Package 'splithalf'

August 11, 2022

Type Package

Title Calculate Task Split Half Reliability Estimates

Version 0.8.2

Maintainer Sam Parsons <sam.parsons@radboudumc.nl>

Description Estimate the internal consistency of your tasks with a permutation based split-half reliability approach.
Unofficial release name: ``I eat stickers all the time, dude!".

Depends R (>= 3.5)

Imports tidyr, dplyr, stats, Rcpp, robustbase, ggplot2, plyr, grid, patchwork, psych, lme4, methods

LinkingTo Rcpp

Suggests knitr, rmarkdown, tools,

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

URL https://github.com/sdparsons/splithalf

BugReports https://github.com/sdparsons/splithalf

VignetteBuilder knitr

NeedsCompilation yes

Author Sam Parsons [aut, cre]

Repository CRAN

Date/Publication 2022-08-11 14:30:02 UTC

R topics documented:

`multiverse.plot` *Visualising reliability multiverses*

Description

This function allows the user to plot the output from splithalf_multiverse or testretest_multiverse. The plot includes an upper panel with all reliability estimates (and CIs) and a lower panel that indicates the data processing specifications corresponding to that reliability estimate. The (unofficial) function version name is "This function will make you a master in bird law"

This function examines the output from splithalf_multiverse or testretest_multiverse to extract the proportions of estimates above or below a set threshold (can be the estimate or the upper or lower CI estimates). The (unofficial) function version name is "This function will get you up to here with it"

Usage

```
multiverse.plot(
  multiverse,
  title = "",
  vline = "none",
  heights = c(4, 5),
  SE = FALSE
)

threshold(multiverse, threshold, use = "estimate", dir = "above")
```

Arguments

`multiverse`	multiverse object
`title`	string add a title to the plot? default is ""
`vline`	add a vertical line to the plot, e.g. use .5 for the median reliability estimate
`heights`	must be a vector of length 2, relative heights of plot panels. Defaults to c(4,5)
`SE`	logical includes an additional panel to plot the standard errors of the scores. Note: the heights parameter must be a vector of length 3, e.g. c(2,2,3). Defaults to FALSE
`threshold`	threshold to look for, e.g. 0.7
`use`	set to check the reliability "estimates", or the "upper" or "lower" CIs
`dir`	look "above" or "below" the 'use' at the set threshold

Value

Returns a visualization of a multiverse object

Examples

```
## Not run:
## see online documentation for examples
https://github.com/sdparsons/splithalf
## also see https://psyarxiv.com/y6tcz

## example simulated data
n_participants = 60 ## sample size
n_trials = 80
n_blocks = 2
sim_data <- data.frame(participant_number = rep(1:n_participants,
                       each = n_blocks * n_trials),
                       trial_number = rep(1:n_trials,
                       times = n_blocks * n_participants),
                       block_name = rep(c("A","B"),
                       each = n_trials,
                       length.out = n_participants * n_trials * n_blocks),
                       trial_type = rep(c("congruent","incongruent"),
                       length.out = n_participants * n_trials * n_blocks),
                       RT = rnorm(n_participants * n_trials * n_blocks,
                       500,
                       200),
                       ACC = 1)

## specify several data processing decisions
specifications <- list(RT_min = c(0, 100, 200),
                       RT_max = c(1000, 2000),
                       averaging_method = c("mean", "median"))
## run splithalf, and save the output
difference <- splithalf(data = sim_data,
                        outcome = "RT",
                        score = "difference",
                        conditionlist = c("A"),
                        halftype = "random",
                        permutations = 5000,
                        var.RT = "RT",
                        var.condition = "block_name",
                        var.participant = "participant_number",
                        var.compare = "trial_type",
                        var.ACC = "ACC",
                        compare1 = "congruent",
                        compare2 = "incongruent",
                        average = "mean")

## run splithalf.multiverse to perform the multiverse of data processing
## and reliability estimation
multiverse <- splithalf.multiverse(input = difference,
                                   specifications = specifications)

## can be plot with:
multiverse.plot(multiverse = multiverse,
                title = "README multiverse")
```

```
## End(Not run)
## Not run:
## see online documentation for examples
https://github.com/sdparsons/splithalf
## also see https://psyarxiv.com/y6tcz

## example simulated data
n_participants = 60 ## sample size
n_trials = 80
n_blocks = 2
sim_data <- data.frame(participant_number = rep(1:n_participants,
                       each = n_blocks * n_trials),
                       trial_number = rep(1:n_trials,
                       times = n_blocks * n_participants),
                       block_name = rep(c("A","B"),
                       each = n_trials,
                       length.out = n_participants * n_trials * n_blocks),
                       trial_type = rep(c("congruent","incongruent"),
                       length.out = n_participants * n_trials * n_blocks),
                       RT = rnorm(n_participants * n_trials * n_blocks,
                       500,
                       200),
                       ACC = 1)

## specify several data processing decisions
specifications <- list(RT_min = c(0, 100, 200),
                       RT_max = c(1000, 2000),
                       averaging_method = c("mean", "median"))
## run splithalf, and save the output
difference <- splithalf(data = sim_data,
                        outcome = "RT",
                        score = "difference",
                        conditionlist = c("A"),
                        halftype = "random",
                        permutations = 5000,
                        var.RT = "RT",
                        var.condition = "block_name",
                        var.participant = "participant_number",
                        var.compare = "trial_type",
                        var.ACC = "ACC",
                        compare1 = "congruent",
                        compare2 = "incongruent",
                        average = "mean")

## run splithalf.multiverse to perform the multiverse of data processing
## and reliability estimation
multiverse <- splithalf.multiverse(input = difference,
                                   specifications = specifications)


## the threshold function can be used to return the number of estimates
```

```
## above or below a certain threshold

threshold(multiverse = multiverse,
          threshold = 0.7,
          use = "estimate",
          dir = "above")

## End(Not run)
```

