Package ‘simcdm’

May 9, 2026

Type Package

Title Simulate Cognitive Diagnostic Model ('CDM') Data

Version 0.1.2

Description Provides efficient R and 'C++' routines to simulate cognitive diagnostic model data for Deterministic Input, Noisy ``And'' Gate ('DINA') and reduced Reparameterized Unified Model ('rRUM') from Culpepper and Hudson (2017) <doi:10.1177/0146621617707511>, Culpepper (2015) <doi:10.3102/1076998615595403>, and de la Torre (2009) <doi:10.3102/1076998607309474>.

URL https://tmsalab.github.io/simcdm/,
https://github.com/tmsalab/simcdm

BugReports https://github.com/tmsalab/simcdm/issues

License GPL (>= 2)

Depends R (>= 4.1.0)

Imports Rcpp (>= 1.0.11)

LinkingTo Rcpp, RcppArmadillo (>= 0.12.6.6.0)

Suggests testthat, covr, knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.2.3

Encoding UTF-8

NeedsCompilation yes

Author James Joseph Balamuta [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-2826-8458>), Steven Andrew Culpepper [aut, cph] (ORCID: <https://orcid.org/0000-0003-4226-6176>), Aaron Hudson [ctb, cph] (ORCID: <https://orcid.org/0000-0002-9731-2224>)

Maintainer James Joseph Balamuta <balamut2@illinois.edu>

Repository CRAN

Date/Publication 2023-11-29 07:50:02 UTC

Contents

`simcdm-package` *simcdm: Simulate Cognitive Diagnostic Model ('CDM') Data*

Description

Provides efficient R and 'C++' routines to simulate cognitive diagnostic model data for Deterministic Input, Noisy "And" Gate ('DINA') and reduced Reparameterized Unified Model ('rRUM') from Culpepper and Hudson (2017) doi: 10.1177/0146621617707511, Culpepper (2015) doi:10.3102/1076998615595403, and de la Torre (2009) doi:10.3102/1076998607309474.

Author(s)

Maintainer: James Joseph Balamuta <balamut2@illinois.edu> (ORCID) [copyright holder]

Authors:

- Steven Andrew Culpepper <sculpepp@illinois.edu> (ORCID) [copyright holder]

Other contributors:

- Aaron Hudson <awhudson@uw.edu> (ORCID) [contributor, copyright holder]

See Also

Useful links:

- https://tmsalab.github.io/simcdm/
- https://github.com/tmsalab/simcdm
- Report bugs at https://github.com/tmsalab/simcdm/issues

`attribute_bijection` *Constructs Unique Attribute Pattern Map*

Description

Computes the powers of 2 from 0 up to $K - 1$ for K-dimensional attribute pattern.

Usage

```
attribute_bijection(K)
```

Arguments

`K`	Number of Attributes.

Value

A vec with length K detailing the power's of 2.

Author(s)

Steven Andrew Culpepper and James Joseph Balamuta

See Also

`attribute_inv_bijection()`

Examples

```
## Construct an attribute bijection ----
biject = attribute_bijection(3)
```

`attribute_classes` *Simulate all the Latent Attribute Profile α_c in Matrix form*

Description

Generate the $\alpha_c = (\alpha_{c1}, \ldots, \alpha_{cK})'$ attribute profile matrix for members of class c such that α_{ck} ' is 1 if members of class c possess skill k and zero otherwise.

Usage

```
attribute_classes(K)
```

Arguments

`K`	Number of Attributes

Value

A 2^K by K `matrix` of latent classes corresponding to entry c of pi based upon mastery and non-mastery of the K skills.

Author(s)

James Joseph Balamuta and Steven Andrew Culpepper

See Also

`sim_subject_attributes()` and `attribute_inv_bijection()`

Examples

```
## Simulate Attribute Class Matrix ----

# Define number of attributes
K = 3

# Generate an Latent Attribute Profile (Alpha) Matrix
alphas = attribute_classes(K)
```

`attribute_inv_bijection` *Perform an Inverse Bijection of an Integer to Attribute Pattern*

Description

Convert an integer between 0 and 2^{K-1} to K-dimensional attribute pattern.

Usage

```
attribute_inv_bijection(K, CL)
```

Arguments

`K`	Number of Attributes.
`CL`	An `integer` between 0 and 2^{K-1}

Value

A K-dimensional vector with an attribute pattern corresponding to `CL`.

