard_stats_wilcox_test(by = "ARM", variables = "AGE")
```

```
# constructing a paired data set,
# where patients receive both treatments
cards::ADSL[c("ARM", "AGE")] |>
  dplyr::filter(ARM %in% c("Placebo", "Xanomeline High Dose")) |>
  dplyr::mutate(.by = ARM, USUBJID = dplyr::row_number()) |>
  dplyr::arrange(USUBJID, ARM) |>
  ard_stats_paired_wilcox_test(by = ARM, variables = AGE, id = USUBJID)
```

ard_stats_wilcox_test_onesample

ARD one-sample Wilcox Rank-sum

Description

Analysis results data for one-sample Wilcox Rank-sum. Result may be stratified by including the `by` argument.

Usage

```
ard_stats_wilcox_test_onesample(
  data,
  variables,
  by = dplyr::group_vars(data),
  conf.level = 0.95,
  ...
)
```

Arguments

<code>data</code>	(data.frame) a data frame. See below for details.
<code>variables</code>	(tidy-select) column names to be analyzed. Independent Wilcox Rank-sum tests will be computed for each variable.
<code>by</code>	(tidy-select) optional column name to stratify results by.
<code>conf.level</code>	(scalar numeric) confidence level for confidence interval. Default is 0.95.
<code>...</code>	arguments passed to <code>wilcox.test(...)</code>

Value

ARD data frame

Examples

```
cards::ADSL |>
  ard_stats_wilcox_test_onesample(by = ARM, variables = AGE)
```

```
ard_summary.survey.design
```

ARD Continuous Survey Statistics

Description

Returns an ARD of weighted statistics using the {survey} package.

Usage

```
## S3 method for class 'survey.design'
ard_summary(
  data,
  variables,
  by = NULL,
  statistic = everything() ~ c("median", "p25", "p75"),
  fmt_fun = NULL,
  stat_label = NULL,
  fmt_fn = deprecated(),
  ...
)
```

Arguments

data	(survey.design) a design object often created with <code>survey::svydesign()</code> .
variables	(tidy-select) columns to include in summaries.
by	(tidy-select) results are calculated for all combinations of the columns specified, including unobserved combinations and unobserved factor levels.
statistic	(formula-list-selector) a named list, a list of formulas, or a single formula where the list element is a character vector of statistic names to include. See below for options.
fmt_fun	(formula-list-selector) a named list, a list of formulas, or a single formula where the list element is a named list of functions (or the RHS of a formula), e.g. <code>list(mpg = list(mean = \(x) round(x, digits</code>
stat_label	(formula-list-selector) a named list, a list of formulas, or a single formula where the list element is either a named list or a list of formulas defining the statistic labels, e.g. <code>everything() ~ list(mean = "Mean", sd = "SD")</code> or <code>everything() ~ list(mean ~ "Mean", sd ~ "SD")</code> .

fmt_fn	[Deprecated]
...	These dots are for future extensions and must be empty.

Value

an ARD data frame of class 'card'

statistic argument

The following statistics are available: 'mean', 'median', 'min', 'max', 'sum', 'var', 'sd', 'mean.std.error', 'deff', 'p##', where 'p##' is the percentiles and ## is an integer between 0 and 100.

The design effect ("deff") is calculated only when requested in the statistic argument.

Examples

```
data(api, package = "survey")
dclus1 <- survey::svydesign(id = ~dnum, weights = ~pw, data = apiclus1, fpc = ~fpc)

ard_summary(
  data = dclus1,
  variables = api00,
  by = stype
)
```

ard_survey_svychisq	<i>ARD Survey Chi-Square Test</i>
---------------------	-----------------------------------

Description

Analysis results data for survey Chi-Square test using [survey::svychisq\(\)](#). Only two-way comparisons are supported.

Usage

```
ard_survey_svychisq(data, by, variables, statistic = "F", ...)
```

Arguments

data	(survey.design) a survey design object often created with the {survey} package
by	(tidy-select) column name to compare by.
variables	(tidy-select) column names to be compared. Independent tests will be computed for each variable.

statistic	(character) statistic used to estimate Chisq p-value. Default is the Rao-Scott second-order correction ("F"). See survey::svychisq for available statistics options.
...	arguments passed to survey::svychisq() .

Value

ARD data frame

Examples

```
data(api, package = "survey")
dclus1 <- survey::svydesign(id = ~dnum, weights = ~pw, data = apiclus1, fpc = ~fpc)

ard_survey_svychisq(dclus1, variables = sch.wide, by = comp.imp, statistic = "F")
```

ard_survey_svyranktest

ARD Survey rank test

Description

Analysis results data for survey wilcox test using [survey::svyranktest\(\)](#).

Usage

