Package ‘mirt’

May 4, 2017

Version 1.24

Type Package

Title Multidimensional Item Response Theory

Description Analysis of dichotomous and polytomous response data using unidimensional and multidimensional latent trait models under the Item Response Theory paradigm. Exploratory and confirmatory models can be estimated with quadrature (EM) or stochastic (MHRM) methods. Confirmatory bi-factor and two-tier analyses are available for modeling item testlets. Multiple group analysis and mixed effects designs also are available for detecting differential item and test functioning as well as modelling item and person covariates. Finally, latent class models such as the DINA, DINO, multidimensional latent class, and several other discrete latent variable models are supported.

VignetteBuilder knitr

Depends stats, R (>= 3.1.0), stats4, lattice, methods

Imports GPArotation, Rcpp, sfsmisc, mgcv, splines, numDeriv

Suggests boot, latticeExtra, directlabels, shiny, knitr, Rsolnp, alabama, sirt, mirtCAT

ByteCompile yes

LazyLoad yes

LazyData yes

LinkingTo Rcpp, RcppArmadillo

License GPL (>= 3)

Repository CRAN

Maintainer Phil Chalmers <rphilip.chalmers@gmail.com>

URL https://github.com/philchalmers/mirt,
https://github.com/philchalmers/mirt/wiki,
https://groups.google.com/forum/#!forum/mirt-package

BugReports https://github.com/philchalmers/mirt/issues?state=open

RoxygenNote 6.0.1

NeedsCompilation yes

Author Phil Chalmers [aut, cre, cph],
Joshua Pritikin [ctb],
Alexander Robitzsch [ctb],
Mateusz Zoltak [ctb],
KwonHyun Kim [ctb],
Carl F. Falk [ctb],
Adam Meade [ctb]

Date/Publication 2017-05-04 07:53:16 UTC

R topics documented:

`mirt-package` *Full information maximum likelihood estimation of IRT models.*

Description

Full information maximum likelihood estimation of multidimensional IRT models

Details

Analysis of dichotomous and polytomous response data using unidimensional and multidimensional latent trait models under the Item Response Theory paradigm. Exploratory and confirmatory models can be estimated with quadrature (EM) or stochastic (MHRM) methods. Confirmatory bi-factor and two-tier analyses are available for modeling item testlets. Multiple group analysis and mixed effects designs also are available for detecting differential item and test functioning as well as modelling item and person covariates. Finally, latent class models such as the DINA, DINO, multidimensional latent class, and several other discrete variable models are supported.

Users interested in the most recent version of this package can visit `https://github.com/philchalmers/mirt` and follow the instructions for installing the package from source. Questions regarding the package can be sent to the mirt-package Google Group, located at `https://groups.google.com/forum/#!forum/mirt-package`. User contributed files, workshop files, and evaluated help files are also available on the package wiki (`https://github.com/philchalmers/mirt/wiki`).

Author(s)

Phil Chalmers <`rphilip.chalmers@gmail.com`>

`anova-method` *Compare nested models with likelihood-based statistics*

Description

Compare nested models using likelihood ratio, AIC, BIC, etc.

Usage

```
## S4 method for signature 'SingleGroupClass'
anova(object, object2, bounded = FALSE,
  mix = 0.5, verbose = TRUE)
```

Arguments

`object`	an object of class `SingleGroupClass`, `MultipleGroupClass`, or `MixedClass`
`object2`	a second model estimated from any of the mirt package estimation methods
`bounded`	logical; are the two models comparing a bounded parameter (e.g., comparing a single 2PL and 3PL model with 1 df)? If `TRUE` then a 50:50 mix of chi-squared distributions is used to obtain the p-value
`mix`	proportion of chi-squared mixtures. Default is 0.5
`verbose`	logical; print additional information to console?

Examples

```
## Not run:
x <- mirt(Science, 1)
x2 <- mirt(Science, 2)
anova(x, x2)

# bounded parameter
dat <- expand.table(LSAT7)
mod <- mirt(dat, 1)
mod2 <- mirt(dat, 1, itemtype = c(rep('2PL', 4), '3PL'))
anova(mod, mod2) #unbounded test
anova(mod, mod2, bounded = TRUE) #bounded


## End(Not run)
```

`areainfo` *Function to calculate the area under a selection of information curves*

Description

Compute the area within test or item information over a definite integral range.

Usage

```
areainfo(x, theta_lim, which.items = 1:extract.mirt(x, "nitems"), ...)
```

Arguments

`x`	an estimated mirt object
`theta_lim`	range of integration to be computed
`which.items`	an integer vector indicating which items to include in the expected information function. Default uses all possible items
`...`	additional arguments passed to `integrate`

Value

a `data.frame` with the lower and upper integration range, the information area within the range (Info), the information area over the range -10 to 10 (Total.Info), proportion of total information given the integration range (Info.Proportion), and the number of items included (nitems)

Author(s)

Phil Chalmers `<rphilip.chalmers@gmail.com>`

Examples

```
## Not run:
dat <- expand.table(LSAT7)
mod <- mirt(dat, 1)

areainfo(mod, c(-2,0), which.items = 1) #item 1
areainfo(mod, c(-2,0), which.items = 1:3) #items 1 to 3
areainfo(mod, c(-2,0)) # all items (total test information)

# plot the area
area <- areainfo(mod, c(-2,0))
Theta <- matrix(seq(-3,3, length.out=1000))
info <- testinfo(mod, Theta)
plot(info ~ Theta, type = 'l')

pick <- Theta >= -2 & Theta <=0
polygon(c(-2, Theta[pick], 0), c(0, info[pick], 0), col='lightblue')
text(x = 2, y = 0.5, labels = paste("Total Information:", round(area$TotalInfo, 3),
           "\n\nInformation in (-2, 0):", round(area$Info, 3),
           paste("(", round(100 * area$Proportion, 2), "%)", sep = "")), cex = 1.2)


## End(Not run)
```

---

averageMI *Collapse values from multiple imputation draws*

Description

This function computes updated parameter and standard error estimates using multiple imputation methodology. Given a set of parameter estimates and their associated standard errors the function returns the weighted average of the overall between and within variability due to the multiple imputations according to Rubin's (1987) methodology.

Usage

```
averageMI(par, SEpar, as.data.frame = TRUE)
```

Arguments

`par`	a list containing parameter estimates which were computed the imputed datasets
`SEpar`	a list containing standard errors associated with `par`
`as.data.frame`	logical; return a data.frame instead of a list? Default is TRUE

Value

returns a list or data.frame containing the updated averaged parameter estimates, standard errors, and t-values with the associated degrees of freedom and two tailed p-values

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

References

Rubin, D.B. (1987) Multiple Imputation for Nonresponse in Surveys. Wiley & Sons, New York.

Examples

```
## Not run:

#simulate data
set.seed(1234)
N <- 1000

# covariates
X1 <- rnorm(N); X2 <- rnorm(N)
covdata <- data.frame(X1, X2)
Theta <- matrix(0.5 * X1 + -1 * X2 + rnorm(N, sd = 0.5))

#items and response data
a <- matrix(1, 20); d <- matrix(rnorm(20))
dat <- simdata(a, d, 1000, itemtype = '2PL', Theta=Theta)

mod1 <- mirt(dat, 1, 'Rasch', covdata=covdata, formula = ~ X1 + X2)
coef(mod1, simplify=TRUE)

#draw plausible values for secondary analyses
pv <- fscores(mod1, plausible.draws = 10)
pvmods <- lapply(pv, function(x, covdata) lm(x ~ covdata$X1 + covdata$X2),
                 covdata=covdata)

# compute Rubin's multiple imputation average
so <- lapply(pvmods, summary)
par <- lapply(so, function(x) x$coefficients[, 'Estimate'])
SEpar <- lapply(so, function(x) x$coefficients[, 'Std. Error'])
averageMI(par, SEpar)


## End(Not run)
```

`bfactor` *Full-Information Item Bi-factor and Two-Tier Analysis*

Description

`bfactor` fits a confirmatory maximum likelihood two-tier/bifactor/testlet model to dichotomous and polytomous data under the item response theory paradigm. The IRT models are fit using a dimensional reduction EM algorithm so that regardless of the number of specific factors estimated the model only uses the number of factors in the second-tier structure plus 1. For the bifactor model the maximum number of dimensions is only 2 since the second-tier only consists of a ubiquitous unidimensional factor. See mirt for appropriate methods to be used on the objects returned from the estimation.

Usage

```
bfactor(data, model, model2 = paste0("G = 1-", ncol(data)), group = NULL,
  quadpts = NULL, invariance = "", ...)
```

Arguments

- `data` — a `matrix` or `data.frame` that consists of numerically ordered data, with missing data coded as `NA`
- `model` — a numeric vector specifying which factor loads on which item. For example, if for a 4 item test with two specific factors, the first specific factor loads on the first two items and the second specific factor on the last two, then the vector is `c(1,1,2,2)`. For items that should only load on the second-tier factors (have no specific component) `NA` values may be used as place-holders. These numbers will be translated into a format suitable for `mirt.model()`, combined with the definition in `model2`, with the letter 'S' added to the respective factor number
- `model2` — a two-tier model specification object defined by `mirt.model()` or a string to be passed to `mirt.model`. By default the model will fit a unidimensional model in the second-tier, and therefore be equivalent to the bifactor model
- `group` — a factor variable indicating group membership used for multiple group analyses
- `quadpts` — number of quadrature nodes to use after accounting for the reduced number of dimensions. Scheme is the same as the one used in `mirt`, however it is in regardes to the reduced dimensions (e.g., a bifactor model has 2 dimensions to be integrated)
- `invariance` — see `multipleGroup` for details, however, the specific factor variances and means will be constrained according to the dimensional reduction algorithm
- `...` — additional arguments to be passed to the estimation engine. See `mirt` for more details and examples

Details

`bfactor` follows the item factor analysis strategy explicated by Gibbons and Hedeker (1992), Gibbons et al. (2007), and Cai (2010). Nested models may be compared via an approximate chi-squared difference test or by a reduction in AIC or BIC (accessible via `anova`). See `mirt` for more details regarding the IRT estimation approach used in this package.

The two-tier model has a specific block diagonal covariance structure between the primary and secondary latent traits. Namely, the secondary latent traits are assumed to be orthogonal to all traits and have a fixed variance of 1, while the primary traits can be organized to vary and covary with other primary traits in the model.

$$\Sigma_{two-tier} = \begin{pmatrix} G & 0 \\ 0 & diag(S) \end{pmatrix}$$

The bifactor model is a special case of the two-tier model when G above is a 1x1 matrix, and therefore only 1 primary factor is being modeled. Evaluation of the numerical integrals for the two-tier model requires only $ncol(G) + 1$ dimensions for integration since the S second order (or 'specific') factors require only 1 integration grid due to the dimension reduction technique.

Note: for multiple group two-tier analyses only the second-tier means and variances should be freed since the specific factors are not treated independently due to the dimension reduction technique.

Value

function returns an object of class `SingleGroupClass` (SingleGroupClass-class) or `MultipleGroupClass`(MultipleGroupClass-class).

Author(s)

Phil Chalmers `<rphilip.chalmers@gmail.com>`

References

Cai, L. (2010). A two-tier full-information item factor analysis model with applications. *Psychometrika, 75*, 581-612.

Chalmers, R., P. (2012). mirt: A Multidimensional Item Response Theory Package for the R Environment. *Journal of Statistical Software, 48*(6), 1-29.

Gibbons, R. D., & Hedeker, D. R. (1992). Full-information Item Bi-Factor Analysis. *Psychometrika, 57*, 423-436.

Gibbons, R. D., Darrell, R. B., Hedeker, D., Weiss, D. J., Segawa, E., Bhaumik, D. K., Kupfer, D. J., Frank, E., Grochocinski, V. J., & Stover, A. (2007). Full-Information item bifactor analysis of graded response data. *Applied Psychological Measurement, 31*, 4-19.

See Also

`mirt`

Examples

```
## Not run:

###load SAT12 and compute bifactor model with 3 specific factors
data(SAT12)
data <- key2binary(SAT12,
  key = c(1,4,5,2,3,1,2,1,3,1,2,4,2,1,5,3,4,4,1,4,3,3,4,1,3,5,1,3,1,5,4,5))
specific <- c(2,3,2,3,3,2,1,2,1,1,1,3,1,3,1,2,1,1,3,3,1,1,3,1,3,3,1,3,2,3,1,2)
mod1 <- bfactor(data, specific)
summary(mod1)
itemplot(mod1, 18, drop.zeros = TRUE) #drop the zero slopes to allow plotting

###Try with fixed guessing parameters added
guess <- rep(.1,32)
mod2 <- bfactor(data, specific, guess = guess)
coef(mod2)
```

```
anova(mod1, mod2)

## don't estimate specific factor for item 32
specific[32] <- NA
mod3 <- bfactor(data, specific)
anova(mod1, mod3)

# same, but decalred manually (not run)
#sv <- mod2values(mod1)
#sv$value[220] <- 0 #parameter 220 is the 32 items specific slope
#sv$est[220] <- FALSE
#mod3 <- bfactor(data, specific, pars = sv) #with excellent starting values


#########
# mixed itemtype example

#simulate data
a <- matrix(c(
1,0.5,NA,
1,0.5,NA,
1,0.5,NA,
1,0.5,NA,
1,0.5,NA,
1,0.5,NA,
1,0.5,NA,
1,NA,0.5,
1,NA,0.5,
1,NA,0.5,
1,NA,0.5,
1,NA,0.5,
1,NA,0.5,
1,NA,0.5),ncol=3,byrow=TRUE)

d <- matrix(c(
-1.0,NA,NA,
-1.5,NA,NA,
 1.5,NA,NA,
 0.0,NA,NA,
2.5,1.0,-1,
3.0,2.0,-0.5,
3.0,2.0,-0.5,
3.0,2.0,-0.5,
2.5,1.0,-1,
2.0,0.0,NA,
-1.0,NA,NA,
-1.5,NA,NA,
 1.5,NA,NA,
 0.0,NA,NA),ncol=3,byrow=TRUE)
items <- rep('2PL', 14)
items[5:10] <- 'graded'

sigma <- diag(3)
```

```
dataset <- simdata(a,d,2000,itemtype=items,sigma=sigma)

specific <- c(rep(1,7),rep(2,7))
simmod <- bfactor(dataset, specific)
coef(simmod)

#########
# testlet response model

#simulate data
set.seed(1234)
a <- matrix(0, 12, 4)
a[,1] <- rlnorm(12, .2, .3)
ind <- 1
for(i in 1:3){
   a[ind:(ind+3),i+1] <- a[ind:(ind+3),1]
   ind <- ind+4
}
print(a)
d <- rnorm(12, 0, .5)
sigma <- diag(c(1, .5, 1, .5))
dataset <- simdata(a,d,2000,itemtype=rep('2PL', 12),sigma=sigma)

# estimate by applying constraints and freeing the latent variances
specific <- c(rep(1,4),rep(2,4), rep(3,4))
model <- "G = 1-12
          CONSTRAIN = (1, a1, a2), (2, a1, a2), (3, a1, a2), (4, a1, a2),
            (5, a1, a3), (6, a1, a3), (7, a1, a3), (8, a1, a3),
            (9, a1, a4), (10, a1, a4), (11, a1, a4), (12, a1, a4)
          COV = S1*S1, S2*S2, S3*S3"

simmod <- bfactor(dataset, specific, model)
coef(simmod, simplify=TRUE)


#########
# Two-tier model

#simulate data
set.seed(1234)
a <- matrix(c(
  0,1,0.5,NA,NA,
  0,1,0.5,NA,NA,
  0,1,0.5,NA,NA,
  0,1,0.5,NA,NA,
  0,1,0.5,NA,NA,
  0,1,NA,0.5,NA,
  0,1,NA,0.5,NA,
  0,1,NA,0.5,NA,
  1,0,NA,0.5,NA,
  1,0,NA,0.5,NA,
  1,0,NA,0.5,NA,
  1,0,NA,NA,0.5,
```

```
  1,0,NA,NA,0.5,
  1,0,NA,NA,0.5,
  1,0,NA,NA,0.5,
  1,0,NA,NA,0.5),ncol=5,byrow=TRUE)

d <- matrix(rnorm(16))
items <- rep('2PL', 16)

sigma <- diag(5)
sigma[1,2] <- sigma[2,1] <- .4
dataset <- simdata(a,d,2000,itemtype=items,sigma=sigma)

specific <- c(rep(1,5),rep(2,6),rep(3,5))
model <- '
    G1 = 1-8
    G2 = 9-16
    COV = G1*G2'

#quadpts dropped for faster estimation, but not as precise
simmod <- bfactor(dataset, specific, model, quadpts = 9, TOL = 1e-3)
coef(simmod, simplify=TRUE)
summary(simmod)
itemfit(simmod, QMC=TRUE)
M2(simmod, QMC=TRUE)
residuals(simmod, QMC=TRUE)


## End(Not run)
```

Bock1997 *Description of Bock 1997 data*

Description

A 3-item tabulated data set extracted from Table 3 in Chapter Two.

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

References

Bock, R. D. (1997). The Nominal Categories Model. In van der Linden, W. J. & Hambleton, R. K. *Handbook of modern item response theory*. New York: Springer.

Examples

```
## Not run:
dat <- expand.table(Bock1997)
head(dat)
mod <- mirt(dat, 1, 'nominal')

#reproduce table 3 in Bock (1997)
fs <- round(fscores(mod, verbose = FALSE, full.scores = FALSE)[,c('F1','SE_F1')],2)
fttd <- residuals(mod, type = 'exp')
table <- data.frame(fttd[,-ncol(fttd)], fs)
table

mod <- mirt(dat, 1, 'nominal')
coef(mod)


## End(Not run)
```

`boot.LR` *Parametric bootstrap likleihood-ratio test*

Description

Given two fitted models, compute a parametric bootstrap test to determine whether the less restrictive models fits significantly better than the more restricted model. Note that this hypothesis test also works when prior parameter distributions are included for either model. Function can be run in parallel after using a stuitable `mirtCluster` definition.

Usage

```
boot.LR(mod, mod2, R = 1000)
```

Arguments

`mod`	an estimated model object
`mod2`	an estimated model object
`R`	number of parametric bootstraps to use.

Value

a p-value evaluating whether the more restrictive model fits significantly worse than the less restrictive model

Author(s)

Phil Chalmers `<rphilip.chalmers@gmail.com>`

Examples

```
## Not run:

#standard
dat <- expand.table(LSAT7)
mod1 <- mirt(dat, 1)
mod2 <- mirt(dat, 1, '3PL')

# standard LR test
anova(mod1, mod2)

# bootstrap LR test (run in parallel to save time)
mirtCluster()
boot.LR(mod1, mod2, R=200)


## End(Not run)
```

---

boot.mirt — *Calculate bootstrapped standard errors for estimated models*

Description

Given an internal mirt object estimate the bootstrapped standard errors. It may be beneficial to run the computations using multi-core architecture (e.g., the `parallel` package). Parameters are organized from the freely estimated values in `mod2values(x)` (equality constraints will also be returned in the bootstrapped estimates).

Usage

```
boot.mirt(x, R = 100, technical = NULL, ...)
```

Arguments

`x`	an estimated model object
`R`	number of draws to use (passed to the `boot()` function)
`technical`	technical arguments passed to estimation engine. See `mirt` for details
`...`	additional arguments to be passed on to `boot(...)` and estimation engine

Author(s)

Phil Chalmers <`rphilip.chalmers@gmail.com`>

Examples

```
## Not run:

#standard
mod <- mirt(Science, 1)
booted <- boot.mirt(mod, R=20)
plot(booted)
booted

#run in parallel using snow back-end using all available cores
mod <- mirt(Science, 1)
booted <- boot.mirt(mod, parallel = 'snow', ncpus = parallel::detectCores())
booted



## End(Not run)
```

---

coef-method — *Extract raw coefs from model object*

Description

Return a list (or data.frame) of raw item and group level coefficients. Note that while the output to the console is rounded to three digits, the returned list of objects is not. Hence, elements from `cfs <- coef(mod); cfs[[1]]` will contain the unrounded results (useful for simulations).

Usage

```
## S4 method for signature 'SingleGroupClass'
coef(object, CI = 0.95, printSE = FALSE,
  rotate = "none", Target = NULL, IRTpars = FALSE, rawug = FALSE,
  as.data.frame = FALSE, simplify = FALSE, unique = FALSE,
  verbose = TRUE, ...)
```

Arguments

`object`	an object of class `SingleGroupClass`, `MultipleGroupClass`, or `MixedClass`
`CI`	the amount of converged used to compute confidence intervals; default is 95 percent confidence intervals
`printSE`	logical; print the standard errors instead of the confidence intervals?
`rotate`	see `summary` method for details. The default rotation is `'none'`
`Target`	a dummy variable matrix indicting a target rotation pattern
`IRTpars`	logical; convert slope intercept parameters into traditional IRT parameters? Only applicable to unidimensional models

rawug	logical; return the untransformed internal g and u parameters? If FALSE, g and u's are converted with the original format along with delta standard errors
as.data.frame	logical; convert list output to a data.frame instead?
simplify	logical; if all items have the same parameter names (indicating they are of the same class) then they are collapsed to a matrix, and a list of length 2 is returned containing a matrix of item parameters and group-level estimates
unique	return the vector of uniquely estimated parameters
verbose	logical; allow information to be printed to the console?
...	additional arguments to be passed

See Also

summary-method

Examples

```
## Not run:
dat <- expand.table(LSAT7)
x <- mirt(dat, 1)
coef(x)
coef(x, IRTpars = TRUE)
coef(x, simplify = TRUE)

#with computed information matrix
x <- mirt(dat, 1, SE = TRUE)
coef(x)
coef(x, printSE = TRUE)
coef(x, as.data.frame = TRUE)

#two factors
x2 <- mirt(Science, 2)
coef(x2)
coef(x2, rotate = 'varimax')


## End(Not run)
```

---

createGroup *Create a user defined group-level object with correct generic functions*

Description

Initializes the proper S4 class and methods necessary for mirt functions to use in estimation for definiting customized group-level functions. To use the defined objects pass to the `mirt(..., customGroup = OBJECT)` command, and ensure that the class parameters are properly labeled.

Usage

```
createGroup(par, est, den, nfact, gr = NULL, hss = NULL, gen = NULL,
  lbound = NULL, ubound = NULL, derivType = "Richardson")
```

Arguments

`par`	a named vector of the starting values for the parameters
`est`	a logical vector indicating which parameters should be freely estimated by default
`den`	the probability density function given the Theta/ability values. First input contains a vector of all the defined parameters and the second input must be a matrix called `Theta`. Function also must return a `numeric` vector object corresponding to the associated densities for each row in the `Theta` input
`nfact`	number of factors required for the model. E.g., for unidimensional models with only one dimension of integration `nfact = 1`
`gr`	gradient function (vector of first derivatives) of the log-likelihood used in estimation. The function must be of the form `gr(x, Theta)`, where `x` is the object defined by `createGroup()` and `Theta` is a matrix of latent trait parameters
`hss`	Hessian function (matrix of second derivatives) of the log-likelihood used in estimation. If not specified a numeric approximation will be used. The input is idential to the `gr` argument
`gen`	a function used when `GenRandomPars = TRUE` is passed to the estimation function to generate random starting values. Function must be of the form `function(object) ...` and must return a vector with properties equivalent to the `par` object. If NULL, parameters will remain at the defined starting values by default
`lbound`	optional vector indicating the lower bounds of the parameters. If not specified then the bounds will be set to -Inf
`ubound`	optional vector indicating the lower bounds of the parameters. If not specified then the bounds will be set to Inf
`derivType`	if the `gr` or `hss` terms are not specified this type will be used to obtain them numerically. Default is 'Richardson'

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

Examples

```
## Not run:

# normal density example
den <- function(obj, Theta) dnorm(Theta, obj@par[1], obj@par[2])
par <- c(mu = 0, sigma = 1)
est <- c(FALSE, TRUE)
```

```
lbound <- c(-Inf, 0)
grp <- createGroup(par, est, den, nfact = 1, lbound=lbound)

mod <- mirt(Science, 1, 'Rasch')
modcustom <- mirt(Science, 1, 'Rasch', customGroup=grp)

coef(mod)
coef(modcustom)


## End(Not run)
```

`createItem` *Create a user defined item with correct generic functions*

Description

Initializes the proper S4 class and methods necessary for mirt functions to use in estimation. To use the defined objects pass to the `mirt(..., customItems = list())` command, and ensure that the classes are properly labeled and unique in the list.

Usage

```
createItem(name, par, est, P, gr = NULL, hss = NULL, gen = NULL,
  lbound = NULL, ubound = NULL, derivType = "forward")
```

Arguments

`name`	a character indicating the item class name to be defined
`par`	a named vector of the starting values for the parameters
`est`	a logical vector indicating which parameters should be freely estimated by default
`P`	the probability trace function for all categories (first column is category 1, second category two, etc). First input contains a vector of all the item parameters, the second input must be a matrix called `Theta`, and the third input must be the number of categories called `ncat`. Function also must return a `matrix` object of category probabilities
`gr`	gradient function (vector of first derivatives) of the log-likelihood used in estimation. The function must be of the form `gr(x, Theta)`, where `x` is the object defined by `createItem()` and `Theta` is a matrix of latent trait parameters. Tabulated (EM) or raw (MHRM) data are located in the `x@dat` slot, and are used to form the complete data log-likelihood. If not specified a numeric approximation will be used
`hss`	Hessian function (matrix of second derivatives) of the log-likelihood used in estimation. If not specified a numeric approximation will be used (required for the MH-RM algorithm only). The input is idential to the `gr` argument

gen — a function used when GenRandomPars = TRUE is passed to the estimation function to generate random starting values. Function must be of the form function(object) ... and must return a vector with properties equivalent to the par object. If NULL, parameters will remain at the defined starting values by default

lbound — optional vector indicating the lower bounds of the parameters. If not specified then the bounds will be set to -Inf

ubound — optional vector indicating the lower bounds of the parameters. If not specified then the bounds will be set to Inf

derivType — if the gr or hss terms are not specified this type will be used to obtain them numerically. Default is the 'forward' method (fastest), but more exact approaches include 'central' and 'Richardson'

Details

The summary() function will not return proper standardized loadings since the function is not sure how to handle them (no slopes could be defined at all!). Instead loadings of .001 are filled in as place-holders.

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

Examples

```
## Not run:

name <- 'old2PL'
par <- c(a = .5, b = -2)
est <- c(TRUE, TRUE)
P.old2PL <- function(par,Theta, ncat){
     a <- par[1]
     b <- par[2]
     P1 <- 1 / (1 + exp(-1*a*(Theta - b)))
     cbind(1-P1, P1)
}

x <- createItem(name, par=par, est=est, P=P.old2PL)

#So, let's estimate it!
dat <- expand.table(LSAT7)
sv <- mirt(dat, 1, c(rep('2PL',4), 'old2PL'), customItems=list(old2PL=x), pars = 'values')
tail(sv) #looks good
mod <- mirt(dat, 1, c(rep('2PL',4), 'old2PL'), customItems=list(old2PL=x))
coef(mod)
mod2 <- mirt(dat, 1, c(rep('2PL',4), 'old2PL'), customItems=list(old2PL=x), method = 'MHRM')
coef(mod2)

#several secondary functions supported
```

```
M2(mod, calcNull=FALSE)
itemfit(mod)
fscores(mod, full.scores=FALSE)
plot(mod)

# fit the same model, but specify gradient function explicitly (use of a browser() may be helpful)
gr <- function(x, Theta){
     # browser()
     a <- x@par[1]
     b <- x@par[2]
     P <- probtrace(x, Theta)
     PQ <- apply(P, 1, prod)
     r_P <- x@dat / P
     grad <- numeric(2)
     grad[2] <- sum(-a * PQ * (r_P[,2] - r_P[,1]))
     grad[1] <- sum((Theta - b) * PQ * (r_P[,2] - r_P[,1]))

     ## check with internal numerical form to be safe
     # numerical_deriv(x@par[x@est], mirt:::EML, obj=x, Theta=Theta, type='Richardson')
     grad
}

x <- createItem(name, par=par, est=est, P=P.old2PL, gr=gr)
mod <- mirt(dat, 1, c(rep('2PL',4), 'old2PL'), customItems=list(old2PL=x))
coef(mod, simplify=TRUE)

###non-linear
name <- 'nonlin'
par <- c(a1 = .5, a2 = .1, d = 0)
est <- c(TRUE, TRUE, TRUE)
P.nonlin <- function(par,Theta, ncat=2){
     a1 <- par[1]
     a2 <- par[2]
     d <- par[3]
     P1 <- 1 / (1 + exp(-1*(a1*Theta + a2*Theta^2 + d)))
     cbind(1-P1, P1)
}

x2 <- createItem(name, par=par, est=est, P=P.nonlin)

mod <- mirt(dat, 1, c(rep('2PL',4), 'nonlin'), customItems=list(nonlin=x2))
coef(mod)

###nominal response model (Bock 1972 version)
Tnom.dev <- function(ncat) {
   T <- matrix(1/ncat, ncat, ncat - 1)
   diag(T[-1, ]) <-  diag(T[-1, ]) - 1
   return(T)
}

name <- 'nom'
par <- c(alp=c(3,0,-3),gam=rep(.4,3))
est <- rep(TRUE, length(par))
```

```
P.nom <- function(par, Theta, ncat){
   alp <- par[1:(ncat-1)]
   gam <- par[ncat:length(par)]
   a <- Tnom.dev(ncat) %*% alp
   c <- Tnom.dev(ncat) %*% gam
   z <- matrix(0, nrow(Theta), ncat)
   for(i in 1:ncat)
       z[,i] <- a[i] * Theta + c[i]
   P <- exp(z) / rowSums(exp(z))
   P
}

nom1 <- createItem(name, par=par, est=est, P=P.nom, derivType = 'central')
nommod <- mirt(Science, 1, 'nom1', customItems=list(nom1=nom1))
coef(nommod)
Tnom.dev(4) %*% coef(nommod)[[1]][1:3] #a
Tnom.dev(4) %*% coef(nommod)[[1]][4:6] #d


## End(Not run)
```

deAyala *Description of deAyala data*

Description

Mathematics data from de Ayala (2009; pg. 14); 5 item dataset in table format.

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

References

de Ayala, R. J. (2009). *The theory and practice of item response theory*. Guilford Press.

Examples

```
## Not run:
dat <- expand.table(deAyala)
head(dat)

## End(Not run)
```

DIF *Differential item functioning statistics*

Description

This function runs the Wald and likelihood-ratio approaches for testing differential item functioning (DIF). This is primarily a convenience wrapper to the `multipleGroup` function for performing standard DIF procedures. Independent models can be estimated in parallel by defining a parallel object with `mirtCluster`, which will help to decrease the runtime. For best results, the baseline model should contain a set of 'anchor' items and have freely estimated hyper-parameters in the focal groups.

