

Package ‘saros’

November 10, 2025

Type Package

Title Semi-Automatic Reporting of Ordinary Surveys

Version 1.6.0

Maintainer Stephan Daus <stephus.daus@gmail.com>

Description Offers a systematic way for conditional reporting of figures and tables for many (and bivariate combinations of) variables, typically from survey data. Contains interactive 'ggiraph'-based (<https://CRAN.R-project.org/package=ggiraph>) plotting functions and data frame-based summary tables (bivariate significance tests, frequencies/proportions, unique open ended responses, etc) with many arguments for customization, and extensions possible. Uses a global options() system for neatly reducing redundant code. Also contains tools for immediate saving of objects and returning a hashed link to the object, useful for creating download links to high resolution images upon rendering in 'Quarto'. Suitable for highly customized reports, primarily intended for survey research.

Note Free to use for non-Norwegian institutions, otherwise see LICENSE.

License MIT + file LICENSE

URL <https://nifu-no.github.io/saros/>, <https://github.com/NIFU-NO/saros>

BugReports <https://github.com/NIFU-NO/saros/issues>

Depends R (>= 4.2.0)

Imports cli, dplyr, forcats, fs, ggiraph, ggplot2, glue, grDevices, lifecycle, mschart, officer, rlang, stringi, stats, tidyr, tidyselect, utils, vctrs

Suggests covr, haven, labelled, quarto, knitr, readr, scales, spelling, srvyr, survey, testthat (>= 3.0.0), tibble, vdiff, withr, writexl, readxl

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.3

Language en-US

VignetteBuilder quarto

Config/Needs/website rmarkdown

Config/testthat/parallel true

LazyData true

NeedsCompilation no

Author Stephan Daus [aut, cre, cph] (ORCID:

<<https://orcid.org/0000-0003-0230-6997>>),

Julia Silge [ctb] (Author of internal scale_x_reordered),

David Robinson [ctb] (Author of internal scale_x_reordered),

Nordic Institute for The Studies of Innovation, Research and Education
(NIFU) [fnd],

Kristiania University College [fnd]

Repository CRAN

Date/Publication 2025-11-10 12:00:02 UTC

Contents

crowd_plots_as_tabset	3
embed_cat_prop_plot	5
embed_cat_table	6
embed_chr_table_html	6
ex_survey	7
fig_height_h_barchart	8
fig_height_h_barchart2	11
get_data_label_opts	12
get_fig_title_suffix_from_ggplot	13
get_makeme_types	14
ggsaver	16
girafe	16
global_settings_get	18
global_settings_reset	19
global_settings_set	19
makeme	20
make_content	28
make_link	29
make_link.default	30
make_link.list	32
n_range	33
n_range2	34
tabular_read	34
tabular_write	35
txt_from_cat_mesos_plots	35

Index

38

crowd_plots_as_tabset *Convert List of Plots to Quarto Tabset*

Description

Creates a Quarto tabset from a named list of ggplot2 objects, typically generated by `makeme()` with `crowd` parameter. Each plot becomes a tab with automatic height calculation and optional download links.

Usage

```
crowd_plots_as_tabset(
  plot_list,
  plot_type = c("cat_plot_html", "int_plot_html", "auto"),
  save = FALSE,
  fig_height = NULL,
  fig_height_int_default = 6
)
```

Arguments

<code>plot_list</code>	A named list of ggplot2 objects. Names become tab labels. Typically created with <code>makeme(crowd = c("target", "others"))</code> .
<code>plot_type</code>	Character. Type of plots in the list. One of: "cat_plot_html" (default): Categorical horizontal bar charts "int_plot_html": Interval plots (violin/box plots) "auto": Auto-detect from first non-NULL plot's data structure
<code>save</code>	Logical. If TRUE (default), generates download links for plot data and images via <code>get_fig_title_suffix_from_ggplot()</code> .
<code>fig_height</code>	Numeric or NULL. Manual figure height override in inches. If NULL (default), height is calculated automatically based on <code>plot_type</code> .
<code>fig_height_int_default</code>	Numeric. Default height for interval plots when auto-calculation is not available (default: 6 inches).

Details

This function is designed to be called within a Quarto document code chunk. It generates markdown that creates a tabset where each non-NULL plot in `plot_list` appears as a separate tab.

Height Calculation:

- For "cat_plot_html": Uses `fig_height_h_barchart2()` which accounts for number of variables, categories, and label lengths
- For "int_plot_html": Uses `fig_height_int_default` (simpler plots need less sophisticated calculation)

- For "auto": Detects type by checking for .category column (categorical) vs numeric statistics columns (interval)

Requirements:

- Must be run within knitr/Quarto context
- Plots should be created with [makeme\(\)](#)
- Plot list should have meaningful names for tab labels

Value

Invisibly returns NULL. The function's purpose is its side effect of printing Quarto markdown that creates a tabset.

See Also

- [makeme\(\)](#) for creating plots with crowd parameter
- [fig_height_h_barchart2\(\)](#) for categorical plot height calculation
- [get_fig_title_suffix_from_ggplot\(\)](#) for caption generation
- [girafe\(\)](#) for interactive plot rendering

Examples

```
## Not run:
# In a Quarto document
plots <- makeme(
  data = ex_survey,
  dep = b_1:b_3,
  crowd = c("target", "others"),
  mesos_var = "f_uni",
  mesos_group = "Uni of A"
)

# Create tabset with auto-detection
crowd_plots_as_tabset(plots)

# Create tabset for interval plots
int_plots <- makeme(
  data = ex_survey,
  dep = c_1:c_2,
  indep = x1_sex,
  type = "int_plot_html",
  crowd = c("target", "others"),
  mesos_var = "f_uni",
  mesos_group = "Uni of A"
)
crowd_plots_as_tabset(int_plots, plot_type = "int_plot_html")

# Without download links
crowd_plots_as_tabset(plots, save = FALSE)
```

```
# With manual height override
crowd_plots_as_tabset(plots, fig_height = 8)

## End(Not run)
```

embed_cat_prop_plot	<i>Embed Interactive Categorical Plot (DEPRECATED!)</i>
---------------------	---

Description

This function has been deprecated. Use instead [makeme\(\)](#)

Usage

```
embed_cat_prop_plot(
  data,
  ...,
  dep = tidyselect::everything(),
  indep = NULL,
  colour_palette = NULL,
  mesos_group = NULL,
  html_interactive = TRUE,
  inverse = FALSE
)
```

Arguments

data	data.frame, tibble or potentially a srvyr-object.
...	Dynamic dots, arguments forwarded to underlying function(s).
dep	tidyselect-syntax for dependent variable(s).
indep	tidyselect-syntax for an optional independent variable.
colour_palette	Character vector. Avoid using this.
mesos_group	String
html_interactive	Flag, whether to include interactivity.
inverse	Flag, whether to flip plot or table.

embed_cat_table	<i>Embed Reactable Table (DEPRECATED!)</i>
-----------------	--

Description

This function has been deprecated. Use instead [makeme\(\)](#)

Usage

```
embed_cat_table(  
  data,  
  ...,  
  dep = tidyselect::everything(),  
  indep = NULL,  
  mesos_group = NULL  
)
```

