Package: NPCD (via r-universe)

September 4, 2024

Type Package

Version 1.0-11

Date 2019-11-15

Title Nonparametric Methods for Cognitive Diagnosis

Maintainer Yi Zheng <yi.isabel.zheng@asu.edu>

Description An array of nonparametric and parametric estimation methods for cognitive diagnostic models, including nonparametric classification of examinee attribute profiles, joint maximum likelihood estimation (JMLE) of examinee attribute profiles and item parameters, and nonparametric refinement of the Q-matrix, as well as conditional maximum likelihood estimation (CMLE) of examinee attribute profiles given item parameters and CMLE of item parameters given examinee attribute profiles. Currently the nonparametric methods in the package support both conjunctive and disjunctive models, and the parametric methods in the package support the DINA model, the DINO model, the NIDA model, the G-NIDA model, and the R-RUM model.

Depends R (>= 2.14.2), BB

Imports R.methodsS3

LazyLoad yes

LazyData yes

License LGPL (>= 2.1)

Repository CRAN

NeedsCompilation no

Author Yi Zheng [aut, cre], Chia-Yi Chiu [aut], Jeffrey A. Douglas [ctb]

Date/Publication 2019-11-15 17:20:07 UTC

Contents

`NPCD-package` *NPCD: The* R *Package for Nonparametric Methods for Cognitive Diagnosis*

Description

This package implements an array of nonparametric and parametric estimation methods for cognitive diagnostic models, including nonparametric classification of examinee attribute profiles, joint maximum likelihood estimation (JMLE) of examinee attribute profiles and item parameters, and nonparametric refinement of the Q-matrix, as well as conditional maximum likelihood estimation (CMLE) of examinee attribute profiles given item parameters and CMLE of item parameters given examinee attribute profiles.

Details

Package:	NPCD
Type:	Package
Version:	1.0-11
Date:	2019-11-15
License:	LGPL (>= 2.1)
Depends:	R (>= 2.14.2), BB, R.oo

Cognitive diagnostic models (CDM) are a group of latent class models specialized for cognitive diagnosis in educational testing. Based on the responses to the items in a test, CDMs classify each examinee into mastery or non-mastery on each of a few attributes, producing an "attribute profile" for the examinee. For different tests, the attributes can represent different constructs (e.g., skills,

traits, strategies). An attribute profile can provide helpful diagnostic information of the examinee regarding the targeted constructs. Some examples of CDMs include the deterministic inputs, noisy "AND" gate (DINA) model (Junker & Sijtsma, 2001), the Deterministic Input, Noisy Output "OR" gate (DINO) model (Templin & Henson, 2006), the NIDA model (Maris, 1999), the reduced reparametrized unified model (R-RUM) (Hartz, Roussos, Henson, & Templin, 2005), the log-linear CDM (Hen son, Templin, & Willse, 2009), the general diagnostic model (GDM) (Davier, 2010), and the generalized DINA model (G-DINA) (de la Torre, 2011).

Despite the different ways to model the relationship between examinees' attribute profiles and their answers to test items, most of the existing CDMs make use of a so-called Q-matrix (Tatsuoka, 1985), which specifies which attributes are required by each item. The Q-matrix and the examinee attribute profile together produce the ideal response pattern to the items by an examinee. However, the real response pattern may not be exactly the same with the ideal response pattern due to other factors. Therefore, CDMs incorporate stochastic elements to account for random deviations from the ideal response pattern. For an acceptable model fit, the stochastic terms should allow some departure from the ideal response pattern, but not so much that the model becomes doubtful. And in such nice conditions, despite the distinctive stochastic terms and assumptions of different CDMs, according to the likelihood functions, the ideal response pattern should still be the most likely one among all possible response patterns. Therefore, classification of examinee attribute profiles based on the ideal response patterns can be effective without fitting the parametric models. Chiu and Douglas (2013) proposed a few nonparametric classification methods based on this idea. This R package NPCD was built to carry out a group of methods that are all based on the above-mentioned nonparametric classification idea, including (1) the nonparametric classification methods proposed in Chiu and Douglas (2013), (2) a joint maximum likelihood estimation (JMLE) approach that utilizes the nonparametric classification methods to give good initial classifications of examinee attribute profiles and then estimates model-based attribute profiles and item parameters simultaneously, and (3) a nonparametric approach to refine the Q-matrix based on the residual sum of squares (RSS) criterion (Chiu, 2013).

Joint maximum likelihood estimation (JMLE) is a parameteric approach to fit a CDM to response data and estimate both the item parameters and examinee attribute profiles. Two other common methods are the expectation maximization (EM) algorithm and the Markov chain Monte Carlo (MCMC) technique. Despite JMLE's attractive advantages of mathematical simplicity and computational rapidity especially for more complex models, so far JMLE has not been successfully implemented for CDMs due to the potential inconsistency resulted from arbitrary initial values. In this package, we try to overcome this limitation by taking the classification results obtained from the nonparametric classification methods (Chiu & Douglas, 2013) as the initial values. Simulation results have shown JMLE is efficient and accurate for the models incorporated in this package.

In most practices, the Q-matrix of a test is constructed from the judgments of content experts. The misspecification of the Q-matrix can impair the estimation of the model parameters as well as classification of examinee attribute profiles. This package implements the Q-matrix refinement method developed by Chiu (2013), which is also based on the aforementioned nonparametric classification methods (Chiu & Douglas, 2013). This Q-matrix refinement method corrects potential misspecified entries of the Q-matrix through comparisons of the residual sum of squares computed from the observed and the ideal item responses. Interested users can refer to Chiu (2013) for more details.

