Package 'eRm'

March 15, 2024

Type Package

Title Extended Rasch Modeling

Version 1.0-6

Date 2024-03-14

Description Fits Rasch models (RM), linear logistic test models (LLTM), rating scale model (RSM), linear rating scale models (LRSM), partial credit models (PCM), and linear partial credit models (LPCM). Missing values are allowed in the data matrix. Additional features are the ML estimation of the person parameters, Andersen's LR-test, item-specific Wald test, Martin-Loef-Test, nonparametric Monte-Carlo Tests, itemfit and personfit statistics including infit and outfit measures, ICC and other plots, automated stepwise item elimination, simulation module for various binary data matrices.

License GPL-3

Imports graphics, grDevices, stats, methods, MASS, splines, Matrix, lattice, colorspace, psych

Depends R (>= 3.5.0)

Encoding UTF-8

LazyData yes

LazyLoad yes

ByteCompile yes

NeedsCompilation yes

Author Patrick Mair [cre, aut],
Thomas Rusch [aut],
Reinhold Hatzinger [aut],
Marco J. Maier [aut],
Rudolf Debelak [ctb]

Maintainer Patrick Mair <mair@fas.harvard.edu>

Repository CRAN

Date/Publication 2024-03-15 08:36:29 UTC

R topics documented:

`Analysis of Deviances` *Analysis of Deviances for Rasch Models*

Description

Performs likelihood ratio tests against the model with the largest number of parameters.

Usage

```
## S3 method for class 'eRm'
anova(object, ...)

## S3 method for class 'eRm_anova'
print(x, ...)
```

Arguments

`object`	Gives the first object to be tested against others which follow, separated by commata.
`x`	An object of class `"eRm_anova"`.
`...`	Further models to test with `anova.eRm()`.

Details

The `anova` method is quite flexible and, as long the used data are identical, every model except the LLRA can be tested against each other. Regardless of the order that models are specified, they will always be sorted by the number of parameters in decreasing order. If ≥ 3 models are passed to the method, all models will be tested against the first model (i.e., the one with the largest amount of parameters).

Value

`anova.eRm` returns a list object of class `eRm_anova` containing:

`calls`	function calls of the different models (character).
`statistics`	the analysis of deviances table (columns are `LLs`: conditional log-likelihoods, `dev`: deviances, `npar`: number of parameters, `LR`: likelihood ratio statistics, `df`: degrees of freedom, `p`: p-values).

Warning

Although, there is a check for identical data matrices used, the models have to be nested for the likelihood ratio test to work. You have to ensure that this is the case, otherwise results will be invalid.

LLRAs cannot be tested with other models (RM, LLTM, RSM, ...); for more information see `anova.llra`.

Author(s)

Marco J. Maier

See Also

`anova.llra`, `anova`

Examples

```
### dichotomous data
dmod1 <- RM(lltmdat1)
dmod2 <- LLTM(lltmdat1, mpoints = 2)
anova(dmod1, dmod2)

### polytomous data
pmod1 <- RSM(rsmdat)
pmod2 <- PCM(rsmdat)
anova(pmod1, pmod2)

W <- cbind(rep(c(1,0), each=9), rep(c(0,1), each=9))
W
pmod3 <- LPCM(rsmdat, W)
anova(pmod3, pmod1, pmod2) # note that models are sorted by npar
```

`anova.llra` *Analysis of Deviance for Linear Logistic Models with Relaxed Assumptions*

Description

Compute an analysis of deviance table for one or more LLRA.

Usage

```
## S3 method for class 'llra'
anova(object, ...)
```

Arguments

`object, ...`	Objects of class "llra", typically the result of a call to LLRA.

Details

An analysis of deviance table will be calculated. The models in rows are ordered from the smallest to the largest model. Each row shows the number of parameters (Npar) and the log-likelihood (logLik). For all but the first model, the parameter difference (df) and the difference in deviance or the likelihood ratio (-2LR) is given between two subsequent models (with increasing complexity). Please note that interpreting these values only makes sense if the models are nested.

The table also contains p-values comparing the reduction in the deviance to the df for each row based on the asymptotic Chi^2-Distribution of the Likelihood ratio test statistic.

Value

An object of class `"anova"` inheriting from class `"data.frame"`.

Warning:

The comparison between two or more models by anova will only be valid if they are fitted to the same dataset and if the models are nested. The function does not check if that is the case.

Author(s)

Thomas Rusch

See Also

The model fitting function LLRA.

Examples

```
## Not run:
##An LLRA with 2 treatment groups and 1 baseline group, 5 items and 4
##time points. Item 1 is dichotomous, all others have 3, 4, 5, 6
##categories respectively.

#fit LLRA
ex2 <- LLRA(llraDat2[,1:20],mpoints=4,groups=llraDat2[,21])

#Imposing a linear trend for items 2 and 3 using collapse_W
collItems2 <- list(c(32,37,42),c(33,38,43))
newNames2 <- c("trend.I2","trend.I3")
Wnew <- collapse_W(ex2$W,collItems2,newNames2)
```

```
#Estimating LLRA with the linear trend for item 2 and 3
ex2new <- LLRA(llraDat2[1:20],W=Wnew,mpoints=4,groups=llraDat2[21])

#comparing models with likelihood ratio test
anova(ex2,ex2new)
## End(Not run)
```

---

build_W *Automatized Construction of LLRA Design Matrix*

Description

Builds a design matrix for LLRA from scratch.

Usage

```
build_W(X, nitems, mpoints, grp_n, groupvec, itmgrps)
```

Arguments

X	Data matrix as described in Hatzinger and Rusch (2009). It must be of long format, e.g. for each person all item answers are written in subsequent rows. The columns correspond to time points. Missing values are not allowed. It can easily be constructed from data in wide format with `matrix(unlist(data),ncol=mpoints)` or from `llra.datprep`.
nitems	The number of items.
mpoints	The number of time points.
grp_n	A vector of number of subjects per g+1 groups (e.g. g treatment or covariate groups and 1 control or baseline group. The sizes must be ordered like the corresponding groups.
groupvec	Assignment vector, i.e. which person belongs to which treatment/item group
itmgrps	Specifies how many groups of items there are.

Details

The function is designed to be modular and calls four internal function `build_effdes` (for treatment/covariate effects), `build_trdes` (for trend effects), `build_catdes` (for category parameter design matrix) and `get_item_cats` (checks how many categories each item has). Those functions are not intended to be used by the user.

Labeling of effects also happens in the internal functions.

Value

An LLRA design matrix as described by Hatzinger and Rusch (2009). This can be passed as the W argument to LLRA or LPCM.

The design matrix specifies every item to lie on its own dimension. Hence at every time point > 1, there are effects for each treatment or covariate group as well as trend effects for every item. Therefore overall there are items x (groups-1) x (time points-1) covariate effect parameters and items x (time points-1) trend parameters specified. For polytomous items there also are parameters for each category with the first and second category being equated for each item. They need not be equidistant. The number of parameters therefore increase quite rapidly for any additional time point, item or covariate group.

Warning

A warning is printed that the first two categories for polytomous items are equated.

Author(s)

Thomas Rusch

References

Hatzinger, R. and Rusch, T. (2009) IRT models with relaxed assumptions in eRm: A manual-like instruction. *Psychology Science Quarterly*, **51**, pp. 87–120.

See Also

This function is used for automatic generation of the design matrix in LLRA.

Examples

```
##An LLRA with 2 treatment groups and 1 baseline group, 5 items and 4
##time points. Item 1 is dichotomous, all others have 3, 4, 5, 6
##categories respectively.
llraDat2a <- matrix(unlist(llraDat2[1:20]),ncol=4)
groupvec <-rep(1:3*5,each=20)
W <- build_W(llraDat2a,nitems=5,mpoints=4,grp_n=c(10,20,40),groupvec=groupvec,itmgrps=1:5)

#There are 55 parameters
dim(W)

## Not run:
#Estimating LLRA by specifiying W
ex2W <- LLRA(llraDat2[1:20],W=W,mpoints=4,groups=llraDat2[21])
## End(Not run)
```

collapse_W *Convenient Collapsing of LLRA Design Matrix*

Description

Collapses columns of a design matrix for LLRA to specify different parameter restrictions in LLRA.

Usage

```
collapse_W(W, listItems, newNames)
```

Arguments

W	A design matrix (for LLRA), typically from a call to build_W or component $W from LLRA or LPCM
listItems	A list of numeric vectors. Each component of the list specifies columns to be collapsed together.
newNames	An (optional) character vector specifying the names of the collapsed effects.

Details

This function is a convenience function to collapse a design matrix, i.e. to specify linear trend or treatment effects and so on. Collapsing here means that effects in columns are summed up. For this, a list of numeric vectors with the column indices of columns to be collapsed have to be passed to the function. For example, if you want to collapse column 3, 6 and 8 into one new effect and 1, 4 and 9 into another it needs to be passed with `list(c(3,6,8),c(1,4,9))`.

The new effects can be given names by passing a character vector to the function with equal length as the list.

Value

An LLRA design matrix as described by Hatzinger and Rusch (2009). This can be passed as the W argument to LLRA or LPCM.

Author(s)

Thomas Rusch

References

Hatzinger, R. and Rusch, T. (2009) IRT models with relaxed assumptions in eRm: A manual-like instruction. *Psychology Science Quarterly*, **51**, pp. 87–120.

See Also

The function to build design matrices from scratch, build_W.

Examples

```
##An LLRA with 2 treatment groups and 1 baseline group, 5 items and 4
##time points. Item 1 is dichotomous, all others have 3, 4, 5, 6
##categories respectively.
llraDat2a <- matrix(unlist(llraDat2[1:20]),ncol=4)
groupvec <-rep(1:3*5,each=20)
W <- build_W(llraDat2a, nitems=5, mpoints=4, grp_n=c(10,20,40), groupvec=groupvec,itmgrps=1:5)

#There are 55 parameters to be estimated
dim(W)

#Imposing a linear trend for the second item ,i.e. parameters in
#columns 32, 37  and 42 need to be collapsed into a single column.
collItems1 <- list(c(32,37,42))
newNames1 <- c("trend.I2")
Wstar1 <- collapse_W(W,collItems1)

#53 parameters need to be estimated
dim(Wstar1)
```

---

eRm.data *Data for Computing Extended Rasch Models*

Description

Artificial data sets for computing extended Rasch models.

Usage

```
raschdat1
raschdat2
raschdat3
raschdat4

lltmdat1
lltmdat2

rsmdat

lrsmdat

pcmdat
pcmdat2

lpcmdat

raschdat1_RM_fitted
raschdat1_RM_plotDIF
raschdat1_RM_lrres2
```

Format

Numeric matrices with subjects as rows, items as columns, missing values as NA.

Details

raschdat1_RM_fitted is the resulting object of RM(raschdat1) and used in examples to reduce computation time. For the generation of raschdat1_RM_plotDIF see the excluded example code of plotDIF. raschdat1_RM_lrres2 results from LRtest(RM(raschdat1), split = "mean")

---

gofIRT *Various model tests and fit indices*

Description

This function computes various model tests and fit indices for objects of class ppar: Collapsed deviance, Casewise deviance, Rost's LR-test, Hosmer-Lemeshow test, R-Squared measures, confusion matrix, ROC analysis.

Usage

```
## S3 method for class 'ppar'
gofIRT(object, groups.hl = 10, cutpoint = 0.5)
```

Arguments

object	Object of class ppar (from person.parameter()).
groups.hl	Number of groups for Hosmer-Lemeshow test (see details).
cutpoint	Integer between 0 and 1 for computing the 0-1 model matrix from the estimated probabilities

Details

So far this test statistics are implemented only for dichotomous models without NA's. The Hosmer-Lemeshow test is computed by splitting the response vector into percentiles, e.g. groups.hl = 10 corresponds to decile splitting.

Value

The function gofIRT returns an object of class gof containing:

test.table	Ouput for model tests.
R2	List with R-squared measures.
classifier	Confusion matrix, accuracy, sensitivity, specificity.
AUC	Area under ROC curve.
Gini	Gini coefficient.
ROC	FPR and TPR for different cutpoints.
opt.cut	Optimal cutpoint determined by ROC analysis.
predobj	Prediction output from ROC analysis (ROCR package)

References

Mair, P., Reise, S. P., and Bentler, P. M. (2008). IRT goodness-of-fit using approaches from logistic regression. UCLA Statistics Preprint Series.

See Also

itemfit.ppar,personfit.ppar,LRtest

Examples

```
#Goodness-of-fit for a Rasch model
res <- RM(raschdat1)
pres <- person.parameter(res)
gof.res <- gofIRT(pres)
gof.res
summary(gof.res)
```

IC *Information criteria*

Description

Computation of information criteria such as AIC, BIC, and cAIC based on unconditional (joint), marginal, and conditional log-likelihood

Usage

```
## S3 method for class 'ppar'
IC(object)
```

Arguments

object	Object of class ppar (from person.parameter().

Details

The joint log-likelihood is established by summation of the logarithms of the estimated solving probabilities. The marginal log-likelihood can be computed directly from the conditional log-likelihood (see vignette for details).

Value

The function IC returns an object of class ICr containing:

ICtable	Matrix containing log-likelihood values, number of parameters, AIC, BIC, and cAIC for the joint, marginal, and conditional log-likelihood.

See Also

LRtest.Rm

Examples

```
#IC's for Rasch model
res <- RM(raschdat2)             #Rasch model
pres <- person.parameter(res)    #Person parameters
IC(pres)

#IC's for RSM
res <- RSM(rsmdat)
pres <- person.parameter(res)
IC(pres)
```

---

itemfit.ppar *Residuals, Personfit and Itemfit Statistics*

Description

pmat computes the theoretical person-item matrix with solving probabilities for each category (except 0th). residuals computes the squared and standardized residuals based on the observed and the expected person-item matrix. Chi-square based itemfit and personfit statistics can be obtained by using itemfit and personfit. Corrected item-test correlations in itemfit are computed using the approach from Cureton (1966).

Usage

```
## S3 method for class 'ppar'
pmat(object)
## S3 method for class 'ppar'
residuals(object,...)
## S3 method for class 'ppar'
itemfit(object)
## S3 method for class 'ppar'
personfit(object)
## S3 method for class 'ifit'
print(x, visible = TRUE,
sort_by = c("none", "p", "outfit_MSQ", "infit_MSQ", "outfit_t", "infit_t", "discrim"),
decreasing = FALSE, digits = 3,...)
## S3 method for class 'pfit'
print(x, visible = TRUE, ...)
## S3 method for class 'resid'
print(x, ...)
```

Arguments

object	Object of class ppar, derived from person.parameter.
x	Object of class ifit, pfit, or resid.
visible	If FALSE, returns the matrix of fit statistics that otherwise would be printed.
sort_by	Optionally the itemfit output can be sorted by one of these criteria.
decreasing	If sort_by is set, whether the output should be sorted in increasing or decreasing order.
digits	How many digits should be printed.
...	Further arguments passed to or from other methods. They are ignored in this function.

Value

pmat	Matrix of theoretical probabilities for each category except 0th (from function pmat).
i.fit	Chi-squared itemfit statistics (from function itemfit).
i.df	Degrees of freedom for itemfit statistics (from function itemfit).
st.res	Standardized residuals (from function itemfit).
i.outfitMSQ	Outfit mean-square statistics (from function itemfit).
i.infitMSQ	Infit mean-square statistics (from function itemfit).
i.disc	Corrected item-test correlations (from function itemfit).
p.fit	Chi-squared personfit statistics (from function personfit).
p.df	Degrees of freedom for personfit statistics (from function personfit).
st.res	Standardized residuals (from function personfit).
p.outfitMSQ	Outfit mean-square statistics (from function personfit).
p.infitMSQ	Infit mean-square statistics (from function personfit).

Author(s)

Patrick Mair, Reinhold Hatzinger, Moritz Heene

References

Smith Jr., E. V., and Smith, R. M. (2004). Introduction to Rasch Measurement. JAM press.

