Package ‘workingfunctions’

May 15, 2025

Title Statistical reporting and visualization for common methods

Version 0.1

Description Statistical reporting and visualization for common methodology used in psychology. Functions to minimize code and automate procedures using common R packages.

Depends R (>= 3.5), ggplot2, tcR

Imports MASS, future, future.apply, gtools, irr, openxlsx, pROC, plyr, psych, reshape2, stringr, xgboost, DescTools, Rmisc, ggfortify, ggrepel, car, sjstats, doSNOW, foreach, ggpubr, gridExtra, scales, pwr, mirt, nlme, MplusAutomation, NLP, openNLP, tm, spelling, lavaan, ggExtra, QuantPsyc, semPlot, purrr, emmeans, ez, htmlwidgets, mixlm, rstatix, rstudioapi, stopwords

License GPL

Encoding UTF-8

LazyData true

Author Dimitrios Zacharatos [aut, cre]

Maintainer Dimitrios Zacharatos <dzacharatos@yahoo.com>

RoxygenNote 7.3.2

Contents

`alpha_diagnostics` *Item total correlation and r drop*

Description

Item total correlation and r drop

Usage

```
alpha_diagnostics(df)
```

Arguments

`df`	dataframe with one dimension

Examples

```
set.seed(12345)
df<-data.frame(matrix(.5,ncol=6,nrow=6))
correlation_martix<-as.matrix(df)
diag(correlation_martix)<-1
df<-round(generate_correlation_matrix(correlation_martix,nrows=1000),0)+5
psych::alpha(df)
alpha_diagnostics(df=df)
```

call_to_string *Model call to string*

Description

Takes a call object and convert it to string

Usage

```
call_to_string(model)
```

Arguments

model	Model object

Examples

```
df<-generate_correlation_matrix()
model<-lm(df$X1~df$X2)
call_to_string(model)
```

cdf *Check dataframe*

Description

dataframe summary

Usage

```
cdf(
  df,
  name_length = (getOption("width")/3),
  digits = 2,
  nuniques = 0,
  parralel = FALSE,
  file = NULL
)
```

Arguments

df	dataframe
name_length	number of characters to be displayed for names
digits	number of rounding digits
nuniques	number of unique items to display
parralel	if TRUE it will run using multiple cores
file	output filename

Examples

```
cdf(df=mtcars,parralel=TRUE)
cdf(df=change_data_type(mtcars,"factor"),nuniques=3)
cdf(df=data.frame(t(mtcars)),file="mtcars",nuniques=10)
cdf(df=mtcars)
cdf(df=generate_missing(mtcars))
cdf(df=infert,nuniques=10)
cdf(df=infert)
df<-data.frame(infert,
               date=seq(as.Date("2010-1-1"),
                    as.Date("2020-1-1"),
                    length.out=nrow(infert)))
cdf(df=df)
```

`cfa_icc_index` *index of items to convert from lavaan to thurstonian order for analysis*

Description

index of items to convert from lavaan to thurstonian order for analysis

Usage

```
cfa_icc_index(nitems, nfactors = 3)
```

Arguments

`nitems`	number of items in the questionnaire
`nfactors`	number of factors

Examples

```
cfa_icc_index(nitems=18,nfactors=3)
```

`change_data_type` *dataframe data type transformations*

Description

dataframe data type transformations

Usage

```
change_data_type(df, type)
```

Arguments

`df`	dataframe
`type`	"character" "numeric" "factor" "factor_character" "character_factor" For "factor_character" if factors are found, are converted to characters For "character_factor" if characters are found, are converted to factors

Examples

```
cdf(df=change_data_type(df=mtcars,"character"))
cdf(df=change_data_type(df=mtcars,"numeric"))
cdf(df=change_data_type(df=mtcars,"factor"))
df<-change_data_type(df=mtcars,"factor")
cdf(df=change_data_type(df=df,"factor_character"))
```

`clear_stopwords` *Remove stopwods*

Description

Remove stopwods

Remove stopwods

Usage

```
clear_stopwords(text, stopwords = stopwords::stopwords("english"))

clear_stopwords(text, stopwords = stopwords::stopwords("english"))
```

Arguments

`text`	character vector
`stopwords`	character words to remove

Examples

```
text1<-"word_one word_two word_three"
text2<-"word_three word_four word_six"
text3<-"All the Lorem Ipsum generators on the Internet tend to repeat predefined
chunks as necessary, making this the first true generator on the Internet."
text4<-"It uses a dictionary of over 200 Latin words, combined with a handful of
model sentence structures, to generate Lorem Ipsum which looks reasonable."
text5<-"The generated Lorem Ipsum is therefore always free from repetition,
injected humour, or non-characteristic words etc."
stopwords<-stopwords::stopwords("english")
text<-c(text1,text2,text3,text4,text5)
clear_stopwords(text,stopwords=stopwords)
text1<-"word_one word_two word_three"
text2<-"word_three word_four word_six"
```

```
text3<-"All the Lorem Ipsum generators on the Internet tend to repeat predefined
chunks as necessary, making this the first true generator on the Internet."
text4<-"It uses a dictionary of over 200 Latin words, combined with a handful of
model sentence structures, to generate Lorem Ipsum which looks reasonable."
text5<-"The generated Lorem Ipsum is therefore always free from repetition,
injected humour, or non-characteristic words etc."
stopwords<-stopwords::stopwords("english")
text<-c(text1,text2,text3,text4,text5)
clear_stopwords(text,stopwords=stopwords)
```

`clear_text` *Clear text*

Description

Clear text

Clear text

Usage

```
clear_text(text)

clear_text(text)
```

Arguments

`text`	character vector

Examples

```
text1<-"word_one word_two word_three"
text2<-"word_three word_four word_six"
text3<-"All the Lorem Ipsum generators on the Internet tend to repeat predefined
chunks as necessary, making this the first true generator on the Internet."
text4<-"It uses a dictionary of over 200 Latin words, combined with a handful of
model sentence structures, to generate Lorem Ipsum which looks reasonable."
text5<-"The generated Lorem Ipsum is therefore always free from repetition,
injected humour, or non-characteristic words etc."
text<-c(text1,text2,text3,text4,text5)
clear_text(text)
text1<-"word_one word_two word_three"
text2<-"word_three word_four word_six"
text3<-"All the Lorem Ipsum generators on the Internet tend to repeat predefined
chunks as necessary, making this the first true generator on the Internet."
text4<-"It uses a dictionary of over 200 Latin words, combined with a handful of
model sentence structures, to generate Lorem Ipsum which looks reasonable."
text5<-"The generated Lorem Ipsum is therefore always free from repetition,
injected humour, or non-characteristic words etc."
text<-c(text1,text2,text3,text4,text5)
clear_text(text)
```

comparison_combinations *Produce combinations for comparisons from dataframe names*

Description

Produce combinations for comparisons from dataframe names

Usage

```
comparison_combinations(df, all_orders = TRUE)
```

Arguments

df	dataframe
all_orders	if TRUE the order of combination is considered i.e. the combination X1 X2 also appears as X2 X1 if FALSE it is assumed that X1 X2 and X2 X1 are the same and only one of them appears

Examples

```
comparison_combinations(generate_correlation_matrix(n=10)[,1:4])
```

compute_ability *Compute subject ability for thurstonian models*

Description

Computes person ability for binary thurstonian coded items for a single dimension

Usage

```
compute_ability(
  response,
  eta,
  gamma,
  lambda,
  psi,
  plot = FALSE,
  map = compute_map(eta = eta, mean = 0, sd = 1)
)
```

Arguments

`response`	item responses
`eta`	eta or ability
`gamma`	gamma or threshold
`lambda`	lambda or loading
`psi`	psi or error
`plot`	if TRUE plots icc curves using the plot_icc_thurstonian function
`map`	vector from compute_map

Examples

```
gamma<-c(0.556,-1.253,-1.729,0.618,0.937,0.295,-0.672,-1.127,-0.446,0.632,1.147,0.498)
psi<-c(2.172,1.883,2.055,1.869,2.231,2.100,1.762,1.803,1.565,1.892,1.794,1.686)
lambda<-c(1.082,1.082,-1.297,-1.297,0.802,0.802,1.083,1.083)
gamma<-gamma[response_dimension(c(1:12),3,c(1,2))]
psi<-psi[response_dimension(c(1:12),3,c(1,2))]
eta<-seq(-6,6,by=0.1)
response1<-c(0,0,0,0,0,0,0,0)
response2<-c(1,1,1,1,1,1,1,1)
response3<-c(1,0,1,0,1,0,1,0)
response4<-c(0,1,0,1,0,1,0,1)
map<-compute_map(eta=eta,mean=0,sd=1)
compute_ability(response1,eta,gamma,lambda,psi,map=map,plot=FALSE)
compute_ability(response2,eta,gamma,lambda,psi,map=map,plot=FALSE)
compute_ability(response3,eta,gamma,lambda,psi,map=map,plot=FALSE)
compute_ability(response4,eta,gamma,lambda,psi,map=map,plot=FALSE)
```

`compute_adjustment` *Compute adjustments*

Description

Compute adjustments

Usage

```
compute_adjustment(a, ntests)
```

Arguments

`a`	alpha criterion
`ntests`	number of tests

Examples

```
compute_adjustment(0.05,100)
```

compute_aggregate *Descriptive statistics*

Description

uses plyr

Usage

```
compute_aggregate(df, iv, file = NULL)
```

Arguments

df	dataframe
iv	index of independent variables
file	output filename

Details

returns xlsx

Examples

```
compute_aggregate(df=mtcars,iv=9)
compute_aggregate(df=mtcars,iv=9:10)
compute_aggregate(df=mtcars,iv=9:11)
compute_aggregate(df=mtcars,iv=9:11,file="descriptives")
```

compute_aov_es *Compute eta and omega*

Description

Computes omega using aov object. Based on http://stats.stackexchange.com/a/126520

Usage

```
compute_aov_es(model, ss = "I")
```

Arguments

model	object aov
ss	Character type of sums of squares "I" "II" "III"

Examples

```
form<-formula(uptake~Treatment)
one_way_between<-aov(form,CO2)
factorial_between<-aov(uptake~Treatment*Type,CO2)
compute_aov_es(model=one_way_between,ss="I")
sjstats::anova_stats(one_way_between,digits=10)
compute_aov_es(model=one_way_between,ss="II")
sjstats::anova_stats(one_way_between,digits=10)
compute_aov_es(model=one_way_between,ss="III")
sjstats::anova_stats(one_way_between,digits=10)
compute_aov_es(model=factorial_between,ss="I")
sjstats::anova_stats(factorial_between,digits=10)
compute_aov_es(model=factorial_between,ss="II")
sjstats::anova_stats(factorial_between,digits=10)
compute_aov_es(model=factorial_between,ss="III")
sjstats::anova_stats(car::Anova(factorial_between,Type=3),digits=10)
```

`compute_confidence_inteval`

Compute confidence interval

Description

Compute confidence interval

Usage

```
compute_confidence_inteval(vector)
```

Arguments

`vector`	vector

Examples

```
set.seed(1)
vector<-rnorm(1000)
compute_confidence_inteval(vector)
```

compute_crosstable *Compute crosstables*

Description

Compute crosstables

Usage

```
compute_crosstable(df, factor_index = NULL, combinations = NULL)
```

Arguments

`df`	dataframe
`factor_index`	index of factors
`combinations`	index of comparisons

Examples

```
combinations<-data.frame(index1=c("vs","am","gear"),index2=c("cyl","cyl","cyl"))
compute_crosstable(df=mtcars,combinations=combinations)
combinations<-data.frame(index1=c("vs","am"),index2=c("cyl","cyl"))
compute_crosstable(df=mtcars,combinations=combinations)
compute_crosstable(df=mtcars,factor_index=8:10)
```

compute_descriptives *Descriptive statistics*

Description

uses psych

Usage

```
compute_descriptives(df, dv, iv = NULL, file = NULL)
```

Arguments

`df`	dataframe
`dv`	index of dependent variables
`iv`	index of independent variables
`file`	output filename

Details

returns xlsx

Examples

```
compute_descriptives(df=mtcars,dv=1:5)
compute_descriptives(df=mtcars,dv=1:2,iv=9:10)
compute_descriptives(df=mtcars,dv=1:2,file="descriptives_no_factor")
compute_descriptives(df=mtcars,dv=1:2,iv=9:10,file="descriptives_factor")
```

---

compute_dissatenuation

Compute dissatenuation

Description

Compute dissatenuation

Usage

```
compute_dissatenuation(variable1, error1, variable2, error2)
```

Arguments

variable1	vector
error1	vector error measurement for variable1
variable2	vector
error2	vector error measurement for variable2

Examples

```
set.seed(1)
compute_dissatenuation(rnorm(10),rnorm(10),rnorm(10),rnorm(10))
```

---

compute_dummy_comparisons

Compute number of dummy comparisons

Description

Compute number of dummy comparisons

Usage

```
compute_dummy_comparisons(items)
```

Arguments

items	number of items per block

Examples

```
compute_dummy_comparisons(1)
compute_dummy_comparisons(2)
compute_dummy_comparisons(3)
compute_dummy_comparisons(4)
compute_dummy_comparisons(5)
compute_dummy_comparisons(6)
```

---

compute_frequencies *Frequencies by levels*

Description

returns frequency proportion percent

Usage

```
compute_frequencies(df, ordered = TRUE, file = NULL)
```

Arguments

df	dataframe
ordered	if TRUE it will output frequencies in descending order
file	output filename

Details

returns xlsx

Examples

```
compute_frequencies(df=generate_missing(generate_factor(nrows=10,ncols=10),missing=5))
compute_frequencies(df=generate_factor())
compute_frequencies(df=generate_factor(),file="descriptives")
```

compute_icc_thurstonian *Compute item characteristic curves for thurstonian models*

Description

Computes icc curves for binary thurstonian coded items for a single dimension

Usage

```
compute_icc_thurstonian(eta, gamma, lambda, psi, plot = FALSE)
```

Arguments

eta	eta or ability
gamma	gamma or threshold
lambda	lambda or loading
psi	psi or error
plot	if TRUE plots icc curves using the plot_icc_thurstonian function

Examples

```
gamma<-c(0.556,-1.253,-1.729,0.618,0.937,0.295,-0.672,-1.127,-0.446,0.632,1.147,0.498)
psi<-c(2.172,1.883,2.055,1.869,2.231,2.100,1.762,1.803,1.565,1.892,1.794,1.686)
lambda<-c(1.082,1.082,-1.297,-1.297,0.802,0.802,1.083,1.083)
gamma<-gamma[response_dimension(c(1:12),3,c(1,2))]
psi<-psi[response_dimension(c(1:12),3,c(1,2))]
eta<-seq(-6,6,by=0.01)
compute_icc_thurstonian(eta=eta,gamma=gamma,lambda=lambda,psi=psi,plot=FALSE)
```

compute_info_1pl *Compute item information for 1PL model*

Description

Compute item information for 1PL model

Usage

```
compute_info_1pl(b, theta)
```

Arguments

b	numeric difficulty parameter
theta	numeric theta

Examples

```
compute_info_1pl(b=1,theta=-3)
compute_info_1pl(b=1,theta=-2)
compute_info_1pl(b=1,theta=-1)
compute_info_1pl(b=1,theta=0)
compute_info_1pl(b=1,theta=1)
compute_info_1pl(b=1,theta=2)
compute_info_1pl(b=1,theta=3)
ti<-compute_info_1pl(b=1,theta=seq(-6,6,by=.01)) # test information
plot(ti,x=seq(-6,6,by=.01))
```

---

compute_info_2pl *Compute item information for 2PL model*

Description

Compute item information for 2PL model

Usage

```
compute_info_2pl(a, b, theta)
```

Arguments

a	numeric discrimination parameter
b	numeric difficulty parameter
theta	numeric theta

Examples

```
compute_info_2pl(a=1.5,b=1,theta=-3)
compute_info_2pl(a=1.5,b=1,theta=-2)
compute_info_2pl(a=1.5,b=1,theta=-1)
compute_info_2pl(a=1.5,b=1,theta=0)
compute_info_2pl(a=1.5,b=1,theta=1)
compute_info_2pl(a=1.5,b=1,theta=2)
compute_info_2pl(a=1.5,b=1,theta=3)
ti<-compute_info_2pl(a=1,b=-2,theta=seq(-6,6,by=.01)) # test information
plot(ti,x=seq(-6,6,by=.01))
ti<-compute_info_2pl(a=2,b=0,theta=seq(-6,6,by=.01)) # test information
plot(ti,x=seq(-6,6,by=.01))
ti<-compute_info_2pl(a=3,b=2,theta=seq(-6,6,by=.01)) # test information
plot(ti,x=seq(-6,6,by=.01))
```

compute_info_3pl *Compute item information for 3PL model*

Description

Compute item information for 3PL model

Usage

```
compute_info_3pl(a, b, g, theta)
```

Arguments

a	numeric discrimination parameter
b	numeric difficulty parameter
g	numeric guessing parameter
theta	numeric theta

Examples

```
compute_info_3pl(a=1.5,b=1,g=.2,theta=-3)
compute_info_3pl(a=1.5,b=1,g=.2,theta=-2)
compute_info_3pl(a=1.5,b=1,g=.2,theta=-1)
compute_info_3pl(a=1.5,b=1,g=.2,theta=0)
compute_info_3pl(a=1.5,b=1,g=.2,theta=1)
compute_info_3pl(a=1.5,b=1,g=.2,theta=2)
compute_info_3pl(a=1.5,b=1,g=.2,theta=3)
ti<-compute_info_3pl(a=1.5,b=1,g=.2,theta=seq(-6,6,by=.01)) # test information
plot(ti,x=seq(-6,6,by=.01))
```

compute_kruskal_wallis_test *Kruskal Wallis test*

Description

Kruskal Wallis test

Usage

```
compute_kruskal_wallis_test(formula, df)
```

Arguments

`formula`	one way formula in form of y~x. It will ignore more complex formulas
`df`	dataframe eta squared ranges between 0 and 1 epsilon squared ranges between 0 and 1 eta squared multiplied by 100 indicates the percentage of variance in the dependent variable explained by the independent variable

Examples

```
form<-formula(bp_before~agegrp)
kruskal.test(formula=form,data=df_blood_pressure)
rcompanion::epsilonSquared(x=df_blood_pressure$bp_before,
                           g=df_blood_pressure$agegrp,
                           group="row",
                           ci=TRUE,
                           conf=0.95,
                           type="perc",
                           R=1000,
                           digits=3)
rstatix::kruskal_effsize(df_blood_pressure,form,ci=TRUE,conf.level=0.95,ci.type="perc",nboot=100)
compute_kruskal_wallis_test(formula=form,df=df_blood_pressure)
```

`compute_kurtosis` *Compute kurtosis*

Description

Compute kurtosis

Usage

```
compute_kurtosis(vector)
```

Arguments

`vector`	vector

Note

b_2 = m_4 / s^4 - 3 = (g_2 + 3) (1 - 1/n)^2 - 3. Used in MINITAB and BMDP.