The aim of the NPCD package is to provide an easy-to-use tool to analyze cognitive diagnostic test data using the aforementioned nonparametric methods as well as a few model-based methods for some CDMs. Currently the nonparametric methods in the package support both conjunctive and disjunctive models, and the parametric methods in the package support the DINA model, the DINO

model, the NIDA model, the G-NIDA model, and the R-RUM model. Note that the G-NIDA model and the R-RUM model are equivalent based on reparametrization; nevertheless, we think it may still provide some convenicence to the users to include both of them us an option. The package also contains demonstrative data generated from some of these models. Besides the numeric outputs, the package also provides diagnostic plots for users to look inside the algorithms.

Author(s)

Yi Zheng (Arizona State University) and Chia-Yi Chiu (Rutgers, the State University of New Jersey)

Maintainer: Yi Zheng <yi.isabel.zheng@asu.edu>

References

Chen, J. de la Torre, J., & Zhang, Z. (2013). Relative and absolute fit evaluation in cognitive diagnosis modeling. *Journal of Educational Measurement, 50*, 123-140.

Chiu, C. Y. (2011). *Flexible approaches to cognitive diagnosis: nonparametric methods and small sample techniques*. Invited session of cognitive diagnosis and item response theory at 2011 Joint Statistical Meeting.

Chiu, C. Y., & Douglas, J. A. (2013). A nonparametric approach to cognitive diagnosis by proximity to ideal response patterns. *Journal of Classification 30*(2), 225-250.

Chiu, C. Y. (2013). Statistical Refinement of the Q-matrix in Cognitive Diagnosis. *Applied Psychological Measurement, 37*(8), 598-618.

Davier, M. (2010). A General Diagnostic Model Applied to Language Testing Data. *British Journal of Mathematical and Statistical Psychology, 61*(2), 287-307.

de la Torre, J. (2011). The Generalized DINA Model Framework. *Psychometrika, 76*(2), 179-199.

Hartz, S., Roussos, L., Henson, R., & Templin, J. (2005). The Fusion Model for Skill Diagnosis: Blending Theory with Practicality. *Unpublished Manuscript*.

Henson, R.A., Templin, J.L., & Willse, J.T. (2009). Defining a Family of Cognitive Diagnosis Models Using Log-Linear Models with Latent Variables. *Psychometrika, 74*(2), 191-210.

Junker, B., Sijtsma, K. (2001). Cognitive Assessment Models with Few Assumptions, and Connections with Nonparametric Item Response Theory. *Applied Psychological Measurement, 25*(3), 258-272.

Maris, E. (1999). Estimating Multiple Classification Latent Class Models. *Psychometrika, 64*(2), 187-212.

Tatsuoka, K. K. (1985). A probabilistic model for diagnosing misconceptions by the pattern classification approach. *Journal of Educational and Behavioral Statistics, 10*, 55-73.

Templin, J.L,, Henson, R.A. (2006). Measurement of Psychological Disorders Using Cognitive Diagnosis Models. *Psychological Methods, 11*(3), 287-305.

AlphaMLE *Maximum likelihood estimation of attribute profile*

Description

This function returns the model-based Maximum likelihood estimator(s) of the cognitive diagnostic attribute profile(s). Currently supported cognitive diagnostic models include the DINA, DINO, NIDA, GNIDA, and R-RUM models.

Usage

```
AlphaMLE(Y, Q, par, model = c("DINA", "DINO", "NIDA", "GNIDA", "RRUM"),
undefined.flag = NULL)
```

Arguments

Y	A matrix of binary responses. Rows represent persons and columns represent items. 1=correct, 0=incorrect.
Q	The Q-matrix of the test. Rows represent items and columns represent attributes. 1=attribute required by the item, 0=attribute not required by the item.
par	A list of parameters. DINA & DINO — par$slip: a vector of slipping parameters for each item; par$guess: a vector of guessing parameters for each item. NIDA — par$slip: a vector of slipping parameters for each attribute; par$guess: a vector of guessing parameters for each attribute. GNIDA — par$slip: a matrix (items by attributes) of slipping parameters; par$guess: a matrix (items by attributes) of guessing parameters. RRUM — par$pi: a vector of pi parameters for each item; par$r: a matrix (items by attributes) of r parameters.
model	Currently supports five models: "DINA", "DINO", "NIDA", "GNIDA", and "RRUM". The default is "DINA".
undefined.flag	A binary vector indicating whether the parameters of each item are undefined. 1=undefined, 0=defined. Generally, this argument is only needed in "JMLE" for the DINA and DINO models, where the data may generate undefined item parameters for some items.

Value

alpha.est	A matrix of estimated attribute profiles for all examinees. Rows represent persons and columns represent attributes. 1=examinee masters the attribute, 0=examinee does not master the attribute.
est.class	The class number (row index in pattern) for each person's attribute profile. It can also be used for locating the log-likelihood value in loglike.matrix for the estimated attribute profile for each person.
n.tie	Number of ties in the log-likelihood among the candidate attribute profiles for each person. When we encounter ties, one of the tied attribute profiles is randomly chosen.

class.tie	The class numbers (row index in pattern) of the tied patterns for each person.
pattern	All possible attribute profiles in the search space.
loglike.matrix	The matrix of the log-likelihood function values. Rows represent candidate attribute profiles in the same order with the pattern matrix; columns represent different examinees.

