Package 'talentlens'

July 6, 2016

Type Package

Title R package for team TalentLens

Version 0.2

Date 2016-06-08

Author Morgan Strom

Maintainer `<morgan.strom@pearson.com>`

Description Functions for test scoring, theta estimation etc.

License No license

LazyData TRUE

Depends dplyr, tidyr, catR, psych, RMySQL, ltm

R topics documented:

`talentlens-package` *Package for TalentLens R&D team*

Description

Contains functions for getting access to data from a Concerto test, monitoring participants during data collection, balancing norm groups, finding optimal anchor items given a score matrix, scoring responses to items, estimating theta from a score matrix given item parameters, calculating bagged IRT parameters and generating target proportions according to UK census 2011.

Access information about these functions using the links below, or by typing e.g. '?fetchConcerto-Data' in the console.

Details

Package:	talentlens
Type:	Package
Version:	0.2
Date:	2016-06-17
License:	No licence

1. fetchConcertoData
2. monitorParticipants
3. balancer
4. findAnchors
5. scoreDataFrame
6. scoreResponses
7. eap
8. baggedParams
9. targetUKCensus

Author(s)

Morgan Strom

`baggedParams` *Function to calculate bagged IRT parameters from a score matrix*

Description

Performs IRT fitting algorithm repeatedly (R times) on a random sample of the score matrix (with replacement), according to a bootstrap procedure. The results are then aggregated (bagging) and a list of parameter and error estimates is returned.

Usage

```
baggedParams(score_mat, R, model = c("2pl", "rasch"),
central.fun = mean, spread.fun = sd)
```

Arguments

`score_mat`	(nxk) numeric matrix rows corresponding to participants and columns corresponding to items. NOTE: columns need to be named after item ids.

`R`	Number of iterations for the bootstrap loop. Higher R increases the time the function needs for completion, but also gives better estimates of bagged parameters and bootstrapped errors. Default: 199 (on a Dell laptop, this takes about 1 min to run on a dataset with 99 rows).
`model`	The IRT model to use. This function currently supports rasch and 2pl models.
`central.fun`	The function used for aggregating results and find the central values. Default: mean.
`spread.fun`	The function used for estimating parameter errors. Default: sd.

Value

`params`	A (kx2) matrix of IRT parameters
`error`	A (kx2) matrix of estimation errors

Author(s)

Morgan Strom

`balancer` *Function to balance a sample with regards to a dominant subgroup*

Description

Takes a group variable (factor vector) f and the desired maximum proportion of the largest subgroup (factor level) p, samples randomly from the largest group and returns a logical vector, indicating which cases to retain to reach the desired proportion of the largest group.

Usage

```
balancer(f, p, silent = TRUE)
```

Arguments

`f`	Factor vector representing a group variable to be balanced, where the levels represent groups/categories in the data. For example, this could be a column in a dataframe, indicating the gender of the participants.
`p`	Desired maximum proportion for the largest subgroup. For example, if we want a balanced sample with regards to gender, we may want to allow for no more than 50 % of the total sample to be from either gender.
`silent`	If TRUE (default), messages are not printed to the console. Set to FALSE if you want to see how many cases are dropped randomly.

Details

Formula for determining the necessary reduction of the sample size (delta_n) for the dominant subgroup:

$$\Delta n = \frac{N(p_2 - p_1)}{(1 - p_2)}$$

Value

Logical vector of the same length as f, with values TRUE for cases to keep and FALSE for cases to drop.

Author(s)

Morgan Strom

Examples

```
##Generate factor vector with unbalanced levels
f <- factor(x = c(rep("A", 20), rep("B", 50), rep("C", 30)))
table(f)

#Calculate proportions of the groups
table(f) / length(f)

#Use balancer() function to reduce the largest group to 40%

#First, create the index vector
i <- balancer(f, p = 0.40, silent = FALSE)

#Second, subset the factor vector and store in a new object, f2
f2 <- f[i]

#Calculate new proportions
table(f2) / length(f2)
```

---

eap *Function to estimate ability using EAP algorithm*

Description

Takes a named score vector and a data frame with known item parameters, and returns the estimated theta.

Usage

```
eap(x, params, D = 1.702)
```

Arguments

x	Vector with correct responses coded as 1 and incorrect responses as 0. NOTE: the values need to be named with item ids
params	Data frame with unique item names and corresponding item parameters on each row, NOTE: columns need to be named "id", "a", "b" and "c"
D	Scaling constant D (Defaults to 1.702 for normal ogive model)

Details

The theta estimation algorithm is based on the implementation of the EAP algorithm in the Get-Feedback platform. Instructions from Louis-Charles Vannier.

