Package 'AATtools'

June 6, 2021

Type Package

Title Reliability and Scoring Routines for the Approach-Avoidance Task

Version 0.0.2

Description Compute approach bias scores using different scoring algorithms, compute bootstrapped and exact split-half reliability estimates, and compute confidence intervals for individual participant scores.

Depends R (¿= 3.6.0)

Imports magrittr, dplyr, doParallel, foreach

License GPL-3

Encoding UTF-8

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

LazyData true

ByteCompile true

RoxygenNote 7.1.1

R topics documented:

aat_bootstrap *Compute bootstrapped approach-bias scores*

Description

Compute bootstrapped approach-bias scores with confidence intervals.

Usage

```
aat_bootstrap(
  ds,
  subjvar,
  pullvar,
  targetvar = NULL,
  rtvar,
  iters,
  algorithm = c("aat_doublemeandiff", "aat_doublemediandiff", "aat_dscore",
    "aat_dscore_multiblock", "aat_regression", "aat_standardregression",
    "aat_singlemeandiff", "aat_singlemediandiff"),
 trialdropfunc = c("prune_nothing", "trial_prune_3SD", "trial_prune_SD_dropcases",
  "trial_recode_SD", "trial_prune_percent_subject", "trial_prune_percent_sample"),
  errortrialfunc = c("prune_nothing", "error_replace_blockmeanplus",
    "error_prune_dropcases"),
  plot = TRUE,
  include.raw = FALSE,
  parallel = TRUE,
  ...
)

## S3 method for class 'aat_bootstrap'
print(x, ...)

## S3 method for class 'aat_bootstrap'
plot(x, ...)
```

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 required to get confidence intervals that make sense.
algorithm	Function (without brackets or quotes) to be used to compute AAT scores. See Algorithms 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 regression scoring approaches, but not when using d-scores or median double-difference scores.

- `prune_nothing` excludes no trials (default)
- `trial_prune_3SD` excludes trials deviating more than 3SD from the mean per participant.
- `trial_prune_SD_dropcases` removes trials deviating more than a specific number of standard deviations from the participant's mean, and removes participants with an excessive percentage of outliers. Required arguments:
  - `trialsd` - trials deviating more than `trialsd` standard deviations from the participant's mean are excluded (optional; default is 3)
  - `maxoutliers` - participants with a higher percentage of outliers are removed from the data. (optional; default is .15)
- `trial_recode_SD` recodes outlying reaction times to the nearest non-outlying value, with outliers defined as reaction times deviating more than a certain number of standard deviations from the participant's mean. Required argument:
  - `trialsd` - trials deviating more than this many standard deviations from the mean are classified as outliers.
- `trial_prune_percent_subject` and `trial_prune_percent_sample` remove trials below and/or above certain percentiles, on a subject-by-subject basis or sample-wide, respectively. The following arguments are available:
  - `lowerpercent` and `uppperpercent` (optional; defaults are .01 and .99).

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

- `prune_nothing` removes no errors (default).
- `error_replace_blockmeanplus` replaces error trial reaction times with the block mean, plus an arbitrary extra quantity. If used, the following additional arguments are required:
  - `blockvar` - Quoted name of the block variable (mandatory)
  - `errorvar` - Quoted name of the error variable, where errors are 1 or TRUE and correct trials are 0 or FALSE (mandatory)
  - `errorbonus` - Amount to add to the reaction time of error trials. Default is 0.6 (recommended by `Greenwald,Nosek,& Banaji,2003`)
- `error_prune_dropcases` removes errors and drops participants if they have more errors than a given percentage. The following arguments are available:
  - `errorvar` - Quoted name of the error variable, where errors are 1 or TRUE and correct trials are 0 or FALSE (mandatory)
  - `maxerrors` - participants with a higher percentage of errors are excluded from the dataset. Default is .15.

`plot` Plot the bias scores and their confidence intervals after computation is complete. This gives a good overview of the data.

include.raw	logical indicating whether raw split-half data should be included in the output object.
parallel	If TRUE (default), will use parallel computing to compute results faster. If a doParallel backend has not been registered beforehand, this function will register a cluster and stop it after finishing, which takes some extra time.
...	Other arguments, to be passed on to the algorithm or outlier rejection functions (see arguments above)
x	An aat_bootstrap object.

Value

A list, containing bootstrapped bias scores, their variance, bootstrapped 95 percent confidence intervals, the number of iterations, and a matrix of bias scores for each iteration.

Author(s)

Sercan Kahveci

Examples

```
# Compute 10 bootstrapped AAT scores.
boot<-aat_bootstrap(ds=erotica[erotica$is_irrelevant==0,], subjvar="subject",
                    pullvar="is_pull", targetvar="is_target",rtvar="RT",
                    iters=10,algorithm="aat_doublemediandiff",
                    trialdropfunc="trial_prune_3SD",
                    plot=FALSE, parallel=FALSE)
plot(boot)
print(boot)
```

---

aat_compute *Compute simple AAT scores*

Description

Compute simple AAT scores, with optional outlier exclusion and error trial recoding.

