Package 'simplyPsycho'

August 28, 2023

Type Package

Title Simply Psycho(metrics)

Version 0.1.0

Author Philip Eaton

Maintainer Philip Eaton <philip.eaton@stockton.edu>

Description Simple functions for doing psychometric analysis of assessment data.

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Imports aws.s3,
BiocManager,
corrplot,
dplyr,
EFAtools,
hasseDiagram,
igraph,
mice,
mirt,
multimode,
nortest,
plotly,
psych,
SuppDists,
stringr,
tidyverse,
tibble,
writexl

R topics documented:

`build.psycho.datafile` *Build simplyPsycho data from external data files. INTERNAL*

Description

Build simplyPsycho data from external data files. INTERNAL

Usage

```
build.psycho.datafile(
  datafile,
  new.data,
  new.course.name = NULL,
  new.term = NULL,
  answerKey = NULL,
  gen.blanks = FALSE,
  blank.ID = NULL
)
```

`contradiction.mat.generator` *Item Tree Analysis: Contradiction Matrix Generator*

Description

Generates the contradiction matrix used in item tree analysis. Finds the number of contradictions in the data to the assumption that a correct response to item ii implies a correct response to item jj. This is found by filtering for the students who all got ii correct, then counting those who got jj incorrect.

Usage

```
contradiction.mat.generator(data)
```

Arguments

`data`	An nS by nQ matrix or data frame of a dichotomous graded (0 or 1) sample, where nS is the number of students in the sample and nQ is the number of questions.

Value

Contradiction matrix to use used in ITA.

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0)
data.alpha <- temp.data$data.alpha
data.num <- temp.data$data.num

# Get contradiction matrix
contradiction.mat.generator(data.num)
```

`cronbachAlpha` *CTT Cronbach's Alpha:*

Description

Calculate Cronbach's Alpha for an assessment from sample data..

Usage

```
cronbachAlpha(data)
```

Arguments

`data`	An nS by nQ matrix or data frame of a dichotomous graded (0 or 1) sample, where nS is the number of students in the sample and nQ is the number of questions.

Value

Cronbach's Alpha value for the given graded data.

Examples

```
# Use sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0)

# Cronbach's Alpha
cronbachAlpha(temp.data)
```

`cttCronbachAlpha` *CTT Cronbach's Alpha (of multiple courses) with bootstrapping for error estimation:*

Description

Calculate Cronbach's Alpha for an assessment from sample data.

Usage

```
cttCronbachAlpha(data, booted = FALSE, nRuns = 100)
```

Arguments

data	Can be: 1) An nS by nQ matrix or data frame of a dichotomous graded (0 or 1) sample, where nS is the number of students in the sample and nQ is the number of questions. 2) The output of piql.data.select() for one or multiple courses.
booted	Logical (default = FALSE). FALSE means no bootstrapping will be performed. TRUE turns on the bootstrapping feature.
nRuns	Number of random samples to use in the bootstrapping (default = 100). For publications it is recommended that 10,000 runs be performed since sample error goes as 1/sqrt(nRuns).

Value

When booted = FALSE (the default setting) then the straight calculated value will be returned. If booted = TRUE, the function will output the mean and standard deviation Cronbach's Alpha values calculated from nRuns randomly sampled with replacement data sets from the given sample.

Examples

```
# Pull in sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0)
data.num <- temp.data$data.num

# Straight Cronbach's Alpha
cttCronbachAlpha(data.num)

# Booted using 100 random samples (default)
cttCronbachAlpha(data.num, booted = TRUE)

# Booted using 1000 random samples by manually setting nRuns
cttCronbachAlpha(data.num, booted = TRUE, nRuns = 1000)
```

---

cttCronbachAlpha.Walkthrough

CTT Cronbach's Alpha WALKTHROUGH:

Description

Walks the user through the set up and use of calculating the Cronbach's Alpha for an assessment using a given sample.

Usage

```
cttCronbachAlpha.Walkthrough(data)
```

Arguments

data	Can be either 1) An nS by nQ matrix or data frame of a dichotomous graded (0 or 1) sample, where nS is the number of students in the sample and nQ is the number of questions, 2) The output of piql.data.select with one or multiple courses selected.

Value

The Cronbach's Alpha calculation as specified in the walkthough.

Examples

```
# Use sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0)

# Begin walkthrough
cttCronbachAlpha.Walkthrough(temp.data)
```

---

cttDifficulty *CTT Difficulty of multiple courses with bootstrapping for error estimation and plotting:*

Description

Calculate Classical Test Theory item difficulty for the items on an assessment using a given sample.

Usage

```
cttDifficulty(data, booted = FALSE, nRuns = 100, plotBarChart = FALSE)
```

Arguments

data	Can be either 1) An nS by nQ matrix or data frame of a dichotomous graded (0 or 1) sample, where nS is the number of students in the sample and nQ is the number of questions, 2) The output of piql.data.select with one or multiple courses selected.
booted	Logical (default = FALSE). FALSE means no bootstrapping will be performed. TRUE turns on the bootstrapping feature.
nRuns	Number of random samples to use in the bootstrapping (default = 100). For publications it is recommended that 10,000 runs be performed since sample error goes as 1/sqrt(nRuns).
plotBarChart	(Default = FALSE). Generate barchart. If bootstrapping was performed, then error bars wll be automatically added (+- 1 standard deviation).

Value

When booted = FALSE (the default setting) then the straight calculated value will be returned. If booted = TRUE, the function will output the mean and standard deviation for the CTT item difficulties calculated from nRuns randomly sampled with replacement data sets from the given sample.

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0)
data.num <- temp.data$data.num

# Straight item difficulty.
cttDifficulty(data.num)

# Booted using 100 random samples (default)
cttDifficulty(data.num, booted = TRUE)

# Booted using 1000 random samples by manually setting nRuns
cttDifficulty(data.num, booted = TRUE, nRuns = 1000)
```

`cttDifficulty.Walkthrough` *CTT Difficulty WALKTHROUGH:*

Description

Walks the user through the set up and use of calculating Classical Test Theory item difficulty for the items on an assessment using a given sample.

Usage

```
cttDifficulty.Walkthrough(data)
```

Arguments

`data`	Can be either 1) An nS by nQ matrix or data frame of a dichotomous graded (0 or 1) sample, where nS is the number of students in the sample and nQ is the number of questions, 2) The output of piql.data.select with one or multiple courses selected.

Value

The CTT Difficulty analysis as specified in the walkthough

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0)

# Begin walkthrough
cttDifficulty.Walkthrough(temp.data)
```

`cttDiscrimination` *CTT Discrimination of multiple courses with bootstrapping for error estimation and plotting:*

Description

Calculate Classical Test Theory item discrimination for the items on an assessment using a given sample.

Usage

```
cttDiscrimination(
  data,
  perc = 0.27,
  as.Percentile = FALSE,
  booted = FALSE,
  nRuns = 100,
  plotBarChart = FALSE
)
```

Arguments

`data`	Can be either 1) An nS by nQ matrix or data frame of a dichotomous graded (0 or 1) sample, where nS is the number of students in the sample and nQ is the number of questions, 2) The output of piql.data.select with one or multiple courses selected.
`perc`	Top and bottom percentage to be compared (default = 0.27).
`as.Percentile`	Calculate item discrimination user percentiles as opposed to raw percentages.
`booted`	Logical (default = FALSE). FALSE means no bootstrapping will be performed. TRUE turns on the bootstrapping feature.
`nRuns`	Number of random samples to use in the bootstrapping (default = 100). For publications it is recommended that 10,000 runs be performed since sample error goes as 1/sqrt(nRuns).
`plotBarChart`	(Default = FALSE). Generate barchart. If bootstrapping was performed, then error bars wll be automatically added (+- 1 standard deviation).

Value

When booted = FALSE (the default setting) then the straight calculated value will be returned. If booted = TRUE, the function will output the mean and standard deviation for the CTT item discrimination calculated from nRuns randomly sampled with replacement data sets from the given sample.

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0)
data.num <- temp.data$data.num
```

```
# Straight item discrimination
cttDiscrimination(data.num)

# Booted using 100 random samples (default)
cttDiscrimination(data.num, booted = TRUE)

# Booted using 1000 random samples by manually setting nRuns
cttDiscrimination(data.num, booted = TRUE, nRuns = 1000)
```

`cttDiscrimination.Walkthrough` *CTT Discrimination WALKTHROUGH:*

Description

Walks the user through the set up and use of calculating Classical Test Theory item Discrimination for the items on an assessment using a given sample.