See Also

AlphaNP, JMLE, print.AlphaMLE, plot.AlphaMLE

Examples

```
# Generate item and examinee profiles

natt <- 3
nitem <- 4
nperson <- 5
Q <- rbind(c(1, 0, 0), c(0, 1, 0), c(0, 0, 1), c(1, 1, 1))
alpha <- rbind(c(0, 0, 0), c(1, 0, 0), c(0, 1, 0), c(0, 0, 1), c(1, 1, 1))

# Generate DINA model-based response data

slip <- c(0.1, 0.15, 0.2, 0.25)
guess <- c(0.1, 0.15, 0.2, 0.25)
my.par <- list(slip=slip, guess=guess)

data <- matrix(NA, nperson, nitem)
eta <- matrix(NA, nperson, nitem)

for (i in 1:nperson) {
  for (j in 1:nitem) {
  eta[i, j] <- prod(alpha[i,] ^ Q[j, ])
  P <- (1 - slip[j]) ^ eta[i, j] * guess[j] ^ (1 - eta[i, j])
  u <- runif(1)
  data[i, j] <- as.numeric(u < P)
  }
}

# Using the function to estimate examinee attribute profile

alpha.est.MLE <- AlphaMLE(data, Q, my.par, model="DINA", undefined.flag=NULL)

nperson <- 1   # Choose an examinee to investigate
print(alpha.est.MLE)   # Print the estimated examinee attribute profiles
plot(alpha.est.MLE, nperson)   # Plot the sorted log-likelihood function
#of different attribute profiles for this examinee
ItemFit(alpha.est.MLE)
```

AlphaNP *Nonparametric estimation of attribute profiles*

Description

This function estimates attribute profiles using nonparametric approaches for both the "AND gate" (conjunctive) and the "OR gate" (disjunctive) cognitive diagnostic models. These algorithms select the attribute profile with the smallest loss function value (plain, weighted, or penalized Hamming distance, see below for details) as the estimate. If more than one attribute profiles have the smallest loss function value, one of them is randomly chosen.

Usage

```
AlphaNP(Y, Q, gate = c("AND", "OR"), method = c("Hamming", "Weighted",
"Penalized"), wg = 1, ws = 1)
```

Arguments

Y	A matrix of binary responses. Rows represent persons and columns represent items. 1=correct, 0=incorrect.
Q	The Q-matrix of the test. Rows represent items and columns represent attributes. 1=attribute required by the item, 0=attribute not required by the item.
gate	"AND": the examinee needs to possess all required attributes of an item in order to answer it correctly; "OR": the examinee needs to possess only one of the required attributes of an item in order to answer it correctly.
method	The method of nonparametric estimation. "Hamming": the plain Hamming distance method; "Weighted": the Hamming distance weighted by inversed item variance; "Penalized": the Hamming distance weighted by inversed item variance and specified penalizing weights for guess and slip.
wg	Additional argument for the "penalized" method. wg is the weight assigned to guessing in the DINA or DINO models. A large value of wg results in a stronger impact on Hamming distance (larger loss function values) caused by guessing.
ws	Additional input for the "penalized" method. ws is the weight assigned to slipping in the DINA or DINO models. A large value of ws results in la stronger impact on Hamming distance (larger loss function values) caused by slipping.

Value

alpha.est	Estimated attribute profiles. Rows represent persons and columns represent attributes. 1=examinee masters the attribute, 0=examinee does not master the attribute.
est.ideal	Estimated ideal response to all items by all examinees. Rows represent persons and columns represent items. 1=correct, 0=incorrect.
est.class	The class number (row index in pattern) for each person's attribute profile. It can also be used for locating the loss function value in loss.matrix for the estimated attribute profile for each person.

n.tie	Number of ties in the Hamming distance among the candidate attribute profiles for each person. When we encounter ties, one of the tied attribute profiles is randomly chosen.
pattern	All possible attribute profiles in the search space.
loss.matrix	The matrix of the values for the loss function (the plain, weighted, or penalized Hamming distance). Rows represent candidate attribute profiles in the same order with the pattern matrix; columns represent different examinees.

References

Chiu, C. (2011). *Flexible approaches to cognitive diagnosis: nonparametric methods and small sample techniques*. Invited session of cognitive diagnosis and item response theory at 2011 Joint Statistical Meeting.

Chiu, C. Y., & Douglas, J. A. (2013). A nonparametric approach to cognitive diagnosis by proximity to ideal response patterns. *Journal of Classification 30*(2), 225-250.

