Package 'dcmr'

February 19, 2015

Type Package

Title Attribute profile estimation using Diagnostic Classification Models and MCMC

Version 1.0

Date 2014-07-08

Author c(person(``Diane'', family=``Losardo'', email='dlosardo@amplify.com``, role = c(''aut``, ''cre``, ''cph``)), person(''Margi``, family=''Dubal``, email='margidubal@gmail.com'', role = c(``aut'', ``cre'', ``cph'')))

Maintainer Diane Losardo <dlosardo@amplify.com>

Description Analysis of dichotomous response data to obtain attribute profile estimates for respondents using Diagnostic Classification Model (DCM) and Markov Chain Monte Carlo (MCMC) method. The estimation procedure uses a loglinear cognitive diagnostic modeling (LDCM) framework that allows for the estimation of a host of DCMs such as NIDO, NIDA, NC-RUM etc.

License GPL (>= 3)

Depends R (>= 3.0.2)

Imports ggplot2, KFAS, plyr, reshape2, tableplot, methods

Suggests testthat

LazyLoad yes

LazyData yes

Repository CRAN

NeedsCompilation no

Date/Publication 2014-07-26 08:27:47

R topics documented:

`all.results.class-class` *all.results.class*

Description

all.results.class

Slots

`attribute.profile.result` An object of `attribute.profile.class`

`attribute.result` An object of `attribute.class`

`parameter.result` An object of `parameter.class`

`attribute.class-class` *attribute.class*

Description

attribute.class

Slots

`results` A data frame of probability of mastering each attribute

attribute.profile.class-class
attribute.profile.class

Description

attribute.profile.class

Slots

`results` A data frame of probability of mastering each attribute profile

`attribute.profile.matrix` A matrix of all possible attribute profiles

`CalculatePiIStarPrime` *Calculate value for pi.i.star.prime for one item*

Description

Given an NCRUM parameterization and values for pi.i.star, r.ia.star, and qmatrix, calculates the value for for pi.i.star.prime for the given item

Usage

```
CalculatePiIStarPrime(pi.i.star, r.ia.star, q.ia)
```

Arguments

`pi.i.star`	a numeric value for the probability that a person responds correctly given they have not mastered any attributes before being penalized for not mastering an attribute.
`r.ia.star`	a numeric 1 x nattributes vector of the penalty probabilities for this item of a correct response for each required attribute that has not been mastered.
`q.ia`	a numeric 1 x nattributes vector of the qmatrix entries for the given item.

Value

a numeric value representing the pi.i.star_prime for item i

class.probabilities.interaction.test

Class probabilities for Q-matrix containing interaction between attributes

Description

This data set list the class probabities for Q-matrix containing interaction between attributes. This data set is used to check output of `ScoreDCM`

Usage

```
class.probabilities.interaction.test
```

Format

a data frame containing 15 students and 3 attributes. 1 student per row

Author(s)

Margi Dubal <margidubal@gmail.com> & Diane Losardo <dlosardo@amplify.com>

class.probabilities.test

Class probabilities for Q-matrix containing no interaction between attributes

Description

This data set list the class probabities for Q-matrix containing no interaction between attributes. This data set is used to check output of `ScoreDCM`

Usage

```
class.probabilities.test
```

Format

a data frame containing 15 students and 3 attributes. 1 student per row

Author(s)

Margi Dubal <margidubal@gmail.com> & Diane Losardo <dlosardo@amplify.com>

dcm.scorer.class-class *dcm.scorer.class*

Description

dcm.scorer.class

Slots

`inputs` A list of inputs provided by user

`mcmc.inputs` A list of inputs for MCMC estimation

`results` An object of `all.results.class`

`mcmc.output` A list of MCMC runs

DrawAlphas *Draw values from multinomial distribution*

Description

Draws values from a mutinomial distribution and matches them with a row in the pmatrix (attribute profile matrix) to obtain the alpha values

Usage

```
DrawAlphas(pmatrix, y, nus0)
```

Arguments

`pmatrix`	a matrix indicating all possible attribute profiles
`y`	number of random vectors to draw
`nus0`	a numeric vector of length `nclasses` for nu parameter estimates

Value

a numeric vector of alpha values

DrawClasses *Draw latent variable values*

Description

Produces values of alphas(s) (i.e., latent variable(s)) by using a rejection/acceptance Metropolis-Hastings procedure.

