Package 'NPCDTools'

November 19, 2026

Title The Nonparametric Classification Methods for Cognitive Diagnosis

Version 1.3.0

Description Statistical tools for analyzing cognitive diagnosis (CD) data collected from small settings using the nonparametric classification (NPCD) framework. The core methods of the NPCD framework includes the nonparametric classification (NPC) method developed by Chiu and Douglas (2013) <DOI:10.1007/s00357-013-9132-9> and the general NPC (GNPC) method developed by Chiu, Sun, and Bian (2018) <DOI:10.1007/s11336-017-9595-4> and Chiu and Köhn (2019) <DOI:10.1007/s11336-019-09660-x>. An extension of the NPCD framework included in the package is the nonparametric method for multiple-choice items (MC-NPC) developed by Wang, Chiu, and Koehn (2023) <DOI:10.3102/10769986221133088>. Functions associated with various extensions concerning the evaluation, validation, and feasibility of the CD analysis are also provided. These topics include the completeness of Q-matrix, Q-matrix refinement method, as well as Q-matrix estimation.

License GPL-3

Encoding UTF-8

Imports GDINA, psych, stats, utils, gtools, Matrix, shiny, MASS

NeedsCompilation no

Depends R (>= 2.10)

LazyData true

Config/roxygen2/version 8.1.0

Author Chia-Yi Chiu [aut, cph],
Weixuan Xiao [aut, cre],
Hans Friedrich Köhn [aut],
Yu Wang [aut],
Xiran Wen [aut]

Maintainer Weixuan Xiao <wx2299@tc.columbia.edu>

Repository CRAN

Date/Publication 2026-11-19 01:00:08 UTC

Contents

AAR . . . 2
bestQperm . . . 3
correction.rate . . . 4
distractor.check . . . 4
GNPC . . . 5
level.x . . . 8
mcdat.generate . . . 10
MCNPC . . . 11
mcQ.completeness . . . 12
mcQ.generate . . . 14
NPC . . . 14
PAR . . . 16
plot.GNPC . . . 17
print.distractor.check . . . 19
print.GNPC . . . 19
print.NPC . . . 20
print.Qcompleteness . . . 20
print.Qrefine . . . 21
Q.completeness . . . 21
Q.generate . . . 23
Q.implausible . . . 24
Q.improper . . . 25
QR . . . 25
Q_Ozaki . . . 27
retention.rate . . . 28
RR . . . 29
run_gnpc_app . . . 29
summary.GNPC . . . 30
summary.NPC . . . 31
TSQE . . . 31

Index **34**

AAR *Attribute-wise agreement rate*

Description

The function is used to compute the attribute-wise agreement rate between two sets of attribute profiles. They need to have the same dimensions.

Usage

```
AAR(x, y)
```

Arguments

x	One set of attribute profiles
y	The other set of attribute profiles

Value

The function returns the attribute-wise agreement rate between two sets of attribute profiles.

See Also

PAR

Examples

```
# see examples used for GNPC.
```

---

bestQperm *Column permutation of a Q-matrix with respect to a benchmark Q-matrix.*

Description

Function bestQperm is used to permute the columns of a Q-matrix so that the order of the columns best matches that of the benchmark Q-matrix. This function is useful in a Q-matrix estimation process.

Usage

```
bestQperm(Q, bench.Q)
```

Arguments

Q	The targeted Q-matrix.
bench.Q	The benchmark Q-matrix.

Value

The function returns a Q-matrix in which the order of the columns best matches that of the benchmark Q-matrix.

Examples

```
# See examples used for TSQE.
```

correction.rate *Correction rate of a Q-matrix refinement method*

Description

This function computes the proportion of corrected q-entries that were originally misspecified in the provisional Q-matrix. This function is used only when the true Q-matrix is known.

Usage

```
correction.rate(ref.Q = ref.Q, mis.Q = mis.Q, true.Q = true.Q)
```

Arguments

ref.Q	The $J \times K$ refined binary Q-matrix.
mis.Q	A $J \times K$ binary provisional Q-matrix.
true.Q	The $J \times K$ binary true Q-matrix.

Value

The function returns a value between 0 and 1 representing the proportion of corrected q-entries in ref.Q that were originally misspecified in mis.Q.

Examples

```
# See examples used for QR function.
```

distractor.check *Detect the implausible and improper distractors in a Q-Matrix for multiple-choice items*

Description

Function distractor.check is used to assess whether the distractors of a given Q-matrix for multiple-choice items are plausible and/or proper.

Usage

```
distractor.check(mcQ, key = NULL)
```

Arguments

mcQ	The given Q-matrix for multiple-choice items. It has to be organized in the following manner. The Q-matrix should contain $K + 2$ columns and $\sum_{j=1}^{J} H_j$ rows, where H_j is the number of coded options for item j. Among the H_j rows for item j, the first row is the key, followed by the coded distractors. The first column of the Q-matrix lists the ID of the items and second column indicates the corresponding options of the coded options. If the Q-matrix is not organized in such a way, an argument key (see below) that indicates the options that the keys are located needs to be given.
key	A vector that indicates the options where the keys are located.

Value

A list with class "distractor.check" containing:

not.plausible A matrix indicating items and options that are not plausible, or NULL if all items are plausible.

not.proper A vector of item IDs that are not proper, or NULL if all items are proper.

all.plausible Logical; TRUE if all items are plausible.

all.proper Logical; TRUE if all items are proper.

References

Chiu, C.-Y., Köhn, H. F. & Wang, Y. (Online first). Plausible and proper multiple-choice items for diagnostic classification. *Psychometrika*.

Examples

```
## Not run:
library(NPCDTools)
Q1 <- Q_Ozaki
distractor.check(Q1)

Q2 <- GDINA::sim10MCDINA2$simQ
key <- c(1, 2, 4, 1, 1, 3, 2, 4, 1, 4)
distractor.check(Q2, key)

## End(Not run)
```

GNPC *Estimation of examinees' attribute profiles using the GNPC method*

Description

Function GNPC is used to estimate examinees' attribute profiles using the general nonparametric classification (GNPC) method (Chiu et al., 2018; Chiu & Köhn, 2019). It can be used with data conforming to any cognitive diagnosis models (CDMs).