`speedtestdata` *Simulated data for runtime of splithalf package*

Description

This simulation was run to estimate the relative runtimes for different possible combinations of sample sizes and trial numbers etc.

Usage

```
data(speedtestdata)
```

Format

A data frame with 225 rows and 6 variables

Details

- Simcodes for the simulation number
- sample_sizecodes for the sample size
- Number_of_conditionscodes for the number of conditions run
- trialscodes for the number of trials
- permutationscodes for the number of permutations
- runtimecodes for the runtime in seconds

`splithalf` *Internal consistency of task measures via a permutation split-half reliability approach*

Description

This function calculates split half reliability estimates via a permutation approach for a wide range of tasks. Most of the user inputs relate to the variables in the dataset splithalf needs to read in order to estimate reliability. Currently supports response time and accuracy outcomes, for several scoring methods: average, difference, difference of difference scores, and a DPrime development. The (unofficial) version name is "This function gives me the power to fight like a crow"

Usage

```
splithalf(
  data,
  outcome = "RT",
  score = "difference",
  conditionlist = FALSE,
  halftype = "random",
  permutations = 5000,
  var.RT = "latency",
  var.ACC = "accuracy",
  var.condition = FALSE,
  var.participant = "subject",
  var.compare = "congruency",
  compare1 = "Congruent",
  compare2 = "Incongruent",
  average = "mean",
  plot = FALSE,
  round.to = 2,
  check = TRUE
)
```

Arguments

data	specifies the raw dataset to be processed
outcome	indicates the type of data to be processed, e.g. "RT" or "accuracy"
score	indicates how the outcome score is calculated, e.g. most commonly the difference score between two trial types. Can be "average", "difference", "difference_of_difference", and "DPrime"
conditionlist	sets conditions/blocks to be processed
halftype	specifies the split method; "oddeven", "halfs", or "random"
permutations	specifies the number of random splits to run - 5000 is good
var.RT	specifies the RT variable name in data
var.ACC	specific the accuracy variable name in data
var.condition	specifies the condition variable name in data - if not specified then splithalf will treat all trials as one condition
var.participant	specifies the subject variable name in data
var.compare	specifies the variable that is used to calculate difference scores (e.g. including congruent and incongruent trials)
compare1	specifies the first trial type to be compared (e.g. congruent trials)
compare2	specifies the second trial type to be compared (e.g. incongruent trials)
average	use "mean" or "median" to calculate average scores?
plot	logical value giving the option to visualise the estimates in a raincloud plot. defaults to FALSE

round.to	sets the number of decimals to round the estimates to defaults to 2
check	runs several checks of the data to detect participants/conditions/trialtypes with too few trials to run splithalf

Value

Returns a data frame containing permutation based split-half reliability estimates

splithalf is the raw estimate of the bias index

spearmanbrown is the spearman-brown corrected estimate of the bias index

Warning: If there are missing data (e.g one condition data missing for one participant) output will include details of the missing data and return a dataframe containing the NA data. Warnings will be displayed in the console.