Usage

```
DrawClasses(nattributes, class0, nus0, observations, nobservations,
  threshold.values, pmatrix)
```

Arguments

nattributes	a numeric value indicating number of alphas to be drawn
class0	a numeric vector of previous classes
nus0	a numeric vector of length nclasses for nu parameter estimates
observations	a data frame or matrix of dichotomous responses
nobservations	a numeric value indicating number of rows of the observation data frame or matrix
threshold.values	an nclasses by nitems numeric matrix with appropriate item threshold values
pmatrix	a numeric nclasses by nattributes matrix of all possible attribute profiles

Value

a numeric vector of classes indicating one of the possible attribute profile for each student

GetAllProbsCorrectNcrum *Calculate all item probabilities of correct response for NCRUM parameterization*

Description

Calculates the probabilities of a correct response for all items given an NCRUM parameterization

Usage

```
GetAllProbsCorrectNcrum(pi.i.star.prime, r.ia.star.qmatrix, qmatrix, alphas)
```

Arguments

pi.i.star.prime	a numeric 1 x nitems vector of the probabilities of a correct response for each item given that no attributes have been mastered.
r.ia.star.qmatrix	a numeric nitems x nattributes vector of the penalty probabilities for each item of a correct response for each required attribute that has not been mastered.
qmatrix	a numeric nitems by nattributes matrix that specifies which items are required for mastery of each attribute (i.e., latent variable)
alphas	a numeric 1 x n_attributes vector with a 1 if a student has mastered an attribute and 0 otherwise.

Value

probabilities of a correct response for all items given a person's attribute profile.

GetAttributeProfiles *Attribute Profiles*

Description

Given a number of attributes, it generates all possible attribute profiles (response patterns of latent variables)

Usage

```
GetAttributeProfiles(nattributes, attribute.names = NULL, profile.names = NULL)
```

Arguments

nattributes	a numeric value for number of attributes
attribute.names	optional character vector of attribute names
profile.names	optional character vector of attribute profile names

Value

a matrix of dimension `2^nattributes X nattributes` containing binary values

Author(s)

Margi Dubal <margidubal@gmail.com> & Diane Losardo <dlosardo@amplify.com>

Examples

```
## Not run:

 GetAttributeProfiles(nattributes=3)

## End(Not run)
```

`GetAttributesProbabilitiesMCMC`
Calculate attribute probabilities

Description

Calculates the attribute (i.e., latent variable) probabilities for all students

Usage

```
GetAttributesProbabilitiesMCMC(class.result, pmatrix)
```

Arguments

`class.result`	a list of classes containing all interactions and chains of MCMC
`pmatrix`	a numeric nclasses by nattributes matrix of all possible attribute profiles

---

`GetClassProbabilitiesMCMC`
Calculate attribute profile probabilities

Description

Calculates the attribute profile probabilities for all students

Usage

```
GetClassProbabilitiesMCMC(class.result, nclasses)
```

Arguments

`class.result`	a list of classes containing all interactions and chains of MCMC
`nclasses`	a numeric value representing the number of unique attribute profiles

GetClassProbsFromMus *Obtain attribute profile probabilities from latent variable means*

Description

Gets all attribute profile probabilities given mu parameters (latent variable means)

Usage

```
GetClassProbsFromMus(mus)
```

Arguments

mus	a 1 by nclasses numeric vector of latent variable means

Value

a numeric vector of all attribute profile probability values

GetGammaName *Obtain name of a particular gamma parameter*

Description

Produces a string of the form: g_effect_attribtues representing gamma parameters

Usage

```
GetGammaName(attributes)
```

Arguments

attributes	a string of the attributes associated with this gamma parameter

Value

a string of the form g_effect_attribtues.

GetGammaNames *Obtain names of all gamma parameters*

Description

Gets names of all gamma parameters

Usage

```
GetGammaNames(required.attributes)
```

Arguments

required.attributes	a numeric vector of attributes required to have mastered to get item correct

Value

a String vector of names for gamma parameters

GetKernelParameterNames *Kernel Parameter Names for all DCM Models Given a Q-matrix and model type it generates item and structural parameter names. These are kernel parameters: item thresholds (*lambdas*) and latent variable thresholds (*gammas*)*

Description

Kernel Parameter Names for all DCM Models Given a Q-matrix and model type it generates item and structural parameter names. These are kernel parameters: item thresholds (lambdas) and latent variable thresholds (gammas)

Usage

```
GetKernelParameterNames(qmatrix, nattributes, model.type = "DCM")
```

Arguments

qmatrix	a matrix of size (nitems X nattributes) that specifies which items are required for mastery of each attribute (i.e., latent variable)
nattributes	a numeric value of number of attriutes
model.type	a string value of DCM, DINA, CRUM, DINO, NIDO, NCRUM. If not specified then the default will be set to DCM

Value

a string vector of threshold parameter names

Author(s)

Margi Dubal <margidubal@gmail.com> & Diane Losardo <dlosardo@amplify.com>

Examples

```
## Not run:

GetKernelParameterNames(qmatrix = qmatrix.test, nattributes=3)

## End(Not run)
```

GetLambdaName *Obtain name of a particular lambda parameter*

Description

Gets the name of a particular lambda parameter and produces a string of the form: l_item_effect_attribtues.

Usage

```
GetLambdaName(item, attributes)
```

Arguments

item	a numeric value representing item number
attributes	a String of the attributes associated with this gamma

Value

a String of the form: l_item_effect_attribtues.

GetLambdaNames *Gets names of lambda parameters*

Description

Gets names of lambda parameters

Usage

```
GetLambdaNames(item.indexes.intercepts, item.indexes.main.effects,
  required.attributes.lambda)
```

Arguments

`item.indexes.intercepts`
: a numeric vector indicating which item is assoicated with the lamdba intercept at the given index and thus allowing for equality constraints. For example, with nitems = 3, this is c(1, 1, 1), that implies that intercept lambdas across items should be contrained to equality.