Author(s)

Steven Andrew Culpepper and James Joseph Balamuta

See Also

`attribute_bijection()`

Examples

```
## Construct an attribute inversion bijection ----
inv_biject1 = attribute_inv_bijection(5, 1)
inv_biject2 = attribute_inv_bijection(5, 2)
```

`sim_dina_attributes` *Simulate a DINA Model's η Matrix*

Description

Generates a DINA model's η matrix based on alphas and the $\mathbf{Q}$ matrix.

Usage

```
sim_dina_attributes(alphas, Q)
```

Arguments

alphas	A N by K `matrix` of latent attributes.
Q	A J by K `matrix` indicating which skills are required for which items.

Value

The η `matrix` with dimensions $N \times J$ under the DINA model.

Author(s)

Steven Andrew Culpepper and James Joseph Balamuta

See Also

`sim_dina_class()` and `sim_dina_items()`

Examples

```
N = 200
K = 5
J = 30
delta0 = rep(1, 2 ^ K)

# Creating Q matrix
Q = matrix(rep(diag(K), 2), 2 * K, K, byrow = TRUE)
for (mm in 2:K) {
  temp = combn(seq_len(K), m = mm)
```

```
    tempmat = matrix(0, ncol(temp), K)
    for (j in seq_len(ncol(temp)))
      tempmat[j, temp[, j]] = 1
    Q = rbind(Q, tempmat)
  }
  Q = Q[seq_len(J), ]

  # Setting item parameters and generating attribute profiles
  ss = gs = rep(.2, J)
  PIs = rep(1 / (2 ^ K), 2 ^ K)
  CLs = c((1:(2 ^ K)) %*% rmultinom(n = N, size = 1, prob = PIs))

  # Defining matrix of possible attribute profiles
  As = rep(0, K)
  for (j in seq_len(K)) {
    temp = combn(1:K, m = j)
    tempmat = matrix(0, ncol(temp), K)
    for (j in seq_len(ncol(temp)))
      tempmat[j, temp[, j]] = 1
    As = rbind(As, tempmat)
  }
  As = as.matrix(As)

  # Sample true attribute profiles
  Alphas = As[CLs, ]

  # Simulate item data under DINA model
  dina_items = sim_dina_items(Alphas, Q, ss, gs)

  # Simulate attribute data under DINA model
  dina_attributes = sim_dina_attributes(Alphas, Q)
```

sim_dina_class — *Simulate Binary Responses for a DINA Model*

Description

Generate the dichotomous item matrix for a DINA Model.

Usage

```
sim_dina_class(N, J, CLASS, ETA, gs, ss)
```

Arguments

N	Number of Observations
J	Number of Assessment Items
CLASS	Does the individual possess all the necessary attributes?
ETA	η Matrix containing indicators.

gs	A vec describing the probability of guessing or the probability subject correctly answers item j when at least one attribute is lacking.
ss	A vec describing the probability of slipping or the probability of an incorrect response for individuals with all of the required attributes

Value

A dichotomous item matrix with dimensions $N \times J$.

Author(s)

Steven Andrew Culpepper and James Joseph Balamuta

See Also

sim_dina_attributes() and sim_dina_items()

Examples

```
# Set
N       = 100
rho     = 0
K       = 3

# Fixed Number of Assessment Items for Q
J = 18

# Specify Q
qbj = c(4, 2, 1, 4, 2, 1, 4, 2, 1, 6, 5, 3, 6, 5, 3, 7, 7, 7)

# Fill Q Matrix
Q = matrix(, J, K)
for (j in seq_len(J)) {
  Q[j,] = attribute_inv_bijection(K, qbj[j])
}

# Item parm vals
ss = gs = rep(.2, J)

# Generating attribute classes depending on correlation
if (rho == 0) {
  PIs = rep(1 / (2 ^ K), 2 ^ K)
  CLs = c(seq_len(2 ^ K) %*% rmultinom(n = N, size = 1, prob = PIs)) - 1
}

if (rho > 0) {
  Z = matrix(rnorm(N * K), N, K)
  Sig = matrix(rho, K, K)
  diag(Sig) = 1
  X = Z %*% chol(Sig)
  thvals = matrix(rep(0, K), N, K, byrow = T)
  Alphas = 1 * (X > thvals)
```

```
  CLs = Alphas %*% attribute_bijection(K)
}

# Simulate data under DINA model
ETA = sim_eta_matrix(K, J, Q)
Y_sim = sim_dina_class(N, J, CLs, ETA, gs, ss)
```

---

sim_dina_items *Simulation Responses from the DINA model*

Description

Sample responses from the DINA model for given attribute profiles, Q matrix, and item parmeters. Returns a `matrix` of dichotomous responses generated under DINA model.

