Package ‘autoFC’

June 7, 2021

Type Package

Title Automatic Construction of Forced-Choice Tests

Version 0.1.2

Author Mengtong Li [cre, aut] (<https://orcid.org/0000-0002-1766-4976>),
Tianjun Sun [aut] (<https://orcid.org/0000-0002-3655-0042>),
Bo Zhang [aut]

Maintainer Mengtong Li <ml70@illinois.edu>

Description

Forced-choice (FC) response has gained increasing popularity and interest for its resistance to faking when well-designed (Cao & Drasgow, 2019 <doi:10.1037/apl0000414>). To established well-designed FC scales, typically each item within a block should measure different trait and have similar level of social desirability (Zhang et al., 2020 <doi:10.1177/1094428119836486>). Recent study also suggests the importance of high inter-item agreement of social desirability between items within a block (Pavlov et al., 2021 <doi:10.31234/osf.io/hmnrc>). In addition to this, FC developers may also need to maximize factor loading differences (Brown & Maydeu-Olivares, 2011 <doi:10.1177/0013164410375112>) or minimize item location differences (Cao & Drasgow, 2019 <doi:10.1037/apl0000414>) depending on scoring models. Decision of which items should be assigned to the same block, termed item pairing, is thus critical to the quality of an FC test. This pairing process is essentially an optimization process which is currently carried out manually. However, given that we often need to simultaneously meet multiple objectives, manual pairing becomes impractical or even not feasible once the number of latent traits and/or number of items per trait are relatively large. To address these problems, autoFC is developed as a practical tool for facilitating the automatic construction of FC tests, essentially exempting users from the burden of manual item pairing and reducing the computational costs and biases induced by simple ranking methods. Given characteristics of each item (and item responses), FC tests can be automatically constructed based on user-defined pairing criteria and weights as well as customized optimization behavior. Users can also construct parallel forms of the same test following the same pairing rules.

URL https://github.com/tspsyched/autoFC

BugReports https://github.com/tspsyched/autoFC/issues

Encoding UTF-8

Imports irrCAC

RoxygenNote 7.1.1

Suggests rmarkdown, knitr

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2021-06-07 07:20:04 UTC

R topics documented:

`cal_block_energy` *Calculation of Item Block "Energy"*

Description

Calculates the total "energy" of one or multiple paired item blocks, which is a linear combination of different functions applied to different item characteristics of interest.

Usage

```
cal_block_energy(block, item_chars, weights, FUN)
```

Arguments

`block`	An n by k integer matrix, where n is the number of item blocks and k is the number of items per block.
`item_chars`	An m by r data frame, where m is the total number of items to sample from, whether it is included in the block or not, whereas r is the number of item characteristics.
`weights`	A vector of length r with weights for each item characteristics in `item_chars`. Should provide a weight of 0 for specific characteristics not of interest, such as item ID.
`FUN`	A vector of customized function names for optimizing each item characteristic within each block, with length r.

Details

This energy calculation function serves as the core for determining the acceptance or rejection of a newly built block over the previous one.

Higher energy is considered more preferable in this case.

Items in the same block can be paired based on characteristics such as:

Mean score, Item Factor, Factor loading, Item IRT Parameters, Reverse Coding, etc.

Pairings of different characteristics can be optimized in different way, by determining the customized function vector `FUN` and the corresponding `weights`.

Value

A numeric value indicating the total energy for the given item block(s).

Note

Use `cal_block_energy_with_iia` if inter-item agreement (IIA) metrics are needed.

Author(s)

Mengtong Li

Examples

```
## Simulate 60 items loading on different Big Five dimensions,
## with different mean and item difficulty

item_dims <- sample(c("Openness","Conscientiousness","Neuroticism",
                      "Extraversion","Agreeableness"), 60, replace = TRUE)
item_mean <- rnorm(60, 5, 2)
item_difficulty <- runif(60, -1, 1)

## Construct data frame for item characteristics and produce
## 20 random triplet blocks with these 60 items

item_df <- data.frame(Dimensions = item_dims, Mean = item_mean,
                      Difficulty = item_difficulty)
solution <- make_random_block(60, 60, 3)

## See ?facfun for its use.
cal_block_energy(solution, item_chars = item_df,
              weights = c(1,1,1), FUN = c("facfun", "var", "var"))
```

cal_block_energy_with_iia

Calculation of Item Block "Energy" with IIAs Included

Description

Calculates the total "energy" of one or multiple paired item blocks, which is a linear combination of different functions applied to different item characteristics of interest.