Usage

```
cttDiscrimination.Walkthrough(data)
```

Arguments

`data`	Can be either 1) An nS by nQ matrix or data frame of a dichotomous graded (0 or 1) sample, where nS is the number of students in the sample and nQ is the number of questions, 2) The output of piql.data.select with one or multiple courses selected.

Value

The CTT Discrimination analysis as specified in the walkthough

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0)

# Begin walkthrough
cttDiscrimination.Walkthrough(temp.data)
```

`cttGeneralAnalysis` *CTT General Analysis*

Description

Performs the following analysis: Cronbach's Alpha, Item point-biserial, Item difficulty, and Item discrimination.

Usage

```
cttGeneralAnalysis(
  data,
  perc = 0.27,
  as.Percentile = FALSE,
  booted = FALSE,
  nRuns = 100,
  plotBarChart = FALSE
)
```

Arguments

`data`	Can be either 1) An nS by nQ matrix or data frame of a dichotomous graded (0 or 1) sample, where nS is the number of students in the sample and nQ is the number of questions, 2) The output of piql.data.select with one or multiple courses selected.
`perc`	(Default = 0.27) percentage or percentile to be used in the analysis.
`as.Percentile`	(Default = FALSE) If true, then students are split on percentile not as percentage in total score.
`booted`	Logical (default = FALSE). FALSE means no bootstrapping will be performed. TRUE turns on the bootstrapping feature.
`nRuns`	Number of random samples to use in the bootstrapping (default = 100). For publications it is recommended that 10,000 runs be performed since sample error goes as 1/sqrt(nRuns).
`plotBarChart`	(Default = FALSE). Generate barchart. If bootstrapping was performed, then error bars wll be automatically added (+- 1 standard deviation).

Value

When booted = FALSE (the default setting) then the straight calculated value will be returned. If booted = TRUE, the function will output the mean and standard deviation for the CTT item difficulties calculated from nRuns randomly sampled with replacement data sets from the given sample.

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0)
data.num <- temp.data$data.num.111
```

```
# Input can be the score matrix
cttGeneralAnalysis(data.num)
# or the output from piql.data.select()
cttGeneralAnalysis(temp.data)
# Supports Bootstrapping of all statistics
cttGeneralAnalysis(data.num, booted = TRUE)
# And will plot the results (with error bars if bootstrapped)
cttGeneralAnalysis(data.num, plotBarChart = TRUE)
cttGeneralAnalysis(data.num, booted = TRUE, plotBarChart = TRUE)
```

`cttGeneralAnalysis.Walkthrough` *CTT General Analysis WALKTHROUGH:*

Description

Walks the user through the set up and use a general CTT analysis of the assessment using a given sample.

Usage

```
cttGeneralAnalysis.Walkthrough(data)
```

Arguments

`data`	Can be either 1) An nS by nQ matrix or data frame of a dichotomous graded (0 or 1) sample, where nS is the number of students in the sample and nQ is the number of questions, 2) The output of piql.data.select with one or multiple courses selected.

Value

The Cronbach's Alpha, Item Difficulty, Item Discrimination, and Item Point-biserial statistics plus plots as specified in the walkthough.

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0)

# Begin walkthrough
cttGeneralAnalysis.Walkthrough(temp.data)
```

`cttpointBiserial` *CTT Point-biserial:*

Description

Calculates the point-biserial correlation between the total score (with focus item removed) and the focus item. This gives a measure if the items on the assessment are all measuring a single similar latent trait. If the point-biserial is poor, then this could be an indication that the assessment is multidimensional.

Usage

```
cttpointBiserial(data, booted = FALSE, nRuns = 100, plotBarChart = FALSE)
```

Arguments

`data`	Can be either 1) An nS by nQ matrix or data frame of a dichotomous graded (0 or 1) sample, where nS is the number of students in the sample and nQ is the number of questions, 2) The output of piql.data.select with one or multiple courses selected.
`booted`	Logical (default = FALSE). FALSE means no bootstrapping will be performed. TRUE turns on the bootstrapping feature.
`nRuns`	Number of random samples to use in the bootstrapping (default = 100). For publications it is recommended that 10,000 runs be performed since sample error goes as 1/sqrt(nRuns).
`plotBarChart`	(Default = FALSE). Generate barchart. If bootstrapping was performed, then error bars wll be automatically added (+- 1 standard deviation).

Value

Point-biserial correlation between the total score (with focus item removed) and the focus item. For all items on the assessment

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0)
data.num <- temp.data$data.num

# Straight item difficulty.
cttpointBiserial(data.num)

# Booted using 100 random samples (default)
cttpointBiserial(data.num, booted = TRUE)

# Booted using 1000 random samples by manually setting nRuns
cttpointBiserial(data.num, booted = TRUE, nRuns = 1000)
```

cttpointBiserial.internal

CTT Point-biserial:

Description

Calculates the point-biserial correlation between the total score (with focus item removed) and the focus item. This gives a measure if the items on the assessment are all measuring a single similar latent trait. If the point-biserial is poor, then this could be an indication that the assessment is multidimensional.

Usage

```
cttpointBiserial.internal(data, booted = FALSE, nRuns = 100)
```

Arguments

data	An nS by nQ matrix or data frame of a dichotomous graded (0 or 1) sample, where nS is the number of students in the sample and nQ is the number of questions.
booted	Logical (default = FALSE). FALSE means no bootstrapping will be performed. TRUE turns on the bootstrapping feature.
nRuns	Number of random samples to use in the bootstrapping (default = 100). For publications it is recommended that 10,000 runs be performed since sample error goes as 1/sqrt(nRuns).

Value

Point-biserial correlation between the total score (with focus item removed) and the focus item. For all items on the assessment

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0)
data.num <- temp.data$data.num

# Point-biserial
cttpointBiserial.internal(data.num)
```

cttPointBiserial.Walkthrough

CTT Point-biserial WALKTHROUGH:

Description

Walks the user through the set up and use of calculating Classical Test Theory item Point-biserial for the items on an assessment using a given sample.

Usage

```
cttPointBiserial.Walkthrough(data)
```

Arguments

data	Can be either 1) An nS by nQ matrix or data frame of a dichotomous graded (0 or 1) sample, where nS is the number of students in the sample and nQ is the number of questions, 2) The output of piql.data.select with one or multiple courses selected.

Value

The CTT Point-biserial analysis as specified in the walkthough

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0)

# Begin walkthrough
cttPointBiserial.Walkthrough(temp.data)
```

---

data.select.Walkthrough

Select specific course from the pulled data - Walkthrough.

Description

Guides the user through selecting the correct data from the pulled data from either AWS or from their own computer.

Usage

```
data.select.Walkthrough(pulled.data)
```

Arguments

pulled.data	Output from pullPQLdata or pullData or import.from.computer

Value

A list of course data:

$data.alpha ALPHABETICAL data for the selected course.

$data.num NUMERICAL data for the selected course.

$nS.details Details about how many students were removed due to the specified numBlanks.allowed value.

Examples

```
# Select data via walkthrough using data pulled
# from AWS or personal computer.
data.select.Walkthrough(pulled.data)
```

`dif.analysis` *Differential Item Functions analysis*

Description

Performs a Differential Item Functioning (DIF) analysis on the given set of data.

