Package ‘RastaRocket’

November 20, 2025

Title Rocket-Fast Clinical Research Reporting

Version 1.0.0

Description Description of the tables, both grouped and not grouped, with some associated data management actions,
such as sorting the terms of the variables and deleting terms with zero numbers.

License GPL (>= 3)

URL https://github.com/biostatusmr/RastaRocket,
https://biostatusmr.github.io/RastaRocket/

BugReports https://github.com/biostatusmr/RastaRocket/issues

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports dplyr, tidyr, labelled, rlang, gtsummary, forcats, gt, cardx,
glue, purrr, ggh4x, ggplot2, ggrepel, scales, viridis

Suggests testthat, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author USMR CHU de Bordeaux [aut, cre],
Valentine Renaudeau [aut],
Marion Kret [aut],
Matisse Decilap [aut],
Sahardid Mohamed Houssein [aut],
Thomas Ferté [aut]

Maintainer USMR CHU de Bordeaux <astreinte.usmr@chu-bordeaux.fr>

Repository CRAN

Date/Publication 2025-11-20 16:00:02 UTC

Contents

add_by_n 2
add_by_n_noNA 4
add_missing_info 5
add_pvalues 6
ajouter_label_ndm 6
base_table 7
css_generator 8
customize_table 9
customize_table_body 11
custom_format 12
custom_headers 13
custom_round 14
desc_ei_per_grade 14
desc_ei_per_pt 15
desc_var 16
df_builder_ae 19
from_name_to_adress 19
intermediate_header 20
modify_table_body_func 21
ordonner_variables_qualitatives 22
plot_butterfly_stacked_barplot 23
plot_dumbell 25
plot_patient_panchart 26
plot_volcano 27
prepare_table 29
reverselog_trans 30
riskdifference 31
select_plus 31
start_new_reporting 32
write_css 34
write_datestamp_output_file 34
write_html_file 35
write_qmd 36
write_quarto_yml 37

Index **38**

`add_by_n` *Add Counts and Percentages of Missing Data by Group*

Description

This function calculates and summarizes the counts and percentages of missing and non-missing values for a specified variable, grouped by another variable. It provides formatted output for integration into summary tables.

Usage

```
add_by_n(data, variable, by, tbl, ...)
```

Arguments

`data`	A data frame containing the dataset to analyze.
`variable`	A character string specifying the target variable for which missing data statistics will be computed.
`by`	A character string specifying the grouping variable. The data will be grouped by this variable before calculating the statistics.
`tbl`	Not used in the current implementation but retained for compatibility with the `gtsummary` framework.
`...`	Additional arguments (not used in the current implementation).

Details

The function performs the following steps:

1. Groups the data by the variable specified in `by`.
2. Computes the number of non-missing values (`nb`), the number of missing values (`nb_NA`), and the percentage of missing values (`nb_percent`) for the specified `variable`.
3. Renames and formats the output columns for clarity and readability.
4. Converts the data into a wide format suitable for integration into summary tables, with calculated statistics included in formatted strings (e.g., "value (missing_count ; missing_percent%)").

The output is designed for use with summary tools, such as `gtsummary`, to display detailed missing data statistics alongside descriptive statistics.

Value

A data frame in wide format, where each row represents a group (as defined by `by`), and columns include statistics for the target variable (`variable`) in a formatted string.

Examples

```
# Example usage:
library(dplyr)
library(tidyr)
data(mtcars)

# Add missing data statistics grouped by 'cyl'
add_by_n(
  data = mtcars,
  variable = "mpg",
  by = "cyl"
)
```

add_by_n_noNA *Add Counts by Group*

Description

This function calculates and summarizes the counts and percentages of non-missing values for a specified variable, grouped by another variable. It provides formatted output for integration into summary tables.

Usage

```
add_by_n_noNA(data, variable, by, tbl, ...)
```

Arguments

`data`	A data frame containing the dataset to analyze.
`variable`	A character string specifying the target variable for which missing data statistics will be computed.
`by`	A character string specifying the grouping variable. The data will be grouped by this variable before calculating the statistics.
`tbl`	Not used in the current implementation but retained for compatibility with the `gtsummary` framework.
`...`	Additional arguments (not used in the current implementation).

Details

The function performs the following steps:

1. Groups the data by the variable specified in `by`.
2. Computes the number of non-missing values (`nb`) for the specified `variable`.
3. Renames and formats the output columns for clarity and readability.
4. Converts the data into a wide format suitable for integration into summary tables, with calculated statistics included in formatted strings.

The output is designed for use with summary tools, such as `gtsummary`, to display detailed missing data statistics alongside descriptive statistics.

Value

A data frame in wide format, where each row represents a group (as defined by `by`), and columns include statistics for the target variable (`variable`) in a formatted string.

Examples

```
# Example usage:
library(dplyr)
library(tidyr)
data(mtcars)

# Add  data statistics grouped by 'cyl'
add_by_n(
  data = mtcars,
  variable = "mpg",
  by = "cyl"
)
```

`add_missing_info` *Add missing value information to a gtsummary table*

Description

This function adds information about missing and non-missing data counts to a gtsummary table. It can also apply custom statistics by group and modify the table body with an external function.

Usage

```
add_missing_info(
  base_table,
  show_missing_data,
  var_group = NULL,
  by_group = FALSE
)
```

Arguments

`base_table`	A `gtsummary` table object.
`show_missing_data`	Logical. If `TRUE`, shows the number of non-missing and missing values with percentages. If `FALSE`, shows only non-missing values.
`var_group`	Optional. A grouping variable name. If not `NULL`, additional stats are added by group.
`by_group`	A boolean (default is FALSE) to analyse by group.

Value

A `gtsummary` table object with missing value information and modifications applied.

`add_pvalues` *Add p-values and separate footnotes to a gtsummary object*

Description

This function adds p-values to a gtsummary table using the specified tests and separates the p-value footnotes.

Usage

```
add_pvalues(res, tests)
```

Arguments

`res`	A `gtsummary` table object.
`tests`	A list of tests to pass to `gtsummary::add_p()`, or `TRUE` to use default tests.

Value

A `gtsummary` table object with p-values added and footnotes separated.

