

# Package ‘sirt’

June 29, 2015

**Type** Package

**Title** Supplementary Item Response Theory Models

**Version** 1.8-9

**Date** 2015-06-28

**Author** Alexander Robitzsch [aut,cre]

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

**Description** Supplementary item response theory models to complement existing functions in R, including multidimensional compensatory and noncompensatory IRT models, MCMC for hierarchical IRT models and testlet models, NOHARM, faceted and hierarchical rater models, ordinal IRT model (ISOP), latent regression models, DETECT statistic, local structural equation modeling (LSEM).

**Depends** R (>= 2.15.0), TAM, CDM

**Imports** stats, graphics, utils, Rcpp, mirt, lavaan, qgraph, semPlot, combinat, ic.infer, sfsmisc, sm, psych, pbivnorm, gtools, igrph, coda, mvtnorm, MASS, Matrix, lavaan.survey, survey

**Suggests** miceadds

**LinkingTo** Rcpp, RcppArmadillo

**License** GPL (>= 2)

**URL** <https://sites.google.com/site/alexanderrobitzsch/software>

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2015-06-29 02:27:40

## R topics documented:

sirt-package . . . . .	4
automatic.recode . . . . .	13
brm-Methods . . . . .	14
btm . . . . .	18

categorize . . . . .	22
ccov.np . . . . .	24
class.accuracy.rasch . . . . .	25
conf.detect . . . . .	26
data.activity.itempars . . . . .	30
data.big5 . . . . .	31
data.bs . . . . .	35
data.eid . . . . .	38
data.ess2005 . . . . .	45
data.g308 . . . . .	46
data.inv4gr . . . . .	48
data.liking.science . . . . .	49
data.long . . . . .	49
data.lsem . . . . .	54
data.math . . . . .	54
data.mcdonald . . . . .	55
data.mixed1 . . . . .	59
data.ml . . . . .	60
data.noharm . . . . .	61
data.pars1.rasch . . . . .	62
data.pirlsmissing . . . . .	63
data.pisaMath . . . . .	64
data.pisaPars . . . . .	65
data.pisaRead . . . . .	65
data.pw . . . . .	66
data.ratings . . . . .	67
data.raw1 . . . . .	68
data.read . . . . .	68
data.reck . . . . .	88
data.sirt . . . . .	94
data.timss . . . . .	97
data.timss07.G8.RUS . . . . .	98
data.wide2long . . . . .	99
detect.index . . . . .	101
dif.logistic.regression . . . . .	102
dif.strata.variance . . . . .	104
dif.variance . . . . .	105
dirichlet.mle . . . . .	106
dirichlet.simul . . . . .	109
eigenvalues.manymatrices . . . . .	110
eigenvalues.sirt . . . . .	112
equating.rasch . . . . .	113
equating.rasch.jackknife . . . . .	115
expl.detect . . . . .	116
f1d.irt . . . . .	117
fit.isop . . . . .	121
fuzcluster . . . . .	123
fuzdiscr . . . . .	126

gom.em . . . . .	129
gom.jml . . . . .	137
greenyang.reliability . . . . .	139
invariance.alignment . . . . .	141
IRT.mle . . . . .	151
isop . . . . .	153
isop.scoring . . . . .	158
isop.test . . . . .	160
latent.regression.em.raschtype . . . . .	161
lavaan2mirt . . . . .	167
lc.2raters . . . . .	174
likelihood.adjustment . . . . .	176
linking.haberman . . . . .	179
linking.robust . . . . .	185
lsdm . . . . .	187
lsem.estimate . . . . .	193
lsem.permutationTest . . . . .	197
marginal.truescore.reliability . . . . .	199
matrixfunctions.sirt . . . . .	201
mcmc.2pno . . . . .	203
mcmc.2pno.ml . . . . .	205
mcmc.2pnoh . . . . .	210
mcmc.3pno.testlet . . . . .	213
mcmc.list.descriptives . . . . .	218
mcmclist2coda . . . . .	220
md.pattern.sirt . . . . .	221
mirt.specify.partitionable . . . . .	222
mirt.wrapper . . . . .	224
mle.pcm.group . . . . .	230
modelfit.sirt . . . . .	232
monoreg.rowwise . . . . .	236
nedelsky-methods . . . . .	237
noharm.sirt . . . . .	241
np.dich . . . . .	247
pbivnorm2 . . . . .	249
pcm.conversion . . . . .	250
pcm.fit . . . . .	251
person.parameter.rasch.copula . . . . .	254
personfit.stat . . . . .	256
pgenlogis . . . . .	258
plausible.value.imputation.raschtype . . . . .	260
plot.mcmc.sirt . . . . .	264
plot.np.dich . . . . .	265
polychoric2 . . . . .	266
prmse.subscores.scales . . . . .	267
prob.guttman . . . . .	268
qmc.nodes . . . . .	273
R2conquest . . . . .	274

R2noharm . . . . .	279
R2noharm.EAP . . . . .	287
R2noharm.jackknife . . . . .	288
rasch.copula2 . . . . .	290
rasch.evm.pcm . . . . .	298
rasch.jml . . . . .	303
rasch.jml.biascorr . . . . .	306
rasch.jml.jackknife1 . . . . .	307
rasch.mirtlc . . . . .	309
rasch.mml2 . . . . .	321
rasch.pairwise . . . . .	338
rasch.pairwise.itemcluster . . . . .	340
rasch.pml2 . . . . .	343
rasch.prox . . . . .	351
rasch.va . . . . .	353
reliability.nonlinearSEM . . . . .	354
rm.facets . . . . .	356
rm.sdt . . . . .	361
sia.sirt . . . . .	366
sim.qm.ramsay . . . . .	369
sim.rasch.dep . . . . .	372
sim.raschtype . . . . .	374
smirt . . . . .	375
stratified.cronbach.alpha . . . . .	385
summary.mcmc.sirt . . . . .	386
tam2mirt . . . . .	387
testlet.marginalized . . . . .	392
testlet.yen.q3 . . . . .	394
tetrachoric2 . . . . .	395
truescore.irt . . . . .	397
unidim.test.csn . . . . .	400
wle.rasch . . . . .	402
wle.rasch.jackknife . . . . .	404
yen.q3 . . . . .	406

<b>Index</b>	<b>409</b>
--------------	------------

---

sirt-package

---

*Supplementary Item Response Theory Models*


---

## Description

Supplementary item response theory models to complement existing functions in R, including multidimensional compensatory and noncompensatory IRT models, MCMC for hierarchical IRT models and testlet models, NOHARM, faceted and hierarchical rater models, ordinal IRT model (ISOP), latent regression models, DETECT statistic, local structural equation modeling (LSEM).

**Details**

Package: **sirt**  
 Type: Package  
 Version: 1.8  
 Publication Year: 2015  
 License: GPL ( $\geq 2$ )  
 URL: <https://sites.google.com/site/alexanderrobitzsch/software>

This package enables the estimation of following models:

- Multidimensional marginal maximum likelihood estimation (MML) of generalized logistic Rasch type models using the generalized logistic link function (Stukel, 1988) can be conducted with [rasch.mm12](#) and the argument `itemtype="raschtype"`. This model also allows the estimation of the 4PL item response model (Loken & Rulison, 2010). Multiple group estimation, latent regression models and plausible value imputation are supported. In addition, pseudo-likelihood estimation for fractional item response data can be conducted.
- Multidimensional noncompensatory, compensatory and partially compensatory item response models for dichotomous item responses (Reckase, 2009) can be estimated with the [smirt](#) function and the options `irtmodel="noncomp"`, `irtmodel="comp"` and `irtmodel="partcomp"`.
- The unidimensional quotient model (Ramsay, 1989) can be estimated using [rasch.mm12](#) with `itemtype="ramsay.qm"`.
- Unidimensional nonparametric item response models can be estimated employing MML estimation (Rossi, Wang & Ramsay, 2002) by making use of [rasch.mm12](#) with `itemtype="npirt"`. Kernel smoothing for item response function estimation (Ramsay, 1991) is implemented in [np.dich](#).
- The multidimensional IRT copula model (Braeken, 2011) can be applied for handling local dependencies, see [rasch.copula3](#).
- Unidimensional joint maximum likelihood estimation (JML) of the Rasch model is possible with the [rasch.jml](#) function. Bias correction methods for item parameters are included in [rasch.jml.jackknife1](#) and [rasch.jml.biascorr](#).
- The multidimensional latent class Rasch and 2PL model (Bartolucci, 2007) which employs a discrete trait distribution can be estimated with [rasch.mirtlc](#).
- The unidimensional 2PL rater facets model (Lincare, 1994) can be estimated with [rm.facets](#). A hierarchical rater model based on signal detection theory (DeCarlo, Kim & Johnson, 2011) can be conducted with [rm.sdt](#). A simple latent class model for two exchangeable raters is implemented in [lc.2raters](#).
- The discrete grade of membership model (Erosheva, Fienberg & Joutard, 2007) and the Rasch grade of membership model can be estimated by [gom.em](#).
- Some hierarchical IRT models and random item models for dichotomous and normally distributed data (van den Noortgate, de Boeck & Meulders, 2003; Fox & Verhagen, 2010) can be estimated with [mcmc.2pno.ml](#).
- Unidimensional pairwise conditional likelihood estimation (PCML; Zwinderman, 1995) is implemented in [rasch.pairwise](#) or [rasch.pairwise.itemcluster](#).

- Unidimensional pairwise marginal likelihood estimation (PMML; Renard, Molenberghs & Geys, 2004) can be conducted using `rasch.pml3`. In this function local dependence can be handled by imposing residual error structure or omitting item pairs within a dependent item cluster from the estimation.  
The function `rasch.evm.pcm` estimates the multiple group partial credit model based on the pairwise eigenvector approach which avoids iterative estimation.
- Some item response models in **sirt** can be estimated via Markov Chain Monte Carlo (MCMC) methods. In `mcmc.2pno` the two-parameter normal ogive model can be estimated. A hierarchical version of this model (Janssen, Tuerlinckx, Meulders & de Boeck, 2000) is implemented in `mcmc.2pnoh`. The 3PNO testlet model (Wainer, Bradlow & Wang, 2007; Glas, 2012) can be estimated with `mcmc.3pno.testlet`. Some hierarchical IRT models and random item models (van den Noortgate, de Boeck & Meulders, 2003) can be estimated with `mcmc.2pno.ml`.
- For dichotomous response data, the free NOHARM software (McDonald, 1997) estimates the multidimensional compensatory 3PL model and the function `R2noharm` runs NOHARM from within R. Note that NOHARM must be downloaded from <http://noharm.niagararesearch.ca/nh4cldl.html> at first. A pure R implementation of the NOHARM model with some extensions can be found in `noharm.sirt`.
- The measurement theoretic founded nonparametric item response models of Scheiblechner (1995, 1999) – the ISOP and the ADISOP model – can be estimated with `isop.dich` or `isop.poly`. Item scoring within this theory can be conducted with `isop.scoring`.
- The functional unidimensional item response model (Ip et al., 2013) can be estimated with `f1d.irt`.
- The Rasch model can be estimated by variational approximation (Rijmen & Vomlel, 2008) using `rasch.va`.
- The unidimensional probabilistic Guttman model (Proctor, 1970) can be specified with `prob.guttman`.
- A jackknife method for the estimation of standard errors of the weighted likelihood trait estimate (Warm, 1989) is available in `wle.rasch.jackknife`.
- Model based reliability for dichotomous data can be calculated by the method of Green and Yang (2009) with `greenyang.reliability` and the marginal true score method of Dimitrov (2003) using the function `marginal.truescore.reliability`.
- Essential unidimensionality can be assessed by the DETECT index (Stout, Habing, Douglas & Kim, 1996), see the function `conf.detect`.
- Item parameters from several studies can be linked using the Haberman method (Haberman, 2009) in `linking.haberman`. See also `equating.rasch` and `linking.robust`. The alignment procedure (Asparouhov & Muthen, 2013) `invariance.alignment` is originally for confirmatory factor analysis and aims at obtaining approximate invariance.
- Some person fit statistics in the Rasch model (Meijer & Sijtsma, 2001) are included in `personfit.stat`.
- An alternative to the linear logistic test model (LLTM), the so called least squares distance model for cognitive diagnosis (LSDM; Dimitrov, 2007), can be estimated with the function `lsdm`.
- Local structural equation models (LSEM) can be estimated with the `lsem.estimate` function.

## R Function Versions

*anova\_sirt\_\_0.08.R, ARb\_utils\_\_0.23.R, attach.environment.sirt\_\_0.01.R, automatic.recode\_\_1.04.R, brm.irf\_\_0.02.R, brm.sim\_\_0.02.R, btm\_\_1.12.R, btm\_fit\_\_0.01.R,*

*categorize\_0.03.R, class.accuracy.rasch\_0.03.R, data.prep\_1.03.R, data.wide2long\_0.14.R, decategorize\_0.02.R, detect\_0.06.R, dif.logisticregression\_1.02.R, dif.variance\_0.06.R, dirichlet\_1.03.R, eigenvalues.manymatrices\_0.03.R, eigenvalues.sirt\_0.05.R, equating.rasch\_0.02.R, fld.irt\_1.06.R, fit.adisop\_2.09.R, fit.gradedresponse\_1.05.R, fit.gradedresponse\_alg\_1.06.R, fit.isop\_2.05.R, fit.logistic\_2.03.R, fit.logistic\_alg\_0.04.R, fuzcluster\_0.07.R, fuzcluster\_alg\_0.10.R, fuzdiscr\_0.02.R, gom.em.alg\_5.06.R, gom.em\_5.12.R, gom.jml\_0.08.R, gom.jml\_alg\_0.04.R, greenyang.reliability\_1.03.R, invariance.alignment.aux\_1.03.R, invariance.alignment\_2.23.R, invariance.alignment2.aux\_0.14.R, invariance.alignment2\_3.25.R, IRT.factor.scores.sirt\_0.04.R, IRT.irfprob.sirt\_0.07.R, IRT.likelihood\_sirt\_0.10.R, IRT.mle\_0.07.R, IRT.modelfit.sirt\_0.09.R, IRT.posterior\_sirt\_0.07.R, isop.dich\_3.08.R, isop.poly\_2.06.R, isop.scoring\_1.03.R, isop.test\_0.05.R, latent.regression.em.normal\_2.02.R, latent.regression.em.raschtype\_2.46.R, lavaan2mirt\_0.49.R, lavaanify.sirt\_1.11.R, lc.2raters.aux\_0.02.R, lc.2raters\_0.11.R, likelihood\_adjustment\_0.07.R, likelihood\_adjustment\_aux\_0.05.R, likelihood\_moments\_0.02.R, linking.haberman.aux\_0.02.R, linking.haberman\_2.11.R, linking.robust\_1.06.R, logLik\_sirt\_0.07.R, lsdm\_1.07.R, lsdm\_aux\_0.01.R, lsem.estimate\_0.16.R, lsem.fitsem\_0.04.R, lsem.group.moderator\_0.03.R, lsem.helper\_0.01.R, lsem.MGM.stepfunctions\_0.01.R, lsem.parameter.summary\_0.07.R, lsem.permutationTest\_0.11.R, lsem.residualize\_0.11.R, marginal.truescore.reliability\_0.01.R, matrix\_functions\_0.03.R, matrixfunctions\_sirt\_0.06.R, mcmc.2pno.ml\_3.08.R, mcmc.2pno.ml\_alg\_3.12.R, mcmc.2pno.ml\_output\_1.02.R, mcmc.2pno\_1.14.R, mcmc.2pno\_alg\_1.08.R, mcmc.2pnoh\_1.02.R, mcmc.2pnoh\_alg\_0.05.R, mcmc.3pno.testlet\_4.05.R, mcmc.3pno.testlet\_alg\_2.11.R, mcmc.3pno.testlet\_output\_1.06.R, mcmc.aux\_0.02.R, mcmc.list.descriptives\_0.06.R, mcmclist2coda\_0.01.R, md.pattern.sirt\_0.03.R, mirt.IRT.functions\_0.03.R, mirt.model.vars\_0.11.R, mirt.specify.partable\_0.01.R, mirt.wrapper.calc.counts\_0.01.R, mirt.wrapper.coef\_0.07.R, mirt.wrapper.fscores\_0.02.R, mirt.wrapper.itemplot\_0.01.R, mirt.wrapper.posterior\_0.16.R, mle.pcm.group\_0.04.R, mle.reliability\_0.02.R, modelfit.cor.poly\_0.03.R, modelfit.cor\_2.25.R, monoreg.rowwise\_0.03.R, nedelsky.irf\_0.04.R, nedelsky.latresp\_0.01.R, nedelsky.sim\_0.04.R, noharm.sirt.est.aux\_4.04.R, noharm.sirt.preprocess\_0.13.R, noharm.sirt\_0.45.R, normal2.cw\_0.04.R, np.dich\_0.09.R, nr.numdiff\_0.01.R, osink\_0.01.R, pbivnorm2\_1.06.R, pcm.conversion\_0.02.R, pcm.fit\_0.05.R, personfit.stat\_0.03.R, personfit\_1.16.R, pgenlogis\_1.01.R, plausible.values.raschtype\_2.08.R, plot.invariance.alignment\_0.01.R, plot.isop\_1.05.R, plot.lsem.permutationTest\_0.10.R, plot.lsem\_0.17.R, plot.mcmc.sirt\_0.10.R, plot.rasch.mml\_0.06.R, plot.rm.sdt\_0.02.R, polychoric2\_0.04.R, prmse.subscores\_0.02.R, prob.guttman\_1.05.R, qmc.nodes\_0.02.R, R2noharm-utility\_1.04.R, R2noharm.EAP\_0.15.R, R2noharm.jackknife\_1.03.R, R2noharm\_2.14.R, rasch.conquest\_1.22.R, rasch.copula\_0.993.R, rasch.copula2\_6.16.R, rasch.copula2\_aux\_1.06.R, rasch.copula3.covariance\_0.05.R, rasch.copula3\_6.39.R, rasch.copula3\_aux\_6.09.R, rasch.evm.pcm.methods\_0.02.R, rasch.evm.pcm\_1.11.R, rasch.evm.pcm\_aux\_0.02.R, rasch.jml.biascorr\_0.03.R, rasch.jml\_3.12.R, rasch.mirtlc\_91.27.R, rasch.mirtlc\_aux\_91.12.R, rasch.mml.npirt\_2.03.R,*



*rasch.mml.ramsay\_2.03.R, rasch.mml.raschtype\_2.39.R, rasch.mml\_2.02.R,*  
*rasch.mml2.missing1\_1.11.R, rasch.mml2\_7.13.R, rasch.pairwise.itemcluster\_0.02.R,*  
*rasch.pairwise\_0.14.R, rasch.pml\_2.14.R, rasch.pml\_aux\_1.03.R, rasch.pml2\_4.10.R,*  
*rasch.pml2\_aux\_3.17.R, rasch.pml3\_6.02.R, rasch.pml3\_aux\_5.01.R, rasch.prox\_1.03.R,*  
*rasch.va\_0.01.R, reliability.nonlinear.sem\_1.06.R, rm.facets\_4.11.R,*  
*rm.facets\_alg\_4.11.R, rm.facets\_IC\_0.02.R, rm.facets\_PP\_0.04.R,*  
*rm.hrm.calcprobs\_0.02.R, rm.hrm.est.tau.item\_0.03.R, rm.sdt\_8.24.R,*  
*rm.sdt\_alg\_8.06.R, rm.smooth.distribution\_0.02.R, rm\_proc\_0.03.R, sia.sirt\_0.11.R,*  
*sim.rasch.dep\_0.06.R, sirtcat\_0.02.R, smirt\_7.14.R, smirt\_alg\_comp\_1.05.R,*  
*smirt\_alg\_noncomp\_2.27.R, smirt\_alg\_partcomp\_0.05.R, smirt\_postproc\_0.02.R,*  
*smirt\_preproc\_1.04.R, stratified.cronbach.alpha\_0.02.R, summary.btm\_0.05.R,*  
*summary.fuzcluster\_0.03.R, summary.gom.em\_0.05.R,*  
*summary.invariance.alignment\_0.09.R, summary.isop\_0.04.R,*  
*summary.latent.regression\_0.01.R, summary.lsem.permutationTest\_0.07.R,*  
*summary.lsem\_0.11.R, summary.mcmc.sirt\_1.02.R, summary.noharm.sirt\_1.05.R,*  
*summary.R2noharm.jackknife\_1.01.R, summary.R2noharm\_0.04.R,*  
*summary.rasch.copula\_2.03.R, summary.rasch.evm.pcm\_0.04.R,*  
*summary.rasch.mirtlc\_7.03.R, summary.rasch.mml2\_1.03.R, summary.rasch.pml\_0.06.R,*  
*summary.rm.facets\_0.06.R, summary.rm.sdt\_1.02.R, summary.smirt\_0.06.R,*  
*tam2mirt.aux\_0.03.R, tam2mirt\_0.12.R, testlet.marginalized\_0.04.R,*  
*tetrachoric2\_1.13.R, truescore.irt\_0.12.R, unidim.csn\_0.05.R, wle.rasch\_1.07.R,*  
*yen.q3\_1.01.R, zzz\_1.07.R,*

### **Rcpp Function Versions**

*eigenvaluessirt\_3.06.cpp, evm\_comp\_matrix\_poly\_1.25.cpp, evm\_eigaux\_fcts\_4.07.h,*  
*evm\_eigenvals2\_0.01.h, first\_eigenvalue\_sirt\_2.18.h, gooijer\_isop\_4.02.cpp,*  
*gooijercsntableaux\_1.08.h, invariance\_alignment\_0.03.cpp,*  
*matrixfunctions\_sirt\_1.07.cpp, mle\_pcm\_group\_c\_1.02.cpp,*  
*noharm\_sirt\_auxfunctions\_2.09.cpp, pbivnorm\_rcpp\_aux\_0.51.h,*  
*polychoric2\_tetrachoric2\_rcpp\_aux\_2.03.cpp,*  
*probs\_multicat\_items\_counts\_csirt\_2.03.cpp, rm\_smirt\_mml2\_code\_4.07.cpp,*

### **Rd Documentation Versions**

*automatic.recode\_0.08.Rd, brm.sim\_0.32.Rd, btm\_0.12.Rd, categorize\_0.06.Rd,*  
*ccov.np\_0.09.Rd, class.accuracy.rasch\_0.12.Rd, conf.detect\_1.24.Rd,*  
*data.activity.itempars\_0.04.Rd, data.big5\_0.38.Rd, data.bs\_0.06.Rd,*  
*data.eid\_0.17.Rd, data.ess2005\_0.05.Rd, data.g308\_0.08.Rd, data.inv4gr\_0.04.Rd,*  
*data.liking.science\_0.04.Rd, data.long\_0.21.Rd, data.lsem\_0.02.Rd,*  
*data.math\_0.09.Rd, data.mcdonald\_0.12.Rd, data.mixed1\_0.07.Rd, data.ml\_1.07.Rd,*  
*data.noharm\_2.07.Rd, data.pars1.rasch\_0.08.Rd, data.pirlsmissing\_0.06.Rd,*  
*data.pisaMath\_0.08.Rd, data.pisaPars\_0.06.Rd, data.pisaRead\_0.06.Rd,*  
*data.pw01\_0.03.Rd, data.ratings1\_0.16.Rd, data.raw1\_0.04.Rd, data.read\_1.89.Rd,*  
*data.reck\_0.18.Rd, data.si\_0.21.Rd, data.timss\_0.06.Rd,*  
*data.timss07.G8.RUS\_0.02.Rd, data.wide2long\_0.13.Rd, detect.index\_0.11.Rd,*  
*dif.logistic.regression\_0.19.Rd, dif.strata.variance\_0.06.Rd,*  
*dif.variance\_0.07.Rd, dirichlet.mle\_0.12.Rd, dirichlet.simul\_0.08.Rd,*  
*eigenvalues.manymatrices\_0.08.Rd, eigenvalues.sirt\_0.04.Rd,*

*equating.rasch.jackknife\_0.13.Rd, equating.rasch\_1.22.Rd, expl.detect\_1.07.Rd, fld.irt\_1.16.Rd, fit.isop\_1.10.Rd, fuzcluster\_0.11.Rd, fuzdiscr\_0.11.Rd, gom.em\_1.56.Rd, gom.jml\_0.12.Rd, greenyang.reliability\_1.20.Rd, invariance.alignment\_1.33.Rd, IRT.mle\_0.08.Rd, isop.scoring\_1.16.Rd, isop.test\_0.11.Rd, isop\_3.16.Rd, latent.regression.em.raschtype\_1.28.Rd, lavaan2mirt\_0.33.Rd, lc.2raters\_0.11.Rd, likelihood.adjustment\_0.03.Rd, linking.haberman\_0.33.Rd, linking.robust\_0.14.Rd, lsdm\_2.05.Rd, lsem.estimate\_0.24.Rd, lsem.permutationTest\_0.17.Rd, marginal.truescore.reliability\_0.15.Rd, matrixfunctions.sirt\_1.10.Rd, mcmc.2pno.ml\_0.28.Rd, mcmc.2pno\_1.20.Rd, mcmc.2pnoh\_0.15.Rd, mcmc.3pno.testlet\_1.13.Rd, mcmc.list.descriptives\_0.14.Rd, mcmclist2coda\_0.09.Rd, md.pattern.sirt\_0.10.Rd, mirt.specify.partable\_0.02.Rd, mirt.wrapper\_1.66.Rd, mle.pcm.group\_0.15.Rd, modelfit.sirt\_0.50.Rd, monoreg.rowwise\_0.09.Rd, nedelsky.sim\_0.06.Rd, noharm.sirt\_0.24.Rd, np.dich\_0.12.Rd, pbivnorm2\_0.13.Rd, pcm.conversion\_0.10.Rd, pcm.fit\_0.11.Rd, person.parameter.rasch.copula\_1.09.Rd, personfit.stat\_0.14.Rd, pgenlogis\_0.15.Rd, plausible.value.imputation.raschtype\_1.19.Rd, plot.mcmc.sirt\_0.07.Rd, plot.np.dich\_0.11.Rd, polychoric2\_0.06.Rd, prmse.subscores.scales\_0.15.Rd, prob.guttman\_1.17.Rd, qmc.nodes\_0.07.Rd, R2conquest\_3.11.Rd, R2noharm.EAP\_0.09.Rd, R2noharm.jackknife\_1.08.Rd, R2noharm\_2.19.Rd, rasch.copula\_1.51.Rd, rasch.evm.pcm\_0.30.Rd, rasch.jml.biascorr\_0.14.Rd, rasch.jml.jackknife1\_2.10.Rd, rasch.jml\_1.22.Rd, rasch.mirtlc\_2.80.Rd, rasch.mml\_3.91.Rd, rasch.pairwise.itemcluster\_0.22.Rd, rasch.pairwise\_0.15.Rd, rasch.pml\_2.49.Rd, rasch.prox\_1.09.Rd, rasch.va\_0.06.Rd, reliability.nonlinearSEM\_0.12.Rd, rm.facets\_0.41.Rd, rm.sdt\_1.17.Rd, sia.sirt\_0.11.Rd, sim.qm.ramsay\_0.27.Rd, sim.rasch.dep\_0.18.Rd, sim.raschtype\_0.11.Rd, sirt-package\_2.17.Rd, smirt\_2.16.Rd, stratified.cronbach.alpha\_0.17.Rd, summary.mcmc.sirt\_0.06.Rd, tam2mirt\_0.13.Rd, testlet.marginalized\_0.13.Rd, testlet.yen.q3\_1.03.Rd, tetrachoric2\_1.25.Rd, truescore.irt\_0.12.Rd, unidim.test.csn\_1.11.Rd, wle.rasch.jackknife\_1.14.Rd, wle.rasch\_1.09.Rd, yen.q3\_1.06.Rd,*

### Author(s)

Alexander Robitzsch  
 Federal Institute for Education Research, Innovation and Development of the Austrian School System (BIFIE Salzburg), Austria

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

URL: <https://sites.google.com/site/alexanderrobitzsch/>

### References

- Asparouhov, T., & Muthen, B. (2014). Multiple-group factor analysis alignment. *Structural Equation Modeling*, **21**, 1-14.
- Bartolucci, F. (2007). A class of multidimensional IRT models for testing unidimensionality and clustering items. *Psychometrika*, **72**, 141-157.

- Braeken, J. (2011). A boundary mixture approach to violations of conditional independence. *Psychometrika*, **76**, 57-76.
- DeCarlo, T., Kim, Y., & Johnson, M. S. (2011). A hierarchical rater model for constructed responses, with a signal detection rater model. *Journal of Educational Measurement*, **48**, 333-356.
- Dimitrov, D. (2003). Marginal true-score measures and reliability for binary items as a function of their IRT parameters. *Applied Psychological Measurement*, **27**, 440-458.
- Dimitrov, D. M. (2007). Least squares distance method of cognitive validation and analysis for binary items using their item response theory parameters. *Applied Psychological Measurement*, **31**, 367-387.
- Erosheva, E. A., Fienberg, S. E., & Joutard, C. (2007). Describing disability through individual-level mixture models for multivariate binary data. *Annals of Applied Statistics*, **1**, 502-537.
- Fox, J.-P., & Verhagen, A.-J. (2010). Random item effects modeling for cross-national survey data. In E. Davidov, P. Schmidt, & J. Billiet (Eds.), *Cross-cultural Analysis: Methods and Applications* (pp. 467-488), London: Routledge Academic.
- Glas, C. A. W. (2012). *Estimating and testing the extended testlet model*. LSAC Research Report Series, RR 12-03.
- Green, S.B., & Yang, Y. (2009). Reliability of summed item scores using structural equation modeling: An alternative to coefficient alpha. *Psychometrika*, **74**, 155-167.
- Haberman, S. J. (2009). *Linking parameter estimates derived from an item response model through separate calibrations*. ETS Research Report ETS RR-09-40. Princeton, ETS.
- Ip, E. H., Molenberghs, G., Chen, S. H., Goegebeur, Y., & De Boeck, P. (2013). Functionally unidimensional item response models for multivariate binary data. *Multivariate Behavioral Research*, **48**, 534-562.
- Janssen, R., Tuerlinckx, F., Meulders, M., & de Boeck, P. (2000). A hierarchical IRT model for criterion-referenced measurement. *Journal of Educational and Behavioral Statistics*, **25**, 285-306.
- Linacre, J. M. (1994). *Many-Facet Rasch Measurement*. Chicago: MESA Press.
- Loken, E. & Rulison, K. L. (2010). Estimation of a four-parameter item response theory model. *British Journal of Mathematical and Statistical Psychology*, **63**, 509-525.
- McDonald, R. P. (1997). Normal-ogive multidimensional model. In W. van der Linden & R. K. Hambleton (1997): *Handbook of modern item response theory* (pp. 257-269). New York: Springer.
- Meijer, R. R., & Sijtsma, K. (2001). Methodology review: Evaluating person fit. *Applied Psychological Measurement*, **25**, 107-135.
- Proctor, C. H. (1970). A probabilistic formulation and statistical analysis for Guttman scaling. *Psychometrika*, **35**, 73-78.
- Ramsay, J. O. (1989). A comparison of three simple test theory models. *Psychometrika*, **54**, 487-499.
- Ramsay, J. O. (1991). Kernel smoothing approaches to nonparametric item characteristic curve estimation. *Psychometrika*, **56**, 611-630.
- Reckase, M.D. (2009). *Multidimensional item response theory*. New York: Springer.
- Rijmen, F., & Vomlel, J. (2008). Assessing the performance of variational methods for mixed logistic regression models. *Journal of Statistical Computation and Simulation*, **78**, 765-779.

- Renard, D., Molenberghs, G., & Geys, H. (2004). A pairwise likelihood approach to estimation in multilevel probit models. *Computational Statistics & Data Analysis*, **44**, 649-667.
- Rossi, N., Wang, X. & Ramsay, J. O. (2002). Nonparametric item response function estimates with the EM algorithm. *Journal of Educational and Behavioral Statistics*, **27**, 291-317.
- Rusch, T., Mair, P., & Hatzinger, R. (2013). *Psychometrics with R: A Review of CRAN Packages for Item Response Theory*. <http://epub.wu.ac.at/4010/1/resrepIRThandbook.pdf>.
- Scheiblechner, H. (1995). Isotonic ordinal probabilistic models (ISOP). *Psychometrika*, **60**, 281-304.
- Scheiblechner, H. (1999). Additive conjoint isotonic probabilistic models (ADISOP). *Psychometrika*, **64**, 295-316.
- Stout, W., Habing, B., Douglas, J., & Kim, H. R. (1996). Conditional covariance-based nonparametric multidimensionality assessment. *Applied Psychological Measurement*, **20**, 331-354.
- Stukel, T. A. (1988). Generalized logistic models. *Journal of the American Statistical Association*, **83**, 426-431.
- Uenlue, A., & Yanagida, T. (2011). R you ready for R?: The CRAN psychometrics task view. *British Journal of Mathematical and Statistical Psychology*, **64**, 182-186.
- van den Noortgate, W., De Boeck, P., & Meulders, M. (2003). Cross-classification multilevel logistic models in psychometrics. *Journal of Educational and Behavioral Statistics*, **28**, 369-386.
- Warm, T. A. (1989). Weighted likelihood estimation of ability in item response theory. *Psychometrika*, **54**, 427-450.
- Wainer, H., Bradlow, E. T., & Wang, X. (2007). *Testlet response theory and its applications*. Cambridge: Cambridge University Press.
- Zwinderman, A. H. (1995). Pairwise parameter estimation in Rasch models. *Applied Psychological Measurement*, **19**, 369-375.

### See Also

- For estimating multidimensional models for polytomous item responses see the **mirt**, **flirt** (<http://faculty.psy.ohio-state.edu/jeon/lab/flirt.php>) and **TAM** packages.
- For conditional maximum likelihood estimation see the **eRm** package.
- For pairwise estimation likelihood methods (also known as composite likelihood methods) see **pln** or **lavaan**.
- The estimation of cognitive diagnostic models is possible using the **CDM** package.
- For the multidimensional latent class IRT model see the **MultiLCIRT** package which also allows the estimation IRT models with polytomous item responses.
- Latent class analysis can be carried out with **covLCA**, **poLCA**, **BayesLCA** or **randomLCA** packages.
- Markov Chain Monte Carlo estimation for item response models can also be found in the **MCMC-pack** package (see the **MCMCirt** functions therein).
- See Rusch, Mair and Hatzinger (2013) and Uenlue and Yanagida (2011) for reviews of psychometrics packages in R.

## Examples

```
##
## |-----|
## | sirt 0.40-4 (2013-11-26) |
## | Supplementary Item Response Theory |
## | Maintainer: Alexander Robitzsch <a.robitzsch at bifie.at > |
## | https://sites.google.com/site/alexanderrobitzsch/software |
## |-----|
##
##          _/
##      _/_/_/  _/  _/_/_/  _/_/_/_/_/
##    _/_/      _/  _/_/_/  _/
##      _/_/  _/  _/      _/
##    _/_/_/  _/  _/      _/_/_/
##
```

---

automatic.recode	<i>Automatic Method of Finding Keys in a Dataset with Raw Item Responses</i>
------------------	--

---

## Description

This function calculates keys of a dataset with raw item responses. It starts with setting the most frequent category of an item to 1. Then, in each iteration keys are changed such that the highest item discrimination is found.

## Usage

```
automatic.recode(data, exclude = NULL, pstart.min = 0.6, allocate = 200,
  maxiter = 20, progress = TRUE)
```

## Arguments

data	Dataset with raw item responses
exclude	Vector with categories to be excluded for searching the key
pstart.min	Minimum probability for an initial solution of keys.
allocate	Maximum number of categories per item. This argument is used in the function <code>tam.ctt3</code> of the <b>TAM</b> package.
maxiter	Maximum number of iterations
progress	A logical which indicates if iteration progress should be displayed

## Value

A list with following entries

item.stat	Data frame with item name, p value, item discrimination and the calculated key
data.scored	Scored data frame using calculated keys in <code>item.stat</code>
categ.stats	Data frame with statistics for all categories of all items

**Author(s)**

Alexander Robitzsch

**Examples**

```
## Not run:
#####
# EXAMPLE 1: data.raw1
#####
data(data.raw1)

# recode data.raw1 and exclude keys 8 and 9 (missing codes) and
# start with initially setting all categories larger than 50
res1 <- automatic.recode( data.raw1 , exclude=c(8,9) , pstart.min=.50 )
# inspect calculated keys
res1$item.stat

#####
# EXAMPLE 2: data.timssAusTwn from TAM package
#####

miceadds::library_install("TAM")
data(data.timssAusTwn,package="TAM")
raw.resp <- data.timssAusTwn[,1:11]
res2 <- automatic.recode( data=raw.resp )

## End(Not run)
```

brm-Methods

*Functions for the Beta Item Response Model***Description**

Functions for simulating and estimating the Beta item response model (Noel & Dauvier, 2007). `brm.sim` can be used for simulating the model, `brm.irf` computes the item response function. The Beta item response model is estimated as a discrete version to enable estimation in *standard* IRT software like **mirt** or **TAM** packages.

**Usage**

```
# simulating the beta item response model
brm.sim(theta, delta, tau, K = NULL)

# computing the item response function of the beta item response model
brm.irf( Theta , delta , tau , ncat , thdim=1 , eps=1E-10 )
```

**Arguments**

theta	Ability vector of $\theta$ values
delta	Vector of item difficulty parameters
tau	Vector item dispersion parameters
K	Number of discretized categories. The default is NULL which means that the simulated item responses are real number values between 0 and 1. If an integer K chosen, then values are discretized such that values of 0, 1, ..., K-1 arise.
Theta	Matrix of the ability vector $\theta$
ncat	Number of categories
thdim	Theta dimension in the matrix Theta on which the item loads.
eps	Nuisance parameter which stabilize probabilities.

**Details**

The discrete version of the beta item response model is defined as follows. Assume that for item  $i$  there are  $K$  categories resulting in values  $k = 0, 1, \dots, K - 1$ . Each value  $k$  is associated with a corresponding the transformed value in  $[0, 1]$ , namely  $q(k) = 1/(2 \cdot K), 1/(2 \cdot K) + 1/K, \dots, 1 - 1/(2 \cdot K)$ . The item response model is defined as

$$P(X_{pi} = x_{pi} | \theta_p) \propto q(x_{pi})^{m_{pi}-1} [1 - q(x_{pi})]^{n_{pi}-1}$$

This density is a discrete version of a Beta distribution with shape parameters  $m_{pi}$  and  $n_{pi}$ . These parameters are defined as

$$m_{pi} = \exp[(\theta_p - \delta_i + \tau_i)/2] \quad \text{and} \quad n_{pi} = \exp[(-\theta_p + \delta_i + \tau_i)/2]$$

The item response function can also be formulated as

$$\log [P(X_{pi} = x_{pi} | \theta_p)] \propto (m_{pi} - 1) \cdot \log[q(x_{pi})] + (n_{pi} - 1) \cdot \log[1 - q(x_{pi})]$$

The item parameters can be reparametrized as  $a_i = \exp[(-\delta_i + \tau_i)/2]$  and  $b_i = \exp[(\delta_i + \tau_i)/2]$ . Then, the original item parameters can be retrieved by  $\tau_i = \log(a_i b_i)$  and  $\delta_i = \log(b_i/a_i)$ . Using  $\gamma_p = \exp(\theta_p/2)$ , we obtain

$$\log [P(X_{pi} = x_{pi} | \theta_p)] \propto a_i \gamma_p \cdot \log[q(x_{pi})] + b_i / \gamma_p \cdot \log[1 - q(x_{pi})] - [\log q(x_{pi}) + \log[1 - q(x_{pi})]]$$

This formulation enables the specification of the Beta item response model as a structured latent class model (see [TAM: : tam.mml.3p1](#); Simulated Example 1).

See Smithson and Verkuilen (2006) for motivations for treating continuous indicators not as normally distributed variables.

**Value**

A simulated dataset of item responses if `brm.sim` is applied.

A matrix of item response probabilities if `brm.irf` is applied.

**Author(s)**

Alexander Robitzsch

**References**

- Gruen, B., Kosmidis, I., & Zeileis, A. (2012). Extended Beta regression in R: Shaken, stirred, mixed, and partitioned. *Journal of Statistical Software*, **48(11)**, 1-25.
- Noel, Y., & Dauvier, B. (2007). A beta item response model for continuous bounded responses. *Applied Psychological Measurement*, **31(1)**, 47-73.
- Smithson, M., & Verkuilen, J. (2006). A better lemon squeezer? Maximum-likelihood regression with beta-distributed dependent variables. *Psychological Methods*, **11(1)**, 54-71.

**See Also**

See also the **betareg** package for fitting Beta regression regression models in R (Gruen, Kosmidis & Zeileis, 2012).