Examples

```
set.seed(1)
vector<-rnorm(1000)
compute_kurtosis(vector)
e1071::kurtosis(vector)
```

compute_map — *Simulate prior distribution*

Description

Simulate prior distribution

Usage

```
compute_map(eta, mean = 0, sd = 1)
```

Arguments

eta	vector
mean	numeric
sd	numeric

Examples

```
eta<-seq(-6,6,by=0.1)
compute_map(eta=eta,mean=0,sd=1)
```

compute_moving_average — *Moving Average*

Description

compute moving average

Usage

```
compute_moving_average(df, w)
```

Arguments

df	dataframe
w	window

Examples

```
compute_moving_average(df=mtcars,w=5)
```

compute_one_way_test *one way test*

Description

one way test

Usage

```
compute_one_way_test(formula, df, var.equal = TRUE)
```

Arguments

formula	one way formula in form of y~x. It will ignore more complex formulas
df	dataframe eta squared ranges between 0 and 1 epsilon squared ranges between 0 and 1 eta squared multiplied by 100 indicates the percentage of variance in the dependent variable explained by the independent variable
var.equal	if TRUE it assumes equal variances

Note

eta and omega for Welch statistics are not adequately tested and they should not be consulted

Examples

```
form<-formula(bp_before~agegrp)
compute_one_way_test(formula=form,df=df_blood_pressure,var.equal=TRUE)
compute_one_way_test(formula=form,df=df_blood_pressure,var.equal=FALSE)
oneway.test(formula=form,data=df_blood_pressure,var.equal=TRUE)
oneway.test(formula=form,data=df_blood_pressure,var.equal=FALSE)
car::Anova(aov(form,data=df_blood_pressure),type=2)
model<-lm(form,data=df_blood_pressure)
lsr::etaSquared(aov(form,data=df_blood_pressure),type=3,anova=TRUE)
sjstats::anova_stats(model,digits=22)
```

compute_posthoc *Games Howell Tukey post hoc tests*

Description

Based on http://www.psych.yorku.ca/cribbie/6130/games_howell.R

Usage

```
compute_posthoc(y, x)
```

Arguments

`y`	Vector continous variable
`x`	Vector factor

Examples

```
compute_posthoc(y=df_blood_pressure$bp_before,x=df_blood_pressure$agegrp)
compute_posthoc(y=df_blood_pressure$bp_after,x=df_blood_pressure$agegrp)
```

`compute_power_r` *Compute r power curve*

Description

Compute r power curve

Usage

```
compute_power_r(
  n = 100,
  r = NULL,
  sig.level = 0.05,
  alternative = c("two.sided", "less", "greater"),
  title = "",
  base_size = 10
)
```

Arguments

`n`	number of observations
`r`	correlation coefficient
`sig.level`	alpha (type I error probability)
`alternative`	a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less"
`title`	plot title
`base_size`	base font size

Examples

```
compute_power_r(n=100,r=.5,sig.level=.05,alternative=c("two.sided"))
```

compute_power_r_matrix *Compute correlation matrix*

Description

Compute correlation matrix

Usage

```
compute_power_r_matrix(m, ...)
```

Arguments

m	correlation matrix
...	arguments passed to compute_power_r

Examples

```
compute_power_r_matrix(m=stats::cor(mtcars,use="pairwise.complete.obs"),n=100)
```

compute_residual_stats *Residuals for matrices*

Description

Root Mean Squared Residual Number of absolute residuals > 0.05 Proportion of absolute residuals > 0.05. It can either accept a psych EFA model or it can compare two correlation or covariance matrices

Usage

```
compute_residual_stats(model, data = NULL)
```

Arguments

model	psych EFA model. It has to be a correlation or covariance matrix if data is not NULL
data	correlation or covariance matrix

Examples

```
model<-psych::fa(mtcars,nfactors=2,rotate="oblimin",fm="pa",oblique.scores=TRUE)
compute_residual_stats(model)
```

compute_scores *Compute subject ability for thurstonian models*

Description

Computes person ability for binary thurstonian coded items for a single dimension

Usage

```
compute_scores(mydata, ...)
```

Arguments

mydata	item responses
...	arguments passed to compute_ability

Examples

```
gamma<-c(0.556,-1.253,-1.729,0.618,0.937,0.295,-0.672,-1.127,-0.446,0.632,1.147,0.498)
psi<-c(2.172,1.883,2.055,1.869,2.231,2.100,1.762,1.803,1.565,1.892,1.794,1.686)
lambda<-c(1.082,1.082,-1.297,-1.297,0.802,0.802,1.083,1.083)
gamma<-gamma[response_dimension(c(1:12),3,c(1,2))]
psi<-psi[response_dimension(c(1:12),3,c(1,2))]
eta<-seq(-6,6,by=0.1)
map<-compute_map(eta=eta,mean=0,sd=1)
response_df<-data.frame(matrix(nrow=0,ncol=8))
response_df[1,]<-c(0,0,0,0,0,0,0,0)
response_df[2,]<-c(1,1,1,1,1,1,1,1)
response_df[3,]<-c(1,0,1,0,1,0,1,0)
response_df[4,]<-c(0,1,0,1,0,1,0,1)
compute_scores(response_df,eta,gamma,lambda,psi,map=map,plot=FALSE)
```

compute_se_theta *Compute the SE of theta*

Description

Compute the SE of theta

Usage

```
compute_se_theta(info)
```

Arguments

info	numeric information

Examples

```
compute_se_theta(1)
ti<-compute_info_2pl(a=10,b=0,theta=seq(-3,3,by=.01)) # test information
plot(compute_se_theta(ti),x=seq(-3,3,by=.01))
```

---

compute_skewness *Compute skewness*

Description

Compute skewness

Usage

```
compute_skewness(vector)
```

Arguments

vector	vector

Note

b_1 = m_3 / s^3 = g_1 ((n-1)/n)^(3/2). Used in MINITAB and BMDP.

Examples

```
set.seed(1)
vector<-rnorm(1000)
compute_skewness(vector)
e1071::skewness(vector)
```

---

compute_standard *compute standard scores*

Description

compute standard scores

Usage

```
compute_standard(vector, mean = 0, sd = 1, type = "z", input = "non_standard")
```

Arguments

vector	vector
mean	numeric applicable to "uz"
sd	numeric applicable to "uz"
type	"z" "uz" "sten" "t" "stanine" "center" "center_reversed" "percent" "percentile" "scale_zero_one" "normal_density" "cumulative_density" "all"
input	"standard" "non_standard" standard inputs are z scores and non standard are raw scores

Examples

```
vector<-c(rnorm(10),NA,rnorm(10))
compute_standard(vector,type="z")
compute_standard(vector,mean=0,sd=1,type="uz")
compute_standard(vector,type="sten")
compute_standard(vector,type="t")
compute_standard(vector,type="stanine")
compute_standard(vector,type="center")
compute_standard(vector,type="center_reversed")
compute_standard(vector,type="percent")
compute_standard(vector,type="scale_zero_one")
ndf<-compute_standard(seq(-6,6,.01),mean=0,sd=1,type="normal_density")
plot(ndf)
cdf<-compute_standard(ndf,mean=0,sd=1,type="cumulative_density")
plot(cdf)
compute_standard(vector,type="all")
compute_standard(seq(-6,6,.1),type="all",input="standard")
```

---

compute_standard_error *Compute standard error*

Description

Compute standard error

Usage

```
compute_standard_error(vector)
```

Arguments

vector	vector

Examples

```
set.seed(1)
vector<-rnorm(1000)
compute_standard_error(vector)
```

compute_unidimensional_ability
Compute theta for unidimensional models

Description

Compute theta for unidimensional models

Usage

```
compute_unidimensional_ability(
  a,
  b,
  g = NULL,
  d = 1.702,
  u,
  lim_theta = c(-6, 6)
)
```

Arguments

a	numeric vector discrimination parameters
b	numeric vector difficulty parameters
g	numeric vector guessing parameters
d	numeric scaling constant usually it is a value that approximating 1.749
u	numeric vector responses
lim_theta	vector minimum and maximum value of theta

Examples

```
a<-c(0.39,0.45,0.52,0.3,0.35,0.43,0.42,0.44,0.34,0.42)
b<-c(-1.96,-1.9,-1.38,-0.58,0.48,-0.81,-0.35,1.59,1.33,2.93)
u<-c(1,1,1,1,0,0,1,0,1,0)
# SHOULD RETURN 0.48402574251176
compute_unidimensional_ability(a=a,b=b,u=u,d=1.7,g=NULL)
a<-c(1.27,0.9,0.94,0.95,0.55,0.6,0.44,0.4)
b<-c(-0.54,0.18,0.21,1.26,1.73,-0.87,1.72,2.67)
u<-c(1,1,1,1,0,0,0,0)
# SHOULD RETURN 1.04621621510192
compute_unidimensional_ability(a=a,b=b,u=u,d=1.7,g=NULL)
a<-c(0.41,0.32,0.33,1.2,0.63,0.62,0.7,0.61,0.38,0.53,0.6,1.16)
b<-c(-1.4,-1.3,-1.17,0.2,0.71,0.86,-0.12,0.12,2.06,1.38,1.18,-0.33)
u<-c(1,0,1,1,0,0,0,1,1,0,1,0)
# SHOULD RETURN 0.0860506282671103
compute_unidimensional_ability(a=a,b=b,u=u,d=1.7,g=NULL)
```

`compute_unidimensional_theta`
Compute theta for unidimensional models

Description

Compute theta for unidimensional models

Usage

```
compute_unidimensional_theta(a, b = 0, g = 0, i = 1, d = 1.702, theta = 0)
```

Arguments

`a`	numeric discrimination parameter
`b`	numeric difficulty parameter
`g`	numeric guessing parameter
`i`	numeric innatentiveness parameter
`d`	numeric scaling constant usually a value 1.749 or 1.702
`theta`	numeric or vector theta

Note

when scaling constant=1 it has no effect in equation
when innatentiveness=1 and guessing=0 function computes a 2PL score
when innatentiveness=1 and guessing!=0 function computes a 3PL score
when innatentiveness!=1 and guessing!=0 function computes a 4PL score

Examples

```
compute_unidimensional_theta(a=10,b=0)
x<-seq(-3,3,by=.01)
plot(compute_unidimensional_theta(a=5,b=0,theta=x),x=x)
plot(compute_unidimensional_theta(a=5,b=-1,theta=x),x=x)
plot(compute_unidimensional_theta(a=5,b=1,theta=x),x=x)
plot(compute_unidimensional_theta(a=.1,b=0,theta=x),x=x)
plot(compute_unidimensional_theta(a=1,b=0,theta=x),x=x)
plot(compute_unidimensional_theta(a=10,b=0,theta=x),x=x)
plot(compute_unidimensional_theta(a=10,b=0,g=0,theta=x),x=x)
plot(compute_unidimensional_theta(a=10,b=0,g=.1,theta=x),x=x)
plot(compute_unidimensional_theta(a=10,b=0,g=.5,theta=x),x=x)
plot(compute_unidimensional_theta(a=10,b=0,g=0,i=1,theta=x),x=x)
plot(compute_unidimensional_theta(a=10,b=0,g=0,i=.9,theta=x),x=x)
plot(compute_unidimensional_theta(a=10,b=0,g=0,i=.6,theta=x),x=x)
```

compute_y_logistic — *Compute y for logistic function*

Description

This function requires x range to produce a vector with y values

Usage

```
compute_y_logistic(intercept, coefficient, x)
```

Arguments

`intercept`	Numeric
`coefficient`	Numeric
`x`	Numeric

Examples

```
x<--10:10
compute_y_logistic(0,1,x)
compute_y_logistic(0,1,1)
plot(x,compute_y_logistic(0,1,x),type="l");grid();abline(b=0,a=.5)
```

confusion — *Create a confusion matrix from observed and predicted vectors*

Description

Generates a confusion matrix from observed and predicted values.

Usage

```
confusion(observed, predicted)
```

Arguments

`observed`	Vector of observed variables. These are the true class labels.
`predicted`	Vector of predicted variables. These are the predicted class labels.

Details

This function creates a confusion matrix by comparing the observed (true) class labels with the predicted class labels. The confusion matrix is a table that is often used to describe the performance of a classification model.

The function performs the following steps: 1. Identifies the unique class labels from both the observed and predicted vectors. 2. Sorts the class labels in a mixed order (if they are character variables) using ‘gtools::mixedsort‘. 3. Constructs a table to represent the confusion matrix with the sorted class labels as levels.

The output is a confusion matrix,where rows represent the predicted class labels and columns represent the observed class labels.

Examples

```
# Example with numeric observed and predicted values
confusion(observed=c(1,2,3,4,5,10),predicted=c(1,2,3,4,5,11))

# Example with repeated observed and predicted values
confusion(observed=c(1,2,2,2,2),predicted=c(1,1,2,2,2))
```

---

confusion_matrix_percent

Confusion matrix with row and column percent

Description

Generates a confusion matrix from observed and predicted values,including row and column percentages.

Usage

```
confusion_matrix_percent(observed, predicted)
```

Arguments

`observed`	Vector of observed variables. These are the true class labels.
`predicted`	Vector of predicted variables. These are the predicted class labels.

Details

This function creates a confusion matrix by comparing the observed (true) class labels with the predicted class labels. Additionally, it calculates row and column percentages to provide a more detailed performance analysis.

The function performs the following steps: 1. Computes the confusion matrix from the observed and predicted values. 2. Calculates the overall accuracy by dividing the sum of diagonal elements by the total number of observations. 3. Appends row and column sums to the confusion matrix. 4. Computes precision and recall for each class and appends these metrics to the matrix. 5. Returns a formatted data frame with the confusion matrix,row and column percentages,and overall accuracy.

Note

Total measures-Accuracy: (TP+TN)/total
Total measures-Prevalence: (TP+FN)/total
Total measures-Proportion Incorrectly Classified: (FN+FP)/total
Horizontal measures-True Positive Rate-Sensitivity: TP/(TP+FN)
Horizontal measures-True Negative Rate-Specificity: TN/(FP+TN)
Horizontal measures-False Negative Rate-Miss Rate: FN/(TP+FN)
Horizontal measures-False Positive Rate-Fall-out: FP/(FP+TN)
Vertical measures-Positive Predictive value-Precision: TP/(TP+FP)
Vertical measures-Negative Predictive value: TN/(FN+TN)
Vertical measures-False Omission Rate: FN/(FN+TN)
Vertical measures-False Discovery Rate: FP/(TP+FP)

Examples

```
# Example with numeric observed and predicted values
confusion_matrix_percent(observed=c(1,2,3,4,5,10),predicted=c(1,2,3,4,5,11))

# Example with repeated observed and predicted values
confusion_matrix_percent(observed=c(1,2,2,2,2),predicted=c(1,1,2,2,2))

# Example with random observed and predicted values
observed<-factor(round(rnorm(10000,m=10,sd=1)))
predicted<-factor(round(rnorm(10000,m=10,sd=1)))
confusion_matrix_percent(observed,predicted)
```

---

convert_excel_unix_timestamp

Convert UNIX EXCEL timestamp

Description

Convert UNIX EXCEL timestamp

Usage

```
convert_excel_unix_timestamp(timestamp)
```

Arguments

`timestamp`	unix or excel timestamp

Examples

```
convert_excel_unix_timestamp(1)
```

c_bind — *cbind dataframes with unequal lengths or row lengths*

Description

cbind dataframes with unequal lengths or row lengths

Usage

```
c_bind(..., first = TRUE)
```

Arguments

`...`	dataframes or vectors to bind
`first`	Logical

Author(s)

Ananda Mahto

Examples

```
c_bind(rnorm(10),rnorm(11),rnorm(12),rnorm(13))
```

data_frame_index — *dataframe index*

Description

dataframe index

Usage

```
data_frame_index(nrow, ncol)
```

Arguments

`nrow`	number of rows
`ncol`	number of collumns

Examples

```
data_frame_index(5,5)
```

decompose_datetime *Decompose datetime objects to dataframe collumns*

Description

Decompose datetime objects to dataframe collumns

Usage

```
decompose_datetime(
  x,
  format = "",
  origin = "1970-01-01",
  tz = "GMT",
  extended = FALSE,
  breaks = c(-1, 5, 13, 16, 20, 23),
  ...
)
```

Arguments

x	datetime object
format	date time format
origin	Starting date. The default is the unix time origin "1970-01-01"
tz	Timezone
extended	if TRUE it will display additional day time categories WEEKDAY MONTH JULIAN QUARTER DAY_PERIOD
breaks	Numeric vector Breaks define hour of day for classifiying into "Night", "Morning", "Noon", "Afternoon", "Evening".
...	arguments passed to as.POSIXct This argument is used if extended=TRUE

Examples

```
timestamp1<-as.numeric(as.POSIXct(Sys.Date()))
timestamp2<-as.numeric(as.POSIXct(Sys.time()))
d1<-Sys.Date()
d2<-Sys.time()
decompose_datetime(x=d1)
decompose_datetime(x=d2)
decompose_datetime(x=d1,extended=TRUE)
decompose_datetime(x=d2,extended=TRUE)
decompose_datetime(x="01/15/1900",format="%m/%e/%Y")
decompose_datetime(x="01/15/1900",format="%m/%e/%Y",extended=TRUE)
decompose_datetime(x=as.Date(as.POSIXct(10000,origin="1970-01-01")))
decompose_datetime(x=as.Date(as.POSIXct(timestamp1,origin="1970-01-01")),
```

```
                        format="%m/%e/%Y")
decompose_datetime(x=as.Date(as.POSIXct(timestamp2,origin="1970-01-01")),
                        format="%m/%e/%Y")
```

deg2rad *Convert degrees to radians*

Description

Convert degrees to radians

Usage

```
deg2rad(degrees)
```

Arguments

degrees	degrees

Examples

```
deg2rad(180)
```

detach_package *Unload library*

Description

Unload library

Usage

```
detach_package(package)
```

Arguments

package	Package name

`df_admission` *Admission Data*

Description

This data set contains information about graduate admission, including GRE scores, GPA, and the ranking of the undergraduate institution.

Usage

```
df_admission
```

Format

A data frame with 8 rows and 4 variables:

admit Binary variable indicating admission (0 = No, 1 = Yes)

gre GRE (Graduate Record Examination) score

gpa Grade Point Average

rank Ranking of the undergraduate institution (1 = highest, 4 = lowest)

Source

researchpy repo

`df_automotive_data` *Automotive Data*

Description

This data set contains various automotive information including engine location, dimensions, weight, engine type, number of cylinders, and other specifications.

Usage

```
df_automotive_data
```

Format

A data frame with 38 rows and 26 variables:

symboling Symboling code for the vehicle

normalized-losses Normalized losses in the vehicle

make Make of the vehicle

fuel-type Type of fuel used (e.g., gas, diesel)

aspiration Aspiration type (e.g., std, turbo)

num-of-doors Number of doors (e.g., two, four)

body-style Body style (e.g., sedan, hatchback, wagon)

drive-wheels Drive wheels (e.g., fwd, rwd, 4wd)

engine-location Location of the engine (e.g., front, rear)

wheel-base Wheelbase of the vehicle in inches

length Length of the vehicle in inches

width Width of the vehicle in inches

height Height of the vehicle in inches

curb-weight Curb weight of the vehicle in pounds

engine-type Type of engine (e.g., dohc, ohcv, ohc, l)

num-of-cylinders Number of cylinders in the engine

engine-size Size of the engine in cubic inches

fuel-system Fuel system used (e.g., mpfi, 2bbl, mfi, 1bbl)

bore Diameter of the cylinders in the engine

stroke Stroke length of the engine

compression-ratio Compression ratio of the engine

horsepower Horsepower generated by the engine

peak-rpm Peak RPM of the engine

city-mpg Miles per gallon in the city

highway-mpg Miles per gallon on the highway

price Price of the vehicle

Source

Downloaded from Kaggle.com by the user Ramakrishnan Srinivasan. see https://www.kaggle.com/toramky/automobile-dataset

---

`df_blood_pressure` *Blood Pressure Data*

Description

This data set contains blood pressure readings for patients before and after a certain treatment or intervention.

Usage

```
df_blood_pressure
```

Format

A data frame with 30 rows and 5 variables:

patient Unique identifier for each patient

sex Sex of the patient (e.g., Male, Female)

agegrp Age group of the patient (e.g., 30-45, 46-59)

bp_before Blood pressure reading before the intervention

bp_after Blood pressure reading after the intervention

Source

researchpy repo

`df_co2` *Carbon Dioxide Uptake in Grass Plants*

Description

The CO2 data frame has 84 rows and 5 columns of data from an experiment on the cold tolerance of the grass species Echinochloa crus-galli.

Usage

```
df_co2
```

Format

A data frame with 84 rows and 5 variables:

Plant an ordered factor with levels Qn1 < Qn2 < Qn3 < ... < Mc1 giving a unique identifier for each plant). Used as a grouping factor.

Type a factor with levels Quebec Mississippi giving the origin of the plant

Treatment a factor with levels nonchilled chilled

conc a numeric vector of ambient carbon dioxide concentrations (mL/L)

uptake a numeric vector of carbon dioxide uptake rates (in μmol/m^2/sec)

Details

Grouped formulas like `uptake ~ conc | Plant` are useful in lattice graphics and mixed-effect models. The vertical bar (‘|‘) separates the grouping variable. This allows modeling or plotting the response (`uptake`) versus the predictor (`conc`) within each level of `Plant`.

Source

Potvin, C., Lechowicz, M. J. and Tardif, S. (1990) “The statistical analysis of ecophysiological response curves obtained from experiments involving repeated measures”, Ecology, 71, 1389–1400.
Pinheiro, J. C. and Bates, D. M. (2000) Mixed-effects Models in S and S-PLUS, Springer.

Examples

```
data(df_co2)
head(df_co2)
```

`df_crop_yield` *Crop Yield Data*

Description

This data set contains information about crop yields based on different fertilizer types and water conditions.

Usage

```
df_crop_yield
```

Format

A data frame with 20 rows and 3 variables:

Fert Type of fertilizer used (A or B)

Water Watering condition (High or Low)

Yield Crop yield (in unspecified units)

Source

researchpy repo (simulated data, not real)

`df_difficile` *Difficile Data*

Description

This data set contains information about the impact of different doses on libido.

Usage

```
df_difficile
```

Format

A data frame with 15 rows and 3 variables:

person Unique identifier for each person

dose Dose received (e.g., 1, 2, 3)

libido Libido level of the person

Source

researchpy repo

`df_insurance` *Insurance Data*

Description

This data set contains information about insurance charges based on various factors such as age, sex, BMI, number of children, smoking status, and region.

Usage

```
df_insurance
```

Format

A data frame with 19 rows and 7 variables:

age Age of the individual

sex Sex of the individual (e.g., male, female)

bmi Body Mass Index of the individual

children Number of children covered by the insurance

smoker Smoking status (yes or no)

region Region where the individual resides (e.g., southwest, southeast, northwest, northeast)

charges Insurance charges

Source

researchpy repo

`df_ocean` *Big Five Personality Test Dataset*

Description

This dataset contains responses to an interactive online Big Five personality test conducted around 2012. Participants rated themselves on 50 personality statements, and also provided demographic and technical metadata. Responses were collected with informed consent, and missing data is coded as 0.