Usage

```
aat_compute(
  ds,
  subjvar,
  pullvar,
  targetvar = NULL,
  rtvar,
  algorithm = c("aat_doublemeandiff", "aat_doublemediandiff", "aat_dscore",
    "aat_dscore_multiblock", "aat_regression", "aat_standardregression",
    "aat_doublemeanquotient", "aat_doublemedianquotient", "aat_singlemeandiff",
    "aat_singlemediandiff"),
 trialdropfunc = c("prune_nothing", "trial_prune_3SD", "trial_prune_SD_dropcases",
  "trial_recode_SD", "trial_prune_percent_subject", "trial_prune_percent_sample"),
```

```
  errortrialfunc = c("prune_nothing", "error_replace_blockmeanplus",
    "error_prune_dropcases"),
  ...
)
```

Arguments

`ds`	a long-format data.frame
`subjvar`	column name of subject variable
`pullvar`	column name of pull/push indicator variable, must be numeric or logical (where pull is 1 or TRUE)
`targetvar`	column name of target stimulus indicator, must be numeric or logical (where target is 1 or TRUE)
`rtvar`	column name of reaction time variable
`algorithm`	Function (without brackets or quotes) to be used to compute AAT scores. See Algorithms 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 regression scoring approaches, but not when using d-scores or median double-difference scores. • `prune_nothing` excludes no trials (default) • `trial_prune_3SD` excludes trials deviating more than 3SD from the mean per participant. • `trial_prune_SD_dropcases` removes trials deviating more than a specific number of standard deviations from the participant's mean, and removes participants with an excessive percentage of outliers. Required arguments: – `trialsd` - trials deviating more than `trialsd` standard deviations from the participant's mean are excluded (optional; default is 3) – `maxoutliers` - participants with a higher percentage of outliers are removed from the data. (optional; default is .15) • `trial_recode_SD` recodes outlying reaction times to the nearest non-outlying value, with outliers defined as reaction times deviating more than a certain number of standard deviations from the participant's mean. Required argument: – `trialsd` - trials deviating more than this many standard deviations from the mean are classified as outliers. • `trial_prune_percent_subject` and `trial_prune_percent_sample` remove trials below and/or above certain percentiles, on a subject-by-subject basis or sample-wide, respectively. The following arguments are available: – `lowerpercent` and `uppperpercent` (optional; defaults are .01 and .99).
`errortrialfunc`	Function (without brackets or quotes) to apply to an error trial. • `prune_nothing` removes no errors (default). • `error_replace_blockmeanplus` replaces error trial reaction times with the block mean, plus an arbitrary extra quantity. If used, the following additional arguments are required:

  - blockvar - Quoted name of the block variable (mandatory)
  - errorvar - Quoted name of the error variable, where errors are 1 or TRUE and correct trials are 0 or FALSE (mandatory)
  - errorbonus - Amount to add to the reaction time of error trials. Default is 0.6 (recommended by Greenwald,Nosek,& Banaji,2003)
- error_prune_dropcases removes errors and drops participants if they have more errors than a given percentage. The following arguments are available:
  - errorvar - Quoted name of the error variable, where errors are 1 or TRUE and correct trials are 0 or FALSE (mandatory)
  - maxerrors - participants with a higher percentage of errors are excluded from the dataset. Default is .15.

... Other arguments, to be passed on to the algorithm or outlier rejection functions (see arguments above)

Examples

```
#Compute the correlation between relevant-feature and irrelevant-feature AAT scores
ds<-erotica[erotica$correct==1,]
relevant <- aat_compute(ds=ds[ds$is_irrelevant==0,],
                        pullvar="is_pull",targetvar="is_target",
                        rtvar="RT",subjvar="subject",
                        trialdropfunc="trial_prune_3SD",
                        algorithm="aat_doublemediandiff")

irrelevant <- aat_compute(ds=ds[ds$is_irrelevant==1,],
                        pullvar="is_pull",targetvar="is_target",
                        rtvar="RT",subjvar="subject",
                        trialdropfunc="trial_prune_3SD",
                        algorithm="aat_doublemediandiff")

comparison.df <- merge(relevant, irrelevant, by = "subject")
cor(comparison.df$ab.x, comparison.df$ab.y)
# 0.1145726
```

---

aat_covreliability *Compute a dataset's reliability from its covariance matrix*

Description

This function computes mean single-difference scores (push minus pull) for individual stimuli, and computes the reliability from that information. Missing values are dealt with using multiple imputation.

Usage

```
aat_covreliability(
  ds,
  subjvar,
  stimvar,
  pullvar,
  targetvar = NULL,
```

```
  rtvar,
  algorithm = c("calpha", "lambda2", "lambda4"),
  iters = 5
)
```

Arguments

ds	the data.frame to use
subjvar	Name of the subject-identifying variable
stimvar	Name of the stimulus-identifying variable
pullvar	Name of the movement-direction identifying variable
targetvar	Optional. Name of the stimulus-category identifying variable
rtvar	Name of the reaction-time identifying variable
algorithm	The reliability formula to use. Defaults to Cronbach's alpha, but Guttman's Lambda-2 is recommended instead.
iters	If there are missing values (which is almost inevitable) then multiple imputation will be used to complete the covariance matrix - this option sets the number of multiple imputations to be used.

Details

When only one stimulus category is indicated, one of the commonly known reliability algorithms provided with the algorithm argument is used. When two stimulus categories are indicated, this function uses Lord's (1963) algorithm to compute the reliability of a double mean difference score, using the algorithms in algorithm to estimate the reliability of indiviau lstimulus categories.