**Examples**

```
#####
# SIMULATED EXAMPLE 1: Simulated data beta response model
#####

#### (1) Simulation of the beta response model
# Table 3 (p. 65) of Noel and Dauvier (2007)
delta <- c( -.942 , -.649 , -.603 , -.398 , -.379 , .523 , .649 , .781 , .907 )
tau <- c( .382 , .166 , 1.799 , .615 , 2.092, 1.988 , 1.899 , 1.439 , 1.057 )
K <- 5 # number of categories for discretization
N <- 500      # number of persons
I <- length(delta) # number of items

set.seed(865)
theta <- rnorm( N )
dat <- brm.sim( theta=theta , delta=delta , tau=tau , K=K)
psych::describe(dat)

#### (2) some preliminaries for estimation of the model in mirt
#### define a mirt function
library(mirt)
Theta <- matrix( seq( -4 , 4, len=21) , ncol=1 )

# compute item response function
ii <- 1      # item ii=1
b1 <- brm.irf( Theta=Theta , delta=delta[ii] , tau=tau[ii] , ncat=K )
# plot item response functions
matplot( Theta[,1] , b1 , type="l" )

#### defining the beta item response function for estimation in mirt
par <- c( 0 , 1 , 1)
names(par) <- c( "delta" , "tau" , "thdim")
est <- c( TRUE , TRUE , FALSE )
```



```

names(est) <- names(par)
brm.icc <- function( par , Theta , ncat ){
  delta <- par[1]
  tau <- par[2]
  thdim <- par[3]
  probs <- brm.irf( Theta=Theta , delta=delta , tau=tau , ncat=ncat ,
    thdim=thdim)
  return(probs)
}
name <- "brm"
# create item response function
brm.itemfct <- mirt::createItem(name, par=par, est=est, P=brm.icc)
*** define model in mirt
mirtmodel <- mirt::mirt.model("
  F1 = 1-9
  " )
itemtype <- rep("brm" , I )
customItems <- list("brm"= brm.itemfct)

# define parameters to be estimated
mod1.pars <- mirt::mirt(dat, mirtmodel , itemtype=itemtype ,
  customItems=customItems, pars = "values")

## Not run:
*** (3) estimate beta item response model in mirt
mod1 <- mirt::mirt(dat,mirtmodel , itemtype=itemtype , customItems=customItems,
  pars = mod1.pars , verbose=TRUE )
# model summaries
print(mod1)
summary(mod1)
coef(mod1)
# estimated coefficients and comparison with simulated data
cbind( mirt.wrapper.coef( mod1 )$coef , delta , tau )
mirt.wrapper.itemplot(mod1 ,ask=TRUE)

#-----
# estimate beta item response model in TAM
library(TAM)

# define the skill space: standard normal distribution
TP <- 21 # number of theta points
theta.k <- diag(TP)
theta.vec <- seq( -6 ,6 , len=TP)
d1 <- dnorm(theta.vec)
d1 <- d1 / sum(d1)
delta.designmatrix <- matrix( log(d1) , ncol=1 )
delta.fixed <- cbind( 1 , 1 , 1 )

# define design matrix E
E <- array(0 , dim=c(I,K,TP,2*I + 1) )
dimnames(E)[[1]] <- items <- colnames(dat)
dimnames(E)[[4]] <- c( paste0( rep( items , each=2 ) ,
  rep( c("_a","_b" ) , I ) ) , "one" )

```

```

for (ii in 1:I){
  for (kk in 1:K){
    for (tt in 1:TP){
      qk <- (2*(kk-1)+1)/(2*K)
      gammap <- exp( theta.vec[tt] / 2 )
      E[ii , kk , tt , 2*(ii-1) + 1 ] <- gammap * log( qk )
      E[ii , kk , tt , 2*(ii-1) + 2 ] <- 1 / gammap * log( 1 - qk )
      E[ii , kk , tt , 2*I+1 ] <- - log(qk) - log( 1 - qk )
    }
  }
}
gamma.slope.fixed <- cbind( 2*I+1 , 1 )
gamma.slope <- exp( rep(0,2*I+1) )

# estimate model in TAM
mod2 <- TAM::tam.mml.3pl(resp= dat , E=E ,control= list(maxiter=100) ,
  skill.space="discrete" , delta.designmatrix=delta.designmatrix ,
  delta.fixed=delta.fixed , theta.k=theta.k , gamma.slope = gamma.slope,
  gamma.slope.fixed = gamma.slope.fixed , notA=TRUE )
summary(mod2)

# extract original tau and delta parameters
m1 <- matrix( mod2$gamma.slope[1:(2*I) ] , ncol=2 , byrow=TRUE )
m1 <- as.data.frame(m1)
colnames(m1) <- c("a" , "b")
m1$delta.TAM <- log( m1$b / m1$a )
m1$tau.TAM <- log( m1$a * m1$b )

# compare estimated parameter
m2 <- cbind( mirt.wrapper.coef( mod1 )$coef , delta , tau )[, -1]
colnames(m2) <- c( "delta.mirt" , "tau.mirt" , "thdim" , "delta.true" , "tau.true" )
m2 <- cbind(m1,m2)
round( m2 , 3 )

## End(Not run)

```

---

btm

*Extended Bradley-Terry Model*


---

## Description

Estimates an extended Bradley-Terry model (Hunter, 2004; see Details).

## Usage

```
btm(data, ignore.ties = FALSE, fix.eta = NULL, fix.delta = NULL, fix.theta = NULL,
  maxiter = 100, conv = 1e-04, eps = 0.3)
```

```
## S3 method for class 'btm'
summary(object, file=NULL, digits=4,...)
```

**Arguments**

<code>data</code>	Data frame with three columns. The first two columns contain labels from the units in the pair comparison. The third column contains the result of the comparison. "1" means that the first units wins, "0" means that the second unit wins and "0.5" means a draw (a tie).
<code>ignore.ties</code>	Logical indicating whether ties should be ignored.
<code>fix.eta</code>	Numeric value for a fixed $\eta$ value
<code>fix.delta</code>	Numeric value for a fixed $\delta$ value
<code>fix.theta</code>	A vector with entries for fixed theta values.
<code>maxiter</code>	Maximum number of iterations
<code>conv</code>	Convergence criterion
<code>eps</code>	The $\varepsilon$ parameter for the $\varepsilon$ -adjustment method (see Bertoli-Barsotti & Punzo, 2012) which reduces bias in ability estimates. In case of $\varepsilon = 0$ , persons with extreme scores are removed from the pairwise comparison.
<code>object</code>	Object of class btm
<code>file</code>	Optional file name for sinking the summary into
<code>digits</code>	Number of digits after decimal to print
<code>...</code>	Further arguments to be passed.

**Details**

The extended Bradley-Terry model for the comparison of individuals  $i$  and  $j$  is defined as

$$P(X_{ij} = 1) \propto \exp(\eta + \theta_i)$$

$$P(X_{ij} = 0) \propto \exp(\theta_j)$$

$$P(X_{ij} = 0.5) \propto \exp(\delta + (\eta + \theta_i + \theta_j)/2)$$

The parameters  $\theta_i$  denote the abilities,  $\delta$  is the tendency of the occurrence of ties and  $\eta$  is the home-advantage effect.

**Value**

List with following entries

<code>pars</code>	Parameter summary for $\eta$ and $\delta$
<code>effects</code>	Parameter estimates for $\theta$ and outfit and infit statistics
<code>summary.effects</code>	Summary of $\theta$ parameter estimates
<code>mle.rel</code>	MLE reliability, also known as separation reliability
<code>sepG</code>	Separation index $G$
<code>probs</code>	Estimated probabilities
<code>data</code>	Used dataset with integer identifiers

**Author(s)**

Alexander Robitzsch

**References**

- Bertoli-Barsotti, L., & Punzo, A. (2012). Comparison of two bias reduction techniques for the Rasch model. *Electronic Journal of Applied Statistical Analysis*, **5**, 360-366.
- Hunter, D. R. (2004). MM algorithms for generalized Bradley-Terry models. *Annals of Statistics*, **32**, 384-406.

**See Also**

See also the R packages **BradleyTerry2**, **psychotools**, **psychomix** and **prefmod**.

**Examples**

```
#####
# EXAMPLE 1: Bradley-Terry model | data.pw01
#####

data(data.pw01)

dat <- data.pw01
dat <- dat[ , c("home_team" , "away_team" , "result") ]

# recode results according to needed input
dat$result[ dat$result == 0 ] <- 1/2 # code for ties
dat$result[ dat$result == 2 ] <- 0   # code for victory of away team

#####
# Model 1: Estimation with ties and home advantage
mod1 <- btm( dat)
summary(mod1)

## Not run:
#####
# Model 2: Estimation with ties, no epsilon adjustment
mod2 <- btm( dat , eps=0 , fix.eta=0)
summary(mod2)

#####
# Model 3: Some fixed abilities
fix.theta <- c("Anhalt Dessau" = -1 )
mod3 <- btm( dat , eps=0, fix.theta=fix.theta)
summary(mod3)

#####
# Model 4: Ignoring ties, no home advantage effect
mod4 <- btm( dat , ignore.ties=TRUE , fix.eta = 0)
summary(mod4)
```

```

#####
# Model 5: Ignoring ties, no home advantage effect (JML approach -> eps=0)
mod5 <- btm( dat , ignore.ties=TRUE , fix.eta = 0 , eps=0)
summary(mod5)

#####
# EXAMPLE 2: Venice chess data
#####

# See http://www.rasch.org/rmt/rmt113o.htm
# Linacre, J. M. (1997). Paired Comparisons with Standard Rasch Software.
# Rasch Measurement Transactions, 11:3, 584-585.

# dataset with chess games -> "D" denotes a draw (tie)
chessdata <- scan( what="character")
  1D.0..1...1...1....1.....D.....D.....1.....1.....1..... Browne
  0.1.D..0...1...1....1.....D.....1.....D.....1.....1..... Mariotti
  .D0..0..1...D...D....1.....1.....1.....1.....1.....D..... Tatai
  ...1D1...D...D...1....D.....D.....D.....1.....0..... Hort
  .....010D...D...D....1.....D.....1.....1.....1.....D..... Kavalek
  .....00DDD...D...D....D.....1.....D.....1.....1..... Damjanovic
  .....00D0DD...D.....1.....1.....1.....1.....0.... Gligoric
  .....00D0DD...D.....1.....1.....1.....D.....1... Radulov
  .....DD0DDD0D.....0.....0.....0.....1... Bobotsov
  .....D00D00001.....1.....1.....1.....1..... Cosulich
  .....0D000D0D10.....1.....1.....1.....1..... Westerinen
  .....00D1D010000 Zichichi

L <- length(chessdata) / 2
games <- matrix( chessdata , nrow=L , ncol=2 , byrow=TRUE )
G <- nchar(games[1,1])
# create matrix with results
results <- matrix( NA , nrow=G , ncol=3 )
for (gg in 1:G){
  games.gg <- substring( games[,1] , gg , gg )
  ind.gg <- which( games.gg != "." )
  results[gg , 1:2 ] <- games[ ind.gg , 2]
  results[gg, 3 ] <- games.gg[ ind.gg[1] ]
}
results <- as.data.frame(results)
results[,3] <- paste(results[,3] )
results[ results[,3] == "D" , 3] <- 1/2
results[,3] <- as.numeric( results[,3] )

# fit model ignoring draws
mod1 <- btm( results , ignore.ties=TRUE , fix.eta = 0 , eps=0 )
summary(mod1)

# fit model with draws
mod2 <- btm( results , fix.eta = 0 , eps=0 )
summary(mod2)

## End(Not run)

```

---

`categorize`*Categorize and Decategorize Variables in a Data Frame*

---

**Description**

The function `categorize` defines categories for variables in a data frame, starting with a user-defined index (e.g. 0 or 1). Continuous variables can be categorized by defining categories by discretizing the variables in different quantile groups.

The function `decategorize` does the reverse operation.

**Usage**

```
categorize(dat, categorical = NULL, quant=NULL , lowest = 0)
```

```
decategorize(dat, categ_design = NULL)
```

**Arguments**

<code>dat</code>	Data frame
<code>categorical</code>	Vector with variable names which should be converted into categories, beginning with integer <code>lowest</code>
<code>quant</code>	Vector with number of classes for each variables. Variables are categorized among quantiles. The vector must have names containing variable names.
<code>lowest</code>	Lowest category index. Default is 0.
<code>categ_design</code>	Data frame containing informations about categorization which is the output of <code>categorize</code> .

**Value**

For `categorize`, it is a list with entries

<code>data</code>	Converted data frame
<code>categ_design</code>	Data frame containing some informations about categorization

For `decategorize` it is a data frame.

**Author(s)**

Alexander Robitzsch

**Examples**

```

## Not run:
library(mice)
library(miceadds)

#####
# EXAMPLE 1: Categorize questionnaire data
#####

data(data.smallscale , package="miceadds")
dat <- data.smallscale

# (0) select dataset
dat <- dat[ , 9:20 ]
summary(dat)
categorical <- colnames(dat)
categorical <- categorical[2:6]

# (1) categorize data
res <- categorize( dat , categorical=categorical )

# (2) multiple imputation using the mice package
dat2 <- res$data
VV <- ncol(dat2)
impMethod <- rep( "sample" , VV ) # define random sampling imputation method
names(impMethod) <- colnames(dat2)
imp <- mice::mice( as.matrix(dat2) , impMethod = impMethod , maxit=1 , m=1 )
dat3 <- mice::complete(imp,action=1)

# (3) decategorize dataset
dat3a <- decategorize( dat3 , categ_design = res$categ_design )

#####
# EXAMPLE 2: Categorize ordinal and continuous data
#####

data(data.ma01,package="miceadds")
dat <- data.ma01
summary(dat[, -c(1:2)] )

# define variables to be categorized
categorical <- c("books" , "paredu" )
# define quantiles
quant <- c(6,5,11)
names(quant) <- c("math" , "read" , "hisei")

# categorize data
res <- categorize( dat , categorical = categorical , quant=quant)
str(res)

## End(Not run)

```

---

`ccov.np`*Nonparametric Estimation of Conditional Covariances of Item Pairs*

---

### Description

This function estimates conditional covariances of itempairs (Stout, Habing, Douglas & Kim, 1996; Zhang & Stout, 1999a). The function is used for the estimation of the DETECT index.

### Usage

```
ccov.np(data, score, bwscale = 1.1, thetagrid = seq(-3, 3, len = 200),  
        progress = TRUE)
```

### Arguments

<code>data</code>	An $N \times I$ data frame of dichotomous responses. Missing responses are allowed.
<code>score</code>	An ability estimate, e.g. the WLE
<code>bwscale</code>	Bandwidth factor for calculation of conditional covariance. The bandwidth used in the estimation is <code>bwscale</code> times $N^{-1/5}$ .
<code>thetagrid</code>	A vector which contains theta values where conditional covariances are evaluated.
<code>progress</code>	Display progress?

### Note

This function is used in [conf.detect](#) and [expl.detect](#).

### Author(s)

Alexander Robitzsch

### References

- Stout, W., Habing, B., Douglas, J., & Kim, H. R. (1996). Conditional covariance-based nonparametric multidimensionality assessment. *Applied Psychological Measurement*, **20**, 331-354.
- Zhang, J., & Stout, W. (1999). Conditional covariance structure of generalized compensatory multidimensional items, *Psychometrika*, **64**, 129-152.



---

class.accuracy.rasch *Classification Accuracy in the Rasch Model*

---

### Description

This function computes the classification accuracy in the Rasch model for the maximum likelihood (person parameter) estimate according to the method of Rudner (2001).

### Usage

```
class.accuracy.rasch(cutscores, b, meantheta, sdtheta, theta.l,  
  n.sims=0, seed=988)
```

### Arguments

cutscores	Vector of cut scores
b	Vector of item difficulties
meantheta	Mean of the trait distribution
sdtheta	Standard deviation of the trait distribution
theta.l	Discretized theta distribution
n.sims	Number of simulated persons in a data set. The default is 0 which means that no simulation is performed.
seed	The random seed for the simulation

### Value

A list with following entries:

class.stats	Data frame with classification accuracies. The column <code>agree0</code> refers to absolute agreement, <code>agree1</code> to the agreement of at most a difference of one level.
class.prob	Probability table of classification

### Author(s)

Alexander Robitzsch

### References

Rudner, L.M. (2001). Computing the expected proportions of misclassified examinees. *Practical Assessment, Research & Evaluation*, **7**(14).

### See Also

Classification accuracy of other IRT models can be obtained with the R package **cacIRT**.

## Examples

```
#####
# EXAMPLE 1: Reading dataset
#####
data( data.read , package="sirt")
dat <- data.read

# estimate the Rasch model
mod <- rasch.mml2( dat )

# estimate classification accuracy (3 levels)
cutscores <- c( -1 , .3 ) # cut scores at theta=-1 and theta=.3
class.accuracy.rasch( cutscores=cutscores , b=mod$item$b ,
  meantheta=0 , sdtheta=mod$sd.traits ,
  theta.l=seq(-4,4,len=200 ) , n.sims=3000)
## Cut Scores
## [1] -1.0 0.3
##
## WLE reliability (by simulation) = 0.671
## WLE consistency (correlation between two parallel forms) = 0.649
##
## Classification accuracy and consistency
##          agree0 agree1 kappa consistency
## analytical 0.68 0.990 0.492          NA
## simulated  0.70 0.997 0.489          0.599
##
## Probability classification table
##          Est_Class1 Est_Class2 Est_Class3
## True_Class1    0.136    0.041    0.001
## True_Class2    0.081    0.249    0.093
## True_Class3    0.009    0.095    0.294
```

---

conf.detect

*Confirmatory DETECT and polyDETECT Analysis*

---

## Description

This function computes the DETECT statistics for dichotomous item responses and the polyDETECT statistic for polytomous item responses under a confirmatory specification of item clusters (Stout, Habing, Douglas & Kim, 1996; Zhang & Stout, 1999a, 1999b; Zhang, 2007). Item responses in a multi-matrix design are allowed (Zhang, 2013).

## Usage

```
conf.detect(data, score, itemcluster, bwscale = 1.1, progress = TRUE,
  thetagrid = seq(-3, 3, len = 200))
```

**Arguments**

data	An $N \times I$ data frame of dichotomous or polytomous responses. Missing responses are allowed.
score	An ability estimate, e.g. the WLE
itemcluster	Item cluster for each item. The order of entries must correspond to the columns in data.
bwscale	Bandwidth factor for calculation of conditional covariance (see <a href="#">ccov.np</a> )
progress	Display progress?
thetagrid	A vector which contains theta values where conditional covariances are evaluated.

**Details**

The result of DETECT are the indices DETECT, ASSI and RATIO (see Zhang 2007 for details) calculated for the options unweighted and weighted. The option unweighted means that all conditional covariances of item pairs are equally weighted, weighted means that these covariances are weighted by the sample size of item pairs. In case of multi matrix item designs, both types of indices can differ.

The classification scheme of these indices are as follows (Jang & Roussos, 2007; Zhang, 2007):

Strong multidimensionality	DETECT > 1.00
Moderate multidimensionality	.40 < DETECT < 1.00
Weak multidimensionality	.20 < DETECT < .40
Essential unidimensionality	DETECT < .20

Maximum value under simple structure	ASSI=1	RATIO=1
Essential deviation from unidimensionality	ASSI > .25	RATIO > .36
Essential unidimensionality	ASSI < .25	RATIO < .36

**Value**

A list with following entries:

detect	Data frame with statistics DETECT, ASSI and RATIO
ccovtable	Individual contributions to conditional covariance
ccov.matrix	Evaluated conditional covariance

**Author(s)**

Alexander Robitzsch

## References

- Jang, E. E., & Roussos, L. (2007). An investigation into the dimensionality of TOEFL using conditional covariance-based nonparametric approach. *Journal of Educational Measurement*, **44**, 1-21.
- Stout, W., Habing, B., Douglas, J., & Kim, H. R. (1996). Conditional covariance-based nonparametric multidimensionality assessment. *Applied Psychological Measurement*, **20**, 331-354.
- Zhang, J., & Stout, W. (1999a). Conditional covariance structure of generalized compensatory multidimensional items. *Psychometrika*, **64**, 129-152.
- Zhang, J., & Stout, W. (1999b). The theoretical DETECT index of dimensionality and its application to approximate simple structure. *Psychometrika*, **64**, 213-249.
- Zhang, J. (2007). Conditional covariance theory and DETECT for polytomous items. *Psychometrika*, **72**, 69-91.
- Zhang, J. (2013). A procedure for dimensionality analyses of response data from various test designs. *Psychometrika*, **78**, 37-58.

## See Also

For a download of the free *DIM-Pack* software (DIMTEST, DETECT) see <http://psychometrictools.measuredprogress.org/home>.

## Examples

```
#####
# EXAMPLE 1: TIMSS mathematics data set (dichotomous data)
#####
data(data.timss)

# extract data
dat <- data.timss$data
dat <- dat[ , substring( colnames(dat),1,1) == "M" ]
# extract item informations
iteminfo <- data.timss$item
# estimate Rasch model
mod1 <- rasch.mml2( dat )
# estimate WLEs
wle1 <- wle.rasch( dat , b = mod1$item$b )$theta

# DETECT for content domains
detect1 <- conf.detect( data = dat , score = wle1 ,
                      itemcluster = iteminfo$Content.Domain )
##           unweighted weighted
## DETECT      0.316    0.316
## ASSI        0.273    0.273
## RATIO       0.355    0.355

## Not run:
# DETECT cognitive domains
detect2 <- conf.detect( data = dat , score = wle1 ,
                      itemcluster = iteminfo$Cognitive.Domain )
##           unweighted weighted
```

```

## DETECT      0.251    0.251
## ASSI       0.227    0.227
## RATIO     0.282    0.282

# DETECT for item format
detect3 <- conf.detect( data = dat , score = wle1 ,
                       itemcluster = iteminfo$Format )
##           unweighted weighted
## DETECT    0.056    0.056
## ASSI     0.060    0.060
## RATIO    0.062    0.062

# DETECT for item blocks
detect4 <- conf.detect( data = dat , score = wle1 ,
                       itemcluster = iteminfo$Block )
##           unweighted weighted
## DETECT    0.301    0.301
## ASSI     0.193    0.193
## RATIO    0.339    0.339
## End(Not run)

# Exploratory DETECT: Application of a cluster analysis employing the Ward method
detect5 <- expl.detect( data = dat , score = wle1 ,
                       nclusters = 10 , N.est = nrow(dat) )
# Plot cluster solution
pl <- plot( detect5$clusterfit , main = "Cluster solution" )
rect.hclust(detect5$clusterfit, k=4, border="red")

## Not run:
#####
# EXAMPLE 2: Big 5 data set (polytomous data)
#####

# attach Big5 Dataset
data(data.big5)

# select 6 items of each dimension
dat <- data.big5
dat <- dat[, 1:30]

# estimate person score by simply using a transformed sum score
score <- qnorm( ( rowMeans( dat )+.5 ) / ( 30 + 1 ) )

# extract item cluster (Big 5 dimensions)
itemcluster <- substring( colnames(dat) , 1 , 1 )

# DETECT Item cluster
detect1 <- conf.detect( data = dat , score = score , itemcluster = itemcluster )
##           unweighted weighted
## DETECT    1.256    1.256
## ASSI     0.384    0.384
## RATIO    0.597    0.597

```

```

# Exploratory DETECT
detect5 <- expl.detect( data = dat , score = score ,
                      nclusters = 9 , N.est = nrow(dat) )
## DETECT (unweighted)
## Optimal Cluster Size is 6 (Maximum of DETECT Index)
##   N.Cluster N.items N.est N.val   size.cluster DETECT.est ASSI.est RATIO.est
## 1         2      30  500   0       6-24      1.073   0.246   0.510
## 2         3      30  500   0      6-10-14    1.578   0.457   0.750
## 3         4      30  500   0      6-10-11-3  1.532   0.444   0.729
## 4         5      30  500   0      6-8-11-2-3  1.591   0.462   0.757
## 5         6      30  500   0      6-8-6-2-5-3  1.610   0.499   0.766
## 6         7      30  500   0      6-3-6-2-5-5-3  1.557   0.476   0.740
## 7         8      30  500   0      6-3-3-2-3-5-5-3  1.540   0.462   0.732
## 8         9      30  500   0      6-3-3-2-3-5-3-3-2  1.522   0.444   0.724

# Plot Cluster solution
pl <- plot( detect5$clusterfit , main = "Cluster solution" )
rect.hclust(detect5$clusterfit, k=6, border="red")

## End(Not run)

```

---

data.activity.itempars

*Item Parameters Cultural Activities*

---

### Description

List with item parameters for cultural activities of Austrian students for 9 Austrian countries.

### Usage

```
data(data.activity.itempars)
```

### Format

The format is a list with number of students per group (N), item loadings (lambda) and item intercepts (nu):

```

List of 3
 $ N      : 'table' int [1:9(1d)] 2580 5279 15131 14692 5525 11005 7080 4112 14274
 ..- attr(*, "dimnames")=List of 1
 .. ..$ : chr [1:9] "1" "2" "3" "4" ...
 $ lambda: num [1:9, 1:5] 0.423 0.485 0.455 0.437 0.502 ...
 ..- attr(*, "dimnames")=List of 2
 .. ..$ : chr [1:9] "country1" "country2" "country3" "country4" ...
 .. ..$ : chr [1:5] "act1" "act2" "act3" "act4" ...
 $ nu    : num [1:9, 1:5] 1.65 1.53 1.7 1.59 1.7 ...
 ..- attr(*, "dimnames")=List of 2
 .. ..$ : chr [1:9] "country1" "country2" "country3" "country4" ...
 .. ..$ : chr [1:5] "act1" "act2" "act3" "act4" ...

```

**Example Index**

[invariance.alignment](#) (Example 1)

---

data.big5

*Dataset Big 5 from **qgraph** Package*

---

**Description**

This is a Big 5 dataset from the **qgraph** package (Dolen, Oorts, Stoel, Wicherts, 2009). It contains 500 subjects on 240 items.

**Usage**

```
data(data.big5)
data(data.big5.qgraph)
```

**Format**

- The format of data.big5 is:
 

```
num [1:500, 1:240] 1 0 0 0 0 1 1 2 0 1 ...
- attr(*, "dimnames")=List of 2
..$ : NULL
..$ : chr [1:240] "N1" "E2" "O3" "A4" ...
```
- The format of data.big5.qgraph is:
 

```
num [1:500, 1:240] 2 3 4 4 5 2 2 1 4 2 ...
- attr(*, "dimnames")=List of 2
..$ : NULL
..$ : chr [1:240] "N1" "E2" "O3" "A4" ...
```

**Details**

In these datasets, there exist 48 items for each dimension. The Big 5 dimensions are Neuroticism (N), Extraversion (E), Openness (O), Agreeableness (A) and Conscientiousness (C). Note that the data.big5 differs from data.big5.qgraph in a way that original items were recoded into three categories 0,1 and 2.

**Source**

See big5 in **qgraph** package.

**References**

Dolan, C. V., Oort, F. J., Stoel, R. D., & Wicherts, J. M. (2009). Testing measurement invariance in the target rotates multigroup exploratory factor model. *Structural Equation Modeling*, **16**, 295-314.

## Examples

```
## Not run:
# list of needed packages for the following examples
packages <- scan(what="character")
  sirt TAM eRm CDM mirt ltm mokken psychotools psychomix
  psych

# load packages. make an installation if necessary
miceadds::library_install(packages)

#####
# EXAMPLE 1: Unidimensional models openness scale
#####

data(data.big5)
# extract first 10 openness items
items <- which( substrings( colnames(data.big5) , 1 , 1 ) == "0" )[1:10]
dat <- data.big5[ , items ]
I <- ncol(dat)
summary(dat)
  ## > colnames(dat)
  ## [1] "03" "08" "013" "018" "023" "028" "033" "038" "043" "048"
# descriptive statistics
psych::describe(dat)

#####
# Model 1: Partial credit model
#####

#-- M1a: rm.facets (in sirt)
m1a <- sirt::rm.facets( dat )
summary(m1a)

#-- M1b: tam.mml (in TAM)
m1b <- TAM::tam.mml( resp=dat )
summary(m1b)

#-- M1c: gdm (in CDM)
theta.k <- seq(-6,6,len=21)
m1c <- CDM::gdm( dat , irtmodel="1PL" ,theta.k=theta.k , skillspace="normal")
summary(m1c)
# compare results with loglinear skillspace
m1c2 <- CDM::gdm( dat , irtmodel="1PL" ,theta.k=theta.k , skillspace="loglinear")
summary(m1c2)

#-- M1d: PCM (in eRm)
m1d <- eRm::PCM( dat )
summary(m1d)

#-- M1e: gpcm (in ltm)
m1e <- ltm::gpcm( dat , constraint = "1PL" , control=list(verbose=TRUE))
summary(m1e)
```



```

#-- M1f: mirt (in mirt)
m1f <- mirt::mirt( dat , model=1 , itemtype="1PL" , verbose=TRUE)
summary(m1f)
coef(m1f)

#-- M1g: PCModel.fit (in psychotools)
mod1g <- psychotools::PCModel.fit(dat)
summary(mod1g)
plot(mod1g)

#*****
# Model 2: Generalized partial credit model
#*****

#-- M2a: rm.facets (in sirt)
m2a <- sirt::rm.facets( dat , est.a.item=TRUE)
summary(m2a)
# Note that in rm.facets the mean of item discriminations is fixed to 1

#-- M2b: tam.mml.2pl (in TAM)
m2b <- TAM::tam.mml.2pl( resp=dat , irtmodel="GPCM")
summary(m2b)

#-- M2c: gdm (in CDM)
m2c <- CDM::gdm( dat , irtmodel="2PL" , theta.k=seq(-6,6,len=21) ,
                skillspace="normal" , standardized.latent=TRUE)
summary(m2c)

#-- M2d: gpcm (in ltm)
m2d <- ltm::gpcm( dat , control=list(verbose=TRUE))
summary(m2d)

#-- M2e: mirt (in mirt)
m2e <- mirt::mirt( dat , model=1 , itemtype="GPCM" , verbose=TRUE)
summary(m2e)
coef(m2e)

#*****
# Model 3: Nonparametric item response model
#*****

#-- M3a: ISOP and ADISOP model - isop.poly (in sirt)
m3a <- sirt::isop.poly( dat )
summary(m3a)
plot(m3a)

#-- M3b: Mokken scale analysis (in mokken)
# Scalability coefficients
mokken::coefH(dat)
# Assumption of monotonicity
monotonicity.list <- mokken::check.monotonicity(dat)
summary(monotonicity.list)

```

```

plot(monotonicity.list)
# Assumption of non-intersecting ISRFs using method restscore
restscore.list <- mokken::check.restscore(dat)
summary(restscore.list)
plot(restscore.list)

#####
# Model 4: Graded response model
#####

#-- M4a: mirt (in mirt)
m4a <- mirt::mirt( dat , model=1 , itemtype="graded" , verbose=TRUE)
print(m4a)
mirt.wrapper.coef(m4a)

#---- M4b: WLSMV estimation with cfa (in lavaan)
lavmodel <- "F =~ 03__048
            F ~~ 1*F
            "

# transform lavaan syntax with lavaanify.IRT
lavmodel <- TAM::lavaanify.IRT( lavmodel , items=colnames(dat) )$lavaan.syntax
mod4b <- lavaan::cfa( data= as.data.frame(dat) , model=lavmodel, std.lv = TRUE,
                    ordered=colnames(dat) , parameterization="theta")
summary(mod4b , standardized=TRUE , fit.measures=TRUE , rsquare=TRUE)
coef(mod4b)

#####
# Model 5: Normally distributed residuals
#####

#---- M5a: cfa (in lavaan)
lavmodel <- "F =~ 03__048
            F ~~ 1*F
            F ~ 0*1
            03__048 ~ 1
            "

lavmodel <- TAM::lavaanify.IRT( lavmodel , items=colnames(dat) )$lavaan.syntax
mod5a <- lavaan::cfa( data= as.data.frame(dat) , model=lavmodel, std.lv = TRUE ,
                    estimator="MLR" )
summary(mod5a , standardized=TRUE , fit.measures=TRUE , rsquare=TRUE)

#---- M5b: mirt (in mirt)

# create user defined function
name <- 'normal'
par <- c("d" = 1 , "a1" = 0.8 , "vy" = 1)
est <- c(TRUE, TRUE,FALSE)
P.normal <- function(par,Theta,ncat){
  d <- par[1]
  a1 <- par[2]
  vy <- par[3]
  psi <- vy - a1^2
  # expected values given Theta

```

```

    mui <- a1*Theta[,1] + d
    TP <- nrow(Theta)
    probs <- matrix( NA , nrow=TP, ncol= ncat )
    eps <- .01
    for (cc in 1:ncat){
      probs[,cc] <- dnorm( cc , mean = mui , sd = sqrt( abs( psi + eps) ) )
    }
    psum <- matrix( rep(rowSums( probs ),each=ncat) , nrow=TP , ncol=ncat , byrow=TRUE)
    probs <- probs / psum
    return(probs)
  }

# create item response function
normal <- mirt::createItem(name, par=par, est=est, P=P.normal)
customItems <- list("normal"=normal)
itemtype <- rep( "normal",I)
# define parameters to be estimated
mod5b.pars <- mirt::mirt(dat, 1, itemtype=itemtype ,
                        customItems=customItems , pars = "values")
ind <- which( mod5b.pars$name == "vy")
vy <- apply( dat , 2 , var , na.rm=TRUE )
mod5b.pars[ ind , "value" ] <- vy
ind <- which( mod5b.pars$name == "a1")
mod5b.pars[ ind , "value" ] <- .5* sqrt(vy)
ind <- which( mod5b.pars$name == "d")
mod5b.pars[ ind , "value" ] <- colMeans( dat , na.rm=TRUE )

# estimate model
mod5b <- mirt::mirt(dat, 1, itemtype=itemtype , customItems=customItems ,
                  pars = mod5b.pars , verbose=TRUE )
sirt::mirt.wrapper.coef(mod5b)$coef

# some item plots
par(ask=TRUE)
plot(mod5b, type = 'trace', layout = c(1,1))
par(ask=FALSE)
# Alternatively:
sirt::mirt.wrapper.itemplot(mod5b)

## End(Not run)

```

---

data.bs

*Datasets from Borg and Staufenbiel (2007)*


---

## Description

Datasets of the book of Borg and Staufenbiel (2007) *Lehrbuch Theorien and Methoden der Skalierung*.

## Usage

```
data(data.bs07a)
```

## Format

- The dataset `data.bs07a` contains the data *Gefechtsangst* (p. 130) and contains 8 of the original 9 items. The items are symptoms of anxiety in engagement.  
GF1: starkes Herzklopfen, GF2: flaes Gefuehl in der Magengegend, GF3: Schwaechegefuehl, GF4: Uebelkeitsgefuehl, GF5: Erbrechen, GF6: Schuettelfrost, GF7: in die Hose urinieren/einkoten, GF9: Gefuehl der Gelaehmtheit

The format is

```
'data.frame':  100 obs. of  9 variables:
 $ idpatt: int  44 29 1 3 28 50 50 36 37 25 ...
 $ GF1   : int  1 1 1 1 1 0 0 1 1 1 ...
 $ GF2   : int  0 1 1 1 1 0 0 1 1 1 ...
 $ GF3   : int  0 0 1 1 0 0 0 0 0 1 ...
 $ GF4   : int  0 0 1 1 0 0 0 1 0 1 ...
 $ GF5   : int  0 0 1 1 0 0 0 0 0 0 ...
 $ GF6   : int  1 1 1 1 1 0 0 0 0 0 ...
 $ GF7   : num  0 0 1 1 0 0 0 0 0 0 ...
 $ GF9   : int  0 0 1 1 1 0 0 0 0 0 ...
```

- MORE DATASETS*

## References

Borg, I., & Staufenbiel, T. (2007). *Lehrbuch Theorie und Methoden der Skalierung*. Bern: Hogrefe.

## Examples

```
## Not run:
#####
# EXAMPLE 07a: Dataset Gefechtsangst
#####

data(data.bs07a)
dat <- data.bs07a
items <- grep( "GF" , colnames(dat) , value=TRUE )

#####
# Model 1: Rasch model
mod1 <- TAM::tam.mml(dat[,items] )
summary(mod1)
IRT.WrightMap(mod1)

#####
# Model 2: 2PL model
mod2 <- TAM::tam.mml.2pl(dat[,items] )
summary(mod2)

#####
# Model 3: Latent class analysis (LCA) with two classes
tammodel <- "
ANALYSIS:
```

```

    TYPE=LCA;
    NCLASSES(2)
    NSTARTS(5,10)
LAVAAN MODEL:
    F =~ GF1__GF9
    "
mod3 <- TAM::tamaan( tammodel , dat )
summary(mod3)

#*****
# Model 4: LCA with three classes
tammodel <- "
ANALYSIS:
    TYPE=LCA;
    NCLASSES(3)
    NSTARTS(5,10)
LAVAAN MODEL:
    F =~ GF1__GF9
    "
mod4 <- TAM::tamaan( tammodel , dat )
summary(mod4)

#*****
# Model 5: Located latent class model (LOCLCA) with two classes
tammodel <- "
ANALYSIS:
    TYPE=LOCLCA;
    NCLASSES(2)
    NSTARTS(5,10)
LAVAAN MODEL:
    F =~ GF1__GF9
    "
mod5 <- TAM::tamaan( tammodel , dat )
summary(mod5)

#*****
# Model 6: Located latent class model with three classes
tammodel <- "
ANALYSIS:
    TYPE=LOCLCA;
    NCLASSES(3)
    NSTARTS(5,10)
LAVAAN MODEL:
    F =~ GF1__GF9
    "
mod6 <- TAM::tamaan( tammodel , dat )
summary(mod6)

#*****
# Model 7: Probabilistic Guttman model
mod7 <- sirt::prob.guttman( dat[,items] )
summary(mod7)

```

```

#-- model comparison
IRT.compareModels( mod1, mod2 , mod3 , mod4 , mod5 , mod6 , mod7 )

## End(Not run)

```

---

data.eid

*Examples with Datasets from Eid and Schmidt (2014)*


---

## Description

Examples with datasets from Eid and Schmidt (2014), illustrations with several R packages. The examples follow closely the online material of Hosoya (2014).

## Usage

```
data(data.eid)
```

## Format

The dataset data.eid is just a placeholder.

## Source

The material can be downloaded from <http://www.hogrefe.de/buecher/lehrbuecher/psychlehrbuchplus/lehrbuecher/testtheorie-und-testkonstruktion/zusatzmaterial/>.

## References

Eid, M., & Schmidt, K. (2014). *Testtheorie und Testkonstruktion*. Goettingen, Hogrefe.

Hosoya, G. (2014). *Einfuehrung in die Analyse testtheoretischer Modelle mit R*. Available at <http://www.hogrefe.de/buecher/lehrbuecher/psychlehrbuchplus/lehrbuecher/testtheorie-und-testkonstruktion/zusatzmaterial/>.

## Examples

```

## Not run:
# The "dataset" data.eid is just a placeholder.
site <- paste0( "http://www.hogrefe.de/fileadmin/redakteure/hogrefe_de/" ,
               "Psychlehrbuchplus/Testtheorie_und_Testkonstruktion/R-Analysen/" )

miceadds::library_install("foreign")
#---- load some IRT packages in R
miceadds::library_install("TAM")           # package (a)
miceadds::library_install("mirt")         # package (b)
miceadds::library_install("sirt")         # package (c)
miceadds::library_install("eRm")         # package (d)
miceadds::library_install("ltm")         # package (e)
miceadds::library_install("psychomix")   # package (f)

```

```
#####
# EXAMPLES Ch. 4: Unidimensional IRT models | dichotomous data
#####

# link to dataset
linkname <- paste0( site , "ids_new.sav")
# load data
data0 <- foreign::read.spss( linkname , to.data.frame=TRUE , use.value.labels=FALSE)
# extract items
dat <- data0[,2:11]

#####
# Model 1: Rasch model
#####

#-----
#-- 1a: estimation with TAM package

# estimation with tam.mml
mod1a <- TAM::tam.mml(dat)
summary(mod1a)

# person parameters in TAM
pp1a <- TAM::tam.wle(mod1a)

# plot item response functions
plot(mod1a,export=FALSE,ask=TRUE)

# Infit and outfit in TAM
itemf1a <- TAM::tam.fit(mod1a)
itemf1a

# model fit
modf1a <- TAM::tam.modelfit(mod1a)
summary(modf1a)

#-----
#-- 1b: estimation with mirt package

# estimation with mirt
mod1b <- mirt::mirt( dat , 1 , itemtype="Rasch")
summary(mod1b)
print(mod1b)

# person parameters
pp1b <- mirt::fscores(mod1b , method="WLE")

# extract coefficients
sirt::mirt.wrapper.coef(mod1b)

# plot item response functions
plot(mod1b, type="trace" )
par(mfrow=c(1,1))
```

```
# item fit
itemf1b <- mirt::itemfit(mod1b)
itemf1b

# model fit
modf1b <- mirt::M2(mod1b)
modf1b

#-----
#-- 1c: estimation with sirt package

# estimation with rasch.mml2
mod1c <- rasch.mml2(dat)
summary(mod1c)

# person parameters (EAP)
pp1c <- mod1c$person

# plot item response functions
plot(mod1c , ask=TRUE )

# model fit
modf1c <- sirt::modelfit.sirt(mod1c)
summary(modf1c)

#-----
#-- 1d: estimation with eRm package

# estimation with RM
mod1d <- eRm::RM(dat)
summary(mod1d)

# estimation person parameters
pp1d <- eRm::person.parameter(mod1d)
summary(pp1d)

# plot item response functions
eRm::plotICC(mod1d)

# person-item map
eRm::plotPImap(mod1d)

# item fit
itemf1d <- eRm::itemfit(pp1d)

# person fit
persf1d <- eRm::personfit(pp1d)

#-----
#-- 1e: estimation with ltm package

# estimation with rasch
```



```
mod1e <- ltm::rasch(dat)
summary(mod1e)

# estimation person parameters
pp1e <- ltm::factor.scores(mod1e)

# plot item response functions
plot(mod1e)

# item fit
itemf1e <- ltm::item.fit(mod1e)

# person fit
persf1e <- ltm::person.fit(mod1e)

# goodness of fit with Bootstrap
modf1e <- ltm::GoF.rasch(mod1e,B=20) # use more bootstrap samples
modf1e

#####
# Model 2: 2PL model
#####

#-----
#-- 2a: estimation with TAM package

# estimation
mod2a <- TAM::tam.mml.2pl(dat)
summary(mod2a)

# model fit
modf2a <- TAM::tam.modelfit(mod2a)
summary(modf2a)

# item response functions
plot(mod2a , export=FALSE , ask=TRUE)

# model comparison
anova(mod1a,mod2a)

#-----
#-- 2b: estimation with mirt package

# estimation
mod2b <- mirt::mirt(dat,1,itemtype="2PL")
summary(mod2b)
print(mod2b)
sirt::mirt.wrapper.coef(mod2b)

# model fit
modf2b <- mirt::M2(mod2b)
modf2b
```

```

#-----
#-- 2c: estimation with sirt package

I <- ncol(dat)
# estimation
mod2c <- sirt::rasch.mml2(dat,est.a=1:I)
summary(mod2c)

# model fit
modf2c <- sirt::modelfit.sirt(mod2c)
summary(modf2c)

#-----
#-- 2e: estimation with ltm package

# estimation
mod2e <- ltm::ltm(dat ~ z1 )
summary(mod2e)

# item response functions
plot(mod2e)

#*****
# Model 3: Mixture Rasch model
#*****

#-----
#-- 3a: estimation with TAM package

# avoid "_" in column names if the "__" operator is used in
# the tamaan syntax
dat1 <- dat
colnames(dat1) <- gsub("_", "__", colnames(dat1) )
# define tamaan model
tammodel <- "
ANALYSIS:
  TYPE=MIXTURE ;
  NCLASSES(2);
  NSTARTS(20,25); # 20 random starts with 25 initial iterations each
LAVAN MODEL:
  F =~ Freude1__Freude2
  F ~~ F
ITEM TYPE:
  ALL(Rasch);
  "

mod3a <- TAM::tamaan( tammodel , resp=dat1 )
summary(mod3a)
# extract item parameters
ipars <- mod2$itempartable_MIXTURE[ 1:10 , ]
plot( 1:10 , ipars[,3] , type="o" , ylim= range( ipars[,3:4] ) , pch=16 ,
      xlab="Item" , ylab="Item difficulty")
lines( 1:10 , ipars[,4] , type="l" , col=2 , lty=2)
points( 1:10 , ipars[,4] , col=2 , pch=2)

```

```

#-----
#-- 3f: estimation with psychomix package

# estimation
mod3f <- psychomix::raschmix( as.matrix(dat) , k=2 , scores="meanvar")
summary(mod3f)
# plot class-specific item difficulties
plot(mod3f)

#####
# EXAMPLES Ch. 5: Unidimensional IRT models | polytomous data
#####

# link to dataset
linkname <- paste0( site , "Daten-kapitel-5-sex.sav")
# load data
data0 <- foreign::read.spss( linkname , to.data.frame=TRUE , use.value.labels=FALSE)
# extract items
dat <- data0[,2:7]

*****
# Model 1: Partial credit model
*****

#-----
#-- 1a: estimation with TAM package

# estimation with tam.mml
mod1a <- TAM::tam.mml(dat)
summary(mod1a)

# person parameters in TAM
pp1a <- tam.wle(mod1a)

# plot item response functions
plot(mod1a,export=FALSE,ask=TRUE)

# Infit and outfit in TAM
itemf1a <- TAM::tam.fit(mod1a)
itemf1a

# model fit
modf1a <- TAM::tam.modelfit(mod1a)
summary(modf1a)

#-----
#-- 1b: estimation with mirt package

# estimation with tam.mml
mod1b <- mirt::mirt( dat , 1 , itemtype="Rasch")
summary(mod1b)
print(mod1b)

```

```

sirt::mirt.wrapper.coef(mod1b)

# plot item response functions
plot(mod1b, type="trace" )
par(mfrow=c(1,1))

# item fit
itemf1b <- mirt::itemfit(mod1b)
itemf1b

#-----
#-- 1c: estimation with sirt package

# estimation with rm.facets
mod1c <- sirt::rm.facets(dat)
summary(mod1c)
summary(mod1a)

#-----
#-- 1d: estimation with eRm package

# estimation
mod1d <- eRm::PCM(dat)
summary(mod1d)

# plot item response functions
eRm::plotICC(mod1d)

# person-item map
eRm::plotPImap(mod1d)

# item fit
itemf1d <- eRm::itemfit(pp1d)

#-----
#-- 1e: estimation with ltm package

# estimation
mod1e <- ltm::gpcm(dat, constraint="1PL")
summary(mod1e)
# plot item response functions
plot(mod1e)

#*****
# Model 2: Generalized partial credit model
#*****

#-----
#-- 2a: estimation with TAM package

# estimation with tam.mml
mod2a <- TAM::tam.mml.2pl(dat, irtmodel="GPCM")
summary(mod2a)

```

```

# model fit
modf2a <- TAM::tam.modelfit(mod2a)
summary(modf2a)

#-----
#-- 2b: estimation with mirt package

# estimation
mod2b <- mirt::mirt( dat , 1 , itemtype="gpcm")
summary(mod2b)
print(mod2b)
sirt::mirt.wrapper.coef(mod2b)

#-----
#-- 2c: estimation with sirt package

# estimation with rm.facets
mod2c <- sirt::rm.facets(dat , est.a.item=TRUE)
summary(mod2c)

#-----
#-- 2e: estimation with ltm package

# estimation
mod2e <- ltm::gpcm(dat)
summary(mod2e)
plot(mod2e)

## End(Not run)

```

---

data.ess2005

*Dataset European Social Survey 2005*


---

### Description

This dataset contains item loadings  $\lambda$  and intercepts  $\nu$  for 26 countries for the European Social Survey (ESS 2005; see Asparouhov & Muthen, 2014).