Usage

```
DIF(MGmodel, which.par, scheme = "add", items2test = 1:extract.mirt(MGmodel,
  "nitems"), seq_stat = "SABIC", Wald = FALSE, p.adjust = "none",
  return_models = FALSE, max_run = Inf, plotdif = FALSE, type = "trace",
  verbose = TRUE, ...)
```

Arguments

`MGmodel` an object returned from `multipleGroup` to be used as the reference model

`which.par` a character vector containing the parameter names which will be inspected for DIF

`scheme` type of DIF analysis to perform, either by adding or dropping constraints across groups. These can be:

- **'add'** parameters in `which.par` will be constrained each item one at a time for items that are specified in `items2test`. This is beneficial when examining DIF from a model with parameters freely estimated across groups, and when inspecting differences via the Wald test
- **'drop'** parameters in `which.par` will be freely estimated for items that are specified in `items2test`. This is useful when supplying an overly restrictive model and attempting to detect DIF with a slightly less restrictive model
- **'add_sequential'** sequentially loop over the items being tested, and at the end of the loop treat DIF tests that satisfy the `seq_stat` criteria as invariant. The loop is then re-run on the remaining invariant items to determine if they are now displaying DIF in the less constrained model, and when no new invariant item is found the algorithm stops and returns the items that displayed DIF
- **'drop_sequential'** sequentially loop over the items being tested, and at the end of the loop treat items that violate the `seq_stat` criteria as demonstrating DIF. The loop is then re-run, leaving the items that previously demonstrated DIF as variable across groups, and the remaining test items that previously showed invariance are re-tested. The algorithm stops when no more items showing DIF are found and returns the items that displayed DIF

items2test — a numeric vector, or character vector containing the item names, indicating which items will be tested for DIF. In models where anchor items are known, omit them from this vector. For example, if items 1 and 2 are anchors in a 10 item test, then `items2test = 3:10` would work for testing the remaining items (important to remember when using sequential schemes)

seq_stat — select a statistic to test for in the sequential schemes. Potential values are (in descending order of power) `'AIC'`, `'AICc'`, `'SABIC'`, and `'BIC'`. If a numeric value is input that ranges between 0 and 1, the 'p' value will be tested (e.g., `seq_stat = .05` will test for the difference of p < .05 in the add scheme, or p > .05 in the drop scheme), along with the specified `p.adjust` input

Wald — logical; perform Wald tests for DIF instead of likelihood ratio test?

p.adjust — string to be passed to the `p.adjust` function to adjust p-values. Adjustments are located in the `adj_pvals` element in the returned list

return_models — logical; return estimated model objects for further analysis? Default is FALSE

max_run — a number indicating the maximum number of cycles to perform in sequential searches. The default is to perform search until no further DIF is found

plotdif — logical; create item plots for items that are displaying DIF according to the `seq_stat` criteria? Only available for 'add' type schemes

type — the `type` of plot argument passed to `plot()`. Default is 'trace', though another good option is 'infotrace'. For ease of viewing, the `facet_item` argument to mirt's `plot()` function is set to `TRUE`

verbose — logical print extra information to the console?

... — additional arguments to be passed to `multipleGroup` and `plot`

Details

Generally, the precomputed baseline model should have been configured with two estimation properties: 1) a set of 'anchor' items, where the anchor items have various parameters that have been constrained to be equal across the groups, and 2) contain freely estimated latent mean and variance terms in all but one group (the so-called 'reference' group). These two properties help to fix the metric of the groups so that item parameter estimates do not contain latent distribution characteristics.

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

See Also

`multipleGroup`

Examples

```
## Not run:

#simulate data where group 2 has a smaller slopes and more extreme intercepts
set.seed(12345)
```

```
a1 <- a2 <- matrix(abs(rnorm(15,1,.3)), ncol=1)
d1 <- d2 <- matrix(rnorm(15,0,.7),ncol=1)
a2[1:2, ] <- a1[1:2, ]/3
d1[c(1,3), ] <- d2[c(1,3), ]/4
head(data.frame(a.group1 = a1, a.group2 = a2, d.group1 = d1, d.group2 = d2))
itemtype <- rep('2PL', nrow(a1))
N <- 1000
dataset1 <- simdata(a1, d1, N, itemtype)
dataset2 <- simdata(a2, d2, N, itemtype, mu = .1, sigma = matrix(1.5))
dat <- rbind(dataset1, dataset2)
group <- c(rep('D1', N), rep('D2', N))

#### no anchors, all items tested for DIF by adding item constrains one item at a time.
# define a parallel cluster (optional) to help speed up internal functions
mirtCluster()

#  Information matrix with Oakes' identity (not controlling for latent group differences)
model <- multipleGroup(dat, 1, group, SE = TRUE)

#test whether adding slopes and intercepts constraints results in DIF. Plot items showing DIF
resulta1d <- DIF(model, c('a1', 'd'), plotdif = TRUE)
resulta1d

#same as above, but using Wald tests with Benjamini & Hochberg adjustment
resulta1dWald <- DIF(model, c('a1', 'd'), Wald = TRUE, p.adjust = 'fdr')
resulta1dWald
round(resulta1dWald$adj_pvals, 4)

#test whether adding only slope constraints results in DIF for all items
resulta1 <- DIF(model, 'a1')
resulta1

#following up on resulta1d, to determine whether it's a1 or d parameter causing DIF
(a1s <- DIF(model, 'a1', items2test = 1:3))
(ds <- DIF(model, 'd', items2test = 1:3))

#### using items 4 to 15 as anchors
itemnames <- colnames(dat)
model_anchor <- multipleGroup(dat, model = 1, group = group,
  invariance = c(itemnames[4:15], 'free_means', 'free_var'))
anchor <- DIF(model_anchor, c('a1', 'd'), items2test = 1:3)
anchor

### drop down approach (freely estimating parameters accross groups) when
### specifying a highly constrained model with estimated latent parameters
model_constrained <- multipleGroup(dat, 1, group,
  invariance = c(colnames(dat), 'free_means', 'free_var'))
dropdown <- DIF(model_constrained, 'd', scheme = 'drop')
dropdown

### sequential searches using SABIC as the selection criteria
# starting from completely different models
model <- multipleGroup(dat, 1, group)
```

```
stepup <- DIF(model, c('a1', 'd'), scheme = 'add_sequential')
stepup

#step down procedure for highly constrained model
model <- multipleGroup(dat, 1, group, invariance = itemnames)
stepdown <- DIF(model, c('a1', 'd'), scheme = 'drop_sequential')
stepdown

## End(Not run)
```

`DiscreteClass-class` *Class "DiscreteClass"*

Description

Defines the object returned from `mdirt`.

Slots

`Call`**:** function call

`Data`**:** list of data, sometimes in different forms

`Options`**:** list of estimation options

`Fit`**:** a list of fit information

`Model`**:** a list of model-based information

`ParObjects`**:** a list of the S4 objects used during estimation

`OptimInfo`**:** a list of arguments from the optimization process

`Internals`**:** a list of internal arguments for secondary computations (inspecting this object is generally not required)

`vcov`**:** a matrix represented the asymtotic covariance matrix of the parameter estimates

`time`**:** a data.frame indicating the breakdown of computation times in seconds

Methods

print `signature(x = "DiscreteClass")`

show `signature(object = "DiscreteClass")`

anova `signature(object = "DiscreteClass")`

coef `signature(x = "DiscreteClass")`

summary `signature(object = "DiscreteClass")`

residuals `signature(object = "DiscreteClass")`

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

DTF *Differential test functioning statistics*

Description

Function performs various omnibus differential test functioning procedures on an object estimated with `multipleGroup()`. If the latent means/covariances are suspected to differ then the input object should contain a set of 'anchor' items to ensure that only differential test features are being detected rather than group differences. Returns signed (average area above and below) and unsigned (total area) statistics, with descriptives such as the percent average bias between group total scores for each statistic. If a grid of Theta values is passed, these can be evaluated as well to determine specific DTF location effects. For best results, the baseline model should contain a set of 'anchor' items and have freely estimated hyper-parameters in the focal groups. See `DIF` for details.

Usage

```
DTF(mod, draws = NULL, CI = 0.95, npts = 1000, theta_lim = c(-6, 6),
  Theta_nodes = NULL, plot = "none", auto.key = TRUE, ...)
```

Arguments

`mod`	a multipleGroup object which estimated only 2 groups
`draws`	a number indicating how many draws to take to form a suitable multiple imputation estimate of the expected test scores (usually 100 or more). Returns a list containing the imputation distribution and null hypothesis test for the sDTF statistic
`CI`	range of confidence interval when using draws input
`npts`	number of points to use in the integration. Default is 1000
`theta_lim`	lower and upper limits of the latent trait (theta) to be evaluated, and is used in conjunction with `npts`
`Theta_nodes`	an optional matrix of Theta values to be evaluated in the draws for the sDTF statistic. However, these values are not averaged across, and instead give the bootstrap confidence intervals at the respective Theta nodes. Useful when following up a large uDTF/sDTF statistic to determine where the difference between the test curves are large (while still accounting for sampling variability). Returns a matrix with observed variability
`plot`	a character vector indicating which plot to draw. Possible values are 'none', 'func' for the test score functions, and 'sDTF' for the evaluated sDTF values across the integration grid. Each plot is drawn with imputed confidence envelopes
`auto.key`	logical; automatically generate key in lattice plot?
`...`	additional arguments to be passed to `lattice` and `boot`

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

References

Chalmers, R. P., Counsell, A., and Flora, D. B. (2016). It might not make a big DIF: Improved Differential Test Functioning statistics that account for sampling variability. *Educational and Psychological Measurement, 76*, 114-140.

See Also

multipleGroup, DIF

Examples

```
## Not run:
set.seed(1234)
n <- 30
N <- 500

# only first 5 items as anchors
model <- 'F = 1-30
          CONSTRAINB = (1-5, a1), (1-5, d)'

a <- matrix(1, n)
d <- matrix(rnorm(n), n)
group <- c(rep('Group_1', N), rep('Group_2', N))

## -------------
# groups completely equal
dat1 <- simdata(a, d, N, itemtype = '2PL')
dat2 <- simdata(a, d, N, itemtype = '2PL')
dat <- rbind(dat1, dat2)
mod <- multipleGroup(dat, model, group=group, SE=TRUE,
                     invariance=c('free_means', 'free_var'))
plot(mod)

DTF(mod)
mirtCluster()
DTF(mod, draws = 1000) #95% C.I. for sDTF containing 0. uDTF is very small
DTF(mod, draws = 1000, plot='sDTF') #sDTF 95% C.I.'s across Theta always include 0

## -------------
## random slopes and intercepts for 15 items, and latent mean difference
##    (no systematic DTF should exist, but DIF will be present)
set.seed(1234)
dat1 <- simdata(a, d, N, itemtype = '2PL', mu=.50, sigma=matrix(1.5))
dat2 <- simdata(a + c(numeric(15), runif(n-15, -.2, .2)),
                d + c(numeric(15), runif(n-15, -.5, .5)), N, itemtype = '2PL')
dat <- rbind(dat1, dat2)
mod1 <- multipleGroup(dat, 1, group=group)
plot(mod1) #does not account for group differences! Need anchors
```

```
mod2 <- multipleGroup(dat, model, group=group, SE=TRUE,
                      invariance=c('free_means', 'free_var'))
plot(mod2)

#significant DIF in multiple items....
# DIF(mod2, which.par=c('a1', 'd'), items2test=16:30)
DTF(mod2)
DTF(mod2, draws=1000) #non-sig DTF due to item cancellation

## -------------
## systematic differing slopes and intercepts (clear DTF)
dat1 <- simdata(a, d, N, itemtype = '2PL', mu=.50, sigma=matrix(1.5))
dat2 <- simdata(a + c(numeric(15), rnorm(n-15, 1, .25)), d + c(numeric(15), rnorm(n-15, 1, .5)),
                N, itemtype = '2PL')
dat <- rbind(dat1, dat2)
mod3 <- multipleGroup(dat, model, group=group, SE=TRUE,
                      invariance=c('free_means', 'free_var'))
plot(mod3) #visable DTF happening

# DIF(mod3, c('a1', 'd'), items2test=16:30)
DTF(mod3) #unsigned bias. Signed bias indicates group 2 scores generally higher on average
DTF(mod3, draws=1000)
DTF(mod3, draws=1000, plot='func')
DTF(mod3, draws=1000, plot='sDTF') #multiple DTF areas along Theta

# evaluate specific values for sDTF
Theta_nodes <- matrix(seq(-6,6,length.out = 100))
sDTF <- DTF(mod3, Theta_nodes=Theta_nodes)
head(sDTF)
sDTF <- DTF(mod3, Theta_nodes=Theta_nodes, draws=100)
head(sDTF)


## End(Not run)
```

`empirical_ES` *Empirical effect sizes based on latent trait estimates*

Description

Computes effect size measures of differential item functioning and differential test/bundle functioning based on expected scores from Meade (2010). Item parameters from both reference and focal group are used in conjunction with focal group empirical theta estimates (and an assumed normally distributed theta) to compute expected scores.

Usage

```
empirical_ES(mod, Theta.focal = NULL, focal_items = 1L:extract.mirt(mod,
  "nitems"), DIF = TRUE, npts = 61, theta_lim = c(-6, 6), ref.group = 1,
```

```
plot = FALSE, par.strip.text = list(cex = 0.7),
par.settings = list(strip.background = list(col = "#9ECAE1"), strip.border =
list(col = "black")), ...)
```

Arguments

`mod`	a multipleGroup object which estimated only 2 groups
`Theta.focal`	an optional matrix of Theta values from the focal group to be evaluated. If not supplied the default values to `fscores` will be used in conjunction with the `...` arguments passed
`focal_items`	a numeric vector indicating which items to include the tests. The default uses all of the items. Selecting fewer items will result in tests of 'differential bundle functioning' when `DIF = FALSE`
`DIF`	logical; return a data.frame of item-level imputation properties? If `FALSE`, only DBF and DTF statistics will be reported
`npts`	number of points to use in the integration. Default is 61
`theta_lim`	lower and upper limits of the latent trait (theta) to be evaluated, and is used in conjunction with `npts`
`ref.group`	either 1 or 2 to indicate which group is considered the 'reference' group. Default is 1
`plot`	logical; plot expected scores of items/test where expected scores are computed using focal group thetas and both focal and reference group item parameters
`par.strip.text`	plotting argument passed to `lattice`
`par.settings`	plotting argument passed to `lattice`
`...`	additional arguments to be passed to `fscores` and `xyplot`

DIF

The default `DIF = TRUE` produces several effect sizes indices at the item level. Signed indices allow DIF favoring the focal group at one point on the theta distribution to cancel DIF favoring the reference group at another point on the theta distribution. Unsigned indices take the absolute value before summing or averaging, thus not allowing cancellation of DIF across theta.

SIDS Signed Item Difference in the Sample. The average difference in expected scores across the focal sample using both focal and reference group item parameters.

UIDS Unsigned Item Difference in the Sample. Same as SIDS except absolute value of expected scores is taken prior to averaging across the sample.

D-Max The maximum difference in expected scores in the sample.

ESSD Expected Score Standardized Difference. Cohen's D for difference in expected scores.

SIDN Signed Item Difference in a Normal distribution. Identical to SIDS but averaged across a normal distribution rather than the sample.

UIDN Unsigned Item Difference in a Normal distribution. Identical to UIDS but averaged across a normal distribution rather than the sample.

DBF/DTF

`DIF = FALSE` produces a series of test/bundle-level indices that are based on item-level indices.

STDS Signed Test Differences in the Sample. The sum of the SIDS across items.

UTDS Unsigned Test Differences in the Sample. The sum of the UIDS across items.

Stark's DTFR Stark's version of STDS using a normal distribution rather than sample estimated thetas.

UDTFR Unsigned Expected Test Scores Differences in the Sample. The difference in observed summed scale scores expected, on average, across a hypothetical focal group with anormally distributed theta, had DF been uniform innature for all items

UETSDS Unsigned Expected Test Score Differences in the Sample. The hypothetical difference inexpected scale scores that would have been present if scale-level DF had been uniform across respondents (i.e., always favoring the focal group).

UETSDN Identical to UETSDS but computed using anormal distribution.

Test D-Max Maximum expected test score differences in the sample.

ETSSD Expected Test Score Standardized Difference. Cohen's D for expected test scores.

Author(s)

Adam Meade and Phil Chalmers <rphilip.chalmers@gmail.com>

References

Meade, A. W. (2010). A taxonomy of effect size measures for the differential functioning of items and scales. *Journal of Applied Psychology, 95*, 728-743.

Examples

```
## Not run:

#no DIF
set.seed(12345)
a <- matrix(abs(rnorm(15,1,.3)), ncol=1)
d <- matrix(rnorm(15,0,.7),ncol=1)
itemtype <- rep('2PL', nrow(a))
N <- 1000
dataset1 <- simdata(a, d, N, itemtype)
dataset2 <- simdata(a, d, N, itemtype, mu = .1, sigma = matrix(1.5))
dat <- rbind(dataset1, dataset2)
group <- c(rep('Ref', N), rep('Focal', N))

mod <- multipleGroup(dat, 1, group = group,
   invariance = c(colnames(dat)[1:5], 'free_means', 'free_var'))
coef(mod, simplify=TRUE)

empirical_ES(mod)
empirical_ES(mod, DIF=FALSE)
empirical_ES(mod, DIF=FALSE, focal_items = 10:15)
```

```
empirical_ES(mod, plot=TRUE)
empirical_ES(mod, plot=TRUE, DIF=FALSE)

###---------------------------------------------
# DIF
set.seed(12345)
a1 <- a2 <- matrix(abs(rnorm(15,1,.3)), ncol=1)
d1 <- d2 <- matrix(rnorm(15,0,.7),ncol=1)
a2[10:15,] <- a2[10:15,] + rnorm(6, 0, .3)
d2[10:15,] <- d2[10:15,] + rnorm(6, 0, .3)
itemtype <- rep('dich', nrow(a1))
N <- 1000
dataset1 <- simdata(a1, d1, N, itemtype)
dataset2 <- simdata(a2, d2, N, itemtype, mu = .1, sigma = matrix(1.5))
dat <- rbind(dataset1, dataset2)
group <- c(rep('Ref', N), rep('Focal', N))

mod <- multipleGroup(dat, 1, group = group,
   invariance = c(colnames(dat)[1:5], 'free_means', 'free_var'))
coef(mod, simplify=TRUE)

empirical_ES(mod)
empirical_ES(mod, DIF = FALSE)
empirical_ES(mod, plot=TRUE)
empirical_ES(mod, plot=TRUE, DIF=FALSE)


## End(Not run)
```

`empirical_plot` *Function to generate empirical unidimensional item and test plots*

Description

Given a dataset containing item responses this function will construct empirical graphics using the observed responses to each item conditioned on the total score. When individual item plots are requested then the total score will be formed without the item of interest (i.e., the total score without that item).

Usage

```
empirical_plot(data, which.items = NULL, smooth = FALSE, formula = resp ~
  s(TS, k = 5), main = NULL, par.strip.text = list(cex = 0.7),
  boxplot = FALSE, par.settings = list(strip.background = list(col =
  "#9ECAE1"), strip.border = list(col = "black")), auto.key = list(space =
  "right"), ...)
```

Arguments

`data`	a `data.frame` or `matrix` of item responses (see `mirt` for typical input)
`which.items`	a numeric vector indicating which items to plot in a faceted image plot. If NULL then a empirical test plot will be constructed instead
`smooth`	logical; include a GAM smoother instead of the raw proportions? Default is FALSE
`formula`	formula used for the GAM smoother
`main`	the main title for the plot. If NULL an internal default will be used
`par.strip.text`	plotting argument passed to `lattice`
`boxplot`	logical; use a boxplot to display the marginal total score differences instead of scatter plots of proportions? Default is FALSE
`par.settings`	plotting argument passed to `lattice`
`auto.key`	plotting argument passed to `lattice`
`...`	additional arguments to be passed to `lattice` and `coef()`

Details

Note that these types of plots should only be used for unidimensional tests with monotonically increasing item response functions. If monotonicity should be true for all items, however, then these plots may serve as a visual diagnostic tool so long as the majority of items are indeed monotonic.

See Also

`itemplot`, `itemGAM`

Examples

```
## Not run:

SAT12[SAT12 == 8] <- NA
data <- key2binary(SAT12,
   key = c(1,4,5,2,3,1,2,1,3,1,2,4,2,1,5,3,4,4,1,4,3,3,4,1,3,5,1,3,1,5,4,5))

#test plot
empirical_plot(data)

#items 1, 2 and 5
empirical_plot(data, c(1, 2, 5))
empirical_plot(data, c(1, 2, 5), smooth = TRUE)
empirical_plot(data, c(1, 2, 5), boxplot = TRUE)

# replace weird looking items with unscored versions for diagnostics
empirical_plot(data, 32)
data[,32] <- SAT12[,32]
empirical_plot(data, 32)
empirical_plot(data, 32, smooth = TRUE)
```

```
## End(Not run)
```

---

empirical_rxx *Function to calculate the empirical (marginal) reliability*

Description

Given secondary latent trait estimates and their associated standard errors returned from `fscores`, compute the empirical reliability.

Usage

```
empirical_rxx(Theta_SE)
```

Arguments

`Theta_SE`	a matrix of latent trait estimates returned from `fscores` with the options `full.scores = TRUE` and `full.scores.SE = TRUE`

Author(s)

Phil Chalmers `<rphilip.chalmers@gmail.com>`

See Also

`fscores`, `marginal_rxx`

Examples

```
## Not run:

dat <- expand.table(deAyala)
mod <- mirt(dat, 1)

theta_se <- fscores(mod, full.scores.SE = TRUE)
empirical_rxx(theta_se)

theta_se <- fscores(mod, full.scores.SE = TRUE, method = 'ML')
empirical_rxx(theta_se)


## End(Not run)
```

expand.table *Expand summary table of patterns and frequencies*

Description

The `expand.table` function expands a summary table of unique response patterns to a full sized data-set. The response frequencies must be on the rightmost column of the input data.

Usage

```
expand.table(tabdata, sample = FALSE)
```

Arguments

`tabdata`	An object of class `data.frame` or `matrix` with the unique response patterns and the number of frequencies in the rightmost column
`sample`	logical; randomly switch the rows in the expanded table? This does not change the expanded data, only the row locations

Value

Returns a numeric matrix with all the response patterns.

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

Examples

```
## Not run:
data(LSAT7)
head(LSAT7)
LSAT7full <- expand.table(LSAT7)
head(LSAT7full)

LSAT7full <- expand.table(LSAT7, sample = TRUE)
head(LSAT7full)


## End(Not run)
```

expected.item *Function to calculate expected value of item*

Description

Given an internal mirt object extracted from an estimated model compute the expected value for an item given the ability parameter(s).

Usage

```
expected.item(x, Theta, min = 0)
```

Arguments

x	an extracted internal mirt object containing item information (see extract.item)
Theta	a vector (unidimensional) or matrix (multidimensional) of latent trait values
min	a constant value added to the expected values indicating the lowest theoretical category. Default is 0

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

See Also

extract.item, expected.test

Examples

```
## Not run:
mod <- mirt(Science, 1)
extr.2 <- extract.item(mod, 2)
Theta <- matrix(seq(-6,6, length.out=200))
expected <- expected.item(extr.2, Theta, min(Science[,1])) #min() of first item
head(data.frame(expected, Theta=Theta))


## End(Not run)
```

`expected.test` *Function to calculate expected test score*

Description

Given an estimated model compute the expected test score. Returns the expected values in the same form as the data used to estimate the model.

Usage

```
expected.test(x, Theta, group = NULL, mins = TRUE, individual = FALSE,
  which.items = NULL)
```

Arguments

`x`	an estimated mirt object
`Theta`	a matrix of latent trait values
`group`	a number signifying which group the item should be extracted from (applies to 'MultipleGroupClass' objects only)
`mins`	logical; include the minimum value constants in the dataset. If FALSE, the expected values for each item are determined from the scoring 0:(ncat-1)
`individual`	logical; return tracelines for individual items?
`which.items`	an integer vector indicating which items to include in the expected test score. Default uses all possible items

See Also

`expected.item`

Examples

```
## Not run:
dat <- expand.table(deAyala)
model <- 'F = 1-5
          CONSTRAIN = (1-5, a1)'
mod <- mirt(dat, model)

Theta <- matrix(seq(-6,6,.01))
tscore <- expected.test(mod, Theta)
tail(cbind(Theta, tscore))

# use only first two items (i.e., a bundle)
bscore <- expected.test(mod, Theta, which.items = 1:2)
tail(cbind(Theta, bscore))


## End(Not run)
```

extract.group *Extract a group from a multiple group mirt object*

Description

Extract a single group from an object defined by multipleGroup.

Usage

```
extract.group(x, group)
```

Arguments

x	mirt model of class 'MultipleGroupClass'
group	a number signifying which group should be extracted

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

See Also

extract.item, extract.mirt

Examples

```
## Not run:
set.seed(12345)
a <- matrix(abs(rnorm(15,1,.3)), ncol=1)
d <- matrix(rnorm(15,0,.7),ncol=1)
itemtype <- rep('2PL', nrow(a))
N <- 1000
dataset1 <- simdata(a, d, N, itemtype)
dataset2 <- simdata(a, d, N, itemtype, mu = .1, sigma = matrix(1.5))
dat <- rbind(dataset1, dataset2)
group <- c(rep('D1', N), rep('D2', N))
models <- 'F1 = 1-15'

mod_configural <- multipleGroup(dat, models, group = group)
group.1 <- extract.group(mod_configural, 1) #extract first group
summary(group.1)
plot(group.1)

## End(Not run)
```

extract.item *Extract an item object from mirt objects*

Description

Extract the internal mirt objects from any estimated model.

Usage

```
extract.item(x, item, group = NULL, drop.zeros = FALSE)
```

Arguments

x	mirt model of class 'SingleGroupClass' or 'MultipleGroupClass'
item	a number or character signifying which item to extract
group	a number signifying which group the item should be extracted from (applies to 'MultipleGroupClass' only)
drop.zeros	logical; drop slope values that are numerically close to zero to reduce dimensionality? Useful in objects returned from bfactor or other confirmatory models that contain several zero slopes

See Also

extract.group, extract.mirt

Examples

```
## Not run:
mod <- mirt(Science, 1)
extr.1 <- extract.item(mod, 1)

## End(Not run)
```

extract.mirt *Extract various elements from estimated model objects*

Description

A generic function to extract the internal objects from estimated models.

Usage

```
extract.mirt(x, what)
```

Arguments

`x`	mirt model of class 'SingleGroupClass', 'MultipleGroupClass', 'MixedClass' or 'DiscreteGroupClass'
`what`	a string indicating what to extract

Details

Objects which can be extracted from mirt objects include:

logLik observed log-likelihood

logPrior log term contributed by prior parameter distributions

G2 goodness of fit statistic

df degrees of freedom

p p-value for G2 statistic

RMSEA root mean-square error of approximation based on G2

CFI CFI fit statistic

TLI TLI fit statistic

AIC AIC

AICc corrected AIC

BIC BIC

SABIC sample size adjusted BIC

DIC DIC

F unrotated standardized loadings matrix

h2 factor communality estimates

LLhistory EM log-likelihood history

tabdata a tabular version of the raw response data input. Frequencies are stored in `freq`

freq frequencies associated with `tabdata`

K an integer vector indicating the number of unique elements for each item

mins an integer vector indicating the lowest category found in the input `data`

model input model syntax

method estimation method used

itemtype a vector of item types for each respective item (e.g., 'graded', '2PL', etc)

itemnames a vector of item names from the input data

data raw input data of item responses

covdata raw input data of data used as covariates

tabdatalong similar to `tabdata`, however the responses have been transformed into dummy coded variables

fulldatalong analogous to `tabdatafull`, but for the raw input data instead of the tabulated frequencies

exp_resp expected probability of the unique response patterns

converged a logical value indicating whether the model terminated within the convergence criteria

iterations number of iterations it took to reach the convergence criteria

nest number of freely estimated parameters

parvec vector containing uniquely estimated parameters

vcov parameter covariance matrix (associated with parvec)

condnum the condition number of the Hessian (if computed). Otherwise NA

constrain a list of item parameter constraints to indicate which item parameters were equal during estimation

Prior prior density distribution for the latent traits

key if supplied, the data scoring key

nfact number of latent traits/factors

nitems number of items

ngroups number of groups

groupNames character vector of unique group names

group a character vector indicating the group membership

secondordertest a logical indicating whether the model passed the second-order test based on the Hessian matrix. Indicates whether model is a potential local maximum solution

SEMconv logical; check whether the supplimented EM information matrix converged. Will be `NA` if not applicable

time estimation time, broken into different sections

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

See Also

`extract.group`, `extract.item`, `mod2values`

Examples

```
## Not run:
mod <- mirt(Science, 1)

extract.mirt(mod, 'logLik')
extract.mirt(mod, 'F')

#multiple group model
grp <- rep(c('G1', 'G2'), each = nrow(Science)/2)
mod2 <- multipleGroup(Science, 1, grp)

grp1 <- extract.group(mod2, 1) #extract single group model
extract.mirt(mod2, 'parvec')
```

```
extract.mirt(grp1, 'parvec')

## End(Not run)
```

fixef *Compute latent regression fixed effect expected values*

Description

Create expected values for fixed effects parameters in latent regression models.

Usage

```
fixef(x)
```

Arguments

x	an estimated model object from the mixedmirt or mirt function

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

See Also

mirt, mixedmirt

Examples

```
## Not run:

#simulate data
set.seed(1234)
N <- 1000

# covariates
X1 <- rnorm(N); X2 <- rnorm(N)
covdata <- data.frame(X1, X2)
Theta <- matrix(0.5 * X1 + -1 * X2 + rnorm(N, sd = 0.5))

#items and response data
a <- matrix(1, 20); d <- matrix(rnorm(20))
dat <- simdata(a, d, 1000, itemtype = '2PL', Theta=Theta)

#conditional model using X1 and X2 as predictors of Theta
mod1 <- mirt(dat, 1, 'Rasch', covdata=covdata, formula = ~ X1 + X2)

#latent regression fixed effects (i.e., expected values)
fe <- fixef(mod1)
```

```
head(fe)

# with mixedmirt()
mod1b <- mixedmirt(dat, covdata, 1, lr.fixed = ~ X1 + X2, fixed = ~ 0 + items)
fe2 <- fixef(mod1b)
head(fe2)


## End(Not run)
```

---

fscores *Compute factor score estimates (a.k.a, ability estimates, latent trait estimates, etc)*

Description

Computes MAP, EAP, ML (Embretson & Reise, 2000), EAP for sum-scores (Thissen et al., 1995), or WLE (Warm, 1989) factor scores with a multivariate normal prior distribution using equally spaced quadrature. EAP scores for models with more than three factors are generally not recommended since the integration grid becomes very large, resulting in slower estimation and less precision if the `quadpts` are too low. Therefore, MAP scores should be used instead of EAP scores for higher dimensional models. Multiple imputation variants are possible for each estimator if a parameter information matrix was computed, which are useful if the sample size/number of items were small. As well, if the model contained latent regression predictors this information will be used in computing MAP and EAP estimates (for these models, `full.scores=TRUE` will always be used). Finally, plausible value imputation is also available, and will also account for latent regression predictor effects.

