Package ‘TAM’

February 19, 2024

Type Package

Title Test Analysis Modules

Version 4.2-21

Date 2024-02-19 18:52:08

Author Alexander Robitzsch [aut,cre] (<https://orcid.org/0000-0002-8226-3132>),
Thomas Kiefer [aut],
Margaret Wu [aut]

Maintainer Alexander Robitzsch <robitzsch@ipn.uni-kiel.de>

Description Includes marginal maximum likelihood estimation and joint maximum likelihood estimation for unidimensional and multidimensional item response models. The package functionality covers the Rasch model, 2PL model, 3PL model, generalized partial credit model, multi-faceted Rasch model, nominal item response model, structured latent class model, mixture distribution IRT models, and located latent class models. Latent regression models and plausible value imputation are also supported. For details see Adams, Wilson and Wang, 1997 <doi:10.1177/0146621697211001>, Adams, Wilson and Wu, 1997 <doi:10.3102/10769986022001047>, Formann, 1982 <doi:10.1002/bimj.4710240209>, Formann, 1992 <doi:10.1080/01621459.1992.10475229>.

Depends R (>= 2.15.1), CDM (>= 6.4-19)

Imports graphics, methods, Rcpp, stats, utils

Suggests coda, GPArotation, grDevices, lattice, lavaan, MASS, miceadds, mvtnorm, plyr, psych, sfsmisc, splines, WrightMap

LinkingTo Rcpp, RcppArmadillo

Enhances LSAmitR

URL http://www.edmeasurementsurveys.com/TAM/Tutorials/,
https://github.com/alexanderrobitzsch/TAM,
https://sites.google.com/view/alexander-robitzsch/software

NeedsCompilation yes

Repository CRAN

Date/Publication 2024-02-19 18:40:02 UTC

R topics documented:

`TAM-package` *Test Analysis Modules*

Description

Includes marginal maximum likelihood estimation and joint maximum likelihood estimation for unidimensional and multidimensional item response models. The package functionality covers the Rasch model, 2PL model, 3PL model, generalized partial credit model, multi-faceted Rasch model, nominal item response model, structured latent class model, mixture distribution IRT models, and located latent class models. Latent regression models and plausible value imputation are also supported. For details see Adams, Wilson and Wang, 1997 <doi:10.1177/0146621697211001>, Adams, Wilson and Wu, 1997 <doi:10.3102/10769986022001047>, Formann, 1982 <doi:10.1002/bimj.4710240209>, Formann, 1992 <doi:10.1080/01621459.1992.10475229>.

Details

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

Author(s)

Alexander Robitzsch [aut,cre] (<https://orcid.org/0000-0002-8226-3132>), Thomas Kiefer [aut], Margaret Wu [aut]

Maintainer: Alexander Robitzsch <robitzsch@ipn.uni-kiel.de>

References

Adams, R. J., Wilson, M., & Wang, W. C. (1997). The multidimensional random coefficients multinomial logit model. *Applied Psychological Measurement, 21*(1), 1-23. doi:10.1177/0146621697211001

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*(1), 47-76. doi:10.3102/10769986022001047

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. doi:10.1007/9780387498393_4

Formann, A. K. (1982). Linear logistic latent class analysis. *Biometrical Journal, 24*(2), 171-190. doi:10.1002/bimj.4710240209

Formann, A. K. (1992). Linear logistic latent class analysis for polytomous data. *Journal of the American Statistical Association, 87*(418), 476-486. doi:10.1080/01621459.1992.10475229

---

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`, `tam.mml.3pl`, `tam.latreg` or `tamaan`.

Usage

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

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

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

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

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

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

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

Arguments

object	Object of class tam, tam.mml, tam.mml.3pl, tam.latreg, tam.np, or tamaan. 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(data.sim.rasch)
# 1PL estimation
mod1 <- TAM::tam.mml(resp=data.sim.rasch)
logLik(mod1)
# 2PL estimation
mod2 <- TAM::tam.mml.2pl(resp=data.sim.rasch, irtmodel="2PL")
logLik(mod2)
# Model comparison
anova( mod1, mod2 )
  ##     Model   loglike Deviance Npars      AIC      BIC    Chisq df       p
  ##   1  mod1 -42077.88 84155.77    41 84278.77 84467.40 54.05078 39 0.05508
  ##   2  mod2 -42050.86 84101.72    80 84341.72 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::tam.mml.2pl(resp=data.read )
# 2-dimensional model
mod2 <- TAM::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    mod1 -1954.888 3909.777    24 3957.777 4048.809 76.66491 11  0
  ##    2    mod2 -1916.556 3833.112    35 3903.112 4035.867       NA NA NA

## End(Not run)
```

`cfa.extract.itempars` *Extracting Item Parameters from a Fitted* `cfa` *Object in* **lavaan**

Description

This function extract item parameters from a fitted `lavaan::cfa` object in **lavaan**. It extract item loadings, item intercepts and the mean and covariance matrix of latent variables in a confirmatory factor analysis model.

Usage

```
cfa.extract.itempars(object)
```

Arguments

`object`	Fitted `cfa` object

Value

List with following entries

`L`	Matrix of item loadings
`nu`	Vector of item intercepts
`psi`	Residual covariance matrix
`Sigma`	Covariance matrix of latent variables
`nu`	Vector of means of latent variables
`...`	Further values

See Also

See IRTLikelihood.cfa for extracting the individual likelihood from fitted confirmatory factor analyses.

lavaan::cfa

Examples

```
#############################################################################
# EXAMPLE 1: CFA data.Students
#############################################################################

library(lavaan)
library(CDM)

data(data.Students, package="CDM")
dat <- data.Students

dat1 <- dat[, paste0( "mj", 1:4 ) ]

#*** Model 1: Unidimensional model scale mj
lavmodel <- "
   mj=~ mj1 + mj2 + mj3 + mj4
   mj ~~ mj
     "
mod1 <- lavaan::cfa( lavmodel, data=dat1, std.lv=TRUE )
summary(mod1, standardized=TRUE, rsquare=TRUE )
# extract parameters
res1 <- TAM::cfa.extract.itempars( mod1 )

## Not run:
#*** Model 2: Scale mj - explicit modelling of item intercepts
lavmodel <- "
   mj=~ mj1 + mj2 + mj3 + mj4
   mj ~~ mj
   mj1 ~ 1
     "
mod2 <- lavaan::cfa( lavmodel, data=dat1, std.lv=TRUE )
summary(mod2, standardized=TRUE, rsquare=TRUE )
res2 <- TAM::cfa.extract.itempars( mod2 )

#*** Model 3: Tau-parallel measurements scale mj
lavmodel <- "
   mj=~ a*mj1 + a*mj2 + a*mj3 + a*mj4
   mj ~~ 1*mj
   mj1 ~ b*1
   mj2 ~ b*1
   mj3 ~ b*1
   mj4 ~ b*1
     "
mod3 <- lavaan::cfa( lavmodel, data=dat1, std.lv=TRUE )
summary(mod3, standardized=TRUE, rsquare=TRUE )
res3 <- TAM::cfa.extract.itempars( mod3 )
```

```
#*** Model 4: Two-dimensional CFA with scales mj and sc
dat2 <- dat[, c(paste0("mj",1:4), paste0("sc",1:4)) ]
# lavaan model with shortage "__" operator
lavmodel <- "
   mj=~ mj1__mj4
   sc=~ sc1__sc4
   mj ~~ sc
   mj ~~ 1*mj
   sc ~~ 1*sc
     "
lavmodel <- TAM::lavaanify.IRT( lavmodel, data=dat2 )$lavaan.syntax
cat(lavmodel)
mod4 <- lavaan::cfa( lavmodel, data=dat2, std.lv=TRUE )
summary(mod4, standardized=TRUE, rsquare=TRUE )
res4 <- TAM::cfa.extract.itempars( mod4 )

## 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 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.

See Also

See the `sirt::R2conquest` function for running ConQuest software from within R.

See the **WrightMap** package for functions connected to reading ConQuest files and creating Wright maps. ConQuest output files can be read into R with the help of the `WrightMap::CQmodel` function. See also the `IRT.WrightMap` function in **TAM**.

See also the **eat** package (https://r-forge.r-project.org/projects/eat/) for elaborate functionality for communication of ConQuest with R.

Examples

```
## Not run:

library(sirt)
library(WrightMap)
# In the following, ConQuest will also be used for estimation.
path.conquest <- "C:/Conquest"             # path of the ConQuest console.exe
setwd( "p:/my_files/ConQuest_analyses" )  # working directory

#############################################################################
# EXAMPLE 01: Rasch model data.cqc01
#############################################################################

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

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

#------- ConQuest

# estimate model
cmod01 <- sirt::R2conquest( dat, name="mod01", path.conquest=path.conquest)
summary(cmod01)   # summary output
# read shw file with some terms
shw01a <- sirt::read.show( "mod01.shw" )
cmod01$shw.itemparameter
# read person item maps
pi01a <- sirt::read.pimap( "mod01.shw" )
cmod01$shw.pimap
# read plausible values (npv=10 plausible values)
pv01a <- sirt::read.pv(pvfile="mod01.pv", npv=10)
cmod01$person

# read ConQuest model
res01a <- WrightMap::CQmodel(p.est="mod01.wle", show="mod01.shw", p.type="WLE" )
print(res01a)
# plot item fit
WrightMap::fitgraph(res01a)
# Wright map
plot(res01a, label.items.srt=90 )

#############################################################################
# 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::tam.mml( dat, irtmodel="PCM2" )
summary(mod02a, file="mod02a")
fit02a <- TAM::tam.fit(mod02a)
summary(fit02a)

#--- ConQuest
# estimate model
maxK <- max( dat, na.rm=TRUE )
cmod02a <- sirt::R2conquest( dat, itemcodes=0:maxK, model="item+item*step",
               name="mod02a", path.conquest=path.conquest)
summary(cmod02a)   # summary output

# read ConQuest model
res02a <- WrightMap::CQmodel(p.est="mod02a.wle", show="mod02a.shw", p.type="WLE" )
print(res02a)
# Wright map
plot(res02a, label.items.srt=90 )
plot(res02a, item.table="item")

#*********************************************
# Model 02b: Rating scale model
mod02b <- TAM::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::tam.mml.mfr( resp, facets=data.cqc03[,"rater",drop=FALSE],
            formulaA=~ item+step+rater, pid=data.cqc03$pid )
summary( mod03a )

#--- ConQuest
X <- data.cqc03[,"rater",drop=FALSE]
X$rater <- as.numeric(substring( X$rater, 2 )) # convert 'rater' in numeric format
maxK <- max( resp, na.rm=TRUE)
cmod03a <- sirt::R2conquest( resp,  X=X, regression="",  model="item+step+rater",
             name="mod03a", path.conquest=path.conquest, set.constraints="cases" )
summary(cmod03a)   # summary output

# read ConQuest model
res03a <- WrightMap::CQmodel(p.est="mod03a.wle", show="mod03a.shw", p.type="WLE" )
print(res03a)
```

```
# Wright map
plot(res03a)

#*********************************************
# Model 03b: 'item:step+rater'
mod03b <- TAM::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::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::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::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::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$xsi )
rownames(xsi.fixed) <- rownames(xsi.est)
xsi.fixed <- xsi.fixed[ c(8:13),]
xsi.fixed[,2] <- 0
  ##   > xsi.fixed
  ##            [,1] [,2]
  ##   raterAM      8    0
  ##   raterBE      9    0
  ##   raterCO     10    0
  ##   topicFami   11    0
  ##   topicScho   12    0
  ##   topicSpor   13    0
mod04c1 <- TAM::tam.mml.mfr( resp, facets=facets,
             formulaA=formulaA, pid=data.cqc04$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

#********************************************
# Model 05a: Partial credit model
mod05a <-tam.mml(resp=resp, irtmodel="PCM2" )

#********************************************
# Model 05b: Partial credit model with latent regressors
mod05b <-tam.mml(resp=resp, irtmodel="PCM2",  Y=data.cqc05[,1:3] )
# Plausible value imputation
pvmod05b <- TAM::tam.pv( mod05b )

## End(Not run)
```

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

Description

Some C-Test datasets.

Usage

```
data(data.ctest1)
data(data.ctest2)
```

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 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 ...
```

- The dataset `data.ctest2` contains two datasets (`$data1` containing item responses, `$data2` containing sum scores of each C-test) and a data frame `$ITEM` with item informations.

```
List of 3
$ data1:'data.frame': 933 obs. of 102 variables:
..$ idstud: num [1:933] 10001 10002 10003 10004 10005 ...
..$ female: num [1:933] 1 1 0 0 0 0 1 1 0 1 ...
..$ A101 : int [1:933] NA NA NA NA NA NA NA NA NA NA ...
..$ A102 : int [1:933] NA NA NA NA NA NA NA NA NA NA ...
..$ A103 : int [1:933] NA NA NA NA NA NA NA NA NA NA ...
..$ A104 : int [1:933] NA NA NA NA NA NA NA NA NA NA ...
..$ A105 : int [1:933] NA NA NA NA NA NA NA NA NA NA ...
..$ A106 : int [1:933] NA NA NA NA NA NA NA NA NA NA ...
..$ E115 : int [1:933] NA NA NA NA NA NA NA NA NA NA ...
..$ E116 : int [1:933] NA NA NA NA NA NA NA NA NA NA ...
..$ E117 : int [1:933] NA NA NA NA NA NA NA NA NA NA ...
.. [list output truncated]
```

```
$ data2:'data.frame': 933 obs. of 7 variables:
..$ idstud: num [1:933] 10001 10002 10003 10004 10005 ...
..$ female: num [1:933] 1 1 0 0 0 0 1 1 0 1 ...
..$ A : num [1:933] NA NA NA NA NA NA NA NA NA NA ...
..$ B : num [1:933] 16 14 15 13 17 11 11 18 19 13 ...
..$ C : num [1:933] 17 15 17 14 17 13 9 15 17 12 ...
..$ D : num [1:933] NA NA NA NA NA NA NA NA NA NA ...
..$ E : num [1:933] NA NA NA NA NA NA NA NA NA NA ...
$ ITEM :'data.frame': 100 obs. of 3 variables:
..$ item : chr [1:100] "A101" "A102" "A103" "A104" ...
..$ ctest : chr [1:100] "A" "A" "A" "A" ...
..$ testlet: int [1:100] 1 1 2 2 2 3 3 3 NA 4 ...
```

`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 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).

- Data `data.ex16` contains dichotomous item response data from three studies corresponding to three grades.
  'data.frame': 3235 obs. of 25 variables:
  $ idstud: num 1e+05 1e+05 1e+05 1e+05 1e+05 ...
  $ grade : num 1 1 1 1 1 1 1 1 1 1 ...
  $ A1 : int 1 1 1 1 1 1 1 1 1 1 ...
  $ B1 : int 1 1 1 1 1 1 1 1 1 1 ...
  $ C1 : int 1 1 1 1 1 1 1 1 1 1 ...
  $ D1 : int 1 1 1 1 1 1 1 1 1 1 ...
  $ E1 : int 0 0 1 0 1 1 1 0 1 1 ...
  $ E2 : int 1 1 1 1 1 1 1 0 1 1 ...
  $ E3 : int 1 1 1 1 1 1 1 0 1 1 ...
  $ F1 : int 1 0 1 1 0 0 1 0 1 1 ...
  $ G1 : int 0 1 1 1 1 0 1 0 1 1 ...
  $ G2 : int 1 1 1 1 1 0 0 0 1 1 ...
  $ G3 : int 1 0 1 1 1 0 0 0 1 1 ...
  $ H1 : int 1 0 1 1 1 0 0 0 1 1 ...
  $ H2 : int 1 0 1 1 1 0 0 0 1 1 ...
  $ I1 : int 1 0 1 0 1 0 0 0 1 1 ...
  $ I2 : int 1 0 1 0 1 0 0 0 1 1 ...
  $ J1 : int NA NA NA NA NA NA NA NA NA NA ...
  $ K1 : int NA NA NA NA NA NA NA NA NA NA ...
  $ L1 : int NA NA NA NA NA NA NA NA NA NA ...
  $ L2 : int NA NA NA NA NA NA NA NA NA NA ...
  $ L3 : int NA NA NA NA NA NA NA NA NA NA ...

```
$ M1 : int NA NA NA NA NA NA NA NA NA NA ...
$ M2 : int NA NA NA NA NA NA NA NA NA NA ...
$ M3 : int NA NA NA NA NA NA NA NA NA NA ...
```

- Data `data.ex17` contains polytomous item response data from three studies corresponding to three grades.

```
'data.frame': 3235 obs. of 15 variables:
$ idstud: num 1e+05 1e+05 1e+05 1e+05 1e+05 ...
$ grade : num 1 1 1 1 1 1 1 1 1 1 ...
$ A : int 1 1 1 1 1 1 1 1 1 1 ...
$ B : int 1 1 1 1 1 1 1 1 1 1 ...
$ C : int 1 1 1 1 1 1 1 1 1 1 ...
$ D : int 1 1 1 1 1 1 1 1 1 1 ...
$ E : num 2 2 3 2 3 3 3 0 3 3 ...
$ F : int 1 0 1 1 0 0 1 0 1 1 ...
$ G : num 2 2 3 3 3 0 1 0 3 3 ...
$ H : num 2 0 2 2 2 0 0 0 2 2 ...
$ I : num 2 0 2 0 2 0 0 0 2 2 ...
$ J : int NA NA NA NA NA NA NA NA NA NA ...
$ K : int NA NA NA NA NA NA NA NA NA NA ...
$ L : num NA NA NA NA NA NA NA NA NA NA ...
$ M : num NA NA NA NA NA NA NA NA NA NA ...
```

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 – male, 2 – female

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::tam.mml(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
stats::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::tam.mml.mfr( resp=scored[,items], facets=facets, formulaA=formulaA )
summary(mod2)

## End(Not run)
```

`data.geiser` *Dataset from Geiser et al. (2006)*

Description

This is a subsample of the dataset used in Geiser et al. (2006) and Geiser and Eid (2010).

Usage

```
data(data.geiser)
```

Format

A data frame with 519 observations on the following 24 variables

```
'data.frame': 519 obs. of 24 variables:
$ mrt1 : num 0 0 0 0 0 0 0 0 0 0 ...
$ mrt2 : num 0 0 0 0 0 0 0 0 0 0 ...
$ mrt3 : num 0 0 0 0 0 0 0 0 1 0 ...
$ mrt4 : num 0 0 0 0 0 1 0 0 0 0 ...
[...]
$ mrt23: num 0 0 0 0 0 0 0 1 0 0 ...
$ mrt24: num 0 0 0 0 0 0 0 0 0 0 ...
```

References

Geiser, C., & Eid, M. (2010). Item-Response-Theorie. In C. Wolf & H. Best (Hrsg.). *Handbuch der sozialwissenschaftlichen Datenanalyse* (S. 311-332). VS Verlag fuer Sozialwissenschaften.

Geiser, C., Lehmann, W., & Eid, M. (2006). Separating rotators from nonrotators in the mental rotations test: A multigroup latent class analysis. *Multivariate Behavioral Research, 41*(3), 261-293. doi:10.1207/s15327906mbr4103_2

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: Latent trait and latent class models (Geiser et al., 2006, MBR)
#############################################################################

data(data.geiser)
dat <- data.geiser

#*********************************************
# Model 1: Rasch model
tammodel <- "
  LAVAAN MODEL:
    F=~ 1*mrt1__mrt12
    F ~~ F
  ITEM TYPE:
    ALL(Rasch)
    "
mod1 <- TAM::tamaan( tammodel, dat)
summary(mod1)

#*********************************************
# Model 2: 2PL model
tammodel <- "
  LAVAAN MODEL:
    F=~ mrt1__mrt12
    F ~~ 1*F
    "
mod2 <- TAM::tamaan( tammodel, dat)
summary(mod2)

# model comparison Rasch vs. 2PL
anova(mod1,mod2)

#*********************************************************************
#*** Model 3: Latent class analysis with four classes

tammodel <- "
ANALYSIS:
  TYPE=LCA;
  NCLASSES(4);   # 4 classes
  NSTARTS(10,20); # 10 random starts with 20 iterations
LAVAAN MODEL:
```

```
  F=~ mrt1__mrt12
    "
mod3 <- TAM::tamaan( tammodel, resp=dat  )
summary(mod3)

# extract item response functions
imod2 <- IRT.irfprob(mod3)[,2,]
# plot class specific probabilities
matplot( imod2, type="o", pch=1:4, xlab="Item", ylab="Probability" )
legend( 10,1, paste0("Class",1:4), lty=1:4, col=1:4, pch=1:4 )

#*********************************************************************
#*** Model 4: Latent class analysis with five classes

tammodel <- "
ANALYSIS:
  TYPE=LCA;
  NCLASSES(5);
  NSTARTS(10,20);
LAVAAN MODEL:
  F=~ mrt1__mrt12
    "
mod4 <- TAM::tamaan( tammodel, resp=dat )
summary(mod4)

# compare different models
AIC(mod1); AIC(mod2); AIC(mod3); AIC(mod4)
BIC(mod1); BIC(mod2); BIC(mod3); BIC(mod4)
# more condensed form
IRT.compareModels(mod1, mod2, mod3, mod4)

#############################################################################
# EXAMPLE 2: Rasch model and mixture Rasch model (Geiser & Eid, 2010)
#############################################################################

data(data.geiser)
dat <- data.geiser

#*********************************************************************
#*** Model 1: Rasch model
tammodel <- "
LAVAAN MODEL:
  F=~ mrt1__mrt6
  F ~~ F
ITEM TYPE:
  ALL(Rasch);
    "
mod1 <- TAM::tamaan( tammodel, resp=dat  )
summary(mod1)

#*********************************************************************
#*** Model 2: Mixed Rasch model with two classes
tammodel <- "
```

```
ANALYSIS:
  TYPE=MIXTURE ;
  NCLASSES(2);
  NSTARTS(20,25);
LAVAAN MODEL:
  F=~ mrt1__mrt6
  F ~~ F
ITEM TYPE:
  ALL(Rasch);
    "
mod2 <- TAM::tamaan( tammodel, resp=dat  )
summary(mod2)

# plot item parameters
ipars <- mod2$itempartable_MIXTURE[ 1:6, ]
plot( 1:6, ipars[,3], type="o", ylim=c(-3,2), pch=16,
        xlab="Item", ylab="Item difficulty")
lines( 1:6, ipars[,4], type="l", col=2, lty=2)
points( 1:6, ipars[,4],  col=2, pch=2)

# extract individual posterior distribution
post2 <- IRT.posterior(mod2)
str(post2)
# num [1:519, 1:30] 0.000105 0.000105 0.000105 0.000105 0.000105 ...
# - attr(*, "theta")=num [1:30, 1:30] 1 0 0 0 0 0 0 0 0 0 ...
# - attr(*, "prob.theta")=num [1:30, 1] 1.21e-05 2.20e-04 2.29e-03 1.37e-02 4.68e-02 ...
# - attr(*, "G")=num 1

# There are 2 classes and 15 theta grid points for each class
# The loadings of the theta grid on items are as follows
mod2$E[1,2,,"mrt1_F_load_Cl1"]
mod2$E[1,2,,"mrt1_F_load_Cl2"]

# compute individual posterior probability for class 1 (first 15 columns)
round( rowSums( post2[, 1:15] ), 3 )
# columns 16 to 30 refer to class 2

#**********************************************************************
#*** Model 3: Mixed Rasch model with three classes
tammodel <- "
ANALYSIS:
  TYPE=MIXTURE ;
  NCLASSES(3);
  NSTARTS(20,25);
LAVAAN MODEL:
  F=~ mrt1__mrt6
  F ~~ F
ITEM TYPE:
  ALL(Rasch);
    "
mod3 <- TAM::tamaan( tammodel, resp=dat  )
summary(mod3)
```

```
# plot item parameters
ipars <- mod3$itempartable_MIXTURE[ 1:6, ]
plot( 1:6, ipars[,3], type="o", ylim=c(-3.7,2), pch=16,
        xlab="Item", ylab="Item difficulty")
lines( 1:6, ipars[,4], type="l", col=2, lty=2)
points( 1:6, ipars[,4],  col=2, pch=2)
lines( 1:6, ipars[,5], type="l", col=3, lty=3)
points( 1:6, ipars[,5],  col=3, pch=17)

# model comparison
IRT.compareModels( mod1, mod2, mod3 )

## 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.janssen` *Dataset from Janssen and Geiser (2010)*

Description

Dataset used in Janssen and Geiser (2010).

Usage

```
data(data.janssen)
data(data.janssen2)
```

Format

- `data.janssen` is a data frame with 346 observations on the 8 items of the following format

  ```
  'data.frame': 346 obs. of 8 variables:
  $ PIS1 : num 1 1 1 0 0 1 1 1 0 1 ...
  $ PIS3 : num 0 1 1 1 1 1 0 1 1 1 ...
  $ PIS4 : num 1 1 1 1 1 1 1 1 1 1 ...
  $ PIS5 : num 0 1 1 0 1 1 1 1 1 0 ...
  $ SCR6 : num 1 1 1 1 1 1 1 1 1 0 ...
  $ SCR9 : num 1 1 1 1 0 0 0 1 0 0 ...
  $ SCR10: num 0 0 0 0 0 0 0 0 0 0 ...
  $ SCR17: num 0 0 0 0 0 1 0 0 0 0 ...
  ```

- `data.janssen2` contains 20 IST items:

  ```
  'data.frame': 346 obs. of 20 variables:
  $ IST01 : num 1 1 1 0 0 1 1 1 0 1 ...
  $ IST02 : num 1 0 1 0 1 1 1 1 0 1 ...
  $ IST03 : num 0 1 1 1 1 1 0 1 1 1 ...
  [...]
  $ IST020: num 0 0 0 1 1 0 0 0 0 0 ...
  ```

References

Janssen, A. B., & Geiser, C. (2010). On the relationship between solution strategies in two mental rotation tasks. *Learning and Individual Differences, 20*(5), 473-478. doi:10.1016/j.lindif.2010.03.002

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: CCT data, Janssen and Geiser (2010, LID)
#            Latent class analysis based on data.janssen
#############################################################################
```

```
data(data.janssen)
dat <- data.janssen
colnames(dat)
  ##   [1] "PIS1"  "PIS3"  "PIS4"  "PIS5"  "SCR6"  "SCR9"  "SCR10" "SCR17"

#**********************************************************************
#*** Model 1: Latent class analysis with two classes

tammodel <- "
ANALYSIS:
  TYPE=LCA;
  NCLASSES(2);
  NSTARTS(10,20);
LAVAAN MODEL:
  # missing item numbers (e.g. PIS2) are ignored in the model
  F=~ PIS1__PIS5 + SCR6__SCR17
    "
mod3 <- TAM::tamaan( tammodel, resp=dat  )
summary(mod3)

# extract item response functions
imod2 <- IRT.irfprob(mod3)[,2,]
# plot class specific probabilities
ncl <- 2
matplot( imod2, type="o", pch=1:ncl, xlab="Item", ylab="Probability" )
legend( 1, .3, paste0("Class",1:ncl), lty=1:ncl, col=1:ncl, pch=1:ncl )

#**********************************************************************
#*** Model 2: Latent class analysis with three classes

tammodel <- "
ANALYSIS:
  TYPE=LCA;
  NCLASSES(3);
  NSTARTS(10,20);
LAVAAN MODEL:
  F=~ PIS1__PIS5 + SCR6__SCR17
    "
mod3 <- TAM::tamaan( tammodel, resp=dat  )
summary(mod3)

# extract item response functions
imod2 <- IRT.irfprob(mod3)[,2,]
# plot class specific probabilities
ncl <- 3
matplot( imod2, type="o", pch=1:ncl, xlab="Item", ylab="Probability" )
legend( 1, .3, paste0("Class",1:ncl), lty=1:ncl, col=1:ncl, pch=1:ncl )

# compare models
AIC(mod1); AIC(mod2)

## End(Not run)
```

`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 1 ...
..$ I02: num [1:143] 1 1 1 0 1 1 1 1 1 1 ...
..$ I03: num [1:143] 1 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::tam.mml(dat)

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

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

#Descriptive statistics of item and person parameters
hist(ItemDiff)
hist(PersonAbility)
mean(ItemDiff)
mean(PersonAbility)
stats::sd(ItemDiff)
stats::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::tam.mml(scored)

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

#CTT statistics
ctt1 <- TAM::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::tam.fit(mod1)
Fit

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

## End(Not run)
```

`data.sim.mfr` *Simulated Multifaceted Data*

Description

Simulated data from multiple facets.

Usage

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

Format

The format of `data.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 `data.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(data.sim.mfr)
data(data.sim.facets)

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

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

  # 3: test_item+rater+topic+ratertopic
  test_3_items <- TAM::.A.matrix( data.sim.mfr, formulaA=~item+rater*topic,
            facets=data.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 <- TAM::.A.matrix( data.sim.mfr, formulaA=~item+step,
            facets=data.sim.facets, constraint="cases" )
```

```
  # 5: test_item+item:step
  test_5_cases <- TAM::.A.matrix( data.sim.mfr, formulaA=~item+item:step,
            facets=data.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 <- TAM::.A.matrix( data.sim.mfr, formulaA=~item+topic:step,
  #                 facets=data.sim.facets, constraint="cases" )
  test_7_cases <- TAM::.A.matrix( data.sim.mfr, formulaA=~item+topic+topic:step,
            facets=data.sim.facets, constraint="cases" )

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

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

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

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

`data.sim.rasch` *Simulated Rasch data*

Description

Simulated Rasch data under unidimensional trait distribution

Usage

```
data(data.sim.rasch)
data(data.sim.rasch.pweights)
data(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( stats::rnorm( N, sd=1.5), stats::rnorm(N, sd=.3 ) )
theta <- stats::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(data.sim.rasch)
N <- 2000
Y <- cbind( stats::rnorm( N, sd=1.5), stats::rnorm(N, sd=.3 ) )

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

# estimate Rasch model
mod1_1 <- TAM::tam.mml(resp=data.sim.rasch, Y=Y)

# standard errors
res1 <- TAM::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::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::tam.mml( resp=data.sim.rasch, xsi.fixed=xsi.fixed )

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

# handling of sample (person) weights
data(data.sim.rasch.pweights)
N <- 1000
pweights <- c(  rep(3,N/2), rep( 1, N/2 ) )
mod1_3 <- TAM::tam.mml( data.sim.rasch.pweights, control=list(conv=.001),
               pweights=pweights )

## 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::tam.mml(Scored)

#Item parameters
mod1$xsi

#Thurstonian thresholds
tthresh <- TAM::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::tam.jml(Scored)
stats::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::tam.mml(Scored2, xsi.fixed=anchor)

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

#CTT statistics
ctt <- TAM::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::tam.mml(dummy, xsi.fixed=anchor)
wle3 <- TAM::tam.wle(mod3)

## End(Not run)
```

DescribeBy — *S3 Method for Descriptive Statistics of Objects*

Description

S3 method for descriptive statistics of objects

Usage

```
DescribeBy(object, ...)
```

Arguments

`object`	An object
`...`	Further arguments to be passed

See Also

psych::describe

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, constraint="cases",...)

