Package ‘reportRmd’

January 24, 2025

Title Tidy Presentation of Clinical Reporting

Version 0.1.1

Date 2024-12-30

Description Streamlined statistical reporting in 'Rmarkdown' environments. Facilitates the automated reporting of descriptive statistics, multiple univariate models, multivariable models and tables combining these outputs. Plotting functions include customisable survival curves, forest plots from logistic and ordinal regression and bivariate comparison plots.

License MIT + file LICENSE

Suggests nlme, tidycmprsk, rmarkdown, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

Imports aod, boot, cmprsk, cowplot, dplyr, geepack, ggplot2, ggpubr, gridExtra, kableExtra, knitr, lifecycle, MASS, pander, plyr, rlang, rstatix, scales, stats, survival, tidyselect

Collate 'helper.R' 'main.R' 'globals.R' 'data.R' 'lblCode.R' 'rm_compactsum.R' 'rm_uvsum2.R' 'autoreg.R' 'autosum.R' 'getVarLevels.R' 'rm_mvsum2.R'

Depends R (>= 4.2)

LazyData no

VignetteBuilder knitr, rmarkdown

NeedsCompilation no

Author Lisa Avery [cre, aut] (<`https://orcid.org/0000-0002-8431-5143`>),
Ryan Del Bel [aut],
Osvaldo Espin-Garcia [aut],
Katherine Lajkosz [aut] (<`https://orcid.org/0000-0003-3760-5401`>),
Clarina Ong [aut],
Tyler Pittman [aut] (<`https://orcid.org/0000-0002-5013-6980`>),
Anna Santiago [aut] (<`https://orcid.org/0000-0002-0932-2386`>),
Yanning Wang [ctr],
Jessica Weiss [aut],
Wei Xu [aut]

Maintainer Lisa Avery <`lisa.avery@uhn.ca`>

Repository CRAN

Date/Publication 2025-01-24 18:40:02 UTC

Contents

`addspace` *Add spaces to strings in LaTeX*

Description

Add spaces to strings in LaTeX. Returns appends ~~~ before the string

Usage

```
addspace(x)
```

Arguments

`x`	string

`boxcoxfitRx` *fit box cox transformed linear model*

Description

Wrapper function to fit fine and gray competing risk model using function crr from package cmprsk

Usage

```
boxcoxfitRx(f, data, lambda = FALSE)
```

Arguments

`f`	formula for the model. Currently the formula only works by using the name of the column in a dataframe. It does not work by using $ or [] notation.
`data`	dataframe containing data
`lambda`	boolean indicating if you want to output the lamda used in the boxcox transformation. If so the function will return a list of length 2 with the model as the first element and a vector of length 2 as the second.

Value

a list containing the linear model (lm) object and, if requested, lambda

`cap` *Capitalize a string*

Description

Capitalize a string

Usage

```
cap(x)
```

Arguments

`x`	string

`clear_labels` *Clear variable labels from a data frame*

Description

This function will remove all label attributes from variables in the data.

Usage

```
clear_labels(data)
```

Arguments

`data`	the data frame to remove labels from

Details

To change or remove individual labels use set_labels or set_var_labels

Examples

```
# Set a few variable labels for ctDNA
data("ctDNA")
ctDNA <- ctDNA |> set_var_labels(
   ctdna_status="detectable ctDNA",
  cohort="A cohort label")
# Clear all variable data frames and check
clear_labels(ctDNA)
```

`covsum` *Get covariate summary dataframe*

Description

Returns a dataframe corresponding to a descriptive table.

Usage

```
covsum(
  data,
  covs,
  maincov = NULL,
  digits = 1,
  numobs = NULL,
  markup = FALSE,
  sanitize = FALSE,
  nicenames = TRUE,
  IQR = FALSE,
  all.stats = FALSE,
  pvalue = TRUE,
  effSize = FALSE,
  show.tests = FALSE,
  dropLevels = TRUE,
  excludeLevels = NULL,
  full = TRUE,
  digits.cat = 0,
  testcont = c("rank-sum test", "ANOVA"),
  testcat = c("Chi-squared", "Fisher"),
  include_missing = FALSE,
  percentage = c("column", "row")
)
```

Arguments

`data`	dataframe containing data
`covs`	character vector with the names of columns to include in table
`maincov`	covariate to stratify table by
`digits`	number of digits for summarizing mean data, does not affect p-values
`numobs`	named list overriding the number of people you expect to have the covariate
`markup`	boolean indicating if you want latex markup
`sanitize`	boolean indicating if you want to sanitize all strings to not break LaTeX
`nicenames`	boolean indicating if you want to replace . and _ in strings with a space
`IQR`	boolean indicating if you want to display the inter quantile range (Q1,Q3) as opposed to (min,max) in the summary for continuous variables

`all.stats`	boolean indicating if all summary statistics (Q1,Q3 + min,max on a separate line) should be displayed. Overrides IQR.
`pvalue`	boolean indicating if you want p-values included in the table
`effSize`	boolean indicating if you want effect sizes included in the table. Can only be obtained if pvalue is also requested. Effect sizes calculated include Cramer's V for categorical variables, Cohen's d, Wilcoxon r, or Eta-squared for numeric/continuous variables.
`show.tests`	boolean indicating if the type of statistical test and effect size used should be shown in a column beside the pvalues. Ignored if pvalue=FALSE.
`dropLevels`	logical, indicating if empty factor levels be dropped from the output, default is TRUE.
`excludeLevels`	a named list of covariate levels to exclude from statistical tests in the form list(varname =c('level1','level2')). These levels will be excluded from association tests, but not the table. This can be useful for levels where there is a logical skip (ie not missing, but not presented). Ignored if pvalue=FALSE.
`full`	boolean indicating if you want the full sample included in the table, ignored if maincov is NULL
`digits.cat`	number of digits for the proportions when summarizing categorical data (default: 0)
`testcont`	test of choice for continuous variables,one of *rank-sum* (default) or *ANOVA*
`testcat`	test of choice for categorical variables,one of *Chi-squared* (default) or *Fisher*
`include_missing`	Option to include NA values of maincov. NAs will not be included in statistical tests
`percentage`	choice of how percentages are presented ,one of *column* (default) or *row*

Details

Comparisons for categorical variables default to chi-square tests, but if there are counts of <5 then the Fisher Exact test will be used and if this is unsuccessful then a second attempt will be made computing p-values using MC simulation. If testcont='ANOVA' then the t-test with unequal variance will be used for two groups and an ANOVA will be used for three or more. The statistical test used can be displayed by specifying show.tests=TRUE.

The number of decimals places to display the statistics can be changed with digits, but this will not change the display of p-values. If more significant digits are required for p-values then use tableOnly=TRUE and format as desired.

References

Ellis, P.D. (2010) The essential guide to effect sizes: statistical power, meta-analysis, and the interpretation of research results. Cambridge: Cambridge University Press.doi:10.1017/CBO9780511761676

Lakens, D. (2013) Calculating and reporting effect sizes to facilitate cumulative science: a practical primer for t-tests and ANOVAs. Frontiers in Psychology, 4; 863:1-12. doi:10.3389/fpsyg.2013.00863

See Also

fisher.test,chisq.test, wilcox.test,kruskal.test,and anova

---

crrRx *fit crr model*

Description

Wrapper function to fit fine and gray competing risk model using function crr from package cmprsk

Usage

```
crrRx(f, data)
```

Arguments

f	formula for the model. Currently the formula only works by using the name of the column in a dataframe. It does not work by using $ or [] notation.
data	dataframe containing data

Value

a competing risk model with the call appended to the list

See Also

cmprsk::crr

Examples

```
# From the crr help file:
set.seed(10)
ftime <- rexp(200)
fstatus <- sample(0:2,200,replace=TRUE)
cov <- matrix(runif(600),nrow=200)
dimnames(cov)[[2]] <- c('x1','x2','x3')
df <- data.frame(ftime,fstatus,cov)
m1 <- crrRx(as.formula('ftime+fstatus~x1+x2+x3'),df)
# Nicely output to report:
rm_mvsum(m1,data=df,showN = TRUE,vif=TRUE)
```

`ctDNA` *Tumour size change over time Longitudinal changes in tumour size since baseline for patients by changes in ctDNA status (clearance, decrease or increase) since baseline.*

Description

Tumour size change over time

Longitudinal changes in tumour size since baseline for patients by changes in ctDNA status (clearance, decrease or increase) since baseline.

Usage

```
data('ctDNA')
```

Format

A data frame with 270 rows and 5 variables:

id Patient ID

cohort Study Cohort

ctdna_status Change in ctDNA since baseline

time Number of weeks on treatment

size_change Percentage change in tumour measurement

Source

`https://www.nature.com/articles/s43018-020-0096-5`

`excelCol` *Retrieve columns number from spreadsheet columns specified as unquoted letters*

Description

Retrieve columns number from spreadsheet columns specified as unquoted letters

Usage

```
excelCol(...)
```

Arguments

`...`	unquoted excel column headers (i.e. excelCol(A,CG,AA)) separated by commas

Value

a numeric vector corresponding to columns in a spreadsheet

Examples

```
## Find the column numbers for excel columns AB, CE and BB
excelCol(AB,CE,bb)
## Get the columns between A and K and Z
excelCol(A-K,Z)
```

---

excelColLetters *Retrieve spreadsheet column letter-names from columns indices*

Description

Creates a vector of spreadsheet-style letter-names corresponding to column numbers

Usage

```
excelColLetters(columnIndices)
```

Arguments

`columnIndices`	vector of integer column indices

Details

This is the inverse function of excelCol

Value

a character vector corresponding to the spreadsheet column headings

Examples

```
## Find the column numbers for excel columns AB, CE and BB
colIndices <- excelCol(AB,CE,bb)
## Go back to the column names
excelColLetters(colIndices)
```

`extract_labels` *Extract variable labels from labelled data frame*

Description

Extract variable labels from data and return a data frame with labels

Usage

```
extract_labels(data, sep = "_")
```

Arguments

`data`	the data frame to extract labels from
`sep`	character used to separate multiple labels, defaults to "_"

Details

All variable names will be returned, even those with no labels. If the label attribute has length greater than one the values will be concatenated and returned as a single string separated by sep

Examples

```
# Set a few variable labels for ctDNA
data("ctDNA")
ctDNA <- ctDNA |> set_var_labels(
   ctdna_status="detectable ctDNA",
  cohort="A cohort label")
# Extract labels
extract_labels(ctDNA)
```

`forestplot2` *Create a forest plot using ggplot2*

Description

This function will accept a log or logistic regression fit from glm or geeglm, and display the OR or RR for each variable on the appropriate log scale.

Usage

```
forestplot2(
  model,
  conf.level = 0.95,
  orderByRisk = TRUE,
  colours = "default",
  showEst = TRUE,
  rmRef = FALSE,
  logScale = getOption("reportRmd.logScale", TRUE),
  nxTicks = 5
)
```

Arguments

`model`	an object output from the glm or geeglm function, must be from a logistic regression
`conf.level`	controls the width of the confidence interval
`orderByRisk`	logical, should the plot be ordered by risk
`colours`	can specify colours for risks less than, 1 and greater than 1.0. Default is red, black, green
`showEst`	logical, should the risks be displayed on the plot in text
`rmRef`	logical, should the reference levels be removed for the plot?
`logScale`	logical, should OR/RR be shown on log scale, defaults to TRUE, or reportRmd.logScale if set. See https://doi.org/10.1093/aje/kwr156 for why you may prefer a linear scale.
`nxTicks`	Number of tick marks supplied to the log_breaks function to produce

Value

a plot object

Examples

```
data("pembrolizumab")
glm_fit = glm(orr~change_ctdna_group+sex+age+l_size,
data=pembrolizumab,family = 'binomial')
forestplot2(glm_fit)
```

`forestplotMV` *Create a multivariable forest plot using ggplot2*

Description

This function will send and take log or logistic regression fit from glm or geeglm from mvsum function, and display the OR or RR for each variable on the appropriate log scale.