Usage

```
GNPC(
  Y,
  Q,
  fixed.w = FALSE,
  initial.dis = "hamming",
  initial.gate = "AND",
  max.iter = 1000,
  tol = 0.001,
  track.convergence = TRUE
)
```

Arguments

`Y`	A $N \times J$ binary data matrix consisting of the responses from N examinees to J items.
`Q`	A $J \times K$ binary Q-matrix where the entry q_{jk} describing whether the kth attribute is required by the jth item.
`fixed.w`	`TRUE` or `FALSE`. When `TRUE` is specified, fixed weights are used as the initial weights to compute fixed-weight ideal response and there is no need to use NPC to produce the initial values for examinees' attribute profiles. Hence, `initial.dis` and `initial.gate` are turned off. The default value is `FALSE`.
`initial.dis`	The type of distance used in the `NPC` to carry out the initial attribute profiles for the GNPC method. Allowable options are `"hamming"` and `"whamming"` representing the Hamming and the weighted Hamming distances, respectively.
`initial.gate`	The type of relation between examinees' attribute profiles and the items. Allowable relations are `"AND"` and `"OR"`, representing the conjunctive and disjunctive relations, respectively.
`max.iter`	Maximum number of iterations allowed. Default is 1000.
`tol`	Convergence tolerance. The algorithm stops when the proportion of examinees whose classification changes is less than this value. Default is 0.001.
`track.convergence`	Logical. If `TRUE`, detailed convergence history is tracked and returned for diagnostic purposes. Default is `TRUE`.

Value

The function returns a list with the following components:

att.est A $N \times K$ matrix of estimated attribute profiles for examinees

class A vector of length N containing the estimated class memberships

ideal.response A $2^K \times J$ matrix of weighted ideal responses

weight A $2^K \times J$ matrix of weights used to compute the weighted ideal responses

Y The response matrix used in the analysis

Q The Q-matrix used in the analysis

pattern All candidate attribute profiles

settings Initialization, iteration, tolerance, and convergence-tracking settings

algorithm.status A list containing the number of iterations, convergence indicator, and the number and proportion of changed classifications at the final iteration

convergence (Only if `track.convergence = TRUE`) A list containing:

- `iteration`: Vector of iteration numbers
- `prop.change`: Proportion of examinees whose classification changed at each iteration
- `total.distance`: Total squared distance between observed and weighted ideal responses
- `n.iter`: Total number of iterations until convergence
- `converged`: Logical indicating whether the algorithm converged within `max.iter`

Details

A weighted ideal response $\eta^{(w)}$, defined as the convex combination of $\eta^{(c)}$ and $\eta^{(d)}$, is used in the GNPC method to compute distances. Suppose item j requires $K_j^* \leq K$ attributes that, without loss of generality, have been moved to the first K_j^* positions of the item attribute vector $\boldsymbol{q_j}$. For each item j and latent class $\mathcal{C}_l$, the weighted ideal response $\eta_{lj}^{(w)}$ is defined as the convex combination $\eta_{lj}^{(w)} = w_{lj}\eta_{lj}^{(c)} + (1 - w_{lj})\eta_{lj}^{(d)}$ where $0 \leq w_{lj} \leq 1$. The distance between the observed responses to item j and the weighted ideal responses $w_{lj}^{(w)}$ of examinees in $\mathcal{C}_l$ is defined as the sum of squared deviations: $d_{lj} = \sum_{i \in \mathcal{C}_l}(y_{ij} - \eta_{lj}^{(w)})^2$. $\hat{w}_{lj}$ can then be obtained by minimizing d_{lj}, which can then be used to compute $\hat{\eta}_{lj}$.

After all the $\hat{\eta}_{lj}$ are obtained, examinees' attribute profiles $\boldsymbol{\alpha}$ can be estimated by minimizing the loss function $\hat{d}_{lj} = \sum_{i \in \mathcal{C}_l}(y_{ij} - \hat{\eta}_{lj}^{(w)})^2$

The algorithm iteratively updates the weighted ideal responses and reclassifies examinees until convergence is achieved. The stopping criterion is based on the proportion of examinees whose classification changes between consecutive iterations: $\frac{\sum_{i=1}^{N} I\left[\alpha_i^{(t)} \neq \alpha_i^{(t-1)}\right]}{N} < \epsilon$ where ϵ is the tolerance level (default = 0.001).

The default initial values of $\boldsymbol{\alpha}$ are obtained by using the NPC method. Chiu et al. (2018) suggested another viable alternative for obtaining initial estimates of the proficiency classes by using an ideal response with fixed weights defined as $\eta_{lj}^{(fw)} = \frac{\sum_{k=1}^{K} \alpha_k q_{jk}}{K}\eta_{lj}^{(c)} + (1 - \frac{\sum_{k=1}^{K} \alpha_k q_{jk}}{K})\eta_{lj}^{(d)}$.

References

programs: The general nonparametric classification method. *Psychometrika*, *83*(2), 355–375. doi:10.1007/s1133601795954

classification method. *Psychometrika*, *84*(3), 830–845. doi:10.1007/s1133601909660x

Examples

```
## Not run:
# Example 1: Basic usage
library(GDINA)
set.seed(123)
N <- 500
```

```
Q <- sim30GDINA$simQ
gs <- data.frame(guess = rep(0.2, nrow(Q)), slip = rep(0.2, nrow(Q)))
sim <- simGDINA(N, Q, gs.parm = gs, model = "DINA")
Y <- extract(sim, what = "dat")
alpha <- extract(sim, what = "attribute")

# Analyze data using GNPC
result <- GNPC(Y, Q, initial.dis = "hamming", initial.gate = "AND")

# View results
head(result$att.est)
table(result$class)

# Plot overall convergence
plot(result)

# Plot individual examinee's convergence
plot(result, type = "individual", examinee.id = 1, true.alpha = alpha[1, ])

# Check attribute agreement rate
PAR(alpha, result$att.est)
AAR(alpha, result$att.est)

# Example 2: Without convergence tracking (Convergence tracking is only used for the GNPC plots.)
result2 <- GNPC(Y, Q, track.convergence = FALSE)

## End(Not run)
```

`level.x` *Identify Level-x or Level-0 Designs for Items*

Description

The `level.x` function searches over all items in a given test and identifies the level-x and level-0 designs, along with their associated attributes, for items involving single-attribute q-vectors.