```
ard_survey_svyranktest(data, by, variables, test, ...)
```

Arguments

data	(survey.design) a survey design object often created with survey::svydesign()
by	(tidy-select) column name to compare by
variables	(tidy-select) column names to be compared. Independent tests will be run for each variable.
test	(string) a string to denote which rank test to use: "wilcoxon", "vanderWaerden", "median", "KruskalWallis"
...	arguments passed to survey::svyranktest()

Value

ARD data frame

Examples

```
data(api, package = "survey")
dclus2 <- survey::svydesign(id = ~ dnum + snum, fpc = ~ fpc1 + fpc2, data = apiclus2)

ard_survey_svyranktest(dclus2, variables = enroll, by = comp.imp, test = "wilcoxon")
ard_survey_svyranktest(dclus2, variables = enroll, by = comp.imp, test = "vanderWaerden")
ard_survey_svyranktest(dclus2, variables = enroll, by = comp.imp, test = "median")
ard_survey_svyranktest(dclus2, variables = enroll, by = comp.imp, test = "KruskalWallis")
```

ard_survey_svytttest *ARD Survey t-test*

Description

Analysis results data for survey t-test using `survey::svytttest()`.

Usage

```
ard_survey_svytttest(data, by, variables, conf.level = 0.95, ...)
```

Arguments

<code>data</code>	(<code>survey.design</code>) a survey design object often created with <code>survey::svydesign()</code>
<code>by</code>	(<code>tidy-select</code>) column name to compare by
<code>variables</code>	(<code>tidy-select</code>) column names to be compared. Independent tests will be run for each variable.
<code>conf.level</code>	(<code>double</code>) confidence level of the returned confidence interval. Must be between $c(0, 1)$. Default is 0.95
<code>...</code>	arguments passed to <code>survey::svytttest()</code>

Value

ARD data frame

Examples

```
data(api, package = "survey")
dclus2 <- survey::svydesign(id = ~ dnum + snum, fpc = ~ fpc1 + fpc2, data = apiclus2)

ard_survey_svytttest(dclus2, variables = enroll, by = comp.imp, conf.level = 0.9)
```

ard_survival_survdiff *ARD for Difference in Survival*

Description

Analysis results data for comparison of survival using `survival::survdiff()`.

Usage

```
ard_survival_survdiff(formula, data, rho = 0, ...)
```

Arguments

formula	(formula) a formula
data	(data.frame) a data frame
rho	(scalar numeric) numeric scalar passed to <code>survival::survdiff(rho)</code> . Default is <code>rho=0</code> .
...	additional arguments passed to <code>survival::survdiff()</code>

Value

an ARD data frame of class 'card'

Examples

```
library(survival)
library(ggsurvfit)

ard_survival_survdiff(Surv_CNSR(AVAL, CNSR) ~ TRTA, data = cards::ADTTE)
```

ard_survival_survfit *ARD Survival Estimates*

Description

Analysis results data for survival quantiles and x-year survival estimates, extracted from a `survival::survfit()` model.

Usage

```
ard_survival_survfit(x, ...)

## S3 method for class 'survfit'
ard_survival_survfit(x, times = NULL, probs = NULL, type = NULL, ...)

## S3 method for class 'data.frame'
ard_survival_survfit(
  x,
  y,
  variables = NULL,
  times = NULL,
  probs = NULL,
  type = NULL,
  method.args = list(conf.int = 0.95, conf.type = "log"),
  ...
)
```

Arguments

x	(survfit or data.frame) an object of class survfit created with <code>survival::survfit()</code> or a data frame. See below for details.								
...	These dots are for future extensions and must be empty.								
times	(numeric) a vector of times for which to return survival probabilities.								
probs	(numeric) a vector of probabilities with values in (0,1) specifying the survival quantiles to return.								
type	(string or NULL) type of statistic to report. Available for Kaplan-Meier time estimates only, otherwise type is ignored. Default is NULL. Must be one of the following:								
	<table border="0" style="margin-left: 100px;"> <tr> <td style="text-align: right; padding-right: 10px;">type</td> <td>transformation</td> </tr> <tr> <td style="text-align: right; padding-right: 10px;">"survival"</td> <td>x</td> </tr> <tr> <td style="text-align: right; padding-right: 10px;">"risk"</td> <td>1 - x</td> </tr> <tr> <td style="text-align: right; padding-right: 10px;">"cumhaz"</td> <td>-log(x)</td> </tr> </table>	type	transformation	"survival"	x	"risk"	1 - x	"cumhaz"	-log(x)
type	transformation								
"survival"	x								
"risk"	1 - x								
"cumhaz"	-log(x)								
y	(Surv or string) an object of class Surv created using <code>survival::Surv()</code> . This object will be passed as the left-hand side of the formula constructed and passed to <code>survival::survfit()</code> . This object can also be passed as a string.								
variables	(tidy-select) stratification variables to be passed as the right-hand side of the formula constructed and passed to <code>survival::survfit()</code> . Default is NULL for an unstratified model, e.g. <code>Surv() ~ 1</code> .								

method.args (named list)
named list of arguments that will be passed to `survival::survfit()`.

Details

- Only one of either the times or probs parameters can be specified.
- Times should be provided using the same scale as the time variable used to fit the provided survival fit model.

Value

an ARD data frame of class 'card'

Formula Specification

When passing a `survival::survfit()` object to `ard_survival_survfit()`, the `survfit()` call must use an evaluated formula and not a stored formula. Including a proper formula in the call allows the function to accurately identify all variables included in the estimation. See below for examples:

```
library(cardx)
library(survival)

# include formula in `survfit()` call
survfit(Surv(time, status) ~ sex, lung) |> ard_survival_survfit(time = 500)

# you can also pass a data frame to `ard_survival_survfit()` as well.
lung |>
  ard_survival_survfit(y = Surv(time, status), variables = "sex", time = 500)
```

You **cannot**, however, pass a stored formula, e.g. `survfit(my_formula, lung)`, but you can use stored formulas with `rlang::inject(survfit(!my_formula, lung))`.

Variable Classes

When the `survfit` method is called, the class of the stratifying variables will be returned as a factor.

When the data frame method is called, the original classes are retained in the resulting ARD.

Examples

```
library(survival)
library(ggsurvfit)

survfit(Surv_CNSR(AVAL, CNSR) ~ TRTA, data = cards::ADTTE) |>
  ard_survival_survfit(times = c(60, 180))

survfit(Surv_CNSR(AVAL, CNSR) ~ TRTA, data = cards::ADTTE, conf.int = 0.90) |>
  ard_survival_survfit(probs = c(0.25, 0.5, 0.75))

cards::ADTTE |>
```

```
ard_survival_survfit(y = Surv_CNSR(AVAL, CNSR), variables = c("TRTA", "SEX"), times = 90)

# Competing Risks Example -----
set.seed(1)
ADTTE_MS <- cards::ADTTE %>%
  dplyr::mutate(
    CNSR = dplyr::case_when(
      CNSR == 0 ~ "censor",
      runif(dplyr::n()) < 0.5 ~ "death from cancer",
      TRUE ~ "death other causes"
    ) %>% factor()
  )

survfit(Surv(AVAL, CNSR) ~ TRTA, data = ADTTE_MS) %>%
  ard_survival_survfit(times = c(60, 180))
```

ard_survival_survfit_diff

ARD Survival Differences

Description

Calculate differences in the Kaplan-Meier estimator of survival using the results from [survival::survfit\(\)](#).

Usage

```
ard_survival_survfit_diff(x, times, conf.level = 0.95)
```

Arguments

x	(survfit) object of class 'survfit' typically created with survival::survfit()
times	(numeric) a vector of times for which to return survival probabilities.
conf.level	(scalar numeric) confidence level for confidence interval. Default is 0.95.

Value

an ARD data frame of class 'card'

Examples

```
library(ggsurvfit)
library(survival)

survfit(Surv_CNSR() ~ TRTA, data = cards::ADTTE) |>
  ard_survival_survfit_diff(times = c(25, 50))
```

ard_tabulate.survey.design

ARD Categorical Survey Statistics

Description

Compute tabulations on survey-weighted data.

The counts and proportion ("N", "n", "p") are calculated using `survey::svytable()`, and the standard errors and design effect ("p.std.error", "deff") are calculated using `survey::svymean()`.

The design effect ("deff") is calculated only when requested in the `statistic` argument.

The unweighted statistics are calculated with `cards::ard_tabulate.data.frame()`.

Usage