Examples

```
library(gtsummary)
tbl <- trial %>% tbl_summary(by = trt)
tbl <- add_pvalues(tbl, tests = TRUE)
```

`ajouter_label_ndm` *Add "n (dm ; %dm)" to Variable Labels*

Description

This function appends the text "n (dm ; %dm)" to the labels of all variables in a dataset. It uses the `labelled` package to modify and update variable labels in-place.

Usage

```
ajouter_label_ndm(data, col_to_skip = NULL)
```

Arguments

`data`	A data frame containing the dataset whose variable labels need to be updated.
`col_to_skip`	A column to skip when adding `"n (dm ; %dm)"`. Default is `NULL`.

Details

The function iterates over all columns in the dataset and performs the following steps:

1. Retrieves the current label of each variable using `labelled::var_label`.
2. Creates a new label by appending the text `"n (dm ; %dm)"` to the existing label.
3. Updates the variable's label using `labelled::set_variable_labels`.

This is useful when preparing a dataset for descriptive analysis, where it is helpful to display missing data statistics (`n`, `dm`, and `%dm`) alongside variable labels in summary tables.

Value

A data frame with updated variable labels.

Examples

```
# Example usage:
library(labelled)

# Create a sample dataset
data <- data.frame(
  var1 = c(1, 2, NA),
  var2 = c("A", "B", NA)
)

# Assign initial labels
data <- labelled::set_variable_labels(
  data,
  var1 = "Variable 1",
  var2 = "Variable 2"
)

# Add "n (dm ; %dm)" to labels
data <- ajouter_label_ndm(data)

# Check updated labels
labelled::var_label(data)
```

`base_table` *Create a Summary Table with Grouping and Custom Formatting*

Description

This function generates a summary table from a data frame with specified grouping and variable types. It uses the `gtsummary` package to create descriptive statistics for categorical and continuous variables, with options for customizing the rounding and labels.

Usage

```
base_table(
  data1,
  by_group = FALSE,
  var_group,
  quali = NULL,
  quanti = NULL,
  digits = list(mean_sd = 1, median_q1_q3_min_max = 1, pct = 1)
)
```

Arguments

`data1`	A data frame containing the data to summarize.
`by_group`	A boolean (default is FALSE) to analyse by group.
`var_group`	A string or NULL, the variable to group by (optional). If NULL, no grouping will be applied.
`quali`	A character vector, the names of categorical variables to treat as categorical in the summary table.
`quanti`	A character vector, the names of continuous variables to treat as continuous in the summary table.
`digits`	A list, the number of decimal places to round categorical and continuous variable. Default is list(mean_sd = 1, median_q1_q3_min_max = 1, pct = 1).

Value

A `gtsummary` table summarizing the specified variables, grouped by `var_group` if provided, with customizable statistics and rounding options.

Examples

```
# Example usage with the iris dataset
base_table(iris, var_group = "Species")
```

`css_generator` *css_generator*

Description

Generate css to be included in quarto.

Usage

```
css_generator(path_logo = NULL)
```

Arguments

`path_logo`	The path to logo, will automatically be guessed on the package.

Value

A css string

`customize_table` *Customize a Summary Table with Grouping, Missing Data, and Custom Titles*

Description

This function customizes a `gtsummary` summary table by adding an overall column, handling missing data, applying group-specific statistics, and updating headers and captions. It provides flexible options for grouping, displaying missing data, and customizing table titles.

Usage

```
customize_table(
  base_table,
  by_group = FALSE,
  var_group,
  add_total,
  show_missing_data,
  show_n_per_group,
  group_title,
  table_title,
  var_title,
  var_tot = NULL,
  var_characteristic = NULL
)
```

Arguments

`base_table`	A `gtsummary` table object, typically generated using functions like `gtsummary::tbl_summary`.
`by_group`	A boolean (default is FALSE) to analyse by group.
`var_group`	A string or NULL, specifying the variable used for grouping in the table. If `NULL`, no group-specific modifications are applied.
`add_total`	A boolean to add total column or not when var_group is specified.
`show_missing_data`	A boolean indicating whether to display missing data counts and percentages in the table. If `TRUE`, columns for missing data will be added.
`show_n_per_group`	A boolean indicating whether to display group sizes (n) for each level of the grouping variable.

`group_title`	A string specifying the title for the group column in the table.
`table_title`	A string specifying the title of the entire table.
`var_title`	A string specifying the title for the variable column in the table.
`var_tot`	A string specifying the name of total column. Default is `NULL` and will guess from `theme_gtsummary_language()`.
`var_characteristic`	A string specifying the name of characteristic column. Default is `NULL` and will guess from `theme_gtsummary_language()`.

Details

- The `show_missing_data` parameter determines whether missing data counts and percentages are displayed:
  - If `TRUE`, missing data columns are added.
  - If `FALSE`, only non-missing data counts are displayed.
- Headers for columns and spanning headers are customized using the `group_title`, `table_title`, and `var_title` arguments.
- An external function `modify_table_body_func` is called to further modify the table body.

Value

A customized `gtsummary` table object with added columns, headers, captions, and modifications based on the provided arguments.

Examples

```
# Example usage with a sample gtsummary table
library(gtsummary)
base_table <- trial %>%
  tbl_summary(by = "trt", missing = "no")

customize_table(
  base_table,
  var_group = "trt",
  add_total = TRUE,
  show_missing_data = TRUE,
  show_n_per_group = FALSE,
  group_title = "Treatment Group",
  table_title = "Summary Statistics",
  var_title = "Variables",
  var_tot = "Total"
)
```

customize_table_body *Customize Table Body*

Description

This function modifies a data frame by updating the `stat_0` column. If any values in `stat_0` are missing (`NA`), they are replaced by the values from the `n` column. After the replacement, the `n` column is removed from the data frame.

Usage

```
customize_table_body(data)
```

Arguments

`data`	A data frame that must contain at least two columns: • `stat_0`: A column whose missing (`NA`) values are to be replaced. • `n`: A column providing replacement values for `stat_0` when its values are missing.

Details

- The function uses `dplyr::case_when` to conditionally update the `stat_0` column.
- After the replacement process, the `n` column is dropped using `dplyr::select(-n)`.
- This function is particularly useful for cleaning and preparing table data.