Wright, B.D., and Masters, G.N. Computation of OUTFIT and INFIT Statistics. Rasch Measurement Transactions, 1990, 3:4 p.84-85

Cureton, E. E. (1966). Corrected item-test correlations. Psychometrika, 31, 93-96

See Also

person.parameter

Examples

```
# Rasch model, estimation of item and person parameters
res <- RM(raschdat2)
p.res <- person.parameter(res)

# Matrix with expected probabilities and corresponding residuals
pmat(p.res)
residuals(p.res)

#Itemfit
itemfit(p.res)

#Personfit
personfit(p.res)
```

`item_info` *Calculate Item Information for 'eRm' objects*

Description

Calculates Samejima's (1969) information for all items.

Usage

```
item_info(ermobject, theta = seq(-5, 5, 0.01))

i_info(hvec, itembeta, theta)
```

Arguments

`ermobject`	An object of class 'eRm'.
`theta`	Supporting or sampling points on the latent trait.
`hvec`	Number of categories of a single item.
`itembeta`	Cumulative item category parameters for a single item.

Details

The function `item_info` calculates information of the whole set of items in the 'eRm' object. The function `i_info` does the same for a single item (and is called by `item_info`).

Value

Returns a list (`i_info`) or a list of lists (where each list element corresponds to an item, `item_info`) and contains

`c.info`	Matrix of category information in columns for the different theta values in rows.
`i.info`	Vector of item information for the different theta values.

Author(s)

Thomas Rusch

References

Samejima, F. (1969) Estimation of latent ability using a response pattern of graded scores. *Psychometric Monographs*, **17**.

See Also

The function to calculate the test information, `test_info` and the plot function `plotINFO`.

Examples

```
res <- PCM(pcmdat)
info <- item_info(res)
plotINFO(res,type="item")
```

LLRA *Fit Linear Logistic Models with Relaxed Assumptions (LLRA)*

Description

Automatically builds design matrix and fits LLRA.

Usage

```
LLRA(X, W, mpoints, groups, baseline, itmgrps = NULL, ...)

## S3 method for class 'llra'
print(x, ...)
```

Arguments

`X`	Data matrix as described in Hatzinger and Rusch (2009). It must be of wide format, e.g. for each person all item answers are written in columns for t1, t2, t3 etc. Hence each row corresponds to all observations for a single person. See llraDat1 for an example. Missing values are not allowed.
`W`	Design Matrix for LLRA to be passed to `LPCM`. If missing, it is generated automatically.
`mpoints`	The number of time points.
`groups`	Vector, matrix or data frame with subject/treatment covariates.
`baseline`	An optional vector with the baseline values for the columns in group.
`itmgrps`	Specifies how many groups of items there are. Currently not functional but may be useful in the future.
`x`	For the print method, an object of class `"llra"`.
`...`	Additional arguments to be passed to and from other methods.

Details

The function LLRA is a wrapper for LPCM to fit Linear Logistic Models with Relaxed Assumptions (LLRA). LLRA are extensions of the LPCM for the measurement of change over a number of discrete time points for a set of items. It can incorporate categorical covariate information. If no design matrix W is passed as an argument, it is built automatically from scratch.

Unless passed by the user, the baseline group is always the one with the lowest (alpha-)numerical value for argument groups. All other groups are labeled decreasingly according to the (alpha)-numerical value, e.g. with 2 treatment groups (TG1 and TG2) and one control group (CG), CG will be the baseline than TG1 and TG2. Hence the group effects are ordered like rev((unique(names(groupvec))) for naming.

Caution is advised as LLRA will fail if all changes for a group will be into a single direction (e.g. all subjects in the treatment group show improvement). Currently only data matrices are supported as arguments.

Value

Returns an object of class 'llra' (also inheriting from class 'eRm') containing

loglik	Conditional log-likelihood.
iter	Number of iterations.
npar	Number of parameters.
convergence	See code output in nlm.
etapar	Estimated basic item parameters. These are the LLRA effect parameters.
se.eta	Standard errors of the estimated basic item parameters.
betapar	Estimated item (easiness) parameters of the virtual items (not useful for interpretation here).
se.beta	Standard errors of virtual item parameters (not useful for interpretation here).
hessian	Hessian matrix if se = TRUE.
W	Design matrix.
X	Data matrix in long format. The columns correspond to the measurement points and each persons item answers are listed susequently in rows.
X01	Dichotomized data matrix.
groupvec	Assignment vector.
call	The matched call.
itms	The number of items.

Warning

A warning is printed that the first two categories for polytomous items are equated to save parameters. See Hatzinger and Rusch (2009) for a justification why this is valid also from a substantive point of view.

Author(s)

Thomas Rusch

References

Fischer, G.H. (1995) Linear logistic models for change. In G.H. Fischer and I. W. Molenaar (eds.), *Rasch models: Foundations, recent developments and applications* (pp. 157–181), New York: Springer.

Glueck, J. and Spiel, C. (1997) Item response models for repeated measures designs: Application and limitations of four different approaches. *Methods of Psychological Research*, **2**.

Hatzinger, R. and Rusch, T. (2009) IRT models with relaxed assumptions in eRm: A manual-like instruction. *Psychology Science Quarterly*, **51**, pp. 87–120.

See Also

The function to build the design matrix `build_W`, and the S3 methods `summary.llra` and `plotTR` and `plotGR` for plotting.

Examples

```
##Example 6 from Hatzinger & Rusch (2009)
groups <- c(rep("TG",30),rep("CG",30))
llra1 <- LLRA(llradat3,mpoints=2,groups=groups)
llra1

## Not run:
##An LLRA with 2 treatment groups and 1 baseline group, 5 items and 4
##time points. Item 1 is dichotomous, all others have 3, 4, 5, 6
##categories respectively.
dats <- llraDat2[1:20]
groups <- llraDat2$group
tps <- 4

#baseline CG
ex2 <- LLRA(dats,mpoints=tps,groups=groups)

#baseline TG1
ex2a <- LLRA(dats,mpoints=tps,groups=groups,baseline="TG1")

#summarize results
summary(ex2)
summary(ex2a)

#plotting
plotGR(ex2)
plotTR(ex2)
## End(Not run)
```

`llra.datprep` *Prepare Data Set for LLRA Analysis*

Description

Converts wide data matrix in long format, sorts subjects according to groups and builds assigment vector.

Usage

```
llra.datprep(X, mpoints, groups, baseline)
```

Arguments

X	Data matrix as described in Hatzinger and Rusch (2009). It must be of wide format, e.g. for each person all item answers are written in columns for t1, t2, t3 etc. Hence each row corresponds to all observations for a single person. Missing values are not allowed.
mpoints	The number of time points.
groups	Vector, matrix or data frame with subject/treatment covariates.
baseline	An optional vector with the baseline values for the columns in group.

Details

The function converts a data matrix from wide to long fromat as needed for LLRA. Additionally it sorts the subjects according to the different treatment/covariate groups. The group with the lowest (alpha-)numerical value will be the baseline.

Treatment and covariate groups are either defined by a vector, or by a matrix or data frame. The latter will be combined to a vector of groups corresponding to a combination of each factor level per column with the factor levels of the other column. The (constructed or passed) vector will then be used to create the assignment vector.

Value

Returns a list with the components

X	Data matrix in long format with subjects sorted by groups.
assign.vec	The assignment vector.
grp_n	A vector of the number of subjects in each group.

Author(s)

Reinhold Hatzinger

See Also

The function that uses this is LLRA. The values from llra.datprep can be passed to build_W.

Examples

```
# example 3 items, 3 timepoints, n=10, 2x2 treatments
dat<-sim.rasch(10,9)
tr1<-sample(c("a","b"),10,r=TRUE)
tr2<-sample(c("x","y"),10,r=TRUE)

# one treatment
res<-llra.datprep(dat,mpoints=3,groups=tr1)
res<-llra.datprep(dat,mpoints=3,groups=tr1,baseline="b")

# two treatments
res<-llra.datprep(dat,mpoints=3,groups=cbind(tr1,tr2))
res<-llra.datprep(dat,mpoints=3,groups=cbind(tr1,tr2),baseline=c("b","x"))

# two treatments - data frame
tr.dfr<-data.frame(tr1, tr2)
res<-llra.datprep(dat,mpoints=3,groups=tr.dfr)
```

---

llraDat1 *An Artificial LLRA Data Set*

Description

Artificial data set of 5 items, 5 time points and 5 groups for LLRA.

Usage

```
llraDat1
```

Format

A data frame with 150 observations of 26 variables.

`t1.I1` Answers to item 1 at time point 1

`t1.I2` Answers to item 2 at time point 1

`t1.I3` Answers to item 3 at time point 1

`t1.I4` Answers to item 4 at time point 1

`t1.I5` Answers to item 5 at time point 1

`t2.I1` Answers to item 1 at time point 2

`t2.I2` Answers to item 2 at time point 2

`t2.I3` Answers to item 3 at time point 2

`t2.I4` Answers to item 4 at time point 2

`t2.I5` Answers to item 5 at time point 2

`t3.I1` Answers to item 1 at time point 3

`t3.I2` Answers to item 2 at time point 3

`t3.I3` Answers to item 3 at time point 3

`t3.I4` Answers to item 4 at time point 3

`t3.I5` Answers to item 5 at time point 3

`t4.I1` Answers to item 1 at time point 4

`t4.I2` Answers to item 2 at time point 4

`t4.I3` Answers to item 3 at time point 4

`t4.I4` Answers to item 4 at time point 4

`t4.I5` Answers to item 5 at time point 4

`t5.I1` Answers to item 1 at time point 5

`t5.I2` Answers to item 2 at time point 5

`t5.I3` Answers to item 3 at time point 5

`t5.I4` Answers to item 4 at time point 5

`t5.I5` Answers to item 5 at time point 5

`groups` The group membership

Details

This is a data set as described in Hatzinger and Rusch (2009). 5 items were measured at 5 time points (in columns). Each row corresponds to one person (P1 to P150). There are 4 treatment groups and a control group. Treatment group G5 has size 10 (the first ten subjects), treatment group G4 has size 20, treatment group G3 has size 30, treatment group G2 has size 40 and the control group CG has size 50 (the last 50 subjects). Item 1 is dichotomous, all others are polytomous. Item 2, 3, 4 and 5 have 3, 4, 5, 6 categories respectively.

References

Hatzinger, R. and Rusch, T. (2009) IRT models with relaxed assumptions in eRm: A manual-like instruction. *Psychology Science Quarterly*, **51**, pp. 87–120.

Examples

```
llraDat1
```

`llraDat2` *An Artificial LLRA Data Set*

Description

Artificial data set of 70 subjects with 5 items, 4 time points and 3 groups for LLRA.

Usage

```
llraDat2
```

Format

A data frame with 70 observations of 21 variables.

`t1.I1` Answers to item 1 at time point 1

`t1.I2` Answers to item 2 at time point 1

`t1.I3` Answers to item 3 at time point 1

`t1.I4` Answers to item 4 at time point 1

`t1.I5` Answers to item 5 at time point 1

`t2.I1` Answers to item 1 at time point 2

`t2.I2` Answers to item 2 at time point 2

`t2.I3` Answers to item 3 at time point 2

`t2.I4` Answers to item 4 at time point 2

`t2.I5` Answers to item 5 at time point 2

`t3.I1` Answers to item 1 at time point 3

`t3.I2` Answers to item 2 at time point 3

`t3.I3` Answers to item 3 at time point 3

`t3.I4` Answers to item 4 at time point 3

`t3.I5` Answers to item 5 at time point 3

`t4.I1` Answers to item 1 at time point 4

`t4.I2` Answers to item 2 at time point 4

`t4.I3` Answers to item 3 at time point 4

`t4.I4` Answers to item 4 at time point 4

`t4.I5` Answers to item 5 at time point 4

`group` The group membership

Details

This is a data set as described in Hatzinger and Rusch (2009). 5 items were measured at 4 time points (in columns). Each persons answers to the items are recorded in the rows. There are 2 treatment groups and a control group. Treatment group 2 has size, 10, treatment group 1 has size 20 and the control group has size 40. Item 1 is dichotomous, all others are polytomous. Item 2, 3, 4 and 5 have 3, 4, 5, 6 categories respectively.

References

Hatzinger, R. and Rusch, T. (2009) IRT models with relaxed assumptions in eRm: A manual-like instruction. *Psychology Science Quarterly*, **51**, pp. 87–120.

Examples

```
llraDat2
```

`llradat3` *An Artificial LLRA Data Set*

Description

Artificial data set of 3 items, 2 time points and 2 groups for LLRA. It is example 6 from Hatzinger and Rusch (2009).

Usage

```
llradat3
```

Format

A data frame with 60 observations of 6 variables.

`V1` Answers to item 1 at time point 1

`V2` Answers to item 2 at time point 1

`V3` Answers to item 3 at time point 1

`V4` Answers to item 1 at time point 2

`V5` Answers to item 2 at time point 2

`V6` Answers to item 3 at time point 2

Details

This is a data set as described in Hatzinger and Rusch (2009).

References

Hatzinger, R. and Rusch, T. (2009) IRT models with relaxed assumptions in eRm: A manual-like instruction. *Psychology Science Quarterly*, **51**, pp. 87–120.

Examples

```
llradat3
```

LLTM *Estimation of linear logistic test models*

Description

This function computes the parameter estimates of a linear logistic test model (LLTM) for binary item responses by using CML estimation.

Usage

```
LLTM(X, W, mpoints = 1, groupvec = 1, se = TRUE, sum0 = TRUE,
   etaStart)
```

Arguments

X	Input 0/1 data matrix or data frame; rows represent individuals (N in total), columns represent items. Missing values have to be inserted as NA.
W	Design matrix for the LLTM. If omitted, the function will compute W automatically.
mpoints	Number of measurement points.
groupvec	Vector of length N which determines the group membership of each subject, starting from 1. If groupvec=1, no group contrasts are imposed.
se	If TRUE, the standard errors are computed.
sum0	If TRUE, the parameters are normalized to sum-0 by specifying an appropriate W. If FALSE, the first parameter is restricted to 0.
etaStart	A vector of starting values for the eta parameters can be specified. If missing, the 0-vector is used.

Details

Through appropriate definition of W the LLTM can be viewed as a more parsimonous Rasch model, on the one hand, e.g. by imposing some cognitive base operations to solve the items. One the other hand, linear extensions of the Rasch model such as group comparisons and repeated measurement designs can be computed. If more than one measurement point is examined, the item responses for the 2nd, 3rd, etc. measurement point are added column-wise in X.

If W is user-defined, it is nevertheless necessary to specify mpoints and groupvec. It is important that first the time contrasts and then the group contrasts have to be imposed.

Available methods for LLTM-objects are:
print, coef, model.matrix, vcov,summary, logLik, person.parameters.

Value

Returns on object of class eRm containing:

loglik	Conditional log-likelihood.
iter	Number of iterations.
npar	Number of parameters.
convergence	See code output in nlm.
etapar	Estimated basic item parameters.
se.eta	Standard errors of the estimated basic parameters.
betapar	Estimated item (easiness) parameters.
se.beta	Standard errors of item parameters.
hessian	Hessian matrix if se = TRUE.
W	Design matrix.
X	Data matrix.
X01	Dichotomized data matrix.
groupvec	Group membership vector.
call	The matched call.

Author(s)

Patrick Mair, Reinhold Hatzinger

References

Fischer, G. H., and Molenaar, I. (1995). Rasch Models - Foundations, Recent Developements, and Applications. Springer.

Mair, P., and Hatzinger, R. (2007). Extended Rasch modeling: The eRm package for the application of IRT models in R. Journal of Statistical Software, 20(9), 1-20.

Mair, P., and Hatzinger, R. (2007). CML based estimation of extended Rasch models with the eRm package in R. Psychology Science, 49, 26-43.

See Also

LRSM,LPCM

Examples

```
#LLTM for 2 measurement points
#100 persons, 2*15 items, W generated automatically
res1 <- LLTM(lltmdat1, mpoints = 2)
res1
summary(res1)

#Reparameterized Rasch model as LLTM (more pasimonious)
W <- matrix(c(1,2,1,3,2,2,2,1,1,1),ncol=2)             #design matrix
```

```
res2 <- LLTM(lltmdat2, W = W)
res2
summary(res2)
```

LPCM *Estimation of linear partial credit models*

Description

This function computes the parameter estimates of a linear partial credit model (LRSM) for polytomuous item responses by using CML estimation.