Usage

```
dif.analysis(
  data,
  booted = FALSE,
  nRuns = 5,
  purify = TRUE,
  p.adjust.method = "BH"
)
```

Arguments

`data`	Must be from the piql.data.select or data.select.Walkthrough functions for multiple courses.
`booted`	Logical (default = FALSE). FALSE means no bootstrapping will be performed. TRUE turns on the bootstrapping feature.
`nRuns`	Number of random samples to use in the bootstrapping (default = 100). For publications it is recommended that 10,000 runs be performed since sample error goes as 1/sqrt(nRuns).

Value

Performs a Differential Item Functioning (DIF) analysis on the given set of data.

For details on DIF analysis see the following papers:

Magis et al. "A general framework and an R package for the detection of dichotomous differential item functioning"

Meade "A Taxonomy of Effect Size Measures for the Differential Functioning of Items and Scales"

Chalmers et al. "It Might Not Make a Big DIF: Improved Differential Test Functioning Statistics That Account for Sampling Variability"

Examples

```
# IN PROCESS
```

diff.ita *Item Tree Analysis: Error in reconstructed condtradiction matrix*

Description

Calculates error between the reconstructed and original contradiction matrices. Goal in ITA is to get this to be as small as possible.

Usage

```
## S3 method for class 'ita'
diff(contradiction.mat.orig, contradiction.mat.cur)
```

Arguments

contradiction.mat.orig	Output from the contradiction.mat.generator() function. Original contradiction matrix for the data.
contradiction.mat.cur	Current estimated contradication matrix. Calculated differently depending on ITA method: Original, Corrected, or Monimized Corrected.

Value

The error (similar to a Chi^2) in the estimated contradiction matrix for the current tree.

efa.walkthrough *Walkthrough of an Exploratory Factor Analysis*

Description

Uses prompts to walks the user through an EFA of the given data.

Usage

```
efa.walkthrough(data)
```

Arguments

data	Can be either 1) An nS by nQ matrix or data frame of a dichotomous graded (0 or 1) sample, where nS is the number of students in the sample and nQ is the number of questions, 2) The output of piql.data.select with one or multiple courses selected.

Value

Walks user through an EFA of the data.

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0)
data.num <- temp.data$data.num.111

Begin walkthrough
efa.walkthrough(data.num)
```

`ensure_library` *Choose Save Location*

Description

Opens a directory GUI to help the user select a save location.

Usage

```
ensure_library(lib.name)
```

Value

Enables users to select directories for saving analysis results.

`error.rate.minCor` *Item Tree Analysis: Error Rate for minimized corrected ITA*

Description

Calculates the minimized corrected error rate between the links carried in the current tree and the contradiction matrix.

Usage

```
error.rate.minCor(current.tree, condradiction.mat, p, nS)
```

Arguments

`current.tree`	Output from the itaForest() function. Current tree being considered.
`condradiction.mat`	Output from the contradiction.mat.generator() function. Contradiction matrix for the data.
`p`	CTT Item Difficulty.
`nS`	Number of students in the data.

Value

Error rate for the current tree using the minimized corrected method.

error.rate.orig *Item Tree Analysis: Error Rate for original and corrected ITA*

Description

Calculates the error rate between the links carried in the current tree and the contradiction matrix.

Usage

```
error.rate.orig(current.tree, condradiction.mat, p, nS)
```

Arguments

current.tree	Output from the itaForest() function. Current tree being considered.
condradiction.mat	Output from the contradiction.mat.generator() function. Contradiction matrix for the data.
p	CTT Item Difficulty.
nS	Number of students in the data.

Value

Error rate for the current tree.

export.as.excel *Export to Spreadsheet*

Description

Export the outputs of simplyPsycho function as an Excel Spreadsheet

Usage

```
export.as.excel(to.be.exported)
```

Arguments

to.be.exported	Output from one of simplyPsycho's analysis functions.

Value

Saves an .xlsx to the current working directory.

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0)
results <- cttDifficulty(temp.data)

export.as.excel(results,"filename.xlsx")
```

`gen.compare.courses` *Compare courses*

Description

Use a t-test to compare total scores between two courses, and ANOVA to compate three or more courses.

NOTE: Use the Left and Right arrow in the Plots window to switch between the plots.

Usage

```
gen.compare.courses(
  data,
  makeBoxPlots = FALSE,
  makeDiffPlots = FALSE,
  makeSankeyPlots = FALSE,
  nBins = 7,
  nRuns = 100,
  return.gains = FALSE,
  makeGainsPlots = FALSE,
  ...
)
```

Arguments

`data`	MUST BE the output of piql.data.select for multiple courses selected.
`makeBoxPlots`	(Default = FALSE) If TRUE, then bar plots will be made for all given courses in a single plot. NOTE: Use the Left and Right arrow in the Plots window to switch between the plots.
`makeDiffPlots`	(Default = FALSE) If TRUE, then point plots will be made for comparing all possible course pairs. NOTE: Use the Left and Right arrow in the Plots window to switch between the plots.
`makeSankeyPlots`	(Default = FALSE) If TRUE, then Sankey plots will be made for comparing all possible course pairs. NOTE: Use the Left and Right arrow in the Viewer window to switch between the plots.
`nBins`	Number of total score bins to use for the Sankey plots. Bins are roughly "equal" ranges of total score. ~(number of total scores possible)/nBins
`nRuns`	Number of runs to use when calculating error in CTT Difficulty.
`return.gains`	(Default = FALSE) If FALSE, the gains are not returned for each student. IF TRUE, then gains are reported for each student.
`makeGainsPlots`	(Default = FALSE) If true, then plots of normalized gain/change versus "pre" scores are created.

Value

Cohen's d with interpretation, p-value and adjusted p-value of difference in means, course normalized gain, difference in item difficulty with error. On request box plots, change in difficulty plots, Sankey plots, and gains versus pre score plots cal all be created. Sankey plots and gains versus pre score plots require matched data.

NOTE: Use the Left and Right arrow in the Plots window to switch between the plots.

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = c(1,2), numBlanks.allowed = 0, Matched = TRUE)
gen.compare.courses(temp.data,  makeSankeyPlots = TRUE)

gen.compare.courses(data)
## Box Plots can be made
gen.compare.courses(temp.data, makeBoxPlots = TRUE)
## Change in item difficulty pots can be made
gen.compare.courses(temp.data, makeDiffPlots = TRUE)

### MATCHED DATA
## Sankey plots can be made
gen.compare.courses(temp.data, makeSankeyPlots = TRUE)
## Plots of normalized gain/change versus pre score for each student
gen.compare.courses(temp.data, makeGainsPlots = TRUE)
## You can get the gain values by setting return.gains = TRUE
gen.compare.courses(temp.data, return.gains = TRUE)
```

---

gen.compare.courses.Walkthrough

General Comparison of Courses WALKTHROUGH:

Description

Walks the user through a general comparision of courses in the given sample.

Usage

```
gen.compare.courses.Walkthrough(data)
```

Arguments

data	Can be either 1) An nS by nQ matrix or data frame of a dichotomous graded (0 or 1) sample, where nS is the number of students in the sample and nQ is the number of questions, 2) The output of piql.data.select with one or multiple courses selected.

Value

Cohen's d with interpretation, p-value and adjusted p-value of difference in means, course normalized gain, difference in item difficulty with error. On request box plots, change in difficulty plots, Sankey plots, and gains versus pre score plots cal all be created. Sankey plots and gains versus pre score plots require matched data.

NOTE: Use the Left and Right arrow in the Plots window to switch between the plots.

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = c(1,2), numBlanks.allowed = 0, Matched = TRUE)

# Begin walkthrough
gen.compare.courses.Walkthrough(temp.data)
```

---

gen.model.TotalScores *General fitting of total scores with a normal distribution.*

Description

Fits a normal distribution to the total scores.