```
## S3 method for class 'survey.design'
ard_tabulate(
  data,
  variables,
  by = NULL,
  statistic = everything() ~ c("n", "N", "p", "p.std.error", "n_unweighted",
    "N_unweighted", "p_unweighted"),
  denominator = c("column", "row", "cell"),
  fmt_fun = NULL,
  stat_label = everything() ~ list(p = "%", p.std.error = "SE(%)", deff =
    "Design Effect", n_unweighted = "Unweighted n", N_unweighted = "Unweighted N",
    p_unweighted = "Unweighted %"),
  fmt_fn = deprecated(),
  ...
)
```

Arguments

<code>data</code>	(<code>survey.design</code>) a design object often created with <code>survey::svydesign()</code> .
<code>variables</code>	(<code>tidy-select</code>) columns to include in summaries.
<code>by</code>	(<code>tidy-select</code>) results are calculated for all combinations of the column specified and the variables. A single column may be specified.
<code>statistic</code>	(<code>formula-list-selector</code>) a named list, a list of formulas, or a single formula where the list element is a character vector of statistic names to include. See default value for options.
<code>denominator</code>	(string) a string indicating the type proportions to calculate. Must be one of "column" (the default), "row", and "cell".

fmt_fun	(formula-list-selector) a named list, a list of formulas, or a single formula where the list element is a named list of functions (or the RHS of a formula), e.g. <code>list(mpg = list(mean = \"(x) round(x, digits\"</code>
stat_label	(formula-list-selector) a named list, a list of formulas, or a single formula where the list element is either a named list or a list of formulas defining the statistic labels, e.g. <code>everything() ~ list(mean = "Mean", sd = "SD")</code> or <code>everything() ~ list(mean ~ "Mean", sd ~ "SD")</code> .
fmt_fn	[Deprecated]
...	These dots are for future extensions and must be empty.

Value

an ARD data frame of class 'card'

Examples

```
svy_titanic <- survey::svydesign(~1, data = as.data.frame(Titanic), weights = ~Freq)

ard_tabulate(svy_titanic, variables = c(Class, Age), by = Survived)
```

ard_tabulate_abnormal *ARD Abnormality Counts*

Description

Function counts participants with abnormal analysis range values.

For each abnormality specified via the abnormal parameter (e.g. Low or High), statistic n is calculated as the number of patients with this abnormality recorded, and statistic N is calculated as the total number of patients with at least one post-baseline assessment. p is calculated as n / N. If `excl_baseline_abn=TRUE` then participants with abnormality at baseline are excluded from all statistic calculations.

Usage

```
ard_tabulate_abnormal(
  data,
  postbaseline,
  baseline,
  id = NULL,
  by = NULL,
  strata = NULL,
  abnormal = list(Low = "LOW", High = "HIGH"),
  excl_baseline_abn = TRUE,
  quiet = FALSE
)
```

Arguments

data	(data.frame) a data frame.
postbaseline	(tidy-select) column name of post-baseline reference range indicator variable.
baseline	(tidy-select) column name of baseline reference range indicator variable.
id	(tidy-select) column name used to identify unique participants in data. If NULL, each row in data is assumed to correspond to a unique participants.
by, strata	(tidy-select) columns to tabulate by/stratify by for summary statistic calculation. Arguments are similar, but with an important distinction: by: results are calculated for all combinations of the columns specified, including unobserved combinations and unobserved factor levels. strata: results are calculated for all observed combinations of the columns specified. Arguments may be used in conjunction with one another.
abnormal	(list) a named list of abnormalities to assess for. Each element should specify all levels of postbaseline/baseline that should be included when assessing for a given abnormality, with the name specifying the name of the abnormality. Any levels specified but not present in the data are ignored.
excl_baseline_abn	(logical) whether participants with baseline abnormality should be excluded from calculations. Defaults to TRUE.
quiet	(scalar logical) logical indicating whether to suppress additional messaging. Default is FALSE.

Value

an ARD data frame of class 'card'

Examples

```
# Load Data -----
set.seed(1)
adlb <- cards::ADLB
adlb$BNRIND <- ifelse(
  adlb$BNRIND != "N",
  sample(c("LOW", "LOW LOW", "HIGH", "HIGH HIGH"), nrow(adlb), replace = TRUE),
  "NORMAL"
)

# Example 1 -----
adlb |>
```

```
ard_tabulate_abnormal(
  postbaseline = LBNRIND, baseline = BNRIND, id = USUBJID, by = TRTA,
  abnormal = list(Low = c("LOW", "LOW LOW"), High = c("HIGH", "HIGH HIGH"))
)
```

ard_tabulate_max	<i>ARD to Calculate Categorical Occurrence Rates by Maximum Level Per Unique ID</i>
------------------	---

Description

Function calculates categorical variable level occurrences rates by maximum level per unique ID. Each variable in `variables` is evaluated independently and then results for all variables are stacked. Only the highest-ordered level will be counted for each unique ID. Unordered, non-numeric variables will be converted to factor and the default level order used for ordering.

Usage