Value

A modified data frame with:

- Updated `stat_0` values (replaced with `n` values where `NA` is found).
- The `n` column removed after integration.

Examples

```
# Example data
data <- data.frame(
  stat_0 = c(NA, "B", "C"),
  n = c(10, 20, 30)
)

# Apply the function
modified_data <- customize_table_body(data)
print(modified_data)
```

`custom_format` *Custom formatting for* `gtsummary` *tables*

Description

This function takes a `gt` table and applies custom formatting. It allows you to align columns, apply bold text to certain rows, and adjust column widths if specified.

Usage

```
custom_format(gt_table, align = "right", column_size = NULL)
```

Arguments

`gt_table`	A `gt` table object (also handles gtsummary tables by converting them).
`align`	A character string defining the alignment of specific columns. Passed to the `gt::cols_align()` function (e.g., "left", "right", "center"). Default is "right".
`column_size`	A named list or vector defining the width of columns (optional). The list should specify the width for one or more columns. If not provided, column widths will not be modified.

Value

A `gt` table object with the specified formatting applied. The table will have columns aligned according to the `align` parameter, and cells in the "label" rows will have bold text. If `column_size` is provided, the column widths will be adjusted accordingly.

Examples

```
# Example usage
tbl <- RastaRocket::desc_var(iris,
  table_title = "test",
  group = TRUE,
  var_group = "Species")
formatted_tbl <- custom_format(tbl,
  align = "center",
  column_size = list(label ~ gt::pct(50)))
formatted_tbl
```

`custom_headers` *Modify gtsummary table headers and add a spanning header*

Description

This function customizes the column headers, optional spanning header, and table caption for a `gtsummary` table. It supports adding a feature name, total label, group title, and formats missing data presentation.

Usage

```
custom_headers(
  base_table_missing,
  var_characteristic = NULL,
  show_missing_data = TRUE,
  show_n_per_group = TRUE,
  var_tot = NULL,
  var_group = NULL,
  group_title = NULL,
  table_title
)
```

Arguments

`base_table_missing`	A `gtsummary` table object (typically output from previous steps).
`var_characteristic`	Optional. A string to label the features column.
`show_missing_data`	Logical. If `TRUE`, adds missing data info to column headers.
`show_n_per_group`	A boolean indicating whether to display group sizes (n) for each level of the grouping variable.
`var_tot`	Optional. A string to label the total column.
`var_group`	Optional. Name of a grouping variable for adding a spanning header.
`group_title`	Optional. Title for the spanning header. If `NULL`, uses the variable label or name.
`table_title`	Title for the entire table.

Value

A `gtsummary` table object with updated headers, spanning header, and caption.

`custom_round` *Custom Round and Format*

Description

Rounds a numeric value to a specified number of decimal places and formats it to always show the specified number of decimal places, including trailing zeros.

Usage

```
custom_round(x, digits = 1)
```

Arguments

`x`	A numeric vector to be rounded and formatted.
`digits`	An integer indicating the number of decimal places to round to. Defaults to 1.

Value

A character vector with the rounded and formatted numbers.

Examples

```
custom_round(3.14159)      # "3.1"
custom_round(3.14159, 3)   # "3.142"
custom_round(c(2, 2.5), 2) # "2.00" "2.50"
```

`desc_ei_per_grade` *desc_ei_per_grade*

Description

A function to describe adverse events (AE) by grade.

Usage

```
desc_ei_per_grade(df_pat_grp, df_pat_grade, severity = TRUE, digits = 1)
```

Arguments

`df_pat_grp`	A dataframe with two columns: USUBJID and RDGRPNAME (the RCT arm).
`df_pat_grade`	A dataframe with three columns: USUBJID, EINUM (the AE id), EIGRDM (the AE grade) and EIGRAV (the AE severity which must be "Grave" and "Non grave").
`severity`	A boolean to show severe adverse event line or not.
`digits`	Number of digits for percentages

Value

A gt table summarizing the AE by grade.

Examples

```
df_pat_grp <- data.frame(USUBJID = paste0("ID_", 1:10),
                         RDGRPNAME = c(rep("A", 3), rep("B", 3), rep("C", 4)))

df_pat_grade <- data.frame(USUBJID = c("ID_1", "ID_1",
                                       "ID_2",
                                       "ID_8",
                                       "ID_9"),
                           EINUM = c(1, 2,
                                     1,
                                     1,
                                     1),
                           EIGRDM = c(1, 3,
                                      4,
                                      2,
                                      4),
                           EIGRAV = c("Grave", "Non grave",
                                      "Non grave",
                                      "Non grave",
                                      "Grave"))

desc_ei_per_grade(df_pat_grp = df_pat_grp,
                  df_pat_grade = df_pat_grade)
```

---

desc_ei_per_pt *desc_ei_per_pt*

Description

A function to describe AE by soc and pt

Usage

```
desc_ei_per_pt(
  df_pat_grp,
  df_pat_llt,
  language = "fr",
  order_by_freq = TRUE,
  digits = 1
)
```

Arguments

`df_pat_grp`	A dataframe with two columns: id_pat and grp (the rct arm)
`df_pat_llt`	A dataframe with two columns: id_pat (patient id), num_ae (AE id), llt (AE LLT), pt (AE PT), soc (AE)
`language`	'fr' default or 'en'
`order_by_freq`	Logical. Should PT and SOC be ordered by frequency? Defaults to TRUE. If FALSE, PT and SOC are ordered alphabetically.
`digits`	Number of digits for percentages

Value

A gt table

Examples

```
df_pat_grp <- data.frame(USUBJID = paste0("ID_", 1:10),
                         RDGRPNAME = c(rep("A", 3), rep("B", 3), rep("C", 4)))

df_pat_llt <- data.frame(USUBJID = c("ID_1", "ID_1",
                                     "ID_2",
                                     "ID_4",
                                     "ID_9"),
                         EINUM = c(1, 2, 1, 1, 1),
                         EILLTN = c("llt1", "llt1",
                                "llt4", "llt3",
                                "llt1"),
                         EIPTN = c("Arrhythmia", "Myocardial Infarction",
                                "Arrhythmia", "Pneumonia",
                                "Pneumonia"),
                         EISOCPN = c("Cardiac Disorders", "Cardiac Disorders",
                                "Cardiac Disorders", "Infections",
                                "Infections"))

desc_ei_per_pt(df_pat_grp = df_pat_grp,
               df_pat_llt = df_pat_llt)
```

`desc_var` *Generate Descriptive Tables for Variables*

Description

This function creates descriptive tables for variables in a dataset. It can handle qualitative and quantitative variables, grouped or ungrouped, and supports multiple configurations for handling missing data (DM).

