Package 'psychomix'

March 17, 2019

Version 1.1-6

Date 2019-03-17

Title Psychometric Mixture Models

Description Psychometric mixture models based on 'flexmix' infrastructure. At the moment Rasch mixture models
with different parameterizations of the score distribution (saturated vs. mean/variance specification),
Bradley-Terry mixture models, and MPT mixture models are implemented. These mixture models can be estimated
with or without concomitant variables. See vignette('raschmix', package = 'psychomix') for details on the
Rasch mixture models.

Depends R (>= 2.10.0), flexmix (>= 2.3-7), psychotools (>= 0.4-2)

Imports graphics, methods, stats, lattice, Formula (>= 1.1-0),
modeltools

Suggests effects, lmtest, mRm, nnet, numDeriv

License GPL-2 | GPL-3

NeedsCompilation no

Author Hannah Frick [aut, cre] (<https://orcid.org/0000-0002-6049-5258>),
Friedrich Leisch [aut],
Carolin Strobl [aut],
Florian Wickelmaier [aut],
Achim Zeileis [aut] (<https://orcid.org/0000-0003-0918-3766>)

Maintainer Hannah Frick <hannah.frick@gmail.com>

Repository CRAN

Date/Publication 2019-03-17 17:40:03 UTC

R topics documented:

`btmix` *Finite Mixtures of Bradley-Terry Models*

Description

Fit finite mixtures of Bradley-Terry models for paired comparisons data via maximum likelihood with the EM algorithm.

Usage

```
btmix(formula, data, k, subset, weights,
         nrep = 3, cluster = NULL, control = NULL,
         verbose = TRUE, drop = TRUE, unique = FALSE, which = NULL,
         type = c("loglin", "logit"), ref = NULL, undecided = NULL,
         position = NULL, ...)

FLXMCbtreg(formula = . ~ ., type = c("loglin", "logit"), ref = NULL,
  undecided = NULL, position = NULL, ...)
```

Arguments

`formula`	Symbolic description of the model (of type `y ~ 1` or `y ~ x`).
`data, subset`	Arguments controlling formula processing.
`k`	A vector of integers indicating the number of components of the finite mixture; passed in turn to the `k` argument of `stepFlexmix`.
`weights`	An optional vector of weights to be used in the fitting process; passed in turn to the `weights` argument of `flexmix`.
`nrep`	Number of runs of the EM algorithm.
`cluster`	Either a matrix with `k` columns of initial cluster membership probabilities for each observation; or a factor or integer vector with the initial cluster assignments of observations at the start of the EM algorithm. Default is random assignment into `k` clusters.
`control`	An object of class `"FLXcontrol"` or a named list; controls the EM algorithm and passed in turn to the `control` argument of `flexmix`.

verbose	A logical; if TRUE progress information is shown for different starts of the EM algorithm.
drop	A logical; if TRUE and k is of length 1, then a single raschmix object is returned instead of a stepRaschmix object.
unique	A logical; if TRUE, then unique() is called on the result; for details see stepFlexmix.
which	number of model to get if k is a vector of integers longer than one. If character, interpreted as number of components or name of an information criterion.
type	Character. Should an auxiliary log-linear Poisson model or logistic binomial be employed for estimation? The latter is only available if not undecided effects are estimated.
ref	Character or numeric. Which object parameter should be the reference category, i.e., constrained to zero?
undecided	Logical. Should an undecided parameter be estimated?
position	Logical. Should a position effect be estimated?
...	Currently not used.

Details

Internally stepFlexmix is called with suitable arguments to fit the finite mixture model with the EM algorithm.

FLXMCbtreg is the flexmix-driver for Bradley-Terry mixture models.

The interface is designed along the same lines as raschmix which is introduced in detail in Frick et al. (2012). However, the btmix function has not yet been fully tested and may change in future versions.

Value

Either an object of class "btmix" containing the best model with respect to the log-likelihood (if k is a scalar) or the one selected according to which (if specified and k is a vector of integers longer than 1) or an object of class "stepBTmix" (if which is not specified and k is a vector of integers longer than 1).

References

Bradley, R.A., and Terry, M.E. (1952). Rank Analysis of Incomplete Block Designs. I. The Method of Paired Comparisons. *Biometrika*, **39**(3/4), 324–345.

Dörr, M. (2011). Bradley Terry Mixture Models: Theory, Implementation in R and Validation. Diploma Thesis, Ludwig-Maximilians-Universität München.

Frick, H., Strobl, C., Leisch, F., and Zeileis, A. (2012). Flexible Rasch Mixture Models with Package psychomix. *Journal of Statistical Software*, **48**(7), 1–25. http://www.jstatsoft.org/v48/i07/.