Usage

```
df_ocean
```

Format

A data frame with 19719 rows and 57 variables:

E1 I am the life of the party.

E2 I don't talk a lot.

E3 I feel comfortable around people.

E4 I keep in the background.

E5 I start conversations.

E6 I have little to say.

E7 I talk to a lot of different people at parties.

E8 I don't like to draw attention to myself.

E9 I don't mind being the center of attention.

E10 I am quiet around strangers.

N1 I get stressed out easily.

N2 I am relaxed most of the time.

N3 I worry about things.

N4 I seldom feel blue.

N5 I am easily disturbed.

N6 I get upset easily.

N7 I change my mood a lot.

N8 I have frequent mood swings.

N9 I get irritated easily.

N10 I often feel blue.

A1 I feel little concern for others.

A2 I am interested in people.

A3 I insult people.

A4 I sympathize with others' feelings.

A5 I am not interested in other people's problems.

A6 I have a soft heart.

A7 I am not really interested in others.

A8 I take time out for others.

A9 I feel others' emotions.

A10 I make people feel at ease.

C1 I am always prepared.

C2 I leave my belongings around.

C3 I pay attention to details.

C4 I make a mess of things.

C5 I get chores done right away.

C6 I often forget to put things back in their proper place.

C7 I like order.

C8 I shirk my duties.

C9 I follow a schedule.

C10 I am exacting in my work.

O1 I have a rich vocabulary.

O2 I have difficulty understanding abstract ideas.

O3 I have a vivid imagination.

O4 I am not interested in abstract ideas.

O5 I have excellent ideas.

O6 I do not have a good imagination.

O7 I am quick to understand things.

O8 I use difficult words.

O9 I spend time reflecting on things.

O10 I am full of ideas.

E1 to E10 Extraversion items

N1 to N10 Neuroticism items

A1 to A10 Agreeableness items

C1 to C10 Conscientiousness items

O1 to O10 Openness to experience items

race Race/ethnic background (1–13, 0 = missing)

age Age (integer; only responses from participants 13 and older included)

engnat Is English your native language? (1 = Yes, 2 = No, 0 = missing)

gender 1 = Male, 2 = Female, 3 = Other, 0 = missing

hand Dominant writing hand: 1 = Right, 2 = Left, 3 = Both, 0 = missing

country Two-letter ISO country code (e.g., "US", "GB")

source How participant came to the test site: 1 = Internal link, 2 = Google, 3 = Facebook, 4 = .edu site, 6 = Other/unknown

Details

Personality items were rated on a five-point Likert scale: #' 1 = Disagree, 3 = Neutral, 5 = Agree. Missing items are coded as 0. **race** Chosen from a drop down menu. 1=Mixed Race, 2=Arctic (Siberian, Eskimo), 3=Caucasian (European), 4=Caucasian (Indian), 5=Caucasian (Middle East), 6=Caucasian (North African, Other), 7=Indigenous Australian, 8=Native American, 9=North East Asian (Mongol, Tibetan, Korean Japanese, etc), 10=Pacific (Polynesian, Micronesian, etc), 11=South East Asian (Chinese, Thai, Malay, Filipino, etc), 12=West African, Bushmen, Ethiopian, 13=Other (0=missed)

age Entered as text (individuals reporting age < 13 were not recorded)

engnat Response to "is English your native language?". 1=yes, 2=no (0=missed)

gender Chosen from a drop down menu. 1=Male, 2=Female, 3=Other (0=missed)

hand "What hand do you use to write with?". 1=Right, 2=Left, 3=Both (0=missed)

On this page users were also asked to confirm that their answers were accurate and could be used for research. Participants who did not were not recorded). Some values were calculated from technical information.

country The participant's technical location. ISO country code.

source How the participant came to the test. Based on HTTP Referer. 1=from another page on the test website, 2=from google, 3=from facebook, 4=from any url with ".edu" in its domain name (e.g. xxx.edu, xxx.edu.au), 6=other source, or HTTP Referer not provided.

In psychological trait theory, the Big Five personality traits, also known as the five-factor model (FFM) and the OCEAN model, is a suggested taxonomy, or grouping, for personality traits, developed from the 1980s onwards. When factor analysis (a statistical technique) is applied to personality survey data, some words used to describe aspects of personality are often applied to the same person. For example, someone described as conscientious is more likely to be described as "always prepared" rather than "messy". This theory is based therefore on semantic associations between words and not on neuropsychological experiments. This theory uses descriptors of common language and suggests five broad dimensions commonly used to describe the human personality and psyche.

The theory identifies five factors:

- **Openness to experience (O)** (inventive/curious vs. consistent/cautious)
- **Conscientiousness (C)** (efficient/organized vs. extravagant/careless)
- **Extraversion (E)** (outgoing/energetic vs. solitary/reserved)
- **Agreeableness (A)** (friendly/compassionate vs. challenging/callous)
- **Neuroticism (N)** (sensitive/nervous vs. resilient/confident)

The five factors are represented using the acronyms OCEAN or CANOE. Beneath each proposed global factor, there are a number of correlated and more specific primary factors. For example, extroversion is typically associated with qualities such as gregariousness, assertiveness, excitement-seeking, warmth, activity, and positive emotions. Family life and the way someone was raised will affect these traits. Twin studies and other research have shown that about half of the variation between individuals results from their genetics and half from their environments. Researchers have found conscientiousness, extroversion, openness to experience, and neuroticism to be relatively stable from childhood through adulthood.

Items are grouped by Big Five traits:

- **Extraversion (E)**: E1 to E10
- **Neuroticism (N)**: N1 to N10
- **Agreeableness (A)**: A1 to A10
- **Conscientiousness (C)**: C1 to C10
- **Openness (O)**: O1 to O10

Negatively keyed items are: E2, E4, E6, E8, E10, N2, N4, A1, A3, A5, A7, C2, C4, C6, C8, O2, O4, O6. These should be reverse-coded prior to scoring.

Source

Collected via an online personality test with informed consent (~2012). Downloaded from Kaggle.com by the user Lucas Greenwell. see https://www.kaggle.com/datasets/lucasgreenwell/ocean-five-factor-personality-test-responses

Examples

```
data(bfi_data)
head(bfi_data)

# Compute Big Five average scores (after reverse scoring)
# library(dplyr)
# bfi_data <- bfi_data %>% mutate(E = rowMeans(select(., E1:E10), na.rm = TRUE))
```

`df_personality` *Personality Dataset*

Description

A dataset containing personality test results.

Usage

```
df_personality
```

Format

A data frame with 22 rows and 44 variables:

pers01 Personality item 1

pers02 Personality item 2

pers03 Personality item 3

pers04 Personality item 4

pers05 Personality item 5

pers06 Personality item 6

pers07 Personality item 7

pers08 Personality item 8

pers09 Personality item 9

pers10 Personality item 10

pers11 Personality item 11

pers12 Personality item 12

pers13 Personality item 13

pers14 Personality item 14

pers15 Personality item 15

pers16 Personality item 16

pers17 Personality item 17

pers18 Personality item 18

pers19 Personality item 19

pers20 Personality item 20

pers21 Personality item 21

pers22 Personality item 22

pers23 Personality item 23

pers24 Personality item 24

pers25 Personality item 25

pers26 Personality item 26

pers27 Personality item 27

pers28 Personality item 28

pers29 Personality item 29

pers30 Personality item 30

pers31 Personality item 31

pers32 Personality item 32

pers33 Personality item 33

pers34 Personality item 34

pers35 Personality item 35

pers36 Personality item 36

pers37 Personality item 37

pers38 Personality item 38

pers39 Personality item 39

pers40 Personality item 40

pers41 Personality item 41

pers42 Personality item 42

pers43 Personality item 43

pers44 Personality item 44

Examples

```
data(df_personality)
head(df_personality)
```

df_responses_state *Responses State Data*

Description

This data set contains simulated state information paired with participant numbers from the responses data set.

Usage

```
df_responses_state
```

Format

A data frame with 28 rows and 2 variables:

Participant Number Unique identifier for each participant

State State code where the participant resides (e.g., MI, OH, CO, CA, MA, WA)

Source

researchpy repo (simulated data, not real)

df_sexual_comp *Sexual Compatibility Data*

Description

This data set contains responses to questions about sexual compatibility, including scores, gender, and age.

Usage

```
df_sexual_comp
```

Format

A data frame with 22 rows and 13 variables:

Q1 Response to question 1

Q2 Response to question 2

Q3 Response to question 3

Q4 Response to question 4

Q5 Response to question 5

Q6 Response to question 6

Q7 Response to question 7

Q8 Response to question 8

Q9 Response to question 9

Q10 Response to question 10

score Total score

gender Gender of the respondent (1 = Male, 2 = Female)

age Age of the respondent

Source

researchpy repo

`df_titanic` *Titanic Dataset*

Description

A dataset containing information about passengers on the Titanic.

Usage

```
df_titanic
```

Format

A data frame with the following variables:

PassengerId Unique identifier for each passenger

survived Survival status (0 = No, 1 = Yes)

pclass Passenger class (1 = 1st, 2 = 2nd, 3 = 3rd)

name Name of the passenger

sex Gender of the passenger

age Age of the passenger

sibsp Number of siblings/spouses aboard the Titanic

parch Number of parents/children aboard the Titanic

ticket Ticket number

fare Passenger fare

cabin Cabin number

embarked Port of embarkation (C = Cherbourg; Q = Queenstown; S = Southampton)

boat Lifeboat number

body Body number

home.dest Home destination

Examples

```
data(df_titanic)
head(df_titanic)
```

`display_upper_lower_triangle`   *Return upper diagonal from one matrix and lower diagonal from another matrix*

Description

Return upper diagonal from one matrix and lower diagonal from another matrix

Usage

```
display_upper_lower_triangle(m_upper, m_lower, diagonal = NA)
```

Arguments

`m_upper`	matrix
`m_lower`	matrix
`diagonal`	if "upper" it returns upper diagonal if "lower" it returns lower diagonal if NA returns NA in diagonal otherwise it returns any value spesified

Examples

```
m1<-matrix(1:9,nrow=3,ncol=3)
m2<-matrix(11:19,nrow=3,ncol=3)
display_upper_lower_triangle(m_upper=m1,m_lower=m2,diagonal="upper")
display_upper_lower_triangle(m_upper=m1,m_lower=m2,diagonal="lower")
display_upper_lower_triangle(m_upper=m1,m_lower=m2,diagonal=NA)
display_upper_lower_triangle(m_upper=m1,m_lower=m2,diagonal=1)
display_upper_lower_triangle(m_upper=m1,m_lower=m2,diagonal=c("X1","X2","X3"))
display_upper_lower_triangle(m_upper=m1,m_lower=m2,diagonal=c(1,2,3))
display_upper_lower_triangle(m_upper=m1,m2)
```

`dotnames`   *Get the names of objects in the arguments*

Description

Get the names of objects in the arguments

Usage

```
dotnames(...)
```

Arguments

...	objects

Author(s)

Ananda Mahto

`drop_levels` *Drops unused factor levels*

Description

Drops unused factor levels

Usage

```
drop_levels(df, factor_index = NULL, minimum_frequency = 5)
```

Arguments

`df`	dataframe
`factor_index`	numeric index of factors. If NULL the function uses is.factor() to discriminate factors
`minimum_frequency`	the minimum frequency each factor will have, levels with frequency bellow or equal to the defined frequency will be renamed "Other"

Examples

```
factor1<-factor(c(rep("A",10),rep("B",10)),levels=c("A","B","C","D"))
factor2<-factor(c(rep("A",10),rep("B",10)),levels=c("A","B","C","D"))
numeric1<-c(1:20)
df<-data.frame(numeric1,factor1,factor2)
df$factor1
drop_levels(df=df,minimum_frequency=9)
drop_levels(df=df,minimum_frequency=10)
```

`dummy_arrange` *Takes a vector with multiple responses and dummy arranges it in a dataframe*

Description

Takes a vector with multiple responses and dummy arranges it in a dataframe

Usage

```
dummy_arrange(vector)
```

Arguments

vector	Vector

Examples

```
vector1<-gsub(" ","",
            generate_multiple_responce_vector(responces=c("Agree","Hi","All"),
            responded=1:3,length=10),fixed=TRUE)
vector2<-gsub(" ","",
            generate_multiple_responce_vector(responces=1:4,responded=1:4,length=10),
            fixed=TRUE)
vector3<-sample(1:4,10,replace=TRUE)
vector4<-sample(LETTERS[1:3],10,replace=TRUE)
dummy_arrange(vector1)
dummy_arrange(vector2)
dummy_arrange(vector3)
dummy_arrange(vector4)
```

`environment_options` *Load environment options*

Description

Load environment options

Usage

```
environment_options()
```

Examples

```
environment_options()
```

excel_confusion_matrix *Write matrix or dataframe to excel sheet*

Description

Usefull for correlation matrices since it uses conditional formatting for matrices

Usage

```
excel_confusion_matrix(
  df,
  workbook,
  title = "Rows: Expected Collumns: Observed"
)
```

Arguments

df	dataframe or matrix
workbook	workbook
title	comment

Examples

```
filename<-"excel_confusion_matrix.xlsx"
if (file.exists(filename)) file.remove(filename)
observed<-factor(round(rnorm(10000,m=10,sd=1)))
predicted<-factor(round(rnorm(10000,m=10,sd=1)))
confusion(observed,predicted)
cm<-confusion_matrix_percent(observed,predicted)
wb<-openxlsx::createWorkbook()
excel_confusion_matrix(cm,wb)
openxlsx::saveWorkbook(wb,invisible(paste(filename)),TRUE)
```

excel_critical_value *Write matrix or dataframe to excel sheet*

Description

Usefull for generic data where conditional formating of a spesific collumn is required

Usage

```
excel_critical_value(
  df,
  workbook,
  sheet = "output",
  title = NULL,
  comment = NULL,
  numFmt = "#0.00",
  critical = NULL
)
```

Arguments

df	dataframe or matrix
workbook	workbook
sheet	sheet
title	title
comment	comment
numFmt	number formatting
critical	list in the form of (collumn1=critical_value1,collumn2=critical_value2...)

Examples

```
comment<-list(mpg="Miles/(US) gallon",
              cyl="Number of cylinders",
              disp="Displacement (cu.in.)",
              hp="Gross horsepower",
              drat="Rear axle ratio",
              wt="Weight (1000 lbs)",
              qsec="1/4 mile time",
              vs="Engine (0=V-shaped,1=straight)",
              am="Transmission (0=automatic,1=manual)",
              gear="Number of forward gears",
              carb="Number of carburetors",
              extra_comment1="test1",
              extra_comment2="test2")
filename<-"excel_critical_value.xlsx"
if (file.exists(filename)) file.remove(filename)
wb<-openxlsx::createWorkbook()
df<-generate_missing(generate_correlation_matrix())
critical<-list(X1="<0.05",X5="<0")
excel_critical_value(df=df,workbook=wb,sheet="critical",comment=list(X1="test"),
                     numFmt="#0.00",critical=critical)
openxlsx::saveWorkbook(wb,invisible(paste(filename)),TRUE)
filename<-"excel_critical_value_comment.xlsx"
if (file.exists(filename)) file.remove(filename)
wb<-openxlsx::createWorkbook()
df<-generate_missing(mtcars)
critical<-list(mpg=">20",am="=0")
```

```
excel_critical_value(df=df,workbook=wb,sheet="critical",comment=comment,
                     numFmt="#0.00",critical=critical)
openxlsx::saveWorkbook(wb,invisible(paste(filename)),TRUE)
filename<-"excel_critical_value_comment_min_max.xlsx"
if (file.exists(filename)) file.remove(filename)
wb<-openxlsx::createWorkbook()
df<-generate_missing(mtcars)
critical<-list(mpg=c(">20","<11"),am="=0")
excel_critical_value(df=df,workbook=wb,sheet="critical",comment=comment,
                     numFmt="#0.00",critical=critical)
openxlsx::saveWorkbook(wb,invisible(paste(filename)),TRUE)
```

---

excel_generic_format *Generic function for creating workbooks and worksheets*

Description

This function is used by excel_matrix and excel_critical_value functions

Usage

```
excel_generic_format(
  df,
  workbook,
  sheet = "output",
  title = NULL,
  comment = NULL,
  numFmt = "#0.00"
)
```

Arguments

df	dataframe or matrix
workbook	workbook
sheet	sheet
title	title
comment	comment
numFmt	number formatting

Examples

```
comment<-list(mpg="Miles/(US) gallon",
              cyl="Number of cylinders",
              disp="Displacement (cu.in.)",
              hp="Gross horsepower",
              drat="Rear axle ratio",
              wt="Weight (1000 lbs)",
```

```
              qsec="1/4 mile time",
              vs="Engine (0=V-shaped,1=straight)",
              am="Transmission (0=automatic,1=manual)",
              gear="Number of forward gears",
              carb="Number of carburetors",
              extra_comment1="test1",
              extra_comment2="test2")
mtcor<-data.frame(cor(mtcars))
filename<-"excel_generic.xlsx"
if (file.exists(filename)) file.remove(filename)
wb<-openxlsx::createWorkbook()
openxlsx::addWorksheet(wb,"sheet")
openxlsx::addWorksheet(wb,"correlation")
openxlsx::writeData(wb,sheet="sheet",x=mtcars,colNames=TRUE,rowNames=TRUE)
openxlsx::writeData(wb,sheet="correlation",x=mtcor,colNames=TRUE,rowNames=TRUE)
excel_generic_format(df=mtcars,workbook=wb,sheet="sheet",title="test",
                     comment=comment,numFmt="#0.00")
excel_generic_format(df=mtcor,workbook=wb,sheet="correlation",title="correlation",
                     comment=comment,numFmt="#0.00")
openxlsx::saveWorkbook(wb,invisible(paste(filename)),TRUE)
```

`excel_matrix` *Write matrix or dataframe to excel sheet*

Description

Usefull for corellation matrices. It uses conditional formatting for matrices,which outlines high and low values using background color

Usage

```
excel_matrix(
  df,
  workbook,
  sheet = "output",
  title = NULL,
  comment = NULL,
  numFmt = "#0.00",
  conditional_formatting = FALSE,
  diagonal = FALSE,
  diagonal_length = nrow(df)
)
```

Arguments

`df`	dataframe or matrix
`workbook`	workbook
`sheet`	sheet

`title` title

`comment` comment

`numFmt` number formatting

`conditional_formatting`
if TRUE it will use conditional formatting

`diagonal` if TRUE it will add background fill to diagonal

`diagonal_length`
length of diagonal for background fill

Examples

```
comment<-list(mpg="Miles/(US) gallon",
              cyl="Number of cylinders",
              disp="Displacement (cu.in.)",
              hp="Gross horsepower",
              drat="Rear axle ratio",
              wt="Weight (1000 lbs)",
              qsec="1/4 mile time",
              vs="Engine (0=V-shaped,1=straight)",
              am="Transmission (0=automatic,1=manual)",
              gear="Number of forward gears",
              carb="Number of carburetors",
              extra_comment1="test1",
              extra_comment2="test2")
mtcor<-data.frame(cor(mtcars))
filename<-"excel_matrix.xlsx"
if (file.exists(filename)) file.remove(filename)
wb<-openxlsx::createWorkbook()
excel_matrix(mtcars,wb,sheet="matrix",comment=comment,
             conditional_formatting=TRUE,diagonal=FALSE)
excel_matrix(mtcars,wb,sheet="diagonal_non_square",comment=comment,
             conditional_formatting=FALSE,diagonal=TRUE)
excel_matrix(mtcars[1:10,1:10],wb,sheet="diagonal_square",comment=comment[1:10],
             conditional_formatting=FALSE,diagonal=TRUE)
excel_matrix(mtcars,wb,sheet="matrix_diagonal_non_square",comment=comment,
             conditional_formatting=TRUE,diagonal=TRUE)
excel_matrix(mtcars[1:10,1:10],wb,sheet="matrix_diagonal_square",comment=comment[1:10],
             conditional_formatting=TRUE,diagonal=TRUE)
excel_matrix(mtcor,wb,sheet="r",comment=comment,
             conditional_formatting=FALSE,diagonal=FALSE)
excel_matrix(mtcor,wb,sheet="conditional_formatting_r",comment=comment,
             conditional_formatting=TRUE,diagonal=TRUE)
openxlsx::saveWorkbook(wb,invisible(paste(filename)),TRUE)
```

`extract_components` *Extract variance components from model*

Description

Extract variance components from model

Usage

```
extract_components(model, title = "")
```

Arguments

model	model containing variance components
title	plot title

Examples

```
design<-expand.grid(time=1:3,item=1:3,person=1:10)
design<-change_data_type(design,type="factor")
design$response<-rowSums(change_data_type(design[,1:2],type="numeric"))+rnorm(90,0,0.1)
model<-mixlm::lm(response~r(time)*r(person)+r(item)*r(person),data=design)
extract_components(model)
```