### Usage

```
data(data.ess2005)
```

### Format

The format of the dataset is:

```

List of 2
 $ lambda: num [1:26, 1:4] 0.688 0.721 0.72 0.687 0.625 ...
 .. attr(*, "dimnames")=List of 2

```

```

.. ..$ : NULL
.. ..$ : chr [1:4] "ipfrule" "ipmodst" "ipbhprp" "imptrad"
$ nu    : num [1:26, 1:4] 3.26 2.52 3.41 2.84 2.79 ...
..- attr(*, "dimnames")=List of 2
.. ..$ : NULL
.. ..$ : chr [1:4] "ipfrule" "ipmodst" "ipbhprp" "imptrad"

```

## References

Asparouhov, T., & Muthen, B. (2014). Multiple-group factor analysis alignment. *Structural Equation Modeling*, **21**, 1-14.

---

data.g308

*C-Test Datasets*

---

## Description

Some datasets of C-tests are provided. The dataset data.g308 was used in Schroeders, Robitzsch and Schipolowski (2014).

## Usage

```
data(data.g308)
```

## Format

- The dataset data.g308 is a C-test containing 20 items and is used in Schroeders, Robitzsch and Schipolowski (2014) and is of the following format

```

'data.frame': 747 obs. of 21 variables:
 $ id      : int  1 2 3 4 5 6 7 8 9 10 ...
 $ G30801: int  1 1 1 1 1 0 0 1 1 1 ...
 $ G30802: int  1 1 1 1 1 1 1 1 1 1 ...
 $ G30803: int  1 1 1 1 1 1 1 1 1 1 ...
 $ G30804: int  1 1 1 1 1 0 1 1 1 1 ...
 [...]
 $ G30817: int  0 0 0 0 1 0 1 0 1 0 ...
 $ G30818: int  0 0 1 0 0 0 0 1 1 0 ...
 $ G30819: int  1 1 1 1 0 0 1 1 1 0 ...
 $ G30820: int  1 1 1 1 0 0 0 1 1 0 ...

```

## References

Schroeders, U., Robitzsch, A., & Schipolowski, S. (2014). A comparison of different psychometric approaches to modeling testlet structures: An example with C-tests. *Journal of Educational Measurement*, **51**(4), 400-418.

## Examples

```

## Not run:
#####
# EXAMPLE 1: Dataset G308 from Schroeders et al. (2014)
#####

data(data.g308)
dat <- data.g308

library(TAM)
library(sirt)
library(combinat)

# define testlets
testlet <- c(1, 1, 2, 2, 2, 2, 2, 2, 3, 3, 4, 4, 4, 4, 4, 5, 5, 6, 6, 6)

#####
*** Model 1: Rasch model
mod1 <- TAM::tam.mml(resp = dat[,-1], pid = dat[,1],
  control = list(maxiter = 300 , snodes = 1500))
summary(mod1)

#####
*** Model 2: Rasch testlet model

# testlets are dimensions, assign items to Q-matrix
TT <- length(unique(testlet))
Q <- matrix(0, nrow = ncol(dat)-1 , ncol = TT + 1)
Q[,1] <- 1 # First dimension constitutes g-factor
for (tt in 1:TT){Q[testlet == tt, tt+1] <- 1}

# In a testlet model, all dimensions are uncorrelated among
# each other, that is, all pairwise correlations are set to 0,
# which can be accomplished with the "variance.fixed" command
variance.fixed <- cbind(t( combinat::combn(TT+1,2)), 0)
mod2 <- TAM::tam.mml(resp = dat[,-1], pid = dat[,1], Q = Q,
  variance.fixed = variance.fixed,
  control = list(snodes = 1500 , maxiter = 300))
summary(mod2)

#####
*** Model 3: Partial credit model

scores <- list()
testlet.names <- NULL
dat.pcm <- NULL
for (tt in 1:max(testlet) ){
  scores[[tt]] <- rowSums (dat[,-1][, testlet == tt, drop = FALSE])
  dat.pcm <- c(dat.pcm, list(c(scores[[tt]])))
  testlet.names <- append(testlet.names, paste0("testlet",tt) )
}
dat.pcm <- as.data.frame(dat.pcm)

```

```
colnames(dat.pcm) <- testlet.names
mod3 <- TAM::tam.mml(resp = dat.pcm, control = list(snodes=1500, maxiter=300) )
summary(mod3)

#*****
#*** Model 4: Copula model

mod4 <- sirt::rasch.copula2 (dat = dat[,-1], itemcluster = testlet)
summary(mod4)

## End(Not run)
```

---

data.inv4gr

*Dataset for Invariance Testing with 4 Groups*

---

### **Description**

Dataset for invariance testing with 4 groups.

### **Usage**

```
data(data.inv4gr)
```

### **Format**

A data frame with 4000 observations on the following 12 variables. The first variable is a group identifier, the other variables are items.

group A group identifier

I01 a numeric vector

I02 a numeric vector

I03 a numeric vector

I04 a numeric vector

I05 a numeric vector

I06 a numeric vector

I07 a numeric vector

I08 a numeric vector

I09 a numeric vector

I10 a numeric vector

I11 a numeric vector

### **Source**

Simulated dataset



---

data.liking.science     *Dataset 'Liking For Science'*

---

**Description**

Dataset 'Liking for science' published by Wright and Masters (1982).

**Usage**

```
data(data.liking.science)
```

**Format**

The format is:

```
num [1:75, 1:24] 1 2 2 1 1 1 2 2 0 2 ...
- attr(*, "dimnames")=List of 2
..$ : NULL
..$ : chr [1:24] "LS01" "LS02" "LS03" "LS04" ...
```

**References**

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

---

data.long                     *Longitudinal Dataset*

---

**Description**

This dataset contains 200 observations on 12 items. 6 items (I1T1, ... ,I6T1) were administered at measurement occasion T1 and 6 items at T2 (I3T2, ... , I8T2). There were 4 anchor items which were presented at both time points. The first column in the dataset contains the student identifier.

**Usage**

```
data(data.long)
```

**Format**

The format of the dataset is

```
'data.frame': 200 obs. of 13 variables:
 $ idstud: int 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 ...
 $ I1T1 : int 1 1 1 1 1 1 1 0 1 1 ...
 $ I2T1 : int 0 0 1 1 1 1 0 1 1 1 ...
 $ I3T1 : int 1 0 1 1 0 1 0 0 0 0 ...
```

```

$ I4T1 : int 1 0 0 1 0 0 0 0 1 1 ...
$ I5T1 : int 1 0 0 1 0 0 0 0 1 0 ...
$ I6T1 : int 1 0 0 0 0 0 0 0 0 0 ...
$ I3T2 : int 1 1 0 0 1 1 1 1 0 1 ...
$ I4T2 : int 1 1 0 0 1 1 0 0 0 1 ...
$ I5T2 : int 1 0 1 1 1 1 1 0 1 1 ...
$ I6T2 : int 1 1 0 0 0 0 0 0 0 1 ...
$ I7T2 : int 1 0 0 0 0 0 0 0 0 1 ...
$ I8T2 : int 0 0 0 0 1 0 0 0 0 0 ...

```

## Examples

```

## Not run:
data(data.long)
dat <- data.long
dat <- dat[,-1]
I <- ncol(dat)

#####
# Model 1: 2-dimensional Rasch model
#####
# define Q-matrix
Q <- matrix(0,I,2)
Q[1:6,1] <- 1
Q[7:12,2] <- 1
rownames(Q) <- colnames(dat)
colnames(Q) <- c("T1","T2")

# vector with same items
itemnr <- as.numeric( substring( colnames(dat) ,2,2) )
# fix mean at T2 to zero
mu.fixed <- cbind( 2,0 )

#--- M1a: rasch.mml2 (in sirt)
mod1a <- sirt::rasch.mml2(dat, Q=Q , est.b=itemnr , mu.fixed=mu.fixed)
summary(mod1a)

#--- M1b: smirt (in sirt)
mod1b <- sirt::smirt(dat, Qmatrix=Q , irtmodel="comp" , est.b= itemnr ,
                    mu.fixed=mu.fixed )

#--- M1c: tam.mml (in TAM)

# assume equal item difficulty of I3T1 and I3T2, I4T1 and I4T2, ...
# create draft design matrix and modify it
A <- TAM::designMatrices(resp=dat)$A
dimnames(A)[[1]] <- colnames(dat)
## > str(A)
## num [1:12, 1:2, 1:12] 0 0 0 0 0 0 0 0 0 0 0 0 ...
## - attr(*, "dimnames")=List of 3
## ..$ : 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 ) ]
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
dimnames(A1)[[3]] <- substring( dimnames(A1)[[3]],1,2)
##      > 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 )
mod1c <- TAM::tam.mml( resp=dat , Q=Q , A=A1 , beta.fixed=beta.fixed)
summary(mod1c)

#####
# Model 2: 2-dimensional 2PL model
#####

# set variance at T2 to 1
variance.fixed <- cbind(2,2,1)

# M2a: rasch.mml2 (in sirt)
mod2a <- sirt::rasch.mml2(dat, Q=Q , est.b=itemnr , est.a=itemnr , mu.fixed=mu.fixed,
      variance.fixed=variance.fixed , mmliter=100)
summary(mod2a)

#####
# Model 3: Concurrent calibration by assuming invariant item parameters
#####

library(mirt) # use mirt for concurrent calibration
data(data.long)
dat <- data.long[,-1]
I <- ncol(dat)

# create user defined function for between item dimensionality 4PL model
name <- "4PLbw"
par <- c("low"=0,"upp"=1,"a"=1,"d"=0 , "dimItem"=1)
est <- c(TRUE, TRUE,TRUE,TRUE,FALSE)

```

```

# item response function
irf <- function(par,Theta,ncat){
  low <- par[1]
  upp <- par[2]
  a <- par[3]
  d <- par[4]
  dimItem <- par[5]
  P1 <- low + ( upp - low ) * plogis( a*Theta[,dimItem] + d )
  cbind(1-P1, P1)
}

# create item response function
fourPLbetw <- mirt::createItem(name, par=par, est=est, P=irf)
head(dat)

# create mirt model (use variable names in mirt.model)
mirtsyn <- "
  T1 = I1T1,I2T1,I3T1,I4T1,I5T1,I6T1
  T2 = I3T2,I4T2,I5T2,I6T2,I7T2,I8T2
  COV = T1*T2,,T2*T2
  MEAN = T1
  CONSTRAIN = (I3T1,I3T2,d),(I4T1,I4T2,d),(I5T1,I5T2,d),(I6T1,I6T2,d),
              (I3T1,I3T2,a),(I4T1,I4T2,a),(I5T1,I5T2,a),(I6T1,I6T2,a)
  "

# create mirt model
mirtmodel <- mirt::mirt.model( mirtsyn , itemnames=colnames(dat) )
# define parameters to be estimated
mod3.pars <- mirt::mirt(dat, mirtmodel$model, rep( "4PLbw",I) ,
                      customItems=list("4PLbw"=fourPLbetw), pars = "values")

# select dimensions
ind <- intersect( grep("T2",mod3.pars$item) , which( mod3.pars$name == "dimItem" ) )
mod3.pars[ind,"value"] <- 2
# set item parameters low and upp to non-estimated
ind <- which( mod3.pars$name %in% c("low","upp") )
mod3.pars[ind,"est"] <- FALSE

# estimate 2PL model
mod3 <- mirt::mirt(dat, mirtmodel$model, itemtype=rep( "4PLbw",I) ,
                  customItems=list("4PLbw"=fourPLbetw), pars = mod3.pars , verbose=TRUE ,
                  technical = list(NCYCLES=50) )
mirt.wrapper.coef(mod3)

#***** estimate model in lavaan
library(lavaan)

# specify syntax
lavamodel <- "
  #**** T1
  F1 =~ a1*I1T1+a2*I2T1+a3*I3T1+a4*I4T1+a5*I5T1+a6*I6T1
  I1T1 | b1*t1 ; I2T1 | b2*t1 ; I3T1 | b3*t1 ; I4T1 | b4*t1
  I5T1 | b5*t1 ; I6T1 | b6*t1
  F1 =~ 1*F1
  #**** T2

```

```

F2 =~ a3*I3T2+a4*I4T2+a5*I5T2+a6*I6T2+a7*I7T2+a8*I8T2
I3T2 | b3*t1 ; I4T2 | b4*t1 ; I5T2 | b5*t1 ; I6T2 | b6*t1
I7T2 | b7*t1 ; I8T2 | b8*t1
F2 ~~ NA*F2
F2 ~ 1
*** covariance
F1 ~~ F2
"

# estimate model using theta parameterization
mod3lav <- lavaan::cfa( data=dat , model=lavmodel,
  std.lv = TRUE , ordered=colnames(dat) , parameterization="theta")
summary(mod3lav , standardized=TRUE , fit.measures=TRUE , rsquare=TRUE)

*****
# Model 4: Linking with items of different item slope groups
*****

data(data.long)
dat <- data.long
# dataset for T1
dat1 <- dat[ , grep( "T1" , colnames(dat) ) ]
colnames(dat1) <- gsub("T1","" , colnames(dat1) )
# dataset for T2
dat2 <- dat[ , grep( "T2" , colnames(dat) ) ]
colnames(dat2) <- gsub("T2","" , colnames(dat2) )

# 2PL model with slope groups T1
mod1 <- rasch.mml2( dat1 , est.a = c( rep(1,2) , rep(2,4) ) )
summary(mod1)

# 2PL model with slope groups T2
mod2 <- rasch.mml2( dat2 , est.a = c( rep(1,4) , rep(2,2) ) )
summary(mod2)

#----- Link 1: Haberman Linking
# collect item parameters
dfr1 <- data.frame( "study1" , mod1$item$item , mod1$item$a , mod1$item$b )
dfr2 <- data.frame( "study2" , mod2$item$item , mod2$item$a , mod2$item$b )
colnames(dfr2) <- colnames(dfr1) <- c("study" , "item" , "a" , "b" )
itempars <- rbind( dfr1 , dfr2 )
# Linking
link1 <- linking.haberman(itempars=itempars)

#----- Link 2: Invariance alignment method
# create objects for invariance.alignment
nu <- rbind( c(mod1$item$thresh,NA,NA) , c(NA,NA,mod2$item$thresh) )
lambda <- rbind( c(mod1$item$a,NA,NA) , c(NA,NA,mod2$item$a) )
colnames(lambda) <- colnames(nu) <- paste0("I",1:8)
rownames(lambda) <- rownames(nu) <- c("T1" , "T2")
# Linking
link2a <- invariance.alignment( lambda , nu )
summary(link2a)

```

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

---

data.lsem	<i>Datasets for Local Structural Equation Models / Moderated Factor Analysis</i>
-----------	--

---

### Description

Datasets for local structural equation models or moderated factor analysis.

### Usage

```
data(data.lsem01)
```

### Format

- The dataset data.lsem01 has the following structure

```
'data.frame':  989 obs. of  6 variables:
 $ age: num  4 4 4 4 4 4 4 4 4 4 ...
 $ v1 : num  1.83 2.38 1.85 4.53 -0.04 4.35 2.38 1.83 4.81 2.82 ...
 $ v2 : num  6.06 9.08 7.41 8.24 6.18 7.4 6.54 4.28 6.43 7.6 ...
 $ v3 : num  1.42 3.05 6.42 -1.05 -1.79 4.06 -0.17 -2.64 0.84 6.42 ...
 $ v4 : num  3.84 4.24 3.24 3.36 2.31 6.07 4 5.93 4.4 3.49 ...
 $ v5 : num  7.84 7.51 6.62 8.02 7.12 7.99 7.25 7.62 7.66 7.03 ...
```

---

data.math	<i>Dataset Mathematics</i>
-----------	----------------------------

---

### Description

This is an example Mathematics dataset. The dataset contains 664 students on 30 items.

### Usage

```
data(data.math)
```

**Format**

The dataset is a list. The list element `data` contains the dataset with the demographical variables `student ID (idstud)` and a dummy variable for female students (`female`). The remaining variables (starting with `M` in the name) are the mathematics items.

The item metadata are included in the list element `item` which contains item name (`item`) and the testlet label (`testlet`). An item not included in a testlet is indicated by `NA`. Each item is allocated to one and only competence domain (`domain`).

The format is:

List of 2

```
$ data:'data.frame':
..$ idstud: int [1:664] 1001 1002 1003 ...
..$ female: int [1:664] 1 1 0 0 1 1 1 0 0 1 ...
..$ MA1   : int [1:664] 1 1 1 0 0 1 1 1 1 1 ...
..$ MA2   : int [1:664] 1 1 1 1 1 0 0 0 0 1 ...
..$ MA3   : int [1:664] 1 1 0 0 0 0 0 1 0 0 ...
..$ MA4   : int [1:664] 0 1 1 1 0 0 1 0 0 0 ...
..$ MB1   : int [1:664] 0 1 0 1 0 0 0 0 0 1 ...
..$ MB2   : int [1:664] 1 1 1 1 0 1 0 1 0 0 ...
..$ MB3   : int [1:664] 1 1 1 1 0 0 0 1 0 1 ...
[...]
..$ MH3   : int [1:664] 1 1 0 1 0 0 1 0 1 0 ...
..$ MH4   : int [1:664] 0 1 1 1 0 0 0 0 1 0 ...
..$ MI1   : int [1:664] 1 1 0 1 0 1 0 0 1 0 ...
..$ MI2   : int [1:664] 1 1 0 0 0 1 1 0 1 1 ...
..$ MI3   : int [1:664] 0 1 0 1 0 0 0 0 0 0 ...
$ item:'data.frame':
..$ item    : Factor w/ 30 levels "MA1","MA2","MA3",...: 1 2 3 4 5 ...
..$ testlet : Factor w/ 9 levels "", "MA", "MB", "MC",...: 2 2 2 3 3 ...
..$ domain  : Factor w/ 3 levels "arithmetic","geometry",...: 1 1 1 ...
..$ subdomain: Factor w/ 9 levels "", "addition",...: 2 2 2 2 7 7 ...
```

---

data.mcdonald

*Some Datasets from McDonald's Test Theory Book*

---

**Description**

Some datasets from McDonald (1999), especially related to using NOHARM for item response modelling. See Examples below.

**Usage**

```
data(data.mcdonald.act15)
data(data.mcdonald.LSAT6)
data(data.mcdonald.rape)
```

**Format**

- The format of the ACT15 data `data.mcdonald.act15` is:
 

```
num [1:15, 1:15] 0.49 0.44 0.38 0.3 0.29 0.13 0.23 0.16 0.16 0.23 ...
- attr(*, "dimnames")=List of 2
..$ : chr [1:15] "A01" "A02" "A03" "A04" ...
..$ : chr [1:15] "A01" "A02" "A03" "A04" ...
```

 The dataset (which is the product-moment covariance matrix) is obtained from Ch. 12 in McDonald (1999).

- The format of the LSAT6 data `data.mcdonald.LSAT6` is:

```
'data.frame': 1004 obs. of 5 variables:
 $ L1: int 0 0 0 0 0 0 0 0 0 0 ...
 $ L2: int 0 0 0 0 0 0 0 0 0 0 ...
 $ L3: int 0 0 0 0 0 0 0 0 0 0 ...
 $ L4: int 0 0 0 0 0 0 0 0 1 ...
 $ L5: int 0 0 0 1 1 1 1 1 1 0 ...
```

The dataset is obtained from Ch. 6 in McDonald (1999).

- The format of the rape myth scale data `data.mcdonald.rape` is

```
List of 2
 $ lambda: num [1:2, 1:19] 1.13 0.88 0.85 0.77 0.79 0.55 1.12 1.01 0.99 0.79 ...
.- attr(*, "dimnames")=List of 2
.. ..$ : chr [1:2] "male" "female"
.. ..$ : chr [1:19] "I1" "I2" "I3" "I4" ...
 $ nu : num [1:2, 1:19] 2.88 1.87 3.12 2.32 2.13 1.43 3.79 2.6 3.01 2.11 ...
.- attr(*, "dimnames")=List of 2
.. ..$ : chr [1:2] "male" "female"
.. ..$ : chr [1:19] "I1" "I2" "I3" "I4" ...
```

The dataset is obtained from Ch. 15 in McDonald (1999).

**Source**

Tables in McDonald (1999)

**References**

McDonald, R. P. (1999). *Test theory: A unified treatment*. Psychology Press.

**Examples**

```
## Not run:
#####
# EXAMPLE 1: LSAT6 data | Chapter 12 McDonald (1999)
#####
data(data.mcdonald.act15)

#*****
```



```

# Model 1: 2-parameter normal ogive model

### NOHARM estimation
I <- ncol(dat)
# covariance structure
P.pattern <- matrix( 0 , ncol=1 , nrow=1 )
P.init <- 1+0*P.pattern
# fix all entries in the loading matrix to 1
F.pattern <- matrix( 1 , nrow=I , ncol=1 )
F.init <- F.pattern
# estimate model
mod1a <- sirt::R2noharm( dat = dat , model.type="CFA" , F.pattern = F.pattern ,
                        F.init = F.init , P.pattern = P.pattern , P.init = P.init ,
                        writename = "LSAT6__1dim_2pno" , noharm.path = noharm.path , dec = "," )
summary(mod1a , logfile="LSAT6__1dim_2pno__SUMMARY")

### pairwise marginal maximum likelihood estimation using the probit link
mod1b <- sirt::rasch.pml3( dat , est.a=1:I , est.sigma=FALSE )

#####
# Model 2: 1-parameter normal ogive model

### NOHARM estimation
# covariance structure
P.pattern <- matrix( 0 , ncol=1 , nrow=1 )
P.init <- 1+0*P.pattern
# fix all entries in the loading matrix to 1
F.pattern <- matrix( 2 , nrow=I , ncol=1 )
F.init <- 1+0*F.pattern
# estimate model
mod2a <- sirt::R2noharm( dat = dat , model.type="CFA" , F.pattern = F.pattern ,
                        F.init = F.init , P.pattern = P.pattern , P.init = P.init ,
                        writename = "LSAT6__1dim_1pno" , noharm.path = noharm.path , dec = "," )
summary(mod2a , logfile="LSAT6__1dim_1pno__SUMMARY")

# PMML estimation
mod2b <- sirt::rasch.pml3( dat , est.a=rep(1,I) , est.sigma=FALSE )
summary(mod2b)

#####
# Model 3: 3-parameter normal ogive model with fixed guessing parameters

### NOHARM estimation
# covariance structure
P.pattern <- matrix( 0 , ncol=1 , nrow=1 )
P.init <- 1+0*P.pattern
# fix all entries in the loading matrix to 1
F.pattern <- matrix( 1 , nrow=I , ncol=1 )
F.init <- 1+0*F.pattern
# estimate model
mod <- sirt::R2noharm( dat = dat , model.type="CFA" , guesses=rep(.2,I) ,
                      F.pattern = F.pattern , F.init = F.init , P.pattern = P.pattern ,
                      P.init = P.init , writename = "LSAT6__1dim_3pno" ,

```

```

        noharm.path = noharm.path , dec = "," )
summary(mod , logfile="LSAT6__1dim_3pno__SUMMARY")

##++ logistic link function employed in smirt function
mod1d <- sirt::smirt(dat, Qmatrix=F.pattern, est.a= matrix(1:I,I,1), c.init=rep(.2,I))
summary(mod1d)

#####
# EXAMPLE 2: ACT15 data | Chapter 6 McDonald (1999)
#####
data(data.mcdonald.act15)
pm <- data.mcdonald.act15

#####
# Model 1: 2-dimensional exploratory factor analysis
mod1 <- sirt::R2noharm( pm=pm , n=1000, model.type="EFA" , dimensions=2 ,
                      writename = "ACT15__efa_2dim" , noharm.path = noharm.path , dec = "," )
summary(mod1)

#####
# Model 2: 2-dimensional independent clusters basis solution
P.pattern <- matrix(1,2,2)
diag(P.pattern) <- 0
P.init <- 1+0*P.pattern
F.pattern <- matrix(0,15,2)
F.pattern[ c(1:5,11:15),1] <- 1
F.pattern[ c(6:10,11:15),2] <- 1
F.init <- F.pattern

# estimate model
mod2 <- sirt::R2noharm( pm=pm , n=1000 , model.type="CFA" , F.pattern = F.pattern ,
                      F.init = F.init , P.pattern = P.pattern ,P.init = P.init ,
                      writename = "ACT15_indep_clusters" , noharm.path = noharm.path , dec = "," )
summary(mod2)

#####
# Model 3: Hierarchical model

P.pattern <- matrix(0,3,3)
P.init <- P.pattern
diag(P.init) <- 1
F.pattern <- matrix(0,15,3)
F.pattern[,1] <- 1 # all items load on g factor
F.pattern[ c(1:5,11:15),2] <- 1 # Items 1-5 and 11-15 load on first nested factor
F.pattern[ c(6:10,11:15),3] <- 1 # Items 6-10 and 11-15 load on second nested factor
F.init <- F.pattern

# estimate model
mod3 <- sirt::R2noharm( pm=pm , n=1000 , model.type="CFA" , F.pattern = F.pattern ,
                      F.init = F.init , P.pattern = P.pattern , P.init = P.init ,
                      writename = "ACT15_hierarch_model" , noharm.path = noharm.path , dec = "," )
summary(mod3)

```

```
#####
# EXAMPLE 3: Rape myth scale | Chapter 15 McDonald (1999)
#####
data(data.mcdonald.rape)
lambda <- data.mcdonald.rape$lambda
nu <- data.mcdonald.rape$nu

# obtain multiplier for factor loadings (Formula 15.5)
k <- sum( lambda[1,] * lambda[2,] ) / sum( lambda[2,]^2 )
## [1] 1.263243

# additive parameter (Formula 15.7)
c <- sum( lambda[2,]*(nu[1,]-nu[2,]) ) / sum( lambda[2,]^2 )
## [1] 1.247697

# SD in the female group
1/k
## [1] 0.7916132

# M in the female group
- c/k
## [1] -0.9876932

# Burt's coefficient of factorial congruence (Formula 15.10a)
sum( lambda[1,] * lambda[2,] ) / sqrt( sum( lambda[1,]^2 ) * sum( lambda[2,]^2 ) )
## [1] 0.9727831

# congruence for mean parameters
sum( (nu[1,]-nu[2,]) * lambda[2,] ) / sqrt( sum( (nu[1,]-nu[2,])^2 ) * sum( lambda[2,]^2 ) )
## [1] 0.968176

## End(Not run)
```

---

data.mixed1

*Dataset with Mixed Dichotomous and Polytomous Item Responses*


---

### Description

Dataset with mixed dichotomous and polytomous item responses.

### Usage

```
data(data.mixed1)
```

### Format

A data frame with 1000 observations on the following 37 variables.

```
'data.frame': 1000 obs. of 37 variables:
 $ I01: num 1 1 1 1 1 1 1 0 1 1 ...
```

```
$ I02: num 1 1 1 1 1 1 1 1 0 1 ...
[...]
```

```
$ I36: num 1 1 1 1 0 0 0 0 1 1 ...
```

```
$ I37: num 0 1 1 1 0 1 0 0 1 1 ...
```

## Examples

```
data(data.mixed1)
apply( data.mixed1 , 2 , max )
## I01 I02 I03 I04 I05 I06 I07 I08 I09 I10 I11 I12 I13 I14 I15 I16
## 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## I17 I18 I19 I20 I21 I22 I23 I24 I25 I26 I27 I28 I29 I30 I31 I32
## 1 1 1 1 4 4 1 1 1 1 1 1 1 1 1 1
## I33 I34 I35 I36 I37
## 1 1 1 1 1
```

---

data.ml

*Multilevel Datasets*

---

## Description

Datasets for conducting multilevel IRT analysis. This dataset is used in the examples of the function [mcmc.2pno.ml](#).

## Usage

```
data(data.ml1)
data(data.ml2)
```

## Format

- data.ml1

A data frame with 2000 student observations in 100 classes on 17 variables. The first variable group contains the class identifier. The remaining 16 variables are dichotomous test items.

```
'data.frame': 2000 obs. of 17 variables:
```

```
$ group: num 1001 1001 1001 1001 1001 ...
```

```
$ X1 : num 1 1 1 1 1 1 1 1 1 1 ...
```

```
$ X2 : num 1 1 1 0 1 1 1 1 1 1 ...
```

```
$ X3 : num 0 1 1 0 1 0 1 0 1 0 ...
```

```
$ X4 : num 1 1 1 0 0 1 1 1 1 1 ...
```

```
$ X5 : num 0 0 0 1 1 1 0 0 1 1 ...
```

```
[...]
```

```
$ X16 : num 0 0 1 0 0 0 1 0 0 0 ...
```

- data.m12

A data frame with 2000 student observations in 100 classes on 6 variables. The first variable group contains the class identifier. The remaining 5 variables are polytomous test items.

```
'data.frame':  2000 obs. of  6 variables:
 $ group: num  1 1 1 1 1 1 1 1 1 1 ...
 $ X1  : num  2 3 4 3 3 3 1 4 4 3 ...
 $ X2  : num  2 2 4 3 3 2 2 3 4 3 ...
 $ X3  : num  3 4 5 4 2 3 3 4 4 2 ...
 $ X4  : num  2 3 3 2 1 3 1 4 4 3 ...
 $ X5  : num  2 3 3 2 3 3 1 3 2 2 ...
```

---

data.noharm

*Datasets for NOHARM Analysis*

---

## Description

Datasets for analyses in NOHARM (see [R2noharm](#)).

## Usage

```
data(data.noharmExC)
data(data.noharm18)
```

## Format

- data.noharmExC

The format of this dataset is

```
'data.frame':  300 obs. of  8 variables:
 $ C1: int  1 1 1 1 1 0 1 1 1 1 ...
 $ C2: int  1 1 1 1 0 1 1 1 1 1 ...
 $ C3: int  1 1 1 1 1 0 0 0 1 1 ...
 $ C4: int  0 0 1 1 1 1 1 0 1 0 ...
 $ C5: int  1 1 1 1 1 0 0 1 1 0 ...
 $ C6: int  1 0 0 0 1 0 1 1 0 1 ...
 $ C7: int  1 1 0 0 1 1 0 0 0 1 ...
 $ C8: int  1 0 1 0 1 0 1 0 1 1 ...
```

- data.noharm18

A data frame with 200 observations on the following 18 variables I01, ..., I18. The format is

```
'data.frame':  200 obs. of  18 variables:
 $ I01: int  1 1 1 1 1 0 1 1 0 1 ...
 $ I02: int  1 1 0 1 1 0 1 1 1 1 ...
 $ I03: int  1 0 0 1 0 0 1 1 0 1 ...
```

```

$I04: int 0 1 0 1 0 0 0 1 1 1 ...
$I05: int 1 0 0 0 1 0 1 1 0 1 ...
$I06: int 1 1 0 1 0 0 1 1 0 1 ...
$I07: int 1 1 1 1 0 1 1 1 1 1 ...
$I08: int 1 1 1 1 1 1 1 1 0 1 ...
$I09: int 1 1 1 1 0 0 1 1 0 1 ...
$I10: int 1 0 0 1 1 0 1 1 0 1 ...
$I11: int 1 1 1 1 0 0 1 1 0 1 ...
$I12: int 0 0 0 0 0 1 0 0 0 0 ...
$I13: int 1 1 1 1 0 1 1 0 1 1 ...
$I14: int 1 1 1 0 1 0 1 1 0 1 ...
$I15: int 1 1 1 0 0 1 1 1 0 1 ...
$I16: int 1 1 0 1 1 0 1 0 1 1 ...
$I17: int 0 1 0 0 0 0 1 1 0 1 ...
$I18: int 0 0 0 0 0 0 0 0 1 0 ...

```

---

data.pars1.rasch	<i>Item Parameters for Three Studies Obtained by 1PL and 2PL Estimation</i>
------------------	---

---

### Description

The datasets contain item parameters to be prepared for linking using the function [linking.haberman](#).

### Usage

```

data(data.pars1.rasch)
data(data.pars1.2pl)

```

### Format

- The format of data.pars1.rasch is:
 

```

'data.frame': 22 obs. of 4 variables:
 $ study: chr "study1" "study1" "study1" "study1" ...
 $ item : Factor w/ 12 levels "M133","M176",...: 1 2 3 4 5 1 6 7 3 8 ...
 $ a : num 1 1 1 1 1 1 1 1 1 1 ...
 $ b : num -1.5862 0.40762 1.78031 2.00382 0.00862 ...

```

 Item slopes a are fixed to 1 in 1PL estimation. Item difficulties are denoted by b.
- The format of data.pars1.2pl is:
 

```

'data.frame': 22 obs. of 4 variables:
 $ study: chr "study1" "study1" "study1" "study1" ...
 $ item : Factor w/ 12 levels "M133","M176",...: 1 2 3 4 5 1 6 7 3 8 ...
 $ a : num 1.238 0.957 1.83 1.927 2.298 ...
 $ b : num -1.16607 0.35844 1.06571 1.17159 0.00792 ...

```

---

data.pirlsmissing      *Dataset from PIRLS Study with Missing Responses*

---

## Description

This is a dataset of the PIRLS 2011 study for 4th graders for the reading booklet 13 (the 'PIRLS reader') and 4 countries (Austria, Germany, France, Netherlands). Missing responses (missing by intention and not reached) are coded by 9.

## Usage

```
data(data.pirlsmissing)
```

## Format

A data frame with 3480 observations on the following 38 variables.

The format is:

```
'data.frame': 3480 obs. of 38 variables:
 $ idstud : int 1000001 1000002 1000003 1000004 1000005 ...
 $ country : Factor w/ 4 levels "AUT","DEU","FRA",...: 1 1 1 1 1 1 1 1 1 1 ...
 $ studwgt : num 1.06 1.06 1.06 1.06 1.06 ...
 $ R31G01M : int 1 1 1 1 1 1 0 1 1 0 ...
 $ R31G02C : int 0 9 0 1 0 0 0 0 1 0 ...
 $ R31G03M : int 1 1 1 1 0 1 0 0 1 1 ...
 [...]
 $ R31P15C : int 1 9 0 1 0 0 0 0 1 0 ...
 $ R31P16C : int 0 0 0 0 0 0 0 9 0 1 ...
```

## Examples

```
data(data.pirlsmissing)
# inspect missing rates
round( colMeans( data.pirlsmissing==9 ) , 3 )
##   idstud  country  studwgt  R31G01M  R31G02C  R31G03M  R31G04C  R31G05M
##   0.000   0.000   0.000   0.009   0.076   0.012   0.203   0.018
##  R31G06M  R31G07M  R31G08CZ  R31G08CA  R31G08CB  R31G09M  R31G10C  R31G11M
##   0.010   0.020   0.189   0.225   0.252   0.019   0.126   0.023
##  R31G12C  R31G13CZ  R31G13CA  R31G13CB  R31G13CC  R31G14M  R31P01M  R31P02C
##   0.202   0.170   0.198   0.220   0.223   0.074   0.013   0.039
##  R31P03C  R31P04M  R31P05C  R31P06C  R31P07C  R31P08M  R31P09C  R31P10M
##   0.056   0.012   0.075   0.043   0.074   0.024   0.062   0.025
##  R31P11M  R31P12M  R31P13M  R31P14C  R31P15C  R31P16C
##   0.027   0.030   0.030   0.126   0.130   0.127
```

---

 data.pisaMath

 Dataset PISA Mathematics
 

---

## Description

This is an example PISA dataset of mathematics items. The dataset contains 565 students on 11 items.

## Usage

```
data(data.pisaMath)
```

## Format

The dataset is a list. The list element `data` contains the dataset with the demographical variables student ID (`idstud`), school ID (`idschool`), a dummy variable for female students (`female`), socioeconomic status (`hisei`) and migrational background (`migra`). The remaining variables (starting with `M` in the name) are the mathematics items.

The item metadata are included in the list element `item` which contains item name (`item`) and the testlet label (`testlet`). An item not included in a testlet is indicated by `NA`.

The format is:

List of 2

```
$ data:'data.frame':
..$ idstud : num [1:565] 9e+10 9e+10 9e+10 9e+10 9e+10 ...
..$ idschool: int [1:565] 900015 900015 900015 900015 ...
..$ female : int [1:565] 0 0 0 0 0 0 0 0 0 ...
..$ hisei : num [1:565] -1.16 -1.099 -1.588 -0.365 -1.588 ...
..$ migra : int [1:565] 0 0 0 0 0 0 0 0 1 ...
..$ M192Q01 : int [1:565] 1 0 1 1 1 1 1 0 0 0 ...
..$ M406Q01 : int [1:565] 1 1 1 0 1 0 0 0 1 0 ...
..$ M406Q02 : int [1:565] 1 0 0 0 1 0 0 0 1 0 ...
..$ M423Q01 : int [1:565] 0 1 0 1 1 1 1 1 1 0 ...
..$ M496Q01 : int [1:565] 1 0 0 0 0 0 0 0 1 0 ...
..$ M496Q02 : int [1:565] 1 0 0 1 0 1 0 1 1 0 ...
..$ M564Q01 : int [1:565] 1 1 1 1 1 1 1 0 0 1 0 ...
..$ M564Q02 : int [1:565] 1 0 1 1 1 0 0 0 0 0 ...
..$ M571Q01 : int [1:565] 1 0 0 0 1 0 0 0 0 0 ...
..$ M603Q01 : int [1:565] 1 0 0 0 1 0 0 0 0 0 ...
..$ M603Q02 : int [1:565] 1 0 0 0 1 0 0 0 1 0 ...
$ item:'data.frame':
..$ item : Factor w/ 11 levels "M192Q01","M406Q01",...: 1 2 3 4 ...
..$ testlet: chr [1:11] NA "M406" "M406" NA ...
```



---

data.pisaPars	<i>Item Parameters from Two PISA Studies</i>
---------------	--

---

**Description**

This data frame contains item parameters from two PISA studies. Because the Rasch model is used, only item difficulties are considered.

**Usage**

```
data(data.pisaPars)
```

**Format**

A data frame with 25 observations on the following 4 variables.

item Item names  
 testlet Items are arranged in corresponding testlets. These names are located in this column.  
 study1 Item difficulties of study 1  
 study2 Item difficulties of study 2

---

data.pisaRead	<i>Dataset PISA Reading</i>
---------------	-----------------------------

---

**Description**

This is an example PISA dataset of reading items. The dataset contains 623 students on 12 items.

**Usage**

```
data(data.pisaRead)
```

**Format**

The dataset is a list. The list element `data` contains the dataset with the demographical variables student ID (`idstud`), school ID (`idschool`), a dummy variable for female students (`female`), socioeconomic status (`hisei`) and migrational background (`migra`). The remaining variables (starting with `R` in the name) are the reading items.

The item metadata are included in the list element `item` which contains item name (`item`), testlet label (`testlet`), item format (`ItemFormat`), text type (`TextType`) and text aspect (`Aspect`).

The format is:

```
List of 2
 $ data: 'data.frame':
 ..$ idstud : num [1:623] 9e+10 9e+10 9e+10 9e+10 9e+10 ...
```

```

..$ idschool: int [1:623] 900003 900003 900003 900003 ...
..$ female  : int [1:623] 1 0 1 0 0 0 1 0 1 0 ...
..$ hisei   : num [1:623] -1.16 -0.671 1.286 0.185 1.225 ...
..$ migra   : int [1:623] 0 0 0 0 0 0 0 0 0 0 ...
..$ R432Q01 : int [1:623] 1 1 1 1 1 1 1 1 1 1 ...
..$ R432Q05 : int [1:623] 1 1 1 1 1 0 1 1 1 0 ...
..$ R432Q06 : int [1:623] 0 0 0 0 0 0 0 0 0 0 ...
..$ R456Q01 : int [1:623] 1 1 1 1 1 1 1 1 1 1 ...
..$ R456Q02 : int [1:623] 1 1 1 1 1 1 1 1 1 1 ...
..$ R456Q06 : int [1:623] 1 1 1 1 1 1 0 0 1 1 ...
..$ R460Q01 : int [1:623] 1 1 0 0 0 0 0 1 1 1 ...
..$ R460Q05 : int [1:623] 1 1 1 1 1 1 1 1 1 1 ...
..$ R460Q06 : int [1:623] 0 1 1 1 1 1 0 0 1 1 ...
..$ R466Q02 : int [1:623] 0 1 0 1 1 0 1 0 0 1 ...
..$ R466Q03 : int [1:623] 0 0 0 1 0 0 0 1 0 1 ...
..$ R466Q06 : int [1:623] 0 1 1 1 1 1 0 1 1 1 ...
$ item: 'data.frame':
..$ item      : Factor w/ 12 levels "R432Q01","R432Q05",...: 1 2 3 4 ...
..$ testlet   : Factor w/ 4 levels "R432","R456",...: 1 1 1 2 ...
..$ ItemFormat: Factor w/ 2 levels "CR","MC": 1 1 2 2 1 1 1 2 2 1 ...
..$ TextType  : Factor w/ 3 levels "Argumentation",...: 1 1 1 3 ...
..$ Aspect    : Factor w/ 3 levels "Access_and_retrieve",...: 2 3 2 1 ...

