Package 'sampcompR'

July 4, 2025

Title Comparing and Visualizing Differences Between Surveys

Version 0.3.1.2

Description
Easily analyze and visualize differences between samples (e.g., benchmark comparisons, nonresponse comparisons in surveys) on three levels. The comparisons can be univariate, bivariate or multivariate. On univariate level the variables of interest of a survey and a comparison survey (i.e. benchmark) are compared, by calculating one of several difference measures (e.g., relative difference in mean), and an average difference between the surveys. On bivariate level a function can calculate significant differences in correlations for the surveys. And on multivariate levels a function can calculate significant differences in model coefficients between the surveys of comparison. All of those differences can be easily plotted and outputted as a table. For more detailed information on the methods and example use see Rohr, B., Silber, H., & Felderer, B. (2024). Comparing the Accuracy of Univariate, Bivariate, and Multivariate Estimates across Probability and Nonprobability Surveys with Population Benchmarks. Sociological Methodology <doi:10.1177/00811750241280963>.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 4.1.0)

Imports boot, data.table, dplyr, forcats, furrr, future, ggplot2, Hmisc, lmtest, magrittr, psych, purrr, readr, reshape2, sandwich, stats, survey, svrep, tibble, tidyr, rlang

Suggests testthat (>= 3.0.0), jtools, utils, parallel, stargazer

Config/testthat/edition 3

URL https://bjoernrohr.github.io/sampcompR/

NeedsCompilation no

Author Bjoern Rohr [aut, cre, cph], Barbara Felderer [aut]

Maintainer Bjoern Rohr <bjoern.rohr@gesis.org>

Repository CRAN

Date/Publication 2025-07-04 09:00:02 UTC

Contents

`biv_bias_per_variable` *Returns a table based on the information of a* `biv_compare_object` *that indicates the Average Absolute Bias (AARB) in Pearson's r or the Average Absolute Relative Bias (AARB) in Pearson's r for every data frame It can be outputted as HTML or LaTex Table, for example with the help of the stargazer function.*

Description

Returns a table based on the information of a `biv_compare_object` that indicates the Average Absolute Bias (AARB) in Pearson's r or the Average Absolute Relative Bias (AARB) in Pearson's r for every data frame It can be outputted as HTML or LaTex Table, for example with the help of the stargazer function.

Usage

```
biv_bias_per_variable(
  biv_compare_object,
  type = "rel_diff",
  final_col = "difference",
  ndigits = 3,
  varlabels = NULL,
  label_df = NULL
)
```

Arguments

biv_compare_object	A object returned by the biv_compare function.
type	A character string, which is "AAB" if the Average Absolute Bias per variable should be displayed in the table, or "AARB" if the Average Absolute Relative Bias per Variable should be displayed in the table.
final_col	A character string, indicating if the last column of the table should display an average bias per variable of over all data frames ("average"), or the difference between the first and the average bias of the first and the last data frame ("difference").
ndigits	Number of digits that is shown in the table.
varlabels	A character vector containing labels for the variables.
label_df	A character vector containing labels for the data frames.

Value

A matrix, that shows the Average Absolute Bias (AAB) or the Average Absolute Relative Bias (AARB) for every individual variable. This is given separately for every comparison data frame, as well as averaged over comparisons, or as the difference between the first and the last comparison.

Examples

```
data("card")

north <- card[card$south==0,]
white <- card[card$black==0,]

## use the function to plot the data
bivar_data<-sampcompR::biv_compare(dfs = c("north","white"),
                                   benchmarks = c("card","card"),
                              variables= c("age","educ","fatheduc","motheduc","wage","IQ"),
                                   data=TRUE)

table1<-sampcompR::biv_bias_per_variable(bivar_data,type="rel_diff",
                                         final_col="average",ndigits=2)
noquote(table1)

table2<-sampcompR::biv_bias_per_variable(bivar_data,type = "diff",
                                         final_col="difference",ndigits=2)
noquote(table2)
```

---

biv_compare *Compare Multiple Data Frames on a Bivariate Level*

Description

Compare multiple data frames on a bivariate level and plot them together.

Usage

```
biv_compare(
  dfs,
  benchmarks,
  variables = NULL,
  corrtype = "r",
  data = TRUE,
  id = NULL,
  weight = NULL,
  strata = NULL,
  id_bench = NULL,
  weight_bench = NULL,
  strata_bench = NULL,
  p_value = NULL,
  p_adjust = NULL,
  varlabels = NULL,
  plot_title = NULL,
  plots_label = NULL,
  diff_perc = TRUE,
  diff_perc_size = 4.5,
  perc_diff_transparance = 0,
  note = FALSE,
  order = NULL,
  breaks = NULL,
  colors = NULL,
  mar = c(0, 0, 0, 0),
  grid = "white",
  gradient = FALSE,
  sum_weights = NULL,
  missings_x = TRUE,
  remove_nas = "pairwise",
  ncol_facet = 3,
  nboots = 0,
  boot_all = FALSE,
  parallel = FALSE,
  adjustment_weighting = "raking",
  adjustment_vars = NULL,
  raking_targets = NULL,
  post_targets = NULL,
  percentile_ci = TRUE
)
```

Arguments

dfs	A character vector containing the names of data frames to compare against the benchmarks.
benchmarks	A character vector containing the names of benchmarks to compare the dfs against, or the names of a list. If it is a list, it has to be of the form, as the output

of rcorr, with a Pearson's r matrix in the first position, a n-matrix (matrix of n for every correlation) in the second position and a p-matrix in the third position. The vector must either be the same length as dfs, or length 1. If it has length one every survey will be compared against the same benchmark.

variables — A character vector that containes the names of the variables for the comparison. If it is NULL, all variables that are named similarly in both the dfs and the benchmarks will be compared. Variables missing in one of the dfs or the benchmarks will be neglected for this comparison.

corrtype — A character string, indicating the type of the bivariate correlation. It can either be "r" for Pearson's r or "rho" for Spearman's "rho". At the moment, rho is only applicable to unweighted data.

data — If TRUE, a biv_compare object is returned, containing the results of the comparison.

strata, strata_bench — A character vector that determines strata variables that are used to weigh the dfs or benchmarks with the help of the survey package. It has to be part of the respective data frame. If fewer characters strings are provided, than in dfs, the first input is used to weigh every df or benchmark, where no input is provided.

id_bench, id — A character vector determining id variables used to weigh the dfs or benchmarks with the help of the survey package. They have to be part of the respective data frame. If less characters strings are provided, than in dfs, the first input is used to weigh every df or benchmark, where no input is provided.

weight_bench, weight — A character vector that determines variables to weigh the dfs of benchmarks. They have to be part of the respective data frame. If fewer characters strings are provided, than in dfs, the first input is used to weigh every df or benchmark, where no input is provided. If a weight variable is provided also an id variable is needed. For weighting, the survey package is used.

p_value — A number between zero and one to determine the maximum significance niveau.

p_adjust — Can be either TRUE or a character string indicating an adjustment method. If p_adjust = TRUE the p_values will be adjusted with the Bonferroni adjustment method, by default, to account for the problem of multiple comparisons. All adjustment methods available in p.adjust can be used here, with the same character strings.

varlabels — A character string or vector of character strings containing the new names of variables that is used in the plot.

plot_title — A character string containing the title of the plot.

plots_label — A character string or vector of character strings containing the new names of the data frames that are used in the plot.

diff_perc — If TRUE a percental difference between surveys and benchmarks is displayed in the plot.

