Package 'TAM'

November 22, 2014

Type Package

Title Test Analysis Modules

Version 1.2

Date 2014-11-22

Author Thomas Kiefer [aut], Alexander Robitzsch [aut, cre], Margaret Wu [aut]

Maintainer Alexander Robitzsch <a.robitzsch@bifie.at>

Description The package includes marginal and joint maximum likelihood
estimation of uni- and multidimensional item response models
(Rasch, 2PL, 3PL, Generalized Partial Credit, Multi Facets,Nominal Item Response, Structured Latent Class Analysis,Mixture Distribution IRT Models, Located Latent Class Models,Cognitive Diagnostic Models), fit statistics and person parameter estimation.

Depends R (>= 2.15.1), CDM (>= 4.0), MASS

Imports Rcpp, mirt (>= 1.6), lavaan (>= 0.5-17), tensor, sfsmisc,GPArotation, psych, lattice, mvtnorm

Suggests sirt (>= 1.0)

LinkingTo Rcpp, RcppArmadillo

License GPL (>= 2)

URL http://www.edmeasurementsurveys.com/TAM/Tutorials/

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-11-22 19:10:30

R topics documented:

TAM-package . . . 2
anova-logLik . . . 3
data.cqc . . . 5
data.ctest . . . 9
data.examples . . . 10
data.fims.Aus.Jpn.scored . . . 12
data.gpcm . . . 14
data.mc . . . 14
data.numeracy . . . 15
data.timssAusTwn . . . 17
designMatrices . . . 19
IRT.factor.scores . . . 22
IRT.irfprob . . . 23
IRT.likelihood . . . 23
lavaanify.IRT . . . 24
plot.tam . . . 27
plotDevianceTAM . . . 28
sim.mfr . . . 29
sim.rasch . . . 31
tam.ctt . . . 32
tam.fa . . . 35
tam.fit . . . 37
tam.jml . . . 41
tam.mml . . . 46
tam.mml.3pl . . . 72
tam.modelfit . . . 85
tam.pv . . . 88
tam.se . . . 90
tam.threshold . . . 92
tam.wle . . . 93

Index **97**

`TAM-package` *Test Analysis Modules*

Description

The package includes marginal and joint maximum likelihood estimation of uni- and multidimensional item response models (Rasch, 2PL, 3PL, Generalized Partial Credit, Multi Facets, Nominal Item Response, Structured Latent Class Analysis, Mixture Distribution IRT Models, Located Latent Class Models, Cognitive Diagnostic Models),fit statistics and person parameter estimation.

Details

Package:	**TAM**
Type:	Package
Version:	1.2
Publication Year:	2014
License:	GPL (>=2.0)
URL:	http://www.edmeasurementsurveys.com/TAM/Tutorials/

See http://www.edmeasurementsurveys.com/TAM/Tutorials/ for tutorials of the **TAM** package.

Author(s)

Thomas Kiefer [aut], Alexander Robitzsch [aut, cre], Margaret Wu [aut]

Maintainer: Alexander Robitzsch <a.robitzsch@bifie.at>

References

Adams, R. J., Wilson, M., & Wu, M. (1997). Multilevel item response models: An approach to errors in variables regression. *Journal of Educational and Behavioral Statistics*, **22**, 47-76.

Adams, R. J., & Wu, M. L. (2007). The mixed-coefficients multinomial logit model. A generalized form of the Rasch model. In M. von Davier & C. H. Carstensen (Eds.): *Multivariate and mixture distribution Rasch models: Extensions and applications* (pp. 55-76). New York: Springer.

`anova-logLik` *Likelihood Ratio Test for Model Comparisons and Log-Likelihood Value*

Description

The anova function compares two models estimated of class `tam` `tam.mml` or `tam.mml.3pl` using a likelihood ratio test. The `logLik` function extracts the value of the log-Likelihood.

The function can be applied for values of `tam.mml`, `tam.mml.2pl`, `tam.mml.mfr`, `tam.fa` and `tam.mml.3pl`.

Usage

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

## S3 method for class 'tam.mml'
anova(object, ...)

## S3 method for class 'tam.mml.3pl'
```

```
anova(object, ...)

## S3 method for class 'tam'
logLik(object, ...)

## S3 method for class 'tam.mml'
logLik(object, ...)

## S3 method for class 'tam.mml.3pl'
logLik(object, ...)
```

Arguments

object	Object of class tam, tam.mml or tam.mml.3pl. Note that for anova two objects (fitted models) must be provided.
...	Further arguments to be passed

Value

A data frame containing the likelihood ratio test statistic and information criteria.

Examples

```
#############################################################################
# EXAMPLE 1: Dichotomous data sim.rasch - 1PL vs. 2PL model
#############################################################################

data(sim.rasch)
# 1PL estimation
mod1 <- tam.mml(resp=sim.rasch)
logLik(mod1)
# 2PL estimation
mod2 <- tam.mml.2pl(resp=sim.rasch , irtmodel="2PL")
logLik(mod2)
# Model comparison
anova( mod1 , mod2 )
  ## > anova( mod1 , mod2 )
  ##     Model   loglike Deviance Npars      AIC      BIC    Chisq df       p
  ## 1 Model 1 -42077.88 84155.77    41 84237.77 84467.40 54.05078 39 0.05508
  ## 2 Model 2 -42050.86 84101.72    80 84261.72 84709.79       NA NA      NA

## Not run:
#############################################################################
# EXAMPLE 2: Dataset reading (sirt package): 1- vs. 2-dimensional model
#############################################################################

data(data.read,package="sirt")

# 1-dimensional model
mod1 <- tam.mml.2pl(resp= data.read )
```

```
# 2-dimensional model
mod2 <- tam.fa(resp= data.read , irtmodel="efa" , nfactors=2 ,
        control=list(maxiter=150) )
# Model comparison
anova( mod1 , mod2 )
  ##       Model   loglike Deviance Npars      AIC      BIC    Chisq df  p
  ##   1 Model 1 -1954.888 3909.777    24 3957.777 4048.809 76.66491 11  0
  ##   2 Model 2 -1916.556 3833.112    35 3903.112 4035.867       NA NA NA

## End(Not run)
```

---

data.cqc *More Datasets and Examples (Similar to ConQuest Examples)*

Description

Datasets and examples similar to the ones in the ConQuest manual (Wu, Adams, Wilson, & Haldane, 2007).

Usage

```
data(data.cqc01)
data(data.cqc02)
data(data.cqc03)
data(data.cqc04)
data(data.cqc05)
```

Format

- data.cqc01 contains 512 persons on 12 dichotomous items of following format

```
'data.frame':   512 obs. of  12 variables:
$ BSMMA01: int  1 1 0 1 1 1 1 0 0 ...
$ BSMMA02: int  1 0 1 1 0 1 1 1 0 0 ...
$ BSMMA03: int  1 1 0 1 1 1 1 1 1 0 ...
[...]
$ BSMSA12: int  0 0 0 0 1 0 1 1 0 0 ...
```

- data.cqc02 contains 431 persons on 8 polytomous variables of following format

```
'data.frame':   431 obs. of  8 variables:
$ It1: int  1 1 2 0 2 1 2 2 2 1 ...
$ It2: int  3 0 1 2 2 3 2 2 1 1 ...
$ It3: int  1 1 1 0 1 1 0 0 1 0 ...
[...]
$ It8: int  3 1 0 0 3 1 3 0 3 0 ...
```

- `data.cqc03` contains 11200 observations for 5600 persons, 16 raters and 2 items (`crit1` and `crit2`)

```
'data.frame':   11200 obs. of  4 variables:
$ pid  : num  10001 10001 10002 10002 10003 ...
$ rater: chr  "R11" "R12" "R13" "R14" ...
$ crit1: int  2 2 2 1 3 2 2 1 1 1 ...
$ crit2: int  3 3 2 1 2 2 2 2 2 1 ...
```

- `data.cqc04` contains 1452 observations for 363 persons, 4 raters, 4 topics and 5 items (`spe`, `coh`, `str`, `gra`, `con`)

```
'data.frame':   1452 obs. of  8 variables:
$ pid  : num  10010 10010 10010 10010 10016 ...
$ rater: chr  "BE" "CO" "BE" "CO" ...
$ topic: chr  "Spor" "Spor" "Spor" "Spor" ...
$ spe  : int  2 0 2 1 3 3 3 3 3 2 ...
$ coh  : int  1 1 2 0 3 3 3 3 3 3 ...
$ str  : int  0 1 3 0 3 2 3 2 3 0 ...
$ gra  : int  0 0 2 0 3 3 3 3 2 1 ...
$ con  : int  0 0 0 0 3 1 2 2 3 0 ...
```

- `data.cqc05` contains 1500 persons, 3 covariates and 157 items.

```
'data.frame':   1500 obs. of  160 variables:
$ gender: int  1 0 1 0 0 0 0 1 0 1 ...
$ level : int  0 1 1 0 0 0 1 0 1 1 ...
$ gbyl  : int  0 0 1 0 0 0 0 0 0 1 ...
$ A001  : num  0 0 0 1 0 1 1 1 0 1 ...
$ A002  : num  1 1 0 1 1 1 1 1 1 0 ...
$ A003  : num  0 0 0 0 1 1 1 0 0 1 ...
[...]
```

References

Wu, M. L., Adams, R. J., Wilson, M. R. & Haldane, S. (2007). *ACER ConQuest Version 2.0.* Mulgrave. https://shop.acer.edu.au/acer-shop/group/CON3

Examples

```
## Not run:
#############################################################################
# EXAMPLE 01: Rasch model data.cqc01
#############################################################################
data(data.cqc01)
dat <- data.cqc01

#*** Model 01: Estimate Rasch model
mod01 <- tam.mml( dat )
summary( mod01 )
```

```
#############################################################################
# EXAMPLE 02: Partial credit model and rating scale model data.cqc02
#############################################################################
data(data.cqc02)
dat <- data.cqc02

# Model 02a: Partial credit model in ConQuest parametrization 'item+item*step'
mod02a <- tam.mml( dat , irtmodel="PCM2" )
summary( mod02a  ) # summary
tam.fit( mod02a ) # item fit

# Model 02b: Rating scale model
mod02b <- tam.mml( dat , irtmodel="RSM" )
summary( mod02b )

#############################################################################
# EXAMPLE 03: Faceted Rasch model for rating data data.cqc03
#############################################################################
data(data.cqc03)
# select items
resp <- data.cqc03[ , c("crit1","crit2") ]

# Model 03a: 'item+step+rater'
mod03a <- tam.mml.mfr( resp , facets=data.cqc03[,"rater",drop=FALSE] ,
            formulaA = ~ item+step+rater , pid = data.cqc03$pid )
summary( mod03a )

# Model 03b: 'item:step+rater'
mod03b <- tam.mml.mfr( resp , facets=data.cqc03[,"rater",drop=FALSE] ,
            formulaA = ~ item + item:step+rater , pid = data.cqc03$pid )
summary( mod03b )

# Model 03c: 'step+rater' for first item 'crit1'
# Restructuring the data is necessary.
# Define raters as items in the new dataset 'dat1'.
persons <- unique( data.cqc03$pid )
raters <- unique( data.cqc03$rater )
dat1 <- matrix( NA , nrow= length(persons) , ncol=length(raters) + 1 )
dat1 <- as.data.frame(dat1)
colnames(dat1) <- c("pid" , raters )
dat1$pid <- persons
for (rr in raters){
    dat1.rr <- data.cqc03[ data.cqc03$rater == rr , ]
    dat1[ match(dat1.rr$pid , persons) ,rr] <- dat1.rr$crit1
        }
  ##   > head(dat1)
  ##       pid R11 R12 R13 R14 R15 R16 R17 R18 R19 R20 R21 R22 R23 R24 R25 R26
  ##   1 10001   2   2  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
  ##   2 10002  NA  NA   2   1  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
  ##   3 10003  NA  NA   3   2  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
  ##   4 10004  NA  NA   2   1  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
  ##   5 10005  NA  NA   1   1  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
  ##   6 10006  NA  NA   1   1  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA  NA
```

```
# estimate model 03c
mod03c <- tam.mml( dat1[,-1] , pid = dat1$pid )
summary( mod03c )

#############################################################################
# EXAMPLE 04: Faceted Rasch model for rating data data.cqc04
#############################################################################
data(data.cqc04)
resp <- data.cqc04[,4:8]
facets <- data.cqc04[ , c("rater" , "topic") ]

# Model 04a: 'item*step+rater+topic'
formulaA <- ~ item*step + rater + topic
mod04a <- tam.mml.mfr( resp , facets= facets ,
            formulaA = formulaA , pid = data.cqc04$pid )
summary( mod04a )

# Model 04b: 'item*step+rater+topic+item*rater+item*topic'
formulaA <- ~ item*step + rater + topic + item*rater + item*topic
mod04b <- tam.mml.mfr( resp , facets= facets ,
            formulaA = formulaA , pid = data.cqc04$pid )
summary( mod04b )

# Model 04c: 'item*step' with fixing rater and topic parameters to zero
formulaA <- ~ item*step + rater + topic
mod04c0 <- tam.mml.mfr( resp , facets= facets ,
            formulaA = formulaA , pid = data.cqc04$pid , control=list(maxiter=4) )
summary( mod04c0 )
# fix rater and topic parameter to zero
xsi.est <- mod04c0$xsi
xsi.fixed <- cbind( seq(1,nrow(xsi.est)) , xsi.est$xsi )
rownames(xsi.fixed) <- rownames(xsi.est)
xsi.fixed <- xsi.fixed[ c(8:13) ,]
xsi.fixed[,2] <- 0
  ##   > xsi.fixed
  ##             [,1] [,2]
  ##   raterAM      8    0
  ##   raterBE      9    0
  ##   raterCO     10    0
  ##   topicFami   11    0
  ##   topicScho   12    0
  ##   topicSpor   13    0
mod04c1 <- tam.mml.mfr( resp , facets= facets ,
            formulaA = formulaA , pid = data.cqc04$pid , xsi.fixed=xsi.fixed )
summary( mod04c1 )

#############################################################################
# EXAMPLE 05: Partial credit model with latent regression and
#             plausible value imputation
#############################################################################

data(data.cqc05)
```

```
resp <- data.cqc05[ , -c(1:3) ] # select item responses

# Partial credit model
mod05a <-tam.mml(resp= resp , irtmodel="PCM2" )
# Partial credit model with latent regressors
mod05b <-tam.mml(resp= resp , irtmodel="PCM2" ,  Y = data.cqc05[,1:3] )
# Plausible value imputation
pvmod05b <- tam.pv( mod05b )

## End(Not run)
```

`data.ctest` *Some C-Test Datasets*

Description

Some C-Test datasets.

Usage

```
data(data.ctest1)
```

Format

- The dataset `data.ctest1` contains item responses of C-tests at two time points. The format is

```
'data.frame':   1675 obs. of  42 variables:
$ idstud : num  100101 100102 100103 100104 100105 ...
$ idclass: num  1001 1001 1001 1001 1001 ...
$ A01T1  : int  0 1 0 1 1 NA 1 0 1 1 ...
$ A02T1  : int  0 1 0 1 0 NA 0 1 1 0 ...
$ A03T1  : int  0 1 1 1 0 NA 0 1 1 1 ...
$ A04T1  : int  1 0 0 0 0 NA 0 0 0 0 ...
$ A05T1  : int  0 0 0 1 1 NA 0 0 1 1 ...
$ B01T1  : int  1 1 0 1 1 NA 0 0 1 0 ...
$ B02T1  : int  0 0 0 1 0 NA 0 0 1 1 ...
[...]
$ C02T2  : int  0 1 1 1 1 0 1 0 1 1 ...
$ C03T2  : int  1 1 0 1 0 0 0 0 1 0 ...
$ C04T2  : int  0 0 1 0 0 0 0 1 0 0 ...
$ C05T2  : int  0 1 0 0 1 0 1 0 0 1 ...
$ D01T2  : int  0 1 1 1 0 1 1 1 1 1 ...
$ D02T2  : int  0 1 1 1 1 1 0 1 1 1 ...
$ D03T2  : int  1 0 0 0 1 0 0 0 0 0 ...
$ D04T2  : int  1 0 1 1 1 0 1 0 1 1 ...
$ D05T2  : int  1 0 1 1 1 1 1 1 1 1 ...
```

data.examples *Datasets* data.ex *in* **TAM** *Package*

Description

Datasets included in the **TAM** package

Usage

```
data(data.ex08)
data(data.ex10)
data(data.ex11)
data(data.ex12)
data(data.ex14)
data(data.ex15)
data(data.exJ03)
```

Format

- Data data.ex08 for Example 8 in tam.mml has the following format:

```
List of 2
$ facets:'data.frame': 1000 obs. of  1 variable:
..$ female: int [1:1000] 1 1 1 1 1 1 1 1 1 1 ...
$ resp  : num [1:1000, 1:10] 1 1 1 0 1 0 1 1 0 1 ...
..- attr(*, "dimnames")=List of 2
.. ..$ : NULL
.. ..$ : chr [1:10] "I0001" "I0002" "I0003" "I0004" ...
```

- Data data.ex10 for Example 10 in tam.mml has the following format:

```
'data.frame':   675 obs. of  7 variables:
$ pid  : int  1 1 1 2 2 3 3 4 4 5 ...
$ rater: int  1 2 3 2 3 1 2 1 3 1 ...
$ I0001: num  0 1 1 1 1 1 1 1 1 1 ...
$ I0002: num  1 1 1 1 1 0 1 1 1 1 ...
$ I0003: num  1 1 1 1 0 0 0 1 0 1 ...
$ I0004: num  0 1 0 0 1 0 1 0 1 0 ...
$ I0005: num  0 0 1 1 1 0 0 1 0 1 ...
```

- Data data.ex11 for Example 11 in tam.mml has the following format:

```
'data.frame':   3400 obs. of  13 variables:
$ booklet: chr  "B1" "B1" "B3" "B2" ...
$ M133   : int  1 1 NA 1 NA 1 NA 1 0 1 ...
$ M176   : int  1 0 1 NA 0 0 0 NA NA NA ...
$ M202   : int  NA NA NA 0 NA NA NA 0 0 0 ...
$ M212   : int  NA NA 1 0 0 NA 0 1 0 0 ...
$ M214   : int  1 0 1 1 0 0 0 0 1 0 ...
```

```
$ M259   : int  NA NA 1 1 1 NA 1 1 1 1 ...
$ M303   : int  NA NA 1 1 1 NA 1 1 1 0 ...
$ M353   : int  NA NA NA 1 NA NA NA 1 1 9 ...
$ M355   : int  NA NA NA 1 NA NA NA 1 1 0 ...
$ M444   : int  0 0 0 NA 0 0 0 NA NA NA ...
$ M446   : int  1 0 0 1 0 1 1 1 0 0 ...
$ M449   : int  NA NA NA 1 NA NA NA 1 1 1 ...
```

Missing responses by design are coded as `NA`, omitted responses are coded as `9`.

- Data `data.ex12` for Example 12 in `tam.mml` has the following format:

```
 num [1:100, 1:10] 1 1 1 1 1 1 1 1 1 1 ...
- attr(*, "dimnames")=List of 2
..$ : NULL
..$ : chr [1:10] "I0001" "I0002" "I0003" "I0004" ...
```

- Data `data.ex14` for Example 14 in `tam.mml` has the following format:

```
'data.frame':   1110 obs. of  11 variables:
$ pid  : num  1001 1001 1001 1001 1001 ...
$ X1   : num  1 1 1 1 1 1 0 0 0 0 ...
$ X2   : int  1 1 1 1 1 1 1 1 1 1 ...
$ rater: int  4 4 4 4 4 4 4 4 4 4 ...
$ crit1: int  0 0 2 1 1 2 0 0 0 0 ...
$ crit2: int  0 0 0 0 0 0 0 0 0 0 ...
$ crit3: int  0 1 1 0 0 1 0 0 1 0 ...
$ crit4: int  0 0 0 1 0 0 0 0 0 0 ...
$ crit5: int  0 0 0 0 1 1 0 0 0 0 ...
$ crit6: int  0 0 0 0 1 0 0 0 0 0 ...
$ crit7: int  1 0 2 0 0 0 0 0 0 0 ...
```

- Data `data.ex15` for Example 15 in `tam.mml` has the following format:

```
'data.frame':   2155 obs. of  182 variables:
$ pid    : num  10001 10002 10003 10004 10005 ...
$ group  : num  1 1 0 0 1 0 1 0 1 1 ...
$ Item001: num  0 NA NA 0 NA NA NA 0 0 NA ...
$ Item002: num  1 NA NA 1 NA NA NA NA 1 NA ...
$ Item003: num  NA NA NA NA 1 NA NA NA NA 1 ...
$ Item004: num  NA NA 0 NA NA NA NA NA NA NA ...
$ Item005: num  NA NA 1 NA NA NA NA NA NA NA ...
[...]
```

This dataset shows an atypical convergence behavior. Look at Example 15 to fix convergence problems using arguments `increment.factor` and `fac.oldxsi`.

- Data `data.exJ03` for Example 4 in `tam.jml` has the following format: `List of 2`

```
$ resp:'data.frame':   40 obs. of  20 variables:
..$ I104: int [1:40] 4 5 6 5 3 4 3 5 4 6 ...
..$ I118: int [1:40] 6 4 6 5 3 2 5 3 5 4 ...
[...]   ..$ I326: int [1:40] 6 1 5 1 4 2 4 1 6 1 ...
```

```
..$ I338: int [1:40] 6 2 6 1 6 2 4 1 6 1 ...
$ X   :'data.frame':   40 obs. of  4 variables:
..$ rater : int [1:40] 40 40 96 96 123 123 157 157 164 164 ...
..$ gender: int [1:40] 2 2 1 1 1 1 2 2 2 2 ...
..$ region: num [1:40] 1 1 1 1 2 2 1 1 1 1 ...
..$ leader: int [1:40] 1 2 1 2 1 2 1 2 1 2 ...
```

It is a rating dataset (a subset of a dataset provided by Matt Barney).

See Also

These examples are used in the `tam.mml` Examples.

`data.fims.Aus.Jpn.scored` *Dataset FIMS Study with Responses of Australian and Japanese Students*

Description

Dataset FIMS study with raw responses (`data.fims.Aus.Jpn.raw`) or scored responses (`data.fims.Aus.Jpn.scored`) of Australian and Japanese Students.

Usage

```
data(data.fims.Aus.Jpn.raw)
data(data.fims.Aus.Jpn.scored)
```

Format

A data frame with 6371 observations on the following 16 variables.

`SEX` Gender: 1 – female, 2 – male

`M1PTI1` A Mathematics item

`M1PTI2` A Mathematics item

`M1PTI3` A Mathematics item

`M1PTI6` A Mathematics item

`M1PTI7` A Mathematics item

`M1PTI11` A Mathematics item

`M1PTI12` A Mathematics item

`M1PTI14` A Mathematics item

`M1PTI17` A Mathematics item

`M1PTI18` A Mathematics item

`M1PTI19` A Mathematics item

M1PTI21 A Mathematics item

M1PTI22 A Mathematics item

M1PTI23 A Mathematics item

country Country: 1 – Australia, 2 – Japan

See Also

http://www.edmeasurementsurveys.com/TAM/Tutorials/7DIF.htm

Examples

```
## Not run:
data(data.fims.Aus.Jpn.scored)
#*****
# Model 1: Differential Item Functioning Gender for Australian students

# extract Australian students
scored <- data.fims.Aus.Jpn.scored[ data.fims.Aus.Jpn.scored$country == 1 , ]

# select items
items <- grep("M1" , colnames(data.fims.Aus.Jpn.scored) , value=TRUE)
##   > items
##    [1] "M1PTI1"  "M1PTI2"  "M1PTI3"  "M1PTI6"  "M1PTI7"  "M1PTI11" "M1PTI12"
##    [8] "M1PTI14" "M1PTI17" "M1PTI18" "M1PTI19" "M1PTI21" "M1PTI22" "M1PTI23"

# Run partial credit model
mod1 <- tam(scored[,items])

# extract values of the gender variable into a variable called "gender".
gender <- scored[,"SEX"]
# computes the test score for each student by calculating the row sum
# of each student's scored responses.
raw_score <- rowSums(scored[,items] )

# compute the mean test score for each gender group: 1=male, and 2=female
aggregate(raw_score,by=list(gender),FUN=mean)
# The mean test score is 6.12 for group 1 (males) and 6.27 for group 2 (females).
# That is, the two groups performed similarly, with girls having a slightly
# higher mean test score. The step of computing raw test scores is not necessary
# for the IRT analyses. But it's always a good practice to explore the data
# a little before delving into more complex analyses.