```

---

data.pw

*Datasets for Pairwise Comparisons*

---

### Description

Some datasets for pairwise comparisons.

### Usage

```
data(data.pw01)
```

### Format

The dataset `data.pw01` contains results of a German football league from the season 2000/01.

---

data.ratings	<i>Rating Datasets</i>
--------------	------------------------

---

### Description

Some rating datasets.

### Usage

```
data(data.ratings1)
data(data.ratings2)
data(data.ratings3)
```

### Format

- Dataset data.ratings1:

Data frame with 274 observations containing 5 criteria (k1, ..., k5), 135 students and 7 raters.

```
'data.frame': 274 obs. of 7 variables:
 $ idstud: int 100020106 100020106 100070101 100070101 100100109 ...
 $ rater : Factor w/ 16 levels "db01","db02",...: 3 15 5 10 2 1 5 4 1 5 ...
 $ k1 : int 1 1 0 1 2 0 1 3 0 0 ...
 $ k2 : int 1 1 1 1 1 0 0 3 0 0 ...
 $ k3 : int 1 1 1 1 2 0 0 3 1 0 ...
 $ k4 : int 1 1 1 2 1 0 0 2 0 1 ...
 $ k5 : int 2 2 1 2 0 1 0 3 1 0 ...
```

Data from a 2009 Austrian survey of national educational standards for 8th graders in German language writing. Variables k1 to k5 denote several rating criteria of writing competency.

- Dataset data.ratings2:

Data frame with 615 observations containing 5 criteria (k1, ..., k5), 178 students and 16 raters.

```
'data.frame': 615 obs. of 7 variables:
 $ idstud: num 1001 1001 1002 1002 1003 ...
 $ rater : chr "R03" "R15" "R05" "R10" ...
 $ k1 : int 1 1 0 1 2 0 1 3 3 0 ...
 $ k2 : int 1 1 1 1 1 0 0 3 3 0 ...
 $ k3 : int 1 1 1 1 2 0 0 3 3 1 ...
 $ k4 : int 1 1 1 2 1 0 0 2 2 0 ...
 $ k5 : int 2 2 1 2 0 1 0 3 2 1 ...
```

- Dataset data.ratings3:

Data frame with 3169 observations containing 4 criteria (crit2, ..., crit6), 561 students and 52 raters.

```
'data.frame': 3169 obs. of 6 variables:
 $ idstud: num 10001 10001 10002 10002 10003 ...
 $ rater : num 840 838 842 808 830 845 813 849 809 802 ...
 $ crit2 : int 1 3 3 1 2 2 2 2 3 3 ...
 $ crit3 : int 2 2 2 2 2 2 2 2 3 3 ...
 $ crit4 : int 1 2 2 2 1 1 1 2 2 2 ...
 $ crit6 : num 4 4 4 3 4 4 4 4 4 4 ...
```

---

data.raw1

*Dataset with Raw Item Responses*

---

### Description

Dataset with raw item responses

### Usage

```
data(data.raw1)
```

### Format

A data frame with raw item responses of 1200 persons on the following 77 items:

```
'data.frame': 1200 obs. of 77 variables:
 $ I101: num 0 0 0 2 0 0 0 0 0 0 ...
 $ I102: int NA NA 2 1 2 1 3 2 NA NA ...
 $ I103: int 1 1 NA NA NA NA NA NA 1 1 ...
 ...
 $ I179: chr "E" "C" "D" "E" ...
```

---

data.read

*Dataset Reading*

---

### Description

This dataset contains  $N = 328$  students and  $I = 12$  items measuring reading competence. All 12 items are arranged into 3 testlets (items with common text stimulus) labeled as A, B and C. The allocation of items to testlets is indicated by their variable names.

### Usage

```
data(data.read)
```

**Format**

A data frame with 328 persons on the following 12 variables. Rows correspond to persons and columns to items. The following items are included in `data.read`:

Testlet A: A1, A2, A3, A4

Testlet B: B1, B2, B3, B4

Testlet C: C1, C2, C3, C4

**Examples**

```
## Not run:
data(data.read)
dat <- data.read
I <- ncol(dat)

# list of needed packages for the following examples
packages <- scan(what="character")
  eRm ltm TAM mRm CDM mirt psychotools IsingFit igraph qgraph pcalg
  polCA randomLCA psychomix MplusAutomation lavaan

# load packages. make an installation if necessary
miceadds::library_install(packages)

#####
# Model 1: Rasch model
#####

#---- M1a: rasch.mml2 (in sirt)
mod1a <- sirt::rasch.mml2(dat)
summary(mod1a)

#---- M1b: smirt (in sirt)
Qmatrix <- matrix(1,nrow=I , ncol=1)
mod1b <- sirt::smirt(dat,Qmatrix=Qmatrix)
summary(mod1b)

#---- M1c: gdm (in CDM)
theta.k <- seq(-6,6,len=21)
mod1c <- CDM::gdm(dat,theta.k=theta.k,irtmodel="1PL", skillspace="normal")
summary(mod1c)

#---- M1d: tam.mml (in TAM)
mod1d <- TAM::tam.mml( resp=dat )
summary(mod1d)

#---- M1e: RM (in eRm)
mod1e <- eRm::RM( dat )
# eRm uses Conditional Maximum Likelihood (CML) as the estimation method.
summary(mod1e)
eRm::plotPImap(mod1e)

#---- M1f: mrm (in mRm)
```

```

mod1f <- mRm::mrm( dat , cl=1) # CML estimation
mod1f$beta # item parameters

#---- M1g: mirt (in mirt)
mod1g <- mirt::mirt( dat , model=1 , itemtype="Rasch" , verbose=TRUE )
print(mod1g)
summary(mod1g)
coef(mod1g)
# arrange coefficients in nicer layout
mirt.wrapper.coef(mod1g)$coef

#---- M1h: rasch (in ltm)
mod1h <- ltm::rasch( dat , control=list(verbose=TRUE ) )
summary(mod1h)
coef(mod1h)

#---- M1i: RaschModel.fit (in psychotools)
mod1i <- psychotools::RaschModel.fit(dat) # CML estimation
summary(mod1i)
plot(mod1i)

#---- M1j: noharm.sirt (in sirt)
Fpatt <- matrix( 0 , I , 1 )
Fval <- 1 + 0*Fpatt
Ppatt <- Pval <- matrix(1,1,1)
mod1j <- sirt::noharm.sirt( dat=dat , Ppatt=Ppatt,Fpatt=Fpatt , Fval=Fval , Pval=Pval )
summary(mod1j)
# Normal-ogive model, multiply item discriminations with constant D=1.7.
# The same holds for other examples with noharm.sirt and R2noharm.
plot(mod1j)

#---- M1k: rasch.pml3 (in sirt)
mod1k <- sirt::rasch.pml3( dat=dat)
# pairwise marginal maximum likelihood estimation
summary(mod1k)

#---- M1l: running Mplus (using MplusAutomation package)
mplus_path <- "c:/Mplus7/Mplus.exe" # locate Mplus executable
# specify Mplus object
mplusmod <- MplusAutomation::mplusObject(
  TITLE = "1PL in Mplus ;" ,
  VARIABLE = paste0( "CATEGORICAL ARE " , paste0(colnames(dat),collapse=" ") ) ,
  MODEL = "
    ! fix all item loadings to 1
    F1 BY A1@1 A2@1 A3@1 A4@1 ;
    F1 BY B1@1 B2@1 B3@1 B4@1 ;
    F1 BY C1@1 C2@1 C3@1 C4@1 ;
    ! estimate variance
    F1 ;
  " ,
  ANALYSIS = "ESTIMATOR=MLR;" ,
  OUTPUT = "stand;" ,
  usevariables = colnames(dat) , rdata = dat )

```

```

# write Mplus syntax
filename <- "mod1u" # specify file name
# create Mplus syntaxes
res2 <- MplusAutomation::mplusModeler(object = mplusmod , dataout = paste0(filename, ".dat") ,
                                     modelout= paste0(filename, ".inp"), run = 0 )
# run Mplus model
MplusAutomation::runModels( filefilter = paste0(filename, ".inp"), Mplus_command = mplus_path)
# alternatively, the system() command can also be used
# get results
mod1l <- MplusAutomation::readModels(target = getwd() , filefilter = filename )
mod1l$summaries # summaries
mod1l$parameters$unstandardized # parameter estimates

#*****
# Model 2: 2PL model
#*****

#---- M2a: rasch.mml2 (in sirt)
mod2a <- sirt::rasch.mml2(dat , est.a=1:I)
summary(mod2a)

#---- M2b: smirt (in sirt)
mod2b <- sirt::smirt(dat,Qmatrix=Qmatrix,est.a="2PL")
summary(mod2b)

#---- M2c: gdm (in CDM)
mod2c <- CDM::gdm(dat,theta.k=theta.k,irtmodel="2PL", skillspace="normal")
summary(mod2c)

#---- M2d: tam.mml (in TAM)
mod2d <- TAM::tam.mml.2pl( resp=dat )
summary(mod2d)

#---- M2e: mirt (in mirt)
mod2e <- mirt::mirt( dat , model=1 , itemtype="2PL" )
print(mod2e)
summary(mod2e)
mirt.wrapper.coef(mod1g)$coef

#---- M2f: ltm (in ltm)
mod2f <- ltm::ltm( dat ~ z1 , control=list(verbose=TRUE ) )
summary(mod2f)
coef(mod2f)
plot(mod2f)

#---- M2g: R2noharm (in NOHARM, running from within R using sirt package)
# define noharm.path where 'NoharmCL.exe' is located
noharm.path <- "c:/NOHARM"
# covariance matrix
P.pattern <- matrix( 1 , ncol=1 , nrow=1 )
P.init <- P.pattern
P.init[1,1] <- 1
# loading matrix

```

```

F.pattern <- matrix(1,I,1)
F.init <- F.pattern
# estimate model
mod2g <- sirt::R2noharm( dat = dat , model.type="CFA" , F.pattern = F.pattern ,
                        F.init = F.init , P.pattern = P.pattern , P.init = P.init ,
                        writename = "ex2g" , noharm.path = noharm.path , dec = "," )
summary(mod2g)

#---- M2h: noharm.sirt (in sirt)
mod2h <- sirt::noharm.sirt( dat=dat , Ppatt=P.pattern,Fpatt=F.pattern ,
                          Fval=F.init , Pval=P.init )
summary(mod2h)
plot(mod2h)

#---- M2i: rasch.pml2 (in sirt)
mod2i <- sirt::rasch.pml2(dat, est.a=1:I)
summary(mod2i)

#---- M2j: WLSMV estimation with cfa (in lavaan)
lavmodel <- "F =~ A1+A2+A3+A4+B1+B2+B3+B4+
            C1+C2+C3+C4"
mod2j <- lavaan::cfa( data=dat , model=lavmodel, std.lv = TRUE, ordered=colnames(dat))
summary(mod2j , standardized=TRUE , fit.measures=TRUE , rsquare=TRUE)

#*****
# Model 3: 3PL model (note that results can be quite unstable!)
#*****

#---- M3a: rasch.mml2 (in sirt)
mod3a <- sirt::rasch.mml2(dat , est.a=1:I, est.c=1:I)
summary(mod3a)

#---- M3b: smirt (in sirt)
mod3b <- sirt::smirt(dat,Qmatrix=Qmatrix,est.a="2PL" , est.c=1:I)
summary(mod3b)

#---- M3c: mirt (in mirt)
mod3c <- mirt::mirt( dat , model=1 , itemtype="3PL" , verbose=TRUE)
summary(mod3c)
coef(mod3c)
# stabilize parameter estimating using informative priors for guessing parameters
mirtmodel <- mirt::mirt.model("
    F = 1-12
    PRIOR = (1-12, g, norm, -1.38, 0.25)
")
# a prior N(-1.38,.25) is specified for transformed guessing parameters: qlogis(g)
# simulate values from this prior for illustration
N <- 100000
logit.g <- rnorm(N, mean=-1.38 , sd=sqrt(.5) )
plot( density(logit.g) ) # transformed qlogis(g)
plot( density(plogis(logit.g)) ) # g parameters
# estimate 3PL with priors
mod3c1 <- mirt::mirt(dat, mirtmodel, itemtype = "3PL",verbose=TRUE)

```



```

coef(mod3c1)
# In addition, set upper bounds for g parameters of .35
mirt.pars <- mirt::mirt( dat , mirtmodel , itemtype = "3PL" , pars="values")
ind <- which( mirt.pars$name == "g" )
mirt.pars[ ind , "value" ] <- plogis(-1.38)
mirt.pars[ ind , "ubound" ] <- .35
# prior distribution for slopes
ind <- which( mirt.pars$name == "a1" )
mirt.pars[ ind , "prior_1" ] <- 1.3
mirt.pars[ ind , "prior_2" ] <- 2
mod3c2 <- mirt::mirt(dat, mirtmodel, itemtype = "3PL",
                    pars=mirt.pars,verbose=TRUE , technical=list(NCYCLES=100) )
coef(mod3c2)
mirt.wrapper.coef(mod3c2)

#---- M3d: ltm (in ltm)
mod3d <- ltm::tpm( dat , control=list(verbose=TRUE ) , max.guessing=.3)
summary(mod3d)
coef(mod3d) # => numerical instabilities

#*****
# Model 4: 3-dimensional Rasch model
#*****

# define Q-matrix
Q <- matrix( 0 , nrow=12 , ncol=3 )
Q[ cbind(1:12 , rep(1:3,each=4) ) ] <- 1
rownames(Q) <- colnames(dat)
colnames(Q) <- c("A","B","C")

# define nodes
theta.k <- seq(-6,6,len=13 )

#---- M4a: smirt (in sirt)
mod4a <- sirt::smirt(dat,Qmatrix=Q,irtmodel="comp" , theta.k=theta.k , maxiter=30)
summary(mod4a)

#---- M4b: rasch.mml2 (in sirt)
mod4b <- sirt::rasch.mml2(dat,Q=Q,theta.k=theta.k , mmliter=30)
summary(mod4b)

#---- M4c: gdm (in CDM)
mod4c <- CDM::gdm( dat , irtmodel="1PL" , theta.k=theta.k , skillspace="normal" ,
                  Qmatrix=Q , maxiter=30 , centered.latent=TRUE )
summary(mod4c)

#---- M4d: tam.mml (in TAM)
mod4d <- TAM::tam.mml( resp=dat , Q=Q , control=list(nodes=theta.k , maxiter=30) )
summary(mod4d)

#---- M4e: R2noharm (in NOHARM, running from within R using sirt package)
noharm.path <- "c:/NOHARM"
# covariance matrix

```

```

P.pattern <- matrix( 1 , ncol=3 , nrow=3 )
P.init <- 0.8+0*P.pattern
diag(P.init) <- 1
  # loading matrix
F.pattern <- 0*Q
F.init <- Q
  # estimate model
mod4e <- sirt::R2noharm( dat = dat , model.type="CFA" , F.pattern = F.pattern ,
  F.init = F.init , P.pattern = P.pattern , P.init = P.init ,
  writename = "ex4e" , noharm.path = noharm.path , dec = "," )
summary(mod4e)

#---- M4f: mirt (in mirt)
cmodel <- mirt::mirt.model("
  F1 = 1-4
  F2 = 5-8
  F3 = 9-12
  # equal item slopes correspond to the Rasch model
  CONSTRAIN = (1-4, a1), (5-8, a2) , (9-12,a3)
  COV = F1*F2, F1*F3 , F2*F3
  " )
mod4f <- mirt::mirt(dat, cmodel , verbose=TRUE)
summary(mod4f)

#*****
# Model 5: 3-dimensional 2PL model
#*****

#---- M5a: smirt (in sirt)
mod5a <- sirt::smirt(dat,Qmatrix=Q,irtmodel="comp" , est.a="2PL" , theta.k=theta.k ,
  maxiter=30)
summary(mod5a)

#---- M5b: rasch.mml2 (in sirt)
mod5b <- sirt::rasch.mml2(dat,Q=Q,theta.k=theta.k ,est.a=1:12, mmliter=30)
summary(mod5b)

#---- M5c: gdm (in CDM)
mod5c <- CDM::gdm( dat , irtmodel="2PL" , theta.k=theta.k , skillspace="loglinear" ,
  Qmatrix=Q , maxiter=30 , centered.latent=TRUE ,
  standardized.latent=TRUE)
summary(mod5c)

#---- M5d: tam.mml (in TAM)
mod5d <- TAM::tam.mml.2pl( resp=dat , Q=Q , control=list(nodes=theta.k , maxiter=30) )
summary(mod5d)

#---- M5e: R2noharm (in NOHARM, running from within R using sirt package)
noharm.path <- "c:/NOHARM"
  # covariance matrix
P.pattern <- matrix( 1 , ncol=3 , nrow=3 )
diag(P.pattern) <- 0
P.init <- 0.8+0*P.pattern

```

```

diag(P.init) <- 1
# loading matrix
F.pattern <- Q
F.init <- Q
# estimate model
mod5e <- sirt::R2noharm( dat = dat , model.type="CFA" , F.pattern = F.pattern ,
  F.init = F.init , P.pattern = P.pattern , P.init = P.init ,
  writename = "ex5e" , noharm.path = noharm.path , dec = "," )
summary(mod5e)

#---- M5f: mirt (in mirt)
cmodel <- mirt::mirt.model("
  F1 = 1-4
  F2 = 5-8
  F3 = 9-12
  COV = F1*F2, F1*F3 , F2*F3
" )
mod5f <- mirt::mirt(dat, cmodel , verbose=TRUE)
summary(mod5f)

#*****
# Model 6: Network models (Graphical models)
#*****

#---- M6a: Ising model using the IsingFit package (undirected graph)
# - fit Ising model using the "OR rule" (AND=FALSE)
mod6a <- IsingFit::IsingFit(x=dat, family="binomial" , AND=FALSE)
summary(mod6a)
##          Network Density:                0.29
##   Gamma:                0.25
##   Rule used:                Or-rule
# plot results
qgraph(mod6a$weiadj,fade = FALSE)

#*-- graph estimation using pcalg package

# some packages from Bioconductor must be downloaded at first (if not yet done)
if (FALSE){ # set 'if (TRUE)' if packages should be downloaded
  source("http://bioconductor.org/biocLite.R")
  biocLite("RBGL")
  biocLite("Rgraphviz")
}

#---- M6b: graph estimation based on Pearson correlations
V <- colnames(dat)
n <- nrow(dat)
mod6b <- pcalg::pc(suffStat = list(C = cor(dat), n = n ),
  indepTest = gaussCITest, ## indep.test: partial correlations
  alpha=0.05, labels = V, verbose = TRUE)
plot(mod6b)
# plot in qgraph package
qgraph::qgraph(mod6b , label.color= rep( c( "red" , "blue","darkgreen" ) , each=4 ) ,
  edge.color="black")

```

```

summary(mod6b)

#---- M6c: graph estimation based on tetrachoric correlations
mod6c <- pcalg::pc(suffStat = list(C = tetrachoric2(dat)$rho, n = n ),
                 indepTest = gaussCItest, alpha=0.05, labels = V, verbose = TRUE)
plot(mod6c)
summary(mod6c)

#---- M6d: Statistical implicative analysis (in sirt)
mod6d <- sirt::sia.sirt(dat , significance=.85 )
# plot results with igraph and qgraph package
plot( mod6d$igraph.obj , vertex.shape="rectangle" , vertex.size=30 )
qgraph::qgraph( mod6d$adj.matrix )

#*****
# Model 7: Latent class analysis with 3 classes
#*****

#---- M7a: randomLCA (in randomLCA)
#       - use two trials of starting values
mod7a <- randomLCA::randomLCA(dat,nclass=3, notrials=2, verbose=TRUE)
summary(mod7a)
plot(mod7a,type="l" , xlab="Item")

#---- M7b: rasch.mirtlc (in sirt)
mod7b <- sirt::rasch.mirtlc( dat , Nclasses = 3 ,seed= -30 , nstarts=2 )
summary(mod7b)
matplot( t(mod7b$pjk) , type="l" , xlab="Item" )

#---- M7c: poLCA (in poLCA)
#       define formula for outcomes
f7c <- paste0( "cbind(" , paste0(colnames(dat),collapse=","), ") ~ 1 " )
dat1 <- as.data.frame( dat + 1 ) # poLCA needs integer values from 1,2,..
mod7c <- poLCA::poLCA( as.formula(f7c),dat1,nclass=3 , verbose=TRUE)
plot(mod7c)

#---- M7d: gom.em (in sirt)
#       - the latent class model is a special grade of membership model
mod7d <- sirt::gom.em( dat , K=3 , problevels=c(0,1) , model="GOM" )
summary(mod7d)

#---- - M7e: mirt (in mirt)
#       define three latent classes
Theta <- diag(3)
#       define mirt model
I <- ncol(dat) # I = 12
mirtmodel <- mirt::mirt.model("
      C1 = 1-12
      C2 = 1-12
      C3 = 1-12
")
#       get initial parameter values
mod.pars <- mirt::mirt(dat, model=mirtmodel , pars = "values")

```

```

# modify parameters: only slopes refer to item-class probabilities
set.seed(9976)
# set starting values for class specific item probabilities
mod.pars[ mod.pars$name == "d" ,"value" ] <- 0
mod.pars[ mod.pars$name == "d" ,"est" ] <- FALSE
b1 <- qnorm( colMeans( dat ) )
mod.pars[ mod.pars$name == "a1" ,"value" ] <- b1
# random starting values for other classes
mod.pars[ mod.pars$name %in% c("a2","a3") ,"value" ] <- b1+runif( 12*2 , -1 ,1 )
mod.pars
*** define prior for latent class analysis
lca_prior <- function(Theta,Etable){
# number of latent Theta classes
TP <- nrow(Theta)
# prior in initial iteration
if ( is.null(Etable) ){ prior <- rep( 1/TP , TP ) }
# process Etable (this is correct for datasets without missing data)
if ( ! is.null(Etable) ){
# sum over correct and incorrect expected responses
prior <- ( rowSums(Etable[ , seq(1,2*I,2)]) + rowSums(Etable[,seq(2,2*I,2)]) )/I
}
prior <- prior / sum(prior)
return(prior)
}
*** estimate model
mod7e <- mirt::mirt(dat, mirtmodel , pars = mod.pars , verbose=TRUE ,
technical = list( customTheta=Theta , customPriorFun = lca_prior ) )
# compare estimated results
print(mod7e)
summary(mod7b)
# The number of estimated parameters is incorrect because mirt does not correctly count
# estimated parameters from the user customized prior distribution.
mod7e@nest <- as.integer(sum(mod.pars$est) + 2) # two additional class probabilities
# extract log-likelihood
mod7e@logLik
# compute AIC and BIC
( AIC <- -2*mod7e@logLik+2*mod7e@nest )
( BIC <- -2*mod7e@logLik+log(mod7e@Data$N)*mod7e@nest )
# RMSEA and SRMSR fit statistic
mirt::M2(mod7e) # TLI and CFI does not make sense in this example
*** extract item parameters
mirt.wrapper.coef(mod7e)
*** extract class-specific item-probabilities
probs <- apply( coef1[ , c("a1","a2","a3") ] , 2 , plogis )
matplot( probs , type="l" , xlab="Item" , main="mirt::mirt")
*** inspect estimated distribution
mod7e@Theta
mod7e@Prior[[1]]

*****
# Model 8: Mixed Rasch model with two classes
*****

```

```

#---- M8a: raschmix (in psychomix)
mod8a <- psychomix::raschmix(data= as.matrix(dat) , k = 2, scores = "saturated")
summary(mod8a)

#---- M8b: mrm (in mRm)
mod8b <- mRm::mrm(data.matrix=dat, cl=2)
mod8b$conv.to.bound
plot(mod8b)
print(mod8b)

#---- M8c: mirt (in mirt)
  #* define theta grid
theta.k <- seq( -5 , 5 , len=9 )
TP <- length(theta.k)
Theta <- matrix( 0 , nrow=2*TP , ncol=4)
Theta[1:TP,1:2] <- cbind(theta.k , 1 )
Theta[1:TP + TP,3:4] <- cbind(theta.k , 1 )
Theta
  # define model
I <- ncol(dat) # I = 12
mirtmodel <- mirt::mirt.model("
    F1a = 1-12 # slope Class 1
    F1b = 1-12 # difficulty Class 1
    F2a = 1-12 # slope Class 2
    F2b = 1-12 # difficulty Class 2
    CONSTRAIN = (1-12,a1),(1-12,a3)
")
  # get initial parameter values
mod.pars <- mirt::mirt(dat, model=mirtmodel , pars = "values")
  # set starting values for class specific item probabilities
mod.pars[ mod.pars$name == "d" ,"value" ] <- 0
mod.pars[ mod.pars$name == "d" ,"est" ] <- FALSE
mod.pars[ mod.pars$name == "a1" ,"value" ] <- 1
mod.pars[ mod.pars$name == "a3" ,"value" ] <- 1
  # initial values difficulties
b1 <- qlogis( colMeans(dat) )
mod.pars[ mod.pars$name == "a2" ,"value" ] <- b1
mod.pars[ mod.pars$name == "a4" ,"value" ] <- b1 + runif(I , -1 , 1)
  #* define prior for mixed Rasch analysis
mixed_prior <- function(Theta,Etable){
  NC <- 2 # number of theta classes
  TP <- nrow(Theta) / NC
  prior1 <- dnorm( Theta[1:TP,1] )
  prior1 <- prior1 / sum(prior1)
  if ( is.null(Etable) ){ prior <- c( prior1 , prior1 ) }
  if ( ! is.null(Etable) ){
    prior <- ( rowSums( Etable[ , seq(1,2*I,2)] ) +
               rowSums( Etable[,seq(2,2*I,2)] ) )/I
    a1 <- aggregate( prior , list( rep(1:NC , each=TP) ) , sum )
    a1[,2] <- a1[,2] / sum( a1[,2] )
    # print some information during estimation
    cat( paste0( " Class proportions: " ,
                 paste0( round(a1[,2] , 3 ) , collapse= " " ) ) , "\n")

```

```

    a1 <- rep( a1[,2] , each=TP )
    # specify mixture of two normal distributions
    prior <- a1*c(prior1,prior1)
    }
  prior <- prior / sum(prior)
  return(prior)
}
#* estimate model
mod8c <- mirt::mirt(dat, mirtmodel , pars=mod.pars , verbose=TRUE ,
  technical = list( customTheta=Theta , customPriorFun = mixed_prior ) )
# Like in Model 7e, the number of estimated parameters must be included.
mod8c@nest <- as.integer(sum(mod.pars$est) + 1)
# two class proportions and therefore one probability is freely estimated.
#* extract item parameters
mirt.wrapper.coef(mod8c)
#* estimated distribution
mod8c@Theta
mod8c@Prior

#---- M8d: tamaan (in TAM)

tammodel <- "
ANALYSIS:
  TYPE=MIXTURE ;
  NCLASSES(2);
  NSTARTS(7,20);
LAVAAAN MODEL:
  F =~ A1__C4
  F ~~ F
ITEM TYPE:
  ALL(Rasch);
  "
mod8d <- TAM::tamaan( tammodel , resp=dat )
summary(mod8d)
# plot item parameters
I <- 12
ipars <- mod8d$itempartable_MIXTURE[ 1:I , ]
plot( 1:I , ipars[,3] , type="o" , ylim= range( ipars[,3:4] ) , pch=16 ,
  xlab="Item" , ylab="Item difficulty")
lines( 1:I , ipars[,4] , type="l" , col=2 , lty=2)
points( 1:I , ipars[,4] , col=2 , pch=2)

#*****
# Model 9: Mixed 2PL model with two classes
#*****

#---- M9a: tamaan (in TAM)

tammodel <- "
ANALYSIS:
  TYPE=MIXTURE ;
  NCLASSES(2);
  NSTARTS(10,30);

```

```

LAVAAN MODEL:
  F =~ A1__C4
  F ~~ F
ITEM TYPE:
  ALL(2PL);
  "

mod9a <- TAM::tamaan( tammodel , resp=dat )
summary(mod9a)

#*****
# Model 10: Rasch testlet model
#*****

#---- M10a: tam.fa (in TAM)
dims <- substring( colnames(dat),1,1 ) # define dimensions
mod10a <- TAM::tam.fa( resp=dat , irtmodel="bifactor1" , dims=dims ,
  control=list(maxiter=60) )
summary(mod10a)

#---- M10b: mirt (in mirt)
cmodel <- mirt::mirt.model("
  G = 1-12
  A = 1-4
  B = 5-8
  C = 9-12
  CONSTRAIN = (1-12,a1), (1-4, a2), (5-8, a3) , (9-12,a4)
")
mod10b <- mirt::mirt(dat, model=cmodel , verbose=TRUE)
summary(mod10b)
coef(mod10b)
mod10b@logLik # equivalent is slot( mod10b , "logLik")

#alternatively, using a dimensional reduction approach (faster and better accuracy)
cmodel <- mirt::mirt.model("
  G = 1-12
  CONSTRAIN = (1-12,a1), (1-4, a2), (5-8, a3) , (9-12,a4)
")
item_bundles <- rep(c(1,2,3), each = 4)
mod10b1 <- mirt::bfactor(dat, model=item_bundles, model2=cmodel , verbose=TRUE)
coef(mod10b1)

#---- M10c: smirt (in sirt)
# define Q-matrix
Qmatrix <- matrix(0,12,4)
Qmatrix[,1] <- 1
Qmatrix[ cbind( 1:12 , match( dims , unique(dims)) +1 ) ] <- 1
# uncorrelated factors
variance.fixed <- cbind( c(1,1,1,2,2,3) , c(2,3,4,3,4,4) , 0 )
# estimate model
mod10c <- sirt::smirt( dat , Qmatrix=Qmatrix , irtmodel="comp" ,
  variance.fixed=variance.fixed , qmcnodes=1000 , maxiter=60)
summary(mod10c)

```



```

*****
# Model 11: Bifactor model
*****

#---- M11a: tam.fa (in TAM)
dims <- substring( colnames(dat),1,1 ) # define dimensions
mod11a <- TAM::tam.fa( resp=dat , irtmodel="bifactor2" , dims=dims ,
                      control=list(maxiter=60) )
summary(mod11a)

#---- M11b: bfactor (in mirt)
dims1 <- match( dims , unique(dims) )
mod11b <- mirt::bfactor(dat, model=dims1 , verbose=TRUE)
summary(mod11b)
coef(mod11b)
mod11b@logLik

#---- M11c: smirt (in sirt)
# define Q-matrix
Qmatrix <- matrix(0,12,4)
Qmatrix[,1] <- 1
Qmatrix[ cbind( 1:12 , match( dims , unique(dims)) +1 ) ] <- 1
# uncorrelated factors
variance.fixed <- cbind( c(1,1,1,2,2,3) , c(2,3,4,3,4,4) , 0 )
# estimate model
mod11c <- sirt::smirt( dat , Qmatrix=Qmatrix , irtmodel="comp" , est.a="2PL" ,
                      variance.fixed=variance.fixed , qmcnodes=1000 , maxiter=60)
summary(mod11c)

*****
# Model 12: Located latent class model: Rasch model with three theta classes
*****

# use 10th item as the reference item
ref.item <- 10
# ability grid
theta.k <- seq(-4,4,len=9)

#---- M12a: rasch.mirtlc (in sirt)
mod12a <- sirt::rasch.mirtlc(dat , Nclasses=3, modeltype="MLC1" , ref.item=ref.item )
summary(mod12a)

#---- M12b: gdm (in CDM)
theta.k <- seq(-1 , 1 , len=3) # initial matrix
b.constraint <- matrix( c(10,1,0) , nrow=1,ncol=3)
# estimate model
mod12b <- CDM::gdm( dat , theta.k = theta.k , skillspace="est" , irtmodel="1PL" ,
                   b.constraint=b.constraint , maxiter=200)
summary(mod12b)

#---- M12c: mirt (in mirt)
items <- colnames(dat)
# define three latent classes

```

```

Theta <- diag(3)
# define mirt model
I <- ncol(dat) # I = 12
mirtmodel <- mirt::mirt.model("
  C1 = 1-12
  C2 = 1-12
  C3 = 1-12
  CONSTRAIN = (1-12,a1),(1-12,a2),(1-12,a3)
")
# get parameters
mod.pars <- mirt(dat, model=mirtmodel , pars = "values")
# set starting values for class specific item probabilities
mod.pars[ mod.pars$name == "d" ,"value" ] <- qlogis( colMeans(dat,na.rm=TRUE) )
# set item difficulty of reference item to zero
ind <- which( ( paste(mod.pars$item) == items[ref.item] ) &
             ( ( paste(mod.pars$name) == "d" ) ) )
mod.pars[ ind ,"value" ] <- 0
mod.pars[ ind ,"est" ] <- FALSE
# initial values for a1, a2 and a3
mod.pars[ mod.pars$name %in% c("a1","a2","a3") ,"value" ] <- c(-1,0,1)
mod.pars
#* define prior for latent class analysis
lca_prior <- function(Theta,Etable){
# number of latent Theta classes
TP <- nrow(Theta)
# prior in initial iteration
if ( is.null(Etable) ){
  prior <- rep( 1/TP , TP )
}
# process Etable (this is correct for datasets without missing data)
if ( ! is.null(Etable) ){
  # sum over correct and incorrect expected responses
  prior <- ( rowSums( Etable[ , seq(1,2*I,2)] ) + rowSums( Etable[ , seq(2,2*I,2)] ) )/I
}
prior <- prior / sum(prior)
return(prior)
}
#* estimate model
mod12c <- mirt(dat, mirtmodel , technical = list(
  customTheta=Theta , customPriorFun = lca_prior) ,
  pars = mod.pars , verbose=TRUE )
# estimated parameters from the user customized prior distribution.
mod12c@nest <- as.integer(sum(mod.pars$est) + 2)
#* extract item parameters
coef1 <- mirt.wrapper.coef(mod12c)
#* inspect estimated distribution
mod12c@Theta
coef1$coef[1,c("a1","a2","a3")]
mod12c@Prior[[1]]

#*****
# Model 13: Multidimensional model with discrete traits
#*****

```

```

# define Q-Matrix
Q <- matrix( 0 , nrow=12,ncol=3)
Q[1:4,1] <- 1
Q[5:8,2] <- 1
Q[9:12,3] <- 1
# define discrete theta distribution with 3 dimensions
Theta <- scan(what="character",nlines=1)
  000 100 010 001 110 101 011 111
Theta <- as.numeric( unlist( lapply( Theta , strsplit , split="" ) ) )
Theta <- matrix(Theta , 8 , 3 , byrow=TRUE )
Theta

#---- Model 13a: din (in CDM)
mod13a <- CDM::din( dat , q.matrix=Q , rule="DINA")
summary(mod13a)
# compare used Theta distributions
cbind( Theta , mod13a$attribute.patt.splitted)

#---- Model 13b: gdm (in CDM)
mod13b <- CDM::gdm( dat , Qmatrix=Q , theta.k=Theta , skillspace="full")
summary(mod13b)

#---- Model 13c: mirt (in mirt)
# define mirt model
I <- ncol(dat) # I = 12
mirtmodel <- mirt::mirt.model("
  F1 = 1-4
  F2 = 5-8
  F3 = 9-12
")
# get parameters
mod.pars <- mirt(dat, model=mirtmodel , pars = "values")
# starting values d parameters (transformed guessing parameters)
ind <- which( mod.pars$name == "d" )
mod.pars[ind,"value"] <- qlogis(.2)
# starting values transformed slipping parameters
ind <- which( ( mod.pars$name %in% paste0("a",1:3) ) & ( mod.pars$est ) )
mod.pars[ind,"value"] <- qlogis(.8) - qlogis(.2)
mod.pars

#* define prior for latent class analysis
lca_prior <- function(Theta,Etable){
  TP <- nrow(Theta)
  if ( is.null(Etable) ){
    prior <- rep( 1/TP , TP )
  }
  if ( ! is.null(Etable) ){
    prior <- ( rowSums( Etable[ , seq(1,2*I,2)] ) + rowSums( Etable[ , seq(2,2*I,2)] ) )/I
  }
  prior <- prior / sum(prior)
  return(prior)
}
#* estimate model

```

```

mod13c <- mirt(dat, mirtmodel , technical = list(
  customTheta=Theta , customPriorFun = lca_prior) ,
  pars = mod.pars , verbose=TRUE )
# estimated parameters from the user customized prior distribution.
mod13c@nest <- as.integer(sum(mod.pars$est) + 2)
#* extract item parameters
coef13c <- mirt.wrapper.coef(mod13c)$coef
#* inspect estimated distribution
mod13c@Theta
mod13c@Prior[[1]]

#-* comparisons of estimated parameters
# extract guessing and slipping parameters from din
dfr <- coef(mod13a)[ , c("guess","slip") ]
colnames(dfr) <- paste0("din.",c("guess","slip") )
# estimated parameters from gdm
dfr$gdm.guess <- plogis(mod13b$item$b)
dfr$gdm.slip <- 1 - plogis( rowSums(mod13b$item[,c("b.Cat1","a.F1","a.F2","a.F3")] ) )
# estimated parameters from mirt
dfr$mirt.guess <- plogis( coef13c$d )
dfr$mirt.slip <- 1 - plogis( rowSums(coef13c[,c("d","a1","a2","a3")] ) )
# comparison
round(dfr[, c(1,3,5,2,4,6)],3)
##      din.guess gdm.guess mirt.guess din.slip gdm.slip mirt.slip
## A1    0.691    0.684    0.686    0.000    0.000    0.000
## A2    0.491    0.489    0.489    0.031    0.038    0.036
## A3    0.302    0.300    0.300    0.184    0.193    0.190
## A4    0.244    0.239    0.240    0.337    0.340    0.339
## B1    0.568    0.579    0.577    0.163    0.148    0.151
## B2    0.329    0.344    0.340    0.344    0.326    0.329
## B3    0.817    0.827    0.825    0.014    0.007    0.009
## B4    0.431    0.463    0.456    0.104    0.089    0.092
## C1    0.188    0.191    0.189    0.013    0.013    0.013
## C2    0.050    0.050    0.050    0.239    0.238    0.239
## C3    0.000    0.002    0.001    0.065    0.065    0.065
## C4    0.000    0.004    0.000    0.212    0.212    0.212

# estimated class sizes
dfr <- data.frame( "Theta" = Theta , "din"=mod13a$attribute.patt$class.prob ,
  "gdm"=mod13b$pi.k , "mirt" = mod13c@Prior[[1]])
# comparison
round(dfr,3)
##      Theta.1 Theta.2 Theta.3  din  gdm  mirt
## 1         0         0         0 0.039 0.041 0.040
## 2         1         0         0 0.008 0.009 0.009
## 3         0         1         0 0.009 0.007 0.008
## 4         0         0         1 0.394 0.417 0.412
## 5         1         1         0 0.011 0.011 0.011
## 6         1         0         1 0.017 0.042 0.037
## 7         0         1         1 0.042 0.008 0.016
## 8         1         1         1 0.480 0.465 0.467

#*****

```

```

# Model 14: DINA model with two skills
#*****

# define some simple Q-Matrix (does not really make in this application)
Q <- matrix( 0 , nrow=12,ncol=2)
Q[1:4,1] <- 1
Q[5:8,2] <- 1
Q[9:12,1:2] <- 1
# define discrete theta distribution with 3 dimensions
Theta <- scan(what="character",nlines=1)
      00 10 01 11
Theta <- as.numeric( unlist( lapply( Theta , strsplit , split="" ) ) )
Theta <- matrix(Theta , 4 , 2 , byrow=TRUE )
Theta

#---- Model 14a: din (in CDM)
mod14a <- CDM::din( dat , q.matrix=Q , rule="DINA")
summary(mod14a)
# compare used Theta distributions
cbind( Theta , mod14a$attribute.patt.splitted)

#---- Model 14b: mirt (in mirt)
# define mirt model
I <- ncol(dat) # I = 12
mirtmodel <- mirt::mirt.model("
      F1 = 1-4
      F2 = 5-8
      (F1*F2) = 9-12
")
#-> constructions like (F1*F2*F3) are also allowed in mirt.model
# get parameters
mod.pars <- mirt(dat, model=mirtmodel , pars = "values")
# starting values d parameters (transformed guessing parameters)
ind <- which( mod.pars$name == "d" )
mod.pars[ind,"value"] <- qlogis(.2)
# starting values transformed slipping parameters
ind <- which( ( mod.pars$name %in% paste0("a",1:3) ) & ( mod.pars$est ) )
mod.pars[ind,"value"] <- qlogis(.8) - qlogis(.2)
mod.pars
#* use above defined prior lca_prior
# lca_prior <- function(prior,Etable) ...
#* estimate model
mod14b <- mirt(dat, mirtmodel , technical = list(
      customTheta=Theta , customPriorFun = lca_prior) ,
      pars = mod.pars , verbose=TRUE )
# estimated parameters from the user customized prior distribution.
mod14b@nest <- as.integer(sum(mod.pars$est) + 2)
#* extract item parameters
coef14b <- mirt.wrapper.coef(mod14b)$coef

#* comparisons of estimated parameters
# extract guessing and slipping parameters from din
dfr <- coef(mod14a)[ , c("guess","slip") ]

```

```

colnames(dfr) <- paste0("din.",c("guess","slip") )
# estimated parameters from mirt
dfr$mirt.guess <- plogis( coef14b$d )
dfr$mirt.slip <- 1 - plogis( rowSums(coef14b[,c("d","a1","a2","a3")]) )
# comparison
round(dfr[, c(1,3,2,4)],3)
##      din.guess mirt.guess din.slip mirt.slip
## A1      0.674      0.671  0.030  0.030
## A2      0.423      0.420  0.049  0.050
## A3      0.258      0.255  0.224  0.225
## A4      0.245      0.243  0.394  0.395
## B1      0.534      0.543  0.166  0.164
## B2      0.338      0.347  0.382  0.380
## B3      0.796      0.802  0.016  0.015
## B4      0.421      0.436  0.142  0.140
## C1      0.850      0.851  0.000  0.000
## C2      0.480      0.480  0.097  0.097
## C3      0.746      0.746  0.026  0.026
## C4      0.575      0.577  0.136  0.137

# estimated class sizes
dfr <- data.frame( "Theta" = Theta , "din"=mod13a$attribute.patt$class.prob ,
                  "mirt" = mod14b@Prior[[1]])
# comparison
round(dfr,3)
##      Theta.1 Theta.2  din mirt
## 1          0        0 0.357 0.369
## 2          1        0 0.044 0.049
## 3          0        1 0.047 0.031
## 4          1        1 0.553 0.551

#####
# Model 15: Rasch model with non-normal distribution
#####

# A non-normal theta distributed is specified by log-linear smoothing
# the distribution as described in
# Xu, X., & von Davier, M. (2008). Fitting the structured general diagnostic model
# to NAEP data. ETS Research Report ETS RR-08-27. Princeton, ETS.

# define theta grid
theta.k <- matrix( seq(-4,4,len=15) , ncol=1 )
# define design matrix for smoothing (up to cubic moments)
delta.designmatrix <- cbind( 1 , theta.k , theta.k^2 , theta.k^3 )
# constrain item difficulty of fifth item (item B1) to zero
b.constraint <- matrix( c(5,1,0) , ncol=3 )

#---- Model 15a: gdm (in CDM)
mod15a <- CDM::gdm( dat , irtmodel="1PL" , theta.k=theta.k ,
                  b.constraint=b.constraint )
summary(mod15a)
# plot estimated distribution
barplot( mod15a$pi.k[,1] , space=0 , names.arg = round(theta.k[,1],2) ,

```

```

      main="Estimated Skewed Distribution (gdm function)")

#---- Model 15b: mirt (in mirt)
# define mirt model
mirtmodel <- mirt::mirt.model("
  F = 1-12
")
# get parameters
mod.pars <- mirt(dat, model=mirtmodel , pars = "values" , itemtype="Rasch")
# fix variance (just for correct counting of parameters)
mod.pars[ mod.pars$name=="COV_11" , "est"] <- FALSE
# fix item difficulty
ind <- which( ( mod.pars$item == "B1" ) & ( mod.pars$name == "d" ) )
mod.pars[ ind , "value"] <- 0
mod.pars[ ind , "est"] <- FALSE

# define prior
loglinear_prior <- function(Theta,Etable){
  TP <- nrow(Theta)
  if ( is.null(Etable) ){
    prior <- rep( 1/TP , TP )
  }
  # process Etable (this is correct for datasets without missing data)
  if ( ! is.null(Etable) ){
    # sum over correct and incorrect expected responses
    prior <- ( rowSums( Etable[ , seq(1,2*I,2)] ) + rowSums( Etable[ , seq(2,2*I,2)] ) )/I
    # smooth prior using the above design matrix and a log-linear model
    # see Xu & von Davier (2008).
    y <- log( prior + 1E-15 )
    lm1 <- lm( y ~ 0 + delta.designmatrix , weights = prior )
    prior <- exp(fitted(lm1)) # smoothed prior
  }
  prior <- prior / sum(prior)
  return(prior)
}

#* estimate model
mod15b <- mirt(dat, mirtmodel , technical = list(
  customTheta= theta.k , customPriorFun = loglinear_prior ) ,
  pars = mod.pars , verbose=TRUE )
# estimated parameters from the user customized prior distribution.
mod15b@nest <- as.integer(sum(mod.pars$est) + 3)
#* extract item parameters
coef1 <- mirt.wrapper.coef(mod15b)$coef

#** compare estimated item parameters
dfr <- data.frame( "gdm"=mod15a$item$b.Cat1 , "mirt"=coef1$d )
rownames(dfr) <- colnames(dat)
round(t(dfr),4)
##      A1      A2      A3      A4 B1      B2      B3      B4      C1      C2      C3      C4
## gdm 0.9818 0.1538 -0.7837 -1.3197 0 -1.0902 1.6088 -0.170 1.9778 0.006 1.1859 0.135
## mirt 0.9829 0.1548 -0.7826 -1.3186 0 -1.0892 1.6099 -0.169 1.9790 0.007 1.1870 0.136
# compare estimated theta distribution

```

```
dfr <- data.frame( "gdm"=mod15a$pi.k , "mirt"= mod15b@Prior[[1]] )
round(t(dfr),4)
##      1 2      3      4      5      6      7      8      9      10     11     12     13
##  gdm 0 0 1e-04 9e-04 0.0056 0.0231 0.0652 0.1299 0.1881 0.2038 0.1702 0.1129 0.0612
##  mirt 0 0 1e-04 9e-04 0.0056 0.0232 0.0653 0.1300 0.1881 0.2038 0.1702 0.1128 0.0611
##
##      14     15
##  gdm 0.0279 0.011
##  mirt 0.0278 0.011

## End(Not run)
```

---

data.reck

*Datasets from Reckase' Book Multidimensional Item Response Theory*

---

## Description

Some simulated datasets from Reckase (2009).

