Package ‘volker’

April 5, 2025

Type Package

Title High-Level Functions for Tabulating, Charting and Reporting Survey Data

Version 3.1.0

Date 2025-04-05

Description Craft polished tables and plots in Markdown reports.
Simply choose whether to treat your data as counts or metrics,
and the package will automatically generate well-designed default tables and plots for you.
Boiled down to the basics, with labeling features and simple interactive reports.
All functions are 'tidyverse' compatible.

URL https://github.com/strohne/volker,
https://strohne.github.io/volker/

BugReports https://github.com/strohne/volker/issues

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

LazyData true

Imports stats, utils, rlang, lifecycle, tibble, dplyr, tidyr, tidyselect, ggplot2 (>= 2.2.1), scales, base64enc, purrr, magrittr, skimr, broom, knitr, kableExtra, rmarkdown, psych, car, effectsize, heplots

Depends R (>= 4.2)

Suggests tidyverse, remotes, usethis, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Jakob Jünger [aut, cre, cph] (<https://orcid.org/0000-0003-1860-6695>),
Henrieke Kotthoff [aut, ctb],
Chantal Gärtner [ctb] (<https://orcid.org/0000-0002-3653-6013>)

Maintainer Jakob Jünger <jakob.juenger@uni-muenster.de>

Repository CRAN

Date/Publication 2025-04-05 20:50:02 UTC

Contents

`add_clusters` *Add cluster number to a data frame*

Description

Clustering is performed using `stats::kmeans`.

[Experimental]

Usage

```
add_clusters(data, cols, newcol = NULL, k = 2, method = "kmeans", clean = TRUE)
```

Arguments

`data`	A dataframe.
`cols`	A tidy selection of item columns.
`newcol`	Name of the new cluster column as a character vector. Set to NULL (default) to automatically build a name from the common column prefix, prefixed with "cls_".
`k`	Number of clusters to calculate. Set to NULL to output a scree plot for up to 10 clusters and automatically choose the number of clusters based on the elbow criterion. The within-sums of squares for the scree plot are calculated by `stats::kmeans`.
`method`	The method as character value. Currently, only kmeans is supported. All items are scaled before performing the cluster analysis using `base::scale`.
`clean`	Prepare data by data_clean.

Value

The input tibble with additional column containing cluster values as a factor. The new column is prefixed with "cls_". The new column contains the fit result in the attribute stats.kmeans.fit. The names of the items used for clustering are stored in the attribute stats.kmeans.items. The clustering diagnostics (Within-Cluster and Between-Cluster Sum of Squares) are stored in the attribute stats.kmeans.wss.

Examples

```
library(volker)
ds <- volker::chatgpt

volker::add_clusters(ds, starts_with("cg_adoption"), k = 3)
```

`add_factors` *Add PCA columns along with summary statistics (KMO and Bartlett test) to a data frame*

Description

PCA is performed using `psych::pca` usind varimax rotation. Bartlett's test for sphericity is calculated with `psych::cortest.bartlett`. The Kaiser-Meyer-Olkin (KMO) measure is computed using `psych::KMO`.

[Experimental]

Usage

```
add_factors(data, cols, newcols = NULL, k = 2, method = "pca", clean = TRUE)
```

Arguments

`data` A dataframe.

`cols` A tidy selection of item columns.

`newcols` Names of the factor columns as a character vector. Must be the same length as k or NULL. Set to NULL (default) to automatically build a name from the common column prefix, prefixed with "fct_", postfixed with the factor number.

`k` Number of factors to calculate. Set to NULL to calculate eigenvalues for all components up to the number of items and automatically choose k. Eigenvalues and the decision on k are calculated by `psych::fa.parallel`.

`method` The method as character value. Currently, only pca is supported.

`clean` Prepare data by data_clean.

Value

The input tibble with additional columns containing factor values. The new columns are prefixed with "fct_". The first new column contains the fit result in the attribute psych.pca.fit. The names of the items used for factor analysis are stored in the attribute psych.pca.items. The summary diagnostics (Bartlett test and KMO) are stored in the attribute psych.kmo.bartlett.