Usage

```
level.x(mcQ)
```

Arguments

`mcQ`	The given Q-matrix for MC items to be checked. It has to be organized in the following manner. The Q-matrix should contain $K + 2$ columns and $\sum_{j=1}^{J} H_j$ rows, where H_j is the number of coded options for item j. Among the H_j rows for item j, the first row is the key, followed by the coded distractors. The first column of the Q-matrix lists the ID of the items and second column indicates the corresponding options of the coded options. The entries must be binary (0/1).

Details

In the MC-DINA setting, the single-attribute q-vectors may be associated with either keys or coded distractors, and several of them may occur within a single item. Thus, the presence of all K unit vectors is necessary but is no longer, by itself, sufficient for completeness. As a further complication, some configurations allow an item to distinguish mastery from nonmastery of attribute k (or multiple attributes) directly; other configurations cannot make this distinction unless the information supplied by additional items is considered jointly. To characterize this organization, Wang and Chiu (conditionally accepted) introduced the concepts of level-x and level-0 item designs, which play a central role in the completeness conditions for MC Q-matrices.

Value

The function returns a matrix with the following components:

item Item number.

level Design level.

attribute Attribute number.

References

Wang, Y. & Chiu, C.-Y. (conditional accepted). Q-matrix completeness for multiple-choice items in cognitive diagnosis. *Psychometrika*.

Chiu, C.-Y., Köhn, H. F. & Wang, Y. (2026). Plausible and proper multiple-choice items for diagnostic classification. *Psychometrika, 91*(1), 177-198.

Examples

```
## Not run:
# Example 1
Q1 <- data.frame(
  Item   = c(1, 1, 1, 2, 2, 3, 3, 4, 5, 5, 5),
  Option = c(4, 6, 1, 5, 3, 6, 2, 3, 2, 3, 4),
  Att1   = c(1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0),
  Att2   = c(1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1),
  Att3   = c(1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0),
  Att4   = c(0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0)
)
level.x(Q1)

# Example 2: Add two items to Q1
Q2 = rbind(Q1, c(6, 1, 0, 0, 1, 0), c(6, 2, 1, 1, 0, 0),
c(7, 1, 1, 1, 0, 0), c(7, 2, 0, 0, 0, 1))
level.x(Q2)

## End(Not run)
```

`mcdat.generate` *Generation of MC Responses*

Description

This function generates MC responses conforming to the MC-DINA model with relaxed MC Q-matrix.

Usage

```
mcdat.generate(
  mcQ,
  N,
  H,
  att.structure = "unif",
  cor.att = 0.5,
  lambda = rep(0.85, J)
)
```

Arguments

`mcQ`	A $(\sum_{j=1}^{J} H_j^*) \times (K+2)$ Q-matrix for MC items. The first two columns are item ID and option ID of the coded options.
`N`	The number of examinees
`H`	The number of maximal attributes required by the items. H takes values 3, 4, 5, or 6.
`att.structure`	The structure of attribute profiles. Possible options are `"unif"` (discrete uniform distribution), `"mvn"` (multivariate normal threshold model), and `"HO"` (higher-order logistic model).
`cor.att`	If `att.structure = "mvn"`, `cor.att` is used to specify the correlation between attributes.
`lambda`	A J-dimensional vector, where the jth entry represents the probability of a match between an examinee's ideal response and his/her observed response to item j.

Value

The function returns

dat A $N \times J$ matrix storing the generated responses

att Examinees' attribute profiles ($N \times K$) used to generate the responses

prob.persons A list storing the probabilities that the examinees are attracted by the options of the items

MCNPC *The Nonparametric Classification Method for Multiple-Choice Items*

Description

Function MCNPC is used to estimate examinees' attribute profiles using the general nonparametric classification method for multiple-choice items (MC-NPC). Different from the traditional multiple-choice items where distractors are not coded, Some or all of the distractors are coded and responses are polytomous. Studies have shown that such a format will increase the classification rates compared to the binary format.

Usage

```
MCNPC(dat, mcQ, H, LS = NULL)
```

Arguments

dat	A $N \times J$ polytomous data matrix consisting of the responses from N examinees to J items.
mcQ	A $\sum_{j=1}^{J} H_j \times (K+2)$ binary matrix where H_j is the number of coded options for item j. The first and second columns contain the IDs of the items and the IDs of the coded options. Columns 3 to $K+2$ contains the q-entries of the coded options. For each item, the first row is designated to the key and the others are to the coded distractors.
H	The number of options. By default, all items have the same number of options.
LS	A matrix representing the possible attribute patterns in the data. The default (= NULL) is the 2^K attribute patterns expanded by the K binary attributes. It can also take user-specified attribute patterns.

Value

The function returns

att The estimated attribute patterns

classID The estimated class memberships

distance order The order of latent classes based on the corresponding distances

distance The calculated distances associated with the possible latent classes

latentclass The possible latent classes in the data

`mcQ.completeness` *Check the completeness status of a Q-matrix with multiple-choice items*

Description

`mcQ.completeness` is primarily used to examine whether a given MC Q-matrix is complete when the data conform to the MC-DINA model. The function also provides useful diagnostic information for constructing a complete MC Q-matrix.

Usage

```
mcQ.completeness(mcQ)
```

Arguments

`mcQ`	The given Q-matrix for MC items to be checked. It has to be organized in the following manner. The Q-matrix should contain $K + 2$ columns and $\sum_{j=1}^{J} H_j$ rows, where H_j is the number of coded options for item j. Among the H_j rows for item j, the first row is the key, followed by the coded distractors. The first column of the Q-matrix lists the ID of the items and second column indicates the corresponding options of the coded options. The entries must be binary (0/1).

Details

While the completeness conditions for the binary DINA model are well established, the corresponding conditions for MC items are more involved because an MC item may contain multiple q-vectors associated with its key and coded distractors. Wang and Chiu (conditionally accepted) extended the theory of Q-matrix completeness from binary items to MC items under the MC-DINA model and establish sufficient and necessary conditions for MC Q-completeness in their work. Their theorem states that when data conform to the MC-DINA model, the MC Q-matrix is complete if and only if the Q-matrix includes all single-attribute q-vectors and either (C1) or (C2) holds:

(C1). At least one item j containing it has a level-x design.