diff_perc_size — A number to determine the size of the displayed percental difference between surveys in the plot.

perc_diff_transparance — A number to determine the transparency of the displayed percental difference between surveys in the plot.

note	If note = TRUE, a note will be displayed to describe the plot.
order	A character vector to determine in which order the variables should be displayed in the plot.
breaks	A vector to label the color scheme in the legend.
colors	A vector to determine the colors in the plot.
mar	A vector that determines the margins of the plot.
grid	A color string, that determines the color of the lines between the tiles of the heatmap.
gradient	If gradient = TRUE, colors in the heatmap will be more or less transparent, depending on the difference in Pearson's r of the data frames of comparison.
sum_weights	A vector containing information for every variable to weigh them in the displayed percental-difference calculation. It can be used if some variables are over- or underrepresented in the analysis.
missings_x	If TRUE, missing pairs in the plot will be marked with an X.
remove_nas	A character string, that indicates how missing values should be removed, can either be "all", to remove all cases that contain NA in any of the variables, or "pairwise", to remove NAs separately for every variable pair when calculating Pearson's r.
ncol_facet	The number of columns used in faced_wrap() for the plots.
nboots	A numeric value indicating the number of bootstrap replications. If nboots = 0 no bootstrapping will be performed. Else nboots must be >2. Note, that bootstrapping can be very computationally heavy and can therefore take a while.
boot_all	If TURE, both, dfs and benchmarks will be bootstrapped. Otherwise the benchmark estimate is assumed to be constant.
parallel	Can be either FALSE or a number of cores that should be used in the function. If it is FALSE, only one core will be used and otherwise the given number of cores will be used.
adjustment_weighting	A character vector indicating if adjustment weighting should be used. It can either be "raking" or "post_start".
adjustment_vars	Variables used to adjust the survey when using raking or post-stratification.
raking_targets	A list of raking targets that can be given to the rake function of rake, to rake the dfs.
post_targets	A list of post_stratification targets that can be given to the postStratify function, to post-stratify the dfs.
percentile_ci	If TURE, cofidence intervals will be calculated using the percentile method. If False, they will be calculated using the normal method.

Details

The plot shows a heatmap of a correlation matrix, where the colors are determined by the similarity of the Pearson's r values in both sets of respondents. Leaving default breaks and colors,

- Same (green) indicates, that the Pearson's r correlation is not significant > 0 in the related data frame or benchmark or the Pearson's r correlations are not significantly different, between data frame and benchmark.
- Small Diff (yellow) indicates that the Pearson's r correlation is significant > 0 in the related data frame or benchmark and the Pearson's r correlations are significantly different, between data frame and benchmark.
- Large Diff (red) indicates, that the same conditions of yellow are fulfilled, and the correlations are either in opposite directions,or one is double the size of the other.

Value

A object generated with the help of ggplot2::ggplot2() visualizes the differences between the data frames and benchmarks. If data = TRUE instead of the plot a list will be returned containing information of the analyses. This biv_compare object can be used in plot_biv_compare to build a plot, or in biv_compare_table, to get a table.

Examples

```
## Get Data for comparison

data("card")

north <- card[card$south==0,]
white <- card[card$black==0,]

## use the function to plot the data
bivar_comp<-sampcompR::biv_compare(dfs = c("north","white"),
                                   benchmarks = c("card","card"),
                               variables= c("age","educ","fatheduc","motheduc","wage","IQ"),
                                   data=FALSE)
bivar_comp
```

`biv_compare_table` *Returns a table based on the information of a* `biv_compare_object` *which can be outputted as HTML or LaTex Table, for example with the help of the stargazer function.*

Description

Returns a table based on the information of a biv_compare_object which can be outputted as HTML or LaTex Table, for example with the help of the stargazer function.

Usage

```
biv_compare_table(
  biv_compare_object,
  type = "diff",
  comparison_number = 1,
  ndigits = 2
)
```

Arguments

biv_compare_object	A object returned by the biv_compare function.
type	A character string, to choose what matrix should be printed. • If "dfs", a correlation matrix of all variables of comparison in the chosen dataframe will be returned. • If "benchmarks", a correlation matrix of all variables of comparison in the chosen benchmark will be returned. • if "diff", a matrix indicating the difference between the chosen dataframe and benchmark will be returned.
comparison_number	A number indicating the data of which data frame, benchmark or comparison should be displayed. The maximum length is equal to the length of the dfs vector that is used to generate the biv_compare_object.
ndigits	Number of digits shown in the table.

Value

A correlation matrix, or difference matrix based on information of a biv_compare_object.

Examples

```
## Get Data for comparison

data("card")

north <- card[card$south==0,]
white <- card[card$black==0,]

## use the function to plot the data
bivar_data<-sampcompR::biv_compare(dfs = c("north","white"),
                                   benchmarks = c("card","card"),
                               variables= c("age","educ","fatheduc","motheduc","wage","IQ"),
                                   data=TRUE)

table<-sampcompR::biv_compare_table(bivar_data, type="diff", comparison_number=1)
noquote(table)
```

`biv_per_variable` *Returns a table based on the information of a* `biv_compare_object` *that indicates the proportion of biased variables. It can be outputted as HTML or LaTex Table, for example with the help of the stargazer function.*

Description

Returns a table based on the information of a `biv_compare_object` that indicates the proportion of biased variables. It can be outputted as HTML or LaTex Table, for example with the help of the stargazer function.

Usage

```
biv_per_variable(
  biv_compare_object,
  ndigits = 1,
  varlabels = NULL,
  label_df = NULL
)
```

Arguments

`biv_compare_object`	A object returned by the `biv_compare` function.
`ndigits`	Number of digits that is shown in the table.
`varlabels`	A character vector containing labels for the variables.
`label_df`	A character vector containing labels for the data frames.

Value

A matrix, that indicates the proportion of bias for every individual variable. This is given separately for every comparison, as well as averaged over comparisons.

Examples

```
data("card")

north <- card[card$south==0,]
white <- card[card$black==0,]

## use the function to plot the data
bivar_data<-sampcompR::biv_compare(dfs = c("north","white"),
                                   benchmarks = c("card","card"),
                               variables= c("age","educ","fatheduc","motheduc","wage","IQ"),
                                   data=TRUE)
```

```
table<-sampcompR::biv_per_variable(bivar_data)
noquote(table)
```

`card` *card*

Description

This data, which originates from D. Card (1995) was released in the Wooldridge R-Package. Sadly the wooldridge package (Shea 2023) was archived on CRAN on the 3rd of December 2024. As we use it, e.g., in our examples to show how our package works, we also added it to our package, so we can further use it. Further we cite the original description of the wooldrigde package. Wooldridge Source: D. Card (1995), Using Geographic Variation in College Proximity to Estimate the Return to Schooling, in Aspects of Labour Market Behavior: Essays in Honour of John Vanderkamp. Ed. L.N. Christophides, E.K. Grant, and R. Swidinsky, 201-222. Toronto: University of Toronto Press. Professor Card kindly provided these data. Data loads lazily.