Grün, B., and Leisch, F. (2008). FlexMix Version 2: Finite Mixtures with Concomitant Variables and Varying and Constant Parameters. *Journal of Statistical Software*, **28**(4), 1–35. http://www.jstatsoft.org/v28/i04/.

Leisch, F. (2004). FlexMix: A General Framework for Finite Mixture Models and Latent Class Regression in R. *Journal of Statistical Software*, **11**(8), 1–18. http://www.jstatsoft.org/v11/i08/.

See Also

flexmix, stepFlexmix

Examples

```
## Data
data("GermanParties2009", package = "psychotools")

## omit single observation with education = 1
gp <- subset(GermanParties2009, education != "1")
gp$education <- factor(gp$education)

## Bradley-Terry mixture models
set.seed(1)
## fit models for k = 1, ..., 4 with concomitant variables
cm <- btmix(preference ~ gender + education + age + crisis,
  data = gp, k = 1:4, nrep = 3)

## inspect results
plot(cm)

## select model
cm4 <- getModel(cm, which = "4")

## inspect mixture and effects
library("lattice")
xyplot(cm4)
effectsplot(cm4)
effectsplot(cm4, selection = "education")

## vis effects package directly
if(require("effects")) {
eff4 <- allEffects(cm4)
plot(eff4)
}
```

`btmix-class` *Class "btmix"*

Description

A fitted btmix model.

Slots

`model`: A `FLXMC` object for a Bradley-Terry mixture model

`prior`: Numeric vector with prior probabilities of classes.

`posterior`: Named list with elements `scaled` and `unscaled`, both matrices with one row per observation and one column per class.

`iter`: Number of EM iterations.

`k`: Number of classes after EM.

`k0`: Number of classes at start of EM.

`cluster`: Class assignments of observations.

`size`: Class sizes.

`logLik`: Log-likelihood at EM convergence.

`df`: Total number of parameters of the model.

`components`: List describing the fitted components using `FLXcomponent` objects.

`formula`: Object of class `"formula"`.

`control`: Object of class `"FLXcontrol"`.

`call`: The function call used to create the object.

`group`: Object of class `"factor"`.

`converged`: Logical, `TRUE` if EM algorithm converged.

`concomitant`: Object of class `"FLXP"`.

`weights`: Optional weights of the observations.

`flx.call`: Internal call to `stepFlexmix`

`nobs`: Number of observations.

`labels`: Labels of objects compared.

`mscale`: Measurement scale of paired comparisons data.

`undecided`: logical. Should an undecided parameter be estimated?

`ref`: character or numeric. Which object parameter should be the reference category, i.e., constrained to zero?

`type`: character. Should an auxiliary log-linear Poisson model or logistic binomial be employed for estimation? The latter is only available if not undecided effects are estimated.

Extends

Class `flexmix`, directly.

Accessor Functions

The following functions should be used for accessing the corresponding slots:

`clusters`: Cluster assignments of observations.

`posterior`: A matrix of posterior probabilities for each observation.

btmix-methods *Methods for btmix Objects*

Description

Methods for btmix-class objects.

Usage

```
## S4 method for signature 'btmix'
worth(object, component = NULL)
```

Arguments

object	An object of class "btmix".
component	Indicates which components are returned. Default is all components.

Details

worth returns the worth parameters from the Bradley-Terry model.

effectsplot *Effects Displays for Concomitant Variables in Finite Mixture Models*

Description

Generic function for visualizing the effects of concomitant variables in finite mixture models.

Usage

```
effectsplot(object, ...)
```

Arguments

object	Fitted model object.
...	Arguments passed to plot.eff, plot.efflist, or plot.effpoly.

Details

effectsplot is set up to be both an S3 and S4 generic. The idea is that it provides the glue needed to extract the concomitant part from a mixture model: First, the concomitant model is refitted as a multinom object or glm object (in case of a mixture with two components). Second, effect or allEffects from the **effects** package is called to extract the effects of the concomitants. Third, the corresponding plot methods from the **effects** package create the display.

Currently, this is implemented for raschmix, btmix, and mptmix objects. The interface is not yet fully tested and may change in future versions.