When one wants to compute the reliability of a double median difference score or D-score, aat_splithalf() is recommended instead.

Value

Returns an aat_covreliability object containing the reliability value as well as the dataset and covariance matrix with replaced missing values. When the argument targetvar is provided, the output also contains the reliability of the individual stimulus categories and their intercorrelation.

References

Lord, F.Y. (1963), "Elementary Models for Measuring Change", in Problems in Measuring Change, C.W. Harris, ed.. Madison. Wisconsin: University of Wisconsin.

Examples

```
#We generate a dataset with 16 stimuli in each category
ds<-aat_simulate(biasfx_jitter=40,nstims=16)
ds$stim<-paste0(ds$stim,"-",ds$is_target)

# If Lord's formula and
# bootstrapped splithalf measure something similar,
# then the outcomes should be close to each other.
aat_covreliability(ds=ds,subjvar="subj",stimvar="stim",pullvar="is_pull",
                           targetvar="is_target",rtvar="rt")
```

```
aat_splithalf(ds=ds,subjvar="subj",pullvar="is_pull",targetvar="is_target",rtvar="rt",
              algorithm="aat_doublemeandiff",iters=100,plot=FALSE)

#Testing reliability for single-difference scores
ds<-ds[ds$is_target==1,]
aat_covreliability(ds=ds,subjvar="subj",stimvar="stim",pullvar="is_pull",rtvar="rt")
```

`aat_simulate` *Simulate AAT datasets and predict parameters*

Description

`aat_simulate()` generates approach-avoidance task datasets.

Usage

```
aat_simulate(
  npps = 40,
  nstims = 32,
  stimreps = 2,
  meanrt = 743,
  meanrt_jitter = 66,
  sdrt = 133,
  sdrt_jitter = 38,
  pullfx = 25,
  pullfx_jitter = 40,
  stimfx = 10,
  stimfx_jitter = 35,
  biasfx = 35,
  biasfx_jitter = 75,
  empirical = FALSE
)

aat_simulate2(
  ...,
  defaults = c("none", "Lender2018_raw", "Lender2018_clean", "Kahveci2021_raw",
    "Kahveci2021_clean"),
  slowols = 0,
  fastols = 0,
  olsd = 3
)
```

Arguments

`npps`	Number of participants
`nstims`	Number of stimuli
`stimreps`	Number of repetitions of each stimulus within each group (i.e. within approach target, avoid target, approach control, avoid control)
`meanrt`	Mean sample reaction time
`meanrt_jitter`	Extent by which participants' mean RTs deviate from mean sample RT.

sdrt	Standard deviation of samplewide RTs, ignoring effects of movement, stimulus, and approach bias. In essence, this represents the amount of pure noise present in the data.
sdrt_jitter	Extent by which standard deviations of individual participants' RTs are larger or smaller than the samplewide SD.
pullfx	size of the effect of approach-versus-avoidance, in milliseconds
pullfx_jitter	Individual variation in the effect of approach-versus-avoidance
stimfx	size of the effect of stimulus category, in milliseconds
stimfx_jitter	Individual variation in the effect of stimulus category
biasfx	Size of the approach bias effect, in milliseconds
biasfx_jitter	Individual variation in the approach bias effect
empirical	If TRUE, then effect sizes and standard deviations will be exact
...	Any parameters of aat_simulate provided here will override the defaults from the defaults parameter.
defaults	Which set of default values should be used?
slowols	Number of slow outliers to insert per participant
fastols	Number of fats outliers to insert per participant
olsd	Number of standard deviations by which (slow) outliers deviate

Details

Defaults of aat_simulate() are based on Kahveci, Van Alebeek, Berking, & Blechert (2021).

"Lender2018" parameters are taken from the relevant-feature AAT of Lender, Meule, Rinck, Brockmeyer, & Blechert (2018). "Kahveci2021" parameters are taken from Kahveci, Van Alebeek, Berking, & Blechert (in review).

Lender, A., Meule, A., Rinck, M., Brockmeyer, T., & Blechert, J. (2018). Measurement of food-related approach–avoidance biases: Larger biases when food stimuli are task relevant. Appetite, 125, 42-47.

Kahveci, S., Van Alebeek, H., Berking, M., & Blechert, J. (in review). Touchscreen based assessment of food approach biases: investigation of reliability and stimulus-specific effects.

Value

aat_simulate() returns a data.frame with the following columns: subj (participant ID), stim (stimulus number), rep (stimulus repetition number), is_pull (0 = avoid, 1 = approach), is_target (0 = control stimulus, 1 = target stimulus), meanrt (participant's mean RT), sdrt (participant's residual standard deviation), pullfx (participant approach-avoidance effect size in ms), stimfx (participant stimulus category effect size in ms), biasfx (participant approach bias effect size in ms), and rt (trial reaction time). Additionally, the data.frame has the attribute population_reliability which represents the expected reliability of the data given the provided parameters.