Usage

```
data('card')
```

Format

A data.frame with 3010 observations on 34 variables:

- **id:** person identifier
- **nearc2:** =1 if near 2 yr college, 1966
- **nearc4:** =1 if near 4 yr college, 1966
- **educ:** years of schooling, 1976
- **age:** in years
- **fatheduc:** father's schooling
- **motheduc:** mother's schooling
- **weight:** NLS sampling weight, 1976
- **momdad14:** =1 if live with mom, dad at 14
- **sinmom14:** =1 if with single mom at 14
- **step14:** =1 if with step parent at 14
- **reg661:** =1 for region 1, 1966
- **reg662:** =1 for region 2, 1966
- **reg663:** =1 for region 3, 1966
- **reg664:** =1 for region 4, 1966
- **reg665:** =1 for region 5, 1966
- **reg666:** =1 for region 6, 1966

- **reg667:** =1 for region 7, 1966
- **reg668:** =1 for region 8, 1966
- **reg669:** =1 for region 9, 1966
- **south66:** =1 if in south in 1966
- **black:** =1 if black
- **smsa:** =1 in in SMSA, 1976
- **south:** =1 if in south, 1976
- **smsa66:** =1 if in SMSA, 1966
- **wage:** hourly wage in cents, 1976
- **enroll:** =1 if enrolled in school, 1976
- **KWW:** knowledge world of work score
- **IQ:** IQ score
- **married:** =1 if married, 1976
- **libcrd14:** =1 if lib. card in home at 14
- **exper:** age - educ - 6
- **lwage:** log(wage)
- **expersq:** exper^2

Notes

Computer Exercise C15.3 is important for analyzing these data. There, it is shown that the instrumental variable, `nearc4`, is actually correlated with `IQ`, at least for the subset of men for which an IQ score is reported. However, the correlation between `nearc4`` and IQ, once the other explanatory variables are net fails the exogeneity requirement in a simple regression model but it passes, at least using the crude test described above, if controls are added to the wage equation. For a more advanced course, a nice extension of Card's analysis is to allow the return to education to differ by race. A relatively simple extension is to include black education (blackeduc) as an additional explanatory variable; its natural instrument is blacknearc4.

Used in Text: pages 526-527, 547

Source

https://www.cengage.com/cgi-wadsworth/course_products_wp.pl?fid=M20b&product_isbn_issn=9781111531041

References

Shea J (2023). *wooldridge: 115 Data Sets from "Introductory Econometrics: A Modern Approach, 7e" by Jeffrey M. Wooldridge*. R package version 1.4-3, https://CRAN.R-project.org/package=wooldridge.

Examples

```
 data("card")
str(card)
```

dataequalizer *Equalize dataframes*

Description

dataequalizer compares two data frames and looks if both data frames contain columns with the same Name. A copy of source_df is returned, containing only columns named identical in target_df and source_df data frames. The function is mainly used in the other functions of the package.

Usage

```
dataequalizer(target_df, source_df, variables = NULL, silence = FALSE)
```

Arguments

target_df	A data frame
source_df	A data frame containing some column-names named equally in target_df
variables	A vector to indicate variable names that should be in the copy of the source_df if they are also in the target_df.
silence	A logic value. If FALSE, warnings will be returned indicating, what variables where removed, from the survey.

Value

Returns a copy of source_df containing only variables with names contained also in the target_df data frame.

Examples

```
## Get Data to equalize
data("card")

##reduce data frame
card2<-card[c("id","age","educ","fatheduc","motheduc","IQ","wage")]

card_equalized<-sampcompR::dataequalizer(card2,card,variables=c("age","educ","IQ","wage"))
card_equalized[1:20,]
```

`descriptive_table` *Get a Descriptive Table for Every Data Frame*

Description

Get a Descriptive Table for every Data Frame, to easy document your Data

Usage

```
descriptive_table(
  dfs,
  variables,
  varlabels = NULL,
  weight = NULL,
  strata = NULL,
  id = NULL,
  value = "mean",
  digits = 3
)
```

Arguments

`dfs`	A character vector, containing the names of the data frames.
`variables`	A character vector containing the variables in the data frame that should be described.
`varlabels`	A character vector containing the Labels for every variable in variables.
`weight`	A character vector, containing either the name of a weight in the respective data frame, or NA, if no weighting should be performed for this data frame.
`strata`	A character vector, containing either the name of a strata in the respective data frame, or NA, if no strata should be used when weighting this data frame.
`id`	A character vector, containing either the name of a id in the respective data frame, or NA, if every row is unique for this data frame.
`value`	A character vector indicating what descriptive value should be displayed for the data frame. It can either be "mean", "percent", "total", or "total_percent".
`digits`	A numeric value indicating the number of digits that the Descriptive table should be rounded to.

Value

Returns a matrix of Descriptive information. Output depends on value.

`heatmap_biv_compare` *Plot Difference or Relative Difference in Pearson's r for Multiple Data Frames*

Description

Plot a object generated by biv_compare function as a heatmap.

Usage

```
heatmap_biv_compare(
  biv_data_object,
  value = "AAB",
  summet_transparance = 0,
  summetric = TRUE,
  summet_size = 4.5,
  ndigits_summet = 3,
  upper_limit = NULL,
  lower_limit = NULL,
  corr_size = 3,
  ndigits_number = 2,
  varlabels = NULL,
  plots_label = NULL,
  grid = "white",
  colors = c("#8ECCEE", "#1F45F9"),
  number_color = "white",
  ncol_facet = 3,
  legend_title = NULL,
  interest_breaks = NULL,
  interest_labels = NULL,
  plot_title = NULL
)
```

Arguments

`biv_data_object`
: A object generated by the biv_compare function.

`value`
: A character string which is either `"AAB"` or `"AARB"`. AAB means that the Absolute Difference in Pearson's r estimates between dfs and the benchmarks should be displayed in the tiles by number and color. AARB means that the Absolute Relative Difference should be displayed instead.

`summet_transparance`
: A number to determine the transparency of the displayed `summetric`.

`summetric`
: If `TRUE` Average Absolute Difference (AAB) and the Average Absolute Relative Difference (AARB) of Pearson's r values between the `dfs` and the `benchmarks` is displayed in the plot.

summet_size A number to determine the size of the displayed summetric in the plot.

ndigits_summet The maximum number of digits for numbers displayed in the summertic of the plot.

upper_limit, lower_limit
A numeric value, indicating the highest or lowest value that should be displayed in the tiles by number and color. This does not affect the summetric. However, it can be used to keep differences between values visible in the heatmap, even in the presence of strong ouliers.

corr_size The font size of correlation numbers displayed in the tiles of the heatmap.

ndigits_number The maximum digits of numbers displyed in the tiles of the heatmap.

varlabels A character string or vector of character strings containing the new labels of variables that are used in the plot.

plots_label A character string or vector of character strings containing the new labels of the data frames that are used in the plot.

grid A character string, that determines the color of the lines between the tiles of the heatmap.

colors A vector of two colors used in the heatmap.

number_color A character string indicating the color of the numbers, displayed in the tiles.

ncol_facet Number of columns used in faced_wrap() for the plots.

legend_title A character string indicating the title of the legend of the plot.

interest_breaks
A numeric vector indicating the breaks for the color scheme displayed in the legend of the heatmap.

interest_labels
A character vector indicating the labels for the breaks displayed in the legend of the heatmap.

plot_title A character string containing the title of the plot.

Details

The plot shows a heatmap of a correlation matrix, where the colors are determined by the Absolue Difference or the Absolute Relative Difference in Pearson's r estimates between the data frames and the benchmarks.

Value

A object generated with the help of ggplot2::ggplot2(), used to visualize a heatmap of the bivariate differences between the data frames and benchmarks.