Arguments

data	data.frame, tibble or potentially a srvyr-object.
...	Dynamic dots, arguments forwarded to underlying function(s).
dep	tidyselect-syntax for dependent variable(s).
indep	tidyselect-syntax for an optional independent variable.
mesos_group	String

embed_chr_table_html	<i>Interactive table of text data (DEPRECATED)</i>
----------------------	--

Description

This function has been deprecated. Use instead [makeme\(\)](#)

Usage

```
embed_chr_table_html(data, dep, ..., mesos_group = NULL)
```

Arguments

data	data.frame, tibble or potentially a srvyr-object.
dep	tidyselect-syntax for dependent variable(s).
...	Dynamic dots, arguments forwarded to underlying function(s).
mesos_group	String

ex_survey

*ex_survey: Mockup dataset of a survey.***Description**

A dataset containing fake respondents' answers to survey questions. The first two, `x_sex` and `x_human`, are intended to be independent variables, whereas the remaining are dependent. The underscore `_` in variable names separates item groups (prefix) from items (suffix) (i.e. `a_1-a_9` => `a` + 1-9), whereas `' - '` separates the same for labels. The latter corresponds with the default in `SurveyXact`.

Usage

ex_survey

Format

A data frame with 100 rows and 29 variables:

x1_sex Gender

x2_human Is respondent human?

x3_nationality Where is the respondent born?

a_1 Do you consent to the following? - Agreement #1

a_2 Do you consent to the following? - Agreement #2

a_3 Do you consent to the following? - Agreement #3

a_4 Do you consent to the following? - Agreement #4

a_5 Do you consent to the following? - Agreement #5

a_6 Do you consent to the following? - Agreement #6

a_7 Do you consent to the following? - Agreement #7

a_8 Do you consent to the following? - Agreement #8

a_9 Do you consent to the following? - Agreement #9

b_1 How much do you like living in - Beijing

b_2 How much do you like living in - Brussels

b_3 How much do you like living in - Budapest

c_1 How many years of experience do you have in - Company A

c_2 How many years of experience do you have in - Company B

d_1 Rate your degree of confidence doing the following - Driving

d_2 Rate your degree of confidence doing the following - Drinking

d_3 Rate your degree of confidence doing the following - Driving

d_4 Rate your degree of confidence doing the following - Dancing

e_1 How often do you do the following? - Eat

e_2 How often do you do the following? - Eavesdrop
e_3 How often do you do the following? - Exercise
e_4 How often do you do the following? - Encourage someone whom you have only recently met
 and who struggles with simple tasks that they cannot achieve by themselves
p_1 To what extent do you agree or disagree to the following policies - Red Party
p_2 To what extent do you agree or disagree to the following policies - Green Party
p_3 To what extent do you agree or disagree to the following policies - Yellow Party
p_4 To what extent do you agree or disagree to the following policies - Blue Party
f_uni Which of the following universities would you prefer to study at?
open_comments Do you have any comments to the survey?
resp_status Response status

fig_height_h_barchart *Estimate figure height for a horizontal bar chart*

Description

This function estimates the height of a figure for a horizontal bar chart based on several parameters including the number of dependent and independent variables, number of categories, maximum characters in the labels, and legend properties.

Usage

```
fig_height_h_barchart(
  n_y,
  n_cats_y,
  max_chars_labels_y = 20,
  max_chars_cats_y = 20,
  n_x = NULL,
  n_cats_x = NULL,
  max_chars_labels_x = NULL,
  max_chars_cats_x = NULL,
  freq = FALSE,
  x_axis_label_width = 20,
  strip_width = 20,
  strip_angle = 0,
  main_font_size = 7,
  legend_location = c("plot", "panel"),
  n_legend_lines = NULL,
  legend_key_chars_equivalence = 5,
  multiplier_per_horizontal_line = 1,
  multiplier_per_vertical_letter = 1,
  multiplier_per_facet = 1,
  multiplier_per_bar = 1,
```

```

multiplier_per_legend_line = 1,
multiplier_per_plot = 1,
fixed_constant = 0,
margin_in_cm = 0,
figure_width_in_cm = 14,
max = 12,
min = 2,
hide_axis_text_if_single_variable = FALSE,
add_n_to_dep_label = FALSE,
add_n_to_indep_label = FALSE,
showNA = c("ifany", "never", "always")
)

```

Arguments

n_y, n_x	Integer. Number of dependent/independent variables.
n_cats_y	Integer. Number of categories across the dependent variables.
max_chars_labels_y	Integer. Maximum number of characters across the dependent variables' labels.
max_chars_cats_y	Integer. Maximum number of characters across the dependent variables' response categories (levels).
n_cats_x	Integer or NULL. Number of categories across the independent variables.
max_chars_labels_x	Integer or NULL. Maximum number of characters across the independent variables' labels.
max_chars_cats_x	Integer or NULL. Maximum number of characters across the independent variables' response categories (levels).
freq	Logical. If TRUE, frequency plot with categories next to each other. If FALSE (default), proportion plot with stacked categories.
x_axis_label_width, strip_width	Numeric. Width allocated for x-axis labels and strip labels respectively.
strip_angle	Integer. Angle of the strip text.
main_font_size	Numeric. Font size for the main text.
legend_location	Character. Location of the legend. "plot" (default) or "panel".
n_legend_lines	Integer. Number of lines in the legend.
legend_key_chars_equivalence	Integer. Approximate number of characters the legend key equals.
multiplier_per_horizontal_line	Numeric. Multiplier per horizontal line.
multiplier_per_vertical_letter	Numeric. Multiplier per vertical letter.
multiplier_per_facet	Numeric. Multiplier per facet height.

<code>multiplier_per_bar</code>	Numeric. Multiplier per bar height (thickness).
<code>multiplier_per_legend_line</code>	Numeric. Multiplier per legend line.
<code>multiplier_per_plot</code>	Numeric. Multiplier for entire plot estimates.
<code>fixed_constant</code>	Numeric. Fixed constant to be added to the height.
<code>margin_in_cm</code>	Numeric. Margin in centimeters.
<code>figure_width_in_cm</code>	Numeric. Width of the figure in centimeters.
<code>max</code>	Numeric. Maximum height.
<code>min</code>	Numeric. Minimum height.
<code>hide_axis_text_if_single_variable</code>	Boolean. Whether the label is hidden for single dependent variable plots.
<code>add_n_to_dep_label, add_n_to_indep_label</code>	Boolean. If TRUE, will add 10 characters to the max label lengths. This is primarily useful when obtaining these settings from the global environment, avoiding the need to compute this for each figure chunk.
<code>showNA</code>	String, one of "ifany", "always" or "never". Not yet in use.

Value

Numeric value representing the estimated height of the figure.