Usage

```
LPCM(X, W , mpoints = 1, groupvec = 1, se = TRUE, sum0 = TRUE,
   etaStart)
```

Arguments

X	Input data matrix or data frame; rows represent individuals (N in total), columns represent items. Missing values are inserted as NA.
W	Design matrix for the LPCM. If omitted, the function will compute W automatically.
mpoints	Number of measurement points.
groupvec	Vector of length N which determines the group membership of each subject, starting from 1
se	If TRUE, the standard errors are computed.
sum0	If TRUE, the parameters are normalized to sum-0 by specifying an appropriate W. If FALSE, the first parameter is restricted to 0.
etaStart	A vector of starting values for the eta parameters can be specified. If missing, the 0-vector is used.

Details

Through appropriate definition of W the LPCM can be viewed as a more parsimonous PCM, on the one hand, e.g. by imposing some cognitive base operations to solve the items. One the other hand, linear extensions of the Rasch model such as group comparisons and repeated measurement designs can be computed. If more than one measurement point is examined, the item responses for the 2nd, 3rd, etc. measurement point are added column-wise in X.

If W is user-defined, it is nevertheless necessary to specify mpoints and groupvec. It is important that first the time contrasts and then the group contrasts have to be imposed.

Available methods for LPCM-objects are:
print, coef, model.matrix, vcov,summary, logLik, person.parameters.

Value

Returns on object of class 'eRm' containing:

loglik	Conditional log-likelihood.
iter	Number of iterations.
npar	Number of parameters.
convergence	See code output in nlm.
etapar	Estimated basic item parameters.
se.eta	Standard errors of the estimated basic item parameters.
betapar	Estimated item (easiness) parameters.
se.beta	Standard errors of item parameters.
hessian	Hessian matrix if se = TRUE.
W	Design matrix.
X	Data matrix.
X01	Dichotomized data matrix.
groupvec	Group membership vector.
call	The matched call.

Author(s)

Patrick Mair, Reinhold Hatzinger

References

Fischer, G. H., and Molenaar, I. (1995). Rasch Models - Foundations, Recent Developements, and Applications. Springer.

Mair, P., and Hatzinger, R. (2007). Extended Rasch modeling: The **eRm** package for the application of IRT models in R. Journal of Statistical Software, 20(9), 1-20.

Mair, P., and Hatzinger, R. (2007). CML based estimation of extended Rasch models with the **eRm** package in R. Psychology Science, 49, 26-43.

See Also

LRSM,LLTM

Examples

```
#LPCM for two measurement points and two subject groups
#20 subjects, 2*3 items
G <- c(rep(1,10),rep(2,10))                  #group vector
res <- LPCM(lpcmdat, mpoints = 2, groupvec = G)
res
summary(res)
```

LRSM *Estimation of linear rating scale models*

Description

This function computes the parameter estimates of a linear rating scale model (LRSM) for polytomuous item responses by using CML estimation.

Usage

```
LRSM(X, W , mpoints = 1, groupvec = 1, se = TRUE, sum0 = TRUE,
   etaStart)
```

Arguments

X	Input data matrix or data frame; rows represent individuals (N in total), columns represent items. Missing values are inserted as NA.
W	Design matrix for the LRSM. If omitted, the function will compute W automatically.
mpoints	Number of measurement points.
groupvec	Vector of length N which determines the group membership of each subject, starting from 1
se	If TRUE, the standard errors are computed.
sum0	If TRUE, the parameters are normalized to sum-0 by specifying an appropriate W. If FALSE, the first parameter is restricted to 0.
etaStart	A vector of starting values for the eta parameters can be specified. If missing, the 0-vector is used.

Details

Through appropriate definition of W the LRSM can be viewed as a more parsimonous RSM, on the one hand, e.g. by imposing some cognitive base operations to solve the items. One the other hand, linear extensions of the Rasch model such as group comparisons and repeated measurement designs can be computed. If more than one measurement point is examined, the item responses for the 2nd, 3rd, etc. measurement point are added column-wise in X.

If W is user-defined, it is nevertheless necessary to specify mpoints and groupvec. It is important that first the time contrasts and then the group contrasts have to be imposed.

Available methods for LRSM-objects are: print, coef, model.matrix, vcov,summary, logLik, person.parameters.

Value

Returns an object of class 'eRm' containing:

loglik	Conditional log-likelihood.
iter	Number of iterations.
npar	Number of parameters.
convergence	See code output in nlm.
etapar	Estimated basic item parameters (item and category parameters).
se.eta	Standard errors of the estimated basic item parameters.
betapar	Estimated item (easiness) parameters.
se.beta	Standard errors of item parameters.
hessian	Hessian matrix if se = TRUE.
W	Design matrix.
X	Data matrix.
X01	Dichotomized data matrix.
groupvec	Group membership vector.
call	The matched call.

Author(s)

Patrick Mair, Reinhold Hatzinger

References

Fischer, G. H., and Molenaar, I. (1995). Rasch Models - Foundations, Recent Developements, and Applications. Springer.

Mair, P., and Hatzinger, R. (2007). Extended Rasch modeling: The **eRm** package for the application of IRT models in R. Journal of Statistical Software, 20(9), 1-20.

Mair, P., and Hatzinger, R. (2007). CML based estimation of extended Rasch models with the **eRm** package in R. Psychology Science, 49, 26-43.

See Also

LLTM,LPCM

Examples

```
#LRSM for two measurement points
#20 subjects, 2*3 items, W generated automatically,
#first parameter set to 0, no standard errors computed.

res <- LRSM(lrsmdat, mpoints = 2, groupvec = 1, sum0 = FALSE, se = FALSE)
res
```

LRtest *Computation of Andersen's LR-test.*

Description

This LR-test is based on subject subgroup splitting.

Usage

```
## S3 method for class 'Rm'
LRtest(object, splitcr = "median", se = TRUE)

## S3 method for class 'LR'
plotGOF(x, beta.subset = "all", main = "Graphical Model Check", xlab, ylab,
    tlab = "item", xlim, ylim, type = "p", pos = 4, conf = NULL, ctrline = NULL,
    smooline = NULL, asp = 1, x_axis = TRUE, y_axis = TRUE, set_par = TRUE,
    reset_par = TRUE, ...)
```

Arguments

`object`	Object of class `"Rm"`.
`splitcr`	Split criterion for subject raw score splitting. `"all.r"` corresponds to a full raw score split, `"median"` uses the median as split criterion, `"mean"` performs a mean split. Optionally `splitcr` can also be a vector which assigns each person to a certain subgroup (e.g., following an external criterion). This vector can be numeric, character or a factor.
`se`	controls computation of standard errors in the submodels (default: `TRUE`).
`x`	Object of class `"LR"`. Also used for visualizing the fit of single items.
`beta.subset`	If `"all"`, all items are plotted. Otherwise numeric subset vector can be specified.
`main`	Title of the plot.
`xlab`	Label on x-axis, default gives name of `splitcr` and level.
`ylab`	Label on y-axis, default gives name of `splitcr` and level.
`tlab`	Specification of item labels: `"item"` prints the item names, `"number"` gives integers corresponding to order of the beta parameters, if `"none"` no labels are printed. `"identify"` allows for an interactive labelling. Initially no labels are printed, after clicking close to an item point the corresponding label is added. The identification process is terminated by clicking the second button and selecting 'Stop' from the menu, or from the 'Stop' menu on the graphics window. For more information and basic operation see `identify`.
`xlim`	Limits on x-axis.
`ylim`	Limits on y-axis.
`type`	Plotting type (see `plot`).
`pos`	Position of the item label (see `text`).

conf	for plotting confidence ellipses for the item parameters. If conf = NULL (the default) no ellipses are drawn. Otherwise, conf must be specified as a list with optional elements: gamma, is the confidence level (numeric), col and lty, color and linetype (see par), which (numeric index vector) specifying for which items ellipses are drawn (must be a subset of beta.subset), and ia, logical, if the ellipses are to be drawn interactively (cf., tlab = "identify" above). For details about the default behavior, if conf is specified as a an empty list, see Details and Examples below. To use conf, the LR object x has to be generated using the option se = TRUE in LRtest(). For specification of col and which see Details and Examples below.
ctrline	for plotting confidence bands (control lines, cf. eg. Wright and Stone, 1999). If ctrline = NULL (the default) no lines are drawn. Otherwise, ctrline must be specified as a list with optional elements: gamma, is the confidence level (numeric), col and lty, color and linetype (see par). If ctrline is specified as ctrline = list(), the default values conf = list(gamma = 0.95, col = "blue", lty = "solid") will be used. See examples below. To use ctrline, the LR object x has to be generated using the option se = TRUE in LRtest().
smooline	spline smoothed confidence bands; must be specified as a list with optional elements: gamma, is the confidence level (numeric), col and lty, color and linetype, spar as smoothing parameter (see smooth.spline).
asp	sets the y/x ratio of the plot (see plot.window).
x_axis	if TRUE, the x-axis will be plotted.
y_axis	if TRUE, the y-axis will be plotted.
set_par	if TRUE, graphical parameters will be set by the function to optimize the plot's appearance. Unless reset_par = FALSE, these will be reset to the previous par settings.
reset_par	if TRUE, graphical parameters will be reset to defaults via par() after plotting (only if set_par = TRUE). To make adjustments *after* using plotGOF, this reset can be switched off. Note that the changed graphical parameters will remain in place unless they are redefined (using par()) or the device is closed.
...	additional parameters.

Details

If the data set contains missing values and mean or median is specified as split criterion, means or medians are calculated for each missing value subgroup and consequently used for raw score splitting.

When using interactive selection for both labelling of single points (tlab = "identify" and drawing confidence ellipses at certain points (ia = TRUE) then first all plotted points are labelled and afterwards all ellipses are generated. Both identification processes can be terminated by clicking the second (right) mouse button and selecting 'Stop' from the menu, or from the 'Stop' menu on the graphics window.

Using the specification which in allows for selectively drawing ellipses for certain items only, e.g., which = 1:3 draws ellipses for items 1 to 3 (as long as they are included in beta.subset). The default is drawing ellipses for all items. The element col in the conf list can either be a single color specification such as "blue" or a vector with color specifications for all items. The length must be

the same as the number of ellipses to be drawn. For color specification a palette can be set up using standard palettes (e.g., rainbow) or palettes from the colorspace or RColorBrewer package. An example is given below.

summary and print methods are available for objects of class LR.

Value

LRtest returns an object of class LR containing:

LR	LR-value.
df	Degrees of freedom of the test statistic.
Chisq	Chi-square value with corresponding df.
pvalue	P-value of the test.
likgroup	Log-likelihood values for the subgroups
betalist	List of beta parameters for the subgroups.
selist	List of standard errors of beta's.
etalist	List of eta parameters for the subgroups.
spl.gr	Names and levels for splitcr.
call	The matched call.
fitobj	List containing model objects from subgroup fit.

Author(s)

Patrick Mair, Reinhold Hatzinger, Marco J. Maier, Adrian Bruegger

References

Fischer, G. H., and Molenaar, I. (1995). Rasch Models - Foundations, Recent Developements, and Applications. Springer.

Mair, P., and Hatzinger, R. (2007). Extended Rasch modeling: The **eRm** package for the application of IRT models in R. Journal of Statistical Software, 20(9), 1-20.

Mair, P., and Hatzinger, R. (2007). CML based estimation of extended Rasch models with the **eRm** package in R. Psychology Science, 49, 26-43.

Wright, B.D., and Stone, M.H. (1999). Measurement essentials. Wide Range Inc., Wilmington. (https://www.rasch.org/measess/me-all.pdf 28Mb).

See Also

Waldtest, MLoef

Examples

```
# the object used is the result of running ... RM(raschdat1)
res <- raschdat1_RM_fitted       # see ? raschdat1_RM_fitted

# LR-test on dichotomous Rasch model with user-defined split
splitvec <- sample(1:2, 100, replace = TRUE)
lrres <- LRtest(res, splitcr = splitvec)
lrres
summary(lrres)

## Not run:
# goodness-of-fit plot with interactive labelling of items w/o standard errors
plotGOF(lrres, tlab = "identify")
## End(Not run)

# LR-test with a full raw-score split
X <- sim.rasch(1000, -2:2, seed = 5)
res2 <- RM(X)
full_lrt <- LRtest(res2, splitcr = "all.r")
full_lrt

## Not run:
# LR-test with mean split, standard errors for beta's
lrres2 <- LRtest(res, split = "mean")
## End(Not run)

# to save computation time, the results are loaded from raschdat1_RM_lrres2
lrres2 <- raschdat1_RM_lrres2                    # see ?raschdat1_RM_lrres2

# goodness-of-fit plot
# additional 95 percent control line with user specified style
plotGOF(lrres2, ctrline = list(gamma = 0.95, col = "red", lty = "dashed"))

# goodness-of-fit plot for items 1, 14, 24, and 25
# additional 95 percent confidence ellipses, default style
plotGOF(lrres2, beta.subset = c(14, 25, 24, 1), conf = list())

## Not run:
# goodness-of-fit plot for items 1, 14, 24, and 25
# for items 1 and 24 additional 95 percent confidence ellipses
# using colors for these 2 items from the colorspace package
library("colorspace")
my_colors <- rainbow_hcl(2)
plotGOF(lrres2, beta.subset = c(14, 25, 24, 1),
        conf = list(which = c(1, 14), col = my_colors))
## End(Not run)

# first, save current graphical parameters in an object
old_par <- par(mfrow = c(1, 2), no.readonly = TRUE)
# plots
plotGOF(lrres2, ctrline = list(gamma = 0.95, col = "red", lty = "dashed"),
  xlim = c(-3, 3), x_axis = FALSE, set_par = FALSE)
```

```
axis(1, seq(-3, 3, .5))

plotGOF(lrres2, conf = list(), xlim = c(-3, 3), x_axis = FALSE, set_par = FALSE)
axis(1, seq(-3, 3, .5))
text(-2, 2, labels = "Annotation")
# reset graphical parameters
par(old_par)
```

MLoef *Martin-Löf's Likelihood-Ratio-Test*

Description

This Likelihood-Ratio-Test is based on item subgroup splitting.

Usage

```
MLoef(robj, splitcr = "median")
```

Arguments

robj	An object of class 'Rm'.
splitcr	Split criterion to define the item groups. "median" and "mean" split items in two groups based on their items' raw scores. splitcr can also be a vector of length k (where k denotes the number of items) that takes two or more distinct values to define groups used for the Martin-Löf Test.

Details

This function implements a generalization of the Martin-Löf test for polytomous items as proposed by Christensen, Bjørner, Kreiner & Petersen (2002), but does currently not allow for missing values.

If the split criterion is "median" or "mean" and one or more items' raw scores are equal the median resp. mean, MLoef will assign those items to the lower raw score group. summary.MLoef gives detailed information about the allocation of all items.

summary and print methods are available for objects of class 'MLoef'.

An 'exact' version of the Martin-Löf test for binary items is implemented in the NPtest function.

Value

MLoef returns an object of class MLoef containing:

LR	LR-value
df	degrees of freedom
p.value	*p*-value of the test
fullModel	the overall Rasch model

subModels	a list containing the submodels
Lf	log-likelihood of the full model
Ls	list of the sub models' log-likelihoods
i.groups	a list of the item groups
splitcr	submitted split criterion
split.vector	binary allocation of items to groups
warning	items equalling median or mean for the respective split criteria
call	the matched call

Author(s)

Marco J. Maier, Reinhold Hatzinger

References

Christensen, K. B., Bjørner, J. B., Kreiner S. & Petersen J. H. (2002). Testing unidimensionality in polytomous Rasch models. *Psychometrika, (67)*4, 563–574.

Fischer, G. H., and Molenaar, I. (1995). *Rasch Models – Foundations, Recent Developements, and Applications.* Springer.

Rost, J. (2004). *Lehrbuch Testtheorie – Testkonstruktion.* Bern: Huber.