Usage

```
desc_var(
  data1,
  table_title = "",
  quali = NULL,
  quanti = NULL,
  add_total = TRUE,
  var_title = "Variable",
  by_group = FALSE,
  var_group = NULL,
  group_title = NULL,
  digits = list(mean_sd = 1, median_q1_q3_min_max = 1, pct = 1),
  drop_levels = TRUE,
  freq_relevel = FALSE,
  tests = FALSE,
  show_n_per_group = FALSE,
  show_missing_data = NULL,
  var_tot = NULL,
  var_characteristic = NULL
)
```

Arguments

`data1`	A data frame containing the dataset to be analyzed.
`table_title`	A character string specifying the title of the table.
`quali`	A vector of qualitative variables to be described. Defaults to `NULL`.
`quanti`	A vector of quantitative variables to be described. Defaults to `NULL`.
`add_total`	A boolean (default is TRUE) to add total column or not when var_group is specified.
`var_title`	A character string for the title of the variable column in the table. Defaults to `"Variable"`.
`by_group`	A boolean (default is FALSE) to analyse by group.
`var_group`	A variable used for grouping (if applicable). Defaults to `NULL`.
`group_title`	A character string specifying the title for the grouping variable. Default is `NULL` and get the label or the variable.
`digits`	A list, the number of decimal places to round categorical and continuous variable. Default is list(mean_sd = 1, median_q1_q3_min_max = 1, pct = 1).
`drop_levels`	Boolean (default = TRUE). Drop unused levels.
`freq_relevel`	Boolean (default = FALSE). Reorder factors by frequency.
`tests`	A value in order to add p value. Default to `FALSE` OPTION : • `FALSE`: No p-value add • `TRUE`: Add p-value made by default by gtsummary. See gtsummary add_p() options. • `list()`: To force tests. See gtsummary add_p() options.

`show_n_per_group`
: Default to `FALSE`. Should the 'N' appears in the column header of the groups. Can be either :
  - `FALSE`: No 'N' is shown
  - `TRUE`: 'N' is shown

`show_missing_data`
: Default to `NULL`. Should the missing data be displayed. Can be either :
  - `FALSE`: No missing data displayed
  - `TRUE`: Missing data displayed
  - `NULL` (default): will be switch to `anyNA(data1)` value.

`var_tot`
: A string specifying the name of total column. Default is `NULL` and will guess from `theme_gtsummary_language()`.

`var_characteristic`
: A string specifying the name of characteristic column. Default is `NULL` and will guess from `theme_gtsummary_language()`.

Details

The function processes the dataset according to the specified parameters and generates descriptive tables.

- It first uses the `ajouter_label_ndm()` function to append missing data statistics to variable labels.
- Depending on the `group` and `DM` arguments, it adjusts the dataset and creates tables using helper functions like `desc_group`, `desc_degroup`, and `desc_degroup_group`.
- Qualitative variables are reordered, and unused levels are dropped when necessary.

Value

A gtsummary table object containing the descriptive statistics.

Examples

```
# Example usage:
library(dplyr)

# Sample dataset
data1 <- data.frame(
  group = c("A", "B", "A", "C"),
  var1 = c(1, 2, 3, NA),
  var2 = c("X", "Y", "X", NA)
)

# Generate descriptive table
table <- desc_var(
  data1 = data1,
  table_title = "Descriptive Table"
)
```

`df_builder_ae` *Prepare a dataframe for creating AE plots*

Description

Prepare a dataframe for creating AE plots

Usage

```
df_builder_ae(df_pat_grp, df_pat_llt, ref_grp = NULL)
```

Arguments

`df_pat_grp`	A data frame of patient groups. Must contain columns `USUBJID` (patient ID) and `RDGRPNAME` (group assignment).
`df_pat_llt`	A data frame with USUBJID (subject ID), EINUM (AE ID), EILLTN (LLT identifier), EIPTN (PT identifier), EISOCPN (soc identifier) and EIGRDM (severity grade)
`ref_grp`	(Optional) A reference group for comparisons. Defaults to the first group in `df_pat_grp`.

Value

A dataframe with all the info to build AE plots

`from_name_to_adress` *Convert a Name to an Email Address*

Description

This function transforms a given name into an email address following the format `firstname.lastname@chu-bordeaux.fr`.

Usage

```
from_name_to_adress(name = "Peter Parker")
```

Arguments

`name`	A character string representing a full name. Default is "Peter Parker".

Value

A character string containing the generated email address.

Examples

```
from_name_to_adress("John Doe")
from_name_to_adress()
```

---

intermediate_header *Intermediate Header*

Description

Combines multiple descriptive tables into a single table with customized row group headers and styling. This function accepts a list of tables and corresponding group headers, applies consistent styling, and outputs a styled `gt` table.

Usage

```
intermediate_header(
  tbls,
  group_header,
  color = "#8ECAE6",
  size = 16,
  align = "center",
  weight = "bold"
)
```

Arguments

`tbls`	A list of descriptive tables (generated by `RastaRocket::desc_var` or similar functions).
`group_header`	A character vector specifying the headers for each group of tables. Must be the same length as `tbls`.
`color`	A character string specifying the background color for the row group headers. Default is `"#8ECAE6"`.
`size`	An integer specifying the font size for the row group headers. Default is `16`.
`align`	A character string specifying text alignment for the row group headers. Options are `"left"`, `"center"`, or `"right"`. Default is `"center"`.
`weight`	A character string specifying the font weight for the row group headers. Options include `"normal"`, `"bold"`, etc. Default is `"bold"`.

Value

A styled `gt` table combining the input tables with row group headers.