This function extends `cal_block_energy` function with consideration of inter item agreement (IIA) metrics.

Usage

```
cal_block_energy_with_iia(block, item_chars, weights,
                                   FUN, rater_chars,
                                   iia_weights = c(BPlin = 1, BPquad = 1,
                                   AClin = 1, ACquad = 1), verbose = FALSE)
```

Arguments

`block`, `item_chars`, `weights`, `FUN`
: See `?cal_block_energy` for details.

`rater_chars`
: A *p* by *m* numeric matrix with scores of each of the *p* participants for the *m* items.

`iia_weights`
: A vector of length 4 indicating weights given to each IIA metric:

  Linearly weighted AC (Gwet, 2008; 2014);

  Quadratic weighted AC;

  Linearly weighted Brennan-Prediger (BP) Index(Brennan & Prediger, 1981; Gwet, 2014);

  Quadratic weighted BP.

`verbose`
: Logical. Should IIAs be printed when this function is called?

Details

This energy calculation function serves as the core for determining the acceptance or rejection of a newly built block over the previous one. Higher energy is considered more preferable in this case.

Items in the same block can be paired based on characteristics such as: Mean score, Item Factor, Factor loading, Item IRT Parameters, Reverse Coding, etc.

In addition, IIAs can be adopted to further estimate rater agreements between different items, if such information is available for the researchers.

Pairings of different characteristics can be optimized in different way, by determining the customized function vector `FUN` and the corresponding `weights`. Currently only linear weighted combination for IIAs can be used in optimization.

Value

A numeric value indicating the total energy for the given item block(s).

Note

Use `cal_block_energy_with_iia` if inter-item agreement (IIA) metrics are needed.

Author(s)

Mengtong Li

References

Brennan, R. L., & Prediger, D. J. (1981). Coefficient kappa: Some uses, misuses, and alternatives. *Educational and Psychological Measurement, 41*(3), 687-699. https://doi.org/10.1177/001316448104100307

Gwet, K. L. (2008). Computing inter rater reliability and its variance in the presence of high agreement. *British Journal of Mathematical and Statistical Psychology, 61*(1), 29-48. https://doi.org/10.1348/000711006X126600

Gwet, K. L. (2014). *Handbook of inter-rater reliability (4th ed.): The definitive guide to measuring the extent of agreement among raters*. Gaithersburg, MD: Advanced Analytics Press.

See Also

`cal_block_energy`

Examples

```
## Simulate 60 items loading on different Big Five dimensions,
## with different mean and item difficulty

item_dims <- sample(c("Openness","Conscientiousness","Neuroticism",
                      "Extraversion","Agreeableness"), 60, replace = TRUE)
item_mean <- rnorm(60, 5, 2)
item_difficulty <- runif(60, -1, 1)

## Construct data frame for item characteristics and produce
## 20 random triplet blocks with these 60 items

item_df <- data.frame(Dimensions = item_dims, Mean = item_mean,
                      Difficulty = item_difficulty)
solution <- make_random_block(60, 60, 3)


## Simple simulation of responses from 600 participants on the 60 items.
## In practice, should use real world data or simluation based on IRT parameters.

item_responses <- matrix(sample(seq(1:5), 600*60, replace = TRUE), ncol = 60, byrow = TRUE)


cal_block_energy_with_iia(solution, item_chars = item_df, weights = c(1,1,1),
                          FUN = c("facfun", "var", "var"),
```

```
                    rater_chars = item_responses, iia_weights = c(1,1,1,1))
```

---

facfun *Function for Checking If All Items in a Vector Are Unique*

Description

Returns *1* if each element in the vector is unique, and *0* otherwise.