## S3 method for class 'designMatrices'
print(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, constraint="cases", Q=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`.
`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*(1), 105-113. doi:10.1007/BF02293856

See Also

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

`doparse` *Parsing a String with* `DO` *Statements*

Description

This function parses a string and expands this string in case of `DO` statements which are shortcuts for writing loops. The statement `DO(n,m,k)` increments an index from `n` to `m` in steps of `k`. The index in the string `model` must be defined as `%`. For a nested loop within a loop, the `DO2` statement can be used using `%1` and `%2` as indices. See Examples for hints on the specification. The loop in `DO2` must not be explicitly crossed, e.g. in applications for specifying covariances or correlations. The formal syntax for
`for (ii in 1:(K-1)){ for (jj in (ii+1):K) { ... } }`
can be written as `DO2(1,K-1,1,%1,K,1 )`. See Example 2.

Usage

```
doparse(model)
```

Arguments

`model`	A string with `DO` or `DO2` statements.

Value

Parsed string in which `DO` statements are expanded.

See Also

This function is also used in `lavaanify.IRT` and `tamaanify`.

Examples

```
#############################################################################
# EXAMPLE 1: doparse example
#############################################################################

# define model
model <- "
 # items I1,...,I10 load on G
 DO(1,10,1)
   G=~ lamg% * I%
 DOEND
 I2 | 0.75*t1
 v10 > 0 ;
 # The first index loops from 1 to 3 and the second index loops from 1 to 2
 DO2(1,3,1,  1,2,1)
   F%2=~ a%2%1 * A%2%1 ;
 DOEND
 # Loop from 1 to 9 with steps of 2
 DO(1,9,2)
   HA1=~ I%
   I% | beta% * t1
 DOEND
 "

# process string
out <- TAM::doparse(model)
cat(out)
  ##    # items I1,...,I10 load on G
  ##       G=~ lamg1 * I1
  ##       G=~ lamg2 * I2
  ##       G=~ lamg3 * I3
  ##       G=~ lamg4 * I4
  ##       G=~ lamg5 * I5
  ##       G=~ lamg6 * I6
  ##       G=~ lamg7 * I7
  ##       G=~ lamg8 * I8
  ##       G=~ lamg9 * I9
  ##       G=~ lamg10 * I10
  ##     I2 | 0.75*t1
  ##     v10 > 0
  ##       F1=~ a11 * A11
  ##       F2=~ a21 * A21
  ##       F1=~ a12 * A12
  ##       F2=~ a22 * A22
  ##       F1=~ a13 * A13
  ##       F2=~ a23 * A23
  ##       HA1=~ I1
  ##       HA1=~ I3
```

```
  ##      HA1=~ I5
  ##      HA1=~ I7
  ##      HA1=~ I9
  ##      I1 | beta1 * t1
  ##      I3 | beta3 * t1
  ##      I5 | beta5 * t1
  ##      I7 | beta7 * t1
  ##      I9 | beta9 * t1

#############################################################################
# EXAMPLE 2: doparse with nested loop example
#############################################################################

# define model
model <- "
 DO(1,4,1)
   G=~ lamg% * I%
 DOEND
 # specify some correlated residuals
 DO2(1,3,1,%1,4,1)
   I%1 ~~ I%2
 DOEND
 "
# process string
out <- TAM::doparse(model)
cat(out)
  ##      G=~ lamg1 * I1
  ##      G=~ lamg2 * I2
  ##      G=~ lamg3 * I3
  ##      G=~ lamg4 * I4
  ##     # specify some correlated residuals
  ##      I1 ~~ I2
  ##      I1 ~~ I3
  ##      I1 ~~ I4
  ##      I2 ~~ I3
  ##      I2 ~~ I4
  ##      I3 ~~ I4
```

IRT.cv *Cross-Validation of a Fitted IRT Model*

Description

This S3 method performs a cross-validation of a fitted item response model.

Usage

```
IRT.cv(object, ...)
```

Arguments

`object`	Object
`...`	Further arguments to be passed

Value

Numeric value: the cross-validated deviance value

`IRT.data.tam` *Extracting Item Response Dataset*

Description

Extracts the used data set for models fitted in **TAM**. See `CDM::IRT.data` for more details.

Usage

```
## S3 method for class 'tam.mml'
IRT.data(object, ...)

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

## S3 method for class 'tamaan'
IRT.data(object, ...)
```

Arguments

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

Value

A dataset with item responses

Examples

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

data(data.sim.rasch)
dat <- data.sim.rasch

# estimate model
mod1 <- TAM::tam.mml(dat)
# extract dataset (and weights and group if available)
```

```
dmod1 <- IRT.data(mod1)
str(dmod1)

## End(Not run)
```

IRT.drawPV *Function for Drawing Plausible Values*

Description

This function draws plausible values of a continuous latent variable based on a fitted object for which the CDM::IRT.posterior method is defined.

Usage

```
IRT.drawPV(object,NPV=5)
```

Arguments

object	Object for which the method CDM::IRT.posterior does exist.
NPV	Number of plausible values to be drawn.

Value

Matrix with plausible values

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: Plausible value imputation for Rasch model in sirt
#############################################################################

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

# fit Rasch model
mod <- rasch.mml2(dat)
# draw 10 plausible values
pv1 <- TAM::IRT.drawPV(mod, NPV=10)

## End(Not run)
```

IRT.expectedCounts *Extracting Expected Counts*

Description

Extracts expected counts for models fitted in **TAM**. See CDM::IRT.expectedCounts for more details.

Usage

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

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

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

## S3 method for class 'tamaan'
IRT.expectedCounts(object, ...)

## S3 method for class 'tam.np'
IRT.expectedCounts(object, ...)
```

Arguments

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

Value

See CDM::IRT.expectedCounts.

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: Dichotomous data data.sim.rasch - extract expected counts
#############################################################################

data(data.sim.rasch)
# 1PL estimation
mod1 <- TAM::tam.mml(resp=data.sim.rasch)
# extract expected counts
IRT.expectedCounts(mod1)

## End(Not run)
```

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

Description

Extracts factor scores for models fitted in **TAM**. See `CDM::IRT.factor.scores` 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", ...)

## S3 method for class 'tamaan'
IRT.factor.scores(object, type="EAP", ...)
```

Arguments

`object`	Object of class `tam`, `tam.mml`, `tam.mml.3pl` or `tamaan`.
`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 `CDM::IRT.factor.scores`.

Examples

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

data(data.sim.rasch)
# 1PL estimation
mod1 <- TAM::tam.mml(resp=data.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.frequencies.tam   *Observed and Expected Frequencies for Univariate and Bivariate Distributions*

Description

Computes observed and expected frequencies for univariate and bivariate distributions for models fitted in **TAM**. See `CDM::IRT.frequencies` for more details.

Usage

```
## S3 method for class 'tam.mml'
IRT.frequencies(object, ...)

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

## S3 method for class 'tamaan'
IRT.frequencies(object, ...)
```

Arguments

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

Value

See `CDM::IRT.frequencies`.

See Also

`CDM::IRT.frequencies`

Examples

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

data(data.sim.rasch)
dat <- data.sim.rasch

# estimate model
```

```
mod1 <- TAM::tam.mml(dat)
# compute observed and expected frequencies
fmod1 <- IRT.frequencies(mod1)
str(fmod1)

## End(Not run)
```

IRT.informationCurves *Item and Test Information Curve*

Description

An S3 method which computes item and test information curves, see Muraki (1993).

Usage

```
IRT.informationCurves(object, ...)

## S3 method for class 'tam.mml'
IRT.informationCurves( object, h=.0001, iIndex=NULL,
          theta=NULL, ... )

## S3 method for class 'tam.mml.2pl'
IRT.informationCurves( object, h=.0001, iIndex=NULL,
          theta=NULL, ... )

## S3 method for class 'tam.mml.mfr'
IRT.informationCurves( object, h=.0001, iIndex=NULL,
          theta=NULL, ... )

## S3 method for class 'tam.mml.3pl'
IRT.informationCurves( object, h=.0001, iIndex=NULL,
          theta=NULL, ... )

## S3 method for class 'IRT.informationCurves'
plot(x, curve_type="test", ...)
```

Arguments

object	Object of class tam.mml, tam.mml.2pl, tam.mml.mfr or tam.mml.3pl.
...	Further arguments to be passed
h	Numerical differentiation parameter
iIndex	Indices of items for which test information should be computed. The default is to use all items.
theta	Optional vector of θ for which information curves should be computed.
curve_type	Type of information to be plotted. It can be "test" for the test information curve and "se" for the standard error curve.
x	Object of class tam.mml, tam.mml.2pl, tam.mml.mfr or tam.mml.3pl.

Value

List with following entries

se_curve Standard error curves

test_info_curve
Test information curve

info_curves_item
Item information curves

info_curves_categories
Item-category information curves

theta Used θ grid

References

Muraki, E. (1993). Information functions of the generalized partial credit model. *Applied Psychological Measurement, 17*(4), 351-363. doi:10.1177/014662169301700403

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: Dichotomous data | data.read
#############################################################################

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

# fit 2PL model
mod1 <- TAM::tam.mml.2pl( dat )
summary(mod1)

# compute information curves at grid seq(-5,5,length=100)
imod1 <- TAM::IRT.informationCurves( mod1, theta=seq(-5,5,len=100)  )
str(imod1)
# plot test information
plot( imod1 )
# plot standard error curve
plot( imod1, curve_type="se", xlim=c(-3,2) )
# cutomized plot
plot( imod1, curve_type="se", xlim=c(-3,2), ylim=c(0,2), lwd=2, lty=3)

#############################################################################
# EXAMPLE 2: Mixed dichotomous and polytomous data
#############################################################################

data(data.timssAusTwn.scored, package="TAM")
dat <- data.timssAusTwn.scored
# select item response data
items <- grep( "M0", colnames(dat), value=TRUE )
resp <- dat[, items ]
```

```
#*** Model 1: Partial credit model
mod1 <- TAM::tam.mml( resp )
summary(mod1)
# information curves
imod1 <- TAM::IRT.informationCurves( mod1, theta=seq(-3,3,len=20)  )

#*** Model 2: Generalized partial credit model
mod2 <- TAM::tam.mml.2pl( resp, irtmodel="GPCM")
summary(mod2)
imod2 <- TAM::IRT.informationCurves( mod2 )

#*** Model 3: Mixed 3PL and generalized partial credit model
psych::describe(resp)
maxK <- apply( resp, 2, max, na.rm=TRUE )
I <- ncol(resp)
# specify guessing parameters, including a prior distribution
est.guess <- 1:I
est.guess[ maxK > 1 ] <- 0
guess <- .2*(est.guess >0)
guess.prior <- matrix( 0, nrow=I, ncol=2 )
guess.prior[ est.guess  > 0, 1] <- 5
guess.prior[ est.guess  > 0, 2] <- 17

# fit model
mod3 <- TAM::tam.mml.3pl( resp, gammaslope.des="2PL", est.guess=est.guess, guess=guess,
           guess.prior=guess.prior,
           control=list( maxiter=100, Msteps=10, fac.oldxsi=0.1,
                        nodes=seq(-8,8,len=41) ),  est.variance=FALSE )
summary(mod3)

# information curves
imod3 <- TAM::IRT.informationCurves( mod3 )
imod3

#*** estimate model in mirt package
library(mirt)
itemtype <- rep("gpcm", I)
itemtype[ maxK==1] <- "3PL"
mod3b <- mirt::mirt(resp, 1, itemtype=itemtype, verbose=TRUE )
print(mod3b)

## End(Not run)
```

`IRT.irfprob` *Extracting Item Response Functions*

Description

Extracts item response functions for models fitted in **TAM**. See `CDM::IRT.irfprob` 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, ...)

## S3 method for class 'tamaan'
IRT.irfprob(object, ...)

## S3 method for class 'tam.np'
IRT.irfprob(object, ...)
```

Arguments

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

Value

See CDM::IRT.irfprob.

Examples

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

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

IRT.itemfit.tam   *RMSD Item Fit Statistics for* **TAM** *Objects*

Description

Computes the RMSD item fit statistic (formerly labeled as RMSEA; Yamamoto, Khorramdel, & von Davier, 2013) for fitted objects in the **TAM** package, see CDM::IRT.itemfit and CDM::IRT.RMSD.

Usage

```
## S3 method for class 'tam.mml'
IRT.itemfit(object, method="RMSD", ...)

## S3 method for class 'tam.mml.2pl'
IRT.itemfit(object, method="RMSD", ...)

## S3 method for class 'tam.mml.mfr'
IRT.itemfit(object, method="RMSD", ...)

## S3 method for class 'tam.mml.3pl'
IRT.itemfit(object, method="RMSD", ...)
```

Arguments

`object`	Object of class `tam.mml`, `tam.mml.2pl`, `tam.mml.mfr` or `tam.mml.3pl`.
`method`	Requested method for item fit calculation. Currently, only the RMSD fit statistic (formerly labeled as the RMSEA statistic, see `CDM::IRT.RMSD`) can be used.
`...`	Further arguments to be passed.

References

Yamamoto, K., Khorramdel, L., & von Davier, M. (2013). Scaling PIAAC cognitive data. In OECD (Eds.). *Technical Report of the Survey of Adults Skills (PIAAC)* (Ch. 17). Paris: OECD.

See Also

`CDM::IRT.itemfit`, `CDM::IRT.RMSD`

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: RMSD item fit statistic data.read
#############################################################################

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

#*** fit 1PL model
mod1 <- TAM::tam.mml( dat )
summary(mod1)

#*** fit 2PL model
mod2 <- TAM::tam.mml.2pl( dat )
summary(mod2)

#*** assess RMSEA item fit
fmod1 <- IRT.itemfit(mod1)
```

```
fmod2 <- IRT.itemfit(mod2)
# summary of fit statistics
summary( fmod1 )
summary( fmod2 )

#############################################################################
# EXAMPLE 2: Simulated 2PL data and fit of 1PL model
#############################################################################

set.seed(987)
N <- 1000    # 1000 persons
I <- 10      # 10 items
# define item difficulties and item slopes
b <- seq(-2,2,len=I)
a <- rep(1,I)
a[c(3,8)] <- c( 1.7, .4 )
# simulate 2PL data
dat <- sirt::sim.raschtype( theta=rnorm(N), b=b, fixed.a=a)

# fit 1PL model
mod <- TAM::tam.mml( dat )

# RMSEA item fit
fmod <- IRT.itemfit(mod)
round( fmod, 3 )

## End(Not run)
```

IRT.likelihood *Extracting Individual Likelihood and Individual Posterior*

Description

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

Usage

```
## S3 method for class 'tam'
IRT.likelihood(object, ...)
## S3 method for class 'tam'
IRT.posterior(object, ...)

## S3 method for class 'tam.mml'
IRT.likelihood(object, ...)
## S3 method for class 'tam.mml'
IRT.posterior(object, ...)

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

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

## S3 method for class 'tamaan'
IRT.likelihood(object, ...)
## S3 method for class 'tamaan'
IRT.posterior(object, ...)

## S3 method for class 'tam.latreg'
IRT.likelihood(object, ...)
## S3 method for class 'tam.latreg'
IRT.posterior(object, ...)

## S3 method for class 'tam.np'
IRT.likelihood(object, ...)
## S3 method for class 'tam.np'
IRT.posterior(object, ...)
```

Arguments

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

Value

See `CDM::IRT.likelihood`.

Examples

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

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

`IRT.linearCFA` *Linear Approximation of a Confirmatory Factor Analysis*

Description

This function approximates a fitted item response model by a linear confirmatory factor analysis. I.e., given item response functions, the expectation $E(X_i|\theta_1, \ldots, \theta_D)$ is linearly approximated by $a_{i1}\theta_1 + \ldots + a_{iD}\theta_D$. See Vermunt and Magidson (2005) for details.

Usage

```
IRT.linearCFA( object, group=1)

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

Arguments

`object`	Fitted item response model for which the `IRT.expectedCounts` method is defined.
`group`	Group identifier which defines the selected group.
`...`	Further arguments to be passed.

Value

A list with following entries

`loadings`	Data frame with factor loadings. `Mlat` and `SDlat` denote the model-implied item mean and standard deviation. The values `ResidVar` and `h2` denote residual variances and item communality.
`stand.loadings`	Data frame with standardized factor loadings.
`M.trait`	Mean of factors
`SD.trait`	Standard deviations of factors

References

Vermunt, J. K., & Magidson, J. (2005). Factor Analysis with categorical indicators: A comparison between traditional and latent class approaches. In A. Van der Ark, M.A. Croon & K. Sijtsma (Eds.), *New Developments in Categorical Data Analysis for the Social and Behavioral Sciences* (pp. 41-62). Mahwah: Erlbaum

See Also

See `tam.fa` for confirmatory factor analysis in **TAM**.

Examples

```
## Not run:
library(lavaan)

#############################################################################
# EXAMPLE 1: Two-dimensional confirmatory factor analysis data.Students
#############################################################################
```

```
data(data.Students, package="CDM")
# select variables
vars <- scan(nlines=1, what="character")
    sc1 sc2 sc3 sc4 mj1 mj2 mj3 mj4
dat <- data.Students[, vars]

# define Q-matrix
Q <- matrix( 0, nrow=8, ncol=2 )
Q[1:4,1] <- Q[5:8,2] <- 1

#*** Model 1: Two-dimensional 2PL model
mod1 <- TAM::tam.mml.2pl( dat, Q=Q, control=list( nodes=seq(-4,4,len=12) ) )
summary(mod1)

# linear approximation CFA
cfa1 <- TAM::IRT.linearCFA(mod1)
summary(cfa1)

# linear CFA in lavaan package
lavmodel <- "
    sc=~ sc1+sc2+sc3+sc4
    mj=~ mj1+mj2+mj3+mj4
    sc1 ~ 1
    sc ~~ mj
    "
mod1b <- lavaan::sem( lavmodel, data=dat, missing="fiml", std.lv=TRUE)
summary(mod1b, standardized=TRUE, fit.measures=TRUE )

#############################################################################
# EXAMPLE 2: Unidimensional confirmatory factor analysis data.Students
#############################################################################

data(data.Students, package="CDM")
# select variables
vars <- scan(nlines=1, what="character")
    sc1 sc2 sc3 sc4
dat <- data.Students[, vars]

#*** Model 1: 2PL model
mod1 <- TAM::tam.mml.2pl( dat )
summary(mod1)

# linear approximation CFA
cfa1 <- TAM::IRT.linearCFA(mod1)
summary(cfa1)

# linear CFA
lavmodel <- "
    sc=~ sc1+sc2+sc3+sc4
    "
mod1b <- lavaan::sem( lavmodel, data=dat, missing="fiml", std.lv=TRUE)
summary(mod1b, standardized=TRUE, fit.measures=TRUE )
```

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

IRT.residuals *Residuals in an IRT Model*

Description

Defines an S3 method for the computation of observed residual values. The computation of residuals is based on weighted likelihood estimates as person parameters, see `tam.wle`. `IRT.residuals` can only be applied for unidimensional IRT models. The methods `IRT.residuals` and `residuals` are equivalent.

Usage

```
IRT.residuals(object, ...)

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

## S3 method for class 'tam.mml.2pl'
IRT.residuals(object, ...)
## S3 method for class 'tam.mml.2pl'
residuals(object, ...)

## S3 method for class 'tam.mml.mfr'
IRT.residuals(object, ...)
## S3 method for class 'tam.mml.mfr'
residuals(object, ...)

## S3 method for class 'tam.jml'
IRT.residuals(object, ...)
## S3 method for class 'tam.jml'
residuals(object, ...)
```

Arguments

`object`	Object of class `tam.mml`, `tam.mml.2pl` or `tam.mml.mfr`.
`...`	Further arguments to be passed

Value

List with following entries

`residuals`	Residuals

`stand_residuals`
: Standardized residuals

`X_exp`
: Expected value of the item response X_{pi}

`X_var`
: Variance of the item response X_{pi}

`theta`
: Used person parameter estimate

`probs`
: Expected item response probabilities

Note

Residuals can be used to inspect local dependencies in the item response data, for example using principle component analysis or factor analysis (see Example 1).

See Also

See also the `eRm::residuals` (**eRm**) or `residuals` (**mirt**) functions.

See also `predict.tam.mml`.

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: Residuals data.read
#############################################################################

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

# for Rasch model
mod <- TAM::tam.mml( dat )
# extract residuals
res <- TAM::IRT.residuals( mod )
str(res)

## End(Not run)
```

`IRT.simulate` *Simulating Item Response Models*

Description

Defines an S3 method for simulation of item response models.

Usage

```
IRT.simulate(object, ...)

## S3 method for class 'tam.mml'
IRT.simulate(object, iIndex=NULL, theta=NULL, nobs=NULL, ...)

## S3 method for class 'tam.mml.2pl'
IRT.simulate(object, iIndex=NULL, theta=NULL, nobs=NULL, ...)

## S3 method for class 'tam.mml.mfr'
IRT.simulate(object, iIndex=NULL, theta=NULL, nobs=NULL, ...)

## S3 method for class 'tam.mml.3pl'
IRT.simulate(object, iIndex=NULL, theta=NULL, nobs=NULL, ...)
```

Arguments

`object`	An object of class `tam.mml`, `tam.mml.2pl`, `tam.mml.mfr` or `tam.mml.3pl`.
`iIndex`	Optional vector of item indices
`theta`	Optional matrix of `theta` values
`nobs`	Optional numeric containing the number of observations to be simulated.
`...`	Further objects to be passed

Value

Data frame with simulated item responses

Examples

```
#############################################################################
# EXAMPLE 1: Simulating Rasch model
#############################################################################

data(data.sim.rasch)

#** (1) estimate model
mod1 <- TAM::tam.mml(resp=data.sim.rasch )

#** (2) simulate data
sim.dat <- TAM::IRT.simulate(mod1)

## Not run:
#** (3) use a subset of items with the argument iIndex
set.seed(976)
iIndex <- sort(sample(ncol(data.sim.rasch), 15))  # randomly select 15 items
sim.dat <- TAM::IRT.simulate(mod1, iIndex=iIndex)
mod.sim.dat <- TAM::tam.mml(sim.dat)

#** (4) specify number of persons in addition
```

```
sim.dat <- TAM::IRT.simulate(mod1, nobs=1500, iIndex=iIndex)

# Rasch - constraint="items" ----
mod1 <- TAM::tam.mml(resp=data.sim.rasch,  constraint="items",
                control=list( xsi.start0=1, fac.oldxsi=.5) )

# provide abilities
theta0 <- matrix( rnorm(1500, mean=0.5, sd=sqrt(mod1$variance)), ncol=1 )
# simulate data
data <- TAM::IRT.simulate(mod1, theta=theta0 )
# estimate model based on simulated data
xsi.fixed <- cbind(1:nrow(mod1$item), mod1$item$xsi.item)
mod2 <- TAM::tam.mml(data, xsi.fixed=xsi.fixed )
summary(mod2)

#############################################################################
# EXAMPLE 2: Simulating 2PL model
#############################################################################

data(data.sim.rasch)
# estimate 2PL
mod2 <- TAM::tam.mml.2pl(resp=data.sim.rasch, irtmodel="2PL")
# simulate 2PL
sim.dat <- TAM::IRT.simulate(mod2)
mod.sim.dat <- TAM::tam.mml.2pl(resp=sim.dat, irtmodel="2PL")

#############################################################################
# EXAMPLE 3: Simulate multiple group model
#############################################################################

# Multi-Group ----
set.seed(6778)
N <- 3000
theta <- c( stats::rnorm(N/2,mean=0,sd=1.5), stats::rnorm(N/2,mean=.5,sd=1)  )
I <- 20
p1 <- stats::plogis( outer( theta, seq( -2, 2, len=I ), "-" ) )
resp <- 1 * ( p1 > matrix( stats::runif( N*I ), nrow=N, ncol=I ) )
colnames(resp) <- paste("I", 1:I, sep="")
group <- rep(1:2, each=N/2 )
mod3 <- TAM::tam.mml(resp, group=group)

# simulate data
sim.dat.g1 <- TAM::IRT.simulate(mod3,
                   theta=matrix( stats::rnorm(N/2, mean=0, sd=1.5), ncol=1) )
sim.dat.g2 <- TAM::IRT.simulate(mod3,
                   theta=matrix( stats::rnorm(N/2, mean=.5, sd=1), ncol=1) )
sim.dat <- rbind( sim.dat.g1, sim.dat.g2)
# estimate model
mod3s <- TAM::tam.mml( sim.dat, group=group)

#############################################################################
# EXAMPLE 4: Multidimensional model and latent regression
#############################################################################
```

```
set.seed(6778)
N <- 2000
Y <- cbind( stats::rnorm(N), stats::rnorm(N))
theta <- mvtnorm::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 <- stats::plogis(outer(theta[, 1], seq(-2, 2, len=I), "-"))
resp1 <- 1 * (p1 > matrix(stats::runif(N * I), nrow=N, ncol=I))
p1 <- stats::plogis(outer(theta[, 2], seq(-2, 2, len=I ), "-" ))
resp2 <- 1 * (p1 > matrix(stats::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
Q

# (3) estimate models
mod4 <- TAM::tam.mml(resp=resp, Q=Q, Y=Y, control=list(  maxiter=15))

# simulate new item responses
theta <- mvtnorm::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

sim.dat <- TAM::IRT.simulate(mod4, theta=theta)
mod.sim.dat <- TAM::tam.mml(resp=sim.dat, Q=Q, Y=Y, control=list( maxiter=15))

## End(Not run)
```

`IRT.threshold` *Thurstonian Thresholds and Wright Map for Item Response Models*

Description

The function `IRT.threshold` computes Thurstonian thresholds for item response models. It is only based on fitted models for which the `IRT.irfprob` does exist.

The function `IRT.WrightMap` creates a Wright map and works as a wrapper to the `WrightMap::wrightMap` function in the **WrightMap** package. Wright maps operate on objects of class `IRT.threshold`.

Usage

```
IRT.threshold(object, prob.lvl=.5, type="category")

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

```
IRT.WrightMap(object, ...)

## S3 method for class 'IRT.threshold'
IRT.WrightMap(object, label.items=NULL, ...)
```

Arguments

`object`	Object of fitted models for which `IRT.irfprob` exists.
`prob.lvl`	Requested probability level of thresholds.
`type`	Type of thresholds to be calculated. The default is category-wise calculation. If only one threshold per item should be calculated, then choose `type="item"`. If an item possesses a maximum score of K_i, then a threshold is defined as a value which produces an expected value of $K_i/2$ (see Ali, Chang & Anderson, 2015).
`x`	Object of class `IRT.threshold`
`label.items`	Vector of item labels
`...`	Further arguments to be passed.

Value

Function `IRT.threshold`:
Matrix with Thurstonian thresholds

Function `IRT.WrightMap`:
A Wright map generated by the **WrightMap** package.

Author(s)

The `IRT.WrightMap` function is based on the `WrightMap::wrightMap` function in the **WrightMap** package.

References

Ali, U. S., Chang, H.-H., & Anderson, C. J. (2015). *Location indices for ordinal polytomous items based on item response theory* (Research Report No. RR-15-20). Princeton, NJ: Educational Testing Service. doi:10.1002/ets2.12065

See Also

See the `WrightMap::wrightMap` function in the **WrightMap** package.

