Package ‘pleLMA’

October 6, 2021

Type Package
Title Pseudo-Likelihood Estimation of Log-Multiplicative Association Models
Version 0.2.1
Author Carolyn J. Anderson
Maintainer Carolyn J. Anderson <cja@illinois.edu>
Description Log-multiplicative association models (LMA) are models for cross-classifications of categorical variables where interactions are represented by products of category scale values and an association parameter. Maximum likelihood estimation (MLE) fails for moderate to large numbers of categorical variables. The 'pleLMA' package overcomes this limitation of MLE by using pseudo-likelihood estimation to fit the models to small or large cross-classifications dichotomous or multi-category variables. Originally proposed by Besag (1974, <doi:10.1111/j.2517-6161.1974.tb00999.x>), pseudo-likelihood estimation takes large complex models and breaks it down into smaller ones. Rather than maximizing the likelihood of the joint distribution of all the variables, a pseudo-likelihood function, which is the product likelihoods from conditional distributions, is maximized. LMA models can be derived from a number of different frameworks including (but not limited to) graphical models and uni-dimensional and multi-dimensional item response theory models. More details about the models and estimation can be found in the vignette.
License GPL (>= 3)
Encoding UTF-8
LazyData true
Imports mlogit, dfidx, stats, graphics
Suggests ggplot2, knitr, rmarkdown, testthat
RoxygenNote 7.1.2
convergence.stats

Computes statistics to assess convergence of the nominal model
convergence.stats

Description
For the nominal model, convergence statistics are computed for each item, as well as the algorithm as a whole. The main argument is the history or log from fitting item regressions. The convergence statistics are the differences between current values of the log likelihoods and item parameter estimates and those from the previous iteration. The maximum over item of the differences of the log likelihood values is used to determine convergence of the pseudo-likelihood algorithm. This function is used internally, but it can also be used after fitting a model to examine how many iterations are required for parameter estimates to get close to the final values and whether any item parameters are still changing.

Usage
convergence.stats(item.log, nitems, nless, LambdaName, NuName)

Arguments
item.log Iteration history of items’ log likelihoods and parameter estimates
nitems Number of items
nless Number of unique marginal terms (i.e., lambdas) and unique category scale values (i.e., nus)
LambdaName Names of lambdas in item regressions
NuName Names of nu in item regressions

Value
diff.last Differences between item loglikes & parameters on last two iterations
criterion.loglike Maximum over items of the absolute value of LogLike differences
criterion.items Sum of item differences of item parameters

Examples

# 9 items from dass data for 250 cases
data(dass)
inData <- dass[1:250,c("d1", "d2", "d3", "a1","a2","a3","s1","s2","s3")]

#--- input for uni-dimensional
inTraitAdj <- matrix(1, nrow=1, ncol=1)
inItemTraitAdj <- matrix(1, nrow=9, ncol=1)

#--- Uni-dimensional Nominal Model
n1 <- ple.lma(inData, model.type="nominal", inItemTraitAdj,inTraitAdj, tol=1e-02)

# Since this function in internal to fit.nominal, need to also run
s <- set.up(inData, model.type='nominal', inTraitAdj, inItemTraitAdj)

convergence.stats(n1$item.log, n1$nitems, n1$nless, s$LambdaName, s$NuName)
convergenceGPCM

Computes statistics to assess convergence for generalized partial credit models

Description

For the generalized partial credit model, convergence statistics are computed for each item, as well as the algorithm as a whole. The convergence statistics are the differences between current values of the log likelihoods and item parameter estimates and those from the previous iteration. The maximum over items’ differences of the log likelihood values is used to determine convergence of the pseudo-likelihood algorithm. This function is used internally, but it can also be used after fitting a model to examine how many iterations are required for parameter estimates to get close to the final values and whether any item parameters are still changing.

Usage

convergenceGPCM(item.log, nitems, ncat, nless, LambdaName)

Arguments

item.log Iteration history of items’ log likelihoods and parameter estimates
nitems Number of items
ncat Number of categories
nless Number of unique lambdas
LambdaName Names of lambdas in formula used to fit item regressions (internal to fit_gpcm)

Value

diff.last Differences between item’s log likelihoods and parameters for each item
criterion.loglike Maximum overs item of the absolute value of differences between item LogLike values
criterions.items Sum of item differences of their parameters
Examples

```r
# 9 items from dass data for 250 cases
data(dass)
inData <- dass[1:250, c("d1", "d2", "d3", "a1", "a2", "a3", "s1", "s2", "s3")]

#--- input for uni-dimensional
inTraitAdj <- matrix(1, nrow=1, ncol=1)
inItemTraitAdj <- matrix(1, nrow=9, ncol=1)

#--- fit Unidimensional gpcm Model
g1 <- ple.lma(inData, model.type="gpcm", inItemTraitAdj, inTraitAdj, tol=1e-03)

# Since convergenceGPCM is internal to fit.gpcm, need to get 'LambdaName'
s <- set.up(inData, model.type='gpcm', inTraitAdj, inItemTraitAdj)
convergenceGPCM(g1$item.log, g1$nitems, g1$ncat, g1$nless, s$LambdaName)

#--- Multidimensional models
#--- re-define inTraitAdj and inItemTraitAdj for 3 dimensional models
inData <- dass[1:250, c("d1", "d2", "d3", "a1", "a2", "a3", "s1", "s2", "s3")]
inTraitAdj <- matrix(1, nrow=3, ncol=3)
dpress <- matrix(c(1, 0, 0), nrow=3, ncol=3, byrow=TRUE)
anxiety <- matrix(c(0, 1, 0), nrow=3, ncol=3, byrow=TRUE)
stress <- matrix(c(0, 0, 1), nrow=3, ncol=3, byrow=TRUE)
das <- list(dpress, anxiety, stress)
inItemTraitAdj <- rbind(das[[1]], das[[2]], das[[3]])

#--- 3 dimensional gpcm
g3 <- ple.lma(inData, model.type="gpcm", inItemTraitAdj, inTraitAdj, tol=1e-03)
s <- set.up(inData, model.type='gpcm', inTraitAdj, inItemTraitAdj)
convergenceGPCM(g1$item.log, g1$nitems, g1$ncat, g1$nless, s$LambdaName)
```

dass

Dateframe of responses to items from depression, anxiety, and stress scales

Description

The dass data are responses from a random sample of 1,000 individuals collected during the period 2017 – 2019. The data were retrieved July 2020. The items included here are on scales designed to measure depression (14 items), anxiety (13 items), and stress (15 items). The 42 items were presented online and in random order. Respondents were instructed to consider the last week when responding to the items using the following categories: (1) Did not apply to me at all; (2) Applied to me to some degree, or some of the time; (3) Applied to me to a considerable degree, or a good part of the time; (4) Applied to me very much, or most of the time.
Usage

dass

Format

A data frame with 1,000 rows (respondents) and 42 columns (items):