Examples

```
ts<- aat_simulate(pullfx = 50, stimfx = 10, biasfx = 100)
mod<-lm(rt~is_pull*is_target,data=ts)
coef(mod) #these should be somewhat close to the provided coefficients
print(q_reliability(ts,"subj",rt~is_pull*is_target,"is_pull:is_target"))
#these two should be not too far apart,
```

```
# and should converge when the process is repeated a bunch


# Here's how to derive the parameters used in this function from a real dataset
## Not run:
mod<-lmer(decisiontime ~ is_pull * is_food + (is_pull * is_food | subjectid),data=dsa)
fixef(mod) # from here, all the fx and mean RTs are derived
ranef(mod)$subjectid %>% apply(2,sd) #from here, all the fx jitters are derived
dsa %>% group_by(subjectid) %>% summarise(sd=sd(resid)) %>%
summarise(m=mean(sd),s=sd(sd)) # from here, sdrt_jitter is derived

## End(Not run)
hist(aat_simulate2(defaults="Lender2018_raw",slowols=10,fastols=10)$rt)
```

---

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

Description

Compute bootstrapped split-half reliability for approach-avoidance task data.

Usage

```
aat_splithalf(
  ds,
  subjvar,
  pullvar,
  targetvar = NULL,
  rtvar,
  iters,
  algorithm = c("aat_doublemeandiff", "aat_doublemediandiff", "aat_dscore",
    "aat_dscore_multiblock", "aat_regression", "aat_standardregression",
    "aat_singlemeandiff", "aat_singlemediandiff"),
 trialdropfunc = c("prune_nothing", "trial_prune_3SD", "trial_prune_SD_dropcases",
  "trial_recode_SD", "trial_prune_percent_subject", "trial_prune_percent_sample"),
  errortrialfunc = c("prune_nothing", "error_replace_blockmeanplus",
    "error_prune_dropcases"),
  casedropfunc = c("prune_nothing", "case_prune_3SD"),
  plot = TRUE,
  include.raw = FALSE,
  parallel = TRUE,
  ...
)

## S3 method for class 'aat_splithalf'
print(x, coef = c("Raju", "FlanaganRulon", "SpearmanBrown"), ...)

## S3 method for class 'aat_splithalf'
plot(x, type = c("median", "minimum", "maximum", "random"), ...)
```

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.
algorithm	Function (without brackets or quotes) to be used to compute AAT scores. See Algorithms 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 regression scoring approaches, but not when using d-scores or median double-difference scores. • prune_nothing excludes no trials (default) • trial_prune_3SD excludes trials deviating more than 3SD from the mean per participant. • trial_prune_SD_dropcases removes trials deviating more than a specific number of standard deviations from the participant's mean, and removes participants with an excessive percentage of outliers. Required arguments: – trialsd - trials deviating more than trialsd standard deviations from the participant's mean are excluded (optional; default is 3) – maxoutliers - participants with a higher percentage of outliers are removed from the data. (optional; default is .15) • trial_recode_SD recodes outlying reaction times to the nearest non-outlying value, with outliers defined as reaction times deviating more than a certain number of standard deviations from the participant's mean. Required argument: – trialsd - trials deviating more than this many standard deviations from the mean are classified as outliers. • trial_prune_percent_subject and trial_prune_percent_sample remove trials below and/or above certain percentiles, on a subject-by-subject basis or sample-wide, respectively. The following arguments are available: – lowerpercent and uppperpercent (optional; defaults are .01 and .99).
errortrialfunc	Function (without brackets or quotes) to apply to an error trial. • prune_nothing removes no errors (default). • error_replace_blockmeanplus replaces error trial reaction times with the block mean, plus an arbitrary extra quantity. If used, the following additional arguments are required:

  - `blockvar` - Quoted name of the block variable (mandatory)
  - `errorvar` - Quoted name of the error variable, where errors are 1 or TRUE and correct trials are 0 or FALSE (mandatory)
  - `errorbonus` - Amount to add to the reaction time of error trials. Default is 0.6 (recommended by `Greenwald,Nosek,& Banaji,2003`)
- `error_prune_dropcases` removes errors and drops participants if they have more errors than a given percentage. The following arguments are available:
  - `errorvar` - Quoted name of the error variable, where errors are 1 or TRUE and correct trials are 0 or FALSE (mandatory)
  - `maxerrors` - participants with a higher percentage of errors are excluded from the dataset. Default is .15.

`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 (default)
- `case_prune_3SD` excludes participants deviating more than 3SD from the sample mean.

`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.

`include.raw` logical indicating whether raw split-half data should be included in the output object.

`parallel` If TRUE (default), will use parallel computing to compute results faster. If a doParallel backend has not been registered beforehand, this function will register a cluster and stop it after finishing, which takes some extra time.

`...` Other arguments, to be passed on to the algorithm or outlier rejection functions (see arguments above)

`x` an `aat_splithalf` object

`coef` Optional character argument, indicating which reliability coefficient should be printed. Defaults to Raju's beta.

`type` Character argument indicating which iteration should be chosen. Must be an abbreviation of `"median"` (default), `"minimum"`, `"maximum"`, or `"random"`.

Details

The calculated split-half coefficients are described in Warrens (2016).

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

References

Warrens, M. J. (2016). A comparison of reliability coefficients for psychometric tests that consist of two parts. Advances in Data Analysis and Classification, 10(1), 71-84.

