Package 'skMisc'

January 18, 2021

Title Sercan Kahveci's Miscellaneous Functions

Version 0.01

Description Contains a wide range of functions.

Depends R (¿= 3.6.1), magrittr, dplyr, doParallel, lmerTest

Imports tidyr, knitr, quanteda

License GPL-3

BugReports https://github.com/Spiritspeak/skMisc/issues

LazyData true

ByteCompile true

RoxygenNote 7.1.1

R topics documented:

`AnovaTable` *Compare multilevel models*

Description

Compare multilevel models

Usage

```
AnovaTable(
  ...,
  fullmodel,
  models,
  serial = F,
  suppress = c("AIC", "deviance", "logLik")
)

## S3 method for class 'AnovaTable'
print(x)
```

Arguments

`...`	Model objects to be compared
`fullmodel`	A model to which all other models are to be compared; only use if `...` is not specified.
`models`	Models to compare to `fullmodel`. Only use if `...` is not specified.
`serial`	If TRUE, models are compared serially; if false, all models will be compared to the first.
`suppress`	Character vector of column names to suppress in printed output.

Value

A data.frame containing model fit metrics such as AIC, BIC, marginal R-squared (the effect size of fixed effects only), conditional R-squared (the effect size of all model terms), loglikelihood, deviance, and a likelihood ratio test.

`AutocorPlot` *Per-subject Autocorrelation Plotting*

Description

Per-subject Autocorrelation Plotting

Usage

```
AutocorPlot(ds, ppvar, rtvar, scope = 64)
```

Arguments

`ds`	a dataset
`ppvar`	name of the variable indicating participant ID
`rtvar`	name of the variable indicating reaction time
`scope`	numeric, the maximum lag at which to compute autocorrelation.

Examples

```
AutocorPlot(ds=ToothGrowth,ppvar="supp",rtvar="len",scope=10)
```

`clamp` *clamp*

Description

clamp

Usage

```
clamp(val, minval, maxval)
```

Arguments

`val`	The vector/matrix to clamp
`minval`	Minimum value; all lower values are clamped to this value
`maxval`	Maximum value; all higher values are clamped to this value

Value

Clamped vector.

Examples

```
clamp(0:10,2,8)
```

coerce *coerce a vector to contain only TRUE and FALSE*

Description

coerce a vector to contain only TRUE and FALSE

Usage

```
coerce(x, default = FALSE)
```

Arguments

x	Numeric/logical vector/matrix to coerce into TRUE/FALSE
default	default returned value if NULL or NA is encountered

Value

logical vector or matrix with only T and F

Examples

```
coerce(NULL)
# FALSE

coerce(c(T,F,NA,NA,T))
# T F F F T

coerce(matrix(c(T,T,F,F,NA,NA),nrow=2))
#     [,1]  [,2]  [,3]
#[1,] TRUE FALSE FALSE
#[2,] TRUE FALSE FALSE
```

colVars *Compute column and row variances*

Description

Compute column and row variances

Usage

```
colVars(x, na.rm = T)

rowVars(x, na.rm = T)
```

Arguments

x	an input matrix of data.frame
na.rm	Logical indicating whether NA values should be omitted before variance computation

Examples

```
colVars(WorldPhones)
rowVars(WorldPhones)
```

---

combobulate *Get all possible combinations of strings*

Description

`combobulate()` returns all possible combinations of the provided character strings, each combination merged into a single string.

Usage

```
combobulate(...)
```

Arguments

...	Character vectors to combobulate.

Value

A character vector.

Examples

```
combobulate("Hello ",c("Sir","Madam"),", ",c("may I take your order?","what shall it be?"))
# [1] "Hello Sir, may I take your order?"
# [2] "Hello Madam, may I take your order?"
# [3] "Hello Sir, what shall it be?"
# [4] "Hello Madam, what shall it be?"
```

---

comboTable *Generate a matrix of combinations of values*

Description

Generate a matrix of combinations of values

Usage

```
comboTable(...)
```

Arguments

...	Character vectors, named or unnamed, or unquoted names of named arguments. Character vectors will be used to generate a matrix where each row represents a unique combination of all values, akin to `expand.grid()`. Arguments which are unquoted names of named arguments will become copies of the column generated by the eponymous named character vector.

Value

A matrix.