d1  I couldn’t seem to experience any positive feeling at all.
d2  I just couldn’t seem to get going
d3  I felt that I had nothing to look forward to
d4  I felt sad and depressed
d5  I felt that I had lost interest in just about everything
d6  I felt I wasn’t worth much as a person
d7  I felt that life wasn’t worthwhile
d8  I couldn’t seem to get any enjoyment out of the things I did
d9  I felt down-hearted and blue
d10 I was unable to become enthusiastic about anything
d11 I felt I was pretty worthless
d12 I could see nothing in the future to be hopeful about
d13 I felt that life was meaningless
d14 I found it difficult to work up the initiative to do things

a1  I was aware of dryness of my mouth
a2  I experienced breathing difficulty (eg, excessively rapid breathing, breathlessness in the absence of physical exertion)
a3  I had a feeling of shakiness (eg, legs going to give way)
a4  I felt that I was using a lot of nervous energy
a5  I had a feeling of faintness
a6  I perspired noticeably (eg, hands sweaty) in the absence of high temperatures or physical exertion
a7  I felt scared without any good reason
a8  I had difficulty in swallowing
a9  I was aware of the action of my heart in the absence of physical exertion (eg, sense of heart rate increase, heart missing a beat)
a10 I felt I was close to panic
a11 I felt terrified
a12 I was worried about situations in which I might panic and make a fool of myself
a13 I experienced trembling (eg, in the hands)
s1  I found myself getting upset by quite trivial things
s2  I tended to over-react to situations
s3  I found it difficult to relax
s4  I found myself in situations that made me so anxious I was most relieved when they ended
s5  I found myself getting upset rather easily
s6  I found myself getting impatient when I was delayed in any way (eg, elevators, traffic lights,
    being kept waiting)
s7  I felt that I was rather touchy
s8  I found it hard to wind down
s9  I found that I was very irritable
s10 I found it hard to calm down after something upset me
s11 I feared that I would be thrown off by some trivial but unfamiliar task
s12 I found it difficult to tolerate interruptions to what I was doing
s13 I was in a state of nervous tension
s14 I was intolerant of anything that kept me from getting on with what I was doing
s15 I found myself getting agitated

Source

https://openpsychometrics.org

Examples

```r
data(dass)
head(dass)
```

error.check | Checks for basic errors in input to the 'ple.lma' function

Description

This function looks at the input to the main function (ple.lma) and checks for 11 different possible
errors. If an error is detected, the function issues a warning and stops any further execution. This
function is internal to 'ple.lma' but can be used outside of the wrapper function.

Usage

```r
error.check(inData, model.type, inTraitAdj = NULL, inItemTraitAdj = NULL)
```

Arguments

- `inData`: Data frame with columns corresponding to categorical variables and rows to the
  number of cases
- `model.type`: Type of model that will be fit to data
- `inTraitAdj`: Trait x Trait adjacency matrix (not required for independence)
- `inItemTraitAdj`: Item x Trait adjacency matrix (not required for independence)
**Value**

Message whether error was detected in input, and if so the nature of the error

**Examples**

```r
#--- some data
data(dass)
inData <- dass[1:250,c("d1", "d2", "d3", "a1","a2","a3","s1","s2","s3")]

#--- no errors
error.check(inData, model.type="independence")

#--- for unidimensional
inTraitAdj <- matrix(1, nrow=1, ncol=1)
inItemTraitAdj <- matrix(1, nrow=9, ncol=1)

#--- no errors
error.check(inData, model.type="rasch", inTraitAdj, inItemTraitAdj)
error.check(inData, model.type="gpcm", inTraitAdj, inItemTraitAdj)
error.check(inData, model.type="nominal", inTraitAdj, inItemTraitAdj)
```

---

**fit.gpcm**  
*Fits LMA model where category scale values equal a_im * x_j*

**Description**

Function estimates the parameters of LMA models with fixed category scores multiplied by an item weight parameter. This function can be used to estimate the LMA model corresponding to is a generalized partial credit model for multi-category items and the 2 parameter logistic model for dichotomous items. The function sets up log objects and model formula. In the case of unidimensional models, the function iterates over item regressions; whereas, for multidimensional models, the function iterates between the item and phi regressions. This function is called from `ple.lma`, but can be run outside of `ple.lma`.

**Usage**

```r
fit.gpcm(
  Master,
  Phi.mat,
  PersonByItem,
  TraitByTrait,
  item.by.trait,
  tol,
  npersons,
  nitems,
  ncat,
)```
Arguments

Master Master data set in long format
Phi.mat Matrix of starting values of association parameters
PersonByItem Person by item matrix of responses (same as inData)
TraitByTrait Trait by trait adjacency matrix (same as inTraitAdj)
item.by.trait Item by trait vector indicating trait item load on (same as inItemTraitAdj)
tol Criterion used to determine convergence
npersons Number of persons
nitems Number of items
ncat Number of categories per item
nless Number of categories minus 1 (i.e., unique lambdas)
ntraits Number of latent traits
Maxnphi Number of phi parameters to be estimated
pq.mat Used to compute rest-scores and totals
starting.sv Fixed category scores
ItemNames Names of items needed label output
LambdaName Names of lambdas needed for formula of the item regressions
LambdaNames Names of lambdas needed for formula of the stacked regression
PhiNames Name of phi parameters (Null for uni-dimensional models)

Value

item.log History over iterations of the algorithm for items’ log likelihood, lambda, and a parameter
phi.log History over iterations of the algorithm for log likelihood, lambdas nd phi parameters
criterion Current value of the convergence statistic which is the maximum of items’ absolute differences between the current and previous value of the log likelihood
estimates An item by parameter matrix of estimated item parameter where the first column are items’ log likelihood
Phi.mat Estimated matrix of association parameters
fitem Formula for item data
fit.independence

Fits the log-linear model of independence

Description

This function fits the log-linear model of independence (i.e., only includes marginal effect terms) using pseudo-likelihood estimation. This provides a baseline model with which to compare other models. The independence maximum of the loglikelihood can be used as a measure of no association. The input to the function is only the Master data set and the names of marginal effect terms and items, all of which are created by the 'set.up' function. This function is called from 'ple.lma' or can be run output of wrapper.

Usage

fit.independence(Master, LambdaNames, LambdaName, ItemNames)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master</td>
<td>Master data set from set.up</td>
</tr>
<tr>
<td>LambdaNames</td>
<td>Needed to define formula</td>
</tr>
<tr>
<td>LambdaName</td>
<td>Used for column names of matrix estimates</td>
</tr>
<tr>
<td>ItemNames</td>
<td>Used for row names of number of item by parameter matrix of estimated Lambda</td>
</tr>
</tbody>
</table>

Examples

data(dass)
inData <- dass[1:250,c("d1", "d2", "a1","a2","s1","s2")]
#--- unidimensional
inTraitAdj <- matrix(1, nrow=1, ncol=1)
inItemTraitAdj <- matrix(1, nrow=6, ncol=1)

# Need to set up data
s <- set.up(inData, model.type='gpcm', inTraitAdj, inItemTraitAdj, tol=le-03)

g <- fit.gpcm(s$Master, s$Phi.mat, s$PersonByItem, s$TraitByTrait, s$Item.by.trait, s$tol, s$npersons, s$nitems, s$ncat, s$nless, s$ntraits, s$Maxnphi, s$pq.mat, s$starting.sv, s$ItemNames, s$LambdaName, s$LambdaNames, s$PhiNames)
Value

phi.mlogit Parameters estimates and mlpl = logLike output from mnlogit
fstack Formual used in stacked regression
estimates Item by parameter estimates matrix
mlpl.phi Maximum of log pseudo-likelihood from stacked regression
AIC Akaike information criterion for pseudo-likelihood (smaller is better)
BIC Bayesian information criterion for pseudo-likelihood (smaller is better)

Examples