---

flatten_list *Flatten two dimensional list*

Description

Flatten two dimensional list

Usage

```
flatten_list(mydata)
```

Arguments

mydata	list with two dimensions

generate_comparisons_matrix
Generate comparisons matrix

Description

Generate comparisons matrix

Usage

```
generate_comparisons_matrix(items)
```

Arguments

items	number of items

Examples

```
generate_comparisons_matrix(2)
generate_comparisons_matrix(3)
generate_comparisons_matrix(4)
generate_comparisons_matrix(5)
generate_comparisons_matrix(6)
```

generate_correlation_matrix
Generate dataframe which outputs a predetermined correlation matrix

Description

Generate dataframe which outputs a predetermined correlation matrix

Usage

```
generate_correlation_matrix(correlation_martix, nrows = 10)
```

Arguments

correlation_martix	correlation matrix of resulting dataframe
nrows	number of rows to generate

Examples

```
df<-data.frame(matrix(.999,ncol=2,nrow=2))
correlation_martix<-as.matrix(df)
diag(correlation_martix)<-1
df<-generate_correlation_matrix(correlation_martix,nrows=100)
stats::cor(df)
```

---

generate_data *Generate dataframe with random numbers*

Description

Generate dataframe with random numbers

Usage

```
generate_data(
  nrows = 10,
  ncols = 5,
  mean = 0,
  sd = 1,
  min = 1,
  max = 5,
  type = "normal"
)
```

Arguments

nrows	number of rows to generate
ncols	number of collumns to generate
mean	mean of generated vectors
sd	standard deviation of generated vectors
min	minimum value in generated vector
max	maximum value in generated vector
type	character "normal" "uniform"

Examples

```
generate_data(nrows=10,ncols=5,mean=0,sd=1,type="normal")
generate_data(nrows=10,ncols=5,min=1,max=5,type="uniform")
```

generate_factor *Generate dataframe of factors*

Description

Generate dataframe of factors

Usage

```
generate_factor(vector = LETTERS[1:5], nrows = 2, ncols = 10, type = "random")
```

Arguments

vector	factor pool
nrows	number of rows to generate
ncols	number of collumns to generate
type	"balanced" or "random" "balanced" generates balanced factor vectrors, "random" generates random factor vectors

Examples

```
generate_factor(vector=LETTERS[1:5],ncols=5,nrows=10,type="random")
generate_factor(vector=LETTERS[1:5],ncols=5,nrows=10,type="balanced")
generate_factor(vector=LETTERS[1:5],ncols=1,nrows=10,type="balanced")
generate_factor(vector=LETTERS[1:5],ncols=1,nrows=10,type="random")
```

generate_matrix_A *Generate Matrix A*

Description

Generate Matrix A

Usage

```
generate_matrix_A(blocks = 3, items = 3)
```

Arguments

blocks	number of blocks
items	number of items per block

Examples

```
generate_matrix_A(blocks=3,items=3)
```

`generate_matrix_lambda_hat` *Generate matrix lambda for spesified number of comparisons*

Description

Generate matrix lambda for spesified number of comparisons

Usage

```
generate_matrix_lambda_hat(blocks = 3, items = 3)
```

Arguments

blocks	number of blocks
items	number of items per block

Examples

```
generate_matrix_lambda_hat(blocks=3,items=4)
```

`generate_missing` *Generate missing data*

Description

Generate missing data

Usage

```
generate_missing(df, missing = 5)
```

Arguments

df	vector or dataframe
missing	number of missing data per vector

Examples

```
generate_missing(rnorm(10),missing=5)
generate_missing(generate_data(nrow=10,ncol=2),missing=5)
```

generate_multiple_responce_vector

Generate multiple responce vector

Description

Generate multiple responce vector

Usage

```
generate_multiple_responce_vector(
  responces = 1:4,
  responded = 1:4,
  length = 10
)
```

Arguments

responces	unique categories allowed
responded	number of categories observed in iteration
length	length of returned vector

Examples

```
generate_multiple_responce_vector(responces=1:4,responded=1:4,length=10)
```

generate_string

Generate random strings

Description

Generate random strings

Usage

```
generate_string(
  vector = c(LETTERS, letters, 0:9),
  vector_length = 1,
  nchar = 5
)
```

Arguments

vector	character pool
vector_length	number of strings to generate
nchar	Length of generated strings

Examples

```
generate_string(nchar=10)
generate_string(nchar=10,vector_length=10)
```

`generate_unique_comparisons_index` *Generate index for unique comparisons*

Description

Generate index for unique comparisons

Usage

```
generate_unique_comparisons_index(items)
```

Arguments

`items`	number of items

Examples

```
generate_unique_comparisons_index(1)
generate_unique_comparisons_index(2)
generate_unique_comparisons_index(3)
generate_unique_comparisons_index(4)
generate_unique_comparisons_index(5)
generate_unique_comparisons_index(6)
```

`getfwp` *Get working file path*

Description

Get working file path

Usage

```
getfwp()
```

Examples

```
#getfwp()
```

get_mplus_thu_3t — *Simulate prior distribution*

Description

Simulate prior distribution

Usage

```
get_mplus_thu_3t(model)
```

Arguments

model	mplus thurstonian cfa model with 3 traits

icc_cfa — *Select responses for each dimension*

Description

Select responses for each dimension

Usage

```
icc_cfa(eta, gamma, lambda, psi)
```

Arguments

eta	eta or ability
gamma	gamma or threshold
lambda	lambda or loading
psi	psi or error

Examples

```
icc_cfa(seq(-6,6,.1),1,1,1)
```

`increase_index` *index dataframe picks*

Description

index dataframe picks

Usage

```
increase_index(blocks, items)
```

Arguments

`blocks`	number of blocks
`items`	number of items per block

Examples

```
increase_index(3,3)
```

`install_all_packages` *Install all packages available in CRAN*

Description

Install all packages available in CRAN

Usage

```
install_all_packages()
```

Details

Install all packages available in CRAN. Already installed packages are not downloaded or installed `install_load` *Install and load multiple packages*

Description

Install and load multiple packages. If packages exist,they are loaded,if packages don't exist,they are downloaded installed and loaded

Usage

```
install_load(package)
```

Arguments

`package`	Vector Package names

Author(s)

Steven Worthington

Examples

```
install_load("car")
install_load(c("car","ggplot2"))
```

`key_to_cfa_model` *Converts key to cfa model spesification*

Description

This function uses the key spesification used in report_alpha function and converts the key to a cfa model spesification

Usage

```
key_to_cfa_model(key)
```

Arguments

`key`	index of trait names and items constituring a trait

Examples

```
population_model<-'t1=~x1+.5*x2+.5*x3
                   t2=~x4+.5*x5+.5*x6
                   t3=~x7+.5*x8+.5*x9'
model_data<-lavaan::simulateData(population_model,sample.nobs=1000)
key<-list(f1=paste0("x",1:3),f2=paste0("x",4:6),f3=paste0("x",7:9))
model<-key_to_cfa_model(key)
fit<-lavaan::cfa(model,model_data)
```

---

k_fold *K-Fold train test sampling*

Description

Splits a dataframe into train and test dataframes for model evaluation. Prepared data include data objects for xgboost.

Usage

```
k_fold(df, model_formula, k = 10)
```

Arguments

df	Dataframe containing the dataset to be split.
model_formula	Model formula specifying the predictors and outcome variable.
k	Integer value representing the number of folds. Defaults to 10.

Details

This function performs k-fold cross-validation by splitting the input dataframe into k folds. Each fold serves as a test set once,while the remaining k-1 folds form the training set.

The function prepares data objects for xgboost model training and evaluation,including train/test datasets and xgboost DMatrix objects.

The output is a list containing the following elements: -‘f‘: List of train and test datasets for each fold. -‘index‘: Vector of fold indices. -‘model_formula‘: Model formula used for generating the datasets. -‘variables‘: Names of the variables in the model formula. -‘predictors‘: Names of the predictor variables. -‘outcome‘: Name of the outcome variable. -‘xgb‘: List of xgboost DMatrix objects for training and testing.

Examples

```
# Example with the 'infert' dataset
infert_formula<-formula(case~education+spontaneous+induced)
result<-k_fold(infert,k=10,model_formula=infert_formula)

# Example with the 'mtcars' dataset
model_formula<-as.formula(mpg~cyl+disp+hp+drat+wt+qsec+vs+am+gear+carb)
```

```
result<-k_fold(mtcars,k=2,model_formula=model_formula)
```

`k_sample` *Train test sampling*

Description

Splits a dataframe into train and test dataframes for model evaluation. Prepared data include data objects for xgboost.

Usage

```
k_sample(df, model_formula, k = 1)
```

Arguments

`df`	Dataframe containing the dataset to be split.
`model_formula`	Model formula specifying the predictors and outcome variable.
`k`	Integer value representing the number of folds. Defaults to 1 (train-test split).

Details

This function performs k-fold cross-validation or a simple train-test split (if k=1) by splitting the input dataframe into k folds. Each fold serves as a test set once,while the remaining k-1 folds form the training set.

The function prepares data objects for xgboost model training and evaluation,including train,test,and validation datasets and xgboost DMatrix objects.

The output is a list containing the following elements: -‘f‘: List of train,test,and validation datasets for each fold. -‘index‘: Vector of fold indices. -‘model_formula‘: Model formula used for generating the datasets. -‘variables‘: Names of the variables in the model formula. -‘predictors‘: Names of the predictor variables. -‘outcome‘: Name of the outcome variable. -‘xgb‘: List of xgboost DMatrix objects for training,testing,and validation.

Examples

```
# Example with the 'infert' dataset
infert_formula<-formula(case~education+spontaneous+induced)
result<-k_sample(df=infert,k=10,model_formula=infert_formula)

# Example with the 'mtcars' dataset
model_formula<-formula(mpg~cyl+disp+hp+drat+wt+qsec+vs+am+gear+carb)
result<-k_sample(df=mtcars,k=10,model_formula=model_formula)
```

matrix_triangle *Return upper or lower matrix triangle*

Description

Return upper or lower matrix triangle

Usage

```
matrix_triangle(m, off_diagonal = NA, diagonal = NULL, type = "lower")
```

Arguments

m	matrix
off_diagonal	off diagonal value
diagonal	diagonal value. If NULL it returns the diagonal of the input matrix
type	"upper" displays upper triangle, "lower" displays lower triangle

Examples

```
m<-matrix(1:9,nrow=3,ncol=3)
matrix_triangle(m=m)
matrix_triangle(m=m,diagonal=NA,type="lower")
matrix_triangle(m=m,diagonal=NULL,type="lower")
matrix_triangle(m=m,diagonal=NA,type="upper")
matrix_triangle(m=m,diagonal=NULL,type="upper")
```

mean_sd_alpha *Mean and SD*

Description

Mean and SD

Usage

```
mean_sd_alpha(df, divisor = NULL)
```

Arguments

df	dataframe with one dimension
divisor	number to use for dividing the rowsums

Examples

```
set.seed(12345)
df<-data.frame(matrix(.5,ncol=6,nrow=6))
correlation_martix<-as.matrix(df)
diag(correlation_martix)<-1
df<-round(generate_correlation_matrix(correlation_martix,nrows=1000),0)+5
mean_sd_alpha(df)
mean_sd_alpha(df,divisor=100)
```

`mgsub` *Sub for multiple patterns*

Description

Sub for multiple patterns

Usage

```
mgsub(mydata, pattern, replacement, ...)
```

Arguments

`mydata`	Character
`pattern`	Character to search for
`replacement`	Replacement character
`...`	arguments passed to gsub

Examples

```
mgsub(mydata="#$%^&*_+",pattern=c("%","*"),"REPLACE",fixed=TRUE)
```

`min_max_index` *Return the minimum and maximum index of a vector*

Description

Return the minimum and maximum index of a vector

Usage

```
min_max_index(vector)
```

Arguments

`vector`	Vector

Examples

```
vector1<-c(1,2,3,4,5,4,3,2,1)
vector2<-c(1,2,3,4,5,5,3,2,1)
vector3<-c(1,2,3,5,5,4,3,2,1)
vector4<-c(1,2,3,4,6,4,3,2,1)
vector5<-c(1,6,3,4,6,4,3,2,1)
vector<-vector1
which(vector==max(vector),arr.ind=TRUE)
which(vector==min(vector),arr.ind=TRUE)
min_max_index(vector1)
min_max_index(vector2)
min_max_index(vector3)
min_max_index(vector4)
min_max_index(vector5)
```

---

model_loadings *Pattern and structure matrix*

Description

Pattern and structure matrix

Usage

```
model_loadings(model, cut = NULL, matrix_type = "pattern", sort = TRUE, ...)
```

Arguments

model	psych EFA model
cut	cut point for loadings
matrix_type	"pattern" "structure" "all"
sort	if TRUE it will sort loadings
...	arguments passed to psych::fa.sort

Note

Check to see if you have multicolinearity values above .8 in the matrix are problematic
Structure matrix represents Loadings after rotation
Pattern matrix represents Loadings before rotation

Examples

```
model<-psych::fa(mtcars,nfactors=2,rotate="oblimin",fm="pa",oblique.scores=TRUE)
model_loadings(model=model,cut=NULL,matrix_type="pattern")
model_loadings(model=model,cut=0.4,matrix_type="structure")
model_loadings(model=model,cut=0.4,matrix_type="all",sort=FALSE)
```

`off_diagonal_index` *index of off diagonal*

Description

index of off diagonal

Usage

```
off_diagonal_index(length)
```

Arguments

`length`	length of diagonal

Examples

```
off_diagonal_index(length=6)
```

`outlier_summary` *Percent of outliers in vector*

Description

Percent of outliers in vector

Usage

```
outlier_summary(vector)
```

Arguments

`vector`	numeric vector

Details

returns dataframe

Examples

```
vector<-generate_missing(rnorm(1000))
df<-generate_missing(mtcars[,1:2])
outlier_summary(vector)
data.frame(sapply(mtcars,outlier_summary))
```

`output_compare_model_logistic` *Compare logistic regression models models*

Description

Compare logistic regression models models

Usage

```
output_compare_model_logistic(model1, model2)
```

Arguments

model1	object glm model
model2	object glm model

Examples

```
modelcategoricalpredictor<-glm(case~education,data=infert,family=binomial)
modelcontinuouspredictor<-glm(case~age,data=infert,family=binomial)
modeltwopredictors<-glm(case~education*age,data=infert,family=binomial)
modelmultiple<-glm(case~education*age*parity,data=infert,family=binomial)
anova(modelcategoricalpredictor,modelcontinuouspredictor)
output_compare_model_logistic(model1=modelcategoricalpredictor,
                              model2=modeltwopredictors)
output_compare_model_logistic(model1=modelcontinuouspredictor,
                              model2=modeltwopredictors)
output_compare_model_logistic(model1=modelcontinuouspredictor,
                              model2=modelcategoricalpredictor)
```

`output_separator` *Output separator*

Description

Heading, main output, and instructions for output for the console environment

Usage

```
output_separator(
  string,
  output = NULL,
  instruction = NULL,
  length = getOption("width")/2
)
```

Arguments

`string`	Title of output
`output`	object to print
`instruction`	Character provided instructions regarding the output
`length`	Numeric Length of separator measured in number of characters

Examples

```
output_separator(string="TEST",output="TEST",instruction="TEST",length=100)
output_separator(string="TEST",instruction="TEST",length=100)
output_separator(string="TEST",output="TEST",length=100)
output_separator(string="TEST")
```

---

padNA *pad NA's to collumns in dataframe*

Description

pad NA's to collumns in dataframe

Usage

```
padNA(df, rowsneeded, first = TRUE)
```

Arguments

`df`	dataframe
`rowsneeded`	Numeric number of rows needed
`first`	Boolean

Author(s)

Ananda Mahto

plot_acf *Plot autocorrelation function of correlation covariance and partial correlation*

Description

uses ggplot

Usage

```
plot_acf(df, lag.max = length(df), base_size = 10, title = "")
```

Arguments

df	ts object
lag.max	maximum lags to include
base_size	base font size
title	plot title

Details

returns plot

Examples

```
ts_data<-ts(UKDriverDeaths,start=1969,end=1984,frequency=12)
plot_acf(df=ts_data,base_size=20)
```

plot_boxplot *Boxplot*

Description

Boxplot

Usage

```
plot_boxplot(df, title = "", base_size = 10)
```

Arguments

df	dataframe or vector with continous or ordinal data
title	Plot title
base_size	numeric base font size

Details

uses ggplot

Examples

```
vector<-generate_missing(rnorm(1000))
df<-generate_missing(mtcars[,1:2])
plot_boxplot(df=vector)
plot_boxplot(df=generate_missing(vector))
plot_boxplot(df=df)
```

`plot_cfa` *Plot cfa model*

Description

Plot cfa model

Usage

```
plot_cfa(model, ...)
```

Arguments

`model`	lavaan object
`...`	arguments passed to semPlot::semPaths

Examples

```
model='LATENT1=~X1+X2+X3
       LATENT2=~X4+X5+X6'
df<-lavaan::simulateData(model=model,model.type="cfa",
                              return.type="data.frame",sample.nobs=100)
df<-generate_missing(df)
fit<-lavaan::cfa(model,data=df,missing="ML")
plot_cfa(fit)
model='LATENT1=~X1+X2+X3+X4+X5+X6
       LATENT2=~X1+X2+X3+X4+X5+X6'
```

`plot_confusion` *Plot confusion matrix*

Description

This function creates a confusion matrix plot with observed and predicted outcomes,including row and column percentages,and various accuracy metrics.

Usage

```
plot_confusion(observed, predicted, base_size = 10, title = "")
```

Arguments

`observed`	Vector of observed outcomes. This can be numeric or factor values representing the true class labels.
`predicted`	Vector of predicted outcomes. This should have the same length as the observed vector and represent the predicted class labels.
`base_size`	Integer value representing the base font size for the plot. Defaults to 10.
`title`	String representing the title of the plot. Defaults to an empty string.

Details

This function generates a confusion matrix plot using ggplot2. It provides a visual representation of the confusion matrix with observed outcomes on the x-axis and predicted outcomes on the y-axis. The cells of the matrix are filled with the count of observations and annotated with the corresponding values.

The plot also includes various accuracy metrics in the caption,such as: -Overall Accuracy: Proportion of correctly classified observations (diagonal elements). -Off-diagonal Accuracy: Proportion of misclassified observations (off-diagonal elements). -Cohen's Kappa (Unweighted,Linear,and Squared): Measures the agreement between observed and predicted outcomes.