## Usage

```
data(data.reck21)
data(data.reck61DAT1)
data(data.reck61DAT2)
data(data.reck73C1a)
data(data.reck73C1b)
data(data.reck75C2)
data(data.reck78ExA)
data(data.reck79ExB)
```

## Format

- The format of the data.reck21 (Table 2.1, p. 45) is:
 

```
List of 2
 $ data: num [1:2500, 1:50] 0 0 0 1 1 0 0 0 1 0 ...
 ..- attr(*, "dimnames")=List of 2
 .. ..$ : NULL
 .. ..$ : chr [1:50] "I0001" "I0002" "I0003" "I0004" ...
 $ pars:'data.frame':
 ..$ a: num [1:50] 1.83 1.38 1.47 1.53 0.88 0.82 1.02 1.19 1.15 0.18 ...
 ..$ b: num [1:50] 0.91 0.81 0.06 -0.8 0.24 0.99 1.23 -0.47 2.78 -3.85 ...
 ..$ c: num [1:50] 0 0 0 0.25 0.21 0.29 0.26 0.19 0 0.21 ...
```
- The format of the datasets data.reck61DAT1 and data.reck61DAT2 (Table 6.1, p. 153) is
 

```
List of 4
 $ data : num [1:2500, 1:30] 1 0 0 1 1 0 0 1 1 0 ...
 ..- attr(*, "dimnames")=List of 2
```



```

.. ..$ : NULL
.. ..$ : chr [1:30] "A01" "A02" "A03" "A04" ...
$ pars :'data.frame':
..$ a1: num [1:30] 0.747 0.46 0.861 1.014 0.552 ...
..$ a2: num [1:30] 0.025 0.0097 0.0067 0.008 0.0204 0.0064 0.0861 ...
..$ a3: num [1:30] 0.1428 0.0692 0.404 0.047 0.1482 ...
..$ d : num [1:30] 0.183 -0.192 -0.466 -0.434 -0.443 ...
$ mu : num [1:3] -0.4 -0.7 0.1
$ sigma: num [1:3, 1:3] 1.21 0.297 1.232 0.297 0.81 ...

```

The dataset data.reck61DAT2 has correlated dimensions while data.reck61DAT1 has uncorrelated dimensions.

- Datasets data.reck73C1a and data.reck73C1b use item parameters from Table 7.3 (p. 188). The dataset C1a has uncorrelated dimensions, while C1b has perfectly correlated dimensions. The items are sensitive to 3 dimensions. The format of the datasets is

```

List of 4
$ data : num [1:2500, 1:30] 1 0 1 1 1 0 1 1 1 1 ...
.- attr(*, "dimnames")=List of 2
.. ..$ : NULL
.. ..$ : chr [1:30] "A01" "A02" "A03" "A04" ...
$ pars :'data.frame': 30 obs. of 4 variables:
..$ a1: num [1:30] 0.747 0.46 0.861 1.014 0.552 ...
..$ a2: num [1:30] 0.025 0.0097 0.0067 0.008 0.0204 0.0064 ...
..$ a3: num [1:30] 0.1428 0.0692 0.404 0.047 0.1482 ...
..$ d : num [1:30] 0.183 -0.192 -0.466 -0.434 -0.443 ...
$ mu : num [1:3] 0 0 0
$ sigma: num [1:3, 1:3] 0.167 0.236 0.289 0.236 0.334 ...

```

- The dataset data.reck75C2 is simulated using item parameters from Table 7.5 (p. 191). It contains items which are sensitive to only one dimension but individuals which have abilities in three uncorrelated dimensions. The format is

```

List of 4
$ data : num [1:2500, 1:30] 0 0 1 1 1 0 0 1 1 1 ...
.- attr(*, "dimnames")=List of 2
.. ..$ : NULL
.. ..$ : chr [1:30] "A01" "A02" "A03" "A04" ...
$ pars :'data.frame': 30 obs. of 4 variables:
..$ a1: num [1:30] 0.56 0.48 0.67 0.57 0.54 0.74 0.7 0.59 0.63 0.64 ...
..$ a2: num [1:30] 0.62 0.53 0.63 0.69 0.58 0.69 0.75 0.63 0.64 0.64 ...
..$ a3: num [1:30] 0.46 0.42 0.43 0.51 0.41 0.48 0.46 0.5 0.51 0.46 ...
..$ d : num [1:30] 0.1 0.06 -0.38 0.46 0.14 0.31 0.06 -1.23 0.47 1.06 ...
$ mu : num [1:3] 0 0 0
$ sigma: num [1:3, 1:3] 1 0 0 0 1 0 0 0 1

```

- The dataset data.reck78ExA contains simulated item responses from Table 7.8 (p. 204 ff.). There are three item clusters and two ability dimensions. The format is

```

List of 4
$ data : num [1:2500, 1:50] 0 1 1 0 1 0 0 0 0 0 ...

```

```

..- attr(*, "dimnames")=List of 2
.. ..$ : NULL
.. ..$ : chr [1:50] "A01" "A02" "A03" "A04" ...
$ pars :'data.frame': 50 obs. of 3 variables:
..$ a1: num [1:50] 0.889 1.057 1.047 1.178 1.029 ...
..$ a2: num [1:50] 0.1399 0.0432 0.016 0.0231 0.2347 ...
..$ d : num [1:50] 0.2724 1.2335 -0.0918 -0.2372 0.8471 ...
$ mu : num [1:2] 0 0
$ sigma: num [1:2, 1:2] 1 0 0 1

```

- The dataset `data.reck79ExB` contains simulated item responses from Table 7.9 (p. 207 ff.). There are three item clusters and three ability dimensions. The format is

```

List of 4
$ data : num [1:2500, 1:50] 1 1 0 1 0 0 0 1 1 0 ...
..- attr(*, "dimnames")=List of 2
.. ..$ : NULL
.. ..$ : chr [1:50] "A01" "A02" "A03" "A04" ...
$ pars :'data.frame': 50 obs. of 4 variables:
..$ a1: num [1:50] 0.895 1.032 1.036 1.163 1.022 ...
..$ a2: num [1:50] 0.052 0.132 0.144 0.13 0.165 ...
..$ a3: num [1:50] 0.0722 0.1923 0.0482 0.1321 0.204 ...
..$ d : num [1:50] 0.2724 1.2335 -0.0918 -0.2372 0.8471 ...
$ mu : num [1:3] 0 0 0
$ sigma: num [1:3, 1:3] 1 0 0 0 1 0 0 0 1

```

## Source

Simulated datasets

## References

Reckase, M. (2009). *Multidimensional item response theory*. New York: Springer.

## Examples

```

## Not run:
#####
# EXAMPLE 1: data.reck21 dataset, Table 2.1, p. 45
#####
data(data.reck21)

dat <- data.reck21$dat      # extract dataset

# items with zero guessing parameters
guess0 <- c( 1 , 2 , 3 , 9 ,11 ,27 ,30 ,35 ,45 ,49 ,50 )
I <- ncol(dat)

###
# Model 1: 3PL estimation using rasch.mm12

```

```

est.c <- est.a <- 1:I
est.c[ guess0 ] <- 0
mod1 <- sirt::rasch.mml2( dat , est.a=est.a , est.c=est.c , mmliter= 300 )
summary(mod1)

####
# Model 2: 3PL estimation using smirt
Q <- matrix(1,I,1)
mod2 <- sirt::smirt( dat , Qmatrix=Q , est.a= "2PL" , est.c=est.c , increment.factor=1.01)
summary(mod2)

####
# Model 3: estimation in mirt package
library(mirt)
itemtype <- rep("3PL" , I )
itemtype[ guess0 ] <- "2PL"
mod3 <- mirt::mirt(dat, 1, itemtype = itemtype , verbose=TRUE)
summary(mod3)

c3 <- unlist( coef(mod3) )[ 1:(4*I) ]
c3 <- matrix( c3 , I , 4 , byrow=TRUE )
# compare estimates of rasch.mml2, smirt and true parameters
round( cbind( mod1$item$c , mod2$item$c , c3[,3] , data.reck21$pars$c ) , 2 )
round( cbind( mod1$item$a , mod2$item$a.Dim1 , c3[,1] , data.reck21$pars$a ) , 2 )
round( cbind( mod1$item$b , mod2$item$b.Dim1 / mod2$item$a.Dim1 , - c3[,2] / c3[,1] ,
             data.reck21$pars$b ) , 2 )

#####
# EXAMPLE 2: data.reck61 dataset, Table 6.1, p. 153
#####

data(data.reck61DAT1)
dat <- data.reck61DAT1$data

####
# Model 1: Exploratory factor analysis

#-- Model 1a: tam.fa in TAM
library(TAM)
mod1a <- TAM::tam.fa( dat , irtmodel="efa" , nfactors=3 )
# varimax rotation
varimax(mod1a$B.stand)

# Model 1b: EFA in NOHARM (Promax rotation)
mod1b <- sirt::R2noharm( dat = dat , model.type="EFA" , dimensions = 3 ,
                       writename = "reck61__3dim_efa" , noharm.path = "c:/NOHARM" , dec = "," )
summary(mod1b)

# Model 1c: EFA with noharm.sirt
mod1c <- sirt::noharm.sirt( dat=dat , dimensions=3 )
summary(mod1c)
plot(mod1c)

```

```

# Model 1d: EFA with 2 dimensions in noharm.sirt
mod1d <- sirt::noharm.sirt( dat=dat , dimensions=2 )
summary(mod1d)
plot(mod1d , efa.load.min=.2) # plot loadings of at least .20

***
# Model 2: Confirmatory factor analysis

#-- Model 2a: tam.fa in TAM
dims <- c( rep(1,10) , rep(3,10) , rep(2,10) )
Qmatrix <- matrix( 0 , nrow=30 , ncol=3 )
Qmatrix[ cbind( 1:30 , dims) ] <- 1
mod2a <- TAM::tam.mml.2pl( dat ,Q=Qmatrix ,
                          control=list( snodes=1000, QMC=TRUE , maxiter=200) )
summary(mod2a)

#-- Model 2b: smirt in sirt
mod2b <- sirt::smirt( dat ,Qmatrix =Qmatrix , est.a="2PL" , maxiter=20 , qmcnodes=1000 )
summary(mod2b)

#-- Model 2c: rasch.mml2 in sirt
mod2c <- sirt::rasch.mml2( dat ,Qmatrix =Qmatrix , est.a= 1:30 ,
                          mmliter =200 , theta.k = seq(-5,5,len=11) )
summary(mod2c)

#-- Model 2d: mirt in mirt
cmodel <- mirt::mirt.model("
  F1 = 1-10
  F2 = 21-30
  F3 = 11-20
  COV = F1*F2, F1*F3 , F2*F3 " )
mod2d <- mirt::mirt(dat, cmodel , verbose=TRUE)
summary(mod2d)
coef(mod2d)

#-- Model 2e: CFA in NOHARM
# specify covariance pattern
P.pattern <- matrix( 1 , ncol=3 , nrow=3 )
P.init <- .4*P.pattern
diag(P.pattern) <- 0
diag(P.init) <- 1
# fix all entries in the loading matrix to 1
F.pattern <- matrix( 0 , nrow=30 , ncol=3 )
F.pattern[1:10,1] <- 1
F.pattern[21:30,2] <- 1
F.pattern[11:20,3] <- 1
F.init <- F.pattern
# estimate model
mod2e <- sirt::R2noharm( dat = dat , model.type="CFA" , P.pattern=P.pattern,
                        P.init=P.init , F.pattern=F.pattern, F.init=F.init ,
                        writename = "reck61__3dim_cfa", noharm.path = "c:/NOHARM" ,dec = "," )
summary(mod2e)

```

```

#-- Model 2f: CFA with noharm.sirt
mod2f <- sirt::noharm.sirt( dat = dat , Fval=F.init , Fpatt = F.pattern ,
                          Pval=P.init , Ppatt = P.pattern )
summary(mod2f)

#####
# EXAMPLE 3: DETECT analysis data.reck78ExA and data.reck79ExB
#####

data(data.reck78ExA)
data(data.reck79ExB)

#*****
# Example A
dat <- data.reck78ExA$data
#- estimate person score
score <- qnorm( ( rowMeans( dat )+.5 ) / ( ncol(dat) + 1 ) )
#- extract item cluster
itemcluster <- substring( colnames(dat) , 1 , 1 )
#- confirmatory DETECT Item cluster
detectA <- sirt::conf.detect( data = dat , score = score , itemcluster = itemcluster )
##          unweighted weighted
## DETECT      0.571    0.571
## ASSI        0.523    0.523
## RATIO       0.757    0.757

#- exploratory DETECT analysis
detect_explA <- sirt::expl.detect(data=dat, score, nclusters=10, N.est = nrow(dat)/2 )
## Optimal Cluster Size is 5 (Maximum of DETECT Index)
##      N.Cluster N.items N.est N.val      size.cluster DETECT.est ASSI.est
## 1         2      50 1250 1250          31-19      0.531  0.404
## 2         3      50 1250 1250          10-19-21    0.554  0.407
## 3         4      50 1250 1250          10-19-14-7    0.630  0.509
## 4         5      50 1250 1250          10-19-3-7-11    0.653  0.546
## 5         6      50 1250 1250          10-12-7-3-7-11    0.593  0.458
## 6         7      50 1250 1250          10-12-7-3-7-9-2    0.604  0.474
## 7         8      50 1250 1250          10-12-7-3-3-9-4-2    0.608  0.481
## 8         9      50 1250 1250          10-12-7-3-3-5-4-2-4    0.617  0.494
## 9         10     50 1250 1250          10-5-7-7-3-3-5-4-2-4    0.592  0.460

# cluster membership
cluster_membership <- detect_explA$itemcluster$cluster3
# Cluster 1:
colnames(dat)[ cluster_membership == 1 ]
## [1] "A01" "A02" "A03" "A04" "A05" "A06" "A07" "A08" "A09" "A10"
# Cluster 2:
colnames(dat)[ cluster_membership == 2 ]
## [1] "B11" "B12" "B13" "B14" "B15" "B16" "B17" "B18" "B19" "B20" "B21" "B22"
## [13] "B23" "B25" "B26" "B27" "B28" "B29" "B30"
# Cluster 3:
colnames(dat)[ cluster_membership == 3 ]
## [1] "B24" "C31" "C32" "C33" "C34" "C35" "C36" "C37" "C38" "C39" "C40" "C41"
## [13] "C42" "C43" "C44" "C45" "C46" "C47" "C48" "C49" "C50"

```

```

#*****
# Example B
dat <- data.reck79ExB$data
#- estimate person score
score <- qnorm( ( rowMeans( dat )+.5 ) / ( ncol(dat) + 1 ) )
#- extract item cluster
itemcluster <- substring( colnames(dat) , 1 , 1 )
#- confirmatory DETECT Item cluster
detectB <- sirt::conf.detect( data = dat , score = score , itemcluster = itemcluster )
##           unweighted weighted
## DETECT      0.715    0.715
## ASSI        0.624    0.624
## RATIO       0.855    0.855

#- exploratory DETECT analysis
detect_explB <- sirt::expl.detect(data=dat, score, nclusters=10, N.est = nrow(dat)/2 )
## Optimal Cluster Size is 4 (Maximum of DETECT Index)
##
##      N.Cluster N.items N.est N.val      size.cluster DETECT.est ASSI.est
## 1           2      50 1250 1250          30-20      0.665    0.546
## 2           3      50 1250 1250          10-20-20     0.686    0.585
## 3           4      50 1250 1250          10-20-8-12     0.728    0.644
## 4           5      50 1250 1250          10-6-14-8-12    0.654    0.553
## 5           6      50 1250 1250          10-6-14-3-12-5   0.659    0.561
## 6           7      50 1250 1250          10-6-14-3-7-5-5   0.664    0.576
## 7           8      50 1250 1250          10-6-7-7-3-7-5-5   0.616    0.518
## 8           9      50 1250 1250          10-6-7-7-3-5-5-5-2 0.612    0.512
## 9          10      50 1250 1250          10-6-7-7-3-5-3-5-2-2 0.613    0.512

## End(Not run)

```

---

data.sirt

*Some Example Datasets for the sirt Package*

---

## Description

Some example datasets for the sirt package.

## Usage

```

data(data.si01)
data(data.si02)
data(data.si03)
data(data.si04)
data(data.si05)
data(data.si06)

```

**Format**

- The format of the dataset `data.si01` is:
 

```
'data.frame': 1857 obs. of 3 variables:
 $ idgroup: int 1 1 1 1 1 1 1 1 1 1 ...
 $ item1 : int NA NA NA NA NA NA NA NA NA NA ...
 $ item2 : int 4 4 4 4 4 4 4 2 4 4 ...
```
- The dataset `data.si02` is the Stouffer-Toby-dataset published in Lindsay, Clogg and Grego (1991; Table 1, p.97, Cross-classification A):
 

```
List of 2
 $ data : num [1:16, 1:4] 1 0 1 0 1 0 1 0 1 0 ...
 ..- attr(*, "dimnames")=List of 2
 .. ..$ : NULL
 .. ..$ : chr [1:4] "I1" "I2" "I3" "I4"
 $ weights: num [1:16] 42 1 6 2 6 1 7 2 23 4 ...
```
- The format of the dataset `data.si03` (containing item parameters of two studies) is:
 

```
'data.frame': 27 obs. of 3 variables:
 $ item : Factor w/ 27 levels "M1","M10","M11",...: 1 12 21 22 ...
 $ b_study1: num 0.297 1.163 0.151 -0.855 -1.653 ...
 $ b_study2: num 0.72 1.118 0.351 -0.861 -1.593 ...
```
- The dataset `data.si04` is adapted from Bartolucci, Montanari and Pandolfi (2012; Table 4, Table 7). The data contains 4999 persons, 79 items on 5 dimensions.
 

```
List of 3
 $ data : num [1:4999, 1:79] 0 1 1 0 1 1 0 0 1 1 ...
 ..- attr(*, "dimnames")=List of 2
 .. ..$ : NULL
 .. ..$ : chr [1:79] "A01" "A02" "A03" "A04" ...
 $ itempars :'data.frame': 79 obs. of 4 variables:
 ..$ item : Factor w/ 79 levels "A01","A02","A03",...: 1 2 3 4 5 6 7 8 9 10 ...
 ..$ dim : num [1:79] 1 1 1 1 1 1 1 1 1 1 ...
 ..$ gamma : num [1:79] 1 1 1 1 1 1 1 1 1 1 ...
 ..$ gamma.beta: num [1:79] -0.189 0.25 0.758 1.695 1.022 ...
 $ distribution: num [1:9, 1:7] 1 2 3 4 5 ...
 ..- attr(*, "dimnames")=List of 2
 .. ..$ : NULL
 .. ..$ : chr [1:7] "class" "A" "B" "C" ...
```
- The dataset `data.si05` contains double ratings of two exchangeable raters for three items which are in Ex1, Ex2 and Ex3, respectively.
 

```
List of 3
 $ Ex1:'data.frame': 199 obs. of 2 variables:
 ..$ C7040: num [1:199] NA 1 0 1 1 0 0 0 1 0 ...
 ..$ C7041: num [1:199] 1 1 0 0 0 0 0 0 1 0 ...
 $ Ex2:'data.frame': 2000 obs. of 2 variables:
```

```

..$ rater1: num [1:2000] 2 0 3 1 2 2 0 0 0 0 ...
..$ rater2: num [1:2000] 4 1 3 2 1 0 0 0 0 2 ...
$ Ex3:'data.frame': 2000 obs. of 2 variables:
..$ rater1: num [1:2000] 5 1 6 2 3 3 0 0 0 0 ...
..$ rater2: num [1:2000] 7 2 6 3 2 1 0 1 0 3 ...

```

- The dataset `data.si06` contains multiple choice item responses. The correct alternative is denoted as 0, distractors are indicated by the codes 1, 2 or 3.

```

'data.frame': 4441 obs. of 14 variables:
$ WV01: num 0 0 0 0 0 0 0 0 0 3 ...
$ WV02: num 0 0 0 3 0 0 0 0 0 1 ...
$ WV03: num 0 1 0 0 0 0 0 0 0 0 ...
$ WV04: num 0 0 0 0 0 0 0 0 0 1 ...
$ WV05: num 3 1 1 1 0 0 1 1 0 2 ...
$ WV06: num 0 1 3 0 0 0 2 0 0 1 ...
$ WV07: num 0 0 0 0 0 0 0 0 0 0 ...
$ WV08: num 0 1 1 0 0 0 0 0 0 0 ...
$ WV09: num 0 0 0 0 0 0 0 0 0 2 ...
$ WV10: num 1 1 3 0 0 2 0 0 0 0 ...
$ WV11: num 0 0 0 0 0 0 0 0 0 0 ...
$ WV12: num 0 0 0 2 0 0 2 0 0 0 ...
$ WV13: num 3 1 1 3 0 0 3 0 0 0 ...
$ WV14: num 3 1 2 3 0 3 1 3 3 0 ...

```

## References

Bartolucci, F., Montanari, G. E., & Pandolfi, S. (2012). Dimensionality of the latent structure and item selection via latent class multidimensional IRT models. *Psychometrika*, *77*, 782-802.

Lindsay, B., Clogg, C. C., & Grego, J. (1991). Semiparametric estimation in the Rasch model and related exponential response models, including a simple latent class model for item analysis. *Journal of the American Statistical Association*, *86*, 96-107.

## See Also

Some free datasets can be obtained from

Psychological questionnaires: [http://personality-testing.info/\\_rawdata/](http://personality-testing.info/_rawdata/)

PISA 2012: <http://pisa2012.acer.edu.au/downloads.php>

PIAAC: <http://www.oecd.org/site/piaac/publicdataandanalysis.htm>

TIMSS 2011: <http://timssandpirls.bc.edu/timss2011/international-database.html>

ALLBUS: <http://www.gesis.org/allbus/datenzugang/>

## Examples

```

## Not run:
#####
# EXAMPLE 1: Nested logit model multiple choice dataset data.si06
#####

```



```

data(data.si06)
dat <- data.si06

*** estimate 2PL nested logit model
library(mirt)
mod1 <- mirt::mirt( dat , model=1 , itemtype="2PLNRM" , key=rep(0,ncol(dat)) ,
                  verbose=TRUE )
summary(mod1)
cmod1 <- mirt.wrapper.coef(mod1)$coef
cmod1[,-1] <- round( cmod1[,-1] , 3)

*** normalize item parameters according Suh and Bolt (2010)
cmod2 <- cmod1

# slope parameters
ind <- grep("ak",colnames(cmod2))
h1 <- cmod2[ ,ind ]
cmod2[,ind] <- t( apply( h1 , 1 , FUN = function(l1){ l1 - mean(l1) } ) )
# item intercepts
ind <- paste0( "d" , 0:9 )
ind <- which( colnames(cmod2) %in% ind )
h1 <- cmod2[ ,ind ]
cmod2[,ind] <- t( apply( h1 , 1 , FUN = function(l1){ l1 - mean(l1) } ) )
cmod2[,-1] <- round( cmod2[,-1] , 3)

## End(Not run)

```

---

data.timss

*Dataset TIMSS Mathematics*


---

## Description

This datasets contains TIMSS mathematics data from 345 students on 25 items.

## Usage

```
data(data.timss)
```

## Format

This dataset is a list. `data` is the dataset containing student ID (`idstud`), a dummy variable for female (`girl`) and student age (`age`). The following variables (starting with `M` in the variable name are items.

The format is:

List of 2

```
$ data: 'data.frame':
```

```
..$ idstud : num [1:345] 4e+09 4e+09 4e+09 4e+09 4e+09 ...
```

```
..$ girl : int [1:345] 0 0 0 0 0 0 0 1 0 ...
```

```
..$ age : num [1:345] 10.5 10 10.25 10.25 9.92 ...
```

```

..$ M031286 : int [1:345] 0 0 0 1 1 0 1 0 1 0 ...
..$ M031106 : int [1:345] 0 0 0 1 1 0 1 1 0 0 ...
..$ M031282 : int [1:345] 0 0 0 1 1 0 1 1 0 0 ...
..$ M031227 : int [1:345] 0 0 0 0 1 0 0 0 0 0 ...
[...]
..$ M041203 : int [1:345] 0 0 0 1 1 0 0 0 0 1 ...
$ item:'data.frame':
..$ item          : Factor w/ 25 levels "M031045","M031068",...: ...
..$ Block         : Factor w/ 2 levels "M01","M02": 1 1 1 1 1 1 ..
..$ Format         : Factor w/ 2 levels "CR","MC": 1 1 1 1 2 ...
..$ Content.Domain : Factor w/ 3 levels "Data Display",...: 3 3 3 3 ...
..$ Cognitive.Domain: Factor w/ 3 levels "Applying","Knowing",...: 2 3 3 ..

```

---

data.timss07.G8.RUS    *TIMSS 2007 Grade 8 Mathematics and Science Russia*

---

### Description

This TIMSS 2007 dataset contains item responses of 4472 eighth grade Russian students in Mathematics and Science.

### Usage

```
data(data.timss07.G8.RUS)
```

### Format

The datasets contains raw responses (raw), scored responses (scored) and item informations (iteminfo).

The format of the dataset is:

List of 3

```

$ raw      : 'data.frame':
..$ idstud  : num [1:4472] 3010101 3010102 3010104 3010105 3010106 ...
..$ M022043 : atomic [1:4472] NA 1 4 NA NA NA NA NA NA NA ...
.. ..- attr(*, "value.labels")= Named num [1:7] 9 6 5 4 3 2 1
.. .. ..- attr(*, "names")= chr [1:7] "OMITTED" "NOT REACHED" "E" "D*" ...
[...]
..$ M032698 : atomic [1:4472] NA NA NA NA NA NA NA NA 2 1 NA ...
.. ..- attr(*, "value.labels")= Named num [1:6] 9 6 4 3 2 1
.. .. ..- attr(*, "names")= chr [1:6] "OMITTED" "NOT REACHED" "D" "C" ...
..$ M032097 : atomic [1:4472] NA NA NA NA NA NA NA NA 2 3 NA ...
.. ..- attr(*, "value.labels")= Named num [1:6] 9 6 4 3 2 1
.. .. ..- attr(*, "names")= chr [1:6] "OMITTED" "NOT REACHED" "D" "C*" ...
.. [list output truncated]
$ scored   : num [1:4472, 1:443] 3010101 3010102 3010104 3010105 3010106 ...
..- attr(*, "dimnames")=List of 2
.. ..$ : NULL

```

```

.. ..$ : chr [1:443] "idstud" "M022043" "M022046" "M022049" ...
$ iteminfo:'data.frame':
..$ item      : Factor w/ 442 levels "M022043","M022046",...: 1 2 3 4 5 6 21 7 8 17 ...
..$ content   : Factor w/ 8 levels "Algebra","Biology",...: 7 7 6 1 6 7 4 6 7 7 ...
..$ topic     : Factor w/ 49 levels "Algebraic Expression",...: 32 32 41 29 ...
..$ cognitive : Factor w/ 3 levels "Applying","Knowing",...: 2 1 3 2 1 1 1 1 2 1 ...
..$ item.type : Factor w/ 2 levels "CR","MC": 2 1 2 2 1 2 2 2 1 ...
..$ N.options : Factor w/ 4 levels "-", " -", "4", "5": 4 1 3 4 1 4 4 4 3 1 ...
..$ key       : Factor w/ 7 levels "-", " -", "A", "B",...: 6 1 6 7 1 5 5 4 6 1 ...
..$ max.points: int [1:442] 1 1 1 1 1 1 1 1 1 2 ...
..$ item.label: Factor w/ 432 levels "1 teacher for every 12 students ",...: 58 351 ...

```

## Source

TIMSS 2007 8th Grade, Russian Sample

---

data.wide2long

*Converting a Data Frame from Wide Format in a Long Format*

---

## Description

Converts a data frame in wide format into long format.

## Usage

```
data.wide2long(dat, id = NULL, X = NULL, Q = NULL)
```

## Arguments

dat	Data frame with item responses and a person identifier if <code>id != NULL</code> .
id	An optional string with the variable name of the person identifier.
X	Data frame with person covariates for inclusion in the data frame of long format
Q	Data frame with item predictors. Item labels must be included as a column named by "item".

## Value

Data frame in long format

## Author(s)

Alexander Robitzsch

## Examples

```

## Not run:
#####
# EXAMPLE 1: data.pisaRead
#####
miceadds::library_install("lme4")

data(data.pisaRead)
dat <- data.pisaRead$data
Q <- data.pisaRead$item # item predictors

# define items
items <- colnames(dat)[ substring( colnames(dat) , 1 , 1 ) == "R" ]
dat1 <- dat[ , c( "idstud" , items ) ]
# matrix with person predictors
X <- dat[ , c("idschool" , "hisei" , "female" , "migra" ) ]

# create dataset in long format
dat.long <- data.wide2long( dat=dat1 , id="idstud" , X=X , Q=Q )

###
# Model 1: Rasch model
mod1 <- lme4::glmer( resp ~ 0 + ( 1 | idstud ) + as.factor(item) , data = dat.long ,
  family="binomial" , verbose=TRUE)
summary(mod1)

###
# Model 2: Rasch model and inclusion of person predictors
mod2 <- lme4::glmer( resp ~ 0 + ( 1 | idstud ) + as.factor(item) + female + hisei + migra,
  data = dat.long , family="binomial" , verbose=TRUE)
summary(mod2)

###
# Model 3: LLTM
mod3 <- lme4::glmer(resp ~ (1|idstud) + as.factor(ItemFormat) + as.factor(TextType),
  data = dat.long , family="binomial" , verbose=TRUE)
summary(mod3)

#####
# SIMULATED EXAMPLE 2: Rasch model in lme4
#####

set.seed(765)
N <- 1000 # number of persons
I <- 10 # number of items
b <- seq(-2,2,length=I)
dat <- sirt::sim.raschtype( rnorm(N,sd=1.2) , b=b )
dat.long <- data.wide2long( dat=dat )
###
# estimate Rasch model with lmer
library(lme4)
mod1 <- lme4::glmer( resp ~ 0 + as.factor( item ) + ( 1 | id_index) , data = dat.long ,

```

```

      verbose=TRUE , family="binomial")
summary(mod1)
## Random effects:
##   Groups   Name      Variance Std.Dev.
##   id_index (Intercept) 1.454    1.206
## Number of obs: 10000, groups: id_index, 1000
##
## Fixed effects:
##              Estimate Std. Error z value Pr(>|z|)
## as.factor(item)I0001  2.16365    0.10541  20.527 < 2e-16 ***
## as.factor(item)I0002  1.66437    0.09400  17.706 < 2e-16 ***
## as.factor(item)I0003  1.21816    0.08700  14.002 < 2e-16 ***
## as.factor(item)I0004  0.68611    0.08184   8.383 < 2e-16 ***
## [...]

## End(Not run)

```

---

detect.index

*Calculation of the DETECT and polyDETECT Index*


---

## Description

This function calculated the DETECT and polyDETECT index (Stout, Habing, Douglas & Kim, 1996; Zhang & Stout, 1999a; Zhang, 2007). At first, conditional covariances have to be estimated using the `ccov.np` function.

## Usage

```
detect.index(ccovtable, itemcluster)
```

## Arguments

<code>ccovtable</code>	A value of <code>ccov.np</code> .
<code>itemcluster</code>	Item cluster for each item. The order of entries must correspond to the columns in data (submitted to <code>ccov.np</code> ).

## Author(s)

Alexander Robitzsch

## References

- Stout, W., Habing, B., Douglas, J., & Kim, H. R. (1996). Conditional covariance-based nonparametric multidimensionality assessment. *Applied Psychological Measurement*, **20**, 331-354.
- Zhang, J., & Stout, W. (1999a). Conditional covariance structure of generalized compensatory multidimensional items. *Psychometrika*, **64**, 129-152.
- Zhang, J., & Stout, W. (1999b). The theoretical DETECT index of dimensionality and its application to approximate simple structure. *Psychometrika*, **64**, 213-249.

Zhang, J. (2007). Conditional covariance theory and DETECT for polytomous items. *Psychometrika*, **72**, 69-91.

### See Also

For examples see [conf.detect](#).

---

dif.logistic.regression

*Differential Item Functioning using Logistic Regression Analysis*

---

### Description

This function estimates differential item functioning using a logistic regression analysis (Zumbo, 1999).

### Usage

```
dif.logistic.regression(dat, group, score, quant=1.645)
```

### Arguments

dat	Data frame with dichotomous item responses
group	Group identifier
score	Ability estimate, e.g. the WLE.
quant	Used quantile of the normal distribution for assessing statistical significance

### Details

Items are classified into A (negligible DIF), B (moderate DIF) and C (large DIF) levels according to the ETS classification system (Longford, Holland & Thayer, 1993, p. 175). See also Monahan et al. (2007) for further DIF effect size classifications.

### Value

A data frame with following variables:

itemnr	Numeric index of the item
sortDIFindex	Rank of item with respect to the uniform DIF (from negative to positive values)
item	Item name
N	Sample size per item
R	Value of group variable for reference group
F	Value of group variable for focal group
nR	Sample size per item in reference group
nF	Sample size per item in focal group

p	Item $p$ value
pR	Item $p$ value in reference group
pF	Item $p$ value in focal group
pdiff	Item $p$ value differences
pdiff.adj	Adjusted $p$ value difference
uniformDIF	Uniform DIF estimate
se.uniformDIF	Standard error of uniform DIF
t.uniformDIF	The $t$ value for uniform DIF
sig.uniformDIF	Significance label for uniform DIF
DIF.ETS	DIF classification according to the ETS classification system (see Details)
uniform.EBDIF	Empirical Bayes estimate of uniform DIF (Longford, Holland & Thayer, 1993) which takes degree of DIF standard error into account
DIF.SD	Value of the DIF standard deviation
nonuniformDIF	Nonuniform DIF estimate
se.nonuniformDIF	Standard error of nonuniform DIF
t.nonuniformDIF	The $t$ value for nonuniform DIF
sig.nonuniformDIF	Significance label for nonuniform DIF

**Author(s)**

Alexander Robitzsch

**References**

- Longford, N. T., Holland, P. W., & Thayer, D. T. (1993). Stability of the MH D-DIF statistics across populations. In P. W. Holland & H. Wainer (Eds.). *Differential Item Functioning* (pp. 171-196). Hillsdale, NJ: Erlbaum.
- Monahan, P. O., McHorney, C. A., Stump, T. E., & Perkins, A. J. (2007). Odds ratio, delta, ETS classification, and standardization measures of DIF magnitude for binary logistic regression. *Journal of Educational and Behavioral Statistics*, **32**, 92-109.
- Zumbo, B. D. (1999). *A handbook on the theory and methods of differential item functioning (DIF): Logistic regression modeling as a unitary framework for binary and Likert-type (ordinal) item scores*. Ottawa ON: Directorate of Human Resources Research and Evaluation, Department of National Defense.

**See Also**

For assessing DIF variance see [dif.variance](#) and [dif.strata.variance](#)

See also [rasch.evm.pcm](#) for assessing differential item functioning in the partial credit model.

See the **difR** package for a large collection of DIF detection methods.

For a download of the free *DIF-Pack* software (SIBTEST, ...) see <http://psychometrictools.measuredprogress.org/home>.

**Examples**

```
#####
# EXAMPLE 1: Mathematics data | Gender DIF
#####

data( data.math )
dat <- data.math$data
items <- grep( "M" , colnames(dat))

# estimate item parameters and WLEs
mod <- rasch.mml2( dat[,items] )
wle <- wle.rasch( dat[,items] , b=mod$item$b )$theta

# assess DIF by logistic regression
mod1 <- dif.logistic.regression( dat=dat[,items] , score=wle , group=dat$female)

# calculate DIF variance
dif1 <- dif.variance( dif=mod1$uniformDIF , se.dif = mod1$se.uniformDIF )
dif1$unweighted.DIFSD
## > dif1$unweighted.DIFSD
## [1] 0.1963958

# calculate stratified DIF variance
# stratification based on domains
dif2 <- dif.strata.variance( dif=mod1$uniformDIF , se.dif = mod1$se.uniformDIF ,
  itemcluster = data.math$item$domain )
## $unweighted.DIFSD
## [1] 0.1455916

## Not run:
#####
# Likelihood ratio test and graphical model test in eRm package
miceadds::library_install("eRm")
# estimate Rasch model
res <- eRm::RM( dat[,items] )
summary(res)
# LR-test with respect to female
lrres <- eRm::LRtest(res, splitcr = dat$female)
summary(lrres)
# graphical model test
eRm::plotGOF(lrres)

## End(Not run)
```

---

dif.strata.variance     *Stratified DIF Variance*

---

**Description**

Calculation of stratified DIF variance



**Usage**

```
dif.strata.variance(dif, se.dif, itemcluster)
```

**Arguments**

dif	Vector of uniform DIF effects
se.dif	Standard error of uniform DIF effects
itemcluster	Vector of item strata

**Value**

A list with following entries:

stratadif	Summary statistics of DIF effects within item strata
weighted.DIFSD	Weighted DIF standard deviation
unweighted.DIFSD	DIF standard deviation

**Author(s)**

Alexander Robitzsch

**References**

Longford, N. T., Holland, P. W., & Thayer, D. T. (1993). Stability of the MH D-DIF statistics across populations. In P. W. Holland & H. Wainer (Eds.). *Differential Item Functioning* (pp. 171-196). Hillsdale, NJ: Erlbaum.

**See Also**

See [dif.logistic.regression](#) for examples.

---

dif.variance

*DIF Variance*

---

**Description**

This function calculates the variance of DIF effects, the so called DIF variance (Longford, Holland & Thayer, 1993).

**Usage**

```
dif.variance(dif, se.dif, items = paste("item", 1:length(dif), sep = "")) )
```

**Arguments**

dif	Vector of uniform DIF effects
se.dif	Standard error of uniform DIF effects
items	Optional vector of item names

**Value**

A list with following entries

weighted.DIFSD	Weighted DIF standard deviation
unweighted.DIFSD	DIF standard deviation
mean.se.dif	Mean of standard errors of DIF effects
eb.dif	Empirical Bayes estimates of DIF effects

**Author(s)**

Alexander Robitzsch

**References**

Longford, N. T., Holland, P. W., & Thayer, D. T. (1993). Stability of the MH D-DIF statistics across populations. In P. W. Holland & H. Wainer (Eds.). *Differential Item Functioning* (pp. 171-196). Hillsdale, NJ: Erlbaum.

