Package 'saros'

May 9, 2026

Type Package

Title Semi-Automatic Reporting of Ordinary Surveys

Version 1.6.2

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, mschart, officer, rlang, stringi, stats, tidyr, tidyselect, utils, vctrs

Suggests covr, haven, labelled, pdftools, quarto, knitr, readr, scales, spelling, srvyr, survey, testthat (>= 3.0.0), tibble, vdiffr, 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 2026-04-29 23:40:02 UTC

Contents

`check_quarto_website_index` *Check Quarto Website Folders for Missing index.qmd*

Description

Scans a Quarto website project directory for folders that contain `.qmd` files (directly or in subfolders) but are missing an `index.qmd` file. Such folders often cause a malfunctioning navigation menu in the rendered Quarto website.

Folders whose names start with `_` or `.` are excluded, as these are typically Quarto internal or hidden directories.

Usage

```
check_quarto_website_index(path = ".", quiet = FALSE)
```

Arguments

`path`	*Path to project root* `scalar<character>` // default: `"."` (`optional`) The root directory of the Quarto website project to check.
`quiet`	*Suppress warnings* `scalar<logical>` // default: `FALSE` (`optional`) If `TRUE`, no cli warnings are issued. The affected paths are still returned invisibly.

Value

A character vector of folder paths (relative to `path`) that contain `.qmd` files but lack an `index.qmd`. Returned invisibly.

Examples

```
## Not run:
# Check the current project
check_quarto_website_index()

# Check a specific directory
check_quarto_website_index("path/to/quarto-project")

## End(Not run)
```

`crowd_output` *Universal Output Function for Crowd Plots and Tables*

Description

Automatically detects the appropriate output method based on the rendering context and input type. Simplifies workflows by providing a single function that works for both Quarto/knitr rendering (HTML tabsets/tables) and officer-based DOCX generation.

Usage

```
crowd_output(
  plot_list,
  path = "crowd_output.docx",
  force_format = c("auto", "html", "docx"),
  ...
)
```

Arguments

`plot_list`	Either: • A named list of ggplot2 objects (for HTML plots) • A named list of mschart objects (for DOCX plots) • A `saros_officer_plots` object (from `crowd_plots_as_officer()`) • A data.frame or list of data.frames (for tables)
`path`	Character. File path for DOCX output (e.g., `"output.docx"`). Only used when not in a knitr/Quarto rendering context. Default: `"crowd_output.docx"`.
`force_format`	Character. Force a specific output format: • `"auto"` (default): Auto-detect based on context • `"html"`: Force HTML tabset output via `crowd_plots_as_tabset()` • `"docx"`: Force DOCX file output via `crowd_plots_as_docx()`
`...`	Additional arguments passed to `crowd_plots_as_tabset()` or `crowd_plots_as_docx()` depending on the detected/forced format.

Details

Context Detection: The function uses `getOption("knitr.in.progress")` to detect if code is running within a knitr/Quarto rendering context:

- **In knitr/Quarto** → Generates HTML tabset via `crowd_plots_as_tabset()`
- **Outside knitr** → Writes DOCX file via `crowd_plots_as_docx()`

This allows the same code to work in multiple contexts:

- Quarto → HTML rendering

- Quarto → DOCX rendering (still uses HTML output in document)
- R script → Officer-based DOCX generation

Input Type Detection: The function automatically detects and handles different input types:

- ggplot2 objects → Uses `crowd_plots_as_tabset()` or `crowd_plots_as_docx()`
- mschart objects → Uses `crowd_plots_as_docx()`
- `saros_officer_plots` → Uses `crowd_plots_as_docx()`
- data.frame/tables → Uses `crowd_tables_as_tabset()` or writes to DOCX

Typical Workflow:

```
# In a Quarto document - works for both HTML and DOCX output formats
plots <- makeme(
  data = ex_survey,
  dep = b_1:b_3,
  crowd = c("target", "others"),
  mesos_var = "f_uni",
  mesos_group = "Uni of A",
  type = if (is_rendering()) "cat_plot_html" else "cat_plot_docx"
)
crowd_output(plots)  # Auto-detects context and renders appropriately
```

Value

- In knitr/Quarto context: Invisibly returns `NULL` (side effect: prints tabset markdown)
- Outside knitr context: Invisibly returns the DOCX file path

See Also

- `crowd_plots_as_tabset()` for HTML tabset generation
- `crowd_plots_as_docx()` for DOCX file creation
- `is_rendering()` for context detection helper
- `makeme()` for creating plots with crowd parameter