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: Fitted unidimensional model with gdm
#############################################################################

data(data.Students)
```

```
dat <- data.Students

# select part of the dataset
resp <- dat[, paste0("sc",1:4) ]
resp[ paste(resp[,1])==3,1] <-  2
psych::describe(resp)

# Model 1: Partial credit model in gdm
theta.k <- seq( -5, 5, len=21 )   # discretized ability
mod1 <- CDM::gdm( dat=resp, irtmodel="1PL", theta.k=theta.k, skillspace="normal",
              centered.latent=TRUE)

# compute thresholds
thresh1 <- TAM::IRT.threshold(mod1)
print(thresh1)
IRT.WrightMap(thresh1)

#############################################################################
# EXAMPLE 2: Fitted mutidimensional model with gdm
#############################################################################

data( data.fraction2 )
dat <- data.fraction2$data
Qmatrix <- data.fraction2$q.matrix3

# Model 1: 3-dimensional Rasch Model (normal distribution)
theta.k <- seq( -4, 4, len=11 )   # discretized ability
mod1 <- CDM::gdm( dat, irtmodel="1PL", theta.k=theta.k, Qmatrix=Qmatrix,
              centered.latent=TRUE, maxiter=10 )
summary(mod1)

# compute thresholds
thresh1 <- TAM::IRT.threshold(mod1)
print(thresh1)

#############################################################################
# EXAMPLE 3: Item-wise thresholds
#############################################################################

data(data.timssAusTwn.scored)
dat <- data.timssAusTwn.scored
dat <- dat[, grep("M03", colnames(dat) ) ]
summary(dat)

# fit partial credit model
mod <- TAM::tam.mml( dat )
# compute thresholds with tam.threshold function
t1mod <- TAM::tam.threshold( mod )
t1mod
# compute thresholds with IRT.threshold function
t2mod <- TAM::IRT.threshold( mod )
t2mod
# compute item-wise thresholds
```

```
t3mod <- TAM::IRT.threshold( mod, type="item")
t3mod

## End(Not run)
```

IRT.truescore *Converts a θ Score into a True Score $\tau(\theta)$*

Description

Converts a θ score into an unweighted true score $\tau(\theta) = \sum_i \sum_h hP_i(\theta)$. In addition, a weighted true score $\tau(\theta) = \sum_i \sum_h q_{ih} P_i(\theta)$ can also be computed by specifying item-category weights q_{ih} in the matrix Q.

Usage

```
IRT.truescore(object, iIndex=NULL, theta=NULL, Q=NULL)
```

Arguments

object	Object for which the CDM::IRT.irfprob S3 method is defined
iIndex	Optional vector with item indices
theta	Optional vector with θ values
Q	Optional weighting matrix

Value

Data frame containing θ values and corresponding true scores $\tau(\theta)$.

See Also

See also sirt::truescore.irt for a conversion function for generalized partial credit models.

Examples

```
#############################################################################
# EXAMPLE 1: True score conversion for a test with polytomous items
#############################################################################

data(data.Students, package="CDM")
dat <- data.Students[, paste0("mj",1:4) ]

# fit partial credit model
mod1 <- TAM::tam.mml( dat,control=list(maxiter=20) )
summary(mod1)

# true score conversion
tmod1 <- TAM::IRT.truescore( mod1 )
```

```
round( tmod1, 4 )
# true score conversion with user-defined theta grid
tmod1b <- TAM::IRT.truescore( mod1, theta=seq( -8,8, len=33 ) )
# plot results
plot( tmod1$theta, tmod1$truescore, type="l",
            xlab=expression(theta), ylab=expression(tau( theta ) ) )
points( tmod1b$theta, tmod1b$truescore, pch=16, col="brown" )

## Not run:
#############################################################################
# EXAMPLE 2: True scores with different category weightings
#############################################################################

data(data.timssAusTwn.scored)
dat <- data.timssAusTwn.scored
# extract item response data
dat <- dat[, grep("M03", colnames(dat) ) ]
# select items with do have maximum score of 2 (polytomous items)
ind <- which( apply( dat,  2, max, na.rm=TRUE )==2 )
I <- ncol(dat)
# define Q-matrix with scoring variant
Q <- matrix( 1, nrow=I, ncol=1 )
Q[ ind, 1 ] <- .5    # score of 0.5 for polyomous items

# estimate model
mod1 <- TAM::tam.mml( dat, Q=Q, irtmodel="PCM2", control=list( nodes=seq(-10,10,len=61) ) )
summary(mod1)

# true score with scoring (0,1,2) which is the default of the function
tmod1 <- TAM::IRT.truescore(mod1)
# true score with user specified weighting matrix
Q <- mod1$B[,,1]
tmod2 <- TAM::IRT.truescore(mod1, Q=Q)

## End(Not run)
```

`IRT.WrightMap` *Wright Map for Item Response Models by Using the* **WrightMap** *Package*

Description

This function creates a Wright map and works as a wrapper to the `wrightMap` function in the **WrightMap** package. The arguments `thetas` and `thresholds` are automatically generated from fitted objects in **TAM**.

Usage

```
## S3 method for class 'tam.mml'
IRT.WrightMap(object, prob.lvl=.5, type="PV", ...)
```

```
## S3 method for class 'tamaan'
IRT.WrightMap(object, prob.lvl=.5, type="PV", ...)
```

Arguments

- `object` Object of class `tam.mml` or class `tamaan`
- `prob.lvl` Requested probability level of thresholds.
- `type` Type of person parameter estimate. `"PV"` (plausible values), `"WLE"` (weighted likelihood estimates) and `"Pop"` (population trait distribution) can be specified.
- `...` Further arguments to be passed in the `wrightMap` (**WrightMap**) function. See Examples.

Details

A Wright map is only created for models with an assumed normal distribution. Hence, not for all models of the `tamaan` functions Wright maps are created.

Value

A Wright map generated by the **WrightMap** package.

Author(s)

The `IRT.WrightMap` function is based on the `WrightMap::wrightMap` function in the **WrightMap** package.

See Also

See the `WrightMap::wrightMap` function in the **WrightMap** package.

Examples

```
## Not run:
library(WrightMap)

#############################################################################
# EXAMPLE 1: Unidimensional models dichotomous data
#############################################################################

data(data.sim.rasch)
str(data.sim.rasch)
dat <- data.sim.rasch

# fit Rasch model
mod1 <- TAM::tam.mml(resp=dat)
# Wright map
IRT.WrightMap( mod1 )
# some customized plots
IRT.WrightMap( mod1, show.thr.lab=FALSE, label.items=c(1:40), label.items.rows=3)
```

```
IRT.WrightMap( mod1,  show.thr.sym=FALSE, thr.lab.text=paste0("I",1:ncol(dat)),
     label.items="", label.items.ticks=FALSE)

#--- direct specification with wrightMap function
theta <- TAM::tam.wle(mod1)$theta
thr <- TAM::tam.threshold(mod1)

# default wrightMap plots
WrightMap::wrightMap( theta, thr, label.items.srt=90)
WrightMap::wrightMap( theta, t(thr), label.items=c("items") )

# stack all items below each other
thr.lab.text <- matrix( "", 1, ncol(dat) )
thr.lab.text[1,] <- colnames(dat)
WrightMap::wrightMap( theta, t(thr), label.items=c("items"),
       thr.lab.text=thr.lab.text, show.thr.sym=FALSE )

#############################################################################
# EXAMPLE 2: Unidimensional model polytomous data
#############################################################################

data( data.Students, package="CDM")
dat <- data.Students

# fit generalized partial credit model using the tamaan function
tammodel <- "
LAVAAN MODEL:
  SC=~ sc1__sc4
  SC ~~ 1*SC
    "
mod1 <- TAM::tamaan( tammodel, dat )
# create item level colors
library(RColorBrewer)
ncat <- 3               # number of category parameters
I <- ncol(mod1$resp)    # number of items
itemlevelcolors <- matrix(rep( RColorBrewer::brewer.pal(ncat, "Set1"), I),
        byrow=TRUE, ncol=ncat)
# Wright map
IRT.WrightMap(mod1, prob.lvl=.625, thr.sym.col.fg=itemlevelcolors,
     thr.sym.col.bg=itemlevelcolors, label.items=colnames( mod1$resp) )

#############################################################################
# EXAMPLE 3: Multidimensional item response model
#############################################################################

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

# fit three-dimensional Rasch model
Q <- matrix( 0, nrow=12, ncol=3 )
Q[1:4,1] <- Q[5:8,2] <- Q[9:12,3] <- 1
mod1 <- TAM::tam.mml( dat, Q=Q, control=list(maxiter=20, snodes=1000)  )
```

```
summary(mod1)
# define matrix with colors for thresholds
c1 <- matrix( c( rep(1,4), rep(2,4), rep(4,4)), ncol=1 )
# create Wright map using WLE
IRT.WrightMap( mod1, prob.lvl=.65, type="WLE", thr.lab.col=c1, thr.sym.col.fg=c1,
        thr.sym.col.bg=c1, label.items=colnames(dat) )
# Wright map using PV (the default)
IRT.WrightMap( mod1, prob.lvl=.65, type="PV" )
# Wright map using population distribution
IRT.WrightMap( mod1, prob.lvl=.65, type="Pop" )

#############################################################################
# EXAMPLE 4: Wright map for a multi-faceted Rasch model
#############################################################################

# This example is copied from
# http://wrightmap.org/post/107431190622/wrightmap-multifaceted-models

library(WrightMap)
data(data.ex10)
dat <- data.ex10

#--- fit multi-faceted Rasch model
facets <- dat[, "rater", drop=FALSE]  # define facet (rater)
pid <- dat$pid  # define person identifier (a person occurs multiple times)
resp <- dat[, -c(1:2)]  # item response data
formulaA <- ~item * rater  # formula
mod <- TAM::tam.mml.mfr(resp=resp, facets=facets, formulaA=formulaA, pid=dat$pid)

# person parameters
persons.mod <- TAM::tam.wle(mod)
theta <- persons.mod$theta
# thresholds
thr <- TAM::tam.threshold(mod)
item.labs <- c("I0001", "I0002", "I0003", "I0004", "I0005")
rater.labs <- c("rater1", "rater2", "rater3")

#--- Plot 1: Item specific
thr1 <- matrix(thr, nrow=5, byrow=TRUE)
WrightMap::wrightMap(theta, thr1, label.items=item.labs,
   thr.lab.text=rep(rater.labs, each=5))

#--- Plot 2: Rater specific
thr2 <- matrix(thr, nrow=3)
WrightMap::wrightMap(theta, thr2, label.items=rater.labs,
   thr.lab.text=rep(item.labs,  each=3), axis.items="Raters")

#--- Plot 3a: item, rater and item*rater parameters
pars <- mod$xsi.facets$xsi
facet <- mod$xsi.facets$facet

item.par <- pars[facet=="item"]
rater.par <- pars[facet=="rater"]
```

```
item_rat <- pars[facet=="item:rater"]

len <- length(item_rat)
item.long <- c(item.par, rep(NA, len - length(item.par)))
rater.long <- c(rater.par, rep(NA, len - length(rater.par)))
ir.labs <- mod$xsi.facets$parameter[facet=="item:rater"]

WrightMap::wrightMap(theta, rbind(item.long, rater.long, item_rat),
    label.items=c("Items",  "Raters", "Item*Raters"),
    thr.lab.text=rbind(item.labs, rater.labs, ir.labs), axis.items="")

#--- Plot 3b: item, rater and item*rater (separated by raters) parameters

# parameters item*rater
ir_rater <- matrix(item_rat, nrow=3, byrow=TRUE)
# define matrix of thresholds
thr <- rbind(item.par, c(rater.par, NA, NA), ir_rater)
# matrix with threshold labels
thr.lab.text <- rbind(item.labs, rater.labs,
           matrix(item.labs, nrow=3, ncol=5, byrow=TRUE))

WrightMap::wrightMap(theta, thresholds=thr,
      label.items=c("Items", "Raters", "Item*Raters (R1)",
                         "Item*Raters (R2)", "Item*Raters (R3)"),
      axis.items="", thr.lab.text=thr.lab.text )

#--- Plot 3c: item, rater and item*rater (separated by items) parameters

# thresholds
ir_item <- matrix(item_rat, nrow=5)
thr <- rbind(item.par, c(rater.par, NA, NA), cbind(ir_item, NA, NA))
# labels
label.items <- c("Items", "Raters", "Item*Raters\n (I1)", "Item*Raters \n(I2)",
     "Item*Raters \n(I3)", "Item*Raters \n (I4)", "Item*Raters \n(I5)")
thr.lab.text <- rbind(item.labs,
          matrix(c(rater.labs, NA, NA), nrow=6, ncol=5, byrow=TRUE))

WrightMap::wrightMap(theta, thr,  label.items=label.items,
      axis.items="", thr.lab.text=thr.lab.text  )

## End(Not run)
```

`IRTLikelihood.cfa` *Individual Likelihood for Confirmatory Factor Analysis*

Description

This function computes the individual likelihood evaluated at a `theta` grid for confirmatory factor analysis under the normality assumption of residuals. Either the item parameters (item loadings `L`, item intercepts `nu` and residual covariances `psi`) or a fitted `cfa` object from the **lavaan** package can be provided. The individual likelihood can be used for drawing plausible values.

Usage

```
IRTLikelihood.cfa(data, cfaobj=NULL, theta=NULL, L=NULL, nu=NULL,
    psi=NULL, snodes=NULL, snodes.adj=2, version=1)
```

Arguments

`data`	Dataset with item responses
`cfaobj`	Fitted `lavaan::cfa` (**lavaan**) object
`theta`	Optional matrix containing the `theta` values used for evaluating the individual likelihood
`L`	Matrix of item loadings (if `cfaobj` is not provided)
`nu`	Vector of item intercepts (if `cfaobj` is not provided)
`psi`	Matrix with residual covariances (if `cfaobj` is not provided)
`snodes`	Number of `theta` values used for the approximation of the distribution of latent variables.
`snodes.adj`	Adjustment factor for quasi monte carlo nodes for more than two latent variables.
`version`	Function version. `version=1` is based on a **Rcpp** implementation while `version=0` is a pure R implementation.

Value

Individual likelihood evaluated at `theta`

See Also

`CDM::IRT.likelihood`

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: Two-dimensional CFA data.Students
#############################################################################

library(lavaan)
library(CDM)

data(data.Students, package="CDM")
dat <- data.Students

dat2 <- dat[, c(paste0("mj",1:4), paste0("sc",1:4)) ]
# lavaan model with DO operator
lavmodel <- "
DO(1,4,1)
   mj=~ mj%
   sc=~ sc%
DOEND
```

```
       mj ~~ sc
       mj ~~ 1*mj
       sc ~~ 1*sc
         "
    lavmodel <- TAM::lavaanify.IRT( lavmodel, data=dat2 )$lavaan.syntax
    cat(lavmodel)

    mod4 <- lavaan::cfa( lavmodel, data=dat2, std.lv=TRUE )
    summary(mod4, standardized=TRUE, rsquare=TRUE )
    # extract item parameters
    res4 <- TAM::cfa.extract.itempars( mod4 )
    # create theta grid
    theta0 <- seq( -6, 6, len=15)
    theta <- expand.grid( theta0, theta0 )
    L <- res4$L
    nu <- res4$nu
    psi <- res4$psi
    data <- dat2
    # evaluate likelihood using item parameters
    like2 <- TAM::IRTLikelihood.cfa( data=dat2, theta=theta, L=L, nu=nu, psi=psi )
    # The likelihood can also be obtained by direct evaluation
    # of the fitted cfa object "mod4"
    like4 <- TAM::IRTLikelihood.cfa( data=dat2, cfaobj=mod4 )
    attr( like4, "theta")
    # the theta grid is automatically created if theta is not
    # supplied as an argument

    ## End(Not run)
```

`IRTLikelihood.ctt` *Computes Individual Likelihood from Classical Test Theory Estimates*

Description

Computes individual likelihood from classical test theory estimates under a unidimensional normal distribution of measurement errors.

Usage

```
IRTLikelihood.ctt(y, errvar, theta=NULL)
```

Arguments

`y`	Vector of observed scores
`errvar`	Vector of error variances
`theta`	Optional vector for θ grid.

Value

Object of class `IRT.likelihood`

Examples

```
#############################################################################
# EXAMPLE 1: Individual likelihood and latent regression in CTT
#############################################################################

set.seed(75)

#--- simulate data
N <- 2000
x1 <- stats::rnorm(N)
x2 <- .7 * x1 + stats::runif(N)
# simulate true score
theta <- 1.2 + .6*x1 + .3 *x2 + stats::rnorm(N, sd=sqrt(.50) )
var(theta)
# simulate measurement error variances
errvar <- stats::runif( N, min=.6, max=.9 )
# simulate observed scores
y <- theta + stats::rnorm( N, sd=sqrt( errvar) )

#--- create likelihood object
like1 <- TAM::IRTLikelihood.ctt( y=y, errvar=errvar, theta=NULL )

#--- estimate latent regression
X <- data.frame(x1,x2)
mod1 <- TAM::tam.latreg( like=like1, Y=X )

## Not run:
#--- draw plausible values
pv1 <- TAM::tam.pv( mod1, normal.approx=TRUE )

#--- create datalist
datlist1 <- TAM::tampv2datalist( pv1, pvnames="thetaPV", Y=X )

#--- statistical inference on plausible values using mitools package
library(mitools)
datlist1a <- mitools::imputationList(datlist1)

# fit linear regression and apply Rubin formulas
mod2 <- with( datlist1a, stats::lm( thetaPV ~ x1 + x2 ) )
summary( mitools::MIcombine(mod2) )

## End(Not run)
```

---

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 lavaan::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).

Nonlinear terms can also be specified for loadings (=~) and regressions (~) (see Example 3).

It is also possible to construct the syntax using a loop by making use of the `DO` statement, see `doparse` for specification.

The operators `MEASERR1` and `MEASERR0` can be used for model specification for variables which contains known measurement error (see Example 6). While `MEASERR1` can be used for endogenous variables, `MEASERR0` provides the specification for exogeneous variables.

Usage

```
lavaanify.IRT(lavmodel, items=NULL, data=NULL, include.residuals=TRUE,
    doparse=TRUE)
```

Arguments

`lavmodel`	A model in `lavaan` syntax plus the additional operators `?=g1`, `?=s1`, `__` and nonlinear terms.
`items`	Optional vector of item names
`data`	Optional data frame with item responses
`include.residuals`	Optional logical indicating whether residual variances should be processed such that they are freely estimated.
`doparse`	Optional logical indicating whether `lavmodel` should be parsed for `DO` statements.

Value

A list with following entries

`lavpartable`	A `lavaan` parameter table
`lavaan.syntax`	Processed syntax for **lavaan** package
`nonlin_factors`	Data frame with renamed and original nonlinear factor specifications
`nonlin_syntable`	Data frame with original and modified syntax if nonlinear factors are used.

See Also

`lavaan::lavaanify`

See `sirt::tam2mirt` for converting objects of class `tam` into `mirt` objects.

See `sirt::lavaan2mirt` for estimating models in the **mirt** package using `lavaan` syntax.

See `doparse` for the `DO` and `DO2` statements.

Examples

```
library(lavaan)

#############################################################################
# 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 <- TAM::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 <- TAM::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 <- TAM::lavaanify.IRT( lavmodel, data=dat)
lav3$lavpartable
cat( lav3$lavaan.syntax )

#############################################################################
# EXAMPLE 3: Nonlinear terms
#############################################################################

#*** define items
items <- paste0("I",1:12)

#*** define lavaan model
lavmodel <- "
    F1=~ I1__I5
    F2=~ I6__I9
    F3=~ I10__I12
    # I3, I4 and I7 load on interaction of F1 and F2
    I(F1*F2)=~ a*I3+a*I4
    I(F1*F2)=~ I7
    # I3 and I5 load on squared factor F1
    I(F1^2)=~ I3 + I5
    # I1 regression on B spline version of factor F1
    I( bs(F1,4) )=~ I1
    F2 ~ F1 + b*I(F1^2) + I(F1>0)
```

```
   F3 ~ F1 + F2 + 1.4*I(F1*F2) + b*I(F1^2) + I(F2^2 )
   # F3 ~ F2 + I(F2^2)      # this line is ignored in the lavaan model
   F1 ~~ 1*F1
    "

#*** process lavaan syntax
lav3 <- TAM::lavaanify.IRT( lavmodel, items=items)

#*** inspect results
lav3$lavpartable
cat( lav3$lavaan.syntax )
lav3$nonlin_syntable
lav3$nonlin_factors

#############################################################################
# EXAMPLE 4: Using lavaanify.IRT for estimation with lavaan
#############################################################################

data(data.big5, package="sirt")
# extract first 10 openness items
items <- which( substring( colnames(data.big5), 1, 1 )=="O"  )[1:10]
dat <- as.data.frame( data.big5[, items ] )
  ##   > colnames(dat)
  ##    [1] "O3"  "O8"  "O13" "O18" "O23" "O28" "O33" "O38" "O43" "O48"
apply(dat,2,var)  # variances

#*** Model 1: Confirmatory factor analysis with one factor
lavmodel <- "
   O=~ O3__O48   # convenient syntax for defining the factor for all items
   O ~~ 1*O
   "
# process lavaan syntax
res <- TAM::lavaanify.IRT( lavmodel, data=dat )
# estimate lavaan model
mod1 <- lavaan::lavaan( model=res$lavaan.syntax, data=dat)
summary(mod1, standardized=TRUE, fit.measures=TRUE, rsquare=TRUE )

## End(Not run)

#############################################################################
# EXAMPLE 5: lavaanify.IRT with do statements
#############################################################################

lavmodel <- "
  DO(1,6,1)
    F=~ I%
  DOEND
  DO(1,5,2)
    A=~ I%
  DOEND
  DO(2,6,2)
    B=~ I%
  DOEND
```

```
  F ~~ 1*F
  A ~~ 1*A
  B ~~ 1*B
  F ~~ 0*A
  F ~~ 0*B
  A ~~ 0*B
   "
res <- TAM::lavaanify.IRT( lavmodel, items=paste("I",1:6) )
cat(res$lavaan.syntax)

#############################################################################
# EXAMPLE 6: Single indicator models with measurement error (MEASERR operator)
#############################################################################

# define lavaan model
lavmodel <- "
  ytrue ~ xtrue + z
  # exogeneous variable error-prone y with error variance .20
  MEASERR1(ytrue,y,.20)
  # exogeneous variable error-prone x with error variance .35
  MEASERR0(xtrue,x,.35)
  ytrue ~~ ytrue
    "
# observed items
items <- c("y","x","z")
# lavaanify
res <- TAM::lavaanify.IRT( lavmodel, items )
cat(res$lavaan.syntax)
  ##   > cat(res$lavaan.syntax)
  ##   ytrue~xtrue
  ##   ytrue~z
  ##   ytrue=~1*y
  ##   y~~0.2*y
  ##   xtrue=~1*x
  ##   x~~0.35*x
  ##   xtrue~~xtrue
  ##   ytrue~~ytrue
  ##   z~~z
```

---

msq.itemfit *Mean Squared Residual Based Item Fit Statistics (Infit, Outfit)*

Description

The function `msq.itemfit` computes computed the outfit and infit statistic for items or item groups. Contrary to `tam.fit`, the function `msq.itemfit` is not based on simulation from individual posterior distributions but rather on evaluating the individual posterior.

The function `msq.itemfit` also computes the outfit and infit statistics but these are based on weighted likelihood estimates obtained from `tam.wle`.

Usage

```
msq.itemfit( object, fitindices=NULL)

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

msq.itemfitWLE( tamobj, fitindices=NULL, ... )

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

Arguments

`object`	Object for which the classes `IRT.data`, `IRT.posterior` and `predict` are defined.
`fitindices`	Vector with parameter labels defining the item groups for which the fit should be evaluated.
`tamobj`	Object of class `tam.mml`, `tam.mml.2pl` or `tam.mml.mfr`.
`file`	Optional name of a file to which the summary should be written
`...`	Further arguments to be passed

Value

List with following entries

`itemfit`	Data frame with outfit and infit statistics.
`summary_itemfit`	Summary statistics of outfit and infit

See Also

See also `tam.fit` for simulation based assessment of item fit.

See also `eRm::itemfit` or `mirt::itemfit`.

Examples

```
## Not run:

#############################################################################
# EXAMPLE 1: Simulated data Rasch model
#############################################################################

#*** 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::tam.mml(resp=dat)
# compute WLEs
wmod1 <- TAM::tam.wle(mod1)$theta

#--- item fit from "msq.itemfit" function
fit1 <- TAM::msq.itemfit(mod1)
summary( fit1 )

#--- item fit using simulation in "tam.fit"
fit0 <- TAM::tam.fit( mod1 )
summary(fit0)

#--- item fit based on WLEs
fit2a <- TAM::msq.itemfitWLE( mod1 )
summary(fit2a)

#++ fit assessment in mirt package
library(mirt)
mod1b <- mirt::mirt( dat, model=1, itemtype="Rasch", verbose=TRUE )
print(mod1b)
sirt::mirt.wrapper.coef(mod1b)
fmod1b <- mirt::itemfit(mod1b, Theta=as.matrix(wmod1,ncol=1),
                 Zh=TRUE, X2=FALSE, S_X2=FALSE )
cbind( fit2a$fit_data, fmod1b )

#++ fit assessment in eRm package
library(eRm)
mod1c <- eRm::RM( dat )
summary(mod1c)
eRm::plotPImap(mod1c)    # person-item map
pmod1c <- eRm::person.parameter(mod1c)
fmod1c <- eRm::itemfit(pmod1c)
print(fmod1c)
plot(fmod1c)

#--- define some item groups for fit assessment

# bases on evaluating the posterior
fitindices <- rep( paste0("IG",c(1,2)), each=10)
fit2 <- TAM::msq.itemfit( mod1, fitindices )
summary(fit2)

# using WLEs
fit2b <- TAM::msq.itemfitWLE( mod1, fitindices )
summary(fit2b)

#############################################################################
# EXAMPLE 2: data.read | fit statistics assessed for testlets
```

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

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

# fit Rasch model
mod <- TAM::tam.mml( dat )

#***** item fit for each item
# based on posterior
res1 <- TAM::msq.itemfit( mod  )
summary(res1)
# based on WLEs
res2 <- TAM::msq.itemfitWLE( mod  )
summary(res2)

#***** item fit for item groups
# define item groups
fitindices <- substring( colnames(dat), 1, 1 )
# based on posterior
res1 <- TAM::msq.itemfit( mod, fitindices )
summary(res1)
# based on WLEs
res2 <- TAM::msq.itemfitWLE( mod, fitindices )
summary(res2)

#############################################################################
# EXAMPLE 3: Fit statistics for rater models
#############################################################################

library(sirt)
data(data.ratings2, package="sirt")
dat <- data.ratings2

# fit rater model "~ item*step + rater"
mod <- TAM::tam.mml.mfr( resp=dat[, paste0( "k",1:5) ],
            facets=dat[, "rater", drop=FALSE],
            pid=dat$pid, formulaA=~ item*step + rater )

# fit for parameter with "tam.fit" function
fmod1a <- TAM::tam.fit( mod )
fmod1b <- TAM::msq.itemfit( mod )
summary(fmod1a)
summary(fmod1b)

# define item groups using pseudo items from object "mod"
pseudo_items <- colnames(mod$resp)
pss <- strsplit( pseudo_items, split="-" )
item_parm <- unlist( lapply( pss, FUN=function(ll){ ll[1] } ) )
rater_parm <- unlist( lapply( pss, FUN=function(ll){ ll[2] } ) )

# fit for items with "msq.itemfit" functions
```

```
res2a <- TAM::msq.itemfit( mod, item_parm )
res2b <- TAM::msq.itemfitWLE( mod, item_parm )
summary(res2a)
summary(res2b)

# fit for raters
res3a <- TAM::msq.itemfit( mod, rater_parm )
res3b <- TAM::msq.itemfitWLE( mod, rater_parm )
summary(res3a)
summary(res3b)

## End(Not run)
```

---

plot.tam *Plot Function 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, type="expected", low=-3, high=3, ngroups=6,
                 groups_by_item=FALSE, wle=NULL, export=TRUE, export.type="png",
                    export.args=list(), observed=TRUE, overlay=FALSE,
                 ask=FALSE, package="lattice", fix.devices=TRUE, nnodes=100, ...)

## S3 method for class 'tam.mml'
plot(x, items=1:x$nitems, type="expected", low=-3, high=3, ngroups=6,
                 groups_by_item=FALSE, wle=NULL, export=TRUE, export.type="png",
                        export.args=list(), observed=TRUE, overlay=FALSE,
                ask=FALSE,  package="lattice",  fix.devices=TRUE, nnodes=100, ...)

## S3 method for class 'tam.jml'
plot(x, items=1:x$nitems, type="expected", low=-3, high=3, ngroups=6,
                 groups_by_item=FALSE, wle=NULL, export=TRUE, export.type="png",
                        export.args=list(), observed=TRUE, overlay=FALSE,
                 ask=FALSE,  package="lattice", fix.devices=TRUE, nnodes=100, ...)
```

Arguments

x	Object of class `tam`, `tam.mml` or `tam.mml`.
items	An index vector giving the items to be visualized.
type	Plot type. `type="expected"` plot the expected item response curves while `type="items"` plots the response curves of all item categories.
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.
groups_by_item	Logical indicating whether grouping of persons should be conducted item-wise. The groupings will differ from item to item in case of missing item responses.
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 subdirectory `'Plots'` of the working directory.
export.type	A string which indicates the type of the graphics export. For currently supported file types, see `grDevices::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`.
package	Used R package for plot. Can be `"lattice"` or `"graphics"`.
fix.devices	Optional logical indicating whether old graphics devices should be saved.
nnodes	Number of θ points at which item response functions are evaluated
...	Further arguments to be passed

Details

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

Value

A plot and list of computed values for `plot` (if saved as an object)

Author(s)

Margaret Wu, Thomas Kiefer, Alexander Robitzsch, Michal Modzelewski

See Also

See `CDM::IRT.irfprobPlot` for a general plot method.

Examples

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

data(data.sim.rasch)
mod <- TAM::tam.mml(data.sim.rasch)
# expected response curves
```

```
plot(mod, items=1:5, export=FALSE)
# export computed values
out <- plot(mod, items=1:5, export=FALSE)
# item response curves
plot(mod, items=1:5, type="items", export=FALSE)
# plot with graphics package
plot(mod, items=1:5, type="items", export=FALSE, ask=TRUE, package="graphics")

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

data(data.Students, package="CDM")
dat <- data.Students[, c("sc3","sc4", "mj1", "mj2" )]
dat <- na.omit(dat)
dat[ dat[,1]==3, 1 ] <- 2   # modify data
dat[ 1:20, 2 ] <- 4

# estimate model
mod1 <- TAM::tam.mml( dat )
# plot item response curves and expected response curves
plot(mod1, type="items", export=FALSE)
plot(mod1, type="expected", export=FALSE )

## End(Not run)
```

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

Description

Plots the deviance change in every iteration.

Usage

```
plotDevianceTAM(tam.obj, omitUntil=1, reverse=TRUE, change=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`.
`change`	An optional logical indicating whether deviance change or the deviance should be plotted.

Author(s)

Martin Hecht, Sebastian Weirich, Alexander Robitzsch

Examples

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

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

`predict` *Expected Values and Predicted Probabilities for Fitted* **TAM** *Models*

Description

Extracts predicted values from the posterior distribution for models fitted in **TAM**.

See `CDM::predict` for more details.

Usage

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

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

## S3 method for class 'tamaan'
predict(object, ...)
```

Arguments

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

Value

List with entries for predicted values (expectations and probabilities) for each person and each item.

See `predict` (**CDM**).

Examples

```
#############################################################################
# EXAMPLE 1: Dichotomous data sim.rasch - predict method
#############################################################################

data(data.sim.rasch)
# 1PL estimation
mod1 <- TAM::tam.mml(resp=data.sim.rasch)
# predict method
prmod1 <- IRT.predict(mod1, data.sim.rasch)
str(prmod1)
```

`Scale` *S3 Method for Standardizations and Transformations of Variables*

Description

S3 method for standardizations and transformations of variables

Usage

```
Scale(object, ...)
```

Arguments

`object`	An object
`...`	Further arguments to be passed

See Also

`base::scale`

`TAM-defunct` *Defunct* **TAM** *Functions*

Description

These functions have been removed or replaced in the **tam.jml2** package.

Usage

```
tam.jml2(...)
```

Arguments

`...`	Arguments to be passed.

Details

The `tam.jml2` is included as the default in `tam.jml`.

`TAM-utilities` *Utility Functions in* **TAM**

Description

Utility functions in **TAM**.