Usage

```
forestplotMV(
  model,
  data,
  conf.level = 0.95,
  orderByRisk = TRUE,
  colours = "default",
  showEst = TRUE,
  rmRef = FALSE,
  digits = getOption("reportRmd.digits", 2),
  logScale = getOption("reportRmd.logScale", TRUE),
  nxTicks = 5,
  showN = TRUE,
  showEvent = TRUE
)
```

Arguments

`model`	an object output from the glm or geeglm function, must be from a logistic regression
`data`	dataframe containing your data
`conf.level`	controls the width of the confidence interval
`orderByRisk`	logical, should the plot be ordered by risk
`colours`	can specify colours for risks less than, 1 and greater than 1.0. Default is red, black, green
`showEst`	logical, should the risks be displayed on the plot in text
`rmRef`	logical, should the reference levels be removed for the plot?
`digits`	number of digits to use displaying estimates
`logScale`	logical, should OR/RR be shown on log scale, defaults to TRUE, or reportRmd.logScale if set. See https://doi.org/10.1093/aje/kwr156 for why you may prefer a linear scale.
`nxTicks`	Number of tick marks supplied to the log_breaks function to produce
`showN`	Show number of observations per variable and category
`showEvent`	Show number of events per variable and category

Value

a plot object

Examples

```
data("pembrolizumab")
glm_fit = glm(orr~change_ctdna_group+sex+age+l_size,
data=pembrolizumab,family = 'binomial')
forestplotMV(glm_fit)
```

forestplotUV *Create an univariable forest plot using ggplot2*

Description

This function will send and take log or logistic regression fit from glm or geeglm from uvsum function, and display the OR or RR for each variable on the appropriate log scale.

Usage

```
forestplotUV(
  response,
  covs,
  data,
  id = NULL,
  corstr = NULL,
  model = "glm",
  family = NULL,
  digits = getOption("reportRmd.digits", 2),
  conf.level = 0.95,
  orderByRisk = TRUE,
  colours = "default",
  showEst = TRUE,
  rmRef = FALSE,
  logScale = getOption("reportRmd.logScale", TRUE),
  nxTicks = 5,
  showN = TRUE,
  showEvent = TRUE
)
```

Arguments

`response`	character vector with names of columns to use for response
`covs`	character vector with names of columns to use for covariates
`data`	dataframe containing your data
`id`	character vector which identifies clusters. Only used for geeglm
`corstr`	character string specifying the correlation structure. Only used for geeglm. The following are permitted: '"independence"', '"exchangeable"', '"ar1"', '"unstructured"' and '"userdefined"'
`model`	fitted model object
`family`	description of the error distribution and link function to be used in the model. Only used for geeglm
`digits`	number of digits to round to
`conf.level`	controls the width of the confidence interval

`orderByRisk`	logical, should the plot be ordered by risk
`colours`	can specify colours for risks less than, 1 and greater than 1.0. Default is red, black, green
`showEst`	logical, should the risks be displayed on the plot in text
`rmRef`	logical, should the reference levels be removed for the plot?
`logScale`	logical, should OR/RR be shown on log scale, defaults to TRUE, or reportRmd.logScale if set. See https://doi.org/10.1093/aje/kwr156 for why you may prefer a linear scale.
`nxTicks`	Number of tick marks supplied to the log_breaks function to produce
`showN`	Show number of observations per variable and category
`showEvent`	Show number of events per variable and category

Value

a plot object

Examples

```
data("pembrolizumab")
forestplotUV(response="orr", covs=c("change_ctdna_group", "sex", "age", "l_size"),
data=pembrolizumab, family='binomial')
```

`forestplotUVMV` *Combine an univariable and multivariable forest plot using ggplot2*

Description

This function will take log or logistic regression fit forest plot output from forestplotUV and forestplotMV functions and display the combined adjusted and unadjusted OR or RR for each variable on the appropriate log scale. Please note that total N and reference-level N is taken from unadjusted model.

Usage

```
forestplotUVMV(
  UVmodel,
  MVmodel,
  model = "glm",
  family = NULL,
  digits = getOption("reportRmd.digits", 2),
  orderByRisk = TRUE,
  colours = "default",
  showEst = TRUE,
  rmRef = FALSE,
  logScale = FALSE,
  nxTicks = 5,
  showN = TRUE,
  showEvent = TRUE
)
```

Arguments

`UVmodel`	an UV model object output from the forestplotUV function
`MVmodel`	a MV model object output from the forestplotMV function
`model`	fitted model object
`family`	description of the error distribution and link function to be used in the model. Only used for geeglm
`digits`	number of digits to round to
`orderByRisk`	logical, should the plot be ordered by risk
`colours`	can specify colours for risks less than, 1 and greater than 1.0. Default is red, black, green
`showEst`	logical, should the risks be displayed on the plot in text
`rmRef`	logical, should the reference levels be removed for the plot?
`logScale`	logical, should OR/RR be shown on log scale, defaults to TRUE. See https://doi.org/10.1093/aje/kwr156 for why you may prefer a linear scale.
`nxTicks`	Number of tick marks supplied to the log_breaks function to produce
`showN`	Show number of observations per variable and category
`showEvent`	Show number of events per variable and category

Value

a plot object

Examples

```
data("pembrolizumab")
UVp = forestplotUV(response="orr", covs=c("change_ctdna_group", "sex", "age",
"l_size"), data=pembrolizumab, family='binomial')
MVp = forestplotMV(glm(orr~change_ctdna_group+sex+age+l_size,
data=pembrolizumab,family = 'binomial'))
forestplotUVMV(UVp, MVp)
```

`formatp` *Specific p-value formatting*

Description

If p < 0.001 returns "<0.001", if p < 0.01 returns p to 3 decimal places otherwise returns p to 2 decimal places

Usage

```
formatp(pvalues)
```

Arguments

`pvalues`	a vector of p values

`geoR_boxcoxfit` *Parameter Estimation for the Box-Cox Transformation*

Description

This function is copied from the geoR package which has been removed from the CRAN repository.

Usage

```
geoR_boxcoxfit(object, xmat, lambda, lambda2 = NULL, add.to.data = 0)
```

Arguments

`object`	a vector with the data
`xmat`	a matrix with covariates values. Defaults to rep(1, length(y)).
`lambda`	numerical value(s) for the transformation parameter lambda. Used as the initial value in the function for parameter estimation. If not provided default values are assumed. If multiple values are passed the one with highest likelihood is used as initial value.
`lambda2`	ogical or numerical value(s) of the additional transformation (see DETAILS below). Defaults to NULL. If TRUE this parameter is also estimated and the initial value is set to the absolute value of the minimum data. A numerical value is provided it is used as the initial value. Multiple values are allowed as for lambda.
`add.to.data`	a constant value to be added to the data.

Details

For more information see: https://cran.r-project.org/web/packages/geoR/index.html

ggkmcif *Plot KM and CIF curves with ggplot*

Description

This function will plot a KM or CIF curve with option to add the number at risk. You can specify if you want confidence bands, the hazard ratio, and pvalues, as well as the units of time used.

Usage

```
ggkmcif(
  response,
  cov = NULL,
  data,
  type = NULL,
  pval = TRUE,
  HR = FALSE,
  HR_pval = FALSE,
  conf.curves = FALSE,
  conf.type = "log",
  table = TRUE,
  times = NULL,
  xlab = "Time",
  ylab = NULL,
  main = NULL,
  stratalabs = NULL,
  strataname = nicename(cov),
  stratalabs.table = NULL,
  strataname.table = strataname,
  median.text = FALSE,
  median.lines = FALSE,
  median.CI = FALSE,
  set.time.text = NULL,
  set.time.line = FALSE,
  set.time = 5,
  set.time.CI = FALSE,
  censor.marks = TRUE,
  censor.size = 3,
  censor.stroke = 1.5,
  fsize = 10,
  nsize = 3,
  lsize = 1,
  psize = 3.5,
  median.size = 3,
  median.pos = NULL,
  median.lsize = 1,
  set.size = 3,
  set.pos = NULL,
  set.lsize = 1,
  ylim = c(0, 1),
  col = NULL,
  linetype = NULL,
  xlim = NULL,
  legend.pos = NULL,
  pval.pos = NULL,
  plot.event = 1,
  event = c("col", "linetype"),
  flip.CIF = FALSE,
  cut = NULL,
  eventlabs = NULL,
  event.name = NULL,
  Numbers_at_risk_text = "Numbers at risk",
  HR.digits = 2,
  HR.pval.digits = 3,
  pval.digits = 3,
  median.digits = 3,
  set.time.digits = 3,
  returns = FALSE,
  print.n.missing = TRUE
)
```

Arguments

`response`	character vector with names of columns to use for response
`cov`	String specifying the column name of stratification variable
`data`	dataframe containing your data
`type`	string indicating he type of univariate model to fit. The function will try and guess what type you want based on your response. If you want to override this you can manually specify the type. Options include "KM", and ,"CIF"
`pval`	boolean to specify if you want p-values in the plot (Log Rank test for KM and Gray's test for CIF)
`HR`	boolean to specify if you want hazard ratios included in the plot
`HR_pval`	boolean to specify if you want HR p-values in the plot
`conf.curves`	boolean to specify if you want confidence interval bands
`conf.type`	One of "none"(the default), "plain", "log" , "log-log" or "logit". Only enough of the string to uniquely identify it is necessary. The first option causes confidence intervals not to be generated. The second causes the standard intervals curve +- k *se(curve), where k is determined from conf.int. The log option calculates intervals based on the cumulative hazard or log(survival). The log-log option bases the intervals on the log hazard or log(-log(survival)), and the logit option on log(survival/(1-survival)).
`table`	Logical value. If TRUE, includes the number at risk table
`times`	Numeric vector of times for the x-axis

`xlab`	String corresponding to xlabel. By default is "Time"
`ylab`	String corresponding to ylabel. When NULL uses "Survival probability" for KM curves, and "Probability of an event" for CIF
`main`	String corresponding to main title. When NULL uses Kaplan-Meier Plot s, and "Cumulative Incidence Plot for CIF"
`stratalabs`	string corresponding to the labels of the covariate, when NULL will use the levels of the covariate
`strataname`	String of the covariate name default is nicename(cov)
`stratalabs.table`	String corresponding to the levels of the covariate for the number at risk table, when NULL will use the levels of the covariate. Can use a string of "-" when the labels are long
`strataname.table`	String of the covariate name for the number at risk table default is nicename(cov
`median.text`	boolean to specify if you want the median values added to the legend (or as added text if there are no covariates), for KM only
`median.lines`	boolean to specify if you want the median values added as lines to the plot, for KM only
`median.CI`	boolean to specify if you want the 95\ with the median text (Only for KM)
`set.time.text`	string for the text to add survival at a specified time (eg. year OS)
`set.time.line`	boolean to specify if you want the survival added as lines to the plot at a specified point
`set.time`	Numeric values of the specific time of interest, default is 5 (Multiple values can be entered)
`set.time.CI`	boolean to specify if you want the 95\ interval with the set time text
`censor.marks`	logical value. If TRUE, includes censor marks (only for KM curves)
`censor.size`	size of censor marks, default is 3
`censor.stroke`	stroke of censor marks, default is 1.5
`fsize`	font size
`nsize`	font size for numbers in the numbers at risk table
`lsize`	line size
`psize`	size of the pvalue
`median.size`	size of the median text (Only when there are no covariates)
`median.pos`	vector of length 2 corresponding to the median position (Only when there are no covariates)
`median.lsize`	line size of the median lines
`set.size`	size of the survival at a set time text (Only when there are no covariates)
`set.pos`	vector of length 2 corresponding to the survival at a set point position (Only when there are no covariates)
`set.lsize`	line size of the survival at set points