Usage

```
fscores(object, method = "EAP", full.scores = TRUE, rotate = "oblimin",
  Target = NULL, response.pattern = NULL, plausible.draws = 0,
  plausible.type = "normal", quadpts = NULL, returnER = FALSE,
  return.acov = FALSE, mean = NULL, cov = NULL, verbose = TRUE,
  full.scores.SE = FALSE, theta_lim = c(-6, 6), MI = 0, QMC = FALSE,
  custom_den = NULL, custom_theta = NULL, min_expected = 1,
  converge_info = FALSE, ...)
```

Arguments

`object` a computed model object of class `SingleGroupClass`, `MultipleGroupClass`, or `DiscreteClass`

`method` type of factor score estimation method. Can be expected a-posteriori (`"EAP"`), Bayes modal (`"MAP"`), weighted likelihood estimation (`"WLE"`), maximum likelihood (`"ML"`), or expected a-posteriori for sum scores (`"EAPsum"`). Can also be `"plausible"` for a single plausible value imputation for each case, and this is equivalent to setting `plausible.draws = 1` `full.scores` if `FALSE` then a summary table with factor scores for each unique pattern is displayed. Otherwise, a matrix of factor scores for each response pattern in the data is returned (default)

`rotate` prior rotation to be used when estimating the factor scores. See `summary-method` for details. If the object is not an exploratory model then this argument is ignored

`Target` target rotation; see `summary-method` for details

`response.pattern`
an optional argument used to calculate the factor scores and standard errors for a given response vector or matrix/data.frame

`plausible.draws`
number of plausible values to draw for future researchers to perform secondary analyses of the latent trait scores. Typically used in conjunction with latent regression predictors (see `mirt` for details), but can also be generated when no predictor variables were modeled. If `plausible.draws` is greater than 0 a list of plausible values will be returned

`plausible.type` type of plausible values to obtain. Can be either `'normal'` (default) to use a normal approximation based on the ACOV matrix, or `'MH'` to obtain Metropolis-Hastings samples from the posterior (silently passes object to `mirt`, therefore arguemnts like `technical` can be supplied to increase the number of burn-in draws and discarded samples)

`quadpts` number of quadratures to use per dimension. If not specified, a suitable one will be created which decreases as the number of dimensions increases (and therefore for estimates such as EAP, will be less accurate). This is determined from the switch statement `quadpts <- switch(as.character(nfact), '1'=61, '2'=31, '3'=15, '4'=9, '5`

`returnER` logical; return empirical reliability (also known as marginal reliability) estimates as a numeric values?

`return.acov` logical; return a list containing covariance matrices instead of factors scores? `impute = TRUE` not supported with this option

`mean` a vector for custom latent variable means. If NULL, the default for 'group' values from the computed mirt object will be used

`cov` a custom matrix of the latent variable covariance matrix. If NULL, the default for 'group' values from the computed mirt object will be used

`verbose` logical; print verbose output messages?

`full.scores.SE` logical; when `full.scores == TRUE`, also return the standard errors associated with each respondent? Default is `FALSE`

`theta_lim` lower and upper range to evaluate latent trait integral for each dimension. If omitted, a range will be generated automatically based on the number of dimensions

`MI` a number indicating how many multiple imputation draws to perform. Default is 0, indicating that no MI draws will be performed

`QMC` logical; use quasi-Monte Carlo integration? If `quadpts` is omitted the default number of nodes is 5000 custom_den a function used to define the integration density (if required). The NULL default assumes that the multivariate normal distribution with the 'GroupPars' hyper-parameters are used. At the minimum must be of the form:

`function(Theta, ...)`

where Theta is a matrix of latent trait values (will be a grid of values if `method == 'EAPsum'` or `method == 'EAP'`, otherwise Theta will have only 1 row). Additional arguments may included and are caught through the `fscores(...)` input. The function *must* return a numeric vector of density weights (one for each row in Theta)

custom_theta a matrix of custom integration nodes to use instead of the default, where each column corresponds to the respective dimension in the model

min_expected when computing goodness of fit tests when `method = 'EAPsum'`, this value is used to collapse across the conditioned total scores until the expected values are greater than this value. Note that this only affect the goodness of fit tests and not the returned EAP for sum scores table

converge_info logical; include a column in the return objects containing a logical for each response pattern indicating whether a maximum value was found (not relavent non-iterative methods, such as EAP and EAPsum). Value is a reflection of the `code` element from `nlm` (e.g., 1 indicates convergence)

... additional arguments to be passed to `nlm`

Details

The function will return either a table with the computed scores and standard errors, the original data matrix with scores appended to the rightmost column, or the scores only. By default the latent means and covariances are determined from the estimated object, though these can be overwritten. Iterative estimation methods can be estimated in parallel to decrease estimation times if a `mirtCluster` object is available.

If the input object is a discrete latent class object estimated from `mdirt` then the returned results will be with respect to the posterior classification for each individual. The method inputs for `'DiscreteClass'` objects may only be `'EAP'`, for posterior classification of each response pattern, or `'EAPsum'` for posterior classification based on the raw sum-score.

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

References

Embretson, S. E. & Reise, S. P. (2000). Item Response Theory for Psychologists. Erlbaum.

Thissen, D., Pommerich, M., Billeaud, K., & Williams, V. S. L. (1995). Item Response Theory for Scores on Tests Including Polytomous Items with Ordered Responses. *Applied Psychological Measurement, 19*, 39-49.

Warm, T. A. (1989). Weighted likelihood estimation of ability in item response theory. *Psychometrika, 54*, 427-450.

See Also

averageMI

Examples

```
## Not run:

mod <- mirt(Science, 1)
tabscores <- fscores(mod, full.scores = FALSE)
head(tabscores)
fullscores <- fscores(mod)
fullscores_with_SE <- fscores(mod, full.scores.SE=TRUE)
head(fullscores)
head(fullscores_with_SE)

#change method argument to use MAP estimates
fullscores <- fscores(mod, method='MAP')
head(fullscores)

#calculate MAP for a given response vector
fscores(mod, method='MAP', response.pattern = c(1,2,3,4))
#or matrix
fscores(mod, method='MAP', response.pattern = rbind(c(1,2,3,4), c(2,2,1,3)))

#use custom latent variable properties (diffuse prior for MAP is very close to ML)
fscores(mod, method='MAP', cov = matrix(1000), full.scores = FALSE)
fscores(mod, method='ML', full.scores = FALSE)

# EAPsum table of values based on total scores
fscores(mod, method = 'EAPsum', full.scores = FALSE)

#WLE estimation, run in parallel using available cores
mirtCluster()
head(fscores(mod, method='WLE', full.scores = FALSE))

#multiple imputation using 30 draws for EAP scores. Requires information matrix
mod <- mirt(Science, 1, SE=TRUE)
fs <- fscores(mod, MI = 30)
head(fs)

# plausible values for future work
pv <- fscores(mod, plausible.draws = 5)
lapply(pv, function(x) c(mean=mean(x), var=var(x), min=min(x), max=max(x)))

## define a custom_den function. EAP with a uniform prior between -3 and 3
fun <- function(Theta, ...) as.numeric(dunif(Theta, min = -3, max = 3))
head(fscores(mod, custom_den = fun))

# custom MAP prior: standard truncated normal between 5 and -2
library(msm)
# need the :: scope for parallel to see the function (not require if no mirtCluster() defined)
```

```
fun <- function(Theta, ...) msm::dtnorm(Theta, mean = 0, sd = 1, lower = -2, upper = 5)
head(fscores(mod, custom_den = fun, method = 'MAP', full.scores = FALSE))


## End(Not run)
```

---

imputeMissing *Imputing plausible data for missing values*

Description

Given an estimated model from any of mirt's model fitting functions and an estimate of the latent trait, impute plausible missing data values. Returns the original data in a `data.frame` without any NA values. If a list of `Theta` values is supplied then a list of complete datasets is returned instead.

Usage

```
imputeMissing(x, Theta, warn = TRUE, ...)
```

Arguments

`x`	an estimated model x from the mirt package
`Theta`	a matrix containing the estimates of the latent trait scores (e.g., via `fscores`)
`warn`	logical; print warning messages?
`...`	additional arguments to pass

Author(s)

Phil Chalmers `<rphilip.chalmers@gmail.com>`

Examples

```
## Not run:
dat <- expand.table(LSAT7)
(original <- mirt(dat, 1))
NAperson <- sample(1:nrow(dat), 20, replace = TRUE)
NAitem <- sample(1:ncol(dat), 20, replace = TRUE)
for(i in 1:20)
    dat[NAperson[i], NAitem[i]] <- NA
(mod <- mirt(dat, 1))
scores <- fscores(mod, method = 'MAP')

#re-estimate imputed dataset (good to do this multiple times and average over)
fulldata <- imputeMissing(mod, scores)
(fullmod <- mirt(fulldata, 1))

#with multipleGroup
set.seed(1)
```

```
group <- sample(c('group1', 'group2'), 1000, TRUE)
mod2 <- multipleGroup(dat, 1, group, TOL=1e-2)
fs <- fscores(mod2)
fulldata2 <- imputeMissing(mod2, fs)


## End(Not run)
```

---

itemfit *Item fit statistics*

Description

Computes item-fit statistics for a variety of unidimensional and multidimensional models. Poorly fitting items should be inspected with the empirical plots/tables for unidimensional models, otherwise `itemGAM` can be used to diagnose where the functional form of the IRT model was misspecified, or models can be refit using more flexible semi-parametric response models (e.g., `itemtype = 'spline'`).

Usage

```
itemfit(x, fit_stats = "S_X2", which.items = 1:extract.mirt(x, "nitems"),
  group.bins = 10, group.size = NA, group.fun = mean, mincell = 1,
  mincell.X2 = 2, S_X2.tables = FALSE, pv_draws = 30, boot = 1000,
  boot_dfapprox = 200, ETrange = c(-2, 2), ETpoints = 11,
  empirical.plot = NULL, empirical.CI = 0.95, empirical.table = NULL,
  method = "EAP", Theta = NULL, impute = 0, par.strip.text = list(cex =
  0.7), par.settings = list(strip.background = list(col = "#9ECAE1"),
  strip.border = list(col = "black")), ...)
```

Arguments

`x` a computed model object of class `SingleGroupClass`, `MultipleGroupClass`, or `DiscreteClass`

`fit_stats` a character vector indicating which fit statistics should be computed. Supported inputs are:

- `'S_X2'` : Orlando and Thissen (2000, 2003) and Kang and Chen's (2007) signed chi-squared test (default)
- `'Zh'` : Drasgow, Levine, & Williams (1985) Zh
- `'X2'` : Bock's (1972) chi-squared method. The default inputs compute Yen's (1981) Q1 variant of the X2 statistic (i.e., uses a fixed `group.bins = 10`). However, Bock's group-size variable median-based method can be computed by passing `group.fun = median` and modifying the `group.size` input to the desired number of bins
- `'G2'` : McKinley & Mills (1985) G2 statistic (similar method to Q1, but with the likelihood-ratio test).
- `'PV_Q1'` : Chalmers and Ng's (forthcoming) plausible-value variant of the Q1 statistic.

- 'PV_Q1*' : Chalmers and Ng's (forthcoming) plausible-value variant of the Q1 statistic that uses parametric boostrapping to obtain a suitable empirical distribution.
- 'X2*' : Stone's (2000) fit statistics that require parametric bootstrapping
- 'X2*_df' : Stone's (2000) fit statistics that require parametric bootstrapping to obtain scaled versions of the X2* and degrees of freedom
- 'infit' : (Unidimensional Rasch model only) compute the infit and outfit statistics. Ignored if models are not from the Rasch family

Note that 'infit', 'S_X2', and 'Zh' cannot be computed when there are missing response data (i.e., will require multiple-imputation techniques).

`which.items` an integer vector indicating which items to test for fit. Default tests all possible items

`group.bins` the number of bins to use for X2 and G2. For example, setting `group.bins = 10` will will compute Yen's (1981) Q1 statistic when `'X2'` is requested

`group.size` approximate size of each group to be used in calculating the χ^2 statistic. The default `NA` disables this command and instead uses the `group.bins` input to try and construct equally sized bins

`group.fun` function used when `'X2'` or `'G2'` are computed. Determines the central tendancy measure within each partitioned group. E.g., setting `group.fun = median` will obtain the median of each respective ability estimate in each subgroup (this is what was used by Bock, 1972)

`mincell` the minimum expected cell size to be used in the S-X2 computations. Tables will be collapsed across items first if polytomous, and then across scores if necessary

`mincell.X2` the minimum expected cell size to be used in the X2 computations. Tables will be collapsed if polytomous, however if this condition can not be met then the group block will be ommited in the computations

`S_X2.tables` logical; return the tables in a list format used to compute the S-X2 stats?

`pv_draws` number of plausible-value draws to obtain for PV_Q1 and PV_Q1*

`boot` number of parametric bootstrap samples to create for PV_Q1* and X2*

`boot_dfapprox` number of parametric bootstrap samples to create for the X2*_df statistic to approximate the scaling factor for X2* as well as the scaled degrees of freedom estimates

`ETrange` rangone of integration nodes for Stone's X2* statistic

`ETpoints` number of integration nodes to use for Stone's X2* statistic

`empirical.plot` a single numeric value or character of the item name indicating which item to plot (via `itemplot`) and overlay with the empirical θ groupings (see `empirical.CI`). Useful for plotting the expected bins based on the `'X2'` or `'G2'` method

`empirical.CI` a numeric value indicating the width of the empirical confidence interval ranging between 0 and 1 (default of 0 plots not interval). For example, a 95 interval would be plotted when `empirical.CI = .95`. Only applicable to dichotomous items

`empirical.table` a single numeric value or character of the item name indicating which item table of expected values should be returned. Useful for visualizing the expected bins based on the `'X2'` or `'G2'` method

`method`	type of factor score estimation method. See `fscores` for more detail
`Theta`	a matrix of factor scores for each person used for statistics that require empirical estimates. If supplied, arguments typically passed to `fscores()` will be ignored and these values will be used instead. Also required when estimating statistics with missing data via imputation
`impute`	a number indicating how many imputations to perform (passed to `imputeMissing`) when there are missing data present. Will return a data.frame object with the mean estimates of the stats and their imputed standard deviations
`par.strip.text`	plotting argument passed to `lattice`
`par.settings`	plotting argument passed to `lattice`
`...`	additional arguments to be passed to `fscores()` and `lattice`

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

References

Bock, R. D. (1972). Estimating item parameters and latent ability when responses are scored in two or more nominal categories. Psychometrika, 37, 29-51.

Chalmers, R. P. & Ng, V. (forthcoming). Plausible-Value Imputation Statistics for Detecting Item Misfit. *Applied Psychological Measurement*.

Drasgow, F., Levine, M. V., & Williams, E. A. (1985). Appropriateness measurement with polychotomous item response models and standardized indices. *British Journal of Mathematical and Statistical Psychology, 38*, 67-86.

Kang, T. & Chen, Troy, T. (2007). An investigation of the performance of the generalized S-X2 item-fit index for polytomous IRT models. ACT

McKinley, R., & Mills, C. (1985). A comparison of several goodness-of-fit statistics. Applied Psychological Measurement, 9, 49-57.

Orlando, M. & Thissen, D. (2000). Likelihood-based item fit indices for dichotomous item response theory models. *Applied Psychological Measurement, 24*, 50-64.

Reise, S. P. (1990). A comparison of item- and person-fit methods of assessing model-data fit in IRT. *Applied Psychological Measurement, 14*, 127-137.

Stone, C. A. (2000). Monte Carlo Based Null Distribution for an Alternative Goodness-of-Fit Test Statistics in IRT Models. *Journal of Educational Measurement, 37*, 58-75.

Wright B. D. & Masters, G. N. (1982). *Rating scale analysis*. MESA Press.

Yen, W. M. (1981). Using simulation results to choose a latent trait model. *Applied Psychological Measurement, 5*, 245-262.

See Also

`personfit`, `itemGAM`

Examples

```
## Not run:

P <- function(Theta){exp(Theta^2 * 1.2 - 1) / (1 + exp(Theta^2 * 1.2 - 1))}

#make some data
set.seed(1234)
a <- matrix(rlnorm(20, meanlog=0, sdlog = .1),ncol=1)
d <- matrix(rnorm(20),ncol=1)
Theta <- matrix(rnorm(2000))
items <- rep('2PL', 20)
ps <- P(Theta)
baditem <- numeric(2000)
for(i in 1:2000)
   baditem[i] <- sample(c(0,1), 1, prob = c(1-ps[i], ps[i]))
data <- cbind(simdata(a,d, 2000, items, Theta=Theta), baditem=baditem)

x <- mirt(data, 1)
raschfit <- mirt(data, 1, itemtype='Rasch')
fit <- itemfit(x)
fit

itemfit(x)
itemfit(x, 'X2') # just X2
itemfit(x, c('S_X2', 'X2')) #both S_X2 and X2
itemfit(x, group.bins=15, empirical.plot = 1) #empirical item plot with 15 points
itemfit(x, group.bins=15, empirical.plot = 21)

# PV and X2* statistics (parametric bootstrap stats not run to save time)
itemfit(x, 'PV_Q1')

# mirtCluster() # improve speed of bootstrap samples by running in parallel
# itemfit(x, 'PV_Q1*')
# itemfit(x, 'X2*') # Stone's 1993 statistic
# itemfit(x, 'X2*_df') # Stone's 2000 scaled statistic with df estimate

#empirical tables
itemfit(x, empirical.table=1)
itemfit(x, empirical.table=21)

#infit/outfit statistics. method='ML' agrees better with eRm package
itemfit(raschfit, 'infit', method = 'ML') #infit and outfit stats

#same as above, but inputting ML estimates instead
Theta <- fscores(raschfit, method = 'ML')
itemfit(raschfit, 'infit', Theta=Theta)

# fit a new more flexible model for the mis-fitting item
itemtype <- c(rep('2PL', 20), 'spline')
x2 <- mirt(data, 1, itemtype=itemtype)
itemfit(x2)
```

```
itemplot(x2, 21)
anova(x2, x)

#--------------------------------------------------------------

#similar example to Kang and Chen 2007
a <- matrix(c(.8,.4,.7, .8, .4, .7, 1, 1, 1, 1))
d <- matrix(rep(c(2.0,0.0,-1,-1.5),10), ncol=4, byrow=TRUE)
dat <- simdata(a,d,2000, itemtype = rep('graded', 10))
head(dat)

mod <- mirt(dat, 1)
itemfit(mod)
itemfit(mod, 'X2') #pretty much useless given inflated Type I error rates
itemfit(mod, empirical.plot = 1)

# collapsed tables (see mincell.X2) for X2 and G2
itemfit(mod, empirical.table = 1)

mod2 <- mirt(dat, 1, 'Rasch')
itemfit(mod2, 'infit')

#massive list of tables
tables <- itemfit(mod, S_X2.tables = TRUE)

#observed and expected total score patterns for item 1 (post collapsing)
tables$O[[1]]
tables$E[[1]]

# fit stats with missing data (run in parallel using all cores)
data[sample(1:prod(dim(data)), 500)] <- NA
raschfit <- mirt(data, 1, itemtype='Rasch')

mirtCluster() # run in parallel
itemfit(raschfit, c('S_X2', 'infit'), impute = 10)

#alternative route: use only valid data, and create a model with the previous parameter estimates
data2 <- na.omit(data)
raschfit2 <- mirt(data2, 1, itemtype = 'Rasch', pars=mod2values(raschfit), TOL=NaN)
itemfit(raschfit2, 'infit')

# note that X2, G2, PV-Q1, and X2* do not require complete datasets
itemfit(raschfit, c('X2', 'G2'))
itemfit(raschfit, empirical.plot=1)
itemfit(raschfit, empirical.table=1)


## End(Not run)
```

`itemGAM` *Parametric smoothed regression lines for item response probability functions*

Description

This function uses a generalized additive model (GAM) to estimate response curves for items that do not seem to fit well in a given model. Using a stable axillary model, traceline functions for poorly fitting dichotomous or polytomous items can be inspected using point estimates (or plausible values) of the latent trait. Plots of the tracelines and their associated standard errors are available to help interpret the misfit. This function may also be useful when adding new items to an existing, well estiablished set of items, especially when the parametric form of the items under investigation are unknown.

Usage

```
itemGAM(item, Theta, formula = resp ~ s(Theta, k = 10), CI = 0.95,
  theta_lim = c(-3, 3), return.models = FALSE, ...)

## S3 method for class 'itemGAM'
plot(x, y = NULL, par.strip.text = list(cex = 0.7),
  par.settings = list(strip.background = list(col = "#9ECAE1"), strip.border =
  list(col = "black")), auto.key = list(space = "right"), ...)
```

Arguments

`item`	a single poorly fitting item to be investigated. Can be a vector or matrix
`Theta`	a list or matrix of latent trait estimates typically returned from `fscores`
`formula`	an R formula to be passed to the `gam` function. Default fits a spline model with 10 nodes. For multidimensional models, the traits are assigned the names 'Theta1', 'Theta2', ..., 'ThetaN'
`CI`	a number ranging from 0 to 1 indicating the confidence interval range. Default provides the 95 percent interval
`theta_lim`	range of latent trait scores to be evaluated
`return.models`	logical; return a list of GAM models for each category? Useful when the GAMs should be inspected directly, but also when fitting multidimensional models (this is set to TRUE automatically for multidimensional models)
`...`	additional arguments to be passed to `gam` or `lattice`
`x`	an object of class 'itemGAM'
`y`	a `NULL` value ignored by the plotting function
`par.strip.text`	plotting argument passed to `lattice`
`par.settings`	plotting argument passed to `lattice`
`auto.key`	plotting argument passed to `lattice`

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

See Also

itemfit

Examples

```
## Not run:
set.seed(10)
N <- 1000
J <- 30

a <- matrix(1, J)
d <- matrix(rnorm(J))
Theta <- matrix(rnorm(N, 0, 1.5))
dat <- simdata(a, d, N, itemtype = '2PL', Theta=Theta)

# make a bad item
ps <- exp(Theta^2 + Theta) / (1 + exp(Theta^2 + Theta))
item1 <- sapply(ps, function(x) sample(c(0,1), size = 1, prob = c(1-x, x)))

ps2 <- exp(2 * Theta^2 + Theta + .5 * Theta^3) / (1 + exp(2 * Theta^2 + Theta + .5 * Theta^3))
item2 <- sapply(ps2, function(x) sample(c(0,1), size = 1, prob = c(1-x, x)))

#' # how the actual item looks in the population
plot(Theta, ps, ylim = c(0,1))
plot(Theta, ps2, ylim = c(0,1))

baditems <- cbind(item1, item2)
newdat <- cbind(dat, baditems)

badmod <- mirt(newdat, 1)
itemfit(badmod) #clearly a bad fit for the last two items
mod <- mirt(dat, 1) #fit a model that does not contain the bad items
itemfit(mod)

#### Pure non-parametric way of investigating the items
library(KernSmoothIRT)
ks <- ksIRT(newdat, rep(1, ncol(newdat)), 1)
plot(ks, item=c(1,31,32))
par(ask=FALSE)

# Using point estimates from the model
Theta <- fscores(mod)
IG0 <- itemGAM(dat[,1], Theta) #good item
IG1 <- itemGAM(baditems[,1], Theta)
IG2 <- itemGAM(baditems[,2], Theta)
plot(IG0)
plot(IG1)
```

```
plot(IG2)

# same as above, but with plausible values to obtain the standard errors
ThetaPV <- fscores(mod, plausible.draws=10)
IG0 <- itemGAM(dat[,1], ThetaPV) #good item
IG1 <- itemGAM(baditems[,1], ThetaPV)
IG2 <- itemGAM(baditems[,2], ThetaPV)
plot(IG0)
plot(IG1)
plot(IG2)

## for polytomous test items
SAT12[SAT12 == 8] <- NA
dat <- key2binary(SAT12,
               key = c(1,4,5,2,3,1,2,1,3,1,2,4,2,1,5,3,4,4,1,4,3,3,4,1,3,5,1,3,1,5,4,5))
dat <- dat[,-32]
mod <- mirt(dat, 1)

# Kernal smoothing is very sensitive to which category is selected as 'correct'
# 5th category as correct
ks <- ksIRT(cbind(dat, SAT12[,32]), c(rep(1, 31), 5), 1)
plot(ks, items = c(1,2,32))

# 3rd category as correct
ks <- ksIRT(cbind(dat, SAT12[,32]), c(rep(1, 31), 3), 1)
plot(ks, items = c(1,2,32))

# splines approach
Theta <- fscores(mod)
IG <- itemGAM(SAT12[,32], Theta)
plot(IG)

ThetaPV <- fscores(mod, plausible.draws=10)
IG2 <- itemGAM(SAT12[,32], ThetaPV)
plot(IG2)

# assuming a simple increasing parametric form (like in a standard IRT model)
IG3 <- itemGAM(SAT12[,32], Theta, formula = resp ~ Theta)
plot(IG3)
IG3 <- itemGAM(SAT12[,32], ThetaPV, formula = resp ~ Theta)
plot(IG3)

### multidimensional example by returning the GAM objects
mod2 <- mirt(dat, 2)
Theta <- fscores(mod2)
IG4 <- itemGAM(SAT12[,32], Theta, formula = resp ~ s(Theta1, k=10) + s(Theta2, k=10),
   return.models=TRUE)
names(IG4)
plot(IG4[[1L]], main = 'Category 1')
plot(IG4[[2L]], main = 'Category 2')
plot(IG4[[3L]], main = 'Category 3')
```

```
## End(Not run)
```

---

iteminfo *Function to calculate item information*

Description

Given an internal mirt item object extracted by using `extract.item`, compute the item information.

Usage

```
iteminfo(x, Theta, degrees = NULL, total.info = TRUE,
  multidim_matrix = FALSE)
```

Arguments

- `x` an extracted internal mirt object containing item information (see `extract.item`)
- `Theta` a vector (unidimensional) or matrix (multidimensional) of latent trait values
- `degrees` a vector of angles in degrees that are between 0 and 90. Only applicable when the input object is multidimensional
- `total.info` logical; return the total information curve for the item? If `FALSE`, information curves for each category are returned as a matrix
- `multidim_matrix` logical; compute the information matrix for each row in `Theta`? If `Theta` contains more than 1 row then a list of matricies will be returned, otherwise if `Theta` has exactly one row then a matrix will be returned

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

See Also

`extract.item`

Examples

```
## Not run:
mod <- mirt(Science, 1)
extr.2 <- extract.item(mod, 2)
Theta <- matrix(seq(-4,4, by = .1))
info.2 <- iteminfo(extr.2, Theta)

#do something with the info?
plot(Theta, info.2, type = 'l', main = 'Item information')

#category information curves
```

```
cat.info <- iteminfo(extr.2, Theta, total.info = FALSE)
plot(Theta, cat.info[,1], type = 'l', ylim = c(0, max(cat.info)),
     ylab = 'info', main = 'Category information')
for(i in 2:ncol(cat.info))
   lines(Theta, cat.info[,i], col = i)

## Customized test information plot
T1 <- T2 <- 0
dat <- expand.table(LSAT7)
mod1 <- mirt(dat, 1)
mod2 <- mirt(dat, 1, 'Rasch')
for(i in 1:5){
  T1 <- T1 + iteminfo(extract.item(mod1, i), Theta)
  T2 <- T2 + iteminfo(extract.item(mod2, i), Theta)
}
plot(Theta, T2/T1, type = 'l', ylab = 'Relative Test Information', las = 1)
lines(Theta, T1/T1, col = 'red')

# multidimensional
mod <- mirt(dat, 2, TOL=1e-2)
ii <- extract.item(mod, 1)
Theta <- as.matrix(expand.grid(-4:4, -4:4))

iteminfo(ii, Theta, degrees=c(45,45)) # equal angle
iteminfo(ii, Theta, degrees=c(90,0)) # first dimension only

# information matricies
iteminfo(ii, Theta, multidim_matrix = TRUE)
iteminfo(ii, Theta[1, , drop=FALSE], multidim_matrix = TRUE)


## End(Not run)
```

---

itemplot *Displays item surface and information plots*

Description

itemplot displays various item based IRT plots, with special options for plotting items that contain several 0 slope parameters. Supports up to three dimensional models.