# Facets analysis
# To conduct a DIF analysis, we set up the variable "gender" as a facet and
# re-run the IRT analysis.
formulaA <- ~item+gender+item*gender # define facets analysis
facets <- as.data.frame(gender)         # data frame with student covariates
# facets model for studying differential item functioning
mod2 <- tam.mml.mfr( resp= scored[,items], facets= facets , formulaA = formulaA )
summary(mod2)

## End(Not run)
```

data.gpcm *Dataset with Ordered Indicators*

Description

Dataset with ordered values of 3 indicators

Usage

```
data(data.gpcm)
```

Format

A data frame with 392 observations on the following 3 items.

`Comfort` a numeric vector

`Work` a numeric vector

`Benefit` a numeric vector

Source

The dataset is copied from the **ltm** package.

Examples

```
data(data.gpcm)
summary(data.gpcm)
```

data.mc *Dataset with Raw and Scored Responses from Multiple Choice Items*

Description

Dataset of responses from multiple choice items, containing 143 students on 30 items.

Usage

```
data(data.mc)
```

Format

The dataset is a list with two elements. The entry `raw` contains unscored (raw) item responses and the entry `scored` contains the scored (recoded) item responses. The format is:

```
List of 2
$ raw   : chr [1:143, 1:30] "A" "A" "A" "A" ...
..- attr(*, "dimnames")=List of 2
.. ..$ : NULL
.. ..$ : chr [1:30] "I01" "I02" "I03" "I04" ...
$ scored:'data.frame':
..$ I01: num [1:143] 1 1 1 1 1 1 1 1 1 ...
..$ I02: num [1:143] 1 1 1 0 1 1 1 1 1 ...
..$ I03: num [1:143] 1 1 1 1 1 1 1 1 1 ...
[...]
..$ I29: num [1:143] NA 0 1 0 1 0 0 0 0 0 ...
..$ I30: num [1:143] NA NA 1 1 1 1 0 1 1 0 ...
```

`data.numeracy` *Dataset Numeracy*

Description

Dataset numeracy with unscored (`raw`) and scored (`scored`) item responses of 876 persons and 15 items.

Usage

```
data(data.numeracy)
```

Format

The format is a list a two entries:

```
List of 2
$ raw   :'data.frame':
..$ I1 : int [1:876] 1 0 1 0 0 0 0 0 1 1 ...
..$ I2 : int [1:876] 0 1 0 0 1 1 1 1 1 0 ...
..$ I3 : int [1:876] 4 4 1 3 4 4 4 4 4 4 ...
..$ I4 : int [1:876] 4 1 2 2 1 1 1 1 1 1 ...
[...]
..$ I15: int [1:876] 1 1 1 1 0 1 1 1 1 1 ...
$ scored:'data.frame':
..$ I1 : int [1:876] 1 0 1 0 0 0 0 0 1 1 ...
..$ I2 : int [1:876] 0 1 0 0 1 1 1 1 1 0 ...
..$ I3 : int [1:876] 1 1 0 0 1 1 1 1 1 1 ...
..$ I4 : int [1:876] 0 1 0 0 1 1 1 1 1 1 ...
[...]
```

```
..$ I15: int [1:876] 1 1 1 1 0 1 1 1 1 1 ...
```

Examples

```
############################################################################
# (1) Scored numeracy data
############################################################################

data(data.numeracy)
dat <- data.numeracy$scored

#Run IRT analysis: Rasch model
mod1 <- tam(dat)

#Item difficulties
mod1$xsi
ItemDiff <- mod1$xsi$xsi
ItemDiff

#Ability estimate - Weighted Likelihood Estimate
Abil <- tam.wle(mod1)
Abil
PersonAbility <- Abil$theta
PersonAbility

#Descriptive statistics of item and person parameters
hist(ItemDiff)
hist(PersonAbility)
mean(ItemDiff)
mean(PersonAbility)
sd(ItemDiff)
sd(PersonAbility)

## Not run:
#Extension
#plot histograms of ability and item parameters in the same graph
oldpar <- par(no.readonly = TRUE)       # save writable default graphic settings
windows (width=4.45, height=4.45, pointsize=12)
layout(matrix(c(1,1,2),3,byrow=TRUE))
layout.show(2)
hist(PersonAbility,xlim=c(-3,3),breaks=20)
hist(ItemDiff,xlim=c(-3,3),breaks=20)

par( oldpar )  # restore default graphic settings
hist(PersonAbility,xlim=c(-3,3),breaks=20)

############################################################################
# (2) Raw numeracy data
############################################################################

raw_resp <- data.numeracy$raw
```

```
#score responses
key <- c(1, 1, 4, 1, 1, 1, 1, 1, 1, 1, 3, 1, 1, 1, 1)
scored <- sapply( seq(1,length(key)) ,
            FUN = function(ii){ 1*(raw_resp[,ii] == key[ii]) } )

#run IRT analysis
mod1 <- tam(scored)

#Ability estimate - Weighted Likelihood Estimate
Abil <- tam.wle(mod1)

#CTT statistics
ctt1 <- tam.ctt(raw_resp, Abil$theta)
write.csv(ctt1,"D1_ctt1.csv")       # write statistics into a file
        # use maybe write.csv2 if ';' should be the column separator

#Fit statistics
Fit <- tam.fit(mod1)
Fit

# plot expected response curves
plot( mod1 , ask=TRUE )

## End(Not run)
```

`data.timssAusTwn` *Dataset TIMSS 2011 of Australian and Taiwanese Students*

Description

Mathematics items of TIMSS 2011 of 1773 Australian and Taiwanese students. The dataset `data.timssAusTwn` contains raw responses while `data.timssAusTwn.scored` contains scored item responses.

Usage

```
data(data.timssAusTwn)
data(data.timssAusTwn.scored)
```

Format

A data frame with 1773 observations on the following 14 variables.

`M032166` a mathematics item

`M032721` a mathematics item

`M032757` a mathematics item

`M032760A` a mathematics item

`M032760B` a mathematics item

`M032760C` a mathematics item

M032761 a mathematics item

M032692 a mathematics item

M032626 a mathematics item

M032595 a mathematics item

M032673 a mathematics item

IDCNTRY Country identifier

ITSEX Gender

IDBOOK Booklet identifier

See Also

http://www.edmeasurementsurveys.com/TAM/Tutorials/5PartialCredit.htm

http://www.edmeasurementsurveys.com/TAM/Tutorials/6Population.htm

Examples

```
data(data.timssAusTwn)
raw_resp <- data.timssAusTwn

#Recode data
resp <- raw_resp[,1:11]
      #Column 12 is country code. Column 13 is gender code. Column 14 is Book ID.
all.na <- rowMeans( is.na(resp) ) == 1
        #Find records where all responses are missing.
resp <- resp[!all.na,]              #Delete records with all missing responses
resp[resp==20 | resp==21] <- 2     #TIMSS double-digit coding: "20" or "21" is a score of 2
resp[resp==10 | resp==11] <- 1     #TIMSS double-digit coding: "10" or "11" is a score of 1
resp[resp==70 | resp==79] <- 0     #TIMSS double-digit coding: "70" or "79" is a score of 0
resp[resp==99] <- 0                #"99" is omitted responses. Score it as wrong here.
resp[resp==96 | resp==6] <- NA     #"96" and "6" are not-reached items. Treat these as missing.

#Score multiple-choice items        #"resp" contains raw responses for MC items.
Scored <- resp
Scored[,9] <- (resp[,9]==4)*1       #Key for item 9 is D.
Scored[,c(1,2)] <- (resp[,c(1,2)]==2)*1  #Key for items 1 and 2 is B.
Scored[,c(10,11)] <- (resp[,c(10,11)]==3)*1  #Key for items 10 and 11 is C.

#Run IRT analysis for partial credit model (MML estimation)
mod1 <- tam(Scored)

#Item parameters
mod1$xsi

#Thurstonian thresholds
tthresh <- tam.threshold(mod1)
tthresh

## Not run:
#Plot Thurstonian thresholds
```

```
windows (width=8, height=7)
par(ps=9)
dotchart(t(tthresh), pch=19)
# plot expected response curves
plot( mod1 , ask=TRUE)

#Re-run IRT analysis in JML
mod1.2 <- tam.jml2(Scored)
var(mod1.2$WLE)

#Re-run the model with "not-reached" coded as incorrect.
Scored2 <- Scored
Scored2[is.na(Scored2)] <- 0

#Prepare anchor parameter values
nparam <- length(mod1$xsi$xsi)
xsi <- mod1$xsi$xsi
anchor <- matrix(c(seq(1,nparam),xsi), ncol=2)

#Run IRT with item parameters anchored on mod1 values
mod2 <- tam(Scored2, xsi.fixed=anchor)

#WLE ability estimates
ability <- tam.wle(mod2)
ability

#CTT statistics
ctt <- tam.ctt(resp, ability$theta)
write.csv(ctt,"TIMSS_CTT.csv")

#plot histograms of ability and item parameters in the same graph
windows (width=4.45, height=4.45, pointsize=12)
layout(matrix(c(1,1,2),3,byrow=TRUE))
layout.show(2)
hist(ability$theta,xlim=c(-3,3),breaks=20)
hist(tthresh,xlim=c(-3,3),breaks=20)

#Extension
#Score equivalence table
dummy <- matrix(0,nrow=16,ncol=11)
dummy[lower.tri(dummy)] <- 1
dummy[12:16,c(3,4,7,8)][lower.tri(dummy[12:16,c(3,4,7,8)])]<-2

mod3 <- tam(dummy, xsi.fixed=anchor)
wle3 <- tam.wle(mod3)

## End(Not run)
```

---

designMatrices *Generation of Design Matrices*

Description

Generate design matrices, and display them at console.

Usage

```
designMatrices(modeltype = c("PCM", "RSM"), maxKi = NULL, resp = resp,
    ndim = 1, A = NULL, B = NULL, Q = NULL, R = NULL, ...)

print.designMatrices(X, ...)

designMatrices.mfr(resp, formulaA = ~ item + item:step, facets = NULL,
    constraint = c("cases", "items"), ndim = 1, Q=NULL, A=NULL, B=NULL ,
    progress=FALSE)
designMatrices.mfr2(resp, formulaA = ~ item + item:step, facets = NULL,
    constraint = c("cases", "items"), ndim = 1, Q=NULL, A=NULL, B=NULL ,
    progress=FALSE)

.A.matrix(resp, formulaA = ~ item + item*step, facets = NULL,
    constraint = c("cases", "items") , progress=FALSE , maxKi=NULL)
rownames.design(X)

.A.PCM2( resp, Kitem=NULL)
   # generates ConQuest parametrization of partial credit model

.A.PCM3( resp, Kitem=NULL ) # parametrization for A matrix in the dispersion model
```

Arguments

modeltype	Type of item response model. Until now, the partial credit model (PCM; 'item+item*step') and the rating scale model (RSM; 'item+step') is implemented.
maxKi	A vector containing the maximum score per item
resp	Data frame of item responses
ndim	Number of dimensions
A	The design matrix for linking item category parameters to generalized item parameters ξ.
B	The scoring matrix of item categories on θ dimensions.
Q	A loading matrix of items on dimensions with number of rows equal the number of items and the number of columns equals the number of dimensions in the item response model.
R	This argument is not used
X	Object generated by designMatrices. This argument is used in print.designMatrices and rownames.design.
formulaA	An R formula object for generating the A design matrix. Variables in formulaA have to be included in facets.
facets	A data frame with observed facets. The number of rows must be equal to the number of rows in resp.

constraint	Constraint in estimation: cases assumes zero means of trait distributions and items a sum constraint of zero of item parameters
Kitem	Maximum number of categories per item
progress	Display progress for creation of design matrices
...	Further arguments

Details

The function .A.PCM2 generates the Conquest parametrization of the partial credit model.

The function .A.PCM3 generates the parametrization for the A design matrix in the dispersion model for ordered data (Andrich, 1982).

Note

The function designMatrices.mfr2 handles multi-faceted design for items with differing number of response options.

References

Andrich, D. (1982). An extension of the Rasch model for ratings providing both location and dispersion parameters. *Psychometrika*, **47**, 105-113.

See Also

See sim.mfr for some examples for creating design matrices.

Examples

```
############################################################
# different parametrizations for ordered data
data( data.gpcm )
resp <- data.gpcm

# parametrization for partial credit model
A1 <- designMatrices( resp= resp )$A
# item difficulty and threshold parametrization
A2 <- .A.PCM2( resp )
# dispersion model of Andrich (1982)
A3 <- .A.PCM3( resp )
# rating scale model
A4 <- designMatrices( resp= resp , modeltype="RSM" )$A
```

IRT.factor.scores *Extracting Factor Scores in* **TAM**

Description

Extracts factor scores for models fitted in **TAM**. See IRT.factor.scores (**CDM**) for more details.

Usage

```
## S3 method for class 'tam'
IRT.factor.scores(object, type="EAP", ...)
## S3 method for class 'tam.mml'
IRT.factor.scores(object, type="EAP", ...)
## S3 method for class 'tam.mml.3pl'
IRT.factor.scores(object, type="EAP", ...)
```

Arguments

object	Object of class tam, tam.mml or tam.mml.3pl.
type	Type of factor score to be used. type="EAP" can be used for all classes in **TAM** while type="WLE" and type="MLE" can only be used for objects of class tam.mml. Further arguments to the used function tam.wle can be specified with ....
...	Further arguments to be passed

Value

See IRT.factor.scores (**CDM**).

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: sim.rasch - Different factor scores in Rasch model
#############################################################################

data(sim.rasch)
# 1PL estimation
mod1 <- tam.mml(resp=sim.rasch)
# EAP
pmod1 <- IRT.factor.scores( mod1 )
# WLE
pmod1 <- IRT.factor.scores( mod1 , type="WLE" )
# MLE
pmod1 <- IRT.factor.scores( mod1 , type="MLE" )

## End(Not run)
```

IRT.irfprob *Extracting Item Response Functions*

Description

Extracts item response functions for models fitted in **TAM**. See `IRT.irfprob` (**CDM**) for more details.

Usage

```
## S3 method for class 'tam'
IRT.irfprob(object, ...)
## S3 method for class 'tam.mml'
IRT.irfprob(object, ...)
## S3 method for class 'tam.mml.3pl'
IRT.irfprob(object, ...)
```

Arguments

`object`	Object of class `tam`, `tam.mml` or `tam.mml.3pl`.
`...`	Further arguments to be passed

Value

See `IRT.irfprob` (**CDM**).

Examples

```
#############################################################################
# EXAMPLE 1: Dichotomous data sim.rasch - item response functions
#############################################################################

data(sim.rasch)
# 1PL estimation
mod1 <- tam.mml(resp=sim.rasch)
IRT.irfprob(mod1)
```

IRT.likelihood *Extracting Individual Likelihood and Individual Posterior*

Description

Extracts individual likelihood and posterior for models fitted in **TAM**. See `IRT.likelihood` (**CDM**) for more details.

Usage

```
## S3 method for class 'tam'
IRT.likelihood(object, ...)
## S3 method for class 'tam.mml'
IRT.likelihood(object, ...)
## S3 method for class 'tam.mml.3pl'
IRT.likelihood(object, ...)

## S3 method for class 'tam'
IRT.posterior(object, ...)
## S3 method for class 'tam.mml'
IRT.posterior(object, ...)
## S3 method for class 'tam.mml.3pl'
IRT.posterior(object, ...)
```

Arguments

`object`	Object of class `tam`, `tam.mml` or `tam.mml.3pl`.
`...`	Further arguments to be passed

Value

See `IRT.likelihood` (**CDM**).

Examples

```
#############################################################################
# EXAMPLE 1: Dichotomous data sim.rasch - extracting likelihood/posterior
#############################################################################

data(sim.rasch)
# 1PL estimation
mod1 <- tam.mml(resp=sim.rasch)
lmod1 <- IRT.likelihood(mod1)
str(lmod1)
pmod1 <- IRT.posterior(mod1)
str(pmod1)
```

`lavaanify.IRT` *Slight Extension of the* `lavaan` *Syntax, with Focus on Item Response Models*

Description

This functions slightly extends the `lavaan` syntax implemented in the **lavaan** package (see `lavaanify`). Guessing and slipping parameters can be specified by using the operators `?=g1` and `?=s1`, respectively. The operator `__` can be used for a convenient specification for groups of items. For example, `I1__I5` refers to items `I1,...,I5`. The operator `__` can also be used for item labels (see Example 2).

Usage

```
lavaanify.IRT(lavmodel, items=NULL, data=NULL)
```

Arguments

lavmodel	A model in lavaan syntax plus the additional operators ?=g1 and ?=s1.
items	Optional vector of item names
data	Optional data frame with item responses

Value

A list with following entries

lavpartable	A lavaan parameter table
lavaan.syntax	Processed syntax for lavaan package

Author(s)

Alexander Robitzsch

See Also

lavaanify (**lavaan**)

See tam2mirt (**sirt**) for converting objects of class tam into mirt objects.

See lavaan2mirt (**sirt**) for estimating models in the **mirt** package using lavaan syntax.

Examples

```
#############################################################################
# EXAMPLE 1: lavaan syntax with guessing and slipping parameters
#############################################################################

# define model in lavaan
lavmodel <- "
    F =~ A1+c*A2+A3+A4
    # define slipping parameters for A1 and A2
    A1 + A2 ?= s1
    # joint guessing parameter for A1 and A2
    A1+A2 ?= c1*g1
    A3 | 0.75*t1
    # fix guessing parameter to .25 and
    # slipping parameter to .01 for item A3
    A3 ?= .25*g1+.01*s1
    A4 ?= c2*g1
    A1 | a*t1
    A2 | b*t1
      "

# process lavaan syntax
lavpartable <- lavaanify.IRT(lavmodel)$lavpartable
```

```
##     id lhs op rhs user group free ustart exo label eq.id unco
##  1   1   F =~  A1    1     1    1     NA   0           0    1
##  2   2   F =~  A2    1     1    2     NA   0     c     0    2
##  3   3   F =~  A3    1     1    3     NA   0           0    3
##  4   4   F =~  A4    1     1    4     NA   0           0    4
##  5   5  A3  |  t1    1     1    0   0.75   0           0    0
##  6   6  A1  |  t1    1     1    5     NA   0     a     0    5
##  7   7  A2  |  t1    1     1    6     NA   0     b     0    6
##  8   8  A1 ?=  s1    1     1    7     NA   0           0    7
##  9   9  A2 ?=  s1    1     1    8     NA   0           0    8
##  10 10  A1 ?=  g1    1     1    9     NA   0    c1     1    9
##  11 11  A2 ?=  g1    1     1    9     NA   0    c1     1   10
##  12 12  A3 ?=  g1    1     1    0   0.25   0           0    0
##  13 13  A3 ?=  s1    1     1    0   0.01   0           0    0
##  14 14  A4 ?=  g1    1     1   10     NA   0    c2     0   11

## Not run:
#############################################################################
# EXAMPLE 2: Usage of "__" and "?=" operators
#############################################################################

library(sirt)
data(data.read, package="sirt")
dat <- data.read
items <- colnames(dat)

lavmodel <- "
   F1 =~ A1+A2+ A3+lam4*A4
   # equal item loadings for items B1 to B4
   F2 =~ lam5*B1__B4
   # different labelled item loadings of items C1 to C4
   F3 =~ lam9__lam12*C1__C4
   # item intercepts
   B1__B2 | -0.5*t1
   B3__C1 | int6*t1
   # guessing parameters
   C1__C3 ?= g1
   C4 + B1__B3 ?= 0.2*g1
   # slipping parameters
   A1__B1 + B3__C2 ?= slip1*s1
   # residual variances
   B1__B3 ~~ errB*B1__B3
   A2__A4 ~~ erra1__erra3*A2__A4
    "
lav2 <- lavaanify.IRT( lavmodel , data=dat)
lav2$lavpartable
cat( lav2$lavaan.syntax )

#** simplified example
lavmodel <- "
   F1 =~ A1+lam4*A2+A3+lam4*A4
   F2 =~ lam5__lam8*B1__B4
   F1 ~~ F2
```

```
   F1 ~~ 1*F1
   F2 ~~ 1*F2
    "
lav3 <- lavaanify.IRT( lavmodel , data=dat)
lav3$lavpartable
cat( lav3$lavaan.syntax )

## End(Not run)
```

---

plot.tam *Plot Expected Response Curves for Unidimensional Item Response Models*

Description

S3 plot method for objects of class tam, tam.mml or tam.mml.

Usage

```
## S3 method for class 'tam'
plot(x, items=1:x$nitems, low=-3, high=3, ngroups=6,
                  wle=NULL, export=TRUE, export.type="png",
                  export.args=list(), observed=TRUE, overlay=FALSE ,
                  ask=FALSE, ...)

## S3 method for class 'tam.mml'
plot(x, items=1:x$nitems, low=-3, high=3, ngroups=6,
                      wle=NULL, export=TRUE, export.type="png",
                      export.args=list(), observed=TRUE, overlay=FALSE ,
                      ask=FALSE, ...)

## S3 method for class 'tam.jml'
plot(x, items=1:x$nitems, low=-3, high=3, ngroups=6,
                      wle=NULL, export=TRUE, export.type="png",
                      export.args=list(), observed=TRUE, overlay=FALSE ,
                      ask=FALSE, ...)
```

Arguments

x	Object of class tam, tam.mml or tam.mml.
items	An index vector giving the items to be visualized.
low	Lowest θ value to be displayed
high	Highest θ value to be displayed
ngroups	Number of score groups to be displayed. The default are six groups.
wle	Use WLE estimate for displaying observed scores.

export	A logical which indicates whether all graphics should be separately exported in files of type export.type in a subfolder 'Plots' of the working directory.
export.type	A string which indicates the type of the graphics export. For currently supported file types, see dev.new.
export.args	A list of arguments that are passed to the export method can be specified. See the respective export device method for supported usage.
observed	A logical which indicates whether observed response curve should be displayed
overlay	A logical indicating whether expected score functions should overlay.
ask	A logical which asks for changing the graphic from item to item. The default is FALSE.
...	Further arguments to be passed

Details

This plot method does not work for multidimensional item response models.

Examples

```
data(sim.rasch)
mod <- tam.mml(sim.rasch)
plot(mod, items=1:5, export=FALSE)
```

---

plotDevianceTAM *Deviance Plot for* **TAM** *Objects*

Description

Plots the deviance change in every iteration.

Usage

```
plotDevianceTAM(tam.obj, omitUntil = 1, reverse = TRUE)
```

Arguments

tam.obj	Object of class tam.mml, tam.mml.2pl or tam.mml.mfr.
omitUntil	An optional value indicating number of iterations to be omitted for plotting.
reverse	A logical indicating whether the deviance change should be multiplied by minus 1. The default is TRUE.

Author(s)

Martin Hecht, Sebastian Weirich

Examples

```
#############################################################################
# EXAMPLE 1: deviance plot dichotomous data
#############################################################################
data(sim.rasch)

# 2PL model
mod1 <- tam.mml.2pl(resp=sim.rasch )
# plot deviance change
plotDevianceTAM( mod1 )
```

`sim.mfr` *Simulated Multifaceted Data*

Description

Simulated data from multiple facets.