`ylim`	vector of length 2 corresponding to limits of y-axis. Default to NULL
`col`	vector of colours
`linetype`	vector of line types
`xlim`	vector of length 2 corresponding to limits of x-axis. Default to NULL
`legend.pos`	Can be either a string corresponding to the legend position ("left","top", "right", "bottom", "none") or a vector of length 2 corresponding to the legend position (uses normalized units (ie the c(0.5,0.5) is the middle of the plot))
`pval.pos`	vector of length 2 corresponding to the p-value position
`plot.event`	Which event(s) to plot (1,2, or c(1,2))
`event`	String specifying if the event should be mapped to the colour, or linetype when plotting both events to colour = "col", line type
`flip.CIF`	boolean to flip the CIF curve to start at 1
`cut`	numeric value indicating where to divide a continuous covariate (default is the median)
`eventlabs`	String corresponding to the event type names
`event.name`	String corresponding to the label of the event types
`Numbers_at_risk_text`	String for the label of the number at risk
`HR.digits`	Number of digits printed of the hazard ratio
`HR.pval.digits`	Number of digits printed of the hazard ratio pvalue
`pval.digits`	Number of digits printed of the Gray's/log rank pvalue
`median.digits`	Number of digits printed of the median pvalue
`set.time.digits`	Number of digits printed of the probability at a specified time
`returns`	Logical value returns a list with all ggplot objects in a list
`print.n.missing`	Logical, should the number of missing be shown !Needs to be checked

Details

Note that for proper pdf output of special characters the following code needs to be included in the first chunk of the rmd knitr::opts_chunk$set(dev="cairo_pdf")

Value

Nothing is returned unless returns = TRUE is used. With returns = TRUE, if table=TRUE (the default) a table style graphic with survival plot and number at risk table is returned. Otherwise a plot with the survival curves is returned.

Examples

```
data("pembrolizumab")
# Simple plot without confidence intervals
ggkmcif(response = c('os_time','os_status'),
cov='cohort',
data=pembrolizumab)

# Plot with median survival time
ggkmcif(response = c('os_time','os_status'),
cov='sex',
data=pembrolizumab,
median.text = TRUE,median.lines=TRUE,conf.curves=TRUE)

# Plot with specified survival times and log-log CI
ggkmcif(response = c('os_time','os_status'),
cov='sex',
data=pembrolizumab,
median.text = FALSE,set.time.text = 'mo OS',
set.time = c(12,24),conf.type = 'log-log',conf.curves=TRUE)

# KM plot with 95% CI and censor marks
ggkmcif(c('os_time','os_status'),'sex',data = pembrolizumab, type = 'KM',
HR=TRUE, HR_pval = TRUE, conf.curves = TRUE,conf.type='log-log',
set.time.CI = TRUE, censor.marks=TRUE)
```

`ggkmcif2` *Plot KM and CIF curves with ggplot*

Description

This function will plot a KM or CIF curve with option to add the number at risk. You can specify if you want confidence bands, the hazard ratio, and pvalues, as well as the units of time used.

Usage

```
ggkmcif2(
  response,
  cov = NULL,
  data,
  pval = TRUE,
  conf.curves = FALSE,
  table = TRUE,
  xlab = "Time",
  ylab = NULL,
  col = NULL,
  times = NULL,
  type = NULL,
  plot.event = 1,
  returns = FALSE,
  ...
)
```

Arguments

`response`	character vector with names of columns to use for response
`cov`	String specifying the column name of stratification variable
`data`	dataframe containing your data
`pval`	boolean to specify if you want p-values in the plot (Log Rank test for KM and Gray's test for CIF)
`conf.curves`	boolean to specify if you want confidence interval bands
`table`	Logical value. If TRUE, includes the number at risk table
`xlab`	String corresponding to xlabel. By default is "Time"
`ylab`	String corresponding to ylabel. When NULL uses "Survival
`col`	vector of colours
`times`	Numeric vector of times for the x-axis probability" for KM cuves, and "Probability of an event" for CIF
`type`	string indicating he type of univariate model to fit. The function will try and guess what type you want based on your response. If you want to override this you can manually specify the type. Options include "KM", and ,"CIF"
`plot.event`	Which event(s) to plot (1,2, or c(1,2))
`returns`	boolean indicating if a list with the objects should be returned. Default is FALSE and plot will be printed
`...`	for additional plotting arguments see ggkmcif2Parameters

Details

Note that for proper pdf output of special characters the following code needs to be included in the first chunk of the rmd knitr::opts_chunk$set(dev="cairo_pdf")

Value

ggplot object; if table = F then only curves are output; if table = T then curves and risk table are output together

Examples

```
# Simple plot without confidence intervals
data("pembrolizumab")
ggkmcif2(response = c('os_time','os_status'),
cov='cohort',
data=pembrolizumab)

# Plot with median survival time
ggkmcif2(response = c('os_time','os_status'),
cov='sex',
data=pembrolizumab,
median.text = TRUE,median.lines=TRUE,conf.curves=TRUE)

# Plot with specified survival times and log-log CI
ggkmcif2(response = c('os_time','os_status'),
cov='sex',
data=pembrolizumab,
median.text = FALSE,set.time.text = 'mo OS',
set.time = c(12,24),conf.type = 'log-log',conf.curves=TRUE)

# KM plot with 95% CI and censor marks
ggkmcif2(c('os_time','os_status'),'sex',data = pembrolizumab, type = 'KM',
HR=TRUE, HR_pval = TRUE, conf.curves = TRUE,conf.type='log-log',
set.time.CI = TRUE, censor.marks=TRUE)
```

ggkmcif_paste *Plot KM and CIF curves with ggplot*

Description

This function puts together a survival curve, and a number at risk table

Usage

```
ggkmcif_paste(list_gg)
```

Arguments

`list_gg`	list containing the results of ggkmcif

Value

a gtable with three elements, the survival curve, a spacer and the number at risk table

Examples

```
data("pembrolizumab")
plot <- ggkmcif(response=c('pfs_time','pfs_status'),
data=pembrolizumab,returns = TRUE)

# Highlighting a section of the curve
plot[[1]] <- plot[[1]] +
ggplot2::geom_rect(xmin=4,xmax=8,ymin=0.15,ymax=0.4,alpha=0.01,fill='yellow')

# Putting the curve back together
ggkmcif_paste(plot)
```

hbld *Bold strings in HTML*

Description

Bold strings in HTML

Usage

```
hbld(strings)
```

Arguments

strings	A vector of strings to bold.

lbld *Bold strings in LaTeX*

Description

Bold strings in LaTeX

Usage

```
lbld(strings)
```

Arguments

strings	A vector of strings to bold.

lpvalue *Formats p-values for LaTeX*

Description

Returns <0.001 if pvalue is <0.001. Else rounds the pvalue to specified significant digits. Will bold the p-value if it is <= 0.05

Usage

```
lpvalue(x, sigdigits = 2)
```

Arguments

x	an integer
sigdigits	number of significant digit to report

mvsum *Get multivariate summary dataframe*

Description

Returns a dataframe with the model summary and global p-value for multi-level variables.

Usage

```
mvsum(
  model,
  data,
  digits = getOption("reportRmd.digits", 2),
  showN = TRUE,
  showEvent = TRUE,
  markup = TRUE,
  sanitize = TRUE,
  nicenames = TRUE,
  CIwidth = 0.95,
  vif = TRUE
)
```

Arguments

`model`	fitted model object
`data`	dataframe containing data
`digits`	number of digits to round to
`showN`	boolean indicating sample sizes should be shown for each comparison, can be useful for interactions
`showEvent`	boolean indicating if number of events should be shown. Only available for logistic.
`markup`	boolean indicating if you want latex markup
`sanitize`	boolean indicating if you want to sanitize all strings to not break LaTeX
`nicenames`	boolean indicating if you want to replace . and _ in strings with a space.
`CIwidth`	width for confidence intervals, defaults to 0.95
`vif`	boolean indicating if the variance inflation factor should be included. See details

Details

Global p-values are likelihood ratio tests for lm, glm and polr models. For lme models an attempt is made to re-fit the model using ML and if,successful LRT is used to obtain a global p-value. For coxph models the model is re-run without robust variances with and without each variable and a LRT is presented. If unsuccessful a Wald p-value is returned. For GEE and CRR models Wald global p-values are returned.

If the variance inflation factor is requested (VIF=T) then a generalised VIF will be calculated in the same manner as the car package.

VIF for competing risk models is computed by fitting a linear model with a dependent variable comprised of the sum of the model independent variables and then calculating VIF from this linear model.

References

John Fox & Georges Monette (1992) Generalized Collinearity Diagnostics, Journal of the American Statistical Association, 87:417, 178-183, DOI: 10.1080/01621459.1992.10475190

John Fox and Sanford Weisberg (2019). An R Companion to Applied Regression, Third Edition. Thousand Oaks CA: Sage.

`nestTable` *Combine two table columns into a single column with levels of one nested within levels of the other.*

Description

This function accepts a data frame (via the data argument) and combines two columns into a single column with values from the head_col serving as headers and values of the to_col displayed underneath each header. The resulting table is then passed to outTable for printing and output, to use the grouped table as a data frame specify tableOnly=TRUE. By default the headers will be bolded and the remaining values indented.

Usage

```
nestTable(
  data,
  head_col,
  to_col,
  colHeader = "",
  caption = NULL,
  indent = TRUE,
  boldheaders = TRUE,
  hdr_prefix = "",
  hdr_suffix = "",
  digits = getOption("reportRmd.digits", 2),
  tableOnly = FALSE,
  fontsize
)
```

Arguments

`data`	dataframe
`head_col`	character value specifying the column name with the headers

`to_col`	character value specifying the column name to add the headers into
`colHeader`	character with the desired name of the first column. The default is to leave this empty for output or, for table only output to use the column name 'col1'.
`caption`	table caption
`indent`	Boolean should the original values in the to_col be indented
`boldheaders`	Boolean should the header column values be bolded
`hdr_prefix`	character value that will prefix headers
`hdr_suffix`	character value that will suffix headers
`digits`	number of digits to round numeric columns to, wither a single number or a vector corresponding to the number of numeric columns
`tableOnly`	boolean indicating if the table should be formatted for printing or returned as a data frame
`fontsize`	PDF/HTML output only, manually set the table fontsize

Details

Note that it is possible to combine multiple tables (more than two) with this function.