Usage

```
itemplot(object, item, type = "trace", degrees = 45, CE = FALSE,
  CEalpha = 0.05, CEdraws = 1000, drop.zeros = FALSE, theta_lim = c(-6,
  6), shiny = FALSE, rot = list(xaxis = -70, yaxis = 30, zaxis = 10),
  par.strip.text = list(cex = 0.7), par.settings = list(strip.background =
  list(col = "#9ECAE1"), strip.border = list(col = "black")),
  auto.key = list(space = "right"), ...)
```

Arguments

`object`	a computed model object of class `SingleGroupClass` or `MultipleGroupClass`. Input may also be a `list` for comparing similar item types (e.g., 1PL vs 2PL)
`item`	a single numeric value, or the item name, indicating which item to plot
`type`	plot type to use, information (`'info'`), standard errors (`'SE'`), item trace lines (`'trace'`), information and standard errors (`'infoSE'`) or information and trace lines (`'infotrace'`), relative efficiency lines (`'RE'`), expected score `'score'`, or information and trace line contours (`'infocontour'` and `'tracecontour'`; not supported for `MultipleGroupClass` objects)
`degrees`	the degrees argument to be used if there are two or three factors. See `iteminfo` for more detail. A new vector will be required for three dimensional models to override the default
`CE`	logical; plot confidence envelope?
`CEalpha`	area remaining in the tail for confidence envelope. Default gives 95% confidence region
`CEdraws`	draws number of draws to use for confidence envelope
`drop.zeros`	logical; drop slope values that are numerically close to zero to reduce dimensionality? Useful in objects returned from `bfactor` or other confirmatory models that contain several zero slopes
`theta_lim`	lower and upper limits of the latent trait (theta) to be evaluated, and is used in conjunction with `npts`
`shiny`	logical; run interactive display for item plots using the `shiny` interface. This primarily is an instructive tool for demonstrating how item response curves behave when adjusting their parameters
`rot`	a list of rotation coordinates to be used for 3 dimensional plots
`par.strip.text`	plotting argument passed to `lattice`
`par.settings`	plotting argument passed to `lattice`
`auto.key`	plotting argument passed to `lattice`
`...`	additional arguments to be passed to `lattice` and `coef()`

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

Examples

```
## Not run:

data(LSAT7)
fulldata <- expand.table(LSAT7)
mod1 <- mirt(fulldata,1,SE=TRUE)
mod2 <- mirt(fulldata,1, itemtype = 'Rasch')
mod3 <- mirt(fulldata,2)

itemplot(mod1, 2)
```

```
itemplot(mod1, 2, CE = TRUE)
itemplot(mod1, 2, type = 'info')
itemplot(mod1, 2, type = 'info', CE = TRUE)

mods <- list(twoPL = mod1, onePL = mod2)
itemplot(mods, 1, type = 'RE')

#multidimensional
itemplot(mod3, 4, type = 'info')
itemplot(mod3, 4, type = 'infocontour')
itemplot(mod3, 4, type = 'tracecontour')

#polytomous items
pmod <- mirt(Science, 1, SE=TRUE)
itemplot(pmod, 3)
itemplot(pmod, 3, CE = TRUE)
itemplot(pmod, 3, type = 'score')
itemplot(pmod, 3, type = 'infotrace')

# use the directlabels package to put labels on tracelines
library(directlabels)
plt <- itemplot(pmod, 3)
direct.label(plt, 'top.points')

# change colour theme of plots
bwtheme <- standard.theme("pdf", color=FALSE)
plot(pmod, type='trace', par.settings=bwtheme)
itemplot(pmod, 1, type = 'trace', par.settings=bwtheme)

itemplot(pmod, 1, type = 'infoSE')
update(trellis.last.object(), par.settings = bwtheme)

# uncomment to run interactive shiny applet
# itemplot(shiny = TRUE)

## End(Not run)
```

`key2binary` *Score a test by converting response patterns to binary data*

Description

The `key2binary` function will convert response pattern data to a dichotomous format, given a response key.

Usage

```
key2binary(fulldata, key)
```

Arguments

`fulldata`	an object of class `data.frame`, `matrix`, or `table` with the response patterns
`key`	a vector or matrix consisting of the 'correct' response to the items. Each value/row corresponds to each column in `fulldata`. If the input is a matrix, multiple scoring keys can be supplied for each item. NA values are used to indicate no scoring key (or in the case of a matrix input, no additional scoring keys)

Value

Returns a numeric matrix with all the response patterns in dichotomous format

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

Examples

```
data(SAT12)
head(SAT12)
key <- c(1,4,5,2,3,1,2,1,3,1,2,4,2,1,5,3,4,4,1,4,3,3,4,1,3,5,1,3,1,5,4,5)

dicho.SAT12 <- key2binary(SAT12, key)
head(dicho.SAT12)

# multiple scoring keys
key2 <- cbind(c(1,4,5,2,3,1,2,1,3,1,2,4,2,1,5,3,4,4,1,4,3,3,4,1,3,5,1,3,1,5,4,5),
              c(2,3,NA,1,rep(NA, 28)))
dicho.SAT12 <- key2binary(SAT12, key2)

# keys from raw character responses
resp <- as.data.frame(matrix(c(
  "B","B","D","D","E",
  "B","A","D","D","E",
  "B","A","D","C","E",
  "D","D","D","C","E",
  "B","C","A","D","A"), ncol=5, byrow=TRUE))

key <- c("B", "D", "D", "C", "E")

d01 <- key2binary(resp, key)
head(d01)
```

`lagrange` *Lagrange test for freeing parameters*

Description

Lagrange (i.e., score) test to test whether parameters should be freed from a more constrained baseline model.

Usage

```
lagrange(mod, parnum, SE.type = "Oakes", type = "central", ...)
```

Arguments

mod	an estimated model
parnum	a vector, or list of vectors, containing one or more parameter locations/sets of locations to be tested. See objects returned from mod2values for the locations
SE.type	type of information matrix estimator to use. See mirt for further details
type	type of numerical algorithm passed to numerical_deriv to obtain the gradient terms
...	additional arguments to pass to mirt

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

See Also

wald

Examples

```
## Not run:
dat <- expand.table(LSAT7)
mod <- mirt(dat, 1, 'Rasch')
(values <- mod2values(mod))

#test all fixed slopes individually
parnum <- values$parnum[values$name == 'a1']
lagrange(mod, parnum)

# compare to LR test for first two slopes
mod2 <- mirt(dat, 'F = 1-5
                   FREE = (1, a1)', 'Rasch')
coef(mod2, simplify=TRUE)$items
anova(mod, mod2)

mod2 <- mirt(dat, 'F = 1-5
                   FREE = (2, a1)', 'Rasch')
coef(mod2, simplify=TRUE)$items
anova(mod, mod2)

mod2 <- mirt(dat, 'F = 1-5
```

```
                   FREE = (3, a1)', 'Rasch')
coef(mod2, simplify=TRUE)$items
anova(mod, mod2)

# test slopes first two slopes and last three slopes jointly
lagrange(mod, list(parnum[1:2], parnum[3:5]))

# test all 5 slopes and first + last jointly
lagrange(mod, list(parnum[1:5], parnum[c(1, 5)]))


## End(Not run)
```

`logLik-method` *Extract log-likelihood*

Description

Extract the observed-data log-likelihood.

Usage

```
## S4 method for signature 'SingleGroupClass'
logLik(object)
```

Arguments

`object`	an object of class `SingleGroupClass`, `MultipleGroupClass`, or `MixedClass`

Examples

```

## Not run:
x <- mirt(Science, 1)
logLik(x)


## End(Not run)
```

LSAT6 *Description of LSAT6 data*

Description

Data from Thissen (1982); contains 5 dichotomously scored items obtained from the Law School Admissions Test, section 6.

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

References

Thissen, D. (1982). Marginal maximum likelihood estimation for the one-parameter logistic model. *Psychometrika, 47*, 175-186.

Examples

```
## Not run:
dat <- expand.table(LSAT6)
head(dat)
model <- 'F = 1-5
         CONSTRAIN = (1-5, a1)'
(mod <- mirt(dat, model))
M2(mod)
itemfit(mod)
coef(mod, simplify=TRUE)

#equivalentely, but with a different parameterization
mod2 <- mirt(dat, 1, itemtype = 'Rasch')
anova(mod, mod2) #equal
M2(mod2)
itemfit(mod2)
coef(mod2, simplify=TRUE)
sqrt(coef(mod2)$GroupPars[2]) #latent SD equal to the slope in mod


## End(Not run)
```

LSAT7 *Description of LSAT7 data*

Description

Data from Bock & Lieberman (1970); contains 5 dichotomously scored items obtained from the Law School Admissions Test, section 7.

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

References

Bock, R. D., & Lieberman, M. (1970). Fitting a response model for *n* dichotomously scored items. *Psychometrika, 35*(2), 179-197.

Examples

```
## Not run:
dat <- expand.table(LSAT7)
head(dat)
(mod <- mirt(dat, 1))
coef(mod)

## End(Not run)
```

M2 *Compute the M2 model fit statistic*

Description

Computes the M2 (Maydeu-Olivares & Joe, 2006) statistic for dichotomous data and the M2* statistic for polytomous data (collapsing over response categories for better stability; see Cai and Hansen, 2013), as well as associated fit indices that are based on fitting the null model. Supports single and multiple-group models.

Usage

```
M2(obj, calcNull = TRUE, quadpts = NULL, theta_lim = c(-6, 6),
  impute = 0, CI = 0.9, residmat = FALSE, QMC = FALSE, suppress = 1,
  ...)
```

Arguments

`obj`	an estimated model object from the mirt package
`calcNull`	logical; calculate statistics for the null model as well? Allows for statistics such as the limited information TLI and CFI. Only valid when items all have a suitable null model (e.g., those created via `createItem` will not)
`quadpts`	number of quadrature points to use during estimation. If `NULL`, a suitable value will be chosen based on the rubric found in `fscores`
`theta_lim`	lower and upper range to evaluate latent trait integral for each dimension
`impute`	a number indicating how many imputations to perform (passed to `imputeMissing`) when there are missing data present. This requires a precomputed `Theta` input. Will return a data.frame object with the mean estimates of the stats and their imputed standard deviations
`CI`	numeric value from 0 to 1 indicating the range of the confidence interval for RMSEA. Default returns the 90% interval
`residmat`	logical; return the residual matrix used to compute the SRMSR statistic? Only the lower triangle of the residual correlation matrix will be returned (the upper triangle is filled with NA's)
`QMC`	logical; use quasi-Monte Carlo integration? Useful for higher dimensional models. If `quadpts` not specified, 5000 nodes are used by default
`suppress`	a numeric value indicating which parameter residual dependency combinations to flag as being too high. Absolute values for the standardized residuals greater than this value will be returned, while all values less than this value will be set to NA. Must be used in conjunction with the argument `residmat = TRUE`
`...`	additional arguments to pass

Value

Returns a data.frame object with the M2 statistic, along with the degrees of freedom, p-value, RMSEA (with 90% confidence interval), SRMSR for each group (if all items were ordinal), and optionally the TLI and CFI model fit statistics of `calcNull = TRUE`.

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

References

Cai, L. & Hansen, M. (2013). Limited-information goodness-of-fit testing of hierarchical item factor models. British Journal of Mathematical and Statistical Psychology, 66, 245-276.

Maydeu-Olivares, A. & Joe, H. (2006). Limited information goodness-of-fit testing in multidimensional contingency tables Psychometrika, 71, 713-732.

Examples

```
## Not run:
dat <- as.matrix(expand.table(LSAT7))
(mod1 <- mirt(dat, 1))
M2(mod1)
M2(mod1, residmat=TRUE) #lower triangle of residual correlation matrix

#M2 imputed with missing data present (run in parallel)
dat[sample(1:prod(dim(dat)), 250)] <- NA
mod2 <- mirt(dat, 1)
mirtCluster()
M2(mod2, impute = 10)


## End(Not run)
```

---

marginal_rxx — *Function to calculate the marginal reliability*

Description

Given an estimated model and a prior density function, compute the marginal reliability. This is only available for unidimensional tests.

Usage

```
marginal_rxx(mod, density = dnorm, ...)
```

Arguments

`mod`	an object of class 'SingleGroupClass'
`density`	a density function to use for integration. Default assumes the latent traits are from a normal (Gaussian) distribution
`...`	additional arguments passed to the density function

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

See Also

empirical_rxx, extract.group, testinfo

Examples

```
## Not run:

dat <- expand.table(deAyala)
mod <- mirt(dat, 1)

# marginal estimate
marginal_rxx(mod)

# empirical estimate (assuming the same prior)
fscores(mod, returnER = TRUE)

# empirical rxx the alternative way, given theta scores and SEs
fs <- fscores(mod, full.scores.SE=TRUE)
head(fs)
empirical_rxx(fs)


## End(Not run)
```

`MDIFF` *Compute multidimensional difficulty index*

Description

Returns a matrix containing the MDIFF values (Reckase, 2009). Only suppored for items of class 'dich' and 'graded'.

Usage

```
MDIFF(x, which.items = NULL)
```

Arguments

`x`	an object of class 'SingleGroupClass'
`which.items`	a vector indicating which items to select. If NULL is used (the default) then MDISC will be computed for all items

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

References

Reckase, M. D. (2009). Multidimensional Item Response Theory. Springer.

See Also

extract.group, MDISC

Examples

```
## Not run:

mod <- mirt(Science, 2)
MDIFF(mod)

mod <- mirt(expand.table(LSAT7), 2)
MDIFF(mod)


## End(Not run)
```

---

mdirt *Multidimensional discrete item response theory*

Description

mdirt fits a variety of item response models with discrete latent variables. These include, but are not limited to, latent class analysis, multidimensional latent class models, multidimensional discrete latent class models, DINA/DINO models, grade of measurement models, and so on.

Usage

```
mdirt(data, model, customTheta = NULL, nruns = 1, method = "EM",
  optimizer = "nlminb", return_max = TRUE, group = NULL,
  GenRandomPars = FALSE, verbose = TRUE, pars = NULL,
  technical = list(), ...)
```

Arguments

data	a matrix or data.frame that consists of numerically ordered data, with missing data coded as NA
model	number of classes to fit, or alternatively a mirt.model definition. Note that when using a mirt.model input in conjunction with the customTheta input defined below, the order with which the syntax factors are defined will be associated with the columns in the customTheta input
customTheta	input passed to techincal = list(customTheta = ...), but is included directly in this function for convienience. This input is most interesting for discrete latent models because it allows for customized patterns of latent class effects. The default builds the pattern customTheta = diag(model), which is the typical pattern for the traditional latent class analysis (whereby classes are completely distinct)

nruns	a numeric value indicating how many times the model should be fit to the data when using random starting values. If greater than 1, GenRandomPars is set to true by default
method	estimation method. Can be 'EM' or 'BL' (see mirt for more details)
optimizer	optimizer used for the M-step, set to 'nlminb' by default. See mirt for more details
return_max	logical; when nruns > 1, return the model that has the most optimal maximum likelihood criteria? If FALSE, returns a list of all the estimated objects
group	a factor variable indicating group membership used for multiple group analyses
GenRandomPars	logical; use random starting values
verbose	logical; turn on messages to the R console
pars	used for modifying starting values; see mirt for details
technical	list of lower-level inputs. See mirt for details
...	additional arguments to be passed to the estimation engine. See mirt for more details and examples

Details

Posterior classification accuracy for each response pattern may be obtained via the fscores function. The summary() function will display the category probability values given the class membership, which can also be displayed graphically with plot(), while coef() displays the raw coefficient values (and their standard errors, if estimated). Finally, anova() is used to compare nested models, while M2 and itemfit may be used for model fitting purposes.

'lca' model definition

The latent class IRT model with two latent classes has the form

$$P(x = k|\theta_1, \theta_2, a1, a2) = \frac{exp(a1\theta_1 + a2\theta_2)}{\sum_j^K exp(a1\theta_1 + a2\theta_2)}$$

where the θ values generally take on discrete points (such as 0 or 1). For proper identification, the first category slope parameters ($a1$ and $a2$) are never freely estimated. Alternatively, supplying a different grid of θ values will allow the estimation of similar models (multidimensional discrete models, grade of membership, etc.). See the examples below.

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

See Also

fscores, mirt.model, M2, itemfit, boot.mirt, mirtCluster, wald, coef-method, summary-method, anova-method, residuals-method

Examples

```
## Not run:
#LSAT6 dataset
dat <- expand.table(LSAT6)

# fit with 2-3 latent classes
(mod2 <- mdirt(dat, 2))
(mod3 <- mdirt(dat, 3))
summary(mod2)
residuals(mod2)
residuals(mod2, type = 'exp')
anova(mod2, mod3)
M2(mod2)
itemfit(mod2)

# generate classification plots
plot(mod2)
plot(mod2, facet_items = FALSE)
plot(mod2, profile = TRUE)

# available for polytomous data
mod <- mdirt(Science, 2)
summary(mod)
plot(mod)
plot(mod, profile=TRUE)

# classification based on response patterns
fscores(mod2, full.scores = FALSE)

# classify individuals either with the largest posterior probability.....
fs <- fscores(mod2)
head(fs)
classes <- matrix(1:2, nrow(fs), 2, byrow=TRUE)
class_max <- classes[t(apply(fs, 1, max) == fs)]
table(class_max)

# ... or by probability sampling (closer to estimated class proportions)
class_prob <- apply(fs, 1, function(x) sample(1:2, 1, prob=x))
table(class_prob)

# plausible value imputations for stocastic classification in both classes
pvs <- fscores(mod2, plausible.draws=10)
tabs <- lapply(pvs, function(x) apply(x, 2, table))
tabs[[1]]


# fit with random starting points (run in parallel to save time)
mirtCluster()
mod <- mdirt(dat, 2, nruns=10)

#--------------------------
```

```
# Grade of measurement model

# define a custom Theta grid for including a 'fuzzy' class membership
(Theta <- matrix(c(1, 0, .5, .5, 0, 1), nrow=3 , ncol=2, byrow=TRUE))
(mod_gom <- mdirt(dat, 2, customTheta = Theta))
summary(mod_gom)

#-----------------
# Multidimensional discrete latent class model

dat <- key2binary(SAT12,
     key = c(1,4,5,2,3,1,2,1,3,1,2,4,2,1,5,3,4,4,1,4,3,3,4,1,3,5,1,3,1,5,4,5))

# define Theta grid for three latent classes
(Theta <- matrix(c(0,0,0, 1,0,0, 0,1,0, 0,0,1, 1,1,0, 1,0,1, 0,1,1, 1,1,1),
   ncol=3, byrow=TRUE))
(mod_discrete <- mdirt(dat, 3, customTheta = Theta))
summary(mod_discrete)

# Located latent class model
model <- mirt.model('C1 = 1-32
                     C2 = 1-32
                     C3 = 1-32
                     CONSTRAIN = (1-32, a1), (1-32, a2), (1-32, a3)')
(mod_located <- mdirt(dat, model, customTheta = diag(3)))
summary(mod_located)

#-----------------
### DINA model example
# generate some suitable data for a two dimensional DINA application
#     (first columns are intercepts)
set.seed(1)
Theta <- expand.table(matrix(c(1,0,0,0, 200,
                               1,1,0,0, 200,
                               1,0,1,0, 100,
                               1,1,1,1, 500), 4, 5, byrow=TRUE))
a <- matrix(c(rnorm(15, -1.5, .5), rlnorm(5, .2, .3), numeric(15), rlnorm(5, .2, .3),
              numeric(15), rlnorm(5, .2, .3)), 15, 4)

guess <- plogis(a[11:15,1]) # population guess
slip <- 1 - plogis(rowSums(a[11:15,])) # population slip

dat <- simdata(a, Theta=Theta, itemtype = 'lca')

# first column is the intercept, 2nd and 3rd are attributes
theta <- matrix(c(1,0,0,
                  1,1,0,
                  1,0,1,
                  1,1,1), 4, 3, byrow=TRUE)
theta <- cbind(theta, theta[,2] * theta[,3]) #DINA interaction of main attributes
model <- mirt.model('Intercept = 1-15
                     A1 = 1-5
                     A2 = 6-10
```

```
                     A1A2 = 11-15')

mod <- mdirt(dat, model, customTheta = theta)
coef(mod)
summary(mod)
M2(mod) # fits well

cfs <- coef(mod, simplify=TRUE)$items[11:15,]
cbind(guess, estguess = plogis(cfs[,1]))
cbind(slip, estslip = 1 - plogis(rowSums(cfs)))


### DINO model example
theta <- matrix(c(1,0,0,
                  1,1,0,
                  1,0,1,
                  1,1,1), 4, 3, byrow=TRUE)
# define theta matrix with negative interaction term
theta <- cbind(theta, -theta[,2] * theta[,3])

model <- mirt.model('Intercept = 1-15
                     A1 = 1-5, 11-15
                     A2 = 6-15
                     Yoshi = 11-15
                     CONSTRAIN = (11,a2,a3,a4), (12,a2,a3,a4), (13,a2,a3,a4),
                                 (14,a2,a3,a4), (15,a2,a3,a4)')

mod <- mdirt(dat, model, customTheta = theta)
coef(mod, simplify=TRUE)
summary(mod)
M2(mod) #doesn't fit as well, because not the generating model


#------------------
#multidimensional latent class model

dat <- key2binary(SAT12,
     key = c(1,4,5,2,3,1,2,1,3,1,2,4,2,1,5,3,4,4,1,4,3,3,4,1,3,5,1,3,1,5,4,5))

# 5 latent classes within 2 different sets of items
model <- mirt.model('C1 = 1-16
                     C2 = 1-16
                     C3 = 1-16
                     C4 = 1-16
                     C5 = 1-16
                     C6 = 17-32
                     C7 = 17-32
                     C8 = 17-32
                     C9 = 17-32
                     C10 = 17-32
                  CONSTRAIN = (1-16, a1), (1-16, a2), (1-16, a3), (1-16, a4), (1-16, a5),
                      (17-32, a6), (17-32, a7), (17-32, a8), (17-32, a9), (17-32, a10)')
```

```
theta <- diag(10)
mod <- mdirt(dat, model, customTheta = theta)
coef(mod, simplify=TRUE)
summary(mod)

#------------------
# multiple group with constrained group probabilities
 dat <- key2binary(SAT12,
   key = c(1,4,5,2,3,1,2,1,3,1,2,4,2,1,5,3,4,4,1,4,3,3,4,1,3,5,1,3,1,5,4,5))
group <- rep(c('G1', 'G2'), each = nrow(SAT12)/2)
Theta <- diag(2)

# the latent class parameters are technically located in the (nitems + 1) location
model <- mirt.model('A1 = 1-32
                     A2 = 1-32
                     CONSTRAINB = (33, c1)')
mod <- mdirt(dat, model, group = group, customTheta = Theta)
coef(mod, simplify=TRUE)
summary(mod)



## End(Not run)
```

MDISC *Compute multidimensional discrimination index*

Description

Returns a vector containing the MDISC values for each item in the model input object (Reckase, 2009).

Usage

```
MDISC(x)
```

Arguments

x	an object of class 'SingleGroupClass'

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

References

Reckase, M. D. (2009). Multidimensional Item Response Theory. Springer.

See Also

extract.group

Examples

```
## Not run:

mod <- mirt(Science, 2)
MDISC(mod)


## End(Not run)
```

---

mirt *Full-Information Item Factor Analysis (Multidimensional Item Response Theory)*

Description

mirt fits an unconditional maximum likelihood factor analysis model to any mixture of dichotomous and polytomous data under the item response theory paradigm using either Cai's (2010) Metropolis-Hastings Robbins-Monro (MHRM) algorithm, with an EM algorithm approach outlined by Bock and Aiken (1981) using rectangular or quasi-Monte Carlo integration grids, or with the stochastic EM (i.e., the first two stages of the MH-RM algorithm). Models containing 'explanatory' person or item level predictors can only be included by using the mixedmirt function, though latent regression models can be fit using the formula input below. Tests that form a two-tier or bifactor structure should be estimated with the bfactor function, which uses a dimension reduction EM algorithm for modeling item parcels. Multiple group analyses (useful for DIF and DTF testing) are also available using the multipleGroup function.

Usage

```
mirt(data, model, itemtype = NULL, guess = 0, upper = 1, SE = FALSE,
  covdata = NULL, formula = NULL, SE.type = "Oakes", method = "EM",
  optimizer = NULL, pars = NULL, constrain = NULL, parprior = NULL,
  calcNull = TRUE, draws = 5000, survey.weights = NULL, quadpts = NULL,
  TOL = NULL, gpcm_mats = list(), grsm.block = NULL, rsm.block = NULL,
  key = NULL, large = FALSE, GenRandomPars = FALSE,
  accelerate = "Ramsay", empiricalhist = FALSE, verbose = TRUE,
  solnp_args = list(), alabama_args = list(), spline_args = list(),
  control = list(), technical = list(), ...)
```

Arguments

data	a matrix or data.frame that consists of numerically ordered data, with missing data coded as NA (to convert from an ordered factor data.frame see data.matrix)

model
: a string to be passed (or an object returned from) `mirt.model`, declaring how the IRT model is to be estimated (loadings, constraints, priors, etc). For exploratory IRT models, a single numeric value indicating the number of factors to extract is also supported

itemtype
: type of items to be modeled, declared as a vector for each item or a single value which will be recycled for each item. The `NULL` default assumes that the items follow a graded or 2PL structure, however they may be changed to the following:

- `'Rasch'` - Rasch/partial credit model by constraining slopes to 1 and freely estimating the variance parameters (alternatively, can be specified by applying equality constraints to the slope parameters in `'gpcm'`; Rasch, 1960)
- `'2PL'`, `'3PL'`, `'3PLu'`, and `'4PL'` - 2-4 parameter logistic model, where `3PL` estimates the lower asymptote only while `3PLu` estimates the upper asymptote only (Lord and Novick, 1968; Lord, 1980)
- `'graded'` - graded response model (Samejima, 1969)
- `'grsm'` and `'grsmIRT'` - graded ratings scale model in the slope-intercept and classical IRT parameterization. `'grsmIRT'` is restricted to unidimensional models (Muraki, 1992)
- `'gpcm'` and `'gpcmIRT'` - generalized partial credit model in the slope-intercept and classical parameterization. `'gpcmIRT'` is restricted to unidimensional models. Note that optional scoring matrices for `'gpcm'` are available with the `gpcm_mats` input (Muraki, 1992)
- `'rsm'` - Rasch rating scale model using the `'gpcmIRT'` structure (unidimensional only; Andrich, 1978)
- `'nominal'` - nominal response model (Bock, 1972)
- `'ideal'` - dichotomous ideal point model (Maydeu-Olivares, 2006)
- `'PC2PL'` and `'PC3PL'` - 2-3 parameter partially compensatory model. Note that constraining the slopes to be equal across items will reduce the model to Embretson's (a.k.a. Whitely's) multicomponent model (1980).
- `'2PLNRM'`, `'3PLNRM'`, `'3PLuNRM'`, and `'4PLNRM'` - 2-4 parameter nested logistic model, where `3PLNRM` estimates the lower asymptote only while `3PLuNRM` estimates the upper asymptote only (Suh and Bolt, 2010)
- `'spline'` - spline response model with the `bs` (default) or the `ns` function (Winsberg, Thissen, and Wainer, 1984)

Additionally, user defined item classes can also be defined using the `createItem` function

guess
: fixed pseudo-guessing parameters. Can be entered as a single value to assign a global guessing parameter or may be entered as a numeric vector corresponding to each item

upper
: fixed upper bound parameters for 4-PL model. Can be entered as a single value to assign a global guessing parameter or may be entered as a numeric vector corresponding to each item

SE
: logical; estimate the standard errors by computing the parameter information matrix? See `SE.type` for the type of estimates available

covdata
: a data.frame of data used for latent regression models

formula
: an R formula (or list of formulas) indicating how the latent traits can be regressed using external covariates in covdata. If a named list of formulas is supplied (where the names correspond to the latent trait names in model) then specific regression effects can be estimated for each factor. Supplying a single formula will estimate the regression parameters for all latent traits by default

SE.type
: type of estimation method to use for calculating the parameter information matrix for computing standard errors and wald tests. Can be:

- 'Richardson', 'forward', or 'central' for the numerical Richardson, forward difference, and central difference evaluation of observed Hessian matrix
- 'crossprod' and 'Louis' for standard error computations based on the variance of the Fisher scores as well as Louis' (1982) exact computation of the observed information matrix
- 'sandwich' for the sandwich covariance estimate based on the 'crossprod' and 'Louis' estimates
- 'Oakes' for Oakes' (1999) method using a central difference approximation
- 'SEM' for the supplemented EM (disables the accelerate option automatically; EM only)
- 'Fisher' for the expected information, 'complete' for information based on the complete-data Hessian used in EM algorithm
- 'MHRM' and 'FMHRM' for stochastic approximations of observed information matrix based on the Robbins-Monro filter or a fixed number of MHRM draws without the RM filter. These are the only options supported when method = 'MHRM'
- 'numerical' to obtain the numerical estimate from a call to optim when method = 'BL'

Note that both the 'SEM' method becomes very sensitive if the ML solution has has not been reached with sufficient precision, and may be further sensitive if the history of the EM cycles is not stable/sufficient for convergence of the respective estimates. Increasing the number of iterations (increasing NCYCLES and decreasing TOL, see below) will help to improve the accuracy, and can be run in parallel if a mirtCluster object has been defined (this will be used for Oakes' method as well). Additionally, inspecting the symmetry of the ACOV matrix for convergence issues by passing technical = list(symmetric = FALSE) can be helpful to determine if a sufficient solution has been reached

method
: a character object specifying the estimation algorithm to be used. The default is 'EM', for the standard EM algorithm with fixed quadrature, 'QMCEM' for quasi-Monte Carlo EM estimation, or 'MCEM' for Monte Carlo EM estimation. The option 'MHRM' may also be passed to use the MH-RM algorithm, 'SEM' for the Stochastic EM algorithm (first two stages of the MH-RM stage using an optimizer other than a single Newton-Raphson iteration), and 'BL' for the Bock and Lieberman approach (generally not recommended for longer tests).

The 'EM' is generally effective with 1-3 factors, but methods such as the 'QMCEM', 'MCEM', 'SEM', or 'MHRM' should be used when the dimensions are 3 or more. Note that when the optimizer is stochastic the associated SE.type is automatically changed to SE.type = 'MHRM' by default to avoid the use of quadrature `optimizer` a character indicating which numerical optimizer to use. By default, the EM algorithm will use the `'BFGS'` when there are no upper and lower bounds box-constraints and `'L-BFGS-B'` when there are. Another good option which supports bound constraints is the `'nlminb'`, which may be more stable than the BFGS family of optimizers (though slightly slower).

Other options include the Newton-Raphson (`'NR'`), which can be more efficient than the `'BFGS'` but not as stable for more complex IRT models (such as the nominal or nested logit models) and the related `'NR1'` which is also the Newton-Raphson but consists of only 1 update that has been coupled with RM Hessian (only applicable when the MH-RM algorithm is used). The MH-RM algorithm uses the `'NR1'` by default, and though currently the `'BFGS'`, `'L-BFGS-B'`, and `'NR'` are also supported with this method (with few iterations by default) to emulate stochastic EM updates. As well, the `'Nelder-Mead'` and `'SANN'` estimators are also available, but their routine use generally is not required or recommended.