Examples

```
library(volker)
ds <- volker::chatgpt

volker::add_factors(ds, starts_with("cg_adoption"))
```

`add_index` *Calculate the mean value of multiple items*

Description

[Experimental]

Usage

```
add_index(data, cols, newcol = NULL, cols.reverse, clean = TRUE)
```

Arguments

`data` A dataframe.

`cols` A tidy selection of item columns.

`newcol` Name of the index as a character value. Set to NULL (default) to automatically build a name from the common column prefix, prefixed with "idx_".

`cols.reverse` A tidy selection of columns with reversed codings.

`clean` Prepare data by data_clean.

Value

The input tibble with an additional column that contains the index values. The column contains the result of the alpha calculation in the attribute named "psych.alpha".

Examples

```
ds <- volker::chatgpt
volker::add_index(ds, starts_with("cg_adoption"))
```

`chatgpt` *ChatGPT Adoption Dataset CG-GE-APR23*

Description

A small random subset of data from a survey about ChatGPT adoption. The survey was conducted in April 2023 within the population of German Internet users.

Usage

```
chatgpt
```

Format

`chatgpt`**:**

A data frame with 101 rows and 19 columns:

case A running case number

adopter Adoption groups inspired by Roger's innovator typology.

use_ Columns starting with use contain data about ChatGPT usage in different contexts.

cg_activities Text answers to the question, what the respondents do with ChatGPT.

cg_adoption_ A scale consisting of items about advantages, fears, and social aspects. The scales match theoretical constructs inspired by Roger's diffusion model and Davis' Technology Acceptance Model

sd_ Columns starting with sd contain sociodemographics of the respondents.

Details

Call codebook(volker::chatgpt) to see the items and answer options.

Source

Communication Department of the University of Münster (gehrau@uni-muenster.de).

codebook *Get variable and value labels from a data set*

Description

Variable labels are extracted from their comment or label attribute. Variable values are extracted from factor levels, the labels attribute, numeric or boolean attributes.

Usage

```
codebook(data, cols, values = TRUE)
```

Arguments

`data`	A tibble.
`cols`	A tidy variable selections to filter specific columns.
`values`	Whether to output values (TRUE) or only items (FALSE)

Details

[Experimental]

Value

A tibble with the columns:

- item_name: The column name.
- item_group: First part of the column name, up to an underscore.
- item_class: The last class value of an item (e.g. numeric, factor).
- item_label: The comment attribute of the column.
- value_name: In case a column has numeric attributes, the attribute names.
- value_label: In case a column has numeric attributes or T/F-attributes, the attribute values. In case a column has a levels attribute, the levels.

Examples

```
volker::codebook(volker::chatgpt)
```

`effect_counts` *Output effect sizes and test statistics for count data*

Description

The type of effect size depends on the number of selected columns:

- One categorical column: see effect_counts_one
- Multiple categorical columns: see effect_counts_items

Cross tabulations:

- One categorical column and one grouping column: see effect_counts_one_grouped
- Multiple categorical columns and one grouping column: see effect_counts_items_grouped (not yet implemented)
- Multiple categorical columns and multiple grouping columns: effect_counts_items_grouped_items (not yet implemented)

By default, if you provide two column selections, the second column is treated as categorical. Setting the metric-parameter to TRUE will call the appropriate functions for correlation analysis:

- One categorical column and one metric column: see effect_counts_one_cor (not yet implemented)
- Multiple categorical columns and one metric column: see effect_counts_items_cor (not yet implemented)
- Multiple categorical columns and multiple metric columns:effect_counts_items_cor_items (not yet implemented)

[Experimental]

Usage

```
effect_counts(data, cols, cross = NULL, metric = FALSE, clean = TRUE, ...)
```

Arguments

`data` A data frame.

`cols` A tidy column selection, e.g. a single column (without quotes) or multiple columns selected by methods such as starts_with().

`cross` Optional, a grouping column. The column name without quotes.

`metric` When crossing variables, the cross column parameter can contain categorical or metric values. By default, the cross column selection is treated as categorical data. Set metric to TRUE, to treat it as metric and calculate correlations.

`clean` Prepare data by data_clean.

`...` Other parameters passed to the appropriate effect function.

Value

A volker tibble.