**See Also**

See [dif.logistic.regression](#) for examples.

---

dirichlet.mle

*Maximum Likelihood Estimation of the Dirichlet Distribution*

---

**Description**

Maximum likelihood estimation of the parameters of the Dirichlet distribution

**Usage**

```
dirichlet.mle(x, weights=NULL, eps = 10-5, convcrit = 1e-05, maxit=1000,  
oldfac = .3, progress=FALSE)
```

**Arguments**

x	Data frame with $N$ observations and $K$ variables of a Dirichlet distribution
weights	Optional vector of frequency weights
eps	Tolerance number which is added to prevent from logarithms of zero
convcrit	Convergence criterion
maxit	Maximum number of iterations
oldfac	Covergence acceleration factor. It must be a parameter between 0 and 1.
progress	Display iteration progress?

**Value**

A list with following entries

alpha	Vector of $\alpha$ parameters
alpha0	The concentration parameter $\alpha_0 = \sum_k \alpha_k$
xsi	Vector of proportions $\xi_k = \alpha_k / \alpha_0$

**Author(s)**

Alexander Robitzsch

**References**

Minka, T. P. (2012). *Estimating a Dirichlet distribution*. Technical Report.

**See Also**

For simulating Dirichlet vectors with matrix-wise  $\alpha$  parameters see [dirichlet.simul](#).

For a variety of functions concerning the Dirichlet distribution see the **DirichletReg** package.

**Examples**

```
#####
# SIMULATED EXAMPLE 1: Simulate and estimate Dirichlet distribution
#####

# (1) simulate data
set.seed(789)
N <- 200
probs <- c(.5 , .3 , .2 )
alpha0 <- .5
alpha <- alpha0*probs
alpha <- matrix( alpha , nrow=N , ncol=length(alpha) , byrow=TRUE )
x <- dirichlet.simul( alpha )

# (2) estimate Dirichlet parameters
dirichlet.mle(x)
## $alpha
```

```

## [1] 0.24507708 0.14470944 0.09590745
## $alpha0
## [1] 0.485694
## $xsi
## [1] 0.5045916 0.2979437 0.1974648

## Not run:
#####
# SIMULATED EXAMPLE 2: Fitting Dirichlet distribution with frequency weights
#####

# define observed data
x <- scan( nlines=1)
  1 0  0 1  .5 .5
x <- matrix( x , nrow=3 , ncol=2 , byrow=TRUE)

# transform observations x into (0,1)
eps <- .01
x <- ( x + eps ) / ( 1 + 2 * eps )

# compare results with likelihood fitting package maxLik
miceadds::library_install("maxLik")
# define likelihood function
dirichlet.ll <- function(param) {
  ll <- sum( weights * log( ddirichlet( x , param ) ) )
  ll
}

### weights 10-10-1
weights <- c(10, 10 , 1 )
mod1a <- dirichlet.mle( x , weights= weights )
mod1a
# estimation in maxLik
mod1b <- maxLik::maxLik(loglik, start=c(.5,.5))
print( mod1b )
coef( mod1b )

### weights 10-10-10
weights <- c(10, 10 , 10 )
mod2a <- dirichlet.mle( x , weights= weights )
mod2a
# estimation in maxLik
mod2b <- maxLik::maxLik(loglik, start=c(.5,.5))
print( mod2b )
coef( mod2b )

### weights 30-10-2
weights <- c(30, 10 , 2 )
mod3a <- dirichlet.mle( x , weights= weights )
mod3a
# estimation in maxLik
mod3b <- maxLik::maxLik(loglik, start=c(.25,.25))
print( mod3b )

```

```
coef( mod3b )
## End(Not run)
```

---

```
dirichlet.simul      Simulation of a Dirichlet Distributed Vectors
```

---

## Description

This function makes random draws from a Dirichlet distribution.

## Usage

```
dirichlet.simul(alpha)
```

## Arguments

alpha            A matrix with  $\alpha$  parameters of the Dirichlet distribution

## Value

A data frame with Dirichlet distributed responses

## Author(s)

Alexander Robitzsch

## Examples

```
#####
# EXAMPLE 1: Simulation with two components
#####

set.seed(789)
N <- 2000
probs <- c(.7 , .3) # define (extremal) class probabilities

### alpha0 = .2 -> nearly crisp latent classes
alpha0 <- .2
alpha <- alpha0*probs
alpha <- matrix( alpha , nrow=N , ncol=length(alpha) , byrow=TRUE )
x <- dirichlet.simul( alpha )
htitle <- expression(paste( alpha[0], "=.2, ", p[1] , "=".7"  ) )
hist( x[,1] , breaks = seq(0,1,len=20) , main=htitle)

### alpha0 = 3 -> strong deviation from crisp membership
alpha0 <- 3
alpha <- alpha0*probs
alpha <- matrix( alpha , nrow=N , ncol=length(alpha) , byrow=TRUE )
x <- dirichlet.simul( alpha )
```

```

htitle <- expression(paste( alpha[0], "=3, ", p[1] , "=.7"  ) )
hist( x[,1] , breaks = seq(0,1,len=20) , main=htitle)

## Not run:
#####
# EXAMPLE 2: Simulation with three components
#####

set.seed(986)
N <- 2000
probs <- c( .5 , .35 , .15 )

*** alpha0 = .2
alpha0 <- .2
alpha <- alpha0*probs
alpha <- matrix( alpha , nrow=N , ncol=length(alpha) , byrow=TRUE )
x <- dirichlet.simul( alpha )
htitle <- expression(paste( alpha[0], "=.2, ", p[1] , "=.7"  ) )
miceadds::library_install("ade4")
ade4::triangle.plot(x, label=NULL , clabel = 1)

*** alpha0 = 3
alpha0 <- 3
alpha <- alpha0*probs
alpha <- matrix( alpha , nrow=N , ncol=length(alpha) , byrow=TRUE )
x <- dirichlet.simul( alpha )
htitle <- expression(paste( alpha[0], "=3, ", p[1] , "=.7"  ) )
ade4::triangle.plot(x, label=NULL , clabel = 1)

## End(Not run)

```

---

eigenvalues.manymatrices

*Computation of Eigenvalues of Many Symmetric Matrices*

---

## Description

This function computes the eigenvalue decomposition of  $N$  symmetric positive definite matrices. The eigenvalues are computed by the Rayleigh quotient method (Lange, 2010, p. 120). In addition, the inverse matrix can be calculated.

## Usage

```
eigenvalues.manymatrices(Sigma.all, itermax = 10, maxconv = 0.001,
  inverse=FALSE )
```

## Arguments

`Sigma.all` An  $N \times D^2$  matrix containing the  $D^2$  entries of  $N$  symmetric matrices of dimension  $D \times D$

itermax	Maximum number of iterations
maxconv	Convergence criterion for convergence of eigenvectors
inverse	A logical which indicates if the inverse matrix shall be calculated

**Value**

A list with following entries

lambda	Matrix with eigenvalues
U	An $N \times D^2$ Matrix of orthonormal eigenvectors
logdet	Vector of logarithm of determinants
det	Vector of determinants
Sigma.inv	Inverse matrix if inverse=TRUE.

**Author(s)**

Alexander Robitzsch

**References**

Lange, K. (2010). *Numerical Analysis for Statisticians*. New York: Springer.

**Examples**

```
# define matrices
Sigma <- diag(1,3)
Sigma[ lower.tri(Sigma) ] <- Sigma[ upper.tri(Sigma) ] <- c(.4,.6,.8 )
Sigma1 <- Sigma

Sigma <- diag(1,3)
Sigma[ lower.tri(Sigma) ] <- Sigma[ upper.tri(Sigma) ] <- c(.2,.1,.99 )
Sigma2 <- Sigma

# collect matrices in a "super-matrix"
Sigma.all <- rbind( matrix( Sigma1 , nrow=1 , byrow=TRUE) ,
                  matrix( Sigma2 , nrow=1 , byrow=TRUE) )
Sigma.all <- Sigma.all[ c(1,1,2,2,1 ) , ]

# eigenvalue decomposition
m1 <- eigenvalues.manymatrices( Sigma.all )
m1

# eigenvalue decomposition for Sigma1
s1 <- svd(Sigma1)
s1
```

---

eigenvalues.sirt      *First Eigenvalues of a Symmetric Matrix*

---

### Description

This function computes the first  $D$  eigenvalues and eigenvectors of a symmetric positive definite matrices. The eigenvalues are computed by the Rayleigh quotient method (Lange, 2010, p. 120).

### Usage

```
eigenvalues.sirt( X , D , maxit=200 , conv=10-6 )
```

### Arguments

X	Symmetric matrix
D	Number of eigenvalues to be estimated
maxit	Maximum number of iterations
conv	Convergence criterion

### Value

A list with following entries:

d	Vector of eigenvalues
u	Matrix with eigenvectors in columns

### Author(s)

Alexander Robitzsch

### References

Lange, K. (2010). *Numerical Analysis for Statisticians*. New York: Springer.

### Examples

```
Sigma <- diag(1,3)
Sigma[ lower.tri(Sigma) ] <- Sigma[ upper.tri(Sigma) ] <- c(.4,.6,.8 )
eigenvalues.sirt(X=Sigma, D=2 )
# compare with svd function
svd(Sigma)
```



---

 equating.rasch

*Equating in the Generalized Logistic Rasch Model*


---

### Description

This function does the linking in the generalized logistic item response model. Only item difficulties ( $b$  item parameters) are allowed. Mean-mean linking and the methods of Haebara and Stocking-Lord are implemented (Kolen & Brennan, 2004).

### Usage

```
equating.rasch(x, y, theta = seq(-4, 4, len = 100),
              alpha1 = 0, alpha2 = 0)
```

### Arguments

x	Matrix with two columns: First column items, second column item difficulties
y	Matrix with two columns: First columns item, second column item difficulties
theta	Vector of theta values at which the linking functions should be evaluated. If a weighting according to a prespecified normal distribution $N(\mu, \sigma^2)$ is aimed, then choose <code>theta=qnorm( seq(.001 , .999 , len=100) , mean=mu, sd=sigma)</code>
alpha1	Fixed $\alpha_1$ parameter in the generalized item response model
alpha2	Fixed $\alpha_2$ parameter in the generalized item response model

### Value

B.est	Estimated linking constants according to the methods Mean.Mean (Mean-mean linking), Haebara (Haebara method) and Stocking.Lord (Stocking-Lord method).
descriptives	Descriptives of the linking. The linking error (linkerror) is calculated under the assumption of simple random sampling of items
anchor	Original and transformed item parameters of anchor items
transf.par	Original and transformed item parameters of all items

### Author(s)

Alexander Robitzsch

### References

Kolen, M. J., & Brennan, R. L. (2004). *Test Equating, Scaling, and Linking: Methods and Practices*. New York: Springer.

**See Also**

For estimating standard errors (due to inference with respect to the item domain) of this procedure see [equating.rasch.jackknife](#).

For linking several studies see [linking.haberman](#) or [invariance.alignment](#).

A robust alternative to mean-mean linking is implemented in [linking.robust](#).

For linking under more general item response models see the **plink** package.

**Examples**

```
#####
# EXAMPLE 1: Linking item parameters of the PISA study
#####

data(data.pisaPars)
pars <- data.pisaPars

# linking the two studies with the Rasch model
mod <- equating.rasch(x=pars[,c("item","study1")], y=pars[,c("item","study2")])
## Mean.Mean Haebara Stocking.Lord
## 1 0.08828 0.08896269 0.09292838

## Not run:
### linking using the plink package
library(plink)
I <- nrow(pars)
pm <- as.poly.mod(I)
# linking parameters
plink.pars1 <- list( "study1" = data.frame( 1 , pars$study1 , 0 ) ,
                   "study2" = data.frame( 1 , pars$study2 , 0 ) )
# the parameters are arranged in the columns:
# Discrimination, Difficulty, Guessing Parameter
# common items
common.items <- cbind("study1"=1:I,"study2"=1:I)
# number of categories per item
cats.item <- list( "study1"=rep(2,I), "study2"=rep(2,I))
# convert into plink object
x <- plink::as.irt.pars( plink.pars, common.items , cat= cats.item,
                       poly.mod=list(pm,pm))
# linking using plink: first group is reference group
out <- plink::plink(x, rescale="MS", base.grp=1, D=1.7)
# summary for linking
summary(out)
## ----- group2/group1* -----
## Linking Constants
##
##           A           B
## Mean/Mean 1.000000 -0.088280
## Mean/Sigma 1.000000 -0.088280
## Haebara    1.000000 -0.088515
## Stocking-Lord 1.000000 -0.096610
# extract linked parameters
```

```
pars.out <- plink::link.pars(out)
## End(Not run)
```

---

equating.rasch.jackknife

*Jackknife Equating Error in Generalized Logistic Rasch Model*

---

## Description

This function estimates the linking error in linking based on Jackknife (Monseur & Berezner, 2007).

## Usage

```
equating.rasch.jackknife(pars.data, display = TRUE,
  se.linkerror = FALSE, alpha1 = 0, alpha2 = 0)
```

## Arguments

<code>pars.data</code>	Data frame with four columns: jackknife unit (1st column), item parameter study 1 (2nd column), item parameter study 2 (3rd column), item (4th column)
<code>display</code>	Display progress?
<code>se.linkerror</code>	Compute standard error of the linking error
<code>alpha1</code>	Fixed $\alpha_1$ parameter in the generalized item response model
<code>alpha2</code>	Fixed $\alpha_2$ parameter in the generalized item response model

## Value

A list with following entries:

<code>pars.data</code>	Used item parameters
<code>itemunits</code>	Used units for jackknife
<code>descriptives</code>	Descriptives for Jackknife. <code>linkingerror.jackknife</code> is the estimated linking error.

## Author(s)

Alexander Robitzsch

## References

Monseur, C., & Berezner, A. (2007). The computation of equating errors in international surveys in education. *Journal of Applied Measurement*, **8**, 323-335.

## See Also

For more details on linking methods see [equating.rasch](#).

## Examples

```
#####
# EXAMPLE 1: Linking errors PISA study
#####

data(data.pisaPars)
pars <- data.pisaPars

# Linking error: Jackknife unit is the testlet
res1 <- equating.rasch.jackknife(pars[ , c("testlet" , "study1" , "study2" , "item" ) ] )
res1$descriptives
##   N.items N.units   shift      SD linkerror.jackknife SE.SD.jackknife
## 1      25      8 0.09292838 0.1487387          0.04491197          0.03466309

# Linking error: Jackknife unit is the item
res2 <- equating.rasch.jackknife(pars[ , c("item" , "study1" , "study2" , "item" ) ] )
res2$descriptives
##   N.items N.units   shift      SD linkerror.jackknife SE.SD.jackknife
## 1      25      25 0.09292838 0.1487387          0.02682839          0.02533327
```

---

expl.detect

*Exploratory DETECT Analysis*


---

## Description

This function estimates the DETECT index (Stout, Habing, Douglas & Kim, 1996; Zhang & Stout, 1999a, 1999b) in an exploratory way. Conditional covariances of itempairs are transformed into a distance matrix such that items are clustered by the hierarchical Ward algorithm (Roussos, Stout & Marden, 1998).

## Usage

```
expl.detect(data, score, nclusters, N.est = NULL, seed = 897, bwscale = 1.1)
```

## Arguments

data	An $N \times I$ data frame of dichotomous responses. Missing responses are allowed.
score	An ability estimate, e.g. the WLE
nclusters	Number of clusters in the analysis
N.est	Number of students in a (possible) validation of the DETECT index. N.est students are drawn at random from data.
seed	Random seed
bwscale	Bandwidth scale factor

**Value**

A list with following entries

detect.unweighted	Unweighted DETECT statistics
detect.weighted	Weighted DETECT statistics. Weighting is done proportionally to sample sizes of item pairs.
clusterfit	Fit of the cluster method
itemcluster	Cluster allocations

**Author(s)**

Alexander Robitzsch

**References**

- Roussos, L. A., Stout, W. F., & Marden, J. I. (1998). Using new proximity measures with hierarchical cluster analysis to detect multidimensionality. *Journal of Educational Measurement*, **35**, 1-30.
- Stout, W., Habing, B., Douglas, J., & Kim, H. R. (1996). Conditional covariance-based nonparametric multidimensionality assessment. *Applied Psychological Measurement*, **20**, 331-354.
- Zhang, J., & Stout, W. (1999a). Conditional covariance structure of generalized compensatory multidimensional items, *Psychometrika*, **64**, 129-152.
- Zhang, J., & Stout, W. (1999b). The theoretical DETECT index of dimensionality and its application to approximate simple structure, *Psychometrika*, **64**, 213-249.

**See Also**

For examples see [conf.detect](#).

---

f1d.irt

*Functional Unidimensional Item Response Model*


---

**Description**

Estimates the functional unidimensional item response model for dichotomous data (Ip et al., 2013). Either the IRT model is estimated using a probit link and employing tetrachoric correlations or item discriminations and intercepts of a pre-estimated multidimensional IRT model are provided as input.

**Usage**

```
f1d.irt(dat = NULL, nnormal = 1000, nfactors = 3, A = NULL, intercept = NULL,
mu = NULL , Sigma = NULL , maxiter = 100, conv = 10^(-5), progress = TRUE)
```

**Arguments**

dat	Data frame with dichotomous item responses
nnormal	Number of $\theta_p$ grid points for approximating the normal distribution
nfactors	Number of dimensions to be estimated
A	Matrix of item discriminations (if the IRT model is already estimated)
intercept	Vector of item intercepts (if the IRT model is already estimated)
mu	Vector of estimated means. In the default it is assumed that all means are zero.
Sigma	Estimated covariance matrix. In the default it is the identity matrix.
maxiter	Maximum number of iterations
conv	Convergence criterion
progress	Display progress? The default is TRUE.

**Details**

The functional unidimensional item response model (F1D model) for dichotomous item responses is based on a multidimensional model with a link function  $g$  (probit or logit):

$$P(X_{pi} = 1 | \theta_p) = g\left(\sum_d a_{id}\theta_{pd} - d_i\right)$$

It is assumed that  $\theta_p$  is multivariate normally distribution with a zero mean vector and identity covariance matrix.

The F1D model estimates unidimensional item response functions such that

$$P(X_{pi} = 1 | \theta_p^*) \approx g(a_i^* \theta_p^* - d_i^*)$$

The optimization function  $F$  minimizes the deviations of the approximation equations

$$a_i^* \theta_p^* - d_i^* \approx \sum_d a_{id}\theta_{pd} - d_i$$

The optimization function  $F$  is defined by

$$F(\{a_i^*, d_i^*\}_i, \{\theta_p^*\}_p) = \sum_p \sum_i w_p (a_{id}\theta_{pd} - d_i - a_i^* \theta_p^* + d_i^*)^2 \rightarrow Min!$$

All items  $i$  are equally weighted whereas the ability distribution of persons  $p$  are weighted according to the multivariate normal distribution (using weights  $w_p$ ). The estimation is conducted using an alternating least squares algorithm (see Ip et al. 2013 for a different algorithm). The ability distribution  $\theta_p^*$  of the functional unidimensional model is assumed to be standardized, i.e. does have a zero mean and a standard deviation of one.

**Value**

A list with following entries:

item	Data frame with estimated item parameters: Item intercepts for the functional unidimensional $a_i^*$ (ai.ast) and the ('ordinary') unidimensional (ai0) item response model. The same holds for item intercepts $d_i^*$ (di.ast and di0 respectively).
person	Data frame with estimated $\theta_p^*$ distribution. Locations are theta.ast with corresponding probabilities in wgt.
A	Estimated or provided item discriminations
intercept	Estimated or provided intercepts
dat	Used dataset
tetra	Object generated by <code>tetrachoric2</code> if dat is specified as input. This list entry is useful for applying <code>greenyang.reliability</code> .

### Author(s)

Alexander Robitzsch

### References

Ip, E. H., Molenberghs, G., Chen, S. H., Goegebeur, Y., & De Boeck, P. (2013). Functionally unidimensional item response models for multivariate binary data. *Multivariate Behavioral Research*, **48**, 534-562.

### See Also

For estimation of bifactor models and Green-Yang reliability based on tetrachoric correlations see `greenyang.reliability`.

For estimation of bifactor models based on marginal maximum likelihood (i.e. full information maximum likelihood) see the `tam.fa` function in the **TAM** package.

### Examples

```
#####
# EXAMPLE 1: Dataset Mathematics data.math | Exploratory multidimensional model
#####
data(data.math)
dat <- ( data.math$data )[ , -c(1,2) ] # select Mathematics items

####*
# Model 1: Functional unidimensional model based on original data

##+ (1) estimate model with 3 factors
mod1 <- f1d.irt( dat =dat , nfactors=3)

##+ (2) plot results
  par(mfrow=c(1,2))
library(MASS)
# Intercepts
plot( mod1$item$di0 , mod1$item$di.ast , pch=16 , main="Item Intercepts" ,
      xlab= expression( paste( d[i] , " (Unidimensional Model)" ) ) ,
```

```

        ylab= expression( paste( d[i] , " (Functional Unidimensional Model)" )))
abline( lm(mod1$item$di.ast ~ mod1$item$di0) , col=2 , lty=2 )
abline( MASS::rlm(mod1$item$di.ast ~ mod1$item$di0) , col=3 , lty=3 )
# Discriminations
plot( mod1$item$ai0 , mod1$item$ai.ast , pch=16 , main="Item Discriminations" ,
      xlab= expression( paste( a[i] , " (Unidimensional Model)" )) ,
      ylab= expression( paste( a[i] , " (Functional Unidimensional Model)" )))
abline( lm(mod1$item$ai.ast ~ mod1$item$ai0) , col=2 , lty=2 )
abline( MASS::rlm(mod1$item$ai.ast ~ mod1$item$ai0) , col=3 , lty=3 )
      par(mfrow=c(1,1))

##+ (3) estimate bifactor model and Green-Yang reliability
gy1 <- greenyang.reliability( mod1$tetra , nfactors = 3 )

## Not run:
#####
# Model 2: Functional unidimensional model based on estimated multidimensional
#       item response model

##+ (1) estimate 2-dimensional exploratory factor analysis with 'smirt'
I <- ncol(dat)
Q <- matrix( 1, I,2 )
Q[1,2] <- 0
variance.fixed <- cbind( 1,2,0 )
mod2a <- smirt( dat , Qmatrix=Q , irtmodel="comp" , est.a="2PL" ,
              variance.fixed=variance.fixed , maxiter=50)
##+ (2) input estimated discriminations and intercepts for
#       functional unidimensional model
mod2b <- f1d.irt( A = mod2a$a , intercept = mod2a$b )

#####
# EXAMPLE 2: Dataset Mathematics data.math | Confirmatory multidimensional model
#####

data(data.math)
library(TAM)

# dataset
dat <- data.math$data
dat <- dat[ , grep("M" , colnames(dat) ) ]
# extract item informations
iteminfo <- data.math$item
I <- ncol(dat)
# define Q-matrix
Q <- matrix( 0 , nrow=I , ncol=3 )
Q[ grep( "arith" , iteminfo$domain ) , 1 ] <- 1
Q[ grep( "Meas" , iteminfo$domain ) , 2 ] <- 1
Q[ grep( "geom" , iteminfo$domain ) , 3 ] <- 1

# fit three-dimensional model in TAM
mod1 <- TAM::tam.mml.2pl( dat , Q=Q , control=list(maxiter=40 , snodes=1000) )
summary(mod1)

```



```
# specify functional unidimensional model
intercept <- mod1$xsi[ , c("xsi") ]
names(intercept) <- rownames(mod1$xsi)
fumod1 <- f1d.irt( A = mod1$B[,2,] , intercept=intercept , Sigma= mod1$variance)
fumod1$item

## End(Not run)
```

---

fit.isop

*Fitting the ISOP and ADISOP Model for Frequency Tables*


---

### Description

Fit the isotonic probabilistic model (ISOP; Scheiblechner, 1995) and the additive isotonic probabilistic model (ADISOP; Scheiblechner, 1999).

### Usage

```
fit.isop(freq.correct, wgt, conv = 1e-04, maxit = 100,
         progress = TRUE, calc.ll=TRUE)
```

```
fit.adisop(freq.correct, wgt, conv = 1e-04, maxit = 100,
           epsilon = 0.01, progress = TRUE, calc.ll=TRUE)
```

### Arguments

freq.correct	Frequency table
wgt	Weights for frequency table (number of persons in each cell)
conv	Convergence criterion
maxit	Maximum number of iterations
epsilon	Additive constant to handle cell frequencies of 0 or 1 in fit.adisop
progress	Display progress?
calc.ll	Calculate log-likelihood values? The default is TRUE.

### Details

See [isop.dich](#) for more details of the ISOP and ADISOP model.

### Value

A list with following entries

fX	Fitted frequency table
ResX	Residual frequency table
fit	Fit statistic: weighted least squares of deviations between observed and expected frequencies

item.sc	Estimated item parameters
person.sc	Estimated person parameters
ll	Log-likelihood of the model
freq.fitted	Fitted frequencies in a long data frame

**Note**

For fitting the ADISOP model it is recommended to first fit the ISOP model and then proceed with the fitted frequency table from ISOP (see Examples).

**Author(s)**

Alexander Robitzsch

**References**

Scheiblechner, H. (1995). Isotonic ordinal probabilistic models (ISOP). *Psychometrika*, **60**, 281-304.

Scheiblechner, H. (1999). Additive conjoint isotonic probabilistic models (ADISOP). *Psychometrika*, **64**, 295-316.

**See Also**

For fitting the ISOP model to dichotomous and polytomous data see [isop.dich](#).

**Examples**

```
#####
# EXAMPLE 1: Dataset Reading
#####

data(data.read)
dat <- as.matrix( data.read)
dat.resp <- 1 - is.na(dat) # response indicator matrix
I <- ncol(dat)

#***
# (1) Data preparation
#   actually only freq.correct and wgt are needed
#   but these matrices must be computed in advance.

# different scores of students
stud.p <- rowMeans( dat , na.rm=TRUE )
# different item p values
item.p <- colMeans( dat , na.rm=TRUE )
item.ps <- sort( item.p, index.return=TRUE)
dat <- dat[ , item.ps$ix ]
# define score groups students
scores <- sort( unique( stud.p ) )
SC <- length(scores)
```

```

# create table
freq.correct <- matrix( NA , SC , I )
wgt <- freq.correct
# percent correct
a1 <- aggregate( dat == 1 , list( stud.p ) , mean , na.rm=TRUE )
freq.correct <- a1[,-1]
# weights
a1 <- aggregate( dat.resp , list( stud.p ) , sum , na.rm=TRUE )
wgt <- a1[,-1]

####
# (2) Fit ISOP model
res.isop <- fit.isop( freq.correct , wgt )
# fitted frequency table
res.isop$fX

####
# (3) Fit ADISOP model
# use monotonely smoothed frequency table from ISOP model
res.adisop <- fit.adisop( freq.correct=res.isop$fX , wgt )
# fitted frequency table
res.adisop$fX

```

---

fuzcluster

*Clustering for Continuous Fuzzy Data*


---

### Description

This function performs clustering for continuous fuzzy data for which membership functions are assumed to be Gaussian (Denoeux, 2013). The mixture is also assumed to be Gaussian and (conditionally cluster membership) independent.

### Usage

```
fuzcluster(dat_m, dat_s, K = 2, nstarts = 7, seed = NULL, maxiter = 100,
           parmconv = 0.001, fac.oldxsi=0.75, progress = TRUE)
```

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

### Arguments

dat_m	Centers for individual item specific membership functions
dat_s	Standard deviations for individual item specific membership functions
K	Number of latent classes
nstarts	Number of random starts. The default is 7 random starts.
seed	Simulation seed. If one value is provided, then only one start is performed.

maxiter	Maximum number of iterations
parmconv	Maximum absolute change in parameters
fac.oldxsi	Convergence acceleration factor which should take values between 0 and 1. The default is 0.75.
progress	An optional logical indicating whether iteration progress should be displayed.
object	Object of class fuzcluster
...	Further arguments to be passed

**Value**

A list with following entries

deviance	Deviance
iter	Number of iterations
pi_est	Estimated class probabilities
mu_est	Cluster means
sd_est	Cluster standard deviations
posterior	Individual posterior distributions of cluster membership
seed	Simulation seed for cluster solution
ic	Information criteria

**Author(s)**

Alexander Robitzsch

**References**

Denooux, T. (2013). Maximum likelihood estimation from uncertain data in the belief function framework. *IEEE Transactions on Knowledge and Data Engineering*, **25**, 119-130.

**See Also**

See [fuzdiscr](#) for estimating discrete distributions for fuzzy data.

**Examples**

```
## Not run:
#####
# SIMULATED EXAMPLE 1: 2 classes and 3 items
#####

#*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
# simulate data (2 classes and 3 items)
set.seed(876)
library(mvtnorm)
Ntot <- 1000 # number of subjects
# define SDs for simulating uncertainty
```

```

sd_uncertain <- c( .2 , 1 , 2 )

dat_m <- NULL # data frame containing mean of membership function
dat_s <- NULL # data frame containing SD of membership function

# *** Class 1
pi_class <- .6
Nclass <- Ntot * pi_class
mu <- c(3,1,0)
Sigma <- diag(3)
# simulate data
dat_m1 <- mvtnorm::rmvnorm( Nclass , mean=mu , sigma = Sigma )
dat_s1 <- matrix( runif( Nclass * 3 ) , nrow=Nclass )
for ( ii in 1:3){ dat_s1[,ii] <- dat_s1[,ii] * sd_uncertain[ii] }
dat_m <- rbind( dat_m , dat_m1 )
dat_s <- rbind( dat_s , dat_s1 )

# *** Class 2
pi_class <- .4
Nclass <- Ntot * pi_class
mu <- c(0,-2,0.4)
Sigma <- diag(c(0.5 , 2 , 2 ) )
# simulate data
dat_m1 <- mvtnorm::rmvnorm( Nclass , mean=mu , sigma = Sigma )
dat_s1 <- matrix( runif( Nclass * 3 ) , nrow=Nclass )
for ( ii in 1:3){ dat_s1[,ii] <- dat_s1[,ii] * sd_uncertain[ii] }
dat_m <- rbind( dat_m , dat_m1 )
dat_s <- rbind( dat_s , dat_s1 )
colnames(dat_s) <- colnames(dat_m) <- paste0("I" , 1:3 )

#*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
# estimation

#### Model 1: Clustering with 8 random starts
res1 <- fuzcluster(K=2,dat_m , dat_s , nstarts = 8 , maxiter=25)
summary(res1)
## Number of iterations = 22 (Seed = 5090 )
## -----
## Class probabilities (2 Classes)
## [1] 0.4083 0.5917
##
## Means
##      I1      I2      I3
## [1,] 0.0595 -1.9070 0.4011
## [2,] 3.0682  1.0233 0.0359
##
## Standard deviations
##      [,1] [,2] [,3]
## [1,] 0.7238 1.3712 1.2647
## [2,] 0.9740 0.8500 0.7523

#### Model 2: Clustering with one start with seed 4550
res2 <- fuzcluster(K=2,dat_m , dat_s , nstarts = 1 , seed= 5090 )

```

```

summary(res2)

**** Model 3: Clustering for crisp data
#           (assuming no uncertainty, i.e. dat_s = 0)
res3 <- fuzcluster(K=2,dat_m , dat_s=0*dat_s , nstarts = 30 , maxiter=25)
summary(res3)
## Class probabilities (2 Classes)
## [1] 0.3645 0.6355
##
## Means
##           I1          I2          I3
## [1,] 0.0463 -1.9221  0.4481
## [2,] 3.0527  1.0241 -0.0008
##
## Standard deviations
##           [,1] [,2] [,3]
## [1,] 0.7261 1.4541 1.4586
## [2,] 0.9933 0.9592 0.9535

**** Model 4: kmeans cluster analysis
res4 <- kmeans( dat_m , centers = 2 )
## K-means clustering with 2 clusters of sizes 607, 393
## Cluster means:
##           I1          I2          I3
## 1 3.01550780  1.035848 -0.01662275
## 2 0.03448309 -2.008209  0.48295067

## End(Not run)

```

---

fuzdiscr

*Estimation of a Discrete Distribution for Fuzzy Data (Data in Belief Function Framework)*

---

## Description

This function estimates a discrete distribution for uncertain data based on the belief function framework (Denoeux, 2013; see Details).

## Usage

```
fuzdiscr(X, theta0 = NULL, maxiter = 200, conv = 1e-04)
```

## Arguments

X	Matrix with fuzzy data. Rows corresponds to subjects and columns to values of the membership function
theta0	Initial vector of parameter estimates
maxiter	Maximum number of iterations
conv	Convergence criterion

## Details

For  $n$  subjects, membership functions  $m_n(k)$  are observed which indicate the belief in data  $X_n = k$ . The membership function is interpreted as *epistemic uncertainty* (Denoeux, 2011). However, associated parameters in statistical models are crisp which means that models are formulated at the basis of precise (crisp) data if they would be observed.

In the present estimation problem of a discrete distribution, the parameters of interest are category probabilities  $\theta_k = P(X = k)$ .

The parameter estimation follows the evidential EM algorithm (Denoeux, 2013).

## Value

Vector of probabilities of the discrete distribution

## Author(s)

Alexander Robitzsch

## References

Denoeux, T. (2011). Maximum likelihood estimation from fuzzy data using the EM algorithm. *Fuzzy Sets and Systems*, **183**, 72-91.

Denoeux, T. (2013). Maximum likelihood estimation from uncertain data in the belief function framework. *IEEE Transactions on Knowledge and Data Engineering*, **25**, 119-130.

## Examples

```
#####
# EXAMPLE 1: Binomial distribution Denoeux Example 4.3 (2013)
#####

*** define uncertain data
X_alpha <- function( alpha ){
  Q <- matrix( 0 , 6 , 2 )
  Q[5:6,2] <- Q[1:3,1] <- 1
  Q[4,] <- c( alpha , 1 - alpha )
  return(Q)
}

# define data for alpha = 0.5
X <- X_alpha( alpha=.5 )
## > X
##      [,1] [,2]
## [1,]  1.0  0.0
## [2,]  1.0  0.0
## [3,]  1.0  0.0
## [4,]  0.5  0.5
## [5,]  0.0  1.0
## [6,]  0.0  1.0

## The fourth observation has equal plausibility for the first and the
```

```

## second category.

# parameter estimate uncertain data
fuzdiscr( X )
## > fuzdiscr( X )
## [1] 0.5999871 0.4000129

# parameter estimate pseudo likelihood
colMeans( X )
## > colMeans( X )
## [1] 0.5833333 0.4166667
##-> Observations are weighted according to belief function values.

#####
# plot parameter estimates as function of alpha
alpha <- seq( 0 , 1 , len=100 )
res <- sapply( alpha , FUN = function(aa){
  X <- X_alpha( alpha=aa )
  c( fuzdiscr( X )[1] , colMeans( X )[1] )
} )

# plot
plot( alpha , res[1,] , xlab = expression(alpha) , ylab=expression( theta[alpha] ) , type="l" ,
      main="Comparison Belief Function and Pseudo-Likelihood (Example 1)")
lines( alpha , res[2,] , lty=2 , col=2)
legend( 0 , .67 , c("Belief Function" , "Pseudo-Likelihood" ) , col=c(1,2) , lty=c(1,2) )

#####
# EXAMPLE 2: Binomial distribution (extends Example 1)
#####

X_alpha <- function( alpha ){
  Q <- matrix( 0 , 6 , 2 )
  Q[6,2] <- Q[1:2,1] <- 1
  Q[3:5,] <- matrix( c( alpha , 1 - alpha ) , 3 , 2 , byrow=TRUE)
  return(Q)
}

X <- X_alpha( alpha=.5 )
alpha <- seq( 0 , 1 , len=100 )
res <- sapply( alpha , FUN = function(aa){
  X <- X_alpha( alpha=aa )
  c( fuzdiscr( X )[1] , colMeans( X )[1] )
} )

# plot
plot( alpha , res[1,] , xlab = expression(alpha) , ylab=expression( theta[alpha] ) , type="l" ,
      main="Comparison Belief Function and Pseudo-Likelihood (Example 2)")
lines( alpha , res[2,] , lty=2 , col=2)
legend( 0 , .67 , c("Belief Function" , "Pseudo-Likelihood" ) , col=c(1,2) , lty=c(1,2) )

#####
# EXAMPLE 3: Multinomial distribution with three categories
#####

```



```

# define uncertain data
X <- matrix( c( 1,0,0 , 1,0,0 , 0,1,0 , 0,0,1 , .7 , .2 , .1 ,
              .4 , .6 , 0 ) , 6 , 3 , byrow=TRUE )
## > X
##      [,1] [,2] [,3]
## [1,] 1.0 0.0 0.0
## [2,] 1.0 0.0 0.0
## [3,] 0.0 1.0 0.0
## [4,] 0.0 0.0 1.0
## [5,] 0.7 0.2 0.1
## [6,] 0.4 0.6 0.0

##-> Only the first four observations are crisp.

#### estimation for uncertain data
fuzdiscr( X )
## > fuzdiscr( X )
## [1] 0.5772305 0.2499931 0.1727764

#### estimation pseudo-likelihood
colMeans(X)
## > colMeans(X)
## [1] 0.5166667 0.3000000 0.1833333

##-> Obviously, the treatment uncertainty is different in belief function
## and in pseudo-likelihood framework.

```

---

gom.em

*Discrete (Rasch) Grade of Membership Model*


---

## Description

This function estimates the grade of membership model (Erosheva, Fienberg & Joutard, 2007; also called mixed membership model) by the EM algorithm assuming a discrete membership score distribution.

## Usage

```

gom.em(dat, K=NULL, problevels=NULL, model="GOM", theta0.k=seq(-5, 5, len=15),
       xsi0.k=exp(seq(-6, 3, len=15)), max.increment=0.3, numdiff.parm=0.001,
       maxdevchange=10^(-5), globconv=0.001, maxiter=1000, msteps=4, mstepconv=0.001,
       progress=TRUE)

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

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

```

```

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

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

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

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

## S3 method for class 'gom'
IRT.modelfit(object,...)

## S3 method for class 'IRT.modelfit.gom'
summary(object,...)

```

### Arguments

dat	Data frame with dichotomous responses
K	Number of classes (only applies for model="GOM")
problevels	Vector containing probability levels for membership functions (only applies for model="GOM"). If a specific space of probability levels should be estimated, then a matrix can be supplied (see Example 1, Model 2a).
model	The type of grade of membership model. The default "GOM" is the nonparametric grade of membership model. The probabilities and membership functions specifications described in Details are called via "GOMRasch".
theta0.k	Vector of $\tilde{\theta}_k$ grid (applies only for model="GOMRasch")
xsi0.k	Vector of $\xi_p$ grid (applies only for model="GOMRasch")
max.increment	Maximum increment
numdiff.parm	Numerical differentiation parameter
maxdevchange	Convergence criterion for change in relative deviance
globconv	Global convergence criterion for parameter change
maxiter	Maximum number of iterations
msteps	Number of iterations within a M step
mstepconv	Convergence criterion within a M step
progress	Display iteration progress? Default is TRUE.
object	Object of class gom
...	Further arguments to be passed

## Details

The item response model of the grade of membership model (Erosheva, Fienberg & Junker, 2002; Erosheva, Fienberg & Joutard, 2007) with  $K$  classes for dichotomous correct responses  $X_{pi}$  of person  $p$  on item  $i$  is as follows (model="GOM")

$$P(X_{pi} = 1 | g_{p1}, \dots, g_{pK}) = \sum_k \lambda_{ik} g_{pk} \quad , \quad \sum_{k=1}^K g_{pk} = 1 \quad , \quad 0 \leq g_{pk} \leq 1$$

In most applications (e.g. Erosheva et al., 2007), the grade of membership function  $\{g_{pk}\}$  is assumed to follow a Dirichlet distribution. In our gom.em implementation the membership function is assumed to be discretely represented by a grid  $u = (u_1, \dots, u_L)$  with entries between 0 and 1 (e.g. seq(0,1,length=5) with  $L = 5$ ). The values  $g_{pk}$  of the membership function can then only take values in  $\{u_1, \dots, u_L\}$  with the restriction  $\sum_k g_{pk} \sum_l \mathbf{1}(g_{pk} = u_l) = 1$ . The grid  $u$  is specified by using the argument `problevels`.

The Rasch grade of membership model (model="GOMRasch") poses constraints on probabilities  $\lambda_{ik}$  and membership functions  $g_{pk}$ . The membership function of person  $p$  is parametrized by a location parameter  $\theta_p$  and a variability parameter  $\xi_p$ . Each class  $k$  is represented by a location parameter  $\tilde{\theta}_k$ . The membership function is defined as

$$g_{pk} \propto \exp \left[ -\frac{(\theta_p - \tilde{\theta}_k)^2}{2\xi_p^2} \right]$$

The person parameter  $\theta_p$  indicates the usual 'ability', while  $\xi_p$  describes the individual tendency to change between classes  $1, \dots, K$  and their corresponding locations  $\tilde{\theta}_1, \dots, \tilde{\theta}_K$ . The extremal class probabilities  $\lambda_{ik}$  follow the Rasch model

$$\lambda_{ik} = \text{invlogit}(\tilde{\theta}_k - b_i) = \frac{\exp(\tilde{\theta}_k - b_i)}{1 + \exp(\tilde{\theta}_k - b_i)}$$

Putting these assumptions together leads to the model equation

$$P(X_{pi} = 1 | g_{p1}, \dots, g_{pK}) = P(X_{pi} = 1 | \theta_p, \xi_p) = \sum_k \frac{\exp(\tilde{\theta}_k - b_i)}{1 + \exp(\tilde{\theta}_k - b_i)} \cdot \exp \left[ -\frac{(\theta_p - \tilde{\theta}_k)^2}{2\xi_p^2} \right]$$

In the extreme case of a very small  $\xi_p = \varepsilon > 0$  and  $\theta_p = \theta_0$ , the Rasch model is obtained

$$P(X_{pi} = 1 | \theta_p, \xi_p) = P(X_{pi} = 1 | \theta_0, \varepsilon) = \frac{\exp(\theta_0 - b_i)}{1 + \exp(\theta_0 - b_i)}$$

See Erosheva et al. (2002), Erosheva (2005, 2006) or Galyart (in press) for a comparison of grade of membership models with latent trait models and latent class models.

The grade of membership model is also published under the name Bernoulli aspect model, see Bingham, Kaban and Fortelius (2009).