Examples

```
fig_height_h_barchart(
  n_y = 5,
  n_cats_y = 3,
  max_chars_labels_y = 20,
  max_chars_cats_y = 8,
  n_x = 1,
  n_cats_x = 4,
  max_chars_labels_x = 12,
  freq = FALSE,
  x_axis_label_width = 20,
  strip_angle = 0,
  main_font_size = 8,
  legend_location = "panel",
  n_legend_lines = 2,
  legend_key_chars_equivalence = 5,
  multiplier_per_horizontal_line = 1,
  multiplier_per_vertical_letter = .15,
  multiplier_per_facet = .95,
  multiplier_per_legend_line = 1.5,
  figure_width_in_cm = 16
)
```

fig_height_h_barchart2

Estimate figure height for a horizontal bar chart

Description

Taking an object from `makeme()`, this function estimates the height of a figure for a horizontal bar chart.

Usage

```
fig_height_h_barchart2(
  ggobj,
  main_font_size = 7,
  strip_angle = 0,
  freq = FALSE,
  x_axis_label_width = 20,
  strip_width = 20,
  legend_location = c("plot", "panel"),
  n_legend_lines = NULL,
  showNA = c("ifany", "never", "always"),
  legend_key_chars_equivalence = 5,
  multiplier_per_horizontal_line = NULL,
  multiplier_per_vertical_letter = 1,
  multiplier_per_facet = 1,
  multiplier_per_legend_line = 1,
  fixed_constant = 0,
  figure_width_in_cm = 14,
  margin_in_cm = 0,
  max = 8,
  min = 1
)
```

Arguments

<code>ggobj</code>	ggplot2-object
<code>main_font_size</code>	Numeric. Font size for the main text.
<code>strip_angle</code>	Integer. Angle of the strip text.
<code>freq</code>	Logical. If TRUE, frequency plot with categories next to each other. If FALSE (default), proportion plot with stacked categories.
<code>x_axis_label_width, strip_width</code>	Numeric. Width allocated for x-axis labels and strip labels respectively.
<code>legend_location</code>	Character. Location of the legend. "plot" (default) or "panel".
<code>n_legend_lines</code>	Integer. Number of lines in the legend.

showNA String, one of "ifany", "always" or "never". Not yet in use.
 legend_key_chars_equivalence Integer. Approximate number of characters the legend key equals.
 multiplier_per_horizontal_line Numeric. Multiplier per horizontal line.
 multiplier_per_vertical_letter Numeric. Multiplier per vertical letter.
 multiplier_per_facet Numeric. Multiplier per facet height.
 multiplier_per_legend_line Numeric. Multiplier per legend line.
 fixed_constant Numeric. Fixed constant to be added to the height.
 figure_width_in_cm Numeric. Width of the figure in centimeters.
 margin_in_cm Numeric. Margin in centimeters.
 max Numeric. Maximum height.
 min Numeric. Minimum height.

Value

Numeric value representing the estimated height of the figure.

Examples

```
fig_height_h_barchart2(makeme(data = ex_survey, dep = b_1:b_2, indep = x1_sex))
```

get_data_label_opts *Get Valid Data Labels for Figures and Tables*

Description

Get Valid Data Labels for Figures and Tables

Usage

```
get_data_label_opts()
```

Value

Character vector

```
get_fig_title_suffix_from_ggplot
```

Generate Figure Title Suffix with N Range and Optional Download Links

Description

Creates a formatted suffix for figure titles that includes the sample size (N) range from a ggplot object. Optionally generates markdown download links for both the plot data and the plot image.

Usage

```
get_fig_title_suffix_from_ggplot(
  plot,
  save = FALSE,
  n_equals_string = "N = ",
  file_suffixes = c(".csv", ".png"),
  link_prefixes = c("[CSV](", "[PNG]("),
  save_fns = list(utils::write.csv, ggsaver::ggsaver),
  sep = ", "
)
```

Arguments

plot	A ggplot2 object, typically created by makeme() .
save	Logical flag. If TRUE, generates download links for the plot data (CSV) and plot image (PNG). If FALSE (default), only returns the N range text.
n_equals_string	String. Prefix text for the sample size display (default: "N = ").
file_suffixes	Character vector. File extensions for the saved plot images (default: ".png"). Should include the dot.
link_prefixes	Character vector. Markdown link text prefixes for the plot download links (default: "[PNG](").
save_fns	List of functions. Functions to save the plot data and images.
sep	String. Separator between N range text and download links (default: ", ").

Details

This function is particularly useful for adding informative captions to plots in reports. The N range is calculated using [n_range2\(\)](#), which extracts the sample size from the plot data. When save = TRUE, the function creates downloadable files using [make_link\(\)](#):

- Plot data as CSV (via `utils::write.csv`)
- Plot image as PNG (via [ggsaver\(\)](#))

The function returns an `AsIs` object to prevent automatic character escaping in markdown/HTML contexts.

Value

An AsIs object (using `I()`) containing a character string with:

- Sample size range formatted as "{n_equals_string}{range}"
- If `save = TRUE`: additional download links for plot data and image, separated by `sep`
- Empty string if `plot` is not a valid ggplot object or has no data

See Also

- `n_range2()` for extracting N range from ggplot objects
- `make_link()` for creating download links
- `ggsaver()` for saving ggplot objects

Examples

```
# Create a sample plot
plot <- makeme(data = ex_survey, dep = b_1:b_3)

# Get just the N range text
get_fig_title_suffix_from_ggplot(plot)

# Custom N prefix
get_fig_title_suffix_from_ggplot(plot, n_equals_string = "Sample size: ")

## Not run:
# Generate with download links (saves files to disk)
get_fig_title_suffix_from_ggplot(plot, save = TRUE)

# Custom separator and link prefix
get_fig_title_suffix_from_ggplot(
  plot,
  save = TRUE,
  sep = " | ",
  link_prefix = "[Download PNG]("
)

## End(Not run)
```

get_makeme_types

Get all registered options for the type-argument in the makeme-function

Description

The `makeme()`-function take for the argument `type` one of several strings to indicate content type and output type. This function collects all registered alternatives. Extensions are possible, see further below.

Built-in types:

Whereas the names of the types can be arbitrary, a pattern is pursued in the built-in types. Prefix indicates what dependent data type it is intended for

"cat" Categorical (ordinal and nominal) data.

"chr" Open ended responses and other character data.

"int" Integer and numeric data.

Suffix indicates output

"html" Interactive html, usually what you want for Quarto, as Quarto can usually convert to other formats when needed

"docx" However, Quarto's and Pandoc's docx-support is currently still limited, for instance as vector graphics are converted to raster graphics for docx output. Hence, saros offers some types that outputs into MS Chart vector graphics. Note that this is experimental and not actively developed.

"pdf" This is basically just a shortcut for "html" with interactive=FALSE

Usage

```
get_makeme_types()
```

Value

Character vector

Further details about some of the built-in types:

"cat_plot_" A Likert style plot for groups of categorical variables sharing the same categories.

"cat_table_" A Likert style table.

"chr_table_" A single-column table listing unique open ended responses.

"sigtest_table_" See below

sigtest_table_*: Make Table with All Combinations of Univariate/Bivariate Significance Tests Based on Variable Types

Although there are hundreds of significance tests for associations between two variables, depending upon the distributions, variables types and assumptions, most fall into a smaller set of popular tests. This function runs for all combinations of dependent and independent variables in data, with a suitable test (but not the only possible) for the combination. Also supports univariate tests, where the assumptions are that of a mean of zero for continuous variables or all equal proportions for binary/categorical.