Usage

```
## RISE item fit statistic of two models
IRT.RISE( mod_p, mod_np, use_probs=TRUE )
## model-implied means
tam_model_implied_means(mod)

## information about used package version
tam_packageinfo(pack)
## call statement in a string format
tam_print_call(CALL)
## information about R session
tam_rsessinfo()
## grep list of arguments for a specific variable
tam_args_CALL_search(args_CALL, variable, default_value)
## requireNamespace with message of needed installation
require_namespace_msg(pkg)
## add leading zeroes
add.lead(x, width=max(nchar(x)))
## round some columns in a data frame
tam_round_data_frame(obji, from=1, to=ncol(obji), digits=3, rownames_null=FALSE)
## round some columns in a data frame and print this data frame
tam_round_data_frame_print(obji, from=1, to=ncol(obji), digits=3, rownames_null=FALSE)
## copy of CDM::osink
tam_osink(file, suffix=".Rout")
## copy of CDM::csink
tam_csink(file)

## base::matrix function with argument value byrow=TRUE
tam_matrix2(x, nrow=NULL, ncol=NULL)
## more efficient base::outer functions for operations "*", "+" and "-"
tam_outer(x, y, op="*")
## row normalization of a matrix
tam_normalize_matrix_rows(x)
## row normalization of a vector
tam_normalize_vector(x)
```

```
## aggregate function for mean and sum based on base::rowsum
tam_aggregate(x, group, mean=FALSE, na.rm=TRUE)
## column index when a value in a matrix is exceeded (used in TAM::tam.pv)
tam_interval_index(matr, rn)
## cumulative sum of row entries in a matrix
tam_rowCumsums(matr)
## extension of mvtnorm::dmvnorm to matrix entries of mean
tam_dmvnorm(x, mean, sigma, log=FALSE )
## Bayesian bootstrap in TAM (used in tam.pv.mcmc)
tam_bayesian_bootstrap(N, sample_integers=FALSE, do_boot=TRUE)
## weighted covariance matrix
tam_cov_wt(x, wt=NULL, method="ML")
## weighted correlation matrix
tam_cor_wt(x, wt=NULL, method="ML")
## generalized inverse
tam_ginv(x, eps=.05)
## generalized inverse with scaled matrix using MASS::ginv
tam_ginv_scaled(x, use_MASS=TRUE)

## remove items or persons with complete missing entries
tam_remove_missings( dat, items, elim_items=TRUE, elim_persons=TRUE )
## compute AXsi given A and xsi
tam_AXsi_compute(A, xsi)
## fit xsi given A and AXsi
tam_AXsi_fit(A, AXsi)

## maximum absolute difference between objects
tam_max_abs( list1, list2, label )
tam_max_abs_list( list1, list2)

## trimming increments in iterations
tam_trim_increment(increment, max.increment, trim_increment="cut",
     trim_incr_factor=2, eps=1E-10, avoid_na=FALSE)
## numerical differentiation by central difference
tam_difference_quotient(d0, d0p, d0m, h)
## assign elements of a list in an environment
tam_assign_list_elements(x, envir)
```

Arguments

mod_p	Fitted model
mod_np	Fitted model
mod	Fitted model
use_probs	Logical
pack	An R package
CALL	An R call
args_CALL	Arguments obtained from as.list(sys.call())

variable	Name of a variable
default_value	Default value of a variable
pkg	String
x	Vector or matrix or list
width	Number of zeroes before decimal
obji	Data frame or vector
from	Integer
to	Integer
digits	Integer
rownames_null	Logical
file	File name
suffix	Suffix for file name of summary output
nrow	Number of rows
ncol	Number of columns
y	Vector
op	An operation "*", "+" or "-"
group	Vector of grouping identifiers
mean	Logical indicating whether mean should be calculated or the sum or vector or matrix
na.rm	Logical indicating whether missing values should be removed
matr	Matrix
sigma	Matrix
log	Logical
N	Integer
sample_integers	Logical indicating whether weights for complete cases should be sampled in bootstrap
do_boot	Logical
wt	Optional vector containing weights
method	Method, see stats::cov.wt
rn	Vector
dat	Data frame
items	Vector
elim_items	Logical
elim_persons	Logical
A	Array
xsi	Vector
AXsi	Matrix

`increment` Vector

`max.increment` Numeric

`trim_increment` One of the methods "half" or "cut"

`trim_incr_factor` Factor of trimming in method "half"

`eps` Small number preventing from division by zero

`use_MASS` Logical indicating whether **MASS** package should be used.

`avoid_na` Logical indicating whether missing values should be set to zero.

`d0` Vector

`d0p` Vector

`d0m` Vector

`h` Vector

`envir` Environment variable

`list1` List

`list2` List

`label` Element of a list

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

Description

The functions 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, max_ncat=30,
          pweights=NULL)

tam.cb( dat, wlescore=NULL, group=NULL, max_ncat=30, progress=TRUE,
             pweights=NULL, digits_freq=5)

plotctt( resp, theta, Ncuts=NULL, ask=FALSE, col.list=NULL,
       package="lattice", ... )
```

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.
`max_ncat`	Maximum number of categories of variables for which frequency tables should be computed
`pweights`	Optional vector of person weights
`dat`	Data frame
`digits_freq`	Number of digits for rounding in frequency table
`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`.
`col.list`	Optional vector of colors for plotting
`package`	Package used for plotting. Can be `"lattice"` or `"graphics"`.
`...`	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::tam.mml( resp=data.mc$scored )
# estimate WLE
w1 <- TAM::tam.wle( mod )
# estimate plausible values
set.seed(789)
p1 <- TAM::tam.pv( mod, ntheta=500, normal.approx=TRUE )$pv

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

# plot empirical item response curves
plotctt( resp=data.mc$raw, theta=w1$theta, Ncuts=5, ask=TRUE)
# use graphics for plot
plotctt( resp=data.mc$raw, theta=w1$theta, Ncuts=5, ask=TRUE, package="graphics")
# change colors
col.list <- c( "darkred",  "darkslateblue", "springgreen4", "darkorange",
                "hotpink4", "navy" )
plotctt( resp=data.mc$raw, theta=w1$theta, Ncuts=5, ask=TRUE,
        package="graphics", col.list=col.list )

# CTT results for scored data
stat2 <- TAM::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::tam.ctt( resp=data.mc$raw, wlescore=w1$theta, pvscores=p1[,-1], group=group)
stat3a <- TAM::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, verbose=TRUE, 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 estimated. 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
`verbose`	Logical indicating whether output should be printed during iterations. This argument replaces `control$progress`.
`control`	See `tam.mml` for more details. Note that the default is Quasi Monte Carlo integration with 1500 nodes (`snodes=1500`, `QMC=TRUE`).
`...`	Further arguments to be passed. These arguments are used in `tam.mml` and `tam.mml.2pl`. For example, `beta.inits` or `xsi.inits` can be supplied.

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 (explained common variation; Reise, Moore & Haviland, 2010; Reise, 2012), ω_t (Omega Total), ω_a (Omega asymptotic) 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. doi:10.1007/s11336-00890993

Reise, S. P. (2012). The rediscovery of bifactor measurement models. *Multivariate Behavioral Research, 47*(5), 667-696. doi:10.1080/00273171.2012.715555

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*(6), 544-559. doi:10.1080/00223891.2010.496477

Revelle, W., & Zinbarg, R. E. (2009). Coefficients alpha, beta, omega and the glb: Comments on Sijtsma. *Psychometrika, 74*(1), 145-154. doi:10.1007/s113360089102z

Wang, W.-C., & Wilson, M. (2005). The Rasch testlet model. *Applied Psychological Measurement, 29*(2), 126-149. doi:10.1177/0146621604271053

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::tam.fa( resp=resp, irtmodel="efa", nfactors=2  )
summary(mod1a)
# varimax rotation
stats::varimax(mod1a$B.stand)
# promax rotation
stats::promax(mod1a$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::tam.fa( resp=resp, irtmodel="bifactor1", dims=dims )
summary(mod1b)

#***
# Model 1c: Bifactor model
mod1c <- TAM::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::tam.fa( resp=resp, irtmodel="bifactor2", dims=dims1 )
summary(mod1d)

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

```
data(data.timssAusTwn.scored, package="TAM")
dat <- data.timssAusTwn.scored
resp <- dat[, grep("M0", colnames(dat))]

#***
# Model 1a: Rasch testlet model with 2 testlets
dims <- c( rep(1,5), rep(2,6))
mod1a <- TAM::tam.fa( resp=resp, irtmodel="bifactor1", dims=dims )
summary(mod1a)

#***
# Model 1b: Bifactor model
mod1b <- TAM::tam.fa( resp=resp, irtmodel="bifactor2", dims=dims )
summary(mod1b)

## 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=NA, fit.facets=TRUE)

tam.jml.fit(tamobj, trim_val=10)

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

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.

`trim_val` Optional trimming value. Squared standardized reaisuals larger than `trim_val` are set to `trim_val`.

`object` Object of class `tam.fit`

`file` Optional file name for summary output

`...` Further arguments to be passed

Value

In case of `tam.mml.fit` a data frame as entry `itemfit` 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_pholm` 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_pholm` 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. doi:10.1007/9780387498393_4

See Also

Fit statistics can be also calculated by the function `msq.itemfit` which avoids simulations and directly evaluates individual posterior distributions.

See `tam.jml.fit` for calculating item fit and person fit statistics for models fitted with JML.

See `tam.personfit` for computing person fit statistics.

Item fit and person fit based on estimated person parameters can also be calculated using the `sirt::pcm.fit` function in the **sirt** package (see Example 1 and Example 2).

Examples

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

data(data.sim.rasch)
# estimate Rasch model
mod1 <- TAM::tam.mml(resp=data.sim.rasch)
# item fit
fit1 <- TAM::tam.fit( mod1 )
summary(fit1)
  ##   > summary(fit1)
  ##      parameter Outfit Outfit_t Outfit_p Infit Infit_t Infit_p
  ##   1         I1  0.966   -0.409    0.171 0.996  -0.087   0.233
  ##   2         I2  1.044    0.599    0.137 1.029   0.798   0.106
  ##   3         I3  1.022    0.330    0.185 1.012   0.366   0.179
  ##   4         I4  1.047    0.720    0.118 1.054   1.650   0.025

## Not run:

#--------
# infit and oufit based on estimated WLEs
library(sirt)

# estimate WLE
wle <- TAM::tam.wle(mod1)
# extract item parameters
b1 <- - mod1$AXsi[, -1 ]
# assess item fit and person fit
fit1a <- sirt::pcm.fit(b=b1, theta=wle$theta, data.sim.rasch )
fit1a$item       # item fit statistic
fit1a$person     # person fit statistic

#############################################################################
# EXAMPLE 2: Partial credit model data.gpcm
#############################################################################

data( data.gpcm )
dat <- data.gpcm

# estimate partial credit model in ConQuest parametrization 'item+item*step'
mod2 <- TAM::tam.mml( resp=dat, irtmodel="PCM2" )
summary(mod2)
# estimate item fit
fit2 <- TAM::tam.fit(mod2)
summary(fit2)

#=> The first three rows of the data frame correspond to the fit statistics
#     of first three items Comfort, Work and Benefit.

#--------
# infit and oufit based on estimated WLEs
```

```
# compute WLEs
wle <- TAM::tam.wle(mod2)
# extract item parameters
b1 <- - mod2$AXsi[, -1 ]
# assess fit
fit1a <- sirt::pcm.fit(b=b1, theta=wle$theta, dat)
fit1a$item

#############################################################################
# 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 <- stats::rnorm(N, 0, 1)
# simulate data
prob <- stats::plogis(outer(theta, delta, "-"))
rand <- matrix( stats::runif(N*I), nrow=N, ncol=I)
resp <- 1*(rand < prob)
colnames(resp) <- paste("I", 1:I, sep="")

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

#run TAM
mod1 <- TAM::tam.mml(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::tam.fit(mod1, FitMatrix=F)
summary(fit)

#############################################################################
# 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::tam.mml(resp=dat)
#*** assess item fit by infit and outfit statistic
fit1 <- TAM::tam.fit( mod1 )$itemfit
round( cbind( "b"=mod1$item$AXsi_.Cat1, fit1$itemfit[,-1] )[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, constraint="cases", verbose=TRUE, control=list(), version=3)
```

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

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

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 θ. The first column includes the index whereas the second column includes the fixed value. This argument can only be applied for version=1.
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.
constraint	Type of constraint for means. Either "cases" (set mean of person parameters to zero) or "items" (set mean of item parameters to zero).
verbose	Logical indicating whether output should be printed during iterations. This argument replaces control$progress.
control	A list of control arguments. See tam.mml for more details.
version	Version function which should be used. version=2 is the former tam.jml2 function in **TAM** (<2.0). The default version=3 allows efficient handling in case of missing data.
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

Value

A list with following entries

`item1`	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
`item`	Extended data frame of item parameters
`theta_summary`	Summary of person parameters
`...`	

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

# compare results with Rasch model estimated by MML
mod1b <- TAM::tam.mml(resp=resp )

# constrain item difficulties to zero
mod1c <- TAM::tam.jml(resp=resp, constraint="items")

# plot estimated parameters
plot( mod1a$xsi, mod1b$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::tam.jml(resp=resp, adj=.05)
# compare item parameters
round( rbind( mod1a$xsi, mod1d$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"=mod1a$PersonScore, "theta_adj0.3"=mod1a$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:
#*** item fit and person fit statistics
fmod1a <- TAM::tam.jml.fit(mod1a)
head(fmod1a$fit.item)
head(fmod1a$fit.person)

#*** 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::tam.jml( resp=resp, xsi.fixed=xsi.fixed )
summary(mod1e)

#*** 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::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, package="TAM")
dat <- data.gpcm

# JML estimation
mod2 <- TAM::tam.jml(resp=dat)
mod2$xsi     # extract item parameters
summary(mod2)
TAM::tam.fit(mod2)    # item and person infit/outfit statistic

#* estimate rating scale model
A <- TAM::designMatrices(resp=dat, modeltype="RSM")$A
#* estimate model with design matrix A
mod3 <- TAM::tam.jml(dat, A=A)
summary(mod3)

#############################################################################
# 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::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$resp
# JML
mod3b <- TAM::tam.jml( resp=resp1, A=mod3a$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::tam.mml.mfr
mod0 <- TAM::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::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, theta.fixed only works for version=1
mod2 <- TAM::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.latreg` *Latent Regression Model*

Description

This function fits a latent regression model $\boldsymbol{\theta} = \boldsymbol{Y}\boldsymbol{\beta} + \varepsilon$. Only the individual likelihood evaluated at a $\boldsymbol{\theta}$ grid is needed as the input. Like in `tam.mml` a multivariate normal distribution is posed on the residual distribution. Plausible values can be drawn by subsequent application of `tam.pv` (see Example 1).

Usage

```
tam.latreg(like, theta=NULL, Y=NULL, group=NULL, formulaY=NULL, dataY=NULL,
   beta.fixed=FALSE, beta.inits=NULL, variance.fixed=NULL,
   variance.inits=NULL, est.variance=TRUE, pweights=NULL, pid=NULL,
   userfct.variance=NULL, variance.Npars=NULL, verbose=TRUE, control=list())

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

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

Arguments

`like`	Individual likelihood. This can be typically extracted from fitted item response models by making use of `IRT.likelihood`.
`theta`	Used $\boldsymbol{\theta}$ grid in the fitted IRT model. If `like` is generated by the `IRT.likelihood` function, then `theta` is automatically extracted as an attribute.
`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 specified, 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.
`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 imposed on β, then set `beta.fixed=FALSE` (see Example 2, Model 2_4) which is the default.
`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.
`pweights`	An optional vector of person weights

pid An optional vector of person identifiers

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

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

verbose Optional logical indicating whether iteration should be displayed.

control List of control parameters, see tam.mml fro details.

object Object of class tam.latreg

file A file name in which the summary output will be written

x Object of class tam.latreg

... Further arguments to be passed

Value

Subset of values of tam.mml. In addition, means (M_post) and standard deviations (SD_post) are computed.

See Also

See also tam.pv for plausible value imputation.

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: Unidimensional latent regression model with fitted IRT model in
#            the sirt package
#############################################################################

library(sirt)
data(data.pisaRead, package="sirt")
dat <- data.pisaRead$data

items <- grep("R4", colnames(dat), value=TRUE )    # select test items from data
# define testlets
testlets <- substring( items, 1, 4 )
itemcluster <- match( testlets, unique(testlets) )
# fit Rasch copula model (only few iterations)
mod <- sirt::rasch.copula2( dat[,items], itemcluster=itemcluster, mmliter=5)
# extract individual likelihood
like1 <- IRT.likelihood( mod )
# fit latent regression model in TAM
Y <- dat[, c("migra", "hisei", "female") ]
mod2 <- TAM::tam.latreg( like1, theta=attr(like1, "theta"), Y=Y, pid=dat$idstud )
summary(mod2)
# plausible value imputation
pv2 <- TAM::tam.pv( mod2 )
# create list of imputed datasets
```

```
Y <- dat[, c("idstud", "idschool", "female", "hisei", "migra") ]
pvnames <- c("PVREAD")
datlist <- TAM::tampv2datalist( pv2, pvnames=pvnames, Y=Y, Y.pid="idstud")

#--- fit some models
library(mice)
library(miceadds)
# convert data list into a mice object
mids1 <- miceadds::datalist2mids( datlist )
# perform an ANOVA
mod3a <- with( mids1, stats::lm(PVREAD ~ hisei*migra) )
summary( pool( mod3a ))
mod3b <- miceadds::mi.anova( mids1, "PVREAD ~ hisei*migra" )

#############################################################################
# EXAMPLE 2: data.pisaRead - fitted IRT model in mirt package
#############################################################################

library(sirt)
library(mirt)

data(data.pisaRead, package="sirt")
dat <- data.pisaRead$data

# define dataset with item responses
items <- grep("R4", colnames(dat), value=TRUE )
resp <- dat[,items]
# define dataset with covariates
X <- dat[, c("female","hisei","migra") ]

# fit 2PL model in mirt
mod <- mirt::mirt( resp, 1, itemtype="2PL", verbose=TRUE)
print(mod)
# extract coefficients
sirt::mirt.wrapper.coef(mod)

# extract likelihood
like <- IRT.likelihood(mod)
str(like)

# fit latent regression model in TAM
mod2 <- TAM::tam.latreg( like, Y=X, pid=dat$idstud )
summary(mod2)
# plausible value imputation
pv2 <- TAM::tam.pv( mod2, samp.regr=TRUE, nplausible=5 )
# create list of imputed datasets
X <- dat[, c("idstud", "idschool", "female", "hisei", "migra") ]
pvnames <- c("PVREAD")
datlist <- TAM::tampv2datalist( pv2, pvnames=pvnames, Y=X, Y.pid="idstud")
str(datlist)

# regression using semTools package
library(semTools)
```

```
lavmodel <- "
   PVREAD ~ hisei + female
           "
mod1a <- semTools::sem.mi( lavmodel, datlist)
summary(mod1a, standardized=TRUE, rsquare=TRUE)

#############################################################################
# EXAMPLE 3: data.Students - fitted confirmatory factor analysis in lavaan
#############################################################################

library(CDM)
library(sirt)
library(lavaan)

data(data.Students, package="CDM")
dat <- data.Students
vars <- scan(what="character", nlines=1)
   urban female sc1 sc2 sc3 sc4 mj1 mj2 mj3 mj4
dat <- dat[, vars]
dat <- na.omit(dat)

# fit confirmatory factor analysis in lavaan
lavmodel <- "
   SC=~ sc1__sc4
   SC ~~ 1*SC
   MJ=~ mj1__mj4
   MJ ~~ 1*MJ
   SC ~~ MJ
        "
# process lavaan syntax
res <- TAM::lavaanify.IRT( lavmodel, dat )
# fit lavaan CFA model
mod1 <- lavaan::cfa( res$lavaan.syntax, dat, std.lv=TRUE)
summary(mod1, standardized=TRUE, fit.measures=TRUE )
# extract likelihood
like1 <- TAM::IRTLikelihood.cfa( dat, mod1 )
str(like1)
# fit latent regression model in TAM
X <- dat[, c("urban","female") ]
mod2 <- TAM::tam.latreg( like1, Y=X  )
summary(mod2)
# plausible value imputation
pv2 <- TAM::tam.pv( mod2, samp.regr=TRUE, normal.approx=TRUE )
# create list of imputed datasets
Y <- dat[, c("urban", "female" ) ]
pvnames <- c("PVSC", "PVMJ")
datlist <- TAM::tampv2datalist( pv2, pvnames=pvnames, Y=Y )
str(datlist)

## End(Not run)
```

tam.linking *Linking of Fitted Unidimensional Item Response Models in* **TAM**

Description

Performs linking of fitted unidimensional item response models in **TAM** according to the Stocking-Lord and the Haebara method (Kolen & Brennan, 2014; Gonzales & Wiberg, 2017). Several studies can either be linked by a chain of linkings of two studies (`method="chain"`) or a joint linking approach (`method="joint"`) comprising all pairwise linkings.

The linking of two studies is implemented in the `tam_linking_2studies` function.

Usage

```
tam.linking(tamobj_list, type="Hae", method="joint", pow_rob_hae=1, eps_rob_hae=1e-4,
   theta=NULL, wgt=NULL, wgt_sd=2, fix.slope=FALSE, elim_items=NULL,
   par_init=NULL, verbose=TRUE)

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

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

tam_linking_2studies( B1, AXsi1, guess1, B2, AXsi2, guess2, theta, wgt, type,
    M1=0, SD1=1, M2=0, SD2=1, fix.slope=FALSE, pow_rob_hae=1)

## S3 method for class 'tam_linking_2studies'
summary(object, file=NULL, ...)

## S3 method for class 'tam_linking_2studies'
print(x, ...)
```

Arguments

tamobj_list	List of fitted objects in **TAM**
type	Type of linking method: `"SL"` (Stocking-Lord), `"Hae"` (Haebara) or `"RobHae"` (robust Haebara). See Details for more information. The default is the Haebara linking method.
method	Chain linking (`"chain"`) or joint linking (`"joint"`)
pow_rob_hae	Power for robust Heabara linking
eps_rob_hae	Value ε for numerical approximation of loss function $\|x\|^p$ in robust Haebara linking
theta	Grid of θ points. The default is `seq(-6,6,len=101)`.
wgt	Weights defined for the `theta` grid. The default is `tam_normalize_vector( stats::dnorm( theta, sd=2 ))`.

`wgt_sd`	Standard deviation for θ grid used for linking function
`fix.slope`	Logical indicating whether the slope transformation constant is fixed to 1.
`elim_items`	List of vectors refering to items which should be removed from linking (see Model 'lmod2' in Example 1)
`par_init`	Optional vector with initial parameter values
`verbose`	Logical indicating progress of linking computation
`object`	Object of class `tam.linking` or `tam_linking_2studies`.
`x`	Object of class `tam.linking` or `tam_linking_2studies`.
`file`	A file name in which the summary output will be written
`...`	Further arguments to be passed
`B1`	Array B for first study
`AXsi1`	Matrix $A\xi$ for first study
`guess1`	Guessing parameter for first study
`B2`	Array B for second study
`AXsi2`	Matrix $A\xi$ for second study
`guess2`	Guessing parameter for second study
`M1`	Mean of first study
`SD1`	Standard deviation of first study
`M2`	Mean of second study
`SD2`	Standard deviation of second study

Details

The Haebara linking is defined by minimizing the loss function

$$\sum_i \sum_k \int \left(P_{ik}(\theta) - P^*_{ik}(\theta) \right)^2$$

A robustification of Haebara linking minimizes the loss function

$$\sum_i \sum_k \int \left(P_{ik}(\theta) - P^*_{ik}(\theta) \right)^p$$

with a power p (defined in `pow_rob_hae`) smaller than 2. He, Cui and Osterlind (2015) consider $p = 1$.

Value

List containing entries

`parameters_list`	List containing transformed item parameters
`linking_list`	List containing results of each linking in the linking chain
`M_SD`	Mean and standard deviation for each study after linking
`trafo_items`	Transformation constants for item parameters
`trafo_persons`	Transformation constants for person parameters

References

Battauz, M. (2015). **equateIRT**: An R package for IRT test equating. *Journal of Statistical Software, 68*(7), 1-22. doi:10.18637/jss.v068.i07

Gonzalez, J., & Wiberg, M. (2017). *Applying test equating methods: Using* R. New York, Springer. doi:10.1007/9783319518244

He, Y., Cui, Z., & Osterlind, S. J. (2015). New robust scale transformation methods in the presence of outlying common items. *Applied Psychological Measurement, 39*(8), 613-626. doi:10.1177/0146621615587003

Kolen, M. J., & Brennan, R. L. (2014). *Test equating, scaling, and linking: Methods and practices*. New York, Springer. doi:10.1007/9781493903177

Weeks, J. P. (2010). **plink**: An R package for linking mixed-format tests using IRT-based methods. *Journal of Statistical Software, 35*(12), 1-33. doi:10.18637/jss.v035.i12

See Also

Linking or equating of item response models can be also conducted with **plink** (Weeks, 2010), **equate**, **equateIRT** (Battauz, 2015), **equateMultiple**, **kequate** and **irteQ** packages.

See also the `sirt::linking.haberman`, `sirt::invariance.alignment` and `sirt::linking.haebara` functions in the **sirt** package.

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: Linking dichotomous data with the 2PL model
#############################################################################

data(data.ex16)
dat <- data.ex16
items <- colnames(dat)[-c(1,2)]

# fit grade 1
rdat1 <- TAM::tam_remove_missings( dat[ dat$grade==1, ], items=items )
mod1 <- TAM::tam.mml.2pl( resp=rdat1$resp[, rdat1$items], pid=rdat1$dat$idstud )
summary(mod1)

# fit grade 2
rdat2 <- TAM::tam_remove_missings( dat[ dat$grade==2, ], items=items )
mod2 <- TAM::tam.mml.2pl( resp=rdat2$resp[, rdat2$items], pid=rdat2$dat$idstud )
summary(mod2)

# fit grade 3
rdat3 <- TAM::tam_remove_missings( dat[ dat$grade==3, ], items=items )
mod3 <- TAM::tam.mml.2pl( resp=rdat3$resp[, rdat3$items], pid=rdat3$dat$idstud )
summary(mod3)

# define list of fitted models
tamobj_list <- list( mod1, mod2, mod3 )
```

```
#-- link item response models
lmod <- TAM::tam.linking( tamobj_list)
summary(lmod)

# estimate WLEs based on transformed item parameters
parm_list <- lmod$parameters_list

# WLE grade 1
arglist <- list( resp=mod1$resp, B=parm_list[[1]]$B, AXsi=parm_list[[1]]$AXsi )
wle1 <- TAM::tam.mml.wle(tamobj=arglist)

# WLE grade 2
arglist <- list( resp=mod2$resp, B=parm_list[[2]]$B, AXsi=parm_list[[2]]$AXsi )
wle2 <- TAM::tam.mml.wle(tamobj=arglist)

# WLE grade 3
arglist <- list( resp=mod3$resp, B=parm_list[[3]]$B, AXsi=parm_list[[3]]$AXsi )
wle3 <- TAM::tam.mml.wle(tamobj=arglist)

# compare result with chain linking
lmod1b <- TAM::tam.linking(tamobj_list)
summary(lmod1b)

#-- linking with some eliminated items

# remove three items from first group and two items from third group
elim_items <- list( c("A1", "E2","F1"), NULL,  c("F1","F2") )
lmod2 <- TAM::tam.linking(tamobj_list, elim_items=elim_items)
summary(lmod2)

#-- Robust Haebara linking with p=1
lmod3a <- TAM::tam.linking(tamobj_list, type="RobHae", pow_rob_hae=1)
summary(lmod3a)

#-- Robust Haeabara linking with initial parameters and prespecified epsilon value
par_init <- lmod3a$par
lmod3b <- TAM::tam.linking(tamobj_list, type="RobHae", pow_rob_hae=.1,
                eps_rob_hae=1e-3, par_init=par_init)
summary(lmod3b)

#############################################################################
# EXAMPLE 2: Linking polytomous data with the partial credit model
#############################################################################

data(data.ex17)
dat <- data.ex17

items <- colnames(dat)[-c(1,2)]

# fit grade 1
rdat1 <- TAM::tam_remove_missings( dat[ dat$grade==1, ], items=items )
mod1 <- TAM::tam.mml.2pl( resp=rdat1$resp[, rdat1$items], pid=rdat1$dat$idstud )
summary(mod1)
```

```
# fit grade 2
rdat2 <- TAM::tam_remove_missings( dat[ dat$grade==2, ], items=items )
mod2 <- TAM::tam.mml.2pl( resp=rdat2$resp[, rdat2$items], pid=rdat2$dat$idstud )
summary(mod2)

# fit grade 3
rdat3 <- TAM::tam_remove_missings( dat[ dat$grade==3, ], items=items )
mod3 <- TAM::tam.mml.2pl( resp=rdat3$resp[, rdat3$items], pid=rdat3$dat$idstud )
summary(mod3)

# list of fitted TAM models
tamobj_list <- list( mod1, mod2, mod3 )

#-- linking: fix slope because partial credit model is fitted
lmod <- TAM::tam.linking( tamobj_list, fix.slope=TRUE)
summary(lmod)

# WLEs can be estimated in the same way as in Example 1.

#############################################################################
# EXAMPLE 3: Linking dichotomous data with the multiple group 2PL models
#############################################################################

data(data.ex16)
dat <- data.ex16
items <- colnames(dat)[-c(1,2)]

# fit grade 1
rdat1 <- TAM::tam_remove_missings( dat[ dat$grade==1, ], items=items )
# create some grouping variable
group <- ( seq( 1, nrow( rdat1$dat ) ) %% 3 ) + 1
mod1 <- TAM::tam.mml.2pl( resp=rdat1$resp[, rdat1$items], pid=rdat1$dat$idstud, group=group)
summary(mod1)

# fit grade 2
rdat2 <- TAM::tam_remove_missings( dat[ dat$grade==2, ], items=items )
group <- 1*(rdat2$dat$dat$idstud > 500)
mod2 <- TAM::tam.mml.2pl( resp=rdat2$resp[, rdat2$items], pid=rdat2$dat$dat$idstud, group=group)
summary(mod2)

# fit grade 3
rdat3 <- TAM::tam_remove_missings( dat[ dat$grade==3, ], items=items )
mod3 <- TAM::tam.mml.2pl( resp=rdat3$resp[, rdat3$items], pid=rdat3$dat$idstud )
summary(mod3)

# define list of fitted models
tamobj_list <- list( mod1, mod2, mod3 )

#-- link item response models
lmod <- TAM::tam.linking( tamobj_list)

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

```
# EXAMPLE 4: Linking simulated dichotomous data with two groups
#############################################################################

library(sirt)

#*** simulate data
N <- 3000  # number of persons
I <- 30    # number of items
b <- seq(-2,2, length=I)
# data for group 1
dat1 <- sirt::sim.raschtype( rnorm(N, mean=0, sd=1), b=b )
# data for group 2
dat2 <- sirt::sim.raschtype( rnorm(N, mean=1, sd=.6), b=b )

# fit group 1
mod1 <- TAM::tam.mml.2pl( resp=dat1 )
summary(mod1)

# fit group 2
mod2 <- TAM::tam.mml.2pl( resp=dat2 )
summary(mod2)

# define list of fitted models
tamobj_list <- list( mod1, mod2 )

#-- link item response models
lmod <- TAM::tam.linking( tamobj_list)
summary(lmod)

# estimate WLEs based on transformed item parameters
parm_list <- lmod$parameters_list

# WLE grade 1
arglist <- list( resp=mod1$resp, B=parm_list[[1]]$B, AXsi=parm_list[[1]]$AXsi )
wle1 <- TAM::tam.mml.wle(tamobj=arglist)

# WLE grade 2
arglist <- list( resp=mod2$resp, B=parm_list[[2]]$B, AXsi=parm_list[[2]]$AXsi )
wle2 <- TAM::tam.mml.wle(tamobj=arglist)
summary(wle1)
summary(wle2)

# estimation with linked and fixed item parameters for group 2
B <- parm_list[[2]]$B
xsi.fixed <- cbind( 1:I, -parm_list[[2]]$AXsi[,2] )
mod2f <- TAM::tam.mml( resp=dat2, B=B, xsi.fixed=xsi.fixed )
summary(mod2f)

## 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=TRUE, constraint="cases", A=NULL,
    B=NULL, B.fixed=NULL,  Q=NULL,  est.slopegroups=NULL, E=NULL,
    pweights=NULL, userfct.variance=NULL,
    variance.Npars=NULL, item.elim=TRUE, verbose=TRUE, 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, gamma.init=NULL,
    pweights=NULL, userfct.variance=NULL, variance.Npars=NULL,
    item.elim=TRUE, verbose=TRUE, control=list() )

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

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

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

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 (only available 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` or `GPCM.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 specified, e. g. `female*race` or `I(age^2)`.
dataY	An optional data frame with possible covariates Y in latent regression. This data frame is 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.setnull	A vector of strings indicating which ξ elements should be set to zero which do have entries in `xsi.setnull` in their labels (see Example 10a).
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 imposed 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.fixed`.
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` and `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"`)

`gamma.init` Optional initial $gamma$ parameter vector (`irtmodel="GPCM.design"`).