```
ard_tabulate_max(
  data,
  variables,
  id,
  by = dplyr::group_vars(data),
  statistic = everything() ~ c("n", "p", "N"),
  denominator = NULL,
  strata = NULL,
  fmt_fun = NULL,
  stat_label = everything() ~ cards::default_stat_labels(),
  quiet = FALSE,
  fmt_fn = deprecated(),
  ...
)
```

Arguments

<code>data</code>	(data.frame) a data frame
<code>variables</code>	(tidy-select) The categorical variables for which occurrence rates per unique ID (by maximum level) will be calculated.
<code>id</code>	(tidy-select) Argument used to subset data to identify rows in data to calculate categorical variable level occurrence rates.
<code>by, strata</code>	(tidy-select) columns to use for grouping or stratifying the table output. Arguments are similar, but with an important distinction:

by: results are tabulated by **all combinations** of the columns specified, including unobserved combinations and unobserved factor levels.

strata: results are tabulated by **all *observed* combinations** of the columns specified.

Arguments may be used in conjunction with one another.

statistic	(formula-list-selector) a named list, a list of formulas, or a single formula where the list element one or more of c("n", "N", "p", "n_cum", "p_cum") (on the RHS of a formula).
denominator	(data.frame, integer) An optional argument to change the denominator used for "N" and "p" statistic calculations. Defaults to NULL, in which case <code>dplyr::distinct(data, dplyr::pick(all_of(c(id, by))))</code> is used for these calculations. See cards::ard_tabulate() for more details on specifying denominators.
fmt_fun	(formula-list-selector) a named list, a list of formulas, or a single formula where the list element is a named list of functions (or the RHS of a formula), e.g. <code>list(mpg = list(mean = \(x) round(x, digits</code>
stat_label	(formula-list-selector) a named list, a list of formulas, or a single formula where the list element is either a named list or a list of formulas defining the statistic labels, e.g. <code>everything() ~ list(n = "n", p = "pct")</code> or <code>everything() ~ list(n ~ "n", p ~ "pct")</code> .
quiet	(scalar logical) Logical indicating whether to suppress additional messaging. Default is FALSE.
fmt_fn	[Deprecated]
...	Arguments passed to methods.

Value

an ARD data frame of class 'card'

Examples

```
# Occurrence Rates by Max Level (Highest Severity) -----
ard_tabulate_max(
  cards::ADAE,
  variables = c(AESER, AESEV),
  id = USUBJID,
  by = TRTA,
  denominator = cards::ADSL
)
```

ard_tabulate_value.survey.design

ARD Dichotomous Survey Statistics

Description

Compute Analysis Results Data (ARD) for dichotomous summary statistics.

Usage

```
## S3 method for class 'survey.design'
ard_tabulate_value(
  data,
  variables,
  by = NULL,
  value = cards::maximum_variable_value(data$variables[variables]),
  statistic = everything() ~ c("n", "N", "p", "p.std.error", "n_unweighted",
    "N_unweighted", "p_unweighted"),
  denominator = c("column", "row", "cell"),
  fmt_fun = NULL,
  stat_label = everything() ~ list(p = "%", p.std.error = "SE(%)", deff =
    "Design Effect", n_unweighted = "Unweighted n", N_unweighted = "Unweighted N",
    p_unweighted = "Unweighted %"),
  fmt_fn = deprecated(),
  ...
)
```

Arguments

data	(survey.design) a design object often created with <code>survey::svydesign()</code> .
variables	(tidy-select) columns to include in summaries.
by	(tidy-select) results are calculated for all combinations of the column specified and the variables. A single column may be specified.
value	(named list) named list of dichotomous values to tabulate. Default is <code>cards::maximum_variable_value(data\$variables[variables])</code> which returns the largest/last value after a sort.
statistic	(formula-list-selector) a named list, a list of formulas, or a single formula where the list element is a character vector of statistic names to include. See default value for options.
denominator	(string) a string indicating the type proportions to calculate. Must be one of "column" (the default), "row", and "cell".

fmt_fun	(formula-list-selector) a named list, a list of formulas, or a single formula where the list element is a named list of functions (or the RHS of a formula), e.g. list(mpg = list(mean = \"(x) round(x, digits\"
stat_label	(formula-list-selector) a named list, a list of formulas, or a single formula where the list element is either a named list or a list of formulas defining the statistic labels, e.g. everything() ~ list(mean = \"Mean\", sd = \"SD\") or everything() ~ list(mean ~ \"Mean\", sd ~ \"SD\").
fmt_fn	[Deprecated]
...	These dots are for future extensions and must be empty.

Value

an ARD data frame of class 'card'

Examples

