# Package 'saros'

June 4, 2025

**Type** Package

**Title** Semi-Automatic Reporting of Ordinary Surveys

**Version** 1.5.4

**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, tidyr, tidyselect, utils, vctrs

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

**Config/testthat/edition** 3

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**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** 2025-06-04 12:10:06 UTC

# Contents

## embed_cat_prop_plot *Embed Interactive Categorical Plot (DEPRECATED!)*

### 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 *Embed Reactable Table (DEPRECATED!)*

### 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 *Interactive table of text data (DEPRECATED)*

---

## 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. |

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.

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.

## 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

| | |
|---|---|
| `ggobj` | `ggplot2`-object |
| `main_font_size` | Numeric. Font size for the main text. |
| `strip_angle` | Integer. 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.

## Value

Numeric value representing the estimated height of the figure.

## Examples

```
fig_height_h_barchart2(makeme(data = ex_survey, dep = b_1:b_3, 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_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

| | |
|---|---|
| `plot` | Plot |
| `filename` | Note |
| `...` | Arguments forwarded to `ggplot2::ggsave()` |

### 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,
  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 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. |
| 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. |

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")
```

### 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 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("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"),
  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_by = ".upper",
  descend = TRUE,
  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 | *Your data.frame/tibble or srvyr-object (experimental)*<br>data.frame // *required*<br>The data to be used for plotting. |
| dep, indep | *Variable selections*<br><tidyselect> // *Default:* NULL, meaning everything for dep, nothing for indep.<br>Columns in data. dep is compulsory. |
| type | *Kind of output*<br>scalar<character> // *default:* "cat_plot_html" (optional)<br>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.

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".

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_by`
: *What to sort output by*
`vector<character>` // *default:* `NULL` (`optional`)

Sort output (and collapse if requested). When using `indep`-argument, sorting differs between ordered factors and unordered factors: Ordering of ordered factors is always respected in output. Unordered factors will be reordered by `sort_by`. Currently, this works best for a single `dep`.

**NULL** No sorting.

**".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. 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::theme_set()). 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.

## Value

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

## Examples

```
makeme(
  data = ex_survey,
  dep = b_1:b_3
)
makeme(
  data = ex_survey,
  dep = b_1, indep = x1_sex
)
makeme(
  data = ex_survey,
  dep = b_1:b_3, indep = c(x1_sex, x2_human),
  type = "sigtest_table_html"
)
makeme(
  data = ex_survey,
  dep = b_1, indep = x1_sex,
  type = "cat_prop_plot_docx"
)
makeme(
  data = ex_survey,
  dep = p_1:p_4, indep = x2_human,
  type = "cat_table_html"
)
makeme(
  data = ex_survey,
  dep = b_1:b_3,
  crowd = c("target", "others", "all"),
  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*<br>scalar<character> with a class named by itself.<br>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_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*<br><data.frame\|tbl\|obj><br>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*<br><dynamic-dots><br>Arguments forwarded to the corresponding functions that create the elements. |
| `folder` | *Where to store file*<br>scalar<character> // *default:* `"."` (optional)<br>Defaults to same folder. |
| `file_prefix, file_suffix` | *File prefix/suffix*<br>scalar<character> // *default:* `""` and `".csv"` (optional)<br>`file_suffix` should include the dot before the extension. |
| `save_fn` | *Saving function*<br>`function` // *default:* `utils::write.csv`<br>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*<br>scalar<character> // *default:* `"[download data]("` and `")"`<br>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*<br>`<data.frame\|tbl\|obj>`<br>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*<br>`<dynamic-dots>`<br>Arguments forwarded to the corresponding functions that create the elements. |
| `folder` | *Where to store file*<br>`scalar<character>` // *default:* `"."` (`optional`)<br>Defaults to same folder. |
| `file_prefix, file_suffix` | *File prefix/suffix*<br>`scalar<character>` // *default:* `""` and `".csv"` (`optional`)<br>`file_suffix` should include the dot before the extension. |
| `save_fn` | *Saving function*<br>`function` // *default:* `utils::write.csv`<br>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*<br>`scalar<character>` // *default:* `"[download data]("` and `")"`<br>The stuff that is returned. |
| `separator_list_items` | *Separator string between multiple list items*<br>`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* ggplot2*-object from* makeme()

## 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))
```

# Index