Package 'cards'

November 27, 2024

Title Analysis Results Data

Version 0.4.0

Description Construct CDISC (Clinical Data Interchange Standards Consortium) compliant Analysis Results Data objects. These objects are used and re-used to construct summary tables, visualizations, and written reports. The package also exports utilities for working with these objects and creating new Analysis Results Data objects.

License Apache License 2.0

URL https://github.com/insightsengineering/cards, https://insightsengineering.github.io/cards/

BugReports https://github.com/insightsengineering/cards/issues

Depends R (>= 4.1)

Imports cli (>= 3.6.1), dplyr (>= 1.1.2), glue (>= 1.6.2), rlang (>= 1.1.1), tidyr (>= 1.3.0), tidyselect (>= 1.2.0)

Suggests spelling (>= 2.2.0), testthat (>= 3.2.0), withr (>= 3.0.0)

Config/Needs/check hms

Config/Needs/website rmarkdown, jsonlite, yaml, gtsummary, tfrmt, insightsengineering/nesttemplate

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.3.2

NeedsCompilation no

Author Daniel D. Sjoberg [aut, cre] (<https://orcid.org/0000-0003-0862-2018>),
Becca Krouse [aut],
Emily de la Rua [aut],
F. Hoffmann-La Roche AG [cph, fnd],
GlaxoSmithKline Research & Development Limited [cph]

Maintainer Daniel D. Sjoberg <danield.sjoberg@gmail.com>

Repository CRAN

Date/Publication 2024-11-27 06:50:02 UTC

Contents

`adam` *Example ADaM Data*

Description

Data frame imported from the CDISC SDTM/ADaM Pilot Project

Usage

```
ADSL

ADAE

ADTTE
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 254 rows and 48 columns.

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 1191 rows and 55 columns.

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 254 rows and 26 columns.

`add_calculated_row` *Add Calculated Row*

Description

Use this function to add a new statistic row that is a function of the other statistics in an ARD.

Usage

```
add_calculated_row(
  x,
  expr,
  stat_name,
  by = c(all_ard_groups(), all_ard_variables(), any_of("context")),
  stat_label = stat_name,
  fmt_fn = NULL
)
```

Arguments

x	(card) data frame of class 'card'
expr	(expression) an expression
stat_name	(string) string naming the new statistic
by	(tidy-select) Grouping variables to calculate statistics within
stat_label	(string) string of the statistic label. Default is the stat_name.
fmt_fn	(integer, function, string) a function of an integer or string that can be converted to a function with alias_as_fmt_fn().

Value

an ARD data frame of class 'card'

Examples

```
ard_continuous(mtcars, variables = mpg) |>
  add_calculated_row(expr = max - min, stat_name = "range")

ard_continuous(mtcars, variables = mpg) |>
  add_calculated_row(
    expr =
      dplyr::case_when(
        mean > median ~ "Right Skew",
        mean < median ~ "Left Skew",
        .default = "Symmetric"
      ),
    stat_name = "skew"
  )
```

---

alias_as_fmt_fn *Convert Alias to Function*

Description

Accepted aliases are non-negative integers and strings.

The integers are converted to functions that round the statistics to the number of decimal places to match the integer.

The formatting strings come in the form "xx", "xx.x", "xx.x%", etc. The number of xs that appear after the decimal place indicate the number of decimal places the statistics will be rounded to. The number of xs that appear before the decimal place indicate the leading spaces that are added to the result. If the string ends in "%", results are scaled by 100 before rounding.

Usage

```
alias_as_fmt_fn(x, variable, stat_name)
```

Arguments

x	(integer, string, or function) a non-negative integer, string alias, or function
variable	(character) the variable whose statistic is to be formatted
stat_name	(character) the name of the statistic that is to be formatted

Value

a function

Examples

```
alias_as_fmt_fn(1)
alias_as_fmt_fn("xx.x")
```

---

apply_fmt_fn *Apply Formatting Functions*

Description

Apply the formatting functions to each of the raw statistics. Function aliases are converted to functions using alias_as_fmt_fn().

Usage

```
apply_fmt_fn(x, replace = FALSE)
```

Arguments

x	(data.frame) an ARD data frame of class 'card'
replace	(scalar logical) logical indicating whether to replace values in the 'stat_fmt' column (if present). Default is FALSE.

Value

an ARD data frame of class 'card'

Examples

```
ard_continuous(ADSL, variables = "AGE") |>
  apply_fmt_fn()
```

`ard_attributes` *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

```
ard_attributes(data, ...)

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

## Default S3 method:
ard_attributes(data, ...)
```

Arguments

`data`	(`data.frame`) a data frame
`...`	These dots are for future extensions and must be empty.
`variables`	(`tidy-select`) variables to include
`label`	(named `list`) named list of variable labels, e.g. `list(cyl = "No. Cylinders")`. Default is `NULL`

Value

an ARD data frame of class 'card'

Examples

```
df <- dplyr::tibble(var1 = letters, var2 = LETTERS)
attr(df$var1, "label") <- "Lowercase Letters"

ard_attributes(df, variables = everything())
```

`ard_categorical` *Categorical ARD Statistics*

Description

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

Usage

```
ard_categorical(data, ...)

## S3 method for class 'data.frame'
ard_categorical(
  data,
  variables,
  by = dplyr::group_vars(data),
  strata = NULL,
  statistic = everything() ~ c("n", "p", "N"),
  denominator = NULL,
  fmt_fn = NULL,
  stat_label = everything() ~ default_stat_labels(),
  ...
)
```

Arguments

`data`	(`data.frame`) a data frame
`...`	Arguments passed to methods.
`variables`	(`tidy-select`) columns to include in summaries. Default is `everything()`.
`by`, `strata`	(`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")` (or the RHS of a formula).
`denominator`	(`data.frame`, `integer`) Specify this *optional* argument to change the denominator, e.g. the `"N"` statistic. Default is `NULL`. See below for details.

`fmt_fn` (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(n = "n", p = "pct")` or `everything() ~ list(n ~ "n", p ~ "pct")`.

Value

an ARD data frame of class 'card'

Denominators

By default, the `ard_categorical()` function returns the statistics `"n"`, `"N"`, and `"p"`, where little `"n"` are the counts for the variable levels, and big `"N"` is the number of non-missing observations. The default calculation for the percentage is merely `p = n/N`.

However, it is sometimes necessary to provide a different `"N"` to use as the denominator in this calculation. For example, in a calculation of the rates of various observed adverse events, you may need to update the denominator to the number of enrolled subjects.

In such cases, use the `denominator` argument to specify a new definition of `"N"`, and subsequently `"p"`. The argument expects one of the following inputs:

- a data frame. Any columns in the data frame that overlap with the `by`/`strata` columns will be used to calculate the new `"N"`.
- an integer. This single integer will be used as the new `"N"`
- a string: one of `"column"`, `"row"`, or `"cell"`. `"column"` is equivalent to `denominator=NULL`. `"row"` gives 'row' percentages where `by`/`strata` columns are the 'top' of a cross table, and the variables are the rows. `"cell"` gives percentages where the denominator is the number of non-missing rows in the source data frame.
- a structured data frame. The data frame will include columns from `by`/`strata`. The last column must be named `"...ard_N..."`. The integers in this column will be used as the updated `"N"` in the calculations.

Other Statistics

In some cases, you may need other kinds of statistics for categorical variables. Despite the name, `ard_continuous()` can be used to obtain these statistics.

In the example below, we calculate the mode of a categorical variable.

