Package 'cards'

February 26, 2024

Title Analysis Results Data

Version 0.1.0

Description Construct Clinical Data Interchange Standards Consortium (CDISC) 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/website rmarkdown, jsonlite, yaml, gtsummary, insightsengineering/nesttemplate

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.3.1

NeedsCompilation no

Author Daniel D. Sjoberg [aut, cre] (<`https://orcid.org/0000-0003-0862-2018`>),
Becca Krouse [aut],
F. Hoffmann-La Roche AG [cph, fnd]

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

Repository CRAN

Date/Publication 2024-02-26 16:50:02 UTC

R topics documented:

`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.

`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 `x`s that appear after the decimal place indicate the number of decimal places the statistics will be rounded to. The number of `x`s 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, call = parent.frame())
```

Arguments

`x`	(`integer`, `string`, or `function`) a non-negative integer, string alias, or function
`call`	(`environment`) frame for error messaging. Default is `parent.frame()`.

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)
```

Arguments

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

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, variables = everything(), label = NULL)
```

Arguments

`data`	(`data.frame`) a data frame
`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,
  variables,
  by = dplyr::group_vars(data),
  strata = NULL,
  statistic = everything() ~ categorical_summary_fns(),
  denominator = NULL,
  fmt_fn = NULL,
  stat_label = everything() ~ default_stat_labels()
)
```

Arguments

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

`by, strata`	(`tidy-select`) columns to tabulate by/stratify by for tabulation. 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 is a named list of functions (or the RHS of a formula), e.g. `list(mpg = categorical_summary_fns())`.
`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_categorical(ADSL, by = "ARM", variables = "AGEGR1")

ADSL |>
  dplyr::group_by(ARM) |>
  ard_categorical(
    variables = "AGEGR1",
    statistic = everything() ~ categorical_summary_fns("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,
  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
`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.
- `data_full`: 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, data_full, variable, ...) {
  list(
    mean = mean(data[[variable]], na.rm = TRUE),
    grand_mean = mean(data_full[[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,
  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
`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,
  variables,
  by = dplyr::group_vars(data),
  strata = NULL,
  value = maximum_variable_value(data[variables]),
  statistic = everything() ~ categorical_summary_fns(),
  denominator = NULL,
  fmt_fn = NULL,
  stat_label = everything() ~ default_stat_labels()
)
```

Arguments

`data`	(`data.frame`) a data frame
`variables`	(`tidy-select`) columns to include in summaries. Default is `everything()`.
`by, strata`	(`tidy-select`) columns to tabulate by/stratify by for tabulation. 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 is a named list of functions (or the RHS of a formula), e.g. `list(mpg = categorical_summary_fns())`.
`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'

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 = ~ categorical_summary_fns("p")
  )
```

`ard_hierarchical` *Hierarchical ARD Statistics*

Description

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, AETERM)` summarizes `AETERM` nested in `AESOC`, and also summarizes the counts of `AESOC`.

Usage

```
ard_hierarchical(
  data,
  variables,
  by = dplyr::group_vars(data),
  statistic = everything() ~ categorical_summary_fns(),
  denominator = NULL,
  fmt_fn = NULL,
  stat_label = everything() ~ default_stat_labels()
)

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
`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 is a named list of functions (or the RHS of a formula), e.g. `list(mpg = categorical_summary_fns())`.

`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'

Examples

```
ard_hierarchical(
  data = ADAE,
  variables = c(AESOC, AETERM),
  by = c(TRTA, AESEV),
  denominator = ADSL |> dplyr::rename(TRTA = ARM)
)

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

`ard_missing` *Missing ARD Statistics*

Description

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

Usage

```
ard_missing(
  data,
  variables,
  by = dplyr::group_vars(data),
  statistic = everything() ~ missing_summary_fns(),
  fmt_fn = NULL,
  stat_label = everything() ~ default_stat_labels()
)
```

Arguments

`data`	(`data.frame`) a data frame
`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 = ~ missing_summary_fns("N_miss")
  )
```

`ard_stack` *Stack ARDs*

Description

Stack multiple ARD calls sharing common input `data` and `by` variables. Optionally incorporate additional information on represented variables (i.e. big N's, missingness, attributes) and/or tidy for use in displays with `shuffle_ard()`.

Usage

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

Arguments

`data`	(`data.frame`) a data frame
`by`	(`tidy-select`) columns to tabulate by in the series of ARD function calls
`...`	(`dynamic-dots`) Series of ARD function calls to be run and stacked
`.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`.
`.shuffle`	(`logical`) logical indicating whether to perform `shuffle_ard()` on the final result. Default is `FALSE`.

Value

a transformed ARD data frame (of class 'card' if `.shuffle = FALSE`)

Examples

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

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

`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(..., .update = FALSE, .order = 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`.
`.update`	(`logical`) logical indicating whether to update duplicate ARD statistics. Default is `FALSE`. If a statistic type is repeated and `.update=TRUE`, the more recently added statistics will be retained, and the others 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.

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)
```

Arguments

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

Value

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

Examples

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

check_pkg_installed *Check Package Installation*

Description

- `check_pkg_installed()`: checks whether a package is installed and returns an error or `FALSE` if not available. If a package search is provided, the function will check whether a minimum version of a package is required.
- `get_pkg_dependencies()` returns a tibble with all dependencies of a specific package.
- `get_min_version_required()` will return, if any, the minimum version of pkg required by `reference_pkg`, `NULL` if no minimum version required.