`pweights` An optional vector of person weights

`formulaA` Design formula (only applies to `tam.mml.mfr`). See Example 8. It is also to possible to set all effects of a facet to zero, e.g. `item*step + 0*rater` (see Example 10a).

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

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

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

`item.elim` Optional logical indicating whether an item with has only zero entries should be removed from the analysis. The default is `TRUE`.

`verbose` Logical indicating whether output should be printed during iterations. This argument replaces `control$progress`.

`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=0, increment.factor=1,
fac.oldxsi=0, acceleration="none", dev_crit="absolute",
trim_increment="half" )
```

`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 (Gonzales at al., 2006; Pan & Thompson, 2007) should be used. The default is `TRUE`. Quasi

Monte Carlo integration 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. A high value of M steps could be helpful in cases of non-convergence. In `tam.mml`, `tam.mml.2pl` and `tam.mml.mfr`, the default is set to 4, in `tam.mml.3pl` it is set to 10.

`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 numeric value. The value of 0 indicates that for all parameters starting values are provided. A value of 1 means that all starting values are set to zero and a value of 2 means that only starting values of item parameters (but not facet parameters) are used.

`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).

`acceleration`: String indicating whether convergence acceleration of the EM algorithm should be employed. Options are `"none"` (no acceleration, the default), the monotone overrelaxation method of `"Yu"` (Yu, 2012) and `"Ramsay"` for the Ramsay (1975) acceleration method.

`dev_crit`: Criterion for convergence in deviance. `dev_crit="absolute"` refers to absolute differences in successive deviance values, while `dev_crit="relative"` refers to relative differences.

`trim_increment`: Type of method for trimming parameter increments in algorithm. Possible types are `"half"` or `""cut"`.

`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 should be written (only for `summary.tam` functions)

`...`
: Further arguments to be passed

`x`
: Object of class `tam` or `tam.mml`

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 response 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 implemented for unidimensional and multidimensional item response models. In this case, variance heterogeneity is allowed

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

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$snodes` with a value larger than zero, then Quasi-Monte Carlo integration (Pan & Thomas, 2007; Gonzales et al., 2006) with `control$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. The column `xsi.item` denotes the item difficulty of polytomous items in the parametrization `irtmodel="PCM2"`.
item_irt	IRT parameterization of 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
- `xsi.fixed.estimated` Matrix of estimated ξ parameters in form of `xsi.fixed` which can be used for parameter fixing in subsequent estimations.
- `B.fixed` Fixed loading parameters (only applies to `tam.mml.2pl`)
- `B.fixed.estimated` Matrix of estimated B parameters in the same format as `B.fixed`.
- `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
- `latreg_stand` List containing standardized regression coefficients
- `...` 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) or Gonzales et al. (2006) for more details.

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

Several choices can be made if your model does not converge. First, the number of iterations within a M step can be increased (`Msteps=10`). Second, the absolute value of increments can be forced with increasing iterations (set a value higher than 1 to `max.increment`, maybe 1.05). Third, change in estimated parameters can be stabilized by `fac.oldxsi` for which a value of 0 (the default) and a value of 1 can be chosen. We recommend values between .5 and .8 if your model does not converge.

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. doi:10.3102/10769986022001047

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. doi:10.1007/9780387498393_4

Embretson, S. E. (1999). Generating items during testing: Psychometric issues and models. *Psychometrika, 64*, 407-433. doi:10.1007/BF02294564

Gonzalez, J., Tuerlinckx, F., De Boeck, P., & Cools, R. (2006). Numerical integration in logistic-normal models. *Computational Statistics & Data Analysis, 51*, 1535-1548. doi:10.1016/j.csda.2006.05.003

Pan, J., & Thompson, R. (2007). Quasi-Monte Carlo estimation in generalized linear mixed models. *Computational Statistics & Data Analysis, 51*, 5765-5775. doi:10.1016/j.csda.2006.10.003

Ramsay, J. O. (1975). Solving implicit equations in psychometric data analysis. *Psychometrika, 40*(3), 337-360. doi:10.1007/BF02291762

Yu, Y. (2012). Monotonically overrelaxed EM algorithms. *Journal of Computational and Graphical Statistics, 21*(2), 518-537. doi:10.1080/10618600.2012.672115

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 `sirt::tam2mirt` for converting `tam` objects into objects of class `mirt::mirt` in the **mirt** package.

Examples

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

#**************************************************************
# Model 1: Rasch model (MML estimation)
mod1 <- TAM::tam.mml(resp=data.sim.rasch)
# extract item parameters
mod1$item    # item difficulties
```

```
## Not run:
# WLE estimation
wle1 <- TAM::tam.wle( mod1 )
# item fit
fit1 <- TAM::tam.fit(mod1)
# plausible value imputation
pv1 <- TAM::tam.pv(mod1, normal.approx=TRUE, ntheta=300)
# standard errors
se1 <- TAM::tam.se( mod1 )
# summary
summary(mod1)

#-- specification with tamaan
tammodel <- "
 LAVAAN MODEL:
   F=~ I1__I40;
   F ~~ F
 ITEM TYPE:
   ALL(Rasch)
   "
mod1t <- TAM::tamaan( tammodel, data.sim.rasch)
summary(mod1t)

#*************************************************************
# Model 1a: 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::tam.mml( resp=data.sim.rasch, xsi.fixed=xsi.fixed )
summary(mod1a)

# Usage of the output value mod1$xsi.fixed.estimated has the right format
# as the input of xsi.fixed
mod1aa <- TAM::tam.mml( resp=data.sim.rasch, xsi.fixed=mod1$xsi.fixed.estimated )
summary(mod1b)

#*************************************************************
# Model 1b: 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::tam.mml( resp=data.sim.rasch, xsi.inits=xsi.inits )

#-- tamaan specification
tammodel <- "
 LAVAAN MODEL:
   F=~ I1__I40
   F ~~ F
   # Fix item difficulties. Note that item intercepts instead of difficulties
   # must be specified.
   I2 | 1.8*t1
   I4 | 1.6*t1
 ITEM TYPE:
   ALL(Rasch)
```

```
  "
mod1bt <- TAM::tamaan( tammodel, data.sim.rasch)
summary(mod1bt)

#*************************************************************
# Model 1c: 1PL estimation with sum constraint on item difficulties
dat <- data.sim.rasch
# modify A design matrix to include the sum constraint
des <- TAM::designMatrices(resp=dat)
A1 <- des$A[,, - ncol(dat) ]
A1[ ncol(dat),2, ] <- 1
A1[,2,]
# estimate model
mod1c <- TAM::tam.mml( resp=dat, A=A1, beta.fixed=FALSE,
           control=list(fac.oldxsi=.1) )
summary(mod1c)

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

#*************************************************************
# Model 1e: This sum constraint can also be obtained by using the argument
# constraint="items" in tam.mml
mod1e <- TAM::tam.mml( resp=data.sim.rasch, constraint="items" )
summary(mod1e)

#*************************************************************
# Model 1d2: 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)) )
formulaA=~ 0 + item_

mod1d2 <- TAM::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)


#*************************************************************
```

```
# Model 2: 2PL model
mod2 <- TAM::tam.mml.2pl(resp=data.sim.rasch,irtmodel="2PL")

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

#--- tamaan specification
tammodel <- "
 LAVAAN MODEL:
   F=~ I1__I40
   F ~~ 1*F
   # item type of 2PL is the default for dichotomous data
   "
# estimate model
mod2t <- TAM::tamaan( tammodel, data.sim.rasch)
summary(mod2t)

## Not run:
#*************************************************************
# Model 2a: 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::tam.mml( resp=data.sim.rasch, xsi.fixed=xsi.fixed,
                 B=mod2$B # fix slopes
            )
summary(mod2a)
mod2a$B     # inspect used slope matrix

#*************************************************************
# Model 3: 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::tam.mml.2pl( resp=data.sim.rasch, irtmodel="2PL.groups",
               est.slopegroups=est.slope )
mod3$B
summary(mod3)

#--- tamaan specification (A)
tammodel <- "
 LAVAAN MODEL:
   F=~ lam1*I1__I10 + lam2*I11__I20 + lam1*I21__I30 + lam3*I31__I40;
   F ~~ 1*F
   "
# estimate model
mod3tA <- TAM::tamaan( tammodel, data.sim.rasch)
summary(mod3tA)
```

```
#--- tamaan specification (alternative B)
tammodel <- "
 LAVAAN MODEL:
   F=~ a1__a40*I1__I40;
   F ~~ 1*F
 MODEL CONSTRAINT:
   a1__a10==lam1
   a11__a20==lam2
   a21__a30==lam1
   a31__a40==lam3
   "
mod3tB <- TAM::tamaan( tammodel, data.sim.rasch)
summary(mod3tB)

#--- tamaan specification (alternative C using DO operator)
tammodel <- "
 LAVAAN MODEL:
 DO(1,10,1)
   F=~ lam1*I%
 DOEND
 DO(11,20,1)
   F=~ lam2*I%
 DOEND
 DO(21,30,1)
   F=~ lam1*I%
 DOEND
 DO(31,40,1)
   F=~ lam3*I%
 DOEND
   F ~~ 1*F
   "
# estimate model
mod3tC <- TAM::tamaan( tammodel, data.sim.rasch)
summary(mod3tC)

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

```
# (2) estimate model
mod2_1 <- TAM::tam.mml(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::tam.mml(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::tam.mml(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::tam.mml(resp=resp, dataY=dataY, formulaY=~ Y1+Y2 )
summary(mod2_4)

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

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

#############################################################################
# EXAMPLE 3: Multiple group estimation
#############################################################################

# (1) simulate data
set.seed(6778)
N <- 3000
theta <- c( stats::rnorm(N/2,mean=0,sd=1.5), stats::rnorm(N/2,mean=.5,sd=1)  )
I <- 20
p1 <- stats::plogis( outer( theta, seq( -2, 2, len=I ), "-" ) )
resp <- 1 * ( p1 > matrix( stats::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::tam.mml( resp,  group=group )
summary(mod3_1)

#############################################################################
# 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( stats::rnorm( N ), stats::rnorm(N) )
theta <- mvtnorm::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 <- stats::plogis( outer( theta[,1], seq( -2, 2, len=I ), "-" ) )
resp1 <- 1 * ( p1 > matrix( stats::runif( N*I ), nrow=N, ncol=I ) )
p1 <- stats::plogis( outer( theta[,2], seq( -2, 2, len=I ), "-" ) )
resp2 <- 1 * ( p1 > matrix( stats::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

#************************************************************
# Model 4.1: Rasch model: without regressors
mod4_1 <- TAM::tam.mml( resp=resp, Q=Q )

#--- tamaan specification
tammodel <- "
  LAVAAN MODEL:
    F1=~ 1*I1__I20
    F2=~ 1*I21__I40
    # Alternatively to the factor 1 one can use the item type Rasch
    F1 ~~ F1
    F2 ~~ F2
    F1 ~~ F2
    "
mod4_1t <- TAM::tamaan( tammodel, resp, control=list(maxiter=100))
summary(mod4_1t)

#************************************************************
# Model 4.1b: estimate model with sum constraint of items for each dimension
mod4_1b <- TAM::tam.mml( resp=resp, Q=Q,constraint="items")

#************************************************************
# Model 4.2: Rasch model: set covariance between dimensions to zero
variance_fixed <- cbind( 1, 2, 0 )
mod4_2 <- TAM::tam.mml( resp=resp, Q=Q, variance.fixed=variance_fixed )
summary(mod4_2)

#--- tamaan specification
tammodel <- "
  LAVAAN MODEL:
    F1=~ I1__I20
    F2=~ I21__I40
```

```
    F1 ~~ F1
    F2 ~~ F2
    F1 ~~ 0*F2
  ITEM TYPE:
    ALL(Rasch)
    "
mod4_2t <- TAM::tamaan( tammodel, resp)
summary(mod4_2t)

#*********************************************************************
# Model 4.3: 2PL model
mod4_3 <- TAM::tam.mml.2pl( resp=resp, Q=Q, irtmodel="2PL" )

#--- tamaan specification
tammodel <- "
  LAVAAN MODEL:
    F1=~ I1__I20
    F2=~ I21__I40
    F1 ~~ F1
    F2 ~~ F2
    F1 ~~ F2
    "
mod4_3t <- TAM::tamaan( tammodel, resp )
summary(mod4_3t)

#*********************************************************************
# Model 4.4: Rasch model with 2000 quasi monte carlo nodes
# -> nodes are useful for more than 3 or 4 dimensions
mod4_4 <- TAM::tam.mml( resp=resp, Q=Q, control=list(snodes=2000) )

#*********************************************************************
# Model 4.5: Rasch model with 2000 stochastic nodes
mod4_5 <- TAM::tam.mml( resp=resp, Q=Q,control=list(snodes=2000,QMC=FALSE))

#*********************************************************************
# Model 4.6: estimate two dimensional Rasch model with regressors
mod4_6 <- TAM::tam.mml( resp=resp, Y=Y, Q=Q )

#--- tamaan specification
tammodel <- "
  LAVAAN MODEL:
    F1=~ I1__I20
    F2=~ I21__I40
    F1 ~~ F1
    F2 ~~ F2
    F1 ~~ F2
  ITEM TYPE:
    ALL(Rasch)
    "
mod4_6t <- TAM::tamaan( tammodel, resp, Y=Y )
summary(mod4_6t)

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

```
# EXAMPLE 5: 2-dimensional estimation with within item dimensionality
#############################################################################
library(mvtnorm)
# (1) simulate data
set.seed(4762)
N <- 2000 # 2000 persons
Y <- stats::rnorm( N )
theta <- mvtnorm::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 <- stats::plogis( outer( theta[,1], seq( -2, 2, len=I ), "-" ) )
resp1 <- 1 * ( p1 > matrix( stats::runif( N*I ), nrow=N, ncol=I ) )
# 10 items load on the second dimension
p1 <- stats::plogis( outer( theta[,2], seq( -2, 2, len=I ), "-" ) )
resp2 <- 1 * ( p1 > matrix( stats::runif( N*I ), nrow=N, ncol=I ) )
# 20 items load on both dimensions
p1 <- stats::plogis( outer( 0.5*theta[,1] + 1.5*theta[,2], seq(-2,2,len=2*I ), "-" ))
resp3 <- 1 * ( p1 > matrix( stats::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::tam.mml.2pl(resp, Q=Q, irtmodel="2PL" )
summary(mod5)

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

#--- tamaan specification
tammodel <- "
  LAVAAN MODEL:
    F1=~ I1__I10 + I21__I40
    F2=~ I11__I20 + I21__I40
    F1 ~~ 1*F1
    F1 ~~ F2
    F2 ~~ 1*F2
    "
mod5t <- TAM::tamaan( tammodel, resp,  control=list(maxiter=10))
summary(mod5t)

#############################################################################
# EXAMPLE 6: ordered data - Generalized partial credit model
#############################################################################
data(data.gpcm, package="TAM")

#***********************************************************
# Ex6.1: Nominal response model (irtmodel="2PL")
```

```
mod6_1 <- TAM::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

# reestimate the model with fixed item parameters
mod6_1a <- TAM::tam.mml.2pl( resp=data.gpcm, irtmodel="2PL",
       xsi.fixed=mod6_1$xsi.fixed.estimated,  B.fixed=mod6_1$B.fixed.estimated,
       est.variance=TRUE )

# estimate the model with initial item parameters from mod6_1
mod6_1b <- TAM::tam.mml.2pl( resp=data.gpcm, irtmodel="2PL",
       xsi.inits=mod6_1$xsi.fixed.estimated,  B=mod6_1$B )

#*************************************************************
# Ex6.2: Generalized partial credit model
mod6_2 <- TAM::tam.mml.2pl( resp=data.gpcm, irtmodel="GPCM", control=list(maxiter=200))
mod6_2$B[,2,]    # 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::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::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::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::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 <- TAM::.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::tam.mml( resp=data.gpcm, A=A1, beta.fixed=FALSE )
summary(mod6_6b)

# estimate model with argument constraint="items"
mod6_6c <- TAM::tam.mml( resp=data.gpcm, irtmodel="PCM2", constraint="items")

# estimate mod6_6 using tam.mml.mfr
mod6_6d <- TAM::tam.mml.mfr( resp=data.gpcm, formulaA=~ 0 + item + item:step,
    control=list(fac.oldxsi=.1), constraint="items" )
summary(mod6_6d)

#*************************************************************
# Ex 6.7: Rating scale model: Conquest parametrization 'item+step'
mod6_7 <- TAM::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 <- TAM::.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::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::tam.mml.mfr( resp=dat, formulaA=~ 0 + item + item:step,
                        control=list(fac.oldxsi=.1), constraint="items" )
```

```
summary(mod6_8b)

#************************************************************
# Ex 6.9: sum constraint on item difficulties for irtmodel="PCM"

data(data.timssAusTwn.scored)
dat <- data.timssAusTwn.scored[,2:11]
dat[ dat==9 ] <- NA

# obtain the design matrix for the PCM parametrization and
# the number of categories for each item
maxKi <- apply(dat, 2, max, na.rm=TRUE)
des <- TAM::designMatrices(resp=dat)
A1 <- des$A

# define the constrained item category and remove the respective parameter
(par <- unlist( strsplit(dimnames(A1)[[3]][dim(A1)[3]], split="_") ))
A1 <- A1[,,-dim(A1)[3]]

# the item category loads on every other item category parameter with
# opposing margin, balancing the number of categories for each item
item.id <- which(colnames(dat)==par[1])
cat.id <- maxKi[par[1]]+1
loading <- 1/rep(maxKi, maxKi)
loading <- loading [-which(names(loading)==par[1])[1]]
A1[item.id, cat.id, ] <- loading
A1[item.id,,]

# estimate model
mod6_9 <- TAM::tam.mml( resp=dat, A=A1, beta.fixed=FALSE )
summary(mod6_9)

## extract fixed item category parameter
# calculate mean for each item
ind.item.cat.pars <- sapply(colnames(dat), grep, rownames(mod6_8$xsi))
item.means <- lapply(ind.item.cat.pars, function(ii) mean(mod6_8$xsi$xsi[ii]))

# these sum up to the negative of the fixed parameter
fix.par <- -sum( unlist(item.means), na.rm=TRUE)

#************************************************************
# Ex 6.10: Generalized partial credit model with equality constraints
#          on item discriminations

data(data.gpcm)
dat <- data.gpcm

# Ex 6.10a: set all slopes of three items equal to each other
E <- matrix( 1, nrow=3, ncol=1 )
mod6_10a <- TAM::tam.mml.2pl( dat, irtmodel="GPCM.design", E=E  )
summary(mod6_10a)
mod6_10a$B[,,]
```

```
# Ex 6.10b: equal slope for first and third item
E <- matrix( 0, nrow=3, ncol=2 )
E[c(1,3),1] <- 1
E[ 2, 2 ] <- 1
mod6_10b <- TAM::tam.mml.2pl( dat, irtmodel="GPCM.design", E=E  )
summary(mod6_10b)
mod6_10b$B[,,]

#############################################################################
# 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

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

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

# recalculate estimated basis parameters
stats::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::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::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 <- TAM::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::tam.mml( resp=des$gresp$gresp.noStep, A=A1 )
summary(mod8b)

#############################################################################
# EXAMPLE 9: Multifaceted Rasch Model
#############################################################################

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

# two way interaction item and rater
formulaA <- ~item+item:step + item*rater
mod9a <- TAM::tam.mml.mfr( resp=data.sim.mfr, facets=data.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::tam.mml.mfr( resp=data.sim.mfr, facets=data.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::tam.mml.mfr( resp=resp, facets=facets, formulaA=formulaA, pid=dat$pid )
summary(mod10)

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

#--- Example 10a
# compare model containing only item
formulaA <- ~ item + rater      # pseudo formula for item
xsi.setnull <- "rater"          # set all rater effects to zero
mod10a <- TAM::tam.mml.mfr( resp=resp, facets=facets, formulaA=formulaA,
```

```
            xsi.setnull=xsi.setnull, pid=dat$pid, beta.fixed=cbind(1,1,0))
summary(mod10a)

# A shorter way for specifying this example is
formulaA <- ~ item + 0*rater        # set all rater effects to zero
mod10a1 <- TAM::tam.mml.mfr( resp=resp, facets=facets, formulaA=formulaA, pid=dat$pid )
summary(mod10a1)

# tam.mml.mfr also appropriately extends the facets data frame with pseudo facets
# if necessary
formulaA <- ~ item     # omitting the rater term
mod10a2 <- TAM::tam.mml.mfr( resp=resp, facets=facets, formulaA=formulaA, pid=dat$pid )
  ##   Item Parameters Xsi
  ##              xsi se.xsi
  ##   I0001   -1.931  0.111
  ##   I0002   -1.023  0.095
  ##   I0003   -0.089  0.089
  ##   I0004    1.015  0.094
  ##   I0005    1.918  0.110
  ##   psfPF11  0.000  0.000
  ##   psfPF12  0.000  0.000

#***
# 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::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::tam.mml( resp=dat1 )
summary(mod11a)
# compute person parameters
```

```
wmod11a <- TAM::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::tam.mml( resp=dat2, xsi.fixed=xsi.fixed )
summary(mod11b)
# WLE estimation
wmod11b <- TAM::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::tam.mml.2pl( resp=data.ex12, control=list( maxiter=1000) )

# avoiding non-convergence with increment.factor=1.02
mod2 <- TAM::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 <- TAM::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::tam.mml( resp=data.long[,-1], Q=Q, A=A1, beta.fixed=beta.fixed)
summary(mod13a)

#--- tamaan specification
tammodel <- "
  LAVAAN MODEL:
    T1=~ 1*I1T1__I6T1
    T2=~ 1*I3T2__I8T2
    T1 ~~ T1
    T2 ~~ T2
    T1 ~~ T2
    # constraint on item difficulties
    I3T1 + I3T2 | b3*t1
    I4T1 + I4T2 | b4*t1
    I5T1 + I5T2 | b5*t1
    I6T1 + I6T2 | b6*t1
    "
```

```
# The constraint on item difficulties can be more efficiently written as
  ##       DO(3,6,1)
  ##         I%T1 + I%T2 | b%*t1
  ##       DOEND
# estimate model
mod13at <- TAM::tamaan( tammodel, resp=data.long,  beta.fixed=beta.fixed )
summary(mod13at)

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

A <- TAM::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::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 <- TAM::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::tam.mml( resp=dat[,-1], Q=Q, A=A1, beta.fixed=beta.fixed,
                   irtmodel="PCM")
summary(mod13c)
wle.mod13c <- TAM::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$rater )     # define facets
formulaA <- ~item+item*step + rater
mod14a <- TAM::tam.mml.mfr( resp, facets=facets, formulaA=formulaA, pid=dat$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::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::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::tam.mml.mfr(resp=data[,-c(1:2)],facets=data[,"group",drop=FALSE],
    formulaA=~ item + group*item, pid=data$pid )
# 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::tam.mml.mfr(resp=data[,-c(1:2)],facets=data[,"group",drop=FALSE],
            formulaA=~ group*item, pid=data$pid,
            control=list( increment.factor=1.03, fac.oldxsi=.4 ) )

#***
# Model 15c: Alternatively, just choose more iterations in M-step by "Msteps=10"
mod15c <- TAM::tam.mml.mfr(resp=data[,-c(1:2)],facets=data[,"group",drop=FALSE],
    formulaA=~ item + group*item, pid=data$pid,
    control=list(maxiter=250, Msteps=10))

#############################################################################
# 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$IDCNTRY)
# create design matrices
des2 <- TAM::designMatrices.mfr2( resp=resp, facets=facets,
                   formulaA=~item*step + item*cnt)
# restructured data set: pseudoitem=item x country
resp2 <- des2$gresp$gresp.noStep
# A design matrix
A <- des2$A$A.3d
    # redundant xsi parameters must be eliminated from design matrix
xsi.elim <- des2$xsi.elim
A <- A[,, - xsi.elim[,2] ]
# extract loading matrix B
B <- des2$B$B.3d
# estimate model
mod1 <- TAM::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

#############################################################################
# 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( stats::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 <- 200 ; 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::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::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::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::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::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::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)

#############################################################################
# EXAMPLE 18: Partial credit model with dimension-specific sum constraints
#             on item difficulties
#############################################################################

data(data.Students, package="CDM")
dat <- data.Students[, c( paste0("sc",1:4), paste0("mj",1:4) ) ]
# specify dimensions in Q-matrix
Q <- matrix( 0,  nrow=8, ncol=2 )
Q[1:4,1] <- Q[5:8,2] <- 1
```

```
# partial credit model with some constraint on item parameters
mod1 <- TAM::tam.mml( dat, Q=Q, irtmodel="PCM2", constraint="items")
summary(mod1)

#############################################################################
# EXAMPLE 19: Partial credit scoring: 0.5 points for partial credit items
#             and 1 point for dichotomous items
#############################################################################

data(data.timssAusTwn.scored)
dat <- data.timssAusTwn.scored
# extrcat item response data
dat <- dat[, grep("M03", colnames(dat) ) ]

# select items with do have maximum score of 2 (polytomous items)
ind <- which( apply( dat,  2, max, na.rm=TRUE )==2 )
I <- ncol(dat)
# define Q-matrix with scoring variant
Q <- matrix( 1, nrow=I, ncol=1 )
Q[ ind, 1 ] <- .5    # score of 0.5 for polyomous items

# estimate model
mod1 <- TAM::tam.mml( dat, Q=Q, irtmodel="PCM2", control=list(nodes=seq(-10,10,len=21)))
summary(mod1)

#############################################################################
# EXAMPLE 20: Specification of loading matrix in multidimensional model
#############################################################################

data(data.gpcm)
psych::describe(data.gpcm)
resp <- data.gpcm

# define three dimensions and different loadings of item categories
# on these dimensions in B loading matrix
I <- 3  # 3 items
D <- 3  # 3 dimensions
# define loading matrix B
# 4 categories for each item (0,1,2,3)
B <- array( 0, dim=c(I,4,D) )
for (ii in 1:I){
    B[ ii, 1:4, 1 ] <- 0:3
    B[ ii, 1,2 ] <- 1
    B[ ii, 4,3 ] <- 1
            }
dimnames(B)[[1]] <- colnames(resp)
B[1,,]
  ##   > B[1,,]
  ##        [,1] [,2] [,3]
  ##   [1,]    0    1    0
  ##   [2,]    1    0    0
  ##   [3,]    2    0    0
  ##   [4,]    3    0    1
```

```
#-- test run
mod1 <- TAM::tam.mml( resp, B=B, control=list( snodes=1000, maxiter=5)  )
summary(mod1)

# Same model with TAM::tam.mml.3pl instead

dim4 <- sum(apply(B, c(1, 3), function(x) any(!(x==0))))
E1 <- array(0, dim=c(dim(B), dim4))

kkk <- 0
for (iii in seq_len(dim(E1)[1])) {
    for (jjj in seq_len(dim(E1)[3])) {
        if (any(B[iii,, jjj] !=0)) {
            kkk <- kkk + 1
            E1[iii,, jjj, kkk] <- B[iii,, jjj]
        }
    }
}
if (kkk !=dim4) stop("Something went wrong in the loop, because 'kkk !=dim4'.")

mod2 <- TAM::tam.mml.3pl(resp, E=E1, est.some.slopes=FALSE, control=list(maxiter=50))
summary(mod2)

cor(mod1$person$EAP.Dim3, mod2$person$EAP.Dim3)
cor(mod1$person$EAP.Dim2, mod2$person$EAP.Dim2)
cor(mod1$person$EAP.Dim1, mod2$person$EAP.Dim1)
cor(mod1$xsi$xsi, mod2$xsi$xsi)

#############################################################################
# EXAMPLE 21: Acceleration of EM algorithm | dichotomous data
#############################################################################

N <- 1000      # number of persons
I <- 100       # number of items
set.seed(987)
# simulate data according to the Rasch model
dat <- sirt::sim.raschtype( stats::rnorm(N), b=seq(-2,2,len=I)  )
# estimate models
mod1n <- TAM::tam.mml( resp=dat, control=list( acceleration="none") )  # no acceler.
mod1y <- TAM::tam.mml( resp=dat, control=list( acceleration="Yu") )  # Yu acceler.
mod1r <- TAM::tam.mml( resp=dat, control=list( acceleration="Ramsay") )  # Ramsay acceler.
# compare number of iterations
mod1n$iter ; mod1y$iter ; mod1r$iter
# log-likelihood values
logLik(mod1n); logLik(mod1y) ; logLik(mod1r)

#############################################################################
# EXAMPLE 22: Acceleration of EM algorithm | polytomous data
#############################################################################

data(data.gpcm)
dat <- data.gpcm
```

```
# no acceleration
mod1n <- TAM::tam.mml.2pl( resp=dat, irtmodel="GPCM",
                control=list( conv=1E-4, acceleration="none") )
# Yu acceleration
mod1y <- TAM::tam.mml.2pl( resp=dat, irtmodel="GPCM",
                control=list( conv=1E-4, acceleration="Yu") )
# Ramsay acceleration
mod1r <- TAM::tam.mml.2pl( resp=dat, irtmodel="GPCM",
                control=list( conv=1E-4, acceleration="Ramsay") )
# number of iterations
mod1n$iter ; mod1y$iter ; mod1r$iter

#############################################################################
# EXAMPLE 23: Multidimensional polytomous Rasch model in which
#             dimensions are defined by categories
#############################################################################

data(data.Students, package="CDM")
dat <- data.Students[, grep( "act", colnames(data.Students) ) ]

# define multidimensional model in which categories of item define dimensions

# * Category 0 -> loading of one on Dimension 0
# * Category 1 -> no loadings
# * Category 2 -> loading of one on Dimension 2

# extract default design matrices
res <- TAM::designMatrices( resp=dat )
A <- res$A
B0 <- 0*res$B
# create design matrix B
B <- array( 0, dim=c( dim(B0)[c(1,2) ], 2  ) )
dimnames(B)[[1]] <- dimnames(B0)[[1]]
dimnames(B)[[2]] <- dimnames(B0)[[2]]
dimnames(B)[[3]] <- c( "Dim0", "Dim2" )
B[,1,1]  <- 1
B[,3,2]  <- 1

# estimate multdimensional Rasch model
mod1 <- TAM::tam.mml( resp=dat, A=A, B=B, control=list( maxiter=100) )
summary(mod1)

# alternative definition of B
# * Category 1: negative loading on Dimension 1 and Dimension 2
B <- array( 0, dim=c( dim(B0)[c(1,2) ], 2  ) )
dimnames(B)[[1]] <- dimnames(B0)[[1]]
dimnames(B)[[2]] <- dimnames(B0)[[2]]
dimnames(B)[[3]] <- c( "Dim0", "Dim2" )
B[,1, 1]  <- 1
B[,3, 2]  <- 1
B[,2, c(1,2)]  <- -1

# estimate model
```

```
mod2 <- TAM::tam.mml( resp=dat, A=A, B=B, control=list( maxiter=100) )
summary(mod2)

#############################################################################
# EXAMPLE 24: Sum constraint on item-category parameters in partial credit model
#############################################################################

data(data.gpcm,package="TAM")
dat <- data.gpcm

# check number of categories
c1 <- TAM::tam.ctt3(dat)

#--- fit with PCM
mod1 <- TAM::tam.mml( dat )
summary(mod1, file="mod1")

#--- fit with constraint on sum of categories
#** redefine design matrix
A1 <- 0*mod1$A
A1 <- A1[,, - dim(A1)[[3]]]
str(A1)
NP <- dim(A1)[[3]]
# define item category parameters
A1[1,2,1] <- A1[1,3,2] <- A1[1,4,3] <- -1
A1[2,2,4] <- A1[2,3,5] <- A1[2,4,6] <- -1
A1[3,2,7] <- A1[3,3,8] <- -1
A1[3,4,1:8] <- 1
# check definition
A1[1,,]
A1[2,,]
A1[3,,]

#** estimate model
mod2 <- TAM::tam.mml( dat, A=A1, beta.fixed=FALSE)
summary(mod2, file="mod2")

#--- compare model fit
IRT.compareModels(mod1, mod2 )  # -> equivalent model fit

#############################################################################
# EXAMPLE 25: Different GPCM parametrizations in IRT packages
#############################################################################

library(TAM)
library(mirt)
library(ltm)

data(data.gpcm, package="TAM")
dat <- data.gpcm

#*** TAM
mod1 <- TAM::tam.mml.2pl(dat, irtmodel="GPCM")
```

```
#*** mirt
mod2 <- mirt::mirt(dat, 1, itemtype="gpcm", verbose=TRUE)
#*** ltm
mod3 <- ltm::gpcm( dat, control=list(verbose=TRUE) )
mod3b <- ltm::gpcm( dat, control=list(verbose=TRUE), IRT.param=FALSE)

#-- comparison log likelihood
logLik(mod1)
logLik(mod2)
logLik(mod3)
logLik(mod3b)

#*** intercept parametrization (like in TAM)

# TAM
mod1$B[,2,1]   # slope
mod1$AXsi      # intercepts
# mirt
coef(mod2)
# ltm
coef(mod3b, IRT.param=FALSE)[, c(4,1:3)]

#*** IRT parametrization

# TAM
mod1$AXsi / mod1$B[,2,1]
# mirt
coef(mod2, IRTpars=TRUE)
# ltm
coef(mod3)[, c(4,1:3)]

#############################################################################
# EXAMPLE 26: Differential item functioning in multdimensional models
#############################################################################

data(data.ex08, package="TAM")
formulaA <- ~ item+female+item*female
resp <- data.ex08[["resp"]]
facets <- as.data.frame(data.ex08[["facets"]])

#***  Model 8a: investigate gender DIF in undimensional model
mod8a <- TAM::tam.mml.mfr(resp=resp, facets=facets, formulaA=formulaA)
summary(mod8a)

#*** multidimensional 2PL Model
I <- 10
Q <- array(0, dim=c(I, 3))
Q[cbind(1:I, c(rep(1, 3), rep(2, 3), rep(3, 4)))] <- 1
rownames(Q) <- colnames(resp)
mod3dim2pl <- TAM::tam.mml.2pl(resp=resp, Q=Q, irtmodel="2PL",
                          control=list(snodes=2000))

#*** Combine both approaches
```

```
thisRows <- gsub("-.*", "", colnames(mod8a$resp)) #select item names

#*** uniform DIF (note irtmodel="2PL.groups" & est.slopegroups)
mod3dim2pl_udiff <- TAM::tam.mml.2pl(resp=mod8a$resp, A=mod8a$A, Q=Q[thisRows, ],
                               irtmodel="2PL.groups",
                               est.slopegroups=as.numeric(as.factor(thisRows)),
                               control=list(snodes=2000))

#*** non-uniform DIF (?); different slope parameters for each item administered to each group
mod3dim2pl_nudiff <- TAM::tam.mml.2pl(resp=mod8a$resp, A=mod8a$A, Q=Q[thisRows, ],
                                irtmodel="2PL", control=list(snodes=2000))

#*** check results
print(mod8a$xsi)
print(mod3dim2pl_udiff$xsi)
summary(mod3dim2pl_nudiff)

#*** within item dimensionality (one item loads on two dimensions)
Q2 <- Q
Q2[4,1] <- 1

# uniform DIF (note irtmodel="2PL.groups" & est.slopegroups)
mod3dim2pl_udiff2 <- TAM::tam.mml.2pl(resp=mod8a$resp, A=mod8a$A, Q=Q2[thisRows, ],
                                irtmodel="2PL.groups",
                                est.slopegroups=as.numeric(as.factor(thisRows)),
                                control=list(snodes=2000))
print(mod8a$xsi)
print(mod3dim2pl_udiff2$xsi)
print(mod3dim2pl_udiff2$xsi)

#############################################################################
# EXAMPLE 27: IRT parameterization for generalized partial credit model (GPCM) in TAM
#############################################################################

#--- read item parameters
pars <- as.numeric(miceadds::scan.vec(
"0.19029 1.25309 0.51737 -1.77046 0.94803
  0.19407 1.22680 0.34986 -1.57666 1.29726
  -0.00888 1.07093 0.31662 -1.38755 1.14809
  -0.33810 1.08205 0.48490 -1.56696 0.79547
  -0.18866 0.99587 0.37880 -1.37468 0.81114" ))
pars <- matrix( pars, nrow=5, byrow=TRUE)
beta <- pars[,1]
alpha <- pars[,5]
tau <- pars[,2:4]

#--- data simulation function for GPCM
sim_gpcm_irt_param <- function(alpha, beta, tau, N, mu=0, sigma=1)
{
    theta <- stats::rnorm(N, mean=mu, sd=sigma)
    I <- length(beta)
    K <- ncol(tau)
    dat <- matrix(0, nrow=N, ncol=I)
```

```
    colnames(dat) <- paste0("I",1:I)
    for (ii in 1:I){
        probs <- matrix(0, nrow=N, ncol=K+1)
        for (kk in 1:K){
            probs[,kk+1] <- probs[,kk] + alpha[ii]*( theta - beta[ii] - tau[ii,kk] )
        }
        probs <- exp(probs)
        probs <- probs/rowSums(probs)
        rn <- stats::runif(N)
        cumprobs <- t(apply(probs,1,cumsum))
        for (kk in 1:K){
            dat[,ii] <- dat[,ii] + ( rn > cumprobs[,kk] )
        }
    }
    return(dat)
}

#-- simulate data
N <- 20000     # number of persons
set.seed(98)
dat1 <- sim_gpcm_irt_param(alpha=alpha, beta=beta, tau=tau, N=N, mu=0, sigma=1)
head(dat1)

#* generate design matrix for IRT parameterization
A1 <- TAM::.A.PCM2( resp=dat1)

#- estimate GPCM model
mod1 <- TAM::tam.mml.2pl( resp=dat1, irtmodel="GPCM", A=A1)
summary(mod1)

# compare true and estimated slope estimates (alpha)
cbind( alpha, mod1$B[,2,] )

# compare true and estimated item difficulties (beta)
cbind( beta, mod1$xsi$xsi[1:5] / mod1$B[,2,1] )

# compare true and estimated tau parameters
cbind( tau[,-3], matrix( mod1$xsi$xsi[-c(1:5)], nrow=5, byrow=TRUE ) / mod1$B[,2,1] )

## 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, Q.fixed=NULL, E=NULL,
  gammaslope.des="2PL", gammaslope=NULL, gammaslope.fixed=NULL,
  est.some.slopes=TRUE, gammaslope.max=9.99, 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, max.guess=0.5, skillspace="normal", theta.k=NULL,
  delta.designmatrix=NULL, delta.fixed=NULL, delta.inits=NULL, pweights=NULL,
  item.elim=TRUE, verbose=TRUE, control=list(), Edes=NULL )

## S3 method for class 'tam.mml.3pl'
summary(object,file=NULL,...)

## S3 method for class 'tam.mml.3pl'
print(x,...)
```