This function does not allow any adjustments - use the original underlying functions for that (chisq.test, t.test, etc.)

Expanding with custom types

makeme() calls the generic **make_content()**, which uses the S3-method system to dispatch to the relevant method (i.e., `paste0("make_content.", type)`). **makeme** forwards all its arguments to **make_content**, with the following exceptions:

1. **dep** and **indep** are converted from **dplyr::dplyr_tidy_select()**-syntax to simple character vectors, for simplifying building your own functions.
2. **data_summary** is attached, which contains many useful pieces of info for many (categorical) displays.

Examples

```
get_makeme_types()
```

```
ggsaver
```

Wrapper Function for `ggplot2::ggsave()`

Description

This only exists to make it easy to use it in `make_link()`

Usage

```
ggsaver(plot, filename, ...)
```

Arguments

<code>plot</code>	Plot
<code>filename</code>	Note
<code>...</code>	Arguments forwarded to <code>ggplot2::ggsave()</code>

Value

No return value, called for side effects

Examples

```
library(ggplot2)
my_plot <- ggplot(data=mtcars, aes(x=hp, y=mpg)) + geom_point()
make_link(my_plot, folder=tempdir(), file_suffix = ".png",
          save_fn = ggsaver, width = 16, height = 16, units = "cm")
```

```
girafe
```

Pull global plotting settings before displaying plot

Description

This function extends `ggiraph::girafe` by allowing colour palettes to be globally specified.

Usage

```

girafe(
  ggobj,
  ...,
  char_limit = 200,
  label_wrap_width = 80,
  interactive = TRUE,
  palette_codes = NULL,
  priority_palette_codes = NULL,
  ncol = NULL,
  byrow = TRUE,
  colour_2nd_binary_cat = NULL,
  checked = NULL,
  not_checked = NULL,
  width_svg = NULL,
  height_svg = NULL,
  pointsize = 12
)

```

Arguments

<code>ggobj</code>	ggplot2-object.
<code>...</code>	Dots forwarded to <code>ggiraph::girafe()</code>
<code>char_limit</code>	Integer. Number of characters to fit on a line of plot (legend-space). Will be replaced in the future with a function that guesses this.
<code>label_wrap_width</code>	Integer. Number of characters fit on the axis text space before wrapping.
<code>interactive</code>	Boolean. Whether to produce a ggiraph-plot with interactivity (defaults to TRUE) or a static ggplot2-plot.
<code>palette_codes</code>	Optional list of named character vectors with names being categories and values being colours. The final character vector of the list is taken as a final resort. Defaults to NULL.
<code>priority_palette_codes</code>	Optional named character of categories (as names) with corresponding colours (as values) which are used first, whereupon the remaining unspecified categories are pulled from the last vector of <code>palette_codes</code> . Defaults to NULL.
<code>ncol</code>	Optional integer or NULL.
<code>byrow</code>	Whether to display legend keys by row or by column.
<code>colour_2nd_binary_cat</code>	Optional string. Color for the second category in binary checkbox plots. When set together with <code>checked</code> and <code>not_checked</code> , reverses the category order so that <code>not_checked</code> appears second and receives this color. Ignored if checkbox criteria are not met.
<code>checked, not_checked</code>	Optional string. If specified and the fill categories of the plot matches these, a special plot is returned where <code>not_checked</code> is hidden. Its usefulness comes in

plots which are intended for checkbox responses where unchecked is not always a conscious choice.

pointsize, height_svg, width_svg

See `ggiraph::girafe()`.

Value

If interactive, only side-effect of generating ggiraph-plot. If interactive=FALSE, returns modified ggobj.

Examples

```
plot <- makeme(data = ex_survey, dep = b_1)
girafe(plot)
```

global_settings_get	<i>Get Global Options for saros-functions</i>
---------------------	---

Description

Get Global Options for saros-functions

Usage

```
global_settings_get(fn_name = "makeme")
```

Arguments

fn_name String, one of "make_link", "fig_height_h_barchart" and "makeme".

Value

List with options in R

Examples

```
global_settings_get()
```

global_settings_reset *Reset Global Options for saros-functions*

Description

Reset Global Options for saros-functions

Usage

```
global_settings_reset(fn_name = "makeme")
```

Arguments

fn_name String, one of "make_link", "fig_height_h_barchart" and "makeme".

Value

Invisibly returned list of old and new values.

Examples

```
global_settings_reset()
```

global_settings_set *Get Global Options for saros-functions*

Description

Get Global Options for saros-functions

Usage

```
global_settings_set(
  new,
  fn_name = "makeme",
  quiet = FALSE,
  null_deletes = FALSE
)
```

Arguments

new	List of arguments (see ?make_link(), ?makeme(), fig_height_h_barchart())
fn_name	String, one of "make_link", "fig_height_h_barchart" and "makeme".
quiet	Flag. If FALSE (default), informs about what has been set.
null_deletes	Flag. If FALSE (default), NULL elements in new become NULL elements in the option. Otherwise, the corresponding element, if present, is deleted from the option.

Value

Invisibly returned list of old and new values.

Examples

```
global_settings_set(new=list(digits=2))
```

makeme

Embed Interactive Plot of Various Kinds Using Tidysselect Syntax

Description

This function allows embedding of interactive or static plots based on various types of data using tidysselect syntax for variable selection.

Usage

```
makeme(
  data,
  dep = tidysselect::everything(),
  indep = NULL,
  type = c("cat_plot_html", "int_plot_html", "cat_table_html", "int_table_html",
    "sigtest_table_html", "cat_prop_plot_docx", "cat_freq_plot_docx", "int_plot_docx"),
  ...,
  require_common_categories = TRUE,
  crowd = c("all"),
  mesos_var = NULL,
  mesos_group = NULL,
  simplify_output = TRUE,
  hide_for_crowd_if_all_na = TRUE,
  hide_for_crowd_if_valid_n_below = 0,
  hide_for_crowd_if_category_k_below = 2,
  hide_for_crowd_if_category_n_below = 0,
  hide_for_crowd_if_cell_n_below = 0,
  hide_for_all_crowds_if_hidden_for_crowd = NULL,
  hide_indep_cat_for_all_crowds_if_hidden_for_crowd = FALSE,
  add_n_to_dep_label = FALSE,
  add_n_to_indep_label = FALSE,
  add_n_to_label = FALSE,
  add_n_to_category = FALSE,
  totals = FALSE,
  categories_treated_as_na = NULL,
  label_separator = " - ",
  error_on_duplicates = TRUE,
  showNA = c("ifany", "always", "never"),
  data_label = c("percentage_bare", "percentage", "proportion", "count", "mean",
    "median"),
```

```

data_label_position = c("center", "bottom", "top", "above"),
html_interactive = TRUE,
hide_axis_text_if_single_variable = TRUE,
hide_label_if_prop_below = 0.01,
inverse = FALSE,
vertical = FALSE,
digits = 0,
data_label_decimal_symbol = ".",
x_axis_label_width = 25,
strip_width = 25,
sort_dep_by = ".variable_position",
sort_indep_by = ".factor_order",
sort_by = NULL,
descend = TRUE,
descend_indep = FALSE,
labels_always_at_top = NULL,
labels_always_at_bottom = NULL,
table_wide = TRUE,
table_main_question_as_header = FALSE,
n_categories_limit = 12,
translations = list(last_sep = " and ", table_heading_N = "Total (N)",
  table_heading_data_label = "%", add_n_to_dep_label_prefix = " (N = ",
  add_n_to_dep_label_suffix = ")", add_n_to_indep_label_prefix = " (N = ",
  add_n_to_indep_label_suffix = ")", add_n_to_label_prefix = " (N = ",
  add_n_to_label_suffix = ")", add_n_to_category_prefix = " (N = [",
  add_n_to_category_infix = ",", add_n_to_category_suffix = "])", by_total =
  "Everyone", sigtest_variable_header_1 = "Var 1", sigtest_variable_header_2 = "Var 2",
  crowd_all = "All",
  crowd_target = "Target", crowd_others = "Others"),
plot_height = 15,
colour_palette = NULL,
colour_2nd_binary_cat = "#ffffff",
colour_na = "grey",
label_font_size = 6,
main_font_size = 6,
strip_font_size = 6,
legend_font_size = 6,
font_family = "sans",
path = NULL,
docx_template = NULL
)