```r
#--- data and set-up
data(dass)
inData <- dass[1:250,c("d1", "d2", "d3", "a1","a2","a3","s1","s2","s3")]
s <- set.up(inData, model.type="QQVar"

#--- fit independence model
ind <- fit.independence(s$Master, s$LambdaNames, s$LambdaName, s$ItemNames)
```

---

**fit.nominal**  
*Fits the nominal model*

Description

Function estimates the parameters of LMA models where the category scale are estimated. The function can be used to estimate the parameters of the LMA model corresponding the nominal model (for multi-category items) and the 2 parameter logistic model for dichotomous items. The function sets up log object(s) and model formula. In the case of unidimensional models, the function iterates over item regressions; whereas, for multidimensional models, the function iterates between the item and phi regressions. This function is called from 'ple.lma', but can be run outside of 'ple.lma'.

Usage

```r
fit.nominal(
  Master,
  Phi.mat,
  starting.sv,
  pq.mat,
  tol,
  PersonByItem,
  TraitByTrait,
  ItemByTrait,
  ItemNames,
  LambdaNames,
```
NuNames,
LambdaName,
NuName,
PhiNames,
npersons,
nitems,
ncat,
nless,
ntraits,
Maxnphi
)

**Arguments**

- **Master**: Master data set in long format
- **Phi.mat**: Matrix of starting values of the association parameters
- **starting.sv**: Matrix starting values category scale values
- **pq.mat**: Array used compute rest scores and total scores
- **tol**: Value used to determine convergence of algorithm
- **PersonByItem**: Same as inData (rows are response patterns)
- **TraitByTrait**: Same as inTraitAdj (trait x trait adjacency)
- **ItemByTrait**: Same as inItemTraitAdj (item x trait adjacency)
- **item.by.trait**: One dimensional array indicating trait item loads on
- **ItemNames**: Names of items in inData (i.e. columns names of categorical variables)
- **LambdaNames**: Lambda names used in the Master and stacked data frames
- **NuNames**: Nu names in Master data frame
- **LambdaName**: Lambda names in formula for items
- **NuName**: Nu names in formula for item regressions
- **PhiNames**: Association parameter names for stacked regression
- **npersons**: Number of persons
- **nitems**: Number of items
- **ncat**: Number of categories per item
- **nless**: ncat-1 = number unique lambda and unique nus
- **ntraits**: Number of traits
- **Maxnphi**: Number of association parameters

**Value**

- **item.log**: Iteration history of LogLike, lambda, and item parameters
- **phi.log**: Iteration history of LogLike, lambdas and phi parameters
- **criterion**: Current value of the convergence statistic
- **estimates**: Item x parameter matrix: LogLike, lambda and scale values
fit.rasch

Fits an LMA using fixed category scores

Description

The LMA model with fixed category scores is fit by this function and the model corresponds to models in the Rasch family of item response models. The category scores can be set by either the user or the package defaults. The default category scores are equally spaced, sum to zero, and sum of squares equal 1. Scores can be set by user by specifying them in the item by category matrix of 'starting.sv'. The pseudo-likelihood algorithm only runs a single stacked regression. This function is called from 'ple.lma' but can also be run outside of the main wrapper function.

Usage

fit.rasch(
  Master,
  npersons,
  nitems,
  ncat,
nless, Maxnphi, pq.mat, starting.sv, LambdaNames, PhiNames, ItemNames, LambdaName, ntraits }

Arguments

Master Master data set in long format
npersons Number of persons
nitems Number of items
ncat Number of categories
nless Number of unique Lambdas (i.e., ncat-1)
Maxnphi Number of phi parameters
pq.mat One dimensional array to compute rest-scores
starting.sv Fixed category scores
LambdaNames Names of lambda parameters in Master and formula for stacked regression
PhiNames Names of association parameters
ItemNames Names of items
LambdaName Names of lambdas used in output
ntraits Number of traits

Value

estimates An item by parameter matrix of the maximum of the log likelihood, estimated item parameters (i.e., Lambdas), and the values of the fixed category scores.

fstack Formula for stacked regression

phi.mlogit Results from mlogit for stacked regression

estimates An item x parameter estimate matrix and fixed category scores used

Phi.mat Estimated phi parameters

mlpl.phi Value of maximum of log pseudo-likelihood function from the stacked regression

AIC Akaike information criterion for pseudo-likelihood (smaller is better)

BIC Bayesian information criterion for pseudo-likelihood (smaller is better)
Examples

```r
#--- data(dass)
inData <- dass[1:250, c("d1", "d2", "d3", "a1", "a2", "a3", "s1", "s2", "s3")]

#--- unidimensional
inTraitAdj <- matrix(1, nrow=1, ncol=1)
inItemTraitAdj <- matrix(1, nrow=9, ncol=1)

s <- set.up(inData, model.type="rasch", inTraitAdj, inItemTraitAdj)

r <- fit.rasch(s$Master, s$npersons, s$nitems, s$ncat, s$nless, s$Maxnphi,
               s$pq.mat, s$starting.sv, s$LambdaNames, s$PhiNames, s$ItemNames,
               s$LambdaName, s$ntraits)
```

---

**FitStack**

*Up-dates association parameters of the nominal model*

**Description**

Discrete choice model (conditional multinomial logistic regression model) is fit to stacked data to update the matrix of association parameters of the LMA that corresponds to the nominal item response model. This is a function internal to `fit.nominal` and is used for multi-dimensional models. The function is similar to `fit.StackGPCM`. This function is unlikely to be run outside of `fit.nominal` or `ple.lma`.

**Usage**