Examples

```
# Example with numeric class labels
plot_confusion(observed=c(1,2,3,1,2,3),predicted=c(1,2,3,1,2,3))

# Example with factor class labels
observed<-c(rep("male",10),rep("female",10),"male","male")
predicted<-c(rep("male",10),rep("female",10),"female","female")
plot_confusion(observed=observed,predicted=predicted)
```

plot_corrplot *Correlation matrix plots*

Description

Correlation matrix plots

Usage

```
plot_corrplot(mydata, title = "", base_size = 10, fill_limits = c(-1, 0, 1))
```

Arguments

mydata	correlation matrix
title	plot title
base_size	base font size
fill_limits	lower and upper limit for fill

Examples

```
plot_corrplot(stats::cor(mtcars),title="Correlation")
plot_corrplot(stats::cor(mtcars),base_size=20)
```

plot_crosstable *Plot crosstables*

Description

Plot crosstables

Usage

```
plot_crosstable(
  df,
  factor_index,
  combinations = NULL,
  shape = 16,
  angle = 0,
  base_size = 10,
  title = ""
)
```

Arguments

df	dataframe
factor_index	index of factors
combinations	index of comparisons
shape	shape of points
angle	angle of xaxis labels
base_size	base font size
title	plot title

Examples

```
combinations<-data.frame(index1=c("vs","am","gear"),index2=c("cyl","cyl","cyl"))
plot_crosstable(df=mtcars,factor_index=8:9)
plot_crosstable(df=mtcars,combinations=combinations)
```

---

plot_histogram *Histograms with density function*

Description

Histograms with density function

Usage

```
plot_histogram(
  df,
  bins = 30,
  title = "",
  base_size = 10,
  xlims = NULL,
  fill = "gray25",
  color = "gray50",
  ylab = "Count"
)
```

Arguments

df	dataframe or vector with continous or ordinal data
bins	number of bars to display
title	plot title
base_size	numeric base font size
xlims	x axis limits
fill	color of bar
color	color of bar outline
ylab	y label

Details

uses ggplot

Examples

```
vector<-generate_missing(rnorm(1000))
df<-generate_missing(mtcars[,1:2])
plot_histogram(df=vector)
plot_histogram(df=df,xlims=c(0,50))
plot_histogram(df=df)
plot_multiplot(plotlist=plot_histogram(df=mtcars),cols=4)
```

---

plot_icc_thurstonian *Plot thurstonian icc*

Description

Plot icc curves for binary thurstonian coded items for a single dimension using the compute_icc_thurstonian function

Usage

```
plot_icc_thurstonian(mydata, title = "Item Characteristic Curve")
```

Arguments

mydata	dataframe from compute_icc_thurstonian function
title	plot title

Examples

```
gamma<-c(0.556,-1.253,-1.729,0.618,0.937,0.295,-0.672,-1.127,-0.446,0.632,1.147,0.498)
psi<-c(2.172,1.883,2.055,1.869,2.231,2.100,1.762,1.803,1.565,1.892,1.794,1.686)
lambda<-c(1.082,1.082,-1.297,-1.297,0.802,0.802,1.083,1.083)
gamma<-gamma[response_dimension(c(1:12),3,c(1,2))]
psi<-psi[response_dimension(c(1:12),3,c(1,2))]
eta<-seq(-6,6,by=1)
result<-compute_icc_thurstonian(eta=eta,gamma=gamma,lambda=lambda,psi=psi,plot=TRUE)
plot_icc_thurstonian(result$icc)
```

`plot_interaction` *Plot two way interaction graphs*

Description

Plot two way interaction graphs

Usage

```
plot_interaction(
  df,
  dv,
  iv,
  base_size = 20,
  type = "se",
  order_factor = TRUE,
  title = "",
  note = ""
)
```

Arguments

`df`	dataframe
`dv`	index of continous variables
`iv`	index of factors
`base_size`	base font size
`type`	error bar type to display (1) "se" for standard error (2) "ci" for confidence interval (3) "sd" for standard deviation (4) "" for no error bar
`order_factor`	if TRUE it will sort the categorical axis by the continous variable value
`title`	plot title
`note`	footnote

Examples

```
nrows=1000
df<-data.frame(generate_factor(vector=LETTERS[1:5],nrows=nrows,ncols=10,type="random"),
              generate_data(nrows=nrows,ncols=5,type="normal"))
#result<-plot_interaction(df=df,dv=11:15,iv=1:10)
plot_interaction(df=mtcars,dv=2:3,iv=8:9,base_size=20,title="",type="se")
plot_interaction(df=mtcars,dv=2,iv=8:9,base_size=20,title="",type="se")
plot_interaction(df=mtcars,dv=2:3,iv=8:9,base_size=20,title="",type="ci")
plot_interaction(df=mtcars,dv=2:3,iv=9:10,base_size=20,title="",type="ci")
plot_interaction(df=mtcars,dv=2:3,iv=9:10,base_size=20,title="",type="sd")
plot_interaction(df=mtcars,dv=2,iv=9:10,base_size=20,
                title="",type="",order_factor=FALSE)
```

`plot_irt_onefactor` *Return data for irt plots*

Description

Return data for irt plots

Usage

```
plot_irt_onefactor(model, theta = seq(-6, 6, 0.1), title = "", base_size = 10)
```

Arguments

`model`	object mirt
`theta`	theta
`title`	plot title
`base_size`	base size

Examples

```
cormatrix<-psych::sim.rasch(nvar=5,n=50000,low=-4,high=4,d=NULL,a=1,mu=0,sd=1)$items
model<-mirt::mirt(cormatrix,1,empiricalhist=TRUE,calcNull=TRUE)
plot_irt_onefactor(model=model,base_size=10,title="Normal Test")
cormatrix<-psych::sim.rasch(nvar=5,n=50000,low=-6,high=-4,d=NULL,a=1,mu=0,sd=1)$items
model<-mirt::mirt(cormatrix,1,empiricalhist=TRUE,calcNull=TRUE)
plot_irt_onefactor(model=model,base_size=10,title="Easy Items")
cormatrix<-psych::sim.rasch(nvar=5,n=50000,low=4,high=6,d=NULL,a=1,mu=0,sd=1)$items
model<-mirt::mirt(cormatrix,1,empiricalhist=TRUE,calcNull=TRUE)
plot_irt_onefactor(model=model,base_size=10,title="Difficult Items")
cormatrix<-psych::sim.rasch(nvar=5,n=50000,low=-4,high=-4,d=NULL,a=0.01,mu=0,sd=1)$items
model<-mirt::mirt(cormatrix,1,empiricalhist=TRUE,calcNull=TRUE)
plot_irt_onefactor(model=model,base_size=10,title="Low Discrimination")
cormatrix<-psych::sim.poly(nvar=5,n=50000,low=-4,high=4,a=1,c=0,z=1,d=NULL,
                           mu=0,sd=1,cat=5,mod="logistic",theta=NULL)$items
model<-mirt::mirt(cormatrix,1,itemtype="graded")
plot_irt_onefactor(model=model,base_size=10,title="graded response")
```

`plot_loadings` *Plot loadings*

Description

Plot loadings

Usage

```
plot_loadings(
  model,
  matrix_type = NULL,
  title = "",
  base_size = 10,
  color = c("#5E912C", "white", "#5F2C91"),
  sort = TRUE
)
```

Arguments

`model`	psych EFA model
`matrix_type`	"pattern" "structure"
`title`	plot title
`base_size`	base font size
`color`	color ranges for heatmap
`sort`	TRUE or FALSE sort loadings

Examples

```
model<-psych::fa(mtcars,nfactors=2,rotate="oblimin",fm="pa",oblique.scores=TRUE)
plot_loadings(model=model,matrix_type="structure")
plot_loadings(model=model,matrix_type="pattern")
cm<-matrix(c(1,.8,.8,.1,.1,.1,
             .8,1,.8,.1,.1,.1,
             .8,.8,1,.1,.1,.1,
             .1,.1,.1,1,.8,.8,
             .1,.1,.1,.8,1,.8,
             .1,.1,.1,.8,.8,1),
             ncol=6,nrow=6)
df1<-generate_correlation_matrix(cm,nrows=10000)
model1<-psych::fa(df1,nfactors=2,rotate="oblimin",fm="pa",oblique.scores=TRUE)
plot_loadings(model=model1,matrix_type="pattern",base_size=30)
cm<-matrix(c(1,.1,.1,.1,.1,.1,
             .1,1,.1,.1,.1,.1,
             .1,.1,1,.1,.1,.1,
             .1,.1,.1,1,.8,.8,
             .1,.1,.1,.8,1,.8,
             .1,.1,.1,.8,.8,1),
             ncol=6,nrow=6)
df1<-generate_correlation_matrix(cm,nrows=10000)
model2<-psych::fa(df1,nfactors=2,rotate="oblimin",fm="pa",oblique.scores=TRUE)
plot_loadings(model=model2,matrix_type="pattern",base_size=30)
cm<-matrix(c(1,.01,.01,.01,.01,.01,
             .01,1,.01,.01,.01,.01,
             .01,.01,1,.01,.01,.01,
             .01,.01,.01,1,.01,.01,
             .01,.01,.01,.01,1,.01,
```

```
              .01,.01,.01,.01,.01,1),
              ncol=6,nrow=6)
df1<-generate_correlation_matrix(cm,nrows=10000)
model3<-psych::fa(df1,nfactors=2,rotate="oblimin",fm="pa",oblique.scores=TRUE)
plot_loadings(model=model3,matrix_type="pattern",base_size=10)
```

`plot_logistic_model` *Logistic model plot*

Description

Logistic model plot

Usage

```
plot_logistic_model(df, outcome = "outcome", title = "", base_size = 10)
```

Arguments

`df`	dataframe with predictor and outcome outcome should be last
`outcome`	name of outcome variable
`title`	Character plot title
`base_size`	base font size

Examples

```
df<-data.frame(outcome=c(rep(1,10),rep(0,10)),
               pd1=c(rep(1,11),rep(0,9)),
               pd2=c(rep(1,9),rep(0,11)),
               pc1=c(rnorm(10,mean=5),rnorm(10,mean=10)),
               pc2=c(rnorm(10,mean=5),rnorm(10,mean=20)))
plot_logistic_model(df=df,base_size=15)
```

`plot_mosaic` *Plot mosaic plots*

Description

Plot mosaic plots

Usage

```
plot_mosaic(df, factor_index, base_size = 10, title = "")
```

Arguments

df	dataframe
factor_index	index of factors
base_size	base font size
title	plot title

Examples

```
plot_mosaic(df=mtcars,factor_index=8:9)
plot_mosaic(df=mtcars,factor_index=9:10)
```

---

plot_mtmm *Plot multitrait multimethod matrix*

Description

Plot multitrait multimethod matrix

Usage

```
plot_mtmm(df, key, method, subject, title = "")
```

Arguments

df	dataframe
key	List index of trait names and items constituring a trait
method	name of dataframe collumn spesifying the method used for the row observed
subject	name of dataframe collumn spesifying subject id
title	plot title

Examples

```
population_model<-'t1=~x1+.9*x2+.9*x3
                   t2=~x4+.9*x5+.9*x6
                   t3=~x7+.9*x8+.9*x9'
model_data<-lavaan::simulateData(population_model,sample.nobs=1000)
model_data<-model_data[sample(1:1000,1000,TRUE),]
model_data<-rbind(model_data,model_data,model_data)
model_data$method<-c(rep("m1",1000),rep("m2",1000),rep("m3",1000))
model_data$id<-rep(1:1000,3)
key<-list(t1=paste0("x",1:3),t2=paste0("x",4:6),t3=paste0("x",7:9))
plot_mtmm(df=model_data,key=key,method="method",subject="id")
```

plot_multiplot — *Multiple ggplot plots in one graph*

Description

Multiple ggplot plots in one graph

Usage

```
plot_multiplot(..., plotlist = NULL, cols = 2, layout = NULL)
```

Arguments

`...`	plot objects
`plotlist`	a list of plots
`cols`	number of columns in layout
`layout`	a matrix specifying the layout. If present,'cols' is ignored

Examples

```
p1<-ggplot(ChickWeight,aes(x=Time,y=weight,colour=Diet,group=Chick))+
          geom_line()+
          ggtitle("Growth curve for individual chicks")+
          theme_bw()
p2<-ggplot(ChickWeight,aes(x=Time,y=weight,colour=Diet))+
          geom_point(alpha=.3)+
          geom_smooth(alpha=.2,size=1,method="loess",formula="y~x")+
          ggtitle("Fitted growth curve per diet")+
          theme_bw()
p3<-ggplot(subset(ChickWeight,Time==21),aes(x=weight,colour=Diet))+
          geom_density()+
          ggtitle("Final weight, by diet")+theme_bw()
p4<-ggplot(subset(ChickWeight,Time==21),aes(x=weight,fill=Diet))+
          geom_histogram(colour="black",binwidth=50)+facet_grid(Diet~.)+
          ggtitle("Final weight, by diet")+theme_bw()
cars_plot<-plot_histogram(mtcars)
plot_multiplot(p1,p2,p3,p4,cols=2)
plot_multiplot(plotlist=plot_histogram(mtcars[,1:4]),cols=2)
plot_multiplot(plotlist=plot_histogram(mtcars),layout=matrix(1:4,ncol=2,byrow=TRUE))
plot_multiplot(plotlist=plot_scatterplot(mtcars[,1:4]),cols=2)
plot_multiplot(plotlist=cars_plot,layout=matrix(1:4,ncol=2,byrow=TRUE))
plot_multiplot(plotlist=cars_plot,cols=3)
```

`plot_normality_diagnostics` *Normality plots*

Description

plot histogram density boxplot qq plot

Usage

```
plot_normality_diagnostics(
  df,
  breaks = NULL,
  title = "",
  file = NULL,
  w = 10,
  h = 10
)
```

Arguments

`df`	dataframe or vector with continous or ordinal data
`breaks`	number of bars to display
`title`	plot title
`file`	output filename
`w`	width of pdf file
`h`	height of pdf file

Details

uses plot base

Examples

```
vector<-generate_missing(rnorm(1000))
df<-generate_missing(mtcars[,1:2])
plot_normality_diagnostics(df=vector,title="",file="rnorm",breaks=30)
plot_normality_diagnostics(df=vector,title="")
plot_normality_diagnostics(df=df,title="mtcars")
plot_normality_diagnostics(df=df,title="mtcars",file="rnorm")
```

`plot_oneway` *Plot means with standard error for every level in a dataframe*

Description

Plot means with standard error for every level in a dataframe

Usage

```
plot_oneway(
  df,
  dv,
  iv,
  base_size = 20,
  type = "se",
  order_factor = TRUE,
  title = "",
  note = "",
  width = 60
)
```

Arguments

df	dataframe
dv	index of continous variables
iv	index of factors
base_size	base font size
type	error bar type to display (1) "se" for standard error (2) "ci" for confidence interval (3) "sd" for standard deviation (4) "" for no error bar
order_factor	if TRUE it will sort the categorical axis by the continous variable value
title	plot title
note	footnote
width	wrap width for x title

Examples

```
nrows=1000
df<-data.frame(generate_factor(vector=LETTERS[1:5],nrows=nrows,ncols=10,type="random"),
               generate_data(nrows=nrows,ncols=5,type="normal"))
result<-plot_oneway(df=df,dv=11:15,iv=1:10)
plot_oneway(df=mtcars,dv=2,iv=9)
plot_oneway(df=mtcars,dv=2,iv=9:10)
plot_oneway(df=mtcars,dv=2:3,iv=10)
plot_oneway(df=mtcars,dv=2:3,iv=9:10)
plot_oneway(df=mtcars,dv=2:3,iv=9:10,type="se")
```

```
plot_oneway(df=mtcars,dv=2:3,iv=9:10,type="ci")
plot_oneway(df=mtcars,dv=2:3,iv=9:10,type="sd")
plot_oneway(df=mtcars,dv=2:3,iv=9:10,type="",order_factor=FALSE)
plot_oneway(df=mtcars,dv=2:3,iv=9:10,type="",order_factor=TRUE)
```

`plot_oneway_diagnostics` *Plot one way diagnostics*

Description

Plot one way diagnostics

Usage

```
plot_oneway_diagnostics(df, dv, iv, base_size = 10)
```

Arguments

`df`	dataframe
`dv`	index of continous variables
`iv`	index of factors
`base_size`	base font size

Note

Residuals vs Fitted should be equally spread horizontally otherwize the assumption of equality of variances is violated
Normal QQ should show values in the diagonal otherwise the assumption of normality is violated

Examples

```
nrows=1000
df<-data.frame(generate_factor(vector=LETTERS[1:5],nrows=nrows,ncols=10,type="random"),
               generate_data(nrows=nrows,ncols=5,type="normal"))
result<-plot_oneway_diagnostics(df=df,dv=11:15,iv=1:10)
plot_oneway_diagnostics(df=mtcars,dv=1:2,iv=9:10)
```

`plot_outlier` *Outlier graph using mean median and boxplot algorythms*

Description

Outlier graph using mean median and boxplot algorythms

Usage

```
plot_outlier(df, method = "mean", title = "", base_size = 10)
```

Arguments

`df`	dataframe or vector with continous or ordinal data
`method`	"mean" "median" "boxplot"
`title`	plot title
`base_size`	base font size

Author(s)

unknown

Examples

```
vector<-generate_missing(rnorm(1000))
df<-generate_missing(mtcars[,1:2])
plot_outlier(df=vector,method="mean",title="random vector")
plot_outlier(df=vector,method="median")
plot_outlier(df=vector,method="boxplot")
plot_outlier(df=df,method="mean",title="random vector")
plot_outlier(df=df,method="median")
plot_outlier(df=df,method="boxplot")
plot_multiplot(plotlist=plot_outlier(df=mtcars[,2:5],method="mean"),cols=2)
```

`plot_qq` *qq plots*

Description

qq plots

Usage

```
plot_qq(df, title = "", base_size = 10)
```

Arguments

`df`	dataframe or vector with continous or ordinal data
`title`	plot title
`base_size`	numeric base font size

Details

uses ggplot

Examples

```
vector<-generate_missing(rnorm(1000))
df<-generate_missing(mtcars[,1:2])
plot_qq(df=vector)
plot_qq(df=df)
plot_multiplot(plotlist=plot_qq(df=mtcars),cols=4)
```

`plot_response_frequencies` *Plot response frequencies*

Description

Plot response frequencies

Usage

```
plot_response_frequencies(
  df,
  factor_index,
  base_size = 10,
  title = "",
  width = 100,
  reorder = FALSE
)
```

Arguments

`df`	dataframe
`factor_index`	index of factors
`base_size`	base font size
`title`	plot title
`width`	Numeric, wrap width for x-axis title
`reorder`	Logical, whether to reorder factors based on frequency

Examples

```
plot_response_frequencies(df=mtcars,factor_index=1:10)
```

---

plot_roc *Plot Receiver Operating Characteristic (ROC) curve*

Description

Generates a ROC curve from observed outcomes and predicted probabilities.

Usage

```
plot_roc(observed, predicted, base_size = 10, title = "")
```

Arguments

observed	Vector of observed outcomes. These are the true class labels.
predicted	Vector of predicted outcome probabilities. These are the predicted probabilities for the positive class.
base_size	Integer value representing the base font size for the plot. Defaults to 10.
title	String representing the title of the plot. Defaults to an empty string.

Details

This function generates a ROC curve to evaluate the performance of a binary classification model. The ROC curve is a plot of the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings.

The function performs the following steps: 1. Computes the ROC curve and its confidence interval using 'pROC::roc'. 2. Generates ROC plots for both reversed and non-reversed order of class levels. 3. Creates a list of ROC plots,each with an AUC value,control level,and direction.

The output is a list of ggplot objects representing the ROC curves for different class level orders.

Examples

```
# Example with random observed and predicted values
observed<-round(abs(rnorm(100,m=0,sd=0.5)))
predicted<-abs(rnorm(100,m=0,sd=0.5))
plot_roc(observed=observed,predicted=predicted)

# Example with generated correlation matrix
df1<-data.frame(matrix(0.999,ncol=2,nrow=2))
correlation_matrix<-as.matrix(df1)
diag(correlation_matrix)<-1
df1<-generate_correlation_matrix(correlation_matrix,nrows=1000)
df1$X1<-ifelse(abs(df1$X1) < 1,0,1)
df1$X2<-abs(df1$X2)
df1$X2<-(df1$X2-min(df1$X2))/(max(df1$X2)-min(df1$X2))
```

```
plot_roc(observed=round(abs(df1$X1),0),predicted=abs(df1$X2))
```

---

plot_scatterplot *Plot plot_scatterplot*

Description

Plot plot_scatterplot

Usage

```
plot_scatterplot(
  df,
  method = lm,
  formula = y ~ x,
  base_size = 10,
  coord_equal = FALSE,
  all_orders = FALSE,
  title = "",
  combinations = NULL,
  string_aes = TRUE
)
```

Arguments

df	dataframe if dataframe consists of 2 collumns the second collumn is the outcome and the first collumn is the predictor
method	smoothing method, "auto", "lm", "glm", "gam", "loess" or a function, e.g. MASS::rlm or mgcv::gam, stats::lm, or stats::loess
formula	formula used in smoothing function for geom_smooth
base_size	base font size
coord_equal	if TRUE axes maintain equal scale
all_orders	if TRUE the order of combination is considered
title	Plot title
combinations	dataframe if not NULL user can provide a dataframe for variable combinations for x and y axis . First column represents x and second column represents y
string_aes	if TRUE string_aes function is used for names

Examples

```
result<-plot_scatterplot(df=mtcars,title="",coord_equal=TRUE,base_size=10)
plot_multiplot(plotlist=result[1:12],cols=4)
plot_scatterplot(df=mtcars[,1:2],base_size=10,coord_equal=TRUE,all_orders=FALSE)
plot_scatterplot(df=mtcars[,1:2],base_size=10,coord_equal=FALSE,all_orders=FALSE)
plot_scatterplot(df=mtcars,base_size=10,coord_equal=TRUE,all_orders=FALSE,
                 combinations=data.frame(x=c("mpg","mpg","mpg"),
                                         y=c("cyl","hp","mpg")))
plot_scatterplot(df=mtcars,base_size=10,coord_equal=TRUE,all_orders=TRUE,
                 combinations=data.frame(x=c("mpg"),y=c("cyl")))
x<-rnorm(1000)
y<-x+rnorm(x,sd=.1)
plot_scatterplot(df=data.frame(x,y),title="Random Simulation",coord_equal=TRUE)
df<-data.frame(matrix(-.999,ncol=2,nrow=2))
correlation_martix<-as.matrix(df)
diag(correlation_martix)<-1
df<-generate_correlation_matrix(correlation_martix,nrows=1000)
plot_scatterplot(df,title="Simulation of -.999 Correlation",coord_equal=TRUE,base_size=20)
```

`plot_scree` *Scree plot displaying the Kaiser and Jolife criteria for factor extraction*

Description

Scree plot displaying the Kaiser and Jolife criteria for factor extraction

Usage

```
plot_scree(df, base_size = 15, title = "", color = c("#5F2C91", "#5E912C"))
```

Arguments

`df`	dataframe
`base_size`	base font size
`title`	plot title
`color`	color of line and point outline

Examples

```
plot_scree(df=mtcars,title="",base_size=15)
```

plot_separability *Plot separability*

Description

This function creates a separability plot showing the density distribution of predicted probabilities for different observed categories.