Examples

```
## Not run:
# Example 1: In a Quarto document (auto-detects HTML context)
plots <- makeme(
  data = ex_survey,
  dep = b_1:b_3,
  crowd = c("target", "others"),
  mesos_var = "f_uni",
  mesos_group = "Uni of A",
  type = if (is_rendering()) "cat_plot_html" else "cat_plot_docx"
)
crowd_output(plots)
```

```
# Example 2: In an R script (auto-detects non-knitr context, writes DOCX)
plots <- makeme(
  data = ex_survey,
  dep = b_1:b_3,
  crowd = c("target", "others"),
  mesos_var = "f_uni",
  mesos_group = "Uni of A",
  type = "cat_plot_docx"
)
crowd_output(plots, path = "my_report.docx")

# Example 3: Force specific format
crowd_output(plots, force_format = "docx", path = "forced_output.docx")

## End(Not run)
```

`crowd_plots_as_docx` *Write Plots to Word Document (DOCX)*

Description

High-level function to write a list of mschart plots directly to a Word document file, with optional section headings and chart labels. Simplifies the workflow for generating DOCX reports from Quarto/RMarkdown by handling all officer boilerplate internally.

Usage

```
crowd_plots_as_docx(
  plot_list,
  path,
  docx_template = NULL,
  heading_style = "heading 2",
  prefix_style = "Normal",
  add_dep_label_prefix = TRUE,
  chart_width = NULL,
  chart_height = NULL,
  extract_metadata = TRUE
)
```

Arguments

`plot_list`	Either: • A named list of mschart objects (from `makeme(..., type = "cat_plot_docx", docx_return_object = TRUE)`) • A `saros_officer_plots` object (from `crowd_plots_as_officer()`)
`path`	Character. File path where the DOCX should be saved (e.g., `"output.docx"`).
`docx_template`	Optional template passed to saros' internal `use_docx()` helper. Can be a file path string, `NULL` for default template, or FALSE to skip template.

`heading_style` Character. officer paragraph style for section headings (plot names). Default is `"heading 2"`.

`prefix_style` Character. officer paragraph style for prefix text (main question labels). Default is `"Normal"`. Only used when `add_dep_label_prefix = TRUE`.

`add_dep_label_prefix` Logical. If `TRUE` (default), adds the main question text (from `get_dep_label_prefix()`) as a paragraph before each chart.

`chart_width` Numeric or `NULL`. Width in inches for charts. If `NULL` (default), uses saros' internal `get_docx_dims()` helper based on template page layout.

`chart_height` Numeric or `NULL`. Height in inches for charts. If `NULL` (default), uses saros' internal `get_docx_dims()` helper based on template page layout.

`extract_metadata` Logical. If `TRUE` (default), automatically extracts metadata from plots using `crowd_plots_as_officer()` when `plot_list` is a plain list. Ignored if `plot_list` is already a `saros_officer_plots` object.

Details

This function provides a simple, single-call interface for writing Word documents from saros plots, ideal for Quarto workflows where you want:

- **One QMD file → One DOCX file** (no chapter merging)
- **Minimal code in .qmd chunks** (just `makeme()` + `crowd_plots_as_docx()`)
- **Automatic layout handling** (page dimensions, chart sizing, heading styles)

Typical Workflow:

1. Generate mschart objects with `makeme(..., type = "cat_plot_docx", docx_return_object = TRUE, crowd = ...)`
2. Call `crowd_plots_as_docx(plots, path = "report.docx")` to write the file

Empty Plot Lists: If `plot_list` contains no valid mschart objects, the function:

- Issues a warning via `cli::cli_warn()`
- Creates an empty DOCX file at the specified path
- Returns the path invisibly

Value

Invisible file path to the created DOCX file.

See Also

- `crowd_plots_as_officer()` for obtaining a structured plot object
- `crowd_plots_as_tabset()` for the Quarto/HTML equivalent
- `makeme()` for creating plots with crowd parameter
- `officer::read_docx()` for manual DOCX assembly

Examples

```
## Not run:
# Generate mschart objects for Word
plots <- makeme(
  data = ex_survey,
  dep = b_1:b_3,
  crowd = c("target", "others"),
  mesos_var = "f_uni",
  mesos_group = "Uni of A",
  type = "cat_plot_docx",
  docx_return_object = TRUE
)

# Write directly to a Word file
crowd_plots_as_docx(plots, path = "survey_results.docx")

# With custom template and styling
crowd_plots_as_docx(
  plots,
  path = "styled_report.docx",
  docx_template = "my_template.docx",
  heading_style = "Heading 1",
  prefix_style = "Body Text",
  chart_width = 6,
  chart_height = 4
)

# Without prefix labels
crowd_plots_as_docx(
  plots,
  path = "minimal_report.docx",
  add_dep_label_prefix = FALSE
)

## End(Not run)
```

`crowd_plots_as_officer` *Convert List of Plots to officer-Compatible Format*

Description

Prepares a named list of mschart objects for insertion into a Word document using officer. Typically used with plots generated by `makeme()` with `crowd` parameter and `type = "cat_plot_docx"`.