(C2). All items containing it have the level-0 design, and Lemma 1 is satisfied.

The theoretical results provide necessary and sufficient conditions, meaning that no other conditions are required to ensure the completeness of the MC Q-matrix.

Value

The function returns a list with the following components:

levelx.option If the Q-matrix is complete and the identifications of the attributes involve level-x items, the function returns a data frame identifying the items and their associated level-x designs and attributes. The data frame contains the following columns:

attribute Attribute number.

level Design level.

item Item number.

levelx.option If the Q-matrix is complete and the identifications of the attributes involve both level-x and level-0 items, the function returns the result of `levelx.option` as well as a data frame indicating the candidate items that could be added to the original level-0 item and whether the algorithm terminates. The data frame contains the following columns:

attribute Attribute number.

add.item Candidate items that could be added to the original level-0 item.

end Indication of whether the algorithm terminates.

The function also prints the status message to the console as a side effect.

References

Wang, Y. & Chiu, C.-Y. (conditional accepted). Q-matrix completeness for multiple-choice items in cognitive diagnosis. *Psychometrika*.

Chiu, C.-Y., Köhn, H. F. & Wang, Y. (2026). Plausible and proper multiple-choice items for diagnostic classification. *Psychometrika, 91*(1), 177-198.

Examples

```
# Example 1: An incomplete MC Q-matrix
Q1 <- data.frame(
  Item   = c(1, 1, 1, 2, 2, 3, 3, 4, 5, 5, 5),
  Option = c(4, 6, 1, 5, 3, 6, 2, 3, 2, 3, 4),
  Att1   = c(1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0),
  Att2   = c(1, 1, 0, 1, 0, 0, 1, 0, 0, 1, 1),
  Att3   = c(1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0),
  Att4   = c(0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0)
)

# Attribute 4 has no single-attribute q-vector.
mcQ.completeness(Q1)

# Example 2: Add two items to obtain a complete MC Q-matrix
# Each item's first row represents its key.
Q2 <- rbind(
  Q1,
  c(6, 1, 0, 1, 1, 0),
  c(6, 2, 0, 0, 1, 0),
  c(7, 1, 0, 0, 0, 1),
  c(7, 2, 1, 0, 0, 0)
)

out <- mcQ.completeness(Q2)
out$levelx.option
out$level0.option
```

mcQ.generate *Generation of Plausible and Proper Q-matrix with Coded Distractors*

Description

This function generates a useful and proper Q-matrix based on a dichotomous Q-matrix.

Usage

```
mcQ.generate(H, Q)
```

Arguments

H	The number of maximal attributes required by the items. H takes values 3, 4, 5, or 6.
Q	The dichotomous Q-matrix that the MC Q-matrix is based on

Value

The function returns

mcq The useful and proper Q-matrix with coded distractors

key The keys of the MC items

H_star The number of coded options

NPC *Estimation of examinees' attribute profiles using the NPC method*

Description

The function estimates examinees' attribute profiles using the nonparametric classification (NPC) method (Chiu & Douglas, 2013). An examinee's attribute profile is estimated by minimizing the distance between the observed and ideal item responses.

Usage

```
NPC(
  Y,
  Q,
  distance = c("hamming", "whamming", "penalized"),
  gate = c("AND", "OR"),
  wg = 1,
  ws = 1
)
```

Arguments

`Y`	A $N \times J$ binary data matrix consisting of the responses from N examinees to J items.
`Q`	A $J \times K$ binary Q-matrix where the entry q_{jk} describes whether the kth attribute is required by the jth item.
`distance`	The type of distance used to compute the loss function. The possible options include (i) "`hamming`" representing the plain Hamming distance method, (ii) "`whamming`" representing the Hamming distance weighted by the inverse of item variance, and (iii) "`penalized`" representing the Hamming distance weighted by the inverse of item variance and specified penalizing weights for guess and slip.
`gate`	A character string specifying the type of gate. The possible options include "`AND`" and "`OR`" standing for conjunctive and disjunctive gate, respectively.
`wg`	Additional argument for the "penalized" method. It is a weight assigned to guesses in the DINA or DINO models. A large value of weight results in a stronger impact on the distance (i.e., larger loss function values) caused by guessing.
`ws`	Additional input for the "penalized" method. It is the weight assigned to slips in the DINA or DINO models. A large value of weight results in a stronger impact on the distance (i.e., larger loss function values) caused by slipping.

Value

The function returns a series of outputs, including:

att.est A $N \times K$ matrix representing the estimated attribute profiles. 1 = examinee masters the attribute, 0 = examinee does not master the attribute.

est.ideal A $N \times J$ matrix indicating the estimated ideal response to all items from all examinees. 1 = correct, 0 = incorrect.

est.class A N-dimensional vector showing the class memberships for all examinees.

n.tie The number of ties in the Hamming distance among the candidate attribute profiles for each person. When ties occur, one of the tied attribute profiles is randomly chosen.

pattern All possible attribute profiles in the latent space.

loss.matrix A $2^K \times N$ matrix containing the values of the loss function (the distances) between each examinee's observed response vector and the 2^K ideal response vectors.

Details

The nonparametric classification (NPC) method (Chiu & Douglas, 2013) assigns examinees to the proficiency classes they belong to by comparing their observed item response patterns with each of the ideal item response patterns of the 2^K proficiency classes. When there is no data perturbation, an examinee's ideal response pattern corresponding to the examinee's true attribute pattern and his/her observed item response patterns are identical, and thus the distance between them is 0. When data perturbations are small, this ideal response pattern remains the one most similar to the observed response pattern, which is exactly the setup of data conforming to the DINA or DINO model. Hence, based on this rationale, an examinee's attribute profile is obtained by minimizing

the distance between the observed and the ideal item response patterns. The nonparametric nature of the NPC method furthermore makes it suitable for data obtained from small-scale settings.