```r
FitStack(
  Master,
  item.log,
  phi.log,
  fstack,
  TraitByTrait,
  pq.mat,
  npersons,
  nitems,
  ncat,
  nless,
  ntraits,
  Maxnphi,
  PhiNames,
  LambdaNames
)
```

**Arguments**

- **Master**  
  Master data set from which stacked data is created
fitStackGPCM

item.log  Last row contains current scale values (item.history)
phi.log Last row contains current estimates of phi
fstack Formula for stacked regression
TraitByTrait inTraitAdj matrix
pq.mat Summing array to get rest scores and totals
n.items Number of persons
n.items Number of items
ncat Number of categories per item
nless Number of categories less 1 (unique lambdas & unique nus)
ntraits Number of latent traits
Maxphi Number of phis to be estimated
PhiNames Names of the Phi parameters
LambdaNames Names of lambdas that correspond to those in Master

Value

Phi.mat Matrix of up-dated estimates of association (phi) parameters
phi.log History of iterations log likelihood and estimates of lambda and phi parameters

Description

Discrete choice model (conditional multinomial logistic regression) is fit to stacked data to update matrix of association parameters of the LMA that corresponds to the generalized partial credit model. This function is called from 'fit.gpcm', which is called from 'ple.lma'. It is unlikely that it would be run outside of these wrappers. It is only slightly different from 'fitStack' for nominal models.

Usage

fitStackGPCM(
  Master,
  item.log,
  phi.log,
  fstack,
  TraitByTrait,
  starting.sv,
  n.persons,
  n.items,
  n.cat,
Arguments

Master Master data set from which stacked data is created
item.log Needed to get most recent values of scale values (item.log)
phi.log History of estimates parameters from stacked regression
fstack Formula for stacked regression
TraitByTrait inTraitAdj matrix
starting.sv Fixed category scores
npersons Number of persons
nitems Number of items
ncat Number of categories per item
nless Number of unique lambdas and unique nus per item
ntraits Number of latent traits
Maxnphi Number of phi parameters to be estimated (NULL for 1 dimensional)
pq.mat Used to compute rest-scores and totals
LambdaNames Needed for formula and data for up-dating phi (stacked regression)
PhiNames Null for 1D models

Value

Phi.mat Updated matrix of phi parameters
item.log of iterations for LogLike, Lambda and phi parameters

Description

This function is internal to the function 'fit.gpcm' and performs the item regressions. It is a core function of the pseudo-likelihood algorithm for items of the GPCM. The function calls function 'itemGPCM.data' to create the data for input into 'mlogit', which is use to fit a conditional multinomial model for each item. The up-dated scale values are put into the Master data frame and the 'item.log' array. It generally would not run outside of 'fit.gpcm' or 'ple.lma'.

item.gpcm Estimates item parameters of LMA with linear restrictions on category scores
Usage

item.gpcm(
  Master,
  item.log,
  Phi.mat,
  fitem,
  TraitByTrait,
  PersonByItem,
  npersons,
  nitems,
  ncat,
  nless,
  ntraits,
  Maxnphi,
  pq.mat,
  starting.sv,
  LambdaName
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master</td>
<td>Master data frame</td>
</tr>
<tr>
<td>item.log</td>
<td>History over iterations of items’ log likelihood and estimates of lambda, and item parameters</td>
</tr>
<tr>
<td>Phi.mat</td>
<td>Starting value of matrix of association parameters (optional)</td>
</tr>
<tr>
<td>fitem</td>
<td>Formula for item regressions</td>
</tr>
<tr>
<td>TraitByTrait</td>
<td>Trait adjacency matrix (same as inTraitAdj)</td>
</tr>
<tr>
<td>PersonByItem</td>
<td>Same as inData</td>
</tr>
<tr>
<td>npersons</td>
<td>Number of persons</td>
</tr>
<tr>
<td>nitems</td>
<td>Number of items</td>
</tr>
<tr>
<td>ncat</td>
<td>Number of categories per item</td>
</tr>
<tr>
<td>nless</td>
<td>Number of unique lambdas and unique nus per item</td>
</tr>
<tr>
<td>ntraits</td>
<td>Number of latent traits</td>
</tr>
<tr>
<td>Maxnphi</td>
<td>Number of phi parameters to be estimated (NULL for 1 dimensional)</td>
</tr>
<tr>
<td>pq.mat</td>
<td>Used to compute rest-scores and totals</td>
</tr>
<tr>
<td>starting.sv</td>
<td>Fixed category scores</td>
</tr>
<tr>
<td>LambdaName</td>
<td>Lambda names for formula for items item regressions</td>
</tr>
</tbody>
</table>

Value

Master Master data frame with up-dated category scores for items
item.log Up-dated history array over iterations of the algorithm of items’ log likelihood and estimated lambda and alpha parameters
ItemData  

Prepares data for up-dating scale value parameters of nominal model

Description

This function creates a data frame, 'item data', to be used in the item regressions for nominal models. It computes weighted rest scores and totals, including correlated traits. This function is internal to 'ItemLoop' and it is unlikely to be run outside of 'fit.nominal' or 'ple.lma'.

Usage

ItemData(
    Master,  
    ItemID,  
    Phi.mat = Phi.mat,  
    npersons,  
    nitems,  
    ntraits,  
    ncat,  
    nless,  
    TraitByTrait,  
    pq.mat,  
    LambdaName,  
    NuName
)

Arguments

Master  
    Master data frame
ItemID  
    The item for which scale values are being up-dated
Phi.mat  
    Current estimate conditional covariance matrix (i.e., association paramters)
npersons  
    Number of persons
nitems  
    Number of items
ntraits  
    Number of traits
ncat  
    Number of categories
nless  
    Number of unique lambdas and unique nus
TraitByTrait  
    Same as inTraitAdj
pq.mat  
    One dimemsinal array used to get rest and totals scores
LambdaName  
    Name of lambdas for in item regression
NuName  
    Name of nus in item regression

Value

ItemFit Data frame used to up-date scale values
**ItemGPCM.data**

*Creates data frame up-dating phi parameters of the gpcm.*

**Description**

This function creates a data frame, 'gpcm.item.data', to be used in the item regressions of LMA models where category scales values are fixed. Sets up data for up-dating alpha parameters of the LMA that corresponds to the GPCM. This function is internal to 'item.gpcm' and it is unlikely to be run outside of 'fit.gpcm' or 'ple.lma'.

**Usage**

```r
ItemGPCM.data(
  Master,
  ItemID,
  Phi.mat,
  TraitByTrait,
  pq.mat,
  starting.sv,
  npersons,
  nitems,
  ncat,
  nless,
  ntraits,
  LambdaName
)
```

**Arguments**

- **Master**: Data frame of all data in long format
- **ItemID**: Specifies the item for which a data frame is created to be input into an item regression
- **Phi.mat**: Starting value of matrix of association parameters
- **TraitByTrait**: Trait by trait adjacency matrix (same as inTraitAdj)
- **pq.mat**: Array used to compute rest scores
- **starting.sv**: Matrix of item category scores that are fixed
- **npersons**: Number of persons
- **nitems**: Number of items
- **ncat**: Number of categories
- **nless**: Number of unique lambdas (ncat-1)
- **ntraits**: Number of latent traits
- **LambdaName**: Names for lambda for item regression
ItemLoop

**Value**

gpcm.item.data Data frame for item to be used up-dated in an item regression for specified item

---

**ItemLoop**  loops through items and up-dates estimates of scale values for each item in Nominal Model

---

**Description**

This is a core function of the pseudo-likelihood algorithm for items of the nominal model. The function calls function 'ItemData' to create the data frame for input into 'mlogit', which is use to fit a conditional multinomial model (i.e., a discrete choice model) for each item. The up-dated scale are put into the Master data frame and added to the item.log array. Generally the function would not run outside of 'fit.nominal' or 'ple.lma'.

**Usage**