Examples

```
# Load necessary libraries
library(RastaRocket)
library(dplyr)

# Generate sample data
data <- data.frame(
  Age = c(rnorm(45, mean = 50, sd = 10), rep(NA, 5)),
  sexe = sample(c("Femme", "Homme"), 50, replace = TRUE, prob = c(0.6, 0.4)),
  quatre_modalites = sample(c("A", "B", "C", "D"), 50, replace = TRUE)
)

# Create descriptive tables
tb1 <- data %>%
  dplyr::select(Age, sexe) %>%
  RastaRocket::desc_var(table_title = "Demographics", group = FALSE)

tb2 <- data %>%
  dplyr::select(quatre_modalites) %>%
  RastaRocket::desc_var(table_title = "Modalities", group = FALSE)

# Combine and style tables
intermediate_header(
  tbls = list(tb1, tb2),
  group_header = c("Demographics", "Modalities")
)
```

`modify_table_body_func` *Modify Table Body*

Description

This function modifies a table by updating the "stat_*" columns with corresponding "add_n_stat_*" columns if they exist. If "stat_*" columns contain missing values (*`NA`*), the function replaces them with the respective "add_n_stat_*" column values. Extra "add_n_stat_*" columns are removed after processing.

Usage

```
modify_table_body_func(data)
```

Arguments

`data`	A data frame that contains columns named `stat_*` and optionally `add_n_stat_*`. The function assumes that `stat_*` columns are to be updated using `add_n_stat_*` columns when necessary.

Details

- The function identifies columns starting with "add_n_stat_" and attempts to use them to fill missing values in columns matching the pattern "^stat_\d+$".
- If all required "add_n_stat_*" columns exist in `data`, they are utilized for this replacement; otherwise, the `add_n_stat_*` columns are removed without modifications to the `stat_*` columns.

Value

A modified data frame where:

- `stat_*` columns are updated to replace `NA` values with the corresponding values from `add_n_stat_*`.
- `add_n_stat_*` columns are removed after processing.
- If no `add_n_stat_*` columns exist, these columns are simply removed.

Examples

```
# Example data
data <- data.frame(
  n = c(1, 2, 3),
  stat_1 = c(NA, 5, 6),
  stat_2 = c(7, NA, 9),
  add_n_stat_1 = c(10, 11, 12),
  add_n_stat_2 = c(13, 14, 15)
)

# Apply the function
modified_data <- modify_table_body_func(data)
print(modified_data)
```

`ordonner_variables_qualitatives` *Reorder Levels of Qualitative Variables by Frequency*

Description

This function reorders the levels of all qualitative (factor) variables in a dataset based on their frequency, in descending order. It ensures that the most frequent levels appear first when analyzing or visualizing the data.

Usage

```
ordonner_variables_qualitatives(data)
```

Arguments

`data`	A data frame containing the dataset with qualitative variables to reorder.

Details

The function applies the following transformations:

- Identifies all columns of type `factor` in the dataset.
- Reorders the levels of each factor variable using the `forcats::fct_infreq()` function, which orders levels by decreasing frequency.

This is particularly useful for preparing datasets for visualization or analysis, where it can be helpful to have the most common levels displayed first.

Value

A data frame with reordered levels for all factor variables. Non-factor variables remain unchanged.

Examples

```
# Example usage:
library(dplyr)
library(forcats)

# Create a sample dataset
data <- data.frame(
  var1 = factor(c("A", "B", "A", "C", "B", "B")),
  var2 = factor(c("X", "Y", "X", "Y", "X", "Z")),
  var3 = c(1, 2, 3, 4, 5, 6)  # Non-factor variable
)

# Reorder qualitative variables by frequency
data <- ordonner_variables_qualitatives(data)

# Check the new order of levels
levels(data$var1) # Output: "B" "A" "C"
levels(data$var2) # Output: "X" "Y" "Z"
```

`plot_butterfly_stacked_barplot`

Butterfly Stacked Bar Plot for Adverse Event Grades

Description

Creates a butterfly stacked bar plot to visualize the frequency of adverse event (AE) grades across patient groups, with system organ class (SOC) and preferred terms (PT) as labels.

Usage

```
plot_butterfly_stacked_barplot(
  df_pat_grp,
  df_pat_llt,
  ref_grp = NULL,
  max_text_width = 9,
  vec_fill_color = viridis::viridis(n = 4)
)
```

Arguments

`df_pat_grp`	A data frame of patient groups. Must contain columns `USUBJID` (patient ID) and `RDGRPNAME` (group assignment).
`df_pat_llt`	A data frame with USUBJID (subject ID), EINUM (AE ID), EILLTN (LLT identifier), EIPTN (PT identifier), EISOCPN (soc identifier) and EIGRDM (severity grade)
`ref_grp`	A character string specifying the reference group (used for alignment in the plot). If NULL (default), the first level of `df_pat_grp$grp` is used.
`max_text_width`	An integer specifying the maximum width (in characters) for SOC labels before wrapping to the next line. Default is 9.
`vec_fill_color`	A vector of colors used for filling the AE grade bars. Default is `viridis::viridis(n = 4)`.

Details

The function processes input data to calculate the frequency of adverse events per patient group and AE grade. It then generates a stacked bar plot where:

- The x-axis represents the percentage of patients experiencing an AE.
- The y-axis represents PTs (with SOCs as facets).
- Bars are stacked by AE grade.
- Labels for PTs are displayed in the center.
- The left and right panels correspond to different patient groups.

The function utilizes the ggh4x package to adjust panel sizes and axes for a symmetrical butterfly plot.

Value

A ggplot2 object representing the butterfly stacked bar plot.