```
survey::svydesign(ids = ~1, data = mtcars, weights = ~1) |>
  ard_tabulate_value(by = vs, variables = c(cyl, am), value = list(cyl = 4))
```

ard_total_n.survey.design
<i>ARD Total N</i>

Description

Returns the total N for a survey object. The placeholder variable name returned in the object is `"..ard_total_n.."`

Usage

```
## S3 method for class 'survey.design'
ard_total_n(data, ...)
```

Arguments

data	(survey.design) a design object often created with <code>survey::svydesign()</code> .
...	These dots are for future extensions and must be empty.

Value

an ARD data frame of class 'card'

Examples

```
svy_titanic <- survey::svydesign(~1, data = as.data.frame(Titanic), weights = ~Freq)

ard_total_n(svy_titanic)
```

construction_helpers *Construction Helpers*

Description

These functions help construct calls to various types of models.

Usage

```
construct_model(data, ...)

## S3 method for class 'data.frame'
construct_model(
  data,
  formula,
  method,
  method.args = list(),
  package = "base",
  env = caller_env(),
  ...
)

## S3 method for class 'survey.design'
construct_model(
  data,
  formula,
  method,
  method.args = list(),
  package = "survey",
  env = caller_env(),
  ...
)

reformulate2(
  termlabels,
  response = NULL,
  intercept = TRUE,
  env = parent.frame(),
  pattern_term = NULL,
  pattern_response = NULL
)
```

```
bt(x, pattern = NULL)
```

```
bt_strip(x)
```

Arguments

data	• <code>construct_model.data.frame()</code> (<code>data.frame</code>) a data frame • <code>construct_model.survey.design()</code> (<code>survey.design</code>) a survey design object
...	These dots are for future extensions and must be empty.
formula	(formula) a formula
method	(string) string of function naming the function to be called, e.g. "glm". If function belongs to a library that is not attached, the package name must be specified in the package argument.
method.args	(named list) named list of arguments that will be passed to method. Note that this list may contain non-standard evaluation components. If you are wrapping this function in other functions, the argument must be passed in a way that does not evaluate the list, e.g. using <code>rlang</code> 's embrace operator <code>{{ . }}</code> .
package	(string) a package name that will be temporarily loaded when function specified in method is executed.
env	The environment in which to evaluate <code>expr</code> . This environment is not applicable for quosures because they have their own environments.
term.labels	character vector giving the right-hand side of a model formula. May be zero-length.
response	a character string, symbol or call giving the left-hand side of a model formula, or <code>NULL</code> .
intercept	logical: should the formula have an intercept?
x	(character) character vector, typically of variable names
pattern, pattern_term, pattern_response	DEPRECATED

Details

- `construct_model()`: Builds models of the form `method(data = data, formula = formula, method.args!!!)`. If the package argument is specified, that package is temporarily attached when the model is evaluated.
- `reformulate2()`: This is a copy of `reformulate()` except that variable names that contain a space are wrapped in backticks.
- `bt()`: Adds backticks to a character vector.
- `bt_strip()`: Removes backticks from a string if it begins and ends with a backtick.

Value

depends on the calling function

Examples

```
construct_model(
  data = mtcars,
  formula = am ~ mpg + (1 | vs),
  method = "glmer",
  method.args = list(family = binomial),
  package = "lme4"
) |>
  broom.mixed::tidy()

construct_model(
  data = mtcars |> dplyr::rename(`M P G` = mpg),
  formula = reformulate2(c("M P G", "cyl"), response = "hp"),
  method = "lm"
) |>
  ard_regression() |>
  dplyr::filter(stat_name %in% c("term", "estimate", "p.value"))
```

proportion_ci

Functions for Calculating Proportion Confidence Intervals

Description

Functions to calculate different proportion confidence intervals for use in `ard_proportion()`.

Usage

```
proportion_ci_wald(x, conf.level = 0.95, correct = FALSE)

proportion_ci_wilson(x, conf.level = 0.95, correct = FALSE)

proportion_ci_clopper_pearson(x, conf.level = 0.95)

proportion_ci_agresti_coull(x, conf.level = 0.95)

proportion_ci_jeffreys(x, conf.level = 0.95)

proportion_ci_strat_wilson(
  x,
  strata,
  weights = NULL,
  conf.level = 0.95,
  max.iterations = 10L,
```