```
ItemLoop(
  Master,
  item.log,
  Phi.mat = Phi.mat,
  PersonByItem,
  npersons,
  nitems,
  ntraits,
  ncat,
  nless,
  TraitByTrait,
  pq.mat,
  LambdaName,
  NuName,
  fitem
)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master</td>
<td>Current master frame</td>
</tr>
<tr>
<td>item.log</td>
<td>Iteration history for the items parameters</td>
</tr>
<tr>
<td>Phi.mat</td>
<td>Current estimate of Phi.mat</td>
</tr>
<tr>
<td>PersonByItem</td>
<td>Person by item adjacency matrix (same as inData)</td>
</tr>
<tr>
<td>npersons</td>
<td>Number of persons</td>
</tr>
<tr>
<td>nitems</td>
<td>Number of items</td>
</tr>
<tr>
<td>ntraits</td>
<td>Number of traits</td>
</tr>
<tr>
<td>ncat</td>
<td>Number of categories</td>
</tr>
<tr>
<td>Term</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>nless</td>
<td>Number of unique lambda and unique nus (ncat-1)</td>
</tr>
<tr>
<td>TraitByTrait</td>
<td>TraitsByTrait adjacency matrix (sam as TraitAdj)</td>
</tr>
<tr>
<td>pq.mat</td>
<td>One dimensional array for computing rest-scores</td>
</tr>
<tr>
<td>LambdaName</td>
<td>Marginal effect names used in formula and item data frame for item regressions</td>
</tr>
<tr>
<td>NuName</td>
<td>Scale values names in used in formula and item data frame for item regressions</td>
</tr>
<tr>
<td>fitem</td>
<td>Formula for item regression</td>
</tr>
</tbody>
</table>

### Value

- **Master**: Master data frame up-dated scale values for all items
- **item.log**: Iteration history of item parameters where the last row showing results from the current iteration

### Description

This is a utility function that plots the estimated item parameters by iterations. The plots can be used to determine how many iterations are required to get close to final values. This functions can be used uni- or multi-dimensional gpcm and models. The number of pages equals the number of items and each page has the plots of marginal effects (left side) and category scale values or alph parameters (right).

### Usage

```r
iterationPlot(model.fit)
```

### Arguments

- **model.fit**: Object from fitting nominal or gpcm model to data

### Value

Plots of estimated parameters by iteration

### Examples

```r
data(dass)
inData <- dass[1:250, c("d1", "d2", "d3", "a1", "a2", "a3", "s1", "s2", "s3")]
#--- input for uni-dimensional
inTraitAdj <- matrix(1, nrow=1, ncol=1)
inItemTraitAdj <- matrix(1, nrow=9, ncol=1)
```
```r
#--- generalized partial credit model
g1 <- ple.lma(inData, model.type="gpcm", inItemTraitAdj, inTraitAdj)
iterationPlot(g1)

#--- nominal response model
n1 <- ple.lma(inData, model.type="nominal", inItemTraitAdj, inTraitAdj)
iterationPlot(n1)

#--- Multidimensional models
inTraitAdj <- matrix(1, nrow=3, ncol=3)
dpress <- matrix(c(1,0,0), nrow=3, ncol=3,
byrow=TRUE)
anxiety <- matrix(c(0,1,0), nrow=3, ncol=3, byrow=TRUE)
stress <- matrix(c(0,0,1), nrow=3, ncol=3, byrow=TRUE)
das <- list(dpress, anxiety, stress)
inItemTraitAdj <- rbind(das[[1]], das[[2]], das[[3]])

inItemTraitAdj <- matrix(adjacency, nrow=3, ncol=3, byrow=TRUE)

#  g3 <- ple.lma(inData, model.type="gpcm", inItemTraitAdj, inTraitAdj)
iterationPlot(g3)

#  n3 <- ple.lma(inData, model.type="nominal", inItemTraitAdj, inTraitAdj)
iterationPlot(n3)
```

---

**Description**

This utility function creates a summary list with five elements. The first is a 'report' that contains a summary of characteristics of the data, the model specification, convergence information, and fit statistics. The second and third elements complete the model specification and are the trait adjacency matrix ('TraitByTrait') and the item x trait adjacency matrix ('ItemByTrait'), respectively. The fourth element, 'estimates', contains a matrix of item parameters, and the fifth element, 'phi.mat' contains association parameter estimates.

**Usage**

```r
lma.summary(model.fit)
```

**Arguments**

- `model.fit` A list object from fitting a model to data

**Value**

- `results` A list with summary information

---

**lma.summury**

*Produces a summary of results*

---

**Description**

This utility function creates a summary list with five elements. The first is a 'report' that contains a summary of characteristics of the data, the model specification, convergence information, and fit statistics. The second and third elements complete the model specification and are the trait adjacency matrix ('TraitByTrait') and the item x trait adjacency matrix ('ItemByTrait'), respectively. The fourth element, 'estimates', contains a matrix of item parameters, and the fifth element, 'phi.mat' contains association parameter estimates.
Examples

```r
#--- 3 items from depression, anxiety and stress scales of
data(dass)
inData <- dass[1:250, c("d1", "d2", "d3", "a1", "a2", "a3", "s1", "s2", "s3")]

#--- log-linear model of independence
ind <- ple.lma(inData, model.type="independence")
noquote(lma.summary(ind))

#--- input for uni-dimensional
inTraitAdj <- matrix(1, nrow=1, ncol=1)
inItemTraitAdj <- matrix(1, nrow=9, ncol=1)

#--- rasch family
r1 <- ple.lma(inData, model.type="rasch", inItemTraitAdj, inTraitAdj)
lma.summary(r1)

#--- Or if specific output is desired
noquote(lma.summary(r1)$report)
lma.summary(r1)$TraitByTrait
lma.summary(r1)$ItemByTrait
lma.summary(r1)$estimates
lma.summary(r1)$phi

#--- generalized partial credit model
g1 <- ple.lma(inData, model.type="gpcm", inItemTraitAdj, inTraitAdj, tol=1e-03)
lma.summary(g1)$report
lma.summary(g1)$estimates
lma.summary(g1)$phi

#--- nominal response model
n1 <- ple.lma(inData, model.type="nominal", inItemTraitAdj, inTraitAdj, tol=1e-03)
noquote(lma.summary(n1))
```

**ple.lma**

*Main function for estimating parameters of LMA models*

**Description**

This function is a wrapper function that checks for errors in the input (i.e., `error.check`), sets up required objects and data (i.e., `set.up`), calls function to fit specified model (either `fit.independence`, `fit.rasch`, `fit.gpcm`, or `fit.nominal`), and outputs an extensive list of details and results. The required input for all models consist of a data frame where elements are consecutive integers (1, 2, ...) indicating the category chosen by each case/individual (rows) for each variable (columns), and the model type. For the LMA models that correspond to item response theory models require an Item x Trait adjacency matrix (`inItemTraitAdj`) and a Trait x Trait adjacency matrix (`inTraitAdj`).
Optional input include the tolerance value (‘tol’) which is used to determine whether the pseudo-likelihood algorithm has converged for a gpcm or nominal model default=1e-6). Additional optional input (‘starting.sv’) is an item by category a matrix of starting scale values for the nominal model or fixed category scores for the gpcm and rasch models. The default category scale values/scores are equally spaced, centered at zero, and the sum of squared values equals 1. The final optional input is a trait x trait (‘starting.phi’) matrix of starting values for the association parameter matrix (default=identity matrix).