Examples

```
df_pat_grp <- data.frame(
 USUBJID = paste0("ID_", 1:10),
 RDGRPNAME = c(rep("A", 5), rep("B", 5))
)
```

```
df_pat_llt <- data.frame(
  USUBJID = c("ID_1", "ID_1", "ID_2", "ID_4", "ID_9"),
  EINUM = c(1, 2, 1, 1, 1),
  EILLTN = c("llt1", "llt2", "llt1", "llt3", "llt4"),
 EIPTN = c("Arrhythmia", "Myocardial Infarction", "Arrhythmia", "Pneumonia", "Pneumonia"),
  EISOCPN = c("Cardiac Disorders", "Cardiac Disorders", "Cardiac Disorders",
  "Infections", "Infections"),
  EIGRDM = c(1, 3, 4, 2, 4)
)

plot_butterfly_stacked_barplot(df_pat_grp, df_pat_llt)
```

---

plot_dumbell *Plot a Dumbbell Chart for Adverse Events Analysis*

Description

This function creates a dumbbell plot comparing the occurrence of adverse events across different patient groups. The plot includes the total number of adverse events, the proportion of patients affected, and the risk difference with confidence intervals.

Usage

```
plot_dumbell(
  df_pat_grp,
  df_pat_llt,
  ref_grp = NULL,
  colors_arm = c("#1b9e77", "#7570b3"),
  color_label = "Arm"
)
```

Arguments

`df_pat_grp`	A data frame of patient groups. Must contain columns `USUBJID` (patient ID) and `RDGRPNAME` (group assignment).
`df_pat_llt`	A data frame with USUBJID (subject ID), EINUM (AE ID), EILLTN (LLT identifier), EIPTN (PT identifier), EISOCPN (soc identifier) and EIGRDM (severity grade)
`ref_grp`	(Optional) A reference group for comparisons. Defaults to the first group in `df_pat_grp`.
`colors_arm`	A vector of colors for the patient groups. Defaults to `c("#1b9e77", "#7570b3")`.
`color_label`	A string specifying the legend label for the groups. Defaults to `"Arm"`.

Value

A `ggplot` object displaying the dumbbell chart.

Examples

```
df_pat_grp <- data.frame(
 USUBJID = paste0("ID_", 1:10),
 RDGRPNAME = c(rep("A", 5), rep("B", 5))
)

df_pat_llt <- data.frame(
  USUBJID = c("ID_1", "ID_1", "ID_2", "ID_4", "ID_9"),
  EINUM = c(1, 2, 1, 1, 1),
  EILLTN = c("llt1", "llt2", "llt1", "llt3", "llt4"),
 EIPTN = c("Arrhythmia", "Myocardial Infarction", "Arrhythmia", "Pneumonia", "Pneumonia"),
  EISOCPN = c("Cardiac Disorders", "Cardiac Disorders", "Cardiac Disorders",
  "Infections", "Infections"),
  EIGRDM = c(1, 3, 4, 2, 4)
)

plot_dumbell(df_pat_llt = df_pat_llt, df_pat_grp = df_pat_grp)
```

`plot_patient_panchart` *Plot a Patient Span Chart (Panchart)*

Description

This function visualizes the timeline of adverse events (AEs), treatments, and randomization for a selected patient. The span chart helps track AE duration and treatment events relative to randomization.

Usage

```
plot_patient_panchart(
  df_soc_pt,
  df_pat_grp_rando,
  df_pat_pt_grade_date,
  df_pat_treatment_date,
  pat_id,
 vec_fill_color = viridis::viridis(n = 4, direction = -1, end = 0.95, option = "magma")
)
```

Arguments

`df_soc_pt`	A data frame mapping System Organ Class (SOC) to Preferred Terms (PT).
`df_pat_grp_rando`	A data frame containing patient IDs, randomization groups, and randomization dates.
`df_pat_pt_grade_date`	A data frame with patient IDs, PT terms, AE grades, start and end dates of AEs.

`df_pat_treatment_date`	A data frame with patient IDs and treatment dates.
`pat_id`	A character string specifying the patient ID to plot.
`vec_fill_color`	A vector of colors for AE grades. Default is `viridis::viridis(n = 4)`.

Value

A ggplot object representing the patient span chart.

Examples

```
df_pat_grp_rando <- data.frame(
  id_pat = c("ID_1", "ID_2"),
  grp = c("A", "B"),
  rando_date = c("2020-12-01", "2021-01-03")
)

df_pat_pt_grade_date <- data.frame(
  id_pat = c("ID_1", "ID_1", "ID_1", "ID_1", "ID_2"),
  pt = c("Arrhythmia", "Myocardial Infarction", "Arrhythmia",
         "Pneumonia", "Pneumonia"),
  grade = c(4, 2, 1, 3, 4),
  start = c("2021-01-01", "2021-02-03", "2021-01-02", "2021-03-05", "2021-02-01"),
  end = c("2021-01-14", "2021-03-03", "2021-01-22", "2021-05-05", "2021-02-03")
)

df_pat_treatment_date <- data.frame(
  id_pat = c("ID_1", "ID_1", "ID_1"),
  treatment_date = c("2021-01-25", "2021-03-01", "2021-01-20")
)
df_soc_pt <- data.frame(
  pt = c("Arrhythmia", "Myocardial Infarction", "Pneumonia", "Sepsis"),
  soc = c("Cardiac Disorders", "Cardiac Disorders", "Infections", "Infections")
)

plot_patient_panchart(
  df_soc_pt = df_soc_pt,
  df_pat_grp_rando = df_pat_grp_rando,
  df_pat_pt_grade_date = df_pat_pt_grade_date,
  df_pat_treatment_date = df_pat_treatment_date,
  pat_id = "ID_1"
)
```

`plot_volcano` *Volcano Plot for Adverse Event Analysis*

Description

Generates a volcano plot to visualize the association between adverse events and patient groups.

Usage

```
plot_volcano(
  df_pat_grp,
  df_pat_llt,
  ref_grp = NULL,
  colors_arm = c("#1b9e77", "#7570b3"),
  size = "nb_pat"
)
```

Arguments

`df_pat_grp`	A data frame of patient groups. Must contain columns `USUBJID` (patient ID) and `RDGRPNAME` (group assignment).
`df_pat_llt`	A data frame with USUBJID (subject ID), EINUM (AE ID), EILLTN (LLT identifier), EIPTN (PT identifier), EISOCPN (soc identifier) and EIGRDM (severity grade)
`ref_grp`	(Optional) A reference group for comparisons. Defaults to the first group in `df_pat_grp`.
`colors_arm`	A character vector of length two specifying the colors for the two patient groups in the plot. Default is `c("#1b9e77", "#7570b3")`.
`size`	A character string specifying the metric used for point sizes in the plot. Options are: • `"nb_pat"`: Number of patients (default). • `"nb_ei"`: Number of adverse events.