Value

A character vector of the table source code, unless tableOnly=TRUE in which case a data frame is returned

Examples

```
## Investigate models to predict baseline ctDNA and tumour size and display together
## (not clinically useful!)
data(pembrolizumab)
fit1 <- lm(baseline_ctdna~age+l_size+pdl1,data=pembrolizumab)
m1 <- rm_mvsum(fit1,tableOnly=TRUE)
m1$Response = 'ctDNA'
fit2 <- lm(l_size~age+baseline_ctdna+pdl1,data=pembrolizumab)
m2 <- rm_mvsum(fit2,tableOnly=TRUE)
m2$Response = 'Tumour Size'
nestTable(rbind(m1,m2),head_col='Response',to_col='Covariate')
```

`nicename` *Lean strings for printing*

Description

Returns strings with . and _ replaced by a space. This is nice when printing column names of your dataframe in a report

Usage

```
nicename(strings, check_numbers = TRUE)
```

Arguments

`strings`	vector of strings to give a nice name
`check_numbers`	boolean indicating if numbers with decimals should be checked for and retained.

`niceNum` *Round retaining digits*

Description

Round retaining digits

Usage

```
niceNum(x, digits = 2)
```

Arguments

`x`	a vector
`digits`	numeric

`outTable` *Print tables to PDF/Latex HTML or Word*

Description

Output the table nicely to whatever format is appropriate. This is the output function used by the rm_* printing functions.

Usage

```
outTable(
  tab,
  row.names = NULL,
  to_indent = numeric(0),
  bold_headers = TRUE,
  rows_bold = numeric(0),
  bold_cells = NULL,
  caption = NULL,
  digits = getOption("reportRmd.digits", 2),
  align,
  applyAttributes = TRUE,
  keep.rownames = FALSE,
  nicenames = TRUE,
  fontsize,
  chunk_label,
  format = NULL
)
```

Arguments

`tab`	a table to format
`row.names`	a string specifying the column name to assign to the rownames. If NULL (the default) then rownames are removed.
`to_indent`	numeric vector indicating which rows to indent in the first column.
`bold_headers`	boolean indicating if the column headers should be bolded
`rows_bold`	numeric vector indicating which rows to bold
`bold_cells`	array indices indicating which cells to bold. These will be in addition to rows bolded by rows_bold.
`caption`	table caption
`digits`	number of digits to round numeric columns to, wither a single number or a vector corresponding to the number of numeric columns in tab
`align`	string specifying column alignment, defaults to left alignment of the first column and right alignment of all other columns. The align argument accepts a single string with 'l' for left, 'c' for centre and 'r' for right, with no separations. For example, to set the left column to be centred, the middle column right-aligned and the right column left aligned use: align='crl'
`applyAttributes`	boolean indicating if the function should use to_indent and bold_cells formatting attributes. This will only work properly if the dimensions of the table output from rm_covsum, rm_uvsum etc haven't changed.
`keep.rownames`	should the row names be included in the output
`nicenames`	boolean indicating if you want to replace . and _ in strings with a space
`fontsize`	PDF/HTML output only, manually set the table fontsize
`chunk_label`	only used knitting to Word docs to allow cross-referencing
`format`	if specified ('html','latex') will override the global pandoc setting

Details

Entire rows can be bolded, or specific cells. Currently indentation refers to the first column only. By default, underscores in column names are converted to spaces. To disable this set rm_ to FALSE

Value

A character vector of the table source code, unless tableOnly=TRUE in which case a data frame is returned

Examples

```
# To make custom changes or change the fontsize in PDF/HTML
data("pembrolizumab")
tab <- rm_covsum(data=pembrolizumab,maincov = 'change_ctdna_group',
covs=c('age','sex','pdl1','tmb','l_size'),show.tests=TRUE,tableOnly = TRUE)
outTable(tab, fontsize=7)

# To bold columns with the variable names
rows_bold <- c(1,4,7,10,13)
outTable(tab,rows_bold = rows_bold)

# To bold the estimates for male/female
bold_cells <- as.matrix(expand.grid(5:6,1:ncol(tab)))
outTable(tab,bold_cells= bold_cells)

# Output the above table to HTML or LaTeX
#cat(outTable(tab=tab)) #Knits to specified global setting
#cat(outTable(tab, format="html"), file = "tab.html") #HTML output
#cat(outTable(tab, format="latex"), file = "tab.tex") #LaTeX output
```

---

pembrolizumab *Survival data Survival status and ctDNA levels for patients receiving pembrolizumab*

Description

Survival data

Survival status and ctDNA levels for patients receiving pembrolizumab

Usage

```
data('pembrolizumab')
```

Format

A data frame with 94 rows and 15 variables:

id Patient ID

age Age at study entry

sex Patient Sex

cohort Study Cohort

l_size Target lesion size at baseline

pdl1 PD L1 percent

tmb log of TMB

baseline_ctdna Baseline ctDNA

change_ctdna_group Did ctDNA increase or decrease from baseline to cycle 3

orr Objective Response

cbr Clinical Beneficial Response

os_status Overall survival status

os_time Overall survival time in months

pfs_status Progression free survival status

pfs_time Progression free survival time in months

Source

https://www.nature.com/articles/s43018-020-0096-5

`plotuv` *Plot multiple bivariate relationships in a single plot*

Description

This function is designed to accompany `rm_uvsum` as a means of visualising the results, and uses similar syntax.

Usage

```
plotuv(
  response,
  covs,
  data,
  showN = FALSE,
  showPoints = TRUE,
  na.rm = TRUE,
  response_title = NULL,
  return_plotlist = FALSE,
  ncol = 2,
  p_margins = c(0, 0.2, 1, 0.2),
  bpThreshold = 20,
  mixed = TRUE,
  violin = FALSE,
  position = c("dodge", "stack", "fill"),
  use_labels = TRUE
)
```

Arguments

`response`	character vector with names of columns to use for response
`covs`	character vector with names of columns to use for covariates
`data`	dataframe containing your data
`showN`	boolean indicating whether sample sizes should be shown on the plots

`showPoints`	boolean indicating whether individual data points should be shown when n>20 in a category
`na.rm`	boolean indicating whether na values should be shown or removed
`response_title`	character value with title of the plot
`return_plotlist`	boolean indicating that the list of plots should be returned instead of a plot, useful for applying changes to the plot, see details
`ncol`	the number of columns of plots to be display in the ggarrange call, defaults to 2
`p_margins`	sets the TRBL margins of the individual plots, defaults to c(0,0.2,1,.2)
`bpThreshold`	Default is 20, if there are fewer than 20 observations in a category then dotplots, as opposed to boxplots are shown.
`mixed`	should a mix of dotplots and boxplots be shown based on sample size? If false then all categories will be shown as either dotplots, or boxplots according the bpThreshold and the smallest category size
`violin`	Show violin plots instead of boxplots. This will override bpThreshold and mixed.
`position`	for categorical variables how should barplots be presented. Default is "dodge" IF stack is TRUE then n will not be shown.
`use_labels`	boolean, default is true if the variables have label attributes this will be shown in the plot instead of the variable names, or if there are no labels then tidy versions of the variable names will be used. If use_labels=FALSE the variable names will be used.

Details

Plots are displayed as follows: If response is continuous For a numeric predictor scatterplot For a categorical predictor: If 20+ observations available boxplot, otherwise dotplot with median line If response is a factor For a numeric predictor: If 20+ observations available boxplot, otherwise dotplot with median line For a categorical predictor barplot Response variables are shown on the ordinate (y-axis) and covariates on the abscissa (x-axis)

Variable names are replaced by their labels if available, or by tidy versions if not. Set use_labels=FALSE to use the variable names.

Value

a list containing plots for each variable in covs

See Also

`ggplot2::ggplot` and `ggpubr::ggarrange` `replace_plot_labels`

Examples

```
## Run multiple univariate analyses on the pembrolizumab dataset to predict cbr and
## then visualise the relationships.
data("pembrolizumab")
```

```
rm_uvsum(data=pembrolizumab,
response='cbr',covs=c('age','sex','l_size','baseline_ctdna'))
plotuv(data=pembrolizumab,  response='cbr',
covs=c('age','sex','l_size','baseline_ctdna'),showN=TRUE)
```

psthr *Round and paste with parentheses*

Description

Round and paste with parentheses

Usage

```
psthr(x, y = 2)
```

Arguments

x	a numeric vector
y	integer corresponding to the number of digits to round by

pstprn *Paste with parentheses*

Description

Paste with parentheses

Usage

```
pstprn(x)
```

Arguments

x	a vector

`pvalue` *Formats p-values*

Description

Returns <0.001 if pvalue is <0.001. Else rounds the pvalue to specified significant digits

Usage

```
pvalue(x, digits)
```

Arguments

`x`	an integer
`digits`	the number of significant digits to return

`replace_plot_labels` *Replace variable names with labels in ggplot*

Description

If the data stored in a ggplot object has variable labels then this will replace the variable names with the variable labels. If no labels are set then the variable names will be tidied and a nicer version used.

Usage

```
replace_plot_labels(plot)
```

Arguments

`plot`	output from a call to ggplot2

See Also

`set_var_labels()` for setting individual variable labels, `set_labels()` for setting variable labels using a data frame, `extract_labels()` for creating a data frame of all variable labels, `clear_labels()` for removing variable labels

Examples

```
## Not run:
data("pembrolizumab")
p <- ggplot(pembrolizumab,aes(x=change_ctdna_group,y=baseline_ctdna)) +
geom_boxplot()
replace_plot_labels(p)
pembrolizumab <- set_var_labels(pembrolizumab,
change_ctdna_group="Change in ctDNA group")
p <- ggplot(pembrolizumab,aes(x=change_ctdna_group,y=baseline_ctdna)) +
geom_boxplot()
replace_plot_labels(p)
# Can also be used with a pipe, but expression needs to be wrapped in a brace
(ggplot(pembrolizumab,aes(x=change_ctdna_group,y=baseline_ctdna)) +
geom_boxplot()) |> replace_plot_labels()

## End(Not run)
```

---

rmds *Replace dollar signs with html for proper HTML output*

Description

Replace dollar signs with html for proper HTML output

Usage

```
rmds(s)
```

Arguments

`s`	a character vector

---

rm_cifsum *Summarize cumulative incidence by group*

Description

Displays event counts and event rates at specified time points for the entire cohort and by group. Gray's test of differences in cumulative incidence is displayed.