Usage

```
gen.model.TotalScores(data, model = "Normal", makePlot = FALSE)
```

Arguments

data	Can be either 1) An nS by nQ matrix or data frame of a dichotomous graded (0 or 1) sample, where nS is the number of students in the sample and nQ is the number of questions, 2) The output of piql.data.select with one or multiple courses selected.
model	(Default = "Normal") Model = "Normal", the mean, standard deviation, skew, kertosis and standard error will be estimated for the given course(s). Model = "Bimodal" will also report the possible location of the two mean values assuming the sample is drawn from a bimodal distribution. Model = "Johnson" will use a Johnson fit method to fit the data including the skew and kurtosis.
makePlot	(Default = FALSE). If TRUE, then a single plot of the fraction of students with each possible total score and the estimated normal distribution for each courses in the data will be created.

Value

Mean, Standard deviation, and optional plot. Additionally, if model = "Bimodal" then the estimated locaitons of maximums (means) assuming a bimodal distribution will be reported.

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0)
data.num <- temp.data$data.num
# -------------------------- #
# For multiple courses
data.multiple <- piql.data.select(simplySampleData, courses = c(1,2), numBlanks.allowed = 0, Matched = TRUE)

## Single Course
# Total score means and standard deviations...
gen.model.TotalScores(data.num)
# ...  with optional plotting.
gen.model.TotalScores(data.num, makePlot = TRUE)
# Supports bimodal estimations as well
gen.model.TotalScores(data.num, model = "Bimodal", makePlot = TRUE)

# Total score means and standard deviations...
gen.model.TotalScores(data.multiple)
# ...  with optional plotting.
gen.model.TotalScores(data.multiple, makePlot = TRUE)
# Supports bimodal estimations as well
gen.model.TotalScores(data.multiple, model = "Bimodal", makePlot = TRUE)
```

---

general.compare.courses.MakeSankey

Makes Sankey plot

Description

Makes Sankey plot for gen.compare.course

Usage

```
general.compare.courses.MakeSankey(
  set1,
  set2,
  nQ,
  nBins = 7,
  couse.names,
  ii,
  jj
)
```

---

import.from.computer

Import data from computer

Description

Select files from your computer to be used in siplyPsycho. The function asks for descriptive information about the files (student data or answer key). If student data, then it asks for the course it is from and for the term. formatting is checked via build.psycho.datafile. See that function's documentation for the formatting extected for student data.

Usage

```
import.from.computer()
```

Value

Data to be used in simplyPsycho with correct formatting.

Examples

```
# Pull in from your computer
## Data must be formatted correctly - see documentation.

import.from.computer()

# Follow the prompts.
# Data should be formatted in the same manner as simplySampleData01 available
# in the simplyPsycho Github repository (https://github.com/PhilipEaton/simplyPsycho).
Read in file from computer and pull the last letters to check file type.
```

---

internal.pointBiserial

CTT Point-biserial:

Description

Calculates the point-biserial correlation between the total score (with focus item removed) and the focus item. This gives a measure if the items on the assessment are all measuring a single similar latent trait. If the point-biserial is poor, then this could be an indication that the assessment is multidimensional.

Usage

```
internal.pointBiserial(data)
```

Arguments

data	An nS by nQ matrix or data frame of a dichotomous graded (0 or 1) sample, where nS is the number of students in the sample and nQ is the number of questions.

Value

Point-biserial correlation between the total score (with focus item removed) and the focus item. For all items on the assessment

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0)
data.num <- temp.data$data.num

# Point-biserial
pointBiserial(data.num)
```

irc.get *Item Response Curve data generator:*

Description

Calculates the plotting data needed to create item response curves for the items on an assessment using a given sample.

Usage

```
irc.get(data.Alpha, data.Num, nO = NULL)
```

Arguments

data.Alpha	An nS by nQ matrix or data frame of ALPHABETICAL data, where nS is the number of students in the sample and nQ is the number of questions.
data.Num	An nS by nQ matrix or data frame of a dichotomous graded (0 or 1) sample, where nS is the number of students in the sample and nQ is the number of questions.
nO	The maximum number of options available on the assessment. Default it set to find this value automatically given the response options present in the sample. It is suggested you enter than value manually if the max number of options is known. For example, if you know item 7 has 6 options, the most out of all the items on the assessment, then set nO = 7.

Value

Plotting data for item response curves. To be put into the ircPlot() function.

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0)
data.alpha <- temp.data$data.alpha
data.num <- temp.data$data.num

# get irc data and plot item 2
irc.data <- irc.get(data.alpha, data.num)
irc.plot(irc.data, 2)
```

irc.get.Booted *Item Response Curve data generator with bootstrapping for error estimation:*

Description

Calculates the plotting data needed to create item response curves for the items on an assessment using a given sample and bootstrapping. Can be used to plot item response curves with error bars/bounds.

Usage

```
irc.get.Booted(data.alpha, data.num, nO = NULL, nRuns = 10)
```

Arguments

data.alpha	An nS by nQ matrix or data frame of ALPHABETICAL data, where nS is the number of students in the sample and nQ is the number of questions.
data.num	An nS by nQ matrix or data frame of a dichotomous graded (0 or 1) sample, where nS is the number of students in the sample and nQ is the number of questions.
nO	The maximum number of options available on the assessment. Default it set to find this value automatically given the response options present in the sample. NOTE: This should almost never need to be changed from its default value!
nRuns	Number of random samples to use in the bootstrapping (default = 10). For publications it is recommended that 10,000 runs be performed since sample error goes as 1/sqrt(nRuns).

Value

Plotting data for item response curves. To be put into the ircPlotErrorBars() or ircPlotErrorBounds() functions.

$means The mean values from bootstrapping.

$stDevs The standard deviation values from bootstrapping.

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0)
data.alpha <- temp.data$data.alpha
data.num <- temp.data$data.num

# Get irc data with default values.
irc.data.booted <- irc.get.Booted(data.alpha,data.num)
irc.data.booted$means # Mean values
irc.data.booted$stDevs # Standard deviation values.

# Plot item 2.
irc.plot.withErrorBars(irc.data.booted, 2)
irc.plot.withErrorBounds(irc.data.booted,2)

# Get irc data and plot item 2 with a larger bootstrapping
irc.data.booted <- irc.get.Booted(data.alpha,data.num, nRuns = 100)
irc.plot.withErrorBars(irc.data.booted, 2)
irc.plot.withErrorBounds(irc.data.booted,2)
```

irc.plot *Item Response Curve plotter:*

Description

Plots the result of irc.get().

Usage

```
irc.plot(irc.data, qq)
```

Arguments

irc.data	Output from irc.get().
qq	Question number you would like plotted.

Value

IRC plot for the requested item using the given irc.get() output.

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0)
data.alpha <- temp.data$data.alpha
data.num <- temp.data$data.num

# get irc data and plot item 2
irc.data <- irc.get(data.alpha, data.num)
irc.plot(irc.data, 2)
```

irc.plot.withErrorBars *Plot Item Response Curves with error bars:*

Description

Given the output from irc.get.Booted() function, create an IRC plot with error bars for a specified item.

Usage

```
irc.plot.withErrorBars(irc.data.from.booted, qq)
```

Arguments

irc.data.from.booted	output from irc.get.Booted() function.
qq	Question number for the generated plot.

Value

IRC plot with error bars for item number qq.

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0)
data.alpha <- temp.data$data.alpha
data.num <- temp.data$data.num

# Get irc data with default values.
irc.data.booted <- irc.get.Booted(data.alpha,data.num)
# Plot item 2.
irc.plot.withErrorBars(irc.data.booted, 2)
# Plot item 4.
irc.plot.withErrorBounds(irc.data.booted, 4)
```

---

irc.plot.withErrorBounds

Plot Item Response Curves with error bounds:

Description

Given the output from irc.get.Booted() function, create an IRC plot with error bounds for a specified item.

NOTE ABOUT WARNING MESSAGES: You will likely get a warning like "zero-length arrow is of indeterminate angle and so skipped". This can be ignored as it does not impact the plotting.