`item.indexes.main.effects`
: a numeric vector indicating which item is assoicated with the lamdba main effect at the given index and thus allowing for equality constraints. For example, with nitems = 3, this is c(1, 1, 1), that implies that main effect (and higher order) lambdas across items should be contrained to equality.

`required.attributes.lambda`
: a nclasses by nitems matrix with each cell containing a String representing the attributes required for the given class and item specific to lambda parameters.

Value

a String vector of names of lambda parameters

GetLambdaNamesForItem *Gets lambda names for a given item*

Description

Gets lambda names for a given item

Usage

```
GetLambdaNamesForItem(item.index.intercept, item.index.main.effect,
  required.attributes)
```

Arguments

`item.index.intercept`
: a numeric value representing item number for the intercept lambda

`item.index.main.effect`
: a numeric value representing item number for the main effects lambda

`required.attributes`
: a numeric vector of attributes required to have mastered to get item correct

Value

a String vector of lambda names for one item

GetMusFromGammas *Calcualte attribute means from gamma parameters*

Description

Give gamma parameters and all attribute profiles, calculates the attribute means (i.e., mus)

Usage

```
GetMusFromGammas(estimates0, pmatrix)
```

Arguments

`estimates0`	a numeric vector of parameter estimates in the form of lambdas and gammas
`pmatrix`	a numeric nclasses by nattributes matrix of all possible attribute profiles

---

GetParameterNames *Parameter Names for All DCM Models*

Description

Given a Q-matrix and parameterization method it generates item and structural parameter names. These are non-kernel parameters: item thresholds (`taus`) and latent variable thresholds (`nus`)

Usage

```
GetParameterNames(qmatrix, nattributes, parameterization.method = 'Mplus')
```

Arguments

qmatrix	a matrix of size (nitems X nattributes) that specifies which items are required for mastery of each attribute (i.e., latent variable)
nattributes	a numeric value of number of attriutes
parameterization.method	optional character string of parameterization method used to calibrated parameters. If not specified then the default will be set to Mplus

Value

a string vector of threshold parameter names

Author(s)

Margi Dubal <margidubal@gmail.com> & Diane Losardo <dlosardo@amplify.com>

Examples

```
## Not run:

 GetParameterNames(qmatrix = qmatrix.test, nattributes=3)

## End(Not run)
```

GetParameterResultsMCMC

Calculate results for attributes and attribute profiles

Description

Calculates results for attributes and attribute profiles for all students based on MCMC output

Usage

```
GetParameterResultsMCMC(parameter.result)
```

Arguments

parameter.result	a list parameter estimates for each iterations and chains

GetProbCorrectNcrum *Calculate item probability of correct response for NCRUM parameterization*

Description

Calculates the probability of a correct response for an item given an NCRUM parameterization

Usage

```
GetProbCorrectNcrum(pi.i.star.prime, r.ia.star, q.ia, alphas)
```

Arguments

pi.i.star.prime	a numeric value for the probability of a correct response for item i given that no attributes have been mastered.
r.ia.star	a numeric 1 x nattributes vector of the penalty probabilities for this item of a correct response for each required attribute that has not been mastered.
q.ia	a numeric 1 x nattributes vector of the qmatrix entries for the given item.
alphas	a numeric 1 x nattributes vector with a 1 if a student has mastered an attribute and 0 otherwise.

Value

the numeric value representing the probability of a correct response for the ith item.

GetRequiredAttributes *Generate required attributes*

Description

Generates a matrix of required attributes

Usage

```
GetRequiredAttributes(qmatrix, pmatrix)
```

Arguments

qmatrix	a nitems by nattributes matrix that specifies which items are required for mastery of each attribute (i.e., latent variable)
pmatrix	a numeric nclasses by nattributes matrix of all possible attribute profiles

Value

a nclasses by nitems character matrix with Strings representing the required attributes for a given item and class.

`GetRequiredAttributesLambdaCrum`
Calculate required attributes for lambdas for a CRUM model.

Description

Calculates required attributes for a model parameterized in the LCDM framework using the CRUM model

Usage

```
GetRequiredAttributesLambdaCrum(required.attributes.tau)
```

Arguments

`required.attributes.tau`
a nclasses by nitems matrix with each cell containing a String representing the attributes required for the given class and item specific to tau parameters.

Value

a nclasses by nitems matrix with each cell containing a String representing the attributes required for the given class and item specific to lambda parameters for the CRUM model.

`GetRequiredAttributesLambdaDina`
Calculate required attributes for lambdas for a DINA model.

Description

Calculates required attributes for a model parameterized in the LCDM framework using the DINA model

Usage

```
GetRequiredAttributesLambdaDina(required.attributes.tau)
```

Arguments

`required.attributes.tau`
a nclasses by nitems matrix with each cell containing a String representing the attributes required for the given class and item specific to tau parameters.