See Also

LRtest, Waldtest

Examples

```
# Martin-Löf-test on dichotomous Rasch model using "median" and a user-defined
# split vector. Note that group indicators can be of character and/or numeric.
splitvec <- c(1, 1, 1, "x", "x", "x", 0, 0, 1, 0)

res <- RM(raschdat1[,1:10])

MLoef.1 <- MLoef(res, splitcr = "median")
MLoef.2 <- MLoef(res, splitcr = splitvec)

MLoef.1

summary(MLoef.2)
```

`NonparametricTests` *A Function to Perform Nonparametric Rasch Model Tests*

Description

A variety of nonparametric tests as proposed by Ponocny (2001), Koller and Hatzinger (2012), and an 'exact' version of the Martin-Löf test are implemented. The function operates on random binary matrices that have been generated using an MCMC algorithm (Verhelst, 2008) from the **RaschSampler** package (Hatzinger, Mair, and Verhelst, 2009).

Usage

```
NPtest(obj, n = NULL, method = "T1", ...)
```

Arguments

`obj`	A binary data matrix (or data frame) or an object containing the output from the **RaschSampler** package.
`n`	If `obj` is a matrix or a data frame, `n` is the number of sampled matrices (default is 500)
`method`	One of the test statistics. See Details below.
`...`	Further arguments according to `method`. See Details below. Additionally, the sampling routine can be controlled by specifying `burn_in`, `step`, and `seed` (for details see below and `rsctrl`). A summary of the sampling object may be obtained using the option `RSinfo = TRUE`.

Details

The function uses the **RaschSampler** package, which is now packaged with **eRm** for convenience. It can, of course, still be accessed and downloaded separately via CRAN.

As an input the user has to supply either a binary data matrix or a **RaschSampler** output object. If the input is a data matrix, the **RaschSampler** is called with default values (i.e., `rsctrl(burn_in = 256, n_eff = n, step = 32)`, see `rsctrl`), where `n` corresponds to `n_eff` (the default number of sampled matrices is 500). By default, the starting values for the random number generators (`seed`) are chosen randomly using system time. Methods other than those listed below can easily be implemented using the **RaschSampler** package directly.

The currently implemented methods (following Ponocny's notation of T-statistics) and their options are:

T_1**:** `method = "T1"`
: Checks for local dependence via increased inter-item correlations. For all item pairs, cases are counted with equal responses on both items.

T_{1m}**:** `method = "T1m"`
: Checks for multidimensionality via decreased inter-item correlations. For all item pairs, cases are counted with equal responses on both items.

T_{1l}: `method = "T1l"`
Checks for learning. For all item pairs, cases are counted with response pattern (1,1).

T_{md}: `method = "Tmd", idx1, idx2`
`idx1` and `idx2` are vectors of indices specifying items which define two subscales, e.g., `idx1 = c(1, 5, 7)` and `idx2 = c(3, 4, 6)`
Checks for multidimensionality based on correlations of person raw scores for the subscales.

T_2: `method = "T2", idx = NULL, stat = "var"`
`idx` is a vector of indices specifying items which define a subscale, e.g., `idx = c(1, 5, 7)`
`stat` defines the used statistic as a character object which can be: `"var"` (variance), `"mad1"` (mean absolute deviation), `"mad2"` (median absolute deviation), or `"range"` (range)
Checks for local dependence within model deviating subscales via increased dispersion of subscale person rawscores.

T_{2m}: `method = "T2m", idx = NULL, stat = "var"`
`idx` is a vector of indices specifying items which define a subscale, e.g., `idx = c(1, 5, 7)`
`stat` defines the used statistic as a character object which can be: `"var"` (variance), `"mad1"` (mean absolute deviation), `"mad2"` (median absolute deviation), `"range"` (range)
Checks for multidimensionality within model deviating subscales via decreased dispersion of subscale person rawscores.

T_4: `method = "T4", idx = NULL, group = NULL, alternative = "high"`
`idx` is a vector of indices specifying items which define a subscale, e.g., `idx = c(1, 5, 7)`
`group` is a logical vector defining a subject group, e.g., `group = ((age >= 20) & (age < 30))`
`alternative` specifies the alternative hypothesis and can be: `"high"` or `"low"`.
Checks for group anomalies (DIF) via too high (low) raw scores on item(s) for specified group.

T_{10}: `method = "T10", splitcr = "median"`
`splitcr` defines the split criterion for subject raw score splitting. `"median"` uses the median as split criterion, `"mean"` performs a mean-split. Optionally, `splitcr` can also be a vector which assigns each person to one of two subgroups (e.g., following an external criterion). This vector can be numeric, character, logical, or a factor.
Global test for subgroup-invariance. Checks for different item difficulties in two subgroups (for details see Ponocny, 2001).

T_{11}: `method = "T11"`
Global test for local dependence. The statistic calculates the sum of absolute deviations between the observed inter-item correlations and the expected correlations.

T_{pbis}: `method = "Tpbis", idxt, idxs`
Test for discrimination. The statistic calculates a point-biserial correlation for a test item (specified via `idxt`) with the person row scores for a subscale of the test sum (specified via `idxs`). If the correlation is too low, the test item shows different discrimination compared to the items of the subscale.

Martin-Löf The 'exact' version of the *Martin-Löf* statistic is specified via `method = "MLoef"` and optionally `splitcr` (see `MLoef`).

Q_{3h}: `method = "Q3h"`
Checks for local dependence by detecting an increased correlation of inter-item residuals. Low p-values correspond to a high ("h") correlation between two items.

Q_{3l}: `method = "Q3l"`
Checks for local dependence by detecting a decreased correlation of inter-item residuals. Low p-values correspond to a low ("l") correlation between two items.

Value

Depending on the `method` argument, a list is returned which has one of the following classes: `'T1obj'`, `'T1mobj'`, `'T1lobj'`, `'Tmdobj'`, `'T2obj'`, `'T2mobj'`, `'T4obj'`, `'T10obj'`, `'T11obj'`, `'Tpbisobj'`, `'Q3hobj'` or `'Q3lobj'`.

The main output element is `prop` giving the one-sided p-value, i.e., the number of statistics from the sampled matrices which are equal or exceed the statistic based on the observed data. For T_1, T_{1m}, and T_{1l}, `prop` is a vector. For the Martin-Löf test, the returned object is of class `'MLobj'`. Besides other elements, it contains a `prop` vector and `MLres`, the output object from the asymptotic Martin-Löf test on the input data.

Note

The **RaschSampler** package is no longer required to use `NPtest` since **eRm** version 0.15-0.

Author(s)

Reinhold Hatzinger

References

Ponocny, I. (2001). Nonparametric goodness-of-fit tests for the Rasch model. *Psychometrika, 66*(3), 437–459. doi:10.1007/BF02294444

Verhelst, N. D. (2008). An efficient MCMC algorithm to sample binary matrices with fixed marginals. *Psychometrika, 73*(4), 705–728. doi:10.1007/s1133600890623

Verhelst, N., Hatzinger, R., & Mair, P. (2007). The Rasch sampler. *Journal of Statistical Software, 20*(4), 1–14. https://www.jstatsoft.org/v20/i04

Koller, I., & Hatzinger, R. (2013). Nonparametric tests for the Rasch model: Explanation, development, and application of quasi-exact tests for small samples. *Interstat, 11*, 1–16. http://interstat.statjournals.net/YEAR/2013/abstracts/1311002.php

Koller, I., Maier, M. J., & Hatzinger, R. (2015). An Empirical Power Analysis of Quasi-Exact Tests for the Rasch Model: Measurement Invariance in Small Samples. *Methodology, 11*(2), 45–54. doi:10.1027/16142241/a000090

Debelak, R., & Koller, I. (2019). Testing the Local Independence Assumption of the Rasch Model With Q3-Based Nonparametric Model Tests. *Applied Psychological Measurement* doi:10.1177/0146621619835501

Examples

```
### Preparation:

# data for examples below
X <- as.matrix(raschdat1)

# generate 100 random matrices based on original data matrix
rmat <- rsampler(X, rsctrl(burn_in = 100, n_eff = 100, seed = 123))

## the following examples can also directly be used by setting
## rmat <- as.matrix(raschdat1)
```

```
## without calling rsampler() first
t1 <- NPtest(rmat, n = 100, method = "T1")


### Examples ###########################################################################

###--- T1 ---------------------------------------------------------------------
t1 <- NPtest(rmat, method = "T1")
# choose a different alpha for selecting displayed values
print(t1, alpha = 0.01)


###--- T2 ---------------------------------------------------------------------
t21 <- NPtest(rmat, method = "T2", idx = 1:5, burn_in = 100, step = 20,
              seed = 7654321, RSinfo = TRUE)
# default stat is variance
t21

t22 <- NPtest(rmat, method = "T2", stat = "mad1",
              idx = c(1, 22, 5, 27, 6, 9, 11))
t22


###--- T4 ---------------------------------------------------------------------
age <- sample(20:90, 100, replace = TRUE)
# group MUST be a logical vector
# (value of TRUE is used for group selection)
age <- age < 30
t41 <- NPtest(rmat, method = "T4", idx = 1:3, group = age)
t41

sex <- gl(2, 50)
# group can also be a logical expression (generating a vector)
t42 <- NPtest(rmat, method = "T4", idx = c(1, 4, 5, 6), group = sex == 1)
t42


###--- T10 --------------------------------------------------------------------
t101 <- NPtest(rmat, method = "T10")       # default split criterion is "median"
t101

## Not run:
split <- runif(100)
t102 <- NPtest(rmat, method = "T10", splitcr = split > 0.5)
t102

t103 <- NPtest(rmat, method = "T10", splitcr = sex)
t103
## End(Not run)


###--- T11 --------------------------------------------------------------------
t11 <- NPtest(rmat, method = "T11")
```

```
t11


###--- Tpbis ----------------------------------------------------------------
tpb <- NPtest(X[, 1:5], method = "Tpbis", idxt = 1, idxs = 2:5)
tpb


###--- Martin-Löf -----------------------------------------------------------
## Not run:
# takes a while ...
split <- rep(1:3, each = 10)
NPtest(raschdat1, n = 100, method = "MLoef", splitcr = split)
## End(Not run)
```

PCM *Estimation of partial credit models*

Description

This function computes the parameter estimates of a partial credit model for polytomous item responses by using CML estimation.

Usage

```
PCM(X, W, se = TRUE, sum0 = TRUE, etaStart)
```

Arguments

X	Input data matrix or data frame with item responses (starting from 0); rows represent individuals, columns represent items. Missing values are inserted as NA.
W	Design matrix for the PCM. If omitted, the function will compute W automatically.
se	If TRUE, the standard errors are computed.
sum0	If TRUE, the parameters are normed to sum-0 by specifying an appropriate W. If FALSE, the first parameter is restricted to 0.
etaStart	A vector of starting values for the eta parameters can be specified. If missing, the 0-vector is used.

Details

Through specification in W, the parameters of the categories with 0 responses are set to 0 as well as the first category of the first item. Available methods for PCM-objects are:
print, coef, model.matrix, vcov, plot, summary, logLik, person.parameters, plotICC, LRtest.

Value

Returns an object of class `Rm, eRm` containing.

`loglik`	Conditional log-likelihood.
`iter`	Number of iterations.
`npar`	Number of parameters.
`convergence`	See code output in `nlm`.
`etapar`	Estimated basic item difficulty parameters.
`se.eta`	Standard errors of the estimated basic item parameters.
`betapar`	Estimated item-category (easiness) parameters.
`se.beta`	Standard errors of item parameters.
`hessian`	Hessian matrix if `se = TRUE`.
`W`	Design matrix.
`X`	Data matrix.
`X01`	Dichotomized data matrix.
`call`	The matched call.

Author(s)

Patrick Mair, Reinhold Hatzinger

References

Fischer, G. H., and Molenaar, I. (1995). Rasch Models - Foundations, Recent Developements, and Applications. Springer.

Mair, P., and Hatzinger, R. (2007). Extended Rasch modeling: The **eRm** package for the application of IRT models in R. Journal of Statistical Software, 20(9), 1-20.

Mair, P., and Hatzinger, R. (2007). CML based estimation of extended Rasch models with the **eRm** package in R. Psychology Science, 49, 26-43.

See Also

`RM`,`RSM`,`LRtest`

Examples

```
##PCM with 10 subjects, 3 items
res <- PCM(pcmdat)
res
summary(res)                #eta and beta parameters with CI
thresholds(res)             #threshold parameters
```

`person.parameter` *Estimation of Person Parameters*

Description

Maximum likelihood estimation of the person parameters with spline interpolation for non-observed and 0/full responses. Extraction of information criteria such as AIC, BIC, and cAIC based on unconditional log-likelihood.

Usage

```
## S3 method for class 'eRm'
person.parameter(object)
## S3 method for class 'ppar'
summary(object, ...)
## S3 method for class 'ppar'
print(x, ...)
## S3 method for class 'ppar'
plot(x, xlab = "Person Raw Scores",
   ylab = "Person Parameters (Theta)", main = NULL, ...)
## S3 method for class 'ppar'
coef(object, extrapolated = TRUE, ...)
## S3 method for class 'ppar'
logLik(object, ...)
## S3 method for class 'ppar'
confint(object, parm, level = 0.95, ...)
```

Arguments

`object`	Object of class `'eRm'` in `person.parameter` and object of class `ppar` in `IC`.

Arguments for `print` and `plot` methods:

`x`	Object of class `ppar`.
`xlab`	Label of the x-axis.
`ylab`	Label of the y-axis.
`main`	Title of the plot.
`...`	Further arguments to be passed to or from other methods. They are ignored in this function.

Arguments for the `coef` method:

`extrapolated`	either returns extrapolated values for raw scores 0 and k or sets them `NA`

Arguments for `confint`:

`parm`	Parameter specification (ignored).
`level`	Alpha-level.

Details

If the data set contains missing values, person parameters are estimated for each missing value subgroup.

Value

The function `person.parameter` returns an object of class `ppar` containing:

`loglik`	Log-likelihood of the collapsed data (for faster estimation persons with the same raw score are collapsed).
`npar`	Number of parameters.
`niter`	Number of iterations.
`thetapar`	Person parameter estimates.
`se.theta`	Standard errors of the person parameters.
`hessian`	Hessian matrix.
`theta.table`	Matrix with person parameters (ordered according to original data) including NA pattern group.
`pers.ex`	Indices with persons excluded due to 0/full raw score
`X.ex`	Data matrix with persons excluded
`gmemb`	NA group membership vector (0/full persons excluded)

The function `coef` returns a vector of the person parameter estimates for each person (i.e., the first column of `theta.table`).

The function `logLik` returns an object of class `loglik.ppar` containing:

`loglik`	Log-likelihood of the collapsed data (see above).
`df`	Degrees of freedom.

Author(s)

Patrick Mair, Reinhold Hatzinger

References

Fischer, G. H., and Molenaar, I. (1995). Rasch Models - Foundations, Recent Developements, and Applications. Springer.

Mair, P., and Hatzinger, R. (2007). Extended Rasch modeling: The **eRm** package for the application of IRT models in R. Journal of Statistical Software, 20(9), 1-20.

Mair, P., and Hatzinger, R. (2007). CML based estimation of extended Rasch models with the **eRm** package in R. Psychology Science, 49, 26-43.

See Also

`itemfit.ppar`,`personfit.ppar`

Examples

```
#Person parameter estimation of a rating scale model
res <- RSM(rsmdat)
pres <- person.parameter(res)
pres
summary(pres)
plot(pres)

#Person parameter estimation for a Rasch model with missing values
res <- RM(raschdat2, se = FALSE) #Rasch model without standard errors
pres <- person.parameter(res)
pres                             #person parameters
summary(pres)
logLik(pres)                     #log-likelihood of person parameter estimation
```

PersonMisfit *Person Misfit*

Description

This function counts the number of persons who do not fit the Rasch model. More specifically, it returns the proportion and frequency of persons - or more generally cases - who exceed a Chi-square based Z-value of 1.96 (suggesting a statistically significant deviation from the predicted response pattern).

Usage

```
## S3 method for class 'ppar'
PersonMisfit(object)
```

Arguments

`object`	Object of class `ppar`.