Examples

```
library(volker)
data <- volker::chatgpt

effect_counts(data, sd_gender, adopter)
```

---

effect_metrics *Output effect sizes and test statistics for metric data*

Description

The calculations depend on the number of selected columns:

- One metric column: see effect_metrics_one
- Multiple metric columns: see effect_metrics_items

Group comparisons:

- One metric column and one grouping column: see effect_metrics_one_grouped
- Multiple metric columns and one grouping column: see effect_metrics_items_grouped
- Multiple metric columns and multiple grouping columns: not yet implemented

By default, if you provide two column selections, the second column is treated as categorical. Setting the metric-parameter to TRUE will call the appropriate functions for correlation analysis:

- Two metric columns: see effect_metrics_one_cor
- Multiple metric columns and one metric column: see effect_metrics_items_cor
- Two metric column selections: see effect_metrics_items_cor_items

[Experimental]

Usage

```
effect_metrics(data, cols, cross = NULL, metric = FALSE, clean = TRUE, ...)
```

Arguments

`data` A data frame.

`cols` A tidy column selection, e.g. a single column (without quotes) or multiple columns selected by methods such as starts_with().

`cross` Optional, a grouping column (without quotes).

`metric` When crossing variables, the cross column parameter can contain categorical or metric values. By default, the cross column selection is treated as categorical data. Set metric to TRUE, to treat it as metric and calculate correlations.

`clean` Prepare data by data_clean.

`...` Other parameters passed to the appropriate effect function.

Value

A volker tibble.

Examples

```
library(volker)
data <- volker::chatgpt

effect_metrics(data, sd_age, sd_gender)
```

`html_report` *Volker style HTML document format*

Description

Based on the standard theme, tweaks the pill navigation to switch between tables and plots. To use the format, in the header of your Markdown document, set `output: volker::html_report`.

Usage

```
html_report(...)
```

Arguments

`...` Additional arguments passed to html_document.

Value

R Markdown output format.

Examples

```
## Not run:
# Add `volker::html_report` to the output options of your Markdown document:
#
# ```
# ---
# title: "How to create reports?"
# output: volker::html_report
# ---
# ```

## End(Not run)
```

`labs_apply` *Set column and value labels*

Description

[Experimental]

Usage

```
labs_apply(data, codes = NULL, cols = NULL, items = TRUE, values = TRUE)
```

Arguments

`data`	A tibble containing the dataset.
`codes`	A tibble in codebook format.
`cols`	A tidy column selection. Set to NULL (default) to apply to all columns found in the codebook. Restricting the columns is helpful when you want to set value labels. In this case, provide a tibble with value_name and value_label columns and specify the columns that should be modified.
`items`	If TRUE, column labels will be retrieved from the codes (the default). If FALSE, no column labels will be changed. Alternatively, a named list of column names with their labels.
`values`	If TRUE, value labels will be retrieved from the codes (default). If FALSE, no value labels will be changed. Alternatively, a named list of value names with their labels. In this case, use the cols-Parameter to define which columns should be changed.

Details

You can either provide a data frame in codebook format to the codes-parameter or provide named lists to the items- or values-parameter.

When working with a codebook in the codes-parameter:

- Change column labels by providing the columns item_name and item_label in the codebook. Set the items-parameter to TRUE (the default setting).
- Change value labels by providing the columns value_name and value_label in the codebook. To tell which columns should be changed, you can either use the item_name column in the codebook or use the cols-parameter. For factor values, the levels and their order are retrieved from the value_label column. For coded values, labels are retrieved from both the columns value_name and value_label.

When working with lists in the items- or values-parameter:

- Change column labels by providing a named list to the items-parameter. The list contains labels named by the columns. Set the parameters codes and cols to NULL (their default value).
- Change value labels by providing a named list to the values-parameter. The list contains labels named by the values. Provide the column selection in the cols-parameter. Set the codes-parameter to NULL (its default value).

Value

A tibble containing the dataset with new labels.