```
get_mode <- function(x) {
  table(x) |> sort(decreasing = TRUE) |> names() |> getElement(1L)
}

ADSL |>
  ard_continuous(
    variables = AGEGR1,
```

```
    statistic = list(AGEGR1 = list(mode = get_mode))
  )
#> {cards} data frame: 1 x 8
#>   variable   context stat_name stat_label  stat fmt_fn
#> 1   AGEGR1 continuo. . .       mode       mode 65-80   <fn>
#> i 2 more variables: warning, error
```

Examples

```
ard_categorical(ADSL, by = "ARM", variables = "AGEGR1")

ADSL |>
  dplyr::group_by(ARM) |>
  ard_categorical(
    variables = "AGEGR1",
    statistic = everything() ~ "n"
  )
```

`ard_complex` *Complex ARD Summaries*

Description

[Experimental]
Function is similar to `ard_continuous()`, but allows for more complex summaries. While `ard_continuous(statistic)` only allows for a univariable function, `ard_complex(statistic)` can handle more complex data summaries.

Usage

```
ard_complex(data, ...)

## S3 method for class 'data.frame'
ard_complex(
  data,
  variables,
  by = dplyr::group_vars(data),
  strata = NULL,
  statistic,
  fmt_fn = NULL,
  stat_label = everything() ~ default_stat_labels(),
  ...
)
```

Arguments

`data`	(`data.frame`) a data frame

`...` Arguments passed to methods.

`variables` (tidy-select)
columns to include in summaries. Default is `everything()`.

`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.

`statistic` (formula-list-selector)
The form of the statistics argument is identical to `ard_continuous(statistic)` argument, except the summary function *must* accept the following arguments:

- `x`: a vector
- `data`: the data frame that has been subset such that the `by`/`strata` columns and rows in which `"variable"` is `NA` have been removed.
- `full_data`: the full data frame
- `by`: character vector of the `by` variables
- `strata`: character vector of the `strata` variables It is unlikely any one function will need *all* of the above elements, and it's recommended the function passed accepts `...` so that any unused arguments will be properly ignored. The `...` also allows this function to perhaps be updated in the future with more passed arguments. For example, if one needs a second variable from the data frame, the function inputs may look like: `foo(x, data, ...)`

`fmt_fn` (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")`.

Value

an ARD data frame of class 'card'

Examples

```
# example how to mimic behavior of `ard_continuous()`
ard_complex(
  ADSL,
  by = "ARM",
  variables = "AGE",
  statistic = list(AGE = list(mean = \(x, ...) mean(x)))
```

```
)

# return the grand mean and the mean within the `by` group
grand_mean <- function(data, full_data, variable, ...) {
  list(
    mean = mean(data[[variable]], na.rm = TRUE),
    grand_mean = mean(full_data[[variable]], na.rm = TRUE)
  )
}

ADSL |>
  dplyr::group_by(ARM) |>
  ard_complex(
    variables = "AGE",
    statistic = list(AGE = list(means = grand_mean))
  )
```

`ard_continuous` *Continuous ARD Statistics*

Description

Compute Analysis Results Data (ARD) for simple continuous summary statistics.

Usage

```
ard_continuous(data, ...)

## S3 method for class 'data.frame'
ard_continuous(
  data,
  variables,
  by = dplyr::group_vars(data),
  strata = NULL,
  statistic = everything() ~ continuous_summary_fns(),
  fmt_fn = NULL,
  stat_label = everything() ~ default_stat_labels(),
  ...
)
```

Arguments

`data`	(`data.frame`) a data frame
`...`	Arguments passed to methods.
`variables`	(`tidy-select`) columns to include in summaries. Default is `everything()`.

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.

statistic (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) mean(x)))`.

The value assigned to each variable must also be a named list, where the names are used to reference a function and the element is the function object. Typically, this function will return a scalar statistic, but a function that returns a named list of results is also acceptable, e.g. `list(conf.low = -1, conf.high = 1)`. However, when errors occur, the messaging will be less clear in this setting.

fmt_fn (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")`.

Value

an ARD data frame of class 'card'

Examples

```
ard_continuous(ADSL, by = "ARM", variables = "AGE")

# if a single function returns a named list, the named
# results will be placed in the resulting ARD
ADSL |>
  dplyr::group_by(ARM) |>
  ard_continuous(
    variables = "AGE",
    statistic =
      ~ list(conf.int = \(x) t.test(x)[["conf.int"]] |>
        as.list() |>
        setNames(c("conf.low", "conf.high")))
  )
```

`ard_dichotomous` *Dichotomous ARD Statistics*

Description

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

Usage

```
ard_dichotomous(data, ...)

## S3 method for class 'data.frame'
ard_dichotomous(
  data,
  variables,
  by = dplyr::group_vars(data),
  strata = NULL,
  value = maximum_variable_value(data[variables]),
  statistic = everything() ~ c("n", "N", "p"),
  denominator = NULL,
  fmt_fn = NULL,
  stat_label = everything() ~ default_stat_labels(),
  ...
)
```

Arguments

`data`	(`data.frame`) a data frame
`...`	Arguments passed to methods.
`variables`	(tidy-select) columns to include in summaries. Default is `everything()`.
`by, strata`	(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.
`value`	(named `list`) named list of dichotomous values to tabulate. Default is `maximum_variable_value(data)`, 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 one or more of `c("n", "N", "p")` (or the RHS of a formula).

`denominator` (`data.frame`, `integer`)
Specify this *optional* argument to change the denominator, e.g. the `"N"` statistic. Default is `NULL`. See below for details.

`fmt_fn` (`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(n = "n", p = "pct")` or `everything() ~ list(n ~ "n", p ~ "pct")`.

Value

an ARD data frame of class 'card'

Denominators

By default, the `ard_categorical()` function returns the statistics `"n"`, `"N"`, and `"p"`, where little `"n"` are the counts for the variable levels, and big `"N"` is the number of non-missing observations. The default calculation for the percentage is merely `p = n/N`.

However, it is sometimes necessary to provide a different `"N"` to use as the denominator in this calculation. For example, in a calculation of the rates of various observed adverse events, you may need to update the denominator to the number of enrolled subjects.

In such cases, use the `denominator` argument to specify a new definition of `"N"`, and subsequently `"p"`. The argument expects one of the following inputs:

- a data frame. Any columns in the data frame that overlap with the `by`/`strata` columns will be used to calculate the new `"N"`.
- an integer. This single integer will be used as the new `"N"`
- a string: one of `"column"`, `"row"`, or `"cell"`. `"column"` is equivalent to `denominator=NULL`. `"row"` gives 'row' percentages where `by`/`strata` columns are the 'top' of a cross table, and the variables are the rows. `"cell"` gives percentages where the denominator is the number of non-missing rows in the source data frame.
- a structured data frame. The data frame will include columns from `by`/`strata`. The last column must be named `"...ard_N..."`. The integers in this column will be used as the updated `"N"` in the calculations.

Examples

```
ard_dichotomous(mtcars, by = vs, variables = c(cyl, am), value = list(cyl = 4))

mtcars |>
  dplyr::group_by(vs) |>
  ard_dichotomous(
    variables = c(cyl, am),
    value = list(cyl = 4),
    statistic = ~"p"
  )
```

`ard_hierarchical` *Hierarchical ARD Statistics*

Description

[Experimental]
Performs hierarchical or nested tabulations, e.g. tabulates AE terms nested within AE system organ class.

- `ard_hierarchical()` includes summaries for the last variable listed in the `variables` argument, nested within the other variables included.
- `ard_hierarchical_count()` includes summaries for *all* variables listed in the `variables` argument each summary nested within the preceding variables, e.g. `variables=c(AESOC, AEDECOD)` summarizes `AEDECOD` nested in `AESOC`, and also summarizes the counts of `AESOC`.

Usage

```
ard_hierarchical(data, ...)

ard_hierarchical_count(data, ...)