Usage

```
data(sim.mfr)
data(sim.facets)
```

Format

The format of `sim.mfr` is:

```
num [1:100, 1:5] 3 2 1 1 0 1 0 1 0 0 ...
- attr(*, "dimnames")=List of 2
..$ : chr [1:100] "V1" "V1.1" "V1.2" "V1.3" ...
..$ : NULL
```

The format of `sim.facets` is:

```
'data.frame':   100 obs. of  3 variables:
$ rater : num  1 2 3 4 5 1 2 3 4 5 ...
$ topic : num  3 1 3 1 3 2 3 2 2 1 ...
$ female: num  2 2 1 2 1 1 2 1 2 1 ...
```

Source

Simulated

Examples

```
#######
# sim multi faceted Rasch model
data(sim.mfr)
data(sim.facets)
```

```
  # 1: A-matrix test_rater
  test_1_items <- .A.matrix( sim.mfr, formulaA = ~rater,
            facets = sim.facets, constraint = "items" )
  test_1_cases <- .A.matrix( sim.mfr, formulaA = ~rater,
            facets = sim.facets, constraint = "cases" )

  # 2: test_item+rater
  test_2_cases <- .A.matrix( sim.mfr, formulaA = ~item+rater,
            facets = sim.facets, constraint = "cases" )

  # 3: test_item+rater+topic+ratertopic
  test_3_items <- .A.matrix( sim.mfr, formulaA = ~item+rater*topic,
            facets = sim.facets, constraint = "items" )
  # conquest uses a different way of ordering the rows
  # these are the first few rows of the conquest design matrix
  # test_3_items$A[grep("item1([[:print:]])*topic1", rownames(test_3_items)),]

  # 4: test_item+step
  test_4_cases <- .A.matrix( sim.mfr, formulaA = ~item+step,
            facets = sim.facets, constraint = "cases" )

  # 5: test_item+item:step
  test_5_cases <- .A.matrix( sim.mfr, formulaA = ~item+item:step,
            facets = sim.facets, constraint = "cases" )
  test_5_cases$A[, grep("item1", colnames(test_5_cases)) ]

  # 5+x: more
  #  => 6: is this even well defined in the conquest-design output
  #          (see test_item+topicstep_cases.cqc / .des)
  #        regardless of the meaning of such a formula;
  #        currently .A.matrix throws a warning
  # test_6_cases <- .A.matrix( sim.mfr, formulaA = ~item+topic:step,
  #                 facets = sim.facets, constraint = "cases" )
  test_7_cases <- .A.matrix( sim.mfr, formulaA = ~item+topic+topic:step,
            facets = sim.facets, constraint = "cases" )

## Not run:
  # => 8: same as with 6
  test_8_cases <- .A.matrix( sim.mfr, formulaA = ~item+rater+item:rater:step,
            facets = sim.facets, constraint = "cases" )
## [1] "Can't proceed the estimation: Lower-order term is missing."
  test_9_cases <- .A.matrix( sim.mfr, formulaA = ~item+step+rater+item:step+item:rater,
            facets = sim.facets, constraint = "cases" )
  test_10_cases <- .A.matrix( sim.mfr, formulaA = ~item+female+item:female,
            facets = sim.facets, constraint = "cases" )

  ### All Design matrices
  test_1_cases <- designMatrices.mfr( sim.mfr, formulaA = ~rater,
            facets = sim.facets, constraint = "cases" )
  test_4_cases <- designMatrices.mfr( sim.mfr, formulaA = ~item+item:step,
            facets = sim.facets, constraint = "cases" )

  ### TAM
```

```
  test_4_cases <- tam.mml.mfr( sim.mfr, formulaA = ~item+item:step )
  test_tam <- tam.mml( sim.mfr )

  test_1_cases <- tam.mml.mfr( sim.mfr, formulaA = ~rater,
            facets = sim.facets, constraint = "cases" )
  test_2_cases <- tam.mml.mfr( sim.mfr, formulaA = ~item+rater,
            facets = sim.facets, constraint = "cases" )
## End(Not run)
```

`sim.rasch` *Simulated Rasch data*

Description

Simulated Rasch data under unidimensional trait distribution

Usage

```
data(sim.rasch)
data(sim.rasch.pweights)
data(sim.rasch.missing)
```

Format

The format is:

```
num [1:2000, 1:40] 1 0 1 1 1 1 1 1 1 1 ...
- attr(*, "dimnames")=List of 2
..$ : NULL
..$ : chr [1:40] "I1" "I2" "I3" "I4" ...
```

Details

```
N <- 2000
# simulate predictors
Y <- cbind( rnorm( N , sd = 1.5) , rnorm(N , sd = .3 ) )
theta <- rnorm( N ) + .4 * Y[,1] + .2 * Y[,2]  # latent regression model
# simulate item responses with missing data
I <- 40
resp[ theta < 0 , c(1,seq(I/2+1 , I)) ] <- NA
# define person weights
pweights <- c(  rep(3,N/2) , rep( 1, N/2 ) )
```

Source

Simulated data (see Details)

Examples

```
## Not run:
data(sim.rasch)
N <- 2000
Y <- cbind( rnorm( N , sd = 1.5) , rnorm(N , sd = .3 ) )

# Loading Matrix
# B <- array( 0 , dim = c( I , 2 , 1 )  )
# B[1:(nrow(B)), 2, 1] <- 1
B <- designMatrices(resp = sim.rasch)[["B"]]

# estimate model
mod1_1 <- tam(resp=sim.rasch , Y=Y)

# standard errors
res1 <- tam.se(mod1_1)

# Compute fit statistics
tam.fit(mod1_1)

# plausible value imputation
# PV imputation has to be adpated for multidimensional case!
pv1 <- tam.pv( mod1_1 , nplausible = 7 , # 7 plausible values
               samp.regr = TRUE       # sampling of regression coefficients
              )

# item parameter constraints
xsi.fixed <- matrix( c( 1, -2,5, -.22,10, 2 ), nrow=3 , ncol=2 , byrow=TRUE)
xsi.fixed
mod1_4 <- tam( resp=sim.rasch , xsi.fixed=xsi.fixed )

# missing value handling
data(sim.rasch.missing)
mod1_2 <- tam(sim.rasch.missing , Y = Y)

# handling of sample (person) weights
data(sim.rasch.pweights)
N <- 1000
pweights <- c(  rep(3,N/2) , rep( 1, N/2 ) )
mod1_3 <- tam( sim.rasch.pweights , control = list(conv = .001) , pweights = pweights )
## End(Not run)
```

`tam.ctt` *Classical Test Theory Based Statistics and Plots*

Description

This function computes some item statistics based on classical test theory.

Usage

```
tam.ctt(resp, wlescore=NULL, pvscores=NULL, group=NULL , progress=TRUE)
tam.ctt2(resp, wlescore=NULL, group=NULL , allocate=30 , progress=TRUE)
tam.ctt3(resp, wlescore=NULL, group=NULL , allocate=30 , progress=TRUE)

plotctt( resp , theta , Ncuts = NULL , ask = FALSE , ... )
```

Arguments

resp	A data frame with unscored or scored item responses
wlescore	A vector with person parameter estimates, e.g. weighted likelihood estimates obtained from tam.wle. If wlescore=NULL is chosen in tam.ctt2, then only a frequency table of all items is produced.
pvscores	A matrix with plausible values, e.g. obtained from tam.pv
group	Vector of group identifiers if descriptive statistics shall be groupwise calculated
progress	An optional logical indicating whether computation progress should be displayed.
allocate	Average number of categories per item. This argument is just used for matrix size allocations. If an error is produced, use a sufficiently higher number.
theta	A score to be conditioned
Ncuts	Number of break points for theta
ask	A logical which asks for changing the graphic from item to item. The default is FALSE.
...	Further arguments to be passed.

Details

The functions tam.ctt2 and tam.ctt3 use **Rcpp** code and are slightly faster. However, only tam.ctt allows the input of wlescore and pvscores.

Value

A data frame with following columns:

index	Index variable in this data frame
group	Group identifier
itemno	Item number
item	Item
N	Number of students responding to this item
Categ	Category label
AbsFreq	Absolute frequency of category
RelFreq	Relative frequency of category
rpb.WLE	Point biserial correlation of an item category and the WLE

M.WLE	Mean of the WLE of students in this item category
SD.WLE	Standard deviation of the WLE of students in this item category
rpb.PV	Point biserial correlation of an item category and the PV
M.PV	Mean of the PV of students in this item category
SD.PV	Standard deviation of the PV of students in this item category

Note

For dichotomously scored data, rpb.WLE is the ordinary point biserial correlation of an item and a test score (here the WLE).

See Also

http://www.edmeasurementsurveys.com/TAM/Tutorials/4CTT.htm

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: Multiple choice data data.mc
#############################################################################

data(data.mc)
# estimate Rasch model for scored data.mc data
mod <- tam.mml( resp=data.mc$scored )
# estimate WLE
w1 <- tam.wle( mod )
# estimate plausible values
set.seed(789)
p1 <- tam.pv( mod , ntheta=500 , normal.approx=TRUE )$pv

# CTT results for raw data
stat1 <- tam.ctt( resp=data.mc$raw , wlescore=w1$theta , pvscores=p1[,-1] )
stat1a <- tam.ctt2( resp=data.mc$raw , wlescore=w1$theta )  # faster
stat1b <- tam.ctt2( resp=data.mc$raw )  # only frequencies
stat1c <- tam.ctt3( resp=data.mc$raw , wlescore=w1$theta )  # faster

# plot empirical item response curves
plotctt( resp=data.mc$raw , theta = w1$theta , Ncuts =5 , ask=TRUE)

# CTT results for scored data
stat2 <- tam.ctt( resp=data.mc$scored , wlescore=w1$theta , pvscores=p1[,-1] )

# descriptive statistics for different groups
# define group identifier
group <- c( rep(1,70) , rep(2,73) )
stat3 <- tam.ctt( resp=data.mc$raw , wlescore=w1$theta , pvscores=p1[,-1] , group=group)
stat3a <- tam.ctt2( resp=data.mc$raw , wlescore=w1$theta ,  group=group)

## End(Not run)
```

tam.fa *Bifactor Model and Exploratory Factor Analysis*

Description

Estimates the bifactor model and exploratory factor analysis with marginal maximum likelihood estimation.

This function is simply a wrapper to `tam.mml` or `tam.mml.2pl`.

Usage

```
tam.fa(resp, irtmodel, dims = NULL, nfactors = NULL, pid = NULL,
    pweights = NULL, control = list())
```

Arguments

`resp`	Data frame with polytomous item responses $k = 0, ..., K$. Missing responses must be declared as `NA`.
`irtmodel`	A string which defines the IRT model to be estmated. Options are `"efa"` (exploratory factor analysis), `"bifactor1"` (Rasch testlet model in case of dichotomous data; Wang & Wilson, 2005; for polytomous data it assumes item slopes of 1) and `"bifactor2"` (bifactor model). See Details for more information.
`dims`	A numeric or string vector which only applies in case of `irtmodel="bifactor1"` or `irtmodel="bifactor2"`. Different entries in the vector indicate different dimensions of items which should load on the nested factor. If items should only load on the general factor, then an `NA` must be specified.
`nfactors`	A numerical value which indicates the number of factors in exploratory factor analysis.
`pid`	An optional vector of person identifiers
`pweights`	An optional vector of person weights
`control`	See `tam.mml` for more details. Note that the default is Quasi Monte Carlo integration with 1500 nodes (`snodes=1500`, `QMC=TRUE`).

Details

The exploratory factor analysis (`irtmodel="efa"` is estimated using an echelon form of the loading matrix and uncorrelated factors. The obtained standardized loading matrix is rotated using oblimin rotation. In addition, a Schmid-Leimann transformation (see Revelle & Zinbarg, 2009) is employed.

The bifactor model (`irtmodel="bifactor2"`; Reise 2012) for dichotomous responses is defined as

$$logit P(X_{pi} = 1 | \theta_{pg}, u_{p1}, \ldots, u_{pD}) = a_{i0}\theta_{pg} + a_{i1} u_{pd(i)}$$

Items load on the general factor θ_{pg} and a specific (nested) factor $u_{pd(i)}$. All factors are assumed to be uncorrelated.

In the Rasch testlet model (`irtmodel="bifactor1"`), all item slopes are set to 1 and variances are estimated.

For polytomous data, the generalized partial credit model is used. The loading structure is defined in the same way as for dichotomous data.

Value

The same list entries as in `tam.mml` but in addition the following statistics are included:

`B.stand`	Standardized factor loadings of the bifactor model or the exploratory factor analysis.
`B.SL`	In case of exploratory factor analysis (`irtmodel="efa"`), loadings form the Schmid-Leimann solution of the **psych** package.
`efa.oblimin`	Output from oblimin rotation in exploratory factor analysis which is produced by the **GPArotation** package
`meas`	Vector of dimensionality and reliability statistics. Included are the ECV measure (Reise, Moore & Haviland, 2010; Reise, 2012), ω_t (Omega Total), ω_a (Omega asymptotical) and ω_h (Omega hierarchical) (Revelle & Zinbarg, 2009). The reliability of the sum score based on the bifactor model for dichotomous item responses is also included (Green & Yang, 2009).

References

Green, S. B., & Yang, Y. (2009). Reliability of summed item scores using structural equation modeling: An alternative to coefficient alpha. *Psychometrika*, **74**, 155-167.

Reise, S. P. (2012). The rediscovery of bifactor measurement models. *Multivariate Behavioral Research*, **47**, 667-696.

Reise, S. P., Moore, T. M., & Haviland, M. G. (2010). Bifactor models and rotations: Exploring the extent to which multidimensional data yield univocal scale scores, *Journal of Personality Assessment*, **92**, 544-559.

Revelle, W., & Zinbarg, R. E. (2009). Coefficients alpha, beta, omega and the glb: Comments on Sijtsma. *Psychometrika*, **74**, 145-154.

Wang, W.-C., & Wilson, M. (2005). The Rasch testlet model. *Applied Psychological Measurement*, **29**, 126-149.

See Also

For more details see `tam.mml` because `tam.fa` is just a wrapper for `tam.mml.2pl` and `tam.mml`.

`logLik.tam`, `anova.tam`

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: Dataset reading from sirt package
#############################################################################
```

```
data(data.read,package="sirt")
resp <- data.read

#***
# Model 1a: Exploratory factor analysis with 2 factors
mod1a <- tam.fa( resp=resp , irtmodel="efa" , nfactors= 2  )
summary(mod1a)
# varimax rotation
varimax(mod1a$B.stand)
# promax rotation
promax(mod1a$B.stand)
# more rotations are included in the GPArotation package
library(GPArotation)
# geomin rotation oblique
GPArotation::geominQ( mod1a$B.stand )
# quartimin rotation
GPArotation::quartimin( mod1a$B.stand )

#***
# Model 1b: Rasch testlet model with 3 testlets
dims <- substring( colnames(resp),1,1 )  # define dimensions
mod1b <- tam.fa( resp=resp , irtmodel="bifactor1" , dims=dims )
summary(mod1b)

#***
# Model 1c: Bifactor model
mod1c <- tam.fa( resp=resp , irtmodel="bifactor2" , dims=dims )
summary(mod1c)

#***
# Model 1d: reestimate Model 1c but assume that items 3 and 5 do not load on
#           specific factors
dims1 <- dims
dims1[c(3,5)] <- NA
mod1d <- tam.fa( resp=resp , irtmodel="bifactor2" , dims=dims1 )
summary(mod1d)

## End(Not run)
```

`tam.fit` *Item Infit and Outfit Statistic*

Description

The item infit and outfit statistic are calculated for objects of classes `tam`, `tam.mml` and `tam.jml`, respectively.

Usage

```
tam.fit(tamobj, ...)
```

```
tam.mml.fit(tamobj, FitMatrix=NULL, Nsimul=NULL,progress=TRUE,
   useRcpp=TRUE, seed=123, fit.facets=TRUE)

tam.jml.fit(tamobj)
```

Arguments

tamobj	An object of class tam, tam.mml or tam.jml
FitMatrix	A fit matrix F for a specific hypothesis of fit of the linear function $F\xi$ (see Simulated Example 3 and Adams & Wu 2007).
Nsimul	Number of simulations used for fit calculation. The default is 100 (less than 400 students), 40 (less than 1000 students), 15 (less than 3000 students) and 5 (more than 3000 students)
progress	An optional logical indicating whether computation progress should be displayed at console.
useRcpp	Optional logical indicating whether **Rcpp** or pure R code should be used for fit calculation. The latter is consistent with **TAM** (<=1.1).
seed	Fixed simulation seed.
fit.facets	An optional logical indicating whether fit for all facet parameters should be computed.
...	Further arguments to be passed

Details

Item fit is automatically calculated in JML estimation using tam.jml.

Value

In case of tam.mml.fit a data frame with four columns:

Outfit	Item outfit statistic
Outfit_t	The t value for the outfit statistic
Outfit_p	Significance p value for outfit statistic
Outfit_phom	Significance p value for outfit statistic, adjusted for multiple testing according to the Holm procedure
Infit	Item infit statistic
Infit_t	The t value for the infit statistic
Infit_p	Significance p value for infit statistic
Infit_phom	Significance p value for infit statistic, adjusted for multiple testing according to the Holm procedure

References

Adams, R. J., & Wu, M. L. (2007). The mixed-coefficients multinomial logit model. A generalized form of the Rasch model. In M. von Davier & C. H. Carstensen (Eds.): *Multivariate and mixture distribution Rasch models: Extensions and applications* (pp. 55-76). New York: Springer.

Examples

```
#############################################################################
# EXAMPLE 1: Dichotomous data sim.rasch
#############################################################################

data(sim.rasch)
# estimate Rasch model
mod1 <- tam.mml(resp=sim.rasch)
# item fit
fit1 <- tam.fit( mod1 )
  ##   > print( round( fit1 ,3 ))
  ##       Outfit Outfit_t Infit Infit_t
  ##   I1   0.951   -1.549 0.994  -0.135
  ##   I2   1.033    1.040 1.028   0.755
  ##   I3   1.012    0.387 1.009   0.273
  ##   I4   1.041    1.296 1.057   1.614

## Not run:
#############################################################################
# EXAMPLE 2: Partial credit model data.gpcm
#############################################################################

data( data.gpcm )
# estimate partial credit model in ConQuest parametrization 'item+item*step'
mod2 <- tam.mml( resp=data.gpcm , irtmodel="PCM2" )
summary(mod2)
# estimate item fit
fit2 <- tam.fit(mod2)
  ## > round( fit2 ,3 )
  ##                 Outfit Outfit_t Infit Infit_t
  ##   Comfort        0.968   -0.426 0.967  -0.375
  ##   Work           1.061    0.867 1.054   0.695
  ##   Benefit        0.984   -0.196 0.987  -0.165
  ##   Comfort_step1  0.932   -0.946 0.955  -0.728
  ##   [...]
#  => The first three rows of the data frame correspond to the fit statistics
#     of first three items Comfort, Work and Benefit.

#############################################################################
# SIMULATED EXAMPLE 3: Fit statistic testing for local independence
#############################################################################

#generate data with local DEPENDENCE and User-defined fit statistics
set.seed(4888)
I <- 40 # 40 items
N <- 1000       # 1000 persons

delta <- seq(-2,2, len=I)
theta <- rnorm(N, 0, 1)
# simulate data
prob <- plogis(outer(theta, delta, "-"))
rand <- matrix(runif(N*I), nrow=N, ncol=I)
```

```
resp <- 1*(rand < prob)
colnames(resp) <- paste("I" , 1:I, sep="")

#Introduce some local dependence
for (item in c(10, 20, 30)){
 #  20
 #are made equal to the previous item
  row <- round(runif(0.2*N)*N + 0.5)
  resp[row, item+1] <- resp[row, item]
}

#run TAM
mod1 <- tam(resp)

#User-defined fit design matrix
F <- array(0, dim=c(dim(mod1$A)[1], dim(mod1$A)[2], 6))
F[,,1] <- mod1$A[,,10] + mod1$A[,,11]
F[,,2] <- mod1$A[,,12] + mod1$A[,,13]
F[,,3] <- mod1$A[,,20] + mod1$A[,,21]
F[,,4] <- mod1$A[,,22] + mod1$A[,,23]
F[,,5] <- mod1$A[,,30] + mod1$A[,,31]
F[,,6] <- mod1$A[,,32] + mod1$A[,,33]
fit <- tam.fit(mod1, FitMatrix=F)
fit

#############################################################################
# SIMULATED EXAMPLE 4: Fit statistic testing for items with differing slopes
#############################################################################

#*** simulate data
library(sirt)
set.seed(9875)
N <- 2000
I <- 20
b <- sample( seq( -2 , 2 , length=I ) )
a <- rep( 1, I )
# create some misfitting items
a[c(1,3)] <- c(.5 , 1.5 )
# simulate data
dat <- sirt::sim.raschtype( rnorm(N) , b=b , fixed.a=a )
#*** estimate Rasch model
mod1 <- tam.mml(resp=dat)
#*** assess item fit by infit and outfit statistic
fit1 <- tam.fit( mod1 )
round( cbind( "b"=mod1$item$AXsi_.Cat1 , fit1 )[1:7,], 3 )

#*** compute item fit statistic in mirt package
library(mirt)
library(sirt)
mod1c <- mirt::mirt( dat , model=1 , itemtype="Rasch" , verbose=TRUE)
print(mod1c)                      # model summary
sirt::mirt.wrapper.coef(mod1c)    # estimated parameters
fit1c <- mirt::itemfit(mod1c , method="EAP")    # model fit in mirt package
```

```
# compare results of TAM and mirt
dfr <- cbind( "TAM"=fit1 , "mirt"=fit1c[,-c(1:2)] )

# S-X2 item fit statistic (see also the output from mirt)
library(CDM)
sx2mod1 <- CDM::itemfit.sx2( mod1 )
summary(sx2mod1)

# compare results of CDM and mirt
sx2comp <-  cbind( sx2mod1$itemfit.stat[ , c("S-X2" , "p") ]  ,
                   dfr[  , c("mirt.S_X2" , "mirt.p.S_X2") ] )
round( sx2comp , 3 )

## End(Not run)
```

`tam.jml` *Joint Maximum Likelihood Estimation*

Description

This function estimate unidimensional item response models with joint maximum likelihood (JML, see e.g. Linacre, 1994).

Usage

```
tam.jml(resp, group = NULL, adj = .3, disattenuate = FALSE, bias = TRUE,
   xsi.fixed = NULL, xsi.inits = NULL, theta.fixed=NULL, A = NULL, B = NULL, Q = NULL,
    ndim = 1, pweights = NULL, control = list())

tam.jml2(resp, group = NULL, adj = .3 , disattenuate = FALSE, bias = TRUE,
    xsi.fixed = NULL, xsi.inits = NULL, A = NULL, B = NULL, Q = NULL,
    ndim = 1, pweights = NULL, control = list())

## S3 method for class 'tam.jml'
summary(object,file=NULL,...)
```

Arguments

`resp`	A matrix of item responses. Missing responses must be declared as `NA`.
`group`	An optional vector of group identifier
`disattenuate`	An optional logical indicating whether the person parameters should be disattenuated due to unreliability? The disattenuation is conducted by applying the Kelley formula.
`adj`	Adjustment constant which is subtracted or added to extreme scores (score of zero or maximum score. The default is a value of 0.3
`bias`	A logical which indicates if JML bias should be reduced by multiplying item parameters by the adjustment factor of $(I-1)/I$

xsi.fixed	An optional matrix with two columns for fixing some of the basis parameters ξ of item intercepts. 1st column: Index of ξ parameter, 2nd column: Fixed value of ξ parameter
xsi.inits	An optional vector of initial ξ parameters. Note that all parameters must be specified and the vector is not of the same format as xsi.fixed.
theta.fixed	Matrix for fixed person parameters $theta$. The first column includes the index whereas the second column includes the fixed value. This argument only applies to tam.jml.
A	A design array A for item category intercepts. For item i and category k, the threshold is specified as $\sum_j a_{ikj}\xi_j$.
B	A design array for scoring item category responses. Entries in B represent item loadings on abilities θ.
Q	A Q-matrix which defines loadings of items on dimensions.
ndim	Number of dimensions in the model. The default is 1.
pweights	An optional vector of person weights.
control	A list of control arguments. See tam.mml for more details.
object	Object of class tam.jml (only for summary.tam function)
file	A file name in which the summary output will be written (only for summary.tam.jml function)
...	Further arguments to be passed

Details

The function tam.jml2 is just a bit more efficient (for most applications) implementation than tam.jml.

Value

A list with following entries

item	Data frame with item parameters
xsi	Vector of item parameters ξ
errorP	Standard error of item parameters ξ
theta	MLE in final step
errorWLE	Standard error of WLE
WLE	WLE in last iteration
WLEreliability	WLE reliability
PersonScores	Scores for each person (sufficient statistic)
ItemScore	Sufficient statistic for each item parameter
PersonMax	Maximum person score
ItemMax	Maximum item score
deviance	Deviance

deviance.history	Deviance history in iterations
resp	Original data frame
resp.ind	Response indicator matrix
group	Vector of group identifiers (if provided as an argument)
pweights	Vector of person weights
A	Design matrix A of item intercepts
B	Loading (or scoring) matrix B
nitems	Number of items
maxK	Maximum number of categories
nstud	Number of persons in resp
resp.ind.list	Like resp.ind, only in the format of a list
xsi.fixed	Fixed ξ item parameters
control	Control list
...	

Note

This joint maximum likelihood estimation procedure should be compatible with Winsteps and Facets software, see also http://www.rasch.org/software.htm.

References

Linacre, J. M. (1994). *Many-Facet Rasch Measurement*. Chicago: MESA Press.

See Also

For estimating the same class of models with marginal maximum likelihood estimation see tam.mml.