See Also

AlphaMLE, JMLE, print.AlphaNP, plot.AlphaNP

Examples

```
# Generate item and examinee profiles

natt <- 3
nitem <- 4
nperson <- 5
Q <- rbind(c(1, 0, 0), c(0, 1, 0), c(0, 0, 1), c(1, 1, 1))
alpha <- rbind(c(0, 0, 0), c(1, 0, 0), c(0, 1, 0), c(0, 0, 1), c(1, 1, 1))

# Generate DINA model-based response data

slip <- c(0.1, 0.15, 0.2, 0.25)
guess <- c(0.1, 0.15, 0.2, 0.25)
my.par <- list(slip=slip, guess=guess)

data <- matrix(NA, nperson, nitem)
eta <- matrix(NA, nperson, nitem)

for (i in 1:nperson) {
  for (j in 1:nitem) {
  eta[i, j] <- prod(alpha[i,] ^ Q[j, ])
  P <- (1 - slip[j]) ^ eta[i, j] * guess[j] ^ (1 - eta[i, j])
  u <- runif(1)
  data[i, j] <- as.numeric(u < P)
  }
}

# Using the function to estimate examinee attribute profile

alpha.est.NP.H <- AlphaNP(data, Q, gate="AND", method="Hamming")
```

```
alpha.est.NP.W <- AlphaNP(data, Q, gate="AND", method="Weighted")
alpha.est.NP.P <- AlphaNP(data, Q, gate="AND", method="Penalized", wg=2, ws=1)

nperson <- 1   # Choose an examinee to investigate
print(alpha.est.NP.H) # Print the estimated examinee attribute profiles
plot(alpha.est.NP.H, nperson)   # Plot the sorted loss function of different
#attribute profiles for this examinee
ItemFit(alpha.est.NP.H, model="DINA", par=list(slip=slip, guess=guess))
ItemFit(alpha.est.NP.W, model="DINA", par=list(slip=slip, guess=guess))
ItemFit(alpha.est.NP.P, model="DINA", par=list(slip=slip, guess=guess))
```

CDL *Log-likelihood for cognitive diagnostic models*

Description

This function returns the log-likelihood of a particular examinee's responses to a set of cognitive diagnostic items. Currently supported cognitive diagnostic models include the DINA model, DINO model, NIDA model, G-NIDA model, and R-RUM model. This function is called by the `AlphaMLE` function and the `JMLE` function in the package.

Usage

```
CDL(Y, Q, par, alpha, model = c("DINA", "DINO", "NIDA", "GNIDA", "RRUM"),
undefined.flag)
```

Arguments

`Y`	A vector of binary examinee responses. 1=correct, 0=incorrect.
`Q`	The Q-matrix of the test. Rows represent items and columns represent attributes. 1=attribute required by the item, 0=attribute not required by the item.
`par`	A list of parameters. DINA & DINO — `par$slip`: a vector of slipping parameters for each item; `par$guess`: a vector of guessing parameters for each item. NIDA — `par$slip`: a vector of slipping parameters for each attribute; `par$guess`: a vector of guessing parameters for each attribute. GNIDA — `par$slip`: a matrix (items by attributes) of slipping parameters; `par$guess`: a matrix (items by attributes) of guessing parameters. RRUM — `par$pi`: a vector of pi parameters for each item; `par$r`: a matrix (items by attributes) of r parameters.
`alpha`	A vector of examinee ability profile. 1=examinee masters the attribute, 0=examinee does not master the attribute.
`model`	Currently supports five models: `"DINA"`, `"DINO"`, `"NIDA"`, `"GNIDA"`, and `"RRUM"`. The default is `"DINA"`.
`undefined.flag`	A binary vector indicating whether the parameters of each item are undefined. 1=undefined, 0=defined.

Value

`loglike` The log likelihood function value for the given data.

See Also

`AlphaMLE`, `JMLE`

Examples

```
# Generate item and examinee profiles

nitem <- 4
Q <- rbind(c(1, 0, 0), c(0, 1, 0), c(0, 0, 1), c(1, 1, 1))
alpha <- c(1, 0, 0)

# Generate DINA model-based response data

slip <- rep(0.1, nitem)
guess <- rep(0.1, nitem)
my.par <- list(slip=slip, guess=guess)

data <- NA
eta <- NA

for (i in 1:nitem) {
  eta[i] <- prod(alpha ^ Q[i, ])
  P <- (1 - slip[i]) ^ eta[i] * guess[i] ^ (1 - eta[i])
  u <- runif(1)
  data[i] <- as.numeric(u < P)
}

# Using the function to compute the log-likelihood of the given data

CDL(data, Q, my.par, alpha, model="DINA", undefined.flag=rep(0, nitem))
```

CDP *Probability of correct response for cognitive diagnostic models*

Description

This function returns the model-predicted probability of correct response of one item for one person given the item parameters, Q vector, and alpha vector. Currently supported cognitive diagnostic models include the DINA model, DINO model, NIDA model, G-NIDA model, and R-RUM model. This function is called by the `ItemFit` function in the package.

Usage

```
CDP(Q, par, alpha, model = c("DINA", "DINO", "NIDA", "GNIDA", "RRUM"))
```

Arguments

Q	The Q-vector of the item. Columns represent attributes. 1=attribute required by the item, 0=attribute not required by the item.
par	A list of parameters. DINA & DINO — par$slip: a scaler slip parameter for the item; par$guess: a scaler guessing parameter for the item. NIDA — par$slip: a vector of slip parameters for each attribute; par$guess: a vector of guessing parameters for each attribute. GNIDA — par$slip: a vector of slip parameters for each attribute for the item; par$guess: a vector of guessing parameters for each attribute for the item. RRUM — par$pi: a scaler pi parameter for the item; par$r: a vector of r parameters for each attribute for the item.
alpha	A vector of examinee ability profile. 1=examinee masters the attribute, 0=examinee does not master the attribute.
model	Currently supports five models: "DINA", "DINO", "NIDA", "GNIDA", and "RRUM". The default is "DINA".

Value

P	The probability of correct response for the item by the person.

Examples

```
# Generate item and examinee profiles

Q <- c(1, 0, 0)
alpha <- c(1, 0, 0)
slip <- 0.2
guess <- 0.1
my.par <- list(slip=slip, guess=guess)
CDP(Q, my.par, alpha, model="DINA")
```

Data.DINA *Example dataset generated using DINA model*

Description

This dataset is a list containing components generated using DINA model, which can be used to test out the estimation functions in the package.