```

Arguments

data	<i>Your data.frame/tibble or srvyr-object (experimental)</i> data.frame // required The data to be used for plotting.
dep, indep	<i>Variable selections</i>

```

<tidyselect> // Default: NULL, meaning everything for dep, nothing for indep.
Columns in data. dep is compulsory.

type          Kind of output
              scalar<character> // default: "cat_plot_html" (optional)
              For a list of registered types in your session, use get_makeme_types().

...          Dynamic dots
              <dynamic-dots>
              Arguments forwarded to the corresponding functions that create the elements.

require_common_categories
              Check common categories
              scalar<logical> // default: TRUE (optional)
              Whether to check if all items share common categories.

crowd         Which group(s) to display results for
              vector<character> // default: c("target", "others", "all") (optional)
              Choose whether to produce results for target (mesos) group, others, all, or combinations of these.

mesos_var     Variable in data indicating groups to tailor reports for
              scalar<character> // default: NULL (optional)
              Column name in data indicating the groups for which mesos reports will be produced.

mesos_group   scalar<character> // default: NULL (optional)
              String, target group.

simplify_output
              scalar<logical> // default: TRUE
              If TRUE, a list output with a single output element will return the element itself,
              whereas list with multiple elements will return the list.

hide_for_crowd_if_all_na
              Hide variable from output if containing all NA
              scalar<boolean> // default: TRUE
              Whether to remove all variables (in particular useful for mesos) if all values are NA.

hide_for_crowd_if_valid_n_below
              Hide variable if variable has < n observations
              scalar<integer> // default: 0
              Whether to hide a variable for a crowd if variable contains fewer than n observations (always ignoring NA).

hide_for_crowd_if_category_k_below
              Hide variable if < k categories
              scalar<integer> // default: 2
              Whether to hide a variable for a crowd if variable contains fewer than k used categories (always ignoring NA). Defaults to 2 because a unitary plot/table is rarely informative.

hide_for_crowd_if_category_n_below
              Hide variable if having a category with < n observations

```

scalar<integer> // default: 0

Whether to hide a variable for a crowd if variable contains a category with less than n observations (ignoring NA). Cells with a 0 count is not considered as these are usually not a problem for anonymity.

hide_for_crowd_if_cell_n_below

Hide variable if having a cell with < n

scalar<integer> // default: 0

Whether to hide a variable for a crowd if the combination of dep-indep results in a cell with less than n observations (ignoring NA). Cells with a 0 count is not considered as these are usually not a problem for anonymity.

hide_for_all_crowds_if_hidden_for_crowd

Conditional hiding

scalar<character> // default: NULL (optional)

Select one of the crowd output groups. If selected, will hide a variable across all crowd-outputs if it for some reason is not displayed for hide_for_all_if_hidden_for_crowd. For instance, say:

```
crowd = c("target", "others"), hide_variable_if_all_na = TRUE,
hide_for_all_if_hidden_for_crowd = "target"
```

will hide variables from both target and others-outputs if all are NA in the target-group.

hide_indep_cat_for_all_crowds_if_hidden_for_crowd

Conditionally hide independent categories

scalar<logical> // default: FALSE

If hide_for_all_crowds_if_hidden_for_crowd is specified, should categories of the indep variable(s) be hidden for a crowd if it does not exist for the crowds specified in hide_for_all_crowds_if_hidden_for_crowd? This is useful when e.g. indep is academic disciplines, mesos_var is institutions, and a specific institution is not interested in seeing academic disciplines they do not offer themselves.

add_n_to_dep_label, add_n_to_indep_label

Add N= to the variable label

scalar<logical> // default: FALSE (optional)

For some plots and tables it is useful to attach the "N=" to the end of the label of the dependent and/or independent variable. Whether it is N or N_valid depends on your showNA-setting. See also translations\$add_n_to_dep_label_prefix, translations\$add_n_to_dep_label_suffix, translations\$add_n_to_indep_label_prefix, translations\$add_n_to_indep_label_suffix.

add_n_to_label *Add N= to the variable label of both dep and indep*

scalar<logical> // default: FALSE (optional)

For some plots and tables it is useful to attach the "N=" to the end of the label. Whether it is N or N_valid depends on your showNA-setting. See also translations\$add_n_to_label_prefix and translations\$add_n_to_label_suffix.

add_n_to_category

Add N= to the category

scalar<logical> // default: FALSE (optional)

For some plots and tables it is useful to attach the "N=" to the end of the category. This will likely produce a range across the variables, hence an infix (comma) between the minimum and maximum can be specified. Whether it is N or N_valid depends on your showNA-setting. See also `translations$add_n_to_category_prefix`, `translations$add_n_to_category_infix`, and `translations$add_n_to_category_suffix`.

<code>totals</code>	<i>Include totals</i> scalar<logical> // <i>default</i>: FALSE (optional) Whether to include totals in the output.
<code>categories_treated_as_na</code>	<i>NA categories</i> vector<character> // <i>default</i>: NULL (optional) Categories that should be treated as NA.
<code>label_separator</code>	<i>How to separate main question from sub-question</i> scalar<character> // <i>default</i>: NULL (optional) Separator for main question from sub-question.
<code>error_on_duplicates</code>	<i>Error or warn on duplicate labels</i> scalar<logical> // <i>default</i>: TRUE (optional) Whether to abort (TRUE) or warn (FALSE) if the same label (suffix) is used across multiple variables.
<code>showNA</code>	<i>Show NA categories</i> vector<character> // <i>default</i>: c("ifany", "always", "never") (optional) Choose whether to show NA categories in the results.
<code>data_label</code>	<i>Data label</i> scalar<character> // <i>default</i>: "proportion" (optional) One of "proportion", "percentage", "percentage_bare", "count", "mean", or "median".
<code>data_label_position</code>	<i>Data label position</i> scalar<character> // <i>default</i>: "center" (optional) Position of data labels on bars. One of "center" (middle of bar), "bottom" (bottom but inside bar), "top" (top but inside bar), or "above" (above bar outside).
<code>html_interactive</code>	<i>Toggle interactive plot</i> scalar<logical> // <i>default</i>: TRUE (optional) Whether the plot is to be interactive (ggiraph) or static (ggplot2).
<code>hide_axis_text_if_single_variable</code>	<i>Hide y-axis text if just a single variable</i> scalar<boolean> // <i>default</i>: FALSE (optional) Whether to hide text on the y-axis label if just a single variable.
<code>hide_label_if_prop_below</code>	<i>Hide label threshold</i> scalar<numeric> // <i>default</i>: NULL (optional) Whether to hide label if below this value.