Usage

```
irc.plot.withErrorBounds(irc.data.from.booted, qq)
```

Arguments

irc.data.from.booted	output from irc.get.Booted() function.
qq	Question number for the generated plot.

Value

IRC plot with error bounds for item number qq.

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0)
data.alpha <- temp.data$data.alpha
data.num <- temp.data$data.num

# Get irc data with default values.
irc.data.booted <- irc.get.Booted(data.alpha,data.num)
# Plot item 2.
```

```
irc.plot.withErrorBars(irc.data.booted, 2)
# Plot item 4.
irc.plot.withErrorBounds(irc.data.booted, 4)
```

`irt2PL` *Item Response theory 2-parameter logistic model with bootstrapping for error estimation:*

Description

Calculate Item Response Theory (IRT) 2-parameter logistic model (2PL) for given sample. Estimation of the error in item parameters can be found using 2 methods.

Method 1: Bootstrap the data. Estimate errors in item parameters by bootstrapping sample. This given population level error estimations.

Method 2: Randomly estimate the initial parameters in mirt. Estimate errors in item parameters by wiggling the initial parameters. This given sample level error estimations.

WARNING: IRT is in general a very computationally expensive process, meaning it is slow. It is recommended you begin with 5 - 10 runs to get a feel for what the data is doing, and then upping to a larger number of runs once you are sure the output is giving you what you want.

Usage

```
irt2PL(data.num, Method = 1, nRuns = 5)
```

Arguments

`data.num`	An nS by nQ matrix or data frame of a dichotomous graded (0 or 1) sample, where nS is the number of students in the sample and nQ is the number of questions.
`Method`	(Default = 1) Method = 1 does traditional bootstrapping for estimations of population-level error. Method = 2 does initial parameter randomization for estimations of sample-level error.
`nRuns`	Number of random samples to use in the bootstrapping (default = 5). For publications it is recommended that 10,000 runs be performed since sample error goes as 1/sqrt(nRuns).

Value

Model fit and item parameter estimation mean and standard deviations for the sample.

$fit.means Model fit mean values from bootstrapping.

$fir.stDevs Model fit standard deviation values from bootstrapping.

$item.means Item parameter mean values from bootstrapping.

$item.stDevs Item parameter standard deviation values from bootstrapping.

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0)
data.alpha <- temp.data$data.alpha
data.num <- temp.data$data.num

# Get irc data with default values.
irt2PL.Method1.res <- irt2PL(data.num)
irt2PL.Method1.res$fit.means
irt2PL.Method1.res$fit.stDevs
irt2PL.Method1.res$item.means
irt2PL.Method1.res$item.stDevs

# Change Method and number of runs
irt2PL.Method2.res <- irt2PL(data.num, Method = 2, nRuns = 10)
irt2PL.Method2.res$fit.means
irt2PL.Method2.res$fit.stDevs
irt2PL.Method2.res$item.means
irt2PL.Method2.res$item.stDevs
```

---

ita.cycle.killer *Item Tree Analysis - Cycle Killer*

Description

TBA

Usage

```
ita.cycle.killer()
```

Value

A plot after killing the detected cycle... hopefully

Examples

```
# TBA
```

---

ita.fullCon *Item Tree Analysis - Full Control*

Description

Performs an ITA where the user gets to control the max amount of contractions between a causal pairing to be retained in the model.

This file does item tree analysis with bootstrapping for error estimation: Relaxed the code to so that models will always 1) includes all of the items on the instrument, and 2) select the model with the best diff from those models.

Not really recommend for publications boasting quantitative robustness. This method is EXTREMELY adhoc and should only be used for qualitative analysis purposes

Usage

```
ita.fullCon(data, perc = 0.2, method = 1)
```

Arguments

data	An nS by nQ matrix or data frame of a dichotomous graded (0 or 1) sample, where nS is the number of students in the sample and nQ is the number of questions.
perc	(Default = 0.20) Percentage of either the number of students or the max number of contradictions to be retained in the model.
method	(Defait = 1) Method 1 is based on the number of students and will be the same for every bootstrapped model. Method 2 is based on the maximum number on contradictions in the initial model and can vary a little between random samples.

Value

ITA map to be plotted using itaPlot() function.

$model Model found from analysis.

$diff Diff for the model.

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0, MCMR.grading = "Selected")
data.num.MCMR <- temp.data$data.num

# Fully Controlled ITA
ita.data.fc <- ita.relaxed(data.num.MCMR)
ita.data.fc$model
ita.data.fc$diff

## PLOTTIING
# Set labels
labs <- c("Plants", "Miner", "Fish", "Hooke's Law", "Ferris",
"Inv. g", "JogAB", "SphBottle", "mkp", "Slide", "Odometer",
"Squareness", "Olive Oil", "Ch.Sph.", "Int. E", "Q&N", "Bhutan-A",
"Bhutan-C", "Bhutan-D", "Work-D", "Work-G", "EField", "Delta v")

# and plot
irt.Plot(ita.data.fc$model, labs = labs)
```

---

ita.fullCon.booted *Item Tree Analysis - Full Control with bootstrapping*

Description

Performs an ITA where the user gets to control the max amount of contractions between a causal pairing to be retained in the model.

This file does item tree analysis with bootstrapping for error estimation: Relaxed the code to so that models will always 1) includes all of the items on the instrument, and 2) select the model with the best diff from those models.

Not really recommend for publications boasting quantitative robustness. This method is EXTREMELY adhoc and should only be used for qualitative analysis purposes

Usage

```
ita.fullCon.booted(
  data,
  perc = 0.2,
  method = 1,
  retain.perc = 0.9,
  nRuns = 10,
  quite = FALSE
)
```

Arguments

data	An nS by nQ matrix or data frame of a dichotomous graded (0 or 1) sample, where nS is the number of students in the sample and nQ is the number of questions.
perc	(Default = 0.20) Percentage of either the number of students or the max number of contradictions to be retained in the model.
method	(Defait = 1) Method 1 is based on the number of students and will be the same for every bootstrapped model. Method 2 is based on the maximum number on contradictions in the initial model and can vary a little between random samples.
retain.perc	(Default = 0.90) Retain the links that are present in this percentage of all the bootstrapped runs.
nRuns	Number of random samples to use in the bootstrapping (default = 10). For publications it is recommended that 10,000 runs be performed since sample error goes as 1/sqrt(nRuns).

Value

ITA map resulting from bootstrapping to be plotted using itaPlot() function.

Examples

```
# Get PIQL data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0, MCMR.grading = "Selected")
data.num.MCMR <- temp.data$data.num
# ITA with bootstrapping
ita.data.fc.booted <- ita.fullCon.booted(data.num.MCMR, type = 3, nRuns = 3)

## PLOTTIING
# Set labels
labs <- c("Plants", "Miner", "Fish", "Hooke's Law", "Ferris",
          "Inv. g", "JogAB", "SphBottle", "mkp", "Slide", "Odometer",
          "Squareness", "Olive Oil", "Ch.Sph.", "Int. E", "Q&N", "Bhutan-A",
          "Bhutan-C", "Bhutan-D", "Work-D", "Work-G", "EField", "Delta v")
# Plot bootstrapped model
ita.Plot(ita.data.fc.booted, labs = labs)
```

ita.Plot *Item Tree Analysis - Create a Plot*

Description

Creates a plot given a ITA tree

Usage