Additionally, estimation subroutines from the `Rsolnp` and `alabama` packages are available by passing the arguments `'solnp'` and `'alabama'`, respectively. This should be used in conjunction with the `solnp_args` and `alabama_args` specified below. If equality constraints were specified in the model definition only the parameter with the lowest `parnum` in the `pars = 'values'` data.frame is used in the estimation vector passed to the objective function, and group hyper-parameters are omitted. Equality an inequality functions should be of the form `function(p, optim_args)`, where `optim_args` is a list of internally parameters that largely can be ignored when defining constraints (though use of `browser()` here may be helpful). Note: for the `'alabama'` optimizer, the starting values should be adjusted such that all constraints are met prior to the first maximization-step. The `'solnp'` optimizer is less sensitive to this initial condition restriction, but it may also if the model is unstable early in the EM cycles

`pars` a data.frame with the structure of how the starting values, parameter numbers, estimation logical values, etc, are defined. The user may observe how the model defines the values by using `pars = 'values'`, and this object can in turn be modified and input back into the estimation with `pars = mymodifiedpars`

`constrain` a list of user declared equality constraints. To see how to define the parameters correctly use `pars = 'values'` initially to see how the parameters are labeled. To constrain parameters to be equal create a list with separate concatenated vectors signifying which parameters to constrain. For example, to set parameters 1 and 5 equal, and also set parameters 2, 6, and 10 equal use `constrain = list(c(1,5), c(2,6,10))`. Constraints can also be specified using the `mirt.model` syntax (recommended)

`parprior` a list of user declared prior item probabilities. To see how to define the parameters correctly use `pars = 'values'` initially to see how the parameters are labeled. Can define either normal (e.g., intercepts, lower/guessing and upper bounds), log-normal (e.g., for univariate slopes), or beta prior probabilities. To specify a prior the form is c('priortype', ...), where normal priors are `parprior = list(c(parnumbers, 'norm', mean, sd))`, `parprior = list(c(parnumbers, 'lnorm` for log-normal, and `parprior = list(c(parnumbers, 'beta', alpha, beta))` for beta, and `parprior = list(c(parnumbers, 'expbeta', alpha, beta))` for the beta distribution after applying the function `plogis` to the input value (note, this is specifically for applying a beta prior to the lower/upper-bound parameters in 3/4PL models). Priors can also be specified using `mirt.model` syntax (recommended)

`calcNull` logical; calculate the Null model for additional fit statistics (e.g., TLI)? Only applicable if the data contains no NA's and the data is not overly sparse, otherwise it is ignored

`draws` the number of Monte Carlo draws to estimate the log-likelihood for the MH-RM algorithm. Default is 5000

`survey.weights` a optional numeric vector of survey weights to apply for each case in the data (EM estimation only). If not specified, all cases are weighted equally (the standard IRT approach). The sum of the `survey.weights` must equal the total sample size for proper weighting to be applied

`quadpts` number of quadrature points per dimension (must be larger than 2). By default the number of quadrature uses the following scheme: `switch(as.character(nfact), '1'=61, '2'=31` However, if the method input is set to `'QMCEM'` and this argument is left blank then the default number of quasi-Monte Carlo integration nodes will be set to 5000 in total

`TOL` convergence threshold for EM or MH-RM; defaults are .0001 and .001. If `SE.type = 'SEM'` and this value is not specified, the default is set to `1e-5`. If `empiricalhist = TRUE` and `TOL` is not specified then the default `3e-5` will be used. To evaluate the model using only the starting values pass `TOL = NaN`, and to evaluate the starting values without the log-likelihood pass `TOL = NA`

`gpcm_mats` a list of matrices specifying how the scoring coefficients in the (generalized) partial credit model should be constructed. If omitted, the standard gpcm format will be used (i.e., `seq(0, k, by = 1)` for each trait). This input should be used if traits should be scored different for each category (e.g., `matrix(c(0:3, 1,0,0,0), 4, 2)` for a two-dimensional model where the first trait is scored like a gpcm, but the second trait is only positively indicated when the first category is selected). Can be used when `itemtypes` are `'gpcm'` or `'Rasch'`, but only when the respective element in `gpcm_mats` is not `NULL`

`grsm.block` an optional numeric vector indicating where the blocking should occur when using the grsm, NA represents items that do not belong to the grsm block (other items that may be estimated in the test data). For example, to specify two blocks of 3 with a 2PL item for the last item: `grsm.block = c(rep(1,3), rep(2,3), NA)`. If NULL the all items are assumed to be within the same group and therefore have the same number of item categories

`rsm.block` same as `grsm.block`, but for `'rsm'` blocks

`key` a numeric vector of the response scoring key. Required when using nested logit item types, and must be the same length as the number of items used. Items that are not nested logit will ignore this vector, so use `NA` in item locations that are not applicable

`large` either a `logical`, indicating whether the internal collapsed data should be returned, or a `list` of internally computed data tables. If `TRUE` is passed, a list containing the organized tables is returned. This list object can then be passed back into `large` to avoid reorganizing the data again (useful when the dataset are very large and computing the tabulated data is computationally burdensome).

The best strategy for large data is to always pass the internal data to the estimation function, shown below:

Compute organized data e.g., `internaldat <- mirt(Science, 1, large = TRUE)`

Pass the organized data to all estimation functions e.g., `mod <- mirt(Science, 1, large = intern`

`GenRandomPars` logical; generate random starting values prior to optimization instead of using the fixed internal starting values?

`accelerate` a character vector indicating the type of acceleration to use. Default is `'Ramsay'`, but may also be `'squarem'` for the SQUAREM procedure (specifically, the gSqS3 approach) described in Varadhan and Roldand (2008). To disable the acceleration, pass `'none'`

`empiricalhist` logical; estimate prior distribution using an empirical histogram approach. Only applicable for unidimensional models estimated with the EM algorithm. The number of cycles, TOL, and quadpts are adjusted accommodate for less precision during estimation (TOL = 3e-5, NCYCLES = 2000, quadpts = 199)

`verbose` logical; print observed- (EM) or complete-data (MHRM) log-likelihood after each iteration cycle? Default is TRUE

`solnp_args` a list of arguments to be passed to the `solnp::solnp()` function for equality constraints, inequality constraints, etc

`alabama_args` a list of arguments to be passed to the `alabama::constrOptim.nl()` function for equality constraints, inequality constraints, etc

`spline_args` a named list of lists containing information to be passed to the `bs` (default) and `ns` for each spline itemtype. Each element must refer to the name of the itemtype with the spline, while the internal list names refer to the arguments which are passed. For example, if item 2 were called 'read2', and item 5 were called 'read5', both of which were of itemtype 'spline' but item 5 should use the `ns` form, then a modified list for each input might be of the form:

`spline_args = list(read2 = list(degree = 4),` `read5 = list(fun = '`

This code input changes the `bs()` splines function to have a `degree = 4` input, while the second element changes to the `ns()` function with knots set a `c(-2, 2)`

`control` a list passed to the respective optimizers (i.e., `optim()`, `nlminb()`, etc). Additional arguments have been included for the `'NR'` optimizer: `'tol'` for the convergence tolerance in the M-step (default is `TOL/1000`), while the default number of iterations for the Newton-Raphson optimizer is 50 (modified with the `'maxit'` control input)

`technical` a list containing lower level technical parameters for estimation. May be:

NCYCLES maximum number of EM or MH-RM cycles; defaults are 500 and 2000

MAXQUAD maximum number of quadratures, which you can increase if you have more than 4GB or RAM on your PC; default 20000

theta_lim range of integration grid for each dimension; default is `c(-6, 6)`

set.seed seed number used during estimation. Default is 12345

SEtol standard error tolerance criteria for the S-EM and MHRM computation of the information matrix. Default is 1e-3

symmetric logical; force S-EM/Oakes information matrix to be symmetric? Default is TRUE so that computation of standard errors are more stable. Setting this to FALSE can help to detect solutions that have not reached the ML estimate

SEM_window ratio of values used to define the S-EM window based on the observed likelihood differences across EM iterations. The default is `c(0, 1 - SEtol)`, which provides nearly the very full S-EM window (i.e., nearly all EM cycles used). To use the a smaller SEM window change the window to to something like `c(.9, .999)` to start at a point farther into the EM history

warn logical; include warning messages during estimation? Default is TRUE

message logical; include general messages during estimation? Default is TRUE

customK a numeric vector used to explicitly declare the number of response categories for each item. This should only be used when constructing mirt model for reasons other than parameter estimation (such as to obtain factor scores), and requires that the input data all have 0 as the lowest category. The format is the same as the `extract.mirt(mod, 'K')` slot in all converged models

customPriorFun a custom function used to determine the normalized density for integration in the EM algorithm. Must be of the form `function(Theta, Etable){...}`, and return a numeric vector with the same length as number of rows in `Theta`. The `Etable` input contains the aggregated table generated from the current E-step computations. For proper integration, the returned vector should sum to 1 (i.e., normalized). Note that if using the `Etable` it will be NULL on the first call, therefore the prior will have to deal with this issue accordingly

customTheta a custom `Theta` grid, in matrix form, used for integration. If not defined, the grid is determined internally based on the number of `quadpts`

delta the deviation term used in numerical estimates when computing the ACOV matrix with the 'forward' or 'central' numerical approaches, as well as Oakes' method with the Richarson extrapolation. Default is 1e-5

parallel logical; use the parallel cluster defined by `mirtCluster`? Default is TRUE

removeEmptyRows logical; remove response vectors that only contain `NA`'s? Default is FALSE

internal_constrains logical; include the internal constrains when using certain IRT models (e.g., 'grsm' itemtype). Disable this if you want to use special optimizers such as the solnp. Default is `TRUE`

gain a vector of two values specifying the numerator and exponent values for the RM gain function $(val1/cycle)^val2$. Default is `c(0.10, 0.75)`

BURNIN number of burn in cycles (stage 1) in MH-RM; default is 150

SEMCYCLES number of SEM cycles (stage 2) in MH-RM; default is 100

MHDRAWS number of Metropolis-Hasting draws to use in the MH-RM at each iteration; default is 5

MHcand a vector of values used to tune the MH sampler. Larger values will cause the acceptance ratio to decrease. One value is required for each group in unconditional item factor analysis (`mixedmirt()` requires additional values for random effect). If null, these values are determined internally, attempting to tune the acceptance of the draws to be between .1 and .4

MHRM_SE_draws number of fixed draws to use when `SE=TRUE` and `SE.type = 'FMHRM'` and the maximum number of draws when `SE.type = 'MHRM'`. Default is 2000

MCEM_draws a function used to determine the number of quadrature points to draw for the `'MCEM'` method. Must include one arguemnt which indicates the iteration number of the EM cycle. Default is `function(cycles) 500 + (cycles - 1)*2`, which starts the number of draws at 500 and increases by 2 after each full EM iteration

info_if_converged logical; compute the information matrix when using the MH-RM algorithm only if the model converged within a suitable number of iterations? Default is `TRUE`

loglik_if_converged logical; compute the observed log-likelihood when using the MH-RM algorithm only if the model converged within a suitable number of iterations? Default is `TRUE`

keep_vcov_PD logical; attempt to keep the variance-covariance matrix of the latent traits positive definite during estimation in the EM algorithm? This generally improves the convergence properties when the traits are highly correlated. Default is `TRUE`

`...` additional arguments to be passed

Value

function returns an object of class `SingleGroupClass` (SingleGroupClass-class)

Confirmatory and Exploratory IRT

Specification of the confirmatory item factor analysis model follows many of the rules in the structural equation modeling framework for confirmatory factor analysis. The variances of the latent factors are automatically fixed to 1 to help facilitate model identification. All parameters may be fixed to constant values or set equal to other parameters using the appropriate declarations. Confirmatory models may also contain 'explanatory' person or item level predictors, though including predictors is currently limited to the `mixedmirt` function.

When specifying a single number greater than 1 as the `model` input to mirt an exploratory IRT model will be estimated. Rotation and target matrix options are available if they are passed to generic functions such as `summary-method` and `fscores`. Factor means and variances are fixed to ensure proper identification.

If the model is an exploratory item factor analysis estimation will begin by computing a matrix of quasi-polychoric correlations. A factor analysis with `nfact` is then extracted and item parameters are estimated by $a_{ij} = f_{ij}/u_j$, where f_{ij} is the factor loading for the *j*th item on the *i*th factor, and u_j is the square root of the factor uniqueness, $\sqrt{1 - h_j^2}$. The initial intercept parameters are determined by calculating the inverse normal of the item facility (i.e., item easiness), q_j, to obtain $d_j = q_j/u_j$. A similar implementation is also used for obtaining initial values for polytomous items.

A note on upper and lower bound parameters

Internally the g and u parameters are transformed using a logit transformation ($log(x/(1-x))$), and can be reversed by using $1/(1+exp(-x))$ following convergence. This also applies when computing confidence intervals for these parameters, and is done so automatically if `coef(mod, rawug = FALSE)`.

As such, when applying prior distributions to these parameters it is recommended to use a prior that ranges from negative infinity to positive infinity, such as the normally distributed prior via the `'norm'` input (see `mirt.model`).

Convergence for quadrature methods

Unrestricted full-information factor analysis is known to have problems with convergence, and some items may need to be constrained or removed entirely to allow for an acceptable solution. As a general rule dichotomous items with means greater than .95, or items that are only .05 greater than the guessing parameter, should be considered for removal from the analysis or treated with prior parameter distributions. The same type of reasoning is applicable when including upper bound parameters as well. For polytomous items, if categories are rarely endorsed then this will cause similar issues. Also, increasing the number of quadrature points per dimension, or using the quasi-Monte Carlo integration method, may help to stabilize the estimation process in higher dimensions. Finally, solutions that are not well defined also will have difficulty converging, and can indicate that the model has been misspecified (e.g., extracting too many dimensions).

Convergence for MH-RM method

For the MH-RM algorithm, when the number of iterations grows very high (e.g., greater than 1500) or when `Max Change = .2500` values are repeatedly printed to the console too often (indicating that the parameters were being constrained since they are naturally moving in steps greater than 0.25) then the model may either be ill defined or have a very flat likelihood surface, and genuine maximum-likelihood parameter estimates may be difficult to find. Standard errors are computed following the model convergence by passing `SE = TRUE`, to perform an addition MH-RM stage but treating the maximum-likelihood estimates as fixed points.

Additional helper functions

Additional functions are available in the package which can be useful pre- and post-estimation. These are:

`mirt.model` Define the IRT model specification use special syntax. Useful for defining between and within group parameter constraints, prior parameter distributions, and specifying the slope coefficients for each factor

`coef-method` Extract raw coefficients from the model, along with their standard errors and confidence intervals

`summary-method` Extract standardized loadings from model. Accepts a `rotate` argument for exploratory item response model

`anova-method` Compare nested models using likelihood ratio statistics as well as information criteria such as the AIC and BIC

`residuals-method` Compute pairwise residuals between each item using methods such as the LD statistic (Chen & Thissen, 1997), as well as response pattern residuals `plot-method` Plot various types of test level plots including the test score and information functions and more

`itemplot` Plot various types of item level plots, including the score, standard error, and information functions, and more

`createItem` Create a customized `itemtype` that does not currently exist in the package

`imputeMissing` Impute missing data given some computed Theta matrix

`fscores` Find predicted scores for the latent traits using estimation methods such as EAP, MAP, ML, WLE, and EAPsum

`wald` Compute Wald statistics follow the convergence of a model with a suitable information matrix

`M2` Limited information goodness of fit test statistic based to determine how well the model fits the data

`itemfit` **and** `personfit` Goodness of fit statistics at the item and person levels, such as the S-X2, infit, outfit, and more

`boot.mirt` Compute estimated parameter confidence intervals via the bootstrap methods

`mirtCluster` Define a cluster for the package functions to use for capitalizing on multi-core architecture to utilize available CPUs when possible. Will help to decrease estimation times for tasks that can be run in parallel

IRT Models

The parameter labels use the follow convention, here using two factors and k as the number of categories.

Rasch Only one intercept estimated, and the latent variance of θ is freely estimated. If the data have more than two categories then a partial credit model is used instead (see 'gpcm' below).

$$P(x = 1|\theta, d) = \frac{1}{1 + exp(-(\theta + d))}$$

2-4PL Depending on the model u may be equal to 1 and g may be equal to 0.

$$P(x = 1|\theta, \psi) = g + \frac{(u - g)}{1 + exp(-(a_1 * \theta_1 + a_2 * \theta_2 + d))}$$

graded The graded model consists of sequential 2PL models, and here k is the predicted category.

$$P(x = k|\theta, \psi) = P(x \geq k|\theta, \phi) - P(x \geq k + 1|\theta, \phi)$$

grsm and grsmIRT A more constrained version of the graded model where graded spacing is equal across item blocks and only adjusted by a single 'difficulty' parameter (c) while the latent variance of θ is freely estimated. Again,

$$P(x = k|\theta, \psi) = P(x \geq k|\theta, \phi) - P(x \geq k + 1|\theta, \phi)$$

but now

$$P = \frac{1}{1 + exp(-(a_1 * \theta_1 + a_2 * \theta_2 + d_k + c))}$$

The grsmIRT model is similar to the `grsm` item type, but uses the IRT parameterization instead (see Muraki, 1990 for this exact form). This is restricted to unidimensional models only, whereas `grsm` may be used for unidimensional or multidimensional models and is more consistent with the form of other IRT models in `mirt`

gpcm/nominal For the gpcm the d values are treated as fixed and ordered values from 0:(k-1) (in the nominal model d_0 is also set to 0). Additionally, for identification in the nominal model $ak_0 = 0$, $ak_{(k-1)} = (k-1)$.

$$P(x = k|\theta, \psi) = \frac{exp(ak_{k-1} * (a_1 * \theta_1 + a_2 * \theta_2) + d_{k-1})}{\sum_1^k exp(ak_{k-1} * (a_1 * \theta_1 + a_2 * \theta_2) + d_{k-1})}$$

For the partial credit model (when `itemtype = 'Rasch'`; unidimensional only) the above model is further constrained so that $ak = (0, 1, \ldots, k-1)$, $a_1 = 1$, and the latent variance of θ_1 is freely estimated. Alternatively, the partial credit model can be obtained by containing all the slope parameters in the gpcms to be equal. More specific scoring function may be included by passing a suitable list or matrices to the `gpcm_mats` input argument.

In the nominal model this parametrization helps to identify the empirical ordering of the categories by inspecting the ak values. Larger values indicate that the item category is more positively related to the latent trait(s) being measured. For instance, if an item was truly ordinal (such as a Likert scale), and had 4 response categories, we would expect to see $ak_0 < ak_1 < ak_2 < ak_3$ following estimation. If on the other hand $ak_0 > ak_1$ then it would appear that the second category is less related to to the trait than the first, and therefore the second category should be understood as the 'lowest score'.

NOTE: The nominal model can become numerical unstable if poor choices for the high and low values are chosen, resulting in `ak` values greater than `abs(10)` or more. It is recommended to choose high and low anchors that cause the estimated parameters to fall between 0 and the number of categories - 1 either by theoretical means or by re-estimating the model with better values following convergence.

gpcmIRT and rsm The gpcmIRT model is the classical generalized partial credit model for unidimensional response data. It will obtain the same fit as the `gpcm` presented above, however the parameterization allows for the Rasch/generalized rating scale model as a special case.

E.g., for a 4 category response model,

$$P(x = 0|\theta, \psi) = exp(1)/G$$

$$P(x = 1|\theta, \psi) = exp(1 + a(\theta - b1) + c)/G$$

$$P(x = 2|\theta, \psi) = exp(1 + a(2\theta - b1 - b2) + 2c)/G$$

$$P(x = 3|\theta, \psi) = exp(1 + a(3\theta - b1 - b2 - b3) + 3c)/G$$

where

$$G = exp(1)+exp(1+a(\theta-b1)+c)+exp(1+a(2\theta-b1-b2)+2c)+a(3\theta-b1-b2-b3)+3c)$$

Here a is the slope parameter, the b parameters are the threshold values for each adjacent category, and c is the so-called difficulty parameter when a rating scale model is fitted (otherwise, $c = 0$ and it drops out of the computations).

The gpcmIRT can be constrained to the partial credit IRT model by either constraining all the slopes to be equal, or setting the slopes to 1 and freeing the latent variance parameter.

Finally, the rsm is a more constrained version of the (generalized) partial credit model where the spacing is equal across item blocks and only adjusted by a single 'difficulty' parameter (c). Note that this is analogous to the relationship between the graded model and the grsm (with an additional constraint regarding the fixed discrimination parameters).

ideal The ideal point model has the form, with the upper bound constraint on d set to 0:

$$P(x = 1|\theta, \psi) = exp(-0.5 * (a_1 * \theta_1 + a_2 * \theta_2 + d)^2)$$

partcomp Partially compensatory models consist of the product of 2PL probability curves.

$$P(x = 1|\theta, \psi) = g + (1 - g)(\frac{1}{1 + exp(-(a_1 * \theta_1 + d_1))} * \frac{1}{1 + exp(-(a_2 * \theta_2 + d_2))})$$

Note that constraining the slopes to be equal across items will reduce the model to Embretson's (a.k.a. Whitely's) multicomponent model (1980).

2-4PLNRM Nested logistic curves for modeling distractor items. Requires a scoring key. The model is broken into two components for the probability of endorsement. For successful endorsement the probability trace is the 1-4PL model, while for unsuccessful endorsement:

$$P(x = 0|\theta, \psi) = (1 - P_{1-4PL}(x = 1|\theta, \psi)) * P_{nominal}(x = k|\theta, \psi)$$

which is the product of the compliment of the dichotomous trace line with the nominal response model. In the nominal model, the slope parameters defined above are constrained to be 1's, while the last value of the ak is freely estimated.

spline Spline response models attempt to model the response curves uses non-linear and potentially non-monotonic patterns. The form is

$$P(x = 1|\theta, \eta) = \frac{1}{1 + exp(-(\eta_1 * X_1 + \eta_2 * X_2 + \cdots + \eta_n * X_n))}$$

where the X_n are from the spline design matrix X organized from the grid of θ values. B-splines with a natural or polynomial basis are supported, and the `intercept` input is set to `TRUE` by default.

HTML help files, exercises, and examples

To access examples, vignettes, and exercise files that have been generated with knitr please visit https://github.com/philchalmers/mirt/wiki.

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

References

Andrich, D. (1978). A rating scale formulation for ordered response categories. *Psychometrika, 43*, 561-573.

Bock, R. D., & Aitkin, M. (1981). Marginal maximum likelihood estimation of item parameters: Application of an EM algorithm. *Psychometrika, 46*(4), 443-459.

Bock, R. D., Gibbons, R., & Muraki, E. (1988). Full-Information Item Factor Analysis. *Applied Psychological Measurement, 12*(3), 261-280.

Bock, R. D. & Lieberman, M. (1970). Fitting a response model for n dichotomously scored items. *Psychometrika, 35*, 179-197.

Cai, L. (2010a). High-Dimensional exploratory item factor analysis by a Metropolis-Hastings Robbins-Monro algorithm. *Psychometrika, 75*, 33-57.

Cai, L. (2010b). Metropolis-Hastings Robbins-Monro algorithm for confirmatory item factor analysis. *Journal of Educational and Behavioral Statistics, 35*, 307-335.

Chalmers, R., P. (2012). mirt: A Multidimensional Item Response Theory Package for the R Environment. *Journal of Statistical Software, 48*(6), 1-29.

Chalmers, R., P. & Flora, D. (2014). Maximum-likelihood Estimation of Noncompensatory IRT Models with the MH-RM Algorithm. *Applied Psychological Measurement, 38*, 339-358.

Chen, W. H. & Thissen, D. (1997). Local dependence indices for item pairs using item response theory. *Journal of Educational and Behavioral Statistics, 22*, 265-289.

Lord, F. M. & Novick, M. R. (1968). Statistical theory of mental test scores. Addison-Wesley.

Ramsay, J. O. (1975). Solving implicit equations in psychometric data analysis. *Psychometrika, 40*, 337-360.

Rasch, G. (1960). Probabilistic models for some intelligence and attainment tests. *Danish Institute for Educational Research*.

Maydeu-Olivares, A., Hernandez, A. & McDonald, R. P. (2006). A Multidimensional Ideal Point Item Response Theory Model for Binary Data. *Multivariate Behavioral Research, 41*, 445-471.

Muraki, E. (1990). Fitting a polytomous item response model to Likert-type data. *Applied Psychological Measurement, 14*, 59-71.

Muraki, E. (1992). A generalized partial credit model: Application of an EM algorithm. *Applied Psychological Measurement, 16*, 159-176.

Muraki, E. & Carlson, E. B. (1995). Full-information factor analysis for polytomous item responses. *Applied Psychological Measurement, 19*, 73-90.

Samejima, F. (1969). Estimation of latent ability using a response pattern of graded scores. *Psychometrika Monographs*, 34.

Suh, Y. & Bolt, D. (2010). Nested logit models for multiple-choice item response data. *Psychometrika, 75*, 454-473.

Sympson, J. B. (1977). A model for testing with multidimensional items. Proceedings of the 1977 Computerized Adaptive Testing Conference.

Thissen, D. (1982). Marginal maximum likelihood estimation for the one-parameter logistic model. *Psychometrika, 47*, 175-186.

Varadhan, R. & Roland, C. (2008). Simple and Globally Convergent Methods for Accelerating the Convergence of Any EM Algorithm. *Scandinavian Journal of Statistics, 35*, 335-353.

Whitely, S. E. (1980). Multicomponent latent trait models for ability tests. *Psychometrika, 45*(4), 470-494.

Wood, R., Wilson, D. T., Gibbons, R. D., Schilling, S. G., Muraki, E., & Bock, R. D. (2003). *TESTFACT 4 for Windows: Test Scoring, Item Statistics, and Full-information Item Factor Analysis* [Computer software]. Lincolnwood, IL: Scientific Software International.

See Also

`bfactor`, `multipleGroup`, `mixedmirt`, `expand.table`, `key2binary`, `mod2values`, `extract.item`, `iteminfo`, `testinfo`, `probtrace`, `simdata`, `averageMI`, `fixef`, `extract.mirt`