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 specified, 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 imposed 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. In case of a single group it is a matrix with three columns for fixing entries in covariance matrix: 1st column: dimension 1, 2nd column: dimension 2, 3rd column: fixed value of covariance/variance. In case of multiple groups, it is a matrix with four columns 1st column: group index (from $1, \ldots, G$, 2nd column: dimension 1, 3rd column: dimension 2, 4th column: fixed value of covariance

`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.fixed`.

`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.

`Q.fixed` Optional $I \times D$ matrix of the same dimensions like `Q`. Non `NA` entries contain values at which item loadings should be fixed to.

`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.max` Value for absolute entries in γ vector

`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 $c_i \sim Beta(a_i, b_i)$. The first column in `gammaslope.prior` is a_i, the second is b_i.

max.guess
: Upper bound for guessing parameters

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.

delta.inits
: Optional initial matrix of δ parameters.

pweights
: Optional vector of person weights.

item.elim
: Optional logical indicating whether an item with has only zero entries should be removed from the analysis. The default is `TRUE`.

verbose
: Logical indicating whether output should be printed during iterations. This argument replaces `control$progress`.

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

Edes
: Compact form of design matrix. This argument is only defined for convenience for models with random starting values to avoid recalculations.

object
: Object of class `tam.mml.3pl`

file
: A file name in which the summary output will be written

x
: Object of class `tam.mml.3pl`

...
: Further arguments to be passed

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

$$\log P(\theta) = D_\delta \delta$$

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
`se.gammaslope`	Standard errors γ item slope parameters
`E`	Used design matrix E
`Edes`	Used design matrix E in compact form
`guess`	Estimated c guessing parameters
`se.guess`	Standard errors c guessing parameters

Note

The implementation of the model builds on pieces work of Anton Formann. See `http://www.antonformann.at/` for more information.

References

Formann, A. K. (1992). Linear logistic latent class analysis for polytomous data. *Journal of the American Statistical Association, 87*, 476-486. doi:10.2307/2290280

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. doi:10.1007/9780387498393_11

Xu, X., & von Davier, M. (2008). *Fitting the structured general diagnostic model to NAEP data*. ETS Research Report ETS RR-08-27. Princeton, ETS. doi:10.1002/j.23338504.2008.tb02113.x

See Also

See also `tam.mml`.

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

`logLik.tam`, `anova.tam`

Examples

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

data(data.sim.rasch)
dat <- data.sim.rasch
# some control arguments
ctl.list <- list(maxiter=100) # increase the number of iterations in applications!

#*** Model 1: Rasch model, normal trait distribution
mod1 <- TAM::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::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::tam.mml.3pl(resp=dat, skillspace="normal", gammaslope.des="2PL",
       control=ctl.list, est.variance=FALSE )
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::tam.mml.3pl(resp=dat, skillspace="normal", gammaslope.des="2PL",
        control=ctl.list, est.guess=est.guess, guess=guess, est.variance=FALSE)
```

```
summary(mod4)

#--- specification in tamaan
tammodel <- "
LAVAAN MODEL:
  F1=~ I1__I40
  F1 ~~ 1*F1
  I3 + I7 ?=g1
  I9 + I12 ?=g912 * g1
    "
mod4a <- TAM::tamaan( tammodel, resp=dat, control=list(maxiter=20))
summary(mod4a)

#*** 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::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)

#--- specification in tamaan
tammodel <- "
LAVAAN MODEL:
  F1=~ I1__I40
  F1 ~~ 1*F1
  I3 + I7 ?=g1
  I9 + I12 ?=g912 * g1
MODEL PRIOR:
  g912 ~ Beta(5,17)
  I3_guess ~ Beta(5,17)
  I7_guess ~ Beta(5,17)
    "
mod5a <- TAM::tamaan( tammodel, resp=dat, control=list(maxiter=20))

#*** Model 6: 2PL model with design matrix for item slopes
I <- 40         # number of items
D <- 1       # dimensions
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::tam.mml.3pl(resp=dat, control=ctl.list,   E=E, est.variance=FALSE  )
summary(mod6)

#*** Model 6b: Truncated normal prior distribution for slope parameters
gammaslope.prior <- matrix( 0, nrow=Ngam, ncol=4 )
gammaslope.prior[,1] <- 2  # mean
```

```
gammaslope.prior[,2] <- 10  # standard deviation
gammaslope.prior[,3] <- -Inf # lower bound
gammaslope.prior[ 4:13,3] <- 1.2
gammaslope.prior[,4] <- Inf  # upper bound
# estimate model
mod6b <- TAM::tam.mml.3pl(resp=dat, E=E, est.variance=FALSE,
                gammaslope.prior=gammaslope.prior, control=ctl.list )
summary(mod6b)

#*** 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::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::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)

#*** Model 9: Multidimensional multiple group model
N <- nrow(dat)
I <- ncol(dat)
group <- c( rep(1,N/4), rep(2,N/4), rep(3,N/2) )
Q <- matrix(0,nrow=I,ncol=2)
Q[ 1:(I/2), 1] <- Q[ seq(I/2+1,I), 2] <- 1
# estimate model
mod9 <- TAM::tam.mml.3pl(resp=dat, skillspace="normal", est.some.slopes=FALSE,
               group=group, Q=Q)
```

```
summary(mod9)

#############################################################################
# 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::tam.mml.3pl(resp=dat, skillspace="normal",
           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::tam.mml.3pl(resp=dat, skillspace="discrete", theta.k=theta.k,
       delta.designmatrix=delta.designmatrix, 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::tam.mml.3pl(resp=dat, gammaslope.des="2PL", Q=Q )
summary(mod3)

#############################################################################
# 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( stats::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::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::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)

#############################################################################
# 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( stats::runif(I,0, .4 ),4)
a2 <- round( stats::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( stats::runif( N1 * I), N1, I ) )
N2 <- round(.7*N)
dat2 <- 1 * ( matrix(a2,N2,I,byrow=TRUE) > matrix( stats::runif( N2 * I), N2, I ) )
dat <- rbind( dat1, dat2 )
colnames(dat) <- paste0("I", 1:I)

#****************************************************
# estimation using tam.mml.3pl function

# 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
                    }

# estimation command
mod1 <- TAM::tam.mml.3pl(resp=dat, E=E, control=list(maxit=20), skillspace="discrete",
          theta.k=theta.k, notA=TRUE)
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

#********************************************************
#** specification with tamaan
tammodel <- "
ANALYSIS:
  TYPE=LCA;
  NCLASSES(2);   # 2 classes
  NSTARTS(5,20); # 5 random starts with 20 iterations
LAVAAN MODEL:
  F=~ I1__I7
    "
mod1b <- TAM::tamaan( tammodel, resp=dat )
summary(mod1b)
# compare with mod1
logLik(mod1)
logLik(mod1b)

#############################################################################
# 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 )
colnames(dat) <- paste0("I",1:I)

#*****************************************************
#*** 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::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 )

#----- specification using tamaan
tammodel <- "
ANALYSIS:
  TYPE=LOCLCA;
  NCLASSES(4)
LAVAAN MODEL:
  F=~ I1__I15
  I8 | 0*t1
    "
mod1b <- TAM::tamaan( tammodel, resp=dat )
summary(mod1b)

#############################################################################
# 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( stats::runif(I, 0, .4 ), 2 )
slip <- round( stats::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::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 )

#############################################################################
# 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( stats::rnorm( wgt*N ), - b1 )
# class 2
dat2 <- sirt::sim.raschtype( stats::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 2 latent classes, then we need
# 7*2=14 dimensions.

#*** Model 1: Rasch model
# theta grid
theta.k1 <- seq( -5, 5, 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 <- stats::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::tam.mml.3pl(resp=dat1, E=E, 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 + stats::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::tam.mml.3pl(resp=dat, E=E, 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
stats::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 ] )

#--- specification in tamaan
```

```
tammodel <- "
ANALYSIS:
  TYPE=MIXTURE;
  NCLASSES(2)
  NSTARTS(5,30)
LAVAAN MODEL:
  F=~ I0001__I0015
ITEM TYPE:
  ALL(Rasch);
    "
mod1b <- TAM::tamaan( tammodel, resp=dat )
summary(mod1b)

#############################################################################
# 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( stats::rnorm( wgt*N ), - b1, fixed.a=a1)
# class 2
dat2 <- sirt::sim.raschtype( stats::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( - stats::qlogis( colMeans( dat ) ), scale=FALSE )
gammaslope <- c( par1, rep(1,I),  scale( par1 + runif(I, - 1.4, 1.4 ),
        scale=FALSE), stats::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::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$gammaslope
# summary
summary(mod1)
# latent class proportions
round( stats::aggregate( mod1$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)

#--- specification in tamaan
tammodel <- "
ANALYSIS:
  TYPE=MIXTURE;
  NCLASSES(2)
  NSTARTS(10,25)
LAVAAN MODEL:
  F=~ I0001__I0020
    "
mod1t <- TAM::tamaan( tammodel, resp=dat )
summary(mod1t)

#############################################################################
# EXAMPLE 9: Toy example: Exact representation of an item by a factor
#############################################################################

data(data.gpcm)
dat <- data.gpcm[,1,drop=FALSE ]   # choose first item
# some descriptives
( t1 <- table(dat) )

# The idea is that we define an IRT model with one latent variable
# which extactly corresponds to the manifest item.

I <- 1    # 1 item
K <- 4    # 4 categories
TP <- 4   # 4 discrete theta points

# define skill space
theta.k <- diag(TP)
# define loading matrix E
E <- array( -99, dim=c(I,K,TP,1 ) )
for (vv in 1:K){
    E[ 1, vv, vv, 1 ] <- 9
                }
# estimate model
mod1 <- TAM::tam.mml.3pl(resp=dat, E=E, skillspace="discrete",
         theta.k=theta.k,  notA=TRUE)
summary(mod1)
# -> the latent distribution corresponds to the manifest distribution, because ...
round( mod1$pi.k, 3 )
round( t1 / sum(t1), 3 )

#############################################################################
# EXAMPLE 10: Some fixed item loadings
#############################################################################

data(data.Students,package="CDM")
dat <- data.Students
# select variables
vars <- scan( nlines=1, what="character")
```

```
    act1 act2 act3 act4 act5 sc1 sc2 sc3 sc4
dat <- data.Students[, vars  ]

# define loading matrix: two-dimensional model
Q <- matrix( 0, nrow=9, ncol=2 )
Q[1:5,1] <- 1
Q[6:9,2] <- 1
# define some fixed item loadings
Q.fixed <- NA*Q
Q.fixed[ c(1,4), 1] <- .5
Q.fixed[ 6:7, 2 ] <- 1

# estimate model
mod3 <- TAM::tam.mml.3pl( resp=dat, gammaslope.des="2PL", Q=Q, Q.fixed=Q.fixed,
            control=list( maxiter=10, nodes=seq(-4,4,len=10) )  )
summary(mod3)

#############################################################################
# EXAMPLE 11: Mixed response formats - Multiple choice and partial credit items
#############################################################################

data(data.timssAusTwn.scored)
dat <- data.timssAusTwn.scored

# select columns with item responses
dat <- dat[, grep("M0", colnames(dat) ) ]
I <- ncol(dat)   # number of items

# The idea is to start with partial credit modelling
# and then to include the guessing parameters

#*** Model 0: Partial Credit Model
mod0 <- TAM::tam.mml(dat)
summary(mod0)

#*** Model 1 and Model 2: include guessing parameters

# multiple choice items
guess_items <- which( apply( dat, 2, max, na.rm=TRUE )==1 )
# define guessing parameters
guess0 <- rep(0,I)
guess0[ guess_items ] <- .25  # set guessing probability to .25

# define which guessing parameters should be estimated
est.guess1 <- rep(0,I)  # all parameters are fixed
est.guess2 <- 1 * ( guess0==.25 )  # joint guessing parameter

# use design matrix from partial credit model
A0 <- mod0$A

#--- Model 1: fixed guessing parameters of .25 and item slopes of 1
mod1 <- TAM::tam.mml.3pl( dat, guess=guess0, est.guess=est.guess1,
            A=A0, est.some.slopes=FALSE, control=list(maxiter=50) )
```

```
summary(mod1)

#--- Model 2: estimate joint guessing parameters and item slopes of 1
mod2 <- TAM::tam.mml.3pl( dat, guess=guess0, est.guess=est.guess2,
            A=A0, est.some.slopes=FALSE, control=list(maxiter=50) )
summary(mod2)

# model comparison
IRT.compareModels(mod0,mod1,mod2)

## End(Not run)
```

---

tam.modelfit *Model Fit Statistics in* **TAM**

Description

The function `tam.modelfit` computes several model fit statistics. It includes the Q3 statistic (Yen, 1984) and an adjusted variant of it (see Details). Effect sizes of model fit (`MADaQ3`, $MADRESIDCOV$, $SRMR$) are also available.

The function `IRT.modelfit` is a wrapper to `tam.modelfit`, but allows convenient model comparisons by using the `CDM::IRT.compareModels` function.

The `tam.modelfit` function can also be used for fitted models outside the **TAM** package by applying `tam.modelfit.IRT` or `tam.modelfit.args`.

The function `tam.Q3` computes the Q_3 statistic based on weighted likelihood estimates (see `tam.wle`).

Usage

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

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

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

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

```
tam.modelfit.IRT( object, progress=TRUE )

tam.modelfit.args( resp, probs, theta, post, progress=TRUE )

tam.Q3(tamobj, ... )

## S3 method for class 'tam.Q3'
summary(object,...)
```

Arguments

tamobj	Object of class tam
progress	An optional logical indicating whether progress should be displayed
object	Object of class tam.modelfit (for summary) or objects for which IRT.data, IRT.irfprob and IRT.posterior have been defined (for tam.modelfit.IRT).
resp	Dataset with item responses
probs	Array with item response functions evaluated at theta
theta	Matrix with used $\boldsymbol{\theta}$ grid
post	Individual posterior distribution
...	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 $Q3_{ij} = Cor(e_{ni}, e_{nj})$. The residuals in tam.modelfit are calculated by integrating values of the individual posterior distribution. Residuals in tam.Q3 are calculated by using weighted likelihood estimates (WLEs) from tam.wle.

It is known that under local independence, the expected value of Q_3 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. An equivalent statistic based on the Q_3 statistic is similar to the standardized generalized dimensionality discrepancy measure (SGDDM; Levy, Xu, Yel & Svetina, 2015).

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 conditional 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
`Q3_summary`	Summary of Q_3 statistics
`N_itempair`	Sample size for each item pair

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. doi:10.1111/j.1745-3984.2012.00185.x

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

Levy, R., Xu, Y., Yel, N., & Svetina, D. (2015). A standardized generalized dimensionality discrepancy measure and a standardized model-based covariance for dimensionality assessment for multidimensional models. *Journal of Educational Measurement, 52*(2), 144–158. doi:10.1111/jedm.12070

Maydeu-Olivares, A. (2013). Goodness-of-fit assessment of item response theory models (with discussion). *Measurement: Interdisciplinary Research and Perspectives, 11*, 71-137. doi:10.1080/15366367.2013.831680

McDonald, R. P., & Mok, M. M.-C. (1995). Goodness of fit in item response models. *Multivariate Behavioral Research, 30*, 23-40. doi:10.1207/s15327906mbr3001_2

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. doi:10.1177/014662168400800201

Examples

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

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

#*****************************************************
#*** Model 1: Rasch model
mod1 <- TAM::tam.mml( dat )
# assess model fit
res1 <- TAM::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:
# IRT.modelfit
fmod1 <- IRT.modelfit(mod1)
summary(fmod1)

#*****************************************************
#*** Model 2: 2PL model
mod2 <- TAM::tam.mml.2pl( dat )
# IRT.modelfit
fmod2 <- IRT.modelfit(mod2)
summary(fmod2)

# model comparison
IRT.compareModels(fmod1, fmod2 )

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

#############################################################################
# EXAMPLE 3: Model fit data.gpcm | Partial credit model
#############################################################################
```

```
data(data.gpcm)
dat <- data.gpcm

# estimate partial credit model
mod1 <- TAM::tam.mml( dat)
summary(mod1)

# assess model fit
tmod1 <- TAM::tam.modelfit( mod1 )
summary(tmod1)

#############################################################################
# EXAMPLE 4: data.read | Comparison Q3 statistic
#############################################################################

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

#**** Model 1: 1PL model
mod1 <- TAM::tam.mml( dat )
summary(mod1)

#**** Model 2: 2PL model
mod2 <- TAM::tam.mml.2pl( dat )
summary(mod2)

#**** assess model fits
# Q3 based on posterior
fmod1 <- TAM::tam.modelfit(mod1)
fmod2 <- TAM::tam.modelfit(mod2)
# Q3 based on WLEs
q3_mod1 <- TAM::tam.Q3(mod1)
q3_mod2 <- TAM::tam.Q3(mod2)
summary(fmod1)
summary(fmod2)
summary(q3_mod1)
summary(q3_mod2)

## End(Not run)
```

---

tam.np *Unidimensional Non- and Semiparametric Item Response Model*

Description

Conducts non- and semiparametric estimation of a unidimensional item response model for a single group allowing polytomous item responses (Rossi, Wang & Ramsay, 2002).

For dichotomous data, the function also allows group lasso penalty (`penalty_type="lasso"`; Breheny & Huang, 2015; Yang & Zhou, 2015) and a ridge penalty (`penalty_type="ridge"`; Rossi et al., 2002) which is applied to the nonlinear part of the basis expansion. This approach automatically detects deviations from a 2PL or a 1PL model (see Examples 2 and 3). See Details for model specification.

Usage

```
tam.np( dat, probs_init=NULL, pweights=NULL, lambda=NULL, control=list(),
    model="2PL", n_basis=0, basis_type="hermite", penalty_type="lasso",
    pars_init=NULL, orthonormalize=TRUE)

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

## S3 method for class 'tam.np'
IRT.cv(object, kfold=10, ...)
```

Arguments

`dat`	Matrix of integer item responses (starting from zero)
`probs_init`	Array containing initial probabilities
`pweights`	Optional vector of person weights
`lambda`	Numeric or vector of regularization parameter
`control`	List of control arguments, see `tam.mml`.
`model`	Specified target model. Can be `"2PL"` or `"1PL"`.
`n_basis`	Number of basis functions
`basis_type`	Type of basis function: `"bspline"` for B-splines or `"hermite"` for Gauss-Hermite polynomials
`penalty_type`	Lasso type penalty (`"lasso"`) or ridge penalty (`"ridge"`)
`pars_init`	Optional matrix of initial item parameters
`orthonormalize`	Logical indicating whether basis functions should be orthonormalized
`object`	Object of class `tam.np`
`file`	Optional file name for summary output
`kfold`	Number of folds in k-fold cross-validation
`...`	Further arguments to be passed

Details

The basis expansion approach is applied for the logit transformation of item response functions for dichotomous data. In more detail, it this assumed that

$$P(X_i = 1|\theta) = \psi(H_0(\theta) + H_1(\theta))$$

where H_0 is the target function type and H_1 is the semiparametric part which parameterizes model deviations. For the 2PL model (`model="2PL"`) it is $H_0(\theta) = d_i + a_i\theta$ and for the 1PL model

(`model="1PL"`) we set $H_1(\theta) = d_i + 1 \cdot \theta$. The model discrepancy is specified as a basis expansion approach

$$H_1(\theta) = \sum_{h=1}^{p} \beta_{ih} f_h(\theta)$$

where f_h are basis functions (possibly orthonormalized) and β_{ih} are item parameters which should be estimated. Penalty functions are posed on the β_{ih} coefficients. For the group lasso penalty, we specify the penalty $J_{i,L1} = N\lambda\sqrt{p}\sqrt{\sum_{h=1}^{p}\beta_{ih}^2}$ while for the ridge penalty it is $J_{i,L2} = N\lambda\sum_{h=1}^{p}\beta_{ih}^2$ (N denoting the sample size).

Value

List containing several entries

`rprobs`	Item response probabilities
`theta`	Used nodes for approximation of θ distribution
`n.ik`	Expected counts
`like`	Individual likelihood
`hwt`	Individual posterior
`item`	Summary item parameter table
`pars`	Estimated parameters
`regularized`	Logical indicating which items are regularized
`ic`	List containing
`...`	Further values

References

Breheny, P., & Huang, J. (2015). Group descent algorithms for nonconvex penalized linear and logistic regression models with grouped predictors. *Statistics and Computing, 25*(2), 173-187. doi:10.1007/s1122201394242

Rossi, N., Wang, X., & Ramsay, J. O. (2002). Nonparametric item response function estimates with the EM algorithm. *Journal of Educational and Behavioral Statistics, 27*(3), 291-317. doi:10.3102/10769986027003291

Yang, Y., & Zou, H. (2015). A fast unified algorithm for solving group-lasso penalized learning problems. *Statistics and Computing, 25*(6), 1129-1141. doi:10.1007/s1122201494985

See Also

Nonparametric item response models can also be estimated with the `mirt::itemGAM` function in the **mirt** package and the `KernSmoothIRT::ksIRT` in the **KernSmoothIRT** package.

See `tam.mml` and `tam.mml.2pl` for parametric item response models.