Usage

```
rm_cifsum(
  data,
  time,
  status,
  group = NULL,
  eventcode = 1,
  cencode = 0,
  eventtimes,
  eventtimeunit,
  eventtimeLbls = NULL,
  CIwidth = 0.95,
  unformattedp = FALSE,
  na.action = "na.omit",
  showCounts = TRUE,
  showGraystest = TRUE,
  digits = 2,
  caption = NULL,
  tableOnly = FALSE
)
```

Arguments

`data`	data frame containing survival data
`time`	string indicating survival time variable
`status`	string indicating event status variable; must have at least 3 levels, e.g. 0 = censor, 1 = event, 2 = competing risk
`group`	string or character vector indicating the variable to group observations by
`eventcode`	numerical variable indicating event, default is 1
`cencode`	numerical variable indicating censored observation, default is 0
`eventtimes`	numeric vector specifying when event probabilities should be calculated
`eventtimeunit`	unit of time to suffix to the time column label if event probabilities are requested, should be plural
`eventtimeLbls`	if supplied, a vector the same length as eventtimes with descriptions (useful for displaying years with data provided in months)
`CIwidth`	width of the event probabilities, default is 95%
`unformattedp`	boolean indicating if you would like the p-value to be returned unformatted (ie not rounded or prefixed with '<'). Should be used in conjunction with the digits argument.
`na.action`	default is to omit missing values, but can be set to throw and error using na.action='na.fail'
`showCounts`	boolean indicating if the at risk, events and censored columns should be output, default is TRUE
`showGraystest`	boolean indicating Gray's test should be included in the final table, default is TRUE

`digits`	the number of digits to report in the event probabilities, default is 2.
`caption`	table caption for markdown output
`tableOnly`	should a dataframe or a formatted object be returned

Value

A character vector of the event table source code, unless tableOnly=TRUE in which case a data frame is returned

Examples

```
library(survival)
data(pbc)

# Event probabilities at various time points with replacement time labels
rm_cifsum(data=pbc,time='time',status='status',
eventtimes=c(1825,3650),eventtimeLbls=c(5,10),eventtimeunit='yr')

# Event probabilities by one group
rm_cifsum(data=pbc,time='time',status='status',group='trt',
eventtimes=c(1825,3650),eventtimeunit='day')


# Event probabilities by multiple groups
rm_cifsum(data=pbc,time='time',status='status',group=c('trt','sex'),
eventtimes=c(1825,3650),eventtimeunit='day')
```

`rm_compactsum` *Output a compact summary table*

Description

Outputs a table formatted for pdf, word or html output with summary statistics

Usage

```
rm_compactsum(
  data,
  xvars,
  grp,
  use_mean,
  caption = NULL,
  tableOnly = FALSE,
  covTitle = "",
  digits = 1,
  digits.cat = 0,
  nicenames = TRUE,
  iqr = TRUE,
  all.stats = FALSE,
  pvalue = TRUE,
  effSize = FALSE,
  p.adjust = "none",
  unformattedp = FALSE,
  show.sumstats = FALSE,
  show.tests = FALSE,
  full = TRUE,
  percentage = "col"
)
```

Arguments

`data`	dataframe containing data
`xvars`	character vector with the names of covariates to include in table
`grp`	character with the name of the grouping variable
`use_mean`	logical indicating whether mean and standard deviation will be returned for continuous variables instead of median. Otherwise, can specify for individual variables using a character vector containing the names of covariates to return mean and sd for (if use_mean is not supplied, all covariates will have median summaries). See examples.
`caption`	character containing table caption (default is no caption)
`tableOnly`	logical, if TRUE then a dataframe is returned, otherwise a formatted printed object is returned (default is FALSE)
`covTitle`	character with the name of the covariate (predictor) column. The default is to leave this empty for output or, for table only output to use the column name 'Covariate'
`digits`	numeric specifying the number of digits for summarizing mean data. Digits can be specified for individual variables using a named vector in the format digits=c("var1"=2,"var2"=3). If a variable is not in the vector the default will be used for it (default is 1). See examples
`digits.cat`	numeric specifying the number of digits for the proportions when summarizing categorical data (default is 0)
`nicenames`	logical indicating if you want to replace . and _ in strings . with a space
`iqr`	logical indicating if you want to display the interquartile range (Q1-Q3) as opposed to (min-max) in the summary for continuous variables
`all.stats`	logical indicating if all summary statistics (Q1, Q3 + min, max on a separate line) should be displayed. Overrides iqr
`pvalue`	logical indicating if you want p-values included in the table
`effSize`	logical indicating if you want effect sizes and their 95% confidence intervals included in the table. Effect sizes calculated include Cramer's V for categorical variables, and Cohen's d, Wilcoxon r, Epsilon-squared, or Omega-squared for numeric/continuous variables
`p.adjust`	p-adjustments to be performed

`unformattedp`	logical indicating if you would like the p-value to be returned unformatted (ie. not rounded or prefixed with '<'). Best used with tableOnly = T and outTable function. See examples
`show.sumstats`	logical indicating if the type of statistical summary (mean, median, etc) used should be shown.
`show.tests`	logical indicating if the type of statistical test and effect size (if effSize = TRUE) used should be shown in a column beside the p-values.
`full`	logical indicating if you want the full sample included in the table, ignored if grp is not specified
`percentage`	choice of how percentages are presented, either column (default) or row

Details

Comparisons for categorical variables default to chi-square tests, but if there are counts of <5 then the Fisher Exact test will be used. For grouping variables with two levels, either t-tests (mean) or wilcoxon tests (median) will be used for numerical variables. Otherwise, ANOVA (mean) or Kruskal- Wallis tests will be used. The statistical test used can be displayed by specifying show.tests = TRUE. Statistical tests and effect sizes for grp and/ or xvars with less than 2 counts in any level will not be shown.

Effect sizes are calculated as Cohen d for between group differences if the variable is summarised with the mean, otherwise Wilcoxon R if summarised with a median. Cramer's V is used for categorical variables, omega is used for differences in means among more than two groups and epsilon for differences in medians among more than two groups. Confidence intervals are calculated using bootstrapping.

tidyselect can only be used for xvars and grp arguments. Additional arguments (digits, use_mean) must be passed in using characters if variable names are used.

Value

A character vector of the table source code, unless tableOnly = TRUE in which case a data frame is returned. The output has the following attribute:

- "description", which describes what is included in the output table and the type of statistical summary for each covariate. When applicable, the types of statistical tests used will be included. If effSize = TRUE, the effect sizes for each covariate will also be mentioned.

References

Smithson, M. (2002). Noncentral Confidence Intervals for Standardized Effect Sizes. (07/140 ed., Vol. 140). SAGE Publications. doi:10.4135/9781412983761.n4

Steiger, J. H. (2004). Beyond the F Test: Effect Size Confidence Intervals and Tests of Close Fit in the Analysis of Variance and Contrast Analysis. Psychological Methods, 9(2), 164–182. doi:10.1037/1082989X.9.2.164

Kelley, T. L. (1935). An Unbiased Correlation Ratio Measure. Proceedings of the National Academy of Sciences - PNAS, 21(9), 554–559. doi:10.1073/pnas.21.9.554

Okada, K. (2013). Is Omega Squared Less Biased? A Comparison of Three Major Effect Size Indices in One-Way ANOVA. Behavior Research Methods, 40(2), 129-147.

Breslow, N. (1970). A generalized Kruskal-Wallis test for comparing K samples subject to unequal patterns of censorship. Biometrika, 57(3), 579-594.

FRITZ, C. O., MORRIS, P. E., & RICHLER, J. J. (2012). Effect Size Estimates: Current Use, Calculations, and Interpretation. Journal of Experimental Psychology. General, 141(1), 2–18. doi:10.1037/a0024338

Examples

```
data("pembrolizumab")
rm_compactsum(data = pembrolizumab, xvars = c("age",
"change_ctdna_group", "l_size", "pdl1"), grp = "sex", use_mean = "age",
digits = c("age" = 2, "l_size" = 3), digits.cat = 1, iqr = TRUE,
show.tests = TRUE)

# Other Examples (not run)
## Include the summary statistic in the variable column
#rm_compactsum(data = pembrolizumab, xvars = c("age",
#"change_ctdna_group"), grp = "sex", use_mean = "age", show.sumstats=TRUE)

## To show effect sizes
#rm_compactsum(data = pembrolizumab, xvars = c("age",
#"change_ctdna_group"), grp = "sex", use_mean = "age", digits = 2,
#effSize = TRUE, show.tests = TRUE)

## To return unformatted p-values
#rm_compactsum(data = pembrolizumab, xvars = c("l_size",
#"change_ctdna_group"), grp = "cohort", effSize = TRUE, unformattedp = TRUE)

## Using tidyselect
#pembrolizumab |> rm_compactsum(xvars = c(age, sex, pdl1), grp = cohort,
#effSize = TRUE)
```

rm_covsum *Outputs a descriptive covariate table*

Description

Returns a data frame corresponding to a descriptive table.

Usage

```
rm_covsum(
  data,
  covs,
  maincov = NULL,
  caption = NULL,
  tableOnly = FALSE,
  covTitle = "",
  digits = 1,
  digits.cat = 0,
  nicenames = TRUE,
  IQR = FALSE,
  all.stats = FALSE,
  pvalue = TRUE,
  effSize = FALSE,
  p.adjust = "none",
  unformattedp = FALSE,
  show.tests = FALSE,
  testcont = c("rank-sum test", "ANOVA"),
  testcat = c("Chi-squared", "Fisher"),
  full = TRUE,
  include_missing = FALSE,
  percentage = c("column", "row"),
  dropLevels = TRUE,
  excludeLevels = NULL,
  numobs = NULL,
  fontsize,
  chunk_label
)
```

Arguments

`data`	dataframe containing data
`covs`	character vector with the names of columns to include in table
`maincov`	covariate to stratify table by
`caption`	character containing table caption (default is no caption)
`tableOnly`	Logical, if TRUE then a dataframe is returned, otherwise a formatted printed object is returned (default).
`covTitle`	character with the names of the covariate (predictor) column. The default is to leave this empty for output or, for table only output to use the column name 'Covariate'.
`digits`	number of digits for summarizing mean data
`digits.cat`	number of digits for the proportions when summarizing categorical data (default: 0)
`nicenames`	boolean indicating if you want to replace . and _ in strings with a space
`IQR`	boolean indicating if you want to display the inter quantile range (Q1,Q3) as opposed to (min,max) in the summary for continuous variables
`all.stats`	boolean indicating if all summary statistics (Q1,Q3 + min,max on a separate line) should be displayed. Overrides IQR.
`pvalue`	boolean indicating if you want p-values included in the table
`effSize`	boolean indicating if you want effect sizes included in the table. Can only be obtained if pvalue is also requested. Effect sizes calculated include Cramer's V for categorical variables, Cohen's d, Wilcoxon r, or Eta-squared for numeric/continuous variables.

`p.adjust`	p-adjustments to be performed. Uses the p.adjust function from base R
`unformattedp`	boolean indicating if you would like the p-value to be returned unformatted (ie not rounded or prefixed with '<'). Best used with tableOnly = T and outTable function. See examples.
`show.tests`	boolean indicating if the type of statistical test and effect size used should be shown in a column beside the pvalues. Ignored if pvalue=FALSE.
`testcont`	test of choice for continuous variables,one of *rank-sum* (default) or *ANOVA*
`testcat`	test of choice for categorical variables,one of *Chi-squared* (default) or *Fisher*
`full`	boolean indicating if you want the full sample included in the table, ignored if maincov is NULL
`include_missing`	Option to include NA values of maincov. NAs will not be included in statistical tests
`percentage`	choice of how percentages are presented, one of *column* (default) or *row*
`dropLevels`	logical, indicating if empty factor levels be dropped from the output, default is TRUE.
`excludeLevels`	a named list of covariate levels to exclude from statistical tests in the form list(varname =c('level1','level2')). These levels will be excluded from association tests, but not the table. This can be useful for levels where there is a logical skip (ie not missing, but not presented). Ignored if pvalue=FALSE.
`numobs`	named list overriding the number of people you expect to have the covariate
`fontsize`	PDF/HTML output only, manually set the table fontsize
`chunk_label`	only used if output is to Word to allow cross-referencing

Details

Comparisons for categorical variables default to chi-square tests, but if there are counts of <5 then the Fisher Exact test will be used and if this is unsuccessful then a second attempt will be made computing p-values using MC simulation. If testcont='ANOVA' then the t-test with unequal variance will be used for two groups and an ANOVA will be used for three or more. The statistical test used can be displayed by specifying show.tests=TRUE.

Effect size can be obtained when p-value is requested.

Further formatting options are available using tableOnly=TRUE and outputting the table with a call to outTable.