Examples

```
## Get Data for comparison

data("card")

north <- card[card$south==0,]
```

```
white <- card[card$black==0,]

## use the function to plot the data
bivar_data<-sampcompR::biv_compare(dfs = c("north","white"),
                                   benchmarks = c("card","card"),
                              variables= c("age","educ","fatheduc","motheduc","wage","IQ"),
                                   data=TRUE)

Absolute_Bias_Plot<-sampcompR::heatmap_biv_compare(bivar_data,value = "AAB")
Absolute_Bias_Plot

Absolute_Relative_Bias_Plot<-sampcompR::heatmap_biv_compare(bivar_data,value = "AARB")
Absolute_Relative_Bias_Plot
```

`missing_table` *Returns a Table indicating the number and proportion of NA values for a selected set of variables.*

Description

Returns a Table indicating the number and proportion of NA values for a selected set of variables.

Usage

```
missing_table(dfs, variables, df_names = NULL, varlabels = NULL)
```

Arguments

`dfs`	A character vector with names of data frames for which the missings per variable should be displayed.
`variables`	A character vector of variable names for which the missings should be displyed.
`df_names`	Either Null or a character vector of names, to relabel the data frames in the table with.
`varlabels`	Either Null, or a character vector of variable names, to relabel the variables in the table with.

Value

Returns a Table indicating the number and proportion of NA values for a selected set of variables. This can be used to get an overview of the data, detect errors after data rangeling, or find items in a survey, with especially, high item nonresponse.

Examples

```
## Get Data for comparison

data("card")

north <- card[card$south==0,]
white <- card[card$black==0,]

variables<- c("age","educ","fatheduc","motheduc","wage","IQ")
varlabels<-c("Age","Education","Father's Education",
             "Mother's Education","Wage","IQ")

missing_tab<-sampcompR::missing_table(dfs = c("north","white"),
                                      variables=variables,
                                      df_names = c("North","White"),
                                      varlabels=varlabels)

missing_tab
```

`multi_compare` *Compares data frames using different regression methods.*

Description

`multi_compare` compares data frames using regression models based on differing methods. All `glm` Models can be compared.

Usage

```
multi_compare(
  df,
  benchmark,
  independent = NULL,
  dependent = NULL,
  formula_list = NULL,
  family = "ols",
  rm_na = "pairwise",
  out_output_list = TRUE,
  out_df = FALSE,
  out_models = FALSE,
  print_p = FALSE,
  print_se = FALSE,
  weight = NULL,
  id = NULL,
  strata = NULL,
```

```
  nest = FALSE,
  weight_bench = NULL,
  id_bench = NULL,
  strata_bench = NULL,
  nest_bench = FALSE,
  robust_se = FALSE,
  p_adjust = NULL,
  names_df_benchmark = NULL,
  silence_summary = FALSE,
  nboots = 0,
  boot_all = FALSE,
  parallel = FALSE,
  adjustment_vars = NULL,
  raking_targets = NULL,
  post_targets = NULL,
  percentile_ci = TRUE
)
```

Arguments

df, benchmark	A data frame containing the set of respondents or benchmark set of respondents to compare, or a character string containing the name of the set of respondents or benchmark set of respondents. All independent and dependent variables must be inside both data frames.
independent	A list of strings containing the independent variables (x) for comparison. Every independent variable will be used in every model to estimate the dependent variable (y). When a formula_list is provided, independent will be ignored.
dependent	A list of strings containing the dependent variables (y) for comparison. One model will be computed for every dependent variable (y) provided. When a formula_list is provided, dependent will be ignored.
formula_list	A list of formulas to use in the regression models. If given, dependent and independent parameters will be ignored.
family	A family input, that can be given to glm or svyglm. Additionally, if "ols" is given, gaussian(link = "identity"), and if "logit" is given, binomial(link = "logit") is used.
rm_na	A character to determine how to handle missing values. For this two options are supported. If rm_na = "pairwise" NAs will be removed separately for every model. Only cases containing NA on one of the variables used in the respective model will be removed (all independent variables but only the respective dependent variable). If rm_na = "listwise" all cases containing NA on one of the dependent or independent variables are removed.
out_output_list	A logical value. If out_output_list = TRUE, a list will be returned, containing the separate interaction models calculated with the glm function or svyglm in case of weighting, as well as a summary object for every model. Standard errors and p-values of these models are always calculated without robustness methods.
out_df	If TRUE, the used data frames will also be part of the output list.

out_models If True, GLM model objects will be part of the returned object.

print_p If TRUE, in addition to the difference in Average Discrete Change (ADC), p-values will be printed.

print_se If TRUE, additionally standard errors will be printed.

weight, weight_bench
A character vector containing the name of the weight variable in the respective data frame. If provided the data frame will be weighted using the svydesign function. Also id must be provided.

id, id_bench A character vector containing the name of the id variable in the respectiv data frame. Only needed for weighting.

strata, strata_bench
A character vector containing the name of the strata variable in the respective data frame. It is used in the svydesign function for weighting.

nest, nest_bench
A logical vector that is used in the svydesign function for the respective data frame.

robust_se A logical value If TRUE instead of normal standard errors, heteroscedasticity-consistent standard errors will be used in the analysis to calculate them the vcovHC and coeftest packages are used.

p_adjust A logical input or character string indicating an adjustment method usable in the method parameter of p.adjust. If set to TRUE the Bonferroni adjusted p-values are used in inference.

names_df_benchmark
A vector containing first the name of df and benchmark.

silence_summary
A logical value, to indicate if the printed summary should not be printed instead.

nboots A numeric value indicating the number of bootstrap replications. If nboots = 0 no bootstrapping will be performed. Else nboots must be >2. Note, that bootstrapping can be very computationaly heavy and can therefore take a while.

boot_all If TURE, both, dfs and benchmarks will be bootstrapped. Otherwise the benchmark estimate is assumed to be constant.

parallel If TRUE, all detected cores will be used in bootstrapping.

adjustment_vars
Variables used to adjust the survey when using raking or post-stratification.

raking_targets A List of raking targets that can be given to the rake function of rake, to rake the df.

post_targets A List of post_stratification targets that can be given to the rake function of postStratify, to post_stratificatify the df.

percentile_ci If TURE, cofidence intervals will be calculated using the percentile method. If False, they will be calculated using the normal method.

Value

A table is printed showing the difference between the set of respondents for each model, as well as an indicator, if they differ significantly from each other. It is generated using the chosen method. Ifout_output_list = TRUE, also a list with additional information will be returned that can be used in some additional packages of this function to reprint the summary or to visualize the results.

Examples

```
#Example 1
## Make a comparison specifiying dependent and independent variables.

## Get Data for comparison

data("card")

north <- card[card$south==0,]


## use the function to plot the data
multi_data1<-sampcompR::multi_compare(df = north,
                                      bench = card,
                                      independent = c("age","fatheduc","motheduc","IQ"),
                                      dependent = c("educ","wage"),
                                      family="ols")

plot_multi_compare("multi_data1")

#Example 2
## Make a comparison with a formula_list
data("card")

north <- card[card$south==0,]

form_list<-list(formula(educ~age+fatheduc+motheduc+IQ),
                formula(wage~age+fatheduc+motheduc+IQ))


multi_data2 <- sampcompR::multi_compare(df = north,
                                        bench = card,
                                        formula_list = form_list,
                                        family="ols")


plot_multi_compare("multi_data2")
```

`multi_compare_merge` *Combine multi_compare_objects*

Description

`multi_compare_merge` combines two `multi_compare_objects` to plot them together.