See Also

q_reliability

Examples

```
split <- aat_splithalf(ds=erotica[erotica$is_irrelevant==0,],
                       subjvar="subject",pullvar="is_pull",targetvar="is_target",
                       rtvar="RT",iters=10,trialdropfunc="trial_prune_3SD",
                       casedropfunc="case_prune_3SD",algorithm="aat_dscore",
                       plot=FALSE,parallel=FALSE)

print(split)
#Mean reliability: 0.521959
#Spearman-Brown-corrected r: 0.6859041
#95%CI: [0.4167018, 0.6172474]

plot(split)


#Regression Splithalf
aat_splithalf(ds=erotica[erotica$is_irrelevant==0,],
              subjvar="subject", pullvar="is_pull", targetvar="is_target",
              rtvar="RT", iters=10, trialdropfunc="trial_prune_3SD",
              casedropfunc="case_prune_3SD", algorithm="aat_regression",
              formula = RT ~ is_pull * is_target, aatterm = "is_pull:is_target",
              plot=FALSE, parallel=FALSE)
#Mean reliability: 0.5313939
#Spearman-Brown-corrected r: 0.6940003
#95%CI: [0.2687186, 0.6749176]
```

---

aat_stimulusscores *Compute stimulus-specific bias scores Computes mean single-difference scores (push - pull) for each stimulus.*

Description

Compute stimulus-specific bias scores Computes mean single-difference scores (push - pull) for each stimulus.

Usage

```
aat_stimulusscores(
  ds,
  subjvar,
  stimvar,
  pullvar,
  targetvar = NULL,
```

```
  rtvar,
  iters = 5
)
```

Arguments

ds	the data.frame to use
subjvar	Name of the subject-identifying variable
stimvar	Name of the stimulus-identifying variable
pullvar	Name of the movement-direction identifying variable
targetvar	Optional. Name of the stimulus-category identifying variable
rtvar	Name of the reaction-time identifying variable
iters	If there are missing values (which is almost inevitable) then multiple imputation will be used to complete the covariance matrix - this argument sets the number of multiple imputations to be used.

Value

Exports a list containing a data.frame with stimulus-specific bias scores, indicated in the column names, a covariance matrix of that same data, and a data.frame indicating to which stimulus category each stimulus belongs.

Examples

```
ds<-aat_simulate(biasfx_jitter=40,nstims=16)
ds$stim<-paste0(ds$stim,"-",ds$is_target)
aat_stimulusscores(ds,"subj","stim","is_pull","is_target","rt")
```

---

aat_stimulus_rest

Compute stimulus-rest correlations of double-difference scores This function provides a statistic that can give an indication of how deviant the responses to specific stimuli are, in comparison to the rest of the stimulus set. The algorithm computes stimulus-rest correlations of stimulus-specific double-difference scores. It takes single-difference approach-avoidance scores for each stimulus, and computes every possible subtraction between individual stimuli from both stimulus categories. It then computes correlations between every such subtraction of stimuli on one hand, and the mean double difference score of all other stimuli. Stimulus-rest correlations are then computed by averaging every such subtraction-rest correlation involving a specific stimulus.

Description

Compute stimulus-rest correlations of double-difference scores This function provides a statistic that can give an indication of how deviant the responses to specific stimuli are, in comparison to the rest of the stimulus set. The algorithm computes stimulus-rest correlations of stimulus-specific double-difference scores. It takes single-difference approach-avoidance scores for each stimulus, and computes every possible subtraction between individual stimuli from both stimulus categories. It then computes correlations between every

such subtraction of stimuli on one hand, and the mean double difference score of all other stimuli. Stimulus-rest correlations are then computed by averaging every such subtraction-rest correlation involving a specific stimulus.

Usage

```
aat_stimulus_rest(ds, subjvar, stimvar, pullvar, targetvar, rtvar)

## S3 method for class 'aat_stimulus_rest'
plot(x, ...)
```

Arguments

ds	a data.frame
subjvar	the label of the participant identifier variable
stimvar	the label of the stimulus identifier variable
pullvar	the label of the movement direction identifier variable
targetvar	the label of the stimulus category identifier variable
rtvar	the label of the reaction time variable
x	an aat_stimulus_rest object
...	Ignored.

Value

Returns a aat_stimulus_rest object containing statistics for each stimulus. Stats include the average stimulus-rest correlation (mcor); the standard deviation of dyad-rest correlations for this stimulus (sdcor); the number of valid correlations involved in these statistic (n); the average percentile of dyad-rest correlations involving the stimulus within the distribution of all other dyad-rest correlations (restpercentile); as well as z-scores (zpercentile) and p-values for this percentile (pval).

Examples

```
ds<-aat_simulate()
stimrest<-aat_stimulus_rest(ds,subjvar="subj",stimvar="stim",pullvar="is_pull",
                    targetvar="is_target",rtvar="rt")
plot(stimrest)
print(stimrest)
```

Algorithms *AAT score computation algorithms*

Description

AAT score computation algorithms

Usage

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

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

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

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

aat_dscore_multiblock(ds, subjvar, pullvar, targetvar, rtvar, blockvar, ...)

aat_regression(ds, subjvar, formula, aatterm, ...)

aat_standardregression(ds, subjvar, formula, aatterm, ...)

aat_singlemeandiff(ds, subjvar, pullvar, rtvar, ...)