Details

Returns the proportion and absolute number of persons who do not fit the Rasch model (Z-values > 1.96).

Value

`PersonMisfit` returns an object of class `MisfittingPersons` containing:

`PersonMisfit`	the proportion of misfitting persons,
`count.misfit.Z`	the frequency of misfitting person,
`total.persons`	the number of persons for whom a fit value was estimated.

Author(s)

Adrian Bruegger

Examples

```
rm1 <- RM(raschdat1)
pers <- person.parameter(rm1)
pmfit <- PersonMisfit(pers)
pmfit
summary(pmfit)
```

---

phi.range *Example User Function*

Description

Calculates the R_ϕ statistic, i.e., the range of the inter-column correlations (ϕ-coefficients) for a binary matrix.

Usage

```
phi.range(mat)
```

Arguments

mat	a binary matrix

Value

The range of the inter-column correlations

Examples

```
ctr <- rsctrl(burn_in = 10, n_eff = 5, step=10, seed = 123, tfixed = FALSE)
mat <- matrix(sample(c(0,1), 50, replace = TRUE), nr = 10)
rso <- rsampler(mat, ctr)
rso_st <- rstats(rso,phi.range)
print(unlist(rso_st))
```

plotDIF *Confidence intervals plot of item parameter estimates.*

Description

Performs an plot of item parameter conficence intervals based on LRtest subgroup splitting.

Usage

```
plotDIF(object, item.subset = NULL, gamma = 0.95, main = NULL,
          xlim = NULL,  xlab = " ", ylab=" ", col = NULL,
          distance, splitnames=NULL, leg = FALSE, legpos="bottomleft", ...)
```

Arguments

object	An object of class LR (if more objects should be plotted, the argument has to be defined as a list).
item.subset	Subset of items to be plotted. Either a numeric vector indicating the items or a character vector indicating the itemnames. If nothing is defined (default), all items are plotted.
gamma	The level for the item parameter's confidence limits (default is gamma = 0.95).
main	Main title for the plot.
xlim	Numeric vector of length 2, giving the x coordinates ranges of the plot (the y coordinates depend on the number of depicted items).
xlab	Label for the x axis.
ylab	Label for the y axis.
col	By default the color for the drawn confidence lines is determined automatically whereas every group (split criterion) is depicted in the same color.
distance	Distance between each item's confidence lines – if omitted, the distance shrinks with increasing numbers of split criteria. Can be overriden using values in (0, 0.5).
splitnames	For labeling the splitobjects in the legend (returns a nicer output).
leg	If TRUE a legend is provided by default.
legpos	Position of the legend with possible values "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" and "center". The default value for the legend is "bottomleft".
...	Further options to be passed to plot.

Details

If there are items that cannot be estimated for some reasons, certainly these ones are not plotted. For plotting several objects of class `LR`, the subgroup splitting by `LRtest` has to be carried out for the same data set (or at least item subsets of it).

Plotting a certain subset of items could be useful if the objects of class `LR` contain a huge number of estimated items.

The default level for the conficence limits is gamma = 0.95. (If the conficence limits should be corrected it is useful to use a correction, e.g., Bonferroni: 1 - (1 - gamma) / number of estimated items.)

Value

`plotCI` returns a list containing the confidence limits of each group in each `LRtest` object.

Author(s)

Kathrin Gruber, Reinhold Hatzinger

See Also

`LRtest`, `confint.threshold`, `thresholds`

Examples

```
# the object used is the result of running   RM(raschdat1)
res <- raschdat1_RM_fitted     # see ? raschdat1_RM_fitted

## Not run:
# LR-test on dichotomous Rasch model with user-defined split
splitvec <- rep(1:2, each = 50)
lrres <- LRtest(res, splitcr = splitvec)

# LR-test with mean split
lrres2 <- LRtest(res, split = "mean")

# combination of LRtest-objects in a list
RMplotCI <- list(lrres, lrres2)
## End(Not run)

# the object raschdat1_RM_plotDIF is the result of the computations outlined
# above and is loaded to save computation time. see ?raschdat1_RM_plotDIF
RMplotCI <- raschdat1_RM_plotDIF

# Confidence intervals plot with default assumptions
plotDIF(RMplotCI)

# Confidence intervals plot with Bonferroni correction
plotDIF(RMplotCI, gamma = (1 - (0.05/10)))

# Confidence intervals plot for an item subset
```

```
plotDIF(RMplotCI, item.subset = 1:6)

# with user defined group color and legend
plotDIF(RMplotCI, col = c("red", "blue"), leg = TRUE, legpos = "bottomright")

# with names for the splitobjects
plotDIF(RMplotCI, col = c("red", "blue"), leg = TRUE, legpos = "bottomright",
        splitnames = c(paste("User", 1:2), paste(rep("Mean", 2), 1:2)))
```

`plotGR` *Plot Treatment or Covariate Effects for LLRA*

Description

Plots treatment or covariate group effects over time.

Usage

```
plotGR(object, ...)
```

Arguments

`object`	an object of class "llra".
`...`	Additional parameters to be passed to and from other methods.

Details

The plot is a lattice plot with each panel corresponding to an item. The effects are plotted for each groups (including baseline) over the different time points. The groups are given the same names as for the parameter estimates (derived from groupvec).

Please note that all effects are to be interpreted relative to the baseline.

Currently, this function only works for a full item x treatment x timepoints LLRA. Collapsed effects will not be displayed properly.

Warning:

Objects of class `"llra"` that contain estimates from a collapsed data matrix will not be displayed correctly.

Author(s)

Thomas Rusch

See Also

The plot method for trend effects `plotTR`.

Examples

```
##Example 6 from Hatzinger & Rusch (2009)
groups <- c(rep("TG",30),rep("CG",30))
llra1 <- LLRA(llradat3,mpoints=2,groups=groups)
summary(llra1)
plotGR(llra1)

## Not run:
##An LLRA with 2 treatment groups and 1 baseline group, 5 items and 4
##time points. Item 1 is dichotomous, all others have 3, 4, 5, 6
##categories respectively.
ex2 <- LLRA(llraDat2[1:20],mpoints=4,groups=llraDat2[21])
plotGR(ex2)
## End(Not run)
```

---

plotICC *ICC Plots*

Description

Plot functions for visualizing the item characteristic curves

Usage

```
## S3 method for class 'Rm'
plotICC(object, item.subset = "all", empICC = NULL, empCI = NULL,
   mplot = NULL, xlim = c(-4, 4), ylim = c(0, 1),
   xlab = "Latent Dimension", ylab = "Probability to Solve", main=NULL,
   col = NULL, lty = 1, legpos = "left", ask = TRUE, ...)
## S3 method for class 'dRm'
plotjointICC(object, item.subset = "all", legend = TRUE,
   xlim = c(-4, 4), ylim = c(0, 1), xlab = "Latent Dimension",
   ylab = "Probability to Solve", lty = 1, legpos = "topleft",
   main="ICC plot",col=NULL,...)
```

Arguments

`object`	object of class `Rm` or `dRm`
`item.subset`	Subset of items to be plotted. Either a numeric vector indicating the column in `X` or a character vector indiciating the column name. If `"all"` (default), all items are plotted.
`empICC`	Plotting the empirical ICCs for objects of class `dRm`. If `empICC=NULL` (the default) the empirical ICC is not drawn. Otherwise, `empICC` must be specified as a list where the first element must be one of `"raw"`, `"loess"`, `"tukey"`, `"kernel"`. The other optional elements are `smooth` (numeric), `type` (line type for empirical ICCs, useful values are `"p"` (default), `"l"`, and `"b"`, see graphics parameter `type` in `plot.default`), `pch`, `col`, and `lty`, plotting 'character', colour and linetype (see `par`). See details and examples below.

empCI	Plotting confidence intervals for the the empirical ICCs. If empCI=NULL (the default) no confidence intervals are drawn. Otherwise, by specifying empCI as a list gives 'exact' confidence intervals for each point of the empirical ICC. The optional elements of this list are gamma, the confidence level, col, colour, and lty, line type. If empCI is specified as an empty list, the default values empCI=list(gamma=0.95,col="red",lty="dotted") will be used.
mplot	if NULL the default setting is in effect. For models of class dRm this is mplot = TRUE, i.e., the ICCs for up to 4 items are plotted in one figure. For Rm models the default is FALSE (each item in one figure) but may be set to TRUE.
xlab	Label of the x-axis.
ylab	Label of the y-axis.
xlim	Range of person parameters.
ylim	Range for probability to solve.
legend	If TRUE, legend is provided, otherwise the ICCs are labeled.
col	If not specified or NULL, line colors are determined automatically. Otherwise, a scalar or vector with appropriate color specifications may be supplied (see par).
lty	Line type.
main	Title of the plot.
legpos	Position of the legend with possible values "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" and "center". If FALSE no legend is displayed.
ask	If TRUE (the default) and the R session is interactive the user is asked for input, before a new figure is drawn. FALSE is only useful if automated figure export is in effect, e.g., when using Sweave.
...	Additional plot parameters.

Details

Empirical ICCs for objects of class dRm can be plotted using the option empICC, a list where the first element specifies the type of calculation of the empirical values. If empICC=list("raw", other specifications) relative frequencies of the positive responses are calculated for each rawscore group and plotted at the position of the corresponding person parameter. The other options use the default versions of various smoothers: "tukey" (see smooth), "loess" (see loess), and "kernel" (see ksmooth). For "loess" and "kernel" a further element, smooth, may be specified to control the span (default is 0.75) or the bandwith (default is 0.5), respectively. For example, the specification could be empirical = list("loess", smooth=0.9) or empirical = list("kernel",smooth=2). Higher values result in smoother estimates of the empirical ICCs.

The optional confidence intervals are obtained by a procedure first given in Clopper and Pearson (1934) based on the beta distribution (see binom.test).

Note

For most of the plot options see plot and par.

Author(s)

Patrick Mair, Reinhold Hatzinger

See Also

plotGOF

Examples

```
## Not run:
# Rating scale model, ICC plot for all items
rsm.res <- RSM(rsmdat)
thresholds(rsm.res)
plotICC(rsm.res)

# now items 1 to 4 in one figure without legends
plotICC(rsm.res, item.subset = 1:4, mplot = TRUE, legpos = FALSE)

# Rasch model for items 1 to 8 from raschdat1
# empirical ICCs displaying relative frequencies (default settings)
rm8.res <- RM(raschdat1[,1:8])
plotICC(rm8.res, empICC=list("raw"))

# the same but using different plotting styles
plotICC(rm8.res, empICC=list("raw",type="b",col="blue",lty="dotted"))

# kernel-smoothed empirical ICCs using bandwidth = 2
plotICC(rm8.res, empICC = list("kernel",smooth=3))

# raw empirical ICCs with confidence intervals
# displaying only items 2,3,7,8
plotICC(rm8.res, item.subset=c(2,3,7,8), empICC=list("raw"), empCI=list())

# Joint ICC plot for items 2, 6, 8, and 15 for a Rasch model
res <- RM(raschdat1)
plotjointICC(res, item.subset = c(2,6,8,15), legpos = "left")

## End(Not run)
```

---

plotINFO *Plot Information For 'eRm' objects*

Description

Calculates and plots the individual item-category information (type='category'), item information (type='item') or test/scale information (i.e., summed item information, type='scale' or 'test')) as defined by Samejima (1969)

Usage

```
plotINFO(ermobject, type = "both", theta = seq(-6, 6, length.out = 1001L),
legpos = "topright", ...)
```

Arguments

ermobject	An object of class 'eRm'.
type	A string denoting the type of information to be plotted. Currently supports "category", "item", "test", "scale" and "both" (which gives item and scale information; default).
theta	Supporting or sampling points on the latent trait.
legpos	Defines the positioning of the legend, as in plotICC.
...	Further arguments. xlab sets the label of the x axis. ylabI and ylabT control the labeling of the item or test information plot. mainI and mainT set the titles for item/test information plots.

Author(s)

Thomas Rusch

References

Samejima, F. (1969) Estimation of latent ability using a response pattern of graded scores. *Psychometric Monographs*, **17**.

See Also

The function to calculate the item or test information, item_info and test_info.

Examples

```
res <- PCM(pcmdat)
plotINFO(res)
```

---

plotPImap *Person-Item Map*

Description

A person-item map displays the location of item (and threshold) parameters as well as the distribution of person parameters.along the latent dimension. Person-item maps are useful to compare the range and position of the item measure distribution (lower panel) to the range and position of the person measure distribution (upper panel). Items should ideally be located along the whole scale to meaningfully measure the 'ability' of all persons.

Usage

```
plotPImap(object, item.subset = "all", sorted = FALSE,
   main = "Person-Item Map", latdim = "Latent Dimension",
   pplabel = "Person\nParameter\nDistribution", cex.gen = 0.7,
   xrange = NULL, warn.ord = TRUE, warn.ord.colour = "black",
   irug = TRUE, pp = NULL)
```

Arguments

object	Object of class Rm or dRm
item.subset	Subset of items to be plotted. Either a numeric vector indicating the column in X or a character vector indicating the column name. If "all", all items are plotted. The number of items to be plotted must be > 1.
sorted	If TRUE, the items are sorted in increasing order according to their location on the latent dimension.
main	Main title of the plot.
latdim	Label of the x-axis, i.e., the latent dimension.
pplabel	Title for the upper panel displaying the person parameter distribution
cex.gen	cex as a graphical parameter specifies a numerical value giving the amount by which plotting text and symbols should be magnified relative to the default. Here cex.gen applies to all text labels. The default is 0.7.
xrange	Range for the x-axis
warn.ord	If TRUE (the default) asterisks are displayed in the right margin of the lower panel to indicate nonordinal threshold locations for polytomous items.
warn.ord.colour	Nonordinal threshold locations for polytomous items are coloured with this colour to make them more visible. This is especially useful when there are many items so that the plot is quite dense. The default is "black", so that there is no distinction made.
irug	If TRUE (the default), all thresholds are plotted below the person distribution to indicate where the included items are most informative.
pp	If non-NULL, this contains the person.parameter data of the data object, avoiding the need to recalculate it.

Details

Item locations are displayed with bullets, threshold locations with circles.

Author(s)

Patrick Mair, Reinhold Hatzinger, patches from Julian Gilbey and Marco J. Maier

References

Bond, T.G., and Fox Ch.M. (2007) Applying the Rasch Model. Fundamental Measurement in the Human Sciences. 2nd Edition. Lawrence Erlbaum Associates.

Examples

```
res <- PCM(pcmdat)
plotPImap(res, sorted=TRUE)
```

---

plotPWmap *Pathway Map*

Description

A Bond-and-Fox Pathway Map displays the location of each item or each person against its infit t-statistic. Pathway maps are useful for identifying misfitting items or misfitting persons. Items or people should ideally have a infit t-statistic lying between about -2 and +2, and these values are marked.

Usage

```
plotPWmap(object, pmap = FALSE, imap=TRUE,
                  item.subset = "all", person.subset = "all",
                  mainitem = "Item Map", mainperson = "Person Map",
                  mainboth="Item/Person Map",
                  latdim = "Latent Dimension",
                  tlab = "Infit t statistic",
                  pp = NULL, cex.gen = 0.6, cex.pch=1,
                  person.pch = 1, item.pch = 16,
                  personCI = NULL, itemCI = NULL, horiz=FALSE)
```

Arguments

object	Object of class Rm or dRm
pmap	Plot a person map if TRUE; the default is FALSE.
imap	Plot an item map if TRUE (the default); do not plot if FALSE. At least one of pmap and imap must be TRUE.
item.subset	Subset of items to be plotted for an item map. Either a numeric vector indicating the item numbers or a character vector indicating the item names. If "all", all items are plotted. The number of items to be plotted must be > 1.
person.subset	Subset of persons to be plotted for a person map. Either a numeric vector indicating the person numbers or a character vector indicating the person names. If "all", all persons are plotted. The number of persons to be plotted must be > 1.
mainitem	Main title of an item plot.
mainperson	Main title of a person plot.
mainboth	Main title of a person/item joint plot.
latdim	Label of the y-axis, i.e., the latent dimension.
tlab	Label of the x-axis, i.e., the t-statistic dimension.

pp — If non-NULL, this contains the person.parameter data of the data object, avoiding the need to recalculate it.

cex.gen — cex as a graphical parameter specifies a numerical value giving the amount by which plotting text and symbols should be magnified relative to the default. Here cex.gen applies to all text labels. The default is 0.6.

cex.pch — applies to all plotting symbols. The default is 1.

person.pch, item.pch — Specifies the symbol used for plotting person data and item data respectively; the defaults are 1 and 16 respectively. See points for more information about pch values.

personCI, itemCI — Plotting confidence intervals for the the person abilities and item difficulties. If personCI=NULL (the default) no confidence intervals are drawn for person abilities. Otherwise, specifying personCI draws approximate confidence intervals for each person's ability. personCI must be specified as a list, and the optional elements of this list are gamma, the confidence level, col, colour, and lty, line type. If personCI is specified as an empty list, or not all of the list items are specified, the default values personCI=list(gamma=0.95,col="orange",lty="dotted") will be used.