Usage

```
crowd_plots_as_officer(plot_list, extract_metadata = TRUE)
```

Arguments

plot_list	A named list of mschart objects. Names become section labels. Typically created with: makeme(crowd = c("target", "others"), type = "cat_plot_docx", docx_return_object = TRUE).
extract_metadata	Logical. If TRUE (default), extracts and includes metadata (like dep_label_prefix) from each plot object.

Details

This function validates and structures plot objects for Word document assembly. Unlike crowd_plots_as_tabset() which generates Quarto markdown directly, this function returns a structured R object that can be used programmatically with officer to build Word documents.

Typical Workflow:

1. Generate mschart objects with makeme(..., type = "cat_plot_docx", docx_return_object = TRUE, crowd = ...)
2. Pass the list to crowd_plots_as_officer() for validation and structuring
3. Use the returned object to insert plots into a Word document via officer

Metadata Extraction: When extract_metadata = TRUE, the function extracts:

- dep_label_prefix: Main question text (from get_dep_label_prefix())
- name: Plot name from the list
- Additional attributes attached to the plot object

Value

A list with class "saros_officer_plots" containing:

- plots: Named list of mschart objects ready for officer insertion
- metadata: List of metadata for each plot (if extract_metadata = TRUE), including plot names, dep_label_prefix, and other attributes; NULL if extract_metadata = FALSE
- n_plots: Integer count of valid mschart objects after filtering

See Also

- crowd_plots_as_tabset() for the Quarto/HTML equivalent
- makeme() for creating plots with crowd parameter
- get_dep_label_prefix() for retrieving main question metadata
- mschart::body_add_chart() for inserting charts into Word documents

Examples

```
## Not run:
# Generate mschart objects for Word
plots <- makeme(
  data = ex_survey,
  dep = b_1:b_3,
  crowd = c("target", "others"),
  mesos_var = "f_uni",
  mesos_group = "Uni of A",
  type = "cat_plot_docx",
  docx_return_object = TRUE
)

# Prepare for officer assembly
officer_plots <- crowd_plots_as_officer(plots)

# Use in officer workflow
library(officer)
doc <- read_docx()
for (plot_name in names(officer_plots$plots)) {
  doc <- doc |>
    body_add_par(plot_name, style = "heading 2") |>
    body_add_par(officer_plots$metadata[[plot_name]]$dep_label_prefix) |>
    mschart::body_add_chart(officer_plots$plots[[plot_name]])
}
print(doc, target = "output.docx")

## End(Not run)
```

---

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,
  pagebreak = c("never", "auto", "always")
)
```

Arguments

plot_list	A named list of ggplot2 objects. Names become tab labels. Typically created with makeme(crowd = c("target", "others")).
plot_type	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
save	Logical. If TRUE (default), generates download links for plot data and images via get_fig_title_suffix_from_ggplot().
fig_height	Numeric or NULL. Manual figure height override in inches. If NULL (default), height is calculated automatically based on plot_type.
fig_height_int_default	Numeric. Default height for interval plots when auto-calculation is not available (default: 6 inches).
pagebreak	Character. Controls page break insertion between plots: • "never" (default): Never insert page breaks • "auto": Insert page breaks for non-HTML/non-Typst formats only • "always": Always insert page breaks between plots

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)
```

`crowd_tables_as_tabset`

Convert List of Tables to Quarto Tabset

Description

Creates a Quarto tabset from a named list of data frames, rendering each as a table in its own tab. Designed to be called within a Quarto document code chunk with `results='asis'`.

Usage

```
crowd_tables_as_tabset(
  tbl_list,
  table_fn = knitr::kable,
  pagebreak = c("never", "auto", "always")
)
```

Arguments

tbl_list	A named list of data frames. Names become tab labels.
table_fn	A function that converts a data frame to a printable table object. Defaults to knitr::kable(). Other options include gt::gt(), tinytable::tt, etc. Can be set globally via global_settings_set(new = list(table_fn = gt::gt), fn_name = "crowd_tables_as_tabset").
pagebreak	Character. Controls page break insertion between tables: • "never" (default): Never insert page breaks • "auto": Insert page breaks for non-HTML/non-Typst formats only • "always": Always insert page breaks between tables

Details

This function outputs raw Quarto markdown (level-5 headings) interleaved with printed tables. The enclosing chunk should use the Quarto tabset panel layout and results: asis.

Value

Called for its side effects (printing tabset markdown and tables). Returns NULL invisibly.

See Also

crowd_plots_as_tabset() for the plot equivalent.