Examples

```
hh<-c("a","b")
comboTable(a=letters[1:3], b=2,a,b,c=c("e","f"),d,c,d=hh,"huh",a,hh)
```

---

compcorr *Test if two correlation coefficients significantly differ*

Description

Uses Fisher's r to z transformation, then performs a z-test on the resulting z-scores

Usage

```
compcorr(cor1, cor2, n1, n2)
```

Arguments

cor1, cor2	Correlation values being compared
n1, n2	Sample sizes of the correlation coefficients

Value

List containing the z-score and p-value

References

http://vassarstats.net/rdiff.html

Examples

```
compcorr(.1,.6,50,100)
```

CorrCrunch *Analyse the robustness of a correlation*

Description

`CorrCrunch()` computes the minimum number of cases that need to be removed from a dataset to flip the sign of a correlation coefficient. This can be useful in distinguishing genuine correlations from spurious findings that hinge on one or two outliers. Cases are removed iteratively; in each iteration the case that maximally shrinks the correlation coefficient is removed.

Usage

```
CorrCrunch(x, y, verbose = F)
```

Arguments

`x, y`	Numeric vectors to correlate.
`verbose`	if TRUE, prints verbose output.

Value

A list containing the number of cases that need to be removed to flip the sign of the correlation coefficient; the proportion removed cases in the data; and a data.frame without these cases.

Examples

```
CorrCrunch(mtcars$mpg,mtcars$wt)
#Holdout needed to flip the sign: 19 (63.33%)
#Final r: 0.01181141
```

`CorTable` *Create a Correlation Table*

Description

Create a Correlation Table

Usage

```
CorTable(df, rowids, columnids, rowdf, columndf)
```

Arguments

`df`	A data.frame.
`rowids, columnids`	character vectors containing column names from `df` that need to be correlated.
`rowdf, columndf`	data.frames whose columns need to be correlated. Either `df,rowids,& columnids` or `rowdf & columndf` are required.

Value

A formatted markdown table containing correlation coefficients, p-values, and the number and percentage of cases that need to be removed to flip the sign of each correlation coefficient.

Examples

```
CorTable(mtcars,rowids=c("mpg","disp","hp"),columnids=c("drat","wt","qsec"))

CorTable(rowdf=mtcars[,c(1,3,4)],columndf=mtcars[,5:7])
```

`df.init` *Initiate an empty data frame*

Description

Initiate an empty data frame

Usage

```
df.init(namelist)
```

Arguments

`namelist`	A character vector of column names.

Value

A data.frame with 0 rows.

`ExpandFormula` *Parse a lme4 formula and return all main effects and interactions as separate terms*

Description

Parse a lme4 formula and return all main effects and interactions as separate terms

Usage

```
ExpandFormula(form)
```

Arguments

`form`	

Value

The same formula, but with all interactions and mai neffects as separate terms

Examples

```
ExpandFormula(rt ~ pull * target + (pull * target | subjectid))
#rt ~ pull + target + pull:target + (pull + target + pull:target | subjectid)
```

ExtractRandomTerms — *Extract random terms from a lme4 formula*

Description

Extract random terms from a lme4 formula

Usage

```
ExtractRandomTerms(form)
```

Arguments

form	A formula

Value

A named list containing character vectors with random terms; names are group variables.

Examples

```
ExtractRandomTerms(grade ~ ChildIQ * TeacherSkill * SchoolType +
                           (ChildIQ * TeacherSkill | School))
#$School
#[1] "ChildIQ"              "TeacherSkill"         "ChildIQ:TeacherSkill"
```

FindTopTerms — *Find all model terms that are not moderated by a higher-order interaction*

Description

Find all model terms that are not moderated by a higher-order interaction

Usage

```
FindTopTerms(form)
```

Arguments

form	a formula

Value

A character vector containing all model terms that are not moderated by a higher-order interaction.

Examples

```
FindTopTerms(speed ~ skill + weight * friction)
#[1] "skill"           "weight:friction"
```

hilight *Plot highlighted text*

Description

Plot highlighted text

Usage

```
hilight(x, y, s, bg = "yellow")
```

Arguments

x	x position
y	y position
s	text
bg	highlight color

Examples

```
plot(mtcars$mpg,mtcars$wt,col=mtcars$cyl)
hilight(27,2.5,"Light and\nefficient")
hilight(17,4.5,"Heavy and\ninefficient")
```

LevenshteinDistance *Levenshtein distance*

Description

Counts the number of single character deletions, insertions, and substitutions that need to be performed to turn the source string into the target string.

Usage

```
LevenshteinDistance(source, target)
```

Arguments

source, target	Strings to be compared.

Value

The Levenshtein distance between the two strings.