## S3 method for class 'data.frame'
ard_hierarchical(
  data,
  variables,
  by = dplyr::group_vars(data),
  statistic = everything() ~ c("n", "N", "p"),
  denominator = NULL,
  fmt_fn = NULL,
  stat_label = everything() ~ default_stat_labels(),
  id = NULL,
  ...
)

## S3 method for class 'data.frame'
ard_hierarchical_count(
  data,
  variables,
  by = dplyr::group_vars(data),
  fmt_fn = NULL,
  stat_label = everything() ~ default_stat_labels(),
  ...
)
```

Arguments

`data`	(`data.frame`) a data frame

`...`	Arguments passed to methods.
`variables`	(tidy-select) variables to perform the nested/hierarchical tabulations within.
`by`	(tidy-select) variables to perform tabulations by. All combinations of the variables specified here appear in results. Default is `dplyr::group_vars(data)`.
`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")` (or the RHS of a formula).
`denominator`	(`data.frame`, `integer`) Specify this *optional* argument to change the denominator, e.g. the `"N"` statistic. Default is `NULL`. See below for details.
`fmt_fn`	(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(n = "n", p = "pct")` or `everything() ~ list(n ~ "n", p ~ "pct")`.
`id`	(tidy-select) an optional argument used to assert there are no duplicates within the `c(id, variables)` columns.

Value

an ARD data frame of class 'card'

Denominators

By default, the `ard_categorical()` function returns the statistics `"n"`, `"N"`, and `"p"`, where little `"n"` are the counts for the variable levels, and big `"N"` is the number of non-missing observations. The default calculation for the percentage is merely `p = n/N`.

However, it is sometimes necessary to provide a different `"N"` to use as the denominator in this calculation. For example, in a calculation of the rates of various observed adverse events, you may need to update the denominator to the number of enrolled subjects.

In such cases, use the `denominator` argument to specify a new definition of `"N"`, and subsequently `"p"`. The argument expects one of the following inputs:

- a data frame. Any columns in the data frame that overlap with the `by`/`strata` columns will be used to calculate the new `"N"`.
- an integer. This single integer will be used as the new `"N"`
- a string: one of `"column"`, `"row"`, or `"cell"`. `"column"` is equivalent to `denominator=NULL`. `"row"` gives 'row' percentages where `by`/`strata` columns are the 'top' of a cross table, and the variables are the rows. `"cell"` gives percentages where the denominator is the number of non-missing rows in the source data frame.

- a structured data frame. The data frame will include columns from by/strata. The last column must be named "...ard_N...". The integers in this column will be used as the updated "N" in the calculations.

Examples

```
ard_hierarchical(
  data = ADAE |>
    dplyr::slice_tail(n = 1L, by = c(USUBJID, TRTA, AESOC, AEDECOD)),
  variables = c(AESOC, AEDECOD),
  by = TRTA,
  id = USUBJID,
  denominator = ADSL |> dplyr::rename(TRTA = ARM)
)

ard_hierarchical_count(
  data = ADAE,
  variables = c(AESOC, AEDECOD),
  by = TRTA
)
```

---

ard_missing *Missing ARD Statistics*

Description

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

Usage

```
ard_missing(data, ...)

## S3 method for class 'data.frame'
ard_missing(
  data,
  variables,
  by = dplyr::group_vars(data),
 statistic = everything() ~ c("N_obs", "N_miss", "N_nonmiss", "p_miss", "p_nonmiss"),
  fmt_fn = NULL,
  stat_label = everything() ~ default_stat_labels(),
  ...
)
```

Arguments

data	(data.frame) a data frame
...	Arguments passed to methods.

variables (tidy-select)
columns to include in summaries. Default is `everything()`.

by (tidy-select)
results are tabulated by **all combinations** of the columns specified.

statistic (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) mean(x)))`.
The value assigned to each variable must also be a named list, where the names are used to reference a function and the element is the function object. Typically, this function will return a scalar statistic, but a function that returns a named list of results is also acceptable, e.g. `list(conf.low = -1, conf.high = 1)`. However, when errors occur, the messaging will be less clear in this setting.

fmt_fn (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")`.

Value

an ARD data frame of class 'card'

Examples

```
ard_missing(ADSL, by = "ARM", variables = "AGE")

ADSL |>
  dplyr::group_by(ARM) |>
  ard_missing(
    variables = "AGE",
    statistic = ~"N_miss"
  )
```

---

ard_pairwise *Pairwise ARD*

Description

Utility to perform pairwise comparisons.

Usage

```
ard_pairwise(data, variable, .f, include = NULL)
```

Arguments

data	(data.frame) a data frame
variable	(tidy-select) Column to perform pairwise analyses for.
.f	(function) a function that creates ARDs. The function accepts a single argument and a subset of data will be passed including the two levels of variable for the pairwise analysis.
include	(vector) a vector of levels of the variable column to include in comparisons. Pairwise comparisons will only be performed for pairs that have a level specified here. Default is NULL and all pairwise computations are included.

Value

list of ARDs

Examples

```
ard_pairwise(
  ADSL,
  variable = ARM,
  .f = \(df) {
    ard_complex(
      df,
      variables = AGE,
    statistic = ~ list(ttest = \(x, data, ...) t.test(x ~ data$ARM)[c("statistic", "p.value")])
    )
  },
  include = "Placebo" # only include comparisons to the "Placebo" group
)
```

---

ard_stack *Stack ARDs*

Description

Stack multiple ARD calls sharing common input data and by variables. Optionally incorporate additional information on represented variables, e.g. overall calculations, rates of missingness, attributes, or transform results with shuffle_ard().

If the ard_stack(by) argument is specified, a univariate tabulation of the by variable will also be returned.

Usage

```
ard_stack(
  data,
  ...,
  .by = NULL,
  .overall = FALSE,
  .missing = FALSE,
  .attributes = FALSE,
  .total_n = FALSE,
  .shuffle = FALSE
)
```

Arguments

data	(data.frame) a data frame
...	(dynamic-dots) Series of ARD function calls to be run and stacked
.by	(tidy-select) columns to tabulate by in the series of ARD function calls. Any rows with NA or NaN values are removed from all calculations.
.overall	(logical) logical indicating whether overall statistics should be calculated (i.e. re-run all ard_*() calls with by=NULL). Default is FALSE.
.missing	(logical) logical indicating whether to include the results of ard_missing() for all variables represented in the ARD. Default is FALSE.
.attributes	(logical) logical indicating whether to include the results of ard_attributes() for all variables represented in the ARD. Default is FALSE.
.total_n	(logical) logical indicating whether to include of ard_total_n() in the returned ARD.
.shuffle	(logical) logical indicating whether to perform shuffle_ard() on the final result. Default is FALSE.

Value

an ARD data frame of class 'card'

Examples