```
ita.Plot(current.tree, labs = NULL)
```

Arguments

current.tree	The tree to be plotted
labs	Labels for the tree, if any.

Value

Hesse Plot of the current tree.

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0, MCMR.grading = "Selected")
data.alpha.MCMR <- temp.data$data.alpha
data.num.MCMR <- temp.data$data.num
# Check number of student removed. Should be less than 10%.
temp.data$nS.details

# Original ITA
ita.data.1 <- ita.strict(data.num.MCMR,type = 1)
ita.data.1$model
ita.data.1$diff
ita.data.1$errorRate

# Corrected ITA
ita.data.2 <- ita.strict(data.num.MCMR,type = 2)
ita.data.2$model
ita.data.2$diff
ita.data.2$errorRate

# Minimized Corredted ITA
ita.data.3 <- ita.strict(data.num.MCMR,type = 3)
ita.data.3$model
ita.data.3$diff
ita.data.3$errorRate

## PLOTTIING
# Set labels
labs <- c("Plants", "Miner", "Fish", "Hooke's Law", "Ferris",
"Inv. g", "JogAB", "SphBottle", "mkp", "Slide", "Odometer",
"Squareness", "Olive Oil", "Ch.Sph.", "Int. E", "Q&N", "Bhutan-A",
"Bhutan-C", "Bhutan-D", "Work-D", "Work-G", "EField", "Delta v")
```

```
# and plot
irt.Plot(ita.data.1$model, labs = labs)
irt.Plot(ita.data.2$model, labs = labs)
irt.Plot(ita.data.3$model, labs = labs)
```

`ita.relaxed` *Item Tree Analysis - Relaxed*

Description

Performs a relaxed ITA. This function is set up to consider every sensible tree by first considering all possible connections in the assessment, then iteratively removing the worst offending connection, one at a time, until no connections are left. The returned model is the one which returns the smallest difference between the estimated and original contradiction matrices AND retains all of the items on the assessment.

Usage

```
ita.relaxed(data, type = 1)
```

Arguments

`data`	An nS by nQ matrix or data frame of a dichotomous graded (0 or 1) sample, where nS is the number of students in the sample and nQ is the number of questions.
`type`	(Default = 1) Type of ITA to be performed. Type = 1 - Original IITA Type = 2 - Corrected IITA Type = 3 - Minimized Corrected IITA

Value

ITA map to be plotted using itaPlot() function.

$model Model found from analysis.

$diff Diff for the model.

$errorRate Error Rate for the model.

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0, MCMR.grading = "Selected")
data.alpha.MCMR <- temp.data$data.alpha
data.num.MCMR <- temp.data$data.num
# Check number of student removed. Should be less than 10%.
temp.data$nS.details

# Original ITA
ita.data.1 <- ita.relaxed(data.num.MCMR,type = 1)
ita.data.1$model
ita.data.1$diff
ita.data.1$errorRate
```

```
# Corrected ITA
ita.data.2 <- ita.relaxed(data.num.MCMR,type = 2)
ita.data.2$model
ita.data.2$diff
ita.data.2$errorRate

# Minimized Corredted ITA
ita.data.3 <- ita.relaxed(data.num.MCMR,type = 3)
ita.data.3$model
ita.data.3$diff
ita.data.3$errorRate

## PLOTTIING
# Set labels
labs <- c("Plants", "Miner", "Fish", "Hooke's Law", "Ferris",
"Inv. g", "JogAB", "SphBottle", "mkp", "Slide", "Odometer",
"Squareness", "Olive Oil", "Ch.Sph.", "Int. E", "Q&N", "Bhutan-A",
"Bhutan-C", "Bhutan-D", "Work-D", "Work-G", "EField", "Delta v")

# and plot
irt.Plot(ita.data.1$model, labs = labs)
irt.Plot(ita.data.2$model, labs = labs)
irt.Plot(ita.data.3$model, labs = labs)
```

`ita.relaxed.booted` *Item Tree Analysis - Relaxed with bootstrapping*

Description

Performs a relaxed ITA. This function is set up to consider every sensible tree by first considering all possible connections in the assessment, then iteratively removing the worst offending connection, one at a time, until no connections are left. The returned model is the one which returns the smallest difference between the estimated and original contradiction matrices AND retains all of the items on the assessment.

Usage

```
ita.relaxed.booted(
  data,
  type = 1,
  retain.perc = 0.9,
  nRuns = 10,
  quite = FALSE
)
```

Arguments

`data`	An nS by nQ matrix or data frame of a dichotomous graded (0 or 1) sample, where nS is the number of students in the sample and nQ is the number of questions.

type	(Default = 1) Type of ITA to be performed. Type = 1 - Original IITA Type = 2 - Corrected IITA Type = 3 - Minimized Corrected IITA
retain.perc	(Default = 0.90) Retain the links that are present in this percentage of all the bootstrapped runs.
nRuns	Number of random samples to use in the bootstrapping (default = 10). For publications it is recommended that 10,000 runs be performed since sample error goes as 1/sqrt(nRuns).

Value

ITA map resulting from bootstrapping to be plotted using itaPlot() function.

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0, MCMR.grading = "Selected")
data.num.MCMR <- temp.data$data.num
# ITA with bootstrapping
ita.data.3.booted <- ita.relaxed.booted(data.num.MCMR, type = 3, nRuns = 3)

## PLOTTIING
# Set labels
labs <- c("Plants", "Miner", "Fish", "Hooke's Law", "Ferris",
          "Inv. g", "JogAB", "SphBottle", "mkp", "Slide", "Odometer",
          "Squareness", "Olive Oil", "Ch.Sph.", "Int. E", "Q&N", "Bhutan-A",
          "Bhutan-C", "Bhutan-D", "Work-D", "Work-G", "EField", "Delta v")
# Plot bootstrapped model
ita.Plot(ita.data.3.booted, labs = labs)
```

---

ita.strict *Item Tree Analysis - Strictly Performed*

Description

Performs ITA using strict procedures. This function is set up to consider every sensible tree by first considering all possible connections in the assessment, then iteratively removing the worst offending connection, one at a time, until no connections are left. The returned model is the one which returns the smallest difference between the estimated and original contradiction matrices.

NOTE: The generally means a handful of questions get likely be removed entirely from the tree due to high diff and error rates in the models that include them.

SUGGESTION: A less strict version of this function is itaRelaxed and itxVeryRelaxed.

Usage

```
ita.strict(data, type = 1)
```

Arguments

data	An nS by nQ matrix or data frame of a dichotomous graded (0 or 1) sample, where nS is the number of students in the sample and nQ is the number of questions.
type	(Default = 1) Type of ITA to be performed. Type = 1 - Original IITA Type = 2 - Corrected IITA Type = 3 - Minimized Corrected IITA

Value

ITA map to be plotted using itaPlot() function.

$model Model found from analysis.

$diff Diff for the model.

$errorRate Error Rate for the model.

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0, MCMR.grading = "Selected")
data.alpha.MCMR <- temp.data$data.alpha
data.num.MCMR <- temp.data$data.num
# Check number of student removed. Should be less than 10%.
temp.data$nS.details

# Original ITA
ita.data.1 <- ita.strict(data.num.MCMR,type = 1)
ita.data.1$model
ita.data.1$diff
ita.data.1$errorRate

# Corrected ITA
ita.data.2 <- ita.strict(data.num.MCMR,type = 2)
ita.data.2$model
ita.data.2$diff
ita.data.2$errorRate

# Minimized Corredted ITA
ita.data.3 <- ita.strict(data.num.MCMR,type = 3)
ita.data.3$model
ita.data.3$diff
ita.data.3$errorRate

## PLOTTIING
# Set labels
labs <- c("Plants", "Miner", "Fish", "Hooke's Law", "Ferris",
"Inv. g", "JogAB", "SphBottle", "mkp", "Slide", "Odometer",
"Squareness", "Olive Oil", "Ch.Sph.", "Int. E", "Q&N", "Bhutan-A",
"Bhutan-C", "Bhutan-D", "Work-D", "Work-G", "EField", "Delta v")