Usage

```
ple.lma(
  inData,
  model.type,
  inItemTraitAdj = NULL,
  inTraitAdj = NULL,
  tol = NULL,
  starting.sv = NULL,
  starting.phi = NULL
)
```

Arguments

- **inData**: A person x item data matrix or data frame with elements equal to response options chosen by an individual.
- **model.type**: Model to be fit (nominal, gpcm, rasch, independence) to data.
- **inItemTraitAdj**: An Item x Trait adjacency matrix indicating what trait an item loads on.
- **inTraitAdj**: A Trait x Trait adjacency matrix indicating relationship among traits.
- **tol**: Convergence criterion, default = 1e-6
- **starting.sv**: Starting category scale values/fixed scores
- **starting.phi**: Starting matrix of phi parameters (i.e., conditional covariance matrix)

Value

- **model.type**: The model (nominal, gpcm, rash, or independence) that was fit to data
- **TraitByTrait**: The Trait x Trait adjacency matrix used.
- **ItemByTrait**: The Item x Trait adjacency matrix.
- **item.by.trait**: One dimensional version of ItemByTrait that gives the number of trait.
- **ItemNames**: Names of items in inData
- **PhiNames**: Names of the association parameters (i.e., phi)
- **formula.item**: Formula used to up-date item parameters via item regressions.
- **formula.phi**: Formula used to up-date association parameters via stacked regression.
- **npersons**: Number of persons in data set.
- **nitems**: Number of items.
- **ncat**: Number of categories per item.
nless Number of unique marginal effects & unique scale values.
Maxnphi Number of association parameters estimated.
ntraits Number of traits.
starting.sv Starting scale values for nominal model or fixed scores for rasch or gpcm.
tol Used to determine convergence default= 1e-7
criterion Final value criterion at convergence
item.log Item iteration history plus maximum of the LogLike for each item
phi.log Association parameter iteration history
estimates Item x Parameter matrix where 1st column is max LogLike for each item and remaining
columns are item parameter estimate
Phi.mat Estimated conditional correlation matrix
item.mlogit Output from final mlogit fit to items
phi.mlogit Output form final mlogit fit to stacked data
mlpl.item Max Log(pseudo-likelihood) function from item models (i.e. sum of first column of
estimates)
mlpl.phi Max Log(pseudo-likelihood) function from stacked regression(s).
AIC Akaike information criterion for pseudo-likelihood (smaller is better)
BIC Bayesian information criterion for pseudo-likelihood (smaller is better)

Examples

```r
#--- some data, 3 items from dpression, anxiety and stress scales
# and only 250 cases out of possible 1000
data(dass)
inData <- dass[1:250,c("d1", "d2", "d3", "a1","a2","a3","s1","s2","s3")]

#--- log-linear model of independence
ind <- ple.lma(inData, model.type="independence")

#--- input for uni-dimensional
inTraitAdj <- matrix(1, nrow=1, ncol=1)
inItemTraitAdj <- matrix(1, nrow=9, ncol=1)

#--- rasch family
r1 <- ple.lma(inData, model.type="rasch", inItemTraitAdj, inTraitAdj)

#--- rasch with alternative scores
scores <- matrix(c(0,1,2,3),nrow=9,ncol=4,byrow=TRUE)
r1b <- ple.lma(inData, model.type="rasch", inItemTraitAdj,
inTraitAdj, starting.sv=scores)

#--- generalized partial credit model
g1 <- ple.lma(inData, model.type="gpcm", inItemTraitAdj, inTraitAdj)

#--- gpcm with alternative scores
```
scores <- matrix(c(0,1,2,3),nrow=9,ncol=4,byrow=TRUE)
g1b <- ple.lma(inData, model.type="gpcm", inItemTraitAdj, inTraitAdj, starting.sv=scores)

#--- nominal response model
n1 <- ple.lma(inData, model.type="nominal", inItemTraitAdj, inTraitAdj)

#--- re-run nominal model with input starting values and phi
# and setting stronger convergence criterion.
sv <- n1$estimates[, 6:9]
phi <- n1$Phi.mat
n1b <- ple.lma(inData, model.type="nominal", inItemTraitAdj, inTraitAdj, starting.sv=sv, starting.phi=phi, tol=1e-8)

#--- Multidimensional models
#--- re-define inTraitAdj and inItemTraitAdj for 3 dimensional models
inTraitAdj <- matrix(1, nrow=3, ncol=3)
dpress <- matrix(c(1,0,0), nrow=3, ncol=3, byrow=TRUE)
anxiety <- matrix(c(0,1,0), nrow=3, ncol=3, byrow=TRUE)
stress <- matrix(c(0,0,1), nrow=3, ncol=3, byrow=TRUE)
das <- list(dpress, anxiety, stress)
inItemTraitAdj <- rbind(das[[1]], das[[2]], das[[3]])

#--- 3 dimensional rasch
r3 <- ple.lma(inData, model.type="rasch", inItemTraitAdj, inTraitAdj)

#--- 3 dimensional gpcm
g3 <- ple.lma(inData, model.type="gpcm", inItemTraitAdj, inTraitAdj)

#--- 3 dimensional nominal
n3 <- ple.lma(inData, model.type="nominal", inItemTraitAdj, inTraitAdj)

#--- 2 parameter logistic IRT model fit to responses to
# 10 dichotomous Vocabulary items from from 2018 GSS
# by 1309 respondents
data(vocab)
inItemTraitAdj <- matrix(1, nrow=10, ncol=1)
inTraitAdj <- matrix(1, nrow=1, ncol=1)

#--- rasch irt
rasch <- ple.lma(inData=vocab, model.type="rasch", inItemTraitAdj, inTraitAdj, tol=1e-03)

#--- 2 pl as a gpcm model
g.2pl <- ple.lma(inData=vocab, model.type="gpcm", inItemTraitAdj, inTraitAdj, tol=1e-03)

#--- 2 pl as a nominal model
n.2pl <- ple.lma(inData=vocab, model.type="nominal", inItemTraitAdj, inTraitAdj, tol=1e-03)
reScaleItem

Re-scales the category scale values and Phi after convergence of the nominal model

Description
This auxiliary function only applies to nominal models after estimating the parameters of the model. During estimation that scaling identification constraint is put on conditional variances (i.e., phi_mm) such that they equal 1. This function provide an alternative identification constraint after the algorithm has converged. This function allow a user to tease apart the strength and structure of associations. The alternative scaling identification constraint is to set the sum of category scale values equals 0 and the sum of squares equal to 1. The phi parameters are adjusted accordingly. Only one item per trait should be selected for the identification constraint and this is indicated by the object anchor.