aat_singlemediandiff(ds, subjvar, pullvar, rtvar, ...)
```

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)
blockvar	name of the variable indicating block number
formula	A regression formula to fit to the data to compute an AAT score
aatterm	A character naming the formula term representing the approach bias. Usually this is the interaction of the movement-direction and stimulus-category terms.

Value

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

Functions

- `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 for a 2-block design (see Greenwald, Nosek, and Banaji, 2003): `((mean(push_target) -mean(pull_target)) -(mean(push_control) -mean(pull_control))) / sd(participant_reaction_times)`
- `aat_mediandscore`: computes a double-difference score usign medians, and divides it by the median absolute deviation of the participant's overall reaction times: `((median(push_target) -median(pull_target)) -(median(push_control) -median(pull_control))) / mad(participant_reactio`

- `aat_dscore_multiblock`: computes D-scores for pairs of sequential blocks and averages the resulting score (see Greenwald, Nosek, and Banaji, 2003). Requires extra `blockvar` argument, indicating the name of the block variable.
- `aat_regression`: `aat_regression` and `aat_standardregression` fit regression models to participants' reaction times and extract a term that serves as AAT score. `aat_regression` extracts the raw coefficient, equivalent to a mean difference score. `aat_standardregression` extracts the t-score of the coefficient, standardized on the basis of the variability of the participant's reaction times. These algorithms can be used to regress nuisance variables out of the data before computing AAT scores. When using these functions, additional arguments must be provided:
  - `formula` - a formula to fit to the data
  - `aatterm` - the term within the formula that indicates the approach bias; this is usually the interaction of the pull and target terms.
- `aat_standardregression`: See above
- `aat_singlemeandiff`: subtracts the mean approach reaction time from the mean avoidance reaction time. Using this algorithm is only sensible if the supplied data contain a single stimulus category.
- `aat_singlemediandiff`: subtracts the median approach reaction time from the median avoidance reaction time. Using this algorithm is only sensible if the supplied data contain a single stimulus category.

`cormean` *Compute a minimally biased average of correlation values*

Description

This function computes a minimally biased average of correlation values. This is needed because simple averaging of correlations is negatively biased, and the often used z-transformation method of averaging correlations is positively biased. The algorithm was developed by Olkin & Pratt (1958) and implemented by Jan Seifert.

Usage

```
cormean(r, n, na.rm = F)
```

Arguments

`r`	a vector containing correlation values
`n`	a single value or vector containing sample sizes
`na.rm`	Logical. Should missing values be removed?

Value

An average correlation.

References

Olkin, I., & Pratt, J. (1958). Unbiased estimation of certain correlation coefficients. The Annals of Mathematical Statistics, 29. https://doi.org/10.1214/aoms/1177706717

https://github.com/SigurdJanson/AveragingCorrelations/blob/master/CorrAggBias.R

https://medium.com/@jan.seifert/averaging-correlations-part-ii-9143b546860b

Examples

```
cormean(c(0,.3,.5),c(30,30,60))
```

`correlation-tools` *Correlation tools*

Description

Helper functions to compute important statistics from correlation coefficients.

Usage

```
r2z(r)

z2r(z)

r2t(r, n)

r2p(r, n)

rconfint(r, n, alpha = 0.05)

compcorr(r1, r2, n1, n2)
```

Arguments

`r, r1, r2`	a correlation value
`z`	a Z-score
`n, n1, n2`	sample sizes
`alpha`	the significance level to use

Functions

- `r2z`: converts correlation coefficients to z-scores
- `z2r`: converts z-scores to correlation coefficients
- `r2t`: Converts correlation coefficients to t-scores
- `r2p`: Computes the two-sided p-value for a given correlation
- `rconfint`: Computes confidence intervals for a given correlation coefficient
- `compcorr`: computes the significance of the difference between two correlation coefficients

See Also

cormean, multiple.cor, partial.cor

Examples

```
z <- r2z(.5)
r <- z2r(z)
t<-r2t(r,30)
r2p(r,30)
print(rconfint(r,30))
print(compcorr(.5,.7,20,20))
```

`covEM` *Covariance matrix computation with multiple imputation*

Description

This function computes a covariance matrix from data with some values missing at random.
The code was written by Eric from StackExchange. https://stats.stackexchange.com/questions/182718/ml-covariance-estimation-from-expectation-maximization-with-missing-data

Usage

```
covEM(dat_missing, iters = 1000)
```

Arguments

`dat_missing`	a matrix with missing values
`iters`	the number of iterations to perform to estimate missing values

References

Beale, E. M. L., & Little, R. J. A.. (1975). Missing Values in Multivariate Analysis. Journal of the Royal Statistical Society. Series B (methodological), 37(1), 129–145.

Examples

```
# make data with missing values
missing_mtcars <- mtcars
for(i in 1:20){
  missing_mtcars[sample(1:nrow(mtcars),1),sample(1:ncol(mtcars),1)]<-NA
}
covmat<-covEM(as.matrix(missing_mtcars))$sigma
calpha(covmat)
```

---

covrel *Covariance Matrix-Based Reliability Coefficients*

Description

These functions allow for the computation of the reliability of a dataset from the covariance matrix of its variables.

Usage

```
calpha(covmat)

lambda2(covmat)

lambda4(covmat)
```

Arguments

covmat	a covariance matrix

Functions

- calpha: Cronbach's alpha
- lambda2: Guttman's Lambda-2
- lambda4: Guttman's Lambda-4. This algorithm tries to get the highest attainable reliability by

See Also

splitrel

Examples