References

Fox, J. (2003). Effect Displays in R for Generalised Linear Models. *Journal of Statistical Software*, **8**(15), 1–27. http://www.jstatsoft.org/v08/i15/

Fox, J., and Hong, J. (2009). Effect Displays in R for Multinomial and Proportional-Odds Logit Models: Extensions to the effects Package. *Journal of Statistical Software*, **32**(1), 1–24. http://www.jstatsoft.org/v32/i01/

See Also

effect, allEffects, multinom, glm

Examples

```
## data on party preferences in Germany
## (omit single observation with education = 1)
data("GermanParties2009", package = "psychotools")
gp <- subset(GermanParties2009, education != "1")
gp$education <- factor(gp$education)

## fit Bradley-Terry mixture, see ?btmix for more details
## and a fully-worked example
set.seed(2)
cm4 <- btmix(preference ~ gender + education + age + crisis, data = gp, k = 4, nrep = 1)

## inspect mixture and effects
library("lattice")
xyplot(cm4)
effectsplot(cm4)

## vis effects package directly
if(require("effects")) {
eff4 <- allEffects(cm4)
plot(eff4)
}
```

mptmix *Finite Mixtures of Multinomial Processing Tree Models*

Description

Fit finite mixtures of multinomial processing tree (MPT) models via maximum likelihood with the EM algorithm.

Usage

```
mptmix(formula, data, k, subset, weights,
          nrep = 3, cluster = NULL, control = NULL,
          verbose = TRUE, drop = TRUE, unique = FALSE, which = NULL,
          spec, treeid = NULL,
          optimargs = list(control = list(reltol =
            .Machine$double.eps^(1/1.2), maxit = 1000)), ...)

FLXMCmpt(formula = . ~ ., spec = NULL, treeid = NULL, optimargs = NULL, ...)
```

Arguments

formula	Symbolic description of the model (of type `y ~ 1` or `y ~ x`).
data, subset	Arguments controlling formula processing.
k	A vector of integers indicating the number of components of the finite mixture; passed in turn to the k argument of `stepFlexmix`.
weights	An optional vector of weights to be used in the fitting process; passed in turn to the `weights` argument of `flexmix`.
nrep	Number of runs of the EM algorithm.
cluster	Either a matrix with k columns of initial cluster membership probabilities for each observation; or a factor or integer vector with the initial cluster assignments of observations at the start of the EM algorithm. Default is random assignment into k clusters.
control	An object of class `"FLXcontrol"` or a named list; controls the EM algorithm and passed in turn to the `control` argument of `flexmix`.
verbose	A logical; if `TRUE` progress information is shown for different starts of the EM algorithm.
drop	A logical; if `TRUE` and k is of length 1, then a single `mptmix` object is returned instead of a `stepMPTmix` object.
unique	A logical; if `TRUE`, then `unique()` is called on the result; for details see `stepFlexmix`.
which	number of model to get if k is a vector of integers longer than one. If character, interpreted as number of components or name of an information criterion.
spec, treeid, optimargs	arguments for the MPT model passed on to `mptmodel`.
...	Currently not used.

Details

Internally stepFlexmix is called with suitable arguments to fit the finite mixture model with the EM algorithm.

FLXMCmpt is the flexmix driver for MPT mixture models.

The interface is designed along the same lines as raschmix which is introduced in detail in Frick et al. (2012). However, the mptmix function has not yet been fully tested and may change in future versions.

The latent-class MPT model (Klauer, 2006) is equivalent to an MPT mixture model without concomitant variables.

MPT models are specified using the mptspec function. See the documentation in the **mpt** package for details.

Value

Either an object of class "mptmix" containing the best model with respect to the log-likelihood (if k is a scalar) or the one selected according to which (if specified and k is a vector of integers longer than 1) or an object of class "stepMPTmix" (if which is not specified and k is a vector of integers longer than 1).

References

Frick, H., Strobl, C., Leisch, F., and Zeileis, A. (2012). Flexible Rasch Mixture Models with Package psychomix. *Journal of Statistical Software*, **48**(7), 1–25. http://www.jstatsoft.org/v48/i07/

Klauer, K.C. (2006). Hierarchical Multinomial Processing Tree Models: A Latent-Class Approach. *Psychometrika*, **71**, 7–31. doi: 10.1007/s1133600411883

See Also

flexmix, stepFlexmix

Examples

```
## Data
data("PairClustering", package = "psychotools")
pc <- reshape(PairClustering, timevar = "trial", idvar = "ID",
              direction = "wide")

## Latent-class MPT model (Klauer, 2006)
set.seed(1)
m <- mptmix(as.matrix(pc[-1]) ~ 1, data = pc, k = 1:3,
            spec = mptspec("SR", .replicates = 2))
m1 <- getModel(m, which = "BIC")

## Inspect results
summary(m1)
parameters(m1)
```

```
plot(m1)

library(lattice)
xyplot(m1)
```

`mptmix-class` *Class "mptmix"*

Description

A fitted `mptmix` model.

Slots

`model`: A `FLXMC` object for an MPT mixture model

`prior`: Numeric vector with prior probabilities of classes.

`posterior`: Named list with elements `scaled` and `unscaled`, both matrices with one row per observation and one column per class.