Usage
reScaleItem(model.fit, anchor)

Arguments
model.fit Model object for a nominal model
anchor Indicator of item(s) to place scaling constraint on.

Value
sNu Re-scaled category scale values
sPhi.mat Re-scale phi matrix (conditional covariance matrix)

Examples

```r
#--- 3 items from depression, anxiety, and stress scales
# for 250 cases out of possible 1000
data(dass)
inData <- dass[1:250,c("d1", "d2", "d3", "a1","a2","a3","s1","s2","s3")]
inTraitAdj <- matrix(1, nrow=1, ncol=3)
inItemTraitAdj <- matrix(1, nrow=9, ncol=1)
#--- nominal response model
n1 <- ple.lma(inData, model.type="nominal",inItemTraitAdj,inTraitAdj,tol=1e-03)
anchor <- c(1,0,0,0,0,0,0,0)
reScaleItem(model.fit=n1, anchor)

#--- Multidimensional models
inTraitAdj <- matrix(1, nrow=1, ncol=3)
dpress <- matrix(c(1,0,0), nrow=3, ncol=3, byrow=TRUE)
anxiety <- matrix(c(0,1,0), nrow=3, ncol=3, byrow=TRUE)
```
stress <- matrix(c(0, 0, 1), nrow=3, ncol=3, byrow=TRUE)
das <- list(dpress, anxiety, stress)
inItemTraitAdj <- rbind(das[[1]], das[[2]], das[[3]])

n3 <- ple.lma(inData, model.type="nominal", inItemTraitAdj, inTraitAdj, tol=1e-03)
anchor <- c(1, 0, 0, 0, 1, 0, 1, 0, 0)
reScaleItem(model.fit=n3, anchor)

---

Scale

*Imposes scaling constraint to identify parameters of the LMA (nominal) model*

**Description**

Scaling is internal to the function `fit.nominal`, which corresponds to the nominal item response theory model. It imposes the required scaling identification constraint by transforming the conditional covariance matrix 'Phi.mat' to a conditional correlation matrix (i.e., set phi_mm=1 for all m). The inverse transformation is applied to the current category scale value estimates and these are put back into the Master data frame so that data are ready for the next iteration of the algorithm.

**Usage**

```r
Scale(
  Master,
  item.log,
  Phi.mat,
  PersonByItem,
  npersons,
  nitems,
  ncat,
  nless,
  ntraits,
  item.by.trait
)
```

**Arguments**

- **Master**: Current Master data frame.
- **item.log**: Iteration history of LogLike, lambda, and item parameters
- **Phi.mat**: Current phi matrix
- **PersonByItem**: inData
- **npersons**: Number of persons
- **nitems**: Number of items
ScaleGPCM

ncat  Number of categories
nless  Number of unique nus (ncat-1)
ntraits Number of (latent) dimensions
item.by.trait Indicates the trait an item load on.

Value

Master Master frame with re-scaled scale values
Phi.mat Re-scaled matrix of association parameters

ScaleGPCM Imposes scaling constraint to identify parameters of LMA (GPCM)

Description

Scaling is internal to the function 'fit.gpcm', which fits the GPCM version of the LMA. It imposes the required scaling identification constraint by transforming the conditional covariance matrix 'Phi.mat' to a conditional correlation matrix. The inverse transformation is applied to the current estimates of the slope or 'a' parameters. Category scale values are recomputed using the re-scale slopes (i.e., nu= a*x) and these are put back into the Master data set so that data are ready for the next iteration of the algorithm.

Usage

ScaleGPCM(
  Master,
  item.log,
  Phi.mat,
  PersonByItem,
  npersons,
  nitems,
  ncat,
  nless,
  ntraits,
  starting.sv,
  item.by.trait
)

Arguments

Master Master/main data set
item.log Iteration history array, last row are current parameters
Phi.mat Current phi matrix
PersonByItem inData (response patterns)
npersons Number of persons
scalingPlot

nitems     Number of items
ncat       Number of categories
nless      Number of unique lambdas (ncat-1)
ntraits    Number of latent traits
starting.sv Matrix of fixed category scores (nitems x ncat)
item.by.trait Object that indicates which trait item loads on

Value

Master Master data set with re-scaled scale values
Phi.mat Re-scaled matrix of association parameters

Description

This function plots the estimated item scale values (i.e., \( nus \)) by integers to see shape of scaling of the categories. A linear regression is overlaid in the plot to help assess linearity. The dashed red line overlaid in the plot is the linear regression line of the scale values on integers.

Usage

scalingPlot(model.fit)

Arguments

model.fit      Output from a nominal model

Value

plots of estimated scale values by integers

Examples

```r
#--- some data, 2 items from depression, anxiety and stress scales
#   for 250 cases out of possible 1000
data(dass)
inData <- dass[1:250, c("d1", "d2", "a1", "a2", "s1", "s2")]
inTraitAdj <- matrix(1, nrow=1, ncol=1)
inItemTraitAdj <- matrix(1, nrow=6, ncol=1)
n1 <- ple.lma(inData, model.type="nominal", inItemTraitAdj, inTraitAdj, tol=1e-03)
scalingPlot(n1)
```
set.up

Sets up the data based on input data and model specifications

**Description**

This function sets up the data and sets constants that are essentially the same for all models. This is used within the main wrapper function ‘ple.lma’, but can also be run independently. If a user wants to run the functions ‘fit.independence’, ‘fit.rasch’, ‘fit.gpcm’, or ‘fit.nominal’, the set up function should be run prior to using these functions to create required input. Such an approach can speed up replication studies because ‘set.up’ would only need to be run once and the response vector (i.e., named ‘y’) in the Master data frame be replaced by a new one.

**Usage**

```r
set.up(
  inData,  
  model.type,  
  inTraitAdj = NULL,  
  inItemTraitAdj = NULL,  
  tol = NULL,  
  starting.sv = NULL,  
  starting.phi = NULL  
)
```

**Arguments**

- `inData`: A person x item Data frame with response patterns
- `model.type`: Type of model to be fit
- `inTraitAdj`: Trait x Trait adjacency matrix (NULL for independence)
- `inItemTraitAdj`: Item x Trait adjacency matrix (NULL for independence)
- `tol`: Tolerance for determining convergence (default: 1e-06)
- `starting.sv`: Starting category scale values/fixed scores (default: sum equal to zero and sum of squares equal to 1)
- `starting.phi`: optional: Starting phi matrix (default: identity matrix)