```
# compute reliability from covariance
h<-cov(iris[,1:4])
calpha(h)
lambda2(h)
lambda4(h)
# Lambda-2 and Lambda-4 are significantly larger because
# some of the variables in the iris dataset are negatively correlated.
```

---

erotica *AAT examining approach bias for erotic stimuli*

Description

AAT

Usage

```
erotica
```

Format

An object of class "data.frame"

Source

osf.io repository

References

Kahveci, S., Van Bockstaele, B.D., & Wiers, R.W. (in preparation). Pulling for Pleasure? Erotic Approach-Bias Associated With Porn Use, Not Problems. DOI:10.17605/OSF.IO/6H2RJ

`multiple.cor` *Multiple correlation Computes the* R*hrefhttps://en.wikipedia.org/wiki/Multiple_correlationmultiple correlation coefficient of variables in* `ymat` *with the variable* `x`

Description

Multiple correlation Computes the multiple correlation coefficient of variables in `ymat` with the variable `x`

Usage

```
multiple.cor(x, ymat, use = "everything")
```

Arguments

`x`	Either a matrix of variables whose multiple correlation with each other is to be estimated; or a vector of which the multiple correlation with variables in `ymat` is to be estimated
`ymat`	a matrix or data.frame of variables of which the multiple correlation with `x` is to be estimated
`use`	optional character indicating how to handle missing values (see cor)

Value

The multiple correlation coefficient

See Also

https://www.personality-project.org/r/book/chapter5.pdf

Examples

```
multiple.cor(mtcars[,1],mtcars[,2:4])
```

`partial.cor` *Partial correlation Compute the correlation between x and y while controlling for z.*

Description

Partial correlation Compute the correlation between x and y while controlling for z.

Usage

```
partial.cor(x, y, z, use = c("complete.obs", "everything"))
```

Arguments

`x, y, z`	x and y will be correlated while controlling for z
`use`	optional character indicating how to handle missing values (see cor)

Examples

```
partial.cor(mtcars$mpg,mtcars$cyl,mtcars$disp)
```

`Preprocessing` *Pre-processing rules*

Description

These are pre-processing rules that can be used in aat_splithalf, aat_bootstrap, and aat_compute.

- The following rules are to be used for the `trialdropfunc` argument. The way you handle outliers for the reliability computation and bootstrapping more broadly 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 regression scoring approaches, but not when using d-scores or median double-difference scores.
  - `prune_nothing` excludes no trials (default)
  - `trial_prune_3SD` excludes trials deviating more than 3SD from the mean per participant.
  - `trial_prune_SD_dropcases` removes trials deviating more than a specific number of standard deviations from the participant's mean, and removes participants with an excessive percentage of outliers. Required arguments:
    - `trialsd` - trials deviating more than `trialsd` standard deviations from the participant's mean are excluded (optional; default is 3)
    - `maxoutliers` - participants with a higher percentage of outliers are removed from the data. (optional; default is .15)
  - `trial_recode_SD` recodes outlying reaction times to the nearest non-outlying value, with outliers defined as reaction times deviating more than a certain number of standard deviations from the participant's mean. Required argument:
    - `trialsd` - trials deviating more than this many standard deviations from the mean are classified as outliers.

  - `trial_prune_percent_subject` and `trial_prune_percent_sample` remove trials below and/or above certain percentiles, on a subject-by-subject basis or sample-wide, respectively. The following arguments are available:
    * `lowerpercent` and `uppperpercent` (optional; defaults are .01 and .99).

- The following pre-procesing rules are to be used for the `errortrialfunc` argument. They determine what is to be done with errors - remove or recode?
  - `prune_nothing` removes no errors (default).
  - `error_replace_blockmeanplus` replaces error trial reaction times with the block mean, plus an arbitrary extra quantity. If used, the following additional arguments are required:
    * `blockvar` - Quoted name of the block variable (mandatory)
    * `errorvar` - Quoted name of the error variable, where errors are 1 or TRUE and correct trials are 0 or FALSE (mandatory)
    * `errorbonus` - Amount to add to the reaction time of error trials. Default is 0.6 (recommended by `Greenwald,Nosek,& Banaji,2003`)
  - `error_prune_dropcases` removes errors and drops participants if they have more errors than a given percentage. The following arguments are available:
    * `errorvar` - Quoted name of the error variable, where errors are 1 or TRUE and correct trials are 0 or FALSE (mandatory)
    * `maxerrors` - participants with a higher percentage of errors are excluded from the dataset. Default is .15.

- These are pre-processing rules to be used for the `casedropfunc` argument. The way you handle outliers here should mimic the way you do it in your regular analyses.
  - `prune_nothing` excludes no participants (default)
  - `case_prune_3SD` excludes participants deviating more than 3SD from the sample mean.

Usage

```
prune_nothing(ds, ...)

trial_prune_percent_subject(
  ds,
  subjvar,
  rtvar,
  lowerpercent = 0.01,
  upperpercent = 0.99,
  ...
)

trial_prune_percent_sample(
  ds,
  rtvar,
  lowerpercent = 0.01,
  upperpercent = 0.99,
  ...
)

trial_prune_3SD(ds, subjvar, rtvar, ...)
```

```
trial_prune_SD_dropcases(
  ds,
  subjvar,
  rtvar,
  trialsd = 3,
  maxoutliers = 0.15,
  ...
)

trial_recode_SD(ds, subjvar, rtvar, trialsd = 3, ...)