$Q: The Q matrix. Numerical, 20 (item) by 3 (attribute). Binary values: 1=attribute required by the item, 0=attribute not required by the item.

$response: The examinee responses to all items. Numerical, 300 (examinee) by 20 (item). Binary values: 1=correct, 0=incorrect.

$true.alpha: The true examinee ability profiles. Numerical, 300 (examinee) by 3 (attribute). Binary values: 1=examinee masters the attribute, 0=examinee does not master the attribute.

$true.par$slip: The true slipping parameters for each item. Numerical, vector of 20.

$true.par$guess: The true guessing parameters for each item. Numerical, vector of 20.

Usage

```
data(Data.DINA)
```

Format

The format is: List of 4 elements.

Data.DINO *Example dataset generated using DINO model*

Description

This dataset is a `list` containing components generated using DINO model, which can be used to test out the estimation functions in the package.

`$Q`: The Q matrix. Numerical, 20 (item) by 3 (attribute). Binary values: 1=attribute required by the item, 0=attribute not required by the item.

`$response`: The examinee responses to all items. Numerical, 300 (examinee) by 20 (item). Binary values: 1=correct, 0=incorrect.

`$true.alpha`: The true examinee ability profiles. Numerical, 300 (examinee) by 3 (attribute). Binary values: 1=examinee masters the attribute, 0=examinee does not master the attribute.

`$true.par$slip`: The true slipping parameters for each item. Numerical, vector of 20.

`$true.par$guess`: The true guessing parameters for each item. Numerical, vector of 20.

Usage

```
data(Data.DINO)
```

Format

The format is: List of 4 elements.

Data.Qrefine *Example dataset used for the* `Qrefine` *function*

Description

This dataset is a `list` containing components that can be used for the `Qrefine` function. The response marix is generated using the DINA model.

`$mis.Q`: The mis-specified Q matrix. Numerical, 20 (item) by 3 (attribute). Binary values: 1=attribute required by the item, 0=attribute not required by the item.

`$true.Q`: The true Q-matrix. Numerical, 20 (item) by 3 (attribute). Binary values: 1=attribute required by the item, 0=attribute not required by the item.

`$response`: The examinee responses to all items, generated using the true Q-matrix. Numerical, 300 (examinee) by 20 (item). Binary values: 1=correct, 0=incorrect.

`$true.alpha`: The true examinee ability profiles. Numerical, 300 (examinee) by 3 (attribute). Binary values: 1=examinee masters the attribute, 0=examinee does not master the attribute.

`$true.par$slip`: The true slipping parameters for each item. Numerical, vector of 20.

`$true.par$guess`: The true guessing parameters for each item. Numerical, vector of 20.

Usage

```
data(Data.Qrefine)
```

Format

The format is: List of 5 elements.

`ItemFit` *Compute item fit statistics for outputs generated by estimation functions in the package*

Description

This function computes item fit statistics for outputs generated by estimation functions in the package, including `AlphaNP`, `AlphaMLE`, `ParMLE`, and`JMLE`. The function currently provides the RMSEA and Chi-square item fit statistics.

Usage

```
ItemFit(x, model=NULL, par=NULL)
```

Arguments

`x`	The output from the function (The list of all outputs).
`model`	This needs to be additionally specified only when `x` is output from `AlphaNP`. Currently support five models: `"DINA"`, `"DINO"`, `"NIDA"`, `"GNIDA"`, and `"RRUM"`.

`par` This needs to be additionally specified only when `x` is output from `AlphaNP`. A list of parameters. DINA & DINO — `par$slip`: a vector of slipping parameters for each item; `par$guess`: a vector of guessing parameters for each item. NIDA — `par$slip`: a vector of slipping parameters for each attribute; `par$guess`: a vector of guessing parameters for each attribute. GNIDA — `par$slip`: a matrix (items by attributes) of slipping parameters; `par$guess`: a matrix (items by attributes) of guessing parameters. RRUM — `par$pi`: a vector of pi parameters for each item; `par$r`: a matrix (items by attributes) of r parameters.

Value

`RMSEA` The model-based root mean square error of approximation (Kunina-Habenicht et al., 2012) of each item based on the estimated or given item parameter, Q-vector, and alpha matrix.

`Chisq` The Q1 Chi-square statistic (Wang et al., 2015; Yen, 1981) of each item based on the estimated or given item parameter, Q-vector, and alpha matrix.

`Chisq.p` The p-values for the Chi-square statistic for each item.

`Chisq.df` The degrees of freedom for the Chi-square statistic for each item.

References

Kunina-Habenicht, O., Rupp, A. A., & Wilhelm, O. (2012). The Impact of Model Misspecification on Parameter Estimation and Item-Fit Assessment in Log0Linear Diagnostic Classification Models. *Journal of Educational Measurement, 49*(1), 59-81.

Wang, C., Shu, Z., Shagn, Z., & Xu, G. (2015). Assessing Item-Level Fit for the DINA Model. *Applied Psychological Measurement*, 1-14.

Yen, W. M. (1981). Using Simulation Results to Choose a Latent Trait Model. *Applied Psychological Measurement, 5*, 245-262.