```
ard_stack(
  data = ADSL,
  ard_categorical(variables = "AGEGR1"),
  ard_continuous(variables = "AGE"),
  .by = "ARM",
```

```
  .overall = TRUE,
  .attributes = TRUE
)

ard_stack(
  data = ADSL,
  ard_categorical(variables = "AGEGR1"),
  ard_continuous(variables = "AGE"),
  .by = "ARM",
  .shuffle = TRUE
)
```

`ard_stack_hierarchical` *Stacked Hierarchical ARD Statistics*

Description

[Experimental]
Use these functions to calculate multiple summaries of nested or hierarchical data in a single call.

- `ard_stack_hierarchical()`: Calculates *rates* of events (e.g. adverse events) utilizing the `denominator` and `id` arguments to identify the rows in `data` to include in each rate calculation.
- `ard_stack_hierarchical_count()`: Calculates *counts* of events utilizing all rows for each tabulation.

Usage

```
ard_stack_hierarchical(
  data,
  variables,
  by = dplyr::group_vars(data),
  id,
  denominator,
  include = everything(),
  statistic = everything() ~ c("n", "N", "p"),
  overall = FALSE,
  over_variables = FALSE,
  attributes = FALSE,
  total_n = FALSE,
  shuffle = FALSE
)

ard_stack_hierarchical_count(
  data,
  variables,
  by = dplyr::group_vars(data),
```

```
  denominator = NULL,
  include = everything(),
  overall = FALSE,
  over_variables = FALSE,
  attributes = FALSE,
  total_n = FALSE,
  shuffle = FALSE
)
```

Arguments

`data`	(`data.frame`) a data frame
`variables`	(tidy-select) Specifies the nested/hierarchical structure of the data. The variables that are specified here and in the `include` argument will have summary statistics calculated.
`by`	(tidy-select) variables to perform tabulations by. All combinations of the variables specified here appear in results. Default is `dplyr::group_vars(data)`.
`id`	(tidy-select) argument used to subset `data` to identify rows in `data` to calculate event rates in `ard_stack_hierarchical()`. See details below.
`denominator`	(`data.frame`, `integer`) used to define the denominator and enhance the output. The argument is required for `ard_stack_hierarchical()` and optional for `ard_stack_hierarchical_count()`. • the univariate tabulations of the by variables are calculated with `denominator`, when a data frame is passed, e.g. tabulation of the treatment assignment counts that may appear in the header of a table. • the `denominator` argument must be specified when `id` is used to calculate the event rates. • if `total_n=TRUE`, the `denominator` argument is used to return the total N
`include`	(tidy-select) Specify the subset a columns indicated in the `variables` argument for which summary statistics will be returned. Default is `everything()`.
`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")` (or the RHS of a formula).
`overall`	(scalar `logical`) logical indicating whether overall statistics should be calculated (i.e. repeat the operations with `by=NULL` in *most cases*, see below for details). Default is `FALSE`.
`over_variables`	(scalar `logical`) logical indicating whether summary statistics should be calculated over or across the columns listed in the `variables` argument. Default is `FALSE`.

attributes (scalar logical)
logical indicating whether to include the results of ard_attributes() for all variables represented in the ARD. Default is FALSE.

total_n (scalar logical)
logical indicating whether to include of ard_total_n(denominator) in the returned ARD.

shuffle (scalar logical)
logical indicating whether to perform shuffle_ard() on the final result. Default is FALSE.

Value

an ARD data frame of class 'card'

Subsetting Data for Rate Calculations

To calculate event rates, the ard_stack_hierarchical() function identifies rows to include in the calculation. First, the primary data frame is sorted by the columns identified in the id, by, and variables arguments.

As the function cycles over the variables specified in the variables argument, the data frame is grouped by id, intersect(by, names(denominator)), and variables utilizing the last row within each of the groups.

For example, if the call is ard_stack_hierarchical(data = ADAE, variables = c(AESOC, AEDECOD), id = USUBJID), then we'd first subset ADAE to be one row within the grouping c(USUBJID, AESOC, AEDECOD) to calculate the event rates in 'AEDECOD'. We'd then repeat and subset ADAE to be one row within the grouping c(USUBJID, AESOC) to calculate the event rates in 'AESOC'.

Overall Argument

When we set overall=TRUE, we wish to re-run our calculations removing the stratifying columns. For example, if we ran the code below, we results would include results with the code chunk being re-run with by=NULL.

```
ard_stack_hierarchical(
  data = ADAE,
  variables = c(AESOC, AEDECOD),
  by = TRTA,
  denominator = ADSL |> dplyr::rename(TRTA = ARM),
  overall = TRUE
)
```

But there is another case to be aware of: when the by argument includes columns that are not present in the denominator, for example when tabulating results by AE grade or severity in addition to treatment assignment. In the example below, we're tabulating results by treatment assignment and AE severity. By specifying overall=TRUE, we will re-run the to get results with by = AESEV and again with by = NULL.

```
ard_stack_hierarchical(
  data = ADAE,
  variables = c(AESOC, AEDECOD),
  by = c(TRTA, AESEV),
  denominator = ADSL |> dplyr::rename(TRTA = ARM),
  overall = TRUE
)
```

Examples

```
ard_stack_hierarchical(
  ADAE,
  variables = c(AESOC, AEDECOD),
  by = TRTA,
  denominator = ADSL |> dplyr::rename(TRTA = ARM),
  id = USUBJID
)

ard_stack_hierarchical_count(
  ADAE,
  variables = c(AESOC, AEDECOD),
  by = TRTA,
  denominator = ADSL |> dplyr::rename(TRTA = ARM)
)
```

---

ard_strata *Stratified ARD*

Description

[Experimental]
General function for calculating ARD results within subgroups.

While the examples below show use with other functions from the cards package, this function would primarily be used with the statistical functions in the cardx functions.

Usage

```
ard_strata(.data, .by = NULL, .strata = NULL, .f, ...)
```

Arguments

`.data`	(`data.frame`) a data frame
`.by`, `.strata`	(`tidy-select`) columns to tabulate by/stratify by for calculation. 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. These argument *should not* include any columns that appear in the `.f` argument.
`.f`	(`function`, `formula`) a function or a formula that can be coerced to a function with `rlang::as_function()` (similar to `purrr::map(.f)`)
`...`	Additional arguments passed on to the `.f` function.

Value

an ARD data frame of class 'card'

Examples

```
ard_strata(
  ADSL,
  .by = ARM,
  .f = ~ ard_continuous(.x, variables = AGE)
)
```

`ard_total_n` *ARD Total N*

Description

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

Usage

```
ard_total_n(data, ...)

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

Arguments

`data`	(`data.frame`) a data frame
`...`	Arguments passed to methods.