Usage

```
check_pkg_installed(pkg, reference_pkg = "cards", call = parent.frame())

is_pkg_installed(pkg, reference_pkg = "cards", call = parent.frame())

get_pkg_dependencies(reference_pkg = "cards", lib.loc = NULL)

get_min_version_required(pkg, reference_pkg = "cards", lib.loc = NULL)
```

Arguments

`pkg`	(`string`) name of required package
`reference_pkg`	(`string`) name of the package the function will search for a minimum required version from.
`call`	(`environment`) frame for error messaging. Default is `parent.frame()`.
`lib.loc`	(`path`) location of `R` library trees to search through, see `utils::packageDescription()`.

Value

`check_pkg_installed()` returns a logical or error, `get_min_version_required()` returns `NULL` or a string with the minimum version required, `get_pkg_dependencies()` returns a tibble.

Examples

```
check_pkg_installed("dplyr")

is_pkg_installed("dplyr")

get_pkg_dependencies()
```

```
get_min_version_required("dplyr")
```

`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

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 `eval_tidy()`.

Usage

```
eval_capture_conditions(expr, data = NULL, env = caller_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.

Value

a named list

Examples

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

# an error is thrown
eval_capture_conditions(stop("Example Error!"))

# 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()
```

`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
)
```

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`.

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

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. Default is `6L`.
`...`	(`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, call = NULL)
```

Arguments

`x`	(`data.frame`) an ARD data frame of class 'card'
`call`	(`environment`) frame for error messaging. Default is `NULL`.

Value

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

Examples

```
ard_continuous(
  ADSL,
  by = ARM,
  variables = AGE
) |>
  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, ..., env = caller_env())

process_formula_selectors(
  data,
  ...,
  env = caller_env(),
  include_env = FALSE,
  allow_empty = TRUE
)

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,
  env = rlang::caller_env(),
  arg_name = rlang::caller_arg(x)
)

cards_select(expr, data, ..., arg_name = NULL, .call = parent.frame())
```

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.
`.call`	(`environment`) calling environment used for error messaging.

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)
```

replace_null_statistic

Replace NULL Statistics with Specified Value

Description

When a statistical summary function errors, the `"statistic"` 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()`: Use this function in dplyr selecting environments, such as `dplyr::select()`. Function selects grouping columns, e.g. columns named `"group##"` or `"group##_level"`.
- `all_ard_variables()`: Use this function in dplyr selecting environments, such as `dplyr::select()`. Function selects variables columns, e.g. columns named `"variable"` or `"variable_level"`.

Usage

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

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

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")`.

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.
- `categorical_summary_fns()` returns a named list of summary statistics for categorical variables. Options are `"n"`, `"N"`, and `"p"`. If a user requests, for example, only `"p"`, the function will return `"n"` and `"N"` as well, since they are needed to calculate `"p"`. These statistics will be stored as a vector within the `tabulation` list element.
- `missing_summary_fns()` returns a named list of summary functions suitable for variable-level summaries, such as number and rate of missing data.

Usage

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

categorical_summary_fns(summaries = c("n", "p", "N"), other_stats = NULL)

missing_summary_fns(
  summaries = c("N_obs", "N_miss", "N_nonmiss", "p_miss", "p_nonmiss")
)
```

Arguments

`summaries`	(`character`) a character vector of results to include in output. • `continuous_summary_fns()`: Select one or more from 'N', 'mean', 'sd', 'median', 'p25', 'p75', 'min', 'max'. • `categorical_summary_fns()`: Select one or more from 'n', 'p', 'N'. • `missing_summary_fns()`: Select one or more from 'N_obs', 'N_miss', 'N_nonmiss', 'p_miss', 'p_nonmiss'.

`other_stats`	(named `list`) named list of other statistic functions to supplement the pre-programmed functions.

Value

`continuous_summary_fns()` and `missing_summary_fns()` return a named list of summary functions, `categorical_summary_fns()` returns a named list of summary statistics.

Examples

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

# categorical variable summaries
ard_categorical(
  ADSL,
  variables = "AGEGR1",
  statistic = ~ categorical_summary_fns(c("n", "N"))
)

# summary for rates of missing data
ard_missing(
  ADSL,
  variables = c("AGE", "AGEGR1"),
  statistic = ~ missing_summary_fns()
)
```

`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")
```

Index