See Also

`AlphaNP`, `AlphaMLE`, `ParMLE`, `JMLE`

Examples

```
# See examples in AlphaNP, AlphaMLE, ParMLE, and JMLE.
```

`JMLE` *Joint maximum likelihood estimation of item parameters and examinee attribute profiles*

Description

This function returns joint maximum likelihood estimates of item parameters and examinee attribute profiles in cognitive diagnostic models. The algorithm starts from the nonparametric estimation of attribute profiles, implemented by the `AlphaNP` function, and then iteratively estimates item parameters and attribute profiles using conditional maximum likelihood estimation until the algorithm converges. Currently supported models include the DINA model, the DINO model, he NIDA model, the G-NIDA model, and the R-RUM model.

Usage

```
JMLE(Y, Q, model = c("DINA", "DINO", "NIDA", "GNIDA", "RRUM"),
NP.method = c("Weighted", "Hamming", "Penalized"), wg=1, ws=1,
conv.crit.par = 0.001, conv.crit.att = 0.01, max.ite = 100)
```

Arguments

`Y`	A matrix of binary responses. Rows represent persons and columns represent items. 1=correct, 0=incorrect.
`Q`	The Q-matrix of the test. Rows represent items and columns represent attributes. 1=attribute required by the item, 0=attribute not required by the item.
`model`	Currently support five models: `"DINA"`, `"DINO"`, `"NIDA"`, `"GNIDA"`, and `"RRUM"`. The default is `"DINA"`.
`NP.method`	The method of the nonparametric estimation in the initial stage. `"Hamming"`: the plain Hamming distance method; `"Weighted"`: the Hamming distance weighted by inversed item variance; `"Penalized"`: the Hamming distance weighted by inversed item variance and specified penalizing weights for guess and slip. The default is `"Weighted"`.
`wg`	Additional argument for the "penalized" NP.method. `wg` is the weight assigned to guessing in the DINA or DINO models. A large value of `wg` results in a stronger impact on Hamming distance (larger loss function values) caused by guessing.
`ws`	Additional input for the "penalized" NP.method. `ws` is the weight assigned to slipping in the DINA or DINO models. A large value of `ws` results in la stronger impact on Hamming distance (larger loss function values) caused by slipping.
`conv.crit.par`	The critical value for the maximum absolute change in all item parameters values to determine convergence.
`conv.crit.att`	The critical value for the percentage of examinee attribute profiles that are changed to determine convergence.
`max.ite`	The maximum number of iterations allowed.

Value

`alpha.est`	JMLE estimates of examinee attribute profiles. Rows represent persons and columns represent attributes. 1=examinee masters the attribute, 0=examinee does not master the attribute.

par.est	JMLE estimates of item parameters, including par.est$slip, par.est$guess, par.est$se.slip, and par.est$se.guess for the DINA, DINO, NIDA, and GNIDA models, and par.est$pi, par.est$r, par.est$se.pi and par.est$se.r for the R-RUM model. Note that for the G-NIDA model and the R-RUM model, the item parameter estimates and standard errors are not available for the entries where the Q-matrix is 0.
n.tie	Number of ties in the final log-likelihood among the candidate attribute profiles for each person. When we encounter ties, one of the tied attribute profiles is randomly chosen.
undefined.flag	A binary vector indicating whether the parameters of each item are undefined. 1=undefined, 0=defined.
loglike	The final overall log-likelihood value from the estimated item parameters and attribute profiles based on the specified model.
convergence	A message on whether the algorithm converged.
n.ite	Number of iterations performed.
loglike.matrix	The values for the log-likelihood function in the last iteration for each candidate attribute profile by each person. Rows represent candidate attribute profiles in the same order with the pattern matrix; columns represent different examinees.
est.class	The final class number (row index in pattern) for each person's attribute profile. It can also be used for locating the log-likelihood value in loglike.matrix for the estimated attribute profile for each person.
NP.loss.matrix	The values for the loss function of the nonparametric estimation of Alpha. Rows represent candidate attribute profiles in the same order with the pattern matrix; columns represent different examinees.
NP.alpha.est	The estimates of examinee attribute profiles from the initial nonparameteric estimation.
NP.est.class	The class number (row index in pattern) for each person's attribute profile from the initial nonparametric classification. It can also be used for locating the loss function value in NP.loss.matrix for the estimated attribute profile for each person.
pattern	All possible attribute profiles in the search space.
model	The chosen model.
Q	The Q-matrix of the test.

References

Chiu, C. (2011). *Flexible approaches to cognitive diagnosis: nonparametric methods and small sample techniques*. Invited session of cognitive diagnosis and item response theory at 2011 Joint Statistical Meeting.

Chiu, C. Y., & Douglas, J. A. (2013). A nonparametric approach to cognitive diagnosis by proximity to ideal response patterns. *Journal of Classification 30*(2), 225-250.

See Also

AlphaMLE, AlphaNP, ParMLE, print.JMLE, plot.JMLE

Examples

```
data("Data.DINA")
JMLE.result <- JMLE(Data.DINA$response, Data.DINA$Q, model="DINA", conv.crit.par=0.001,
conv.crit.att=0.001, max.ite=100)
print(JMLE.result)  # Print the estimated item parameters, standard errors,
#and examinee attribute profiles
plot(JMLE.result, nperson=1)   # Plot the sorted loss function of different
#attribute profiles for this examinee
ItemFit(JMLE.result)
ModelFit(JMLE.result)
```

`ModelFit` *Compute overall model fit statistics for outputs generated by estimation functions in the package*

Description

This function computes overall model fit statistics for outputs generated by estimation functions in the package, including `ParMLE` and `JMLE`. The function currently provides the AIC and BIC statistics.

Usage

```
ModelFit(x)
```

Arguments

`x`	The output from the function (The list of all outputs).