Value

an ARD data frame of class 'card'

Examples

```
ard_total_n(ADSL)
```

as_card *Data Frame as ARD*

Description

Convert data frames to ARDs of class 'card'.

Usage

```
as_card(x)
```

Arguments

x	(data.frame) a data frame

Value

an ARD data frame of class 'card'

Examples

```
data.frame(
  stat_name = c("N", "mean"),
  stat_label = c("N", "Mean"),
  stat = c(10, 0.5)
) |>
  as_card()
```

as_cards_fn *As card function*

Description

Add attributes to a function that specify the expected results. It is used when `ard_continuous()` or `ard_complex()` errors and constructs an ARD with the correct structure when the results cannot be calculated.

Usage

```
as_cards_fn(f, stat_names)

is_cards_fn(f)

get_cards_fn_stat_names(f)
```

Arguments

f	(function) a function
stat_names	(character) a character vector of the expected statistic names returned by function f

Value

an ARD data frame of class 'card'

Examples

```
# When there is no error, everything works as if we hadn't used `as_card_fn()`
ttest_works <-
  as_cards_fn(
    \(x) t.test(x)[c("statistic", "p.value")],
    stat_names = c("statistic", "p.value")
  )
ard_continuous(
  mtcars,
  variables = mpg,
  statistic = ~ list(ttest = ttest_works)
)

# When there is an error and we use `as_card_fn()`,
#   we will see the same structure as when there is no error
ttest_error <-
  as_cards_fn(
    \(x) {
      t.test(x)[c("statistic", "p.value")]
      stop("Intentional Error")
    },
    stat_names = c("statistic", "p.value")
  )
ard_continuous(
  mtcars,
  variables = mpg,
  statistic = ~ list(ttest = ttest_error)
)

# if we don't use `as_card_fn()` and there is an error,
#   the returned result is only one row
ard_continuous(
  mtcars,
  variables = mpg,
  statistic = ~ list(ttest = \(x) {
    t.test(x)[c("statistic", "p.value")]
    stop("Intentional Error")
  })
)
```

as_nested_list *ARD as Nested List*

Description

[Experimental]
Convert ARDs to nested lists.

Usage

```
as_nested_list(x)
```

Arguments

x	(data.frame) an ARD data frame of class 'card'

Value

a nested list

Examples

```
ard_continuous(mtcars, by = "cyl", variables = c("mpg", "hp")) |>
  as_nested_list()
```

bind_ard *Bind ARDs*

Description

Wrapper for dplyr::bind_rows() with additional checks for duplicated statistics.

Usage

```
bind_ard(
  ...,
  .distinct = TRUE,
  .update = FALSE,
  .order = FALSE,
  .quiet = FALSE
)
```

Arguments

`...`	(dynamic-dots) ARDs to combine. Each argument can either be an ARD, or a list of ARDs. Columns are matched by name, and any missing columns will be filled with NA.
`.distinct`	(logical) logical indicating whether to remove non-distinct values from the ARD. Duplicates are checked across grouping variables, primary variables, context (if present), the **statistic name and the statistic value**. Default is FALSE. If a statistic name and value is repeated and `.distinct=TRUE`, the more recently added statistics will be retained, and the other(s) omitted.
`.update`	(logical) logical indicating whether to update ARD and remove duplicated named statistics. Duplicates are checked across grouping variables, primary variables, and the **statistic name**. Default is FALSE. If a statistic name is repeated and `.update=TRUE`, the more recently added statistics will be retained, and the other(s) omitted.
`.order`	(logical) logical indicating whether to order the rows of the stacked ARDs, allowing statistics that share common group and variable values to appear in consecutive rows. Default is FALSE. Ordering will be based on the order of the group/variable values prior to stacking.
`.quiet`	(logical) logical indicating whether to suppress any messaging. Default is FALSE

Value

an ARD data frame of class 'card'

Examples

```
ard <- ard_categorical(ADSL, by = "ARM", variables = "AGEGR1")

bind_ard(ard, ard, .update = TRUE)
```

---

check_ard_structure *Check ARD Structure*

Description

Function tests the structure and returns notes when object does not conform to expected structure.

Usage

```
check_ard_structure(x, column_order = TRUE, method = TRUE)
```

Arguments

x	(data.frame) an ARD data frame of class 'card'
column_order	(scalar logical) check whether ordering of columns adheres to to cards::tidy_ard_column_order().
method	(scalar logical) check whether a "stat_name" equal to "method" appears in results.

Value

an ARD data frame of class 'card' (invisible)

Examples

```
ard_continuous(ADSL, variables = "AGE") |>
  dplyr::select(-warning, -error) |>
  check_ard_structure()
```

---

default_stat_labels *Defaults for Statistical Arguments*

Description

Returns a named list of statistics labels

Usage

```
default_stat_labels()
```

Value

named list

Examples

```
# stat labels
default_stat_labels()
```

`eval_capture_conditions` *Evaluate and Capture Conditions*

Description

[Experimental]

`eval_capture_conditions()`

Evaluates an expression while also capturing error and warning conditions. Function always returns a named list `list(result=, warning=, error=)`. If there are no errors or warnings, those elements will be `NULL`. If there is an error, the result element will be `NULL`.

Messages are neither saved nor printed to the console.

Evaluation is done via `rlang::eval_tidy()`. If errors and warnings are produced using the `{cli}` package, the messages are processed with `cli::ansi_strip()` to remove styling from the message.

`captured_condition_as_message()`/`captured_condition_as_error()`

These functions take the result from `eval_capture_conditions()` and return errors or warnings as either messages (via `cli::cli_inform()`) or errors (via `cli::cli_abort()`). These functions handle cases where the condition messages may include curly brackets, which would typically cause issues when processed with the `cli::cli_*()` functions.

Functions return the `"result"` from `eval_capture_conditions()`.

Usage

```
eval_capture_conditions(expr, data = NULL, env = caller_env())

captured_condition_as_message(
  x,
  message = c("The following {type} occured:", x = "{condition}"),
  type = c("error", "warning"),
  envir = rlang::current_env()
)

captured_condition_as_error(
  x,
  message = c("The following {type} occured:", x = "{condition}"),
  type = c("error", "warning"),
  call = get_cli_abort_call(),
  envir = rlang::current_env()
)
```

Arguments

`expr`	An expression or quosure to evaluate.

data	A data frame, or named list or vector. Alternatively, a data mask created with as_data_mask() or new_data_mask(). Objects in data have priority over those in env. See the section about data masking.
env	The environment in which to evaluate expr. This environment is not applicable for quosures because they have their own environments.
x	(captured_condition) a captured condition created by eval_capture_conditions().
message	(character) message passed to cli::cli_inform() or cli::cli_abort(). The condition being printed is saved in an object named condition, which should be included in this message surrounded by curly brackets.
type	(string) the type of condition to return. Must be one of 'error' or 'warning'.
envir	Environment to evaluate the glue expressions in.
call	(environment) Execution environment of currently running function. Default is get_cli_abort_call().