The same goes for itemCI, except that the default settings are itemCI=list(gamma=0.95,col="red",lt

horiz — if TRUE, the plot is horizontal, i.e., the latent dimension is on the x-axis. The default is FALSE.

Details

This code uses vertical(horizontal) error bars rather than circles or boxes to indicate standard errors. It also offers the possibility of plotting item or person data on its own; this can considerably simplify the reading of the plots for large datasets.

Author(s)

Julian Gilbey

References

Bond T.G., Fox C.M. (2007) *Applying the Rasch Model: Fundamental Measurement in the Human Sciences* (2nd ed.) chapter 3, Lawrence Erlbaum Associates, Inc.

Linacre J.M., Wright B.D. (1994) Dichotomous Infit and Outfit Mean-Square Fit Statistics / Chi-Square Fit Statistics. *Rasch Measurement Transactions* **8:2** p. 350, https://www.rasch.org/rmt/rmt82a.htm

Linacre J.M. (2002) What do Infit and Outfit, Mean-square and Standardized mean? *Rasch Measurement Transactions* **16:2** p. 878, https://www.rasch.org/rmt/rmt162f.htm

Wright B.D., Masters G.N. (1990) Computation of OUTFIT and INFIT Statistics. *Rasch Measurement Transactions* **3:4** p. 84–85, https://www.rasch.org/rmt/rmt34e.htm

Examples

```
res <- PCM(pcmdat)
pparm <- person.parameter(res)
plotPWmap(res, pp = pparm)
plotPWmap(res, pp = pparm, pmap = TRUE)
```

`plotTR` *Plot Trend Effects for LLRA*

Description

Plots trend effects over time.

Usage

```
plotTR(object, ...)
```

Arguments

`object`	an object of class `"llra"`
`...`	Additional parameters to be passed to and from other methods

Details

The plot is a lattice plot with one panel. The effects for each items are plotted over the different time points.

Please note that all effects are to be interpreted relative to the baseline (i.e. t1).

Currently, this function only works for a full item x treatment x timepoints LLRA. Collapsed effects will not be displayed properly.

Warning:

Objects of class `"llra"` that contain estimates from a collapsed data matrix will not be displayed correctly.

Author(s)

Thomas Rusch

See Also

The plot method for treatment effects `"plotGR"`.

Examples

```
##Example 6 from Hatzinger & Rusch (2009)
groups <- c(rep("TG",30),rep("CG",30))
llra1 <- LLRA(llradat3,mpoints=2,groups=groups)
summary(llra1)
plotTR(llra1)

## Not run:
##An LLRA with 2 treatment groups and 1 baseline group, 5 items and 4
##time points. Item 1 is dichotomous, all others have 3, 4, 5, 6
##categories respectively.
ex2 <- LLRA(llraDat2[1:20],mpoints=4,groups=llraDat2[21])
plotTR(ex2)
## End(Not run)
```

`predict.ppar` *Predict methods*

Description

Returns data matrix based on model probabilites. So far implemented for dichotomous models only.

Usage

```
## S3 method for class 'ppar'
predict(object, cutpoint = "randomized", ...)
```

Arguments

`object`	Object of class `ppar` (from `person.parameter()`).
`cutpoint`	Either single integer value between 0 and 1 or `"randomized"` for randomized 0-1 assignment (see details)
`...`	Additional arguments ignored

Details

A randomized assignment implies that for each cell an additional random number is drawn. If the model probability is larger than this value, the person gets 1 on this particular item, if smaller, 0 is assigned. Alternatively, a numeric probability cutpoint can be assigned and the 0-1 scoring is carried out according to the same rule.

Value

Returns data matrix based on model probabilities

Author(s)

Patrick Mair, Reinhold Hatzinger

See Also

gofIRT.ppar

Examples

```
#Model-based data matrix for RSM
res <- RM(raschdat2)
pres <- person.parameter(res)
predict(pres)
```

---

print.eRm *Methods for extended Rasch models*

Description

Several methods for objects of class 'eRm'.

Usage

```
## S3 method for class 'eRm'
print(x, ...)
## S3 method for class 'eRm'
summary(object, ...)
## S3 method for class 'eRm'
coef(object, parm="beta", ...)
## S3 method for class 'eRm'
model.matrix(object, ...)
## S3 method for class 'eRm'
vcov(object, ...)
## S3 method for class 'eRm'
logLik(object, ...)
## S3 method for class 'eRm'
confint(object, parm = "beta", level = 0.95, ...)
```

Arguments

x	Object of class eRm.
object	Object of class eRm.
parm	Either "eta" or "beta".
level	Alpha-level.
...	Further arguments to be passed to or from other methods. They are ignored in this function.

Details

The `print` method displays the value of the log-likelihood, parameter estimates (basic parameters eta) and their standard errors. For RM, RSM, and PCM models, the etas are difficulty parameters, for the LLTM, LRSM, LPCM the sign of the parameters depend on the design matrix and are easiness effects by default. The `summary` method additionally gives the full set of item parameters beta as easiness parameters for all models.

Print methods are also available for the functions `logLik` and `confint` (see below).

Value

The methods below are extractor functions and return various quantities: `vcov` returns the variance-covariance matrix of the parameter estimates, `coef` a vector of estimates of the eta or beta basic parameters, `model.matrix` the design matrix, `logLik` an object with elements `loglik` and `df` containing the log-likelihood value and df. `confint` a matrix of confidence interval values for eta or beta.

Author(s)

Patrick Mair, Reinhold Hatzinger

Examples

```
res <- RM(raschdat1)
res
summary(res)
coef(res)
vcov(res)
model.matrix(res)
logLik(res)
```

`RaschSampler` *Rasch Sampler Package*

Description

The package implements an MCMC algorithm for sampling of binary matrices with fixed margins complying to the Rasch model. Its stationary distribution is uniform. The algorithm also allows for square matrices with fixed diagonal.

Parameter estimates in the Rasch model only depend on the marginal totals of the data matrix that is used for the estimation. From this it follows that, if the model is valid, all binary matrices with the same marginals as the observed one are equally likely. For any statistic of the data matrix, one can approximate the null distribution, i.e., the distribution if the Rasch model is valid, by taking a random sample from the collection of equally likely data matrices and constructing the observed distribution of the statistic. One can then simply determine the exceedence probability of the statistic in the observed sample, and thus construct a non-parametric test of the Rasch model. The main purpose of this package is the implementation of a methodology to build nonparametric tests for the

Rasch model.

In the context of social network theories, where the structure of binary asymmetric relations is studied, for example, person a esteems person b, which correponds to a 1 in cell (a, b) of the associated adjacency matrix. If one wants to study the distribution of a statistic defined on the adjacency matrix and conditional on the marginal totals, one has to exclude the diagonal cells from consideration, i.e., by keeping the diagonal cells fixed at an arbitrary value. The `RaschSampler` package has implemented an appropriate option, thus it can be also used for sampling random adjacency matrices with given marginal totals.

Details

Package:	RaschSampler
Type:	Package
Version:	0.8-6
Date:	2012-07-03
License:	GNU GPL 2, June 1991

The user has to supply a binary input matrix. After defining appropriate control parameters using `rsctrl` the sampling function `rsampler` may be called to obtain an object of class `RSmpl` which contains the generated random matrices in encoded form. After defining an appropriate function to operate on a binary matrix (e.g., calculate a statistic such as `phi.range`) the application of this function to the sampled matrices is performed using `rstats`. Prior to applying the user defined function, `rstats` decodes the matrices packed in the `RSmpl`-object.

The package also defines a utility function `rsextrobj` for extracting certains parts from the `RSmpl`-object resulting in an object of class `RSmplext`. Both types of objects can be saved and reloaded for later use.

Summary methods are available to print information on these objects, as well as on the control object `RSctr` which is obtained from using `rsctrl` containing the specification for the sampling routine.

Note

The current implementation allows for data matrices up to 4096 rows and 128 columns. This can be changed by setting `nmax` and `kmax` in `RaschSampler.f90` to values which are a power of 2. These values should also be changed in `rserror.R`.

For convenience, we reuse the Fortran code of package version 0.8-1 which cicumvents the compiler bug in Linux distributions of GCC 4.3. The following note from package version 0.8-3 is thus obsolete: In case of compilation errors (due to a bug in Linux distributions of GCC 4.3) please use `RaschSampler.f90` from package version 0.8-1 and change `nmax` and `kmax` accordingly (or use GCC 4.4).

Author(s)

Reinhold Hatzinger, Patrick Mair, Norman D. Verhelst

References

Verhelst, N. D. (2008) An Efficient MCMC Algorithm to Sample Binary Matrices with Fixed Marginals. Psychometrika, Volume 73, Number 4
Verhelst, N. D., Hatzinger, R., and Mair, P. (2007) The Rasch Sampler, Journal of Statistical Software, Vol. 20, Issue 4, Feb 2007

`RM` *Estimation of Rasch Models*

Description

This function computes the parameter estimates of a Rasch model for binary item responses by using CML estimation.

Usage

```
RM(X, W, se = TRUE, sum0 = TRUE, etaStart)
```

Arguments

`X`	Input 0/1 data matrix or data frame; rows represent individuals, columns represent items. Missing values are inserted as `NA`.
`W`	Design matrix for the Rasch model. If omitted, the function will compute W automatically.
`se`	If `TRUE`, the standard errors are computed.
`sum0`	If `TRUE`, the parameters are normed to sum-0 by specifying an appropriate `W`. If `FALSE`, the first parameter is restricted to 0.
`etaStart`	A vector of starting values for the eta parameters can be specified. If missing, the 0-vector is used.

Details

For estimating the item parameters the CML method is used. Available methods for RM-objects are:
`print`, `coef`, `model.matrix`, `vcov`, `summary`, `logLik`, `person.parameter`, `LRtest`, `Waldtest`, `plotICC`, `plotjointICC`.

Value

Returns an object of class dRm, Rm, eRm and contains the log-likelihood value, the parameter estimates and their standard errors.

loglik	Conditional log-likelihood.
iter	Number of iterations.
npar	Number of parameters.
convergence	See code output in nlm.
etapar	Estimated basic item difficulty parameters.
se.eta	Standard errors of the estimated basic item parameters.
betapar	Estimated item (easiness) parameters.
se.beta	Standard errors of item parameters.
hessian	Hessian matrix if se = TRUE.
W	Design matrix.
X	Data matrix.
X01	Dichotomized data matrix.
call	The matched call.

Author(s)

Patrick Mair, Reinhold Hatzinger

References

Fischer, G. H., and Molenaar, I. (1995). Rasch Models - Foundations, Recent Developements, and Applications. Springer.

Mair, P., and Hatzinger, R. (2007). Extended Rasch modeling: The **eRm** package for the application of IRT models in R. Journal of Statistical Software, 20(9), 1-20.

Mair, P., and Hatzinger, R. (2007). CML based estimation of extended Rasch models with the **eRm** package in R. Psychology Science, 49, 26-43.

See Also

RSM,PCM, LRtest, Waldtest

Examples

```
# Rasch model with beta.1 restricted to 0
res <- RM(raschdat1, sum0 = FALSE)
res
summary(res)
res$W                                    #generated design matrix

# Rasch model with sum-0 beta restriction; no standard errors computed
res <- RM(raschdat1, se = FALSE, sum0 = TRUE)
```

```
res
summary(res)
res$W                                    #generated design matrix

#Rasch model with missing values
res <- RM(raschdat2)
res
summary(res)
```

`rsampler` *Sampling Binary Matrices*

Description

The function implements an MCMC algorithm for sampling of binary matrices with fixed margins complying to the Rasch model. Its stationary distribution is uniform. The algorithm also allows for square matrices with fixed diagonal.

Usage

```
rsampler(inpmat, controls = rsctrl())
```

Arguments

`inpmat`	A binary (data) matrix with n rows and k columns.
`controls`	An object of class `RSctr`. If not specified, the default parameters as returned by function `rsctrl` are used.

Details

`rsampler` is a wrapper function for a Fortran routine to generate binary random matrices based on an input matrix. On output the generated binary matrices are integer encoded. For further processing of the generated matrices use the function `rstats`.

Value

A list of class `RSmpl` with components

`n`	number of rows of the input matrix
`k`	number of columns of the input matrix
`inpmat`	the input matrix
`tfixed`	`TRUE`, if diagonals of `inpmat` are fixed
`burn_in`	length of the burn in process
`n_eff`	number of generated matrices (effective matrices)
`step`	controls the number number of void matrices generated in the the burn in process and when effective matrices are generated (see note in `rsctrl`).

seed	starting value for the random number generator
n_tot	number of matrices in outvec, n_tot = n_eff + 1
outvec	vector of encoded random matrices
ier	error code

Note

An element of outvec is a four byte (or 32 bits) integer. The matrices to be output are stored bitwise (some bits are unused, since a integer is used for every row of a matrix). So the number of integers per row needed equals $(k + 31)/32$ (integer division), which is one to four in the present implementation since the number of columns and rows must not exceed 128 and 4096, respectively.

The summary method (summary.RSmpl) prints information on the content of the output object.

Author(s)

Reinhold Hatzinger, Norman Verhelst

References

Verhelst, N. D. (2008). An Efficient MCMC Algorithm to Sample Binary Matrices with Fixed Marginals. *Psychometrika, 73* (4)

See Also

rsctrl, rstats

Examples

```
data(xmpl)
ctr<-rsctrl(burn_in=10, n_eff=5, step=10, seed=0, tfixed=FALSE)
res<-rsampler(xmpl,ctr)
summary(res)
```

RSctr *Control Object*

Description

The object of class RSctr represents the control parameter specification for the sampling function rsampler.

Value

A legitimate RSctr object is a list with components

burn_in	the number of matrices to be sampled to come close to a stationary distribution.
n_eff	the number of effective matrices, i.e., the number of matrices to be generated by the sampling function rsampler.
step	controls the number number of void matrices generated in the the burn in process and when effective matrices are generated (see note in rsctrl).
seed	is the indicator for the seed of the random number generator. If the value of seed at equals zero, a seed is generated by the sampling function rsampler
tfixed	TRUE or FALSE. tfixed = TRUE has no effect if the input matrix is not quadratic, i.e., all matrix elements are considered free (unrestricted). If the input matrix is quadratic, and tfixed = TRUE, the main diagonal of the matrix is considered as fixed.

Generation

This object is returned from function rsctrl.

Methods

This class has a method for the generic summary function.

See Also

rsctrl

---

rsctrl *Controls for the Sampling Function*

Description

Various parameters that control aspects of the random generation of binary matrices.