Value

`AIC`	The AIC statistic of the overall model. See the reference for details.
`BIC`	The BIC statistic of the overall model. See the reference for details.

References

Chen, J. de la Torre, J., & Zhang, Z. (2013). Relative and absolute fit evaluation in cognitive diagnosis modeling. *Journal of Educational Measurement, 50*, 123-140.

See Also

`ParMLE`, `JMLE`

Examples

```
# See examples in ParMLE and JMLE.
```

ParMLE *Maximum likelihood estimation of item parameters for cognitive diagnostic models.*

Description

This function returns maximum likelihood estimates of item parameters for cognitive diagnostic models when examinee ability patterns are known. This function can either be used independently or called in the JMLE function. Currently supported cognitive diagnostic models include the DINA model, the DINO model, the NIDA model, the G-NIDA model, and the R-RUM model.

Usage

```
ParMLE(Y, Q, alpha, model = c("DINA", "DINO", "NIDA", "GNIDA", "RRUM"))
```

Arguments

Y	A matrix of binary responses. Rows represent persons and columns represent items. 1=correct, 0=incorrect.
Q	The Q-matrix of the test. Rows represent items and columns represent attributes. 1=attribute required by the item, 0=attribute not required by the item.
alpha	Examinee attribute profiles. Rows represent persons and columns represent attributes. 1=examinee masters the attribute, 0=examinee does not master the attribute.
model	Currently support five models: "DINA", "DINO", "NIDA", "GNIDA", and "RRUM". The default is "DINA".

Value

For the DINA, DINO, and NIDA models:

slip	a vector of slip parameters.
guess	a vector of guessing parameters.
se.slip	a vector of the standard errors for slip parameters.
se.guess	a vector of the standard errors for guessing parameters.

For the G-NIDA model:

slip	a matrix (# items by # attributes) of slip parameters.
guess	a matrix (# items by # attributes) of guessing parameters.
se.slip	a matrix (# items by # attributes) of the standard errors for slip parameters.
se.guess	a matrix (# items by # attributes) of the standard errors for guessing parameters.

For the R-RUM model:

pi	a vector of pi parameters for each item.

r	a matrix (# items by # attributes) of r parameters.
se.pi	a vector of the standard errors for pi parameters.
se.r	a matrix (# items by # attributes) of the standard errors for r parameters.

Note that for the G-NIDA model and the R-RUM model, the item parameter estimates and standard errors are not available for the entries where the Q-matrix is 0.

Additionally, for all models:

model	The chosen model.
Q	The Q-matrix of the test.

See Also

JMLE, print.ParMLE

Examples

```
# Generate item and examinee profiles

natt <- 3
nitem <- 4
nperson <- 5
Q <- rbind(c(1, 0, 0), c(0, 1, 0), c(0, 0, 1), c(1, 1, 1))
alpha <- rbind(c(0, 0, 0), c(1, 0, 0), c(0, 1, 0), c(0, 0, 1), c(1, 1, 1))

# Generate DINA model-based response data

slip <- c(0.1, 0.15, 0.2, 0.25)
guess <- c(0.1, 0.15, 0.2, 0.25)
my.par <- list(slip=slip, guess=guess)

data <- matrix(NA, nperson, nitem)
eta <- matrix(NA, nperson, nitem)

for (i in 1:nperson) {
  for (j in 1:nitem) {
  eta[i, j] <- prod(alpha[i,] ^ Q[j, ])
  P <- (1 - slip[j]) ^ eta[i, j] * guess[j] ^ (1 - eta[i, j])
  u <- runif(1)
  data[i, j] <- as.numeric(u < P)
  }
}

# Using the function to estimate item parameters

parMLE.result <- ParMLE(data, Q, alpha, model="DINA")
print(parMLE.result)  # Print the estimated item parameters and standard errors
ItemFit(parMLE.result)
ModelFit(parMLE.result)
```

plot.NPCD *Produce diagnostic plots*

Description

This function produces diagnostic plots of various outputs generated from the functions in this package, including AlphaNP, AlphaMLE, JMLE, andQrefine.

Usage

```
## S3 method for class 'AlphaNP'
plot(x, nperson, cex.main, cex.legend, ...)
## S3 method for class 'AlphaMLE'
plot(x, nperson, cex.main, ...)
## S3 method for class 'JMLE'
plot(x, nperson, cex.main, ...)
## S3 method for class 'Qrefine'
plot(x, filename="Qrefine.plot.png", cex.main,
                cex.lab, cex.axis, cex.legend, ...)
```

Arguments

x	The output from the function (The list of all outputs).
nperson	The choice of examinee to be investigated.
filename	The filename of the plot, ending in ".png". Directory can be included; otherwise the plot will be saved in the working directory.
cex.main	A numerical value giving the amount by which title text should be magnified relative to the default. This starts as 1 when a device is opened.
cex.axis	The magnification to be used for axis annotation.
cex.lab	The magnification to be used for x and y labels.
cex.legend	The magnification to be used for the legend if there is one.
...	Other arguments.

Value

AlphaNP	Bar plot of sorted loss function values (plain, weighted, or penalized Hamming distance) for each candidate attribute profile for the chosen examinee. The bar with the tilted shade is the estimated attribute profile for this examinee, which ideally should have the smallest loss function value.
AlphaMLE	Bar plot of sorted negative log-likelihood function values for each candidate attribute profile for the chosen examinee.The bar with the tilted shade is the estimated attribute profile for this examinee, which ideally should have the smallest value.