Examples

```
library(volker)

# Set column labels using the items-parameter
volker::chatgpt %>%
  labs_apply(
   items = list(
     "cg_adoption_advantage_01" = "Allgemeine Vorteile",
     "cg_adoption_advantage_02" = "Finanzielle Vorteile",
     "cg_adoption_advantage_03" = "Vorteile bei der Arbeit",
     "cg_adoption_advantage_04" = "Macht mehr Spaß"
   )
 ) %>%
 tab_metrics(starts_with("cg_adoption_advantage_"))

# Set value labels using the values-parameter
 volker::chatgpt %>%
   labs_apply(
     cols=starts_with("cg_adoption"),
     values = list(
       "1" = "Stimme überhaupt nicht zu",
       "2" = "Stimme nicht zu",
       "3" = "Unentschieden",
       "4" = "Stimme zu",
       "5" =  "Stimme voll und ganz zu"
     )
   ) %>%
   plot_metrics(starts_with("cg_adoption"))
```

labs_clear *Remove all comments from the selected columns*

Description

[Experimental]

Usage

```
labs_clear(data, cols, labels = NULL)
```

Arguments

data	A tibble.
cols	Tidyselect columns.
labels	The attributes to remove. NULL to remove all attributes except levels and class.

Value

A tibble with comments removed.

Examples

```
library(volker)
volker::chatgpt |>
  labs_clear()
```

labs_restore *Restore labels from the codebook store in the codebook attribute.*

Description

[Experimental]

Usage

```
labs_restore(data, cols = NULL)
```

Arguments

data	A data frame.
cols	A tidyselect column selection.

Details

You can store labels before mutate operations by calling labs_store.

Value

A data frame.

Examples

```
library(dplyr)
library(volker)

volker::chatgpt |>
  labs_store() |>
  mutate(sd_age = 2024 - sd_age) |>
  labs_restore() |>
  tab_metrics(sd_age)
```

---

labs_store *Get the current codebook and store it in the codebook attribute.*

Description

[Experimental]

Usage

```
labs_store(data)
```

Arguments

data	A data frame.

Details

You can restore the labels after mutate operations by calling labs_restore.

Value

A data frame.

Examples

```
library(dplyr)
library(volker)

volker::chatgpt |>
  labs_store() |>
  mutate(sd_age = 2024 - sd_age) |>
  labs_restore() |>
  tab_metrics(sd_age)
```

pdf_report *Volker style PDF document format*

Description

Based on the standard theme, tweaks tex headers. To use the format, in the header of your Markdown document, set `output: volker::pdf_report`.

Usage

```
pdf_report(...)
```

Arguments

...	Additional arguments passed to pdf_document.

Value

R Markdown output format.

Examples

```
## Not run:
# Add `volker::pdf_report` to the output options of your Markdown document:
#
# ```
# ---
# title: "How to create reports?"
# output: volker::pdf_report
# ---
# ```

## End(Not run)
```

plot_counts *Output a frequency plot*

Description

The type of frequency plot depends on the number of selected columns:

- One categorical column: see plot_counts_one
- Multiple categorical columns: see plot_counts_items

Cross tabulations:

- One categorical column and one grouping column: see plot_counts_one_grouped

- Multiple categorical columns and one grouping column: see plot_counts_items_grouped
- Two categorical column selections: see plot_counts_items_grouped_items (not yet implemented)

By default, if you provide two column selections, the second selection is treated as categorical. Setting the metric-parameter to TRUE will call the appropriate functions for correlation analysis:

- One categorical column and one metric column: see plot_counts_one_cor
- Multiple categorical columns and one metric column: see plot_counts_items_cor
- Multiple categorical columns and multiple metric columns: see plot_counts_items_cor_items (not yet implemented)

Parameters that may be passed to the count functions (see the respective function help):

- **ci**: Add confidence intervals to proportions.
- **ordered**: The values of the cross column can be nominal (0), ordered ascending (1), or ordered descending (-1). The colors are adjusted accordingly.
- **category**: When you have multiple categories in a column, you can focus one of the categories to simplify the plots. By default, if a column has only TRUE and FALSE values, the outputs focus the TRUE category.
- **prop**: For stacked bar charts, displaying row percentages instead of total percentages gives a direct visual comparison of groups.
- **limits**: The scale limits are automatically guessed by the package functions (work in progress). Use the limits-parameter to manually fix any misleading graphs.
- **title**: All plots usually get a title derived from the column attributes or column names. Set to FALSE to suppress the title or provide a title of your choice as a character value.
- **labels**: Labels are extracted from the column attributes. Set to FALSE to output bare column names and values.
- **numbers**: Set the numbers parameter to "n" (frequency), "p" (percentage) or c("n","p"). To prevent cluttering and overlaps, numbers are only plotted on bars larger than 5%.
- **width**: When comparing groups by row of column percentages, by default, the bar or column width reflects the number of cases. You can disable this behavior by setting width to FALSE.