# and plot
irt.Plot(ita.data.1$model, labs = labs)
irt.Plot(ita.data.2$model, labs = labs)
irt.Plot(ita.data.3$model, labs = labs)
```

`ita.strict.booted` *Item Tree Analysis - Strictly Performed with bootstrapping*

Description

Performs ITA using strict procedures with bootstrapping This function is set up to consider every sensible tree by first considering all possible connections in the assessment, then iteratively removing the worst offending connection, one at a time, until no connections are left. The returned model is the one which returns the smallest difference between the estimated and original contradiction matrices.

NOTE: The generally means a handful of questions get likely be removed entirely from the tree due to high diff and error rates in the models that include them.

SUGGESTION: A less strict version of this function is itaRelaxed and itxVeryRelaxed.

Usage

```
ita.strict.booted(data, type = 1, retain.perc = 0.9, nRuns = 10, quite = FALSE)
```

Arguments

`data`	An nS by nQ matrix or data frame of a dichotomous graded (0 or 1) sample, where nS is the number of students in the sample and nQ is the number of questions.
`type`	(Default = 1) Type of ITA to be performed. Type = 1 - Original IITA Type = 2 - Corrected IITA Type = 3 - Minimized Corrected IITA
`retain.perc`	(Default = 0.90) Retain the links that are present in this percentage of all the bootstrapped runs.
`nRuns`	Number of random samples to use in the bootstrapping (default = 10). For publications it is recommended that 10,000 runs be performed since sample error goes as 1/sqrt(nRuns).

Value

ITA map resulting from bootstrapping to be plotted using itaPlot() function.

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0, MCMR.grading = "Selected")
data.num.MCMR <- temp.data$data.num
# ITA with bootstrapping
ita.data.3.booted <- ita.strict.booted(data.num.MCMR, type = 3, nRuns = 3)

## PLOTTIING
# Set labels
labs <- c("Plants", "Miner", "Fish", "Hooke's Law", "Ferris",
          "Inv. g", "JogAB", "SphBottle", "mkp", "Slide", "Odometer",
          "Squareness", "Olive Oil", "Ch.Sph.", "Int. E", "Q&N", "Bhutan-A",
```

```
          "Bhutan-C", "Bhutan-D", "Work-D", "Work-G", "EField", "Delta v")
# Plot bootstrapped model
ita.Plot(ita.data.3.booted, labs = labs)
```

`MCMR.response.analysis`

MCMR response frequency and correlations

Description

Calculates the response frequency and correlations for the MCMR items on the assessment.

Usage

```
MCMR.response.analysis(data, booted = FALSE, nRuns = 5, makeCorPlot = FALSE)
```

Arguments

`data`	Must be from the piql.data.select or data.select.Walkthrough functions
`booted`	Logical (default = FALSE). FALSE means no bootstrapping will be performed. TRUE turns on the bootstrapping feature.
`nRuns`	Number of random samples to use in the bootstrapping (default = 5). For publications it is recommended that 10,000 runs be performed since sample error goes as 1/sqrt(nRuns).
`makeCorPlot`	(Default = FALSE) If TRUE, then visualizations of the correlation matrices will be produced. When booted = true, the mean correlations will be plotted for each course.

Value

The response frequency and correlations for the MCMR items on the assessment.

Examples

```
# Pull in PIQL data from AWS and get some course data.
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0)

# Get selection numbers and correlations
MCMR.response.analysis(temp.data)

# Can be booted
MCMR.response.analysis(temp.data, booted = TRUE)

# Can make plots of the correlations matrices
MCMR.response.analysis(temp.data, makeCorPlot = TRUE)
```

multiple.imputation.pooled *Pooled Imputed results*

Description

Takes in results from simplyPsycho functions applied to multiply imputed data sets and returns the mean and sd of the results.

Usage

```
multiple.imputation.pooled(to.be.pooled)
```

Arguments

to.be.pooled	Must be the results of cttDifficulty, cttDiscrimination, cttCronbachAlpha or cttpointBiserial.

Value

Mean and sd of the results for multiple imputed data sets.

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0)
answerKey <- simplySampleData$answerkey

meh <- multiple.imputations(temp.data, answerKey)

thing <- cttDifficulty(meh)
#thing <- cttDiscrimination(meh)
#thing <- cttCronbachAlpha(meh)
#thing <- cttpointBiserial(meh)
```

multiple.imputations *Apply multple imputation to given data sets*

Description

Uses multiple implutation to create nImps complete data sets.

Uses multiple implutation to create nImps complete data sets.

Usage

```
multiple.imputations(
  data,
  answerKey,
  remove.lessthan.50perc.answered = TRUE,
  plot.missing.pattern = FALSE,
  nImps = 5,
  nMaxIterations = 5
)

multiple.imputations(
  data,
  answerKey,
  remove.lessthan.50perc.answered = TRUE,
  plot.missing.pattern = FALSE,
  nImps = 5,
  nMaxIterations = 5
)
```

Arguments

data	Can be either 1) An nS by nQ matrix or data frame of a dichotomous graded (0 or 1) sample, where nS is the number of students in the sample and nQ is the number of questions, 2) The output of piql.data.select with one or multiple courses selected.
answerKey	Answer key for the assessment being analyzted. Generally can be entered as something like pulled.data$answerKey from the pulldata function.
remove.lessthan.50perc.answered	(Default = TRUE) Removed students who answered less than 50 percent of the questions on the assessment. Removal of these students is standard prectice in multiple imputation practice. If you want to keep all students, set remove.lessthan.50perc.answered = TRUE.
plot.missing.pattern	(Default = FALSE) If TRUE, then a plot of the missing data patterns will be produced.
nImps	(Dfault = 5) Number of imputed data sets to be created.
nMaxIterations	(Default = 5) Number of iterations to be used to create mean values used to estimate missing data. Larger numbers take longer to run, but will give for consistent results.

Value

List of complete, imputed data sets for the given data formatted to be readily used in all simplyPsycho functions. Suggestion, do not make plots of this data using other functions. Get the results and then use the pool.imputations function to get the pooled results for the imputed courses.

https://datascienceplus.com/imputing-missing-data-with-r-mice-package/

List of complete, imputed data sets for the given data formatted to be readily used in all simplyPsycho functions. Suggestion, do not make plots of this data using other functions. Get the results and then use the pool.imputations function to get the pooled results for the imputed courses.

https://datascienceplus.com/imputing-missing-data-with-r-mice-package/

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 2, MCMR.items = NA)
answerKey <- PIQLdata$answerkey

multiple.imputations(temp.data, answerKey)
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0)
answerKey <- simplySampleData$answerkey

multiple.imputations(temp.data, answerKey)
```

`netA.Community.Memberships`

Community membership data for plotting

Description

Generates community membership data using multiple network analysis community detection methods and bootstrapping.

Usage

```
netA.Community.Memberships(
  data.num,
  cor.type = 1,
  lans.alpha = 0.05,
  nRuns = 100,
  printRunNumber = FALSE
)
```

Arguments

`data.num`	An nS by nQ matrix or data frame of a dichotomous graded (0 or 1) sample, where nS is the number of students in the sample and nQ is the number of questions.
`cor.type`	(default = 1). 1 = Pearson r, 2 = polychoric, 3 = partial correlation.
`lans.alpha`	Cutoff value for the LANS sparcification procedure. Default is set to 0.05, since this acts similar to a p-value.
`nRuns`	Number of random samples to use in the bootstrapping (default = 100). For publications it is recommended that 10,000 runs be performed since sample error goes as 1/sqrt(nRuns).
`printRunNumber`	Default = FALSE. Print the run number currently being computed. Useful when doing long runs and you want to keep track of there the code is.

Value

Community membership data to be put unto netA.Generate.PairedMat() for the generation of heatmaps.