`iter`: Number of EM iterations.

`k`: Number of classes after EM.

`k0`: Number of classes at start of EM.

`cluster`: Class assignments of observations.

`size`: Class sizes.

`logLik`: Log-likelihood at EM convergence.

`df`: Total number of parameters of the model.

`components`: List describing the fitted components using `FLXcomponent` objects.

`formula`: Object of class `"formula"`.

`control`: Object of class `"FLXcontrol"`.

`call`: The function call used to create the object.

`group`: Object of class `"factor"`.

`converged`: Logical, `TRUE` if EM algorithm converged.

`concomitant`: Object of class `"FLXP"`.

`weights`: Optional weights of the observations.

`flx.call`: Internal call to `stepFlexmix`

`nobs`: Number of observations.

Extends

Class `flexmix`, directly.

Accessor Functions

The following functions should be used for accessing the corresponding slots:

`clusters`: Cluster assignments of observations.

`posterior`: A matrix of posterior probabilities for each observation.

`raschmix` *Finite Mixtures of Rasch Models*

Description

Fit finite mixtures of Rasch models for item response data via conditional maximum likelihood with the EM algorithm.

Usage

```
raschmix(formula, data, k, subset, weights, scores = c("saturated", "meanvar"),
  restricted = FALSE, nrep = 3, cluster = NULL, control = NULL, verbose = TRUE,
  drop = TRUE, unique = FALSE, which = NULL, reltol = 1e-10, deriv = "sum",
  hessian = FALSE, restart = TRUE, model = NULL, gradtol = reltol, ...)

FLXMCrasch(formula = . ~ ., scores = "saturated", delta = NULL, nonExtremeProb = 1,
  ref = 1, reltol = 1e-10, deriv = "sum", hessian = FALSE,
  restart = TRUE, ...)
```

Arguments

`formula`	Symbolic description of the model (of type `y ~ 1` or `y ~ x`).
`data, subset`	Arguments controlling formula processing.
`k`	A vector of integers indicating the number of components of the finite mixture; passed in turn to the `k` argument of `stepFlexmix`.
`weights`	An optional vector of weights to be used in the fitting process; passed in turn to the `weights` argument of `flexmix`.
`scores`	Indicates which model should be fitted for the score probabilities: either a `saturated` model with a separate parameter for each score probability, or, for `meanvar`, a multinomial logit model with a location and a scale parameter.
`restricted`	Logical. Should the score distributions be restricted to being equal across components? See Frick et al. (2015) for details.
`nrep`	Number of runs for the starting values for the EM algorithm (if `cluster = "mrm"`) or number of runs of the EM algorithm itself.
`cluster`	Either a matrix with `k` columns of initial cluster membership probabilities for each observation; or a factor or integer vector with the initial cluster assignments of observations at the start of the EM algorithm. If `cluster = "mrm"`, the `mrm` function is used to generate starting values. Default is random assignment into `k` clusters.

control	An object of class "FLXcontrol" or a named list; controls the EM algorithm and passed in turn to the control argument of flexmix.
verbose	A logical; if TRUE progress information is shown for different starts of the EM algorithm.
drop	A logical; if TRUE and k is of length 1, then a single raschmix object is returned instead of a stepRaschmix object.
unique	A logical; if TRUE, then unique() is called on the result; for details see stepFlexmix.
which	number of model to get if k is a vector of integers longer than one. If character, interpreted as number of components or name of an information criterion.
nonExtremeProb	A numeric giving the probability of scoring either none or all items.
ref	Reference category for the saturated score model.
reltol, gradtol, deriv, hessian	Control parameters passed to RaschModel.fit for the M-step. The gradtol argument is deprecated and reltol should be used instead.
restart	Logical. Should the estimation of the item parameters be restarted in each iteration? If FALSE, the estimates from the previous M-step are used as starting values.
delta	Parameters of score model. If NULL, a score model is estimated.
model	An object inheriting from class "FLXM" for the flexmix-driver, as typically produced by FLXMCrasch. By default FLXMCrasch is called automatically with the parameters computed from raschmix.
...	Currently not used.

Details

raschmix is intended as a convenience interface to the stepFlexmix function from the **flexmix** package (Leisch 2004, Grün and Leisch 2008). The formula argument of raschmix is used to describe the model in terms of both items and concomitant variables, if any. On the left-hand side of the formula the item are specified, either as a matrix y or as single items `y1 + y2 + y3 + ...`. On the right-hand side, the concomitant variables are specified. If no concomitant variables are to be included in the model, the right-hand side of the is just written as `~ 1`. See Frick et al. (2012) for a detailed introduction.

raschmix processes this model description and calls stepFlexmix with the suitable driver FLXMCrasch. Usually, the driver does not need to be called by itself, but it is of course also possible to call stepFlexmix directly with this driver to fit Rasch mixture models.