Value

a nclasses by nitems matrix with each cell containing a String representing the attributes required for the given class and item specific to lambda parameters for the DINA model.

`GetRequiredAttributesLambdaDino` *Calculate required attributes for lambdas for a DINO model.*

Description

Calculates required attributes for a model parameterized in the LCDM framework using the DINO model

Usage

```
GetRequiredAttributesLambdaDino(required.attributes.tau)
```

Arguments

`required.attributes.tau`
: a nclasses by nitems matrix with each cell containing a String representing the attributes required for the given class and item specific to tau parameters.

Value

a nclasses by nitems matrix with each cell containing a String representing the attributes required for the given class and item specific to lambda parameters for the DINO model.

`GetRequiredAttributesLambdaFullDCM` *Calculate required attributes for lamdbas fully specified DCM*

Description

Calculates required attributes for a model parameterized in the LCDM framework using the fully specified DCM model.

Usage

```
GetRequiredAttributesLambdaFullDCM(required.attributes.tau)
```

Arguments

`required.attributes.tau`
: a nclasses by nitems matrix with each cell containing a String representing the attributes required for the given class and item specific to tau parameters.

Value

a nclasses by nitems matrix with each cell containing a String representing the attributes required for the given class and item specific to lambda parameters for the a fully specific DCM model.

`GetRequiredAttributesLambdaNido` *Calculate required attributes for lambdas for a NIDO model.*

Description

Calculates required attributes for a model parameterized in the LCDM framework using the NIDO model

Usage

```
GetRequiredAttributesLambdaNido(required.attributes.tau)
```

Arguments

`required.attributes.tau`
: a nclasses by nitems matrix with each cell containing a String representing the attributes required for the given class and item specific to tau parameters.

Value

a nclasses by nitems matrix with each cell containing a String representing the attributes required for the given class and item specific to lambda parameters for the NIDO model.

`GetThresholdLabels` *Obtain item threshold labels*

Description

Gets item threshold labels for a given qmatrix and pmatrix

Usage

```
GetThresholdLabels(qmatrix, pmatrix)
```

Arguments

`qmatrix`	a numeric nitems by nattributes matrix that specifies which items are required for mastery of each attribute (i.e., latent variable)
`pmatrix`	a numeric nclasses by nattributes matrix of all possible attribute profiles

Value

an nclasses by nitems character matrix with appropriate threshold labels.

`GetThresholdValues` *GetThresholdValues*

Description

GetThresholdValues

Usage

```
GetThresholdValues(threshold.labels, taus)
```

Arguments

`threshold.labels`	an nclasses by nitems character matrix with appropriate threshold labels.
`taus`	numeric vector of taus correctly named threshold.labels.cls: String vector of names of thresholds for given class

`GetThresholdValuesKernel` *Calculate item threshold values for kernel parameterization*

Description

Calculates the item threshold values for a kernel parameterization of a DCM model using lambdas

Usage

```
GetThresholdValuesKernel(lambda.equations, estimates0, threshold.labels)
```

Arguments

`lambda.equations`	an nclasses by nitems character matrix with tau values represented as a function of lambda values
`estimates0`	a numeric vector of parameter estimates in the form of lambdas and gammas
`threshold.labels`	an nclasses by nitems character matrix with appropriate threshold labels.

Value

a list of a numeric vector of item threshold values and an nclasses by nitems numeric matrix of item threshold values

GetThresholdValuesKernelPiR *Calculate item threshold values for NCRUM parameterization*

Description

Calculates all item threshold values for an NCRUM parameterization given parameter estimates, qmatrix, attribute profiles, and threshold label matrix.

Usage

```
GetThresholdValuesKernelPiR(estimates0, qmatrix, pmatrix, threshold.labels)
```

Arguments

estimates0	a numeric vector of parameter estimates in the form of pis and rs
qmatrix	a numeric nitems by nattributes matrix that specifies which items are required for mastery of each attribute (i.e., latent variable)
pmatrix	a numeric nclasses by nattributes matrix of all possible attribute profiles
threshold.labels	an nclasses by nitems character matrix with appropriate item threshold labels

Value

a numeric vector of item threshold values

head,attribute.class-method *headattribute.class*

Description

headattribute.class

Usage

```
## S4 method for signature 'attribute.class'
head(x)
```

Arguments

x	attribute.class input object

head,attribute.profile.class-method *head attribute.profile.class*

Description

head attribute.profile.class

Usage

```
## S4 method for signature 'attribute.profile.class'
head(x)
```

Arguments

x	attribute.profile.class input object

InitializeParameters *Initialize parameter estimates*

Description

Obtains appropriate parameter estimates for a non-kernel model (e.g., taus and nus)

Usage

```
InitializeParameters(estimates, nclasses)
```

Arguments

estimates	a vector of parameter estimates
nclasses	a numeric value representing the number of unique attribute profiles

`iterate` *Perform one iteration of MCMC procedure*

Description

If applicable, randomly samples new set of parameter estimates, obtains applicable estimates and uses those to calculate threshold values for both items and latent variables, draws new set of alpha values.