Usage

```
multi_compare_merge(multi_reg_object1, multi_reg_object2, p_adjust = FALSE)
```

Arguments

multi_reg_object1, multi_reg_object2
: Multireg objects that should be combined.

p_adjust
: A logical input or character string indicating an adjustment method that isusable in the method parameter of p.adjust. If set to TRUE the Bonferroni adjusted p-values are used in inference.

Value

A combined multi_reg_object that can be used in plot functions to create a visualization.

Examples

```
## Get Data for comparison
data("card")

north <- card[card$south==0,]
white <- card[card$black==0,]

## use the function to plot the data
multi_data1 <- sampcompR::multi_compare(df = north,
                                        bench = card,
                                      independent = c("age","fatheduc","motheduc","IQ"),
                                        dependent = c("educ"),
                                        family = "ols")

multi_data2 <- sampcompR::multi_compare(df = white,
                                        bench = card,
                                      independent = c("age","fatheduc","motheduc","IQ"),
                                        dependent = c("wage"),
                                        family = "ols")
 ### merge two objects ###
 merged_object<-multi_compare_merge(multi_data1,multi_data2)

 ### Plot the merged object ###
 plot_multi_compare("merged_object")
```

---

multi_compare_table *Create an Output-Table of a multi_compare_object*

Description

Returns a table based on the information of a multi_compare_object which can be outputted as HTML or LaTex Table, for example with the help of the stargazer function.

Usage

```
multi_compare_table(
  multi_compare_objects,
  type = "diff",
  names = NULL,
  ndigits = 3,
  envir = parent.frame()
)
```

Arguments

multi_compare_objects	One or more object that were returned by multi_compare.
type	A character string, to determine the type of regression table. • If "dfs" a regression table based on the data frame(s) is returned. • If "benchmarks" a regression table based on the benchmark(s) is returned. • If "diff" a table indicating the difference between the df(s) and the benchmark(s) is returned.
names	A character vector to rename the data frames of comparison.
ndigits	The Number of digits that is shown in the table.
envir	The environment, where the multi_core_objects can be found.

Value

A table containing information on the multivariate comparison based on the multi_compare function.

Examples

```
## Get Data for comparison

data("card")

north <- card[card$south==0,]
white <- card[card$black==0,]

## use the function to plot the data
multi_data1 <- sampcompR::multi_compare(df = north,
                                        bench = card,
                                    independent = c("age","fatheduc","motheduc","IQ"),
                                        dependent = c("educ","wage"),
                                        family = "ols")

multi_data2 <- sampcompR::multi_compare(df = white,
                                        bench = card,
                                    independent = c("age","fatheduc","motheduc","IQ"),
                                        dependent = c("educ","wage"),
                                        family = "ols")
```

```
table<-multi_compare_table(c("multi_data1","multi_data2"),type="diff")

noquote(table)
```

`multi_per_variable` *Returns a table based on the information of a* `multi_compare_object` *that indicates the proportion of biased variables. It can be outputted as HTML or LaTex Table, for example with the help of the stargazer function.*

Description

Returns a table based on the information of a `multi_compare_object` that indicates the proportion of biased variables. It can be outputted as HTML or LaTex Table, for example with the help of the stargazer function.

Usage

```
multi_per_variable(
  multi_compare_objects,
  type = "coefs",
  label_df = NULL,
  lables_coefs = NULL,
  lables_models = NULL,
  ndigits = 1
)
```

Arguments

`multi_compare_objects`	A object returned by the `multi_compare` function. Object can either be inserted as single object or as a character string containing the names of multiple objects.
`type`	The type of table, can either be `"coefs"`, `"models"`, or `"complete"`. When coefs is chosen, the average bias of the coefficients is outputted, when models is chosen, the average bias for the models is outputted, and when complete is chosen, both are outputted.
`label_df`	A character vector containing labels for the data frames.
`lables_coefs`	A character vector containing labels for the coefficients.
`lables_models`	A character vector containing labels for the models.
`ndigits`	Number of digits that is shown in the table.

Value

A matrix, that indicates the proportion of bias for every individual coefficient or model for multivariate comparisons. This is given separately for every comparison, as well as averaged over comparisons.

Examples

```
data("card")

north <- card[card$south==0,]
white <- card[card$black==0,]

## use the function to plot the data
multi_data1 <- sampcompR::multi_compare(df = north,
                                        bench = card,
                                      independent = c("age","fatheduc","motheduc","IQ"),
                                        dependent = c("educ","wage"),
                                        family = "ols")

multi_data2 <- sampcompR::multi_compare(df = white,
                                        bench = card,
                                      independent = c("age","fatheduc","motheduc","IQ"),
                                        dependent = c("educ","wage"),
                                        family = "ols")

table<-sampcompR::multi_per_variable(multi_compare_objects = c("multi_data1","multi_data2"))
noquote(table)
```

`plot_biv_compare` *Plot Comparison of Multiple Data Frames on a Bivariate Level*

Description

Plot a object generated by biv_compare function.

Usage

```
plot_biv_compare(
  biv_data_object,
  plot_title = NULL,
  plots_label = NULL,
  p_value = NULL,
  varlabels = NULL,
  mar = c(0, 0, 0, 0),
  note = FALSE,
  grid = "white",
  diff_perc = TRUE,
  diff_perc_size = 4.5,
  perc_diff_transparance = 0,
  gradient = FALSE,
  sum_weights = NULL,
  missings_x = TRUE,
  order = NULL,
```

```
  breaks = NULL,
  colors = NULL,
  ncol_facet = 3
)
```

Arguments

biv_data_object	A object generated by the biv_compare function.
plot_title	A character string containing the title of the plot.
plots_label	A character string or vector of character strings containing the new labels of the data frames that are used in the plot.
p_value	A number between 0 and one to determine the maximum significance niveau.
varlabels	A character string or vector of character strings containing the new labels of variables that are used in the plot.
mar	A vector that determines the margins of the plot.
note	If note = TRUE, a note will be displayed to describe the plot.
grid	A character string, that determines the color of the lines between the tiles of the heatmap.
diff_perc	If TRUE a percental measure of difference between dfs and benchmarks is displayed in the plot.
diff_perc_size	A number to determine the size of the displayed percental difference between surveys in the plot.
perc_diff_transparance	A number to determine the transparency of the displayed percental-difference between surveys in the plot.
gradient	If gradient = TRUE, colors in the heatmap will be more or less transparent, depending on the difference in Pearson's r of the data frames of comparison.
sum_weights	A vector containing information for every variable to weigh them in the displayed percental difference calculation. It can be used if some variables are over- or underrepresented in the analysis.
missings_x	If TRUE, missing pairs in the plot will be marked with an X.
order	A character vector to determine in which order the variables should be displayed in the plot.
breaks	A vector to label the color scheme in the legend.
colors	A vector to determine the colors in the plot.
ncol_facet	Number of columns used in faced_wrap() for the plots.

Details

The plot shows a heatmap of a correlation matrix, where the colors are determined by the similarity of the Pearson's r value in both sets of respondents. Leaving default breaks and colors,

- `Same` (green) indicates, that the Pearson's r correlation is not significant > 0 in the related data frame or benchmark or the Pearson's r correlations are not significant different, between data frame and benchmark.
- `Small Diff` (yellow) indicates that the Pearson's r correlation is significant > 0 in the related data frame or benchmark and the Pearson's r correlations are significant different, between data frame and benchmark.
- `Large Diff` (red) indicates, that the same coditions of yellow are fulfilled, and the correlations are either in opposite directions,or one is double the size of the other.

Value

A object generated with the help of `ggplot2::ggplot2()`, used to visualize the differences between the data frames and benchmarks.