Examples

```
#############################################################################
# EXAMPLE 1: Dichotomous data
#############################################################################
data(sim.rasch)
resp <- sim.rasch[1:700 , seq( 1 , 40 , len=10)  ]  # subsample
# estimate the Rasch model with JML (function 'tam.jml')
mod1a <- tam.jml(resp=resp)
summary(mod1a)
itemfit <- tam.fit(mod1a)$fit.item

# the same model but using function 'tam.jml2'
mod1b <- tam.jml2(resp=resp)
summary(mod1b)
itemfit <- tam.fit(mod1b)$fit.item

# compare results with Rasch model estimated by MML
```

```
mod1c <- tam.mml(resp=resp )
# plot estimated parameters
plot( mod1b$xsi , mod1c$xsi$xsi , pch=16 ,
    xlab= expression( paste( xi[i] , " (JML)" )) ,
    ylab= expression( paste( xi[i] , " (MML)" )) ,
    main="Item Parameter Estimate Comparison")
lines( c(-5,5) , c(-5,5) , col="gray" )

# Now, the adjustment pf .05 instead of the default .3 is used.
mod1d <- tam.jml2(resp=resp , adj=.05)
# compare item parameters
round( rbind( mod1b$xsi , mod1d$xsi ) , 3 )
  ##          [,1]   [,2]   [,3]   [,4]   [,5]  [,6]  [,7]  [,8]  [,9] [,10]
  ##   [1,] -2.076 -1.743 -1.217 -0.733 -0.338 0.147 0.593 1.158 1.570 2.091
  ##   [2,] -2.105 -1.766 -1.233 -0.746 -0.349 0.139 0.587 1.156 1.574 2.108

# person parameters for persons with a score 0, 5 and 10
pers1 <- data.frame( "score_adj0.3"= mod1b$PersonScore , "theta_adj0.3"= mod1b$theta ,
           "score_adj0.05"= mod1d$PersonScore , "theta_adj0.05"= mod1d$theta  )
round( pers1[ c(698, 683, 608) , ] ,3  )
  ##       score_adj0.3 theta_adj0.3 score_adj0.05 theta_adj0.05
  ##   698          0.3       -4.404          0.05        -6.283
  ##   683          5.0       -0.070          5.00        -0.081
  ##   608          9.7        4.315          9.95         6.179

## Not run:
#*** Models in which some item parameters are fixed
xsi.fixed <- cbind( c(1,3,9,10) , c(-2 , -1.2 , 1.6 , 2 ) )
mod1e <- tam.jml2( resp=resp , xsi.fixed = xsi.fixed )
summary(mod1e)
# the same fixing in tam.jml
mod1f <- tam.jml( resp=resp , xsi.fixed = xsi.fixed )
summary(mod1f)

#*** Model in which also some person parameters theta are fixed
# fix theta parameters of persons 2, 3, 4 and 33 to values -2.9, ...
theta.fixed <- cbind( c(2,3,4,33) , c( -2.9 , 4 , -2.9 , -2.9 ) )
mod1g <- tam.jml( resp=resp , xsi.fixed = xsi.fixed , theta.fixed=theta.fixed )
# look at estimated results
ind.person <- c( 1:5 , 30:33 )
cbind( mod1g$WLE , mod1g$errorWLE )[ind.person,]

#############################################################################
# EXAMPLE 2: Partial credit model
#############################################################################

data(data.gpcm)
# the same model but using function 'tam.jml2'
mod2 <- tam.jml2(resp=data.gpcm)
mod2$xsi     # extract item parameters
summary(mod2)
tam.fit(mod2) # item and person infit/outfit statistic
```

```
#############################################################################
# EXAMPLE 3: Facet model estimation using joint maximum likelihood
#            data.ex10; see also Example 10 in tam.mml
#############################################################################

data(data.ex10)
dat <- data.ex10
  ## > head(dat)
  ##  pid rater I0001 I0002 I0003 I0004 I0005
  ##    1     1     0     1     1     0     0
  ##    1     2     1     1     1     1     0
  ##    1     3     1     1     1     0     1
  ##    2     2     1     1     1     0     1
  ##    2     3     1     1     0     1     1

facets <- dat[ , "rater" , drop=FALSE ] # define facet (rater)
pid <- dat$pid      # define person identifier (a person occurs multiple times)
resp <- dat[ , -c(1:2) ]        # item response data
formulaA <- ~ item * rater      # formula

# use MML function only to restructure data and input obtained design matrices
# and processed response data to tam.jml (-> therefore use only 2 iterations)
mod3a <- tam.mml.mfr( resp=resp , facets=facets , formulaA = formulaA ,
             pid=dat$pid ,  control=list(maxiter=2) )

# use modified response data mod3a$resp and design matrix mod3a$A
resp1 <- mod3a$resp
# JML with 'tam.jml'
mod3b <- tam.jml( resp=resp1 , A=mod3a$A , control=list(maxiter=200) )
# 'tam.jml2'
mod3c <- tam.jml2( resp=resp1 , A=mod3a$A , control=list(maxiter=200) )

#############################################################################
# EXAMPLE 4: Multi faceted model with some anchored item and person parameters
#############################################################################

data(data.exJ03)
resp <- data.exJ03$resp
X <- data.exJ03$X

#*** (0) preprocess data with tam.mml.facets
mod0 <- tam.mml.mfr( resp=resp , facets = X , pid = X$rater ,
                formulaA = ~ leader + item + step ,
                control=list(maxiter=2) )
summary(mod0)

#*** (1) estimation with tam.jml (no parameter fixings)

# extract processed data and design matrix from tam.mml.mfr
resp1 <- mod0$resp
A1 <- mod0$A
# estimate model with tam.jml
mod1 <- tam.jml( resp=resp1 , A=A1 , control=list( Msteps=4 , maxiter=100 ) )
```

```
summary(mod1)

#*** (2) fix some parameters (persons and items)

# look at indices in mod1$xsi
mod1$xsi
# fix step parameters
xsi.index1 <- cbind( 21:25 , c( -2.44 , 0.01 , -0.15 , 0.01 ,  1.55 ) )
# fix some item parameters of items 1,2,3,6 and 13
xsi.index2 <- cbind( c(1,2,3,6,13) , c(-2,-1,-1,-1.32 , -1 ) )
xsi.index <- rbind( xsi.index1 , xsi.index2 )
# fix some theta parameters of persons 1, 15 and 20
theta.fixed <- cbind(  c(1,15,20) , c(0.4 , 1 , 0 ) )
# estimate model
mod2 <- tam.jml( resp=resp1 , A=A1 , xsi.fixed=xsi.fixed , theta.fixed=theta.fixed ,
            control=list( Msteps=4 , maxiter=100 ) )
summary(mod2)
cbind( mod2$WLE , mod2$errorWLE )

## End(Not run)
```

---

tam.mml *Test Analysis Modules: Marginal Maximum Likelihood Estimation*

Description

Modules for psychometric test analysis demonstrated with the help of artificial example data. The package includes MML and JML estimation of uni- and multidimensional IRT (Rasch, 2PL, Generalized Partial Credit, Rating Scale, Multi Facets, Nominal Item Response) models, fit statistic computation, standard error estimation, as well as plausible value imputation and weighted likelihood estimation of ability.

Usage

```
tam(resp , irtmodel ="1PL" , formulaA=NULL, ...)

tam.mml(resp, Y = NULL, group = NULL, irtmodel = "1PL", formulaY = NULL,
    dataY = NULL, ndim = 1,  pid = NULL, xsi.fixed = NULL, xsi.inits = NULL,
    beta.fixed = NULL, beta.inits = NULL, variance.fixed = NULL,
    variance.inits = NULL, est.variance = FALSE, A = NULL, B = NULL,
    B.fixed = NULL,  Q = NULL,  est.slopegroups = NULL, E = NULL,
    pweights = NULL, userfct.variance = NULL , variance.Npars = NULL ,
    control = list())

tam.mml.2pl(resp, Y = NULL, group = NULL, irtmodel = "2PL", formulaY = NULL,
    dataY = NULL, ndim = 1, pid = NULL, xsi.fixed = NULL, xsi.inits = NULL,
    beta.fixed = NULL, beta.inits = NULL, variance.fixed = NULL,
    variance.inits = NULL, est.variance = FALSE, A = NULL, B = NULL,
    B.fixed = NULL, Q = NULL, est.slopegroups = NULL, E = NULL,
```

```
    pweights = NULL, userfct.variance = NULL , variance.Npars = NULL ,
    control = list())

tam.mml.mfr(resp, Y = NULL, group = NULL, irtmodel = "1PL", formulaY = NULL,
    dataY = NULL, ndim = 1, pid = NULL, xsi.fixed = NULL, xsi.inits = NULL,
    beta.fixed = NULL, beta.inits = NULL, variance.fixed = NULL ,
    variance.inits = NULL , est.variance = FALSE , formulaA=~item+item:step,
    constraint="cases", A=NULL , B=NULL ,  B.fixed = NULL , Q=NULL ,
    facets=NULL, est.slopegroups=NULL , E = NULL , pweights = NULL ,
    control = list() , delete.red.items=TRUE )

## S3 method for class 'tam'
summary(object,file=NULL,...)

## S3 method for class 'tam.mml'
summary(object,file=NULL,...)
```

Arguments

resp	Data frame with polytomous item responses $k = 0, ..., K$. Missing responses must be declared as NA.
Y	A matrix of covariates in latent regression. Note that the intercept is automatically included as the first predictor.
group	An optional vector of group identifiers
irtmodel	For fixed item slopes (in tam.mml) options include PCM (partial credit model), PCM2 (partial credit model with ConQuest parametrization 'item+item*step' and RSM (rating scale model; the ConQuest parametrization 'item+step'). For estimated item slopes (in tam.mml.2pl) options are 2PL (all slopes of item categories are estimated; Nominal Item Response Model), GPCM (generalized partial credit model in which every item gets one and only slope parameter per dimension) and 2PL.groups (subsets of items get same item slope estimates) and a design matrix E on item slopes in the generalized partial credit model (GPCM.design, see Examples). Note that item slopes can not be estimated with faceted designs using the function tam.mml.mfr. However, it is easy to use pre-specified design matrices and apply some restrictions to tam.mml.2pl (see Example 14, Model 3).
formulaY	An R formula for latent regression. Transformations of predictors in Y (included in dataY) can be easily spcified, e. g. female*race or I(age^2).
dataY	An optional data frame with possible covariates Y in latent regression. This data frame will be used if an R formula in formulaY is specified.
ndim	Number of dimensions (is not needed to determined by the user)
pid	An optional vector of person identifiers
xsi.fixed	A matrix with two columns for fixing ξ parameters. 1st column: index of ξ parameter, 2nd column: fixed value
xsi.inits	A matrix with two columns (in the same way defined as in xsi.fixed with initial value for ξ.

`beta.fixed` A matrix with three columns for fixing regression coefficients. 1st column: Index of Y value, 2nd column: dimension, 3rd column: fixed β value.
If no constraints should be inposed on β, then set `beta.fixed=FALSE` (see Example 2, Model `2_4`).

`beta.inits` A matrix (same format as in `beta.fixed`) with initial β values

`variance.fixed` An optional matrix with three columns for fixing entries in covariance matrix: 1st column: dimension 1, 2nd column: dimension 2, 3rd column: fixed value

`variance.inits` Initial covariance matrix in estimation. All matrix entries have to be specified and this matrix is NOT in the same format like `variance.inits`.

`est.variance` Should the covariance matrix be estimated? This argument applies to estimated item slopes in `tam.mml.2pl`. The default is `FALSE` which means that latent variables (in the first group) are standardized in 2PL estimation.

`constraint` Set sum constraint for parameter identification for `items` or `cases` (applies to `tam.mml.mfr`)

`A` An optional array of dimension $I \times (K+1) \times N_\xi$. Only ξ parameters are estimated, entries in A only correspond to the design.

`B` An optional array of dimension $I \times (K+1) \times D$. In case of `tam.mml.2pl` entries of the B matrix can be estimated.

`B.fixed` An optional matrix with four columns for fixing B matrix entries in 2PL estimation. 1st column: item index, 2nd column: category, 3rd column: dimension, 4th column: fixed value.

`Q` An optional $I \times D$ matrix (the Q-matrix) which specifies the loading structure of items on dimensions.

`est.slopegroups`
A vector of integers of length I for estimating item slope parameters of item groups. This function only applies to the generalized partial credit model (`irtmodel="2PL.groups"`).

`E` An optional design matrix for estimating item slopes in the generalized partial credit model (`irtmodel="GPCM.design"`)

`pweights` An optional vector of person weights

`formulaA` Design formula (only applies to `tam.mml.mfr`). See Example 8.

`facets` A data frame with facet entries (only applies to `tam.mml.mfr`)

`userfct.variance`
Optional user customized function for variance speicification (See Simulated Example 17).

`variance.Npars` Number of estimated parameters of variance matrix if a `userfct.variance` is provided.

`control` A list of control arguments for the algorithm:

```
list( nodes = seq(-6,6,len=21) , snodes = 0 , QMC=TRUE,
convD = .001 ,conv = .0001 , convM = .0001 , Msteps = 4 ,
maxiter = 1000 , max.increment = 1 ,
min.variance = .001 , progress = TRUE , ridge=0 ,
seed = NULL , xsi.start0=FALSE, , increment.factor=1 ,
```

fac.oldxsi=0)

nodes: the discretized θ nodes for numerical integration

snodes: number of simulated θ nodes for stochastic integration. If snodes=0, numerical integration is used.

QMC: A logical indicating whether quasi Monte Carlo integration (Pan & Thomas, 2007) should be used. The default is TRUE. Quasi Monte Carlo integrationn is a nonstochastic integration approach but prevents the very demanding numeric integration using Gaussian quadrature. In case of QMC=FALSE, "ordinary" stochastic integration is used (see the section *Integration* in Details).

convD: Convergence criterion for deviance

conv: Convergence criterion for item parameters and regression coefficients

convM: Convergence criterion for item parameters within each M step

Msteps: Number of M steps for item parameter estimation

maxiter: Maximum number of iterations

max.increment: Maximum increment for item parameter change for every iteration

min.variance: Minimum variance to be estimated during iterations.

progress: A logical indicating whether computation progress should be displayed at R console

ridge: A numeric value or a vector of ridge parameter(s) for the latent regression which is added to the covariance matrix $Y'Y$ of predictors in the diagonal.

seed: An optional integer defining the simulation seed (important for reproducible results for stochastic integration)

xsi.start0: A logical indicating whether all ξ starting values should be initially set to zero.

increment.factor: A value (larger than one) which defines the extent of the decrease of the maximum increment of item parameters in every iteration. The maximum increment in iteration iter is defined as max.increment*increment.factor^(-iter) where max.increment=1. Using a value larger than 1 helps to reach convergence in some non-converging analyses (see Example 12).

fac.oldxsi: An optional numeric value f between 0 and 1 which defines the weight of parameter values in previous iteration. If ξ_t denotes a parameter update in iteration t, ξ_{t-1} is the parameter value of iteration $t-1$, then the modified parameter value is defined as $\xi_t^* = (1-f) \cdot \xi_t + f \cdot \xi_{t-1}$. Especially in cases where the deviance increases, setting the parameter larger than 0 (maybe .4 or .5) is helpful in stabilizing the algorithm (see Example 15).

delete.red.items

An optional logical indicating whether redundant generalized items (with no observations) should be eliminated.

object Object of class tam or tam.mml (only for summary.tam functions)

file A file name in which the summary output will be written (only for summary.tam functions)

... Further arguments to be passed

Details

The multidimensional item response model in **TAM** is described in Adams, Wilson and Wu (1997) or Adams and Wu (2007).

The data frame `resp` contains item responses of N persons (in rows) at I items (in columns), each item having at most K categories $k = 0, ..., K$. The item reponse model has D dimensions of the θ ability vector and can be written as

$$P(X_{pi} = k|\theta_p) \propto exp(b_{ik}\theta_p + a_{ik}\xi)$$

The symbol $\propto$ means that response probabilities are normalized such that $\sum_k P(X_{pi} = k|\theta_p) = 1$.

Item category thresholds for item i in category k are written as a linear combination $a_i\xi$ where the vector ξ of length N_ξ contains generalized item parameters and $A = (a_{ik})_{ik} = (a_i)_i$ is a three-dimensional design array (specified in `A`).

The scoring vector b_{ik} contains the fixed (in `tam.mml`) or estimated (in `tam.mml.2pl`) scores of item i in category k on dimension d.

For `tam.mml.2pl` and `irtmodel="GPCM.design"`, item slopes a_i can be written as a linear combination $a_i = (E\gamma)_i$ of basis item slopes which is an analogue of the LLTM for item slopes (see Example 7; Embretson, 1999).

The latent regression model regresses the latent trait θ_p on covariates Y which results in

$$\theta_p = Y\beta + \epsilon_p, \epsilon_p \sim N_D(0, \Sigma)$$

Where β is a N_Y times D matrix of regression coefficients for N_Y covariates in Y.

The multiple group model for groups $g = 1, ..., G$ is only implemented for unidimensional item response models. In this case, variance heterogeneity is allowed

$$\theta_p = Y\beta + \epsilon_p, \epsilon_p \sim N(0, \sigma^2_g)$$

Integration: Uni- and multidimensional integrals are approximated by posing a uni- or multivariate normality assumption. The default is Gaussian quadrature with nodes defined in `control$nodes`. For D-dimensional IRT models, the D-dimensional cube consisting of the vector `control$nodes` in all dimensions is used. If the user specifies `control$snodes` with a value larger than zero, then Quasi-Monte Carlo integration (Pan & Thomas, 2007) with `control$snodes` is used (because `control$QMC = TRUE` is set by default). If `control$QMC=FALSE` is specified, then stochastic (Monte Carlo) integration is employed with `control$snodes` stochastic nodes.

Value

A list with following entries:

`xsi`	Vector of ξ parameter estimates and their corresponding standard errors
`xsi.facets`	Data frame of ξ parameters and corresponding constraints for multifacet models
`beta`	Matrix of β regression coefficient estimates
`variance`	Covariance matrix. In case of multiple groups, it is a vector indicating heteroscedastic variances
`item`	Data frame with item parameters

person	Matrix with person parameter estimates. EAP is the mean of the posterior distribution and SD.EAP the corresponding standard deviation
pid	Vector of person identifiers
EAP.rel	EAP reliability
post	Posterior distribution for item response pattern
rprobs	A three-dimensional array with estimated response probabilities (dimensions are items $\times$ categories $\times$ theta length)
itemweight	Matrix of item weights
theta	Theta grid
n.ik	Array of expected counts: theta class $\times$ item $\times$ category $\times$ group
pi.k	Marginal trait distribution at grid theta
Y	Matrix of covariates
resp	Original data frame
resp.ind	Response indicator matrix
group	Group identifier
G	Number of groups
formulaY	Formula for latent regression
dataY	Data frame for latent regression
pweights	Person weights
time	Computation time
A	Design matrix A for ξ parameters
B	Fixed or estimated loading matrix
se.B	Standard errors of B parameters
nitems	Number of items
maxK	Maximum number of categories
AXsi	Estimated item intercepts $a_{ik}\xi$
AXsi_	Estimated item intercepts -$a_{ik}\xi$. Note that in summary.tam, the parameters AXsi_ are displayed.
se.AXsi	Standard errors of $a_{ik}\xi$ parameters
nstud	Number of persons
resp.ind.list	List of response indicator vectors
hwt	Individual posterior distribution
like	Individual likelihood
ndim	Number of dimensions
xsi.fixed	Fixed ξ parameters
B.fixed	Fixed loading parameters (only applies to tam.mml.2pl)
est.slopegroups	An index vector of item groups of common slope parameters (only applies to tam.mml.2pl)

`E`	Design matrix for estimated item slopes in the generalized partial credit model (only applies to `tam.mml.2pl` in case of `irtmodel="GPCM.design"`)
`basispar`	Vector of γ parameters of the linear combination $a_i = (E\gamma)_i$ for item slopes (only applies to `tam.mml.2pl` in case of `irtmodel='GPCM.design'`)
`formulaA`	Design formula (only applies to `tam.mml.mfr`)
`facets`	Data frame with facet entries (only applies to `tam.mml.mfr`)
`variance.fixed`	Fixed covariance matrix
`nnodes`	Number of theta nodes
`deviance`	Final deviance
`ic`	Vector with information criteria
`deviance.history`	Deviance history in iterations
`control`	List of control arguments
`...`	Further values

Note

For more than three dimensions, quasi-Monte Carlo or stochastic integration is recommended because otherwise problems in memory allocation and large computation time will result. Choose in `control` a suitable value of the number of quasi Monte Carlo or stochastic nodes `snodes` (say, larger than 1000). See Pan and Thompson (2007) for more details.

In faceted models (`tam.mml.mfr`), parameters which cannot be estimated are fixed to `99`.

References

Adams, R. J., Wilson, M., & Wu, M. (1997). Multilevel item response models: An approach to errors in variables regression. *Journal of Educational and Behavioral Statistics*, **22**, 47-76.

Adams, R. J., & Wu, M. L. (2007). The mixed-coefficients multinomial logit model. A generalized form of the Rasch model. In M. von Davier & C. H. Carstensen (Eds.): *Multivariate and mixture distribution Rasch models: Extensions and applications* (pp. 55-76). New York: Springer.

Embretson, S. E. (1999). Generating items during testing: Psychometric issues and models. *Psychometrika*, **64**, 407-433.

Pan, J., & Thompson, R. (2007). Quasi-Monte Carlo estimation in generalized linear mixed models. *Computational Statistics & Data Analysis*, **51**, 5765-5775.

Wu, M. L., Adams, R. J., Wilson, M. R. & Haldane, S. (2007). *ACER ConQuest Version 2.0*. Mulgrave. https://shop.acer.edu.au/acer-shop/group/CON3

See Also

See `data.cqc01` for more examples which is are similar to the ones in the ConQuest manual (Wu, Adams, Wilson & Haldane, 2007).

See `tam.jml` for joint maximum likelihood estimation.

Standard errors are estimated by a rather crude (but quick) approximation. Use `tam.se` for improved standard errors.

For model comparisons see anova.tam.

See tam2mirt (**sirt**) for converting tam objects into objects of class mirt in the **mirt** package.