An alternative to using this function is `eapEst` from the `catR` package.

Value

Estimated theta value.

Author(s)

Morgan Strom

See Also

`eapEst` `eapSem`

Examples

```
#Create named score vector
score_vec <- c(1,1,0,1,0)
names(score_vec) <- c("item1", "item2", "item3", "item4", "item5")

#Create parameter data frame
params <- data.frame(id = c("item1", "item2", "item3", "item4", "item5"),
                     a = c(0.7, 0.8, 0.9, 1, 1.1),
                     b = c(-2, -1, 0, 1, 2),
                     c = c(0,0,0,0,0))

#Estimate theta
theta <- eap(score_vec, params)

#Using the function for a matrix with 4 observations
score_mat <- matrix(c(1,1,0,1,0,
                      1,0,0,0,0,
                      1,1,1,1,0,
                      1,1,1,1,1),
                    nrow = 4, ncol = 5, byrow = TRUE)
colnames(score_mat) <- c("item1", "item2", "item3", "item4", "item5")

thetas <- apply(score_mat, 1, eap, params=params)
```

`fetchConcertoData` *Function to fetch data from Concerto*

Description

Returns candidate scores and demographics in a data frame format

Usage

```
fetchConcertoData(dbname, host, user, password, backup = FALSE, fetch = "scores")
```

Arguments

`dbname`	String containing the name of the database where the test is stored. E.g "concerto_2" for the UK version of Watson-Glaser. This information can be found in the spreadsheet "Concerto data collection" on Google Drive.
`host`	String containing the IP address of the Concerto server. This information can also be found in the spreadsheet "Concerto data collection" on Google Drive.
`user`	String containing the user name for accessing the database. To get access to this information, please email the author of this package.
`password`	String containing the password for accessing the database. To get access to this information, please email the author of this package.
`backup`	Boolean value - if TRUE (default), all candidate information from the server (tables candidate_summary and candidate_responses) will be stored as .csv files in the working directory.
`fetch`	String defining what data to fetch. If "scores" (default), the data will contain the scores matrix with 1 representing a correct response and 0 representing an incorrect response. If "responses", the data will instead contain participants' responses to the items.

Value

`complete_data`	A dataframe containing both demographic information and item scores from the concerto test. If you stored this in a dataframe called `df`, and you wish to save this as a .csv file in your working directory, type `write.csv(df, file = "df.csv", na = "", row.` in the console and press Enter.

Author(s)

Morgan Strom, TalentLens UK

`findAnchors` *Function to suggest optimal anchor items from a score matrix*

Description

This function selects optimal anchor items in two steps: first, based on CTT analyses (proportion of participants answering the item correctly and item-total biserial correlation) and second, based on a chi-square test of item fit under the chosen IRT model.

Usage

```
findAnchors(score_matrix, model = c("rasch", "2pl"),
bagging = FALSE, R = NA,
min_cor = 0.2, min_p = 0.1, max_p = 0.9, verbose = FALSE)
```

Arguments

`score_matrix`	A numeric matrix where each row represent the scores of a participant, and each row represent a item. NOTE the columns need to be named with item ids!
`model`	The IRT model to use for parameter fitting. Either "rasch" (for smaller sample sizes) or "2pl" (for larger sample sizes)
`bagging`	If TRUE, the parameters will be estimated using bagging (see package bagged-Params). If FALSE, parameters will be estimated using one single run of the item fitting algorithm.
`R`	For bagging, this is the number of bootstrap iterations.
`min_cor`	The minimum allowed item-total biserial correlation. Default: 0.2
`min_p`	The minimum allowed proportion of participants answering a given item correctly. Default: 0.1
`max_p`	The maximum allowed proportion of participants answering a given item correctly. Default: 0.9
`verbose`	If TRUE, information about the item selection will be printed to the console.

Value

A two-column matrix of item parameters for the suggested anchor items. The rows are named after the item ids, and the columns are named "b" (difficulty parameter) and "a" (discrimination parameter).

Author(s)

Morgan Strom

`monitorParticipants` *Function to monitor participants from MTurk projects*

Description

Provides MTurk Worker ID along with time of completion, number of attempted items and total time for completion (in minutes).