[Experimental]

Usage

```
plot_counts(data, cols, cross = NULL, metric = FALSE, clean = TRUE, ...)
```

Arguments

`data`	A data frame.
`cols`	A tidy column selection, e.g. a single column (without quotes) or multiple columns selected by methods such as starts_with().
`cross`	Optional, a grouping column. The column name without quotes.

`metric`	When crossing variables, the cross column parameter can contain categorical or metric values. By default, the cross column selection is treated as categorical data. Set metric to TRUE, to treat it as metric and calculate correlations.
`clean`	Prepare data by data_clean.
`...`	Other parameters passed to the appropriate plot function.

Value

A ggplot2 plot object.

Examples

```
library(volker)
data <- volker::chatgpt

plot_counts(data, sd_gender)
```

`plot_metrics` *Output a plot with distribution parameters such as the mean values*

Description

The plot type depends on the number of selected columns:

- One metric column: see plot_metrics_one
- Multiple metric columns: see plot_metrics_items

Group comparisons:

- One metric column and one grouping column: see plot_metrics_one_grouped
- Multiple metric columns and one grouping column: see plot_metrics_items_grouped
- Multiple metric columns and multiple grouping columns: see plot_metrics_items_grouped_items (not yet implemented)

By default, if you provide two column selections, the second selection is treated as categorical. Setting the metric-parameter to TRUE will call the appropriate functions for correlation analysis:

- Two metric columns: see plot_metrics_one_cor
- Multiple metric columns and one metric column : see plot_metrics_items_cor
- Two metric column selections: see plot_metrics_items_cor_items

Parameters that may be passed to the metric functions (see the respective function help):

- **ci**: Plot confidence intervals for means or correlation coefficients.
- **box**: Visualise the distribution by adding boxplots.
- **log**: In scatter plots, you can use a logarithmic scale. Be aware, that zero values will be omitted because their log value is undefined.

- **method**: By default, correlations are calculated using Pearson's R. You can choose Spearman's Rho with the methods-parameter.
- **limits**: The scale limits are automatically guessed by the package functions (work in progress). Use the limits-parameter to manually fix any misleading graphs.
- **title**: All plots usually get a title derived from the column attributes or column names. Set to FALSE to suppress the title or provide a title of your choice as a character value.
- **labels**: Labels are extracted from the column attributes. Set to FALSE to output bare column names and values.
- **numbers**: Controls whether to display correlation coefficients on the plot.

[Experimental]

Usage

```
plot_metrics(data, cols, cross = NULL, metric = FALSE, clean = TRUE, ...)
```

Arguments

`data` A data frame.

`cols` A tidy column selection, e.g. a single column (without quotes) or multiple columns selected by methods such as starts_with().

`cross` Optional, a grouping column (without quotes).

`metric` When crossing variables, the cross column parameter can contain categorical or metric values. By default, the cross column selection is treated as categorical data. Set metric to TRUE, to treat it as metric and calculate correlations.

`clean` Prepare data by data_clean.

`...` Other parameters passed to the appropriate plot function.

Value

A ggplot object.

Examples

```
library(volker)
data <- volker::chatgpt

plot_metrics(data, sd_age)
```

report_counts *Create table and plot for categorical variables*

Description

Depending on your column selection, different types of plots and tables are generated. See plot_counts and tab_counts.

Usage

```
report_counts(
  data,
  cols,
  cross = NULL,
  metric = FALSE,
  index = FALSE,
  effect = FALSE,
  numbers = NULL,
  title = TRUE,
  close = TRUE,
  clean = TRUE,
  ...
)
```

Arguments

`data` A data frame.

`cols` A tidy column selection, e.g. a single column (without quotes) or multiple columns selected by methods such as starts_with().

`cross` Optional, a grouping column (without quotes).

`metric` When crossing variables, the cross column parameter can contain categorical or metric values. By default, the cross column selection is treated as categorical data. Set metric to TRUE, to treat it as metric and calculate correlations.