```

    correct = FALSE
  )

  is_binary(x)

```

Arguments

<code>x</code>	(binary numeric/logical) vector of a binary values, i.e. a logical vector, or numeric with values c(0, 1)
<code>conf.level</code>	(scalar numeric) a scalar in (0, 1) indicating the confidence level. Default is 0.95
<code>correct</code>	(scalar logical) include the continuity correction. For further information, see for example stats::prop.test() .
<code>strata</code>	(factor) variable with one level per stratum and same length as <code>x</code> .
<code>weights</code>	(numeric) weights for each level of the strata. If NULL, they are estimated using the iterative algorithm that minimizes the weighted squared length of the confidence interval.
<code>max.iterations</code>	(positive integer) maximum number of iterations for the iterative procedure used to find estimates of optimal weights.

Value

Confidence interval of a proportion.

Functions

- `proportion_ci_wald()`: Calculates the Wald interval by following the usual textbook definition for a single proportion confidence interval using the normal approximation.

$$\hat{p} \pm z_{\alpha/2} \sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$$

- `proportion_ci_wilson()`: Calculates the Wilson interval by calling [stats::prop.test\(\)](#). Also referred to as Wilson score interval.

$$\frac{\hat{p} + \frac{z_{\alpha/2}^2}{2n} \pm z_{\alpha/2} \sqrt{\frac{\hat{p}(1-\hat{p})}{n} + \frac{z_{\alpha/2}^2}{4n^2}}}{1 + \frac{z_{\alpha/2}^2}{n}}$$

- `proportion_ci_clopper_pearson()`: Calculates the Clopper-Pearson interval by calling [stats::binom.test\(\)](#). Also referred to as the exact method.

$$\left(\frac{k}{n} \pm z_{\alpha/2} \sqrt{\frac{\frac{k}{n}(1-\frac{k}{n})}{n} + \frac{z_{\alpha/2}^2}{4n^2}} \right) / \left(1 + \frac{z_{\alpha/2}^2}{n} \right)$$

- `proportion_ci_agresti_coull()`: Calculates the Agresti-Coull interval (created by Alan Agresti and Brent Coull) by (for 95% CI) adding two successes and two failures to the data and then using the Wald formula to construct a CI.

$$\left(\frac{\tilde{p} + z_{\alpha/2}^2/2}{n + z_{\alpha/2}^2} \pm z_{\alpha/2} \sqrt{\frac{\tilde{p}(1 - \tilde{p})}{n} + \frac{z_{\alpha/2}^2}{4n^2}} \right)$$

- `proportion_ci_jeffreys()`: Calculates the Jeffreys interval, an equal-tailed interval based on the non-informative Jeffreys prior for a binomial proportion.

$$\left(\text{Beta} \left(\frac{k}{2} + \frac{1}{2}, \frac{n-k}{2} + \frac{1}{2} \right)_{\alpha}, \text{Beta} \left(\frac{k}{2} + \frac{1}{2}, \frac{n-k}{2} + \frac{1}{2} \right)_{1-\alpha} \right)$$

- `proportion_ci_strat_wilson()`: Calculates the stratified Wilson confidence interval for unequal proportions as described in Xin YA, Su XG. Stratified Wilson and Newcombe confidence intervals for multiple binomial proportions. *Statistics in Biopharmaceutical Research*. 2010;2(3).

$$\frac{\hat{p}_j + \frac{z_{\alpha/2}^2}{2n_j} \pm z_{\alpha/2} \sqrt{\frac{\hat{p}_j(1-\hat{p}_j)}{n_j} + \frac{z_{\alpha/2}^2}{4n_j^2}}}{1 + \frac{z_{\alpha/2}^2}{n_j}}$$

- `is_binary()`: Helper to determine if vector is binary (logical or 0/1)

Examples

```
x <- c(
  TRUE, TRUE, TRUE, TRUE, TRUE,
  FALSE, FALSE, FALSE, FALSE, FALSE
)

proportion_ci_wald(x, conf.level = 0.9)
proportion_ci_wilson(x, correct = TRUE)
proportion_ci_clopper_pearson(x)
proportion_ci_agresti_coull(x)
proportion_ci_jeffreys(x)

# Stratified Wilson confidence interval with unequal probabilities

set.seed(1)
rsp <- sample(c(TRUE, FALSE), 100, TRUE)
strata_data <- data.frame(
  "f1" = sample(c("a", "b"), 100, TRUE),
  "f2" = sample(c("x", "y", "z"), 100, TRUE),
  stringsAsFactors = TRUE
)
strata <- interaction(strata_data)
n_strata <- ncol(table(rsp, strata)) # Number of strata

proportion_ci_strat_wilson(
```

```
x = rsp, strata = strata,  
  conf.level = 0.90  
)  
  