Usage

```
rsctrl(burn_in = 100, n_eff = 100, step = 16, seed = 0, tfixed = FALSE)
```

Arguments

burn_in	the number of sampled matrices to come close to a stationary distribution. The default is burn_in = 100. (The actual number is 2 * burn_in * step.)
n_eff	the number of effective matrices, i.e., the number of matrices to be generated by the sampling function rsampler. n_eff must be positive and not larger than 8191 ($2^{13} - 1$). The default is n_eff = 100.

step	controls the number number of void matrices generated in the the burn in process and when effective matrices are generated (see note below). The default is step = 16.
seed	is the indicator for the seed of the random number generator. Its value must be in the range 0 and 2147483646 (2**31-2). If the value of seed equals zero, a seed is generated by the sampling function rsampler (dependent on the system's clock) and its value is returned in the output. If seed is not equal to zero, its value is used as the seed of the random number generator. In that case its value is unaltered at output. The default is seed = 0.
tfixed	logical, – specifies if in case of a quadratic input matrix the diagonal is considered fixed (see note below). The default is tfixed = FALSE.

Value

A list of class RSctr with components burn_in, n_eff, step, seed, tfixed.,

Note

If one of the components is incorrectly specified the error function rserror is called and some informations are printed. The ouput object will not be defined.

The specification of step controls the sampling algorithm as follows: If , e.g., burn_in = 10, n_eff = 5, and step = 2, then during the burn in period step * burn_in = 2 * 10 matrices are generated. After that, n_eff * step = 5 * 2 matrices are generated and every second matrix of these last ten is returned from link{rsampler}.

tfixed has no effect if the input matrix is not quadratic, i.e., all matrix elements are considered free (unrestricted). If the input matrix is quadratic, and tfixed = TRUE, the main diagonal of the matrix is considered as fixed. On return from link{rsampler} all diagonal elements of the generated matrices are set to zero. This specification applies, e.g., to analyzing square incidence matrices representing binary asymmetric relation in social network theory.

The summary method (summary.RSctr) prints the current definitions.

See Also

rsampler

Examples

```
ctr <- rsctrl(n_eff = 1, seed = 987654321)  # specify new controls
summary(ctr)

## Not run:
# incorrect specifications will lead to an error
ctr2 <- rsctrl(step = -3, n_eff = 10000)
## End(Not run)
```

rsextrmat *Extracting a Matrix*

Description

Convenience function to extract a matrix.

Usage

```
rsextrmat(RSobj, mat.no = 1)
```

Arguments

RSobj	object as obtained from using rsampler or rsextrobj
mat.no	number of the matrix to extract from the sample object.

Value

One of the matrices (either the original or a sampled matrix)

See Also

rsampler, rsextrobj,rstats,

Examples

```
ctr <- rsctrl(burn_in = 10, n_eff = 3, step=10, seed = 0, tfixed = FALSE)
mat <- matrix(sample(c(0,1), 50, replace = TRUE), nr = 10)
all_m <- rsampler(mat, ctr)
summary(all_m)

# extract the third sampled matrix (here the fourth)
third_m <- rsextrmat(all_m, 4)
head(third_m)
```

rsextrobj *Extracting Encoded Sample Matrices*

Description

Utility function to extract some of the generated matrices, still in encoded form.

Usage

```
rsextrobj(RSobj, start = 1, end = 8192)
```

Arguments

RSobj	object as obtained from using rsampler
start	number of the matrix to start with. When specifying 1 (the default value) the original input matrix is included in the output object.
end	last matrix to be extracted. If end is not specified, all matrices from RSobj are extracted (the maximal value is 8192, see rsctrl). If end is larger than the number of matrices stored in RSobj, end is set to the highest possible value (i.e., n_tot).

Value

A list of class RSmpl with components

n	number of rows of the input matrix
k	number of columns of the input matrix
inpmat	the input matrix
tfixed	TRUE, if diagonals of inpmat are fixed
burn_in	length of the burn in process
n_eff	number of generated matrices (effective matrices)
step	controls the number number of void matrices generated in the burn in process and when effective matrices are generated (see note in rsctrl).
seed	starting value for the random number generator
n_tot	number of matrices in outvec.
outvec	vector of encoded random matrices
ier	error code

Note

By default, all generated matrices plus the original matrix (in position 1) are contained in outvec, thus n_tot = n_eff + 1. If the original matrix is not in outvec then n_tot = n_eff.
For saving and loading objects of class RSobj see the example below.

For extracting a decoded (directly usable) matrix use rsextrmat.

See Also

rsampler, rsextrmat

Examples

```
ctr <- rsctrl(burn_in = 10, n_eff = 3, step=10, seed = 0, tfixed = FALSE)
mat <- matrix(sample(c(0,1), 50, replace = TRUE), nr = 10)
all_m <- rsampler(mat, ctr)
summary(all_m)

some_m <- rsextrobj(all_m, 1, 2)
summary(some_m)
```

```
## Not run:
save(some_m, file = "some.RSobj.RData")
rm(some_m)
ls()

load("some.RSobj.RData")
summary(some_m)
## End(Not run)
```

RSM *Estimation of rating scale models*

Description

This function computes the parameter estimates of a rating scale model for polytomous item responses by using CML estimation.

Usage

```
RSM(X, W, se = TRUE, sum0 = TRUE, etaStart)
```

Arguments

X	Input data matrix or data frame with item responses (starting from 0); rows represent individuals, columns represent items. Missing values are inserted as NA.
W	Design matrix for the RSM. If omitted, the function will compute W automatically.
se	If TRUE, the standard errors are computed.
sum0	If TRUE, the parameters are normed to sum-0 by specifying an appropriate W. If FALSE, the first parameter is restricted to 0.
etaStart	A vector of starting values for the eta parameters can be specified. If missing, the 0-vector is used.

Details

The design matrix approach transforms the RSM into a partial credit model and estimates the corresponding basic parameters by using CML. Available methods for RSM-objects are print, coef, model.matrix, vcov, summary, logLik, person.parameters, plotICC, LRtest.

Value

Returns an object of class 'Rm', 'eRm' and contains the log-likelihood value, the parameter estimates and their standard errors.

loglik	Conditional log-likelihood.

iter	Number of iterations.
npar	Number of parameters.
convergence	See code output in nlm.
etapar	Estimated basic item difficulty parameters (item and category parameters).
se.eta	Standard errors of the estimated basic item parameters.
betapar	Estimated item-category (easiness) parameters.
se.beta	Standard errors of item parameters.
hessian	Hessian matrix if se = TRUE.
W	Design matrix.
X	Data matrix.
X01	Dichotomized data matrix.
call	The matched call.

Author(s)

Patrick Mair, Reinhold Hatzinger

References

Fischer, G. H., and Molenaar, I. (1995). Rasch Models - Foundations, Recent Developements, and Applications. Springer.

Mair, P., and Hatzinger, R. (2007). Extended Rasch modeling: The **eRm** package for the application of IRT models in R. Journal of Statistical Software, 20(9), 1-20.

Mair, P., and Hatzinger, R. (2007). CML based estimation of extended Rasch models with the **eRm** package in R. Psychology Science, 49, 26-43.

See Also

RM,PCM,LRtest

Examples

```
##RSM with 10 subjects, 3 items
res <- RSM(rsmdat)
res
summary(res)                          #eta and beta parameters with CI
thresholds(res)                       #threshold parameters
```

RSmpl *Sample Objects*

Description

The objects of class RSmpl and RSmplext contain the original input matrix, the generated (encoded) random matrices, and some information about the sampling process.

Value

A list of class RSmpl or RSmplext with components

n	number of rows of the input matrix
k	number of columns of the input matrix
inpmat	the input matrix
tfixed	TRUE, if diagonals of inpmat are fixed
burn_in	length of the burn in process
n_eff	number of generated matrices (effective matrices)
step	controls the number number of void matrices generated in the the burn in process and when effective matrices are generated (see note in rsctrl).
seed	starting value for the random number generator
n_tot	number of matrices in outvec.
outvec	vector of encoded random matrices
ier	error code (see below)

Generation

These classes of objects are returned from rsampler and rsextrobj.

Methods

Both classes have methods for the generic summary function.

Note

By default, all generated matrices plus the original matrix (in position 1) are contained in outvec, thus n_tot = n_eff + 1. If the original matrix is not in outvec then n_tot = n_eff.

If ier is 0, no error was detected. Otherwise use the error function rserror(ier) to obtain some informations.

For saving and loading objects of class RSmpl or RSmplext see the example in rsextrobj.

See Also

rsampler, rsextrobj

rstats *Calculating Statistics for the Sampled Matrices*

Description

This function is used to calculate user defined statistics for the (original and) sampled matrices. A user defined function has to be provided.

Usage

```
rstats(RSobj, userfunc, ...)
```

Arguments

RSobj	object as obtained from using rsampler or rsextrobj
userfunc	a user defined function which performs operations on the (original and) sampled matrices. The first argument in the definition of the user function must be an object of type matrix.
...	further arguments, that are passed to the user function

Value

A list of objects as specified in the user supplied function

Note

The encoded matrices that are contained in the input object RSobj are decoded and passed to the user function in turn. If RSobj is not an object obtained from either rsampler or rsextrobj or no user function is specified an error message is printed. A simple user function, phi.range, is included in the RaschSampler package for demonstration purposes.

rstats can be used to obtain the 0/1 values for any of the sampled matrices (see second example below). Please note, that the output from the user function is stored in a list where the number of components corresponds to the number of matrices passed to the user function (see third example).

See Also

rsampler, rsextrobj

Examples

```
ctr <- rsctrl(burn_in = 10, n_eff = 5, step=10, seed = 12345678, tfixed = FALSE)
mat <- matrix(sample(c(0,1), 50, replace = TRUE), nr = 10)
rso <- rsampler(mat, ctr)
rso_st <- rstats(rso,phi.range)
unlist(rso_st)
```

```
# extract the third generated matrix
# (here, the first is the input matrix)
# and decode it into rsmat

rso2 <- rsextrobj(rso,4,4)
summary(rso2)
rsmat <- rstats(rso2, function(x) matrix(x, nr = rso2$n))
print(rsmat[[1]])

# extract only the first r rows of the third generated matrix

mat<-function(x, nr = nr, r = 3){
  m <- matrix(x, nr = nr)
  m[1:r,]
}
rsmat2 <- rstats(rso2, mat, nr=rso$n, r = 3)
print(rsmat2[[1]])

# apply a user function to the decoded object
print(phi.range(rsmat[[1]]))
```

`Separation Reliability`

Person Separation Reliability

Description

This function calculates the proportion of person variance that is not due to error. The concept of person separation reliability is very similar to reliability indices such as Cronbach's α.

Usage

```
SepRel(pobject)

## S3 method for class 'eRm_SepRel'
print(x, ...)

## S3 method for class 'eRm_SepRel'
summary(object, ...)
```

Arguments

`pobject`	Object of class `ppar` (see `person.parameter`).
`x`	Object of class `eRm_SepRel`.
`object`	Object of class `eRm_SepRel`.
`...`	Further arguments.

Details

Returns the person separation reliability $\frac{\text{SSD}-\text{MSE}}{\text{SSD}}$ where SSD is the squared standard deviation and MSE the mean squared error. Note that persons with raw scores of 0 or k are ignored in the computation.

Caveats:

Please note that the concept of *reliability* and associated problems are fundamentally different between IRT and CTT (Classical Test Theory). Separation reliability is more like a workaround to make the "change" from CTT to IRT easier for users by providing something "familiar." Hence, we recommend not to put too much emphasis on this particular measure and use it with caution.

Varying results in different programs:

If you compare the separation reliability obtained using **eRm** with values by other software, you will find that they are most likely not equal. This has a couple of reasons, one of the most important is the employed estimation method.

eRm uses a conditional maximum likelihood (CML) framework and handles missing values as separate groups during the estimation of item parameters. Person parameters are computed in a second step using unconditional or joint maximum likelihood (UML or JML) estimation with item parameters assumed to be known from the first step. Other programs might do JML to estimate item and person parameters at the same time, or employ marginal maximum likelihood MML to estimate item parameters, assuming a certain distribution for person parameters. In the latter case person parameters might be obtained by various methods like EAP, MAP, ... Even CML-based programs yield different values, for example, if they use Warm's weighted maximum likelihood estimation WLE to compute person parameters in the second step.

The bottom line is that, since there is not "definite" solution for this problem, you will end up with different values under different circumstances. This is another reason to take results and implications with a grain of salt.

Value

`SepRel` returns a list object of class `eRm_SepRel` containing:

`sep.rel`	the person separation reliability,
`SSD.PS`	the squared standard deviation (i.e., total person variability),
`MSE`	the mean square measurement error (i.e., model error variance).

Author(s)

Original code by Adrian Brügger (<Adrian.Bruegger@imu.unibe.ch>), adapted by Marco J. Maier

References

Wright, B.D., and Stone, M.H. (1999). *Measurement essentials.* Wide Range Inc., Wilmington. (https://www.rasch.org/measess/me-all.pdf 28Mb).

Examples

```
# Compute Separation Reliability for a Rasch Model:
pers <- person.parameter(RM(raschdat1))
res <- SepRel(pers)
res
summary(res)
```

`sim.2pl` *Simulation of 2-PL Data*

Description

This utility function returns a 0-1 matrix violating the parallel ICC assumption in the Rasch model.

Usage

```
sim.2pl(persons, items, discrim = 0.25, seed = NULL, cutpoint = "randomized")
```

Arguments

`persons`	Either a vector of person parameters or an integer indicating the number of persons (see details).
`items`	Either a vector of item parameters or an integer indicating the number of items (see details).
`discrim`	Standard deviation on the log scale.
`seed`	A seed for the random number generated can be set.
`cutpoint`	Either `"randomized"` for a randomized transformation of the model probability matrix into the model 0-1 matrix or an integer value between 0 and 1 (see details).

Details

If `persons` and/or `items` (using single integers) are specified to determine the number of subjects or items, the corresponding parameter vector is drawn from N(0,1). The `cutpoint` argument refers to the transformation of the theoretical probabilities into a 0-1 data matrix. A randomized assingment implies that for each cell an additional random number is drawn. If the model probability is larger than this value, the person gets 1 on this particular item, if smaller, 0 is assigned. Alternatively, a numeric probability cutpoint can be assigned and the 0-1 scoring is carried out according to the same rule.

The `discrim` argument can be specified either as a vector of length `items` defining the item discrimination parameters in the 2-PL (e.g., `c(1,1,0.5,1,1.5)`), or as a single value. In that case, the discrimination parameters are drawn from a lognormal distribution with `meanlog = 0`, where the specified value in `discrim` refers to the standard deviation on the log-scale. The larger the values, the stronger the degree of Rasch violation. Reasonable values are up to 0.5. If 0, the data are Rasch homogeneous.

References

Su\'arez-Falc\'on, J. C., & Glas, C. A. W. (2003). Evaluation of global testing procedures for item fit to the Rasch model. British Journal of Mathematical and Statistical Society, 56, 127-143.

See Also

sim.rasch, sim.locdep, sim.xdim

Examples

```
#simulating 2-PL data
#500 persons, 10 items, sdlog = 0.30, randomized cutpoint
X <- sim.2pl(500, 10, discrim = 0.30)

#item and discrimination parameters from uniform distribution,
#cutpoint fixed
dpar <- runif(50, 0, 2)
ipar <- runif(50, -1.5, 1.5)
X <- sim.2pl(500, ipar, dpar, cutpoint = 0.5)
```

---

sim.locdep *Simulation locally dependent items*

Description

This utility function returns a 0-1 matrix violating the local independence assumption.

Usage

```
sim.locdep(persons, items, it.cor = 0.25, seed = NULL,
   cutpoint = "randomized")
```

Arguments

persons	Either a vector of person parameters or an integer indicating the number of persons (see details).
items	Either a vector of item parameters or an integer indicating the number of items (see details).
it.cor	Either a single correlation value between 0 and 1 or a positive semi-definite VC matrix.
seed	A seed for the random number generated can be set.
cutpoint	Either "randomized" for a randomized tranformation of the model probability matrix into the model 0-1 matrix or an integer value between 0 and 1 (see details).

Details

If `persons` or `items` is an integer value, the corresponding parameter vector is drawn from N(0,1). The `cutpoint` argument refers to the transformation of the theoretical probabilities into a 0-1 data matrix. A randomized assingment implies that for each cell an additional random number is drawn. If the model probability is larger than this value, the person gets 1 on this particular item, if smaller, 0 is assigned. Alternatively, a numeric probability cutpoint can be assigned and the 0-1 scoring is carried out according to the same rule.