Examples

```
## Get Data for comparison

data("card")

north <- card[card$south==0,]
white <- card[card$black==0,]

## use the function to plot the data
bivar_data<-sampcompR::biv_compare(dfs = c("north","white"),
                                   benchmarks = c("card","card"),
                               variables= c("age","educ","fatheduc","motheduc","wage","IQ"),
                                   data=TRUE)

sampcompR::plot_biv_compare(bivar_data)
```

`plot_multi_compare` *Plot Multiple multi_compare_objects*

Description

`plot_multi_compare` plots multipe `multi_compare_objects` together.

Usage

```
plot_multi_compare(
  multi_compare_objects,
  plots_label = NULL,
  plot_title = NULL,
  p_value = 0.05,
  breaks = NULL,
  plot_data = FALSE,
```

```
  colors = NULL,
  variant = "one",
  p_adjust = NULL,
  note = FALSE,
  grid = "white",
  diff_perc = TRUE,
  diff_perc_size = 4.5,
  ncol_facet = 3,
  perc_diff_transparance = 0,
  diff_perc_position = "top_right",
  gradient = FALSE,
  sum_weights_indep = NULL,
  sum_weights_dep = NULL,
  label_x = NULL,
  label_y = NULL,
  missings_x = TRUE
)
```

Arguments

`multi_compare_objects`
: A character vector containing the names of one or more `multi_compare_objects`. Every object will be displayed separately in `facet_wrap` of `ggplot`.

`plots_label`
: A character vector of the same lengths as `multi_compare_objects`, to name the different objects in facet_wrap of ggplot.

`plot_title`
: A string containing the title of the visualization.

`p_value`
: A number between zero and one, that is used as p-value in significance analyses.

`breaks`
: A vector, containing several of strings, to rename the categories in the legend. Its possible length depends on the `variant`.

`plot_data`
: A logical value. If `TRUE`, instead of a plot a data frame will be returned, that is used for the plot.

`colors`
: A vector of colors, usable in ggplot, for every break. It's possible length depends on the `variant`.

`variant`
: Variant can be either "one", "two", "three","four","five", or "six".

  `variant = "one"` The plot will show whether the coefficients in the regression models are significantly different from each other (Diff). When they are, it will also show if they differ in strength (one is twice the size of the other) or direction as well (Large Diff).

  `variant = "two"` The plot will show whether coefficients in the regression models differ significantly from each other (Large Diff). If not it will show whether they still differ in direction (Diff in Direction) or whether one is significant while the other is not (Diff in Significance).

  `variant = "three"` The plot will show whether coefficients in the regression models differ from each other in various aspects. Whether one is significant, while the other is not (Diff in Significance), whether they differ in direction (Diff in Direction) or whether one is double the size of the other (Diff in

Strength). When variables meet the criteria for multiple categories they will classified in the latest fitting category.

`variant = "four"` The plot will show if the coefficient in the df is significant, while the coefficient is not significant in the benchmark or the other way around (Diff in Significance).

`variant = "five"` The plot will show if the coefficient in the df is positive, while the coefficient in the benchmark is negative or the other way around (Diff in Direction).

`variant = "six"` The plot will show if the coefficient in the df is double the size of the coefficient in the benchmark or the other way around (Diff in Strength).

`p_adjust` If `TRUE` results based on adjusted p-values will be used. Adjustment methods depend on the method used to generate the `multi_compare_objects`.

`note` A logical value. If `TRUE`, a note will be displayed under the plot describing the `variant`.

`grid` A string, that can either be "none" or a color, for the edges of every tile. If "none", no grid will be displayed.

`diff_perc` A logical value. If `TRUE`, the percent of the differing categories, decided by the variant, will be displayed in the corner of the plot.

`diff_perc_size` A number to decide the size of the text in `diff_perc`.

`ncol_facet` A number of columns used in faced_wrap() for the plots.

`perc_diff_transparance`
A number between zero and one, to decide the background transparency of `diff_perc`.

`diff_perc_position`
A character string, to choose the position of `diff_perc` Can either be "top_right"(default), "bottom_right", "bottom_left", or "top_left".

`gradient` A logical Value. If `TRUE`, the transparency of the tiles depends on the coefficient.

`sum_weights_indep`, `sum_weights_dep`
A vector of weights for every dependent or independent variable. Must be `NULL`, or the same length as the dependent variables or independent variables.

`label_x`, `label_y`
A character string or vector of character strings containing a label for the x-axis and y-axis.

`missings_x` If `TRUE`, missing pairs in the plot will be marked with an X.

Value

Returns a a heat matrix-like plot created with ggplot, to visualize the multivariate differences. If multiple objects are used, they will be displayed separately with ggplot's facet_wrap function. On the y-axis, the independent variables are displayed, while on the x-axis the independent variables are displayed. Depending on the variant, the displayed tile colors must be interpreted differently. FALSEor more information on interpretation look at `variant`.

Examples

```
## Get Data for comparison

data("card")

north <- card[card$south==0,]
white <- card[card$black==0,]

## use the function to plot the data
multi_data1 <- sampcompR::multi_compare(df = north,
                                        bench = card,
                                      independent = c("age","fatheduc","motheduc","IQ"),
                                        dependent = c("educ","wage"),
                                        family = "ols")

multi_data2 <- sampcompR::multi_compare(df = white,
                                        bench = card,
                                      independent = c("age","fatheduc","motheduc","IQ"),
                                        dependent = c("educ","wage"),
                                        family = "ols")

plot_multi_compare(c("multi_data1","multi_data2"))
```

---

plot_uni_compare *plot univar data*

Description

plot_uni_compare This uses ggplot2 to generate a plot based on an object generated by the uni_compare function.

Usage

```
plot_uni_compare(
  uni_compare_objects,
  name_dfs = NULL,
  name_benchmarks = NULL,
  summetric = NULL,
  colors = NULL,
  shapes = NULL,
  legendlabels = NULL,
  legendtitle = NULL,
  label_x = NULL,
  label_y = NULL,
  summet_size = NULL,
  point_size = NULL,
```

```
  errorbar_size = NULL,
  plot_title = NULL,
  conf_adjustment = FALSE,
  varlabels = NULL,
  ndigits = 3
)
```

Arguments

uni_compare_objects	A object generated by the uni_compare function.
name_dfs, name_benchmarks	A character string or vector of character strings containing the new names of the data frames and the benchmarks, that are used in the plot.
summetric	If "avg1","avg2", "mse1","mse2", "rmse1","rmse2", or "R" the respective measure is calculated for the biases of each survey. The values "avg1", "mse1" and "rmse1" lead to similar results as in "avg2", "mse2" and "rmse2", with slightly different visualization in the plot. If summetric = "none", no summetric will be displayed in the plot, and if summetric = NULL the summetric specified in the uni_compare_object is used.
colors	A vector of colors that is used in the plot for the different comparisons. The color has to be specified separately for every comparison, with one value of the vector.
shapes	A vector of shapes applicable in ggplot2::ggplot2() that is used in the plot for the different comparisons. The shapes has to be specified separately for every comparison, with one value of the vector.
legendlabels	A character string or vector of strings containing a label for the legend.
legendtitle	A character string containing the title of the legend.
label_x, label_y	A character string or vector of character strings containing a label for the x-axis and y-axis.
summet_size	A number to determine the size of the displayed summetric in the plot.
point_size	Either NULL or a number indicating the size of the dots in the plot. If Null by default the size is specified by ggplot.
errorbar_size	Either NULL or a number indicating the size of the errorbars in the plot. If Null by default the size is specified by ggplot.
plot_title	A character string containing the title of the plot.
conf_adjustment	If conf_adjustment = TRUE the confidence level of the confidence interval will be adjusted with a Bonferroni adjustment, to account for the problem of multiple comparisons.
varlabels	A character string or vector of character strings containing the new names of the variables, also used in plot.
ndigits	The number of digits to round the numbers in the plot.