`index` When the cols contain items on a metric scale (as determined by get_direction), an index will be calculated using the 'psych' package. Set to FALSE to suppress index generation.

`effect` Whether to report statistical tests and effect sizes. See effect_counts for further parameters.

`numbers` The numbers to print on the bars: "n" (frequency), "p" (percentage) or both. Set to NULL to remove numbers.

`title` A character providing the heading or TRUE (default) to output a heading. Classes for tabset pills will be added.

`close` Whether to close the last tab (default value TRUE) or to keep it open. Keep it open to add further custom tabs by adding headers on the fifth level in Markdown (e.g. ##### Method).

`clean` Prepare data by data_clean.

`...` Parameters passed to the plot_counts and tab_counts and effect_counts functions.

Details

For item batteries, an index is calculated and reported. When used in combination with the Markdown-template "html_report", the different parts of the report are grouped under a tabsheet selector.

[Experimental]

Value

A volker report object.

Examples

```
library(volker)
data <- volker::chatgpt

report_counts(data, sd_gender)
```

`report_metrics` *Create table and plot for metric variables*

Description

Depending on your column selection, different types of plots and tables are generated. See plot_metrics and tab_metrics.

Usage

```
report_metrics(
  data,
  cols,
  cross = NULL,
  metric = FALSE,
  ...,
  index = FALSE,
  factors = FALSE,
  clusters = FALSE,
  effect = FALSE,
  title = TRUE,
  close = TRUE,
  clean = TRUE
)
```

Arguments

`data` A data frame.

`cols` A tidy column selection, e.g. a single column (without quotes) or multiple columns selected by methods such as starts_with().

`cross` Optional, a grouping or correlation column (without quotes).

`metric` When crossing variables, the cross column parameter can contain categorical or metric values. By default, the cross column selection is treated as categorical data. Set metric to TRUE, to treat it as metric and calculate correlations.

`...` Parameters passed to the plot_metrics and tab_metrics and effect_metrics functions.

`index` When the cols contain items on a metric scale (as determined by get_direction), an index will be calculated using the 'psych' package. Set to FALSE to suppress index generation.

`factors` The number of factors to calculate. Set to FALSE to suppress factor analysis. Set to TRUE to output a scree plot and automatically choose the number of factors. When the cols contain items on a metric scale (as determined by get_direction), factors will be calculated using the 'psych' package. See add_factors.

`clusters` The number of clusters to calculate. Cluster are determined using kmeans after scaling the items. Set to FALSE to suppress cluster analysis. Set to TRUE to output a scree plot and automatically choose the number of clusters based on the elbow criterion. See add_clusters.

`effect` Whether to report statistical tests and effect sizes. See effect_counts for further parameters.

`title` A character providing the heading or TRUE (default) to output a heading. Classes for tabset pills will be added.

`close` Whether to close the last tab (default value TRUE) or to keep it open. Keep it open to add further custom tabs by adding headers on the fifth level in Markdown (e.g. ##### Method).

`clean` Prepare data by data_clean.

Details

For item batteries, an index is calculated and reported. When used in combination with the Markdown-template "html_report", the different parts of the report are grouped under a tabsheet selector.

[Experimental]

Value

A volker report object.

Examples

```
library(volker)
data <- volker::chatgpt

report_metrics(data, sd_age)
```

`tab_counts` *Output a frequency table*

Description

The type of frequency table depends on the number of selected columns:

- One categorical column: see tab_counts_one
- Multiple categorical columns: see tab_counts_items

Cross tabulations:

- One categorical column and one grouping column: see tab_counts_one_grouped
- Multiple categorical columns and one grouping column: see tab_counts_items_grouped
- Multiple categorical columns and multiple grouping columns: see tab_counts_items_grouped_items (not yet implemented)

By default, if you provide two column selections, the second column is treated as categorical. Setting the metric-parameter to TRUE will call the appropriate functions for correlation analysis:

- One categorical column and one metric column: see tab_counts_one_cor
- Multiple categorical columns and one metric column: see tab_counts_items_cor
- Multiple categorical columns and multiple metric columns: tab_counts_items_cor_items (not yet implemented)

Parameters that may be passed to specific count functions:

- **ci**: Add confidence intervals to proportions.
- **percent**: Frequency tables show percentages by default. Set to FALSE to get raw proportions.
- **prop**: For cross tables you can choose between total, row or column percentages.
- **values**: The values to output: n (frequency) or p (percentage) or both (the default).
- **category**: When you have multiple categories in a column, you can focus one of the categories to simplify the plots. By default, if a column has only TRUE and FALSE values, the outputs focus the TRUE category.
- **labels**: Labels are extracted from the column attributes. Set to FALSE to output bare column names and values.