Usage

```
plot_separability(observed, predicted, base_size = 10, title = "")
```

Arguments

`observed`	Vector of observed outcomes. This can be numeric or factor values representing the true class labels.
`predicted`	Vector of predicted outcome probabilities. This should have the same length as the observed vector and represent the predicted probabilities.
`base_size`	Integer value representing the base font size for the plot. Defaults to 10.
`title`	String representing the title of the plot. Defaults to an empty string.

Details

This function generates a separability plot using ggplot2. It shows the density distribution of predicted probabilities for different observed categories. The plot helps to visualize how well the predicted probabilities separate the different observed categories.

The plot includes the following components: -Density curves for each observed category,representing the distribution of predicted probabilities. -A legend indicating the observed categories. -The total number of observations is included in the plot caption.

Examples

```
# Example with numeric class labels
df1<-data.frame(matrix(.999,ncol=2,nrow=2))
correlation_matrix<-as.matrix(df1)
diag(correlation_matrix)<-1
df1<-generate_correlation_matrix(correlation_matrix,nrows=1000)
df1$X1<-ifelse(abs(df1$X1) < 1,0,1)
df1$X2<-abs(df1$X2)
df1$X2<-(df1$X2-min(df1$X2))/(max(df1$X2)-min(df1$X2))
plot_separability(observed=round(abs(df1$X1),0),predicted=abs(df1$X2))
```

plot_trees_xgboost *Plot trees for xgboost::xgb.train*

Description

Plot trees for xgboost::xgb.train

Usage

```
plot_trees_xgboost(model, train, file = "xgboost")
```

Arguments

`model`	object from xgboost::xgb.train
`train`	Train dataset
`file`	output filename

Examples

```
infert_formula<-formula(case~education+spontaneous+induced)
boston_formula<-formula(medv~crim+zn+indus+chas+nox+rm+age+dis+rad+tax+ptratio+black+lstat)
train_test_classification<-k_fold(df=infert,model_formula=infert_formula)
train_test_regression<-k_fold(df=MASS::Boston,model_formula=boston_formula)
xgb_classification<-xgboost::xgb.train(
                    data=train_test_classification$xgb$f1$train,
                    watchlist=train_test_classification$xgb$f1$watchlist,
                    eta=.1,
                    nthread=8,
                    nround=20,
                    objective="binary:logistic")
xgb_regression<-xgboost::xgb.train(
                data=train_test_regression$xgb$f1$train,
                watchlist=train_test_regression$xgb$f1$watchlist,
                eta=.3,
                nthread=8,
                nround=20)
# xgboost::xgb.plot.multi.trees(model=xgb_classification,features_keep=2)
# plot_trees_xgboost(model=xgb_classification,
#                    train=train_test_classification$xgb$f1,
#                    file="Classification")
# plot_trees_xgboost(model=xgb_regression,
#                    train=train_test_regression$xbg$f1,
#                    file="Regression")
```

plot_ts *Plot timeseries*

Description

Plot timeseries

Usage

```
plot_ts(df, base_size = 10, ylab = "Count", title = "")
```

Arguments

df	ts object
base_size	base font size
ylab	y label
title	plot title

Details

returns plot

Examples

```
ts_data<-ts(UKDriverDeaths,start=1969,end=1984,frequency=12)
result<-plot_ts(ts_data,title="UK driver deaths")
for(i in 1969:1984)
  result<-result+geom_vline(xintercept=i,color="blue",size=1,alpha=.5)
result
autoplot(stl(ts_data,s.window='periodic'))+
  theme_bw(base_size=10)+
  labs(title="UK driver deaths")
forecast::gglagplot(data.frame(ts_data),do.lines=FALSE,lags=100)+
  theme_bw(base_size=10)+labs(title="UK driver deaths",y="count")
```

proper *Capitalize first character and lowercase the rest*

Description

Capitalize first character and lowercase the rest

Usage

```
proper(x)
```

Arguments

`x`	Character

Examples

```
x<-generate_string(nchar=10,vector=LETTERS,vector_length=10)
proper(x)
```

`proportion_accurate` *Proportion overall accuracy of a confusion matrix*

Description

Calculates the overall accuracy and Cohen's kappa statistics of a confusion matrix.

Usage

```
proportion_accurate(observed, predicted)
```

Arguments

`observed`	Vector of observed variables. These are the true class labels.
`predicted`	Vector of predicted variables. These are the predicted class labels.

Details

This function evaluates the performance of a confusion matrix by calculating the overall accuracy and Cohen's kappa statistics.

The function performs the following steps: 1. Computes the confusion matrix from the observed and predicted values. 2. Calculates the diagonal proportion (overall accuracy) and the off-diagonal proportion. 3. Computes Cohen's kappa statistics (unweighted,linear,and squared weights).

The output is a data.frame containing the following metrics: -‘cm_diagonal‘: Proportion of correct classifications (diagonal elements). -‘cm_off_diagonal‘: Proportion of misclassified observations (off-diagonal elements). -‘kappa_unweighted‘: Cohen's kappa statistic with no weights. -‘kappa_linear‘: Cohen's kappa statistic with linear weights. -‘kappa_squared‘: Cohen's kappa statistic with squared weights.

Examples

```
# Example with numeric observed and predicted values
proportion_accurate(observed=c(1,2,3,4,5,10),predicted=c(1,2,3,4,5,11))
```

questions_by_keys *Convert key to index list*

Description

Convert key to index list

Usage

```
questions_by_keys(key)
```

Arguments

key	a vector indicating the dimension of each question. The order of the elements in the key represents the order of the questions, the numeric values represent the dimension the question belongs to

Examples

```
key<-c(1,2,3,4,5,1,2,3,4,5)
questions_by_keys(key)
```

questions_dimensions_dataframe *Question dimension table*

Description

Return a dataframe with the order of the questions, their respective dimensions, and the description of the dimensions

Usage

```
questions_dimensions_dataframe(
  key,
  dimensions,
  elaborate_dimensions,
  questions
)
```

Arguments

key	a vector indicating the dimension of each question. The order of the elements in the key represents the order of the questions, the numeric values represent the dimension the question belongs to
dimensions	dimension names
elaborate_dimensions	full dimension names
questions	question names

Examples

```
key<-c(1,2,3,4,5,1,2,3,4,5)
dimensions<-paste0("Dimension",1:10)
elaborate_dimensions<-paste0("Elaborated_Dimension",1:10)
questions<-paste0("Question",1:65)
questions_dimensions_dataframe(key,dimensions,elaborate_dimensions,questions)
```

---

rad2deg *Convert radians to degrees*

Description

Convert radians to degrees

Usage

```
rad2deg(radians)
```

Arguments

radians	radians

Examples

```
rad2deg(pi)
```

rank3_to_triplets *Convert thurstonian binary triplets to scale*

Description

Convert thurstonian binary triplets to scale

Usage

```
rank3_to_triplets(mydata)
```

Arguments

mydata	dataframe

Examples

```
set.seed(12345)
mydata<-data.frame(i1=rnorm(10,mean=2,sd=.5),
                   i2=rnorm(10,mean=2,sd=.5),
                   i3=rnorm(10,mean=2,sd=.5),
                   i4=rnorm(10,mean=2,sd=.5),
                   i5=rnorm(10,mean=2,sd=.5),
                   i6=rnorm(10,mean=2,sd=.5))
result<-rank_to_binary(mydata[,1:3])
rank3_to_triplets(result)
```

rank_df_to_binary *Convert scale to thurstonian binary with n items per block and n blocks*

Description

Convert scale to thurstonian binary with n items per block and n blocks

Usage

```
rank_df_to_binary(mydata, items, reverse = TRUE)
```

Arguments

mydata	dataframe
items	number of items in block
reverse	if TRUE assumes that the highest value is first item in rank if FALSE the lowest value is the first item in rank

Examples

```
set.seed(12345)
mydata<-data.frame(i1=rnorm(10,mean=2,sd=.5),
                   i2=rnorm(10,mean=2,sd=.5),
                   i3=rnorm(10,mean=2,sd=.5),
                   i4=rnorm(10,mean=2,sd=.5),
                   i5=rnorm(10,mean=2,sd=.5),
                   i6=rnorm(10,mean=2,sd=.5))
rank_df_to_binary(mydata[,c("i1","i2","i3","i4")],4)
rank_df_to_binary(mydata,3)
```

`rank_to_binary` *Convert scale to thurstonian binary with n items per ranking block*

Description

Convert scale to thurstonian binary with n items per ranking block

Usage

```
rank_to_binary(mydata, items, reverse = TRUE)
```

Arguments

`mydata`	dataframe
`items`	number of items in block
`reverse`	if TRUE assumes that the highest value is first item in rank if FALSE the lowest value is the first item in rank

Examples

```
set.seed(12345)
mydata<-data.frame(i1=round(rnorm(10,mean=2,sd=1),2),
                   i2=round(rnorm(10,mean=2,sd=1),2),
                   i3=round(rnorm(10,mean=2,sd=1),2),
                   i4=round(rnorm(10,mean=2,sd=1),2),
                   i5=round(rnorm(10,mean=2,sd=1),2),
                   i6=round(rnorm(10,mean=2,sd=1),2))
rank_to_binary(mydata[,c("i1","i2","i3")],items=3)
rank_to_binary(mydata[,c("i1","i2","i3")],items=3,reverse=FALSE)
rank_to_binary(mydata,items=3)
```

raw_alpha *Raw alpha*

Description

Raw alpha

Usage

```
raw_alpha(df)
```

Arguments

df	dataframe with one dimension

Examples

```
set.seed(12345)
df<-data.frame(matrix(.5,ncol=6,nrow=6))
correlation_martix<-as.matrix(df)
diag(correlation_martix)<-1
df<-round(generate_correlation_matrix(correlation_martix,nrows=1000),0)+5
psych::alpha(df)
raw_alpha(df=df)
```

rbind_all *rbind dataframes or matrices with different lengths or collumn names*

Description

rbind dataframes or matrices with different lengths or collumn names

Usage

```
rbind_all(df1, df2)
```

Arguments

df1	dataframe or matrix
df2	dataframe or matrix

Examples

```
df1<-generate_correlation_matrix(n=10)
df2<-generate_correlation_matrix(n=10)
names(df2)[4]<-"X11"
rbind_all(df1=df1,df2=df2)
row.names(df1)<-21:30
rbind_all(df1=df1,df2=df2)
```

recode_scale_dummy   *Scale and dummy code*

Description

Scales numeric variables between 0 and 1 and creates dummy coding for character and factor variables.

Usage

```
recode_scale_dummy(df, categories = 10)
```

Arguments

`df`	Dataframe containing the dataset to be scaled and dummy coded.
`categories`	Numeric value representing the number of unique values a vector must have to perform dummy coding. Defaults to 10.

Details

This function processes a dataframe by scaling numeric variables and creating dummy codes for character and factor variables. The numeric variables are scaled between 0 and 1,while the character and factor variables are converted to dummy variables if they have fewer unique values than the specified 'categories' parameter.

The function performs the following steps: 1. Identifies numeric variables in the dataframe and scales them. 2. Identifies character and factor variables and creates dummy variables if they meet the criteria. 3. Combines the scaled numeric variables and dummy variables into a single dataframe.

The output is a dataframe with scaled numeric variables and dummy-coded character/factor variables.

Examples

```
# Example with the 'infert' dataset
recode_scale_dummy(infert)

# Example with a custom dataframe
df<-data.frame(numeric_var=c(1,2,3,4,5),
               factor_var=factor(c('A','B','A','B','C')))
recode_scale_dummy(df)
```

remove_nc — *Replace remove non computable values*

Description

Replace remove non computable values

Usage

```
remove_nc(
  df,
  value = NA,
  remove_rows = FALSE,
  aggressive = FALSE,
  remove_cols = FALSE,
  remove_zero_variance = FALSE
)
```

Arguments

`df`	dataframe
`value`	replacement
`remove_rows`	if TRUE it will remove rows with non computable values
`aggressive`	if TRUE it will remove entire row if a single non computable value exists if FALSE it will remove row if all values are non computable
`remove_cols`	if TRUE it will remove collumns with non computable values
`remove_zero_variance`	if TRUE it will remove collumns with no variance

Details

Non computable values are NA, NAN, inf and empty cells.

Note

This function internally replaces non computable values with the value choosen the default value is NA. Then it removes rows and collumns with NA values or zero variance

Examples

```
df<-mtcars
df[1,]<-as.numeric(NaN)
df[2,]<-as.numeric(Inf)
df[3,]<-as.numeric(-Inf)
df[4,]<-as.numeric(NA)
df[5,]<-""
remove_nc(df=df,value=NA)
```

```
cdf(remove_nc(df=df,value=NA))
df<-generate_missing(mtcars,missing=5)
remove_nc(df,remove_rows=TRUE,aggressive=FALSE)
remove_nc(df,remove_rows=TRUE,aggressive=TRUE)
df<-generate_missing(generate_correlation_matrix(nrows=5),missing=2)
df$X2<-NA
df$X3<-1
remove_nc(df,remove_cols=TRUE,remove_zero_variance=FALSE)
remove_nc(df,remove_cols=TRUE,remove_zero_variance=TRUE)
```

`remove_outliers` *Remove outliers*

Description

Remove outliers

Usage

```
remove_outliers(vector, probs = c(0.25, 0.75), na.rm = TRUE, ...)
```

Arguments

`vector`	numeric
`probs`	numeric vector with lowest and highest quantiles
`na.rm`	if TRUE removes NA values
`...`	arguments passed to quantile

Examples

```
vector<-generate_missing(rnorm(1000))
df<-generate_missing(mtcars[,1:2])
remove_outliers(vector)
data.frame(sapply(df,remove_outliers))
```

`remove_user_packages` *Remove all user packages*

Description

Remove all user packages

Usage

```
remove_user_packages()
```

replace_na_with_previous *Replace NA with the previous element in a vector*

Description

Replace NA with the previous element in a vector

Usage

```
replace_na_with_previous(vector)
```

Arguments

vector	Vector

Examples

```
df1<-generate_missing(rnorm(10),missing=5)
df2<-generate_missing(rnorm(10),missing=5)
df3<-generate_missing(rnorm(10),missing=5)
df4<-generate_missing(rnorm(10),missing=5)
df5<-generate_missing(rnorm(10),missing=5)
df<-data.frame(df1,df2,df3,df4,df5)
row.names(df)<-paste0("A",row.names(df))
replace_na_with_previous(df1)
df[]<-lapply(df,replace_na_with_previous)
```

report_alpha *Estimate alpha for several dimensions and export results to xlsx*

Description

Uses an arbitrary input

Usage

```
report_alpha(
  df,
  key = NULL,
  questions = NULL,
  reverse = NULL,
  mini = NULL,
  maxi = NULL,
  file = NULL,
  ...
)
```

Arguments

`df`	dataframe
`key`	index of trait names and items constituting a trait
`questions`	trait names and items constituting a trait
`reverse`	index of trait names and index for reversal
`mini`	minimum rating in scale if NULL reversal will be performed using the empirical minimum
`maxi`	maximum rating in scale if NULL reversal will be performed using the empirical maximum
`file`	output filename
`...`	arguments passed to psych::alpha

Examples

```
set.seed(12345)
df<-data.frame(matrix(.5,ncol=6,nrow=6))
correlation_martix<-as.matrix(df)
diag(correlation_martix)<-1
df<-round(generate_correlation_matrix(correlation_martix,nrows=1000),0)+5
key<-list(f1=c("X1","X2","X3"),
          f2=c("X4","X5","X6"))
reverse<-list(f1=c(1,1,1),
              f2=c(1,1,1))
report_alpha(df=df,key=key,cumulative=TRUE,n.iter=1)
report_alpha(df=df,key=key,reverse=reverse,check.keys=FALSE,n.iter=2)
report_alpha(df=df,key=key,check.keys=FALSE,n.iter=2,file="alpha")
```

`report_cfa` *Report*

Description

Report

Usage

```
report_cfa(model, file = NULL, w = 10, h = 10)
```

Arguments

`model`	lavaan object
`file`	output filename
`w`	width of pdf file
`h`	height of pdf file

Examples

```
model='LATENT=~ITEM1+ITEM2+ITEM3+ITEM4+ITEM5'
df<-lavaan::simulateData(model=model,model.type="cfa",
                          return.type="data.frame",sample.nobs=100)
df<-generate_missing(df)
fit<-lavaan::cfa(model,data=df,missing="ML")
report_cfa(fit)
report_cfa(fit,file="cfa")
```

`report_choric_serial` *Report polychoric tetrachoric polyserial biserial correlation*

Description

Report polychoric tetrachoric polyserial biserial correlation

Usage

```
report_choric_serial(
  x,
  y = NULL,
  file = NULL,
  w = 10,
  h = 10,
  type = "tetrachoric",
  ...
)
```

Arguments

`x`	The input may be in one of four forms: a) a data frame or matrix of dichotmous data (e.g., the lsat6 from the bock data set) or discrete numerical (i.e., not too many levels, e.g., the big 5 data set, bfi) for polychoric, or continuous for the case of biserial and polyserial b) a 2 x 2 table of cell counts or cell frequencies (for tetrachoric) or an n x m table of cell counts (for both tetrachoric and polychoric) c) a vector with elements corresponding to the four cell frequencies (for tetrachoric) d) a vector with elements of the two marginal frequencies (row and column) and the comorbidity (for tetrachoric)
`y`	matrix or dataframe of discrete scores. In the case of tetrachoric, these should be dichotomous, for polychoric not too many levels, for biserial they should be discrete (e.g., item responses) with not too many (<10?) categories
`file`	output filename
`w`	width of pdf file

h	height of pdf file
type	"tetrachoric" "polychoric" "polyserial" "biserial"
...	arguments passed to psych::polychoric

Examples

```
report_choric_serial(generate_data(min=0,max=1,type="uniform"),
              type="tetrachoric",file="tetrachoric")
report_choric_serial(generate_data(min=1,max=5,type="uniform"),
              type="polychoric")
report_choric_serial(x=psych::lsat6,y=psych::lsat6,
                     type="polyserial",file="polyserial")
report_choric_serial(x=psych::lsat6,y=psych::lsat6,
                     type="biserial",file="biserial")
```

---

report_correlation *Report correlation matrix*

Description

Report correlation matrix

Usage

```
report_correlation(
  x,
  y = NULL,
  use = "pairwise",
  method = "pearson",
  adjust = "holm",
  alpha = 0.05,
  ci = TRUE,
  file = NULL,
  w = 10,
  h = 10,
  base_size = 20,
  scatterplot = TRUE
)
```

Arguments

x	matrix or dataframe
y	a second matrix or dataframe with the same number of rows as x
use	"pairwise" is the default value and will do pairwise deletion of cases. "complete" will select just complete cases
method	"pearson" "spearman" "kendall"

adjust	"holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none"
alpha	alpha level of confidence intervals
ci	By default, confidence intervals are found. However, this leads to a great slow-down of speed. So, for just the rs, ts and ps, set ci=FALSE
file	output filename
w	width of pdf file
h	height of pdf file
base_size	base font size
scatterplot	if TRUE it will outpu scatterplots

Examples

```
report_correlation(x=generate_missing(mtcars[,1:3],10))
report_correlation(x=generate_missing(mtcars[,1:3],10),
                   file="correlation",scatterplot=TRUE)
report_correlation(x=mtcars[,1:3],file="correlation")
```

`report_dataframe` *Write matrix or dataframe to excel sheet*

Description

Usefull for generic data where conditional formating of a spesific collumn is required

Usage

```
report_dataframe(df, file = NULL, type = "critical_value", ...)
```

Arguments

df	dataframe or matrix
file	output filename of excel file
type	"critical_value" "matrix"
...	arguments passed to excel_critical_value or to excel_matrix

Examples

```
comment<-list(mpg="Miles/(US) gallon",
              cyl="Number of cylinders",
              disp="Displacement (cu.in.)",
              hp="Gross horsepower",
              drat="Rear axle ratio",
              wt="Weight (1000 lbs)",
              qsec="1/4 mile time",
              vs="Engine (0=V-shaped,1=straight)",
              am="Transmission (0=automatic,1=manual)",
```

```
              gear="Number of forward gears",
              carb="Number of carburetors")
report_dataframe(mtcars,sheet="report",file="mtcars",comment=comment,numFmt="#0.00",
                 critical=list(am="<0.05"))
report_dataframe(mtcars,sheet="report",file=NULL,comment=comment,numFmt="#0.00",
                 critical=list(am="<0.05"))
```

---

report_efa *Output EFA model*

Description

Output EFA model

Usage

```
report_efa(
  model,
  df,
  file = NULL,
  w = 10,
  h = 5,
  cut = 0,
  base_size = 10,
  scores = FALSE
)
```

Arguments

model	psych EFA model
df	dataframe
file	output filename
w	width of pdf file
h	height of pdf file
cut	cut point for loadings
base_size	base font size
scores	if TRUE it will output factor scores in excel file