Value

Plot of a uni_compare object using ggplot2::ggplot2() which shows the difference between two or more data frames.

Examples

```
## Get Data for comparison

data("card")

south <- card[card$south==1,]
north <- card[card$south==0,]
black <- card[card$black==1,]
white <- card[card$black==0,]

## use the function to plot the data
univar_data<-sampcompR::uni_compare(dfs = c("north","white"),
                                    benchmarks = c("south","black"),
                              variables= c("age","educ","fatheduc","motheduc","wage","IQ"),
                                    funct = "abs_rel_mean",
                                    nboots=0,
                                    summetric="rmse2",
                                    data=TRUE)

sampcompR::plot_uni_compare(univar_data)
```

R_indicator *Calculate the R-Indicator*

Description

Calculates the R-Indicator of the (weighted) data frame.

Usage

```
R_indicator(
  dfs,
  response_identificators,
  variables,
  id = NULL,
  weight = NULL,
  strata = NULL,
  get_r2 = FALSE
)
```

Arguments

dfs	A character vector containing the names of data frames to calculate the R indicator.
response_identificators	A character vector, naming response identificators for every df. Response identificators should indicate if respondents are part of the set of respondents (respondents = 1) or not part of the set of respondents. (non-respondents = 0). If only one character is provided, the same variable is used in every df.
variables	A character vector with the names of variables that should be used in the model to calculate the R indicator.
id	A character vector that determines id variables that are used to weight the dfs with the help of the survey package. They have to be part of the respective data frame. If only one character is provided, the same variable is used to weight every df.
weight	A character vector that determines variables to weight the dfs. They have to be part of the respective data frame. If only one character is provided, the same variable used to weight every df. If a weight variable is provided also an id variable is needed. For weighting, the survey package is used.
strata	A character vector that determines strata variables that are used to weight the dfs with the help of the survey package. They have to be part of the respective data frame. If only one character is provided, the same variable is used to weight every df.
get_r2	If true, Pseudo R-squared of the propensity model will be returned, based on the method of McFadden.

Value

A list containing the R-indicator, and its standard error for every data frame.

Note

The calculated R-indicator is based on Shlomo et al., (2012).

References

- Shlomo, N., Skinner, C., & Schouten, B. (2012). Estimation of an indicator of the representativeness of survey response. Journal of Statistical Planning and Inference, 142(1), 201–211. https://doi.org/10.1016/j.jspi.2011.07.008

Examples

```
data("card")

# For the purpose of this example, we assume that only respondents living in
# the south or only white respondents have participated in the survey.

sampcompR::R_indicator(dfs=c("card","card"),
```

```
response_identificators = c("south","black"),
variables = c("age","educ","fatheduc","motheduc","wage","IQ"),
weight = c("weight","weight"))
```

---

sampcompR *sampcompR: A package for the comparison of samples*

Description

Easily analyze and visualize differences between samples (e.g., benchmark comparisons, nonresponse comparisons in surveys) on three levels. The comparisons can be univariate, bivariate or multivariate. On univariate level the variables of interest of a survey and a comparison survey (i.e. benchmark) are compared, by calculating one of several difference measures (e.g., relative difference in mean), and an average difference between the surveys. On bivariate level a function can calculate significant differences in correlations for the surveys. And on multivariate levels a function can calculate significant differences in model coefficients between the surveys of comparison. All of those differences can be easily plotted and outputted as a table. Visualization is based on `ggplot` and can be edited as other plots of ggplot afterwards. For more detailed information on the methods and example use see: Rohr, B., Silber, H., & Felderer, B. (2024). „Comparing the Accuracy of Univariate, Bivariate, and Multivariate Estimates across Probability and Non-Probability Surveys with Population Benchmarks“ https://doi.org/10.31235/osf.io/n6ehf.

sampcompR functions

uni_compare Compare Datasets Univariate and Plot Differences

plot_uni_compare Plot uni_compare objects

uni_compare_table Get a table output of a uni_compare object

R_indicator Calculate the R_indicator for several surveys

biv_compare Compare Datasets Bivariate and Plot Differences

plot_biv_compare Plot biv_compare objects

biv_compare_table Get a table output of a biv_compare object

multi_compare Compare two Datasets on a Multivariate Level (Any GLM Model)

plot_multi_compare Plot multi_compare objects

multi_compare_table Get a table output of multi_compare objects

multi_compare_merge Combine two multi_compare objects, to plot them together

descriptive_table Get a Descriptive Table for Every Data Frame

dataequalizer Equalize dataframes

uni_compare function

uni_compare Returns data or a plot showing the difference of two or more data frames The differences are calculated on the base of differing metrics, chosen in the funct argument. Results can be visualized using `plot_uni_compare`.

biv_compare function

biv_compare Returns data or heatmap of difference between two or more data frames, by comparing their correlation matrices. The comparison is based on Pearson's r, calculated using the `rcorr` function. Results can be visualized using `plot_biv_compare`.

multi_compare function

multi_compare Returns data of difference between two data frames on a multivariate level. Similar (multivariate) regression models are compared between the surveys. Only GLM models are possible. Results can be visualized using `plot_multi_compare`.

dataequalizer function

dataequalizer compares two data frames and looks if data frames contain columns with the same name. A copy of y is returned, containing only columns named identical in x and y data frames. The function is mainly used in the other functions of the package.

_PACKAGE

`uni_compare` *Compare data frames and Plot Differences*

Description

Returns data or a plot showing the difference of two or more data frames The differences are calculated on the base of differing metrics, chosen in the funct argument. All used data frames must contain at least one column named equal in all data frames, that has equal values.

Usage

```
uni_compare(
  dfs,
  benchmarks,
  variables = NULL,
  nboots = 2000,
  n_bench = NULL,
  boot_all = FALSE,
  funct = "rel_mean",
  data = TRUE,
  type = "comparison",
  legendlabels = NULL,
  legendtitle = NULL,
  colors = NULL,
  shapes = NULL,
  summetric = "rmse2",
  label_x = NULL,
  label_y = NULL,
```

```
  plot_title = NULL,
  varlabels = NULL,
  name_dfs = NULL,
  name_benchmarks = NULL,
  summet_size = 4,
  silence = TRUE,
  conf_level = 0.95,
  conf_adjustment = NULL,
  percentile_ci = TRUE,
  weight = NULL,
  id = NULL,
  strata = NULL,
  weight_bench = NULL,
  id_bench = NULL,
  strata_bench = NULL,
  adjustment_weighting = "raking",
  adjustment_vars = NULL,
  raking_targets = NULL,
  post_targets = NULL,
  ndigits = 3,
  parallel = FALSE
)
```

Arguments

dfs	A character vector containing the names of data frames to compare against the benchmarks.
benchmarks	A character vector containing the names of benchmarks to compare the data frames against. The vector must either be the same length as dfs, or length 1. If it has length 1 every df will be compared against the same benchmark. Benchmarks can either be the name of data frames, the name of a list of tables, or a named vector of means. The tables in the list need to be named as the respective variables in the data frame of comparison. When they are a named vector of means, the means need to be named as the respective variables in the dfs.
variables	A character vector containing the names of the variables for the comparison. If NULL, all variables named similarly in both the dfs and the benchmarks will be compared. Variables missing in one of the data frames or the benchmarks will be neglected for this comparison.
nboots	The number of bootstraps used to calculate standard errors. Must either be >2 or 0. If >2 bootstrapping is used to calculate standard errors with nboots iterations. If 0, SE is calculated analytically. We do not recommend using nboots =0 because this method is not yet suitable for every funct used and every method. Depending on the size of the data and the number of bootstraps, uni_compare can take a while.
n_bench	A list of vectors containing the number of cases for every variable in the benchmark. This is only needed, if the benchmark is given as a vector. The list should be as long as the number of dataframes

`boot_all` If TURE, both, dfs and benchmarks will be bootstrapped. Otherwise the benchmark estimate is assumed to be constant.