inverse	<i>Flag to swap x-axis and faceting</i> scalar<logical> // default: FALSE (optional) If TRUE, swaps x-axis and faceting.
vertical	<i>Display plot vertically</i> scalar<logical> // default: FALSE (optional) If TRUE, display plot vertically.
digits	<i>Decimal places</i> scalar<integer> // default: 0L (optional) Number of decimal places.
data_label_decimal_symbol	<i>Decimal symbol</i> scalar<character> // default: "." (optional) Decimal marker, some might prefer a comma ',' or something else entirely.
x_axis_label_width, strip_width	<i>Label width of x-axis and strip texts in plots</i> scalar<integer> // default: 20 (optional) Width of the labels used for the categorical column names in x-axis texts and strip texts.
sort_dep_by	<i>What to sort dependent variables by</i> vector<character> // default: ".variable_position" (optional) Sort dependent variables in output. When using indep-argument, sorting differs between ordered factors and unordered factors: Ordering of ordered factors is always respected in output (their levels define the base order). Unordered factors will be reordered by sort_dep_by. NULL or ".variable_position" Sort by variable position in the supplied data frame (default). ".variable_label" Sort by the variable labels. ".variable_name" Sort by the variable names. ".top" The proportion for the highest category available in the variable. ".upper" The sum of the proportions for the categories above the middle category. ".mid_upper" The sum of the proportions for the categories including and above the middle category. ".mid_lower" The sum of the proportions for the categories including and below the middle category. ".lower" The sum of the proportions for the categories below the middle category. ".bottom" The proportions for the lowest category available in the variable.
sort_indep_by	<i>What to sort independent variable categories by</i> vector<character> // default: ".factor_order" (optional) Sort independent variable categories in output. When ".factor_order", preserves the original factor level order for the independent variable. Passing NULL is accepted and treated as ".factor_order". NULL No sorting - preserves original factor level order (default).

".top" The proportion for the highest category available.
".upper" The sum of the proportions for the categories above the middle category.
".mid_upper" The sum of the proportions for the categories including and above the middle category.
".mid_lower" The sum of the proportions for the categories including and below the middle category.
".lower" The sum of the proportions for the categories below the middle category.
".bottom" The proportions for the lowest category available.
character() Character vector of category labels to sum together.
What to sort output by (legacy)
vector<character> // *default:* NULL (optional)
DEPRECATED: Use `sort_dep_by` and `sort_indep_by` instead for clearer control. When specified, this parameter will be used for both dependent and independent sorting. If NULL (default), dependent variables will be sorted by `.variable_position`.
NULL Uses `.variable_position` for dependent variables, no sorting for independent.
".top" The proportion for the highest category available in the variable.
".upper" The sum of the proportions for the categories above the middle category.
".mid_upper" The sum of the proportions for the categories including and above the middle category.
".mid_lower" The sum of the proportions for the categories including and below the middle category.
".lower" The sum of the proportions for the categories below the middle category.
".bottom" The proportions for the lowest category available in the variable.
".variable_label" Sort by the variable labels.
".variable_name" Sort by the variable names.
".variable_position" Sort by the variable position in the supplied data frame.
".by_group" The groups of the `by` argument.
character() Character vector of category labels to sum together.
descend *Sorting order*
scalar<logical> // *default:* FALSE (optional)
 Reverse sorting of `sort_by` in figures and tables. Works with both ordered and unordered factors - for ordered factors, it reverses the display order while preserving the inherent level ordering. See `arrange_section_by` for sorting of report sections.
descend_indep *Sorting order for independent variables*
scalar<logical> // *default:* FALSE (optional)
 Reverse sorting of `sort_indep_by` in figures and tables. Works with both ordered and unordered factors - for ordered factors, it reverses the display order while preserving the inherent level ordering. See `arrange_section_by` for sorting of report sections.

labels_always_at_top, labels_always_at_bottom
Top/bottom variables
 vector<character> // default: NULL (optional)
 Column names in data that should always be placed at the top or bottom of figures/tables.

table_wide *Pivot table wider*
 scalar<logical> // default: FALSE (optional)
 Whether to pivot table wider.

table_main_question_as_header
Table main question as header
 scalar<logical> // default: FALSE (optional)
 Whether to include the main question as a header in the table.

n_categories_limit
Limit for cat_table_wide format
 scalar<integer> // default: 12 (optional)
 If there are more than this number of categories in the categorical variable, cat_table_* will have a long format instead of wide format.

translations *Localize your output*
 list<character>
 A list of translations where the name is the code and the value is the translation. See the examples.

plot_height *DOCX-setting*
 scalar<numeric> // default: 12 (optional)
 DOCX plots need a height, which currently cannot be set easily with a Quarto chunk option.

colour_palette *Colour palette*
 vector<character> // default: NULL (optional)
 Must contain at least the number of unique values (including missing) in the data set.

colour_2nd_binary_cat
Colour for second binary category
 scalar<character> // default: "#ffffff" (optional)
 Colour for the second category in binary variables. Often useful to hide this.

colour_na *Colour for NA category*
 scalar<character> // default: NULL (optional)
 Colour as a single string for NA values, if showNA is "ifany" or "always".

main_font_size, label_font_size, strip_font_size, legend_font_size
Font sizes
 scalar<integer> // default: 6 (optional)
 ONLY FOR DOCX-OUTPUT. Other output is adjusted using e.g. ggplot2::theme() or set with a global theme (ggplot2::set_theme()). Font sizes for general text (6), data label text (3), strip text (6) and legend text (6).

font_family *Font family*
 scalar<character> // default: "sans" (optional)
 Word font family. See officer::fp_text.

path	<i>Output path for DOCX</i> scalar<character> // default: NULL (optional) Path to save docx-output.
docx_template	<i>Filename or rdocx object</i> scalar<character> <rdocx>-object // default: NULL (optional) Can be either a valid character path to a reference Word file, or an existing rdocx-object in memory.

Value

ggplot-object, optionally an extended ggplot object with ggiraph features.

Examples

```

makeme(
  data = ex_survey,
  dep = b_1:b_2
)
makeme(
  data = ex_survey,
  dep = b_1:b_3, indep = c(x1_sex, x2_human),
  type = "sigtest_table_html"
)
makeme(
  data = ex_survey,
  dep = p_1:p_4, indep = x2_human,
  type = "cat_table_html"
)
makeme(
  data = ex_survey,
  dep = c_1:c_2, indep = x1_sex,
  type = "int_table_html"
)
makeme(
  data = ex_survey,
  dep = b_1:b_2,
  crowd = c("target", "others"),
  mesos_var = "f_uni",
  mesos_group = "Uni of A"
)

```

make_content

Method for Creating Saros Contents

Description

Takes the same arguments as makeme, except that dep and indep in make_content are character vectors, for ease of user-customized function programming.