Examples

```
## Not run:
tbl_list <- list(
  "Group A" = head(mtcars),
  "Group B" = tail(mtcars)
)
crowd_tables_as_tabset(tbl_list)

# Use gt::gt instead
crowd_tables_as_tabset(tbl_list, table_fn = gt::gt)

## End(Not run)
```

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,
  multiplier_hide_axis_single_var = 0.6,
  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	Numeric. 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.

multiplier_per_bar
: Numeric. Multiplier per bar height (thickness).

multiplier_per_legend_line
: Numeric. Multiplier per legend line.

multiplier_per_plot
: Numeric. Multiplier for entire plot estimates.

fixed_constant
: Numeric. Fixed constant to be added to the height.

margin_in_cm
: Numeric. Margin in centimeters.

figure_width_in_cm
: Numeric. Width of the figure in centimeters.

max
: Numeric. Maximum height.

min
: Numeric. Minimum height.

hide_axis_text_if_single_variable
: Boolean. Whether the label is hidden for single dependent variable plots.

multiplier_hide_axis_single_var
: Numeric. Multiplier to reduce panel height when hiding axis text for single variable (default 0.6).

add_n_to_dep_label, add_n_to_indep_label
: 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.

showNA
: 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. Works with both ggplot2 and mschart objects.

Usage

```
fig_height_h_barchart2(plot_obj, ...)

fig_height_h_barchart2.ggplot(
  plot_obj,
  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 = 12,
  min = 1,
  multiplier_hide_axis_single_var = 0.6
)

fig_height_h_barchart2.ms_chart(
  plot_obj,
  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 = 12,
  min = 1,
  multiplier_hide_axis_single_var = 0.6
)

fig_height_h_barchart2.default(plot_obj, ...)
```

Arguments

`plot_obj`	A plot object from `makeme()` - either a `ggplot2` object or an `ms_chart` object
`...`	Additional parameters passed to the specific method (`fig_height_h_barchart2.ggplot` or `fig_height_h_barchart2.ms_chart`)
`main_font_size`	Numeric. Font size for the main text.
`strip_angle`	Numeric. Angle of the strip text.
`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.
`legend_location`	Character. Location of the legend. "plot" (default) or "panel".
`n_legend_lines`	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.

`multiplier_hide_axis_single_var`
Numeric. Multiplier to reduce panel height when hiding axis text for single variable (default 0.6).

Value

Numeric value representing the estimated height of the figure.

Examples

```
# With ggplot2 (cat_plot_html)
fig_height_h_barchart2(makeme(data = ex_survey, dep = b_1:b_2, indep = x1_sex))

# With mschart (cat_plot_docx)
## Not run:
fig_height_h_barchart2(
  makeme(data = ex_survey, dep = b_1:b_2,
         type = "cat_plot_docx", docx_return_object = TRUE)
)

## End(Not run)
```

---

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_dep_label_prefix *Retrieve the dep label prefix from a saros output object*

Description

Retrieves the "dep_label_prefix" attribute that saros attaches to every object returned by makeme() / make_content.*(). This is the main question text — the shared label prefix of all dependent variables used to produce the object.

Usage

```
get_dep_label_prefix(obj)
```

Arguments

obj	Any object returned by makeme() or a make_content.*() method (ggplot, data.frame, mschart, ...).

Details

Storage location by class:

- **ggplot / gg** and **ms_barchart**: attribute is stored on obj$data (when obj$data is a data.frame) so that it survives further + operations. This function reads from obj$data first for both classes.
- **data.frame and other objects**: attribute stored directly on obj.

Value

A character scalar: the dep label prefix if present and non-empty, otherwise "".

Examples

```
p <- makeme(data = ex_survey, dep = b_1:b_3)
get_dep_label_prefix(p)
```

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 = ",
  folder = NULL,
  file_prefix = NULL,
  file_suffixes = c(".csv", ".png"),
  link_prefixes = c("[CSV](", "[PNG]("),
  save_fns = NULL,
  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 = ").
folder	String. Folder path where files should be saved. If NULL, uses global settings or defaults to "." (current directory).
file_prefix	String. Prefix for saved filenames. If NULL, uses global settings or defaults to "" (no prefix).
file_suffixes	Character vector. File extensions for the saved plot images (default: c(".csv", ".png")). Should include the dot. (default: c(".csv", ".png")). Should include the dot.
link_prefixes	Character vector. Markdown link text prefixes for the plot download links (default: c("[CSV](", "[PNG]("). (default: c("[CSV](", "[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() *with Palette Support*

Description

Saves ggplot2 objects with automatic palette application from global settings. Inherits palette settings from girafe() global options, ensuring saved plots match the appearance of interactive plots.

Usage

```
ggsaver(
  plot,
  filename,
  palette_codes = NULL,
  priority_palette_codes = NULL,
  label_wrap_width = 80,
  ncol = NULL,
  byrow = TRUE,
  ...
)
```

Arguments

plot	A ggplot2 object to save.
filename	File path where the plot should be saved.
palette_codes	Optional list of character vectors. Each vector contains colours. Vectors can optionally be named, where names are categories and values are colours. Inherits from girafe() global settings if not specified. The final character vector of the list is used as a fallback. Defaults to NULL.
priority_palette_codes	Optional named character of categories (as names) with corresponding colours (as values) which are used first. Inherits from girafe() global settings if not specified. Defaults to NULL.
label_wrap_width	Integer. Number of characters fit on the legend labels before wrapping. Inherits from girafe() global settings. Defaults to 80.
ncol	Optional integer or NULL for legend columns. Inherits from girafe() global settings. Defaults to NULL.
byrow	Whether to display legend keys by row or by column. Inherits from girafe() global settings. Defaults to TRUE.
...	Arguments forwarded to ggplot2::ggsave()

Details

This function extends ggplot2::ggsave() by applying colour palettes before saving, ensuring consistency between interactive plots (via girafe()) and saved static images. Palette settings are inherited from global settings set via global_settings_set() for the "girafe" function.

