# Package 'skMisc'

October 1, 2019

**Title** Sercan Kahveci's Miscellaneous Functions

**Version** 0.01

**Description** Contains a wide range of functions.

**Depends** R (¿= 3.6.1)

**Imports** magrittr, dplyr, doParallel, lmerTest, knitr

**License** GPL (¿=2)

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

**LazyData** true

**RoxygenNote** 6.1.1

## R topics documented:

aat_doublemeandiff *AAT score computation algorithms*

## Description

- `aat_doublemeandiff` computes a mean-based double-difference score:
  `(mean(push_target) -mean(pull_target)) -(mean(push_control) -mean(pull_control))`
- `aat_doublemediandiff` computes a median-based double-difference score:
  `(median(push_target) -median(pull_target)) -(median(push_control) -median(pull_control))`
- `aat_dscore` computes D-scores (see Greenwald, Nosek, and Banaji, 2003):
  `((mean(push_target) -mean(pull_target)) -(mean(push_control) -mean(pull_control))) / sd(participant_reaction_times)`
- `aat_multilevelscore` fits a multilevel model using lme4 and extracts a random effect serving as AAT score. When using this function, additional arguments must be provided:
  - `formula` - a quoted formula to fit to the data;
  - `aatterm` the quoted random effect within the subject variable that indicates the approach bias; this is usually the interaction of the pull and target terms.

## Usage

```
aat_doublemeandiff(ds, subjvar, pullvar, targetvar, rtvar, ...)

aat_doublemediandiff(ds, subjvar, pullvar, targetvar, rtvar, ...)

aat_dscore(ds, subjvar, pullvar, targetvar, rtvar, ...)

aat_multilevelscore(ds, subjvar, formula, aatterm, ...)
```

## Arguments

| | |
|---|---|
| `ds` | A long-format data.frame |
| `subjvar` | Column name of the participant identifier variable |
| `pullvar` | Column name of the movement variable (0: avoid; 1: approach) |
| `targetvar` | Column name of the stimulus category variable (0: control stimulus; 1: target stimulus) |
| `rtvar` | Column name of the reaction time variable |
| `...` | Other arguments passed on by functions (ignored) |
| `formula` | A character string containing a formula to fit to the data and derive multilevel scores from |
| `aatterm` | The random term, grouped under the subject variable, which represents the approach bias. Usually this is the interaction of the pull and target terms. |

## Value

A data.frame containing participant number and computed AAT score.

`aat_splithalf` *Compute bootstrapped split-half reliability for approach-avoidance task data*

## Description

Compute bootstrapped split-half reliability for approach-avoidance task data. `aat_splithalf()` uses multicore computation, which is fast, but provides no clear output when there are errors. `aat_splithalf_singlecore()` is much slower, but more easily debugged.

## Usage

```
aat_splithalf(ds, subjvar, pullvar, targetvar, rtvar, iters, plot = T,
  algorithm = c(aat_doublemeandiff, aat_doublemediandiff, aat_dscore,
  aat_multilevelscore), trialdropfunc = c(prune_nothing,
  trial_prune_3SD), errortrialfunc = c(prune_nothing,
  error_replace_blockmeanplus), casedropfunc = c(prune_nothing,
  case_prune_3SD), ...)

aat_splithalf_singlecore(ds, subjvar, pullvar, targetvar, rtvar, iters,
  plot = F, algorithm = c(aat_doublemeandiff, aat_doublemediandiff,
  aat_dscore, aat_multilevelscore), trialdropfunc = c(prune_nothing,
  trial_prune_3SD), casedropfunc = c(prune_nothing, case_prune_3SD), ...)
```