JMLE	A pair of barplots. Plot1: Bar plot of unsorted loss function values for each candidate attribute profile for the chosen examinee using the nonparametric estimation; Plot2: bar plot of unsorted negative log-likelihood function values for each candidate attribute profile for the chosen examinee in the last iteration of the JMLE estimation.The bar with the tilted shade is the estimated attribute profile for this examinee by each method, which ideally should have the smallest value.
Qrefine	A panel plot for all items with refined q-vectors. Each subplot shows the unsorted residual sum of squares (RSS) between the real response patterns and the ideal response patterns generated from each possible q-vector. The bar with the sparsely tilted shade is the initial q-vector for this item, and the bar with the densely tilted shade is the refined q-vector for this item, which ideally should have the smallest RSS value.

See Also

AlphaNP, AlphaMLE, JMLE, Qrefine

Examples

```
# See examples in AlphaNP, AlphaMLE, JMLE, and Qrefine.
```

---

print.NPCD *Print outputs generated from the functions in the package.*

Description

This function prints outputs generated from the functions in this package, including AlphaNP, AlphaMLE, ParMLE, JMLE, andQrefine.

Usage

```
## S3 method for class 'AlphaNP'
print(x, ...)
## S3 method for class 'AlphaMLE'
print(x, ...)
## S3 method for class 'ParMLE'
print(x, ...)
## S3 method for class 'JMLE'
print(x, ...)
## S3 method for class 'Qrefine'
print(x, ...)
```

Arguments

x	The output from the function (The list of all outputs).
...	Other arguments.

Value

AlphaNP	The estimated examinee attribute profiles.
AlphaMLE	The estimated examinee attribute profiles.
ParMLE	The estimated item parameters and standard errors.
JMLE	The estimated item parameters and examinee attribute profiles.
Qrefine	The modified Q-matrix and the modified entries

See Also

AlphaNP, AlphaMLE, ParMLE, JMLE, Qrefine

Examples

```
# See examples in AlphaNP, AlphaMLE, ParMLE, JMLE, and Qrefine.
```

Qrefine *Refine the Q-matrix by minimizing the residual sum of square (RSS)*

Description

Refine the Q-matrix by minimizing the residual sum of square (RSS) betweenn the real responses and ideal responses. Examinee attribute profiles are estimated using the nonparametric method (plain Hamming) implemented by AlphaNP.

Usage

```
Qrefine(Y, Q, gate=c("AND", "OR"), max.ite = 50)
```

Arguments

Y	A matrix of binary responses. Rows represent persons and columns represent items. 1=correct, 0=incorrect.
Q	The Q-matrix of the test. Rows represent items and columns represent attributes. 1=attribute required by the item, 0=attribute not required by the item.
gate	"AND": the examinee needs to possess all attributes required by an item in order to answer it correctly; "OR": the examinee needs to possess only one of the attributes required by an item in order to answer it correctly.
max.ite	The maximum number of iterations allowed.

Value

patterns	All possible attribute profiles. Rows represent different patterns of attribute profiles and columns represent attributes. 1=examinee masters the attribute, 0=examinee does not master the attribute.
initial.Q	The initial Q-matrix. Rows represent items and columns represent attributes. 1=attribute required by the item, 0=attribute not required by the item. This is the preliminary Q-matrix to be refined.
initial.class	The row indices of patterns in the initial estimation of examinee attribute profiles.
terminal.class	The The row indices on patterns in the terminal estimation of examinee attribute profiles after the Q-matrix has been refined.
modified.Q	The modified Q-matrix. Rows represent items and columns represent attributes. 1=attribute required by the item, 0=attribute not required by the item.
modified.entries	The modified q-entries. Column 1 is the item ID of the modified entry; column 2 is the attribute ID of the modified entry.

References

Chiu, C. Y. (2013). Statistical Refinement of the Q-matrix in Cognitive Diagnosis. *Applied Psychological Measurement, 37*(8), 598-618.

See Also

AlphaNP, print.Qrefine, plot.Qrefine

Examples

```
# Generate item and examinee profiles

natt <- 3
nitem <- 4
nperson <- 16
Q <- rbind(c(1, 0, 0), c(0, 1, 0), c(0, 0, 1), c(1, 1, 1))
alpha <- rbind(c(0, 0, 0), c(1, 0, 0), c(0, 1, 0), c(0, 0, 1),
  c(1, 1, 0), c(1, 0, 1), c(0, 1, 1), c(1, 1, 1))
alpha <- rbind(alpha, alpha)

# Generate DINA model-based response data

slip <- c(0.1, 0.15, 0.2, 0.25)
guess <- c(0.1, 0.15, 0.2, 0.25)
my.par <- list(slip=slip, guess=guess)

data <- matrix(NA, nperson, nitem)
eta <- matrix(NA, nperson, nitem)

for (i in 1:nperson) {
  for (j in 1:nitem) {
```

```
  eta[i, j] <- prod(alpha[i,] ^ Q[j, ])
  P <- (1 - slip[j]) ^ eta[i, j] * guess[j] ^ (1 - eta[i, j])
  u <- runif(1)
  data[i, j] <- as.numeric(u < P)
  }
}

# Generate misspecified Q-matrix

Q_mis <- Q
Q_mis[c(1,2), 1] <- 1 - Q_mis[c(1,2), 1]

# Run Qrefine and create diagnostic plots

Qrefine.out <- Qrefine(data, Q_mis, gate="AND", max.ite=50)
print(Qrefine.out)
plot(Qrefine.out)
```

Index