References

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. doi:10.1007/s0035701391329

See Also

GNPC

Examples

```
## Not run:
library(GDINA)
N <- 500
Q <- sim30GDINA$simQ
gs <- data.frame(guess = rep(0.2, nrow(Q)), slip = rep(0.2, nrow(Q)))
sim <- simGDINA(N, Q, gs.parm = gs, model = "DINA")
Y <- extract(sim, what = "dat")
alpha <- extract(sim, what = "attribute")

# Estimate attribute profiles using NPC
result <- NPC(Y, Q, distance = "hamming", gate = "AND")
print(result)
result$att.est

# Check attributed agreement rate
PAR(alpha, result$att.est)
AAR(alpha, result$att.est)

## End(Not run)
```

PAR *Pattern-wise agreement rate*

Description

The function is used to compute the pattern-wise agreement rate between two sets of attribute profiles. They need to have the same dimensions.

Usage

```
PAR(x, y)
```

Arguments

x	One set of attribute profiles
y	The other set of attribute profiles

Value

The function returns the pattern-wise agreement rate between two sets of attribute profiles.

See Also

AAR

Examples

```
# see examples used for GNPC.
```

`plot.GNPC` *Plot Diagnostics for GNPC*

Description

This function gives two types of diagnostic plots for the outcomes of the GNPC algorithm.

The type = "convergence" option gives two graphs: The upper panel displays the trajectory of the proportion of membership switches and the lower panel shows the total squared distance along with the iterations. They illustrate the detailed information about how and whether the algorithm has converged.

The type = "individual" option returns a sequence of squared distances for a single examinee and the estimates of the examinee's attribute profile along with the iterations. The plots allow users to investigate how the algorithm arrives at its final classifications. In simulation studies, the true attribute profile can be provided as a reference.

Usage

```
## S3 method for class 'GNPC'
plot(
  x,
  type = c("convergence", "individual"),
  examinee.id = NULL,
  true.alpha = NULL,
  top.n.pattern = NULL,
  ...
)
```

Arguments

x	An object of class "GNPC" when track.convergence = TRUE.
type	"convergence" (default) for overall convergence diagnostics, or "individual" for a single examinee's estimation trajectory.
examinee.id	An integer indicating which examinee to be plotted. This argument is required if "individual" is specified.

`true.alpha`	A numeric vector of length K when the true attribute profile of the examinee is available (optional; usually used for simulation studies).
`top.n.pattern`	An integer specifying the maximum number of patterns to be displayed. The default is `min(2^K, 6)`. The GNPC estimate and true pattern (if provided) are always included.
`...`	Additional arguments passed to `plot`.

Details

For `"individual"` plots, the visual elements are:

- **Red line**: the attribute pattern ultimately selected by GNPC.
- **Black line**: the true attribute pattern (only when `true.alpha` is provided). A small vertical jitter is applied when it overlaps with the red line.
- **Other colored lines**: the most competitive candidate patterns, selected by proximity at the final iteration.
- **Filled circle at each iteration**: indicates which attribute profile GNPC assigned to the examinee at each iteration, drawn in the corresponding line's color. The line for the true attribute profile always has black circles at every iteration as a fixed reference.

See Also

`GNPC`

Examples

```
## Not run:
library(GDINA)
set.seed(123)
N <- 500
Q <- sim30GDINA$simQ
gs <- data.frame(guess = rep(0.2, nrow(Q)), slip = rep(0.2, nrow(Q)))
sim <- simGDINA(N, Q, gs.parm = gs, model = "DINA")
Y <- extract(sim, what = "dat")
alpha <- extract(sim, what = "attribute")

# Analyze data using GNPC
result <- GNPC(Y, Q, initial.dis = "hamming", initial.gate = "AND")

# Convergence
plot(result)

# Individual with true attribute profile (simulation)
plot(result, type = "individual", examinee.id = 1, true.alpha = alpha[1, ])

# Individual without true attribute profile (real data)
plot(result, type = "individual", examinee.id = 1)

## End(Not run)
```

print.distractor.check *Print the Summary of Distractor Check Results*

Description

Prints a summary of whether the distractors of a given Q-matrix for multiple-choice items are plausible and/or proper.

Usage

```
## S3 method for class 'distractor.check'
print(x, ...)
```

Arguments

x	An object of class "distractor.check" returned by distractor.check.
...	Additional arguments (currently not used).

print.GNPC *Print GNPC Classification Results*

Description

Prints concise data, convergence, and classification information for a GNPC object.

Usage

```
## S3 method for class 'GNPC'
print(x, ...)
```

Arguments

x	An object of class GNPC.
...	Additional arguments (not used).

print.NPC *Print NPC Classification Results*

Description

Prints concise data, classification, and tie information for an NPC object.

Usage

```
## S3 method for class 'NPC'
print(x, ...)
```

Arguments

x	An object of class "NPC".
...	Additional arguments (not used).

print.Qcompleteness *Print Summary of a Qcompleteness Object*

Description

Print method for objects of class "Qcompleteness".

Usage

```
## S3 method for class 'Qcompleteness'
print(x, ...)
```

Arguments

x	An object of class "Qcompleteness"
...	Additional arguments (not used)

print.Qrefine *Print Summary of the Q-Matrix Refinement Result*

Description

Prints a summary of the Q-matrix refinement process, including which entries were modified and the final refined Q-matrix.

Usage

```
## S3 method for class 'Qrefine'
print(x, ...)
```

Arguments

x	An object of class Qrefine.
...	Additional arguments passed to print methods.

Q.completeness *Check the completeness status of a binary Q-matrix*

Description

Q.completeness is used to examine whether a given Q-matrix is complete when data conform to a specified CDM. A Q-matrix is said to be complete if it allows for the unique identification of all possible attribute profiles among examinees. So far, the function can only be used for a binary Q-matrix with binary responses.

Usage

```
Q.completeness(raw.Q, model = NULL)
```

Arguments

raw.Q	The Q-matrix that is to be checked, where J is the number of items and K is the number of attributes. It must be a binary (0/1) matrix or data frame that can be coerced to a matrix.
model	Character string specifying the cognitive diagnosis model. Valid options are "DINA", "DINO", or "General" (for general CDMs including G-DINA, LCDM, etc.). Case-insensitive. If not specified or invalid, the default is "General" with a warning.

Details

The conditions for one Q-matrix completeness are model-dependent: a Q-matrix may be complete for one CDM but incomplete for another. This function implements the theoretical work developed by Chiu et al. (2009) and Köhn and Chiu (2017).

For DINA and DINO models:

A Q-matrix is complete if and only if it contains all K single-attribute items (Chiu et al., 2009).