The Rasch mixture model with saturated score distribution as proposed by Rost (1990) is also known as "Mixed Rasch Model". The mean-variance score distribution was suggested by Rost and von Davier (1995). A more recent extension is the restricted score specification by Frick et al. (2015) who also provide an extensive comparison using Monte Carlo studies.

Value

Either an object of class "raschmix" containing the best model with respect to the log-likelihood (if k is a scalar) or the one selected according to which (if specified and k is a vector of integers longer than 1) or an object of class "stepRaschmix" (if which is not specified and k is a vector of integers longer than 1).

References

Frick, H., Strobl, C., Leisch, F., and Zeileis, A. (2012). Flexible Rasch Mixture Models with Package psychomix. *Journal of Statistical Software*, **48**(7), 1–25. http://www.jstatsoft.org/v48/i07/.

Frick, H., Strobl, C., and Zeileis, A. (2015). Rasch Mixture Models for DIF Detection: A Comparison of Old and New Score Specifications. *Educational and Psychological Measurement*, **75**(2), 208–234. doi:10.1177/0013164414536183.

Grün, B., and Leisch, F. (2008). FlexMix Version 2: Finite Mixtures with Concomitant Variables and Varying and Constant Parameters. *Journal of Statistical Software*, **28**(4), 1–35. http://www.jstatsoft.org/v28/i04/.

Leisch, F. (2004). FlexMix: A General Framework for Finite Mixture Models and Latent Class Regression in R. *Journal of Statistical Software*, **11**(8), 1–18. http://www.jstatsoft.org/v11/i08/.

Rost, J. (1990). Rasch Models in Latent Classes: An Integration of Two Approaches to Item Analysis. *Applied Psychological Measurement*, **14**(3), 271–282.

Rost, J., and von Davier, M. (1995). Mixture Distribution Rasch Models. In Fischer, G.H., and Molenaar, I.W. (eds.), *Rasch Models: Foundations, Recent Developments, and Applications*, chapter 14, pp. 257–268. Springer-Verlag, New York.

See Also

flexmix, stepFlexmix, simRaschmix

Examples

```
##########
## Data ##
##########

## simulate response from Rost's scenario 2 (with 2 latent classes)
set.seed(1)
r2 <- simRaschmix(design = "rost2")

## plus informative and non-informative concomitants
d <- data.frame(
  x1 = rbinom(nrow(r2), prob = c(0.4, 0.6)[attr(r2, "cluster")], size = 1),
  x2 = rnorm(nrow(r2))
)
d$resp <- r2

## fit model with 2 latent classes (here the number is known a priori)
m <- raschmix(r2, k = 2, scores = "saturated")
summary(m)

## see below for examples which do not use this a priori information
## (these take a little longer to compute)
```

```
#####################################################
## Rasch mixture model with saturated score model ##
## (Rost, 1990)                                    ##
#####################################################

## fit models for k = 1, 2, 3
m1 <- raschmix(r2, k = 1:3, score = "saturated")
## equivalently: m1 <- raschmix(resp ~ 1, data = d, k = 1:3, score = "saturated")

## inspect results
m1
plot(m1)

## select best BIC model
BIC(m1)
m1b <- getModel(m1, which = "BIC")
summary(m1b)

## compare estimated with true item parameters
parameters(m1b, "item") ##  9 items, item_1 = 0
worth(m1b)              ## 10 items, sum = 0
attr(r2, "difficulty")

## graphical comparison
plot(m1b, pos = "top")
for(i in 1:2) lines(attr(r2, "difficulty")[,i], lty = 2, type = "b")

## extract estimated raw score probabilities
## (approximately equal across components and roughly uniform)
scoreProbs(m1b)

## note: parameters() and worth() take "component" argument
parameters(m1b, "item",  component = 2)
parameters(m1b, "score", component = 1)
worth(m1b, component = 2:1)

## inspect posterior probabilities
histogram(m1b)
head(posterior(m1b)) ## for first observations only

## compare resulting clusters with true groups
table(model = clusters(m1b), true = attr(r2, "cluster"))


## optionally: leverage mRm package for faster computation of
## starting values
## Not run:
library("mRm")
## fit 2-component model
m1b_mrm <- raschmix(r2, k = 2, score = "saturated", cluster = "mrm")
## essentially identical to previous solution
table(clusters(m1b), clusters(m1b_mrm))
worth(m1b) - worth(m1b_mrm)
```

```
## End(Not run)


#####################################################################
##  Rasch mixture model with mean/variance score distribution ##
## (Rost & von Davier, 1995)                                       ##
#####################################################################

## more parsimonious parameterization,
## fit multinomial logit model for score probabilities

## fit models and select best BIC
m2 <- raschmix(r2, k = 1:3, score = "meanvar")
plot(m2)
m2b <- getModel(m2, which = "BIC")

## compare number of estimated parameters
dim(parameters(m2b))
dim(parameters(m1b))

## graphical comparison with true parameters
plot(m2b, pos = "top")
for(i in 1:2) lines(attr(r2, "difficulty")[,i], lty = 2, type = "b")

## results from non-parametric and parametric specification
## essentially identical
max(abs(worth(m1b) - worth(m2b, component = 2:1)))



############################
## Concomitant variables ##
############################

## employ concomitant variables (x1 = informative, x2 = not)
## Not run:
## fit model
cm2 <- raschmix(resp ~ x1 + x2, data = d, k = 2:3, score = "meanvar")

## BIC selection
rbind(m2 = BIC(m2), cm2 = c(NA, BIC(cm2)))
cm2b <- getModel(cm2, which = "BIC")

## concomitant coefficients
parameters(cm2b, which = "concomitant")

## End(Not run)


##########
## Misc ##
##########
```

```
## note: number of clusters can either be chosen directly
## or directly selected via AIC (or BIC, ICL)
## Not run:
raschmix(r2, k = 2)
raschmix(r2, k = 1:3, which = "AIC")

## End(Not run)
```