Examples

```
## Not run:
## see online documentation for full examples
https://github.com/sdparsons/splithalf
## example simulated data
n_participants = 60 ## sample size
n_trials = 80
n_blocks = 2
sim_data <- data.frame(participant_number = rep(1:n_participants,
                       each = n_blocks * n_trials),
                       trial_number = rep(1:n_trials,
                       times = n_blocks * n_participants),
                       block_name = rep(c("A","B"),
                       each = n_trials,
                       length.out = n_participants * n_trials * n_blocks),
                       trial_type = rep(c("congruent","incongruent"),
                       length.out = n_participants * n_trials * n_blocks),
                       RT = rnorm(n_participants * n_trials * n_blocks,
                       500,
                       200),
                       ACC = 1)

## example run of splithalf on a difference score
splithalf(data = sim_data,
          outcome = "RT",
          score = "difference",
          conditionlist = c("A", "B"),
          halftype = "random",
          permutations = 5000,
          var.RT = "RT",
          var.condition = "block_name",
          var.participant = "participant_number",
          var.compare = "trial_type",
          compare1 = "congruent",
          compare2 = "incongruent",
          average = "mean",
          plot = TRUE)
```

```
## example run of splithalf on an average score
splithalf(data = sim_data,
          outcome = "RT",
          score = "average",
          conditionlist = c("A", "B"),
          halftype = "random",
          permutations = 5000,
          var.RT = "RT",
          var.condition = "block_name",
          var.participant = "participant_number",
          average = "mean")

## example run of splithalf on a difference of differences score
splithalf(data = sim_data,
          outcome = "RT",
          score = "difference_of_difference",
          conditionlist = c("A", "B"),
          halftype = "random",
          permutations = 5000,
          var.RT = "RT",
          var.condition = "block_name",
          var.participant = "participant_number",
          var.compare = "trial_type",
          compare1 = "congruent",
          compare2 = "incongruent",
          average = "mean")


## End(Not run)
```

`splithalf.multiverse` *Multiverse of data processing decisions on internal consistency reliability estimates.*

Description

This function enables the user to run a multiverse of data processing options and extract the resulting (internal consistency) reliability estimates generated by splithalf. The user specifies a set of data processing decisions and passes this to the function, along with a splithalf object. The output can then be explored and plotted as desired.

Usage

```
splithalf.multiverse(input, specifications)
```

Arguments

`input`	splithalf object or list of splithalf objects
`specifications`	list of data processing specifications

Details

The (unofficial) function version name is "This function will let you get honey from a hornets nest"

Value

Returns a multiverse object containing the reliability estimates and dataframes from all data processing specifications provided

Examples

```
## Not run:
## see online documentation for examples
https://github.com/sdparsons/splithalf
## also see https://psyarxiv.com/y6tcz

## example simulated data
n_participants = 60 ## sample size
n_trials = 80
n_blocks = 2
sim_data <- data.frame(participant_number = rep(1:n_participants,
                       each = n_blocks * n_trials),
                       trial_number = rep(1:n_trials,
                       times = n_blocks * n_participants),
                       block_name = rep(c("A","B"),
                       each = n_trials,
                       length.out = n_participants * n_trials * n_blocks),
                       trial_type = rep(c("congruent","incongruent"),
                       length.out = n_participants * n_trials * n_blocks),
                       RT = rnorm(n_participants * n_trials * n_blocks,
                       500,
                       200),
                       ACC = 1)

## specify several data processing decisions
specifications <- list(RT_min = c(0, 100, 200),
                       RT_max = c(1000, 2000),
                       averaging_method = c("mean", "median"))
## run splithalf, and save the output
difference <- splithalf(data = sim_data,
                        outcome = "RT",
                        score = "difference",
                        conditionlist = c("A"),
                        halftype = "random",
                        permutations = 5000,
                        var.RT = "RT",
                        var.condition = "block_name",
                        var.participant = "participant_number",
                        var.compare = "trial_type",
                        var.ACC = "ACC",
                        compare1 = "congruent",
                        compare2 = "incongruent",
                        average = "mean")
```

```
## run splithalf.multiverse to perform the multiverse of data processing
## and reliability estimation
multiverse <- splithalf.multiverse(input = difference,
                                   specifications = specifications)

## can be plot with:
multiverse.plot(multiverse = multiverse,
                title = "README multiverse")

## End(Not run)
```