`funct` A character string, indicating the function to calculate the difference between the data frames.

Predefined functions are:

- `"d_mean"`, `"ad_mean"` A function to calculate the (absolute) difference in mean of the variables in `dfs` and benchmarks with the same name. Only applicable for metric variables.
- `"d_prop"`, `"ad_prop"` A function to calculate the (absolute) difference in proportions of the variables in `dfs` and benchmarks with the same name. Only applicable for dummy variables.
- `"rel_mean"`, `"abs_rel_mean"` A function to calculate the (absolute) relative difference in mean of the variables in `dfs` and benchmarks with the same name. #' For more information on the formula for difference and analytic variance, see Felderer et al. (2019). Only applicable for metric variables.
- `"rel_prop"`, `"abs_rel_prop"` A function to calculate the (absolute) relative difference in proportions of the variables in `dfs` and benchmarks with the same name. It is calculated similar to the relative difference in mean (see Felderer et al., 2019), however the default label for the plot is different. Only applicable for dummy variables.

`data` If TRUE, a uni_compare_object is returned, containing results of the comparison.

`type` Define the type of comparison. Can either be `"comparison"` or `"nonresponse"`.

`legendlabels` A character string or vector of strings containing a label for the legend.

`legendtitle` A character string containing the title of the legend.

`colors` A vector of colors, that is used in the plot for the different comparisons.

`shapes` A vector of shapes applicable in `ggplot2::ggplot2()`, that is used in the plot for the different comparisons.

`summetric` If `"avg1"`, `"mse1"`, `"rmse1"`, or `"R"` the respective measure is calculated for the biases of each survey. The values `"mse1"` and `"rmse1"` lead to similar results as in `"mse2"` and `"rmse2"`, with slightly different visualization in the plot. If `summetric = NULL`, no summetric will be displayed in the Plot. When `"R"` is chosen, also `response_identificator` is needed.

`label_x`, `label_y` A character string or vector of character strings containing a label for the x-axis and y-axis.

`plot_title` A character string containing the title of the plot.

`varlabels` A character string or vector of character strings containing the new names of variables, also used in plot.

`name_dfs`, `name_benchmarks` A character string or vector of character strings containing the new names of the `dfs` and `benchmarks`, that is also used in plot.

`summet_size` A number to determine the size of the displayed `summetric` in the plot.

`silence` If `silence = FALSE` a warning will be displayed, if variables a re excluded from either the data frame or benchmark, for not existing in both.

`conf_level` A numeric value between zero and one to determine the confidence level of the confidence interval.

`conf_adjustment`
If `conf_adjustment = TRUE` the confidence level of the confidence interval will be adjusted with a Bonferroni adjustment, to account for the problem of multiple comparisons.

`percentile_ci` If TURE, cofidence intervals will be calculated using the percentile method. If False, they will be calculated using the normal method.

`weight, weight_bench`
A character vector determining variables to weight the `dfs` or `benchmarks`. They have to be part of the respective data frame. If only one character is provided, the same variable is used to weigh every `df` or `benchmark`. If a weight variable is provided also an `id` variable is needed.For weighting, the `survey` package is used.

`id, id_bench` A character vector determining `id` variables used to weigh the `dfs` or `benchmarks` with the help of the `survey` package. They have to be part of the respective data frame. If only one character is provided, the same variable is used to weigh every `df` or `benchmark`.

`strata, strata_bench`
A character vector determining strata variables used to weigh the `dfs` or `benchmarks` with the help of the `survey` package.They have to be part of the respective data frame. If only one character is provided, the same variable is used to weight every `df` or `benchmark`.

`adjustment_weighting`
A character vector indicating if adjustment weighting should be used. It can either be `"raking"` or `"post_start"`.

`adjustment_vars`
Variables used to adjust the survey when using raking or post stratification.

`raking_targets` A list of raking targets that can be given to the rake function of `rake`, to rake the `dfs`.

`post_targets` A list of post-stratification targets that can be given to the `postStratify` function, to post-stratify the `dfs`.

`ndigits` The number of digits to round the numbers in the plot.

`parallel` Can be either `FALSE` or a number of cores that should be used in the function. If it is `FALSE`, only one core will be used and otherwise the given number of cores will be used.

Value

A plot based on `ggplot2::ggplot2()` (or data frame if data==TRUE) which shows the difference between two or more data frames on predetermined variables, named identical in both data frames.

References

Felderer, B., Kirchner, A., & Kreuter, FALSE. (2019). The Effect of Survey Mode on Data Quality: Disentangling Nonresponse and Measurement Error Bias. Journal of Official Statistics, 35(1), 93–115. https://doi.org/10.2478/jos-2019-0005

Examples

```
## Get Data for comparison

data("card")

north<-card[card$south==0,]
white<-card[card$black==0,]

## use the function to plot the data
univar_comp<-sampcompR::uni_compare(dfs = c("north","white"),
                                    benchmarks = c("card","card"),
                              variables= c("age","educ","fatheduc","motheduc","wage","IQ"),
                                    funct = "abs_rel_mean",
                                    nboots=200,
                                    summetric="rmse2",
                                    data=FALSE)

 univar_comp
```

`uni_compare_table` *Create an Output-Table of a uni_compare_object*

Description

Returns a table based on the information of an `uni_compare_object` which can be outputted as HTML or LaTex Table, for example with the help of the stargazer function.

Usage

```
uni_compare_table(
  uni_compare_object,
  conf_adjustment = FALSE,
  df_names = NULL,
  varlabels = NULL,
  ci_line = TRUE,
  ndigits = 3
)
```

Arguments

uni_compare_object	A object returned by uni_compare.
conf_adjustment	A logical parameter determining if adjusted confidence intervals should be returned.
df_names	A character vector to relabel the data frames of comparison.
varlabels	A character vector to relabel the variables in the table.
ci_line	If TRUE, confidence intervals will be displayed in a separate line, otherwise, they are shown in the same line instead.
ndigits	The number of digits to round the numbers in table.

Value

A table containing information on the univariate comparison based on the uni_compare function.

Examples

```
## Get Data for comparison

data("card")

north <- card[card$south==0,]
white <- card[card$black==0,]

## use the function to plot the data
univar_data<-sampcompR::uni_compare(dfs = c("north","white"),
                                    benchmarks = c("card","card"),
                            variables= c("age","educ","fatheduc","motheduc","wage","IQ"),
                                    funct = "abs_rel_mean",
                                    nboots=0,
                                    summetric="rmse2",
                                    data=TRUE)

table<-sampcompR::uni_compare_table(univar_data)
noquote(table)
```

Index