Usage

```
monitorParticipants(dbname, host, user, password)
```

Arguments

`dbname`	String containing the name of the database where the test is stored. E.g "concerto_2" for the UK version of Watson-Glaser. This information can be found in the spreadsheet "Concerto data collection" on Google Drive.
`host`	String containing the IP address of the Concerto server. This information can also be found in the spreadsheet "Concerto data collection" on Google Drive.
`user`	String containing the user name for accessing the database. To get access to this information, please email the author of this package.
`password`	String containing the password for accessing the database. To get access to this information, please email the author of this package.

Value

Data frame containing the columns `session_id`, `worker_id`, `timestamp`, `attempted` and `total_time`.

Author(s)

Morgan Strom

`scoreDataFrame` *Function to score an entire data frame*

Description

Loops over all items in "key" and returns a score matrix with named columns. NOTE: the rows will be in the same order as the original data frame "df", but the columns may be in a different order.

Usage

```
scoreDataFrame(df, key)
```

Arguments

`df`	A data frame containing one candidate per row and one item per column (may also include additional columns, like candidate id etc). NOTE: the column names for the items need to correspond to the item ids in the column "id" in the "key"" data frame.
`key`	A data frame containing one column for item ids ("id") and one column for the correct responses ("key"). If the correct response is a range, this should be represented in the following format: "RANGE: 1 - 1.1".

Value

A numeric score matrix, where the rows correspond to participants (same order as in the original data frame df) and the columns correspond to item ids (may be in different order from the original data frame). 1 represents a correct response, 0 an incorrect response and NA represents missing data.

Author(s)

Morgan Strom

Examples

```
key <- data.frame(id = c("item_1", "item_2", "item_3"),
                  key = c("2", 3, "RANGE: 1 - 1.1"))

df <- data.frame(id = c("cand_1", "cand_2", "cand_3"),
                 item_1 = c(2, 1.99, 2.01),
                 item_2 = c(3, 2.99, 3.01),
                 item_3 = c(1, 0.99, 1.01))

scoreDataFrame(df, key)
```

`scoreResponses` *Function to score a vector of responses*

Description

Returns a vector of scores, given a vector of responses to a specific item, a vector with the correct response(s) ("key") and the item id.

Usage

```
scoreResponses(responses, key, item_id)
```

Arguments

`responses`	A numeric vector with the responses to an item
`key`	A numeric vector with the correct response to the item. If the response format is Multiple Choice, this will be a single integer value. If the response format is Free (such as some items in Athena), this will be two values representing the lower and the upper bounds of the range of acceptable responses.
`item_id`	A character vector with the item id

Value

A numeric score vector with 1 for correct responses, 0 for incorrect responses and NA for missing values

Author(s)

Morgan Strom

Examples

```
responses <- c(1, 0, 1.05, 0.99, 2)

#Multiple choice with 2 as the correct answer
key <- 2
scoreResponses(responses, key, "test_item")

#Free response with lower bound 1 and upper bound 1.1
key <- c(1, 1.1)
scoreResponses(responses, key, "test_item")
```

`scoringKey` *Function to create a scoring key for the* `scoreData` *function*

Description

Creates a list with one entry per item, where the name is the item id and the value is the correct response. The value can either be a single number, when there is only one correct answer, or a vector of two numbers, where the correct answer is within a range (Athena items for example)

Usage

```
scoringKey(key.df)
```

Arguments

`key.df`	Data frame containing one column with item ID (named "id") and one column with the corresponding correct responses (named "key"). If the correct response is a range, use the following format: "RANGE: 1 - 1.1".

Value

List with one entry per item. The entry is named after the item id, and the value is the correct response to the item.

Author(s)

Morgan Strom

Examples

```
df <- data.frame(id = c("item_1", "item_2", "item_3"),
                 key = c("2", 3, "RANGE: 1 - 1.1"))

scoringKey(df)
```

`targetUKCensus` *Function to calculate target proportions from UK Census data 2011*

Description

Returns a dataframe with 60 rows, with target proportions for each possible subgroup with regards to age, gender, ethnicity and education level

Usage

```
targetUKCensus()
```

Details

Age is divided into 2 groups: 16-24 and 25-64. Gender is also divided into 2 groups: Female and Male. Ethnicity is divided into 5 groups: White, Mixed, Asian, Black/African/Caribbean (BAC) and Other. Finally, Education is divided into 3 groups: No University Education (Levels 1-3), University Education (Level 4) and Other.

Value

Data frame containing all possible combinations of the groups described above (2 * 2 * 5 * 3 = 60 rows), where each row represent a unique combination. The column "ExpectedP" contains the target proportion for that specific intersection of groups.

Author(s)

Morgan Strom

Index