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: Nonparametric estimation polytomous data
#############################################################################

data(data.cqc02, package="TAM")
dat <- data.cqc02

#** nonparametric estimation
mod <- TAM::tam.np(dat)

#** extractor functions for objects of class 'tam.np'
lmod <- IRT.likelihood(mod)
pmod <- IRT.posterior(mod)
rmod <- IRT.irfprob(mod)
emod <- IRT.expectedCounts(mod)

#############################################################################
# EXAMPLE 2: Semiparametric estimation and detection of item misfit
#############################################################################

#- simulate data with two misfitting items
set.seed(998)
I <- 10
N <- 1000
a <- stats::rnorm(I, mean=1, sd=.3)
b <- stats::rnorm(I, mean=0, sd=1)
dat <- matrix(NA, nrow=N, ncol=I)
colnames(dat) <- paste0("I",1:I)
theta <- stats::rnorm(N)
for (ii in 1:I){
    dat[,ii] <- 1*(stats::runif(N) < stats::plogis( a[ii]*(theta-b[ii] ) ))
}

#* first misfitting item with lower and upper asymptote
ii <- 1
l <- .3
u <- 1
b[ii] <- 1.5
dat[,ii] <- 1*(stats::runif(N) < l + (u-l)*stats::plogis( a[ii]*(theta-b[ii] ) ))

#* second misfitting item with non-monotonic item response function
ii <- 3
dat[,ii] <- (stats::runif(N) < stats::plogis( theta-b[ii]+.6*theta^2))

#- 2PL model
mod0 <- TAM::tam.mml.2pl(dat)

#- lasso penalty with lambda of .05
mod1 <- TAM::tam.np(dat, n_basis=4, lambda=.05)
```

```
#- lambda value of .03 using starting value of previous model
mod2 <- TAM::tam.np(dat, n_basis=4, lambda=.03, pars_init=mod1$pars)
cmod2 <- TAM::IRT.cv(mod2)  # cross-validated deviance

#- lambda=.015
mod3 <- TAM::tam.np(dat, n_basis=4, lambda=.015, pars_init=mod2$pars)
cmod3 <- TAM::IRT.cv(mod3)

#- lambda=.007
mod4 <- TAM::tam.np(dat, n_basis=4, lambda=.007, pars_init=mod3$pars)

#- lambda=.001
mod5 <- TAM::tam.np(dat, n_basis=4, lambda=.001, pars_init=mod4$pars)

#- final estimation using solution of mod3
eps <- .0001
lambda_final <- eps+(1-eps)*mod3$regularized   # lambda parameter for final estimate
mod3b <- TAM::tam.np(dat, n_basis=4, lambda=lambda_final, pars_init=mod3$pars)
summary(mod1)
summary(mod2)
summary(mod3)
summary(mod3b)
summary(mod4)

# compare models with respect to information criteria
IRT.compareModels(mod0, mod1, mod2, mod3, mod3b, mod4, mod5)

#-- compute item fit statistics RISE
# regularized solution
TAM::IRT.RISE(mod_p=mod1, mod_np=mod3)
# regularized solution, final estimation
TAM::IRT.RISE(mod_p=mod1, mod_np=mod3b, use_probs=TRUE)
TAM::IRT.RISE(mod_p=mod1, mod_np=mod3b, use_probs=FALSE)
# use TAM::IRT.RISE() function for computing the RMSD statistic
TAM::IRT.RISE(mod_p=mod1, mod_np=mod1, use_probs=FALSE)

#############################################################################
# EXAMPLE 3: Mixed 1PL/2PL model
#############################################################################

#* simulate data with 2 2PL items and 8 1PL items
set.seed(9877)
N <- 2000
I <- 10
b <- seq(-1,1,len=I)
a <- rep(1,I)
a[c(3,8)] <- c(.5, 2)
theta <- stats::rnorm(N, sd=1)
dat <- sirt::sim.raschtype(theta, b=b, fixed.a=a)

#- 1PL model
mod1 <- TAM::tam.mml(dat)
#- 2PL model
```

```
mod2 <- TAM::tam.mml.2pl(dat)
#- 2PL model with penalty on slopes
mod3 <- TAM::tam.np(dat, lambda=.04, model="1PL", n_basis=0)
summary(mod3)
#- final mixed 1PL/2PL model
lambda <- 1*mod3$regularized
mod4 <- TAM::tam.np(dat, lambda=lambda, model="1PL", n_basis=0)
summary(mod4)

IRT.compareModels(mod1, mod2, mod3, mod4)

## End(Not run)
```

`tam.personfit` *Person Outfit and Infit Statistics*

Description

Computes person outfit and infit statistics.

Usage

```
tam.personfit(tamobj)
```

Arguments

`tamobj`	Fitted object in **TAM**

Value

Data frame containing person outfit and infit statistics

See Also

See `tam.fit` and `msq.itemfit` for item fit statistics.

Examples

```
#############################################################################
# EXAMPLE 1: Person fit dichotomous data
#############################################################################

data(data.sim.rasch, package="TAM")
resp <- data.sim.rasch

#*** estimate Rasch model
mod1 <- TAM::tam.mml(resp=resp)
summary(mod1)

#*** compute person fit statistics
```

```
fmod1 <- TAM::tam.personfit(mod1)
head(fmod1)
```

---

tam.pv *Plausible Value Imputation*

Description

Plausible value imputation for objects of the classes tam and tam.mml (Adams & Wu, 2007). For converting generated plausible values into a list of multiply imputed datasets see tampv2datalist and the Examples 2 and 3 of this function.

The function tam.pv.mcmc employs fully Bayesian estimation for drawing plausible values and is recommended in cases when the latent regression model is unreliably estimated (multidimensional model with stochastic nodes). The parameters of the latent regression (regression coefficients and residual covariance matrices) are drawn by Bayesian bootstrap (Rubin, 1981). Either case probabilities (i.e., non-integer weights for cases in resampling; argument sample_integers=FALSE) or ordinary bootstrap (i.e., sampling cases with replacement; argument sample_integers=TRUE) can be used for the Bootstrap step by obtaining posterior draws of regression parameters.

Usage

```
tam.pv(tamobj, nplausible=10, ntheta=2000, normal.approx=FALSE,
    samp.regr=FALSE, theta.model=FALSE, np.adj=8, na.grid=5, verbose=TRUE)

tam.pv.mcmc( tamobj, Y=NULL, group=NULL, beta_groups=TRUE, nplausible=10, level=.95,
    n.iter=1000, n.burnin=500, adj_MH=.5, adj_change_MH=.05, refresh_MH=50,
    accrate_bound_MH=c(.45, .55), sample_integers=FALSE, theta_init=NULL,
    print_iter=20, verbose=TRUE, calc_ic=TRUE)

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

## S3 method for class 'tam.pv.mcmc'
plot(x, ...)
```

Arguments

tamobj	Object of class tam or tam.mml. For tam.pv.mcmc, it must not be an object of this class but rather a list with (at least the) entries AXsi, B, resp.
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. The normal approximation is implemented for unidimensional and multidimensional 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. Sampled regression coefficients are obtained by nonparametric bootstrap.

theta.model
: Logical indicating whether the theta grid from the tamobj object should be used for plausible value imputation. In case of normal.approx=TRUE, this should be sufficient in many applications.

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}.

na.grid
: Range of the grid in normal approximation. Default is from -5 to 5.

...
: Further arguments to be passed

Y
: Optional matrix of regressors

group
: Optional vector of group identifiers

beta_groups
: Logical indicating whether group specific beta coefficients shall be estimated.

level
: Confidence level

n.iter
: Number of iterations

n.burnin
: Number of burnin-iterations

adj_MH
: Adjustment factor for Metropolis-Hastings (MH) steps which controls the variance of the proposal distribution for θ. Can be also a vector of length equal to the number of persons.

adj_change_MH
: Allowed change for MH adjustment factor after refreshing

refresh_MH
: Number of iterations after which the variance of the proposal distribution should be updated

accrate_bound_MH
: Bounds for target acceptance rates of sampled θ values.

sample_integers
: Logical indicating whether weights for complete cases should be sampled in bootstrap

theta_init
: Optional matrix with initial $\boldsymbol{\theta}$ values

print_iter
: Print iteration progress every print_iterth iteration

verbose
: Logical indicating whether iteration progress should be displayed.

calc_ic
: Logical indicating whether information criteria should be computed.

object
: Object of class tam.pv.mcmc

x
: Object of class tam.pv.mcmc

file
: A file name in which the summary output will be written

Value

The value of `tam.pv` is a list with following entries:

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

The value of `tam.pv.mcmc` is a list containing entries

- `pv` — Data frame containing plausible values
- `parameter_samples` — Sampled regression parameters
- `ic` — Information criteria
- `beta` — Estimate of regression parameters β
- `variance` — Estimate of residual variance matrix Σ
- `correlation` — Estimate of residual correlation matrix corresponding to `variance`
- `theta_acceptance_MH` — Acceptance rates and acceptance MH factors for each individual
- `theta_last` — Last sampled θ value
- `...` — Further values

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. doi:10.1007/9780387498393_4

Rubin, D. B. (1981). The Bayesian bootstrap. *The Annals of Statistics, 9*(1), 130-134.

See Also

See `tam.latreg` for further examples of fitting latent regression models and drawing plausible values from models which provides an individual likelihood as an input.

Examples

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

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

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

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

# draw 5 plausible values with a normality
# assumption of the posterior and 500 ability nodes
pv1b <- TAM::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:
# draw plausible values with tam.pv.mcmc function
Y <- matrix(1, nrow=500, ncol=1)
pv2 <- TAM::tam.pv.mcmc( tamobj=mod, Y=Y, nplausible=5 )
str(pv2)

# summary output
summary(pv2)
# assessing convergence with traceplots
plot(pv2, ask=TRUE)

# use starting values for theta and MH factors which fulfill acceptance rates
# from previously fitted model
pv3 <- TAM::tam.pv.mcmc( tamobj=mod, Y=Y, nplausible=5, theta_init=pv2$theta_last,
            adj_MH=pv2$theta_acceptance_MH$adj_MH )

#############################################################################
# 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::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$B
# matrix of latent regressors female, hisei and migra
Y <- dat[, c("female", "hisei", "migra") ]
mod2b <- TAM::tam.mml( resp=dat[,items], B=mod2a$B, xsi.fixed=xsi.fixed, Y=Y,
            pid=dat$idstud)

# plausible value imputation with normality assumption
# and ignoring uncertainty about regression coefficients
#    -> the default is samp.regr=FALSE
pv2c <- TAM::tam.pv( mod2b, nplausible=10, ntheta=500, normal.approx=TRUE )
# sampling of regression coefficients
pv2d <- TAM::tam.pv( mod2b, nplausible=10, ntheta=500, samp.regr=TRUE)
# sampling of regression coefficients, normal approximation using the
# theta grid from the model
pv2e <- TAM::tam.pv( mod2b, samp.regr=TRUE, theta.model=TRUE, normal.approx=TRUE)

#--- create list of multiply imputed datasets with plausible values
# define dataset with covariates to be matched
Y <- dat[, c("idstud", "idschool", "female", "hisei", "migra") ]

# define plausible value names
pvnames <- c("PVREAD")
# create list of imputed datasets
datlist1 <- TAM::tampv2datalist( pv2e, pvnames=pvnames, Y=Y, Y.pid="idstud")
str(datlist1)

# create a matrix of covariates with different set of students than in pv2e
Y1 <- Y[ seq( 1, 600, 2 ), ]
# create list of multiply imputed datasets
datlist2 <- TAM::tampv2datalist( pv2e, pvnames=c("PVREAD"), Y=Y1, Y.pid="idstud")

#--- fit some models in lavaan and semTools
library(lavaan)
library(semTools)

#*** Model 1: Linear regression
lavmodel <- "
   PVREAD ~ migra + hisei
   PVREAD ~~ PVREAD
        "
mod1 <- semTools::lavaan.mi( lavmodel, data=datlist1, m=0)
summary(mod1, standardized=TRUE, rsquare=TRUE)

# apply lavaan for third imputed dataset
mod1a <- lavaan::lavaan( lavmodel, data=datlist1[[3]] )
summary(mod1a, standardized=TRUE, rsquare=TRUE)

# compare with mod1 by looping over all datasets
mod1b <- lapply( datlist1, FUN=function(dat0){
    mod1a <- lavaan( lavmodel, data=dat0 )
    coef( mod1a)
        } )
mod1b
```

```
mod1b <- matrix( unlist( mod1b ), ncol=length( coef(mod1)), byrow=TRUE )
mod1b
round( colMeans(mod1b), 3 )
coef(mod1)   # -> results coincide

#*** Model 2: Path model
lavmodel <- "
   PVREAD ~ migra + hisei
   hisei ~ migra
   PVREAD ~~ PVREAD
   hisei ~~ hisei
        "
mod2 <- semTools::lavaan.mi( lavmodel, data=datlist1 )
summary(mod2, standardized=TRUE, rsquare=TRUE)
# fit statistics
inspect( mod2, what="fit")

#--- using mitools package
library(mitools)
# convert datalist into an object of class imputationList
datlist1a <- mitools::imputationList( datlist1 )
# fit linear regression
mod1c <- with( datlist1a, stats::lm( PVREAD ~ migra + hisei ) )
summary( mitools::MIcombine(mod1c) )

#--- using mice package
library(mice)
library(miceadds)
# convert datalist into a mids object
mids1 <- miceadds::datalist2mids( datlist1 )
# fit linear regression
mod1c <- with( mids1, stats::lm( PVREAD ~ migra + hisei ) )
summary( mice::pool(mod1c) )

#############################################################################
# EXAMPLE 3: Multidimensional plausible value imputation
#############################################################################

# (1) simulate some data
set.seed(6778)
library(mvtnorm)
N <- 1000
Y <- cbind( stats::rnorm(N), stats::rnorm(N) )
theta <- mvtnorm::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 <- stats::plogis( outer( theta[,1], seq( -2, 2, len=I ), "-" ) )
resp1 <- 1 * ( p1 > matrix( stats::runif( N*I ), nrow=N, ncol=I ) )
p1 <- stats::plogis( outer( theta[,2], seq( -2, 2, len=I ), "-" ) )
resp2 <- 1 * ( p1 > matrix( stats::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) fit latent regression model
mod <- TAM::tam.mml( resp=resp, Y=Y, Q=Q )

# (4) draw plausible values
pv1 <- TAM::tam.pv( mod, theta.model=TRUE )

# (5) convert plausible values to list of imputed datasets
Y1 <- data.frame(Y)
colnames(Y1) <- paste0("Y",1:2)
pvnames <- c("PVFA","PVFB")
# create list of imputed datasets
datlist1 <- TAM::tampv2datalist( pv1, pvnames=pvnames, Y=Y1 )
str(datlist1)

# (6) apply statistical models
library(semTools)
# define linear regression
lavmodel <- "
   PVFA ~ Y1 + Y2
   PVFA ~~ PVFA
        "
mod1 <- semTools::lavaan.mi( lavmodel, data=datlist1 )
summary(mod1, standardized=TRUE, rsquare=TRUE)

# (7) draw plausible values with tam.pv.mcmc function
Y1 <- cbind( 1, Y )
pv2 <- TAM::tam.pv.mcmc( tamobj=mod, Y=Y1, n.iter=1000, n.burnin=200 )

# (8) group-specific plausible values
set.seed(908)
# create artificial grouping variable
group <- sample( 1:3, N, replace=TRUE )
pv3 <- TAM::tam.pv.mcmc( tamobj, Y=Y1, n.iter=1000, n.burnin=200, group=group )

# (9) plausible values with no fitted object in TAM

# fit IRT model without covariates
mod4a <- TAM::tam.mml( resp=resp, Q=Q )
# define input for tam.pv.mcmc
tamobj1 <- list( AXsi=mod4a$AXsi, B=mod4a$B, resp=mod4a$resp )
pmod4 <- TAM::tam.pv.mcmc( tamobj1, Y=Y1 )

#############################################################################
# EXAMPLE 4: Plausible value imputation with measurement errors in covariates
#############################################################################

library(sirt)
set.seed(7756)
```

```
N <- 2000    # number of persons
I <- 10     # number of items

# simulate covariates
X <- mvrnorm( N, mu=c(0,0), Sigma=matrix( c(1,.5,.5,1),2,2 ) )
colnames(X) <- paste0("X",1:2)
# second covariate with measurement error with variance var.err
var.err <- .3
X.err <- X
X.err[,2] <-X[,2] + rnorm(N, sd=sqrt(var.err) )
# simulate theta
theta <- .5*X[,1] + .4*X[,2] + rnorm( N, sd=.5 )
# simulate item responses
itemdiff <- seq( -2, 2, length=I)  # item difficulties
dat <- sirt::sim.raschtype( theta, b=itemdiff )

#***********************
#*** Model 0: Regression model with true variables
mod0 <- stats::lm( theta ~ X )
summary(mod0)

#***********************
#*** Model 1: latent regression model with true covariates X
xsi.fixed <- cbind( 1:I, itemdiff )
mod1 <- TAM::tam.mml( dat, xsi.fixed=xsi.fixed, Y=X)
summary(mod1)

# draw plausible values
res1a <- TAM::tam.pv( mod1, normal.approx=TRUE, ntheta=200, samp.regr=TRUE)
# create list of multiply imputed datasets
library(miceadds)
datlist1a <- TAM::tampv2datalist( res1a, Y=X )
imp1a <- miceadds::datalist2mids( datlist1a )

# fit linear model
# linear regression with measurement errors in X
lavmodel <- "
   PV.Dim1 ~ X1 + X2true
   X2true=~ 1*X2
   X2 ~~ 0.3*X2  #=var.err
   PV.Dim1 ~~ PV.Dim1
   X2true ~~ X2true
        "
mod1a <- semTools::lavaan.mi( lavmodel, datlist1a)
summary(mod1a, standardized=TRUE, rsquare=TRUE)

#***********************
#*** Model 2: latent regression model with error-prone covariates X.err
mod2 <- TAM::tam.mml( dat, xsi.fixed=xsi.fixed, Y=X.err)
summary(mod2)

#***********************
#*** Model 3: Adjustment of covariates
```

```
cov.X.err <- cov( X.err )
# matrix of variance of measurement errors
measerr <- diag( c(0,var.err) )
# true covariance matrix
cov.X <- cov.X.err - measerr
# mean of X.err
mu <- colMeans(X.err)
muM <- matrix( mu, nrow=nrow(X.err), ncol=ncol(X.err), byrow=TRUE)
# reliability matrix
W <- solve( cov.X.err ) %*% cov.X
ident <- diag(2)
# adjusted scores of X
X.adj <- ( X.err - muM ) %*% W   + muM %*% ( ident - W )

# fit latent regression model
mod3 <- TAM::tam.mml( dat, xsi.fixed=xsi.fixed, Y=X.adj)
summary(mod3)

# draw plausible values
res3a <- TAM::tam.pv( mod3, normal.approx=TRUE, ntheta=200, samp.regr=TRUE)

# create list of multiply imputed datasets
library(semTools)

#*** PV dataset 1
# datalist with error-prone covariates
datlist3a <- TAM::tampv2datalist( res3a, Y=X.err )
# datalist with adjusted covariates
datlist3b <- TAM::tampv2datalist( res3a, Y=X.adj )

# linear regression with measurement errors in X
lavmodel <- "
   PV.Dim1 ~ X1 + X2true
   X2true=~ 1*X2
   X2 ~~ 0.3*X2  #=var.err
   PV.Dim1 ~~ PV.Dim1
   X2true ~~ X2true
        "
mod3a <- semTools::lavaan.mi( lavmodel, datlist3a)
summary(mod3a, standardized=TRUE, rsquare=TRUE)

lavmodel <- "
   PV.Dim1 ~ X1 + X2
   PV.Dim1 ~~ PV.Dim1
        "
mod3b <- semTools::lavaan.mi( lavmodel, datlist3b)
summary(mod3b, standardized=TRUE, rsquare=TRUE)
#=> mod3b leads to the correct estimate.

#**********************************************
# plausible value imputation for abilities and error-prone
# covariates using the mice package
```

```
library(mice)
library(miceadds)

# creating the likelihood for plausible value for abilities
mod11 <- TAM::tam.mml( dat, xsi.fixed=xsi.fixed )
likePV <- IRT.likelihood(mod11)
# creating the likelihood for error-prone covariate X2
lavmodel <- "
  X2true=~ 1*X2
  X2 ~~ 0.3*X2
    "
mod12 <- lavaan::cfa( lavmodel, data=as.data.frame(X.err) )
summary(mod12)
likeX2 <- TAM::IRTLikelihood.cfa( data=X.err, cfaobj=mod12)
str(likeX2)

#-- create data input for mice package
data <- data.frame( "PVA"=NA, "X1"=X[,1], "X2"=NA  )
vars <- colnames(data)
V <- length(vars)
predictorMatrix <- 1 - diag(V)
rownames(predictorMatrix) <- colnames(predictorMatrix) <- vars
imputationMethod <- rep("norm", V )
names(imputationMethod) <- vars
imputationMethod[c("PVA","X2")] <- "plausible.values"

#-- create argument lists for plausible value imputation
# likelihood and theta grid of plausible value derived from IRT model
like <- list( "PVA"=likePV, "X2"=likeX2 )
theta <- list( "PVA"=attr(likePV,"theta"), "X2"=attr(likeX2, "theta") )
#-- initial imputations
data.init <- data
data.init$PVA <- mod11$person$EAP
data.init$X2 <- X.err[,"X2"]

#-- imputation using the mice and miceadds package
imp1 <- mice::mice( as.matrix(data), predictorMatrix=predictorMatrix, m=4, maxit=6,
             method=imputationMethod,  allow.na=TRUE,
             theta=theta, like=like, data.init=data.init )
summary(imp1)

# compute linear regression
mod4a <- with( imp1, stats::lm( PVA ~ X1 + X2 ) )
summary( mice::pool(mod4a) )

## 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, item_pars=TRUE, ...)

tam_mml_se_quick(tamobj, numdiff.parm=0.001, item_pars=TRUE )

tam_latreg_se_quick(tamobj, numdiff.parm=0.001 )
```

Arguments

`tamobj`	An object generated by `tam.mml`
`item_pars`	Logical indicating whether standard errors should also be computed for item parameters
`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, data.sim.rasch
#############################################################################

data(data.sim.rasch)
# estimate Rasch model
mod1 <- TAM::tam.mml(resp=data.sim.rasch[1:500,1:10])
# standard error estimation
se1 <- TAM::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::tam.mml.mfr( resp=resp, facets=facets, formulaA=formulaA )
summary(mod)
# estimate standard errors
semod <- TAM::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.

See Also

See the **WrightMap** package and Example 3 for creating Wright maps with fitted models in **TAM**, see wrightMap.

Examples

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

# Model 1: partial credit model
mod1 <- TAM::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::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::tam.mml( dat, Q=Q )
# compute thresholds (specify a probability level of .625)
tmod1 <- TAM::tam.threshold( mod1, prob.lvl=.625 )

#############################################################################
# EXAMPLE 3: Creating Wright maps with the WrightMap package
#############################################################################

library(WrightMap)
# For conducting Wright maps in combination with TAM, see
# http://wrightmap.org/post/100850738072/using-wrightmap-with-the-tam-package
data(sim.rasch)
dat <- sim.rasch

# estimate Rasch model in TAM
mod1 <- TAM::tam.mml(dat)
summary(mod1)

#--- A: creating a Wright map with WLEs

# compute WLE
wlemod1 <- TAM::tam.wle(mod1)$theta
# extract thresholds
tmod1 <- TAM::tam.threshold(mod1)
# create Wright map
WrightMap::wrightMap( thetas=wlemod1, thresholds=tmod1, label.items.srt=-90)

#--- B: creating a Wright Map with population distribution

# extract ability distribution and replicate observations
uni.proficiency <- rep( mod1$theta[,1], round( mod1$pi.k * mod1$ic$n) )
# draw WrightMap
WrightMap::wrightMap( thetas=uni.proficiency, thresholds=tmod1, label.items.rows=3)

## 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=.3, Msteps=20,
       convM=.0001, progress=TRUE,    output.prob=FALSE )

tam.mml.wle2(tamobj, score.resp=NULL, WLE=TRUE, adj=0.3, Msteps=20, convM=1e-04,
        progress=TRUE, output.prob=FALSE, pid=NULL, theta_init=NULL )

tam_jml_wle(tamobj, resp, resp.ind, A, B, nstud, nitems, maxK, convM,
   PersonScores, theta, xsi, Msteps, WLE=FALSE, theta.fixed=NULL, progress=FALSE,
    output.prob=TRUE, damp=0, version=2)

## S3 method for class 'tam.wle'
summary(object, file=NULL, digits=3, ...)

## S3 method for class 'tam.wle'
print(x, digits=3, ...)
```

Arguments

`tamobj`	An object generated by `tam.mml` or `tam.jml`. The object can also be a list containing (at least the) inputs `AXsi`, `B` and `resp` and therefore allows WLE estimation without fitting models in **TAM**.
`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 MLE estimation for extreme scores (i.e. all or none items were correctly solved). This argument is not used if `WLE=TRUE`.
`Msteps`	Maximum number of iterations
`convM`	Convergence criterion
`progress`	Logical indicating whether progress should be displayed.
`output.prob`	Logical indicating whether evaluated probabilities should be included in the list of outputs.

pid	Optional vector of person identifiers
theta_init	Initial theta values
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 θ. The first column includes the index whereas the second column includes the fixed value.
damp	Numeric value between 0 and 1 indicating amount of dampening increments in θ estimates during iterations
version	Integer with possible values 2 or 3. In case of missing item responses, version=3 will typically be more efficient.
...	Further arguments to be passed
object	Object of class tam.wle
x	Object of class tam.wle
file	Optional file name in which the object summary should be written.
digits	Number of digits for rounding

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

Penfield, R. D., & Bergeron, J. M. (2005). Applying a weighted maximum likelihood latent trait estimator to the generalized partial credit model. *Applied Psychological Measurement, 29*, 218-233.

Warm, T. A. (1989). Weighted likelihood estimation of ability in item response theory. *Psychometrika, 54*, 427-450. doi:10.1007/BF02294627

See Also

See the PP::PP_gpcm function in the **PP** package for more person parameter estimators for the partial credit model (Penfield & Bergeron, 2005).

See the S3 method IRT.factor.scores.tam.

Examples

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

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

print(wle1)
summary(wle1)

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

#--- WLE estimation without using a TAM object

#* create an input list
input <- list( resp=data.sim.rasch, AXsi=mod1$AXsi, B=mod1$B )
#* estimation
wle2b <- TAM::tam.mml.wle2( input )

## Not run:
#############################################################################
# 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::tam.mml( resp=resp, Q=Q, control=list(snodes=1000,maxiter=50) )
summary(mod)

# WLE estimates
wmod <- TAM::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.

#############################################################################
# EXAMPLE 3: Partial credit model | Comparison WLEs with PP package
#############################################################################

library(PP)
data(data.gpcm)
dat <- data.gpcm
I <- ncol(dat)

#****************************************
#*** Model 1: Partial Credit Model
```

```
# estimation in TAM
mod1 <- TAM::tam.mml( dat )
summary(mod1)

#-- WLE estimation in TAM
tamw1 <- TAM::tam.wle( mod1 )

#-- WLE estimation with PP package
# convert AXsi parameters into thres parameters for PP
AXsi0 <- - mod1$AXsi[,-1]
b <- AXsi0
K <- ncol(AXsi0)
for (cc in 2:K){
    b[,cc] <- AXsi0[,cc] - AXsi0[,cc-1]
}
# WLE estimation in PP
ppw1 <- PP::PP_gpcm( respm=as.matrix(dat),  thres=t(b), slopes=rep(1,I) )

#-- compare results
dfr <- cbind( tamw1[, c("theta","error") ], ppw1$resPP)
head( round(dfr,3))
  ##      theta error resPP.estimate resPP.SE nsteps
  ##   1 -1.006 0.973         -1.006    0.973      8
  ##   2 -0.122 0.904         -0.122    0.904      8
  ##   3  0.640 0.836          0.640    0.836      8
  ##   4  0.640 0.836          0.640    0.836      8
  ##   5  0.640 0.836          0.640    0.836      8
  ##   6 -1.941 1.106         -1.941    1.106      8
plot( dfr$resPP.estimate, dfr$theta, pch=16, xlab="PP", ylab="TAM")
lines( c(-10,10), c(-10,10) )

#*****************************************
#*** Model 2: Generalized partial Credit Model

# estimation in TAM
mod2 <- TAM::tam.mml.2pl( dat, irtmodel="GPCM" )
summary(mod2)

#-- WLE estimation in TAM
tamw2 <- TAM::tam.wle( mod2 )

#-- WLE estimation in PP
# convert AXsi parameters into thres and slopes parameters for PP
AXsi0 <- - mod2$AXsi[,-1]
slopes <- mod2$B[,2,1]
K <- ncol(AXsi0)
slopesM <- matrix( slopes, I, ncol=K )
AXsi0 <- AXsi0 / slopesM
b <- AXsi0
for (cc in 2:K){
    b[,cc] <- AXsi0[,cc] - AXsi0[,cc-1]
}
```

```
# estimation in PP
ppw2 <- PP::PP_gpcm( respm=as.matrix(dat),  thres=t(b), slopes=slopes )

#-- compare results
dfr <- cbind( tamw2[, c("theta","error") ], ppw2$resPP)
head( round(dfr,3))
  ##      theta error resPP.estimate resPP.SE nsteps
  ##   1 -0.476 0.971         -0.476    0.971     13
  ##   2 -0.090 0.973         -0.090    0.973     13
  ##   3  0.311 0.960          0.311    0.960     13
  ##   4  0.311 0.960          0.311    0.960     13
  ##   5  1.749 0.813          1.749    0.813     13
  ##   6 -1.513 1.032         -1.513    1.032     13

## End(Not run)
```

---

tamaan *Wrapper Function for* **TAM** *Language*

Description

This function is a convenience wrapper function for several item response models in **TAM**. Using the `tamaanify` framework, multidimensional item response models, latent class models, located and ordered latent class models and mixture item response models can be estimated.