`testretest.multiverse` *Multiverse of data processing decisions on test retest reliability estimates.*

Description

This function enables the user to run a multiverse of data processing options and extract the resulting test-retest reliability estimates. The user specifies a set of data processing decisions and passes this to the function, along with specifying key variables within several "var." inputs (so that the function knows where to find your participant ids and RTs for example)

Usage

```
testretest.multiverse(
  data,
  specifications,
  test = "ICC2",
  outcome = "RT",
  score = "difference",
  var.participant = "subject",
  var.ACC = "correct",
  var.RT = "RT",
  var.time = "time",
  var.compare = "congruency",
  compare1 = "Congruent",
  compare2 = "Incongruent"
)
```

Arguments

`data`	dataset
`specifications`	list of data processing specifications
`test`	test retest statistic, "ICC2", "cor", "ICC3"
`outcome`	from splithalf() specifies the RT outcome - only "RT" available currently
`score`	currently only "difference" scores are supported

var.participant
: = "subject",

var.ACC
: = "correct",

var.RT
: = "RT"

var.time
: codes the time variable (currently only works for 2 timepoints)

var.compare
: = "congruency" trial type used to create difference scores

compare1
: specifies the first trial type to be compared (e.g. "Congruent" trials)

compare2
: specifies the second trial type to be compared (e.g. "Incongruent" trials)

Details

The (unofficial) function version name is "This function will help you pay the troll toll"

Value

Returns a multiverse object containing the reliability estimates and dataframes from all data processing specifications provided

Examples

```
## Not run:
## see online documentation for examples
https://github.com/sdparsons/splithalf
## also see https://psyarxiv.com/y6tcz

n_participants <- 80 ## sample size
n_trials <- 120
n_blocks <- 2

sim_data_mv <- data.frame(participant_number = rep(1:n_participants,
                                                   each = n_blocks * n_trials),
                          trial_number = rep(1:n_trials,
                                             times = n_blocks * n_participants),
                          block_name = rep(c(1,2),
                                           each = n_trials,
                                       length.out = n_participants * n_trials * n_blocks),
                          trial_type = rep(c("congruent","congruent",
                                             "incongruent","incongruent"),
                                   length.out = n_participants * n_trials * n_blocks / 2),
                          RT = rnorm(n_participants * n_trials * n_blocks,
                                     500,
                                     200),
                          ACC = c(rbinom(n_participants *
                                           n_trials *
                                           n_blocks / 6,
                                         1, .5),
                                  rbinom(n_participants *
                                           n_trials *
                                           n_blocks / 6,
                                         1, .7),
```

```
                                    rbinom(n_participants *
                                             n_trials *
                                             n_blocks / 6,
                                           1, .9),
                                    rbinom(n_participants *
                                             n_trials *
                                             n_blocks / 6,
                                           1, .5),
                                    rbinom(n_participants *
                                             n_trials *
                                             n_blocks / 6,
                                           1, .7),
                                    rbinom(n_participants *
                                             n_trials *
                                             n_blocks / 6,
                                           1, .9)))

specifications <- list(
ACC_cutoff = c(0, 0.5),
RT_min           = c(0, 200),
RT_max            = c(2000, 3000),
RT_sd_cutoff      = c(0, 2),
split_by          = c("subject", "trial"),
averaging_method  = c("mean")
)

icc2 <- testretest.multiverse(data = sim_data_acc,
specifications,
test = "ICC2",
score = "difference",
var.participant = "participant_number",
var.ACC = "ACC",
var.RT = "RT",
var.time = "block_name",
var.compare = "trial_type",
compare1 = "congruent",
compare2 = "incongruent")

multiverse.plot(icc2)


## End(Not run)
```

Index