Examples

```
## Not run:
#load LSAT section 7 data and compute 1 and 2 factor models
data <- expand.table(LSAT7)

(mod1 <- mirt(data, 1))
coef(mod1)
(mod2 <- mirt(data, 1, SE = TRUE)) #standard errors via the Oakes method
(mod2 <- mirt(data, 1, SE = TRUE, SE.type = 'SEM')) #standard errors with SEM method
coef(mod2)
(mod3 <- mirt(data, 1, SE = TRUE, SE.type = 'Richardson')) #with numerical Richardson method
residuals(mod1)
plot(mod1) #test score function
plot(mod1, type = 'trace') #trace lines
plot(mod2, type = 'info') #test information
plot(mod2, MI=200) #expected total score with 95% confidence intervals

#estimated 3PL model for item 5 only
(mod1.3PL <- mirt(data, 1, itemtype = c('2PL', '2PL', '2PL', '2PL', '3PL')))
coef(mod1.3PL)

#internally g and u pars are stored as logits, so usually a good idea to include normal prior
#  to help stabilize the parameters. For a value around .182 use a mean
#  of -1.5 (since 1 / (1 + exp(-(-1.5))) == .182)
model <- 'F = 1-5
         PRIOR = (5, g, norm, -1.5, 3)'
mod1.3PL.norm <- mirt(data, model, itemtype = c('2PL', '2PL', '2PL', '2PL', '3PL'))
coef(mod1.3PL.norm)
#limited information fit statistics
M2(mod1.3PL.norm)

#unidimensional ideal point model
idealpt <- mirt(data, 1, itemtype = 'ideal')
plot(idealpt, type = 'trace', facet_items = TRUE)
plot(idealpt, type = 'trace', facet_items = FALSE)

#two factors (exploratory)
mod2 <- mirt(data, 2)
coef(mod2)
summary(mod2, rotate = 'oblimin') #oblimin rotation
residuals(mod2)
plot(mod2)
plot(mod2, rotate = 'oblimin')

anova(mod1, mod2) #compare the two models
scoresfull <- fscores(mod2) #factor scores for each response pattern
head(scoresfull)
scorestable <- fscores(mod2, full.scores = FALSE) #save factor score table
head(scorestable)

#confirmatory (as an example, model is not identified since you need 3 items per factor)
```

```
# Two ways to define a confirmatory model: with mirt.model, or with a string

# these model definitions are equivalent
cmodel <- mirt.model('
   F1 = 1,4,5
   F2 = 2,3')
cmodel2 <- 'F1 = 1,4,5
            F2 = 2,3'

cmod <- mirt(data, cmodel)
# cmod <- mirt(data, cmodel2) # same as above
coef(cmod)
anova(cmod, mod2)
#check if identified by computing information matrix
(cmod <- mirt(data, cmodel, SE = TRUE))

###########
#data from the 'ltm' package in numeric format
pmod1 <- mirt(Science, 1)
plot(pmod1)
plot(pmod1, type = 'trace')
plot(pmod1, type = 'itemscore')
summary(pmod1)

#Constrain all slopes to be equal with the constrain = list() input or mirt.model() syntax
#first obtain parameter index
values <- mirt(Science,1, pars = 'values')
values #note that slopes are numbered 1,5,9,13, or index with values$parnum[values$name == 'a1']
(pmod1_equalslopes <- mirt(Science, 1, constrain = list(c(1,5,9,13))))
coef(pmod1_equalslopes)

# using mirt.model syntax, constrain all item slopes to be equal
model <- 'F = 1-4
          CONSTRAIN = (1-4, a1)'
(pmod1_equalslopes <- mirt(Science, model))
coef(pmod1_equalslopes)

coef(pmod1_equalslopes)
anova(pmod1_equalslopes, pmod1) #significantly worse fit with almost all criteria

pmod2 <- mirt(Science, 2)
summary(pmod2)
plot(pmod2, rotate = 'oblimin')
itemplot(pmod2, 1, rotate = 'oblimin')
anova(pmod1, pmod2)

#unidimensional fit with a generalized partial credit and nominal model
(gpcmod <- mirt(Science, 1, 'gpcm'))
coef(gpcmod)

#for the nominal model the lowest and highest categories are assumed to be the
#  theoretically lowest and highest categories that related to the latent trait(s)
(nomod <- mirt(Science, 1, 'nominal'))
```

```
coef(nomod) #ordering of ak values suggest that the items are indeed ordinal
anova(gpcmod, nomod)
itemplot(nomod, 3)

## example applying survey weights.
# weight the first half of the cases to be more representative of population
survey.weights <- c(rep(2, nrow(Science)/2), rep(1, nrow(Science)/2))
survey.weights <- survey.weights/sum(survey.weights) * nrow(Science)
unweighted <- mirt(Science, 1)
weighted <- mirt(Science, 1, survey.weights=survey.weights)

###########
#empirical dimensionality testing that includes 'guessing'

data(SAT12)
data <- key2binary(SAT12,
  key = c(1,4,5,2,3,1,2,1,3,1,2,4,2,1,5,3,4,4,1,4,3,3,4,1,3,5,1,3,1,5,4,5))

mod1 <- mirt(data, 1)
extract.mirt(mod1, 'time') #time elapsed for each estimation component

#optionally use Newton-Raphson for (generally) faster convergence in the M-step's
mod1 <- mirt(data, 1, optimizer = 'NR')
extract.mirt(mod1, 'time')

mod2 <- mirt(data, 2, optimizer = 'NR')
#difficulty converging with reduced quadpts, reduce TOL
mod3 <- mirt(data, 3, TOL = .001, optimizer = 'NR')
anova(mod1,mod2)
anova(mod2, mod3) #negative AIC, 2 factors probably best

#same as above, but using the QMCEM method for generally better accuracy in mod3
mod3 <- mirt(data, 3, method = 'QMCEM', TOL = .001, optimizer = 'NR')
anova(mod2, mod3)

#with fixed guessing parameters
mod1g <- mirt(data, 1, guess = .1)
coef(mod1g)

###########
#graded rating scale example

#make some data
set.seed(1234)
a <- matrix(rep(1, 10))
d <- matrix(c(1,0.5,-.5,-1), 10, 4, byrow = TRUE)
c <- seq(-1, 1, length.out=10)
data <- simdata(a, d + c, 2000, itemtype = rep('graded',10))

mod1 <- mirt(data, 1)
mod2 <- mirt(data, 1, itemtype = 'grsm')
coef(mod2)
anova(mod2, mod1) #not sig, mod2 should be preferred
```

```
itemplot(mod2, 1)
itemplot(mod2, 5)
itemplot(mod2, 10)

###########
# 2PL nominal response model example (Suh and Bolt, 2010)
data(SAT12)
SAT12[SAT12 == 8] <- NA #set 8 as a missing value
head(SAT12)

#correct answer key
key <- c(1,4,5,2,3,1,2,1,3,1,2,4,2,1,5,3,4,4,1,4,3,3,4,1,3,5,1,3,1,5,4,5)
scoredSAT12 <- key2binary(SAT12, key)
mod0 <- mirt(scoredSAT12, 1)

#for first 5 items use 2PLNRM and nominal
scoredSAT12[,1:5] <- as.matrix(SAT12[,1:5])
mod1 <- mirt(scoredSAT12, 1, c(rep('nominal',5),rep('2PL', 27)))
mod2 <- mirt(scoredSAT12, 1, c(rep('2PLNRM',5),rep('2PL', 27)), key=key)
coef(mod0)$Item.1
coef(mod1)$Item.1
coef(mod2)$Item.1
itemplot(mod0, 1)
itemplot(mod1, 1)
itemplot(mod2, 1)

#compare added information from distractors
Theta <- matrix(seq(-4,4,.01))
par(mfrow = c(2,3))
for(i in 1:5){
    info <- iteminfo(extract.item(mod0,i), Theta)
    info2 <- iteminfo(extract.item(mod2,i), Theta)
   plot(Theta, info2, type = 'l', main = paste('Information for item', i), ylab = 'Information')
    lines(Theta, info, col = 'red')
}
par(mfrow = c(1,1))

#test information
plot(Theta, testinfo(mod2, Theta), type = 'l', main = 'Test information', ylab = 'Information')
lines(Theta, testinfo(mod0, Theta), col = 'red')

###########
# using the MH-RM algorithm
data(LSAT7)
fulldata <- expand.table(LSAT7)
(mod1 <- mirt(fulldata, 1, method = 'MHRM'))

#Confirmatory models

#simulate data
a <- matrix(c(
1.5,NA,
0.5,NA,
```

```
1.0,NA,
1.0,0.5,
 NA,1.5,
 NA,0.5,
 NA,1.0,
 NA,1.0),ncol=2,byrow=TRUE)

d <- matrix(c(
-1.0,NA,NA,
-1.5,NA,NA,
 1.5,NA,NA,
 0.0,NA,NA,
3.0,2.0,-0.5,
2.5,1.0,-1,
2.0,0.0,NA,
1.0,NA,NA),ncol=3,byrow=TRUE)

sigma <- diag(2)
sigma[1,2] <- sigma[2,1] <- .4
items <- c(rep('2PL',4), rep('graded',3), '2PL')
dataset <- simdata(a,d,2000,items,sigma)

#analyses
#CIFA for 2 factor crossed structure

model.1 <- '
  F1 = 1-4
  F2 = 4-8
  COV = F1*F2'

#compute model, and use parallel computation of the log-likelihood
mirtCluster()
mod1 <- mirt(dataset, model.1, method = 'MHRM')
coef(mod1)
summary(mod1)
residuals(mod1)

#####
#bifactor
model.3 <- '
  G = 1-8
  F1 = 1-4
  F2 = 5-8'

mod3 <- mirt(dataset,model.3, method = 'MHRM')
coef(mod3)
summary(mod3)
residuals(mod3)
anova(mod1,mod3)

#####
#polynomial/combinations
data(SAT12)
```

```
data <- key2binary(SAT12,
               key = c(1,4,5,2,3,1,2,1,3,1,2,4,2,1,5,3,4,4,1,4,3,3,4,1,3,5,1,3,1,5,4,5))

model.quad <- '
       F1 = 1-32
  (F1*F1) = 1-32'


model.combo <- '
       F1 = 1-16
       F2 = 17-32
  (F1*F2) = 1-8'

(mod.quad <- mirt(data, model.quad))
summary(mod.quad)
(mod.combo <- mirt(data, model.combo))
anova(mod.quad, mod.combo)

#non-linear item and test plots
plot(mod.quad)
plot(mod.combo, type = 'SE')
itemplot(mod.quad, 1, type = 'score')
itemplot(mod.combo, 2, type = 'score')
itemplot(mod.combo, 2, type = 'infocontour')

## empirical histogram examples (normal, skew and bimodality)
#make some data
set.seed(1234)
a <- matrix(rlnorm(50, .2, .2))
d <- matrix(rnorm(50))
ThetaNormal <- matrix(rnorm(2000))
ThetaBimodal <- scale(matrix(c(rnorm(1000, -2), rnorm(1000,2)))) #bimodal
ThetaSkew <- scale(matrix(rchisq(2000, 3))) #positive skew
datNormal <- simdata(a, d, 2000, itemtype = '2PL', Theta=ThetaNormal)
datBimodal <- simdata(a, d, 2000, itemtype = '2PL', Theta=ThetaBimodal)
datSkew <- simdata(a, d, 2000, itemtype = '2PL', Theta=ThetaSkew)

normal <- mirt(datNormal, 1, empiricalhist = TRUE)
plot(normal, type = 'empiricalhist')
histogram(ThetaNormal, breaks=30)

bimodal <- mirt(datBimodal, 1, empiricalhist = TRUE)
plot(bimodal, type = 'empiricalhist')
histogram(ThetaBimodal, breaks=30)

skew <- mirt(datSkew, 1, empiricalhist = TRUE)
plot(skew, type = 'empiricalhist')
histogram(ThetaSkew, breaks=30)

#####
# non-linear parameter constraints with Rsolnp package (alabama supported as well):
# Find Rasch model subject to the constraint that the intercepts sum to 0
```

```
dat <- expand.table(LSAT6)

#free latent mean and variance terms
model <- 'Theta = 1-5
          MEAN = Theta
          COV = Theta*Theta'

#view how vector of parameters is organized internally
sv <- mirt(dat, model, itemtype = 'Rasch', pars = 'values')
sv[sv$est, ]

#constraint: create function for solnp to compute constraint, and declare value in eqB
eqfun <- function(p, optim_args) sum(p[1:5]) #could use browser() here, if it helps
LB <- c(rep(-15, 6), 1e-4) # more reasonable lower bound for variance term

mod <- mirt(dat, model, sv=sv, itemtype = 'Rasch', optimizer = 'solnp',
   solnp_args=list(eqfun=eqfun, eqB=0, LB=LB))
print(mod)
coef(mod)
(ds <- sapply(coef(mod)[1:5], function(x) x[,'d']))
sum(ds)

# same likelihood location as: mirt(dat, 1, itemtype = 'Rasch')


#######
# latent regression Rasch model

#simulate data
set.seed(1234)
N <- 1000

# covariates
X1 <- rnorm(N); X2 <- rnorm(N)
covdata <- data.frame(X1, X2)
Theta <- matrix(0.5 * X1 + -1 * X2 + rnorm(N, sd = 0.5))

#items and response data
a <- matrix(1, 20); d <- matrix(rnorm(20))
dat <- simdata(a, d, 1000, itemtype = '2PL', Theta=Theta)

#unconditional Rasch model
mod0 <- mirt(dat, 1, 'Rasch')

#conditional model using X1 and X2 as predictors of Theta
mod1 <- mirt(dat, 1, 'Rasch', covdata=covdata, formula = ~ X1 + X2)
coef(mod1, simplify=TRUE)
anova(mod0, mod1)

#bootstrapped confidence intervals
boot.mirt(mod1, R=5)

#draw plausible values for secondary analyses
```

```
pv <- fscores(mod1, plausible.draws = 10)
pvmods <- lapply(pv, function(x, covdata) lm(x ~ covdata$X1 + covdata$X2),
                 covdata=covdata)
#population characteristics recovered well, and can be averaged over
so <- lapply(pvmods, summary)
so

# compute Rubin's multiple imputation average
par <- lapply(so, function(x) x$coefficients[, 'Estimate'])
SEpar <- lapply(so, function(x) x$coefficients[, 'Std. Error'])
averageMI(par, SEpar)

############
# Example using Gauss-Hermite quadrature with custom input functions

library(fastGHQuad)
data(SAT12)
data <- key2binary(SAT12,
               key = c(1,4,5,2,3,1,2,1,3,1,2,4,2,1,5,3,4,4,1,4,3,3,4,1,3,5,1,3,1,5,4,5))
GH <- gaussHermiteData(50)
Theta <- matrix(GH$x)

# This prior works for uni- and multi-dimensional models
prior <- function(Theta, Etable){
    P <- grid <- GH$w / sqrt(pi)
    if(ncol(Theta) > 1)
        for(i in 2:ncol(Theta))
            P <- expand.grid(P, grid)
     if(!is.vector(P)) P <- apply(P, 1, prod)
     P
}

GHmod1 <- mirt(data, 1, optimizer = 'NR',
              technical = list(customTheta = Theta, customPriorFun = prior))
coef(GHmod1, simplify=TRUE)

Theta2 <- as.matrix(expand.grid(Theta, Theta))
GHmod2 <- mirt(data, 2, optimizer = 'NR', TOL = .0002,
              technical = list(customTheta = Theta2, customPriorFun = prior))
summary(GHmod2, suppress=.2)


## End(Not run)
```

`mirt.model` *Specify model loadings*

Description

The `mirt.model` function scans/reads user input to specify the confirmatory model. Item locations must be used in the specifications if no `itemnames` argument is supplied. This is called implicitly by estimation functions when a string is passed to the `model` argument.

Usage

```
mirt.model(input = NULL, itemnames = NULL, file = "", COV = NULL,
  quiet = TRUE, ...)
```

Arguments

`input`	input for writing out the model syntax. Can either be a string declaration of class character or the so-called Q-matrix or class `matrix` that specifies the model either with integer or logical values. If the Q-matrix method is chosen covariances terms can be specified with the `COV` input
`itemnames`	a character vector or factor indicating the item names. If a data.frame or matrix object is supplied the names will be extracted using `colnames(itemnames)`. Supplying this input allows the syntax to be specified with the raw item names rather than item locations
`file`	a input specifying an external file that declares the input.
`COV`	a symmetric, logical matrix used to declare which covariance terms are estimated
`quiet`	logical argument passed to `scan()` to suppress console read message
`...`	additional arguments for `scan()`

Details

Factors are first named and then specify which numerical items they affect (i.e., where the slope is not equal to 0), separated either by commas or by - to indicate a range of items. Products between factors may be specified by enclosing the left hand term within brackets. To finish the declaration of a model simply enter a blank line with only a carriage return (i.e., the 'enter' or 'return' key), or instead read in an input version of the model syntax.

There is an optional keyword for specifying the correlation between relationships between factors called `COV`, and non-linear factor products can be included by enclosing the product combination on the left hand side of the declaration (e.g., `(F1*F1)` would create a quadratic factor for `F1`).

COV Specify the relationship between the latent factors. Estimating a correlation between factors is declared by joining the two factors with an asterisk (e.g., F1*F2), or with an asterisk between three or more factors to estimate all the possible correlations (e.g., F1*F2*F3)

MEAN A comma separated list specifying which latent factor means to freely estimate. E.g., `MEAN = F1, F2` will free the latent means for factors F1 and F2

CONSTRAIN A bracketed, comma separated list specifying equality constrains between items. The input format is `CONSTRAIN = (items, ..., parameterName(s), OptionalGroup), (items, ..., paramete`
If `OptionalGroup` is omitted then the constraints are applied within all groups.

For example, in a single group 10-item dichotomous tests, using the default 2PL model, the first and last 5 item slopes (a1) can be constrained to be equal by using `CONSTRAIN = (1-5, a1), (6-10, a1)`, or some combination such as `CONSTRAIN = (1-3,4,5,a1), (6,7,8-10,a1)`.

When constraining parameters to be equal across items with different parameter names, a balanced bracketed vector must be supplied. E.g., setting the first slope for item 1 equal to the second slope in item 3 would be `CONSTRAIN = (1, 3, a1, a2)`

CONSTRAINB A bracketed, comma separate list specifying equality constrains between groups. The input format is `CONSTRAINB = (items, ..., parameterName), (items, ..., parameterName)`.

For example, in a two group 10-item dichotomous tests, using the default 2PL model, the first 5 item slopes (a1) can be constrained to be equal across both groups by using `CONSTRAINB = (1-5, a1)`, or some combination such as `CONSTRAINB = (1-3,4,5,a1)`

PRIOR A bracketed, comma separate list specifying prior parameter distributions. The input format is `PRIOR = (items, ..., parameterName, priorType, val1, val2, OptionalGroup), (items, ..., par` If `OptionalGroup` is omitted then the priors are defined for all groups. For example, in a single group 10-item dichotomous tests, using the default 2PL model, defining a normal prior of N(0,2) for the first 5 item intercepts (d) can be defined by `PRIOR = (1-5, d, norm, 0, 2)`

Currently supported priors are of the form: `(items, norm, mean, sd)` for the normal/Gaussian, `(items, lnorm, log_mean, log_sd)` for log-normal, `(items, beta, alpha, beta)` for beta, and `(items, expbeta, alpha, beta)` for the beta distribution after applying the function `plogis` to the input value (note, this is specifically for applying a beta prior to the lower-bound parameters in 3/4PL models)

LBOUND A bracketed, comma separate list specifying lower bounds for estimated parameters (used in optimizers such as `L-BFGS-B` and `nlminb`). The input format is `LBOUND = (items, ..., parameterName, va`

For example, in a single group 10-item dichotomous tests, using the 3PL model and setting lower bounds for the 'g' parameters for the first 5 items to 0.2 is accomplished with `LBOUND = (1-5, g, 0.2)`

UBOUND same as LBOUND, but specifying upper bounds in estimated parameters

START A bracketed, comma separate list specifying the starting values for individual parameters. The input is of the form `(items, ..., parameterName, value)`. For instance, setting the 10th and 12th to 15th item slope parameters (a1) to 1.0 is specified with `START = (10, 12-15, a1, 1.0)`

For more hands on control of the starting values pass the argument `pars = 'values'` through whatever estimation function is being used

FIXED A bracketed, comma separate list specifying which parameters should be fixed at their starting values (i.e., not freely estimated). The input is of the form `(items, ..., parameterName)`. For instance, fixing the 10th and 12th to 15th item slope parameters (a1) is accomplished with `FIXED = (10, 12-15, a1)`

For more hands on control of the estimated values pass the argument `pars = 'values'` through whatever estimation function is being used

FREE Equivalent to the `FIXED` input, except that parameters are freely estimated instead of fixed at their starting value

NEXPLORE Number of exploratory factors to extract. Usually this is not required because passing a numeric value to the `model` argument in the estimation function will generate an exploratory factor analysis model, however if different start values, priors, lower and upper bounds, etc, are desired then this input can be used

Value

Returns a model specification object to be used in `mirt`, `bfactor`, `multipleGroup`, or `mixedmirt`

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com> and Alexander Robitzsch

Examples

```
## Not run:

# interactively through the console (not run)
#model <- mirt.model()
#  F1 = 1,2,3,4-10
#  F2 = 10-20
#  (F1*F2) = 1,2,3,4-10
#  COV = F1*F2


#Or alternatively with a string input
s <- 'F1 = 1,2,3,4-10
      F2 = 10-20
      (F1*F2) = 1,2,3,4-10
      COV = F1*F2'
model <- mirt.model(s)

# strings can also be passed to the estimation functions directly,
#   which silently calls mirt.model(). E.g., using the string above:
# mod <- mirt(data, s)


#Q-matrix specification
Q <- matrix(c(1,1,1,0,0,0,0,0,0,1,1,1), ncol=2, dimnames = list(NULL, c('Factor1', 'Factor2')))
COV <- matrix(c(FALSE, TRUE, TRUE, FALSE), 2)
model <- mirt.model(Q, COV=COV)

## constrain various items slopes and all intercepts in single group model to be equal,
#   and use a log-normal prior for all the slopes
s <- 'F = 1-10
      CONSTRAIN = (1-3, 5, 6, a1), (1-10, d)
      PRIOR = (1-10, a1, lnorm, .2, .2)'
model <- mirt.model(s)


## constrain various items slopes and intercepts across groups for use in multipleGroup(),
#  and constrain first two slopes within 'group1' to be equal
s <- 'F = 1-10
      CONSTRAIN = (1-2, a1)
      CONSTRAINB = (1-3, 5, 6, a1), (1-10, d)'
model <- mirt.model(s)


## specify model using raw item names
data(data.read, package = 'sirt')
dat <- data.read

# syntax with variable names
mirtsyn2 <- "
       F1 = A1,B2,B3,C4
```

```
      F2 = A1-A4,C2,C4
      MEAN = F1
      COV = F1*F1, F1*F2
      CONSTRAIN=(A2-A4,a2),(A3,C2,d)
      PRIOR = (C3,A2-A4,a2,lnorm, .2, .2),(B3,d,norm,0,.0001)"
# create a mirt model
mirtmodel <- mirt.model(mirtsyn2, itemnames=dat)
# or equivelently:
# mirtmodel <- mirt.model(mirtsyn2, itemnames=colnames(dat))

# mod <- mirt(dat , mirtmodel)


## End(Not run)
```

`mirtCluster` *Define a parallel cluster object to be used in internal functions*

Description

This function defines a object that is placed in a relevant internal environment defined in mirt. Internal functions such as `calcLogLik`, `fscores`, etc, will utilize this object automatically to capitalize on parallel processing architecture. The object defined is a call from `parallel::makeCluster()`. Note that if you are defining other parallel objects (for simulation desings, for example) it is not recommended to define a mirtCluster.

Usage

```
mirtCluster(spec, ..., remove = FALSE)
```

Arguments

`spec`	input that is passed to `parallel::makeCluster()`. If no input is given the maximum number of available local cores will be used
`...`	additional arguments to pass to `parallel::makeCluster`
`remove`	logical; remove previously defined `mirtCluster()`?

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

Examples

```
## Not run:

#make 4 cores available for parallel computing
mirtCluster(4)
```

```
#' #stop and remove cores
mirtCluster(remove = TRUE)


## End(Not run)
```

MixedClass-class *Class "MixedClass"*

Description

Defines the object returned from `mixedmirt`.

Slots

`Call`: function call

`Data`: list of data, sometimes in different forms

`Options`: list of estimation options

`Fit`: a list of fit information

`Model`: a list of model-based information

`ParObjects`: a list of the S4 objects used during estimation

`OptimInfo`: a list of arguments from the optimization process

`Internals`: a list of internal arguments for secondary computations (inspecting this object is generally not required)

`vcov`: a matrix represented the asymtotic covariance matrix of the parameter estimates

`time`: a data.frame indicating the breakdown of computation times in seconds

Methods

coef `signature(object = "MixedClass")`

print `signature(x = "MixedClass")`

residuals `signature(object = "MixedClass")`

show `signature(object = "MixedClass")`

summary `signature(object = "MixedClass")`

logLik `signature(object = "MixedClass")`

anova `signature(object = "MixedClass")`

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

mixedmirt *Mixed effects modeling for MIRT models*

Description

`mixedmirt` fits MIRT models using FIML estimation to dichotomous and polytomous IRT models conditional on fixed and random effect of person and item level covariates. This can also be understood as 'explanatory IRT' if only fixed effects are modeled, or multilevel/mixed IRT if random and fixed effects are included. The method uses the MH-RM algorithm exclusively. Additionally, computation of the log-likelihood can be sped up by using parallel estimation via `mirtCluster`.

Usage

```
mixedmirt(data, covdata = NULL, model, fixed = ~1, random = NULL,
  itemtype = "Rasch", lr.fixed = ~1, lr.random = NULL,
  itemdesign = NULL, constrain = NULL, pars = NULL,
  return.design = FALSE, SE = TRUE, internal_constraints = TRUE,
  technical = list(SEtol = 1e-04), ...)
```

Arguments

`data` — a `matrix` or `data.frame` that consists of numerically ordered data, with missing data coded as `NA`

`covdata` — a `data.frame` that consists of the `nrow(data)` by `K` 'person level' fixed and random predictors

`model` — an object returned from, or a string to be passed to, `mirt.model()` to declare how the IRT model is to be estimated. See `mirt.model` for more details

`fixed` — a right sided R formula for specifying the fixed effect (aka 'explanatory') predictors from `covdata` and `itemdesign`. To estimate the intercepts for each item the keyword `items` is reserved and automatically added to the `itemdesign` input. If any polytomous items are being model the `items` are argument is not valid since all intercept parameters are freely estimated and identified with the parameterizations found in `mirt`, and the first column in the fixed design matrix (commonly the intercept or a reference group) is omitted

`random` — a right sided formula or list of formulas containing crossed random effects of the form `v1 + ... v_n | G`, where `G` is the grouping variable and `v_n` are random numeric predictors within each group. If no intercept value is specified then by default the correlations between the `v`'s and `G` are estimated, but can be suppressed by including the `~ -1 + ...` or 0 constant. `G` may contain interaction terms, such as `group:items` to include cross or person-level interactions effects

`itemtype` — same as itemtype in `mirt`, except when the `fixed` or `random` inputs are used does not support the following item types: `c('PC2PL', 'PC3PL', '2PLNRM', '3PLNRM', '3PLuNRM', '4PL`

`lr.fixed` — an R formula (or list of formulas) to specify regression effects in the latent variables from the variables in `covdata`. This is used to construct models such as the so-called 'latent regression model' to explain person-level ability/trait differences. If a named list of formulas is supplied (where the names correspond to the latent trait names in `model`) then specific regression effects can be estimated for each factor. Supplying a single formula will estimate the regression parameters for all latent traits by default.

`lr.random` a list of random effect terms for modeling variability in the latent trait scores, where the syntax uses the same style as in the `random` argument. Useful for building so-called 'multilevel IRT' models which are non-Rasch (multilevel Rasch models do not technically require these because they can be built using the `fixed` and `random` inputs alone)

`itemdesign` a `data.frame` object used to create a design matrix for the items, where each `nrow(itemdesign) == nitems` and the number of columns is equal to the number of fixed effect predictors (i.e., item intercepts). By default an `items` variable is reserved for modeling the item intercept parameters

`constrain` a list indicating parameter equality constrains. See `mirt` for more detail

`pars` used for parameter starting values. See `mirt` for more detail

`return.design` logical; return the design matrices before they have (potentially) been reassigned?

`SE` logical; compute the standard errors by approximating the information matrix using the MHRM algorithm? Default is TRUE

`internal_constraints` logical; use the internally defined constraints for constraining effects across persons and items? Default is TRUE. Setting this to FALSE runs the risk of underidentification

`technical` the technical list passed to the MH-RM estimation engine, with the SEtol default increased to .0001. Additionally, the argument `RANDSTART` is available to indicate at which iteration (during the burn-in stage) the additional random effect variables should begin to be approximated (i.e., elements in `lr.random` and `random`). The default for `RANDSTART` is to start at iteration 100, and when random effects are included the default number of burn-in iterations is incrased from 150 to 200. See `mirt` for further details

`...` additional arguments to be passed to the MH-RM estimation engine. See `mirt` for more details and examples

Details

For dichotomous response models, `mixedmirt` follows the general form

$$P(x = 1|\theta, \psi) = g + \frac{(u - g)}{1 + exp(-1 * [\theta a + X\beta + Z\delta])}$$

where X is a design matrix with associated β fixed effect intercept coefficients, and Z is a design matrix with associated δ random effects for the intercepts. For simplicity and easier interpretation, the unique item intercept values typically found in $X\beta$ are extracted and reassigned within mirt's 'intercept' parameters (e.g., `'d'`). To observe how the design matrices are structured prior to reassignment and estimation pass the argument `return.design = TRUE`.

Polytomous IRT models follow a similar format except the item intercepts are automatically estimated internally, rendering the `items` argument in the fixed formula redundant and therefore must be omitted from the specification. If there are a mixture of dichotomous and polytomous items the intercepts for the dichotomous models are also estimated for consistency.

The decomposition of the θ parameters is also possible to form latent regression and multilevel IRT models by using the `lr.fixed` and `lr.random` inputs. These effects decompose θ such that

$$\theta = V\Gamma + W\zeta + \epsilon$$

where V and W are fixed and random effects design matrices for the associated coefficients.

To simulate maximum a posteriori estimates for the random effect terms use the `randef` function.

Value

function returns an object of class `MixedClass` (MixedClass-class).

Author(s)

Phil Chalmers `<rphilip.chalmers@gmail.com>`

References

Chalmers, R. P. (2015). Extended Mixed-Effects Item Response Models with the MH-RM Algorithm. *Journal of Educational Measurement, 52*, 200-222.

See Also

`mirt`, `randef`, `fixef`, `boot.mirt`

Examples

```
## Not run:

#make some data
set.seed(1234)
N <- 750
a <- matrix(rlnorm(10,.3,1),10,1)
d <- matrix(rnorm(10), 10)
Theta <- matrix(sort(rnorm(N)))
pseudoIQ <- Theta * 5 + 100  + rnorm(N, 0 , 5)
pseudoIQ <- (pseudoIQ - mean(pseudoIQ))/10  #rescale variable for numerical stability
group <- factor(rep(c('G1','G2','G3'), each = N/3))
data <- simdata(a,d,N, itemtype = rep('2PL',10), Theta=Theta)
covdata <- data.frame(group, pseudoIQ)
#use parallel computing
mirtCluster()

#specify IRT model
model <- 'Theta = 1-10'
```

```
#model with no person predictors
mod0 <- mirt(data, model, itemtype = 'Rasch')

#group as a fixed effect predictor (aka, uniform dif)
mod1 <- mixedmirt(data, covdata, model, fixed = ~ 0 + group + items)
anova(mod0, mod1)
summary(mod1)
coef(mod1)

#same model as above in lme4
wide <- data.frame(id=1:nrow(data),data,covdata)
long <- reshape2::melt(wide, id.vars = c('id', 'group', 'pseudoIQ'))
library(lme4)
lmod0 <- glmer(value ~ 0 + variable + (1|id), long, family = binomial)
lmod1 <- glmer(value ~ 0 + group + variable + (1|id), long, family = binomial)
anova(lmod0, lmod1)

#model using 2PL items instead of Rasch
mod1b <- mixedmirt(data, covdata, model, fixed = ~ 0 + group + items, itemtype = '2PL')
anova(mod1, mod1b) #better with 2PL models using all criteria (as expected, given simdata pars)

#continuous predictor with group
mod2 <- mixedmirt(data, covdata, model, fixed = ~ 0 + group + items + pseudoIQ)
summary(mod2)
anova(mod1b, mod2)

#view fixed design matrix with and without unique item level intercepts
withint <- mixedmirt(data, covdata, model, fixed = ~ 0 + items + group, return.design = TRUE)
withoutint <- mixedmirt(data, covdata, model, fixed = ~ 0 + group, return.design = TRUE)

#notice that in result above, the intercepts 'items1 to items 10' were reassigned to 'd'
head(withint$X)
tail(withint$X)
head(withoutint$X) #no intercepts design here to be reassigned into item intercepts
tail(withoutint$X)

###################################################
### random effects
#make the number of groups much larger
covdata$group <- factor(rep(paste0('G',1:50), each = N/50))

#random groups
rmod1 <- mixedmirt(data, covdata, 1, fixed = ~ 0 + items, random = ~ 1|group)
summary(rmod1)
coef(rmod1)

#random groups and random items
rmod2 <- mixedmirt(data, covdata, 1, random = list(~ 1|group, ~ 1|items))
summary(rmod2)
eff <- randef(rmod2) #estimate random effects

#random slopes with fixed intercepts (suppressed correlation)
```

```
rmod3 <- mixedmirt(data, covdata, 1, fixed = ~ 0 + items, random = ~ -1 + pseudoIQ|group)
summary(rmod3)
eff <- randef(rmod3)
str(eff)

###################################################
##LLTM, and 2PL version of LLTM
data(SAT12)
data <- key2binary(SAT12,
                   key = c(1,4,5,2,3,1,2,1,3,1,2,4,2,1,5,3,4,4,1,4,3,3,4,1,3,5,1,3,1,5,4,5))
model <- 'Theta = 1-32'

# Suppose that the first 16 items were suspected to be easier than the last 16 items,
#   and we wish to test this item structure hypothesis (more intercept designs are possible
#   by including more columns).
itemdesign <- data.frame(itemorder = factor(c(rep('easier', 16), rep('harder', 16))))

#notice that the 'fixed = ~ ... + items' argument is omitted
LLTM <- mixedmirt(data, model = model, fixed = ~ 0 + itemorder, itemdesign = itemdesign,
   SE = TRUE) # SE argument ensures that the information matrix is computed accurately
summary(LLTM)
coef(LLTM)
wald(LLTM)
L <- matrix(c(-1, 1, 0), 1)
wald(LLTM, L) #first half different from second

#compare to items with estimated slopes (2PL)
twoPL <- mixedmirt(data, model = model, fixed = ~ 0 + itemorder, itemtype = '2PL',
                   itemdesign = itemdesign)
#twoPL not mixing too well (AR should be between .2 and .5), decrease MHcand
twoPL <- mixedmirt(data, model = model, fixed = ~ 0 + itemorder, itemtype = '2PL',
                  itemdesign = itemdesign, technical = list(MHcand = 0.8))
anova(twoPL, LLTM) #much better fit
summary(twoPL)
coef(twoPL)

wald(twoPL)
L <- matrix(0, 1, 34)
L[1, 1] <- 1
L[1, 2] <- -1
wald(twoPL, L) #n.s., which is the correct conclusion. Rasch approach gave wrong inference

##LLTM with item error term
LLTMwithError <- mixedmirt(data, model = model, fixed = ~ 0 + itemorder, random = ~ 1|items,
    itemdesign = itemdesign)
summary(LLTMwithError)
#large item level variance after itemorder is regressed; not a great predictor of item difficulty
coef(LLTMwithError)

###################################################
### Polytomous example

#make an arbitrary group difference
```

```
covdat <- data.frame(group = rep(c('m', 'f'), nrow(Science)/2))

#partial credit model
mod <- mixedmirt(Science, covdat, model=1, fixed = ~ 0 + group)
coef(mod)

#gpcm to estimate slopes
mod2 <- mixedmirt(Science, covdat, model=1, fixed = ~ 0 + group,
                 itemtype = 'gpcm')
summary(mod2)
anova(mod, mod2)

#graded model
mod3 <- mixedmirt(Science, covdat, model=1, fixed = ~ 0 + group,
                 itemtype = 'graded')
coef(mod3)


###################################################
# latent regression with Rasch and 2PL models

set.seed(1)
n <- 300
a <- matrix(1, 10)
d <- matrix(rnorm(10))
Theta <- matrix(c(rnorm(n, 0), rnorm(n, 1), rnorm(n, 2)))
covdata <- data.frame(group=rep(c('g1','g2','g3'), each=n))
dat <- simdata(a, d, N=n*3, Theta=Theta, itemtype = '2PL')

#had we known the latent abilities, we could have computed the regression coefs
summary(lm(Theta ~ covdata$group))

#but all we have is observed test data. Latent regression helps to recover these coefs
#Rasch model approach (and mirt equivalent)
rmod0 <- mirt(dat, 1, 'Rasch') # unconditional

# these two models are equivalent
rmod1a <- mirt(dat, 1, 'Rasch', covdata = covdata, formula = ~ group)
rmod1b <- mixedmirt(dat, covdata, 1, fixed = ~ 0 + items + group)
anova(rmod0, rmod1b)
coef(rmod1a, simplify=TRUE)
summary(rmod1b)

# 2PL, requires different input to allow Theta variance to remain fixed
mod0 <- mirt(dat, 1) # unconditional
mod1a <- mirt(dat, 1, covdata = covdata, formula = ~ group, itemtype = '2PL')
mod1b <- mixedmirt(dat, covdata, 1, fixed = ~ 0 + items, lr.fixed = ~group, itemtype = '2PL')
anova(mod0, mod1b)
coef(mod1a)$lr.betas
summary(mod1b)

# specifying specific regression effects is accomplished by passing a list of formula
model <- 'F1 = 1-5
```

```
         F2 = 6-10'
covdata$contvar <- rnorm(nrow(covdata))
mod2 <- mirt(dat, model, itemtype = 'Rasch', covdata=covdata,
        formula = list(F1 = ~ group + contvar, F2 = ~ group))
coef(mod2)[11:12]
mod2b <- mixedmirt(dat, covdata, model, fixed = ~ 0 + items,
        lr.fixed = list(F1 = ~ group + contvar, F2 = ~ group))
summary(mod2b)

####################################################
## Simulated Multilevel Rasch Model

set.seed(1)
N <- 2000
a <- matrix(rep(1,10),10,1)
d <- matrix(rnorm(10))
cluster = 100
random_intercept = rnorm(cluster,0,1)
Theta = numeric()
for (i in 1:cluster)
    Theta <- c(Theta, rnorm(N/cluster,0,1) + random_intercept[i])

group = factor(rep(paste0('G',1:cluster), each = N/cluster))
covdata <- data.frame(group)
dat <- simdata(a,d,N, itemtype = rep('2PL',10), Theta=matrix(Theta))

# null model
mod1 <- mixedmirt(dat, covdata, 1, fixed = ~ 0 + items, random = ~ 1|group)
summary(mod1)

# include level 2 predictor for 'group' variance
covdata$group_pred <- rep(random_intercept, each = N/cluster)
mod2 <- mixedmirt(dat, covdata, 1, fixed = ~ 0 + items + group_pred, random = ~ 1|group)

# including group means predicts nearly all variability in 'group'
summary(mod2)
anova(mod1, mod2)

# can also be fit for Rasch/non-Rasch models with the lr.random input
mod1b <- mixedmirt(dat, covdata, 1, fixed = ~ 0 + items, lr.random = ~ 1|group)
summary(mod1b)

mod2b <- mixedmirt(dat, covdata, 1, fixed = ~ 0 + items + group_pred, lr.random = ~ 1|group)
summary(mod2b)
anova(mod1b, mod2b)

mod3 <- mixedmirt(dat, covdata, 1, fixed = ~ 0 + items, lr.random = ~ 1|group, itemtype = '2PL')
summary(mod3)
anova(mod1b, mod3)

head(cbind(randef(mod3)$group, random_intercept))
```

```
## End(Not run)
```