Usage

```
make_content(type, ...)
```

Arguments

<code>type</code>	<i>Method name</i> scalar<character> with a class named by itself. Optional string indicating the specific method. Occasionally useful for error messages, etc.
<code>...</code>	<i>Dots</i> Arguments provided by makeme

Value

The returned object class depends on the type. `type="*_table_html"` always returns a tibble. `type="*_plot_html"` always returns a ggplot. `type="*_docx"` always returns a rdocx object if `path=NULL`, or has side-effect of writing docx file to disk if path is set.

make_link	Save data to a file and return a Markdown link
-----------	--

Description

The file is automatically named by a hash of the object, removing the need to come up with unique file names inside a Quarto report. This has the added benefit of reducing storage needs if the objects needing linking to are identical, and all are stored in the same folder. It also allows the user to download multiple files without worrying about accidentally overwriting them.

Usage

```
make_link(
  data,
  folder = NULL,
  file_prefix = NULL,
  file_suffix = ".csv",
  save_fn = utils::write.csv,
  link_prefix = "[download figure data](",
  link_suffix = ")",
  ...
)
```

Arguments

data	<i>Data or object</i> <data.frame tbl obj> Data frame if using a tabular data save_fn, or possibly any R object, if a serializing save_fn is provided (e.g. saveRDS()).
folder	<i>Where to store file</i> scalar<character> // default: "." (optional) Defaults to same folder.
file_prefix, file_suffix	<i>File prefix/suffix</i> scalar<character> // default: "" and ".csv" (optional) file_suffix should include the dot before the extension.
save_fn	<i>Saving function</i> function // default: utils::write.csv Can be any saving/writing function. However, first argument must be the object to be saved, and the second must be the path. Hence, ggplot2::ggsave() must be wrapped in another function with filename and object swapped. See ggsaver() for an example of such a wrapper function.
link_prefix, link_suffix	<i>Link prefix/suffix</i> scalar<character> // default: "[download data](" and ")" The stuff that is returned.
...	<i>Dynamic dots</i> <dynamic-dots> Arguments forwarded to the corresponding functions that create the elements.

Value

String.

Examples

```
make_link(mtcars, folder = tempdir())
```

make_link.default	Save data to a file and return a Markdown link
-------------------	--

Description

The file is automatically named by a hash of the object, removing the need to come up with unique file names inside a Quarto report. This has the added benefit of reducing storage needs if the objects needing linking to are identical, and all are stored in the same folder. It also allows the user to download multiple files without worrying about accidentally overwriting them.

Usage

```
## Default S3 method:
make_link(
  data,
  ...,
  folder = NULL,
  file_prefix = NULL,
  file_suffix = ".csv",
  save_fn = utils::write.csv,
  link_prefix = "[download figure data](",
  link_suffix = ")"
)
```

Arguments

data	<i>Data or object</i> <code><data.frame tbl obj></code> Data frame if using a tabular data save_fn, or possibly any R object, if a serializing save_fn is provided (e.g. saveRDS()).
...	<i>Dynamic dots</i> <code><dynamic-dots></code> Arguments forwarded to the corresponding functions that create the elements.
folder	<i>Where to store file</i> <code>scalar<character> // default: "."</code> (optional) Defaults to same folder.
file_prefix, file_suffix	<i>File prefix/suffix</i> <code>scalar<character> // default: ""</code> and <code>".csv"</code> (optional) file_suffix should include the dot before the extension.
save_fn	<i>Saving function</i> <code>function // default: utils::write.csv</code> Can be any saving/writing function. However, first argument must be the object to be saved, and the second must be the path. Hence, ggplot2::ggsave() must be wrapped in another function with filename and object swapped. See ggsaver() for an example of such a wrapper function.
link_prefix, link_suffix	<i>Link prefix/suffix</i> <code>scalar<character> // default: "[download data]("</code> and <code>)"</code> The stuff that is returned.

Value

String.

Examples

```
make_link(mtcars, folder = tempdir())
```

make_link.list

Save data to a file and return a Markdown link

Description

The file is automatically named by a hash of the object, removing the need to come up with unique file names inside a Quarto report. This has the added benefit of reducing storage needs if the objects needing linking to are identical, and all are stored in the same folder. It also allows the user to download multiple files without worrying about accidentally overwriting them.

Usage

```
## S3 method for class 'list'
make_link(
  data,
  ...,
  folder = NULL,
  file_prefix = NULL,
  file_suffix = ".csv",
  save_fn = utils::write.csv,
  link_prefix = "[download figure data](",
  link_suffix = ")",
  separator_list_items = ". "
)
```

Arguments

data	<i>Data or object</i> <data.frame tbl obj> Data frame if using a tabular data save_fn, or possibly any R object, if a serializing save_fn is provided (e.g. saveRDS()).
...	<i>Dynamic dots</i> <dynamic-dots> Arguments forwarded to the corresponding functions that create the elements.
folder	<i>Where to store file</i> scalar<character> // default: "." (optional) Defaults to same folder.
file_prefix, file_suffix	<i>File prefix/suffix</i> scalar<character> // default: "" and ".csv" (optional) file_suffix should include the dot before the extension.
save_fn	<i>Saving function</i> function // default: utils::write.csv Can be any saving/writing function. However, first argument must be the object to be saved, and the second must be the path. Hence, ggplot2::ggsave()

must be wrapped in another function with filename and object swapped. See `ggsaver()` for an example of such a wrapper function.

`link_prefix, link_suffix`
Link prefix/suffix
 scalar<character> // default: "[download data](" and ")"

`separator_list_items`
Separator string between multiple list items
 scalar<character> // default: ". " (optional)

n_range	<i>Provides a range (or single value) for N in data, given dep and indep</i>
---------	--

Description

Provides a range (or single value) for N in data, given dep and indep

Usage

```
n_range(
  data,
  dep,
  indep = NULL,
  mesos_var = NULL,
  mesos_group = NULL,
  glue_template_1 = "{n}",
  glue_template_2 = "[{n[1]}-{n[2]}]"
)
```

Arguments

<code>data</code>	Dataset
<code>dep, indep</code>	Tidyselect syntax
<code>mesos_var</code>	Optional, NULL or string specifying name of variable used to split dataset.
<code>mesos_group</code>	Optional, NULL or string specifying value in <code>mesos_var</code> indicating the target group.
<code>glue_template_1, glue_template_2</code>	String, for the case of a single value (1) or a range with minimum-maximum of values (2).

Value

String.

Examples

```
n_range(data = ex_survey, dep = b_1:b_3, indep = x1_sex)
```

n_range2	<i>Provides a range (or single value) for N in a ggplot2-object from makeme()</i>
----------	---

Description

Provides a range (or single value) for N in a ggplot2-object from makeme()

Usage

```
n_range2(ggobj, glue_template_1 = "{n}", glue_template_2 = "[{n[1]}-{n[2]}]")
```

Arguments

ggobj	A ggplot2-object.
glue_template_1, glue_template_2	String, for the case of a single value (1) or a range with minimum-maximum of values (2).