If palette_codes is provided (either directly or via global settings), the function applies the same palette transformation that girafe() uses for interactive plots.

Value

No return value, called for side effects (saves plot to file)

See Also

- girafe() for interactive plots with palette support
- global_settings_set() for setting default palettes
- ggplot2::ggsave() for the underlying save function

Examples

```
library(ggplot2)
my_plot <- ggplot(data=mtcars, aes(x=hp, y=mpg, fill=factor(cyl))) + geom_point()

## Not run:
# Save with default settings
ggsaver(my_plot, tempfile(fileext = ".png"))

# Set global palette and save
global_settings_set(
  fn_name = "girafe",
  new = list(palette_codes = list(c("red", "blue", "green")))
)
ggsaver(my_plot, tempfile(fileext = ".png"))

# Override global palette for specific save
ggsaver(
  my_plot,
  tempfile(fileext = ".png"),
  palette_codes = list(c("purple", "orange", "yellow"))
)

## End(Not run)
```

---

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

ggobj	ggplot2-object.
...	Dots forwarded to ggiraph::girafe()
char_limit	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.
label_wrap_width	Integer. Number of characters fit on the axis text space before wrapping.
interactive	Boolean. Whether to produce a ggiraph-plot with interactivity (defaults to TRUE) or a static ggplot2-plot.
palette_codes	Optional list of character vectors. Each vector contains colours. Vectors can optionally be named, where names are categories and values are colours. The final character vector of the list is used as a fallback. Defaults to NULL.
priority_palette_codes	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 palette_codes. Defaults to NULL.
ncol	Optional integer or NULL.
byrow	Whether to display legend keys by row or by column.
colour_2nd_binary_cat	Optional string. Color for the second category in binary checkbox plots. When set together with checked and not_checked, reverses the category order so that not_checked appears second and receives this color. Ignored if checkbox criteria are not met.
checked, not_checked	Optional string. If specified and the fill categories of the plot matches these, a special plot is returned where not_checked 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` *Get Global Options for saros-functions*

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", quiet = FALSE)
```

Arguments

fn_name	String, one of "make_link", "fig_height_h_barchart" and "makeme".
quiet	Flag. If FALSE (default), informs about what has been set.

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))
```

`insert_text` *Insert Text from a Data Frame by Chunk Name*

Description

Looks up a text string from a data frame based on a chunk name and position (before/after). Optionally expands knitr templating syntax (`{{...}}`) found in the text.

Usage

```
insert_text(data, chunk, before = TRUE, error_on_empty = NULL, enabled = TRUE)
```

Arguments

`data`	A data frame with columns `chunk`, `before`, and `text`.
`chunk`	Character. The chunk name to look up in `data`. If the file extension is `"rmarkdown"`, it is stripped automatically.
`before`	Logical. Whether to retrieve the text marked as "before" (`TRUE`, default) or "after" (`FALSE`) the chunk.
`error_on_empty`	Controls behaviour when no matching text is found: • `TRUE`: throws an error via `cli::cli_abort()`. • `FALSE`: issues a warning via `cli::cli_warn()`. • `NULL` (default): silently returns an empty `AsIs` object.
`enabled`	Logical. If `FALSE`, the function returns `I(character(0))` immediately without any lookup. Can be set globally via `global_settings_set(new = list(enabled = FALSE), fn_name = "insert_text")`.

Details

The function filters `data` for rows matching the given `chunk` and `before` values. If exactly one row matches, its `text` column is returned. If the text contains knitr templating delimiters (`{{`), it is expanded with `knitr::knit_expand()`.

Value

An `I()`-wrapped character string. If no match is found and `error_on_empty` is `NULL`, an empty `AsIs` character vector.

Examples

```
## Not run:
texts <- data.frame(
  chunk = c("intro", "intro"),
  before = c(TRUE, FALSE),
  text = c("Before the chunk.", "After the chunk.")
)
insert_text(texts, "intro", before = TRUE)
insert_text(texts, "intro", before = FALSE)

## End(Not run)
```

---

is_rendering *Detect if Running in knitr/Quarto Rendering Context*

Description

Helper function to detect if code is running within a knitr/Quarto rendering context. Useful for creating unified code that works in both interactive R sessions and when rendering documents.