Usage

```
iterate(nattributes, class0, estimates0, threshold.labels, lambda.equations,
  is.pi.r, parameter.means, parameter.acov, observations, nobservations,
  is.parameter.randomized, qmatrix, pmatrix)
```

Arguments

`nattributes`	numberic value for number of attributes
`class0`	The previous value of attribute profile for each respondent
`estimates0`	a numeric vector of parameter estimates
`threshold.labels`	an nclasses by nitems character matrix with appropriate threshold labels
`lambda.equations`	equations for lambda parameters
`is.pi.r`	If `FALSE` (the default), parameter values are the type of `taus` and `nus` or `lambdas` and `gammas` else they are the type `pis` and `rs` as used in NC-RUM parameterization
`parameter.means`	a numerical vector of calibrated item and structural parameters
`parameter.acov`	a numerical matrix of covariances of item and structural parameters
`observations`	a data frame or matrix of dichotomous responses
`nobservations`	a numeric value of number of observations
`is.parameter.randomized`	if true parameter estimates are randomized using acov matrix
`qmatrix`	a data frame or matrix of 1s and 0s indicating relation between items and attributes. This matrix specifies which items are required for mastery of each attribute (i.e., latent variable). A matrix must be a size of `nItems X nAttributes`
`pmatrix`	a numeric nclasses by nattributes matrix of all possible attribute profiles

Value

a list of newly sampled classes and parameter estimates

LongFormatResults *Tranform dataframe to long format*

Description

Given a dataframe in wide format, transform to long format

Usage

```
LongFormatResults(d, result.type = "class")
```

Arguments

d	a list of MCMC output either classes or parameter estimates
result.type	optional character strings containing result.type: class or parameter

mcmc *Performs MCMC routine for DCM*

Description

Performs MCMC routine for DCM

Usage

```
mcmc(observations, nattributes, qmatrix, pmatrix, parameter.means,
  parameter.acov, nobservations, nreps, initial.class, nchains,
  threshold.labels, lambda.equations, is.pi.r, is.parameter.randomized,
  parameterization.method, percent.reps.to.discard)
```

Arguments

observations	a data frame or matrix of dichotomous responses
nattributes	numeric value of number of attributes
qmatrix	a data frame or matrix of 1s and 0s indicating relation between items and attributes. This matrix specifies which items are required for mastery of each attribute (i.e., latent variable). A matrix must be a size of nItems X nAttributes
pmatrix	a numeric nclasses by nattributes matrix of all possible attribute profiles
parameter.means	a numerical vector of calibrated item and structural parameters
parameter.acov	a numerical matrix of covariances of item and structural parameters
nobservations	a numeric value indicating number of rows of the observation data frame or matrix

nreps The number of iterations in MCMC per chain

initial.class The initial value of attribute profile for each respondent

nchains The number of chains in MCMC

threshold.labels
an nclasses by nitems character matrix with appropriate item threshold labels

lambda.equations
lambda parameter equations

is.pi.r If FALSE (the default), parameter values are the type of taus and nus or lambdas and gammas else they are the type pis and rs as used in NC-RUM parameterization

is.parameter.randomized
if true parameter estimates are randomized using acov matrix

parameterization.method
optional character string of parameterization method used to calibrate parameters

percent.reps.to.discard
The percent of iterations to be discarded

Value

a list of class and parameter data frame containing all accepted iteraction of MCMC

---

observations.test *Obervations*

Description

This data set list the dichotomous responses of students for a test containing 11 items.

Usage

```
observations.test
```

Format

a data frame containing 15 students and 11 items. 1 student per row

Author(s)

Margi Dubal <margidubal@gmail.com> & Diane Losardo <dlosardo@amplify.com>

parameter.acov.DCM.Mplus.interaction.test

Covariance matrix of parameter estimates calibrated using Mplus for fully saturated model and Q-matrix with interaction.

Description

matrix of covariances of all model parameters. If NULL (the default) model parameters are not randomized for each iteration of MCMC

Usage

```
parameter.acov.DCM.Mplus.interaction.test
```

Format

A data frame

Author(s)

Margi Dubal <margidubal@gmail.com> & Diane Losardo <dlosardo@amplify.com>

parameter.acov.DCM.Mplus.test

Covariance matrix of parameter estimates calibrated using Mplus for fully saturated model.

Description

matrix of covariances of all model parameters. If NULL (the default) model parameters are not randomized for each iteration of MCMC

Usage

```
parameter.acov.DCM.Mplus.test
```

Format

A data frame

Author(s)

Margi Dubal <margidubal@gmail.com> & Diane Losardo <dlosardo@amplify.com>

`parameter.class-class` *parameter.class*

Description

parameter.class

Slots

`results` A data frame of parameter estimates

`parameter.means.DCM.kernel.Mplus.interaction.test`
Kernel parameter estimates calibrated using Mplus for fully saturated model and Q-matrix with interaction.