[Experimental]

Usage

```
tab_counts(data, cols, cross = NULL, metric = FALSE, clean = TRUE, ...)
```

Arguments

`data` A data frame.

`cols` A tidy column selection, e.g. a single column (without quotes) or multiple columns selected by methods such as starts_with().

`cross` Optional, a grouping column. The column name without quotes.

`metric` When crossing variables, the cross column parameter can contain categorical or metric values. By default, the cross column selection is treated as categorical data. Set metric to TRUE, to treat it as metric and calculate correlations.

`clean` Prepare data by data_clean.

`...` Other parameters passed to the appropriate table function.

Value

A volker tibble.

Examples

```
library(volker)
data <- volker::chatgpt

tab_counts(data, sd_gender)
```

`tab_metrics` *Output a table with distribution parameters*

Description

The table type depends on the number of selected columns:

- One metric column: see tab_metrics_one
- Multiple metric columns: see tab_metrics_items

Group comparisons:

- One metric column and one grouping column: see tab_metrics_one_grouped
- Multiple metric columns and one grouping column: see tab_metrics_items_grouped
- Multiple metric columns and multiple grouping columns: see tab_metrics_items_grouped_items (not yet implemented)

By default, if you provide two column selections, the second column is treated as categorical. Setting the metric-parameter to TRUE will call the appropriate functions for correlation analysis:

- Two metric columns: see tab_metrics_one_cor
- Multiple metric columns and one metric column: see tab_metrics_items_cor
- Two metric column selections: see tab_metrics_items_cor_items

Parameters that may be passed to specific metric functions:

- **ci**: Add confidence intervals for means or correlation coefficients.
- **values**: The output metrics, mean (m), the standard deviation (sd) or both (the default).
- **digits**: Tables containing means and standard deviations by default round values to one digit. Increase the number to show more digits
- **method**: By default, correlations are calculated using Pearson's R. You can choose Spearman's Rho with the methods-parameter.
- **labels**: Labels are extracted from the column attributes. Set to FALSE to output bare column names and values.

[Experimental]

Usage

```
tab_metrics(data, cols, cross = NULL, metric = FALSE, clean = TRUE, ...)
```

Arguments

`data`	A data frame.
`cols`	A tidy column selection, e.g. a single column (without quotes) or multiple columns selected by methods such as starts_with().
`cross`	Optional, a grouping column (without quotes).
`metric`	When crossing variables, the cross column parameter can contain categorical or metric values. By default, the cross column selection is treated as categorical data. Set metric to TRUE, to treat it as metric and calculate correlations.
`clean`	Prepare data by data_clean.
`...`	Other parameters passed to the appropriate table function.

Value

A volker tibble.

Examples

```
library(volker)
data <- volker::chatgpt

tab_metrics(data, sd_age)
```

theme_vlkr *Define a default theme for volker plots*

Description

Set ggplot colors, sizes and layout parameters.

Usage

```
theme_vlkr(
  base_size = 11,
  base_color = "black",
  base_fill = VLKR_FILLDISCRETE,
  base_gradient = VLKR_FILLGRADIENT
)
```

Arguments

`base_size`	Base font size.
`base_color`	Base font color.
`base_fill`	A list of fill color sets or at least one fill color set. Example: `list(c("red"), c("red", "blue", "green"))`. Each set can contain different numbers of colors. Depending on the number of colors needed, the set with at least the number of required colors is used. The first color is always used for simple bar charts.
`base_gradient`	A color vector used for creating gradient fill colors, e.g. in stacked bar plots.

Details

[Experimental]

Value

A theme function.

Examples

```
library(volker)
library(ggplot2)
data <- volker::chatgpt

theme_set(theme_vlkr(base_size=15, base_fill = list("red")))
plot_counts(data, sd_gender)
```

Index