Value

a named list

Examples

```
# function executes without error or warning
eval_capture_conditions(letters[1:2])

# an error is thrown
res <- eval_capture_conditions(stop("Example Error!"))
res
captured_condition_as_message(res)

# if more than one warning is returned, all are saved
eval_capture_conditions({
  warning("Warning 1")
  warning("Warning 2")
  letters[1:2]
})

# messages are not printed to the console
eval_capture_conditions({
  message("A message!")
  letters[1:2]
})
```

`get_ard_statistics` *ARD Statistics as List*

Description

Returns the statistics from an ARD as a named list.

Usage

```
get_ard_statistics(x, ..., .column = "stat", .attributes = NULL)
```

Arguments

`x`	(`data.frame`) an ARD data frame of class 'card'
`...`	(dynamic-dots) optional arguments indicating rows to subset of the ARD. For example, to return only rows where the column `"AGEGR1"` is `"65-80"`, pass `AGEGR1 %in% "65-80"`.
`.column`	(`string`) string indicating the column that will be returned in the list. Default is `"statistic"`
`.attributes`	(`character`) character vector of column names that will be returned in the list as attributes. Default is `NULL`

Value

named list

Examples

```
ard <- ard_categorical(ADSL, by = "ARM", variables = "AGEGR1")

get_ard_statistics(
  ard,
  group1_level %in% "Placebo",
  variable_level %in% "65-80",
  .attributes = "stat_label"
)
```

label_cards *Generate Formatting Function*

Description

Returns a function with the requested rounding and scaling schema.

Usage

```
label_cards(digits = 1, scale = 1, width = NULL)
```

Arguments

digits	(integer) a non-negative integer specifying the number of decimal places round statistics to
scale	(numeric) a scalar real number. Before rounding, the input will be scaled by this quantity
width	(integer) a non-negative integer specifying the minimum width of the returned formatted values

Value

a function

Examples

```
label_cards(2)(pi)
label_cards(1, scale = 100)(pi)
label_cards(2, width = 5)(pi)
```

maximum_variable_value *Maximum Value*

Description

For each column in the passed data frame, the function returns a named list with the value being the largest/last element after a sort. For factors, the last level is returned, and for logical vectors TRUE is returned. This is used as the default value in ard_dichotomous(value) if not specified by the user.

Usage

```
maximum_variable_value(data)
```

Arguments

data	(data.frame) a data frame

Value

a named list

Examples

```
ADSL[c("AGEGR1", "BMIBLGR1")] |> maximum_variable_value()
```

---

mock *Mock ARDs*

Description

[Experimental]
Create empty ARDs used to create mock tables or table shells. Where applicable, the formatting functions are set to return 'xx' or 'xx.x'.

Usage

```
mock_categorical(
  variables,
  statistic = everything() ~ c("n", "p", "N"),
  by = NULL
)

mock_continuous(
  variables,
 statistic = everything() ~ c("N", "mean", "sd", "median", "p25", "p75", "min", "max"),
  by = NULL
)

mock_dichotomous(
  variables,
  statistic = everything() ~ c("n", "p", "N"),
  by = NULL
)

mock_missing(
  variables,
 statistic = everything() ~ c("N_obs", "N_miss", "N_nonmiss", "p_miss", "p_nonmiss"),
  by = NULL
)
```

```
mock_attributes(label)

mock_total_n()
```

Arguments

variables	(character or named list) a character vector of variable names for functions mock_continuous(), mock_missing(), and mock_attributes(). a named list for functions mock_categorical() and mock_dichotomous(), where the list element is a vector of variable values. For mock_dichotomous(), only a single value is allowed for each variable.
statistic	(formula-list-selector) a named list, a list of formulas, or a single formula where the list elements are character vectors of statistic names to appear in the ARD.
by	(named list) a named list where the list element is a vector of variable values.
label	(named list) named list of variable labels, e.g. list(cyl = "No. Cylinders").

Value

an ARD data frame of class 'card'

Examples

```
mock_categorical(
  variables =
    list(
      AGEGR1 = factor(c("<65", "65-80", ">80"), levels = c("<65", "65-80", ">80"))
    ),
  by = list(TRTA = c("Placebo", "Xanomeline High Dose", "Xanomeline Low Dose"))
) |>
  apply_fmt_fn()

mock_continuous(
  variables = c("AGE", "BMIBL"),
  by = list(TRTA = c("Placebo", "Xanomeline High Dose", "Xanomeline Low Dose"))
) |>
  # update the mock to report 'xx.xx' for standard deviations
 update_ard_fmt_fn(variables = c("AGE", "BMIBL"), stat_names = "sd", fmt_fn = \(x) "xx.xx") |>
  apply_fmt_fn()
```

`nest_for_ard` *ARD Nesting*

Description

This function is similar to `tidyr::nest()`, except that it retains rows for unobserved combinations (and unobserved factor levels) of by variables, and unobserved combinations of stratifying variables.

The levels are wrapped in lists so they can be stacked with other types of different classes.

Usage

```
nest_for_ard(
  data,
  by = NULL,
  strata = NULL,
  key = "data",
  rename_columns = TRUE,
  list_columns = TRUE,
  include_data = TRUE
)
```

Arguments

`data`	(`data.frame`) a data frame
`by`, `strata`	(`character`) columns to nest by/stratify by. Arguments are similar, but with an important distinction: `by`: data frame is nested by **all combinations** of the columns specified, including unobserved combinations and unobserved factor levels. `strata`: data frame is nested by **all *observed* combinations** of the columns specified. Arguments may be used in conjunction with one another.
`key`	(`string`) the name of the new column with the nested data frame. Default is `"data"`.
`rename_columns`	(`logical`) logical indicating whether to rename the `by` and `strata` variables. Default is `TRUE`.
`list_columns`	(`logical`) logical indicating whether to put levels of `by` and `strata` columns in a list. Default is `TRUE`.
`include_data`	(scalar `logical`) logical indicating whether to include the data subsets as a list-column. Default is `TRUE`.

Value

a nested tibble

Examples

```
nest_for_ard(
  data =
    ADAE |>
      dplyr::left_join(ADSL[c("USUBJID", "ARM")], by = "USUBJID") |>
      dplyr::filter(AOCCSFL %in% "Y"),
  by = "ARM",
  strata = "AESOC"
)
```

`print.card` *Print*

Description

[Experimental]
Print method for objects of class 'card'

Usage

```
## S3 method for class 'card'
print(x, n = NULL, columns = c("auto", "all"), n_col = 6L, ...)
```

Arguments

`x`	(`data.frame`) object of class 'card'
`n`	(`integer`) integer specifying the number of rows to print
`columns`	(`string`) string indicating whether to print a selected number of columns or all.
`n_col`	(`integer`) some columns are removed when there are more than a threshold of columns present. This argument sets that threshold. This is only used when `columns='auto'` and default is `6L`. Columns `'error'`, `'warning'`, `'context'`, and `'fmt_fn'` *may* be removed from the print. All other columns will be printed, even if more than `n_col` columns are present.
`...`	(dynamic-dots) not used

Value

an ARD data frame of class 'card' (invisibly)

Examples

```
ard_categorical(ADSL, variables = AGEGR1) |>
  print()
```

---

print_ard_conditions *Print ARD Condition Messages*

Description

Function parses the errors and warnings observed while calculating the statistics requested in the ARD and prints them to the console as messages.

Usage

```
print_ard_conditions(x)
```

Arguments

x	(data.frame) an ARD data frame of class 'card'

Value

returns invisible if check is successful, throws all condition messages if not.