Description

a numerical vector of calibrated item and structural parameters. Values must be in the order of `GetParameterNames` if parametrization method is Mplus and non-kernel parameters are used. If kernel parameters values are used must be in order of `GetKernelParameterNames`

Usage

```
parameter.means.DCM.kernel.Mplus.interaction.test
```

Format

a vector or a dataframe

Author(s)

Margi Dubal <margidubal@gmail.com> & Diane Losardo <dlosardo@amplify.com>

`parameter.means.DCM.kernel.Mplus.test`

Kernel parameter estimates calibrated using Mplus for fully saturated model.

Description

a numerical vector of calibrated item and structural kernel parameters. Values must be in the order of `GetParameterNames` if parametrization method is Mplus and non-kernel parameters are used. If kernel parameters values are used must be in order of `GetKernelParameterNames`

Usage

```
parameter.means.DCM.kernel.Mplus.test
```

Format

a vector or a dataframe

Author(s)

Margi Dubal <margidubal@gmail.com> & Diane Losardo <dlosardo@amplify.com>

`parameter.means.DCM.Mplus.interaction.test`

Parameter estimates calibrated using Mplus for fully saturated model and Q-matrix with interaction.

Description

a numerical vector of calibrated item and structural parameters. Values must be in the order of `GetParameterNames` if parametrization method is Mplus and non-kernel parameters are used. If kernel parameters values are used must be in order of `GetKernelParameterNames`

Usage

```
parameter.means.DCM.Mplus.interaction.test
```

Format

a vector or a dataframe

Author(s)

Margi Dubal <margidubal@gmail.com> & Diane Losardo <dlosardo@amplify.com>

`parameter.means.DCM.Mplus.test`

Parameter estimates calibrated using Mplus for fully saturated model.

Description

a numerical vector of calibrated item and structural parameters. Values must be in the order of `GetParameterNames` if parametrization method is Mplus and non-kernel parameters are used. If kernel parameters values are used must be in order of `GetKernelParameterNames`

Usage

```
parameter.means.DCM.Mplus.test
```

Format

a vector or a dataframe

Author(s)

Margi Dubal <margidubal@gmail.com> & Diane Losardo <dlosardo@amplify.com>

`parameter.means.names.DCM.kernel.Mplus.interaction.test`

Names of kernel parameter estimates calibrated using Mplus for fully saturated model and Q-matrix with interaction.

Description

a vector of string containing names of calibrated parameter estimates.

Usage

```
parameter.means.names.DCM.kernel.Mplus.interaction.test
```

Format

A vector of string

Author(s)

Margi Dubal <margidubal@gmail.com> & Diane Losardo <dlosardo@amplify.com>

parameter.means.names.DCM.Mplus.interaction.test

Names of parameter estimates calibrated using Mplus for fully saturated model and Q-matrix with interaction.

Description

a vector of string containing names of calibrated parameter estimates.

Usage

```
parameter.means.names.DCM.Mplus.interaction.test
```

Format

A vector of string

Author(s)

Margi Dubal <margidubal@gmail.com> & Diane Losardo <dlosardo@amplify.com>

parameter.means.names.DCM.Mplus.test

Names of parameter estimates calibrated using Mplus for fully saturated model and Q-matrix with no interaction.

Description

a vector of string containing names of calibrated parameter estimates.

Usage

```
parameter.means.names.DCM.Mplus.test
```

Format

A vector of string

Author(s)

Margi Dubal <margidubal@gmail.com> & Diane Losardo <dlosardo@amplify.com>

parameter.means.names.NCRUM.interaction.test

Names of parameter estimates calibrated using Mplus for fully saturated model and Q-matrix with interaction.

Description

a vector of string containing names of calibrated parameter estimates.

Usage

```
parameter.means.names.NCRUM.interaction.test
```

Format

A vector of string

Author(s)

Margi Dubal <margidubal@gmail.com> & Diane Losardo <dlosardo@amplify.com>

parameter.means.NCRUM.interaction.test

Parameter estimates calibrated using Mplus for NC-RUM model and Q-matrix with interaction.

Description

a numerical vector of calibrated item and structural parameters. Values must be in the order of GetParameterNames if parametrization method is Mplus and non-kernel parameters are used. If kernel parameters values are used must be in order of GetKernelParameterNames

Usage

```
parameter.means.NCRUM.interaction.test
```

Format

a vector or a dataframe

Author(s)

Margi Dubal <margidubal@gmail.com> & Diane Losardo <dlosardo@amplify.com>

`plot,attribute.class,missing-method`
plot attribute.class

Description

plot attribute.class

Usage

```
## S4 method for signature 'attribute.class,missing'
plot(x, y, type = "mean", ...)
```

Arguments

`x`	`attribute.class` input object
`type`	a string containing either `mean` or `profile`
`y`	the y coordinates of points in the plot, *optional* if `x` is an appropriate structure.
`...`	Arguments to be passed to methods, such as graphical parameters (see `par`). Many methods will accept the following arguments: `type` what type of plot should be drawn. Possible types are • `"p"` for **p**oints, • `"l"` for **l**ines, • `"b"` for **b**oth, • `"c"` for the lines part alone of `"b"`, • `"o"` for both '**o**verplotted', • `"h"` for '**h**istogram' like (or 'high-density') vertical lines, • `"s"` for stair **s**teps, • `"S"` for other **s**teps, see 'Details' below, • `"n"` for no plotting. All other `type`s give a warning or an error; using, e.g., `type = "punkte"` being equivalent to `type = "p"` for S compatibility. Note that some methods, e.g. `plot.factor`, do not accept this. `main` an overall title for the plot: see `title`. `sub` a sub title for the plot: see `title`. `xlab` a title for the x axis: see `title`. `ylab` a title for the y axis: see `title`. `asp` the y/x aspect ratio, see `plot.window`.