A newer version of this function is rm_compactsum which is more flexible and displays few rows of output.

Value

A character vector of the table source code, unless tableOnly=TRUE in which case a data frame is returned

References

Ellis, P.D. (2010) The essential guide to effect sizes: statistical power, meta-analysis, and the interpretation of research results. Cambridge: Cambridge University Press.doi:10.1017/CBO9780511761676

Lakens, D. (2013) Calculating and reporting effect sizes to facilitate cumulative science: a practical primer for t-tests and ANOVAs. Frontiers in Psychology, 4; 863:1-12. doi:10.3389/fpsyg.2013.00863

See Also

`covsum`,`fisher.test`, `chisq.test`, `wilcox.test`, `kruskal.test`, `anova`, `rstatix::cramer_v`, `rstatix:eta_squared`, and `outTable`

Examples

```
data("pembrolizumab")
rm_covsum(data=pembrolizumab, maincov = 'orr',
covs=c('age','sex','pdl1','tmb','l_size','change_ctdna_group'),
show.tests=TRUE)

# To Show Effect Sizes
rm_covsum(data=pembrolizumab, maincov = 'orr',
covs=c('age','sex'),
effSize=TRUE)

# To make custom changes or change the fontsize in PDF/HTML
tab <- rm_covsum(data=pembrolizumab,maincov = 'change_ctdna_group',
covs=c('age','sex','pdl1','tmb','l_size'),show.tests=TRUE,tableOnly = TRUE)
outTable(tab, fontsize=7)

# To return unformatted p-values
tab <- rm_covsum(data=pembrolizumab, maincov = 'orr',
covs=c('age','sex','pdl1','tmb','l_size','change_ctdna_group'),
show.tests=TRUE,unformattedp=TRUE,tableOnly=TRUE)
outTable(tab,digits=5)
outTable(tab,digits=5, applyAttributes=FALSE) # remove bold/indent
```

`rm_mvsum` *Format a regression model nicely for 'Rmarkdown'*

Description

Multivariable (or univariate) regression models are re-formatted for reporting and a global p-value is added for the evaluation of factor variables.

Multivariable (or univariate) regression models are re-formatted for reporting and a global p-value is added for the evaluation of factor variables.

Usage

```
rm_mvsum(
  model,
  data,
  digits = getOption("reportRmd.digits", 2),
  covTitle = "",
  showN = TRUE,
  showEvent = TRUE,
  CIwidth = 0.95,
  vif = TRUE,
  whichp = c("levels", "global", "both"),
  caption = NULL,
  tableOnly = FALSE,
  p.adjust = "none",
  unformattedp = FALSE,
  nicenames = TRUE,
  chunk_label,
  fontsize
)

rm_mvsum(
  model,
  data,
  digits = getOption("reportRmd.digits", 2),
  covTitle = "",
  showN = TRUE,
  showEvent = TRUE,
  CIwidth = 0.95,
  vif = TRUE,
  whichp = c("levels", "global", "both"),
  caption = NULL,
  tableOnly = FALSE,
  p.adjust = "none",
  unformattedp = FALSE,
  nicenames = TRUE,
  chunk_label,
  fontsize
)
```

Arguments

`model`	model fit
`data`	data that model was fit on (an attempt will be made to extract this from the model)
`digits`	number of digits to round estimates to, does not affect p-values
`covTitle`	character with the names of the covariate (predictor) column. The default is to leave this empty for output or, for table only output to use the column name 'Covariate'.

`showN`	boolean indicating sample sizes should be shown for each comparison, can be useful for interactions
`showEvent`	boolean indicating if number of events should be shown. Only available for logistic.
`CIwidth`	width for confidence intervals, defaults to 0.95
`vif`	boolean indicating if the variance inflation factor should be included. See details
`whichp`	string indicating whether you want to display p-values for levels within categorical data ("levels"), global p values ("global"), or both ("both"). Irrelevant for continuous predictors.
`caption`	table caption
`tableOnly`	boolean indicating if unformatted table should be returned
`p.adjust`	p-adjustments to be performed. Uses the p.adjust function from base R
`unformattedp`	boolean indicating if you would like the p-value to be returned unformatted (ie not rounded or prefixed with '<'). Should be used in conjuction with the digits argument.
`nicenames`	boolean indicating if you want to replace . and _ in strings with a space
`chunk_label`	only used if output is to Word to allow cross-referencing
`fontsize`	PDF/HTML output only, manually set the table fontsize

Details

Global p-values are likelihood ratio tests for lm, glm and polr models. For lme models an attempt is made to re-fit the model using ML and if,successful LRT is used to obtain a global p-value. For coxph models the model is re-run without robust variances with and without each variable and a LRT is presented. If unsuccessful a Wald p-value is returned. For GEE and CRR models Wald global p-values are returned. For negative binomial models a deviance test is used.

If the variance inflation factor is requested (VIF=T, default) then a generalised VIF will be calculated in the same manner as the car package.

As of version 0.1.1 if global p-values are requested they will be included in the p-value column.

As of R 4.4.0 profile likelihood confidence intervals will be calculated automatically and there is no longer an option to force Wald tests.

The number of decimals places to display the statistics can be changed with digits, but this will not change the display of p-values. If more significant digits are required for p-values then use tableOnly=TRUE and format as desired.

Global p-values are likelihood ratio tests for lm, glm and polr models. For lme models an attempt is made to re-fit the model using ML and if,successful LRT is used to obtain a global p-value. For coxph models the model is re-run without robust variances with and without each variable and a LRT is presented. If unsuccessful a Wald p-value is returned. For GEE and CRR models Wald global p-values are returned. For negative binomial models a deviance test is used.

If the variance inflation factor is requested (VIF=T) then a generalised VIF will be calculated in the same manner as the car package.

The number of decimals places to display the statistics can be changed with digits, but this will not change the display of p-values. If more significant digits are required for p-values then use tableOnly=TRUE and format as desired.

Value

A character vector of the table source code, unless tableOnly=TRUE in which case a data frame is returned

A character vector of the table source code, unless tableOnly=TRUE in which case a data frame is returned

References

John Fox & Georges Monette (1992) Generalized Collinearity Diagnostics, Journal of the American Statistical Association, 87:417, 178-183, doi:10.1080/01621459.1992.10475190

John Fox and Sanford Weisberg (2019). An R Companion to Applied Regression, Third Edition. Thousand Oaks CA: Sage.

John Fox & Georges Monette (1992) Generalized Collinearity Diagnostics, Journal of the American Statistical Association, 87:417, 178-183, doi:10.1080/01621459.1992.10475190

John Fox and Sanford Weisberg (2019). An R Companion to Applied Regression, Third Edition. Thousand Oaks CA: Sage.

Examples

```
data("pembrolizumab")
glm_fit = glm(change_ctdna_group~sex:age+baseline_ctdna+l_size,
data=pembrolizumab,family = 'binomial')
rm_mvsum(glm_fit)

#linear model with p-value adjustment
lm_fit=lm(baseline_ctdna~age+sex+l_size+tmb,data=pembrolizumab)
rm_mvsum(lm_fit,p.adjust = "bonferroni")
#Coxph
require(survival)
res.cox <- coxph(Surv(os_time, os_status) ~ sex+age+l_size+tmb, data = pembrolizumab)
rm_mvsum(res.cox, vif=TRUE)
```

```
data("pembrolizumab")
glm_fit = glm(change_ctdna_group~sex:age+baseline_ctdna+l_size,
data=pembrolizumab,family = 'binomial')
rm_mvsum(glm_fit)

#linear model with p-value adjustment
lm_fit=lm(baseline_ctdna~age+sex+l_size+tmb,data=pembrolizumab)
rm_mvsum(lm_fit,p.adjust = "bonferroni")
#Coxph
require(survival)
res.cox <- coxph(Surv(os_time, os_status) ~ sex+age+l_size+tmb, data = pembrolizumab)
rm_mvsum(res.cox, vif=TRUE)
```

`rm_survdiff` *Display event counts, expected event counts and logrank test of differences*

Description

This is a wrapper function around the survdiff function to display overall event rates and group-specific rates along with the log-rank test of a difference in survival between groups in a single table suitable for markdown output. Median survival times are included by default but can be removed setting median=FALSE

Usage

```
rm_survdiff(
  data,
  time,
  status,
  covs,
  strata,
  includeVarNames = FALSE,
  digits = 1,
  showCols = c("N", "Observed", "Expected"),
  CIwidth = 0.95,
  conf.type = "log",
  caption = NULL,
  tableOnly = FALSE,
  fontsize
)
```

Arguments

`data`	data frame containing survival data
`time`	string indicating survival time variable
`status`	string indicating event status variable
`covs`	character vector indicating variables to group observations by
`strata`	string indicating the variable to stratify observations by
`includeVarNames`	boolean indicating if the variable names should be included in the output table, default is FALSE
`digits`	the number of digits in the survival rate
`showCols`	character vector indicating which of the optional columns to display, defaults to c('N','Observed','Expected')
`CIwidth`	width of the median survival estimates, default is 95%
`conf.type`	type of confidence interval see `survival::survfit` for details. Default is 'log'.

`caption`	table caption
`tableOnly`	should a dataframe or a formatted object be returned
`fontsize`	PDF/HTML output only, manually set the table fontsize

Value

A character vector of the survival table source code, unless tableOnly=TRUE in which case a data frame is returned

See Also

`survival::survdiff`

Examples

```
#' # Differences between sex
data("pembrolizumab")
rm_survdiff(data=pembrolizumab,time='os_time',status='os_status',
covs='sex',digits=1)

# Differences between sex, stratified by cohort
rm_survdiff(data=pembrolizumab,time='os_time',status='os_status',
covs='sex',strata='cohort',digits=1)
# Differences between sex/cohort groups
rm_survdiff(data=pembrolizumab,time='os_time',status='os_status',
covs=c('sex','cohort'),digits=1)
```

`rm_survsum` *Summarise survival data by group*

Description

Displays event counts, median survival time and survival rates at specified times points for the entire cohort and by group. The logrank test of differences in survival curves is displayed.

Usage

```
rm_survsum(
  data,
  time,
  status,
  group = NULL,
  survtimes = NULL,
  survtimeunit,
  survtimesLbls = NULL,
  CIwidth = 0.95,
  unformattedp = FALSE,
  conf.type = "log",
  na.action = "na.omit",
  showCounts = TRUE,
  showLogrank = TRUE,
  eventProb = FALSE,
  digits = getOption("reportRmd.digits", 2),
  caption = NULL,
  tableOnly = FALSE,
  fontsize
)
```

Arguments

`data`	data frame containing survival data
`time`	string indicating survival time variable
`status`	string indicating event status variable
`group`	string or character vector indicating the variable(s) to group observations by. If this is left as NULL (the default) then summaries are provided for the entire cohort.
`survtimes`	numeric vector specifying when survival probabilities should be calculated.
`survtimeunit`	unit of time to suffix to the time column label if survival probabilities are requested, should be plural
`survtimesLbls`	if supplied, a vector the same length as survtimes with descriptions (useful for displaying years with data provided in months)
`CIwidth`	width of the survival probabilities, default is 95%
`unformattedp`	boolean indicating if you would like the p-value to be returned unformatted (ie not rounded or prefixed with '<'). Should be used in conjunction with the digits argument.
`conf.type`	type of confidence interval see `survival::survfit` for details. Default is 'log'.
`na.action`	default is to omit missing values, but can be set to throw and error using na.action='na.fail'
`showCounts`	boolean indicating if the at risk, events and censored columns should be output; default is TRUE
`showLogrank`	boolean indicating if the log-rank test statistic and p-value should be output; default is TRUE
`eventProb`	boolean indicating if event probabilities, rather than survival probabilities, should be displayed; default is FALSE
`digits`	the number of digits in the survival rate, default is 2, unless the reportRmd.digits option is set
`caption`	table caption for markdown output
`tableOnly`	should a dataframe or a formatted object be returned
`fontsize`	PDF/HTML output only, manually set the table fontsize

Details

This summary table is supplied for simple group comparisons only. To examine differences in groups with stratification see `rm_survdiff`. To summarise differences in survival rates controlling for covariates see `rm_survtime`.