Note

Orthogonal=varimax, Oblique=oblimin

Examples

```
model<-psych::fa(mtcars,nfactors=2,rotate="oblimin",fm="minres",oblique.scores=TRUE)
report_efa(model=model,df=mtcars,file="efa")
model<-psych::fa(mtcars,nfactors=2,rotate="oblimin",fm="uls",oblique.scores=TRUE)
report_efa(model=model,df=mtcars)
model<-psych::fa(mtcars,nfactors=2,rotate="oblimin",fm="ols",oblique.scores=TRUE)
report_efa(model=model,df=mtcars)
model<-psych::fa(mtcars,nfactors=2,rotate="oblimin",fm="wls",oblique.scores=TRUE)
report_efa(model=model,df=mtcars)
model<-psych::fa(mtcars,nfactors=2,rotate="oblimin",fm="gls",oblique.scores=TRUE)
report_efa(model=model,df=mtcars)
model<-psych::fa(mtcars,nfactors=2,rotate="oblimin",fm="pa",oblique.scores=TRUE)
report_efa(model=model,df=mtcars)
model<-psych::fa(mtcars,nfactors=2,rotate="oblimin",fm="ml",oblique.scores=TRUE)
report_efa(model=model,df=mtcars)
model<-psych::fa(mtcars,nfactors=2,rotate="oblimin",fm="minchi",oblique.scores=TRUE)
report_efa(model=model,df=mtcars)
model<-psych::fa(mtcars,nfactors=2,rotate="oblimin",fm="minrank",oblique.scores=TRUE)
report_efa(model=model,df=mtcars)
model<-psych::fa(mtcars,nfactors=2,rotate="oblimin",fm="old.min",oblique.scores=TRUE)
report_efa(model=model,df=mtcars)
model<-psych::fa(mtcars,nfactors=2,rotate="oblimin",fm="alpha",oblique.scores=TRUE)
#report_efa(model=model,df=mtcars)
```

---

report_factorial_anova

Plot means with standard error for every level in a dataframe

---

Description

Plot means with standard error for every level in a dataframe

Usage

```
report_factorial_anova(
  df,
  dv,
  wid,
  within = NULL,
  within_full = NULL,
  between = NULL,
  within_covariates = NULL,
  between_covariates = NULL,
  observed = NULL,
  diff = NULL,
  reverse_diff = FALSE,
  type = 3,
  white.adjust = TRUE,
```

```
  detailed = TRUE,
  return_aov = TRUE,
  file = NULL,
  post_hoc_test = TRUE,
  base_size = 15
)
```

Arguments

df	dataframe
dv	names of dependent variables
wid	names of
within	names of within factors
within_full	names of within factors after data are collapsed to means per condition
between	names of between factors
within_covariates	names of within covariates
between_covariates	mames of between covariates
observed	names in data that are already specified in either within or between that contain predictor variables that are observed variables (not manipulated)
diff	names of variables to collapse in a different score
reverse_diff	If TRUE, triggers reversal of the difference collapse requested by diff
type	sum of squares 1 2 3
white.adjust	if TRUE corrects for heteroscedasticity
detailed	if TRUE returns detailed information
return_aov	if TRUE returns aov object
file	output filename
post_hoc_test	if TRUE outputs post hoc in file
base_size	base font size

Examples

```
set.seed(12345)
df<-data.frame(id=rep(seq(1,80),each=81,1),
               IV1=rep(LETTERS[1:3],each=1,2160),
               IV2=rep(LETTERS[4:6],each=3,720),
               IV3=rep(LETTERS[7:9],each=9,240),
               IV4=rep(LETTERS[10:12],each=27,80),
               stringsAsFactors=FALSE)
cdf<-data.frame(matrix(.01,ncol=4,nrow=4))
correlation_martix<-as.matrix(cdf)
diag(correlation_martix)<-1
cdf<-generate_correlation_matrix(correlation_martix,nrows=nrow(df))+10
names(cdf)<-paste0("DV",1:4)
```

```
df<-data.frame(df,cdf)
df$DV2<-df$DV2+10
df$DV3<-df$DV3+20
df$DV4<-df$DV4+30
df[df$IV1%in%"A",]$DV1<-df[df$IV1%in%"A",]$DV1+1
df[df$IV1%in%"B",]$DV1<-df[df$IV1%in%"B",]$DV1+2
df[df$IV1%in%"C",]$DV1<-df[df$IV1%in%"C",]$DV1+3
cdf(df)
r1<-report_factorial_anova(df=df,wid="id",dv=c("DV1","DV2"),
                           within=c("IV1","IV2"),within_full=c("IV1","IV2"),
                           between=NULL,
                           within_covariates=NULL,between_covariates=NULL,
                           file="anova_within",
                           post_hoc=TRUE)
r2<-report_factorial_anova(df=df,wid="id",dv=c("DV1","DV2"),
                           within=NULL,within_full=NULL,
                           between=c("IV1","IV2"),
                           within_covariates=NULL,between_covariates=NULL,
                           file="anova_between",
                           post_hoc=TRUE)
r3<-report_factorial_anova(df=df,wid="id",dv=c("DV1","DV2"),
                           within=c("IV3","IV4"),within_full=c("IV3","IV4"),
                           between=c("IV1","IV2"),
                           within_covariates=NULL,between_covariates=NULL,
                           file="anova_mixed",
                           post_hoc=FALSE)
r4<-report_factorial_anova(df=df,wid="id",dv=c("DV1","DV2"),
                           within=c("IV1","IV2"),within_full=c("IV1","IV2"),
                           between=NULL,
                           within_covariates=c("DV3","DV4"),between_covariates=NULL,
                           file="anova_within_cov",
                           post_hoc=TRUE)
```

`report_hlr` *Report HLR*

Description

Report HLR

Usage

```
report_hlr(
  df,
  corlist,
  factorlist,
  predictor,
  random_effect,
  file = NULL,
  sheet = "report"
)
```

Arguments

`df`	dataframe
`corlist`	Numeric outcome index
`factorlist`	Numeric predictor index
`predictor`	Character predictor name
`random_effect`	Character random effect name
`file`	Character file
`sheet`	Character sheet

Examples

```
report_hlr(df=infert,corlist=8,factorlist=1,
          predictor="case",random_effect="case")
```

---

report_irt — *Output for irt model*

Description

Output for irt model

Usage

```
report_irt(model, m2 = TRUE, file = NULL)
```

Arguments

`model`	object mirt
`m2`	if TRUE report m2 statistics
`file`	output filename

Examples

```
set.seed(12345)
cormatrix<-psych::sim.rasch(nvar=5,n=50000,low=-4,high=4,d=NULL,a=1,mu=0,sd=1)$items
irt_onefactor<-mirt::mirt(cormatrix,1,empiricalhist=TRUE,calcNull=TRUE)
irt_twofactor<-mirt::mirt(cormatrix,2,empiricalhist=TRUE,calcNull=TRUE)
irt_threefactor<-mirt::mirt(cormatrix,3,empiricalhist=TRUE,calcNull=TRUE)
report_irt(model=irt_onefactor,file="one_factor")
report_irt(model=irt_twofactor,file="two_factors")
report_irt(model=irt_threefactor,file="three_factors")
```

report_lda *Report for MASS::lda*

Description

Report for MASS::lda

Usage

```
report_lda(model, file = NULL, w = 10, h = 10, base_size = 10, title = "")
```

Arguments

model	object from MASS::lda
file	output filename
w	width of pdf file
h	height of pdf file
base_size	base font size
title	plot title

Examples

```
model<-MASS::lda(case~.,data=infert)
result<-report_lda(model=model)
result<-report_lda(model=model,file="lda")
model<-MASS::lda(Species~.,data=iris)
result<-report_lda(model=model,file="lda")
```

report_logistic *Report logistic regression*

Description

Report logistic regression

Usage

```
report_logistic(
  model,
  validation_data = NULL,
  file = NULL,
  title = "",
  w = 10,
  h = 10,
  base_size = 10,
  fast = FALSE
)
```

Arguments

model	object glm
validation_data	validation data
file	output filename
title	plot title
w	width of pdf file. Relevant only when file string is not empty
h	height of pdf file. Relevant only when file string is not empty
base_size	base font size
fast	if TRUE it will not output individual scores and residuals

Note

(1) Problematic values for standardized residuals > +-1.96
Standardized residuals are residuals divided by an estimated standard deviation and they can be interpreted as z scores in that:
95 99 99.99 (2) Problematic values for dfBeta >=1
dfBeta estimates coefficients if the respective case is removed from the dataset
(3) Problematic values for Hat values (leverage) 2 or 3 times the average (k+1/n)
Hat values (leverage),gauge the influence of the observed value of the outcome variable over the predicted values
The average leverage value is defined as (k+1)/n, k=number of predictors, n=number of participants.
Leverage values lie between 0 (no influence) and 1 (complete influence over prediction)
If no cases exert undue influence over the model then all leverage values should be close to (k+1)/n
Hoaglin and Welsch (1978) recommends investigating cases with values greater than twice the average (2(k+1)/n)
Stevens (2002) recommends investigating cases with values greater than three times the average (3(k+1)/n)
(4) Problematic values for VIFs > 10
ASSUMPTIONS
(1) Linearity between continous predictors and the logit (test wether the interaction term between the predictor and its log transformation is significant)
(2) Independence of errors
(3) No multicolinearity

Examples

```
modelcategoricalpredictor0<-glm(case~education,data=infert,family=binomial)
modelcategoricalpredictor1<-glm(case~education,data=infert,family=gaussian)
#modelcategoricalpredictor2<-glm(case~education,data=infert,family=Gamma)
#modelcategoricalpredictor3<-glm(case~education,data=infert,family=inverse.gaussian)
modelcategoricalpredictor4<-glm(case~education,data=infert,family=poisson)
modelcategoricalpredictor5<-glm(case~education,data=infert,family=quasi)
modelcategoricalpredictor6<-glm(case~education,data=infert,family=quasibinomial)
modelcategoricalpredictor7<-glm(case~education,data=infert,family=quasipoisson)
modelcontinuouspredictor0<-glm(case~stratum,data=infert,family=binomial)
```

```
modeltwopredictors0<-glm(case~education+stratum,data=infert,family=binomial)
modeltwopredictors1<-glm(case~education+stratum,data=infert,family=gaussian)
#modeltwopredictors2<-glm(case~education+stratum,data=infert,family=Gamma)
#modeltwopredictors3<-glm(case~education+stratum,data=infert,family=inverse.gaussian)
modeltwopredictors4<-glm(case~education+stratum,data=infert,family=poisson)
modeltwopredictors5<-glm(case~education+stratum,data=infert,family=quasi)
modeltwopredictors6<-glm(case~education+stratum,data=infert,family=quasibinomial)
modeltwopredictors7<-glm(case~education+stratum,data=infert,family=quasipoisson)
report_logistic(model=modelcategoricalpredictor0)
report_logistic(model=modelcategoricalpredictor1)
#report_logistic(model=modelcategoricalpredictor2)
#report_logistic(model=modelcategoricalpredictor3)
report_logistic(model=modelcategoricalpredictor4)
report_logistic(model=modelcategoricalpredictor5)
report_logistic(model=modelcategoricalpredictor6)
report_logistic(model=modelcategoricalpredictor7)
report_logistic(model=modelcontinuouspredictor0)
report_logistic(model=modeltwopredictors0)
report_logistic(model=modelcategoricalpredictor0,
                file="logistic_categorical_predictor",
                validation_data=infert)
report_logistic(model=modelcontinuouspredictor0,
                file="logistic_continuous_predictor",
                validation_data=infert)
report_logistic(model=modeltwopredictors0,
                file="logistic_two_predictors",
                validation_data=infert[1:10,])
```

`report_manova` *Manova result*

Description

Manova result

Usage

```
report_manova(model, file = NULL)
```

Arguments

`model`	object of manova model
`file`	output filename

Note

Pillai-Bartlett trace (V): Represents the sum of the proportion of explained variance on the discriminant functions. As such,it is similar to the ratio of SS M /SS T,which is known as R 2.
Hotelling-s T 2: Represents the sum of the eigenvalues for each variate it compares directly to the

F-ratio in ANOVA
Wilks-s lambda (L): Represents the ratio of error variance to total variance (SS R /SS T) for each variate.
Roy-s largest root: Represents the proportion of explained variance to unexplained variance (SS M /SS R) for the first discriminant function.
ASSUMPTIONS
Independence: Observations should be statistically independent.
Random sampling: Data should be randomly sampled from the population of interest and measured at an interval level.
Multivariate normality: In ANOVA,we assume that our dependent variable is normally distributed within each group. In the case of MANOVA,we assume that the dependent variables (collectively) have multivariate normality within groups.
Homogeneity of covariance matrices: In ANOVA,it is assumed that the variances in each group are roughly equal (homogeneity of variance). In MANOVA we must assume that this is true for each dependent variable,but also that the correlation between any two dependent variables is the same in all groups. This assumption is examined by testing whether the population variance-covariance matrices of the different groups in the analysis are equal.

Examples

```
## Set orthogonal contrasts.
op<-options(contrasts=c("contr.helmert","contr.poly"))
model_mixed<-manova(cbind(yield,foo)~N*P*K,within(npk,foo<-rnorm(24)))
model_between<-manova(cbind(rnorm(24),rnorm(24))~round(rnorm(24),0)*round(rnorm(24),0))
report_manova(model=model_mixed)
report_manova(model=model_between)
```

`report_normality_tests` *Normality tests*

Description

Shapiro-Wilk Anderson-Darling Cramer-von-Mises Shapiro-Francia
Jarque-Bera Kolmogorov-Smirnov Lilliefors Pearson X2

Usage

```
report_normality_tests(df, file = NULL)
```

Arguments

`df`	dataframe with continous or ordinal data
`file`	output filename

Details

returns xlsx file

Examples

```
vector<-generate_missing(rnorm(1000))
df<-generate_missing(mtcars[,1:2])
report_normality_tests(df=df)
report_normality_tests(df=vector,file="normality_tests")
```

`report_oneway` *One way*

Description

One way

Usage

```
report_oneway(
  df,
  dv,
  iv,
  file = NULL,
  w = 10,
  h = 10,
  base_size = 10,
  note = "",
  title = "",
  type = "ci",
  plot_means = FALSE,
  plot_diagnostics = FALSE
)
```

Arguments

`df`	dataframe
`dv`	index of continous variables
`iv`	index of factors
`file`	output filename
`w`	width of pdf file
`h`	height of pdf file
`base_size`	base font size
`note`	text for footnote
`title`	plot title
`type`	type of bar to display "se" "ci" "sd" ""
`plot_means`	if TRUE it will output mean plots and descriptives for plots
`plot_diagnostics`	if TRUE it will output ANOVA diagnostics plots

Note

(1) The Fisher procedure assumes heteroscedasticity
(2) The Welch procedure does not assume heteroscedasticity
(3) The Kruskal Wallis procedure does not assume normality but it is not an alternative for violations of heteroscedasticity
(4) Posthoc Tuckey: not good for unequal sample sizes or heteroscedasticity
(5) Posthoc Games Howell: good for unequal sample sizes and heteroscedasticity

Examples

```
report_oneway(df=df_blood_pressure,
              dv=c(which("bp_before"==names(df_blood_pressure)),
                   which("bp_after"==names(df_blood_pressure))),
              iv=c(which("sex"==names(df_blood_pressure)),
                   which("agegrp"==names(df_blood_pressure))),
              file="anova",
              plot_diagnostics=FALSE,
              plot_means=FALSE)
report_oneway(df=mtcars,dv=2:4,iv=9:10,file="anova_oneway_two_factor")
report_oneway(df=mtcars,dv=2:4,iv=9,file="anova_oneway_one_factor")
report_oneway(df=mtcars,dv=2:4,iv=9,file="anova_oneway_one_factor",
              plot_means=TRUE,plot_diagnostics=TRUE)
```

---

report_pdf *Report pdf*

Description

Report pdf

Usage

```
report_pdf(
  ...,
  plotlist = NULL,
  file = NULL,
  title = NULL,
  w = 10,
  h = 10,
  print_plot = TRUE
)
```

Arguments

`...`	plot objects
`plotlist`	list of plot objects
`file`	output filename

title	output filename
w	width of pdf file
h	height of pdf file
print_plot	if TRUE it prints plot on graphics device

Examples

```
p1<-ggplot(ChickWeight,aes(x=Time,y=weight,colour=Diet,group=Chick))+
          geom_line()+
          ggtitle("Growth curve for individual chicks")+
          theme_bw()
p2<-ggplot(ChickWeight,aes(x=Time,y=weight,colour=Diet))+
          geom_point(alpha=.3)+
          geom_smooth(alpha=.2,size=1,method="loess",formula="y~x")+
          ggtitle("Fitted growth curve per diet")+theme_bw()
cars_plot_multiplot<-plot_multiplot(plotlist=plot_histogram(mtcars[,1:4]),cols=2)
cars_plot_base<-plot_normality_diagnostics(mtcars)
report_pdf(p1,p2,print_plot=TRUE)
report_pdf(p1,p2,file="report",print_plot=FALSE)
report_pdf(plotlist=cars_plot_multiplot,print_plot=TRUE)
report_pdf(plotlist=cars_plot_multiplot,file="report",print_plot=FALSE)
report_pdf(plotlist=cars_plot_base,print_plot=TRUE)
report_pdf(plotlist=cars_plot_base,file="report",print_plot=FALSE)
```

---

report_regression *Regression*

Description

Regression

Usage

```
report_regression(
  model,
  base_size = 10,
  title = "",
  file = NULL,
  w = 10,
  h = 10,
  plot_diagnostics = TRUE
)
```

Arguments

model	object ml
base_size	base font size

`title` plot title

`file` output filename

`w` width of pdf file. Relevant only when file string is not empty

`h` height of pdf file. Relevant only when file string is not empty

`plot_diagnostics`
if TRUE it will output linear model diagnostics plots

Note

(1) Problematic values for standardized residuals > +-1.96
Standardized residuals are residuals divided by an estimated standard deviation and they can be interpreted as z scores in that:
- 95.00 - 99.00 - 99.99 (2) **Studentized residuals** indicate the the ability of the model to predict that case. They follow a t distribution
(3) **DFFits** indicate the difference between the adjusted predicted value and the original predicted value. Adjusted predicted value for a case refers to the predicted value of that case, when that case is excluded from model fit.
(4) **Cook's distance** indicates leverage. Problematic values for cook's distance > 1 Cook and Weisberg (1982).
(5) **Hat values** indicate leverage. Problematic values for Hat values 2 or 3 times the average (k+1/n)
The average leverage value is defined as (k+1)/n, k=number of predictors, n=number of participants. Leverage values lie between 0 (no influence) and 1 (complete influence over prediction)
- Hoaglin and Welsch (1978) recommends investigating cases with values greater than twice the average (2(k+1)/n)
- Stevens (2002) recommends investigating cases with values greater than three times the average (3(k+1)/n)
T-tests test the hypothesis that b's are different from 0
Multiple R^2: Variance Explained
Adjusted R^2: Indicates how much variance in Y would be accounted for if the model is derived from the population from which the sample was taken. Idealy, R^2 = Adjusted R^2
F-Statistic tests the null hypothesis is that the overall model has no effect
Covariance ratios critical values CVR>1+[3(k+1)/n] CRV<1-[3(k+1)/n]. In general we should obtain small values or we may have to remove cases
ASSUMPTIONS
(1) variable types: All predictors must be quantitative or categorical (with two levels), and the outcome variable must be quantitative (interval data), continuous and unbounded (no constraints on the variability of the outcome) (2) Non-zero variance
(3) No perfect multicollinearity
(4) Predictors are uncorrelated with -external variables-
(5) Homoscedasticity: At each level of the predictor variable(s), the variance of the residual terms should be constant. Residuals at each level of the predictor(s) should have similar variance (homoscedasticity)
(6) Independent errors: For any two observations the residual terms should be uncorrelated (or independent)
This eventuality is sometimes described as a lack of autocorrelation. This assumption can be tested with the Durbin-Watson test,which tests for serial correlations between errors. Specifically, it tests whether adjacent residuals are correlated The size of the Durbin-Watson statistic depends upon the

number of predictors in the model and the number of observations As a very conservative rule of thumb, values less than 1 or greater than 3 are definitely cause for concern; however,values closer to 2 may still be problematic depending on your sample and model R also provides a p-value of the autocorrelation. Be very careful with the Durbin-Watson test, though, as it depends on the order of the data: if you reorder your data, you-ll get a different value
(7) Normally distributed errors: It is assumed that the residuals in the model are random, normally distributed variables with a mean of 0
(8) Independence: It is assumed that all of the values of the outcome variable are independent (in other words, each value of the outcome variable comes from a separate entity)
(9) Linearity: The mean values of the outcome variable for each increment of the predictor(s) lie along a straight line

Examples

```
form<-formula(mpg~qsec)
regressionmodel<-lm(form,data=mtcars)
multipleregressionmodel<-lm(mpg~qsec*hp*wt*drat,data=mtcars)
res<-report_regression(model=regressionmodel,plot_diagnostics=TRUE)
res<-report_regression(model=multipleregressionmodel)
res<-report_regression(model=regressionmodel,file="regression")
res<-report_regression(model=multipleregressionmodel,
                       file="regression",
                       plot_diagnostics=TRUE)
```

`report_ttests` *T test*

Description

T test

Usage

```
report_ttests(df, dv, iv, file = NULL, ...)
```

Arguments

`df`	dataframe
`dv`	index of continous variables
`iv`	index of factors
`file`	output filename
`...`	Arguments passed on to `stats::t.test`
	`x` a (non-empty) numeric vector of data values.