`plot,attribute.profile.class,missing-method`
plot attribute.profile.class

Description

plot attribute.profile.class

Usage

```
## S4 method for signature 'attribute.profile.class,missing'
plot(x, y, type = "mean", ...)
```

Arguments

`x`	`attribute.profile.class` input object
`type`	a string containing either `mean` or `profile`
`y`	the y coordinates of points in the plot, *optional* if `x` is an appropriate structure.
`...`	Arguments to be passed to methods, such as graphical parameters (see `par`). Many methods will accept the following arguments:

`type` what type of plot should be drawn. Possible types are

- `"p"` for **p**oints,
- `"l"` for **l**ines,
- `"b"` for **b**oth,
- `"c"` for the lines part alone of `"b"`,
- `"o"` for both '**o**verplotted',
- `"h"` for '**h**istogram' like (or 'high-density') vertical lines,
- `"s"` for stair **s**teps,
- `"S"` for other **s**teps, see 'Details' below,
- `"n"` for no plotting.

All other `type`s give a warning or an error; using, e.g., `type = "punkte"` being equivalent to `type = "p"` for S compatibility. Note that some methods, e.g. `plot.factor`, do not accept this.

`main` an overall title for the plot: see `title`.

`sub` a sub title for the plot: see `title`.

`xlab` a title for the x axis: see `title`.

`ylab` a title for the y axis: see `title`.

`asp` the y/x aspect ratio, see `plot.window`.

`plot,dcm.scorer.class,missing-method` *plot of dcm.scorer.class*

Description

plot of dcm.scorer.class

Usage

```
## S4 method for signature 'dcm.scorer.class,missing'
plot(x, y, type = "attr.means", ...)
```

Arguments

`x`	`dcm.scorer.class` input object
`type`	a string containing one of the result type
`y`	the y coordinates of points in the plot, *optional* if `x` is an appropriate structure.
`...`	Arguments to be passed to methods, such as graphical parameters (see `par`). Many methods will accept the following arguments:

`type` what type of plot should be drawn. Possible types are

- `"p"` for **p**oints,
- `"l"` for **l**ines,
- `"b"` for **b**oth,
- `"c"` for the lines part alone of `"b"`,
- `"o"` for both '**o**verplotted',
- `"h"` for '**h**istogram' like (or 'high-density') vertical lines,
- `"s"` for stair **s**teps,
- `"S"` for other **s**teps, see 'Details' below,
- `"n"` for no plotting.

All other `type`s give a warning or an error; using, e.g., `type = "punkte"` being equivalent to `type = "p"` for S compatibility. Note that some methods, e.g. `plot.factor`, do not accept this.

`main` an overall title for the plot: see `title`.

`sub` a sub title for the plot: see `title`.

`xlab` a title for the x axis: see `title`.

`ylab` a title for the y axis: see `title`.

`asp` the y/x aspect ratio, see `plot.window`.

---

print,attribute.class-method
print attribute.class

Description

print attribute.class

Usage

```
## S4 method for signature 'attribute.class'
print(x)
```

Arguments

x	attribute.class input object

---

print,attribute.profile.class-method
print attribute.profile.class

Description

print attribute.profile.class

Usage

```
## S4 method for signature 'attribute.profile.class'
print(x)
```

Arguments

x	attribute.profile.class input object

qmatrix.test *Q-matrix*

Description

This data set list 1s and 0s indicating relation between items and attributes. This matrix specifies which items are required for mastery of each attribute (i.e., latent variable). The matrix must be a size of `nitems X nattributes`

Usage

```
qmatrix.test
```

Format

a matrix containing 11 items and 3 attributes

Author(s)

Margi Dubal <margidubal@gmail.com> & Diane Losardo <dlosardo@amplify.com>

qmatrix.test.interaction *Q-matrix with interaction*

Description

This data set list 1s and 0s indicating relation between items and attributes. This matrix specifies which items are required for mastery of each attribute (i.e., latent variable). The matrix must be a size of `nitems X nattributes`

Usage

```
qmatrix.interaction.test
```

Format

a matrix containing 11 items and 3 attributes

Author(s)

Margi Dubal <margidubal@gmail.com> & Diane Losardo <dlosardo@amplify.com>

SampleParameterEstimates *Randomly sample parameter estimates*

Description

Given parameter estimates and asymptotic covaraince matrix of parameter estimates, randomly samples a new set of estimates