Usage

```
tamaan(tammodel, resp, tam.method=NULL, control=list(), doparse=TRUE, ...)

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

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

Arguments

`tammodel`	String for specification in **TAM**, see also `tamaanify`.
`resp`	Dataset with item responses
`tam.method`	One of the **TAM** methods `tam.mml`, `tam.mml.2pl` or `tam.mml.3pl`.
`control`	List with control arguments. See `tam.mml`.
`doparse`	Optional logical indicating whether `lavmodel` should be parsed for `DO` statements, see `doparse`.
`...`	Further arguments to be passed to `tam.mml`, `tam.mml.2pl` or `tam.mml.3pl`.
`object`	Object of class `tamaan`
`file`	A file name in which the summary output will be written
`x`	Object of class `tamaan`

Value

Values generated by `tam.mml`, `tam.mml.2pl` or `tam.mml.3pl`. In addition, the list also contains the (optional) entries

`tamaanify`	Output produced by `tamaanify`
`lcaprobs`	Matrix with probabilities for latent class models
`locs`	Matrix with cluster locations (for `TYPE="LOCLCA"`)
`probs_MIXTURE`	Class probabilities (for `TYPE="MIXTURE"`)
`moments_MIXTURE`	Distribution parameters (for `TYPE="MIXTURE"`)
`itempartable_MIXTURE`	Item parameters (for `TYPE="MIXTURE"`)
`ind_classprobs`	Individual posterior probabilities for latent classes (for `TYPE="MIXTURE"`)

See Also

See `tamaanify` for more details about model specification using `tammodel`.

See `tam.mml` or `tam.mml.3pl` for more examples.

Examples

```
## Not run:

#############################################################################
# EXAMPLE 1: Examples dichotomous data data.read
#############################################################################

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

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

tammodel <- "
LAVAAN MODEL:
  F1=~ A1__C4
  F1 ~~ F1
ITEM TYPE:
  ALL(Rasch);
    "
# estimate model
mod1 <- TAM::tamaan( tammodel, resp=dat)
summary(mod1)

#**********************************************************************
#*** Model 2: 2PL model with some selected items

tammodel <- "
```

```
LAVAAN MODEL:
  F1=~ A1__B1 + B3 + C1__C3
  F1 ~~ F1
    "
mod2 <- TAM::tamaan( tammodel, resp=dat)
summary(mod2)

#*******************************************************************
#*** Model 3: Multidimensional IRT model

tammodel <- "
LAVAAN MODEL:
  G=~ A1__C4
  F1=~ A1__B4
  F2=~ C1__C4
  F1 ~~ F2
  # specify fixed entries in covariance matrix
  F1 ~~ 1*F1
  F2 ~~ 1*F2
  G  ~~ 0*F1
  G  ~~ 0.3*F2
  G  ~~ 0.7*G
    "
mod3 <- TAM::tamaan( tammodel, resp=dat, control=list(maxiter=30))
summary(mod3)

#*******************************************************************
#*** Model 4: Some linear constraints for item slopes and intercepts

tammodel <- "
LAVAAN MODEL:
  F=~ lam1__lam10*A1__C2
  F=~ 0.78*C3
  F ~~ F
  A1 | a1*t1
  A2 | a2*t1
  A3 | a3*t1
  A4 | a4*t1
  B1 | b1*t1
  B2 | b2*t1
  B3 | b3*t1
  C1 | t1
MODEL CONSTRAINT:
  # defined parameters
  # only linear combinations are permitted
  b2==1.3*b1 + (-0.6)*b3
  a1==q1
  a2==q2 + t
  a3==q1 + 2*t
  a4==q2 + 3*t
  # linear constraints for loadings
  lam2==1.1*lam1
  lam3==0.9*lam1 + (-.1)*lam0
```

```
  lam8==lam0
  lam9==lam0
    "
mod4 <- TAM::tamaan( tammodel, resp=dat, control=list(maxiter=5) )
summary(mod4)

#*********************************************************************
#*** Model 5: Latent class analysis with three classes

tammodel <- "
ANALYSIS:
  TYPE=LCA;
  NCLASSES(3);   # 3 classes
  NSTARTS(5,20); # 5 random starts with 20 iterations
LAVAAN MODEL:
  F=~ A1__C4
    "
mod5 <- TAM::tamaan( tammodel, resp=dat, control=list(maxiter=100)  )
summary(mod5)

#*********************************************************************
#*** Model 6: Ordered latent class analysis with three classes

tammodel <- "
ANALYSIS:
  TYPE=OLCA;
  NCLASSES(3);    # 3 classes
  NSTARTS(20,40); # 20 random starts with 40 iterations
LAVAAN MODEL:
  F=~ A1__C4
    "
mod6 <- TAM::tamaan( tammodel, dat )
summary(mod6)

#*********************************************************************
#*** Model 7: Unidimensional located latent class model with three classes

tammodel <- "
ANALYSIS:
  TYPE=LOCLCA;
  NCLASSES(3)
  NSTARTS(10,40)
LAVAAN MODEL:
  F=~ A1__C4
  B2 | 0*t1
    "
mod7 <- TAM::tamaan( tammodel, resp=dat)
summary(mod7)

#*********************************************************************
#*** Model 8: Two-dimensional located latent class analysis with some
#             priors and equality constraints among thresholds
```

```
tammodel <- "
ANALYSIS:
  TYPE=LOCLCA;
  NCLASSES(4);
  NSTARTS(10,20);
LAVAAN MODEL:
  AB=~ A1__B4
  C=~ C1__C4
  A1 | a1diff*t1
  B2 | 0*t1
  C2 | 0*t1
  B1 | a1diff*t1
MODEL PRIOR:
  # prior distributions for cluster locations
  DO2(1,4,1,1,2,1)
    Cl%1_Dim%2 ~ N(0,2);
  DOEND
    "
# estimate model
mod8 <- TAM::tamaan( tammodel, resp=dat )
summary(mod8)

#*******************************************************************
#*** Model 9: Two-dimensional model with constraints on parameters

tammodel <- "
LAVAAN MODEL:
  FA=~ A1+b*A2+A3+d*A4
  FB=~ B1+b*B2+B3+d*B4
  FA ~~ 1*FA
  FA ~~ FB
  FB ~~ 1*FB
  A1 | c*t1
  B1 | c*t1
  A2 | .7*t1
    "
# estimate model
mod9 <- TAM::tamaan( tammodel, resp=dat, control=list(maxiter=30) )
summary(mod9)

#############################################################################
# EXAMPLE 2: Examples polytomous data | data.Students
#############################################################################

library(CDM)
data( data.Students, package="CDM")
dat <- data.Students[,3:13]
  ##   > colnames(dat)
  ##    [1] "act1" "act2" "act3" "act4" "act5" "sc1"  "sc2"  "sc3"  "sc4"  "mj1"  "mj2"

#*******************************************************************
#*** Model 1: Two-dimensional generalized partial credit model
```

```
tammodel <- "
LAVAAN MODEL:
  FA=~ act1__act5
  FS=~ sc1__sc4
  FA ~~ 1*FA
  FS ~~ 1*FS
  FA ~~ FS
    "
# estimate model
mod1 <- TAM::tamaan( tammodel, dat, control=list(maxiter=10)  )
summary(mod1)

#*********************************************************************
#*** Model 2: Two-dimensional model, some constraints

tammodel <- "
LAVAAN MODEL:
  FA=~ a1__a4*act1__act4 + 0.89*act5
  FS=~ 1*sc1 + sc2__sc4
  FA ~~ FA
  FS ~~ FS
  FA ~~ FS
  # some equality constraints
  act1 + act3 | a13_t1 * t1
  act1 + act3 | a13_t2 * t2
    "
# only create design matrices with tamaanify
mod2 <- TAM::tamaanify( tammodel, dat  )
mod2$lavpartable
# estimate model (only few iterations as a test)
mod2 <- TAM::tamaan( tammodel, dat, control=list(maxiter=10)  )
summary(mod2)

#*********************************************************************
#*** Model 3: Two-dimensional model, some more linear constraints

tammodel <- "
LAVAAN MODEL:
  FA=~ a1__a5*act1__act5
  FS=~ b1__b4*sc1__sc4
  FA ~~ 1*FA
  FA ~~ FS
  FS ~~ 1*FS
  act1 + act3 | a13_t1 * t1
  act1 + act3 | a13_t2 * t2
MODEL CONSTRAINT:
  a1==q0
  a2==q0
  a3==q0    + q1
  a4==q2
  a5==q2 + q1
    "
# estimate
```

```
mod3 <- TAM::tamaan( tammodel, dat, control=list(maxiter=300 )  )
summary(mod3)

#*********************************************************************
#*** Model 4: Latent class analysis with three latent classes

tammodel <- "
ANALYSIS:
  TYPE=LCA;
  NCLASSES(3);    # 3 classes
  NSTARTS(10,30); # 10 random starts with 30 iterations
LAVAAN MODEL:
  F=~ act1__act5
    "
# estimate model
mod4 <- TAM::tamaan( tammodel, resp=dat)
summary(mod4)

#*********************************************************************
#*** Model 5: Partial credit model with "PCM2" parametrization

# select data
dat1 <- dat[, paste0("act",1:5) ]
# specify tamaan model
tammodel <- "
  LAVAAN MODEL:
    F=~ act1__act5
    F ~~ F
    # use DO statement as shortages
    DO(1,5,1)
      act% | b%_1 * t1
      act% | b%_2 * t2
    DOEND
  MODEL CONSTRAINT:
    DO(1,5,1)
      b%_1==delta% + tau%_1
      b%_2==2*delta%
    DOEND
  ITEM TYPE:
    ALL(PCM)
  "
# estimate model
mod5 <- TAM::tamaan( tammodel, dat1 )
summary(mod5)
# compare with PCM2 parametrization in tam.mml
mod5b <- TAM::tam.mml( dat1, irtmodel="PCM2" )
summary(mod5b)

#*********************************************************************
#*** Model 6: Rating scale model

# select data
dat1 <- dat[, paste0("sc",1:4) ]
```

```
psych::describe(dat1)

# specify tamaan model
tammodel <- "
  LAVAAN MODEL:
    F=~ sc1__sc4
    F ~~ F
    # use DO statement as shortages
    DO(1,4,1)
      sc% | b%_1 * t1
      sc% | b%_2 * t2
      sc% | b%_3 * t3
    DOEND
  MODEL CONSTRAINT:
    DO(1,4,1)
      b%_1==delta% + step1
      b%_2==2*delta% + step1 + step2
      b%_3==3*delta%
    DOEND
  ITEM TYPE:
    ALL(PCM)
  "
# estimate model
mod6 <- TAM::tamaan( tammodel, dat1 )
summary(mod6)
# compare with RSM in tam.mml
mod6b <- TAM::tam.mml( dat1, irtmodel="RSM" )
summary(mod6b)

#********************************************************************
#*** Model 7: Partial credit model with Fourier basis for
#             item intercepts (Thissen, Cai & Bock, 2010)
# see ?tamaanify manual

# define tamaan model
tammodel <- "
LAVAAN MODEL:
   mj=~ mj1__mj4
   mj ~~ 1*mj
ITEM TYPE:
  mj1(PCM,2)
  mj2(PCM,3)
  mj3(PCM)
  mj4(PCM,1)
   "
# estimate model
mod7 <- TAM::tamaan( tammodel, dat )
summary(mod7)
# -> This function can also be applied for the generalized partial credit
#    model (GPCM).

#############################################################################
# EXAMPLE 3: Rasch model and mixture Rasch model (Geiser & Eid, 2010)
```

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

data(data.geiser, package="TAM")
dat <- data.geiser

#*********************************************************************
#*** Model 1: Rasch model
tammodel <- "
LAVAAN MODEL:
  F=~ mrt1__mrt6
  F ~~ F
ITEM TYPE:
  ALL(Rasch);
    "
mod1 <- TAM::tamaan( tammodel, resp=dat  )
summary(mod1)

#*********************************************************************
#*** Model 2: Mixed Rasch model with two classes
tammodel <- "
ANALYSIS:
  TYPE=MIXTURE ;
  NCLASSES(2);
  NSTARTS(20,25);
LAVAAN MODEL:
  F=~ mrt1__mrt6
  F ~~ F
ITEM TYPE:
  ALL(Rasch);
    "
mod2 <- TAM::tamaan( tammodel, resp=dat  )
summary(mod2)

# plot item parameters
ipars <- mod2$itempartable_MIXTURE[ 1:6, ]
plot( 1:6, ipars[,3], type="o", ylim=c(-3,2), pch=16,
        xlab="Item", ylab="Item difficulty")
lines( 1:6, ipars[,4], type="l", col=2, lty=2)
points( 1:6, ipars[,4],  col=2, pch=2)

# extract individual posterior distribution
post2 <- IRT.posterior(mod2)
str(post2)
# num [1:519, 1:30] 0.000105 0.000105 0.000105 0.000105 0.000105 ...
# - attr(*, "theta")=num [1:30, 1:30] 1 0 0 0 0 0 0 0 0 0 ...
# - attr(*, "prob.theta")=num [1:30, 1] 1.21e-05 2.20e-04 2.29e-03 1.37e-02 4.68e-02 ...
# - attr(*, "G")=num 1

# There are 2 classes and 15 theta grid points for each class
# The loadings of the theta grid on items are as follows
mod2$E[1,2,,"mrt1_F_load_Cl1"]
mod2$E[1,2,,"mrt1_F_load_Cl2"]
```

```
# compute individual posterior probability for class 1 (first 15 columns)
round( rowSums( post2[, 1:15] ), 3 )
# columns 16 to 30 refer to class 2


## End(Not run)
```

---

tamaanify *Function for Parsing* **TAM** *Input*

Description

This function parses a so called `tammodel` which is a string used for model estimation in **TAM**. The function is based on the **lavaan** syntax and operates at the extension `lavaanify.IRT`.

Usage

```
tamaanify(tammodel, resp, tam.method=NULL, doparse=TRUE )
```

Arguments

`tammodel`	String for model definition following the rules described in Details and in Examples.
`resp`	Item response dataset
`tam.method`	One of the **TAM** methods `tam.mml`, `tam.mml.2pl` or `tam.mml.3pl`.
`doparse`	Optional logical indicating whether `lavmodel` should be parsed for `DO` statements.

Details

The model syntax `tammodel` consists of several sections. Some of them are optional.

`ANALYSIS:`
Possible model types are unidimensional and multidimensional item response models (`TYPE="TRAIT"`), latent class models (`"LCA"`), located latent class models (`"LOCLCA"`; e.g. Formann, 1989; Bartolucci, 2007), ordered latent class models (`"OLCA"`; only works for dichotomous item responses; e.g. Hoijtink, 1997; Shojima, 2007) and mixture distribution models (`"MIXTURE"`; e.g. von Davier, 2007).

`LAVAAN MODEL:`
For specification of the syntax, see `lavaanify.IRT`.

`MODEL CONSTRAINT:`
Linear constraints can be specified by using conventional specification in R syntax. All terms must be combined with the + operator. Equality constraints are set by using the == operator as in **lavaan**.

ITEM TYPE:
The following item types can be defined: Rasch model (Rasch), the 2PL model (2PL), partial credit model (PCM) and the generalized partial credit model (GPCM).

The item intercepts can also be smoothed for the PCM and the GPCM by using a Fourier basis proposed by Thissen, Cai and Bock (2010). For an item with a maximum of score of K, a smoothed partial credit model is requested by PCM(kk) where kk is an integer between 1 and K. With kk=1, only a linear function is used. The subsequent integers correspond to Fourier functions with decreasing periods. See Example 2, Model 7 of the tamaan function.

PRIOR:
Possible prior distributions: Normal distribution N(mu,sd), truncated normal distribution TN(mu,sd,low,upp) and Beta distribution Beta(a,b). Parameter labels and prior specification must be separated by ~.

Value

A list with following (optional) entries which are used as input in one of the **TAM** functions tam.mml, tam.mml.2pl or tam.mml.3pl:

tammodel	Model input for **TAM**
tammodel.dfr	Processed tammodel input
ANALYSIS	Syntax specified in ANALYSIS
ANALYSIS.list	Parsed specifications in ANALYSIS
LAVAANMODEL	Syntax specified in LAVAAN MODEL
lavpartable	Parameter table processed by the syntax in LAVAAN MODEL
items	Informations about items: Number of categories, specified item response function
maxcat	Maximum number of categories
ITEMTYPE	Syntax specified in ITEM TYPE
MODELCONSTRAINT	Syntax specified in MODEL CONSTRAINT
MODELCONSTRAINT.dfr	Processed syntax in MODEL CONSTRAINT
modelconstraint.thresh	Processed data frame for model constraint of thresholds
modelconstraint.loading	Processed data frame for loadings
resp	Data set for usage
method	Used **TAM** function
A	Design matrix A
Q	Design matrix for loadings
Q.fixed	Fixed values in Q matrix
B.fixed	Matrix with fixed item loadings (used for tam.mml.2pl)

`L`	Processed design matrix for loadings when there are model constraints for loadings
`variance.fixed`	Matrix for specification of fixed values in covariance matrix
`est.variance`	Logical indicating whether variance should be estimated (`tam.mml.2pl`)
`theta.k`	Theta design matrix
`E`	Design matrix E
`notA`	Logical indicating whether A matrix is defined
`gammaslope.fixed`	Fixed `gammaslope` parameters
`gammaslope.prior`	Prior distributions for `gammaslope` parameters
`xsi.fixed`	Fixed ξ parameter
`xsi.prior`	Prior distributions for ξ parameters

References

Bartolucci, F. (2007). A class of multidimensional IRT models for testing unidimensionality and clustering items. *Psychometrika, 72*, 141-157. doi:10.1007/s1133600513769

Formann, A. K. (1989). Constrained latent class models: Some further applications. *British Journal of Mathematical and Statistical Psychology, 42*, 37-54. doi:10.1111/j.20448317.1989.tb01113.x

Hojtink, H., & Molenaar, I. W. (1997). A multidimensional item response model: Constrained latent class analysis using the Gibbs sampler and posterior predictive checks. *Psychometrika, 62*(2), 171-189. doi:10.1007/BF02295273

Thissen, D., Cai, L., & Bock, R. D. (2010). The nominal categories item response model. In M. L. Nering & Ostini, R. (Eds.). *Handbook of Polytomous Item Response Models* (pp. 43-75). New York: Routledge.

Shojima, K. (2007). *Latent rank theory: Estimation of item reference profile by marginal maximum likelihood method with EM algorithm*. DNC Research Note 07-12.

von Davier, M. (2007). *Mixture distribution diagnostic models*. ETS Research Report ETS RR-07-32. Princeton, ETS. doi:10.1002/j.23338504.2007.tb02074.x

See Also

See `tamaan` for more examples. Other examples are included in `tam.mml` and `tam.mml.3pl`.

`lavaanify.IRT`

Examples

```
## Not run:

#############################################################################
# EXAMPLE 1: Examples dichotomous data data.read
#############################################################################

library(sirt)
```

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

#*********************************************************************
#*** Model 1: 2PL estimation with some fixed parameters and
#             equality constraints
tammodel <- "
LAVAAN MODEL:
  F2=~ C1__C2 + 1.3*C3 + C4
  F1=~ A1__B1
  # fixed loading of 1.4 for item B2
  F1=~ 1.4*B2
  F1=~ B3
  F1 ~~ F1
  F2 ~~ F2
  F1 ~~ F2
  B1 | 1.23*t1 ; A3 | 0.679*t1
  A2 | a*t1 ; C2 | a*t1 ; C4 | a*t1
  C3 | x1*t1 ; C1 | x1*t1
ITEM TYPE:
  A1__A3 (Rasch) ;
  A4 (2PL) ;
  B1__C4 (Rasch) ;
    "
# process model
out <- TAM::tamaanify( tammodel, resp=dat)
# inspect some output
out$method          # used TAM function
out$lavpartable     # lavaan parameter table

#*********************************************************************
#*** Model 2: Latent class analysis with three classes
tammodel <- "
ANALYSIS:
  TYPE=LCA;
  NCLASSES(3);   # 3 classes
  NSTARTS(5,20); # 5 random starts with 20 iterations
LAVAAN MODEL:
  F=~ A1__C4
    "
# process syntax
out <- TAM::tamaanify( tammodel, resp=dat)
str(out$E)     # E design matrix for estimation with tam.mml.3pl function

#*********************************************************************
#*** Model 3: Linear constraints for item intercepts and item loadings
tammodel <- "
LAVAAN MODEL:
  F=~ lam1__lam10*A1__C2
  F ~~ F
  A1 | a1*t1
  A2 | a2*t1
  A3 | a3*t1
```

```
  A4 | a4*t1
  B1 | b1*t1
  B2 | b2*t1
  B3 | b3*t1
  C1 | t1
MODEL CONSTRAINT:
  # defined parameters
  # only linear combinations are permitted
  b2==1.3*b1 + (-0.6)*b3
  a1==q1
  a2==q2 + t
  a3==q1 + 2*t
  a4==q2 + 3*t
  # linear constraints for loadings
  lam2==1.1*lam1
  lam3==0.9*lam1 + (-.1)*lam0
  lam8==lam0
  lam9==lam0
    "
# parse syntax
mod1 <- TAM::tamaanify( tammodel, resp=dat)
mod1$A          # design matrix A for intercepts
mod1$L[,1,]     # design matrix L for loadings

## End(Not run)

#############################################################################
# EXAMPLE 2: Examples polytomous data data.Students
#############################################################################

library(CDM)
data( data.Students, package="CDM")
dat <- data.Students[,3:13]

#**********************************************************************
#*** Model 1: Two-dimensional generalized partial credit model
tammodel <- "
LAVAAN MODEL:
  FA=~ act1__act5
  FS=~ sc1__sc4
  FA ~~ 1*FA
  FS ~~ 1*FS
  FA ~~ FS
  act1__act3 | t1
  sc2 | t2
    "
out <- TAM::tamaanify( tammodel, resp=dat)
out$A    # design matrix for item intercepts
out$Q    # loading matrix for items

#**********************************************************************
#*** Model 2: Linear constraints
```

```
# In the following syntax, linear equations for multiple constraints
# are arranged over multiple lines.
tammodel <- "
  LAVAAN MODEL:
    F=~ a1__a5*act1__act5
    F ~~ F
  MODEL CONSTRAINT:
      a1==delta +
                tau1
      a2==delta
      a3==delta + z1
      a4==1.1*delta +
              2*tau1
                + (-0.2)*z1
  "
# tamaanify model
res <- TAM::tamaanify( tammodel, dat )
res$MODELCONSTRAINT.dfr
res$modelconstraint.loading
```

---

tampv2datalist *Conversion of Plausible Value Object into Datalist*

Description

Converts a `tam.pv` object and a matrix of covariates into a list of multiply imputed datasets. This list can be conveniently analyzed by R packages such as **semTools**, **Zelig**, **mice** or **BIFIEsurvey**.

Usage

```
tampv2datalist(tam.pv.object, pvnames=NULL, Y=NULL, Y.pid="pid",
      as_mids=FALSE, stringsAsFactors=FALSE)
```

Arguments

`tam.pv.object`	Generated `tam.pv` object
`pvnames`	Variable names of generated plausible values
`Y`	Matrix with covariates
`Y.pid`	Person identifier in `Y` matrix. It is not required that a person identifier is provided. In this case, the merge of the datasets will be conducted as for `rbind`.
`as_mids`	Logical indicating whether the datalist should be converted into an object of class `mids` for analysis in the **mice** package. This functionality uses the the function `miceadds::datalist2mids`.
`stringsAsFactors`	Logical indicating whether strings in the data frame should be converted into factors

Value

List of multiply imputed datasets or an `mids` object

See Also

For examples see `tam.pv`.

`tam_downcode` *Downcoding an Item Response Dataset*

Description

Recodes item categories in a data frame such that each item has values $0, 1, \ldots, K_i$.

Usage

```
tam_downcode(dat)
```

Arguments

`dat`	Data frame containing item responses

Value

List with following entries

`dat`	Recoded dataset
`rec`	Recoding table

Examples

```
#############################################################################
# EXAMPLE 1: Downcoding in a toy example
#############################################################################

#-- simulate some toy data
set.seed(989)
# values to be sampled
vals <- c(NA, 0:6)
# number of persons and items
N <- 10; I <- 5
dat <- as.data.frame(matrix(NA, nrow=N, ncol=I))
colnames(dat) <- paste0("I",1:I)
for (ii in 1L:I){
    dat[,ii] <- sample(vals, size=N, replace=TRUE)
}

#-- apply downcoding
res <- TAM::tam_downcode(dat)
```

```
dat <- res$dat   # extract downcoded dataset
rec <- res$rec   # extract recoded table
```

---

tam_irf_3pl *Item Response Function for the 3PL Model*

Description

Computes the item response function for the 3PL model in the **TAM** package.

Usage

```
tam_irf_3pl(theta, AXsi, B, guess=NULL, subtract_max=TRUE)
```

Arguments

theta	Matrix or vector of θ values
AXsi	Matrix of item-category parameters
B	Array containing item-category loadings
guess	Optional parameter of guessing parameters
subtract_max	Logical indicating whether numerical underflow in probabilities should be explicitly avoided

Value

Array containing item response probabilities arranged by the dimensions theta points $\times$ items $\times$ categories

Examples

```
## Not run:
#############################################################################
# EXAMPLE 1: 2PL example
#############################################################################

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

#* estimate 2PL model
mod <- TAM::tam.mml.2pl( resp=dat )
#* define theta vector
theta <- seq(-3,3, len=41)
#* compute item response probabilities
probs <- TAM::tam_irf_3pl( theta=theta, AXsi=mod$AXsi, B=mod$B )
str(probs)

## End(Not run)
```

tam_NA_pattern *Missing Data Patterns*

Description

Determines patterns of missing values or pattern of dichotomous item responses.

Usage

```
tam_NA_pattern(x)

tam_01_pattern(x)
```

Arguments

x	Matrix or data frame

Value

List containing pattern identifiers and indices

Examples

```
#############################################################################
# EXAMPLE 1: Missing data patterns
#############################################################################

data(data.sim.rasch.missing, package="TAM")
dat <- data.sim.rasch.missing

res <- TAM::tam_NA_pattern(dat)
str(res)

## Not run:
#############################################################################
# EXAMPLE 2: Item response patterns
#############################################################################

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

res <- TAM::tam_01_pattern(dat)
str(res)

## End(Not run)
```

weighted_Stats *Descriptive Statistics for Weighted Data*

Description

Some descriptive statistics for weighted data: variance, standard deviation, means, skewness, excess kurtosis, quantiles and frequency tables. Missing values are automatically removed from the data.

Usage

```
weighted_mean(x, w=rep(1, length(x)), select=NULL  )

weighted_var(x, w=rep(1, length(x)), method="unbiased", select=NULL )

weighted_sd(x, w=rep(1, length(x)), method="unbiased", select=NULL )

weighted_skewness( x, w=rep(1,length(x)), select=NULL  )

weighted_kurtosis( x, w=rep(1,length(x)), select=NULL  )

weighted_quantile( x, w=rep(1,length(x)), probs=seq(0,1,.25), type=NULL, select=NULL )

weighted_table( x, w=NULL, props=FALSE )
```

Arguments

`x`	A numeric vector. For `weighted_table`, a matrix with two columns can be used as input for cross-tabulation.
`w`	Optional vector of sample weights
`select`	Vector referring to selected cases
`method`	Computation method (can be `"unbiased"` or `"ML"`)), see `stats::cov.wt`
`probs`	Vector with probabilities
`type`	Quantile type. For unweighted data, quantile types 6 and 7 can be used (see `stats::quantile`). For weighted data, the quantile type `"i/n"` is used (see `Hmisc::wtd.quantile`)).
`props`	Logical indicating whether relative or absolute frequencies should be calculated.

Value

Numeric value

See Also

See `stats::weighted.mean` for computing a weighted mean.

See `stats::var` for computing unweighted variances.

See `stats::quantile` and `Hmisc::wtd.quantile`) for quantiles.

Examples

```
#############################################################################
# EXAMPLE 1: Toy example for weighted_var function
#############################################################################

set.seed(9897)
# simulate data
N <- 10
x <- stats::rnorm(N)
w <- stats::runif(N)

#---- variance

# use weighted_var
weighted_var( x=x, w=w )
# use cov.wt
stats::cov.wt( data.frame(x=x), w=w )$cov[1,1]
## Not run:
# use wtd.var from Hmisc package
Hmisc::wtd.var(x=x, weights=w, normwt=TRUE, method="unbiased")

#---- standard deviation
weighted_sd( x=x, w=w )

#---- mean
weighted_mean( x=x, w=w )
stats::weighted.mean( x=x, w=w )

#---- weighted quantiles for unweighted data
pvec <- c(.23, .53, .78, .99 )  # choose probabilities
type <- 7

# quantiles for unweighted data
weighted_quantile( x, probs=pvec, type=type)
quantile( x, probs=pvec, type=type)
Hmisc::wtd.quantile(x,probs=pvec, type=type)

# quantiles for weighted data
pvec <- c(.23, .53, .78, .99 )  # probabilities
weighted_quantile( x, w=w, probs=pvec)
Hmisc::wtd.quantile(x, weights=w, probs=pvec)

#--- weighted skewness and kurtosis
weighted_skewness(x=x, w=w)
weighted_kurtosis(x=x, w=w)

#############################################################################
# EXAMPLE 2: Descriptive statistics normally distributed data
#############################################################################

# simulate some normally distributed data
set.seed(7768)
```

```
x <- stats::rnorm( 10000, mean=1.7, sd=1.2)
# some statistics
weighted_mean(x=x)
weighted_sd(x=x)
weighted_skewness(x=x)
weighted_kurtosis(x=x)

#############################################################################
# EXAMPLE 3: Frequency tables
#############################################################################

#*****
# simulate data for weighted frequency tables
y <- scan()
   1 0  1 1   1 2   1 3    1 4
   2 0  2 1   2 2   2 3    2 4

y <- matrix( y, ncol=2, byrow=TRUE)
# define probabilities
set.seed(976)
pr <- stats::runif(10)
pr <- pr / sum(pr)
# sample data
N <- 300
x <- y[ sample( 1:10, size=300, prob=pr, replace=TRUE ), ]
w <- stats::runif( N, 0.5, 1.5 )

# frequency table unweighted data
weighted_table(x[,2] )
table( x[,2] )

# weighted data and proportions
weighted_table(x[,2], w=w, props=TRUE)

#*** contingency table
table( x[,1], x[,2] )
weighted_table( x )
# table using weights
weighted_table( x, w=w )

## End(Not run)
```

WLErel *Reliability Estimation in* **TAM**

Description

Functions for computing reliability estimates.

Usage

```
WLErel(theta, error, w=rep(1, length(theta)), select=NULL)

EAPrel(theta, error, w=rep(1, length(theta)), select=NULL)
```

Arguments

`theta`	Vector with theta estimates
`error`	Vector with standard errors of theta estimates
`w`	Optional vector of person weights
`select`	Optional vector for selecting cases

Details

The reliability formulas follow Adams (2005). Let v denote the variance of `theta` estimates and let s denote the average of the squared `error`. Then, the WLE reliability is defined as $1 - s/v = (v - s)/v$ while the EAP reliability is defined as $1 - s/(s + v) = v/(s + v)$.

Value

Numeric value

References

Adams, R. J. (2005). Reliability as a measurement design effect. *Studies in Educational Evaluation, 31*(2), 162-172. doi:10.1016/j.stueduc.2005.05.008

Examples

```
#############################################################################
# EXAMPLE 1: Toy example for reliability functions
#############################################################################

set.seed(9897)
N <- 100
# simulate theta and error SDs
x <- stats::rnorm(N,sd=2)
error <- stats::runif(N, .7, 1.3)
# compute WLE reliability
WLErel(x,error)
# compute EAP reliaility
EAPrel(x,error)
```

Index