Value

A character vector of the survival table source code, unless tableOnly=TRUE in which case a data frame is returned

See Also

`survival::survfit`

Examples

```
# Simple median survival table
data("pembrolizumab")
rm_survsum(data=pembrolizumab,time='os_time',status='os_status')

# Survival table with yearly survival rates
rm_survsum(data=pembrolizumab,time='os_time',status='os_status',
survtimes=c(12,24),survtimesLbls=1:2, survtimeunit='yr')

#Median survival by group
rm_survsum(data=pembrolizumab,time='os_time',status='os_status',group='sex')

# Survival Summary by cohort, displayed in years
rm_survsum(data=pembrolizumab,time='os_time',status='os_status',
group="cohort",survtimes=seq(12,72,12),
survtimesLbls=seq(1,6,1),
survtimeunit='years')

# Survival Summary by Sex and ctDNA group
rm_survsum(data=pembrolizumab,time='os_time',status='os_status',
group=c('sex','change_ctdna_group'),survtimes=c(12,24),survtimeunit='mo')
```

---

rm_survtime *Display survival rates and events for specified times*

Description

This is a wrapper for the survfit function to output a tidy display for reporting. Either Kaplan Meier or Cox Proportional Hazards models may be used to estimate the survival probabilities.

Usage

```
rm_survtime(
  data,
  time,
  status,
  covs = NULL,
  strata = NULL,
  type = "KM",
  survtimes,
  survtimeunit,
  strata.prefix = NULL,
  survtimesLbls = NULL,
  showCols = c("At Risk", "Events", "Censored"),
  CIwidth = 0.95,
  conf.type = "log",
  na.action = "na.omit",
  showCounts = TRUE,
  digits = getOption("reportRmd.digits", 2),
  caption = NULL,
  tableOnly = FALSE,
  fontsize
)
```

Arguments

`data`	data frame containing survival data
`time`	string indicating survival time variable
`status`	string indicating event status variable
`covs`	character vector with the names of variables to adjust for in coxph fit
`strata`	string indicating the variable to group observations by. If this is left as NULL (the default) then event counts and survival rates are provided for the entire cohort.
`type`	survival function, if no covs are specified defaults to Kaplan-Meier, otherwise the Cox PH model is fit. Use type='PH' to fit a Cox PH model with no covariates.
`survtimes`	numeric vector specifying when survival probabilities should be calculated.
`survtimeunit`	unit of time to suffix to the time column label if survival probabilities are requested, should be plural
`strata.prefix`	character value describing the grouping variable
`survtimesLbls`	if supplied, a vector the same length as survtimes with descriptions (useful for displaying years with data provided in months)
`showCols`	character vector specifying which of the optional columns to display, defaults to c('At Risk','Events','Censored')
`CIwidth`	width of the survival probabilities, default is 95%
`conf.type`	type of confidence interval see `survival::survfit` for details. Default is 'log'.
`na.action`	default is to omit missing values, but can be set to throw and error using na.action='na.fail'
`showCounts`	boolean indicating if the at risk, events and censored columns should be output, default is TRUE
`digits`	the number of digits in the survival rate, default is 2.
`caption`	table caption for markdown output
`tableOnly`	should a dataframe or a formatted object be returned
`fontsize`	PDF/HTML output only, manually set the table fontsize

Details

If covariates are supplied then a Cox proportional hazards model is fit for the entire cohort and each strata. Otherwise the default is for Kaplan-Meier estimates. Setting type = 'PH' will force a proportional hazards model.

Value

A character vector of the survival table source code, unless tableOnly=TRUE in which case a data frame is returned

See Also

`survival::survfit`

Examples

```
# Kaplan-Mieir survival probabilities with time displayed in years
data("pembrolizumab")
rm_survtime(data=pembrolizumab,time='os_time',status='os_status',
strata="cohort",type='KM',survtimes=seq(12,72,12),
survtimesLbls=seq(1,6,1),
survtimeunit='years')

# Cox Proportional Hazards survivial probabilities
rm_survtime(data=pembrolizumab,time='os_time',status='os_status',
strata="cohort",type='PH',survtimes=seq(12,72,12),survtimeunit='months')

# Cox Proportional Hazards survivial probabilities controlling for age
rm_survtime(data=pembrolizumab,time='os_time',status='os_status',
covs='age',strata="cohort",survtimes=seq(12,72,12),survtimeunit='months')
```

`rm_uvsum` *Output several univariate models nicely in a single table*

Description

#'A table with the model parameters from running separate univariate models on each covariate. For factors with more than two levels a Global p-value is returned.

A table with the model parameters from running separate univariate models on each covariate. For factors with more than two levels a Global p-value is returned.

Usage

```
rm_uvsum(
  response,
  covs,
  data,
  digits = getOption("reportRmd.digits", 2),
  covTitle = "",
  caption = NULL,
  tableOnly = FALSE,
  removeInf = FALSE,
  p.adjust = "none",
  unformattedp = FALSE,
  whichp = c("levels", "global", "both"),
  chunk_label,
  gee = FALSE,
  id = NULL,
  corstr = NULL,
  family = NULL,
  type = NULL,
  offset = NULL,
  strata = 1,
  nicenames = TRUE,
  showN = TRUE,
  showEvent = TRUE,
  CIwidth = 0.95,
  reflevel = NULL,
  returnModels = FALSE,
  fontsize,
  forceWald = FALSE
)

rm_uvsum(
  response,
  covs,
  data,
  digits = getOption("reportRmd.digits", 2),
  covTitle = "",
  caption = NULL,
  tableOnly = FALSE,
  removeInf = FALSE,
  p.adjust = "none",
  unformattedp = FALSE,
  whichp = c("levels", "global", "both"),
  chunk_label,
  gee = FALSE,
  id = NULL,
  corstr = NULL,
  family = NULL,
  type = NULL,
  offset = NULL,
  strata = 1,
  nicenames = TRUE,
  showN = TRUE,
  showEvent = TRUE,
  CIwidth = 0.95,
  reflevel = NULL,
  returnModels = FALSE,
  fontsize,
  forceWald = FALSE
)
```

Arguments

`response`	string vector with name of response
`covs`	character vector with the names of columns to fit univariate models to
`data`	dataframe containing data
`digits`	number of digits to round estimates and CI to. Does not affect p-values.
`covTitle`	character with the names of the covariate (predictor) column. The default is to leave this empty for output or, for table only output to use the column name 'Covariate'.
`caption`	character containing table caption (default is no caption)
`tableOnly`	boolean indicating if unformatted table should be returned
`removeInf`	boolean indicating if infinite estimates should be removed from the table
`p.adjust`	p-adjustments to be performed. Uses the p.adjust function from base R
`unformattedp`	boolean indicating if you would like the p-value to be returned unformatted (ie not rounded or prefixed with '<'). Should be used in conjunction with the digits argument.
`whichp`	string indicating whether you want to display p-values for levels within categorical data ("levels"), global p values ("global"), or both ("both"). Irrelevant for continuous predictors.
`chunk_label`	only used if output is to Word to allow cross-referencing
`gee`	boolean indicating if gee models should be fit to account for correlated observations. If TRUE then the id argument must specify the column in the data which indicates the correlated clusters.
`id`	character vector which identifies clusters. Only used for geeglm
`corstr`	character string specifying the correlation structure. Only used for geeglm. The following are permitted: '"independence"', '"exchangeable"', '"ar1"', '"unstructured"' and '"userdefined"'
`family`	description of the error distribution and link function to be used in the model. Only used for geeglm
`type`	string indicating the type of univariate model to fit. The function will try and guess what type you want based on your response. If you want to override this you can manually specify the type. Options include "linear", "logistic", "poisson",coxph", "crr", "boxcox", "ordinal", "geeglm"
`offset`	string specifying the offset term to be used for Poisson or negative binomial regression. Example: offset="log(follow_up)"

`strata`	character vector of covariates to stratify by. Only used for coxph and crr
`nicenames`	boolean indicating if you want to replace . and _ in strings with a space
`showN`	boolean indicating if you want to show sample sizes
`showEvent`	boolean indicating if you want to show number of events. Only available for logistic.
`CIwidth`	width of confidence interval, default is 0.95
`reflevel`	manual specification of the reference level. Only used for ordinal regression This will allow you to see which model is not fitting if the function throws an error
`returnModels`	boolean indicating if a list of fitted models should be returned. If this is TRUE then the models will be returned, but the output will be suppressed. In addition to the model elements a data element will be appended to each model so that the fitted data can be examined, if necessary. See Details
`fontsize`	PDF/HTML output only, manually set the table fontsize
`forceWald`	**[Deprecated]** `forceWald = TRUE` is no longer supported; this function will always use profile likelihoods as per the inclusion of the MASS confidence intervals into base from from R 4.4.0

Details

Global p-values are likelihood ratio tests for lm, glm and polr models. For lme models an attempt is made to re-fit the model using ML and if,successful LRT is used to obtain a global p-value. For coxph models the model is re-run without robust variances with and without each variable and a LRT is presented. If unsuccessful a Wald p-value is returned. For GEE and CRR models Wald global p-values are returned.

As of version 0.1.1 if global p-values are requested they will be included in the p-value column.

The number of decimals places to display the statistics can be changed with digits, but this will not change the display of p-values. If more significant digits are required for p-values then use tableOnly=TRUE and format as desired.

tidyselect can only be used for response and covs variables. Additional arguments must be passed in using characters

Global p-values are likelihood ratio tests for lm, glm and polr models. For lme models an attempt is made to re-fit the model using ML and if,successful LRT is used to obtain a global p-value. For coxph models the model is re-run without robust variances with and without each variable and a LRT is presented. If unsuccessful a Wald p-value is returned. For GEE and CRR models Wald global p-values are returned.

The number of decimals places to display the statistics can be changed with digits, but this will not change the display of p-values. If more significant digits are required for p-values then use tableOnly=TRUE and format as desired.