Usage

```
facfun(vec)
```

Arguments

vec	Input vector.

Value

1 if each element in the vector is unique, and *0* otherwise.

Author(s)

Mengtong Li

Examples

```
facfun(c("Openness", "Neuroticism", "Agreeableness"))
facfun(c("Openness", "Openness", "Agreeableness"))
```

---

get_iia *Helper Function for Outputting IIA Characteristics of Each Block*

Description

This function prints IIA metrics for select items, given the individual responses for the items.

Usage

```
get_iia(block, data)
```

Arguments

block	An n by k integer matrix, where n is the number of item blocks and k is the number of items per block.
data	A p by m numeric matrix with scores of each of the p participants for the m items.

Value

An *n* by *k* matrix indicating the four IIA metrics for each item block.

Author(s)

Mengtong Li

Examples

```
item_responses <- matrix(sample(seq(1:5), 600*60, replace = TRUE), ncol = 60, byrow = TRUE)
get_iia(matrix(seq(1:60), ncol = 3, byrow = TRUE), item_responses)
```

---

make_random_block — *Construction of Random Item Blocks*

Description

Returns a matrix of randomly paired blocks where each row represents a block.

Usage

```
make_random_block(total_items, target_items = total_items, item_per_block)
```

Arguments

`total_items`	Integer value. Determines the total number of items we sample from.
`target_items`	Integer value. Determines the number of items to use from `total_items` to build item blocks. Default to be equal to `total_items`. Should be no more than `total_items`.
`item_per_block`	Integer value. Determines the number of items in each item block. Should be no less than 2.

Details

Given the total number of items to pair from, number of items to build paired blocks and number of items in each block, `make_random_block` produces a matrix randomly paired blocks where each row represents a block.

It can also accommodate cases when `target_items` is not a multiple of `item_per_block`.

Can be used as initial solution for other functions in this package.

Value

A matrix of integers indicating the item numbers, where the number of rows equals `target_items` divided by `item_per_block`, rounded up, and number of columns equals `item_per_block`.

Note

If target_items is not a multiple of item_per_block, the item set produced by target_items will be looped until number of sampled items becomes a multiple of item_per_block.

Author(s)

Mengtong Li

Examples

```
# Try out cases where you make target_items the default.
make_random_block(60, item_per_block = 3)

# You can also set your own values of target_items.
make_random_block(60, 45, item_per_block = 3)

# Also see what happens if target_items is not a multiple of item_per_block.
make_random_block(60, 50, item_per_block = 3)
```

---

sa_pairing_generalized

Automatic Item Pairing Method in Forced-Choice Test Construction

---

Description

Automatic construction of forced-choice tests based on Simulated Annealing algorithm. Allows items to be:

1. Matched in either pairs, triplets, quadruplets or blocks of any size;

2. Matched based on any number of item-level characteristics (e.g. Social desirability, factor) based on any customized criteria;

3. Matched based on person-level inter-item agreement (IIA) metrics.

Usage

```
sa_pairing_generalized(block, total_items, Temperature,
                                eta_Temperature = 0.01, r = 0.999,
                                end_criteria = 10^(-6),
                                item_chars, weights, FUN,
                                n_exchange = 2, prob_newitem = 0.25,
                                use_IIA = FALSE, rater_chars,
                                iia_weights = c(BPlin = 1, BPquad = 1,
                                ACLin = 1, ACquad = 1))
```

Arguments

block
: An n by k integer matrix, where n is the number of item blocks and k is the number of items per block.

  Serves as the initial starting blocks for the automatic pairing method.

total_items
: Integer value. How many items do we sample from in order to build this block? Should be more than number of unique values in block.

Temperature
: Initial temperature value. Can be left blank and be computed based on the absolute value of initial energy of block (Recommended), and scaled by eta_Temperature.