case_prune_3SD(ds, ...)

error_replace_blockmeanplus(
  ds,
  subjvar,
  rtvar,
  blockvar,
  errorvar,
  errorbonus,
  ...
)

error_prune_dropcases(ds, subjvar, errorvar, maxerrors = 0.15, ...)
```

Arguments

`ds`	A data.frame.
`...`	Other arguments (ignored).
`subjvar`	The name of the subject variable.
`rtvar`	The name of the reaction time variable.
`lowerpercent, upperpercent`	for `trial_prune_percent_subject` and `trial_prune_percent_sample`, the lower and upper proportions beyond which trials are considered outliers and removed (defaults to .01 and .99).
`trialsd`	The amount of deviation from the participant mean (in SD) after which a trial is considered an outlier and excluded (defaults to 3).
`maxoutliers`	for `trial_prune_SD_dropcases`, the maximum percentage of outliers, after which a participant is excluded from the data.
`blockvar`	The name of the block variable.
`errorvar`	The name of the error variable.
`errorbonus`	for `error_replace_blockmeanplus`, the amount of seconds to add to the block mean and use as a replacement for error trial reaction times (default is 0.6).
`maxerrors`	for `error_prune_dropcases`, the maximum percentage of errors, after which a participant is excluded from the data.

`q_reliability` *Compute psychological experiment reliability*

Description

This function can be used to compute an exact reliability score for a psychological task whose results involve a difference score. The resulting intraclass correlation coefficient is equivalent to the average all possible split-half reliability scores. It ranges from -1 to 1, with -1 implying that all variance in the data is explained by within-subjects variability, 1 implying that all variance is explained by between-subjects variability, and 0 implying that within-subjects and between-subjects variability contribute equally to the total variance in the sample.

Usage

```
q_reliability(ds, subjvar, formula, aatterm = NA)

q_reliability2(ds, subjvar, splitvars, rtvar, na.rm = F)

## S3 method for class 'qreliability'
print(x, ...)

## S3 method for class 'qreliability'
plot(x, ...)
```

Arguments

`ds`	a long-format data.frame
`subjvar`	name of the subject variable
`formula`	a formula predicting the participant's reaction time using trial-level variables such as movement direction and stimulus category
`aatterm`	a string denoting the term in the formula that contains the participant's approach bias
`splitvars`	Vector of column names over which to split the data to compute difference scores. This can be used to compute the reliability of single, double, or even triple difference scores.
`rtvar`	Column name of the variable containing reaction times
`na.rm`	If true, remove rows with missing values from the data
`x`	a `qreliability` object
`...`	Other arguments passed to the generic `print` and `plot` functions.

Value

a qreliability object, containing the reliability coefficient, and a data.frame with participants' bias scores and score variance.

Please note that the valence of the bias scores may or may not correspond with approach and avoidance. If you plan to use these scores in your analyses, always verify that they are in the right direction by correlating them with independently calculated bias scores, for example using `aat_compute()`.

Author(s)

Sercan Kahveci

Examples

```
# Double-difference score reliability
q_reliability(ds=erotica,subjvar="subject",
              formula= RT ~ is_pull * is_target, aatterm = "is_pull:is_target")

# Single-difference reliability for target stimuli
q_reliability(ds=erotica[erotica$is_target ==1,],subjvar="subject",
              formula= RT ~ is_pull, aatterm = "is_pull")

# Reliability of the mean reaction time of approaching target stimuli (no difference score)
q_reliability(ds=erotica[erotica$is_target ==1 & erotica$is_pull ==1,],subjvar="subject",
              formula= RT ~ 1, aatterm = "1")

q_reliability2(ds=erotica,subjvar="subject",
              splitvars=c("is_pull", "is_target"),rtvar="RT")
```

`splitrel` *Split Half-Based Reliability Coefficients*

Description

Split Half-Based Reliability Coefficients

Usage

```
SpearmanBrown(
  corr,
  ntests = 2,
  fix.negative = c("nullify", "bilateral", "none")
)

FlanaganRulon(x1, x2, fix.negative = c("nullify", "bilateral", "none"))

RajuCoefficient(x1, x2, prop, fix.negative = c("nullify", "bilateral", "none"))
```

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.
`fix.negative`	Determines how to deal with a negative value. "nullify" sets it to zero, "bilateral" applies the correction as if it were a positive number, and then sets it to negative. "none" gives the raw value. It should be noted that negative values are not supposed to occur, and there is no commonly accepted way to deal with them when they do occur.

`x1`	scores from half 1
`x2`	scores from half 2
`prop`	Proportion of the first half to the complete sample

Value

Spearman-Brown-corrected correlation coefficient.

Functions

- `SpearmanBrown`: Perform a Spearman-Brown correction on the provided correlation score.
- `FlanaganRulon`: Compute the true reliability using the Flanagan-Rulon formula, which takes into account inequal variances between split halves.
- `RajuCoefficient`: Compute split-half reliability using the Raju formula, which takes into account unequal split-halves and variances.

See Also

covrel

Examples

```
SpearmanBrown(.5)
FlanaganRulon(a<-rnorm(50),rnorm(50)+a*.5,fix.negative="bilateral")
a<-rnorm(50)
b<-rnorm(50)+a*.5
RajuCoefficient(a,b,prop=.4,fix.negative="bilateral")
```

Index