## Arguments

| | |
|---|---|
| `ds` | a longformat data.frame |
| `subjvar` | Quoted name of the participant identifier column |
| `pullvar` | Quoted name of the column indicating pull trials. Pull trials should either be represented by 1, or by the second level of a factor. |
| `targetvar` | Name of the column indicating trials featuring the target stimulus. Target stimuli should either be represented by 1, or by the second level of a factor. |
| `rtvar` | Name of the reaction time column. |
| `iters` | Total number of desired iterations. At least 200 are recommended for reasonable confidence intervals; If you want to see plots of your data, 1 iteration is enough. |
| `plot` | Create a scatterplot of the AAT scores computed from each half of the data from the last iteration. This is highly recommended, as it helps to identify outliers that can inflate or diminish the reliability. |
| `algorithm` | Function (without brackets or quotes) to be used to compute AAT scores. See aat_doublemeandiff for a list of usable algorithms. |
| `trialdropfunc` | Function (without brackets or quotes) to be used to exclude outlying trials in each half. The way you handle outliers for the reliability computation should mimic the way you do it in your regular analyses. It is recommended to exclude outlying trials when computing AAT scores using the mean double-dfference scores and multilevel scoring approaches, but not when using d-scores or median double-difference scores. `prune_nothing` excludes no trials, while `trial_prune_3SD` excludes trials deviating more than 3SD from the mean per participant. |

errortrialfunc — Function (without brackets or quotes) to apply to an error trial.

error_replace_blockmeanplus replaces error trial reaction times with the block mean plus an arbitrary extra amount of time. If used, the following additional arguments are required:

- blockvar - Quoted name of the block variable
- errorvar - Quoted name of the error variable, where errors are 1 or TRUE and correct trials are 0 or FALSE
- errorbonus - Amount to add to the reaction time of error trials. Default is 0.6 (recommended by Greenwald,Nosek,& Banaji,2003)

casedropfunc — Function (without brackets or quotes) to be used to exclude outlying participant scores in each half. The way you handle outliers here should mimic the way you do it in your regular analyses. prune_nothing excludes no participants, while case_prune_3SD excludes participants deviating more than 3SD from the sample mean.

... — Other arguments, to be passed on to the algorithm functions (see algorithm above)

## Value

A list, containing the mean bootstrapped split-half reliability, bootstrapped 95 a list of data.frames used over each iteration, and a vector containing the split-half reliability of each iteration.

## Author(s)

Sercan Kahveci

## Examples

```
#Not Run
aat_splithalf(ds=ds2,subjvar="subjectid",pullvar="is_pull",targetvar="is_food",
              rtvar="rt",iters=1000,trialdropfunc=trial_prune_3SD,
              casedropfunc=case_prune_3SD,plot=T,algorithm=aat_dscore)
#Mean reliability: 0.521959
#Spearman-Brown-corrected r: 0.6859041
#95%CI: [0.4167018, 0.6172474]

#Multilevel Splithalf
aat_splithalf(ds=ds2,subjvar="subjectid",pullvar="is_pull",targetvar="is_food",
              rtvar="rt",iters=100,trialdropfunc=trial_prune_3SD,
              casedropfunc=case_prune_3SD,plot=T,algorithm=aat_multilevelscore,
              formula = "rt ~ is_pull * is_food + (is_pull * is_food | subjectid)",
              aatterm = "is_pull:is_food")
#Mean reliability: 0.5313939
#Spearman-Brown-corrected r: 0.6940003
#95%CI: [0.2687186, 0.6749176]
```

`AnovaTable` *Compare multilevel models*

## Description

Compare multilevel models

## Usage

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

## 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.

---

`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
```

## 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?"
```

---

## 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

---

## 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"
```

## 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

## 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`. |
| `df` | A data.frame |

### 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,"numeric","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"
```

---

## `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)
```

### 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")
```

---

`SpearmanBrown` *Correct a correlation coefficient for being based on only a subset of the data.*

---

### Description

Perform a Spearman-Brown correction on the provided correlation score.

### Usage

```
SpearmanBrown(corr, ntests = 2)
```

### Arguments

| | |
|---|---|
| `corr` | To-be-corrected correlation coefficient |
| `ntests` | An integer indicating how many times larger the full test is, for which the corrected correlation coefficient is being computed. When `ntests=2`, the formula will compute what the correlation coefficient would be if the test were twice as long. |

### Value

Spearman-Brown-corrected correlation coefficient. Values are bounded at zero.

---

`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)
```

# Index