**Value**

A list with following entries:

deviance	Deviance
ic	Information criteria
item	Data frame with item parameters
person	Data frame with person parameters
EAP.rel	EAP reliability (only applies for model="GOMRasch")
MAP	Maximum a posteriori estimate of the membership function
classdesc	Descriptives for class membership
lambda	Estimated response probabilities $\lambda_{ik}$
se.lambda	Standard error for stimated response probabilities $\lambda_{ik}$
mu	Mean of the distribution of $(\theta_p, \xi_p)$ (only applies for model="GOMRasch")
Sigma	Covariance matrix of $(\theta_p, \xi_p)$ (only applies for model="GOMRasch")
b	Estimated item difficulties (only applies for model="GOMRasch")
se.b	Standard error of estimated difficulties (only applies for model="GOMRasch")
f.yi.qk	Individual likelihood
f.qk.yi	Individual posterior
probs	Array with response probabilities
n.ik	Expected counts
iter	Number of iterations
I	Number of items
K	Number of classes
TP	Number of discrete integration points for $(g_{p1}, \dots, g_{pK})$
theta.k	Used grid of membership functions
...	Further values

**Author(s)**

Alexander Robitzsch

**References**

- Bingham, E., Kaban, A., & Fortelius, M. (2009). The aspect Bernoulli model: multiple causes of presences and absences. *Pattern Analysis and Applications*, **12**(1), 55-78.
- Erosheva, E. A. (2005). Comparing latent structures of the grade of membership, Rasch, and latent class models. *Psychometrika*, **70**, 619-628.
- Erosheva, E. A. (2006). *Latent class representation of the grade of membership model*. Seattle: University of Washington.
- Erosheva, E. A., Fienberg, S. E., & Junker, B. W. (2002). Alternative statistical models and representations for large sparse multi-dimensional contingency tables. *Annales-Faculte Des Sciences Toulouse Mathematiques*, **11**, 485-505.

Erosheva, E. A., Fienberg, S. E., & Joutard, C. (2007). Describing disability through individual-level mixture models for multivariate binary data. *Annals of Applied Statistics*, **1**, 502-537.

Galyardt, A. (in press). Interpreting mixed membership models: Implications of Erosheva's representation theorem. In E. M. Airoldi, D. Blei, E. A. Erosheva, & S. E. Fienberg: *Handbook of Mixed Membership Models*. Chapman & Hall.

### See Also

For joint maximum likelihood estimation of the grade of membership model see [gom.jml](#).

See also the **mixedMem** package for estimating mixed membership models by a variational EM algorithm.

The C code of Erosheva et al. (2007) can be downloaded from <http://projecteuclid.org/euclid.aos/1196438029#supplemental>.

Code from Manrique-Vallier can be downloaded from <http://pages.iu.edu/~dmanriqu/software.html>.

See [http://users.ics.aalto.fi/ella/publications/aspect\\_bernoulli.m](http://users.ics.aalto.fi/ella/publications/aspect_bernoulli.m) for a Matlab implementation of the algorithm in Bingham, Kaban and Fortelius (2009).

### Examples

```
#####
# EXAMPLE 1: PISA data mathematics
#####

data(data.pisaMath)
dat <- data.pisaMath$data
dat <- dat[ , grep("M" , colnames(dat)) ]

###
# Model 1: Discrete GOM with 3 classes and 5 probability levels
problevels <- seq( 0 , 1 , len=5 )
mod1 <- gom.em( dat , K=3 , problevels , model="GOM" )
summary(mod1)

## Not run:
###
# Model 2: Discrete GOM with 4 classes and 5 probability levels
problevels <- seq( 0 , 1 , len=5 )
mod2 <- gom.em( dat , K=4 , problevels , model="GOM" )
summary(mod2)

# model comparison
smod1 <- IRT.modelfit(mod1)
smod2 <- IRT.modelfit(mod2)
IRT.compareModels(smod1,smod2)

###
# Model 2a: Estimate discrete GOM with 4 classes and restricted space of probability levels
# the 2nd, 4th and 6th class correspond to "intermediate stages"
problevels <- scan()
```

```

1 0 0 0
.5 .5 0 0
0 1 0 0
0 .5 .5 0
0 0 1 0
0 0 .5 .5
0 0 0 1

problevels <- matrix( problevels, ncol=4 , byrow=TRUE)
mod2a <- gom.em( dat , K=4 , problevels , model="GOM" )
# probability distribution for latent classes
cbind( mod2a$theta.k , mod2a$pi.k )
##          [,1] [,2] [,3] [,4]          [,5]
## [1,]  1.0  0.0  0.0  0.0  0.17214630
## [2,]  0.5  0.5  0.0  0.0  0.04965676
## [3,]  0.0  1.0  0.0  0.0  0.09336660
## [4,]  0.0  0.5  0.5  0.0  0.06555719
## [5,]  0.0  0.0  1.0  0.0  0.27523678
## [6,]  0.0  0.0  0.5  0.5  0.08458620
## [7,]  0.0  0.0  0.0  1.0  0.25945016

## End(Not run)

###
# Model 3: Rasch GOM
mod3 <- gom.em( dat , model="GOMRasch" , maxiter=20 )
summary(mod3)

###
# Model 4: 'Ordinary' Rasch model
mod4 <- rasch.mml2( dat )
summary(mod4)

## Not run:
#####
# SIMULATED EXAMPLE 2: Grade of membership model with 2 classes
#####

***** DATASET 1 *****
# define an ordinary 2 latent class model
set.seed(8765)
I <- 10
prob.class1 <- runif( I , 0 , .35 )
prob.class2 <- runif( I , .70 , .95 )
probs <- cbind( prob.class1 , prob.class2 )

# define classes
N <- 1000
latent.class <- c( rep( 1 , 1/4*N ) , rep( 2,3/4*N ) )

# simulate item responses
dat <- matrix( NA , nrow=N , ncol=I )
for (ii in 1:I){

```

```

    dat[,ii] <- probs[ ii , latent.class ]
    dat[,ii] <- 1 * ( runif(N) < dat[,ii] )
  }
colnames(dat) <- paste0( "I" , 1:I)

# Model 1: estimate latent class model
mod1 <- gom.em(dat, K=2, problevels= c(0,1) , model="GOM" )
summary(mod1)
# Model 2: estimate GOM
mod2 <- gom.em(dat, K=2, problevels= seq(0,1,0.5) , model="GOM" )
summary(mod2)
# estimated distribution
cbind( mod2$theta.k , mod2$pi.k )
##      [,1] [,2]      [,3]
## [1,]  1.0  0.0 0.243925644
## [2,]  0.5  0.5 0.006534278
## [3,]  0.0  1.0 0.749540078

***** DATASET 2 *****
# define a 2-class model with graded membership
set.seed(8765)
I <- 10
prob.class1 <- runif( I , 0 , .35 )
prob.class2 <- runif( I , .70 , .95 )
prob.class3 <- .5*prob.class1+.5*prob.class2 # probabilities for 'fuzzy class'
probs <- cbind( prob.class1 , prob.class2 , prob.class3)
# define classes
N <- 1000
latent.class <- c( rep(1,round(1/3*N)),rep(2,round(1/2*N)),rep(3,round(1/6*N)))
# simulate item responses
dat <- matrix( NA , nrow=N , ncol=I )
for (ii in 1:I){
  dat[,ii] <- probs[ ii , latent.class ]
  dat[,ii] <- 1 * ( runif(N) < dat[,ii] )
}
colnames(dat) <- paste0( "I" , 1:I)

*** Model 1: estimate latent class model
mod1 <- gom.em(dat, K=2, problevels= c(0,1) , model="GOM" )
summary(mod1)

*** Model 2: estimate GOM
mod2 <- gom.em(dat, K=2, problevels= seq(0,1,0.5) , model="GOM" )
summary(mod2)
# inspect distribution
cbind( mod2$theta.k , mod2$pi.k )
##      [,1] [,2]      [,3]
## [1,]  1.0  0.0 0.3335666
## [2,]  0.5  0.5 0.1810114
## [3,]  0.0  1.0 0.4854220

***
# Model2m: estimate discrete GOM in mirt

```

```

# define latent classes
Theta <- scan( nlines=1)
  1 0 .5 .5  0 1
Theta <- matrix( Theta , nrow=3 , ncol=2,byrow=TRUE)
# define mirt model
I <- ncol(dat)
*** create customized item response function for mirt model
name <- 'gom'
par <- c("a1" = -1 , "a2" = 1 )
est <- c(TRUE, TRUE)
P.gom <- function(par,Theta,ncat){
  # GOM for two extremal classes
  pext1 <- plogis(par[1])
  pext2 <- plogis(par[2])
  P1 <- Theta[,1]*pext1 + Theta[,2]*pext2
  cbind(1-P1, P1)
}
# create item response function
icc_gom <- mirt::createItem(name, par=par, est=est, P=P.gom)
*** define prior for latent class analysis
lca_prior <- function(Theta,Etable){
  # number of latent Theta classes
  TP <- nrow(Theta)
  # prior in initial iteration
  if ( is.null(Etable) ){ prior <- rep( 1/TP , TP ) }
  # process Etable (this is correct for datasets without missing data)
  if ( ! is.null(Etable) ){
    # sum over correct and incorrect expected responses
    prior <- ( rowSums(Etable[ , seq(1,2*I,2)]) + rowSums(Etable[,seq(2,2*I,2)]) )/I
  }
  prior <- prior / sum(prior)
  return(prior)
}
*** estimate discrete GOM in mirt package
mod2m <- mirt::mirt(dat, 1, rep( "icc_gom",I) , customItems=list("icc_gom"=icc_gom),
  technical = list( customTheta=Theta , customPriorFun = lca_prior) )
# correct number of estimated parameters
mod2m@nest <- as.integer(sum(mod.pars$est) + nrow(Theta)-1 )
# extract log-likelihood and compute AIC and BIC
mod2m@logLik
( AIC <- -2*mod2m@logLik+2*mod2m@nest )
( BIC <- -2*mod2m@logLik+log(mod2m@Data$N)*mod2m@nest )
# extract coefficients
( cmod2m <- mirt.wrapper.coef(mod2m) )
# compare estimated distributions
round( cbind( "sirt" = list(2$pi.k , "mirt" = mod2m@Prior[[1]]) ) , 5 )
##          sirt    mirt
## [1,] 0.33357 0.33627
## [2,] 0.18101 0.17789
## [3,] 0.48542 0.48584
# compare estimated item parameters
dfr <- data.frame( "sirt" = mod2m$item[,4:5] )
dfr$mirt <- apply(cmod2m$coef[ , c("a1" , "a2") ] , 2 , plogis )

```



```

round(dfr,4)
##      sirt.lam.C11 sirt.lam.C12 mirt.a1 mirt.a2
## 1      0.1157      0.8935 0.1177 0.8934
## 2      0.0790      0.8360 0.0804 0.8360
## 3      0.0743      0.8165 0.0760 0.8164
## 4      0.0398      0.8093 0.0414 0.8094
## 5      0.1273      0.7244 0.1289 0.7243
## [...]

## End(Not run)

```

---

gom.jml

*Grade of Membership Model (Joint Maximum Likelihood Estimation)*


---

## Description

This function estimates the grade of membership model employing a joint maximum likelihood estimation method (Erosheva, 2002; p. 23ff.).

## Usage

```
gom.jml(dat, K=2, seed=NULL, globconv=0.001, maxdevchange=0.001,
        maxiter=600, min.lambda=0.001, min.g=0.001)
```

## Arguments

dat	Data frame of dichotomous item responses
K	Number of classes
seed	Seed value of random number generator. Deterministic starting values are used for the default value NULL.
globconv	Global parameter convergence criterion
maxdevchange	Maximum change in relative deviance
maxiter	Maximum number of iterations
min.lambda	Minimum $\lambda_{ik}$ parameter to be estimated
min.g	Minimum $g_{pk}$ parameter to be estimated

## Details

The item response model of the grade of membership model with  $K$  classes for dichotomous correct responses  $X_{pi}$  of person  $p$  on item  $i$  is

$$P(X_{pi} = 1 | g_{p1}, \dots, g_{pK}) = \sum_k \lambda_{ik} g_{pk} \quad , \quad \sum_k g_{pk} = 1$$

**Value**

A list with following entries:

lambda	Data frame of item parameters $\lambda_{ik}$
g	Data frame of individual membership scores $g_{pk}$
g.mean	Mean membership scores
gcut	Discretized membership scores
gcut.distr	Distribution of discretized membership scores
K	Number of classes
deviance	Deviance
ic	Information criteria
N	Number of students
score	Person score
iter	Number of iterations
datproc	List with processed data (recoded data, starting values, ...)
...	Further values

**Author(s)**

Alexander Robitzsch

**References**

Erosheva, E. A. (2002). *Grade of membership and latent structure models with application to disability survey data*. PhD thesis, Carnegie Mellon University, Department of Statistics.

**See Also**

S3 method [summary.gom](#)

**Examples**

```
#####
# EXAMPLE 1: TIMSS data
#####

data( data.timss)
dat <- data.timss$data[ , grep("M" , colnames(data.timss$data) ) ]

# 2 Classes (deterministic starting values)
m2 <- gom.jml(dat,K=2 , maxiter=10 )
summary(m2)

## Not run:
# 3 Classes with fixed seed and maximum number of iterations
m3 <- gom.jml(dat,K=3 , maxiter=50,seed=89)
summary(m3)
```

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

---

greenyang.reliability *Reliability for Dichotomous Item Response Data Using the Method of Green and Yang (2009)*

---

## Description

This function estimates the model-based reliability of dichotomous data using the Green & Yang (2009) method. The underlying factor model is  $D$ -dimensional where the dimension  $D$  is specified by the argument `nfactors`. The factor solution is subject to the application of the Schmid-Leiman transformation (see Reise, 2012; Reise, Bonifay, & Haviland, 2013; Reise, Moore, & Haviland, 2010).

## Usage

```
greenyang.reliability(object.tetra, nfactors)
```

## Arguments

<code>object.tetra</code>	Object as the output of the function <code>tetrachoric</code> , the <code>fa.parallel.poly</code> from the <b>psych</b> package or the <code>tetrachoric2</code> function (from <b>sirt</b> ). This object can also be created as a list by the user where the tetrachoric correlation must be in the list entry <code>rho</code> and the thresholds must be in the list entry <code>thresh</code> .
<code>nfactors</code>	Number of factors (dimensions)

## Value

A data frame with columns:

<code>coefficient</code>	Name of the reliability measure. <code>omega_1</code> (Omega) is the reliability estimate for the total score for dichotomous data based on a one-factor model, <code>omega_t</code> (Omega Total) is the estimate for a $D$ -dimensional model. For the nested factor model, <code>omega_h</code> (Omega Asymptotic) is the reliability of the general factor model, <code>omega_ha</code> (Omega Hierarchical Asymptotic) eliminates item-specific variance. The explained common variance (ECV) explained by the common factor is based on the $D$ -dimensional but does not take item thresholds into account. The amount of explained variance <code>ExplVar</code> is defined as the quotient of the first eigenvalue of the tetrachoric correlation matrix to the sum of all eigenvalues. The statistic <code>EigenvalRatio</code> is the ratio of the first and second eigenvalue.
<code>dimensions</code>	Number of dimensions
<code>estimate</code>	Reliability estimate

## Note

This function needs the **psych** package.

**Author(s)**

Alexander Robitzsch

**References**

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

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

Reise, S. P., Bonifay, W. E., & Haviland, M. G. (2013). Scoring and modeling psychological measures in the presence of multidimensionality. *Journal of Personality Assessment*, **95**, 129-140.

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

**See Also**

See [f1d.irt](#) for estimating the functional unidimensional item response model.

This function uses [reliability.nonlinearSEM](#).

**Examples**

```
## Not run:
#####
# EXAMPLE 1: Reliability estimation of Reading dataset data.read
#####
miceadds::library_install("psych")
set.seed(789)
data( data.read )
dat <- data.read

# calculate matrix of tetrachoric correlations
dat.tetra <- psych::tetrachoric(dat)      # using tetrachoric from psych package
dat.tetra2 <- tetrachoric2(dat)          # using tetrachoric2 from sirt package

# perform parallel factor analysis
fap <- psych::fa.parallel.poly(dat , n.iter = 1 )
## Parallel analysis suggests that the number of factors = 3
## and the number of components = 2

# parallel factor analysis based on tetrachoric correlation matrix
## (tetrachoric2)
fap2 <- psych::fa.parallel(dat.tetra2$rho , n.obs=nrow(dat) , n.iter = 1 )
## Parallel analysis suggests that the number of factors = 6
## and the number of components = 2
## Note that in this analysis, uncertainty with respect to thresholds is ignored.

# calculate reliability using a model with 4 factors
greenyang.reliability( object.tetra = dat.tetra , nfactors =4 )
## coefficient dimensions estimate
```

```

## Omega Total (1D)                omega_1      1  0.771
## Omega Total (4D)                omega_t      4  0.844
## Omega Hierarchical (4D)         omega_h      4  0.360
## Omega Hierarchical Asymptotic (4D) omega_ha     4  0.427
## Explained Common Variance (4D)   ECV          4  0.489
## Explained Variance (First Eigenvalue) ExplVar     NA 35.145
## Eigenvalue Ratio (1st to 2nd Eigenvalue) EigenvalRatio NA  2.121

# calculation of Green-Yang-Reliability based on tetrachoric correlations
# obtained by tetrachoric2
greenyang.reliability( object.tetra = dat.tetra2 , nfactors =4 )

# The same result will be obtained by using fap as the input
greenyang.reliability( object.tetra = fap , nfactors =4 )
## End(Not run)

```

---

invariance.alignment    *Alignment Procedure for Linking under Approximate Invariance*

---

## Description

This function does alignment under approximate invariance for  $G$  groups and  $I$  items (Asparouhov & Muthen, 2014; Muthen, 2014). It is assumed that item loadings and intercepts are previously estimated under the assumption of a factor with zero mean and a variance of one.

## Usage

```

invariance.alignment(lambda, nu, wgt=NULL, align.scale = c(1, 1),
  align.pow = c(1, 1), eps = .01, h = 0.001, max.increment = 0.2,
  increment.factor = c(1.001,1.02,1.04,1.08), maxiter = 300, conv = 1e-04,
  fac.oldpar= c(.01,.2,.5,.85), psi0.init = NULL, alpha0.init = NULL, progress = TRUE)

## S3 method for class 'invariance.alignment'
summary(object,...)

```

## Arguments

lambda	A $G \times I$ matrix with item loadings
nu	A $G \times I$ matrix with item intercepts
wgt	A $G \times I$ matrix for weighing groups for each item
align.scale	A vector of length two containing scale parameter $a_\lambda$ and $a_\nu$ (see Details)
align.pow	A vector of length two containing power $p_\lambda$ and $p_\nu$ (see Details)
eps	A parameter in the optimization function
h	Numerical differentiation parameter
max.increment	Maximum increment in each iteration

increment.factor	A numerical larger than one indicating the extent of the decrease of max.increment in every iteration.
maxiter	Maximum number of iterations
conv	Maximum parameter change of the optimization function
fac.oldpar	Convergence acceleration parameter between 0 and 1. This parameter defines the relative weight the previous parameter value for the calculation of the parameter update. The default is .85. But experiment with this value and study the obtained results.
psi0.init	An optional vector of initial $\psi_0$ parameters
alpha0.init	An optional vector of initial $\alpha_0$ parameters
progress	An optional logical indicating whether computational should be printed.
object	Object of class invariance.alignment
...	Further optional arguments to be passed

### Details

For  $G$  groups and  $I$  items, item loadings  $\lambda_{ig0}$  and intercepts  $\nu_{ig0}$  are available and have been estimated in a 1-dimensional factor analysis assuming a standardized factor.

The alignment procedure searches means  $\alpha_{g0}$  and standard deviations  $\psi_{g0}$  using an alignment optimization function  $F$ . This function is defined as

$$F = \sum_i \sum_{g_1 < g_2} w_{i,g_1} w_{i,g_2} f_\lambda(\lambda_{ig_1,1} - \lambda_{ig_2,1}) + \sum_i \sum_{g_1 < g_2} w_{i,g_1} w_{i,g_2} f_\nu(\nu_{ig_1,1} - \nu_{ig_2,1})$$

where the aligned item parameters  $\lambda_{ig,1}$  and  $\nu_{ig,1}$  are defined such that

$$\lambda_{ig,1} = \lambda_{ig0} / \psi_{g0} \quad \text{and} \quad \nu_{ig,1} = \nu_{ig0} - \alpha_{g0} \lambda_{ig0} / \psi_{g0}$$

and the optimization functions are defined as

$$f_\lambda(x) = [(x/a_\lambda)^2 + \varepsilon]^{p_\lambda} \quad \text{and} \quad f_\nu(x) = [(x/a_\nu)^2 + \varepsilon]^{p_\nu}$$

using a small  $\varepsilon > 0$  (e.g. .0001) to obtain a differentiable optimization function.

For identification reasons, the product  $\prod_g \psi_{g0}$  of all group standard deviations is set to one. The mean  $\alpha_{g0}$  of the first group is set to zero.

Note that the standard deviations  $\psi_g$  are estimated due to minimizing the sum of  $f_\lambda$  functions while means  $\alpha_g$  are obtained by minimizing the  $f_\nu$  part with fixed  $\psi_g$  parameters. Therefore, the original approach of Asparouhov and Muthen (2014) is split into a two-step procedure.

Note that Asparouhov and Muthen (2014) use  $a_\lambda = a_\nu = 1$  (which can be modified in align.scale) and  $p_\lambda = p_\nu = 1/4$  (which can be modified in align.pow). In case of  $p_\lambda = 1$ , the penalty is approximately  $f_\lambda(x) = x^2$ , in case of  $p_\lambda = 1/4$  it is approximately  $f_\lambda(x) = \sqrt{|x|}$ .

Effect sizes of approximate invariance based on  $R^2$  have been proposed by Asparouhov and Muthen (2014). These are calculated separately for item loading and intercepts, resulting in  $R_\lambda^2$  and  $R_\nu^2$  measures which are included in the output es.invariance. In addition, the average correlation of aligned item parameters among groups (rbar) is reported.

*Metric invariance* means that all aligned item loadings  $\lambda_{ig,1}$  are equal across groups and therefore  $R_\lambda^2 = 1$ . *Scalar invariance* means that all aligned item loadings  $\lambda_{ig,1}$  and aligned item intercepts  $\nu_{ig,1}$  are equal across groups and therefore  $R_\lambda^2 = 1$  and  $R_\nu^2 = 1$  (see Vandenberg & Lance, 2000).

**Value**

A list with following entries

pars	Aligned distribution parameters
itempars.aligned	Aligned item parameters for all groups
es.invariance	Effect sizes of approximate invariance
lambda.aligned	Aligned $\lambda_{ig,1}$ parameters
lambda.resid	Residuals of $\lambda_{ig,1}$ parameters
nu.aligned	Aligned $\nu_{ig,1}$ parameters
nu.resid	Residuals of $\nu_{ig,1}$ parameters
Niter	Number of iterations for $f_\lambda$ and $f_\nu$ optimization functions
miniter	Iteration index with minimum optimization value
fopt	Minimum optimization value
align.scale	Used alignment scale parameters
align.pow	Used alignment power parameters

**Author(s)**

Alexander Robitzsch

**References**

- Asparouhov, T., & Muthen, B. (2014). Multiple-group factor analysis alignment. *Structural Equation Modeling*, **21**, 1-14. <http://www.statmodel.com/Alignment.shtml>
- Muthen, B., & Asparouhov, T. (2014). IRT studies of many groups: The alignment method. *Frontiers in Psychology | Quantitative Psychology and Measurement*, **5:978**. doi: 10.3389/fpsyg.2014.00978, <http://journal.frontiersin.org/Journal/10.3389/fpsyg.2014.00978/abstract>.
- Vandenberg, R. J., & Lance, C. E. (2000). A review and synthesis of the measurement invariance literature: Suggestions, practices, and recommendations for organizational research. *Organizational Research Methods*, **3**, 4-70.

**See Also**

For IRT linking see also [linking.haberman](#).

For modelling random item effects for loadings and intercepts see [mcmc.2pno.ml](#).

**Examples**

```
#####
# EXAMPLE 1: Item parameters cultural activities
#####

data( data.activity.itempars )
lambda <- data.activity.itempars$lambda
```

```

nu <- data.activity.itempars$nu
Ng <- data.activity.itempars$N
wgt <- matrix( sqrt(Ng) , length(Ng) , ncol(nu) )

####
# Model 1: Alignment using a quadratic loss function
# -> use the default of align.pow=c(1,1) and align.scale=c(1,1)
mod1 <- invariance.alignment( lambda , nu , wgt )
summary(mod1)
## Effect Sizes of Approximate Invariance
## loadings intercepts
## R2 0.9944 0.9988
## sqrtU2 0.0748 0.0346
## rbar 0.9265 0.9735

####
# Model 2: Different powers for alignment
mod2 <- invariance.alignment( lambda , nu , wgt , align.pow=c(.25,1/2) ,
                             align.scale=c(.95,.95) , max.increment=.1)
summary(mod2)

# compare means from Models 1 and 2
plot( mod1$pars$alpha0 , mod2$pars$alpha0 , pch=16 ,
      xlab= "M (Model 1)" , ylab="M (Model 2)" , xlim=c(-.3,.3) , ylim=c(-.3,.3) )
lines( c(-1,1) , c(-1,1) , col="gray")
round( cbind( mod1$pars$alpha0 , mod2$pars$alpha0 ) , 3 )
round( mod1$nu.resid , 3)
round( mod2$nu.resid , 3 )

####
# Model 3: Low powers for alignment of scale and power
# Note that setting increment.factor larger than 1 seems necessary
mod3 <- invariance.alignment( lambda , nu , wgt , align.pow=c(.25,.35) ,
                             align.scale=c(.55,.55) , psi0.init=mod1$psi0 , alpha0.init = mod1$alpha0 )
summary(mod3)

# compare mean and SD estimates of Models 1 and 3
plot( mod1$pars$alpha0 , mod3$pars$alpha0 , pch=16)
plot( mod1$pars$psi0 , mod3$pars$psi0 , pch=16)

# compare residuals for Models 1 and 3
# plot lambda
plot( abs(as.vector(mod1$lambda.resid)) , abs(as.vector(mod3$lambda.resid)) ,
      pch=16 , xlab="Residuals lambda (Model 1)" ,
      ylab="Residuals lambda (Model 3)" , xlim=c(0,.1) , ylim=c(0,.1))
lines( c(-3,3),c(-3,3) , col="gray")
# plot nu
plot( abs(as.vector(mod1$nu.resid)) , abs(as.vector(mod3$nu.resid)) ,
      pch=16 , xlab="Residuals nu (Model 1)" , ylab="Residuals nu (Model 3)" ,
      xlim=c(0,.4),ylim=c(0,.4))
lines( c(-3,3),c(-3,3) , col="gray")

## Not run:

```



```
#####
# EXAMPLE 2: Comparison 4 groups | data.inv4gr
#####

data(data.inv4gr)
dat <- data.inv4gr
miceadds::library_install("semTools")

model1 <- "
  F =~ I01 + I02 + I03 + I04 + I05 + I06 + I07 + I08 + I09 + I10 + I11
  F ~~ 1*F
"

res <- semTools::measurementInvariance(model1, std.lv =TRUE , data=dat , group="group")
## Measurement invariance tests:
##
## Model 1: configural invariance:
##   chisq      df    pvalue      cfi    rmsea      bic
##   162.084   176.000    0.766    1.000    0.000 95428.025
##
## Model 2: weak invariance (equal loadings):
##   chisq      df    pvalue      cfi    rmsea      bic
##   519.598   209.000    0.000    0.973    0.039 95511.835
##
## [Model 1 versus model 2]
##   delta.chisq    delta.df delta.p.value    delta.cfi
##     357.514         33.000         0.000         0.027
##
## Model 3: strong invariance (equal loadings + intercepts):
##   chisq      df    pvalue      cfi    rmsea      bic
##   2197.260   239.000    0.000    0.828    0.091 96940.676
##
## [Model 1 versus model 3]
##   delta.chisq    delta.df delta.p.value    delta.cfi
##     2035.176         63.000         0.000         0.172
##
## [Model 2 versus model 3]
##   delta.chisq    delta.df delta.p.value    delta.cfi
##     1677.662         30.000         0.000         0.144
##

# extract item parameters separate group analyses
ipars <- lavaan::parameterEstimates(res$fit.configural)
# extract lambda's: groups are in rows, items in columns
lambda <- matrix( ipars[ ipars$op == "~" , "est" ] , nrow=4 , byrow=TRUE)
colnames(lambda) <- colnames(dat)[-1]
# extract nu's
nu <- matrix( ipars[ ipars$op == "~1" & ipars$sse != 0 , "est" ] , nrow=4 , byrow=TRUE)
colnames(nu) <- colnames(dat)[-1]

# Model 1: least squares optimization
mod1 <- invariance.alignment( lambda=lambda , nu=nu )
summary(mod1)
```

```

## Effect Sizes of Approximate Invariance
##      loadings intercepts
## R2      0.9826      0.9972
## sqrtU2  0.1319      0.0526
## rbar    0.6237      0.7821
## -----
## Group Means and Standard Deviations
##      alpha0 psi0
## 1  0.000 0.965
## 2 -0.105 1.098
## 3 -0.081 1.011
## 4  0.171 0.935

# Model 2: sparse target function
mod2 <- invariance.alignment( lambda=lambda , nu=nu , align.pow=c(1/4,1/4) )
summary(mod2)
## Effect Sizes of Approximate Invariance
##      loadings intercepts
## R2      0.9824      0.9972
## sqrtU2  0.1327      0.0529
## rbar    0.6237      0.7856
## -----
## Group Means and Standard Deviations
##      alpha0 psi0
## 1 -0.002 0.965
## 2 -0.107 1.098
## 3 -0.083 1.011
## 4  0.170 0.935

#####
# EXAMPLE 3: European Social Survey data.ess2005
#####

data(data.ess2005)
lambda <- data.ess2005$lambda
nu <- data.ess2005$nu

# Model 1: least squares optimization
mod1 <- invariance.alignment( lambda=lambda , nu=nu )
summary(mod1)

# Model 2: sparse target function and definition of scales
mod2 <- invariance.alignment( lambda=lambda , nu=nu , align.pow=c(1/4,1/4) ,
                             align.scale= c( .2 , .3) )
summary(mod2)

# compare results of Model 1 and Model 2
round( cbind( mod1$pars , mod2$pars ) , 2 )
##      alpha0 psi0 alpha0 psi0
## 1  0.06 0.87  0.05 0.91
## 2 -0.51 1.03 -0.37 0.99
## 3  0.18 0.97  0.25 1.04
## 4 -0.67 0.90 -0.53 0.90

```

```

## 5  0.09 0.98  0.10 0.99
## 6  0.23 1.03  0.28 1.00
## 7  0.27 0.97  0.14 1.10
## 8  0.18 0.90  0.07 0.89
## [...]

# look at nu residuals to explain differences in means
round( mod1$nu.resid , 2)
##      ipfrule ipmodst ipbhprp imptrad
## [1,]  0.15 -0.25 -0.01  0.01
## [2,] -0.18  0.23  0.10 -0.24
## [3,]  0.22 -0.34  0.05 -0.02
## [4,]  0.29 -0.04  0.12 -0.53
## [5,] -0.32  0.19  0.00  0.13
## [6,]  0.05 -0.21  0.05  0.04
## [7,] -0.26  0.54 -0.15 -0.02
## [8,]  0.07 -0.05 -0.10  0.12
round( mod2$nu.resid , 2)
##      ipfrule ipmodst ipbhprp imptrad
## [1,]  0.16 -0.25  0.00  0.02
## [2,] -0.27  0.14  0.00 -0.30
## [3,]  0.18 -0.37  0.00 -0.05
## [4,]  0.19 -0.13  0.00 -0.60
## [5,] -0.33  0.19 -0.01  0.12
## [6,]  0.00 -0.23  0.00  0.01
## [7,] -0.16  0.64 -0.01  0.04
## [8,]  0.15  0.02 -0.02  0.19

round( rowMeans( mod1$nu.resid )[1:8] , 2 )
## [1] -0.02 -0.02 -0.02 -0.04  0.00 -0.02  0.03  0.01
round( rowMeans( mod2$nu.resid )[1:8] , 2 )
## [1] -0.02 -0.11 -0.06 -0.14 -0.01 -0.06  0.13  0.09

#####
# SIMULATED EXAMPLE 4: Linking with item parameters containing outliers
#####

# see Help file in linking.robust

# simulate some item difficulties in the Rasch model
I <- 38
set.seed(18785)
itempars <- data.frame("item" = paste0("I",1:I) )
itempars$study1 <- rnorm( I , mean = .3 , sd =1.4 )
# simulate DIF effects plus some outliers
bdif <- rnorm(I,mean=.4,sd=.09)+(runif(I)>.9)* rep( 1*c(-1,1)+.4 , each=I/2 )
itempars$study2 <- itempars$study1 + bdif
# create input for function invariance.alignment
nu <- t( itempars[,2:3] )
colnames(nu) <- itempars$item
lambda <- 1+0*nu

# linking using least squares optimization

```

```

mod1 <- invariance.alignment( lambda=lambda , nu=nu )
summary(mod1)
##   Group Means and Standard Deviations
##           alpha0 psi0
##   study1 -0.286   1
##   study2  0.286   1

# linking using powers of .5
mod2 <- invariance.alignment( lambda=lambda , nu=nu , align.pow=c(.5,.5) )
summary(mod2)
##   Group Means and Standard Deviations
##           alpha0 psi0
##   study1 -0.213   1
##   study2  0.213   1

# linking using powers of .25
mod3 <- invariance.alignment( lambda=lambda , nu=nu , align.pow=c(.25,.25) )
summary(mod3)
##   Group Means and Standard Deviations
##           alpha0 psi0
##   study1 -0.207   1
##   study2  0.207   1

#####
# EXAMPLE 5: Linking gender groups with data.math
#####

data(data.math)
dat <- data.math$data
dat.male <- dat[ dat$female == 0 , substring( colnames(dat) ,1,1) == "M" ]
dat.female <- dat[ dat$female == 1 , substring( colnames(dat) ,1,1) == "M" ]

#*****
# Model 1: Linking using the Rasch model
mod1m <- rasch.mml2( dat.male )
mod1f <- rasch.mml2( dat.female )

# create objects for invariance.alignment
nu <- rbind( mod1m$item$thresh , mod1f$item$thresh )
colnames(nu) <- mod1m$item$item
rownames(nu) <- c("male" , "female")
lambda <- 1+0*nu

# mean of item difficulties
round( rowMeans(nu) , 3 )
##           male female
##   -0.081 -0.049

# Linking using least squares optimization
res1a <- invariance.alignment( lambda , nu , align.scale = c( .3 , .5 ) )
summary(res1a)
##   Effect Sizes of Approximate Invariance
##           loadings intercepts

```

```

## R2          1      0.9801
## sqrtU2      0      0.1412
## rbar        1      0.9626
## -----
## Group Means and Standard Deviations
##           alpha0 psi0
## male    -0.016   1
## female   0.016   1

# Linking using optimization with absolute values
res1b <- invariance.alignment( lambda , nu , align.scale = c( .3 , .5 ) ,
                             align.pow=c( .5 , .5 ) )
summary(res1b)
## Group Means and Standard Deviations
##           alpha0 psi0
## male    -0.045   1
## female   0.045   1

#-- compare results with Haberman linking
I <- ncol(dat.male)
itempartable <- data.frame( "study" = rep( c("male" , "female") , each=I ) )
itempartable$item <- c( paste0(mod1m$item$item) , paste0(mod1f$item$item) )
itempartable$a <- 1
itempartable$b <- c( mod1m$item$b , mod1f$item$b )
# estimate linking parameters
res1c <- linking.haberman( itempars= itempartable )
## Transformation parameters (Haberman linking)
##      study At      Bt
## 1 female  1  0.000
## 2  male  1 -0.032
## Linear transformation for person parameters theta
##      study A_theta B_theta
## 1 female      1  0.000
## 2  male      1  0.032
## R-Squared Measures of Invariance
##           slopes intercepts
## R2          1      0.9801
## sqrtU2      0      0.1412

#-- results of equating.rasch
x <- itempartable[ 1:I , c("item" , "b") ]
y <- itempartable[ I + 1:I , c("item" , "b") ]
res1d <- equating.rasch( x , y )
round( res1d$B.est , 3 )
##      Mean.Mean Haebara Stocking.Lord
## 1      0.032  0.032      0.029

#*****
# Model 2: Linking using the 2PL model
I <- ncol(dat.male)
mod2m <- rasch.mm12( dat.male , est.a=1:I)
mod2f <- rasch.mm12( dat.female , est.a=1:I)

```

```

# create objects for invariance.alignment
nu <- rbind( mod2m$item$thresh , mod2f$item$thresh )
colnames(nu) <- mod2m$item$item
rownames(nu) <- c("male" , "female")
lambda <- rbind( mod2m$item$a , mod2f$item$a )
colnames(lambda) <- mod2m$item$item
rownames(lambda) <- c("male" , "female")

res2a <- invariance.alignment( lambda , nu , align.scale = c( .3 , .5 ) )
summary(res2a)
## Effect Sizes of Approximate Invariance
## loadings intercepts
## R2 0.9589 0.9682
## sqrtU2 0.2027 0.1782
## rbar 0.5177 0.9394
## -----
## Group Means and Standard Deviations
## alpha0 psi0
## male -0.044 0.968
## female 0.047 1.034

res2b <- invariance.alignment( lambda , nu , align.scale = c( .3 , .5 ) ,
                              align.pow=c( .5 , .5 ) )
summary(res2b)
## Group Means and Standard Deviations
## alpha0 psi0
## male -0.046 1.053
## female 0.041 0.951

# compare results with Haberman linking
I <- ncol(dat.male)
itempartable <- data.frame( "study" = rep( c("male" , "female") , each=I ) )
itempartable$item <- c( paste0(mod2m$item$item) , paste0(mod2f$item$item) )
itempartable$a <- c( mod2m$item$a , mod2f$item$a )
itempartable$b <- c( mod2m$item$b , mod2f$item$b )
# estimate linking parameters
res2c <- linking.haberman( itempars= itempartable )
## Transformation parameters (Haberman linking)
## study At Bt
## 1 female 1.000 0.00
## 2 male 1.041 0.09
## Linear transformation for person parameters theta
## study A_theta B_theta
## 1 female 1.000 0.00
## 2 male 1.041 -0.09
## R-Squared Measures of Invariance
## slopes intercepts
## R2 0.9554 0.9484
## sqrtU2 0.2111 0.2273

## End(Not run)

```

IRT.mle

*Person Parameter Estimation***Description**

Computes the maximum likelihood estimate (MLE), weighted likelihood estimate (WLE) and maximum a posterior estimate (MAP) of ability in unidimensional item response models (Penfield & Bergeron, 2005; Warm, 1989). Item response functions can be defined by the user.

**Usage**

```
IRT.mle(data, irffct, arg.list, theta=rep(0,nrow(data)), type = "MLE",
        mu=0, sigma=1, maxiter = 20, maxincr = 3, h = 0.001, convP = 1e-04,
        maxval = 9, progress = TRUE)
```

**Arguments**

data	Data frame with item responses
irffct	User defined item response (see Examples). Arguments must be specified in arg.list. The function must contain theta and ii (item index) as arguments.
theta	Initial ability estimate
arg.list	List of arguments for irffct.
type	Type of ability estimate. It can be "MLE" (the default), "WLE" or "MAP".
mu	Mean of normal prior distribution (for type="MAP")
sigma	Standard deviation of normal prior distribution (for type="MAP")
maxiter	Maximum number of iterations
maxincr	Maximum increment
h	Numerical differentiation parameter
convP	Convergence criterion
maxval	Maximum ability value to be estimated
progress	Logical indicating whether iteration progress should be displayed

**Value**

Data frame with estimated abilities (est) and its standard error (se).

**Author(s)**

Alexander Robitzsch

**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.

**See Also**

See also the **PP** package for further person parameter estimation methods.

**Examples**

```
## Not run:
#####
# EXAMPLE 1: Generalized partial credit model
#####

data(data.ratings1)
dat <- data.ratings1

# estimate model
mod1 <- rm.facets( dat[ , paste0( "k",1:5) ], rater=dat$rater,
                  pid=dat$idstud , maxiter=15)
# extract dataset and item parameters
data <- mod1$procdata$dat2.NA
a <- mod1$ipars.dat2$a
b <- mod1$ipars.dat2$b
theta0 <- mod1$person$EAP
# define item response function for item ii
calc.pcm <- function( theta , a , b , ii ){
  K <- ncol(b)
  N <- length(theta)
  matrK <- matrix( 0:K , nrow=N , ncol=K+1 , byrow=TRUE)
  eta <- a[ii] * theta * matrK - matrix( c(0,b[ii,]), nrow=N, ncol=K+1, byrow=TRUE)
  eta <- exp(eta)
  probs <- eta / rowSums(eta, na.rm=TRUE)
  return(probs)
}
arg.list <- list("a"=a , "b"=b )

# MLE
abil1 <- IRT.mle( data, irffct=calc.pcm, theta=theta0, arg.list=arg.list )
str(abil1)
# WLE
abil2 <- IRT.mle( data, irffct=calc.pcm, theta=theta0, arg.list=arg.list, type="WLE")
str(abil2)
# MAP with prior distribution N(.2, 1.3)
abil3 <- IRT.mle( data, irffct=calc.pcm, theta=theta0, arg.list=arg.list,
                 type="MAP", mu=.2, sigma=1.3 )
str(abil3)

#####
# EXAMPLE 2: Rasch model
#####

data(data.read)
dat <- data.read
I <- ncol(dat)
```



```

# estimate Rasch model
mod1 <- rasch.mml2( dat )
summary(mod1)

# define item response function
irffct <- function( theta, b , ii){
  eta <- exp( theta - b[ii] )
  probs <- eta / ( 1 + eta )
  probs <- cbind( 1 - probs , probs )
  return(probs)
}

# initial person parameters and item parameters
theta0 <- mod1$person$EAP
arg.list <- list( "b" = mod1$item$b )

# estimate WLE
source.all(pfsirt)
abil <- IRT.mle( data = dat , irffct=irffct , arg.list=arg.list ,
               theta=theta0, type="WLE")
# compare with wle.rasch function
theta <- wle.rasch( dat , b= mod1$item$b )
cbind( abil[,1] , theta$theta , abil[,2] , theta$se.theta )

#####
# EXAMPLE 3: Ramsay quotient model
#####

data(data.read)
dat <- data.read
I <- ncol(dat)

# estimate Ramsay model
mod1 <- rasch.mml2( dat , irtmodel ="ramsay.qm" )
summary(mod1)
# define item response function
irffct <- function( theta, b , K , ii){
  eta <- exp( theta / b[ii] )
  probs <- eta / ( K[ii] + eta )
  probs <- cbind( 1 - probs , probs )
  return(probs)
}

# initial person parameters and item parameters
theta0 <- exp( mod1$person$EAP )
arg.list <- list( "b" = mod1$item2$b , "K"=mod1$item2$K )
# estimate MLE
res <- IRT.mle( data = dat , irffct=irffct , arg.list=arg.list , theta=theta0 ,
               maxval=20 , maxiter=50)

## End(Not run)

```

---

isop

*Fit Unidimensional ISOP and ADISOP Model to Dichotomous and Polytomous Item Responses*

---

## Description

Fit the unidimensional isotonic probabilistic model (ISOP; Scheiblechner, 1995, 2007) and the additive isotonic probabilistic model (ADISOP; Scheiblechner, 1999). The `isop.dich` function can be used for dichotomous data while the `isop.poly` function can be applied to polytomous data. Note that for applying the ISOP model for polytomous data it is necessary that all items do have the same number of categories.