mod2values — *Convert an estimated mirt model to a data.frame*

Description

Given an estimated model from any of mirt's model fitting functions this function will convert the model parameters into the design data frame of starting values and other parameter characteristics (similar to using the `pars = 'values'` for obtaining starting values).

Usage

```
mod2values(x)
```

Arguments

x	an estimated model x from the mirt package

Author(s)

Phil Chalmers `<rphilip.chalmers@gmail.com>`

See Also

`extract.mirt`

Examples

```
## Not run:
dat <- expand.table(LSAT7)
mod <- mirt(dat, 1)
values <- mod2values(mod)
values

#use the converted values as starting values in a new model, and reduce TOL
mod2 <- mirt(dat, 1, pars = values, TOL=1e-5)


## End(Not run)
```

multipleGroup *Multiple Group Estimation*

Description

multipleGroup performs a full-information maximum-likelihood multiple group analysis for any combination of dichotomous and polytomous data under the item response theory paradigm using either Cai's (2010) Metropolis-Hastings Robbins-Monro (MHRM) algorithm or with an EM algorithm approach. This function may be used for detecting differential item functioning (DIF), thought the DIF function may provide a more convenient approach.

Usage

```
multipleGroup(data, model, group, invariance = "", method = "EM",
  rotate = "oblimin", ...)
```

Arguments

- `data` — a `matrix` or `data.frame` that consists of numerically ordered data, with missing data coded as `NA`
- `model` — string to be passed to, or a model object returned from, `mirt.model` declaring how the global model is to be estimated (useful to apply constraints here)
- `group` — a character vector indicating group membership
- `invariance` — a character vector containing the following possible options:
  - `'free_means'` for freely estimating all latent means (reference group constrained to 0)
  - `'free_var'` for freely estimating all latent variances (reference group constrained to 1's)
  - `'slopes'` to constrain all the slopes to be equal across all groups
  - `'intercepts'` to constrain all the intercepts to be equal across all groups, note for nominal models this also includes the category specific slope parameters

  Additionally, specifying specific item name bundles (from `colnames(data)`) will constrain all freely estimated parameters in each item to be equal across groups. This is useful for selecting 'anchor' items for vertical and horizontal scaling, and for detecting differential item functioning (DIF) across groups
- `method` — a character object that is either `'EM'`, `'QMCEM'`, or `'MHRM'` (default is `'EM'`). See `mirt` for details
- `rotate` — rotation if models are exploratory (see `mirt` for details)
- `...` — additional arguments to be passed to the estimation engine. See `mirt` for details and examples

Details

By default the estimation in `multipleGroup` assumes that the models are maximally independent, and therefore could initially be performed by sub-setting the data and running identical models with `mirt` and aggregating the results (e.g., log-likelihood). However, constrains may be automatically imposed across groups by invoking various `invariance` keywords. Users may also supply a list of parameter equality constraints to by `constrain` argument, of define equality constraints using the `mirt.model` syntax (recommended).

Value

function returns an object of class `MultipleGroupClass` (MultipleGroupClass-class).

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

See Also

`mirt`, `DIF`, `extract.group`, `DTF`

Examples

```
## Not run:
#single factor
set.seed(12345)
a <- matrix(abs(rnorm(15,1,.3)), ncol=1)
d <- matrix(rnorm(15,0,.7),ncol=1)
itemtype <- rep('2PL', nrow(a))
N <- 1000
dataset1 <- simdata(a, d, N, itemtype)
dataset2 <- simdata(a, d, N, itemtype, mu = .1, sigma = matrix(1.5))
dat <- rbind(dataset1, dataset2)
group <- c(rep('D1', N), rep('D2', N))
models <- 'F1 = 1-15'

mod_configural <- multipleGroup(dat, models, group = group) #completely separate analyses
#limited information fit statistics
M2(mod_configural)

mod_metric <- multipleGroup(dat, models, group = group, invariance=c('slopes')) #equal slopes
#equal intercepts, free variance and means
mod_scalar2 <- multipleGroup(dat, models, group = group,
                             invariance=c('slopes', 'intercepts', 'free_var','free_means'))
mod_scalar1 <- multipleGroup(dat, models, group = group,  #fixed means
                               invariance=c('slopes', 'intercepts', 'free_var'))
mod_fullconstrain <- multipleGroup(dat, models, group = group,
                               invariance=c('slopes', 'intercepts'))
slot(mod_fullconstrain, 'time') #time of estimation components

#optionally use Newton-Raphson for (generally) faster convergence in the M-step's
mod_fullconstrain <- multipleGroup(dat, models, group = group, optimizer = 'NR',
```

```
                             invariance=c('slopes', 'intercepts'))
slot(mod_fullconstrain, 'time') #time of estimation components

summary(mod_scalar2)
coef(mod_scalar2, simplify=TRUE)
residuals(mod_scalar2)
plot(mod_configural)
plot(mod_configural, type = 'info')
plot(mod_configural, type = 'trace')
plot(mod_configural, type = 'trace', which.items = 1:4)
itemplot(mod_configural, 2)
itemplot(mod_configural, 2, type = 'RE')

anova(mod_metric, mod_configural) #equal slopes only
anova(mod_scalar2, mod_metric) #equal intercepts, free variance and mean
anova(mod_scalar1, mod_scalar2) #fix mean
anova(mod_fullconstrain, mod_scalar1) #fix variance


#test whether first 6 slopes should be equal across groups
values <- multipleGroup(dat, models, group = group, pars = 'values')
values
constrain <- list(c(1, 63), c(5,67), c(9,71), c(13,75), c(17,79), c(21,83))
equalslopes <- multipleGroup(dat, models, group = group, constrain = constrain)
anova(equalslopes, mod_configural)

#same as above, but using mirt.model syntax
newmodel <- '
    F = 1-15
    CONSTRAINB = (1-6, a1)'
equalslopes <- multipleGroup(dat, newmodel, group = group)
coef(equalslopes, simplify=TRUE)

############
# vertical scaling (i.e., equating when groups answer items others do not)
dat2 <- dat
dat2[group == 'D1', 1:2] <- dat2[group != 'D1', 14:15] <- NA
head(dat2)
tail(dat2)

# items with missing reponses need to be constrained across groups for identification
nms <- colnames(dat2)
mod <- multipleGroup(dat2, 1, group, invariance = nms[c(1:2, 14:15)])

# this will throw an error without proper constraints (SEs cannot be computed either)
# mod <- multipleGroup(dat2, 1, group)

# model still does not have anchors, therefore need to add a few (here use items 3-5)
mod_anchor <- multipleGroup(dat2, 1, group,
                           invariance = c(nms[c(1:5, 14:15)], 'free_means', 'free_var'))
coef(mod_anchor, simplify=TRUE)

# check if identified by computing information matrix
```

```
mod_anchor <- multipleGroup(dat2, 1, group, pars = mod2values(mod_anchor), TOL=NaN, SE=TRUE,
                           invariance = c(nms[c(1:5, 14:15)], 'free_means', 'free_var'))
mod_anchor
coef(mod_anchor)
coef(mod_anchor, printSE=TRUE)


#############
#DIF test for each item (using all other items as anchors)
itemnames <- colnames(dat)
refmodel <- multipleGroup(dat, models, group = group, SE=TRUE,
                             invariance=c('free_means', 'free_var', itemnames))

#loop over items (in practice, run in parallel to increase speed). May be better to use ?DIF
estmodels <- vector('list', ncol(dat))
for(i in 1:ncol(dat))
   estmodels[[i]] <- multipleGroup(dat, models, group = group, verbose = FALSE, calcNull=FALSE,
                             invariance=c('free_means', 'free_var', itemnames[-i]))

(anovas <- lapply(estmodels, anova, object2=refmodel, verbose=FALSE))

#family-wise error control
p <- do.call(rbind, lapply(anovas, function(x) x[2, 'p']))
p.adjust(p, method = 'BH')

#same as above, except only test if slopes vary (1 df)
#constrain all intercepts
estmodels <- vector('list', ncol(dat))
for(i in 1:ncol(dat))
   estmodels[[i]] <- multipleGroup(dat, models, group = group, verbose = FALSE, calcNull=FALSE,
                             invariance=c('free_means', 'free_var', 'intercepts',
                             itemnames[-i]))

(anovas <- lapply(estmodels, anova, object2=refmodel, verbose=FALSE))

#quickly test with Wald test using DIF()
mod_configural2 <- multipleGroup(dat, models, group = group, SE=TRUE)
DIF(mod_configural2, which.par = c('a1', 'd'), Wald=TRUE, p.adjust = 'fdr')

#############
#multiple factors

a <- matrix(c(abs(rnorm(5,1,.3)), rep(0,15),abs(rnorm(5,1,.3)),
     rep(0,15),abs(rnorm(5,1,.3))), 15, 3)
d <- matrix(rnorm(15,0,.7),ncol=1)
mu <- c(-.4, -.7, .1)
sigma <- matrix(c(1.21,.297,1.232,.297,.81,.252,1.232,.252,1.96),3,3)
itemtype <- rep('2PL', nrow(a))
N <- 1000
dataset1 <- simdata(a, d, N, itemtype)
dataset2 <- simdata(a, d, N, itemtype, mu = mu, sigma = sigma)
dat <- rbind(dataset1, dataset2)
group <- c(rep('D1', N), rep('D2', N))
```

```
#group models
model <- '
   F1 = 1-5
   F2 = 6-10
   F3 = 11-15'

#define mirt cluster to use parallel architecture
mirtCluster()

#EM approach (not as accurate with 3 factors, but generally good for quick model comparisons)
mod_configural <- multipleGroup(dat, model, group = group) #completely separate analyses
mod_metric <- multipleGroup(dat, model, group = group, invariance=c('slopes')) #equal slopes
mod_fullconstrain <- multipleGroup(dat, model, group = group, #equal means, slopes, intercepts
                             invariance=c('slopes', 'intercepts'))

anova(mod_metric, mod_configural)
anova(mod_fullconstrain, mod_metric)

#same as above, but with MHRM (generally  more accurate with 3+ factors, but slower)
mod_configural <- multipleGroup(dat, model, group = group, method = 'MHRM')
mod_metric <- multipleGroup(dat, model, group = group, invariance=c('slopes'), method = 'MHRM')
mod_fullconstrain <- multipleGroup(dat, model, group = group, method = 'MHRM',
                             invariance=c('slopes', 'intercepts'))

anova(mod_metric, mod_configural)
anova(mod_fullconstrain, mod_metric)

############
#polytomous item example
set.seed(12345)
a <- matrix(abs(rnorm(15,1,.3)), ncol=1)
d <- matrix(rnorm(15,0,.7),ncol=1)
d <- cbind(d, d-1, d-2)
itemtype <- rep('graded', nrow(a))
N <- 1000
dataset1 <- simdata(a, d, N, itemtype)
dataset2 <- simdata(a, d, N, itemtype, mu = .1, sigma = matrix(1.5))
dat <- rbind(dataset1, dataset2)
group <- c(rep('D1', N), rep('D2', N))
model <- 'F1 = 1-15'

mod_configural <- multipleGroup(dat, model, group = group)
plot(mod_configural)
plot(mod_configural, type = 'SE')
itemplot(mod_configural, 1)
itemplot(mod_configural, 1, type = 'info')
plot(mod_configural, type = 'trace') # messy, score function typically better
plot(mod_configural, type = 'itemscore')

fs <- fscores(mod_configural, full.scores = FALSE)
head(fs[["D1"]])
fscores(mod_configural, method = 'EAPsum', full.scores = FALSE)
```

```
# constrain slopes within each group to be equal (but not across groups)
model2 <- 'F1 = 1-15
           CONSTRAIN = (1-15, a1)'
mod_configural2 <- multipleGroup(dat, model2, group = group)
plot(mod_configural2, type = 'SE')
plot(mod_configural2, type = 'RE')
itemplot(mod_configural2, 10)

############
## empirical histogram example (normal and bimodal groups)
set.seed(1234)
a <- matrix(rlnorm(50, .2, .2))
d <- matrix(rnorm(50))
ThetaNormal <- matrix(rnorm(2000))
ThetaBimodal <- scale(matrix(c(rnorm(1000, -2), rnorm(1000,2)))) #bimodal
Theta <- rbind(ThetaNormal, ThetaBimodal)
dat <- simdata(a, d, 4000, itemtype = '2PL', Theta=Theta)
group <- rep(c('G1', 'G2'), each=2000)

EH <- multipleGroup(dat, 1, group=group, empiricalhist = TRUE, invariance = colnames(dat))
coef(EH, simplify=TRUE)
plot(EH, type = 'empiricalhist', npts = 60)

#dif test for item 1
EH1 <- multipleGroup(dat, 1, group=group, empiricalhist = TRUE, invariance = colnames(dat)[-1])
anova(EH, EH1)


## End(Not run)
```

MultipleGroupClass-class

Class "MultipleGroupClass"

Description

Defines the object returned from multipleGroup.

Slots

Call: function call

Data: list of data, sometimes in different forms

Options: list of estimation options

Fit: a list of fit information

Model: a list of model-based information

ParObjects: a list of the S4 objects used during estimation

OptimInfo: a list of arguments from the optimization process

`Internals`: a list of internal arguments for secondary computations (inspecting this object is generally not required)

`vcov`: a matrix represented the asymtotic covariance matrix of the parameter estimates

`time`: a data.frame indicating the breakdown of computation times in seconds

Methods

coef `signature(object = "MultipleGroupClass")`

print `signature(x = "MultipleGroupClass")`

show `signature(object = "MultipleGroupClass")`

anova `signature(object = "MultipleGroupClass")`

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

`numerical_deriv` *Compute numerical derivatives*

Description

Compute numerical derivatives using forward/backword difference, central difference, or Richardson extropolation.

Usage

```
numerical_deriv(par, f, ..., delta = 1e-05, gradient = TRUE,
  type = "forward")
```

Arguments

`par`	a vector of parameters
`f`	the objective function being evaluated
`...`	additional arguments to be passed to `f` and the `numDeriv` package when the Richardson type is used
`delta`	the term used to perturb the `f` function. Default is 1e-5
`gradient`	logical; compute the gradient terms? If FALSE then the Hessian is computed instead
`type`	type of difference to compute. Can be either `'forward'` for the forward difference, `'central'` for the central difference, or `'Richardson'` for the Richardson extropolation. Backword difference is acheived by supplying a negative `delta` value

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

Examples

```
## Not run:
f <- function(x) 3*x[1]^3 - 4*x[2]^2
par <- c(3,8)

# grad = 9 * x^2 , -8 * y
(actual <- c(9 * par[1]^2, -8 * par[2]))
numerical_deriv(par, f, type = 'forward')
numerical_deriv(par, f, type = 'central')
numerical_deriv(par, f, type = 'Richardson')

# hessian = h11 -> 18 * x, h22 -> -8, h12 -> h21 -> 0
(actual <- matrix(c(18 * par[1], 0, 0, -8), 2, 2))
numerical_deriv(par, f, type = 'forward', gradient = FALSE)
numerical_deriv(par, f, type = 'central', gradient = FALSE)
numerical_deriv(par, f, type = 'Richardson', gradient = FALSE)


## End(Not run)
```

---

personfit *Person fit statistics*

Description

personfit calculates the Zh values from Drasgow, Levine and Williams (1985) for unidimensional and multidimensional models. For Rasch models infit and outfit statistics are also produced. The returned object is a data.frame consisting either of the tabulated data or full data with the statistics appended to the rightmost columns.

Usage

```
personfit(x, method = "EAP", Theta = NULL, stats.only = TRUE, ...)
```

Arguments

x	a computed model object of class SingleGroupClass or MultipleGroupClass
method	type of factor score estimation method. See fscores for more detail
Theta	a matrix of factor scores used for statistics that require emperical estimates. If supplied, arguments typically passed to fscores() will be ignored and these values will be used instead
stats.only	logical; return only the person fit statistics without their associated response pattern?
...	additional arguments to be passed to fscores()

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

References

Drasgow, F., Levine, M. V., & Williams, E. A. (1985). Appropriateness measurement with polychotomous item response models and standardized indices. *British Journal of Mathematical and Statistical Psychology, 38*, 67-86.

Reise, S. P. (1990). A comparison of item- and person-fit methods of assessing model-data fit in IRT. *Applied Psychological Measurement, 14*, 127-137.

Wright B. D. & Masters, G. N. (1982). *Rating scale analysis*. MESA Press.

See Also

itemfit

Examples

```
## Not run:

#make some data
set.seed(1234)
a <- matrix(rlnorm(20),ncol=1)
d <- matrix(rnorm(20),ncol=1)
items <- rep('2PL', 20)
data <- simdata(a,d, 2000, items)

x <- mirt(data, 1)
fit <- personfit(x)
head(fit)

#using precomputed Theta
Theta <- fscores(x, method = 'MAP', full.scores = TRUE)
head(personfit(x, Theta=Theta))

#muliple group Rasch model example
set.seed(12345)
a <- matrix(rep(1, 15), ncol=1)
d <- matrix(rnorm(15,0,.7),ncol=1)
itemtype <- rep('dich', nrow(a))
N <- 1000
dataset1 <- simdata(a, d, N, itemtype)
dataset2 <- simdata(a, d, N, itemtype, sigma = matrix(1.5))
dat <- rbind(dataset1, dataset2)
group <- c(rep('D1', N), rep('D2', N))
models <- 'F1 = 1-15'
mod_Rasch <- multipleGroup(dat, models, itemtype = 'Rasch', group = group)
coef(mod_Rasch, simplify=TRUE)
pf <- personfit(mod_Rasch, method='MAP')
head(pf)
```

```
## End(Not run)
```

PLCI.mirt *Compute profiled-likelihood (or posterior) confidence intervals*

Description

Computes profiled-likelihood based confidence intervals. Supports the inclusion of equality constraints. Object returns the confidence intervals and whether the respective interval could be found.

Usage

```
PLCI.mirt(mod, parnum = NULL, alpha = 0.05, search_bound = TRUE,
  step = 0.5, lower = TRUE, upper = TRUE, inf2val = 30,
  NealeMiller = FALSE, ...)
```

Arguments

mod	a converged mirt model
parnum	a numeric vector indicating which parameters to estimate. Use mod2values to determine parameter numbers. If NULL, all possible parameters are used
alpha	two-tailed alpha critical level
search_bound	logical; use a fixed grid of values around the ML estimate to determine more suitable optimization bounds? Using this has much better behaviour than setting fixed upper/lower bound values and searching from more extreme ends
step	magnitude of steps used when search_bound is TRUE. Smaller values create more points to search a suitable bound for (up to the lower bound value visible with mod2values). When upper/lower bounds are detected this value will be adjusted accordingly
lower	logical; search for the lower CI?
upper	logical; search for the upper CI?
inf2val	a numeric used to change parameter bounds which are infinity to a finite number. Decreasing this too much may not allow a suitable bound to be located. Default is 30
NealeMiller	logical; use the Neale and Miller 1997 approximation? Default is FALSE
...	additional arguments to pass to the estimation functions

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

References

Chalmers, R. P., Pek, J., & Liu, Y. (in press). Profile-likelihood Confidence Intervals in Item Response Theory Models. *Multivariate Behavioral Research*.

Neale, M. C. & Miller, M. B. (1997). The use of likelihood-based confidence intervals in genetic models. *Behavior Genetics, 27*, 113-120.

See Also

boot.mirt

Examples

```
## Not run:
mirtCluster() #use all available cores to estimate CI's in parallel
dat <- expand.table(LSAT7)
mod <- mirt(dat, 1)

result <- PLCI.mirt(mod)
result

# model with constraints
mod <- mirt(dat, 'F = 1-5
                  CONSTRAIN = (1-5, a1)')

result <- PLCI.mirt(mod)
result

mod2 <- mirt(Science, 1)
result2 <- PLCI.mirt(mod2)
result2

#only estimate CI's slopes
sv <- mod2values(mod2)
parnum <- sv$parnum[sv$name == 'a1']
result3 <- PLCI.mirt(mod2, parnum)
result3


## End(Not run)
```

---

plot-method *Plot various test-implied functions from models*

Description

Plot various test implied response functions from models estimated in the mirt package.

Usage

```
## S4 method for signature 'SingleGroupClass,missing'
plot(x, y, type = "score", npts = 50,
  degrees = 45, theta_lim = c(-6, 6), which.items = 1:extract.mirt(x,
  "nitems"), MI = 0, CI = 0.95, rot = list(xaxis = -70, yaxis = 30, zaxis
  = 10), facet_items = TRUE, main = NULL, drape = TRUE, colorkey = TRUE,
  ehist.cut = 1e-10, add.ylab2 = TRUE, par.strip.text = list(cex = 0.7),
  par.settings = list(strip.background = list(col = "#9ECAE1"), strip.border =
  list(col = "black")), auto.key = list(space = "right"), profile = FALSE,
  ...)
```

Arguments

`x`	an object of class `SingleGroupClass`, `MultipleGroupClass`, or `DiscreteClass`
`y`	an arbitrary missing argument required for `R CMD check`
`type`	type of plot to view; can be `'info'` to show the test information function, `'rxx'` for the reliability function, `'infocontour'` for the test information contours, `'SE'` for the test standard error function, `'trace'`, `'infotrace'`, and `'itemscore'` for all item probability, information, and scoring or trace lines, `'infoSE'` for a combined test information and standard error plot, and `'score'` and `'scorecontour'` for the expected total score surface and contour plots. If `empiricalhist = TRUE` was used in estimation then the type `'empiricalhist'` also will be available to generate the empirical histogram plot
`npts`	number of quadrature points to be used for plotting features. Larger values make plots look smoother
`degrees`	numeric value ranging from 0 to 90 used in `plot` to compute angle for information-based plots with respect to the first dimension. If a vector is used then a bubble plot is created with the summed information across the angles specified (e.g., `degrees = seq(0, 90, by=10)`)
`theta_lim`	lower and upper limits of the latent trait (theta) to be evaluated, and is used in conjunction with `npts`
`which.items`	numeric vector indicating which items to be used when plotting. Default is to use all available items
`MI`	a single number indicating how many imputations to draw to form bootstrapped confidence intervals for the selected test statistic. If greater than 0 a plot will be drawn with a shaded region for the interval
`CI`	a number from 0 to 1 indicating the confidence interval to select when MI input is used. Default uses the 95% confidence (CI = .95)
`rot`	allows rotation of the 3D graphics
`facet_items`	logical; apply grid of plots across items? If `FALSE`, items will be placed in one plot for each group
`main`	argument passed to lattice. Default generated automatically
`drape`	logical argument passed to lattice. Default generated automatically
`colorkey`	logical argument passed to lattice. Default generated automatically

ehist.cut	a probability value indicating a threshold for excluding cases in empirical histogram plots. Values larger than the default will include more points in the tails of the plot, potentially squishing the 'meat' of the plot to take up less area than visually desired
add.ylab2	logical argument passed to lattice. Default generated automatically
par.strip.text	plotting argument passed to lattice
par.settings	plotting argument passed to lattice
auto.key	plotting argument passed to lattice
profile	logical; provide a profile plot of response probabilities (objects returned from mdirt only)
...	additional arguments to be passed to lattice

Examples

```
## Not run:
x <- mirt(Science, 1, SE=TRUE)
plot(x)
plot(x, type = 'info')
plot(x, type = 'infotrace')
plot(x, type = 'infotrace', facet_items = FALSE)
plot(x, type = 'infoSE')
plot(x, type = 'rxx')

# confidence interval plots when information matrix computed
plot(x)
plot(x, MI=100)
plot(x, type='info', MI=100)
plot(x, type='SE', MI=100)
plot(x, type='rxx', MI=100)

# use the directlabels package to put labels on tracelines
library(directlabels)
plt <- plot(x, type = 'trace')
direct.label(plt, 'top.points')

set.seed(1234)
group <- sample(c('g1','g2'), nrow(Science), TRUE)
x2 <- multipleGroup(Science, 1, group)
plot(x2)
plot(x2, type = 'trace')
plot(x2, type = 'trace', which.items = 1:2)
plot(x2, type = 'trace', which.items = 1, facet_items = FALSE) #facet by group
plot(x2, type = 'info')

x3 <- mirt(Science, 2)
plot(x3, type = 'info')
plot(x3, type = 'SE', theta_lim = c(-3,3))


## End(Not run)
```

poly2dich *Change polytomous items to dichotomous item format*

Description

Tranforms a matrix of items into a new matrix where the select polytomous items have been converted into comperable dichotomous items with the same information.

Usage

```
poly2dich(data, which.items = 1:ncol(data))
```

Arguments

`data`	an object of class `data.frame` or `matrix`
`which.items`	a vector indicating which items should be transformed into the dichotomous form. Default uses all input items

Value

Returns an integer matrix

Author(s)

Phil Chalmers `<rphilip.chalmers@gmail.com>`

Examples

```
## Not run:
data(Science)

head(Science)
newScience <- poly2dich(Science)
head(newScience)

newScience2 <- poly2dich(Science, which.items = 2)
head(newScience2)


## End(Not run)
```

print-method *Print the model objects*

Description

Print model object summaries to the console.

Usage

```
## S4 method for signature 'SingleGroupClass'
print(x)
```

Arguments

x	an object of class SingleGroupClass, MultipleGroupClass, or MixedClass

Examples

```
## Not run:
x <- mirt(Science, 1)
print(x)

## End(Not run)
```

print.mirt_df *Print generic for customized data.frame console output*

Description

Privides a nicer output for most printed `data.frame` objects defined by functions in `mirt`.

Usage

```
## S3 method for class 'mirt_df'
print(x, digits = 3, ...)
```

Arguments

x	object of class 'mirt_df'
digits	number of digits to round
...	additional arguments passed to print(...)

print.mirt_list *Print generic for customized list console output*

Description

Privides a nicer output for most printed `list` objects defined by functions in `mirt`.

Usage

```
## S3 method for class 'mirt_list'
print(x, digits = 3, ...)
```

Arguments

`x`	object of class `'mirt_list'`
`digits`	number of digits to round
`...`	additional arguments passed to `print(...)`

print.mirt_matrix *Print generic for customized matrix console output*

Description

Privides a nicer output for most printed `matrix` objects defined by functions in `mirt`.