logLik.tam, anova.tam

Examples

```
#############################################################################
# EXAMPLE 1: dichotomous data
# sim.rasch: 2000 persons, 40 items
#############################################################################
data(sim.rasch)

#***
# Rasch model (MML estimation)
mod1 <- tam.mml(resp=sim.rasch)
# extract item parameters
mod1$item    # item difficulties

## Not run:
# WLE estimation
wle1 <- tam.wle( mod1 )
# item fit
fit1 <- tam.fit(mod1)
# plausible value imputation
pv1 <- tam.pv(mod1)
# standard errors
se1 <- tam.se( mod1 )
# summary
summary(mod1)

#***
# Rasch model with fixed item difficulties from 'mod1'
xsi0 <- mod1$xsi$xsi
xsi.fixed <- cbind( 1:(length(xsi0)) , xsi0 )
        # define vector with fixed item difficulties
mod1a <- tam.mml( resp=sim.rasch , xsi.fixed=xsi.fixed )
summary(mod1a)

# Rasch model with initial xsi parameters for items 2 (item difficulty b=-1.8),
# item 4 (b=-1.6) and item 40 (b=2)
xsi.inits <- cbind( c(2,4,40) , c(-1.8,-1.6,2))
mod1b <- tam.mml( resp=sim.rasch , xsi.inits=xsi.inits )

#***
# 1PL estimation with sum constraint on item difficulties
dat <- sim.rasch
# modify A design matrix to include the sum constraint
des <- designMatrices(resp=dat)
A1 <- des$A[ , , - ncol(dat) ]
A1[ ncol(dat) ,2 , ] <- 1
A1[,2,]
# estimate model
```

```
mod1c <- tam.mml( resp=dat , A=A1 , beta.fixed=FALSE ,
           control=list(fac.oldxsi=.1) )
summary(mod1c)

# estimate constraint='items' using tam.mml.mfr
formulaA= ~ 0 + item
mod1d <- tam.mml.mfr( resp=dat , formulaA=formulaA ,
                     control=list(fac.oldxsi=.1) , constraint="items")
summary(mod1d)

# estimate constraint='items' using tam.mml.mfr
# long format response data
resp.long <- c(dat)

# pid and item facet specifications are necessary
#     Note, that we recommend the facet labels to be sortable in the same order that the
#     results are desired.
#     compare to: facets <- data.frame( "item"=rep(colnames(dat), each=nrow(dat)) )
pid <- rep(1:nrow(dat), ncol(dat))
itemnames <- paste0("I", sprintf(paste('%0', max(nchar(1:ncol(dat))), 'i', sep='' ),
                    c(1:ncol(dat)) ) )
facets   <- data.frame( "item"=rep(itemnames, each=nrow(dat)) )

mod1d2 <- tam.mml.mfr( resp=resp.long , formulaA=formulaA , control=list(fac.oldxsi=.1) ,
                       constraint="items" , facets=facets , pid=pid)
stopifnot( all(mod1d$xsi.facets$xsi == mod1d2$xsi.facets$xsi) )

## End(Not run)


#***
# 2PL model
mod2 <- tam.mml.2pl(resp=sim.rasch,irtmodel="2PL")

# extract item parameters
mod2$xsi    # item difficulties
mod2$B      # item slopes

## Not run:
# 2PL with fixed item difficulties and slopes from 'mod2'
xsi0 <- mod2$xsi$xsi
xsi.fixed <- cbind( 1:(length(xsi0)) , xsi0 )
        # define vector with fixed item difficulties
mod2a <- tam.mml( resp=sim.rasch , xsi.fixed=xsi.fixed ,
        B = mod2$B # fix slopes
            )
summary(mod2a)
mod2a$B     # inspect used slope matrix

#****
# constrained 2PL estimation
# estimate item parameters in different slope groups
# items 1-10, 21-30 group 1
```

```
# items 11-20 group 2 and items 31-40 group 3
est.slope <- rep(1,40)
est.slope[ 11:20 ] <- 2
est.slope[ 31:40 ] <- 3
mod3 <- tam.mml.2pl( resp=sim.rasch , irtmodel="2PL.groups" ,
               est.slopegroups = est.slope )
mod3$B
summary(mod3)

#############################################################################
# SIMULATED EXAMPLE 2: Unidimensional calibration with latent regressors
#############################################################################

# (1) simulate data
set.seed(6778)     # set simulation seed
N <- 2000          # number of persons
# latent regressors Y
Y <- cbind( rnorm( N , sd = 1.5) , rnorm(N , sd = .3 ) )
# simulate theta
theta <- rnorm( N ) + .4 * Y[,1] + .2 * Y[,2]  # latent regression model
# number of items
I <- 40
p1 <- plogis( outer( theta , seq( -2 , 2 , len=I ) , "-" ) )
# simulate response matrix
resp <- 1 * ( p1 > matrix( runif( N*I ) , nrow=N , ncol=I ) )
colnames(resp) <- paste("I" , 1:I, sep="")

# (2) estimate model
mod2_1 <- tam(resp=resp , Y=Y)
summary(mod2_1)

# (3) setting initial values for beta coefficients
#   beta_2 = .20 , beta_3 = .35 for dimension 1
beta.inits <- cbind( c(2,3) , 1 , c(.2, .35 ) )
mod2_2 <- tam(resp=resp , Y=Y , beta.inits=beta.inits)

# (4) fix intercept to zero and third coefficient to .3
beta.fixed <- cbind( c(1,3) , 1 , c(0, .3 ) )
mod2_3 <- tam(resp=resp , Y=Y , beta.fixed=beta.fixed )

# (5) same model but with R regression formula for Y
dataY <- data.frame(Y)
colnames(dataY) <- c("Y1","Y2")
mod2_4 <- tam(resp=resp , dataY=dataY , formulaY = ~ Y1+Y2 )
summary(mod2_4)

# (6) model with interaction of regressors
mod2_5 <- tam(resp=resp , dataY=dataY , formulaY = ~ Y1*Y2 )
summary(mod2_5)

# (7) no constraint on regressors (removing constraint from intercept)
mod2_6 <- tam(resp=resp , Y=Y , beta.fixed=FALSE )
```

```
#############################################################################
# SIMULATED EXAMPLE 3: Multiple group estimation
#############################################################################

# (1) simulate data
set.seed(6778)
N <- 3000
theta <- c( rnorm(N/2,mean=0,sd = 1.5) , rnorm(N/2,mean=.5,sd = 1)  )
I <- 20
p1 <- plogis( outer( theta , seq( -2 , 2 , len=I ) , "-" ) )
resp <- 1 * ( p1 > matrix( runif( N*I ) , nrow=N , ncol=I ) )
colnames(resp) <- paste("I" , 1:I, sep="")
group <- rep(1:2 , each=N/2 )

# (2) estimate model
mod3_1 <- tam.mml( resp ,  group = group )
summary(mod3_1)

#############################################################################
# SIMULATED EXAMPLE 4: Multidimensional estimation
# with two dimensional theta's - simulate some bivariate data,
# and regressors
# 40 items: first 20 items load on dimension 1,
#           second 20 items load on dimension 2
#############################################################################

# (1) simulate some data
set.seed(6778)
library(mvtnorm)
N <- 1000
Y <- cbind( rnorm( N ) , rnorm(N) )
theta <- rmvnorm( N,mean=c(0,0), sigma=matrix( c(1,.5,.5,1) , 2 , 2 ))
theta[,1] <- theta[,1] + .4 * Y[,1] + .2 * Y[,2]  # latent regression model
theta[,2] <- theta[,2] + .8 * Y[,1] + .5 * Y[,2]  # latent regression model
I <- 20
p1 <- plogis( outer( theta[,1] , seq( -2 , 2 , len=I ) , "-" ) )
resp1 <- 1 * ( p1 > matrix( runif( N*I ) , nrow=N , ncol=I ) )
p1 <- plogis( outer( theta[,2] , seq( -2 , 2 , len=I ) , "-" ) )
resp2 <- 1 * ( p1 > matrix( runif( N*I ) , nrow=N , ncol=I ) )
resp <- cbind(resp1,resp2)
colnames(resp) <- paste("I" , 1:(2*I), sep="")

# (2) define loading Matrix
Q <- array( 0 , dim = c( 2*I , 2 ))
Q[cbind(1:(2*I), c( rep(1,I) , rep(2,I) ))] <- 1

# (3) estimate models

# Rasch model: without regressors
mod4_1 <- tam.mml( resp=resp , Q=Q )

# Rasch model: set covariance between dimensions to zero
variance_fixed <- cbind( 1 , 2 , 0 )
```

```
mod4_2 <- tam.mml( resp=resp , Q=Q , variance.fixed = variance_fixed )
summary(mod4_2)

# 2PL model
mod4_3 <- tam.mml.2pl( resp=resp , Q=Q , irtmodel="2PL" )

# Rasch model with 2000 quasi monte carlo nodes and a maximum of 15 iterations
# -> nodes are useful for more than 3 or 4 dimensions
mod4_4 <- tam.mml( resp=resp , Q=Q , control=list(snodes=2000,maxiter=15) )

# Rasch model with 2000 stochastic nodes
mod4_5 <- tam.mml( resp=resp, Q=Q,control=list(snodes=2000,QMC=FALSE,maxiter=15))

# estimate two dimensional Rasch model with regressors
mod4_6 <- tam.mml( resp=resp , Y=Y , Q=Q )

############################################################################
# SIMULATED EXAMPLE 5: 2-dimensional estimation with within item dimensionality
############################################################################
library(mvtnorm)
# (1) simulate data
set.seed(4762)
N <- 2000 # 2000 persons
Y <- rnorm( N )
theta <- rmvnorm( N,mean=c(0,0), sigma=matrix( c(1,.5,.5,1) , 2 , 2 ))
I <- 10
# 10 items load on the first dimension
p1 <- plogis( outer( theta[,1] , seq( -2 , 2 , len=I ) , "-" ) )
resp1 <- 1 * ( p1 > matrix( runif( N*I ) , nrow=N , ncol=I ) )
# 10 items load on the second dimension
p1 <- plogis( outer( theta[,2] , seq( -2 , 2 , len=I ) , "-" ) )
resp2 <- 1 * ( p1 > matrix( runif( N*I ) , nrow=N , ncol=I ) )
# 20 items load on both dimensions
p1 <- plogis( outer( 0.5*theta[,1] + 1.5*theta[,2] , seq( -2 , 2 , len=2*I ) , "-" ) )
resp3 <- 1 * ( p1 > matrix( runif( N*2*I ) , nrow=N , ncol=2*I ) )
#Combine the two sets of items into one response matrix
resp <- cbind(resp1, resp2, resp3 )
colnames(resp) <- paste("I" , 1:(4*I), sep="")

# (2) define loading matrix
Q <- cbind(c(rep(1,10),rep(0,10),rep(1,20)), c(rep(0,10),rep(1,10),rep(1,20)))

# (3) model: within item dimensionality and 2PL estimation
mod5 <- tam.mml.2pl(resp, Q=Q, irtmodel="2PL" , control=list(maxiter=20) )
summary(mod5)

# item difficulties
mod5$item
# item loadings
mod5$B

############################################################################
# EXAMPLE 6: ordered data - Generalized partial credit model
```

```
#############################################################################
data( data.gpcm )

# Ex6.1: Nominal response model (irtmodel="2PL")
mod6_1 <- tam.mml.2pl( resp=data.gpcm , irtmodel="2PL" , control=list( maxiter=200) )
mod6_1$item # item intercepts
mod6_1$B    # for every category a separate slope parameter is estimated

# Ex6.2: Generalized partial credit model
mod6_2 <- tam.mml.2pl( resp= data.gpcm , irtmodel="GPCM" , control=list( maxiter=200) )
mod6_2$B    # joint slope parameter for all categories

# Ex6.3: some fixed entries of slope matrix B
# B: nitems x maxK x ndim
#   ( number of items x maximum number of categories x number of dimensions)
# set two constraints
B.fixed <- matrix( 0 , 2 , 4 )
# set second item, score of 2 (category 3), at first dimension to 2.3
B.fixed[1,] <- c(2,3,1,2.3)
# set third item, score of 1 (category 2), at first dimension to 1.4
B.fixed[2,] <- c(3,2,1,1.4)

# estimate item parameter with variance fixed (by default)
mod6_3 <- tam.mml.2pl( resp=data.gpcm , irtmodel="2PL" , B.fixed = B.fixed ,
                control=list( maxiter=200) )
mod6_3$B

# Ex 6.4: estimate the same model, but estimate variance
mod6_4 <- tam.mml.2pl( resp=data.gpcm , irtmodel="2PL" , B.fixed = B.fixed ,
               est.variance = TRUE , control=list( maxiter=350) )
mod6_4$B

# Ex 6.5: partial credit model
mod6_5 <- tam.mml( resp=data.gpcm ,control=list( maxiter=200) )
mod6_5$B

# Ex 6.6: partial credit model: Conquest parametrization 'item+item*step'
mod6_6 <- tam.mml( resp=data.gpcm , irtmodel="PCM2" )
summary(mod6_6)

# estimate mod6_6 applying the sum constraint of item difficulties
# modify design matrix of xsi paramters
A1 <- .A.PCM2(resp=data.gpcm )
A1[3,2:4,"Comfort"] <- 1:3
A1[3,2:4,"Work"] <- 1:3
A1 <- A1[ , , -3] # remove Benefit xsi item parameter
# estimate model
mod6_6b <- tam.mml( resp=data.gpcm , A=A1 , beta.fixed=FALSE )
summary(mod6_6b)

# estimate mod6_6 using tam.mml.mfr
mod6_6c <- tam.mml.mfr( resp=data.gpcm , formulaA= ~ 0 + item + item:step ,
    control=list(fac.oldxsi=.1)  , constraint="items" )
```

```
summary(mod6_6c)

# Ex 6.7: Rating scale model: Conquest parametrization 'item+step'
mod6_7 <- tam.mml( resp=data.gpcm , irtmodel="RSM" )
summary(mod6_7)

#***
# Ex 6.8: sum constraint on item difficulties
#         partial credit model: ConQuest parametrization 'item+item*step'
#         polytomous scored TIMMS data
#         compare to Example 16
#

data(data.timssAusTwn.scored)
dat <- data.timssAusTwn.scored[,1:11]

## > tail(sort(names(dat)),1) # constrained item
## [1] "M032761"

# modify design matrix of xsi paramters
A1 <- .A.PCM2( resp=dat )
# constrained item loads on every other main item parameter
# with opposing margin it had been loaded on its own main item parameter
A1["M032761",,setdiff(colnames(dat), "M032761")] <- -A1["M032761",,"M032761"]
# remove main item parameter for constrained item
A1 <- A1[ , , setdiff(dimnames(A1)[[3]],"M032761")]

# estimate model
mod6_8a <- tam.mml( resp=dat , A=A1 , beta.fixed=FALSE )
summary(mod6_8a)
# extract fixed item parameter for item M032761
## - sum(mod6_8a$xsi[setdiff(colnames(dat), "M032761"),"xsi"])

# estimate mod6_8a using tam.mml.mfr
## fixed a bug in 'tam.mml.mfr' for differing number of categories
## per item -> now a xsi vector with parameter fixings to values
## of 99 is used
mod6_8b <- tam.mml.mfr( resp=dat , formulaA= ~ 0 + item + item:step ,
                        control=list(fac.oldxsi=.1), constraint="items" )
summary(mod6_8b)

#############################################################################
# SIMULATED EXAMPLE 7: design matrix for slopes for the
#            generalized partial credit model
#############################################################################

# (1) simulate data from a model with a (item slope) design matrix E
set.seed(789)
I <- 42
b <- seq( -2 , 2 , len=I)
# create design matrix for loadings
E <- matrix( 0 , I , 5 )
E[ seq(1,I,3) , 1 ] <- 1
```

```
E[ seq(2,I,3) , 2 ] <- 1
E[ seq(3,I,3) , 3 ] <- 1
ind <- seq( 1, I , 2 ) ; E[ ind , 4 ] <- rep( c( .3 , -.2 ) , I )[ 1:length(ind) ]
ind <- seq( 2, I , 4 ) ; E[ ind , 5 ] <- rep( .15 , I )[ 1:length(ind) ]
E

# true basis slope parameters
lambda <- c( 1 , 1.2 , 0.8 , 1 , 1.1 )
# calculate item slopes
a <- E %*% lambda

N <- 4000
theta <- rnorm( N )
aM <- outer( rep(1,N) , a[,1] )
bM <- outer( rep(1,N) , b )
pM <- plogis( aM * ( matrix( theta , nrow=N , ncol=I  ) - bM ) )
dat <- 1 * ( pM > runif( N*I ) )
colnames(dat) <- paste("I" , 1:I , sep="")

mod7 <- tam.mml.2pl( resp = dat , irtmodel="GPCM.design" , E=E )
mod7$B

# recalculate estimated basis parameters
lm( mod7$B[,2,1] ~ 0+ as.matrix(E ) )
  ##   Call:
  ##   lm(formula = mod7$B[, 2, 1] ~ 0 + as.matrix(E))
  ##   Coefficients:
  ##   as.matrix(E)1  as.matrix(E)2  as.matrix(E)3  as.matrix(E)4  as.matrix(E)5
  ##          0.9904         1.1896         0.7817         0.9601         1.2132

#############################################################################
# EXAMPLE 8: Differential item functioning                                  #
#  A first example of a Multifaceted Rasch Model                            #
#  Facet is only female; 10 items are studied                               #
#############################################################################
data(data.ex08)

formulaA <- ~ item+female+item*female
# this formula is in R equivalent to 'item*female'
resp <- data.ex08[["resp"]]
facets <- as.data.frame( data.ex08[["facets"]] )

#***
# Model 8a: investigate gender DIF on all items
mod8a <- tam.mml.mfr( resp= resp , facets= facets , formulaA = formulaA )
summary(mod8a)

#***
# Model 8a 2: specification with long format response data
resp.long <- c( data.ex08[["resp"]] )
pid <- rep( 1:nrow(data.ex08[["resp"]]), ncol(data.ex08[["resp"]]) )

itemnames <- rep(colnames(data.ex08[["resp"]]), each=nrow(data.ex08[["resp"]]))
```

```
facets.long <- cbind( data.frame( "item"=itemnames ),
                 data.ex08[["facets"]][pid,,drop=F] )

mod8a_2 <- tam.mml.mfr( resp=resp.long , facets=facets.long , formulaA=formulaA, pid= pid )
stopifnot( all(mod8a$xsi.facets$xsi==mod8a_2$xsi.facets$xsi) )

#***
# Model 8b: Differential bundle functioning (DBF)
#   - investigate differential item functioning in item groups

# modify pre-specified design matrix to define 'appropriate' DBF effects
formulaA <- ~ item*female
des <- designMatrices.mfr( resp=resp , facets=facets , formulaA=formulaA)
A1 <- des$A$A.3d
# item group A: items 1-5
# item group B: items 6-8
# item group C: items 9-10
A1 <- A1[,,1:13]
dimnames(A1)[[3]][ c(12,13) ] <- c("A:female1" , "B:female1")
# item group A
A1[,2,12] <- 0
A1[c(1,5,7,9,11),2,12] <- -1
A1[c(1,5,7,9,11)+1,2,12] <- 1
# item group B
A1[,2,13] <- 0
A1[c(13,15,17),2,13] <- -1
A1[c(13,15,17)+1,2,13] <- 1
# item group C (define effect(A)+effect(B)+effect(C)=0)
A1[c(19,3),2,c(12,13)] <- 1
A1[c(19,3)+1,2,c(12,13)] <- -1
#   A1[,2,]   # look at modified design matrix
# estimate model
mod8b <- tam.mml( resp= des$gresp$gresp.noStep , A=A1 )
summary(mod8b)

#############################################################################
# EXAMPLE 9: Multifaceted Rasch Model
#############################################################################
data(sim.mfr) ; data(sim.facets)

# two way interaction item and rater
formulaA <- ~item+item:step + item*rater
mod9a <- tam.mml.mfr( resp=sim.mfr , facets=sim.facets , formulaA = formulaA )
mod9a$xsi.facets
summary(mod9a)

# three way interaction item, female and rater
formulaA <- ~item+item:step + female*rater + female*item*step
mod9b <- tam.mml.mfr( resp=sim.mfr , facets=sim.facets , formulaA = formulaA )
summary(mod9b)

#############################################################################
# EXAMPLE 10: Model with raters.
```

```
#   Persons are arranged in multiple rows which is indicated
#   by multiple person identifiers.
#############################################################################
data(data.ex10)
dat <- data.ex10
head(dat)
  ##     pid rater I0001 I0002 I0003 I0004 I0005
  ## 1     1     1     0     1     1     0     0
  ## 451   1     2     1     1     1     1     0
  ## 901   1     3     1     1     1     0     1
  ## 452   2     2     1     1     1     0     1
  ## 902   2     3     1     1     0     1     1

facets <- dat[ , "rater" , drop=FALSE ] # define facet (rater)
pid <- dat$pid      # define person identifier (a person occurs multiple times)
resp <- dat[ , -c(1:2) ]        # item response data
formulaA <- ~ item * rater      # formula

mod10 <- tam.mml.mfr( resp=resp , facets=facets , formulaA = formulaA , pid=dat$pid )
summary(mod10)

# estimate person parameter with WLE
wmod10 <- tam.wle( mod10 )

#***
# Model 10 2: specification with long format response data
resp.long=c(unlist( dat[ , -c(1:2) ] ))

pid <- rep( dat$pid, ncol(dat[ , -c(1:2) ]) )
itemnames <- rep(colnames(dat[ , -c(1:2) ]), each=nrow(dat[ , -c(1:2) ]))

# quick note: the 'trick' to use pid as the row index of the facet  (cf., used in Ex 8a_2)
# is not working here, since pid already occures multiple times in the original response data
facets <- cbind( data.frame("item"=itemnames),
                 dat[ rep(1:nrow(dat), ncol(dat[,-c(1:2)])), "rater" ,drop=F]
)

mod10_2 <- tam.mml.mfr( resp=resp.long , facets= facets , formulaA = formulaA, pid= pid )

stopifnot( all(mod10$xsi.facets$xsi==mod10_2$xsi.facets$xsi) )

#############################################################################
# EXAMPLE 11: Dichotomous data with missing and omitted responses
#############################################################################
data(data.ex11) ; dat <- data.ex11

#***
# Model 11a: Calibration (item parameter estimating) in which omitted
#            responses (code 9) are set to missing
dat1 <- dat[,-1]
dat1[ dat1 == 9 ] <- NA
# estimate Rasch model
mod11a <- tam.mml( resp= dat1 )
```

```
summary(mod11a)
# compute person parameters
wmod11a <- tam.wle( mod11a )

#***
# Model 11b: Scaling persons (WLE estimation) setting omitted
#            responses as incorrect and using fixed
#            item parameters form Model 11a

# set matrix with fixed item difficulties as the input
xsi1 <- mod11a$xsi # xsi output from Model 11a
xsi.fixed <- cbind( seq(1,nrow(xsi1) ) , xsi1$xsi )
# recode 9 to 0
dat2 <- dat[,-1]
dat2[ dat2 == 9 ] <- 0
# run Rasch model with fixed item difficulties
mod11b <- tam.mml( resp= dat2 , xsi.fixed=xsi.fixed )
summary(mod11b)
# WLE estimation
wmod11b <- tam.wle( mod11b )

#############################################################################
# EXAMPLE 12: Avoiding nonconvergence using the argument increment.factor
#############################################################################
data(data.ex12)
dat <- data.ex12

# non-convergence without increment.factor
mod1 <- tam.mml.2pl( resp=data.ex12 , control=list( maxiter=1000) )

# avoiding non-convergence with increment.factor=1.02
mod2 <- tam.mml.2pl( resp=data.ex12 ,
            control=list( maxiter=1000 , increment.factor=1.02) )
summary(mod1)
summary(mod2)

#############################################################################
# EXAMPLE 13: Longitudinal data 'data.long' from sirt package
#############################################################################
library(sirt)
data(data.long, package="sirt")
dat <- data.long
  ##   > colnames(dat)
  ##    [1] "idstud" "I1T1"   "I2T1"   "I3T1"   "I4T1"   "I5T1"   "I6T1"
  ##    [8] "I3T2"   "I4T2"   "I5T2"   "I6T2"   "I7T2"   "I8T2"

## item 1 to 6 administered at T1 and items 3 to 8 at T2
## items 3 to 6 are anchor items

#***
# Model 13a: 2-dimensional Rasch model assuming invariant item difficulties

# define matrix loadings
```

```
Q <- matrix(0,12,2)
colnames(Q) <- c("T1","T2")
Q[1:6,1] <- 1       # items at T1
Q[7:12,2] <- 1      # items at T2

# assume equal item difficulty of I3T1 and I3T2, I4T1 and I4T2, ...
# create draft design matrix and modify it
A <- designMatrices(resp=data.long[,-1])$A
dimnames(A)[[1]] <- colnames(data.long)[-1]
  ##   > str(A)
  ##    num [1:12, 1:2, 1:12] 0 0 0 0 0 0 0 0 0 0 ...
  ##    - attr(*, "dimnames")=List of 3
  ##     ..$ : chr [1:12] "Item01" "Item02" "Item03" "Item04" ...
  ##     ..$ : chr [1:2] "Category0" "Category1"
  ##     ..$ : chr [1:12] "I1T1" "I2T1" "I3T1" "I4T1" ...
A1 <- A[  ,  , c(1:6 , 11:12 ) ]
dimnames(A1)[[3]] <- substring( dimnames(A1)[[3]],1,2)
A1[7,2,3] <- -1     # difficulty(I3T1) = difficulty(I3T2)
A1[8,2,4] <- -1     # I4T1 = I4T2
A1[9,2,5] <- A1[10,2,6] <- -1
  ##   > A1[,2,]
  ##        I1 I2 I3 I4 I5 I6 I7 I8
  ##   I1T1 -1  0  0  0  0  0  0  0
  ##   I2T1  0 -1  0  0  0  0  0  0
  ##   I3T1  0  0 -1  0  0  0  0  0
  ##   I4T1  0  0  0 -1  0  0  0  0
  ##   I5T1  0  0  0  0 -1  0  0  0
  ##   I6T1  0  0  0  0  0 -1  0  0
  ##   I3T2  0  0 -1  0  0  0  0  0
  ##   I4T2  0  0  0 -1  0  0  0  0
  ##   I5T2  0  0  0  0 -1  0  0  0
  ##   I6T2  0  0  0  0  0 -1  0  0
  ##   I7T2  0  0  0  0  0  0 -1  0
  ##   I8T2  0  0  0  0  0  0  0 -1

# estimate model
# set intercept of second dimension (T2) to zero
beta.fixed <- cbind( 1 , 2 , 0 )
mod13a <- tam.mml( resp=data.long[,-1] , Q=Q , A=A1 , beta.fixed=beta.fixed)
summary(mod13a)

#***
# Model 13b: invariant item difficulties with zero mean item difficulty
#           of anchor items

A <- designMatrices(resp=data.long[,-1])$A
dimnames(A)[[1]] <- colnames(data.long)[-1]
A1 <- A[  ,  , c(1:5 , 11:12 ) ]
dimnames(A1)[[3]] <- substring( dimnames(A1)[[3]],1,2)
A1[7,2,3] <- -1     # difficulty(I3T1) = difficulty(I3T2)
A1[8,2,4] <- -1     # I4T1 = I4T2
A1[9,2,5] <- -1
A1[6,2,3] <- A1[6,2,4] <- A1[6,2,5] <- 1  # I6T1=-(I3T1+I4T1+I5T1)
```

```
A1[10,2,3] <- A1[10,2,4] <- A1[10,2,5] <- 1  # I6T2=-(I3T2+I4T2+I5T2)
A1[,2,]
  ##      I1 I2 I3 I4 I5 I7 I8
  ## I1T1 -1  0  0  0  0  0  0
  ## I2T1  0 -1  0  0  0  0  0
  ## I3T1  0  0 -1  0  0  0  0
  ## I4T1  0  0  0 -1  0  0  0
  ## I5T1  0  0  0  0 -1  0  0
  ## I6T1  0  0  1  1  1  0  0
  ## I3T2  0  0 -1  0  0  0  0
  ## I4T2  0  0  0 -1  0  0  0
  ## I5T2  0  0  0  0 -1  0  0
  ## I6T2  0  0  1  1  1  0  0
  ## I7T2  0  0  0  0  0 -1  0
  ## I8T2  0  0  0  0  0  0 -1

mod13b <- tam.mml( resp=data.long[,-1] , Q=Q , A=A1 , beta.fixed=FALSE)
summary(mod13b)

#***
# Model 13c: longitudinal polytomous data
#

# modifiy Items I1T1, I4T1, I4T2 in order to be trichotomous (codes: 0,1,2)
set.seed(42)
dat <- data.long
dat[(1:50),2] <- sample(c(0,1,2), 50, replace = TRUE)
dat[(1:50),5] <- sample(c(0,1,2), 50, replace = TRUE)
dat[(1:50),9] <- sample(c(0,1,2), 50, replace = TRUE)
  ##   > colnames(dat)
  ##    [1] "idstud" "I1T1"   "I2T1"   "I3T1"   "I4T1"   "I5T1"   "I6T1"
  ##    [8] "I3T2"   "I4T2"   "I5T2"   "I6T2"   "I7T2"   "I8T2"

## item 1 to 6 administered at T1, items 3 to 8 at T2
## items 3 to 6 are anchor items

# (1) define matrix loadings
Q <- matrix(0,12,2)
colnames(Q) <- c("T1","T2")
Q[1:6,1] <- 1       # items at T1
Q[7:12,2] <- 1      # items at T2

# (2) assume equal item difficulty of anchor items
#     create draft design matrix and modify it
A <- designMatrices(resp=dat[,-1])$A
dimnames(A)[[1]] <- colnames(dat)[-1]
  ## > str(A)
  ## num [1:12, 1:3, 1:15] 0 0 0 0 0 0 0 0 0 0 ...
  ## - attr(*, "dimnames")=List of 3
  ## ..$ : chr [1:12] "I1T1" "I2T1" "I3T1" "I4T1" ...
  ## ..$ : chr [1:3] "Category0" "Category1" "Category2"
  ## ..$ : chr [1:15] "I1T1_Cat1" "I1T1_Cat2" "I2T1_Cat1" "I3T1_Cat1" ...
```

```
# define matrix A
# Items 1 to 3 administered at T1, Items 3 to 6 are anchor items
# Item 7 to 8 administered at T2
# Item I1T1, I4T1, I4T2 are trichotomous (codes: 0,1,2)
A1 <- A[  ,  , c(1:8, 14:15) ]
dimnames(A1)[[3]] <- gsub("T1|T2", "",  dimnames(A1)[[3]])

# Modifications are shortened compared to Model 13 a, but are still valid
A1[7,,] <- A1[3,,]  # item 7, i.e. I3T2, loads on same parameters as
                    # item 3, I3T1
A1[8,,] <- A1[4,,]  # same for item 8 and item 4
A1[9,,] <- A1[5,,]  # same for item 9 and item 5
A1[10,,] <- A1[6,,] # same for item 10 and item 6
  ## > A1[8,,]
  ##           I1_Cat1 I1_Cat2 I2_Cat1 I3_Cat1 I4_Cat1 I4_Cat2 I5_Cat1 ...
  ## Category0       0       0       0       0       0       0       0
  ## Category1       0       0       0       0      -1       0       0
  ## Category2       0       0       0       0      -1      -1       0

# (3) estimate model
#     set intercept of second dimension (T2) to zero
beta.fixed <- cbind( 1 , 2 , 0 )
mod13c <- tam.mml( resp=dat[,-1] , Q=Q , A=A1 , beta.fixed=beta.fixed,
                   irtmodel = "PCM")
summary(mod13c)
wle.mod13c <- tam.wle(mod13c) # WLEs of dimension T1 and T2


#############################################################################
# EXAMPLE 14: Facet model with latent regression
#############################################################################
data( data.ex14 )
dat <- data.ex14

#***
# Model 14a: facet model
resp <- dat[ , paste0("crit",1:7,sep="") ] # item data
facets <- data.frame( "rater" = dat$rater )     # define facets
formulaA <- ~item+item*step + rater
mod14a <- tam.mml.mfr( resp , facets=facets , formulaA=formulaA , pid=dat$pid )
summary(mod14a)

#***
# Model 14b: facet model with latent regression
#   Note that dataY must correspond to rows in resp and facets which means
#   that there must be the same rows in Y for a person with multiple rows
#   in resp
dataY <- dat[ , c("X1","X2") ] # latent regressors
formulaY <- ~ X1+X2 # latent regression formula
mod14b <- tam.mml.mfr( resp , facets=facets , formulaA= formulaA ,
            dataY=dataY , formulaY=formulaY , pid=dat$pid)
summary(mod14b)
```

```
#***
# Model 14c: Multi-facet model with item slope estimation
# use design matrix and modified response data from Model 1
# item-specific slopes

resp1 <- mod14a$resp      # extract response data with generalized items
A <- mod14a$A             # extract design matrix for item intercepts
colnames(resp1)

# define design matrix for slopes
E <- matrix( 0 , nrow= ncol(resp1) , ncol=7 )
colnames(E) <- paste0("crit",1:7)
rownames(E) <- colnames(resp1)
E[ cbind( 1:(7*7) , rep(1:7,each=7) ) ] <- 1

mod14c <- tam.mml.2pl( resp=resp1 , A=A , irtmodel="GPCM.design" , E=E ,
        control=list(maxiter=100) )
summary(mod14c)

#############################################################################
# EXAMPLE 15: Coping with nonconvergent models
#############################################################################

data(data.ex15)
data <- data.ex15
# facet model 'group*item' is of interest

#***
# Model 15a:
mod15a <- tam.mml.mfr(resp = data[,-c(1:2)],facets=data[,"group",drop=FALSE],
    formulaA = ~ item + group*item , pid = data$pid , control=list(maxiter=50))
# See output:
  ##
  ##   Iteration 47     2013-09-10 16:51:39
  ##   E Step
  ##   M Step Intercepts   |----
  ##     Deviance = 75510.2868 | Deviance change: -595.0609
  ##   !!! Deviance increases!                                        !!!!
  ##   !!! Choose maybe fac.oldxsi > 0 and/or increment.factor > 1    !!!!
  ##     Maximum intercept parameter change: 0.925045
  ##     Maximum regression parameter change: 0
  ##     Variance:  0.9796  | Maximum change: 0.009226

#***
# Model 15b: Follow the suggestions of changing the default of fac.oldxsi and
#            increment.factor
mod15b <- tam.mml.mfr(resp = data[,-c(1:2)],facets=data[,"group",drop=FALSE],
            formulaA = ~ group*item , pid = data$pid ,
            control=list(maxiter= 50 , increment.factor=1.03 , fac.oldxsi=.4 ) )

#############################################################################
# EXAMPLE 16: Differential item function for polytomous items and
#       differing number of response options per item
```

```
#############################################################################

data(data.timssAusTwn.scored)
dat <- data.timssAusTwn.scored
# extract item response data
resp <- dat[ , sort(grep("M" , colnames(data.timssAusTwn.scored) , value=TRUE)) ]
# some descriptives
psych::describe(resp)
# define facets: 'cnt' is group identifier
facets <- data.frame( "cnt" = dat$IDCNTRY)
# create design matrices
des2 <- designMatrices.mfr2( resp=resp , facets=facets , formulaA = ~item*step + item*cnt)
# restructured data set: pseudoitem = item x country
resp2 <- des2$gresp$gresp.noStep
# A design matrix
A <- des2$A$A.3d
# redundant xsi parameters to be eliminated from design matrix
xsi.elim <- des2$xsi.elim
A <- A[ , , - xsi.elim[,2] ]
# extract loading matrix B
B <- des2$B$B.3d
# estimate model
mod1 <- tam.mml( resp=resp2 , A = A , B= B , control=list(maxiter=100) )
summary(mod1)
# The sum of all DIF parameters is set to zero. The DIF parameter for the last
# item is therefore obtained
xsi1 <- mod1$xsi
difxsi <- xsi1[ intersect( grep("cnt",rownames(xsi1)),grep("M03" ,rownames(xsi1))) , ]
- colSums(difxsi) # this is the DIF effect of the remaining item

#############################################################################
# SIMULATED EXAMPLE 17: Several multidimensional and subdimension models
#############################################################################

library(mirt)
#*** (1) simulate data in mirt package
set.seed(9897)
# simulate data according to the four-dimensional Rasch model
# variances
variances <- c( 1.45 , 1.74 , .86 , 1.48  )
# correlations
corrs <- matrix( 1 , 4 , 4 )
dd1 <- 1 ; dd2 <- 2 ; corrs[dd1,dd2] <- corrs[dd2,dd1] <- .88
dd1 <- 1 ; dd2 <- 3 ; corrs[dd1,dd2] <- corrs[dd2,dd1] <- .85
dd1 <- 1 ; dd2 <- 4 ; corrs[dd1,dd2] <- corrs[dd2,dd1] <- .87
dd1 <- 2 ; dd2 <- 3 ; corrs[dd1,dd2] <- corrs[dd2,dd1] <- .84
dd1 <- 2 ; dd2 <- 4 ; corrs[dd1,dd2] <- corrs[dd2,dd1] <- .90
dd1 <- 3 ; dd2 <- 4 ; corrs[dd1,dd2] <- corrs[dd2,dd1] <- .90
# covariance matrix
covar <- outer( sqrt( variances) , sqrt(variances) )*corrs
# item thresholds and item discriminations
d <- matrix( runif(40, -2 , 2 ) , ncol=1 )
a <- matrix(NA , nrow=40,ncol=4)
```

```
a[1:10,1] <- a[11:20,2] <- a[21:30,3] <- a[31:40,4] <- 1
# simulate data
dat <- mirt::simdata(a=a, d=d, N=1000, itemtype="dich", sigma = covar )
# define Q-matrix for testlet and subdimension models estimated below
Q <- matrix( 0 , nrow=40 , ncol=5 )
colnames(Q) <- c("g" , paste0( "subd"  , 1:4) )
Q[,1] <- 1
Q[1:10,2] <- Q[11:20,3] <- Q[21:30,4] <- Q[31:40,5] <- 1

# define maximum number of iterations and number of quasi monte carlo nodes
maxit <- 5  ; snodes <- 300 # this specification is only for speed reasons
# maxit <- 50 ; snodes <- 1500

#*****************
# Model 1: Rasch testlet model
#*****************

# define a user function for restricting the variance according to the
# Rasch testlet model
variance.fct1 <- function( variance ){
            ndim <- ncol(variance)
            variance.new <- matrix( 0 , ndim , ndim )
            diag(variance.new) <- diag(variance)
            variance <- variance.new
            return(variance)
                    }
variance.Npars <- 5 # number of estimated parameters in variance matrix
# estimation using tam.mml
mod1 <- tam.mml( dat , Q=Q , userfct.variance=variance.fct1 , variance.Npars=variance.Npars ,
                control=list(maxiter=maxit , QMC=TRUE , snodes=snodes) )
summary(mod1)

#*****************
# Model 2: Testlet model with correlated testlet effects
#*****************

# specify a testlet model with general factor g and testlet effects
# u_1,u_2,u_3 and u_4. Assume that Cov(g,u_t)=0 for all t=1,2,3,4.
# Additionally, assume that \sum_t,t' Cov( u_t , u_t') = 0, i.e.
# the sum of all testlet covariances is equal to zero
# => testlet effects are uncorrelated on average.

# set Cov(g,u_t)=0 and sum of all testlet covariances equals to zero
variance.fct2 <- function( variance ){
            ndim <- ncol(variance)
            variance.new <- matrix( 0 , ndim , ndim )
            diag(variance.new) <- diag(variance)
            variance.new[1,2:ndim] <- variance.new[2:ndim,1] <- 0
            # calculate average covariance between testlets
            v1 <- variance[ -1 , -1] - variance.new[-1,-1]
            M1 <- sum(v1) / ( ( ndim-1)^2 - ( ndim - 1))
            v1 <- v1 - M1
            variance.new[ -1 , -1 ] <- v1
```

```
            diag(variance.new) <- diag(variance)
            variance <- variance.new
            return(variance)
                    }
variance.Npars <- 1 + 4 + (4*3)/2 - 1
# estimate model in TAM
mod2 <- tam.mml( dat , Q=Q , userfct.variance=variance.fct2 , variance.Npars=variance.Npars ,
               control=list(maxiter=maxit , QMC=TRUE , snodes=snodes) )
summary(mod2)

#*****************
# Model 3: Testlet model with correlated testlet effects (different identification)
#*****************

# Testlet model like in Model 2. But now the constraint is
# \sum _t,t' Cov(u_t , u_t') + \sum_t Var(u_t) = 0 , i.e.
# the sum of all testlet covariances and variances is equal to zero.
variance.fct3 <- function( variance ){
            ndim <- ncol(variance)
            variance.new <- matrix( 0 , ndim , ndim )
            diag(variance.new) <- diag(variance)
            variance.new[1,2:ndim] <- variance.new[2:ndim,1] <- 0
            # calculate average covariance and variance between testlets
            v1 <- variance[ -1 , -1]
            M1 <- mean(v1)
            v1 <- v1 - M1
            variance.new[ -1 , -1 ] <- v1
            # ensure positive definiteness of covariance matrix
            eps <- 10^(-2)
            diag(variance.new) <- diag( variance.new) + eps
            variance.new <- psych::cor.smooth( variance.new )  # smoothing in psych
            variance <- variance.new
            return(variance)
                    }
variance.Npars <- 1 + 4 + (4*3)/2 - 1
# estimate model in TAM
mod3 <- tam.mml( dat , Q=Q , userfct.variance=variance.fct3 , variance.Npars=variance.Npars ,
               control=list(maxiter=maxit , QMC=TRUE , snodes=snodes) )
summary(mod3)

#*****************
# Model 4: Rasch subdimension model
#*****************

# The Rasch subdimension model is specified according to Brandt (2008).
# The fourth testlet effect is defined as u4 = - (u1+u2+u3)
# specify an alternative Q-matrix with 4 dimensions
Q2 <- Q[,-5]
Q2[31:40,2:4] <- -1

# set Cov(g,u1)=Cov(g,u2)=Cov(g,u3)=0
variance.fixed <- rbind( c(1,2,0) , c(1,3,0) , c(1,4,0) )
# estimate model in TAM
```

```
mod4 <- tam.mml( dat , Q=Q2 ,variance.fixed=variance.fixed ,
                control=list(maxiter=maxit , QMC=TRUE , snodes=snodes) )
summary(mod4)

#*****************
# Model 5: Higher-order model
#*****************

# A four-dimensional model with a higher-order factor is specified.
# F_t = a_t g + eps_g
Q3 <- Q[,-1]
# define fitting function using the lavaan package and ULS estimation
N0 <- nrow(dat)         # sample size of dataset
library(lavaan) # requires lavaan package for fitting covariance
variance.fct5 <- function( variance ){
    ndim <- ncol(variance)
    rownames(variance) <- colnames(variance) <- paste0("F",1:ndim)
    lavmodel <- paste0(
        "FHO =~" , paste0( paste0( "F" , 1:ndim ) , collapse="+" ) )
    lavres <- lavaan::cfa( model=lavmodel , sample.cov=variance , estimator = "ULS" ,
                      std.lv=TRUE , sample.nobs=N0)
    variance.new <- fitted(lavres)$cov
    variance <- variance.new
    # print coefficients
    cat( paste0( "\n **** Higher order loadings: " ,
            paste0( paste0( round( coef(lavres)[ 1:ndim ] , 3 )) , collapse=" ")
                        ) , "\n")
    return(variance)
                    }
variance.Npars <- 4+4
# estimate model in TAM
mod5 <- tam.mml( dat , Q=Q3 , userfct.variance=variance.fct5 , variance.Npars=variance.Npars ,
                control=list(maxiter=maxit , QMC=TRUE , snodes=snodes) )
summary(mod5)

#*****************
# Model 6: Generalized Rasch subdimension model (Brandt, 2012)
#*****************

Q2 <- Q[,-5]
Q2[31:40,2:4] <- -1
# fixed covariances
variance.fixed2 <- rbind( c(1,2,0) , c(1,3,0) , c(1,4,0)  )
# design matrix for item loading parameters
#      items x category x dimension x xsi parameter
E <- array( 0 , dim = c( 40 , 2 , 4 , 4 ) )
E[ 1:10 , 2 , c(1,2) , 1 ] <- 1
E[ 11:20 , 2 , c(1,3) , 2 ] <- 1
E[ 21:30 , 2 , c(1,4) , 3 ] <- 1
E[ 31:40 , 2 , 1 , 4 ] <- 1
E[ 31:40 , 2 , 2:4 , 4 ] <- -1

# constraint on slope parameters, see Brandt (2012)
```

```
gammaconstr <- function( gammaslope ){
        K <- length( gammaslope)
        g1 <- sum( gammaslope^2  )
        gammaslope.new <- sqrt(K) / sqrt(g1) * gammaslope
        return(gammaslope.new)
                    }
# estimate model
mod6 <- tam.mml.3pl( dat, E=E, Q=Q2, variance.fixed=variance.fixed2,  skillspace="normal",
                userfct.gammaslope = gammaconstr , gammaslope.constr.Npars = 1 ,
                control=list(maxiter=maxit , QMC=TRUE , snodes=snodes ) )
summary(mod6)

## End(Not run)
```