Usage

```
is_rendering()
```

Details

This function checks `getOption("knitr.in.progress")` which is set by knitr during document rendering. This works for:

- Quarto documents (all output formats: HTML, DOCX, PDF, etc.)
- R Markdown documents
- Any knitr-based rendering systems

Returns `FALSE` when running in:

- Interactive R sessions (RStudio console, R terminal)
- R scripts executed outside knitr
- Shiny applications (unless explicitly using knitr)

Value

Logical. `TRUE` if rendering a document, `FALSE` otherwise.

See Also

`crowd_output()` for automatic context-aware output generation

Examples

```
# Check if rendering a document
if (is_rendering()) {
  message("Rendering a document with knitr/Quarto")
} else {
  message("Running in regular R session")
}

# Use for conditional logic
plot_type <- if (is_rendering()) "cat_plot_html" else "cat_plot_docx"
```

---

makeme *Embed Interactive Plot of Various Kinds Using Tidyselect Syntax*

Description

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

Usage

```
makeme(
  data,
  dep = tidyselect::everything(),
  indep = NULL,
 type = c("auto", "cat_plot_html", "int_plot_html", "cat_table_html", "int_table_html",
    "sigtest_table_html", "cat_plot_docx", "int_plot_docx", "cat_table_docx",
    "chr_table_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 = 9,
  main_font_size = 9,
  strip_font_size = 6,
  legend_font_size = 6,
  font_family = "sans",
  path = NULL,
  docx_template = NULL,
  docx_return_object = TRUE
)
```

Arguments

data	*Your data.frame/tibble or srvyr-object (experimental)* data.frame // *required* The data to be used for plotting.
dep, indep	*Variable selections* <tidyselect> // *Default:* NULL, meaning everything for dep, nothing for indep. Columns in data. dep is compulsory.
type	*Kind of output* scalar<character> // *default:* "auto" (optional) The type of output to generate. Use "auto" (default) to automatically detect the appropriate type based on your dependent variables: • Numeric/integer variables → "int_plot_html" • Factor/character variables → "cat_plot_html" For a list of all 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.

totals

Include totals

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

Whether to include totals in the output.

categories_treated_as_na

NA categories

vector<character> // *default:* NULL (optional)

Categories that should be treated as NA.

label_separator

How to separate main question from sub-question

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

Separator for main question from sub-question.

error_on_duplicates

Error or warn on duplicate labels

scalar<logical> // *default:* TRUE (optional)

Whether to abort (TRUE) or warn (FALSE) if the same label (suffix) is used across multiple variables.

showNA

Show NA categories

vector<character> // *default:* c("ifany", "always", "never") (optional)

Choose whether to show NA categories in the results.

data_label

Data label

scalar<character> // *default:* "proportion" (optional)

One of "proportion", "percentage", "percentage_bare", "count", "mean", or "median".

data_label_position

Data label position

scalar<character> // *default:* "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).

html_interactive

Toggle interactive plot

scalar<logical> // *default:* TRUE (optional)

Whether the plot is to be interactive (ggiraph) or static (ggplot2).

hide_axis_text_if_single_variable
: *Hide y-axis text if just a single variable*
scalar<boolean> // *default:* FALSE (optional)
Whether to hide text on the y-axis label if just a single variable.

hide_label_if_prop_below
: *Hide label threshold*
scalar<numeric> // *default:* NULL (optional)
Whether to hide label if below this value.

inverse
: *Flag to swap x-axis and faceting*
scalar<logical> // *default:* FALSE (optional)
If TRUE, swaps x-axis and faceting.

vertical
: *Display plot vertically*
scalar<logical> // *default:* FALSE (optional)
If TRUE, display plot vertically.

digits
: *Decimal places*
scalar<integer> // *default:* 0L (optional)
Number of decimal places.

data_label_decimal_symbol
: *Decimal symbol*
scalar<character> // *default:* "." (optional)
Decimal marker, some might prefer a comma ',' or something else entirely.

x_axis_label_width, strip_width
: *Label width of x-axis and strip texts in plots*
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
: *What to sort dependent variables by*
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` *What to sort independent variable categories by*
`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.

`sort_by` *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`
: *Output path for DOCX*
`scalar<character>` // *default:* `NULL` (`optional`)
Path to save docx-output.

`docx_template`
: *Filename or rdocx object*
`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.

`docx_return_object`
: *Return underlying object instead of rdocx*
`scalar<logical>` // *default:* `TRUE` (`optional`)
For DOCX output types: if TRUE, return the underlying object (mschart for plots, data.frame for tables) instead of embedding it in an rdocx document.

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

`type`	*Method name* `scalar<character>` with a class named by itself. Optional string indicating the specific method. Occasionally useful for error messages, etc.
`...`	*Dots* 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_file_links` *Create Markdown Links to Files with Document Titles*

Description

Scans a folder for files matching a pattern and generates a markdown list with links to each file. The link text is extracted from the document's title metadata (for DOCX, PPTX, PDF files) or uses the filename as fallback.