The argument `it.cor` reflects the pair-wise inter-item correlation. If this should be constant across the items, a single value between 0 (i.e. Rasch model) and 1 (strong violation) can be specified. Alternatively, a symmetric VC-matrix of dimension number of items can be defined.

References

Jannarone, R. J. (1986). Conjunctive item response theory kernels. Psychometrika, 51, 357-373.

Su\'arez-Falc\'on, J. C., & Glas, C. A. W. (2003). Evaluation of global testing procedures for item fit to the Rasch model. British Journal of Mathematical and Statistical Society, 56, 127-143.

See Also

`sim.rasch`, `sim.2pl`, `sim.xdim`

Examples

```
#simulating locally-dependent data
#500 persons, 10 items, inter-item correlation of 0.5
X <- sim.locdep(500, 10, it.cor = 0.5)

#500 persons, 4 items, correlation matrix specified
sigma <- matrix(c(1,0.2,0.2,0.3,0.2,1,0.4,0.1,0.2,0.4,1,0.8,0.3,0.1,0.8,1),
   ncol = 4)
X <- sim.locdep(500, 4, it.cor = sigma)
```

---

sim.rasch *Simulation of Rasch homogeneous data*

Description

This utility function returns a 0-1 matrix which fits the Rasch model.

Usage

```
sim.rasch(persons, items, seed = NULL, cutpoint = "randomized")
```

Arguments

persons	Either a vector of person parameters or an integer indicating the number of persons (see details)
items	Either a vector of item parameters or an integer indicating the number of items (see details)
seed	A seed for the random number generated can be set.
cutpoint	Either "randomized" for a randomized tranformation of the model probability matrix into the model 0-1 matrix or an integer value between 0 and 1 (see details)

Details

If persons or items is an integer value, the corresponding parameter vector is drawn from N(0,1). The cutpoint argument refers to the transformation of the theoretical probabilities into a 0-1 data matrix. A randomized assingment implies that for each cell an additional random number is drawn. If the model probability is larger than this value, the person gets 1 on this particular item, if smaller, 0 is assigned. Alternatively, a numeric probability cutpoint can be assigned and the 0-1 scoring is carried out according to the same rule.

References

Su\'arez-Falc\'on, J. C., & Glas, C. A. W. (2003). Evaluation of global testing procedures for item fit to the Rasch model. British Journal of Mathematical and Statistical Society, 56, 127-143.

See Also

sim.xdim, sim.locdep, sim.2pl

Examples

```
#simulating Rasch homogenous data
#100 persons, 10 items, parameter drawn from N(0,1)
X <- sim.rasch(100, 10)

#person parameters drawn from uniform distribution, fixed cutpoint
ppar <- runif(100,-2,2)
X <- sim.rasch(ppar, 10, cutpoint = 0.5)
```

---

sim.xdim *Simulation of multidimensional binary data*

Description

This utility function simulates a 0-1 matrix violating the unidimensionality assumption in the Rasch model.

Usage

```
sim.xdim(persons, items, Sigma, weightmat, seed = NULL,
   cutpoint = "randomized")
```

Arguments

persons	Either a matrix (each column corresponds to a dimension) of person parameters or an integer indicating the number of persons (see details).
items	Either a vector of item parameters or an integer indicating the number of items (see details).
Sigma	A positive-definite symmetric matrix specifying the covariance matrix of the variables.
weightmat	Matrix for item-weights for each dimension (columns).
seed	A seed for the random number generated can be set.
cutpoint	Either "randomized" for a randomized tranformation of the model probability matrix into the model 0-1 matrix or an integer value between 0 and 1 (see details).

Details

If persons is specified as matrix, Sigma is ignored. If items is an integer value, the corresponding parameter vector is drawn from N(0,1). The cutpoint argument refers to the transformation of the theoretical probabilities into a 0-1 data matrix. A randomized assingment implies that for each cell an additional random number is drawn. If the model probability is larger than this value, the person gets 1 on this particular item, if smaller, 0 is assigned. Alternatively, a numeric probability cutpoint can be assigned and the 0-1 scoring is carried out according to the same rule.

If weightmat is not specified, a random indicator matrix is generated where each item is a measurement of only one dimension. For instance, the first row for a 3D-model could be (0,1,0) which means that the first item measures the second dimension only. This corresponds to the between-item multidimensional model presented by Adams et al. (1997).

Sigma reflects the VC-structure for the person parameters drawn from a multivariate standard normal distribution. Thus, the diagonal elements are typically 1 and the lower the covariances in the off-diagonal, the stronger the model violation.

References

Adams, R. J., Wilson, M., & Wang, W. C. (1997). The multidimensional random coefficients multinomial logit model. Applied Psychological Measurement, 21, 1-23.

Glas, C. A. W. (1992). A Rasch model with a multivariate distribution of ability. In M. Wilson (Ed.), Objective Measurement: Foundations, Recent Developments, and Applications (pp. 236-258). Norwood, NJ: Ablex.

See Also

sim.rasch, sim.locdep, sim.2pl

Examples

```
# 500 persons, 10 items, 3 dimensions, random weights.
Sigma <- matrix(c(1, 0.01, 0.01, 0.01, 1, 0.01, 0.01, 0.01, 1), 3)
X <- sim.xdim(500, 10, Sigma)

#500 persons, 10 items, 2 dimensions, weights fixed to 0.5
itemvec <- runif(10, -2, 2)
Sigma <- matrix(c(1, 0.05, 0.05, 1), 2)
weights <- matrix(0.5, ncol = 2, nrow = 10)
X <- sim.xdim(500, itemvec, Sigma, weightmat = weights)
```

---

stepwiseIt *Stepwise item elimination*

Description

This function eliminates items stepwise according to one of the following criteria: itemfit, Wald test, Andersen's LR-test

Usage

```
## S3 method for class 'eRm'
stepwiseIt(object, criterion = list("itemfit"), alpha = 0.05,
           verbose = TRUE, maxstep = NA)
```

Arguments

`object`	Object of class `eRm`.
`criterion`	List with either `"itemfit"`, `"Waldtest"` or `"LRtest"` as first element. Optionally, for the Waldtest and LRtest a second element containing the split criterion can be specified (see details).
`alpha`	Significance level.
`verbose`	If `TRUE` intermediate results are printed out.
`maxstep`	Maximum number of elimination steps. If `NA` the procedure stops when the itemset is Rasch homogeneous.

Details

If `criterion = list("itemfit")` the elimination stops when none of the p-values in itemfit is significant. Within each step the item with the largest chi-squared itemfit value is excluded.

If `criterion = list("Waldtest")` the elimination stops when none of the p-values resulting from the Wald test is significant. Within each step the item with the largest z-value in Wald test is excluded.

If `criterion = list("LRtest")` the elimination stops when Andersen's LR-test is not significant. Within each step the item with the largest z-value in Wald test is excluded.

Value

The function returns an object of class `step` containing:

`X`	Reduced data matrix (bad items eliminated)
`fit`	Object of class `eRm` with the final item parameter elimination
`it.elim`	Vector contaning the names of the eliminated items
`res.wald`	Elimination results for Wald test criterion
`res.itemfit`	Elimination results for itemfit criterion
`res.LR`	Elimination results for LR-test criterion
`nsteps`	Number of elimination steps

See Also

`LRtest.Rm`, `Waldtest.Rm`, `itemfit.ppar`

Examples

```
## 2pl-data, 100 persons, 10 items
set.seed(123)
X <- sim.2pl(500, 10, 0.4)
res <- RM(X)

## elimination according to itemfit
stepwiseIt(res, criterion = list("itemfit"))

## Wald test based on mean splitting
stepwiseIt(res, criterion = list("Waldtest","mean"))

## Andersen LR-test based on random split
set.seed(123)
groupvec <- sample(1:3, 500, replace = TRUE)
stepwiseIt(res, criterion = list("LRtest",groupvec))
```

`summary.llra` *Summarizing Linear Logistic Models with Relaxed Assumptions (LLRA)*

Description

`summary` method for class `"llra"`

Usage

```
## S3 method for class 'llra'
summary(object, level, ...)

## S3 method for class 'summary.llra'
print(x, ...)
```

Arguments

object	an object of class "llra", typically result of a call to LLRA.
x	an object of class "summary.llra", usually, a result of a call to summary.llra.
level	The level of confidence for the confidence intervals. Default is 0.95.
...	further arguments passed to or from other methods.

Details

Objects of class "summary.llra" contain all parameters of interest plus the confidence intervals.

print.summary.llra rounds the values to 3 digits and displays them nicely.

Value

The function summary.lllra computes and returns a list of summary statistics of the fitted LLRA given in object, reusing the components (list elements) call, etapar, iter, loglik, model, npar and se.etapar from its argument, plus

ci	The upper and lower confidence interval borders.

Author(s)

Thomas Rusch

See Also

The model fitting function LLRA.

Examples

```
##Example 6 from Hatzinger & Rusch (2009)
groups <- c(rep("TG",30),rep("CG",30))
llra1 <- LLRA(llradat3,mpoints=2,groups=groups)
summary(llra1)

## Not run:
##An LLRA with 2 treatment groups and 1 baseline group, 5 items and 4
##time points. Item 1 is dichotomous, all others have 3, 4, 5, 6
##categories respectively.
ex2 <- LLRA(llraDat2[1:20],mpoints=4,llraDat2[21])
sumEx2 <- summary(ex2, level=0.95)
```

```
#print a summary
sumEx2

#get confidence intervals
sumEx2$ci
## End(Not run)
```

---

summary.RSctr *Summary Method for Control Objects*

Description

Prints the current definitions for the sampling function.

Usage

```
## S3 method for class 'RSctr'
summary(object, ...)
```

Arguments

object	object of class RSctr as obtained from rsctrl
...	potential further arguments (ignored)

See Also

rsctrl

Examples

```
ctr <- rsctrl(n_eff = 1, seed = 123123123)  # specify controls
summary(ctr)
```

---

summary.RSmpl *Summary Methods for Sample Objects*

Description

Prints a summary list for sample objects of class RSmpl and RSmplext.

Usage

```
## S3 method for class 'RSmpl'
summary(object, ...)
## S3 method for class 'RSmplext'
summary(object, ...)
```

Arguments

object	object as obtained from rsampler or rsextrobj
...	potential further arguments (ignored)

Details

Describes the status of an sample object.

See Also

rsampler, rsextrobj

Examples

```
ctr <- rsctrl(burn_in = 10, n_eff = 3, step=10, seed = 0, tfixed = FALSE)
mat <- matrix(sample(c(0,1), 50, replace = TRUE), nr = 10)
all_m <- rsampler(mat, ctr)
summary(all_m)

some_m <- rsextrobj(all_m, 1, 2)
summary(some_m)
```

---

test_info *Calculate Test Information For* eRm *objects*

Description

Calculates the information of a test or a scale as the sum of Samejima's (1969) information for all items.

Usage

```
test_info(ermobject, theta=seq(-5,5,0.01))
```

Arguments

ermobject	An object of class 'eRm'.
theta	Supporting or sampling points on the latent trait.

Details

The function test_info calculates the test or scale information of the whole set of items in the 'eRm' object.

Value

Returns the vector of test information for all values of theta.

Author(s)

Thomas Rusch

References

Samejima, F. (1969) Estimation of latent ability using a response pattern of graded scores. *Psychometric Monographs*, **17**.

See Also

The function to calculate the item information, `item_info` and the plot function `plotINFO`.

Examples

```
res <- PCM(pcmdat)
tinfo <- test_info(res)
plotINFO(res, type="test")
```

---

thresholds *Computation of item-category treshold parameters.*

Description

This function transforms the beta parameters into threshold parameters. These can be interpreted by means of log-odds as visualized in ICC plots.

Usage

```
## S3 method for class 'eRm'
thresholds(object)
## S3 method for class 'threshold'
print(x, ...)
## S3 method for class 'threshold'
summary(object, ...)
## S3 method for class 'threshold'
confint(object, parm, level = 0.95, ...)
```

Arguments

Arguments for `thresholds`:

`object`	Object of class `eRm`.

Arguments for `print`, `summary`, and `confint` methods:

`x`	Object of class `threshold`.
`parm`	Parameter specification (ignored).
`level`	Alpha-level.
`...`	Further arguments to be passed to methods. They are ignored.

Details

For dichotomous models (i.e., RM and LLTM) threshold parameters are not computed. The `print` method returns a location parameter for each item which is the mean of the corresponding threshold parameters. For LPCM and LRSM the thresholds are computed for each design matrix block (i.e., measurement point/group) separately (PCM and RSM have only 1 block).

Value

The function `thresholds` returns an object of class `threshold` containing:

`threshpar`	Vector with threshold parameters.
`se.thresh`	Vector with standard errors.
`threshtable`	Data frame with location and threshold parameters.

References

Andrich, D. (1978). Application of a psychometric rating model to ordered categories which are scored with successive integers. Applied Psychological Measurement, 2, 581-594.

See Also

`plotICC.Rm`

Examples

```
#Threshold parameterization for a rating scale model
res <- RSM(rsmdat)
th.res <- thresholds(res)
th.res
confint(th.res)
summary(th.res)

#Threshold parameters for a PCM with ICC plot
res <- PCM(pcmdat)
th.res <- thresholds(res)
th.res
plotICC(res)

#Threshold parameters for a LPCM:
#Block 1: t1, g1; Block 2: t1, g2; ...; Block 6: t2,g3
G <- c(rep(1,7),rep(2,7),rep(3,6)) # group vector for 3 groups
res <- LPCM(lpcmdat, mpoints = 2, groupvec = G)
th.res <- thresholds(res)
th.res
```

`Waldtest` *Item-Specific Wald Test*

Description

Performs a Wald test on item-level by splitting subjects into subgroups.

Usage

```
## S3 method for class 'Rm'
Waldtest(object, splitcr = "median")
## S3 method for class 'wald'
print(x,...)
```

Arguments

`object`	Object of class `RM`.
`splitcr`	Split criterion for subject raw score splitting. `median` uses the median as split criterion, `mean` performs a mean-split. Optionally `splitcr` can also be a dichotomous vector which assigns each person to a certain subgroup (e.g., following an external criterion). This vector can be numeric, character or a factor.
`x`	Object of class `wald`.
`...`	Further arguments passed to or from other methods. They are ignored in this function.

Details

Items are eliminated if they not have the same number of categories in each subgroup. To avoid this problem, for RSM and PCM it is considered to use a random or another user-defined split. If the data set contains missing values and `mean` or `median` is specified as splitcriterion, means or medians are calculated for each missing value subgroup and consequently used for raw score splitting.

Value

Returns an object of class `wald` containing:

`coef.table`	Data frame with test statistics, z- and p-values.
`betapar1`	Beta parameters for first subgroup
`se.beta1`	Standard errors for first subgroup
`betapar2`	Beta parameters for second subgroup
`se.beta2`	Standard errors for second subgroup
`se.beta2`	Standard errors for second subgroup
`spl.gr`	Names and levels for `splitcr`.
`call`	The matched call.

Author(s)

Patrick Mair, Reinhold Hatzinger

References

Fischer, G. H., and Molenaar, I. (1995). Rasch Models - Foundations, Recent Developements, and Applications. Springer.

Fischer, G. H., and Scheiblechner, H. (1970). Algorithmen und Programme fuer das probabilistische Testmodell von Rasch [Algorithms and programs for Rasch's probabilistic test model]. Psychologische Beitraege, 12, 23-51.

See Also

LRtest, MLoef

Examples

```
#Wald test for Rasch model with user-defined subject split
res <- RM(raschdat2)
splitvec <- sample(1:2,25,replace=TRUE)
Waldtest(res, splitcr = splitvec)
```

---

xmpl *Example Data*

Description

Ficitious data sets - matrices with binary responses

Usage

```
data(xmpl)
```

Format

The format of `xmpl` is:
300 rows (referring to subjects)
30 columns (referring to items)

The format of `xmplbig` is:
4096 rows (referring to subjects)
128 columns (referring to items)
`xmplbig` has the maximum dimensions that the RaschSampler package can handle currently.

Examples

```
data(xmpl)
print(head(xmpl))
```

Index