Usage

```
SampleParameterEstimates(means, acov)
```

Arguments

means	a numerical vector of calibrated item and structural parameters
acov	a numerical matrix of covariances of item and structural parameters

Value

A vector of randomly sampled parameter estimate values

ScoreDCM *Score dichotomous response data using DCM and MCMC*

Description

Given dichotomous response data, a Q-matrix (relation between items and attributes) and calibrated item and structural parameters ScoreDCM estimates posterior probabilities of attribute profiles of respondents using a Diagnostic Classification Model (DCM) and Markov Chain Monte Carlo (MCMC) method. The estimation procedure uses the loglinear cognitive diagnostic modeling (LDCM) framework that allows for the estimation of a host of DCMs such as DCM, DINA, C-RUM, NIDO, NIDA, NC-RUM etc.

Usage

```
ScoreDCM(observations, qmatrix, parameter.means, parameter.acov= NULL,
parameterization.method = "Mplus", is.kernel.parameters = FALSE, model.type = NULL,
nreps = 1000 , nchains = 1, initial.class = 1,  percent.reps.to.discard = 5)
```

Arguments

observations	a data frame or matrix of dichotomous responses in wide format (nobservations X nitems)
qmatrix	a data frame or matrix of 1s and 0s indicating relation between items and attributes. This matrix specifies which items are required for mastery of each attribute (i.e., latent variable). The matrix must be a size of nitems X nattributes
parameter.means	a numerical vector of calibrated item and structural parameters. Values must be in the order of GetParameterNames if parametrization method is Mplus and non-kernel parameters are used. If kernel parameters values are used must be in order of GetKernelParameterNames
parameter.acov	optional matrix of covariances of all model parameters. If NULL (the default) model parameters are not randomized for each iteration of MCMC
parameterization.method	optional character string of parameterization method used to calibrate parameters. If not specified then the default will be set to Mplus
is.kernel.parameters	If FALSE (the default), parameter values are of type taus and nus else they are of type kernel parameters, i.e., lambdas and gammas
model.type	If is.kernel.parameter is TRUE, model type must be one of DCM, DINA, CRUM, DINO, NIDO, NCRUM. Kernel parameters are different for each model type
nreps	The number of iterations in MCMC per chain. If not specified the default is 1000
nchains	The number of chains in MCMC. If not specified the default is 1
initial.class	The initial value of the attribute profile for each respondent. If not specified the default is 1
percent.reps.to.discard	The percent of iterations to be discarded. If not specifed then the default is 5

Value

ScoreDCM returns an object of class dcm.scorer.class; a list consisting of

inputs a list of all input arguments to the funciton like observations, Q-matrix, parameter.means, etc.

mcmc.inputs a list of all mcmc inputs

results an object of all.results.class class; a list consisting of

- **attribute.profile.result** an object of attribute.profile.class
- **attribute.result** an object of attribute.class
- **parameter.result** an object of parameter.class

mcmc.outputs a list of all mcmc output

The function summary is used to obtain and print a summary of MCMC runs in the form of probabilities of mastering each attribute and attribute profile probabilities. The function plot is used to plot the aggregated mean of both attribute mastery (type = "attr.means") and attribute profile probably (type = "attr.profile.means") across all respondents. Other plot options include attribute mastery (type = "attr.profiles") and attribute profile probabilities (type = "attr.profile.profiles") of individual respondents.

Author(s)

Margi Dubal <margidubal@gmail.com> & Diane Losardo <dlosardo@amplify.com>

Examples

```
## Not run:

data(obervations.test)
data(qmatrix.test)
parameter.names <- GetParameterNames(qmatrix.test, ncol(qmatrix.test))
parameter.names
data(parameter.means.DCM.Mplus.test)
obj <- ScoreDCM(observations = observations.test, qmatrix = qmatrix.test
, parameter.means = parameter.means.DCM.Mplus.test)
summary(obj)
plot(obj)

## End(Not run)
```

---

summary,attribute.class-method

summary attribute.class

Description

summary attribute.class

Usage

```
## S4 method for signature 'attribute.class'
summary(object, verbose = TRUE, ...)
```

Arguments

object	attribute.class input object
verbose	a logical. If TRUE, additional diagnostics are printed.
...	additional arguments affecting the summary produced.

`summary,attribute.profile.class-method`

summary attribute.profile.class

Description

summary attribute.profile.class

Usage

```
## S4 method for signature 'attribute.profile.class'
summary(object, verbose = TRUE, ...)
```

Arguments

`object`	`attribute.profile.class` input object
`verbose`	a logical. If TRUE, additional diagnostics are printed.
`...`	additional arguments affecting the summary produced.

`summary,dcm.scorer.class-method`

summary of dcm.scorer.class

Description

summary of dcm.scorer.class

Usage

```
## S4 method for signature 'dcm.scorer.class'
summary(object, verbose = TRUE, ...)
```

Arguments

`object`	`dcm.scorer.class` input object
`verbose`	a logical. If TRUE, additional diagnostics are printed
`...`	additional arguments affecting the summary produced.

Index