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0)
data.num <- temp.data$data.num

# Get community membership data
data.memberships <- netA.Community.Memberships(temp.data$data.num)
# Generate the pared matrix
data.PairedMat <- netA.Generate.PairedMat(thing.memberships$fastgreedy)
# Create heat map
heatmap(data.PairedMat)
```

---

netA.Filter.LANS *Locally Adaptive Network Sparcification*

Description

Takes in an undirected network and performs LANS on it.

Usage

```
netA.Filter.LANS(initial.network, lans.alpha)
```

Arguments

initial.network	Initial network of the data. Output from graph.adjacency() function.
lans.alpha	Cutoff value for the LANS sparcification procedure. Default is set to 0.05, since this acts similar to a p-value.

Value

A LANS sparcified network.

---

netA.Generate.PairedMat *Create paired matrix for network analysis community detection heat maps*

Description

Generates paired matrix from community membership data.

Usage

```
netA.Generate.PairedMat(memberMatrix)
```

Arguments

memberMatrix	Output from netA.Community.Memberships() function.

Value

Paired matrix from community membership data to be plotted using heatmap().

Examples

```
# Pull sample data
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0)
data.num <- temp.data$data.num

# Get community membership data
data.memberships <- netA.Community.Memberships(temp.data$data.num)
# Generate the pared matrix
data.PairedMat <- netA.Generate.PairedMat(thing.memberships$fastgreedy)
# Create heat map
heatmap(data.PairedMat)
```

---

netA.lans.plot *Locally Adaptive Network Sparcification Plot*

Description

Takes in an undirected network and performs LANS on it.

Usage

```
netA.lans.plot(data.num, cor.type = 1, lans.alpha = 0.05, makePlot = TRUE)
```

Arguments

data.num	An nS by nQ matrix or data frame of a dichotomous graded (0 or 1) sample, where nS is the number of students in the sample and nQ is the number of questions.
cor.type	(default = 1). 1 = Pearson r, 2 = polychoric, 3 = partial correlation.
lans.alpha	Cutoff value for the LANS sparcification procedure. Default is set to 0.05, since this acts similar to a p-value.
makePlot	Generate plot or not.

Value

A plot of the LANS sparcified network for the given data.

Examples

```
# Pull in PIQL data from AWS and get some course data.
temp.data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0)
data.num <- temp.data$data.num

# Get LANS plot of data
netA.lans.plot(temp.data$data.num, cor.type = 1)
```

`piql.data.select` *Select specific course from the pulled PIQL data.*

Description

Selects a specific course from the pulled PIQL data and separates the correct MCMR options marked 1 if they were selected and 0 if they were not.

Usage

```
piql.data.select(
  pulled.PIQL.data,
  MCMR.grading = "Dichotomous",
  MCMR.items = c(15:20),
  courses = 1,
  numBlanks.allowed = 0,
  minTime.allowed = NA,
  maxTime.allowed = 999,
  Matched = FALSE
)
```

Arguments

`pulled.PIQL.data`
: Output from pullPQLdata

`MCMR.grading`
: (Default = "Dichotomous") If undefined, then MCMR items will be graded dichotomously.

  If MCMR.grading = "Selected", then the correct options to the indicated MCMR items will be included in the output as their own columns (e.g., Q17A and Q17C) populated with 0's and 1's. 0 = student DID NOT select the option, and 1 = student did select the option.

  If MCMR.grading = "FourScale", then MCMR items will be graded o the 4 scale: (0 = completely incorrect, 1 = correct and incorrect, 2 = some correct, 3 = completely correct)

`MCMR.items`
: Which items are MCMR. Defaulted to PIQL (15 through 20).

`numBlanks.allowed`
: (default = 0) Number of blanks allowed in the resulting data.

`minTime.allowed`
: (default = NA) If NA, then no time cutting will be done. If not NA, then data will be constrained between the two times minTime.allowed and maxTime.allowed

`maxTime.allowed`
: (default = 999) Max time allowed for students if time cutting is enabled.

`Matched`
: (Default = FALSE). If TRUE, only matching student IDs will be extracted across all courses.

`course`
: (Default = 1 = 121) Course number (1 = 121, 2 = 122, 3 = 123, etc.) or position in course list:

  Course: 121, 122, 123, 141, 142, 143, 224, 225, 226, 321, 322, 323, 325

  For all courses in the data set to "all" (include quotations).

Value

A list of course data for the PIQL stored in AWS.

$data.alpha ALPHABETICAL data for the selected course.

$data.num NUMERICAL data for the selected course.

$nS.details Details about how many students were removed due to the specified numBlanks.allowed value.

Examples

```
# Using Sample data in simplyPsycho
temp.data <- piql.data.select(simplySampleData)
data.alpha <- temp.data$data.alpha
data.num <- temp.data$data.num
# Check number of student removed. Should be less than 10%.
temp.data$nS.details
```

---

pulldata *Pull data from AWS*

Description

Walks the user through pulling available data from AWS.

Usage

```
pulldata(
  accessID,
  accessSecret,
  data.directory = "piqlgerqndata",
  name.assessment = NULL,
  region = "us-west-2"
)
```

Arguments

accessID	AWS access ID.
accessSecret	AWS secret access ID.
data.directory	(Default = "piqlgerqndata") Name of the directory in AWS the user is extracting data from.
name.assessment	Name of the assessment who data is to be extracted. Default option will result in a walk through helping the user select the correct data.
region	(Default = "us-west-2") AWS region ID for database.

Value

A list of course data stored in AWS for the requested assessment.

Examples

```
# Pull in data from AWS and get some course data.
pulledData <- pulldata()
pulledData$courses # Course data
pulledData$answerkey # answer key
```

---

pullPIQLdata *Pull PIQL data from AWS*

Description

Pulls PIQL data from AWS

Usage

```
pullPIQLdata(accessID, accessSecret)
```

Value

A list of course data for the PIQL stored in AWS.

$courses course data as a list.

$answerkey answer key stored in AWS.

Examples

```
# Pull in PIQL data from AWS and get some course data.
PIQLdata <- pullPIQLdata()
PIQLdata$courses # Course data
PIQLdata$answerkey # answer key
```

---

response.Frequency *Response Frequency*

Description

Calculates the frequency each response option is selected in the given data

Usage

```
response.Frequency(data, booted = FALSE, nRuns = 10, MCMR.separate = TRUE)
```

Arguments

data	Can be either 1) An nS by nQ matrix or data frame of ALPHABETICAL data, where nS is the number of students in the sample and nQ is the number of questions, 2) The output of piql.data.select with one or multiple courses selected.
booted	Logical (default = FALSE). FALSE means no bootstrapping will be performed. TRUE turns on the bootstrapping feature.
nRuns	Number of random samples to use in the bootstrapping (default = 100). For publications it is recommended that 10,000 runs be performed since sample error goes as 1/sqrt(nRuns).
MCMR.separate	(Default = TRUE). MCMR.separate = TRUE separates multiple selections to get percentage for individual options on the MCMR items. MCMR.separate = FALSE treats grouped selections as individual options.

Value

When booted = FALSE (the default setting) then the straight calculated value will be returned. If booted = TRUE, the function will output the mean and standard deviation for the CTT item difficulties calculated from nRuns randomly sampled with replacement data sets from the given sample.

Examples

```
# Pull sample data
data <- piql.data.select(simplySampleData, courses = 1, numBlanks.allowed = 0)

# Get Response frequencies for each item in each selected course
response.Frequency(data)
# Supports bootstrapping
response.Frequency(data, booted = TRUE)
```

---

select.questions *Select specific course from the pulled PIQL data.*

Description

Selects a specific course from the pulled PIQL data and separates the correct MCMR options marked 1 if they were selected and 0 if they were not.

Usage

```
select.questions(selected.data, selected.questions = NULL)
```

Arguments

selected.data	Output from a data selection function (like piql.data.select)
selected.questions	A list of questions for each pulled course. Each course needs its own list of questions. (NOT IMPLEMENTED YET)

Value

A list of course data filtered to include only the selected questions.

$data.alpha ALPHABETICAL data for the selected course.

$data.num NUMERICAL data for the selected course.

$nS.details Details about how many students were removed due to the specified numBlanks.allowed value.

Examples

```
# Using Sample data in simplyPsycho
temp.data <- piql.data.select(simplySampleData, courses = c(1,2))
temp.data <- select.questions(temp.data)
```

Index