`tam.mml.3pl` *3PL Structured Item Response Model in* **TAM**

Description

This estimates a 3PL model with design matrices for item slopes and item intercepts. Discrete distributions of the latent variables are allowed which can be log-linearly smoothed.

Usage

```
tam.mml.3pl(resp, Y = NULL, group = NULL, formulaY = NULL, dataY = NULL, ndim = 1,
  pid = NULL, xsi.fixed = NULL, xsi.inits = NULL, xsi.prior = NULL,
  beta.fixed = NULL, beta.inits = NULL, variance.fixed = NULL, variance.inits = NULL,
  est.variance = TRUE, A = NULL, notA=FALSE , Q = NULL, E = NULL,gammaslope.des = "2PL",
   gammaslope = NULL, gammaslope.fixed = NULL,
  est.some.slopes = TRUE, gammaslope.constr.V = NULL, gammaslope.constr.c = NULL,
   gammaslope.center.index=NULL ,  gammaslope.center.value=NULL ,
  gammaslope.prior = NULL, userfct.gammaslope = NULL , gammaslope.constr.Npars = 0 ,
   est.guess = NULL, guess = rep(0, ncol(resp)),
  guess.prior = NULL, skillspace = "normal", theta.k = NULL, delta.designmatrix = NULL,
   delta.fixed = NULL, pweights = NULL, control = list() )
```

Arguments

`resp`	Data frame with polytomous item responses $k = 0, ..., K$. Missing responses must be declared as `NA`.
`Y`	A matrix of covariates in latent regression. Note that the intercept is automatically included as the first predictor.
`group`	An optional vector of group identifiers
`formulaY`	An R formula for latent regression. Transformations of predictors in Y (included in `dataY`) can be easily spcified, e. g. `female*race` or `I(age^2)`.
`dataY`	An optional data frame with possible covariates Y in latent regression. This data frame will be used if an R formula in `formulaY` is specified.

`ndim` Number of dimensions (is not needed to determined by the user)

`pid` An optional vector of person identifiers

`xsi.fixed` A matrix with two columns for fixing ξ parameters. 1st column: index of ξ parameter, 2nd column: fixed value

`xsi.inits` A matrix with two columns (in the same way defined as in `xsi.fixed` with initial value for ξ.

`xsi.prior` Item-specific prior distribution for ξ parameters. It is assumed that $\xi_k \sim N(\mu_k, \sigma_k^2)$. The first column in `xsi.prior` is μ_k, the second is σ_k.

`beta.fixed` A matrix with three columns for fixing regression coefficients. 1st column: Index of Y value, 2nd column: dimension, 3rd column: fixed β value.
If no constraints should be inposed on β, then set `beta.fixed=FALSE` (see Example 2, Model `2_4`).

`beta.inits` A matrix (same format as in `beta.fixed`) with initial β values

`variance.fixed` An optional matrix with three columns for fixing entries in covariance matrix: 1st column: dimension 1, 2nd column: dimension 2, 3rd column: fixed value

`variance.inits` Initial covariance matrix in estimation. All matrix entries have to be specified and this matrix is NOT in the same format like `variance.inits`.

`est.variance` Should the covariance matrix be estimated? This argument applies to estimated item slopes in `tam.mml.2pl`. The default is `FALSE` which means that latent variables (in the first group) are standardized in 2PL estimation.

`A` An optional array of dimension $I \times (K+1) \times N_\xi$. Only ξ parameters are estimated, entries in A only correspond to the design.

`notA` An optional logical indicating whether all entries in the A matrix are set to zero and no item intercept ξ should be estimated.

`Q` An optional $I \times D$ matrix (the Q-matrix) which specifies the loading structure of items on dimensions.

`E` Optional design array for item slopes γ. It is a four dimensional array of size $I \times (K+1) \times D \times N_\gamma$ containing items, categories, dimensions, γ parameter.

`gammaslope.des` Optional string indicating type of item slope parameter to be estimated. `gammaslope.des="2PL"` estimates a slope parameter for an item, `gammaslope.des="2PLcat"` for an item and a

`gammaslope` Initial or fixed vector of γ parameters

`gammaslope.fixed`
An optional matrix containing fixed values in the γ vector. First column: parameter index; second colunmn: fixed value.

`est.some.slopes`
An optional logical indicating whether some item slopes should be estimated.

`gammaslope.constr.V`
An optional constraint matrix V for item slope parameters γ

`gammaslope.constr.c`
An optional constraint vector c for item slope parameters γ

`gammaslope.center.index`
Indices of `gammaslope` parameters which should be fixed to sum specified in `gammaslope.center.value` (see Example 7).

`gammaslope.center.value`
: Specified values of sum of subset of `gammaslope` parameters.

`gammaslope.prior`
: Item-specific prior distribution for γ parameters. It is assumed that $\gamma_k \sim N(\mu_k, \sigma_k^2)$. The first column in `gammaslope.prior` is μ_k, the second is σ_k.

`userfct.gammaslope`
: A user specified function for constraints or transformations of the γ parameters within the algorithm. See Example 17 in `tam.mml`.

`gammaslope.constr.Npars`
: Number of constrained γ parameters in `userfct.gammaslope`

`est.guess`
: An optional vector of integers indicating for which items a guessing parameter should be estimated. Same integers correspond to same estimated guessing parameters. A value of 0 denotes an item for which no guessing parameter is estimated.

`guess`
: Fixed or initial guessing parameters

`guess.prior`
: Item-specific prior distribution for guessing parameters c_i. It is assumed that $c_i \sim N(a_i, b_i)$. The first column in `gammaslope.prior` is a_i, the second is b_i.

`skillspace`
: Skill space: normal distribution (`"normal"`) or discrete distribution (`"discrete"`).

`theta.k`
: A matrix of the $\boldsymbol{\theta}$ skill space in case of a discrete distribution (`skillspace="discrete"`).

`delta.designmatrix`
: A design matrix of the log-linear model for the skill space in case of a discrete distribution (`skillspace="discrete"`).

`delta.fixed`
: Fixed δ values of the log-linear skill space. `delta.fixed` must be a matrix with three columns. First column: δ parameter index, Second column: Group index, Third column: Fixed δ parameter value.

`pweights`
: Optional vector of person weights.

`control`
: See `tam.mml` for more details.

Details

The item response model for item i and category h and no guessing parameters can be written as

$$P(X_i = h|\boldsymbol{\theta}) \propto exp(\sum_d b_{ihd}\theta_d + \sum_k a_{ih}\xi_k)$$

For item slopes, a linear decomposition is allowed

$$b_{ihd} = \sum_k e_{ihdk}\gamma_k$$

In case of a guessing parameter, the item response function is

$$P(X_i = h|\boldsymbol{\theta}) = c_i + (1 - c_i) \cdot (1 + exp(-\sum_d b_{ihd}\theta_d - \sum_k a_{ih}\xi_k))^{-1}$$

For possibilities of definitions of the design matrix $E = (e_{ihdk})$ see Formann (2007), for the strongly related linear logistic latent class model see Formann (1992). Linear constraints on γ can be specified by equations $V\gamma = c$ and using the arguments `gammaslope.constr.V` and `gammaslope.constr.c`.

Like in tam.mml, a multivariate linear regression

$$\boldsymbol{\theta} = Y\beta + \boldsymbol{\epsilon}$$

assuming a multivariate normal distribution on the residuals $\boldsymbol{\epsilon}$ can be specified (skillspace="normal").

Alternatively, a log-linear distribution of the skill classes $P(\theta)$ (skillspace="discrete") is performed

$$logP(\theta) = D_\delta \delta$$

See Xu and von Davier (2008). The design matrix D_δ can be specified in delta.designmatrix. The theta grid $\boldsymbol{\theta}$ of the skill space can be defined in theta.k.

Value

The same output as in tam.mml and additional entries

delta	Parameter vector δ
gammaslope	Estimated γ item slope parameters
E	Used design matrix E

References

Formann, A. K. (1992). Linear logistic latent class analysis for polytomous data. *Journal of the American Statistical Association*, **87**, 476-486.

Formann, A. K. (2007). (Almost) Equivalence between conditional and mixture maximum likelihood estimates for some models of the Rasch type. In M. von Davier & C. H. Carstensen (Eds.), *Multivariate and mixture distribution Rasch models* (pp. 177-189). New York: Springer.

Xu, X., & von Davier, M. (2008). *Fitting the structured general diagnostic model to NAEP data*. ETS Research Report ETS RR-08-27. Princeton, ETS.

See Also

See also tam.mml.

See the slca function in the **CDM** package for a similar method.

logLik.tam, anova.tam

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: Dichotomous data | sim.rasch
#############################################################################

data(sim.rasch)
dat <- sim.rasch
# some control arguments
ctl.list <- list(maxiter= 20 ) # increase the number of iterations in applications!