For More General CDMs:

The function implements a sequential procedure based on Theorems 3-4 and Propositions 1-2 in the work by Köhn and Chiu (2017).

1. If Q contains all K single-attribute items, it is complete (Proposition 1).
2. If Q has rank $< K$, it is incomplete (Theorem 3).
3. For full-rank Q-matrices without all single-attribute items, the function examines non-nested attribute pairs using indicator vectors to determine if distinct expected response patterns $\mathbf{S}(\boldsymbol{\alpha})$ can be guaranteed.

The theoretical framework establishes the sufficient conditions for Q completeness, which means completeness implies distinct expected item response patterns for all 2^K possible attribute profiles.

Value

A list of class "`Qcompleteness`" containing:

`is_complete`	Logical value indicating completeness: `TRUE` if complete, `FALSE` if incomplete, `NA` if uncertain.
`status`	Character string: "`complete`", "`incomplete`", or "`uncertain`".
`message`	Character string with detailed explanation of the result.
`model`	The CDM used for assessment.
`K`	Number of attributes in the Q-matrix.
`J`	Number of items in the Q-matrix.

The function also prints the status message to the console as a side effect.

References

Chiu, C.-Y., Douglas, J. A., & Li, X. (2009). Cluster analysis for cognitive diagnosis: Theory and applications. *Psychometrika, 74*(4), 633-665. doi:10.1007/s1133600991250

Köhn, H.-F., & Chiu, C.-Y. (2017). A procedure for assessing the completeness of the Q-matrices of cognitively diagnostic tests. *Psychometrika, 82*(1), 112-132. doi:10.1007/s1133601695367

Köhn, H.-F., & Chiu, C.-Y. (2018). How to build a complete Q-matrix for a #' cognitively diagnostic test. *Journal of Classification, 35*(2), 273-299. doi:10.1007/s0035701892550

Examples

```
## Not run:
# Example 1: Complete Q-matrix for DINA model
# (contains all 3 single-attribute items)
Q1 <- matrix(c(1, 0, 0,
               0, 1, 0,
               0, 0, 1,
               1, 1, 0,
               1, 0, 1), ncol = 3, byrow = TRUE)
result1 <- Q.completeness(Q1, model = "DINA")
print(result1$is_complete)  # TRUE

# Example 2: Incomplete Q-matrix for DINA model
# (missing single-attribute items)
Q2 <- matrix(c(1, 1, 0,
               1, 0, 1,
               0, 1, 1), ncol = 3, byrow = TRUE)
result2 <- Q.completeness(Q2, model = "DINA")
print(result2$is_complete)  # FALSE

# Example 3: Check completeness for general CDM
Q3 <- matrix(c(1, 0, 0,
               0, 1, 0,
               0, 0, 1,
               1, 1, 0,
               1, 0, 1,
               0, 1, 1), ncol = 3, byrow = TRUE)
result3 <- Q.completeness(Q3, model = "General")

## End(Not run)
```

`Q.generate` *Generation of dichotomous Q-matrix*

Description

The function generates a complete Q-matrix based on a pre-specified probability that each q-entry equals 1.

Usage

```
Q.generate(K, J, p, single.att = TRUE)
```

Arguments

`K`	The number of attributes.
`J`	The number of items.
`p`	The probability that each q-entry equals 1.
`single.att`	Whether all the single-attribute patterns are included. If `TRUE`, the completeness of the Q-matrix is guaranteed.

Value

The function returns a dichotomous Q-matrix. A complete Q-matrix is the default unless `single.att = F` is specified.

See Also

`Q.completeness`

Examples

```
## Not run:
# Example 1: A complete Q-matrix with items requiring fewer attributes.
Q1 = Q.generate(3, 20, 0.5, single.att = TRUE)

# Example 2: A Q-matrix with items requiring more attributes but completeness is not guaranteed.
Q2 = Q.generate(5, 30, 0.6, single.att = FALSE)

## End(Not run)
```

Q.implausible *Generation of a Q-matrix that contains implausible MC items*

Description

This function turns a proper and plausible Q-matrix into an implausible Q-matrix with the user-specified number of implausible distractors.

Usage

```
Q.implausible(Q, n.implausible)
```

Arguments

`Q`	A proper and plausible Q-matrix for MC items.
`n.implausible`	The number of items that have implausible distractors.

Value

The function returns

Q.implausible The generated Q-matrix

item.implausible The ID of the items that have implausible distractors.

References

Chiu, C.-Y., Köhn, H. F. & Wang, Y. (Online first). Plausible and proper multiple-choice items for diagnostic classification. *Psychometrika*. doi:10.1017/psy.2025.10074

`Q.improper` *Generation of a Q-matrix with improper MC items that are not proper based on a 'plausible and proper' Q*

Description

This function turns a proper and plausible Q-matrix into an improper Q-matrix with the user-specified number of improper distractors.

Usage

```
Q.improper(Q, n.improper)
```

Arguments

`Q`	A proper and plausible Q-matrix for MC items.
`n.improper`	The number of improper distractors.

Value

The function returns

Q.improper The improper Q-matrix generated from Q

item.improper The ID of the items that are improper.

References

Chiu, C.-Y., Köhn, H. F. & Wang, Y. (Online first). Plausible and proper multiple-choice items for diagnostic classification. *Psychometrika*. doi:10.1017/psy.2025.10074

`QR` *Q-matrix refinement method*

Description

The `QR` function refines a provisional Q-matrix by minimizing the residual sum of squares (RSS) between the observed and ideal item responses across all possible q-vectors, given the estimates of examinees' attribute profiles.

Usage

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

Arguments

`Y`	A $N \times J$ matrix of binary responses (1=correct, 0=incorrect). Rows represent persons and columns represent items.
`Q`	A $J \times K$ provisional Q-matrix to be refined. Rows represent items and columns represent attributes.
`gate`	A string, `"AND"` or `"OR"`. `"AND"` is specified when a conjunctive relation between the attributes an examinee possesses and the attributes required by an item is assumed. `"OR"` is specified when a disjunctive relation between the attributes an examinee possesses and the attributes required by an item is assumed.
`max.ite`	The number of iterations to run until the RSS's of all items are stationary.