Usage

```
sim_dina_items(alphas, Q, ss, gs)
```

Arguments

alphas	A N by K `matrix` of latent attributes.
Q	A J by K `matrix` indicating which skills are required for which items.
ss	A J `vector` of item slipping parameters.
gs	A J `vector` of item guessing parameters.

Value

A N by J `matrix` of responses from the DINA model.

Author(s)

Steven Andrew Culpepper and James Joseph Balamuta

See Also

`sim_dina_class()` and `sim_dina_attributes()`

Examples

```
N = 200
K = 5
J = 30
delta0 = rep(1, 2 ^ K)

# Creating Q matrix
Q = matrix(rep(diag(K), 2), 2 * K, K, byrow = TRUE)
for (mm in 2:K) {
```

```
    temp = combn(seq_len(K), m = mm)
    tempmat = matrix(0, ncol(temp), K)
    for (j in seq_len(ncol(temp)))
      tempmat[j, temp[, j]] = 1
    Q = rbind(Q, tempmat)
  }
  Q = Q[seq_len(J), ]

  # Setting item parameters and generating attribute profiles
  ss = gs = rep(.2, J)
  PIs = rep(1 / (2 ^ K), 2 ^ K)
  CLs = c((1:(2 ^ K)) %*% rmultinom(n = N, size = 1, prob = PIs))

  # Defining matrix of possible attribute profiles
  As = rep(0, K)
  for (j in seq_len(K)) {
    temp = combn(1:K, m = j)
    tempmat = matrix(0, ncol(temp), K)
    for (j in seq_len(ncol(temp)))
      tempmat[j, temp[, j]] = 1
    As = rbind(As, tempmat)
  }
  As = as.matrix(As)

  # Sample true attribute profiles
  Alphas = As[CLs, ]

  # Simulate item data under DINA model
  dina_items = sim_dina_items(Alphas, Q, ss, gs)

  # Simulate attribute data under DINA model
  dina_attributes = sim_dina_attributes(Alphas, Q)
```

---

sim_eta_matrix *Generate ideal response η Matrix*

Description

Creates the ideal response matrix for each trait

Usage

```
sim_eta_matrix(K, J, Q)
```

Arguments

K	Number of Attribute Levels
J	Number of Assessment Items
Q	Q Matrix with dimensions $K \times J$.

Value

A `mat` with dimensions $J \times 2^K$.

Author(s)

Steven Andrew Culpepper and James Joseph Balamuta

See Also

`sim_q_matrix()`, `attribute_bijection()`, and `attribute_inv_bijection()`

Examples

```
## Simulation Settings ----

# Fixed Number of Assessment Items for Q
J = 18

# Fixed Number of Attributes for Q
K = 3

## Pre-specified configuration ----

# Specify Q
qbj = c(4, 2, 1, 4, 2, 1, 4, 2, 1, 6, 5, 3, 6, 5, 3, 7, 7, 7)

# Fill Q Matrix
Q = matrix(, J, K)
for (j in seq_len(J)) {
  Q[j,] = attribute_inv_bijection(K, qbj[j])
}

# Create an eta matrix
ETA = sim_eta_matrix(K, J, Q)

## Random generation of Q matrix with ETA matrix ----

# Construct a random q matrix
Q_sim = sim_q_matrix(J, K)

# Generate the eta matrix
ETA_gen = sim_eta_matrix(K, J, Q_sim)
```

---

sim_q_matrix — *Generate a Random Identifiable Q Matrix*

Description

Simulates a Q matrix containing three identity matrices after a row permutation that is identifiable.

Usage

```
sim_q_matrix(J, K)
```

Arguments

`J`	Number of Items
`K`	Number of Attributes

Value

A dichotomous `matrix` for Q.