**Value**

- `PersonByItem inData` (rows are response patterns)
- `TraitByTrait Trait x Trait adjacency matrix`
- `ItemByTrait Item x Trait adjacency matrix`
- `item.by.trait` Need for re-scaling phi.mat
- `starting.sv` An item by number of category matrix with starting values for scale values for nominal model and fixed category scores for gpcm and rasch models
- `ItemNames` Names of items in inData and PersonByItem
LambdaName Short list of lambda names needed for item regressions
NuName Short list of nu names needed for item regressions
LambdaNames Long list of lambdas using in Master data set
NuNames Long list of nu using in Master data set
PhiNames Names of the unique phi parameters
npersons Number of individual or persons in data
nitems Number of items
ncat Number of categories
nless Number of unique lambdas and unique nus
ntraits Number of traits
Maxnphi Number of phis to estimate
Nstack Length of master data set
pq.mat An array used to computed (weighted) rest-scores
Phi.mat A number of traits x number of traits Phi matrix (defual: the identity matrix)
Master Master data set formated for input to to mlogit
tol Tolerance for deteriming convergence

Examples

data(dass)
inData <- dass[1:250,c("d1", "d2", "d3", "a1","a2","a3","s1","s2","s3")]

#--- to set data up for model of independence
ind.setup <- set.up(inData, model.type="independence")

#--- for model specification for uni-dimensional models
inTraitAdj <- matrix(1, nrow=1, ncol=1)
inItemTraitAdj <- matrix(1, nrow=9, ncol=1)
i.setup <- set.up(inData, model.type="independence")

r.setup <- set.up(inData, model.type='rasch', inTraitAdj, inItemTraitAdj)
g.setup <- set.up(inData, model.type='gpcm', inTraitAdj, inItemTraitAdj)
n.setup <- set.up(inData, model.type='nominal', inTraitAdj, inItemTraitAdj)
## StackData

Prepares data for up-dating association parameters of a multidimensional nominal LMA

### Description

Prepares data frame for input to `mnlogit` for the stacked regression to obtain association parameters of the multidimensional LMA models corresponding to the Nominal item response model. This function is called from `fit.nominal`. It generally would not run outside of either `fit.nominal` or `ple.lma`.

### Usage

```r
StackData(
  Master,  
  item.log,  
  phi.log,  
  pq.mat,  
  npersons,  
  nitems,  
  ncat,  
  nless,  
  ntraits,  
  Maxnphi,  
  PhiNames,  
  LambdaNames
)
```

### Arguments

- **Master**: Master data frame from which stacked data is created
- **item.log**: Last row contains current scale values (item.history)
- **phi.log**: Last row contains current estimates of phi
- **pq.mat**: Summing array to get rest scores and totals
- **npersons**: Number of persons
- **nitems**: Number of items
- **ncat**: Number of categories per item
- **nless**: Number of categories less 1 (i.e., unique lambdas and unique nus)
- **ntraits**: Number of latent traits
- **Maxnphi**: Number of phis to be estimated
- **PhiNames**: Names of the Phi parameters
- **LambdaNames**: Names of lambdas that correspond to those in Master
StackDataGPCM

Value

Phi.mat Up-dated matrix of phi parameters
phi.log History of iterations for LogLike, Lambda and phi parameters

StackDataGPCM Prepares data for up-dating association parameters of LMA (GPCM) model

Description

Prepares data frame for input to ‘mnlogit’ for the stacked regression to obtain association parameters of the multidimensional LMA models corresponding to the GPCM. This function is called from ‘fit.gpcm’. It generally would not run outside of ‘fit.nominal’ or ‘ple.lma’.  

Usage

StackDataGPCM(
  Master,
  item.log,
  starting.sv,
  npersons,
  nitems,
  ncat,
  nless,
  Maxnphi,
  pq.mat,
  LambdaNames,
  PhiNames
)

Arguments

Master Current master data frame
item.log Current history of iterations from which current a parameters are drawn
starting.sv Fixed category scores
npersons Number of persons
nitems Number of items
ncat Number of categories
nless Number of unique lambdas
Maxnphi Number of estimated phi parameters
pq.mat Array needed for rest-total scores
LambdaNames Names of lambdas in Master/Stacked data set
PhiNames Names of phi parameters
theta.estimates

Computes estimates of theta (values on latent trait(s)) for all LMA models.

Description

The final estimates of the item scale values and the conditional covariance matrix (i.e., Phi.mat) are used to compute values on latent traits for each individual or case. The estimated thetas are the (conditional) mean values of response patterns. The correlations between the estimated thetas equal the marginal correlations.

Usage

theta.estimates(inData, model.fit)

Arguments

- inData: Matrix of response patterns
- model.fit: Object containing output from running ple.lma

Value

theta.est A person by trait matrix of values on the latent traits

Examples

```r
data(dass)
inData <- dass[1:250,c("d1", "d2", "d3", "a1","a2","a3","s1","s2","s3")]
inTraitAdj <- matrix(1, nrow=1, ncol=1)
inItemTraitAdj <- matrix(1, nrow=9, ncol=1)
r1 <- ple.lma(inData, model.type="rasch", inItemTraitAdj, inTraitAdj)
theta.r1 <- theta.estimates(inData, r1)

g1 <- ple.lma(inData, model.type="gpcm", inItemTraitAdj, inTraitAdj)
theta.g1 <- theta.estimates(inData, g1)

n1 <- ple.lma(inData, model.type="nominal", inItemTraitAdj, inTraitAdj)
theta.n1 <- theta.estimates(inData, n1)
```
vocab

**Description**

These data are responses to 10 vocabulary items from the GSS collected in 2018 and retrieved July 2019 from https://gss.norc.org. These data are provided as an example of binary items and how the package can fit two parameter logistic models as either a GPCM or nominal model. Both models should give the same results. There are 10 words and responses to them were either correct or incorrect. There are 1309 respondents in the data who gave answers to all items. The specific words used are not released due to test security reasons. The instructions given to answering these items are as follows: "We would like to know something about how people go about guessing words they do not know. On this card are listed some words—you may know some of them, and you may not know quite a few of them. On each line the first word is in capital letters – like BEAST. Then there are five other words. Tell me the number of the word that comes closest to the meaning of the word in capital letters. For example, if the word in capital letters is BEAST, you would say "4" since "animal" comes closer to BEAST than any of the other words. If you wish, I will read the words to you. These words are difficult for almost everyone—just give me your best guess if you are not sure of the answer. CIRCLE ONE CODE NUMBER FOR EACH ITEM BELOW."

**Usage**

vocab

**Format**

A data frame with 1309 rows and 10 columns (items):

- wordA  word A
- wordB  word B
- wordC  word C
- wordD  word D
- wordE  word E
- wordF  word F
- wordG  word G
- wordH  word H
- wordI  word I
- wordJ  word J

**Source**

https://gss.norc.org
Examples

data(vocab)
head(vocab)
Index

* datasets
  dass, 5
  vocab, 37

cornergence.stats, 2
convergenceGPCM, 4
dass, 5
error.check, 7
fit.gpcm, 8
fit.independence, 10
fit.nominal, 11
fit.rasch, 13
FitStack, 15
fitStackGPCM, 16

item.gpcm, 17
ItemData, 19
ItemGPCM.data, 20
ItemLoop, 21
iterationPlot, 22

lma.summary, 23

ple.lma, 24
reScaleItem, 28
Scale, 29
ScaleGPCM, 30
scalingPlot, 31
set.up, 32
StackData, 34
StackDataGPCM, 35

theta.estimates, 36
vocab, 37