Details

This function implements the Q-matrix refinement (QR) method developed by Chiu (2013). The NPC method (Chiu & Douglas, 2013) is first used to classify examinees and the best q-vector for an item is identified by minimizing its RSS. Specifically, the RSS of item j for examinee i is defined as

$$RSS_j = \sum_{m=1}^{2^K} \sum_{i \in C_m} (Y_{ij} - \eta_{jm})^2,$$

where C_m for $m = 1, \ldots, 2^K$ is the mth proficiency class, and N is the number of examinees. Chiu (2013) proved that the expected value of RSS_j corresponding to the correct q-vector is the minimum among the $2^K - 1$ candidates.

Value

A list containing:

`initial.class`	Initial classifications of examinees
`terminal.class`	Terminal classification of examinees
`modified.Q`	The modified Q-matrix
`modified.entries`	The modified q-entries

References

Chiu, C. Y. (2013). Statistical Refinement of the Q-matrix in Cognitive Diagnosis. *Applied Psychological Measurement, 37*(8), 598-618. doi:10.1177/0146621613488436

See Also

NPC

Examples

```
## Not run:
## Generate data
library(GDINA)
N = 500
Q = sim30GDINA$simQ
J = nrow(Q)
K= ncol(Q)
gs = data.frame(guess = rep(0.2,J), slip = rep(0.2,J))
sim = simGDINA(N, Q, gs.parm = gs, model = "DINA")
Y = extract(sim,what = "dat")

## Randomly generate a misspecified Q with 20% of misspecifications
mis.Q = matrix(0, J, K)
while (any(rowSums(mis.Q)==0)==T){
  mis.q = sample(J*K, J*K*0.2) ## percentage of misspecified q
  ind = arrayInd(mis.q, dim(Q))
  mis.Q = Q
  mis.Q[ind] = 1-mis.Q[ind]
}

## Refine the misspecified Q-matrix
ref = QR(Y, mis.Q)
ref.Q = ref$modified.Q

## Compute the entry-wise and item-wise recovery rates
rr = RR(ref.Q, Q)
rr$entry.wise
rr$item.wise

## Compute the retention rate
retention.rate(ref.Q, mis.Q, Q)

## Compute the correction rate
correction.rate(ref.Q, mis.Q, Q)

## End(Not run)
```

Q_Ozaki *Q-Matrix from Ozaki (2015)*

Description

A Q-matrix for 30 multiple-choice items measuring 5 attributes, originally used in Ozaki (2015) to demonstrate structured MC-DINA models. Items 1–10 are single-attribute items with no coded distractors. Items 11–20 have one coded distractor each, and items 21–30 have two or three coded distractors.

Usage

```
Q_Ozaki
```

Format

A data frame with 66 rows and 7 columns:

V1 Item index (1–30).

V2 Option index. For each item, the first row is the key; subsequent rows are coded distractors.

V3 Attribute 1 indicator (0 or 1).

V4 Attribute 2 indicator (0 or 1).

V5 Attribute 3 indicator (0 or 1).

V6 Attribute 4 indicator (0 or 1).

V7 Attribute 5 indicator (0 or 1).

References

Ozaki, K. (2015). DINA models for multiple-choice items with few parameters: Considering incorrect answers. *Applied Psychological Measurement*, *39*, 431–447.

`retention.rate` *Retention rate of a Q-matrix refinement method*

Description

This function computes the proportion of correctly specified q-entries in a provisional Q-matrix that remain correctly specified after a Q-matrix refinement procedure is applied. This function is used only when the true Q-matrix is known.

Usage

```
retention.rate(ref.Q = ref.Q, mis.Q = mis.Q, true.Q = true.Q)
```

Arguments

`ref.Q`	The $J \times K$ refined binary Q-matrix.
`mis.Q`	A $J \times K$ binary provisional Q-matrix.
`true.Q`	The $J \times K$ binary true Q-matrix.

Value

The function returns a value between 0 and 1 indicating the proportion of correctly specified q-entries in `mis.Q` that remain correctly specified in `ref.Q` after a Q-matrix refinement procedure is applied to `mis.Q`.

See Also

See examples used for `QR`.

RR *Entry-wise and vector-wise agreement rate between two Q-matrices*

Description

RR is used to compute the agreement rate between two Q-matrices with identical dimensions.

Usage

```
RR(Q1, Q2)
```

Arguments

Q1	The first Q-matrix.
Q2	The second Q-matrix that has the same dimensionality as Q1.

Value

The function returns

entry.wise The entry-wise agreement rate

item.wise The item-wise agreement rate

See Also

See the examples for the QR and TSQE functions.

run_gnpc_app *Launch the GNPC Shiny app*

Description

Launches the interactive demo shipped with the package (can be found in inst/shiny/GNPC_app). The app demonstrates NPC, GNPC, and G-DINA workflows and the ECPE real-data example.

Usage

```
run_gnpc_app(
  launch.browser = interactive(),
  host = "127.0.0.1",
  port = NULL,
  ...
)
```

Arguments

launch.browser	Logical; open in a web browser? Defaults to interactive().
host	Host interface passed to shiny::runApp. Defaults to "127.0.0.1".
port	Optional integer port. If NULL, Shiny picks a free port.
...	Additional arguments forwarded to shiny::runApp().

Examples

```
## Not run:
  run_gnpc_app()

## End(Not run)
```

---

summary.GNPC *Summarize GNPC Classification Results*

Description

Summarize GNPC Classification Results

Usage

```
## S3 method for class 'GNPC'
summary(object, top.n = 5, digits = 1, max.iter.print = 10, ...)

## S3 method for class 'summary.GNPC'
print(x, ...)
```

Arguments

object	An object of class "GNPC".
top.n	Number of estimated attribute profiles to include.
digits	Number of decimal places used for percentages when printed.
max.iter.print	Maximum number of convergence-history rows to print.
...	Additional arguments (not used).
x	An object of class "summary.GNPC".

Value

An object of class "summary.GNPC" containing attribute-level results, profile-level results, and convergence results.

summary.NPC *Summarize NPC Classification Results*

Description

Summarize NPC Classification Results

Usage

```
## S3 method for class 'NPC'
summary(object, top.n = 5, digits = 1, ...)

## S3 method for class 'summary.NPC'
print(x, ...)
```

Arguments

object	An object of class "NPC".
top.n	Number of estimated attribute profiles to include.
digits	Number of decimal places used for percentages when printed.
...	Additional arguments (not used).
x	An object of class "summary.NPC".