Value

String.

Examples

```
n_range2(makeme(data = ex_survey, dep = b_1:b_3))
```

tabular_read	<i>Read tabular data from various formats</i>
--------------	---

Description

A wrapper function to read data from different file formats

Usage

```
tabular_read(path, format, ...)
```

Arguments

path	Character string specifying the file path
format	Character string specifying the format: "delim", "xlsx", "csv", "csv2", "tsv", "sav", "dta"
...	Additional arguments passed to the underlying read functions

Value

A data frame containing the loaded data

tabular_write	<i>Write tabular data to various formats</i>
---------------	--

Description

A wrapper function to write data frames to different file formats

Usage

```
tabular_write(object, path, format)
```

Arguments

object	A data frame to write
path	Character string specifying the output file path
format	Character string specifying the format: "delim", "xlsx", "csv", "csv2", "tsv", "sav", "dta"

Value

Invisibly returns TRUE on success, used for side effects

Examples

```
data <- data.frame(x = 1:3, y = letters[1:3])

# Write as CSV
tabular_write(data, tempfile(fileext = ".csv"), format = "csv")

# Write as Excel
tabular_write(data, tempfile(fileext = ".xlsx"), format = "xlsx")

# Write as SPSS
tabular_write(data, tempfile(fileext = ".sav"), format = "sav")
```

txt_from_cat_mesos_plots	<i>Extract Text Summary from Categorical Mesos Plots</i>
--------------------------	--

Description

Generates text summaries comparing two groups from categorical mesos plot data. The function identifies meaningful differences between groups based on proportions of respondents selecting specific categories and produces narrative text descriptions.

Usage

```
txt_from_cat_mesos_plots(
  plots,
  min_prop_diff = 0.1,
  n_highest_categories = 1,
  flip_to_lowest_categories = FALSE,
  digits = 2,
  selected_categories_last_split = " or ",
  fallback_string = character(),
  glue_str_pos =
    c(paste0("For {var}, the target group has a higher proportion of respondents ",
      "({group_1}) than all others ({group_2}) who answered {selected_categories}. "),
      paste0("More respondents answered {selected_categories} for {var} in the ",
        "target group ({group_1}) than in other groups ({group_2}). "),
      paste0("The statement {var} shows {selected_categories} responses are more ",
        "common in the target group ({group_1}) compared to others ({group_2}). ")),
  glue_str_neg =
    c(paste0("For {var}, the target group has a lower proportion of respondents ",
      "({group_1}) than all others ({group_2}) who answered {selected_categories}. "),
      paste0("Fewer respondents answered {selected_categories} for {var} in the ",
        "target group ({group_1}) than in other groups ({group_2}). "),
      paste0("The statement {var} shows {selected_categories} responses are less ",
        "common in the target group ({group_1}) compared to others ({group_2}). "))
)
```

Arguments

<code>plots</code>	A list of two plot objects (or data frames with plot data) to compare. Each must contain columns: <code>.variable_label</code> , <code>.category</code> , <code>.category_order</code> , <code>.proportion</code> .
<code>min_prop_diff</code>	Numeric. Minimum proportion difference (default 0.10) required between groups to generate text. Differences below this threshold are ignored.
<code>n_highest_categories</code>	Integer. Number of top categories to include in the comparison (default 1). Categories are selected based on <code>.category_order</code> .
<code>flip_to_lowest_categories</code>	Logical. If TRUE, compare lowest categories instead of highest (default FALSE).
<code>digits</code>	Integer. Number of decimal places for rounding proportions (default 2).
<code>selected_categories_last_split</code>	Character. Separator for the last item when listing multiple categories (default " or ").
<code>fallback_string</code>	Character. String to return when validation fails (default <code>character()</code>).
<code>glue_str_pos</code>	Character vector. Templates for positive differences (<code>group_1 > group_2</code>). Available placeholders: <code>{var}</code> , <code>{group_1}</code> , <code>{group_2}</code> , <code>{selected_categories}</code> .
<code>glue_str_neg</code>	Character vector. Templates for negative differences (<code>group_2 > group_1</code>). Same placeholders as <code>glue_str_pos</code> .

Details

The function compares proportions between two groups for each variable in the plot data. One template is randomly selected from the provided vectors for variety in output text.

Value

A character vector of text summaries, one per variable with meaningful differences. Returns empty character vector if no plots provided or no meaningful differences found.

Examples

```
## Not run:
# Create sample plot data
plot_data_1 <- data.frame(
  .variable_label = rep("Job satisfaction", 3),
  .category = factor(c("Low", "Medium", "High"), levels = c("Low", "Medium", "High")),
  .category_order = 1:3,
  .proportion = c(0.2, 0.3, 0.5)
)

plot_data_2 <- data.frame(
  .variable_label = rep("Job satisfaction", 3),
  .category = factor(c("Low", "Medium", "High"), levels = c("Low", "Medium", "High")),
  .category_order = 1:3,
  .proportion = c(0.3, 0.4, 0.3)
)

plots <- list(
  list(data = plot_data_1),
  list(data = plot_data_2)
)

# Generate text summaries
txt_from_cat_mesos_plots(plots, min_prop_diff = 0.10)

# Compare lowest categories instead
txt_from_cat_mesos_plots(
  plots,
  flip_to_lowest_categories = TRUE,
  min_prop_diff = 0.05
)

## End(Not run)
```

Index

- * **datasets**
 - ex_survey, [7](#)
- crowd_plots_as_tabset, [3](#)
- dplyr::dplyr_tidy_select(), [15](#)
- embed_cat_prop_plot, [5](#)
- embed_cat_table, [6](#)
- embed_chr_table_html, [6](#)
- ex_survey, [7](#)
- fig_height_h_barchart, [8](#)
- fig_height_h_barchart2, [11](#)
- fig_height_h_barchart2(), [3](#), [4](#)
- get_data_label_opts, [12](#)
- get_fig_title_suffix_from_ggplot, [13](#)
- get_fig_title_suffix_from_ggplot(), [3](#), [4](#)
- get_makeme_types, [14](#)
- ggiraph::girafe, [16](#)
- ggiraph::girafe(), [17](#), [18](#)
- ggplot2::ggsave(), [16](#), [30–32](#)
- ggsaver, [16](#)
- ggsaver(), [13](#), [14](#), [30](#), [31](#), [33](#)
- girafe, [16](#)
- girafe(), [4](#)
- global_settings_get, [18](#)
- global_settings_reset, [19](#)
- global_settings_set, [19](#)
- I(), [14](#)
- make_content, [28](#)
- make_content(), [15](#)
- make_link, [29](#)
- make_link(), [13](#), [14](#), [16](#)
- make_link.default, [30](#)
- make_link.list, [32](#)
- makeme, [20](#)
- makeme(), [3–6](#), [13–15](#)
- n_range, [33](#)
- n_range2, [34](#)
- n_range2(), [13](#), [14](#)
- saveRDS(), [30–32](#)
- tabular_read, [34](#)
- tabular_write, [35](#)
- txt_from_cat_mesos_plots, [35](#)