Examples

```
LevenshteinDistance("Yoghurt","Youtube")
```

`logit.weightfun` *Downweight outliers*

Description

Computes weights; trials within certain bounds of the mean receive the maximum weight while trials outside these bounds are downweighted to 0 or an optional minimum.

Usage

```
logit.weightfun(
  x,
  mean = mean(x),
  s = sd(x),
  sdist = 3,
  taper = 10,
  scale = c("max", "norm"),
  min = 0
)
```

Arguments

`x`	A numeric vector
`mean`	An optional mean of the vector
`s`	An optional standard deviation of the vector
`sdist`	The number of standard deviations beyond which values should be downweighted
`taper`	A number indicating how strongly values exceeding the standard deviation should taper off
`scale`	How the weight vector should be scaled: "norm" sets the sum to 1, "max" sets the maximum to 1.
`min`	A minimum weight.

Value

A numeric vector of weights

`multimerge` *Merge Multiple Data Frames*

Description

This function makes calls to `merge()` to merge every other dataset with the one next to it, repeating until only one dataset remains.

Usage

```
multimerge(x, ...)
```

Arguments

x	a list of data frames
...	all other arguments for merge can be provided here

Value

A single, merged data.frame

Author(s)

Sercan Kahveci

Examples

```
#generate test data
testlist<-list()
lsize<-50
for(i in 1:lsize){
  testlist[[i]]<-data.frame(key=sample(1:500,100),
                            junk=letters[sample(1:26,100,replace=T)])
  colnames(testlist[[i]])[2]<-paste0("info",i)
}
multimerge(testlist,by="key",all=T)
```

OLcrunch *Crunch Outliers*

Description

Crunch Outliers

Usage

```
OLcrunch(x, DS = 3, hardlimit = NULL)
```

Arguments

x	Numeric vector to remove outliers from
DS	A positive numeric value. If value exceeds this many standard deviations, it is counted as an outlier
hardlimit	A numeric vector with two values. If set, values below the first value and above the second will be counted as outliers, and means/standard deviations will be computed from values within these bounds only.

Value

Vector with outlying values set to NA

pair *Create unique pairs*

Description

Combines vectors such that unique unordered sets are derived from the vectors' cross sections.

Usage

```
pair(...)
```

Arguments

...	two or more vectors of equal length

Value

a character vector consisting of all input vectors concatenated term-by-term and in alphabetic order.

Examples

```
pair(1:4,4:1)
#[1] "1-4" "2-3" "2-3" "1-4"
```

read.csv.folder *Read and merge all .csv files in a folder*

Description

Read and merge all .csv files in a folder

Usage

```
read.csv.folder(
  folder = "./",
  readfunc = list(read.csv, read.csv2, read.table)
)
```

Arguments

folder	path to a folder
readfunc	list of functions that will be used to read the files; if the first function fails, the second function will be used, etc.

Value

A data.frame containing all merged .csv files

RemoveTopTerms *Remove all possible models with one unmoderated term removed*

Description

Remove all possible models with one unmoderated term removed

Usage

```
RemoveTopTerms(form, randeff = "")
```

Arguments

form	A formula
randeff	The name of the group from which unmoderated terms should be removed. To remove from fixed effects, use "" (the default).

Value

A list of formulas which have one unmoderated term removed each. The name of each list item is the term which was removed.

Examples

```
RemoveTopTerms(a ~ b * c + d + (1|e))
#$d
#a ~ b + c + b:c + (1 | e)
#$`b:c`
#a ~ b + c + d + (1 | e)
```

retype *Change classes of columns in a data.frame*

Description

`retype()` changes the class of specific columns; `retype_all()` changes the class of all columns of a given class.

Usage

```
retype(df, ...)

retype_all(df, from, to)
```

Arguments

`df`	A data.frame
`...`	Unquoted column names, paired with the desired class, e.g. `age = numeric(),language = character()`
`from`	An empty vector of the class to convert from, or a string. Columns sharing the class of argument `from` will be converted to the class of argument `to`.
`to`	An empty vector of the class to convert to, or a string. Columns sharing the class of argument `from` will be converted to the class of argument `to`.