---

raschmix-class *Class "raschmix"*

Description

A fitted raschmix model.

Slots

model: A FLXMC object for a Rasch mixture model

prior: Numeric vector with prior probabilities of classes.

posterior: Named list with elements scaled and unscaled, both matrices with one row per observation and one column per class.

iter: Number of EM iterations.

k: Number of classes after EM.

k0: Number of classes at start of EM.

cluster: Class assignments of observations.

size: Class sizes.

logLik: Log-likelihood at EM convergence.

df: Total number of parameters of the model.

components: List describing the fitted components using FLXcomponent objects.

formula: Object of class "formula".

control: Object of class "FLXcontrol".

call: The function call used to create the object.

group: Object of class "factor".

converged: Logical, TRUE if EM algorithm converged.

concomitant: Object of class "FLXP"..

weights: Optional weights of the observations.

scores: Type of score model employed.

restricted: Logical. Is the score model equal across components?

deriv: Type of derivatives used for computing gradient and Hessian matrix. Analytical with sum algorithm ("sum"), analytical with difference algorithm ("diff", faster but numerically unstable), or numerical.

`extremeScoreProbs`: Estimated probability of solving either all or no items.

`rawScoresData`: Table of raw scores from the data.

`flx.call`: Internal call to `stepFlexmix`

`nobs`: Number of observations without missing values, excluding observations with an extreme score.

`identified.items`: Factor indicating which items are identified.

Extends

Class `flexmix`, directly.

Accessor Functions

The following functions should be used for accessing the corresponding slots:

`clusters`: Cluster assignments of observations.

`posterior`: A matrix of posterior probabilities for each observation.

`raschmix-methods` *Methods for raschmix Objects*

Description

Methods for `raschmix-class` objects.

Usage

```
## S4 method for signature 'raschmix'
summary(object, eps = 1e-4, ...)

## S4 method for signature 'raschmix'
parameters(object,
       which = c("model", "item", "score", "concomitant"),
       difficulty = TRUE, component = NULL, simplify = TRUE)

## S4 method for signature 'raschmix'
worth(object, difficulty = TRUE, component = NULL)

## S4 method for signature 'raschmix'
itempar(object, ref = NULL, alias = TRUE, ...)

scoreProbs(object, component = NULL, simplify = TRUE)
```

Arguments

object	An object of class "raschmix".
eps	Probabilities below this threshold are treated as zero in the summary method.
which	Indicates which type of parameters are used. model refers to both item and score parameters, item and score to their corresponding parameters separately. The parameters of the concomitant model are accessed through concomitant.
difficulty	Indicates whether item difficulty (default) or easiness parameters are used.
component	Indicates which components are returned. Default is all components.
ref	a vector of labels or position indices of item parameters or a contrast matrix which should be used as restriction/for normalization. If 'NULL' (the default), all items are used (sum zero restriction).
alias	logical. If 'TRUE' (the default), the aliased parameter is included in the return vector. If 'FALSE', it is removed. If the restriction given in 'ref' depends on several parameters, the first parameter of the restriction specified is (arbitrarily) chosen to be removed if 'alias' is 'FALSE'.
simplify	Should the result be simplified if possible?
...	Currently not used.