  In general, higher temperature represents a higher probability of accepting an inferior solution.

eta_Temperature
: A positive numeric value. The ratio of initial temperature to initial energy of block, if Temperature is not designated.

r
: A positive numeric value less than 1. Determines the reduction rate of Temperature after each iteration.

end_criteria
: A positive numeric value less than 1. Iteration stops when temperature drops to below end_criteria * Temperature. Default to be 10^-6.

item_chars
: An m by r data frame, where m is the total number of items to sample from, whether it is included in the block or not, whereas r is the number of item characteristics.

weights
: A vector of length r with weights for each item characteristics in item_chars. Should provide a weight of 0 for specific characteristics not of interest, such as item ID.

FUN
: A vector of customized function names for optimizing each item characteristic within each block, with length r.

n_exchange
: Integer value. Determines how many blocks are exchanged in order to produce a new solution for each iteration. Should be a value larger than 1 and less than nrow(block).

prob_newitem
: A value between *0* and *1*. Probability of choosing the strategy of picking a new item, when not all candidate items are used to build the FC scale.

use_IIA
: Logical. Are IIA metrics used when performing automatic pairing?

rater_chars
: A p by m numeric matrix with scores of each of the p participants for the m items. Ignored when use_IIA == FALSE.

iia_weights
: A vector of length 4 indicating weights given to each IIA metric:

  Linearly weighted AC (Gwet, 2008; 2014);

  Quadratic weighted AC;

  Linearly weighted Brennan-Prediger (BP) Index(Brennan & Prediger, 1981; Gwet, 2014);

  Quadratic weighted BP.

Value

A list containing:

`block_initial` Initial starting block
`energy_initial` Initial energy for `block_initial`
`block_final` Final paired block after optimization by SA
`energy_final` Final energy for `block_final`

Note

The essence of SA is the probablistic acceptance of solutions inferior to the current state, which avoids getting stuck in local maxima/minima. It is also recommended to try out different values of `weights,iia_weights,eta_Temperature` to find out the best combination of initial temperature and energy value in order to provide optimally paired blocks.

Use `cal_block_energy_with_iia` if inter-item agreement (IIA) metrics are needed.

Author(s)

Mengtong Li

References

Brennan, R. L., & Prediger, D. J. (1981). Coefficient kappa: Some uses, misuses, and alternatives. *Educational and Psychological Measurement, 41*(3), 687-699. https://doi.org/10.1177/001316448104100307

Gwet, K. L. (2008). Computing inter rater reliability and its variance in the presence of high agreement. *British Journal of Mathematical and Statistical Psychology, 61*(1), 29-48. https://doi.org/10.1348/000711006X126600

Gwet, K. L. (2014). *Handbook of inter-rater reliability (4th ed.): The definitive guide to measuring the extent of agreement among raters*. Gaithersburg, MD: Advanced Analytics Press.

Examples

```
## Simulate 60 items loading on different Big Five dimensions,
## with different mean and item difficulty

item_dims <- sample(c("Openness","Conscientiousness","Neuroticism",
                      "Extraversion","Agreeableness"), 60, replace = TRUE)
item_mean <- rnorm(60, 5, 2)
item_difficulty <- runif(60, -1, 1)


item_df <- data.frame(Dimensions = item_dims,
                      Mean = item_mean, Difficulty = item_difficulty)
solution <- make_random_block(60, 60, 3)


item_responses <- matrix(sample(seq(1:5), 600*60, replace = TRUE), nrow = 60, byrow = TRUE)

## Automatic pairing, without use of IIAs
## See ?facfun for information about what it does

sa_pairing_generalized(solution, 60, eta_Temperature = 0.01,
                                    r = 0.999, end_criteria = 0.001,
                                    weights = c(1,1,1),
```

```
                                    item_chars = item_df,
                                    FUN = c("facfun", "var", "var"))


## Automatic pairing, with IIAs

sa_pairing_generalized(solution, 60, eta_Temperature = 0.01,
                                    r = 0.999, end_criteria = 0.001,
                                    weights = c(1,1,1),
                                    item_chars = item_df,
                                    FUN = c("facfun", "var", "var"),
                                    use_IIA = TRUE,
                                    rater_chars = item_responses,
                                    iia_weights = c(BPlin = 1, BPquad = 1,
                                    AClin = 1, ACquad = 1))
```

Index