Examples

```
sapply(ToothGrowth,class)
#      len      supp      dose
#"numeric"  "factor" "numeric"
NewToothGrowth <- retype(ToothGrowth, supp = character(), dose = factor())
sapply(NewToothGrowth,class)
#      len        supp        dose
#"numeric" "character"    "factor"

sapply(mtcars,class)
#       mpg       cyl      disp        hp      drat        wt
# "numeric" "numeric" "numeric" "numeric" "numeric" "numeric"
#      qsec        vs        am      gear      carb
# "numeric" "numeric" "numeric" "numeric" "numeric"

newmtcars <- retype_all(mtcars,from="numeric",to="character")
sapply(newmtcars,class)
#         mpg         cyl        disp          hp        drat
# "character" "character" "character" "character" "character"
#          wt        qsec          vs          am        gear        carb
# "character" "character" "character" "character" "character" "character"
```

`setColNames` *Set column and row names of an object These are convenience functions that return an object with its column or row names changed. Use it in pipes.*

Description

Set column and row names of an object These are convenience functions that return an object with its column or row names changed. Use it in pipes.

Usage

```
setColNames(x, names)

setRowNames(x, names)
```

Arguments

`x`	an object
`names`	column or row names to be assigned to the object

Examples

```
setColNames(ToothGrowth,c("length","supplement","dosage"))
setRowNames(BOD,BOD$Time)
```

---

smoothvect *Smooth a numeric vector using a moving window algorithm*

Description

Smooth a numeric vector using a moving window algorithm

Usage

```
smoothvect(vect, width = 2, both.sides = T, alg = c("mean", "gauss"))
```

Arguments

vect	
width	Over how many values should the vector be averaged?
both.sides	If TRUE (default), takes the mean of `width` values before and after the current index. If FALSE, only takes values ahead of the current index.

Value

Smoothed numeric vector

Examples

```
temp<- smoothvect(beaver1$temp)
plot(temp,type="l")
```

---

splitColumn *Split a character column into multiple values*

Description

Split a character column into multiple values

Usage

```
splitColumn(x, sep = ";")
```

Arguments

x	a character vector to split into columns
sep	a caracter separating the different values

Value

a `data.frame` of boolean values, with rows representing the unpacked vector entries and columns indicating whether the specific value

Examples

```
unsplit<-c("flour;salt;baking soda;steak;sugar;water;sauce;vinegar",
"flour;sauce;mustard;salt;pepper;vinegar;baking soda;water;tomatoes;onion;steak")
splitColumn(unsplit)
```

---

theme_pecher *Pecher theme for ggplot Based on the plot design style of prof. Diane Pecher.*

Description

Pecher theme for ggplot Based on the plot design style of prof. Diane Pecher.

Usage

```
theme_pecher()
```

Examples

```
ggplot(mtcars,aes(x=mpg,y=wt)) + geom_point() + theme_pecher()
```

---

tokens_compound_stepwise *Compound tokens without overflowing memory and crashing R*

Description

A wrapper around tokens_compound that processes your tokens in chunks, set by argument `stepsize`. See tokens_compound for more info.

Usage

```
tokens_compound_stepwise(
  x,
  pattern,
  stepsize = 100,
  concatenator = "_",
  valuetype = c("glob", "regex", "fixed"),
  case_insensitive = TRUE,
  join = TRUE
)
```

Examples

```
toks<-tokens(data_corpus_inaugural)
compounded<-tokens_compound_stepwise(x=toks,pattern="I am",stepsize=10)

#note: does not work?
```

TransformPlots *Data Transformation Plots*

Description

Visualize how different transformations of the data will fit to a normal distribution.

Usage

```
TransformPlots(x)
```

Arguments

x	A numeric vector.

Examples

```
TransformPlots(mtcars$disp)
```

trypackages *Install packages if neccesary, then load them.*

Description

Install packages if neccesary, then load them.

Usage

```
trypackages(...)
```

Arguments

...	Unquoted names of packages to try loading, and if unable, install and load.

Examples

```
trypackages(stats,utils,compiler)
```

verify_types *Verify variable types in bulk*

Description

Verify variable types in bulk

Usage

```
verify_types(...)
```

Arguments

...	Named arguments, where the argument is the object to be checked and the name of the argument is the mode (numeric, list, character, etc)

Value

Returns true on success, causes error if not.

Examples

```
try(verify_types(character="test",numeric=0000,character=12345))
```

wtd.median *Weighted Median*

Description

Weighted Median

Usage

```
wtd.median(x, wts, na.rm = T)
```

Arguments

x	an input vector
wts	a vector of weights
na.rm	Logical indicating whether NA values in the input and weight vectors should be stripped.

Value

A weighted median of the input values and weights.

Examples

```
wtd.median(1:5,c(.5,4,1,2,1))
```

Index