Author(s)

Steven Andrew Culpepper and James Joseph Balamuta

See Also

`attribute_bijection()` and `attribute_inv_bijection()`

Examples

```
## Simulate identifiable Q matrices ----

# 7 items and 2 attributes
q_matrix_j7_k2 = sim_q_matrix(7, 2)

# 10 items and 3 attributes
q_matrix_j10_k3 = sim_q_matrix(10, 3)
```

`sim_rrum_items` *Generate data from the rRUM*

Description

Randomly generate response data according to the reduced Reparameterized Unified Model (rRUM).

Usage

```
sim_rrum_items(Q, rstar, pistar, alpha)
```

Arguments

- `Q` — A `matrix` with J rows and K columns indicating which attributes are required to answer each of the items, where J represents the number of items and K the number of attributes. An entry of 1 indicates attribute k is required to answer item j. An entry of one indicates attribute k is not required.
- `rstar` — A `matrix` a matrix with J rows and K columns indicating the penalties for failing to have each of the required attributes, where J represents the number of items and K the number of attributes. `rstar` and `Q` must share the same 0 entries.
- `pistar` — A `vector` of length J indicating the probabiliies of answering each item correctly for individuals who do not lack any required attribute, where J represents the number of items.
- `alpha` — A `matrix` with N rows and K columns indicating the subjects attribute acquisition, where K represents the number of attributes. An entry of 1 indicates individual i has attained attribute k. An entry of 0 indicates the attribute has not been attained.

Value

Y A `matrix` with N rows and J columns indicating the indviduals' responses to each of the items, where J represents the number of items.

Author(s)

Steven Andrew Culpepper, Aaron Hudson, and James Joseph Balamuta

References

Culpepper, S. A. & Hudson, A. (In Press). An improved strategy for Bayesian estimation of the reduced reparameterized unified model. Applied Psychological Measurement.

Hudson, A., Culpepper, S. A., & Douglas, J. (2016, July). Bayesian estimation of the generalized NIDA model with Gibbs sampling. Paper presented at the annual International Meeting of the Psychometric Society, Asheville, North Carolina.

Examples

```
# Set seed for reproducibility
set.seed(217)

# Define Simulation Parameters
N = 1000 # number of individuals
J = 6 # number of items
K = 2 # number of attributes

# Matrix where rows represent attribute classes
As = attribute_classes(K)

# Latent Class probabilities
pis = c(.1, .2, .3, .4)
```

```
# Q Matrix
Q = rbind(c(1, 0),
          c(0, 1),
          c(1, 0),
          c(0, 1),
          c(1, 1),
          c(1, 1)
    )

# The probabiliies of answering each item correctly for individuals
# who do not lack any required attribute
pistar = rep(.9, J)

# Penalties for failing to have each of the required attributes
rstar  = .5 * Q

# Randomized alpha profiles
alpha  = As[sample(1:(K ^ 2), N, replace = TRUE, pis),]

# Simulate data
rrum_items = sim_rrum_items(Q, rstar, pistar, alpha)
```

---

sim_subject_attributes

Simulate Subject Latent Attribute Profiles α_c

Description

Generate a sample from the $\alpha_c = (\alpha_{c1}, \ldots, \alpha_{cK})'$ attribute profile matrix for members of class c such that α_{ck} ' is 1 if members of class c possess skill k and zero otherwise.

Usage

```
sim_subject_attributes(N, K, probs = NULL)
```

Arguments

N	Number of Observations
K	Number of Skills
probs	A vector of probabilities that sum to 1.

Value

A N by K matrix of latent classes corresponding to entry c of pi based upon mastery and nonmastery of the K skills.

Author(s)

James Joseph Balamuta and Steven Andrew Culpepper

See Also

`attribute_classes()` and `attribute_inv_bijection()`

Examples

```
# Define number of subjects and attributes
N = 100
K = 3

# Generate a sample from the Latent Attribute Profile (Alpha) Matrix
# By default, we sample from a uniform distribution weighting of classes.
alphas_builtin = sim_subject_attributes(N, K)

# Generate a sample using custom probabilities from the
# Latent Attribute Profile (Alpha) Matrix
probs = rep(1 / (2 ^ K), 2 ^ K)
alphas_custom = sim_subject_attributes(N, K, probs)
```

Index