Value

A character vector of the table source code, unless tableOnly=TRUE in which case a data frame is returned

A character vector of the table source code, unless tableOnly=TRUE in which case a data frame is returned

See Also

uvsum,lm,glm, cmprsk::crr, survival::coxph, nlme::lme, geepack::geeglm, MASS::glm.nb
covsum,fisher.test, chisq.test, wilcox.test, kruskal.test, anova, rstatix::cramer_v, rstatix:eta_squared, and outTable

Examples

```
# Examples are for demonstration and are not meaningful
# Coxph model with 90% CI
data("pembrolizumab")
rm_uvsum(response = c('os_time','os_status'),
covs=c('age','sex','baseline_ctdna','l_size','change_ctdna_group'),
data=pembrolizumab,CIwidth=.9)

# Linear model with default 95% CI
rm_uvsum(response = 'baseline_ctdna',
covs=c('age','sex','l_size','pdl1','tmb'),
data=pembrolizumab)

# Logistic model with default 95% CI
rm_uvsum(response = 'os_status',
covs=c('age','sex','l_size','pdl1','tmb'),
data=pembrolizumab,family = binomial)
# Poisson models returned as model list
mList <- rm_uvsum(response = 'baseline_ctdna',
covs=c('age','sex','l_size','pdl1','tmb'),
data=pembrolizumab, returnModels=TRUE)
#'
# GEE on correlated outcomes
data("ctDNA")
rm_uvsum(response = 'size_change',
covs=c('time','ctdna_status'),
gee=TRUE,
id='id', corstr="exchangeable",
family=gaussian("identity"),
data=ctDNA,showN=TRUE)

# Using tidyselect
pembrolizumab |> rm_uvsum(response = sex,
covs = c(age, cohort))
# Examples are for demonstration and are not meaningful
# Coxph model with 90% CI
data("pembrolizumab")
rm_uvsum(response = c('os_time','os_status'),
covs=c('age','sex','baseline_ctdna','l_size','change_ctdna_group'),
data=pembrolizumab,CIwidth=.9)

# Linear model with default 95% CI
rm_uvsum(response = 'baseline_ctdna',
covs=c('age','sex','l_size','pdl1','tmb'),
data=pembrolizumab)
```

```
# Logistic model with default 95% CI
rm_uvsum(response = 'os_status',
covs=c('age','sex','l_size','pdl1','tmb'),
data=pembrolizumab,family = binomial)
# Poisson models returned as model list
mList <- rm_uvsum(response = 'baseline_ctdna',
covs=c('age','sex','l_size','pdl1','tmb'),
data=pembrolizumab, returnModels=TRUE)
#'
# GEE on correlated outcomes
data("ctDNA")
rm_uvsum(response = 'size_change',
covs=c('time','ctdna_status'),
gee=TRUE,
id='id', corstr="exchangeable",
family=gaussian("identity"),
data=ctDNA,showN=TRUE)
```

`rm_uv_mv` *Combine univariate and multivariable regression tables*

Description

This function will combine rm_uvsum and rm_mvsum outputs into a single table. The tableOnly argument must be set to TRUE when tables to be combined are created. The resulting table will be in the same order as the uvsum table and will contain the same columns as the uvsum and mvsum tables, but the p-values will be combined into a single column. There must be a variable overlapping between the uvsum and mvsum tables and all variables in the mvsum table must also appear in the uvsum table.

Usage

```
rm_uv_mv(
  uvsumTable,
  mvsumTable,
  covTitle = "",
  vif = FALSE,
  showN = FALSE,
  showEvent = FALSE,
  caption = NULL,
  tableOnly = FALSE,
  chunk_label,
  fontsize
)
```

Arguments

`uvsumTable`	Output from rm_uvsum, with tableOnly=TRUE

`mvsumTable`	Output from rm_mvsum, with tableOnly=TRUE
`covTitle`	character with the names of the covariate (predictor) column. The default is to leave this empty for output or, for table only output to use the column name 'Covariate'.
`vif`	boolean indicating if the variance inflation factor should be shown if present in the mvsumTable. Default is FALSE.
`showN`	boolean indicating if sample sizes should be displayed.
`showEvent`	boolean indicating if number of events (dichotomous outcomes) should be displayed.
`caption`	table caption
`tableOnly`	boolean indicating if unformatted table should be returned
`chunk_label`	only used if output is to Word to allow cross-referencing
`fontsize`	PDF/HTML output only, manually set the table fontsize

Value

A character vector of the table source code, unless tableOnly=TRUE in which case a data frame is returned

See Also

`rm_uvsum`,`rm_mvsum`

Examples

```
require(survival)
data("pembrolizumab")
uvTab <- rm_uvsum(response = c('os_time','os_status'),
covs=c('age','sex','baseline_ctdna','l_size','change_ctdna_group'),
data=pembrolizumab,tableOnly=TRUE)
mv_surv_fit <- coxph(Surv(os_time,os_status)~age+sex+
baseline_ctdna+l_size+change_ctdna_group, data=pembrolizumab)
uvTab <- rm_mvsum(mv_surv_fit)

#linear model
uvtab<-rm_uvsum(response = 'baseline_ctdna',
covs=c('age','sex','l_size','pdl1','tmb'),
data=pembrolizumab,tableOnly=TRUE)
lm_fit=lm(baseline_ctdna~age+sex+l_size+tmb,data=pembrolizumab)
mvtab<-rm_mvsum(lm_fit,tableOnly = TRUE)
rm_uv_mv(uvtab,mvtab,tableOnly=TRUE)

#logistic model
uvtab<-rm_uvsum(response = 'os_status',
covs=c('age','sex','l_size','pdl1','tmb'),
data=pembrolizumab,family = binomial,tableOnly=TRUE)
logis_fit<-glm(os_status~age+sex+l_size+pdl1+tmb,data = pembrolizumab,family = 'binomial')
mvtab<-rm_mvsum(logis_fit,tableOnly = TRUE)
rm_uv_mv(uvtab,mvtab,tableOnly=TRUE)
```

`sanitizestr` *Sanitizes strings to not break LaTeX*

Description

Strings with special characters will break LaTeX if returned 'asis' by knitr. This happens every time we use one of the main reportRmd functions. We first sanitize our strings with this function to stop LaTeX from breaking.

Usage

```
sanitizestr(str)
```

Arguments

`str`	a vector of strings to sanitize

`scrolling_table` *Output a scrollable table*

Description

This function accepts the output of a aa call to knitr::kable or reportRmd::outTable and, if the output format is html, will produce a scrollable table. Otherwise a regular table will be output for pandoc/latex

Usage

```
scrolling_table(knitrTable, pixelHeight = 500)
```

Arguments

`knitrTable`	output from a call to knitr::kable or outTable
`pixelHeight`	the height of the scroll box in pixels,defulat is 500

Examples

```
data("pembrolizumab")
tab <- rm_covsum(data=pembrolizumab,maincov = 'change_ctdna_group',
covs=c('age','cohort','sex','pdl1','tmb','l_size'),full=F)
scrolling_table(tab,pixelHeight=300)
```

`set_labels` *Set variable labels*

Description

Assign variable labels to a data.frame from a lookup table.

Usage

```
set_labels(data, names_labels)
```

Arguments

`data`	data frame to be labelled
`names_labels`	data frame with column 1 containing variable names from data and column 2 containing variable labels. Other columns will be ignored.

Details

Useful if variable labels have been imported from a data dictionary. The first column in names_labels must contain the variable name and the second column the variable label. The column names are not used.

If no label is provided then the existing label will not be changed. To remove a label set the label to NA.

See Also

`set_var_labels()` for setting individual variable labels, `extract_labels()` for creating a data frame of all variable labels, `clear_labels()` for removing variable labels

Examples

```
data("ctDNA")
# create data frame with labels
lbls <- data.frame(c1=c('cohort','size_change'),
c2=c('Cancer cohort','Change in tumour size'))
# set labels and return labelled data frame
set_labels(ctDNA,lbls)
```

`set_var_labels` *Set variable labels*

Description

Set variable labels for a data frame using name-label pairs.

Usage

```
set_var_labels(data, ...)
```

Arguments

`data`	data frame containing variables to be labelled
`...`	Name-label pairs the name gives the name of the column in the output and the label is a character vector of length one.

Details

If no label is provided for a variable then the existing label will not be changed. To remove a label set the label to NA.

See Also

`set_labels()` for setting variable labels using a data frame, `extract_labels()` for creating a data frame of all variable labels, `clear_labels()` for removing variable labels

Examples

```
# set labels using name-label pairs
# and return labelled data frame
data("ctDNA")
ctDNA |> set_var_labels(
   ctdna_status="detectable ctDNA",
  cohort="A cohort label")
```

`uvmodels` *Aligning models and parameters*

Description

Aligning models and parameters

Usage

```
uvmodels
```

Format

A data frame with 16 rows and 5 variables:

type model type

family model linking family

gee boolean indicating if gee are used

autoreg_class class of model fit

beta description of output parameter

Source

internal

`uvsum` *Get univariate summary dataframe*

Description

Returns a dataframe corresponding to a univariate regression table

Usage

```
uvsum(
  response,
  covs,
  data,
  digits = getOption("reportRmd.digits", 2),
  id = NULL,
  corstr = NULL,
  family = NULL,
  type = NULL,
  offset = NULL,
  gee = FALSE,
  strata = 1,
  markup = TRUE,
  sanitize = TRUE,
  nicenames = TRUE,
  showN = TRUE,
  showEvent = TRUE,
  CIwidth = 0.95,
  reflevel = NULL,
  returnModels = FALSE,
  forceWald
)
```

Arguments

`response`	string vector with name of response
`covs`	character vector with the names of columns to fit univariate models to
`data`	dataframe containing data
`digits`	number of digits to round to
`id`	character vector which identifies clusters. Used for GEE and coxph models.
`corstr`	character string specifying the correlation structure. Only used for geeglm. The following are permitted: '"independence"', '"exchangeable"', '"ar1"', '"unstructured"' and '"userdefined"'
`family`	specify details of the model used. This argument does not need to be specified and should be used with caution. By default, gaussian errors are used for linear models, the binomial family with logit link is used for logistic regression and poisson with log link is used for poisson regression. This can be specified with the type argument, or will be inferred from the data type. See `family`. Ignored for ordinal and survival regression and if the type argument is not explicitly specified.
`type`	string indicating he type of univariate model to fit. The function will try and guess what type you want based on your response. If you want to override this you can manually specify the type. Options include "linear", "logistic", "poisson", coxph", "crr", "boxcox", "ordinal" and "negbin"
`offset`	string specifying the offset term to be used for Poisson or negative binomial regression. Example: offset="log(follow_up)"
`gee`	boolean indicating if gee models should be fit to account for correlated observations. If TRUE then the id argument must specify the column in the data which indicates the correlated clusters.
`strata`	character vector of covariates to stratify by. Only used for coxph and crr
`markup`	boolean indicating if you want latex markup
`sanitize`	boolean indicating if you want to sanitize all strings to not break LaTeX
`nicenames`	boolean indicating if you want to replace . and _ in strings with a space
`showN`	boolean indicating if you want to show sample sizes
`showEvent`	boolean indicating if you want to show number of events. Only available for logistic.
`CIwidth`	width of confidence interval, default is 0.95
`reflevel`	manual specification of the reference level. Only used for ordinal. This may allow you to debug if the function throws an error.
`returnModels`	boolean indicating if a list of fitted models should be returned.
`forceWald`	boolean indicating if Wald confidence intervals should be used instead of profile likelihood. This is not recommended, but can speed up computations. To use throughout a document use options(reportRmd.forceWald=TRUE)

Details

Univariate summaries for a number of covariates, the type of model can be specified. If unspecified the function will guess the appropriate model based on the response variable.

Confidence intervals are extracted using confint where possible. Otherwise Student t distribution is used for linear models and the Normal distribution is used for proportions.

returnModels can be used to return a list of the univariate models, which will be the same length as covs. The data used to run each model will include all cases with observations on the response and covariate. For gee models the data are re-ordered so that the ids appear sequentially and proper estimates are given.

See Also

`lm`,`glm`,`cmprsk::crr`,`survival::coxph`, `nlme::lme`,`geepack::geeglm`,`MASS::polr`,`MASS::glm.nb`

Index