Details

worth transforms the item parameters so that the sum over all item parameters (within each component) is zero.

itempar allows for the flexible specification of the restriction applied to the item parameters.

scoreProbs does not include any aliased parameters if a certain raw score is not present in the data.

---

raschmix-plot-method *Profile Plot of Item Parameters*

Description

The plot method for raschmix-class objects gives a base plot of the item parameter profiles in each class. A lattice plot of the item parameters is returned by xyplot. A rootogram or histogram of the posterior probabilities is plotted via histogram.

Usage

```
## S4 method for signature 'raschmix,missing'
plot(x, y, component = NULL, difficulty = TRUE,
                   center = TRUE, index = TRUE, names = TRUE,
                   abbreviate = FALSE, ref = TRUE, col =  "black",
                   refcol = "lightgray", linecol = NULL, lty = 2, cex = 1,
                   pch = 19, type = NULL, ylim = NULL, xlab = "Items",
                   ylab = NULL, legend = TRUE, pos = "topright",
```

```
                    srt = 45, adj = c(1.1, 1.1), ...)
## S3 method for class 'raschmix'
histogram(x, data, root = TRUE, ...)
## S3 method for class 'raschmix'
xyplot(x, data, component = NULL, item = NULL,
  difficulty = TRUE, plot.type = c("multiple", "single"),
  auto.key = NULL, type = "b", lty = NULL, xlab = "Item", ylab = NULL,
  panel = NULL, scales = NULL, ...)
```

Arguments

x	An object of class "raschmix".
y	Not used.
component	A vector indicating which components should be plotted.
difficulty	Logical. Should item difficulty parameters be used?
center	Logical. Should the item parameters be centered around 0?
index	Logical. Should the index be used for labelling the items?
names	Either logical or an optional vector of names used for labeling of the items.
abbreviate	Logical. Should the labels of the items be abbreviated?
ref	Logical. Should a reference line be drawn?
col	Point color. If col is a vector, it is interpreted as the color of the components respectively. Individual coloring within components is possible if col is given as a matrix with each column representing one component.
refcol	Color of the reference line.
linecol	Line color. Defaults to the point color.
lty, cex, pch, type, ylim, xlab, ylab	Further standard graphical parameters.
legend	Logical. Should a legend be included?
pos	Position of the legend.
srt, adj	Passed on to text() if names = TRUE.
...	Further graphical parameters.
data	Ignored.
root	Logical. Should a rootogram be drawn?
item	A vector indicating which items should be plotted.
plot.type	Should the item profiles be drawn in multiple panels or a single panel?
auto.key, panel, scales	Further graphical parameters for lattice.

Details

For a graphical representation of the item parameter in each class use plot (for a base graph) or xyplot (for a lattice plot).

For a graphical representation of the quality of the mixture use histogram. For details see plot-methods.

References

Frick, H., Strobl, C., Leisch, F., and Zeileis, A. (2012). Flexible Rasch Mixture Models with Package psychomix. *Journal of Statistical Software*, **48**(7), 1–25. http://www.jstatsoft.org/v48/i07/.

Leisch, F. (2004). FlexMix: A General Framework for Finite Mixture Models and Latent Class Regression in R. *Journal of Statistical Software*, **11**(8), 1–18. http://www.jstatsoft.org/v11/i08/.

Leisch, F. (2004). Exploring the Structure of Mixture Model Components. In Jaromir Antoch, editor, *Compstat 2004 – Proceedings in Computational Statistics*, pages 1405–1412. Physika Verlag, Heidelberg, Germany. ISBN 3-7908-1554-3.

`simRaschmix` *Simulate Data from Rasch Mixture Models*

Description

Generate simulated data from mixtures of Rasch models. The latent classes of the mixture can differ regarding size as well as item and person parameters of the corresponding Rasch model.

Usage

```
simRaschmix(design, extremes = FALSE, attributes = TRUE, ...)
```

Arguments

`design`	Type of data generating process. Can be provided as a character or a named list. See Details.
`extremes`	Logical. Should observations with none or all items solved be included in the data?
`attributes`	Logical. Should the true group membership as well as true item and person parameters be attached to the data as attributes `"cluster"`, `"difficulty"`, and `"ability"`?
`...`	Currently not used.

Details

The `design` of the data generating process (DGP) can be provided in essentially three different ways.

If the `design` argument is one of `"rost1"`, `"rost2"` or `"rost3"`, responses from the three DGPs introduced in Rost (1990) will be drawn.