Value

An object of class "summary.NPC" containing attribute-level results, profile-level results, and classification ambiguity results.

TSQE *Two-step Q-matrix estimation method*

Description

The function estimates the Q-matrix based on the response data using the two-step Q-matrix estimation method.

Usage

```
TSQE(
  Y,
  K,
  input.cor = c("tetrachoric", "pearson"),
  ref.method = c("QR", "GDI"),
  GDI.model = c("GDINA", "DINA", "ACDM", "RRUM"),
  cutoff = 0.8
)
```

Arguments

`Y`	A $N \times J$ binary data matrix consisting of responses from N examinees to J items
`K`	The number of attributes in the Q-matrix
`input.cor`	The type of correlation used as input for the provisional attribute extraction (PAE) algorithm. It could be the `tetrachoric` or `pearson` correlation.
`ref.method`	The refinement method used to polish the provisional Q-matrix obtained from the PAE. Currently available methods include the Q-matrix refinement (`QR`) method and the G-DINA discrimination index (`GDI`).
`GDI.model`	The CDM used in the GDI algorithm to fit the data. Currently available models include the DINA model, the ACDM, the RRUM, and the G-DINA model.
`cutoff`	The cutoff used to dichotomize the entries in the provisional Q-matrix. The default is 0.8.

Value

The function returns the estimated Q-matrix.

Details

The TSQE method estimates a Q-matrix by integrating the provisional attribute extraction (PAE) algorithm with a Q-matrix refinement-and-validation method, such as the Q-Matrix Refinement (QR) method and the G-DINA Model Discrimination Index (GDI). Specifically, the PAE algorithm relies on classic exploratory factor analysis (EFA) combined with a unique stopping rule for identifying a provisional Q-matrix, and the resulting provisional Q-Matrix is "polished" by a refinement method to derive the finalized estimation of Q-matrix.

The PAE Algorithm starts with computing the inter-item tetrachoric correlation matrix. The reason for using tetrachoric correlation is that the examinee responses are binary, so it is more appropriate than the Pearson product moment correlation coefficient. See Köhn et al. (2025) for details. The next step is to use factor analysis on the item-correlation matrix, and treat the extracted factors as proxies for the latent attributes. The third step concerns the identification of specific attributes required for each item. The detailed algorithm is described below:

(1) Initialize the item index as $j = 1$.

(2) Let l_{jk} denote the loading of item j on factor k, where $k = 1, 2, ..., K$.

(3) Arrange the loadings in descending order. Define a mapping function $f(k) = t$, where t is the order index. Hence, $l_{j(1)}$ will indicate the maximum loading, while $l_{j(K)}$ will indicate the minimum loading.

(4) Define

$$p_j(t) = \frac{\sum_{h=1}^{t} l_{j(h)}^2}{\sum_{k=1}^{K} l_{jk}^2}$$

as the proportion of the communality of item j accounted for by the first t factors.

(5) Define

$$K_j = \min\{t \mid p_j(t) \geq \lambda\}$$

, where λ is the cut-off value for the desired proportion of item variance-accounted-for. Then, the ordered entries of the provisional q-vector of item j are obtained as

$$q^*_{j(t)} = \begin{cases} 1 & \text{if } t \leq K_j \\ 0 & \text{if } t > K_j \end{cases}$$

.

(6) Identify $q^*_j = (q^*_{j1}, q^*_{j2}, ..., q^*_{jK})$ by rearranging the ordered entries of the q-vector using the inverse function $k = f^{-1}(t)$.

(7) Set $j = j + 1$ and repeat (2) to (6) until $j = J$. Then denote the provisional Q-matrix as $\mathbf{Q}^*$.

The provisional Q-matrix $\mathbf{Q}^*$ is then refined by using either the `QR` or `GDI` method.

References

Chiu, C. Y. (2013). Statistical Refinement of the Q-matrix in Cognitive Diagnosis. *Applied Psychological Measurement, 37(8)*, 598-618. doi:10.1177/0146621613488436

de la Torre, J., & Chiu, C.-Y. (2016). A general method of empirical Q-matrix validation. *Psychometrika, 81*, 253-273. doi:10.1007/s1133601594678

Köhn, H. F., Chiu, C.-Y., Oluwalana, O., Kim, H. & Wang, J. (2025). A two-step Q-matrix estimation method, *Applied Psychological Measurement, 49*(1-2), 3-28. doi:10.1177/01466216241284418

See Also

`QR`

Examples

```
## Not run:
library(GDINA)
N = 1000
Q = sim30GDINA$simQ
J = nrow(Q)
K= ncol(Q)
gs = data.frame(guess=rep(0.2,J),slip=rep(0.2,J))
sim = simGDINA(N,Q,gs.parm = gs,model = "DINA")
Y = extract(sim,what = "dat")

## Run TSQE method with QR
est.Q = TSQE(Y, K, input.cor = "tetrachoric", ref.method = "QR", cutoff = 0.8)

## If the recovery rate is to be computed, the columns of the estimated Q-matrix
## should be permuted so that they align with those of the true Q-matrix.
best.est.Q = bestQperm(est.Q, Q)

## Compute the recovery rate
RR(best.est.Q, Q)

## End(Not run)
```

Index

∗ **datasets**
    Q_Ozaki, 27

AAR, 2, *17*

bestQperm, 3

correction.rate, 4

distractor.check, 4, *19*

GNPC, 5, *16*, *18*

level.x, 8

mcdat.generate, 10
MCNPC, 11
mcQ.completeness, 12
mcQ.generate, 14

NPC, 14, *26*

PAR, *3*, 16
plot, *18*
plot.GNPC, 17
print.distractor.check, 19
print.GNPC, 19
print.NPC, 20
print.Qcompleteness, 20
print.Qrefine, 21
print.summary.GNPC *(*summary.GNPC*)*, 30
print.summary.NPC *(*summary.NPC*)*, 31

Q.completeness, 21, *24*
Q.generate, 23
Q.implausible, 24
Q.improper, 25
Q_Ozaki, 27
QR, 25, *28*, *29*, *33*

retention.rate, 28
RR, 29
run_gnpc_app, 29

summary.GNPC, 30
summary.NPC, 31

TSQE, *29*, 31