# Not automatic setting of weights  
proportion_ci_strat_wilson(  
  x = rsp, strata = strata,  
  weights = rep(1 / n_strata, n_strata),  
  conf.level = 0.90  
)
```

Index

`aod::wald.test()`, 3
`ard_aod_wald_test`, 3
`ard_attributes.survey.design`, 4
`ard_car_anova`, 5
`ard_car_vif`, 5
`ard_categorical_ci`, 6
`ard_categorical_ci.survey.design`, 7
`ard_continuous_ci`, 9
`ard_continuous_ci.survey.design`, 10
`ard_effectsize_cohens_d`, 11
`ard_effectsize_hedges_g`, 12
`ard_effectsize_paired_cohens_d`
 (`ard_effectsize_cohens_d`), 11
`ard_effectsize_paired_hedges_g`
 (`ard_effectsize_hedges_g`), 12
`ard_emmeans_contrast`, 13
`ard_emmeans_emmeans`
 (`ard_emmeans_contrast`), 13
`ard_incidence_rate`, 16
`ard_missing.survey.design`, 18
`ard_regression`, 19
`ard_regression()`, 21
`ard_regression_basic`, 21
`ard_smd_smd`, 22
`ard_stats_anova`, 23
`ard_stats_aov`, 25
`ard_stats_chisq_test`, 26
`ard_stats_fisher_test`, 27
`ard_stats_kruskal_test`, 27
`ard_stats_mantelhaen_test`, 28
`ard_stats_mcnemar_test`, 29
`ard_stats_mcnemar_test_long`
 (`ard_stats_mcnemar_test`), 29
`ard_stats_mood_test`, 30
`ard_stats_oneway_test`, 31
`ard_stats_paired_t_test`
 (`ard_stats_t_test`), 34
`ard_stats_paired_wilcox_test`
 (`ard_stats_wilcox_test`), 36
`ard_stats_poisson_test`, 32
`ard_stats_prop_test`, 33
`ard_stats_t_test`, 34
`ard_stats_t_test_onesample`, 35
`ard_stats_wilcox_test`, 36
`ard_stats_wilcox_test_onesample`, 37
`ard_summary.survey.design`, 38
`ard_survey_svychisq`, 39
`ard_survey_svyranktest`, 40
`ard_survey_svytttest`, 41
`ard_survival_survdiff`, 42
`ard_survival_survfit`, 42
`ard_survival_survfit_diff`, 45
`ard_tabulate.survey.design`, 46
`ard_tabulate_abnormal`, 47
`ard_tabulate_max`, 49
`ard_tabulate_value.survey.design`, 51
`ard_total_n.survey.design`, 52

`broom.helpers::tidy_plus_plus()`, 20, 21
`broom.helpers::tidy_with_broom_or_parameters`,
 3, 20, 22
`bt (construction_helpers)`, 53
`bt_strip (construction_helpers)`, 53

`car::Anova()`, 5
`car::vif()`, 5
`cards::ard_tabulate()`, 50
`construct_model (construction_helpers)`,
 53
`construction_helpers`, 53

`effectsize::cohens_d()`, 11
`effectsize::hedges_g()`, 12

`is_binary (proportion_ci)`, 55

`model.frame`, 31

`poisson.test()`, 32
`proportion_ci`, 55

`proportion_ci_agresti_coull`
 (`proportion_ci`), 55
`proportion_ci_clopper_pearson`
 (`proportion_ci`), 55
`proportion_ci_jeffreys` (`proportion_ci`),
 55
`proportion_ci_strat_wilson`
 (`proportion_ci`), 55
`proportion_ci_wald` (`proportion_ci`), 55
`proportion_ci_wilson` (`proportion_ci`), 55

`reformulate2` (`construction_helpers`), 53

`smd::smd()`, 22
`stats::binom.test()`, 56
`stats::mcnemar.test()`, 29
`stats::prop.test()`, 33, 56
`survey::svychisq`, 40
`survey::svychisq()`, 39, 40
`survey::svyciprop()`, 7
`survey::svydesign()`, 4, 8, 10, 18, 23, 38,
 40, 41, 46, 51, 52
`survey::svyranktest()`, 40
`survey::svyttest()`, 41
`survival::Surv()`, 43
`survival::survdiff()`, 42
`survival::survfit()`, 42–45