Details

The function first processes input data using `df_builder_ae()`, then calculates relevant statistics such as risk difference (`RD`) and p-values. The volcano plot displays:

- `RD` on the x-axis (risk difference between groups).
- `-log10(p-value)` on the y-axis (significance level).
- Point colors indicating which group has an increased risk.
- Point sizes reflecting either the number of patients or events.
- A horizontal dashed line at p = 0.05 for significance threshold.

Value

A ggplot2 object representing the volcano plot.

Examples

```
df_pat_grp <- data.frame(
 USUBJID = paste0("ID_", 1:10),
 RDGRPNAME = c(rep("A", 5), rep("B", 5))
)
```

```
df_pat_llt <- data.frame(
  USUBJID = c("ID_1", "ID_1", "ID_2", "ID_4", "ID_9"),
  EINUM = c(1, 2, 1, 1, 1),
  EILLTN = c("llt1", "llt2", "llt1", "llt3", "llt4"),
 EIPTN = c("Arrhythmia", "Myocardial Infarction", "Arrhythmia", "Pneumonia", "Pneumonia"),
  EISOCPN = c("Cardiac Disorders", "Cardiac Disorders", "Cardiac Disorders",
  "Infections", "Infections"),
  EIGRDM = c(1, 3, 4, 2, 4)
)

plot_volcano(df_pat_grp, df_pat_llt)
```

`prepare_table` *Prepare a Data Frame for Summarization with Custom Missing Data Handling and Factor Ordering*

Description

This function prepares a data frame for summarization by handling missing data based on the `show_missing_data` argument and applying the specified data manipulation (DM) option to factor variables. It provides flexibility for data cleaning and ordering before summarizing with functions like `gtsummary`.

Usage

```
prepare_table(
  data1,
  by_group = FALSE,
  var_group = NULL,
  drop_levels = TRUE,
  freq_relevel = FALSE,
  show_missing_data = TRUE
)
```

Arguments

`data1`	A data frame containing the data to be prepared.
`by_group`	A boolean (default is FALSE) to analyse by group.
`var_group`	The group variable (used to correctly update the label if needed).
`drop_levels`	Boolean (default = TRUE). Drop unused levels.
`freq_relevel`	Boolean (default = FALSE). Reorder factors by frequency.
`show_missing_data`	Should the missing data be displayed. Can be either : • `FALSE`: No missing data displayed • `TRUE`(default): Missing data displayed

Details

- The `DM` option defines the data manipulation to be applied to factor variables:
  - `"tout"`: Both order factor levels and drop unused levels.
  - `"tri"`: Only order factor levels.
  - `"remove"`: Drop unused factor levels without ordering.

Value

A data frame that has been prepared based on the `show_missing_data` and `DM` arguments. The function modifies the input data frame by applying labels, ordering factor variables, and potentially dropping unused levels.

Examples

```
# Example usage with the iris dataset
prepare_table(iris)
```

`reverselog_trans` *Reverse Log Transformation*

Description

Creates a transformation object for a reverse log scale, which can be used in ggplot2 scales.

Usage

```
reverselog_trans(base = exp(1))
```

Arguments

`base`	A numeric value specifying the logarithm base. Default is the natural logarithm (`exp(1)`).

Details

This function defines a reverse logarithmic transformation, where the transformation function is

$$-\log(x, \text{base})$$

and the inverse function is

$$\text{base}^{-x}$$

. It is useful for cases where a decreasing log scale is needed.

Value

A transformation object compatible with ggplot2 scales.

Examples

```
library(scales)
rev_log <- reverselog_trans(10)
rev_log$trans(100)  # -2
rev_log$inverse(-2) # 100
```

---

riskdifference *riskdifference*

Description

A function from the fmsb package to compute risk difference. Calculate risk difference (a kind of attributable risk / excess risk) and its confidence intervals based on approximation, followed by null hypothesis (risk difference equals to 0) testing.

Usage

```
riskdifference(a, b, N1, N0, CRC = FALSE, conf.level = 0.95)
```

Arguments

a	The number of disease occurence among exposed cohort.
b	The number of disease occurence among non-exposed cohort.
N1	The population at risk of the exposed cohort.
N0	The population at risk of the unexposed cohort.
CRC	Logical. If TRUE, calculate confidence intervals for each risk. Default is FALSE.
conf.level	Probability for confidence intervals. Default is 0.95.

Value

A list with the results

---

select_plus *Column selection with optional grouping variable*

Description

This function extends `dplyr::select()` by allowing the dynamic addition of one or more grouping variables (`var_group`) to the selection.

Usage

```
select_plus(.data, ..., var_group = NULL)
```

Arguments

`.data`	A data frame.
`...`	Columns to select (as in `dplyr::select()`).
`var_group`	A character string or vector of column names to additionally include, typically one or more grouping variables. Can be `NULL`.

Details

It is especially useful when switching between an ungrouped analysis (e.g., all observations together) and a grouped analysis (e.g., stratified or including interaction terms), without rewriting code.

For instance, this allows you to write a single analysis command for both the RDD (Rapport de Démarrage des Données) and the final report, simply by changing the `.qmd` file, without modifying the core analysis code.

Value

A data frame with the selected columns, including `var_group` if specified.

Examples

```
library(dplyr)
df <- tibble(x = 1:3, y = 4:6, z = 7:9)

# Simple selection
select_plus(df, x, y)

# Selection with grouping variable
select_plus(df, x, var_group = "z")
```

`start_new_reporting` *Generate qmd, html and css files for reporting*

Description

This function creates and writes a qmd file with css and html to report statistical analysis.