Examples

```
# passing a character variable for numeric summary
ard_continuous(ADSL, variables = AGEGR1) |>
  print_ard_conditions()
```

---

process_selectors *Process tidyselectors*

Description

Functions process tidyselect arguments passed to functions in the cards package. The processed values are saved to the calling environment, by default.

- `process_selectors()`: the arguments will be processed with tidyselect and converted to a vector of character column names.
- `process_formula_selectors()`: for arguments that expect named lists or lists of formulas (where the LHS of the formula is a tidyselector). This function processes these inputs and returns a named list. If a name is repeated, the last entry is kept.

- `fill_formula_selectors()`: when users override the default argument values, it can be important to ensure that each column from a data frame is assigned a value. This function checks that each column in `data` has an assigned value, and if not, fills the value in with the default value passed here.
- `compute_formula_selector()`: used in `process_formula_selectors()` to evaluate a single argument.
- `check_list_elements()`: used to check the class/type/values of the list elements, primarily those processed with `process_formula_selectors()`.
- `cards_select()`: wraps `tidyselect::eval_select() |> names()`, and returns better contextual messaging when errors occur.

Usage

```
process_selectors(data, ...)

process_formula_selectors(data, ...)

fill_formula_selectors(data, ...)

## S3 method for class 'data.frame'
process_selectors(data, ..., env = caller_env())

## S3 method for class 'data.frame'
process_formula_selectors(
  data,
  ...,
  env = caller_env(),
  include_env = FALSE,
  allow_empty = TRUE
)

## S3 method for class 'data.frame'
fill_formula_selectors(data, ..., env = caller_env())

compute_formula_selector(
  data,
  x,
  arg_name = caller_arg(x),
  env = caller_env(),
  strict = TRUE,
  include_env = FALSE,
  allow_empty = TRUE
)

check_list_elements(
  x,
  predicate,
  error_msg = NULL,
```

```
  arg_name = rlang::caller_arg(x)
)

cards_select(expr, data, ..., arg_name = NULL)
```

Arguments

data	(data.frame) a data frame
...	(dynamic-dots) named arguments where the value of the argument is processed with tidyselect. • process_selectors(): the values are tidyselect-compatible selectors • process_formula_selectors(): the values are named lists, list of formulas a combination of both, or a single formula. Users may pass ~value as a shortcut for everything() ~ value. • check_list_elements(): named arguments where the name matches an existing list in the env environment, and the value is a predicate function to test each element of the list, e.g. each element must be a string or a function.
env	(environment) env to save the results to. Default is the calling environment.
include_env	(logical) whether to include the environment from the formula object in the returned named list. Default is FALSE
allow_empty	(logical) Logical indicating whether empty result is acceptable while process formula-list selectors. Default is TRUE.
x	• compute_formula_selector(): (formula-list-selector) a named list, list of formulas, or a single formula that will be converted to a named list. • check_list_elements(): (named list) a named list
arg_name	(string) the name of the argument being processed. Used in error messaging. Default is caller_arg(x).
strict	(logical) whether to throw an error if a variable doesn't exist in the reference data (passed to tidyselect::eval_select())
predicate	(function) a predicate function that returns TRUE or FALSE
error_msg	(character) a character vector that will be used in error messaging when mis-specified arguments are passed. Elements "{arg_name}" and "{variable}" are available using glue syntax for messaging.
expr	(expression) Defused R code describing a selection according to the tidyselect syntax.

Value

process_selectors(), fill_formula_selectors(), process_formula_selectors() and check_list_elements() return NULL. compute_formula_selector() returns a named list.

Examples

```
example_env <- rlang::new_environment()

process_selectors(ADSL, variables = starts_with("TRT"), env = example_env)
get(x = "variables", envir = example_env)

fill_formula_selectors(ADSL, env = example_env)

process_formula_selectors(
  ADSL,
  statistic = list(starts_with("TRT") ~ mean, TRTSDT = min),
  env = example_env
)
get(x = "statistic", envir = example_env)

check_list_elements(
  get(x = "statistic", envir = example_env),
  predicate = function(x) !is.null(x),
  error_msg = c(
    "Error in the argument {.arg {arg_name}} for variable {.val {variable}}.",
    "i" = "Value must be a named list of functions."
  )
)

# process one list
compute_formula_selector(ADSL, x = starts_with("U") ~ 1L)
```

---

rename_ard_columns *Rename ARD Columns*

Description

This function combines a pair of group/group_level or variable/variable_level columns into a single column. The group_level or variable_level column is renamed according to the value of the group or variable column, respectively.

Usage

```
rename_ard_columns(
  x,
  columns = c(all_ard_groups(), all_ard_variables()),
  unlist = NULL
)
```

Arguments

x	(data.frame) a data frame
columns	(tidy-select) Name of columns to coalesce together and rename.
unlist	(tidy-select) Columns to unlist. Often useful when performing visual inspection of the results where the list-columns are more difficult to work with.

Value

data frame

Examples

```
ADSL |>
  ard_categorical(by = ARM, variables = AGEGR1) |>
  apply_fmt_fn() |>
  rename_ard_columns(unlist = c(stat, stat_fmt))
```

---

replace_null_statistic

Replace NULL Statistics with Specified Value

Description

When a statistical summary function errors, the "stat" column will be NULL. It is, however, sometimes useful to replace these values with a non-NULL value, e.g. NA.

Usage

```
replace_null_statistic(x, value = NA, rows = TRUE)
```

Arguments

x	(data.frame) an ARD data frame of class 'card'
value	(usually a scalar) The value to replace NULL values with. Default is NA.
rows	(data-masking) Expression that return a logical value, and are defined in terms of the variables in .data. Only rows for which the condition evaluates to TRUE are replaced. Default is TRUE, which applies to all rows.

Value

an ARD data frame of class 'card'

Examples

```
# the quantile functions error because the input is character, while the median function returns NA
data.frame(x = rep_len(NA_character_, 10)) |>
  ard_continuous(
    variables = x,
    statistic = ~ continuous_summary_fns(c("median", "p25", "p75"))
  ) |>
  replace_null_statistic(rows = !is.null(error))
```

---

round5 *Rounding of Numbers*

Description

Rounds the values in its first argument to the specified number of decimal places (default 0). Importantly, `round5()` **does not** use Base R's "round to even" default. Standard rounding methods are implemented, for example, `round5(0.5) = 1`.

Usage

```
round5(x, digits = 0)
```

Arguments

`x`	(`numeric`) a numeric vector
`digits`	(`integer`) integer indicating the number of decimal places

Details

Function inspired by `janitor::round_half_up()`.

Value

a numeric vector

Examples

```
x <- 0:4 / 2
round5(x) |> setNames(x)

# compare results to Base R
round(x) |> setNames(x)
```

`selectors` *ARD Selectors*

Description

These selection helpers match variables according to a given pattern.

- `all_ard_groups()`: Function selects grouping columns, e.g. columns named `"group##"` or `"group##_level"`.
- `all_ard_variables()`: Function selects variables columns, e.g. columns named `"variable"` or `"variable_level"`.
- `all_ard_group_n()`: Function selects `n` grouping columns.
- `all_missing_columns()`: Function selects columns that are all `NA` or empty.

