psychomix - Psychometric Mixture Models: Version 0.1-1

Frick, H ; Strobl, Carolin ; Leisch, F ; Zeileis, A

Abstract: Psychometric mixture models based on flexmix infrastructure. At the moment only Rasch mixture models are implemented in various flavors: with/without concomitant variables, different parametrizations of the score distribution (saturated vs. mean/variance specification). See vignette(‘‘raschmix’’, package = ‘‘psychomix’’) for details. Depends R (>= 2.10.0), graphics, stats, methods, Formula, flexmix,psychotools

Other titles: Package ‘psychomix’

Originally published at:
Frick, H; Strobl, Carolin; Leisch, F; Zeileis, A (2011). psychomix - Psychometric Mixture Models: Version 0.1-1. Wien: The R Foundation for Statistical Computing.

Package ‘psychomix’

October 5, 2011

Version 0.1-1

Date 2011-10-05

Title Psychometric Mixture Models

Authors@R c(person(given = ‘‘Hannah’’, family = ‘‘Frick’’, role = c(‘‘aut’’,‘‘cre’’), email = ‘‘Hannah.Frick@uibk.ac.at’’), person(given = ‘‘Carolin’’, family = ‘‘Strobl’’, role = ‘‘aut’’, email = ‘‘Carolin.Strobl@psychologie.uzh.ch’’), person(given = ‘‘Friedrich’’, family = ‘‘Leisch’’, role = ‘‘aut’’, email = ‘‘Friedrich.Leisch@R-project.org’’), person(given = ‘‘Achim’’,family = ‘‘Zeileis’’, role = ‘‘aut’’, email = ‘‘Achim.Zeileis@R-project.org’’))

Author Hannah Frick [aut, cre], Carolin Strobl [aut], Friedrich Leisch [aut], Achim Zeileis [aut]

Maintainer Hannah Frick <`Hannah.Frick@uibk.ac.at`>

Description Psychometric mixture models based on flexmix infrastructure. At the moment only Rasch mixture models are implemented in various flavors: with/without concomitant variables, different parametrizations of the score distribution (saturated vs. mean/variance specification). See vignette(‘‘raschmix’’, package = ‘‘psychomix’’) for details.

Depends R (>= 2.10.0), graphics, stats, methods, Formula, flexmix,psychotools

Imports graphics, stats, lattice

Suggests mRm

License GPL-2

Repository CRAN

Date/Publication 2011-10-05 12:18:10

R topics documented:

`raschmix` *Finite Mixtures of Rasch Models*

Description

Fit finte 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"),
         nrep = 3, cluster = NULL, control = list(minprior = 0),
         verbose = TRUE, drop = TRUE, unique = FALSE, which = NULL,
 gradtol = 1e-6, deriv = "sum", hessian = FALSE, ...)

FLXMCrasch(formula = . ~ ., scores = c("saturated", "meanvar"),
  nonExtremeProb = 1, gradtol = 1e-6, deriv = "sum", hessian = FALSE, ...)

simRaschmix(nobs = 1800, itemp = NULL, mean = NULL, sd = NULL,
            design = c("rost1", "rost2", "rost3", "cont1",
                       "cont1-2", "cont2", "cont2-2"),
            extremes = FALSE, attributes = 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.

`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.

`gradtol, deriv, hessian`
Control parameters passed to `RaschModel.fit` for the M-step.

`nobs` Number of observations.

`itemp` Optional item parameters for designs `"cont1"`, `"cont1-2"`, `"cont2"`, and `"cont2-2"`. Given as a `matrix` with each column representing one latent class.

`mean, sd` Parameters of the normal distribution used to generate person parameters.

`design` Type of data generating process.

`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 `"group"`, `"item"`, and `"person"`?

`...` Currently not used.

Details

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

`FLXMCrasch` is the `flexmix`-driver for the Rasch mixture models with saturated score distribution as proposed by Rost (1990), also known as “Mixed Rasch Model”.

For the `design` argument of `simRaschmix`, `"rost1"`, `"rost2"`, and `"rost3"` refer to the 3 data generating processes (dgps) introduced in Rost (1990). The other arguments refer to similar dgps with the same sets of item parameters but the person parameters drawn from normal distributions. `"cont1"` is the counterpart to `"rost1"` with the person parameters drawn from a standard normal distribution. `"cont1-2"` is similar, but the person parameters stem from two different normal distributions with means 2 and -2. `"cont2"` and `"cont2-2"` are the counterparts to `"rost2"`. For `"cont2"` for all observations with the same item parameters, the same normal distribution is used to generate person parameters. For `"cont2-2"`, in each group regarding the item parameters extra heterogenity is introduced via sampling the person parameters from two different normal distributions (with means 2 and -2), similar to dgp `"cont1-2"`.

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

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`

Examples

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

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

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


####################################################
## 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, "item")

## graphical comparison
plot(m1b, pos = "top")
for(i in 1:2) lines(attr(r2, "item")[,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, "group"))

## 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 parametrization,
## fit multinomial logit model for score probabilites

## 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, ”item”)[,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, either for the Rost or the conditional version of the mixture Rasch 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`: FIXME

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

`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:

`cluster`: 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)

scoreProbs(object, component = NULL, simplify = TRUE, drop = 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.

`simplify` Should the result be simplified if possible?

`drop` Argument passed on directly to the parameters method for `flexmix` objects.

`...` Currently not used.

Details

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

`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 lettice 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 = NULL,
                   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", ...)
## 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` 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.

`...` 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

Friedrich Leisch. FlexMix: A general framework for finite mixture models and latent class regression in R. Journal of Statistical Software, 11(8), 2004. http://www.jstatsoft.org/v11/i08/

Friedrich Leisch. Exploring the structure of mixture model components. In Jaromir Antoch, editor, Compstat 2004 - Proceedings in Computational Statistics, pages 1405-1412. Physika Verlag, Heidelberg, Germany, 2004. ISBN 3-7908-1554-3.

Index