Usage

```
start_new_reporting(
  folder_path,
  output_folder,
  name = "report",
  structure = "USMR",
  path_logo = NULL,
  confidential = FALSE,
  report_type = "Data review report",
  study_id = "CHUBXYYYY/NN",
  study_name = "The Study Name",
  study_abbreviation = "TSN",
  investigator = "Investigator name",
  methodologist = "Jean Dupont",
  biostatistician = "George Frais",
  datamanager = "Peter Parker",
  methodologist_mail = NULL,
  biostatistician_mail = NULL,
  datamanager_mail = NULL,
  language = "fr"
)
```

Arguments

`folder_path`	The folder where this should be created
`output_folder`	The folder where the html will be recorded.
`name`	The name of the files
`structure`	Character string indicating the organizational structure, either "USMR" or "EUCLID". Default is "USMR".
`path_logo`	Character string specifying the path to the logo image. If NULL, a default logo is used.
`confidential`	Logical value indicating whether the report should be marked as confidential. Default is FALSE.
`report_type`	Character string specifying the type of report. Default is "Data review report".
`study_id`	Character string representing the study identifier. Default is "CHUBXYYYY/NN".
`study_name`	Character string specifying the name of the study. Default is "The Study Name".
`study_abbreviation`	Character string providing the abbreviation of the study. Default is "TSN".
`investigator`	Character string representing the investigator's name. Default is "Investigator name".
`methodologist`	Character string specifying the methodologist's name. Default is "Jean Dupont".
`biostatistician`	Character string specifying the biostatistician's name. Default is "George Frais".
`datamanager`	Character string specifying the data manager's name. Default is "Peter Parker".
`methodologist_mail`	Character string specifying the methodologist's email. If NULL, it is generated automatically.
`biostatistician_mail`	Character string specifying the biostatistician's email. If NULL, it is generated automatically.
`datamanager_mail`	Character string specifying the data manager's email. If NULL, it is generated automatically.
`language`	Character string indicating the language of the report, either "fr" (French) or "en" (English). Default is "fr".

Value

None. The function writes an HTML report to the specified file path.

`write_css` *Generate a CSS File*

Description

This function creates and writes a CSS file with predefined styling for tables and text formatting.

Usage

```
write_css(path)
```

Arguments

`path`	Character string specifying the file path where the CSS file will be saved.

Value

None. The function writes a CSS file to the specified file path.

`write_datestamp_output_file` *write_datestamp_output_file*

Description

A function to write a R file and add datestamp

Usage

```
write_datestamp_output_file(output_folder, path, from_file)
```

Arguments

`output_folder`	The output folder
`path`	The path of the R script
`from_file`	The initial html file to be renamed

Value

Nothing

write_html_file *Generate an HTML Report File*

Description

This function creates and writes an HTML report file based on specified study and structure details.

Usage

```
write_html_file(
  path,
  structure = "USMR",
  path_logo = NULL,
  confidential = FALSE,
  report_type = "Data review report",
  study_id = "CHUBXYYYY/NN",
  study_name = "The Study Name",
  study_abbreviation = "TSN",
  investigator = "Investigator name",
  methodologist = "Jean Dupont",
  biostatistician = "George Frais",
  datamanager = "Peter Parker",
  methodologist_mail = NULL,
  biostatistician_mail = NULL,
  datamanager_mail = NULL,
  language = "fr"
)
```

Arguments

`path`	Character string specifying the file path where the HTML file will be saved.
`structure`	Character string indicating the organizational structure, either "USMR" or "EUCLID". Default is "USMR".
`path_logo`	Character string specifying the path to the logo image. If NULL, a default logo is used.
`confidential`	Logical value indicating whether the report should be marked as confidential. Default is FALSE.
`report_type`	Character string specifying the type of report. Default is "Data review report".
`study_id`	Character string representing the study identifier. Default is "CHUBXYYYY/NN".
`study_name`	Character string specifying the name of the study. Default is "The Study Name".
`study_abbreviation`	Character string providing the abbreviation of the study. Default is "TSN".
`investigator`	Character string representing the investigator's name. Default is "Investigator name".

`methodologist`	Character string specifying the methodologist's name. Default is "Jean Dupont".
`biostatistician`	Character string specifying the biostatistician's name. Default is "George Frais".
`datamanager`	Character string specifying the data manager's name. Default is "Peter Parker".
`methodologist_mail`	Character string specifying the methodologist's email. If NULL, it is generated automatically.
`biostatistician_mail`	Character string specifying the biostatistician's email. If NULL, it is generated automatically.
`datamanager_mail`	Character string specifying the data manager's email. If NULL, it is generated automatically.
`language`	Character string indicating the language of the report, either "fr" (French) or "en" (English). Default is "fr".

Value

None. The function writes an HTML report to the specified file path.

`write_qmd` *Write a Quarto Markdown (.qmd) file*

Description

This function generates a Quarto Markdown (.qmd) file with predefined metadata and a sample table.

Usage

```
write_qmd(path, path_html, path_css)
```

Arguments

`path`	Character string specifying the output file path for the .qmd file.
`path_html`	Character string specifying the path to an HTML file to be included before the body of the document.
`path_css`	Character string specifying the path to a CSS file for styling the document.

Details

The function creates a Quarto Markdown file with metadata fields such as title, author, date, and format settings. The HTML file specified in `path_html` is included before the body, and the CSS file specified in `path_css` is used for styling. The generated document includes an example of a table with a caption.

Value

None. The function writes a .qmd file to the specified `path`.

`write_quarto_yml` *write_quarto_yml*

Description

Write quarto extension

Usage

```
write_quarto_yml(path)
```

Arguments

`path`	The path toward quarto yaml file

Value

nothing

Index

add_by_n, 2
add_by_n_noNA, 4
add_missing_info, 5
add_pvalues, 6
ajouter_label_ndm, 6

base_table, 7

css_generator, 8
custom_format, 12
custom_headers, 13
custom_round, 14
customize_table, 9
customize_table_body, 11

desc_ei_per_grade, 14
desc_ei_per_pt, 15
desc_var, 16
df_builder_ae, 19

from_name_to_adress, 19

intermediate_header, 20

modify_table_body_func, 21

ordonner_variables_qualitatives, 22

plot_butterfly_stacked_barplot, 23
plot_dumbell, 25
plot_patient_panchart, 26
plot_volcano, 27
prepare_table, 29

reverselog_trans, 30
riskdifference, 31

select_plus, 31
start_new_reporting, 32

write_css, 34
write_datestamp_output_file, 34
write_html_file, 35
write_qmd, 36
write_quarto_yml, 37