Usage

```
all_ard_groups(types = c("names", "levels"))

all_ard_variables(types = c("names", "levels"))

all_ard_group_n(n)

all_missing_columns()
```

Arguments

`types`	(`character`) type(s) of columns to select. `"names"` selects the columns variable name columns, and `"levels"` selects the level columns. Default is `c("names", "levels")`.
`n`	(`integer`) integer(s) indicating which grouping columns to select.

Value

tidyselect output

Examples

```
ard <- ard_categorical(ADSL, by = "ARM", variables = "AGEGR1")

ard |> dplyr::select(all_ard_groups())
ard |> dplyr::select(all_ard_variables())
```

shuffle_ard *Shuffle ARD*

Description

This function ingests an ARD object and shuffles the information to prepare for analysis. Helpful for streamlining across multiple ARDs. Combines each group/group_level into 1 column, back fills missing grouping values from the variable levels where possible, and optionally trims statistics-level metadata.

Usage

```
shuffle_ard(x, trim = TRUE)
```

Arguments

x	(data.frame) an ARD data frame of class 'card'
trim	(logical) logical representing whether or not to trim away statistic-level metadata and filter only on numeric statistic values.

Value

a tibble

Examples

```
bind_ard(
  ard_categorical(ADSL, by = "ARM", variables = "AGEGR1"),
  ard_categorical(ADSL, variables = "ARM")
) |>
  shuffle_ard()
```

summary_functions *Summary Functions*

Description

- `continuous_summary_fns()` returns a named list of summary functions for continuous variables. Some functions include slight modifications to their base equivalents. For example, the `min()` and `max()` functions return `NA` instead of `Inf` when an empty vector is passed. Statistics `"p25"` and `"p75"` are calculated with `quantile(type = 2)`, which matches SAS's default value.

Usage

```
continuous_summary_fns(
  summaries = c("N", "mean", "sd", "median", "p25", "p75", "min", "max"),
  other_stats = NULL
)
```

Arguments

summaries	(character) a character vector of results to include in output. Select one or more from 'N', 'mean', 'sd', 'median', 'p25', 'p75', 'min', 'max'.
other_stats	(named list) named list of other statistic functions to supplement the pre-programmed functions.

Value

named list of summary statistics

Examples

```
# continuous variable summaries
ard_continuous(
  ADSL,
  variables = "AGE",
  statistic = ~ continuous_summary_fns(c("N", "median"))
)
```

---

tidy_ard_order *Standard Order of ARD*

Description

ARD functions for relocating columns and rows to the standard order.

- `tidy_ard_column_order()` relocates columns of the ARD to the standard order.
- `tidy_ard_row_order()` orders rows of ARD according to variables, groups, and strata, while retaining the order of the input ARD.

Usage

```
tidy_ard_column_order(x)

tidy_ard_row_order(x)
```

Arguments

x	(data.frame) an ARD data frame of class 'card'

Value

an ARD data frame of class 'card'

Examples

```
# order columns
ard <-
  dplyr::bind_rows(
    ard_continuous(mtcars, variables = "mpg"),
    ard_continuous(mtcars, variables = "mpg", by = "cyl")
  )

tidy_ard_column_order(ard) |>
  tidy_ard_row_order()
```

---

tidy_as_ard *Build ARD from Tidier*

Description

Function converts a model's one-row tidy data frame into an ARD structure. The tidied data frame must have been constructed with eval_capture_conditions().

This function is primarily for developers and few consistency checks have been included.

Usage

```
tidy_as_ard(
  lst_tidy,
  tidy_result_names,
  fun_args_to_record = character(0L),
  formals = list(),
  passed_args = list(),
  lst_ard_columns
)
```

Arguments

lst_tidy	(named list) list of tidied results constructed with eval_capture_conditions(), e.g. eval_capture_conditions(t ~ mtcars$am) |> broom::tidy()).

`tidy_result_names`
: (`character`)
character vector of column names expected by the tidier method. This is used to construct blank results in the event of an error.

`fun_args_to_record`
: (`character`)
character vector of function argument names that are added to the ARD.

`formals`
: (`pairlist`)
the results from `formals()`, e.g. `formals(fisher.test)`. This is used to get the default argument values from unspecified arguments.

`passed_args`
: (named `list`)
named list of additional arguments passed to the modeling function.

`lst_ard_columns`
: (named `list`)
named list of values that will be added to the ARD data frame.

Value

an ARD data frame of class 'card'

Examples

```
# example how one may create a fisher.test() ARD function
my_ard_fishertest <- function(data, by, variable, ...) {
  # perform fisher test and format results ----------------------------------
  lst_tidy_fisher <-
    eval_capture_conditions(
      # this manipulation is similar to `fisher.test(...) |> broom::tidy()`
    stats::fisher.test(x = data[[variable]], y = data[[by]], ...)[c("p.value", "method")] |>
        as.data.frame()
    )

  # build ARD ---------------------------------------------------------------
  tidy_as_ard(
    lst_tidy = lst_tidy_fisher,
    tidy_result_names = c("p.value", "method"),
    fun_args_to_record =
      c(
        "workspace", "hybrid", "hybridPars", "control", "or",
        "conf.int", "conf.level", "simulate.p.value", "B"
      ),
    formals = formals(stats::fisher.test),
    passed_args = dots_list(...),
    lst_ard_columns = list(group1 = by, variable = variable, context = "fishertest")
  )
}

my_ard_fishertest(mtcars, by = "am", variable = "vs")
```

update_ard *Update ARDs*

Description

Functions used to update ARD formatting functions and statistic labels.

This is a helper function to streamline the update process. If it does not exactly meet your needs, recall that an ARD is just a data frame and it can be modified directly.

Usage

```
update_ard_fmt_fn(
  x,
  variables = everything(),
  stat_names,
  fmt_fn,
  filter = TRUE
)

update_ard_stat_label(
  x,
  variables = everything(),
  stat_names,
  stat_label,
  filter = TRUE
)
```

Arguments

x	(data.frame) an ARD data frame of class 'card'
variables	(tidy-select) variables in x$variable to apply update. Default is everything().
stat_names	(character) character vector of the statistic names (i.e. values from x$stat_name) to apply the update.
fmt_fn	(function) a function or alias recognized by alias_as_fmt_fn().
filter	(expression) an expression that evaluates to a logical vector identifying rows in x to apply the update to. Default is TRUE, and update is applied to all rows.
stat_label	(function) a string of the updated statistic label.

Value

an ARD data frame of class 'card'

Examples

```
ard_continuous(ADSL, variables = AGE) |>
  update_ard_fmt_fn(stat_names = c("mean", "sd"), fmt_fn = 8L) |>
  update_ard_stat_label(stat_names = c("mean", "sd"), stat_label = "Mean (SD)") |>
  apply_fmt_fn()

# same as above, but only apply update to the Placebo level
ard_continuous(
  ADSL,
  by = ARM,
  variables = AGE,
  statistic = ~ continuous_summary_fns(c("N", "mean"))
) |>
 update_ard_fmt_fn(stat_names = "mean", fmt_fn = 8L, filter = group1_level == "Placebo") |>
  apply_fmt_fn()
```

Index