#*** Model 1: Rasch model, normal trait distribution
```

```
mod1 <- tam.mml.3pl(resp=dat , skillspace= "normal" , est.some.slopes=FALSE ,
        control=ctl.list)
summary(mod1)

#*** Model 2: Rasch model, discrete trait distribution
#  choose theta grid
theta.k <- seq( -3 , 3 , len=7 ) # discrete theta grid distribution
# define symmetric trait distribution
delta.designmatrix <- matrix( 0 , nrow=7 , ncol= 4 )
delta.designmatrix[4,1] <- 1
delta.designmatrix[c(3,5),2] <- 1
delta.designmatrix[c(2,6),3] <- 1
delta.designmatrix[c(1,7),4] <- 1
mod2 <- tam.mml.3pl(resp=dat , skillspace= "discrete" , est.some.slopes=FALSE ,
           theta.k=theta.k, delta.designmatrix=delta.designmatrix , control=ctl.list)
summary(mod2)

#*** Model 3: 2PL model
mod3 <- tam.mml.3pl(resp=dat , skillspace= "normal" , gammaslope.des="2PL" ,control=ctl.list)
summary(mod3)

#*** Model 4: 3PL model
# estimate guessing parameters for items 3,7,9 and 12
I <- ncol(dat)
est.guess <- rep(0, I )
# set parameters 9 and 12 equal -> same integers
est.guess[ c(3,7,9,12) ] <- c( 1 , 3 , 2 , 2 )
# starting values guessing parameter
guess <- .2*(est.guess > 0)
# estimate model
mod4 <- tam.mml.3pl(resp=dat , skillspace= "normal" , gammaslope.des="2PL" ,
        control=ctl.list , est.guess = est.guess , guess=guess)
summary(mod4)

#*** Model 5: 3PL model, add some prior Beta distribution
guess.prior <- matrix( 0 , nrow=I , ncol=2 )
guess.prior[ est.guess  > 0 , 1] <- 5
guess.prior[ est.guess  > 0 , 2] <- 17
mod5 <- tam.mml.3pl(resp=dat , skillspace= "normal" , gammaslope.des="2PL" ,
       control=ctl.list , est.guess = est.guess , guess=guess , guess.prior=guess.prior)
summary(mod5)

#*** Model 6: 2PL model with design matrix for item slopes
I <- 40      # number of items
D <- 1       # dimension
maxK <- 2    # maximum number of categories
Ngam <- 13   # number of different slope parameters
E <- array( 0 , dim=c(I,maxK,D,Ngam) )
# joint slope parameters for items 1 to 10, 11 to 20, 21 to 30
E[ 1:10 , 2 , 1 , 2 ] <- 1
E[ 11:20 , 2 , 1 , 1 ] <- 1
E[ 21:30 , 2 , 1 , 3 ] <- 1
for (ii in 31:40){   E[ii,2,1,ii - 27 ] <- 1 }
```

```
# estimate model
mod6 <- tam.mml.3pl(resp= dat , control= ctl.list ,   E=E , est.variance=FALSE  )
summary(mod6)

#*** Model 7: 2PL model with design matrix of slopes and slope constraints
Ngam <- dim(E)[4]   # number of gamma parameters
# define two constraint equations
gammaslope.constr.V <- matrix( 0 , nrow=Ngam , ncol=2 )
gammaslope.constr.c <- rep(0,2)
# set sum of first two xlambda entries to 1.8
gammaslope.constr.V[1:2,1] <- 1
gammaslope.constr.c[1] <- 1.8
# set sum of entries 4, 5 and 6 to 3
gammaslope.constr.V[4:6,2] <- 1
gammaslope.constr.c[2] <- 3
mod7 <- tam.mml.3pl(resp= dat, control= ctl.list ,  E=E ,  est.variance=FALSE ,
   gammaslope.constr.V=gammaslope.constr.V , gammaslope.constr.c=gammaslope.constr.c)
summary(mod7)

#**** Model 8: Located latent class Rasch model with estimated three skill points

# three classes of theta's are estimated
TP <- 3
theta.k <- diag(TP)
# because item difficulties are unrestricted, we define the sum of the estimated
# theta points equal to zero
Ngam <- TP  # estimate three gamma loading parameters which are discrete theta points
E <- array( 0 , dim=c(I,2,TP,Ngam) )
E[,2,1,1] <- E[,2,2,2] <- E[,2,3,3] <- 1
gammaslope.constr.V <- matrix( 1 , nrow=3 , ncol=1 )
gammaslope.constr.c <- c(0)
# initial gamma values
gammaslope <- c( -2 , 0 , 2 )
# estimate model
mod8 <- tam.mml.3pl(resp= dat, control= ctl.list ,  E=E ,  skillspace="discrete" ,
     theta.k=theta.k , gammaslope=gammaslope , gammaslope.constr.V=gammaslope.constr.V ,
     gammaslope.constr.c=gammaslope.constr.c )
summary(mod8)

#############################################################################
# EXAMPLE 2: Polytomous data
#############################################################################

data( data.mg , package="CDM")
dat <- data.mg[1:1000, paste0("I",1:11)]

#*** Model 1: 1-dimensional 1PL estimation, normal skill distribution
mod1 <- tam.mml.3pl(resp=dat , skillspace="normal" ,
         control= list( maxiter= 30 )  , gammaslope.des = "2PL" ,
         est.some.slopes=FALSE ,  est.variance=TRUE  )
summary(mod1)

#*** Model 2: 1-dimensional 2PL estimation, discrete skill distribution
```

```
# define skill space
theta.k <- matrix( seq(-5,5,len=21) )
# allow skew skill distribution
delta.designmatrix <- cbind( 1 , theta.k , theta.k^2 , theta.k^3 )
# fix 13th xsi item parameter to zero
xsi.fixed <- cbind( 13 , 0 )
# fix 10th slope paremeter to one
gammaslope.fixed <- cbind( 10 , 1 )
# estimate model
mod2 <- tam.mml.3pl(resp=dat , skillspace="discrete" , theta.k=theta.k ,
          delta.designmatrix=delta.designmatrix,  control= list(maxiter=30),
          gammaslope.des = "2PL" , xsi.fixed=xsi.fixed ,
          gammaslope.fixed=gammaslope.fixed)
summary(mod2)

#*** Model 3: 2-dimensional 2PL estimation, normal skill distribution

# define loading matrix
Q <- matrix(0,11,2)
Q[1:6,1] <- 1   # items 1 to 6 load on dimension 1
Q[7:11,2] <- 1  # items 7 to 11 load on dimension 2
# estimate model
mod3 <- tam.mml.3pl(resp=dat , control= list(maxiter=30),
          gammaslope.des = "2PL" , Q=Q )
summary(mod3)

############################################################################
# SIMULATED EXAMPLE 3: dichotomous data with guessing
############################################################################

#*** simulate data
set.seed(9765)
N <- 4000   # number of persons
I <- 20     # number of items
b <- seq( -1.5 , 1.5 , len=I )
guess <- rep(0 , I )
guess.items <- c(6,11,16)
guess[ guess.items ] <- .33
library(sirt)
dat <- sirt::sim.raschtype( rnorm(N) , b=b , fixed.c=guess )

#*** Model 1: Difficulty + guessing model, i.e. fix slopes to 1
est.guess <- rep(0,I)
est.guess[ guess.items ] <- seq(1, length(guess.items) )
# define prior distribution
guess.prior <- matrix( cbind( 5 , 17 ) , I , 2 , byrow=TRUE )
guess.prior[ ! est.guess , ] <- 0
# estimate model
mod1 <- tam.mml.3pl(resp= dat , guess = guess , est.guess = est.guess ,
        guess.prior=guess.prior , control= ctl.list ,est.variance=TRUE  ,
        est.some.slopes=FALSE )
summary(mod1)
```

```
#*** Model 2: estimate a joint guessing parameter
est.guess <- rep(0,I)
est.guess[ guess.items ] <- 1
# estimate model
mod2 <- tam.mml.3pl(resp= dat , guess = guess , est.guess = est.guess ,
        guess.prior=guess.prior , control= ctl.list ,est.variance=TRUE  ,
        est.some.slopes=FALSE )
summary(mod2)

############################################################################
# SIMULATED EXAMPLE 4: Latent class model with two classes
#      See slca Simulated Example 2 in the CDM package
############################################################################

#*** simulate data
set.seed(9876)
I <- 7  # number of items
# simulate response probabilities
a1 <- round(runif(I,0 , .4 ),4)
a2 <- round(runif(I , .6 , 1 ),4)
N <- 1000   # sample size
# simulate data in two classes of proportions .3 and .7
N1 <- round(.3*N)
dat1 <- 1 * ( matrix(a1,N1,I,byrow=TRUE) > matrix( runif( N1 * I) , N1 , I ) )
N2 <- round(.7*N)
dat2 <- 1 * ( matrix(a2,N2,I,byrow=TRUE) > matrix( runif( N2 * I) , N2 , I ) )
dat <- rbind( dat1 , dat2 )
colnames(dat) <- paste0("I" , 1:I)

# define design matrices
TP <- 2     # two classes
theta.k <- diag(TP)     # there are theta dimensions -> two classes
# The idea is that latent classes refer to two different "dimensions".
# Items load on latent class indicators 1 and 2, see below.
E <- array(0 , dim=c(I,2,2,2*I) )
items <- colnames(dat)
dimnames(E)[[4]] <- c(paste0( colnames(dat) , "Class" , 1),
          paste0( colnames(dat) , "Class" , 2) )
# items, categories , classes , parameters
# probabilities for correct solution
for (ii in 1:I){
    E[ ii , 2 , 1 , ii ] <- 1    # probabilities class 1
    E[ ii , 2 , 2 , ii+I ] <- 1  # probabilities class 2
                    }
# define A matrix for xsi parameters which does not contain any parameter contributions
A <- array( 0 , dim=c(I,2,2) )
xsi.fixed <- cbind( cbind(1,2) , 0 )
# estimate model
mod1 <- tam.mml.3pl(resp= dat , E=E , A=A , control= list(maxit=20) , skillspace="discrete" ,
          theta.k=theta.k )
summary(mod1)
# compare simulated and estimated data
cbind( mod1$rprobs[,2,1] , a2 )  # Simulated class 2
```

```
cbind( mod1$rprobs[,2,2] , a1 )  # Simulated class 1

#############################################################################
# SIMULATED EXAMPLE 5: Located latent class model, Rasch model
#      See slca Simulated Example 4 in the CDM package
#############################################################################

#*** simulate data
set.seed(487)
I <- 15  # I items
b1 <- seq( -2 , 2 , len=I)      # item difficulties
N <- 2000    # number of persons
# simulate 4 theta classes
theta0 <- c( -2.5 , -1 , 0.3 , 1.3 )  # skill classes
probs0 <- c( .1 , .4 , .2 , .3 )      # skill class probabilities
TP <- length(theta0)
theta <- theta0[ rep(1:TP, round(probs0*N)  ) ]
library(sirt)
dat <- sirt::sim.raschtype( theta , b1 )

#*** Model 1: Located latent class model with 4 classes
maxK <- 2
Ngam <- TP
E <- array( 0 , dim=c(I , maxK , TP ,  Ngam ) )
dimnames(E)[[1]] <- colnames(dat)
dimnames(E)[[2]] <- paste0("Cat", 1:(maxK) )
dimnames(E)[[3]] <- paste0("Class", 1:TP)
dimnames(E)[[4]] <- paste0("theta", 1:TP)
# theta design
for (tt in 1:TP){   E[1:I , 2 , tt ,  tt] <- 1       }
theta.k <- diag(TP)
# set eighth item difficulty to zero
xsi.fixed <- cbind( 8 , 0 )
# initial gamma parameter
gammaslope <- seq( -1.5 , 1.5 , len=TP)
# estimate model
mod1 <- tam.mml.3pl(resp= dat , E=E , xsi.fixed=xsi.fixed ,
           control= list(maxiter=100) , skillspace="discrete" ,
           theta.k=theta.k , gammaslope=gammaslope)
summary(mod1)
# compare estimated and simulated theta class locations
cbind( mod1$gammaslope , theta0 )
# compare estimated and simulated latent class proportions
cbind( mod1$pi.k , probs0 )

#############################################################################
# SIMULATED EXAMPLE 6: DINA model with two skills
#      See slca Simulated Example 5 in the CDM package
#############################################################################

#*** simulate data
set.seed(487)
N <- 3000   # number of persons
```

```
# define Q-matrix
I <- 9  # 9 items
NS <- 2 # 2 skills
TP <- 4 # number of skill classes
Q <- scan(nlines=3, text=
  "1 0   1 0   1 0
   0 1   0 1   0 1
   1 1   1 1   1 1",
   )
Q <- matrix(Q , I, ncol=NS, byrow=TRUE)
# define skill distribution
alpha0 <- matrix( c(0,0,1,0,0,1,1,1) , nrow=4,ncol=2,byrow=TRUE)
prob0 <- c( .2 , .4 , .1 , .3 )
alpha <- alpha0[ rep( 1:TP , prob0*N) ,]
# define guessing and slipping parameters
guess <- round( runif(I, 0 , .4 ) , 2 )
slip <- round( runif(I , 0 , .3 ) , 2 )
# simulate data according to the DINA model
dat <- CDM::sim.din( q.matrix=Q , alpha=alpha , slip=slip , guess=guess )$dat

#*** Model 1: Estimate DINA model
# define E matrix which contains the anti-slipping parameters
maxK <- 2
Ngam <- I
E <- array( 0 , dim=c(I , maxK , TP ,  Ngam ) )
dimnames(E)[[1]] <- colnames(dat)
dimnames(E)[[2]] <- paste0("Cat" , 1:(maxK) )
dimnames(E)[[3]] <- c("S00","S10","S01","S11")
dimnames(E)[[4]] <- paste0( "antislip" , 1:I )
# define anti-slipping parameters in E
for (ii in 1:I){
        # define latent responses
        latresp <- 1*( alpha0 %*% Q[ii,]  == sum(Q[ii,]) )[,1]
        # model slipping parameters
        E[ii , 2 , latresp == 1, ii ] <- 1
                }
# skill space definition
theta.k <- diag(TP)
gammaslope <- rep( qlogis( .8 ) , I )

# estimate model
mod1 <- tam.mml.3pl(resp= dat , E=E , control= list(maxiter=100) , skillspace="discrete" ,
          theta.k=theta.k , gammaslope=gammaslope)
summary(mod1)
# compare estimated and simulated latent class proportions
cbind( mod1$pi.k , probs0 )
# compare estimated and simulated guessing parameters
cbind( mod1$rprobs[,2,1] , guess )
# compare estimated and simulated slipping parameters
cbind( 1 - mod1$rprobs[,2,4] , slip )

#############################################################################
# SIMULATED EXAMPLE 7: Mixed Rasch model with two classes
```

```
#      See slca Simulated Example 3 in the CDM package
#############################################################################

#*** simulate data
set.seed(987)
library(sirt)
# simulate two latent classes of Rasch populations
I <- 15  # 6 items
b1 <- seq( -1.5 , 1.5 , len=I)      # difficulties latent class 1
b2 <- b1    # difficulties latent class 2
b2[ c(4,7, 9 , 11 , 12, 13) ] <- c(1 , -.5 , -.5 , .33 , .33 , -.66 )
b2 <- b2 - mean(b2)
N <- 3000    # number of persons
wgt <- .25       # class probability for class 1
# class 1
dat1 <- sirt::sim.raschtype( rnorm( wgt*N ) , - b1 )
# class 2
dat2 <- sirt::sim.raschtype( rnorm( (1-wgt)*N , mean= 1 , sd=1.7) , - b2 )
dat <- rbind( dat1 , dat2 )

# The idea is that each grid point class x theta is defined as new
# dimension. If we approximate the trait distribution by 7 theta points
# and are interested in estimating 3 latent classes, then we need
# 7*3=21 dimensions.

#*** Model 1: Rasch model
# theta grid
theta.k1 <- seq( -3 , 3 , len=7 )
TT <- length(theta.k1)
#-- define theta design matrix
theta.k <- diag(TT)
#-- delta designmatrix
delta.designmatrix <- matrix( 0 , TT , ncol=3 )
delta.designmatrix[ , 1] <- 1
delta.designmatrix[, 2:3] <- cbind( theta.k1 , theta.k1^2 )

#-- define loading matrix E
E <- array( 0 , dim=c(I,2,TT,I + 1) )  # last parameter is constant 1
for (ii in 1:I){
    E[ ii , 2 , 1:TT , ii ] <- -1   # '-b' in '1*theta - b'
    E[ ii , 2 , 1:TT , I+1] <- theta.k1  # '1*theta' in '1*theta - b'
                }
# initial gammaslope parameters
par1 <- qlogis( colMeans( dat ) )
gammaslope <- c( par1 , 1 )
# sum constraint of zero on item difficulties
gammaslope.constr.V <- matrix( 0 , I+1 , 1 )
gammaslope.constr.V[ 1:I , 1] <- 1  # Class 1
gammaslope.constr.c <- c(0)
# fixed gammaslope parameter
gammaslope.fixed <- cbind( I+1 , 1 )
# estimate model
mod1 <- tam.mml.3pl(resp= dat1 , E=E , control= list(maxiter=10) , skillspace="discrete" ,
```

```
      theta.k=theta.k , delta.designmatrix=delta.designmatrix ,
      gammaslope=gammaslope , gammaslope.constr.V=gammaslope.constr.V ,
      gammaslope.constr.c=gammaslope.constr.c , gammaslope.fixed=gammaslope.fixed ,
      notA=TRUE , est.variance=FALSE)
summary(mod1)

#*** Model 2: Mixed Rasch model with two latent classes
# theta grid
theta.k1 <- seq( -4 , 4 , len=7 )
TT <- length(theta.k1)
#-- define theta design matrix
theta.k <- diag(2*TT)   # 2*7=14 classes
#-- delta designmatrix
delta.designmatrix <- matrix( 0 , 2*TT , ncol=6 )
# Class 1
delta.designmatrix[1:TT , 1] <- 1
delta.designmatrix[1:TT , 2:3] <- cbind( theta.k1 , theta.k1^2 )
# Class 2
delta.designmatrix[TT+1:TT , 4] <- 1
delta.designmatrix[TT+1:TT , 5:6] <- cbind( theta.k1 , theta.k1^2 )

#-- define loading matrix E
E <- array( 0 , dim=c(I,2,2*TT,2*I + 1) )  # last parameter is constant 1
dimnames(E)[[1]] <- colnames(dat)
dimnames(E)[[2]] <- c("Cat0","Cat1")
dimnames(E)[[3]] <- c( paste0("Class1_theta" , 1:TT) , paste0("Class2_theta" , 1:TT) )
dimnames(E)[[4]] <- c( paste0("b_Class1_" , colnames(dat)) ,
       paste0("b_Class2_" , colnames(dat)) , "One")
for (ii in 1:I){
  # Class 1 item parameters
    E[ ii , 2 , 1:TT , ii ] <- -1   # '-b' in '1*theta - b'
    E[ ii , 2 , 1:TT , 2*I+1] <- theta.k1  # '1*theta' in '1*theta - b'
  # Class 2 item parameters
    E[ ii , 2 , TT + 1:TT , I + ii ] <- -1
    E[ ii , 2 , TT + 1:TT , 2*I+1] <- theta.k1
               }
# initial gammaslope parameters
par1 <- qlogis( colMeans( dat ) )
gammaslope <- c( par1 , par1 + runif(I, -2  ,2 ) , 1 )
# sum constraint of zero on item difficulties within a class
gammaslope.center.index <- c( rep( 1, I ) , rep(2,I) , 0 )
gammaslope.center.value <- c(0,0)

# estimate model
mod1 <- tam.mml.3pl(resp= dat , E=E , control= list(maxiter=30) , skillspace="discrete" ,
      theta.k=theta.k , delta.designmatrix=delta.designmatrix ,
      gammaslope=gammaslope , gammaslope.center.index=gammaslope.center.index ,
     gammaslope.center.value=gammaslope.center.value, gammaslope.fixed=gammaslope.fixed ,
      notA=TRUE)
summary(mod1)
# latent class proportions
aggregate( mod1$pi.k , list( rep(1:2, each=TT)) , sum )
# compare simulated and estimated item parameters
```

```
cbind( b1 , b2 , - mod1$gammaslope[1:I] , - mod1$gammaslope[I + 1:I ] )

#############################################################################
# SIMULATED EXAMPLE 8: 2PL mixture distribution model
#############################################################################

#*** simulate data
set.seed(9187)
library(sirt)
# simulate two latent classes of Rasch populations
I <- 20
b1 <- seq( -1.5 , 1.5 , len=I)      # difficulties latent class 1
b2 <- b1    # difficulties latent class 2
b2[ c(4,7, 9 , 11 , 12, 13 , 16 , 18) ] <- c(1 , -.5 , -.5 , .33 , .33 , -.66 , -1 , .3)
# b2 <- scale( b2 , scale=FALSE)
b2 <- b2 - mean(b2)
N <- 4000       # number of persons
wgt <- .75       # class probability for class 1
# item slopes
a1 <- rep( 1 , I )  # first class
a2 <- rep( c(.5,1.5) , I/2 )

# class 1
dat1 <- sirt::sim.raschtype( rnorm( wgt*N ) , - b1 , fixed.a=a1)
# class 2
dat2 <- sirt::sim.raschtype( rnorm( (1-wgt)*N , mean= 1 , sd=1.4) , - b2 , fixed.a=a2)
dat <- rbind( dat1 , dat2 )

#*** Model 1: Mixed 2PL model with two latent classes

theta.k1 <- seq( -4 , 4 , len=7 )
TT <- length(theta.k1)
#-- define theta design matrix
theta.k <- diag(2*TT)   # 2*7=14 classes
#-- delta designmatrix
delta.designmatrix <- matrix( 0 , 2*TT , ncol=6 )
# Class 1
delta.designmatrix[1:TT , 1] <- 1
delta.designmatrix[1:TT , 2:3] <- cbind( theta.k1 , theta.k1^2 )
# Class 2
delta.designmatrix[TT+1:TT , 4] <- 1
delta.designmatrix[TT+1:TT , 5:6] <- cbind( theta.k1 , theta.k1^2 )

#-- define loading matrix E
E <- array( 0 , dim=c(I,2,2*TT,4*I ) )
dimnames(E)[[1]] <- colnames(dat)
dimnames(E)[[2]] <- c("Cat0","Cat1")
dimnames(E)[[3]] <- c( paste0("Class1_theta" , 1:TT) , paste0("Class2_theta" , 1:TT) )
dimnames(E)[[4]] <- c( paste0("b_Class1_" , colnames(dat)) ,  paste0("a_Class1_" , colnames(dat)) ,
       paste0("b_Class2_" , colnames(dat)) , paste0("a_Class2_" , colnames(dat)) )

for (ii in 1:I){
  # Class 1 item parameters
```

```
    E[ ii , 2 , 1:TT , ii ] <- -1           # '-b' in 'a*theta - b'
    E[ ii , 2 , 1:TT , I + ii] <- theta.k1  # 'a*theta' in 'a*theta - b'
  # Class 2 item parameters
    E[ ii , 2 , TT + 1:TT , 2*I + ii ] <- -1
    E[ ii , 2 , TT + 1:TT , 3*I + ii ] <- theta.k1
                }

# initial gammaslope parameters
par1 <- scale( - qlogis( colMeans( dat ) ) , scale =FALSE )
gammaslope <- c( par1 , rep(1,I) ,  scale( par1 + runif(I, - 1.4  , 1.4 ),
        scale=FALSE)  , runif( I,.6,1.4) )

# constraint matrix
gammaslope.constr.V <- matrix( 0 , 4*I , 4 )
# sum of item intercepts equals zero
gammaslope.constr.V[ 1:I , 1] <- 1        # Class 1  (b)
gammaslope.constr.V[ 2*I + 1:I , 2] <- 1  # Class 2 (b)
# sum of item slopes equals number of items -> mean slope of 1
gammaslope.constr.V[ I + 1:I , 3] <- 1    # Class 1 (a)
gammaslope.constr.V[ 3*I + 1:I , 4] <- 1  # Class 2 (a)
gammaslope.constr.c <- c(0,0,I,I)

# estimate model
mod1 <- tam.mml.3pl(resp= dat , E=E , control= list(maxiter=80) , skillspace="discrete" ,
      theta.k=theta.k , delta.designmatrix=delta.designmatrix ,
      gammaslope=gammaslope , gammaslope.constr.V=gammaslope.constr.V ,
      gammaslope.constr.c=gammaslope.constr.c ,  gammaslope.fixed=gammaslope.fixed ,
      notA=TRUE)

# estimated item parameters
mod1$gammaslope
# summary
summary(mod1)
# latent class proportions
round( aggregate( mod1$pi.k , list( rep(1:2, each=TT)) , sum ) , 3 )
# compare simulated and estimated item intercepts
int <- cbind( b1*a1 , b2 * a2 , - mod1$gammaslope[1:I] , - mod1$gammaslope[2*I + 1:I ] )
round( int , 3 )
# simulated and estimated item slopes
slo <- cbind( a1 , a2 ,  mod1$gammaslope[I+1:I] , mod1$gammaslope[3*I + 1:I ] )
round(slo,3)

## End(Not run)
```

`tam.modelfit` *Q3 and Adjusted Q3 Statistic and Further Model Fit Statistics*

Description

This function computes the Q3 statistic (Yen, 1984) and an adjusted variant of it (see Details). In addition, some effect sizes of model fit (`MADaQ3`, $MADRESIDCOV$, $SRMR$) are implemented.

Usage

```
tam.modelfit(tamobj, progress = TRUE)

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

Arguments

tamobj	Object of class tam
progress	An optional logical indicating whether progress should be displayed
object	Object of class tam.modelfit
...	Further arguments to be passed

Details

For each item i and each person n, residuals $e_{ni} = X_{ni} - E(X_{ni})$ are computed. The expected value $E(X_{ni})$ is obtained by integrating the individual posterior distribution.

The Q3 statistic of item pairs i and j is defined as the correlation $Q3_{ij} = Cor(e_{ni}, e_{nj})$. It is known that under local independence, the expected value is slightly smaller than zero. Therefore, an adjusted Q3 statistic (aQ3; $aQ3_{ij}$) is computed by subtracting the average of all Q3 statistics from Q3. To control for multiple testing, a p value adjustment by the method of Holm (p.holm) is employed (see Chen, de la Torre & Zhang, 2013).

An effect size of model fit (MADaQ3) is defined as the average of absolute values of $aQ3$ statistics.