Usage

```
make_file_links(
  folder = ".",
  pattern = "",
  bullet_style = "-",
  recurse = FALSE,
  relative_links = TRUE
)
```

Arguments

folder	String. Path to the folder containing files. Defaults to current directory ("..").
pattern	String. Regular expression pattern for file matching (e.g., "\\.pptx$", "\\.pdf$", "^report_.*\\.docx$"). Defaults to "" (all files).
bullet_style	String. Markdown list style. One of "-" (unordered), "*" (unordered), or "1." (ordered). Defaults to "-".
recurse	Logical. Whether to search recursively in subdirectories. Defaults to FALSE.
relative_links	Logical. Whether to use relative or absolute paths in links. If TRUE, paths are relative to folder. If FALSE, uses absolute paths. Defaults to TRUE.

Details

The function attempts to extract document titles from file metadata:

- **DOCX files**: Extracts title from document properties (requires officer package)
- **PPTX files**: Extracts title from presentation properties (requires officer package)
- **PDF files**: Extracts title from PDF metadata (requires pdftools package)

If title extraction fails or the file type is unsupported, the filename (without extension) is used as the link text.

The function preserves the order of files as returned by fs::dir_ls(), which typically sorts alphabetically.

Value

A character string containing a markdown-formatted list of links. Each line contains a bullet point and a link with the document title as link text. Returns empty string if no files found.

Optional Packages

The pdftools package is optional (in Suggests) and only needed for PDF title extraction.

Examples

```
## Not run:
# Create links to all PowerPoint files in a folder
make_file_links(folder = "presentations", pattern = "\\.pptx$")
```

```
# Create links to PDF reports with numbered list
make_file_links(
  folder = "reports",
  pattern = "^report_.*\\.pdf$",
  bullet_style = "1."
)

# Recursively find all Office documents
make_file_links(
  folder = "documents",
  pattern = "\\.(docx|pptx)$",
  recurse = TRUE
)

## End(Not run)
```

`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`	*Data or object* `<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`	*Where to store file* `scalar<character>` // *default:* `"."` (`optional`) Defaults to same folder.

`file_prefix`, `file_suffix`
: *File prefix/suffix*
`scalar<character>` // *default:* `""` and `".csv"` (`optional`)
`file_suffix` should include the dot before the extension.

`save_fn`
: *Saving function*
`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 `")"`
The stuff that is returned.

`...`
: *Dynamic dots*
`<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`	*Data or object* `<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()`).
`...`	*Dynamic dots* <dynamic-dots> Arguments forwarded to the corresponding functions that create the elements.
`folder`	*Where to store file* `scalar<character>` // *default:* `"."` (`optional`) Defaults to same folder.
`file_prefix, file_suffix`	*File prefix/suffix* `scalar<character>` // *default:* `""` and `".csv"` (`optional`) `file_suffix` should include the dot before the extension.
`save_fn`	*Saving function* `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 `")"` 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`	*Data or object* `<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()`).
`...`	*Dynamic dots* `<dynamic-dots>` Arguments forwarded to the corresponding functions that create the elements.
`folder`	*Where to store file* `scalar<character>` // *default:* `"."` (`optional`) Defaults to same folder.
`file_prefix, file_suffix`	*File prefix/suffix* `scalar<character>` // *default:* `""` and `".csv"` (`optional`) `file_suffix` should include the dot before the extension.
`save_fn`	*Saving function* `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 `")"`
The stuff that is returned.

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

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

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

`data`	Dataset
`dep`, `indep`	Tidyselect syntax
`mesos_var`	Optional, NULL or string specifying name of variable used to split dataset.
`mesos_group`	Optional, NULL or string specifying value in `mesos_var` indicating the target group.
`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_range(data = ex_survey, dep = b_1:b_3, indep = x1_sex)
```

n_range2 *Provides a range (or single value) for N in a plot object from* makeme()

Description

Takes a plot object from makeme() and returns the sample size (N) range as a formatted string. Works with both ggplot2 objects and mschart objects.

Usage

```
n_range2(plot_obj, ...)

n_range2.ggplot(
  plot_obj,
  glue_template_1 = "{n}",
  glue_template_2 = "[{n[1]}-{n[2]}]"
)

n_range2.ms_chart(
  plot_obj,
  glue_template_1 = "{n}",
  glue_template_2 = "[{n[1]}-{n[2]}]"
)

n_range2.default(plot_obj, ...)
```

Arguments

plot_obj	A plot object from makeme() - either a ggplot2 object or an ms_chart object
...	Additional parameters passed to the specific method
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

```
# With ggplot2 (cat_plot_html)
n_range2(makeme(data = ex_survey, dep = b_1:b_3))

# With mschart (cat_plot_docx)
## Not run:
n_range2(
  makeme(data = ex_survey, dep = b_1:b_3,
```

```
          type = "cat_plot_docx", docx_return_object = TRUE)
)