Examples

```
report_ttests(df=df_insurance,dv=which("charges"==names(df_insurance)),iv=c(2))
report_ttests(df=df_insurance,dv=which("charges"==names(df_insurance)),iv=c(4))
report_ttests(df=df_insurance,dv=which("charges"==names(df_insurance)),iv=c(2,4))
report_ttests(df=df_insurance,dv=which("charges"==names(df_insurance)),iv=c(2,4),alternative="two.sided")
report_ttests(df=df_insurance,dv=which("charges"==names(df_insurance)),iv=c(2,4),alternative="less")
report_ttests(df=df_insurance,dv=which("charges"==names(df_insurance)),iv=c(2,4),alternative="greater")
report_ttests(df=df_insurance,dv=which("charges"==names(df_insurance)),iv=c(2,4),var.equal=TRUE)
report_ttests(df=mtcars,dv=1:7,iv=8:10,var.equal=TRUE,file="ttest")
```

---

report_wtests   *Wilcoxon test*

Description

Wilcoxon test

Usage

```
report_wtests(df, dv, iv, file = NULL, ...)
```

Arguments

df	dataframe
dv	index of continous variables
iv	index of factors
file	output filename
...	Arguments passed on to `stats::wilcox.test` `x` numeric vector of data values. Non-finite (e.g., infinite or missing) values will be omitted.

Examples

```
report_wtests(df=df_insurance,dv=which("charges"==names(df_insurance)),iv=c(2))
report_wtests(df=df_insurance,dv=which("charges"==names(df_insurance)),iv=c(4))
report_wtests(df=df_insurance,dv=which("charges"==names(df_insurance)),iv=c(2,4))
report_wtests(df=df_insurance,dv=which("charges"==names(df_insurance)),iv=c(2,4),alternative="two.sided")
report_wtests(df=df_insurance,dv=which("charges"==names(df_insurance)),iv=c(2,4),alternative="less")
report_wtests(df=df_insurance,dv=which("charges"==names(df_insurance)),iv=c(2,4),alternative="greater")
report_wtests(df=df_insurance,dv=which("charges"==names(df_insurance)),iv=c(2,4),var.equal=TRUE)
report_wtests(df=df_insurance,dv=which("charges"==names(df_insurance)),iv=c(2,4),var.equal=TRUE,file="wilcoxon
```

report_xgboost *Report for xgboost::xgb.train*

Description

Report for xgboost::xgb.train

Usage

```
report_xgboost(
  model,
  validation_data = NULL,
  label = NULL,
  file = "xgboost",
  w = 10,
  h = 10,
  base_size = 10,
  title = "",
  fast = FALSE
)
```

Arguments

model	object from xgboost::xgb.train
validation_data	validation data
label	outcome variable name
file	output filename
w	width of pdf file
h	height of pdf file
base_size	base font size
title	plot title
fast	if TRUE error values are not saved in output

Examples

```
infert_formula<-formula(case~education+spontaneous+induced)
boston_formula<-formula(medv~crim+zn+indus+chas+nox+rm+age+dis+rad+tax+ptratio+black+lstat)
train_test_classification<-k_fold(df=infert,model_formula=infert_formula)
train_test_regression<-k_fold(df=MASS::Boston,model_formula=boston_formula)
xgb_classification<-xgboost::xgb.train(
                  data=train_test_classification$xgb$f1$train,
                  watchlist=train_test_classification$xgb$f1$watchlist,
                  eta=.1,
                  nthread=8,
                  nround=20,
```

```
                    objective="binary:logistic")
xgb_regression<-xgboost::xgb.train(
                data=train_test_regression$xgb$f1$train,
                watchlist=train_test_regression$xgb$f1$watchlist,
                eta=.3,
                nthread=8,
                nround=20)
report_xgboost(model=xgb_classification,
               validation_data=train_test_classification$f$test$f1,
               label=train_test_classification$outcome,
               file="Classification")
report_xgboost(model=xgb_regression,
               validation_data=train_test_regression$f$test$f1,
               label=train_test_regression$outcome,
               file="Regression")
```

`response_dimension` *index parameter and items relative to their dimensions*

Description

index parameter and items relative to their dimensions

Usage

```
response_dimension(response, dimensions, items)
```

Arguments

`response`	vector one to number of items
`dimensions`	number of dimensions
`items`	item comparisons

Examples

```
response_dimension(c(1:18),3,c(1,2))
response_dimension(c(1:18),3,c(1,3))
response_dimension(c(1:18),3,c(2,3))
```

`response_frequency` *Response frequencies*

Description

returns count proportion percent

Usage

```
response_frequency(
  df,
  max = 10,
  uniqueitems = NULL,
  type = "percent",
  file = NULL
)
```

Arguments

df	dataframe
max	maximum score
uniqueitems	number of unique items
type	"frequency" "proportion" "percent" "all"
file	output filename

Details

returns dataframe

Examples

```
response_frequency(mtcars[,c("gear","carb")],uniqueitems=1:8,type="frequency")
response_frequency(mtcars[,c("gear")],uniqueitems=1:8,type="proportion")
response_frequency(mtcars[,c("gear","carb")],uniqueitems=1:8,type="percent")
response_frequency(mtcars[,c("gear","carb")],uniqueitems=1:8,type="all")
response_frequency(mtcars[,c("gear","carb")],uniqueitems=1:8,type="all",
                  file="descriptives")
```

`result_confusion_performance` *Plot performance of confusion matrix for different cut off points*

Description

This function generates a plot to visualize the performance of a confusion matrix at various cut-off points. It evaluates the proportion of correct classifications and identifies the optimal cut-off point.

Usage

```
result_confusion_performance(
  observed,
  predicted,
  step = 0.1,
  base_size = 10,
  title = ""
)
```

Arguments

`observed`	Vector of observed outcomes. This can be numeric or factor values representing the true class labels.
`predicted`	Vector of predicted outcome probabilities. This should have the same length as the observed vector and represent the predicted probabilities.
`step`	Numeric value representing the stepping for tested cut values. Defaults to 0.1.
`base_size`	Integer value representing the base font size for the plot. Defaults to 10.
`title`	String representing the title of the plot. Defaults to an empty string.

Details

This function evaluates the performance of a confusion matrix at different cut-off points. It iterates through a range of cut-off points, calculates the confusion matrix,and evaluates the proportion of correct classifications for each cut-off.

The function generates a plot that includes: -The proportion of correct classifications for different cut-off points. -Vertical lines indicating the optimal cut-off point. -A legend representing different performance metrics. -A caption showing the number of observations and the optimal cut-off point.

The function returns a list containing the plot,the data frame with cut-off performance,the optimal cut-off point, and the confusion matrix at the optimal cut-off.

Examples

```
# Example with numeric class labels
df<-data.frame(matrix(.999,ncol=2,nrow=2))
correlation_matrix<-as.matrix(df)
diag(correlation_matrix)<-1
```

```
df<-generate_correlation_matrix(correlation_matrix,nrows=1000)
df$X1<-ifelse(abs(df$X1) < 1,0,1)
df$X2<-abs(df$X2)
df$X2<-(df$X2-min(df$X2))/(max(df$X2)-min(df$X2))
result_confusion_performance(observed=round(abs(df$X1),0),
                             predicted=abs(df$X2),
                             step=0.01)
result_confusion_performance(observed=c(1,2,3,1,2,3),
                             predicted=abs(rnorm(6,0,sd=0.1)))
```

---

round_dataframe *Round dataframe*

Description

It only processes numeric values in a dataframe

Usage

```
round_dataframe(df, digits = 0, type = "round")
```

Arguments

df	dataframe
digits	decimal points to return. It works only with "round" type
type	"round" "ceiling" "floor" "tenth"

Examples

```
round_dataframe(df=change_data_type(df=mtcars,type="factor"),digits=0)
round_dataframe(df=change_data_type(df=mtcars,type="character"),digits=0)
round_dataframe(df=mtcars,digits=0)
round_dataframe(df=mtcars,digits=0,type="ceiling")
round_dataframe(df=mtcars,digits=0,type="floor")
round_dataframe(df=mtcars*100,digits=2,type="tenth")
```

---

shrout *Shrout reliability*

Description

Shrout reliability

Usage

```
shrout(sperson, spersonitem, stime, spersontime, serror, m, k)
```

Arguments

sperson	variance component of participant
spersonitem	variance component of participant by item interaction
stime	variance component of time
spersontime	variance component of participant by time interaction
serror	variance component of error
m	m item reports
k	k time points

Examples

```
design<-expand.grid(time=1:3,item=1:2,person=1:10)
design<-change_data_type(design,type="factor")
design$response<-rnorm(30,0,0.1)
model<-mixlm::lm(response~r(time)*r(person)+r(item)*r(person),data=design)
result<-extract_components(model)
vc<-result$components
shrout(sperson=vc[2,3],spersonitem=vc[5,3],stime=vc[1,3],
       spersontime=vc[4,3],serror=vc[6,3],3,3)
```

---

simulate_cfa_fit   *Simulate CFA from coefficients*

Description

Simulates cfa from coefficients Simulates cfa from correlations of obeserved data Returns fit indices for predefined set of sample sizes

Usage

```
simulate_cfa_fit(
  model_sim = NULL,
  model = NULL,
  df = NULL,
  minnobs = 50,
  maxnobs = 1000,
  stepping = 10,
  file = NULL,
  w = 10,
  h = 10
)
```

Arguments

`model_sim`	lavaan model spesification with defined coefficients
`model`	lavaan model spesification with free coefficients
`df`	dataframe
`minnobs`	start sample size
`maxnobs`	end sample size
`stepping`	stepping
`file`	output filename
`w`	width of pdf file
`h`	height of pdf file

Examples

```
model_sim='LATENT=~1*X1+0.5*X2+1.5*X3+1.5*X4+X5'
model='LATENT=~X1+X2+X3+X4+X5'
df<-lavaan::simulateData(model=model_sim,model.type="cfa",
                         return.type="data.frame",sample.nobs=1000)
# simulate_cfa_fit(model_sim=model_sim,model=model,
#                  minnobs=50,maxnobs=1000,stepping=100,file="report")
# simulate_cfa_fit(model=model,df=df,
#                  minnobs=50,maxnobs=1000,stepping=100,file="report")
```

---

simulate_correlation_from_sample

Generate a dataframe that produces the same correlation matrix as the input dataframe

---

Description

Generate a dataframe that produces the same correlation matrix as the input dataframe

Usage

```
simulate_correlation_from_sample(cordata, nrows = 10)
```

Arguments

`cordata`	dataframe
`nrows`	number of rows to generate

Examples

```
correlation_matrix<-generate_correlation_matrix()
stats::cor(correlation_matrix)
simulate_correlation_from_sample(correlation_matrix,nrows=1000)
stats::cor(simulate_correlation_from_sample(correlation_matrix,nrows=1000))
```

split_str *Split string to dataframe*

Description

Split string to dataframe

Usage

```
split_str(vector, split = "/", include_original = FALSE)
```

Arguments

vector	String
split	Separation character
include_original	if TRUE it will return the input on a separate collumn

Examples

```
string<-paste0(1:10,"/",
              generate_string(nchar=2,vector_length=10),"/",
              generate_string(nchar=2,vector_length=10),"/",
              generate_string(nchar=2,vector_length=10))
split_str(string,split="/")
```

split_str_df *Split string in dataframe*

Description

Split string in dataframe

Usage

```
split_str_df(df, split = "/", type = "row", index, ...)
```

Arguments

df	dataframe
split	Separation character
type	"row" "collumn" if "row" it will split the string of row names and it will display it on seperate collumns if "collumn" it will split the string of a spesified collumn and it will display it on separate collumns
index	Numeric index of collumn to split. This is only relevant if type="collumn"
...	arguments passed to split_str

Examples

```
df<-generate_correlation_matrix()
string<-paste0(1:nrow(df),"/",
              generate_string(nchar=2,vector_length=nrow(df)),"/",
              generate_string(nchar=2,vector_length=nrow(df)),"/",
              generate_string(nchar=2,vector_length=nrow(df)))
row.names(df)<-string
split_str_df(df,split="/",type="row")
df[,1]<-string
split_str_df(df,split="/",type="collumn",index=1)
```

`stat_word_char` *Text similarity measures*

Description

Text similarity measures

Text similarity measures

Usage

```
stat_word_char(text)

stat_word_char(text)
```

Arguments

`text`	character vector

Examples

```
text<-"There are many variations of passages of Lorem Ipsum available,
but the majority have suffered alteration in some form, by injected humour,
or randomised words which don't look even slightly believable."
stat_word_char(text)
text<-"There are many variations of passages of Lorem Ipsum available,
but the majority have suffered alteration in some form, by injected humour,
or randomised words which don't look even slightly believable."
stat_word_char(text)
```

string_aes — *Adjust string aesthetics*

Description

Treats spesific characters such as ".", as separating characters and separates strings with space. Trims leading and trailing spaces and capitalizes the first letter of the string and lowers the rest.

Usage

```
string_aes(
  vector,
  characterlist = c(".", "_", "-", ",", "$", "<p>", "</p>", "<br>", "<br/>", "<B>",
    "</B>", "<BR/>", "|", "/", " "),
  proper = TRUE
)
```

Arguments

vector	Vector
characterlist	List the list of characters to treat as separating characters
proper	Logical TRUE capitalizes the first letter in sentense format

Examples

```
vector<-c("TES.T","TES<p>T","TES&nbspT")
string_aes(vector=vector)
string_aes(vector=vector,proper=FALSE)
string_aes(vector=vector,proper=TRUE)
```

sub_str — *Return n characters from left or right*

Description

Return n characters from left or right

Usage

```
sub_str(x, n = 2, type)
```

Arguments

x	Character
n	Number of characters to return
type	"right" "left"

Examples

```
sub_str("12345",n=2,type="right")
sub_str("12345",n=2,type="left")
```

`swap` *Reverse a numeric vector*

Description

Reverse a numeric vector

Usage

```
swap(vector)
```

Arguments

`vector`	numeric

Examples

```
swap(c(1:10,1,2,3))
```

`symmetric_matrix` *Symmetric Matrix*

Description

Symmetric Matrix

Usage

```
symmetric_matrix(matrix, duplicate = "lower", diagonal = NULL)
```

Arguments

`matrix`	matrix
`duplicate`	"upper" duplicates upper triangle "lower" duplicates lower triangle
`diagonal`	diagonal values

Examples

```
m_lower<-matrix_triangle(matrix(1:9,nrow=3,ncol=3),type="lower",diagonal=NA)
m_upper<-matrix_triangle(matrix(11:19,nrow=3,ncol=3),type="upper",diagonal=NA)
symmetric_matrix(matrix=m_lower,duplicate="lower",diagonal=NA)
symmetric_matrix(matrix=m_upper,duplicate="upper",diagonal=NA)
```

`tag_pos` *Part of speech tagging*

Description

Part of speech tagging

Part of speech tagging

Usage

```
tag_pos(text)

tag_pos(text)
```

Arguments

`text`	character vector

Examples

```
text1<-"word_one word_two word_three"
text2<-"word_three word_four word_six"
text3<-"All the Lorem Ipsum generators on the Internet tend to repeat predefined
chunks as necessary, making this the first true generator on the Internet."
text4<-"It uses a dictionary of over 200 Latin words, combined with a handful of
model sentence structures, to generate Lorem Ipsum which looks reasonable."
text5<-"The generated Lorem Ipsum is therefore always free from repetition,
injected humour, or non-characteristic words etc."
text<-c(text1,text2,text3,text4,text5)
tag_pos(text)
text1<-"word_one word_two word_three"
text2<-"word_three word_four word_six"
text3<-"All the Lorem Ipsum generators on the Internet tend to repeat predefined
chunks as necessary, making this the first true generator on the Internet."
text4<-"It uses a dictionary of over 200 Latin words, combined with a handful of
model sentence structures, to generate Lorem Ipsum which looks reasonable."
text5<-"The generated Lorem Ipsum is therefore always free from repetition,
injected humour, or non-characteristic words etc."
text<-c(text1,text2,text3,text4,text5)
tag_pos(text)
```

`text_similarity` *Text similarity measures*

Description

Text similarity measures

Text similarity measures

Usage

```
text_similarity(text1, text2)

text_similarity(text1, text2)
```

Arguments

`text1`	character vector
`text2`	character vector

Examples

```
text1<-"word_one word_two word_three"
text2<-"word_three word_four word_six"
text3<-"All the Lorem Ipsum generators on the Internet tend to repeat predefined
chunks as necessary, making this the first true generator on the Internet."
text4<-"It uses a dictionary of over 200 Latin words, combined with a handful of
model sentence structures, to generate Lorem Ipsum which looks reasonable."
text5<-"The generated Lorem Ipsum is therefore always free from repetition,
injected humour, or non-characteristic words etc."
text<-c(text1,text2,text3,text4,text5)
text<-unlist(strsplit(text,split=" "))
text1<-unlist(strsplit(text1,split=" "))
text2<-unlist(strsplit(text2,split=" "))
text3<-unlist(strsplit(text3,split=" "))
text4<-unlist(strsplit(text4,split=" "))
text5<-unlist(strsplit(text5,split=" "))
text_similarity(text1,text1)
text_similarity(text1,text2)
text_similarity(text1,text3)
text_similarity(text1,text4)
text1<-"word_one word_two word_three"
text2<-"word_three word_four word_six"
text3<-"All the Lorem Ipsum generators on the Internet tend to repeat predefined
chunks as necessary, making this the first true generator on the Internet."
text4<-"It uses a dictionary of over 200 Latin words, combined with a handful of
model sentence structures, to generate Lorem Ipsum which looks reasonable."
text5<-"The generated Lorem Ipsum is therefore always free from repetition,
injected humour, or non-characteristic words etc."
text<-c(text1,text2,text3,text4,text5)
```

```
text<-unlist(strsplit(text,split=" "))
text1<-unlist(strsplit(text1,split=" "))
text2<-unlist(strsplit(text2,split=" "))
text3<-unlist(strsplit(text3,split=" "))
text4<-unlist(strsplit(text4,split=" "))
text5<-unlist(strsplit(text5,split=" "))
text_similarity(text1,text1)
text_similarity(text1,text2)
text_similarity(text1,text3)
text_similarity(text1,text4)
```

`trim_df` *Trim whitespace in dataframe*

Description

Trim whitespace in dataframe

Usage

```
trim_df(df)
```

Arguments

`df`	dataframe

Examples

```
string<-data.frame(str1=rep(paste0(sample(c(LETTERS,rep(" ",10))),collapse=""),10),
                   str2=rep(paste0(sample(c(LETTERS,rep(" ",10))),collapse=""),10),
                   num1=rnorm(10),
                   stringsAsFactors=FALSE)
trim_df(string)
```

`ts_smoothing` *Smoothing*

Description

smoothing for timeseries. uses base plot

Usage

```
ts_smoothing(
  df,
  start = 0.01,
  stop = 2,
  step = 0.001,
  title = "",
  type = "kernel"
)
```

Arguments

df	ts object
start	start value
stop	stop value
step	step
title	plot title
type	"default" "kernel" "lowess" "friedman" "splines" "polynomial" "linear"

Details

returns plot

Examples

```
ts_data<-ts(UKDriverDeaths,start=1969,end=1984,frequency=12)
par(mfrow=c(2,2))
ts_smoothing(ts_data,start=.01,stop=2,step=.01,title="Driver Deaths in UK",type="default")
ts_smoothing(ts_data,start=.01,stop=2,step=.01,title="Driver Deaths in UK",type="polynomial")
ts_smoothing(ts_data,start=.01,stop=2,step=.01,title="Driver Deaths in UK",type="linear")
ts_smoothing(ts_data,start=.01,stop=2,step=.01,title="Driver Deaths in UK",type="kernel")
ts_smoothing(ts_data,start=.01,stop=2,step=.01,title="Driver Deaths in UK",type="lowess")
ts_smoothing(ts_data,start=.01,stop=2,step=.01,title="Driver Deaths in UK",type="friedman")
ts_smoothing(ts_data,start=.01,stop=2,step=.01,title="Driver Deaths in UK",type="splines")
```

---

wrapper *Wrap string*

Description

Wrap string

Usage

```
wrapper(x, ...)
```

Arguments

`x`	title
`...`	arguments passed to strwrap

Examples

```
wrapper(rep("sting",50),30)
```

`write_txt` *Log console in file*

Description

Logs console in file and then displays log in console

Usage

```
write_txt(input, file = NULL)
```

Arguments

`input`	Script to log in log file
`file`	Filename of log

Examples

```
write_txt(mtcars)
write_txt(mtcars,file="mtcars")
```

Index