The SRMSR (standardized root mean square root of squared residuals, Maydeu-Olivaras, 2013) is based on comparing residual correlations of item pairs

$$SRMSR = \sqrt{\frac{1}{J(J-1)/2} \sum_{i<j} (r_{ij} - \hat{r}_{ij})^2}$$

Additionally, the SRMR is computed as

$$SRMR = \frac{1}{J(J-1)/2} \sum_{i<j} |r_{ij} - \hat{r}_{ij}|$$

The $MADRESIDCOV$ statistic (McDonald & Mok, 1995) is based on comparing residual covariances of item pairs

$$MADRESIDCOV = \frac{1}{J(J-1)/2} \sum_{i<j} |c_{ij} - \hat{c}_{ij}|$$

This statistic is just multiplied by 100 in the output of this function.

Value

A list with following entries

stat.MADaQ3	Global fit statistic MADaQ3 and global model test with p value obtained by Holm adjustment

`chi2.stat`	Data frame with chi square tests of condional independence for every item pair (Chen & Thissen, 1997)
`fitstat`	Model fit statistics $100 \cdot MADRESIDCOV$, $SRMR$ and $SRMSR$
`modelfit.test`	Test statistic of global fit based on multiple testing correction of χ^2 statistics
`stat.itempair`	Q3 and adjusted Q3 statistic for all item pairs
`residuals`	Residuals
`Q3.matr`	Matrix of Q_3 statistics
`aQ3.matr`	Matrix of adjusted Q_3 statistics

References

Chen, J., de la Torre, J., & Zhang, Z. (2013). Relative and absolute fit evaluation in cognitive diagnosis modeling. *Journal of Educational Measurement*, **50**, 123-140.

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

Maydeu-Olivares, A. (2013). Goodness-of-fit assessment of item response theory models (with discussion). *Measurement: Interdisciplinary Research and Perspectives*, **11**, 71-137.

McDonald, R. P., & Mok, M. M.-C. (1995). Goodness of fit in item response models. *Multivariate Behavioral Research*, **30**, 23-40.

Yen, W. M. (1984). Effects of local item dependence on the fit and equating performance of the three-parameter logistic model. *Applied Psychological Measurement*, **8**, 125-145.

Examples

```
############################################################################
# EXAMPLE 1: data.cqc01
############################################################################

data(data.cqc01)
dat <- data.cqc01

#*** estimation
mod1 <- tam.mml( dat )
#*** model fit
res1 <- tam.modelfit( tamobj = mod1 )
summary(res1)
# display item pairs with five largest adjusted Q3 statistics
res1$stat.itempair[1:5,c("item1","item2","aQ3","p","p.holm")]

## Not run:
############################################################################
# SIMULATED EXAMPLE 2: Rasch model
############################################################################

set.seed(8766)
N <- 1000    # number of persons
I <- 20      # number of items
# simulate responses
```

```
library(sirt)
dat <- sirt::sim.raschtype( rnorm(N) , b=seq(-1.5,1.5,len=I) )
#*** estimation
mod1 <- tam.mml( dat )
summary(dat)
#*** model fit
res1 <- tam.modelfit( tamobj = mod1)
summary(res1)

## End(Not run)
```

---

tam.pv *Plausible Value Imputation*

Description

Plausible value imputation for objects of the classes tam and tam.mml (Adams & Wu, 2007).

Usage

```
tam.pv(tamobj, nplausible = 10, ntheta = 2000, normal.approx = FALSE,
    samp.regr = FALSE , np.adj=8 )
```

Arguments

tamobj	Object of class tam or tam.mml
nplausible	Number of plausible values to be drawn
ntheta	Number of ability nodes for plausible value imputation. Note that in this function ability nodes are simulated for the whole sample, not for every person (contrary to the software ConQuest).
normal.approx	An optional logical indicating whether the individual posterior distributions should be approximated by a normal distribution? The default is FALSE. In the case normal.approx=TRUE (normal distribution approximation), the number of ability nodes ntheta can be substantially smaller than 2000, say 200 or 500. Note that normal approximation is only implemented for unidimensional models.
samp.regr	An optional logical indicating whether regression coefficients should be fixed in the plausible value imputation or also sampled from their posterior distribution? The default is FALSE. Note that sampling of regression coefficients is only implemented for unidimensional models.
np.adj	This parameter defines the "spread" of the random theta values for drawing plausible values when normal.approx=FALSE. If s_{EAP} denotes the standard deviation of the posterior distribution of theta (in the one-dimensional case), then theta is simulated from a normal distribution with standard deviation np.adj times s_{EAP}.
...	Further arguments to be passed

Value

A list with following entries:

pv	A data frame containing a person identifier (pid) and plausible values denoted by PVxx.Dimyy which is the xxth plausible value of dimension yy.
hwt	Individual posterior distribution evaluated at the ability grid theta
hwt1	Cumulated individual posterior distribution
theta	Simulated ability nodes

References

Adams, R. J., & Wu, M. L. (2007). The mixed-coefficients multinomial logit model. A generalized form of the Rasch model. In M. von Davier & C. H. Carstensen (Eds.): *Multivariate and mixture distribution Rasch models: Extensions and applications* (pp. 55-76). New York: Springer.

Examples

```
#############################################################################
# EXAMPLE 1: Dichotomous unidimensional data sim.rasch
#############################################################################

data(sim.rasch)
resp <- sim.rasch[ 1:500 , 1:15 ]  # select subsample of students and items

# estimate Rasch model
mod <- tam.mml(resp)

# draw 5 plausible values without a normality
# assumption of the posterior and 2000 ability nodes
pv1a <- tam.pv( mod , nplausible=5 , ntheta=2000 )

# draw 5 plausible values with a normality
# assumption of the posterior and 500 ability nodes
pv1b <- tam.pv( mod , nplausible=5 , ntheta=500 , normal.approx=TRUE )

# distribution of first plausible value from imputation pv1
hist(pv1a$pv$PV1.Dim1 )
# boxplot of all plausible values from imputation pv2
boxplot(pv1b$pv[ , 2:6 ] )

## Not run:
#############################################################################
# EXAMPLE 2: Unidimensional plausible value imputation with
#            background variables; dataset data.pisaRead from sirt package
#############################################################################

data(data.pisaRead, package="sirt")
dat <- data.pisaRead$data
  ##   > colnames(dat)
  ##    [1] "idstud"   "idschool" "female"   "hisei"    "migra"    "R432Q01"
  ##    [7] "R432Q05"  "R432Q06"  "R456Q01"  "R456Q02"  "R456Q06"  "R460Q01"
```

```
  ##   [13] "R460Q05"  "R460Q06"  "R466Q02"  "R466Q03"  "R466Q06"

## Note that reading items have variable names starting with R4

# estimate 2PL model without covariates
items <- grep("R4" , colnames(dat) ) # select test items from data
mod2a <- tam.mml.2pl( resp=dat[,items] )
summary(mod2a)

# fix item parameters for plausible value imputation
   # fix item intercepts by defining xsi.fixed
xsi0 <- mod2a$xsi$xsi
xsi.fixed <- cbind( seq(1,length(xsi0)) , xsi0 )
   # fix item slopes using mod2$B
# matrix of latent regressors female, hisei and migra
Y <- dat[ , c("female" , "hisei" , "migra") ]
mod2b <- tam.mml( resp=dat[,items] , B=mod2a$B , xsi.fixed=xsi.fixed , Y=Y)

# plausible value imputation with normality assumption
# and ignoring uncertainty about regression coefficients
#    -> the default is samp.regr=FALSE
pv2c <- tam.pv( mod2b , nplausible=10 , ntheta=500 )
# sampling of regression coefficients
pv2d <- tam.pv( mod2b , nplausible=10 , ntheta=500 , samp.regr=TRUE)

## End(Not run)
```

`tam.se` *Standard Error Estimation*

Description

Standard error computation for objects of the classes `tam` and `tam.mml`.

Usage

```
tam.se(tamobj, ...)

tam.mml.se(tamobj, numdiff.parm = 0.001)
```

Arguments

`tamobj`	An object generated by `tam.mml`
`numdiff.parm`	Step width parameter for numerical differentiation
`...`	Further arguments to be passed

Details

Covariances between parameters estimates are ignored in this standard error calculation. The standard error is obtained by numerical differentiation.

Value

A list with following entries:

`xsi`	Data frame with ξ parameters (`est`) and their corresponding standard errors (`se`)
`beta`	Data frame with β regression parameters and their standard error estimates
`B`	Data frame with loading parameters and their corresponding standard errors

Note

Standard error estimation for variances and covariances is not yet implemented. Standard error estimation for loading parameters in case of `irtmodel='GPCM.design'` is highly experimental.

Examples

```
#############################################################################
# EXAMPLE 1: 1PL model, sim.rasch data
#############################################################################

data(sim.rasch)
# estimate Rasch model
mod1 <- tam.mml(resp=sim.rasch[1:500,1:10])
# standard error estimation
se1 <- tam.se( mod1 )
# proportion of standard errors estimated by 'tam.se' and 'tam.mml'
prop1 <- se1$xsi$se / mod1$xsi$se
##   > summary( prop1 )
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     1.030   1.034   1.035   1.036   1.039   1.042
## => standard errors estimated by tam.se are a bit larger

## Not run:
#############################################################################
# EXAMPLE 2: Standard errors differential item functioning
#############################################################################
data(data.ex08)

formulaA <- ~ item*female
resp <- data.ex08[["resp"]]
facets <- as.data.frame( data.ex08[["facets"]] )
# investigate DIF
mod <- tam.mml.mfr( resp= resp , facets= facets , formulaA = formulaA )
summary(mod)
# estimate standard errors
semod <- tam.se(mod)
prop1 <- semod$xsi$se / mod$xsi$se
summary(prop1)
# plot differences in standard errors
plot( mod$xsi$se , semod$xsi$se , pch=16 , xlim=c(0,.15) , ylim=c(0,.15) ,
    xlab="Standard error 'tam.mml'" , ylab="Standard error 'tam.se'" )
lines( c(-6,6) , c(-6,6) , col="gray")
```

```
round( cbind( mod$xsi , semod$xsi[,-1] ) , 3 )
  ##                     xsi se.xsi   N    est    se
  ##   I0001         -1.956  0.092 500 -1.956 0.095
  ##   I0002         -1.669  0.085 500 -1.669 0.088
  ##   [...]
  ##   I0010          2.515  0.108 500  2.515 0.110
  ##   female1       -0.091  0.025 500 -0.091 0.041
  ##   I0001:female1 -0.051  0.070 500 -0.051 0.071
  ##   I0002:female1  0.085  0.067 500  0.085 0.068
  ##   [...]
  ##   I0009:female1 -0.019  0.068 500 -0.019 0.068
  ##
# => The largest discrepancy in standard errors is observed for the
#    main female effect (.041 in 'tam.se' instead of .025 in 'tam.mml')

## End(Not run)
```

`tam.threshold` *Calculation of Thurstonian Thresholds*

Description

This function estimates Thurstonian thresholds for item category parameters of (generalized) partial credit models (see Details).

Usage

```
tam.threshold(tamobj, prob.lvl=0.5)
```

Arguments

`tamobj`	Object of class `tam`
`prob.lvl`	A numeric specifying the probability level of the threshold. The default is `prob.lvl=0.5`.

Details

This function only works appropriately for unidimensional models or between item multidimensional models.

Value

A data frame with Thurstonian thresholds. Rows correspond to items and columns to item steps.

Examples

```
#############################################################################
# EXAMPLE 1: ordered data - Partial credit model
#############################################################################
data( data.gpcm )

# Model 1: partial credit model
mod1 <- tam.mml( resp=data.gpcm ,control=list( maxiter=200) )
summary(mod1)
  ##   Item Parameters -A*Xsi
  ##      item   N    M AXsi_.Cat1 AXsi_.Cat2 AXsi_.Cat3 B.Cat1.Dim1 B.Cat2.Dim1 B.Cat3.Dim1
  ##   1 Comfort 392 0.880    -1.302     1.154     3.881          1          2          3
  ##   2    Work 392 1.278    -1.706    -0.847     0.833          1          2          3
  ##   3 Benefit 392 1.163    -1.233    -0.404     1.806          1          2          3

# Calculation of Thurstonian thresholds
tam.threshold(mod1)
  ##                Cat1      Cat2     Cat3
  ##   Comfort -1.325226 2.0717468 3.139801
  ##   Work    -1.777679 0.6459045 1.971222
  ##   Benefit -1.343536 0.7491760 2.403168

## Not run:
#############################################################################
# EXAMPLE 2: Multidimensional model data.math
#############################################################################

library(sirt)
data(data.math, package="sirt")
dat <- data.math$data
# select items
items1 <- grep("M[A-D]" , colnames(dat) , value=TRUE)
items2 <- grep("M[H-I]" , colnames(dat) , value=TRUE)
# select dataset
dat <- dat[ c(items1,items2)]
# create Q-matrix
Q <- matrix( 0 , nrow=ncol(dat) , ncol=2 )
Q[ seq(1,length(items1) ) , 1 ] <- 1
Q[ length(items1) + seq(1,length(items2) ) , 2 ] <- 1

# fit two-dimensional model
mod1 <- tam.mml( dat , Q=Q )
# compute thresholds (specify a probability level of .625)
tmod1 <- tam.threshold( mod1  , prob.lvl = .625 )

## End(Not run)
```

`tam.wle` *Weighted Likelihood Estimation and Maximum Likelihood Estimation of Person Parameters*

Description

Compute the weighted likelihood estimator (Warm, 1989) for objects of classes `tam`, `tam.mml` and `tam.jml`, respectively.

Usage

```
tam.wle(tamobj, ...)

tam.mml.wle(tamobj, score.resp=NULL, WLE=TRUE, adj=0.3, Msteps=20, convM=1e-04)

tam.mml.wle2(tamobj, score.resp=NULL, WLE=TRUE, adj=0.3, Msteps=20, convM=1e-04)

tam.jml.WLE(tamobj, resp, resp.ind, A, B, nstud, nitems, maxK, convM,
    PersonScores, theta, xsi, Msteps, WLE=FALSE, theta.fixed = NULL)
```

Arguments

`tamobj`	An object generated by `tam.mml` or `tam.jml`
`score.resp`	An optional data frame for which WLEs or MLEs should be calculated. In case of the default `NULL`, `resp` from `tamobj` (i.e. `tamobj$resp`) is chosen. Note that items in `score.resp` must be the same (and in the same order) as in `tamobj$resp`.
`WLE`	A logical indicating whether the weighted likelihood estimate (WLE, `WLE=TRUE`) or the maximum likelihood estimate (MLE, `WLE=FALSE`) should be used.
`adj`	Adjustment in WLE estimation for extreme scores (i.e. all or none items were correctly solved)
`Msteps`	Maximum number of iterations
`convM`	Convergence criterion
`resp`	Data frame with item responses (only for `tam.jml.WLE`)
`resp.ind`	Data frame with response indicators (only for `tam.jml.WLE`)
`A`	Design matrix A (applies only to `tam.jml.WLE`)
`B`	Design matrix B (applies only to `tam.jml.WLE`)
`nstud`	Number of persons (applies only to `tam.jml.WLE`)
`nitems`	Number of items (applies only to `tam.jml.WLE`)
`maxK`	Maximum item score (applies only to `tam.jml.WLE`)
`PersonScores`	A vector containing the sufficient statistics for the person parameters (applies only to `tam.jml.WLE`)
`theta`	Initial θ estimate (applies only to `tam.jml.WLE`)
`xsi`	Parameter vector ξ (applies only to `tam.jml.WLE`)
`theta.fixed`	Matrix for fixed person parameters $theta$. The first column includes the index whereas the second column includes the fixed value.
`...`	Further arguments to be passed

Value

For `tam.wle.mml` and `tam.wle.mml2`, it is a data frame with following columns:

`pid`	Person identifier
`PersonScores`	Score of each person
`PersonMax`	Maximum score of each person
`theta`	Weighted likelihood estimate (WLE) or MLE
`error`	Standard error of the WLE or MLE
`WLE.rel`	WLE reliability (same value for all persons)

For `tam.jml.WLE`, it is a list with following entries:

`theta`	Weighted likelihood estimate (WLE) or MLE
`errorWLE`	Standard error of the WLE or MLE
`meanChangeWLE`	Mean change between updated and previous ability estimates from last iteration

References

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

Examples

```
#############################################################################
# EXAMPLE 1: 1PL model, sim.rasch data
#############################################################################
data(sim.rasch)
# estimate Rasch model
mod1 <- tam.mml(resp=sim.rasch)
# WLE estimation
wle1 <- tam.wle( mod1 )
  ## ----
  ## WLE Reliability = 0.894

# scoring for a different dataset containing same items (first 10 persons in
#  sim.rasch)
wle2 <- tam.wle( mod1 , score.resp=sim.rasch[1:10,])

## Not run:
# compare results to earlier wle-implementation
wle3 <- tam.mml.wle( mod1 )
stopifnot( all(wle3==wle1) )

#############################################################################
# EXAMPLE 2: 3-dimensional Rasch model | data.read from sirt package
#############################################################################

data(data.read , package="sirt")
```

```
# define Q-matrix
Q <- matrix(0,12,3)
Q[ cbind( 1:12 , rep(1:3,each=4) ) ] <- 1
# redefine data: create some missings for first three cases
resp <- data.read
resp[1:2 , 5:12] <- NA
resp[3,1:4] <- NA
  ##   > head(resp)
  ##      A1 A2 A3 A4 B1 B2 B3 B4 C1 C2 C3 C4
  ##   2   1  1  1  1 NA NA NA NA NA NA NA NA
  ##   22  1  1  0  0 NA NA NA NA NA NA NA NA
  ##   23 NA NA NA NA  1  0  1  1  1  1  1  1
  ##   41  1  1  1  1  1  1  1  1  1  1  1  1
  ##   43  1  0  0  1  0  0  1  1  1  0  1  0
  ##   63  1  1  0  0  1  0  1  1  1  1  1  1

# estimate 3-dimensional Rasch model
mod <- tam.mml( resp=resp , Q=Q , control=list(snodes=1000,maxiter=50) )
summary(mod)

# WLE estimates
wmod <- tam.wle(mod , Msteps=3)
summary(wmod)
  ##   head(round(wmod,2))
  ##      pid N.items PersonScores.Dim01 PersonScores.Dim02 PersonScores.Dim03
  ##   2    1       4               3.7               0.3               0.3
  ##   22   2       4               2.0               0.3               0.3
  ##   23   3       8               0.3               3.0               3.7
  ##   41   4      12               3.7               3.7               3.7
  ##   43   5      12               2.0               2.0               2.0
  ##   63   6      12               2.0               3.0               3.7
  ##      PersonMax.Dim01 PersonMax.Dim02 PersonMax.Dim03 theta.Dim01 theta.Dim02
  ##   2              4.0             0.6             0.6        1.06          NA
  ##   22             4.0             0.6             0.6       -0.96          NA
  ##   23             0.6             4.0             4.0          NA       -0.07
  ##   41             4.0             4.0             4.0        1.06        0.82
  ##   43             4.0             4.0             4.0       -0.96       -1.11
  ##   63             4.0             4.0             4.0       -0.96       -0.07
  ##      theta.Dim03 error.Dim01 error.Dim02 error.Dim03 WLE.rel.Dim01
  ##   2           NA        1.50          NA          NA          -0.1
  ##   22          NA        1.11          NA          NA          -0.1
  ##   23        0.25          NA        1.17        1.92          -0.1
  ##   41        0.25        1.50        1.48        1.92          -0.1
  ##   43       -1.93        1.11        1.10        1.14          -0.1

# (1) Note that estimated WLE reliabilities are not trustworthy in this example.
# (2) If cases do not possess any observations on dimensions, then WLEs
#     and their corresponding standard errors are set to NA.

## End(Not run)
```

Index

∗Topic **3PL model**
tam.mml.3pl, 72
∗Topic **Bifactor model**
tam.fa, 35
∗Topic **Classical test theory statistics**
tam.ctt, 32
∗Topic **Design matrices**
designMatrices, 19
∗Topic **Exploratory factor analysis**
tam.fa, 35
∗Topic **Factor scores**
IRT.factor.scores, 22
∗Topic **Fit statistics**
tam.fit, 37
∗Topic **Individual likelihood**
IRT.likelihood, 23
∗Topic **Individual posterior**
IRT.likelihood, 23
∗Topic **Infit**
tam.fit, 37
∗Topic **Item fit**
tam.fit, 37
∗Topic **Item response functions**
IRT.irfprob, 23
∗Topic **Joint maximum likelihood estimation (JML)**
tam.jml, 41
∗Topic **Likelihood ratio test**
anova-logLik, 3
∗Topic **MLE**
tam.wle, 93
∗Topic **Marginal maximum likelihood estimation (MML)**
tam.mml, 46
∗Topic **Model fit**
tam.modelfit, 85
∗Topic **Multidimensional item response model**
tam.mml, 46
∗Topic **Outfit**
tam.fit, 37
∗Topic **Person parameter estimation**
tam.wle, 93
∗Topic **Plausible value imputation**
tam.pv, 88
∗Topic **Q3 statistic**
tam.modelfit, 85
∗Topic **Standard errors**
tam.se, 90
∗Topic **Structured item response model**
tam.mml.3pl, 72
∗Topic **Testlet model**
tam.fa, 35
∗Topic **Thurstonian thresholds**
tam.threshold, 92
∗Topic **WLE**
tam.wle, 93
∗Topic **anova**
anova-logLik, 3
∗Topic **datasets**
data.cqc, 5
data.ctest, 9
data.examples, 10
data.fims.Aus.Jpn.scored, 12
data.gpcm, 14
data.mc, 14
data.numeracy, 15
data.timssAusTwn, 17
sim.mfr, 29
sim.rasch, 31
∗Topic **lavaan**
lavaanify.IRT, 24
∗Topic **logLik**
anova-logLik, 3
∗Topic **package**
TAM-package, 2
∗Topic **plot**

plot.tam, 27
plotDevianceTAM, 28
tam.ctt, 32
∗Topic **summary**
tam.jml, 41
tam.mml, 46
tam.modelfit, 85
.A.PCM2 *(*designMatrices*)*, 19
.A.PCM3 *(*designMatrices*)*, 19
.A.matrix *(*designMatrices*)*, 19

anova-logLik, 3
anova.tam, *36*, *53*, *75*
anova.tam *(*anova-logLik*)*, 3

data.cqc, 5
data.cqc01, *52*
data.cqc01 *(*data.cqc*)*, 5
data.cqc02 *(*data.cqc*)*, 5
data.cqc03 *(*data.cqc*)*, 5
data.cqc04 *(*data.cqc*)*, 5
data.cqc05 *(*data.cqc*)*, 5
data.ctest, 9
data.ctest1 *(*data.ctest*)*, 9
data.ex08 *(*data.examples*)*, 10
data.ex10 *(*data.examples*)*, 10
data.ex11 *(*data.examples*)*, 10
data.ex12 *(*data.examples*)*, 10
data.ex14 *(*data.examples*)*, 10
data.ex15 *(*data.examples*)*, 10
data.examples, 10
data.exJ03 *(*data.examples*)*, 10
data.fims.Aus.Jpn.raw *(*data.fims.Aus.Jpn.scored*)*, 12
data.fims.Aus.Jpn.scored, 12
data.gpcm, 14
data.mc, 14
data.numeracy, 15
data.timssAusTwn, 17
designMatrices, 19
dev.new, *28*

IRT.factor.scores, *22*, 22
IRT.irfprob, *23*, 23
IRT.likelihood, *23*, 23, *24*
IRT.posterior.tam *(*IRT.likelihood*)*, 23

lavaan2mirt, *25*
lavaanify, *24*, *25*
lavaanify.IRT, 24
logLik.tam, *36*, *53*, *75*
logLik.tam *(*anova-logLik*)*, 3

mirt, *53*

plot.tam, 27
plotctt *(*tam.ctt*)*, 32
plotDevianceTAM, 28
print.designMatrices *(*designMatrices*)*, 19

rownames.design *(*designMatrices*)*, 19

sim.facets *(*sim.mfr*)*, 29
sim.mfr, *21*, 29
sim.rasch, 31
slca, *75*
summary.tam *(*tam.mml*)*, 46
summary.tam.jml *(*tam.jml*)*, 41
summary.tam.modelfit *(*tam.modelfit*)*, 85

TAM *(*TAM-package*)*, 2
tam, *3*, *4*, *22–24*
tam *(*tam.mml*)*, 46
TAM-package, 2
tam.ctt, 32
tam.ctt2 *(*tam.ctt*)*, 32
tam.ctt3 *(*tam.ctt*)*, 32
tam.fa, *3*, 35
tam.fit, 37
tam.jml, *11*, *38*, 41, *52*
tam.jml.fit *(*tam.fit*)*, 37
tam.jml.WLE *(*tam.wle*)*, 93
tam.jml2 *(*tam.jml*)*, 41
tam.mml, *3*, *4*, *10–12*, *22–24*, *35*, *36*, *42*, *43*, 46, *74*, *75*
tam.mml.2pl, *3*, *35*, *36*
tam.mml.3pl, *3*, *4*, *22–24*, 72
tam.mml.fit *(*tam.fit*)*, 37
tam.mml.mfr, *3*
tam.mml.se *(*tam.se*)*, 90
tam.mml.wle *(*tam.wle*)*, 93
tam.mml.wle2 *(*tam.wle*)*, 93
tam.modelfit, 85
tam.pv, 88
tam.se, *52*, 90
tam.threshold, 92
tam.wle, *22*, 93
tam2mirt, *25*, *53*