Usage

```
## S3 method for class 'mirt_matrix'
print(x, digits = 3, ...)
```

Arguments

`x`	object of class `'mirt_matrix'`
`digits`	number of digits to round
`...`	additional arguments passed to `print(...)`

probtrace — *Function to calculate probability trace lines*

Description

Given an internal mirt object extracted from an estimated model compute the probability trace lines for all categories.

Usage

```
probtrace(x, Theta)
```

Arguments

`x`	an extracted internal mirt object containing item information (see `extract.item`)
`Theta`	a vector (unidimensional) or matrix (unidimensional/multidimensional) of latent trait values

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

See Also

`extract.item`

Examples

```
## Not run:
mod <- mirt(Science, 1)
extr.2 <- extract.item(mod, 2)
Theta <- matrix(seq(-4,4, by = .1))
traceline <- probtrace(extr.2, Theta)

head(data.frame(traceline, Theta=Theta))


## End(Not run)
```

randef — *Compute posterior estimates of random effect*

Description

Stochastically compute random effects for `MixedClass` objects with Metropolis-Hastings samplers and averaging over the draws. Returns a list of the estimated effects.

Usage

```
randef(x, ndraws = 1000, thin = 10, return.draws = FALSE)
```

Arguments

`x`	an estimated model object from the `mixedmirt` function
`ndraws`	total number of draws to perform. Default is 1000
`thin`	amount of thinning to apply. Default is to use every 10th draw
`return.draws`	logical; return a list containing the thinned draws of the posterior?

Author(s)

Phil Chalmers `<rphilip.chalmers@gmail.com>`

Examples

```
## Not run:
#make an arbitrary groups
covdat <- data.frame(group = rep(paste0('group', 1:49), each=nrow(Science)/49))

#partial credit model
mod <- mixedmirt(Science, covdat, model=1, random = ~ 1|group)
summary(mod)

effects <- randef(mod, ndraws = 2000, thin = 20)
head(effects$Theta)
head(effects$group)


## End(Not run)
```

`residuals-method` *Compute model residuals*

Description

Return model implied residuals for linear dependencies between items or at the person level.

Usage

```
## S4 method for signature 'SingleGroupClass'
residuals(object, type = "LD", df.p = FALSE,
  full.scores = FALSE, QMC = FALSE, printvalue = NULL, tables = FALSE,
  verbose = TRUE, Theta = NULL, suppress = 1, theta_lim = c(-6, 6),
  quadpts = NULL, ...)
```

Arguments

`object`	an object of class `SingleGroupClass` or `MultipleGroupClass`. Bifactor models are automatically detected and utilized for better accuracy
`type`	type of residuals to be displayed. Can be either `'LD'` or `'LDG2'` for a local dependence matrix based on the X2 or G2 statistics (Chen & Thissen, 1997), `'Q3'` for the statistic proposed by Yen (1984), or `'exp'` for the expected values for the frequencies of every response pattern. For the 'LD' and 'LDG2' types, the upper diagonal elements represent the standardized residuals in the form of signed Cramers V coefficients
`df.p`	logical; print the degrees of freedom and p-values?
`full.scores`	logical; compute relevant statistics for each subject in the original data?
`QMC`	logical; use quasi-Monte Carlo integration? If `quadpts` is omitted the default number of nodes is 5000
`printvalue`	a numeric value to be specified when using the `res='exp'` option. Only prints patterns that have standardized residuals greater than `abs(printvalue)`. The default (NULL) prints all response patterns
`tables`	logical; for LD type, return the observed, expected, and standardized residual tables for each item combination?
`verbose`	logical; allow information to be printed to the console?
`Theta`	a matrix of factor scores used for statistics that require empirical estimates (i.e., Q3). If supplied, arguments typically passed to `fscores()` will be ignored and these values will be used instead
`suppress`	a numeric value indicating which parameter local dependency combinations to flag as being too high. Absolute values for the standardized estimates greater than this value will be returned, while all values less than this value will be set to NA
`theta_lim`	range for the integration grid

`quadpts`	number of quadrature nodes to use. The default is extracted from model (if available) or generated automatically if not available
`...`	additional arguments to be passed to `fscores()`

References

Chen, W. H. & Thissen, D. (1997). Local dependence indices for item pairs using item response theory. *Journal of Educational and Behavioral Statistics, 22*, 265-289.

Yen, W. (1984). Effects of local item dependence on the fit and equating performance of the three parameter logistic model. *Applied Psychological Measurement, 8*, 125-145.

Examples

```
## Not run:

x <- mirt(Science, 1)
residuals(x)
residuals(x, tables = TRUE)
residuals(x, type = 'exp')
residuals(x, suppress = .15)

# with and without supplied factor scores
Theta <- fscores(x)
residuals(x, type = 'Q3', Theta=Theta)
residuals(x, type = 'Q3', method = 'ML')


## End(Not run)
```

SAT12 — *Description of SAT12 data*

Description

Data obtained from the TESTFACT (Woods et al., 2003) manual, with 32 response pattern scored items for a grade 12 science assessment test (SAT) measuring topics of chemistry, biology, and physics. The scoring key for these data is [1, 4, 5, 2, 3, 1, 2, 1, 3, 1, 2, 4, 2, 1, 5, 3, 4, 4, 1, 4, 3, 3, 4, 1, 3, 5, 1, 3, 1, 5, 4, 5], respectively. However, careful analysis using the nominal response model suggests that the scoring key for item 32 may be incorrect, and should be changed from 5 to 3.

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

References

Wood, R., Wilson, D. T., Gibbons, R. D., Schilling, S. G., Muraki, E., & Bock, R. D. (2003). TESTFACT 4 for Windows: Test Scoring, Item Statistics, and Full-information Item Factor Analysis [Computer software]. Lincolnwood, IL: Scientific Software International.

Examples

```
## Not run:
#score the data (missing scored as 0)
head(SAT12)
data <- key2binary(SAT12,
    key = c(1,4,5,2,3,1,2,1,3,1,2,4,2,1,5,3,4,4,1,4,3,3,4,1,3,5,1,3,1,5,4,5))
head(data)

#score the data, missing (value of 8) treated as NA
SAT12missing <- SAT12
SAT12missing[SAT12missing == 8] <- NA
data <- key2binary(SAT12missing,
    key = c(1,4,5,2,3,1,2,1,3,1,2,4,2,1,5,3,4,4,1,4,3,3,4,1,3,5,1,3,1,5,4,5))
head(data)

#potentially better scoring for item 32 (based on nominal model finding)
data <- key2binary(SAT12,
    key = c(1,4,5,2,3,1,2,1,3,1,2,4,2,1,5,3,4,4,1,4,3,3,4,1,3,5,1,3,1,5,4,3))

## End(Not run)
```

`Science` *Description of Science data*

Description

A 4-item data set borrowed from `ltm` package in R, first example of the `grm()` function. See more complete documentation therein.

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

Examples

```
## Not run:
mod <- mirt(Science, 1)
plot(mod, type = 'trace')

## End(Not run)
```

show-method *Show model object*

Description

Print model object summaries to the console.

Usage

```
## S4 method for signature 'SingleGroupClass'
show(object)
```

Arguments

object	an object of class SingleGroupClass, MultipleGroupClass, or MixedClass

Examples

```
## Not run:
x <- mirt(Science, 1)
show(x)

## End(Not run)
```

SIBTEST *Simultaneous Item Bias Test (SIBTEST)*

Description

Classical test theory approach to detecting DIF for unidimensional tests by applying a regression-corrected matched-total score approach. SIBTEST is similar to the Mantel-Haenszel approach for detecting DIF but uses a regression correction based on the KR-20/coefficient alpha reliability index to correct the observed differences when the latent trait distributions are not equal. Function supports the standard SIBTEST for dichotomous and poltomous data (compensatory) and also supports crossed DIF testing (i.e., non-compensatory).

Usage

```
SIBTEST(dat, group, focal_set, match_set, focal_name, guess_correction = 0,
  Jmin = 2, cross = FALSE, permute = 1000, pk_focal = FALSE,
  correction = TRUE, details = FALSE)
```

Arguments

- `dat` — integer dataset to be tested containing dichotomous or polytomous responses
- `group` — a vector indicating group membership
- `focal_set` — an integer vector indicating which items to inspect with SIBTEST. Including only one value will perform a DIF test, while including more than one will perform a simultaneous bundle test (DBF); including all non-matched items will perform DTF. If missing, a simultaneous test using all the items not listed in match_set will be used (i.e., DTF)
- `match_set` — an integer vector indicating which items to use as the items which are matched (i.e., contain no DIF). These are analogous to 'achor' items in the likelihood method to locate DIF. If missing, all items other than the items found in the focal_set will be used
- `focal_name` — name of the focal group; e.g., 'focal'. If not specified then one will be selected automatically
- `guess_correction` — a vector of numbers from 0 to 1 indicating how much to correct the items for guessing. It's length should be the same as ncol(dat)
- `Jmin` — the minimum number of observations required when splitting the data into focal and reference groups conditioned on the matched set
- `cross` — logical; perform the crossing test for non-compensatory bias? Default is `FALSE`
- `permute` — number of permutations to perform when `cross = TRUE`. Default is 1000
- `pk_focal` — logical; using the group weights from the focal group instead of the total sample? Default is FALSE as per Shealy and Stout's recommendation
- `correction` — logical; apply the composite correction for the difference between focal composite scores using the true-score regression technique? Default is `TRUE`, reflecting Shealy and Stout's method
- `details` — logical; return a data.frame containing the details required to compute SIBTEST?

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

References

Chang, H. H., Mazzeo, J. & Roussos, L. (1996). DIF for Polytomously Scored Items: An Adaptation of the SIBTEST Procedure. Journal of Educational Measurement, 33, 333-353.

Li, H.-H. & Stout, W. (1996). A new procedure for detetion of crossing DIF. Psychometrika, 61, 647-677.

Shealy, R. & Stout, W. (1993). A model-based standardization approach that separates true bias/DIF from group ability differences and ddetect test bias/DTF as well as item bias/DIF. Psychometrika, 58, 159-194.

Examples

```
## Not run:

library(mirt)

set.seed(1234)
n <- 30
N <- 500
a <- matrix(1, n)
d <- matrix(rnorm(n), n)
group <- c(rep('reference', N), rep('focal', N*2))

## -------------
# groups completely equal
dat1 <- simdata(a, d, N, itemtype = '2PL')
dat2 <- simdata(a, d, N*2, itemtype = '2PL')
dat <- rbind(dat1, dat2)

#DIF (all other items as anchors)
SIBTEST(dat, group, focal_set = 6)

#DIF (specific anchors)
SIBTEST(dat, group, match_set = 1:5, focal_set = 6)

# DBF (all and specific anchors, respectively)
SIBTEST(dat, group, focal_set = 11:30)
SIBTEST(dat, group, match_set = 1:5, focal_set = 11:30)

#DTF
SIBTEST(dat, group, focal_set = 11:30)
SIBTEST(dat, group, match_set = 1:10) #equivalent

# different hyper pars
dat1 <- simdata(a, d, N, itemtype = '2PL')
dat2 <- simdata(a, d, N*2, itemtype = '2PL', mu = .5, sigma = matrix(1.5))
dat <- rbind(dat1, dat2)
SIBTEST(dat, group, 6:30)
SIBTEST(dat, group, 11:30)

#DIF testing with anchors 1 through 5
SIBTEST(dat, group, 6, match_set = 1:5)
SIBTEST(dat, group, 7, match_set = 1:5)
SIBTEST(dat, group, 8, match_set = 1:5)

#DIF testing with all other items as anchors
SIBTEST(dat, group, 6)
SIBTEST(dat, group, 7)
SIBTEST(dat, group, 8)

#crossed SIBTEST
SIBTEST(dat, group, 6, match_set = 1:5, cross=TRUE)
SIBTEST(dat, group, 7, match_set = 1:5, cross=TRUE)
```

```
SIBTEST(dat, group, 8, match_set = 1:5, cross=TRUE)

## -------------
## systematic differing slopes and intercepts (clear DTF)
dat1 <- simdata(a, d, N, itemtype = '2PL')
dat2 <- simdata(a + c(numeric(15), rnorm(n-15, 1, .25)), d + c(numeric(15), rnorm(n-15, 1, 1)),
  N*2, itemtype = '2PL')
dat <- rbind(dat1, dat2)
SIBTEST(dat, group, 6:30)
SIBTEST(dat, group, 11:30)

#DIF testing using valid anchors
SIBTEST(dat, group, focal_set = 6, match_set = 1:5)
SIBTEST(dat, group, focal_set = 7, match_set = 1:5)
SIBTEST(dat, group, focal_set = 30, match_set = 1:5)

SIBTEST(dat, group, focal_set = 11, match_set = 1:10, cross=TRUE)
SIBTEST(dat, group, focal_set = 30, match_set = 1:15, cross=TRUE)


## End(Not run)
```

---

simdata *Simulate response patterns*

Description

Simulates response patterns for compensatory and noncompensatory MIRT models from multivariate normally distributed factor (θ) scores, or from a user input matrix of θ's.

Usage

```
simdata(a, d, N, itemtype, sigma = NULL, mu = NULL, guess = 0,
  upper = 1, nominal = NULL, Theta = NULL, gpcm_mats = list(),
  returnList = FALSE, model = NULL, which.items = NULL, mins = 0,
  lca_cats = NULL, prob.list = NULL)
```

Arguments

a	a matrix/vector of slope parameters. If slopes are to be constrained to zero then use NA or simply set them equal to 0
d	a matrix/vector of intercepts. The matrix should have as many columns as the item with the largest number of categories, and filled empty locations with NA. When a vector is used the test is assumed to consist only of dichotomous items (because only one intercept per item is provided). When itemtype = 'lca' intercepts will not be used
N	sample size

itemtype	a character vector of length nrow(a) (or 1, if all the item types are the same) specifying the type of items to simulate. Inputs can either be the same as the inputs found in the itemtype argument in mirt or the internal clases defined by the package. Typical itemtype inputs that are passed to mirt are used then these will be converted into the respective internal classes automatically. If the internal class of the object is specified instead, the inputs can be 'dich', 'graded', 'gpcm','nomi or 'lca', for dichotomous, graded, generalized partial credit, nominal, nested logit, partially compensatory, and latent class analysis model. Note that for the gpcm, nominal, and nested logit models there should be as many parameters as desired categories, however to parametrized them for meaningful interpretation the first category intercept should equal 0 for these models (second column for 'nestlogit', since first column is for the correct item traceline). For nested logit models the 'correct' category is always the lowest category (i.e., == 1). It may be helpful to use mod2values on data-sets that have already been estimated to understand the itemtypes more intimately
sigma	a covariance matrix of the underlying distribution. Default is the identity matrix. Used when Theta is not supplied
mu	a mean vector of the underlying distribution. Default is a vector of zeros. Used when Theta is not supplied
guess	a vector of guessing parameters for each item; only applicable for dichotomous items. Must be either a scalar value that will affect all of the dichotomous items, or a vector with as many values as to be simulated items
upper	same as guess, but for upper bound parameters
nominal	a matrix of specific item category slopes for nominal models. Should be the dimensions as the intercept specification with one less column, with NA in locations where not applicable. Note that during estimation the first slope will be constrained to 0 and the last will be constrained to the number of categories minus 1, so it is best to set these as the values for the first and last categories as well
Theta	a user specified matrix of the underlying ability parameters, where nrow(Theta) == N and ncol(Theta) == ncol(a). When this is supplied the N input is not required
gpcm_mats	a list of matricies specifying the scoring scheme for generalized partial credit models (see mirt for details)
returnList	logical; return a list containing the data, item objects defined by mirt containing the population parameters and item structure, and the latent trait matrix Theta? Default is FALSE
model	a single group object, typically returned by functions such as mirt or bfactor. Supplying this will render all other parameter elements (excluding the Theta, N, mu, and sigma inputs) redundent (unless explicitly provided)
which.items	an integer vector used to indicate which items to simulate when a model input is included. Default simulates all items
mins	an integer vector (or single value to be used for each item) indicating what the lowest category should be. If model is supplied then this will be extracted from slot(mod, 'Data')$mins, otherwise the default is 0

`lca_cats`	a vector indicating how many categories each lca item should have. If not supplied then it is assumed that 2 categories should be generated for each item
`prob.list`	an optional list containing matrix/data.frames of probabilities values for each category to be simulated. This is useful when creating customized probability functions to be sampled from

Details

Returns a data matrix simulated from the parameters, or a list containing the data, item objects, and Theta matrix.

Author(s)

Phil Chalmers `<rphilip.chalmers@gmail.com>`

References

Reckase, M. D. (2009). *Multidimensional Item Response Theory*. New York: Springer.

Examples

```
## Not run:
### Parameters from Reckase (2009), p. 153

set.seed(1234)

a <- matrix(c(
 .7471, .0250, .1428,
 .4595, .0097, .0692,
 .8613, .0067, .4040,
1.0141, .0080, .0470,
 .5521, .0204, .1482,
1.3547, .0064, .5362,
1.3761, .0861, .4676,
 .8525, .0383, .2574,
1.0113, .0055, .2024,
 .9212, .0119, .3044,
 .0026, .0119, .8036,
 .0008, .1905,1.1945,
 .0575, .0853, .7077,
 .0182, .3307,2.1414,
 .0256, .0478, .8551,
 .0246, .1496, .9348,
 .0262, .2872,1.3561,
 .0038, .2229, .8993,
 .0039, .4720, .7318,
 .0068, .0949, .6416,
 .3073, .9704, .0031,
 .1819, .4980, .0020,
 .4115,1.1136, .2008,
 .1536,1.7251, .0345,
```

```
 .1530, .6688, .0020,
 .2890,1.2419, .0220,
 .1341,1.4882, .0050,
 .0524, .4754, .0012,
 .2139, .4612, .0063,
 .1761,1.1200, .0870),30,3,byrow=TRUE)*1.702

d <- matrix(c(.1826,-.1924,-.4656,-.4336,-.4428,-.5845,-1.0403,
  .6431,.0122,.0912,.8082,-.1867,.4533,-1.8398,.4139,
  -.3004,-.1824,.5125,1.1342,.0230,.6172,-.1955,-.3668,
  -1.7590,-.2434,.4925,-.3410,.2896,.006,.0329),ncol=1)*1.702

mu <- c(-.4, -.7, .1)
sigma <- matrix(c(1.21,.297,1.232,.297,.81,.252,1.232,.252,1.96),3,3)

dataset1 <- simdata(a, d, 2000, itemtype = '2PL')
dataset2 <- simdata(a, d, 2000, itemtype = '2PL', mu = mu, sigma = sigma)

#mod <- mirt(dataset1, 3, method = 'MHRM')
#coef(mod)

### Unidimensional graded response model with 5 categories each

a <- matrix(rlnorm(20,.2,.3))

# for the graded model, ensure that there is enough space between the intercepts,
# otherwise closer categories will not be selected often (minimum distance of 0.3 here)
diffs <- t(apply(matrix(runif(20*4, .3, 1), 20), 1, cumsum))
diffs <- -(diffs - rowMeans(diffs))
d <- diffs + rnorm(20)

dat <- simdata(a, d, 500, itemtype = 'graded')
# mod <- mirt(dat, 1)

### An example of a mixed item, bifactor loadings pattern with correlated specific factors

a <- matrix(c(
.8,.4,NA,
.4,.4,NA,
.7,.4,NA,
.8,NA,.4,
.4,NA,.4,
.7,NA,.4),ncol=3,byrow=TRUE)

d <- matrix(c(
-1.0,NA,NA,
 1.5,NA,NA,
 0.0,NA,NA,
0.0,-1.0,1.5,  #the first 0 here is the recommended constraint for nominal
0.0,1.0,-1, #the first 0 here is the recommended constraint for gpcm
2.0,0.0,NA),ncol=3,byrow=TRUE)

nominal <- matrix(NA, nrow(d), ncol(d))
```

```
#the first 0 and last (ncat - 1) = 2 values are the recommended constraints
nominal[4, ] <- c(0,1.2,2)

sigma <- diag(3)
sigma[2,3] <- sigma[3,2] <- .25
items <- c('2PL','2PL','2PL','nominal','gpcm','graded')

dataset <- simdata(a,d,2000,items,sigma=sigma,nominal=nominal)

#mod <- bfactor(dataset, c(1,1,1,2,2,2), itemtype=c(rep('2PL', 3), 'nominal', 'gpcm','graded'))
#coef(mod)

#### Convert standardized factor loadings to slopes

F2a <- function(F, D=1.702){
    h2 <- rowSums(F^2)
    a <- (F / sqrt(1 - h2)) * D
    a
}

(F <- matrix(c(rep(.7, 5), rep(.5,5))))
(a <- F2a(F))
d <- rnorm(10)

dat <- simdata(a, d, 5000, itemtype = '2PL')
mod <- mirt(dat, 1)
coef(mod, simplify=TRUE)$items
summary(mod)

mod2 <- mirt(dat, 'F1 = 1-10
                   CONSTRAIN = (1-5, a1), (6-10, a1)')
summary(mod2)
anova(mod, mod2)

#### Unidimensional nonlinear factor pattern

theta <- rnorm(2000)
Theta <- cbind(theta,theta^2)

a <- matrix(c(
.8,.4,
.4,.4,
.7,.4,
.8,NA,
.4,NA,
.7,NA),ncol=2,byrow=TRUE)
d <- matrix(rnorm(6))
itemtype <- rep('2PL',6)

nonlindata <- simdata(a=a, d=d, itemtype=itemtype, Theta=Theta)

#model <- '
#F1 = 1-6
```

```
#(F1 * F1) = 1-3'
#mod <- mirt(nonlindata, model)
#coef(mod)

#### 2PLNRM model for item 4 (with 4 categories), 2PL otherwise

a <- matrix(rlnorm(4,0,.2))

#first column of item 4 is the intercept for the correct category of 2PL model,
#    otherwise nominal model configuration
d <- matrix(c(
-1.0,NA,NA,NA,
 1.5,NA,NA,NA,
 0.0,NA,NA,NA,
 1, 0.0,-0.5,0.5),ncol=4,byrow=TRUE)

nominal <- matrix(NA, nrow(d), ncol(d))
nominal[4, ] <- c(NA,0,.5,.6)

items <- c(rep('2PL',3),'nestlogit')

dataset <- simdata(a,d,2000,items,nominal=nominal)

#mod <- mirt(dataset, 1, itemtype = c('2PL', '2PL', '2PL', '2PLNRM'), key=c(NA,NA,NA,1))
#coef(mod)
#itemplot(mod,4)

#return list of simulation parameters
listobj <- simdata(a,d,2000,items,nominal=nominal, returnList=TRUE)
str(listobj)

# generate dataset from converged model
mod <- mirt(Science, 1, itemtype = c(rep('gpcm', 3), 'nominal'))
sim <- simdata(model=mod, N=1000)
head(sim)

Theta <- matrix(rnorm(100))
sim <- simdata(model=mod, Theta=Theta)
head(sim)

# alternatively, define a suitable object with functions from the mirtCAT package
# help(generate.mirt_object)
library(mirtCAT)

nitems <- 50
a1 <- rlnorm(nitems, .2,.2)
d <- rnorm(nitems)
g <- rbeta(nitems, 20, 80)
pars <- data.frame(a1=a1, d=d, g=g)
head(pars)

obj <- generate.mirt_object(pars, '3PL')
dat <- simdata(N=200, model=obj)
```

```
######
# prob.list example

# custom probabilty function that returns a matrix
fun <- function(a, b, theta){
    P <- 1 / (1 + exp(-a * (theta-b)))
    cbind(1-P, P)
}

set.seed(1)
theta <- matrix(rnorm(100))
prob.list <- list()
nitems <- 5
a <- rlnorm(nitems, .2, .2); b <- rnorm(nitems, 0, 1/2)
for(i in 1:nitems) prob.list[[i]] <- fun(a[i], b[i], theta)
str(prob.list)

dat <- simdata(prob.list=prob.list)
head(dat)


## End(Not run)
```

SingleGroupClass-class

Class "SingleGroupClass"

Description

Defines the object returned from `mirt` when model is exploratory.

Slots

`Call`: function call

`Data`: list of data, sometimes in different forms

`Options`: list of estimation options

`Fit`: a list of fit information

`Model`: a list of model-based information

`ParObjects`: a list of the S4 objects used during estimation

`OptimInfo`: a list of arguments from the optimization process

`Internals`: a list of internal arguments for secondary computations (inspecting this object is generally not required)

`vcov`: a matrix represented the asymtotic covariance matrix of the parameter estimates

`time`: a data.frame indicating the breakdown of computation times in seconds

Methods

anova `signature(object = "SingleGroupClass")`

coef `signature(object = "SingleGroupClass")`

plot `signature(x = "SingleGroupClass", y = "missing")`

print `signature(x = "SingleGroupClass")`

residuals `signature(object = "SingleGroupClass")`

show `signature(object = "SingleGroupClass")`

summary `signature(object = "SingleGroupClass")`

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

---

summary-method *Summary of model object*

Description

Transforms coefficients into a standardized factor loading's metric. For `MixedClass` objects, the fixed and random coefficients are printed. Note that while the output to the console is rounded to three digits, the returned list of objects is not. For simulations, use `output <- summary(mod, verbose = FALSE)` to suppress the console messages.

Usage

```
## S4 method for signature 'SingleGroupClass'
summary(object, rotate = "oblimin",
  Target = NULL, suppress = 0, verbose = TRUE, ...)
```

Arguments

`object`	an object of class `SingleGroupClass`, `MultipleGroupClass`, or `MixedClass`
`rotate`	a string indicating which rotation to use for exploratory models, primarily from the `GPArotation` package (see documentation therein). Rotations currently supported are: `'promax'`, `'oblimin'`, `'varimax'`, `'quartimin'`, `'targetT'`, `'targetQ'`, `'pstT'`, `'pstQ'`, `'oblimax'`, `'entropy'`, `'quartimax'`, `'simplimax'`, `'bentlerT'`, `'bentlerQ'`, `'tandemI'`, `'tandemII'`, `'geominT'`, `'geominQ'`, `'cfT'`, `'cfQ'`, `'infomaxT'`, `'infomaxQ'`, `'mccammon'`, `'bifactorT'`, `'bifactorQ'`. For models that are not exploratory this input will automatically be set to `'none'`
`Target`	a dummy variable matrix indicting a target rotation pattern. This is required for rotations such as `'targetT'`, `'targetQ'`, `'pstT'`, and `'pstQ'`

suppress	a numeric value indicating which (possibly rotated) factor loadings should be suppressed. Typical values are around .3 in most statistical software. Default is 0 for no suppression
verbose	logical; allow information to be printed to the console?
...	additional arguments to be passed

See Also

coef-method

Examples

```
## Not run:
x <- mirt(Science, 2)
summary(x)
summary(x, rotate = 'varimax')


## End(Not run)
```

---

testinfo *Function to calculate test information*

Description

Given an estimated model compute the test information.

Usage

```
testinfo(x, Theta, degrees = NULL, group = NULL, individual = FALSE,
  which.items = 1:extract.mirt(x, "nitems"))
```

Arguments

x	an estimated mirt object
Theta	a matrix of latent trait values
degrees	a vector of angles in degrees that are between 0 and 90. Only applicable when the input object is multidimensional
group	a number signifying which group the item should be extracted from (applies to 'MultipleGroupClass' objects only)
individual	logical; return a data.frame of information traceline for each item?
which.items	an integer vector indicating which items to include in the expected information function. Default uses all possible items

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

Examples

```
## Not run:
dat <- expand.table(deAyala)
(mirt(dat, 1, '2PL', pars = 'values'))
mod <- mirt(dat, 1, '2PL', constrain = list(c(1,5,9,13,17)))

Theta <- matrix(seq(-4,4,.01))
tinfo <- testinfo(mod, Theta)
plot(Theta, tinfo, type = 'l')

#compare information loss between two tests
tinfo_smaller <- testinfo(mod, Theta, which.items = 3:5)

#removed item informations
plot(Theta, iteminfo(extract.item(mod, 1), Theta), type = 'l')
plot(Theta, iteminfo(extract.item(mod, 2), Theta), type = 'l')

#most loss of info around -1 when removing items 1 and 2; expected given item info functions
plot(Theta, tinfo_smaller - tinfo, type = 'l')



## End(Not run)
```

`vcov-method` *Extract parameter variance covariance matrix*

Description

Extract parameter variance covariance matrix

Usage

```
## S4 method for signature 'SingleGroupClass'
vcov(object)
```

Arguments

`object`	an object of class `SingleGroupClass`, `MultipleGroupClass`, or `MixedClass`

Examples

```
## Not run:
x <- mirt(Science, 1, SE=TRUE)
vcov(x)


## End(Not run)
```

---

wald — *Wald statistics for mirt models*

Description

Compute a Wald test given an `L` vector or matrix of numeric contrasts. Requires that the model information matrix be computed (including `SE = TRUE` when using the EM method). Use `wald(model)` to observe how the information matrix columns are named, especially if the estimated model contains constrained parameters (e.g., 1PL).

Usage

```
wald(object, L, C = 0)
```

Arguments

`object`	estimated object from `mirt`, `bfactor`, `multipleGroup`, `mixedmirt`, or `mdirt`
`L`	a coefficient matrix with dimensions nconstrasts x npars. Omitting this value will return the column names of the information matrix used to identify the (potentially constrained) parameters
`C`	a constant vector of population parameters to be compared along side L, where `length(C) == ncol(L)`. By default a vector of 0's is constructed

Author(s)

Phil Chalmers <rphilip.chalmers@gmail.com>

Examples

```
## Not run:
#View parnumber index
data(LSAT7)
data <- expand.table(LSAT7)
mod <- mirt(data, 1, SE = TRUE)
coef(mod)

# see how the information matrix relates to estimated parameters, and how it lines up
#   with the parameter index
(infonames <- wald(mod))
```

```
index <- mod2values(mod)
index[index$est, ]

#second item slope equal to 0?
L <- matrix(0, 1, 10)
L[1,3] <- 1
wald(mod, L)

#simultaneously test equal factor slopes for item 1 and 2, and 4 and 5
L <- matrix(0, 2, 10)
L[1,1] <- L[2, 7] <- 1
L[1,3] <- L[2, 9] <- -1
L
wald(mod, L)

#logLiklihood tests (requires estimating a new model)
cmodel <- 'theta = 1-5
           CONSTRAIN = (1,2, a1), (4,5, a1)'
mod2 <- mirt(data, cmodel)
#or, eqivalently
#mod2 <- mirt(data, 1, constrain = list(c(1,5), c(13,17)))
anova(mod2, mod)


## End(Not run)
```

Index