## End(Not run)
```

`output_format` *Detect the Current Output Format*

Description

Returns the output format of the current rendering context. When called inside a Quarto/knitr document, delegates to `knitr::pandoc_to()`. When called outside of Quarto (e.g. in an officer-based script), returns `"officer"`.

Usage

```
output_format()
```

Value

A character string: `"html"`, `"docx"`, `"typst"`, `"officer"`, or another format reported by `knitr::pandoc_to()`.

Examples

```
## Not run:
output_format()

## End(Not run)
```

`quarto_pdf_post_render`
Quarto Post-Render: Enrich PDF Files with DOCX Titles

Description

A post-render function for Quarto projects that processes rendered PDF output files. For each PDF, it checks if a corresponding DOCX file with the same base name exists, extracts the title from the DOCX document properties, sets it as the PDF metadata title, and updates the link text in the accompanying `index.html`.

Usage

```
quarto_pdf_post_render(
  output_files = strsplit(Sys.getenv("QUARTO_PROJECT_OUTPUT_FILES"), "\n")[[1L]]
)
```

Arguments

output_files	Character vector of output file paths from Quarto. Defaults to the QUARTO_PROJECT_OUTPUT_FILES environment variable (newline-separated paths set by Quarto during project render). If that variable is empty, falls back to reading from stdin as provided by Quarto's post-render hook.

Details

To use as a Quarto post-render script, add to _quarto.yml:

```
project:
  post-render:
    - "Rscript -e 'saros::quarto_pdf_post_render()'"
```

Processing steps for each .pdf file:

1. Checks if a .docx with the same base name exists in the same directory
2. Extracts the title from the DOCX document properties (via officer)
3. Sets the extracted title as the PDF file's metadata title (requires Ghostscript)
4. Locates index.html in the same directory as the PDF
5. Replaces the <a> link text for that PDF with the extracted title

Value

Invisible NULL. Called for side effects.

System Requirements

Setting PDF metadata title requires Ghostscript to be installed and available on the system PATH:

- **Linux/macOS**: gs
- **Windows**: gswin64c or gswin32c

If Ghostscript is not found, a warning is issued and only the HTML link text update is performed.

See Also

extract_docx_title() for the DOCX title extraction logic.

Examples

```
## Not run:
# Called automatically by Quarto post-render, or manually:
quarto_pdf_post_render(c("_site/report/report.pdf"))

## End(Not run)
```

`txt_from_cat_mesos_plots` *Extract Text Summary from Categorical Mesos Plots*

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,
  checked = NULL,
  not_checked = NULL,
  digits = 2,
  selected_categories_last_split = " or ",
  fallback_string = character(),
  reverse = FALSE,
  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

`plots`	A list of two plot objects (or data frames with plot data) to compare. Each must contain columns: `.variable_label`, `.category`, `.category_order`, `.proportion`.
`min_prop_diff`	Numeric. Minimum proportion difference (default 0.10) required between groups to generate text. Differences below this threshold are ignored.

`n_highest_categories`
: Integer. Number of top categories to include in the comparison (default 1). Categories are selected based on `.category_order`. Only applied if the variable has more categories than this value.

`flip_to_lowest_categories`
: Logical. If TRUE, compare lowest categories instead of highest (default FALSE).

`checked`, `not_checked`
: Optional string. When the categories of a variable exactly match these two values, the comparison is always made on `checked` — mirroring the visual convention in the bar chart where the checked category is rendered in colour on the left. Defaults to `NULL`; when `NULL`, the function tries to auto-detect the values from `global_settings_get("girafe")$checked` / `$not_checked`; if those are also `NULL`, checkbox handling is disabled and normal order-based category selection applies.

`digits`
: Integer. Number of decimal places for rounding proportions (default 2).

`selected_categories_last_split`
: Character. Separator for the last item when listing multiple categories (default " or ").

`fallback_string`
: Character. String to return when validation fails (default `character()`).

`reverse`
: Logical. If TRUE, reverses the order of the output text summaries (default FALSE).

`glue_str_pos`
: Character vector. Templates for positive differences (group_1 > group_2). Available placeholders: `{var}`, `{group_1}`, `{group_2}`, `{selected_categories}`.

`glue_str_neg`
: Character vector. Templates for negative differences (group_2 > group_1). Same placeholders as `glue_str_pos`.

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.

Checkbox (checked/not_checked) variables: When `checked` and `not_checked` are both strings, any variable whose categories exactly match that pair is treated as a checkbox variable. For such variables the comparison is always made on the `checked` category, regardless of `flip_to_lowest_categories`. This mirrors the visual convention in the bar chart where the checked category is rendered in colour on the left — the semantically meaningful side — even though its `.category_order` may not be the highest. If `checked`/`not_checked` are `NULL`, the function tries to auto-detect them from `global_settings_get("girafe")$checked` / `$not_checked`; if those are also `NULL`, checkbox handling is disabled.

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