Alternatively, the `design` can be provided as a named list with elements `nobs`, `weights`, `ability`, and `difficulty`. The `weights` can be provided in three formats: If provided as a vector of probabilities (summing to 1), class membership will be drawn with these probabilities. If `weights` is a vector of integer weights (summing to `nobs`, or an integer division thereof), Class sizes will be either the weights directly or a multiple thereof. As a third alternative, the `weights` can be provided

as a function of the number of observations (nobs). The ability specification can also be provided in three formats: If provided as a matrix of dimension 2xk with mean and standard deviation for each of the k clusters, the ability parameters are drawn from a normal distribution with the corresponding parameters. Second, ability can be an array of dimension (., 2, k) with abilities and corresponding weights/probabilities per cluster. Third, it can also be provided as a list of k functions which take the number of observations as an argument. The specification of the item difficulty can be provided either as a matrix with k columns with the item difficulties per cluster or as a matrix with nobs rows with the item difficulties per subject.

As a third option, design may also be a named list containing a vector of ability parameters and a matrix difficulty of dimension (number of observation x number of items).

Value

A matrix of item responses with dimension (number of observations x number of items). If attributes = TRUE, the matrix has attributes cluster, ability, and difficulty. The class memberships cluster are only returned when not provided implicitly through and a vector of abilties and a difficulty matrix with entries for each subject.

References

Frick, H., Strobl, C., Leisch, F., and Zeileis, A. (2012). Flexible Rasch Mixture Models with Package psychomix. *Journal of Statistical Software*, **48**(7), 1–25. http://www.jstatsoft.org/v48/i07/.

Rost, J. (1990). Rasch Models in Latent Classes: An Integration of Two Approaches to Item Analysis. *Applied Psychological Measurement*, **14**(3), 271–282.

See Also

raschmix

Examples

```
#################
## Rost's DGPs ##
#################

set.seed(1990)

## DGP 1 with just one latent class
r1 <- simRaschmix(design = "rost1")
## less than 1800 observations because the extreme scorers have been removed
table(attr(r1, "ability"))
table(rowSums(r1))

## DGP 2 with 2 equally large latent classes
r2 <- simRaschmix(design = "rost2", extreme = TRUE)
## exactly 1800 observations including the extreme scorers
table(attr(r2, "ability"))
table(rowSums(r2))
```

```
## DGP 3 with 3 latent classes
r3 <- simRaschmix(design = "rost3")
## item parameters in the three latent classes
attr(r3, "difficulty")


###################################
## flexible specification of DGPs ##
###################################

set.seed(482)

## number of observations
nobs <- 8

## relative weights
weights <- c(1/4, 3/4)
## exact weights: either summing to nobs or an integer division thereof
weights <- c(2, 6)
weights <- c(1, 3)
## weights as function
## here the result is the same as when specifying relative weights
weights <- function(n) sample(size = n, 1:2, prob = c(1/4, 3/4), replace
= TRUE)

## class 1: only ability level 0
## class 2: normally distributed abilities with mean = 2 and sd = 1
ability <- cbind(c(0, 0), c(2, 1))
## class 1: 3 ability levels (-1, 0, 1); class 2: 2 ability levels (-0.5, 0.5)
## with equal probabilities and frequencies, repectively
ability <- array(c(cbind(-1:1, rep(1/3, 3)), cbind(-1:1/2, c(0.5, 0, 0.5))),
  dim = c(3, 2, 2))
ability <- array(c(cbind(-1:1, rep(1, 3)), cbind(-1:1/2, c(1, 0, 1))),
  dim = c(3, 2, 2))
## ability as function
ability <- list(
  function(n) rnorm(n, mean = 0, sd = 0.5),
  function(n) sample(c(-0.5, 0.5), size = n, replace = TRUE)
)

## difficulty per latent class
difficulty <- cbind(c(-1,1,rep(0,8)), c(rep(0,8),1,-1))

## simulate data
dat <- simRaschmix(design = list(nobs = nobs, weights = weights,
  ability = ability, difficulty = difficulty))

## inspect attributes and raw scores
table(attr(dat, "cluster"))
hist(attr(dat, "ability"))
barplot(table(rowSums(dat)))
attr(dat, "difficulty")
```

```
## specification of DGP only via ability and difficulty
## one vector of abilities of all subjects
ability <- c(rnorm(4, mean = 0, sd = 0.5), sample(c(-0.5, 0.5), size = 4,
  replace = TRUE))
## difficulty per subject
difficulty <- matrix(c(rep(c(-1,1,rep(0,8)), 4), rep(c(rep(0,8),1,-1), 4)),
  nrow = 8, byrow = TRUE)
## simulate data
dat <- simRaschmix(design = list(ability = ability, difficulty = difficulty))

## inspect attributes and raw scores
hist(attr(dat, "ability"))
barplot(table(rowSums(dat)))
attr(dat, "difficulty")
```

Index