## Usage

```
isop.dich(dat, score.breaks = NULL, merge.extreme = TRUE,
          conv = .0001, maxit = 1000, epsilon = .025, progress = TRUE)

isop.poly( dat , score.breaks=seq(0,1,len=10 ) ,
          conv = .0001, maxit = 1000 , epsilon = .025 , progress=TRUE )

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

## S3 method for class 'isop'
plot(x,ask=TRUE,...)
```

## Arguments

<code>dat</code>	Data frame with dichotomous or polytomous item responses
<code>score.breaks</code>	Vector with breaks to define score groups. For dichotomous data, the person score grouping is applied for the mean person score, for polytomous data it is applied to the modified percentile score.
<code>merge.extreme</code>	Merge extreme groups with zero and maximum score with succeeding score categories? The default is TRUE.
<code>conv</code>	Convergence criterion
<code>maxit</code>	Maximum number of iterations
<code>epsilon</code>	Additive constant to handle cell frequencies of 0 or 1 in <a href="#">fit.adisop</a>
<code>progress</code>	Display progress?
<code>object</code>	Object of class <code>isop</code> (generated by <code>isop.dich</code> or <code>isop.poly</code> )
<code>x</code>	Object of class <code>isop</code> (generated by <code>isop.dich</code> or <code>isop.poly</code> )
<code>ask</code>	Ask for a new plot?
<code>...</code>	Further arguments to be passed

## Details

The ISOP model for dichotomous data was firstly proposed by Irtel and Schmalhofer (1982). Consider person groups  $p$  (ordered from low to high scores) and items  $i$  (ordered from difficult to easy items). Here,  $F(p, i)$  denotes the proportion correct for item  $i$  in score group  $p$ , while  $n_{pi}$  denotes the number of persons in group  $p$  and on item  $i$ . The isotonic probabilistic model (Scheiblechner, 1995) monotonely smoothes this distribution function  $F$  such that

$$P(X_{pi} = 1|p, i) = F^*(p, i)$$

where the two-dimensional distribution function  $F^*$  is isotonic in  $p$  and  $i$ . Model fit is assessed by the square root of weighted squares of deviations

$$Fit = \sqrt{\frac{1}{I} \sum_{p,i} w_{pi} (F(p, i) - F^*(p, i))^2}$$

with frequency weights  $w_{pi}$  and  $\sum_p w_{pi} = 1$  for every item  $i$ . The additive isotonic model (ADISOP; Scheiblechner, 1999) assumes the existence of person parameters  $\theta_p$  and item parameters  $\delta_i$  such that

$$P(X_{pi} = 1|p) = g(\theta_p + \delta_i)$$

and  $g$  is a nonparametrically estimated isotonic function. The functions `isop.dich` and `isop.poly` uses  $F^*$  from the ISOP models and estimates person and item parameters of the ADISOP model. For comparison, `isop.dich` also fits a model with the logistic function  $g$  which results in the Rasch model.

For polytomous data, the starting point is the empirical distribution function

$$P(X_i \leq k|p) = F(k; p, i)$$

which is increasing in the argument  $k$  (the item categories). The ISOP model is defined to be antitonic in  $p$  and  $i$  while items are ordered with respect to item P-scores and persons are ordered according to modified percentile scores (Scheiblechner, 2007). The estimated ISOP model results in a distribution function  $F^*$ . Using this function, the additive isotonic probabilistic model (ADISOP) aims at estimating a distribution function

$$P(X_i \leq k; p) = F^{**}(k; p, i) = F^{**}(k, \theta_p + \delta_i)$$

which is antitonic in  $k$  and in  $\theta_p + \delta_i$ . Due to this additive relation, the ADISOP scale values are claimed to be measured at interval scale level (Scheiblechner, 1999).

The ADISOP model is compared to the graded response model which is defined by the response equation

$$P(X_i \leq k; p) = g(\theta_p + \delta_i + \gamma_k)$$

where  $g$  denotes the logistic function. Estimated parameters are in the value `fit.grm`: person parameters  $\theta_p$  (`person.sc`), item parameters  $\delta_i$  (`item.sc`) and category parameters  $\gamma_k$  (`cat.sc`).

The calculation of person and item scores is explained in [isop.scoring](#).

For an application of the ISOP and ADISOP model see Scheiblechner and Lutz (2009).

**Value**

A list with following entries:

freq.correct	Used frequency table (distribution function) for dichotomous and polytomous data
wgt	Used weights (frequencies)
prob.saturated	Frequencies of the saturated model
prob.isop	Fitted frequencies of the ISOP model
prob.adisop	Fitted frequencies of the ADISOP model
prob.logistic	Fitted frequencies of the logistic model (only for isop.dich)
prob.grm	Fitted frequencies of the graded response model (only for isop.poly)
ll	List with log-likelihood values
fit	Vector of fit statistics
person	Data frame of person parameters
item	Data frame of item parameters
p.itemcat	Frequencies for every item category
score.itemcat	Scoring points for every item category
fit.isop	Values of fitting the ISOP model (see <a href="#">fit.isop</a> )
fit.adisop	Values of fitting the ADISOP model (see <a href="#">fit.adisop</a> )
fit.logistic	Values of fitting the logistic model (only for isop.dich)
fit.grm	Values of fitting the graded response model (only for isop.poly)
...	Further values

**Author(s)**

Alexander Robitzsch

**References**

- Irtel, H., & Schmalhofer, F. (1982). Psychodiagnostik auf Ordinalskalenniveau: Messtheoretische Grundlagen, Modelltest und Parameterschaetzung. *Archiv fuer Psychologie*, **134**, 197-218.
- Scheiblechner, H. (1995). Isotonic ordinal probabilistic models (ISOP). *Psychometrika*, **60**, 281-304.
- Scheiblechner, H. (1999). Additive conjoint isotonic probabilistic models (ADISOP). *Psychometrika*, **64**, 295-316.
- Scheiblechner, H. (2007). A unified nonparametric IRT model for d-dimensional psychological test data (d-ISOP). *Psychometrika*, **72**, 43-67.
- Scheiblechner, H., & Lutz, R. (2009). Die Konstruktion eines optimalen eindimensionalen Tests mittels nichtparametrischer Testtheorie (NIRT) am Beispiel des MR SOC. *Diagnostica*, **55**, 41-54.

**See Also**

This function uses `isop.scoring`, `fit.isop` and `fit.adisop`.

Tests of the W1 axiom of the ISOP model (Scheiblechner, 1995) can be performed with `isop.test`.

See also the **ISOP** package at *Rforge*: <http://www.rforge.net/ISOP/>.

Install this package using

```
install.packages("ISOP", repos="http://www.rforge.net/")
```

**Examples**

```
#####
# EXAMPLE 1: Dataset Reading (dichotomous items)
#####

data(data.read)
dat <- as.matrix( data.read)
I <- ncol(dat)

# Model 1: ISOP Model (11 score groups)
mod1 <- isop.dich( dat )
summary(mod1)
plot(mod1)

## Not run:
# Model 2: ISOP Model (5 score groups)
score.breaks <- seq( -.005 , 1.005 , len=5+1 )
mod2 <- isop.dich( dat , score.breaks=score.breaks)
summary(mod2)

#####
# EXAMPLE 2: Dataset PISA mathematics (dichotomous items)
#####

data(data.pisaMath)
dat <- data.pisaMath$data
dat <- dat[ , grep("M" , colnames(dat) ) ]

# fit ISOP model
# Note that for this model many iterations are needed
# to reach convergence for ADISOP
mod1 <- isop.dich( dat , maxit=4000)
summary(mod1)

## End(Not run)

#####
# EXAMPLE 3: Dataset Students (polytomous items)
#####

# Dataset students: scale cultural activities
library(CDM)
```

```

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

# fit models
mod1 <- isop.poly( dat )
summary(mod1)
plot(mod1)

```

---

isop.scoring

*Scoring Persons and Items in the ISOP Model*


---

### Description

This function does the scoring in the isotonic probabilistic model (Scheiblechner, 1995, 2003, 2007). Person parameters are ordinally scaled but the ISOP model also allows *specific objective* (ordinal) comparisons for persons (Scheiblechner, 1995).

### Usage

```
isop.scoring(dat, score.itemcat=NULL)
```

### Arguments

`dat` Data frame with dichotomous or polytomous item responses  
`score.itemcat` Optional data frame with scoring points for every item and every category (see Example 2).

### Details

This function extracts the scoring rule of the ISOP model (if `score.itemcat != NULL`) and calculates the modified percentile score for every person. The score  $s_{ik}$  for item  $i$  and category  $k$  is calculated as

$$s_{ik} = \sum_{j=0}^{k-1} f_{ij} - \sum_{j=k+1}^K f_{ij} = P(X_i < k) - P(X_i > k)$$

where  $f_{ik}$  is the relative frequency of item  $i$  in category  $k$  and  $K$  is the maximum category. The modified percentile score  $\rho_p$  for subject  $p$  (mpsc in person) is defined by

$$\rho_p = \frac{1}{I} \sum_{i=1}^I \sum_{j=0}^K s_{ij} \mathbf{1}(X_{pi} = k)$$

Note that for dichotomous items, the sum score is a sufficient statistic for  $\rho_p$  but this is not the case for polytomous items. The modified percentile score  $\rho_p$  ranges between -1 and 1.

The modified item P-score  $\rho_i$  (Scheiblechner, 2007, p. 52) is defined by

$$\rho_i = \frac{1}{I-1} \cdot \sum_j [P(X_j < X_i) - P(X_j > X_i)]$$

**Value**

A list with following entries:

person	A data frame with person parameters. The modified percentile score $\rho_p$ is denoted by <code>mpsc</code> .
item	Item statistics and scoring parameters. The item P-scores $\rho_i$ are labeled as <code>pscore</code> .
<code>p.itemcat</code>	Frequencies for every item category
<code>score.itemcat</code>	Scoring points for every item category
<code>distr.fct</code>	Empirical distribution function

**Author(s)**

Alexander Robitzsch

**References**

Scheiblechner, H. (1995). Isotonic ordinal probabilistic models (ISOP). *Psychometrika*, **60**, 281-304.

Scheiblechner, H. (2003). *Nonparametric IRT: Scoring functions and ordinal parameter estimation of isotonic probabilistic models (ISOP)*. Technical Report, Philipps-Universitaet Marburg.

Scheiblechner, H. (2007). A unified nonparametric IRT model for d-dimensional psychological test data (d-ISOP). *Psychometrika*, **72**, 43-67.

**See Also**

For fitting the ISOP and ADISOP model see [isop.dich](#) or [fit.isop](#).

**Examples**

```
#####
# EXAMPLE 1: Dataset Reading
#####

data( data.read )
dat <- data.read

# Scoring according to the ISOP model
msc <- isop.scoring( dat )
# plot student scores
boxplot( msc$person$mpsc ~ msc$person$score )

#####
# EXAMPLE 2: Dataset students from CDM package | polytomous items
#####

library("CDM")
data( data.Students , package="CDM")
dat <- na.omit(data.Students[ , -c(1:2) ])
```

```

# Scoring according to the ISOP model
msc <- isop.scoring( dat )
# plot student scores
boxplot( msc$person$mpsc ~ msc$person$score )

# scoring with known scoring rule for activity items
items <- paste0( "act" , 1:5 )
score.itemcat <- msc$score.itemcat
score.itemcat <- score.itemcat[ items , ]
msc2 <- isop.scoring( dat[,items] , score.itemcat=score.itemcat )

```

---

isop.test

*Testing the ISOP Model*


---

### Description

This function performs tests of the  $W1$  axiom of the ISOP model (Scheiblechner, 2003). Standard errors of the corresponding  $W1_i$  statistics are obtained by Jackknife.

### Usage

```
isop.test(data, jackunits = 20, weights = rep(1, nrow(data)))
```

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

### Arguments

data	Data frame with item responses
jackunits	A number of Jackknife units (if an integer is provided as the argument value) or a vector in the Jackknife units are already defined.
weights	Optional vector of sampling weights
object	Object of class <code>isop.test</code>
...	Further arguments to be passed

### Value

A list with following entries

itemstat	Data frame with test and item statistics for the $W1$ axiom. The $W1_i$ statistic is denoted as <code>est</code> while <code>se</code> is the corresponding standard error of the statistic. The sample size per item is <code>N</code> and <code>M</code> denotes the item mean.
Es	Number of concordancies per item
Ed	Number of discordancies per item

The  $W1_i$  statistics are printed by the `summary` method.



**Author(s)**

Alexander Robitzsch

**References**

Scheiblechner, H. (2003). Nonparametric IRT: Testing the bi-isotonicity of isotonic probabilistic models (ISOP). *Psychometrika*, **68**, 79-96.

**See Also**

Fit the ISOP model with [isop.dich](#) or [isop.poly](#).

See also the **ISOP** package at *R*forge: <http://www.rforge.net/ISOP/>.

**Examples**

```
#####
# EXAMPLE 1: ISOP model data.Students
#####

data(data.Students, package="CDM")
dat <- data.Students[ , paste0("act",1:5) ]
dat <- dat[1:300 , ] # select first 300 students

# perform the ISOP test
mod <- isop.test(dat)
summary(mod)
## -> Wli statistics
##   parm  N    M  est  se    t
## 1 test 300   NA 0.430 0.036 11.869
## 2 act1 278 0.601 0.451 0.048  9.384
## 3 act2 275 0.473 0.473 0.035 13.571
## 4 act3 274 0.277 0.352 0.098  3.596
## 5 act4 291 1.320 0.381 0.054  7.103
## 6 act5 276 0.460 0.475 0.042 11.184
```

---

latent.regression.em.raschtype

*Latent Regression Model for the Generalized Logistic Item Response Model and the Linear Model for Normal Responses*

---

**Description**

This function estimates a unidimensional latent regression model if a likelihood is specified, parameters from the generalized item response model (Stukel, 1988) or a mean and a standard error estimate for individual scores is provided as input. Item parameters are treated as fixed in the estimation.

**Usage**

```
latent.regression.em.raschtype(data=NULL, f.yi.qk=NULL , X ,
  weights=rep(1, nrow(X)), beta.init=rep(0,ncol(X)),
  sigma.init=1, b=rep(0,ncol(X)), a=rep(1,length(b)),
  c=rep(0, length(b)), d=rep(1, length(b)), alpha1=0, alpha2=0,
  max.parchange=1e-04, theta.list=seq(-5, 5, len=20),
  maxiter=300 , progress=TRUE )

latent.regression.em.normal(y, X, sig.e, weights = rep(1, nrow(X)),
  beta.init = rep(0, ncol(X)), sigma.init = 1, max.parchange = 1e-04,
  maxiter = 300, progress = TRUE)

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

**Arguments**

data	An $N \times I$ data frame of dichotomous item responses. If no data frame is supplied, then a user can input the individual likelihood <code>f.yi.qk</code> .
f.yi.qk	An optional matrix which contains the individual likelihood. This matrix is produced by <code>rasch.mml2</code> or <code>rasch.copula2</code> . The use of this argument allows the estimation of the latent regression model independent of the parameters of the used item response model.
X	An $N \times K$ matrix of $K$ covariates in the latent regression model. Note that the intercept (i.e. a vector of ones) must be included in $X$ .
weights	Student weights (optional).
beta.init	Initial regression coefficients (optional).
sigma.init	Initial residual standard deviation (optional).
b	Item difficulties (optional). They must only be provided if the likelihood <code>f.yi.qk</code> is not given as an input.
a	Item discriminations (optional).
c	Guessing parameter (lower asymptotes) (optional).
d	One minus slipping parameter (upper asymptotes) (optional).
alpha1	Upper tail parameter $\alpha_1$ in the generalized logistic item response model. Default is 0.
alpha2	Lower tail parameter $\alpha_2$ parameter in the generalized logistic item response model. Default is 0.
max.parchange	Maximum change in regression parameters
theta.list	Grid of person ability where theta is evaluated
maxiter	Maximum number of iterations
progress	An optional logical indicating whether computation progress should be displayed.
y	Individual scores

sig.e	Standard errors for individual scores
object	Object of class latent.regression
...	Further arguments to be passed

### Details

In the output *Regression Parameters* the fraction of missing information (fmi) is reported which is the increase of variance in regression parameter estimates because ability is defined as a latent variable. The effective sample size pseudoN.latent corresponds to a sample size when the ability would be available with a reliability of one.

### Value

A list with following entries

iterations	Number of iterations needed
maxiter	Maximal number of iterations
max.pchange	Maximum change in parameter estimates
coef	Coefficients
summary.coef	Summary of regression coefficients
sigma	Estimate of residual standard deviation
vcov.simple	Covariance parameters of estimated parameters (simplified version)
vcov.latent	Covariance parameters of estimated parameters which accounts for latent ability
post	Individual posterior distribution
EAP	Individual EAP estimates
SE.EAP	Standard error estimates of EAP
explvar	Explained variance in latent regression
totalvar	Total variance in latent regression
rsquared	Explained variance $R^2$ in latent regression

### Note

Using the defaults in a, c, d, alpha1 and alpha2 corresponds to the Rasch model.

### Author(s)

Alexander Robitzsch

### References

- Adams, R., & Wu. M. (2007). The mixed-coefficients multinomial logit model: A generalized form of the Rasch model. In M. von Davier & C. H. Carstensen: *Multivariate and Mixture Distribution Rasch Models: Extensions and Applications* (pp. 57-76). New York: Springer.
- Mislevy, R. J. (1991). Randomization-based inference about latent variables from complex samples. *Psychometrika*, **56**, 177-196.
- Stukel, T. A. (1988). Generalized logistic models. *Journal of the American Statistical Association*, **83**, 426-431.

**See Also**

See also [plausible.value.imputation.raschtype](#) for plausible value imputation of generalized logistic item type models.

**Examples**

```
#####
# EXAMPLE 1: PISA Reading | Rasch model for dichotomous data
#####

data( data.pisaRead)
dat <- data.pisaRead$data
items <- grep("R" , colnames(dat))
# define matrix of covariates
X <- cbind( 1 , dat[ , c("female","hisei","migra" ) ] )

###
# Model 1: Latent regression model in the Rasch model
# estimate Rasch model
mod1 <- rasch.mml2( dat[,items] )
# latent regression model
lm1 <- latent.regression.em.raschtype( data=dat[,items ], X = X , b = mod1$item$b )

## Not run:
###
# Model 2: Latent regression with generalized link function
# estimate alpha parameters for link function
mod2 <- rasch.mml2( dat[,items] , est.alpha=TRUE)
# use model estimated likelihood for latent regression model
lm2 <- latent.regression.em.raschtype( f.yi.qk=mod2$f.yi.qk ,
      X = X , theta.list=mod2$theta.k)

###
# Model 3: Latent regression model based on Rasch copula model
testlets <- paste( data.pisaRead$item$testlet)
itemclusters <- match( testlets , unique(testlets) )
# estimate Rasch copula model
mod3 <- rasch.copula2( dat[,items] , itemcluster=itemclusters )
# use model estimated likelihood for latent regression model
lm3 <- latent.regression.em.raschtype( f.yi.qk=mod3$f.yi.qk ,
      X = X , theta.list=mod3$theta.k)

#####
# SIMULATED EXAMPLE 2: Simulated data according to the Rasch model
#####

set.seed(899)
I <- 21      # number of items
b <- seq(-2,2, len=I)  # item difficulties
n <- 2000    # number of students

# simulate theta and covariates
```

```

theta <- rnorm( n )
x <- .7 * theta + rnorm( n , .5 )
y <- .2 * x + .3*theta + rnorm( n , .4 )
dfr <- data.frame( theta , 1 , x , y )

# simulate Rasch model
dat1 <- sim.raschtype( theta = theta , b = b )

# estimate latent regression
mod <- latent.regression.em.raschtype( data = dat1 , X = dfr[,-1] , b = b )
## Regression Parameters
##
##          est se.simple      se          t p  beta    fmi N.simple pseudoN.latent
## X1 -0.2554   0.0208 0.0248 -10.2853 0 0.0000 0.2972    2000    1411.322
## x   0.4113   0.0161 0.0193  21.3037 0 0.4956 0.3052    2000    1411.322
## y   0.1715   0.0179 0.0213   8.0438 0 0.1860 0.2972    2000    1411.322
##
## Residual Variance = 0.685
## Explained Variance = 0.3639
## Total Variance    = 1.049
##
##                      R2 = 0.3469

# compare with linear model (based on true scores)
summary( lm( theta ~ x + y , data = dfr ) )
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.27821    0.01984  -14.02  <2e-16 ***
## x            0.40747    0.01534   26.56  <2e-16 ***
## y            0.18189    0.01704   10.67  <2e-16 ***
## ---
##
## Residual standard error: 0.789 on 1997 degrees of freedom
## Multiple R-squared: 0.3713, Adjusted R-squared: 0.3707

#*****
# define guessing parameters (lower asymptotes) and
# upper asymptotes ( 1 minus slipping parameters)
cI <- rep(.2, I) # all items get a guessing parameter of .2
cI[ c(7,9) ] <- .25 # 7th and 9th get a guessing parameter of .25
dI <- rep( .95 , I ) # upper asymptote of .95
dI[ c(7,11) ] <- 1 # 7th and 9th item have an asymptote of 1

# latent regression model
mod1 <- latent.regression.em.raschtype( data = dat1 , X = dfr[,-1] ,
                                       b = b , c = cI , d = dI )
## Regression Parameters
##
##          est se.simple      se          t p  beta    fmi N.simple pseudoN.latent
## X1 -0.7929   0.0243 0.0315 -25.1818 0 0.0000 0.4044    2000    1247.306
## x   0.5025   0.0188 0.0241  20.8273 0 0.5093 0.3936    2000    1247.306
## y   0.2149   0.0209 0.0266   8.0850 0 0.1960 0.3831    2000    1247.306
##
## Residual Variance = 0.9338

```

```

## Explained Variance = 0.5487
## Total Variance      = 1.4825
##                    R2 = 0.3701

#####
# SIMULATED EXAMPLE 3: Measurement error in dependent variable
#####

set.seed(8766)
N <- 4000      # number of persons
X <- rnorm(N)  # independent variable
Z <- rnorm(N)  # independent variable
y <- .45 * X + .25 * Z + rnorm(N) # dependent variable true score
sig.e <- runif( N , .5 , .6 )     # measurement error standard deviation
yast <- y + rnorm( N , sd = sig.e ) # dependent variable measured with error

#####
#****
# Model 1: Estimation with latent.regression.em.raschtype using
#           individual likelihood
# define theta grid for evaluation of density
theta.list <- mean(yast) + sd(yast) * seq( - 5 , 5 , length=21)
# compute individual likelihood
f.yi.qk <- dnorm( outer( yast , theta.list , "-" ) / sig.e )
f.yi.qk <- f.yi.qk / rowSums(f.yi.qk)
# define predictor matrix
X1 <- as.matrix(data.frame( "intercept"=1 , "X"=X , "Z"=Z ))

# latent regression model
res <- latent.regression.em.raschtype( f.yi.qk=f.yi.qk ,
                                       X= X1 , theta.list=theta.list)

## Regression Parameters
##
##           est se.simple  se      t      p  beta  fmi N.simple pseudoN.latent
## intercept 0.0112   0.0157 0.0180  0.6225 0.5336 0.0000 0.2345    4000    3061.998
## X         0.4275   0.0157 0.0180 23.7926 0.0000 0.3868 0.2350    4000    3061.998
## Z         0.2314   0.0156 0.0178 12.9868 0.0000 0.2111 0.2349    4000    3061.998
##
## Residual Variance = 0.9877
## Explained Variance = 0.2343
## Total Variance    = 1.222
##                    R2 = 0.1917

#####
#****
# Model 2: Estimation with latent.regression.em.normal
res2 <- latent.regression.em.normal( y = yast , sig.e = sig.e , X = X1)
## Regression Parameters
##
##           est se.simple  se      t      p  beta  fmi N.simple pseudoN.latent
## intercept 0.0112   0.0157 0.0180  0.6225 0.5336 0.0000 0.2345    4000    3062.041
## X         0.4275   0.0157 0.0180 23.7927 0.0000 0.3868 0.2350    4000    3062.041
## Z         0.2314   0.0156 0.0178 12.9870 0.0000 0.2111 0.2349    4000    3062.041
##
## Residual Variance = 0.9877

```

```

## Explained Variance = 0.2343
## Total Variance      = 1.222
##                    R2 = 0.1917

## -> Results between Model 1 and Model 2 are identical because they use
## the same input.

###
# Model 3: Regression model based on true scores y
mod3 <- lm( y ~ X + Z )
summary(mod3)
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.02364    0.01569   1.506   0.132
## X            0.42401    0.01570  27.016 <2e-16 ***
## Z            0.23804    0.01556  15.294 <2e-16 ***
## Residual standard error: 0.9925 on 3997 degrees of freedom
## Multiple R-squared:  0.1923, Adjusted R-squared:  0.1919
## F-statistic: 475.9 on 2 and 3997 DF, p-value: < 2.2e-16

###
# Model 4: Regression model based on observed scores yast
mod4 <- lm( yast ~ X + Z )
summary(mod4)
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.01101    0.01797   0.613   0.54
## X            0.42716    0.01797  23.764 <2e-16 ***
## Z            0.23174    0.01783  13.001 <2e-16 ***
## Residual standard error: 1.137 on 3997 degrees of freedom
## Multiple R-squared:  0.1535, Adjusted R-squared:  0.1531
## F-statistic: 362.4 on 2 and 3997 DF, p-value: < 2.2e-16

## End(Not run)

```

---

lavaan2mirt

---

*Converting a lavaan Model into a mirt Model*


---

## Description

Converts a lavaan model into a mirt model. Optionally, the model can be estimated with the `mirt::mirt` function (`est.mirt=TRUE`) or just mirt syntax is generated (`est.mirt=FALSE`).

Extensions of the lavaan syntax include guessing and slipping parameters (operators `?=g1` and `?=s1`) and a shortage operator for item groups (see `__`). See [TAM::lavaanify.IRT](#) for more details.

## Usage

```
lavaan2mirt(dat, lavmodel, est.mirt = TRUE, poly.itemtype="gpcm" , ...)
```

**Arguments**

<code>dat</code>	Dataset with item responses
<code>lavmodel</code>	Model specified in lavaan syntax (see <a href="#">lavaan::lavaanify</a> )
<code>est.mirt</code>	An optional logical indicating whether the model should be estimated with <a href="#">mirt::mirt</a>
<code>poly.itemtype</code>	Item type for polytomous data. This can be <code>gpcm</code> for the generalized partial credit model or <code>graded</code> for the graded response model.
<code>...</code>	Further arguments to be passed for estimation in <a href="#">mirt</a>

**Details**

This function uses the [lavaan::lavaanify](#) (**lavaan**) function.

Only single group models are supported (for now).

**Value**

A list with following entries

<code>mirt</code>	Object generated by mirt function if <code>est.mirt=TRUE</code>
<code>mirt.model</code>	Generated mirt model
<code>mirt.syntax</code>	Generated mirt syntax
<code>mirt.pars</code>	Generated parameter specifications in mirt
<code>lavaan.model</code>	Used lavaan model transformed by lavaanify function
<code>dat</code>	Used dataset. If necessary, only items used in the model are included in the dataset.

**Author(s)**

Alexander Robitzsch

**See Also**

See <http://lavaan.ugent.be/> for **lavaan** resources.

See <https://groups.google.com/forum/#!forum/lavaan> for discussion about the **lavaan** package.

See [mirt.wrapper](#) for convenience wrapper functions for [mirt::mirt](#) objects.

See [TAM::lavaanify.IRT](#) for extensions of `lavaanify`.

See [tam2mirt](#) for converting fitted objects in the **TAM** package into fitted [mirt::mirt](#) objects.



## Examples

```

## Not run:
#####
# EXAMPLE 1: Convert some lavaan syntax to mirt syntax for data.read
#####

library(mirt)
data(data.read)
dat <- data.read

#*****
#*** Model 1: Single factor model
lavmodel <- "
  # omit item C3
  F =~ A1+A2+A3+A4 + C1+C2+C4 + B1+B2+B3+B4
  F ~~ 1*F
  "

# convert syntax and estimate model
res <- lavaan2mirt( dat , lavmodel , verbose=TRUE , technical=list(NCYCLES=3) )
# inspect coefficients
coef(res$mirt)
mirt.wrapper.coef(res$mirt)
# converted mirt model and parameter table
cat(res$mirt.syntax)
res$mirt.pars

#*****
#*** Model 2: Rasch Model with first six items
lavmodel <- "
  F =~ a*A1+a*A2+a*A3+a*A4+a*B1+a*B2
  F ~~ 1*F
  "

# convert syntax and estimate model
res <- lavaan2mirt( dat , lavmodel , est.mirt=FALSE)
# converted mirt model
cat(res$mirt.syntax)
# mirt parameter table
res$mirt.pars
# estimate model using generated objects
res2 <- mirt::mirt( res$dat , res$mirt.model , pars=res$mirt.pars )
mirt.wrapper.coef(res2) # parameter estimates

#*****
#*** Model 3: Bifactor model
lavmodel <- "
  G =~ A1+A2+A3+A4 + B1+B2+B3+B4 + C1+C2+C3+C4
  A =~ A1+A2+A3+A4
  B =~ B1+B2+B3+B4
  C =~ C1+C2+C3+C4
  G ~~ 1*G
  A ~~ 1*A

```

```

      B ~~ 1*B
      C ~~ 1*C
      "

res <- lavaan2mirt( dat , lavmodel , est.mirt=FALSE )
# mirt syntax and mirt model
cat(res$mirt.syntax)
res$mirt.model
res$mirt.pars

#*****
#*** Model 4: 3-dimensional model with some parameter constraints
lavmodel <- "
  # some equality constraints among loadings
  A =~ a*A1+a*A2+a2*A3+a2*A4
  B =~ B1+B2+b3*B3+B4
  C =~ c*C1+c*C2+c*C3+c*C4
  # some equality constraints among thresholds
  A1 | da*t1
  A3 | da*t1
  B3 | da*t1
  C3 | dg*t1
  C4 | dg*t1
  # standardized latent variables
  A ~~ 1*A
  B ~~ 1*B
  C ~~ 1*C
  # estimate Cov(A,B) and Cov(A,C)
  A ~~ B
  A ~~ C
  # estimate mean of B
  B ~ 1
  "

res <- lavaan2mirt( dat , lavmodel , verbose=TRUE , technical=list(NCYCLES=3) )
# estimated parameters
mirt.wrapper.coef(res$mirt)
# generated mirt syntax
cat(res$mirt.syntax)
# mirt parameter table
mirt::mod2values(res$mirt)

#*****
#*** Model 5: 3-dimensional model with some parameter constraints and
#           parameter fixings
lavmodel <- "
  A =~ a*A1+a*A2+1.3*A3+A4 # set loading of A3 to 1.3
  B =~ B1+1*B2+b3*B3+B4
  C =~ c*C1+C2+c*C3+C4
  A1 | da*t1
  A3 | da*t1
  C4 | dg*t1
  B1 | 0*t1
  B3 | -1.4*t1 # fix item threshold of B3 to -1.4
  A ~~ 1*A

```

```

      B ~~ B          # estimate variance of B freely
      C ~~ 1*C
      A ~~ B          # estimate covariance between A and B
      A ~~ .6 * C     # fix covariance to .6
      A ~ .5*1        # set mean of A to .5
      B ~ 1           # estimate mean of B
      "

res <- lavaan2mirt( dat , lavmodel , verbose=TRUE , technical=list(NCYCLES=3) )
mirt.wrapper.coef(res$mirt)

#####
*** Model 6: 1-dimensional model with guessing and slipping parameters
#####

lavmodel <- "
  F =~ c*A1+c*A2+1*A3+1.3*A4 + C1__C4 + a*B1+b*B2+b*B3+B4
  # guessing parameters
  A1+A2 ?= guess1*g1
  A3 ?= .25*g1
  B1+C1 ?= g1
  B2__B4 ?= 0.10*g1
  # slipping parameters
  A1+A2+C3 ?= slip1*s1
  A3 ?= .02*s1
  # fix item intercepts
  A1 | 0*t1
  A2 | -.4*t1
  F ~ 1 # estimate mean of F
  F ~~ 1*F # fix variance of F
  "

# convert syntax and estimate model
res <- lavaan2mirt( dat , lavmodel , verbose=TRUE , technical=list(NCYCLES=3) )
# coefficients
mirt.wrapper.coef(res$mirt)
# converted mirt model
cat(res$mirt.syntax)

#####
# EXAMPLE 2: Convert some lavaan syntax to mirt syntax for
#           longitudinal data data.long
#####

data(data.long)
dat <- data.long[, -1]

#####
*** Model 1: Rasch model for T1
#####

lavmodel <- "
  F =~ 1*I1T1 +1*I2T1+1*I3T1+1*I4T1+1*I5T1+1*I6T1
  F ~~ F
  "

# convert syntax and estimate model
res <- lavaan2mirt( dat , lavmodel , verbose=TRUE , technical=list(NCYCLES=20) )

```

```

# inspect coefficients
mirt.wrapper.coef(res$mirt)
# converted mirt model
cat(res$mirt.syntax)

#####
*** Model 2: Rasch model for two time points
lavamodel <- "
  F1=~ 1*I1T1 +1*I2T1+1*I3T1+1*I4T1+1*I5T1+1*I6T1
  F2=~ 1*I3T2 +1*I4T2+1*I5T2+1*I6T2+1*I7T2+1*I8T2
  F1 ~~ F1
  F1 ~~ F2
  F2 ~~ F2
  # equal item difficulties of same items
  I3T1 | i3*t1
  I3T2 | i3*t1
  I4T1 | i4*t1
  I4T2 | i4*t1
  I5T1 | i5*t1
  I5T2 | i5*t1
  I6T1 | i6*t1
  I6T2 | i6*t1
  # estimate mean of F1, but fix mean of F2
  F1 ~ 1
  F2 ~ 0*1
  "

# convert syntax and estimate model
res <- lavaan2mirt( dat , lavamodel , verbose=TRUE , technical=list(NCYCLES=20) )
# inspect coefficients
mirt.wrapper.coef(res$mirt)
# converted mirt model
cat(res$mirt.syntax)

#-- compare estimation with smirt function
# define Q-matrix
I <- ncol(dat)
Q <- matrix(0,I,2)
Q[1:6,1] <- 1
Q[7:12,2] <- 1
rownames(Q) <- colnames(dat)
colnames(Q) <- c("T1","T2")
# vector with same items
itemnr <- as.numeric( substring( colnames(dat) ,2,2) )
# fix mean at T2 to zero
mu.fixed <- cbind( 2,0 )
# estimate model in smirt
mod1 <- smirt(dat, Qmatrix=Q , irtmodel="comp" , est.b= itemnr, mu.fixed=mu.fixed )
summary(mod1)

#####
# EXAMPLE 3: Converting lavaan syntax for polytomous data
#####

```

```

data(data.big5)
# select some items
items <- c( grep( "0" , colnames(data.big5) , value=TRUE )[1:6] ,
           grep( "N" , colnames(data.big5) , value=TRUE )[1:4] )
# 03 08 013 018 023 028 N1 N6 N11 N16
dat <- data.big5[ , items ]
library(psych)
psych::describe(dat)

#####
### Model 1: Partial credit model
lavmodel <- "
    0 =~ 1*03+1*08+1*013+1*018+1*023+1*028
    0 ~~ 0
    "

# estimate model in mirt
res <- lavaan2mirt( dat , lavmodel , technical=list(NCYCLES=20) , verbose=TRUE)
# estimated mirt model
mres <- res$mirt
# mirt syntax
cat(res$mirt.syntax)
## 0=1,2,3,4,5,6
## COV = 0*0
# estimated parameters
mirt.wrapper.coef(mres)
# some plots
mirt::itemplot( mres , 3 ) # third item
plot(mres) # item information
plot(mres,type="trace") # item category functions

# graded response model with equal slopes
res1 <- lavaan2mirt( dat , lavmodel , poly.itemtype="graded" , technical=list(NCYCLES=20),
                    verbose=TRUE )
mirt.wrapper.coef(res1$mirt)

#####
### Model 2: Generalized partial credit model with some constraints
lavmodel <- "
    0 =~ 03+08+013+a*018+a*023+1.2*028
    0 ~ 1 # estimate mean
    0 ~~ 0 # estimate variance
    # some constraints among thresholds
    03 | d1*t1
    013 | d1*t1
    03 | d2*t2
    08 | d3*t2
    028 | (-0.5)*t1
    "

# estimate model in mirt
res <- lavaan2mirt( dat , lavmodel , technical=list(NCYCLES=5) , verbose=TRUE)
# estimated mirt model
mres <- res$mirt
# estimated parameters

```

```

mirt.wrapper.coef(mres)

#### generate syntax for mirt for this model and estimate it in mirt package
# Items: 03 08 013 018 023 028
mirtmodel <- mirt::mirt.model( "
  0 = 1-6
  # a(018)=a(023), t1(03)=t1(018), t2(03)=t2(08)
  CONSTRAIN= (4,5,a1), (1,3,d1), (1,2,d2)
  MEAN = 0
  COV = 0*0
  ")
# initial table of parameters in mirt
mirt.pars <- mirt::mirt( dat[,1:6] , mirtmodel , itemtype="gpcm" , pars="values")
# fix slope of item 028 to 1.2
ind <- which( ( mirt.pars$item == "028" ) & ( mirt.pars$name == "a1" ) )
mirt.pars[ ind , "est"] <- FALSE
mirt.pars[ ind , "value"] <- 1.2
# fix d1 of item 028 to -0.5
ind <- which( ( mirt.pars$item == "028" ) & ( mirt.pars$name == "d1" ) )
mirt.pars[ ind , "est"] <- FALSE
mirt.pars[ ind , "value"] <- -0.5
# estimate model
res2 <- mirt::mirt( dat[,1:6] , mirtmodel , pars=mirt.pars ,
  verbose=TRUE , technical=list(NCYCLES=4) )
mirt.wrapper.coef(res2)
plot(res2, type="trace")

## End(Not run)

```

---

lc.2raters

*Latent Class Model for Two Exchangeable Raters and One Item*


---

## Description

This function computes a latent class model for ratings on an item based on exchangeable raters (Uebersax & Grove, 1990). Additionally, several measures of rater agreement are computed (see e.g. Gwet, 2010).

## Usage

```

lc.2raters(data, conv = 0.001, maxiter = 1000, progress = TRUE)

## S3 method for class 'lc.2raters'
summary(object,...)

```

## Arguments

**data** Data frame with item responses (must be ordered from 0 to  $K$ ) and two columns which correspond to ratings of two (exchangeable) raters.

conv	Convergence criterion
maxiter	Maximum number of iterations
progress	An optional logical indicating whether iteration progress should be displayed.
object	Object of class lc.2raters
...	Further arguments to be passed

### Details

For two exchangeable raters which provide ratings on an item, a latent class model with  $K + 1$  classes (if there are  $K + 1$  item categories  $0, \dots, K$ ) is defined. Where  $P(X = x, Y = y|c)$  denotes the probability that the first rating is  $x$  and the second rating is  $y$  given the true but unknown item category (class)  $c$ . Ratings are assumed to be locally independent, i.e.

$$P(X = x, Y = y|c) = P(X = x|c) \cdot P(Y = y|c) = p_{x|c} \cdot p_{y|c}$$

Note that  $P(X = x|c) = P(Y = x|c) = p_{x|c}$  holds due to the exchangeability of raters. The latent class model estimates true class proportions  $\pi_c$  and conditional item probabilities  $p_{x|c}$ .

### Value

A list with following entries

classprob.1rater.like	Classification probability $P(c x)$ of latent category $c$ given a manifest rating $x$ (estimated by maximum likelihood)
classprob.1rater.post	Classification probability $P(c x)$ of latent category $c$ given a manifest rating $x$ (estimated by the posterior distribution)
classprob.2rater.like	Classification probability $P(c (x, y))$ of latent category $c$ given two manifest ratings $x$ and $y$ (estimated by maximum likelihood)
classprob.2rater.post	Classification probability $P(c (x, y))$ of latent category $c$ given two manifest ratings $x$ and $y$ (estimated by posterior distribution)
f.yi.qk	Likelihood of each pair of ratings
f.qk.yi	Posterior of each pair of ratings
probs	Item response probabilities $p_{x c}$
pi.k	Estimated class proportions $\pi_c$
pi.k.obs	Observed manifest class proportions
freq.long	Frequency table of ratings in long format
freq.table	Symmetrized frequency table of ratings
agree.stats	Measures of rater agreement. These measures include percentage agreement (agree0, agree1), Cohen's kappa and weighted Cohen's kappa (kappa.wtd.kappa.linear), Gwet's AC1 agreement measures (AC1; Gwet, 2008, 2010) and Aickin's alpha (alpha.aickin; Aickin, 1990).
data	Used dataset
N.categ	Number of categories

**Author(s)**

Alexander Robitzsch

**References**

Aickin, M. (1990). Maximum likelihood estimation of agreement in the constant predictive probability model, and its relation to Cohen's kappa. *Biometrics*, **46**, 293-302.

Gwet, K. L. (2008). Computing inter-rater reliability and its variance in the presence of high agreement. *British Journal of Mathematical and Statistical Psychology*, **61**, 29-48.

Gwet, K. L. (2010). *Handbook of Inter-Rater Reliability*. Advanced Analytics, Gaithersburg. <http://www.agreestat.com/>

Uebersax, J. S., & Grove, W. M. (1990). Latent class analysis of diagnostic agreement. *Statistics in Medicine*, **9**, 559-572.

**See Also**

See also [rm.facets](#) and [rm.sdt](#) for specifying rater models.

See also the **irr** package for measures of rater agreement.

**Examples**

```
#####
# EXAMPLE 1: Latent class models for rating datasets data.si05
#####

data(data.si05)

### Model 1: one item with two categories
mod1 <- lc.2raters( data.si05$Ex1)
summary(mod1)

### Model 2: one item with five categories
mod2 <- lc.2raters( data.si05$Ex2)
summary(mod2)

### Model 3: one item with eight categories
mod3 <- lc.2raters( data.si05$Ex3)
summary(mod3)
```



**Description**

Approximates individual likelihood functions  $L(\mathbf{X}_p|\theta)$  by normal distributions (see Mislevy, 1990). Extreme response patterns are handled by adding pseudo-observations of items with extreme item difficulties (see argument `extreme.item`). The individual standard deviations of the likelihood, used in the normal approximation, can be modified by individual adjustment factors which are specified in `adjfac`. In addition, a reliability of the adjusted likelihood can be specified in `target.EAP.rel`.

**Usage**

```
likelihood.adjustment(likelihood, theta = NULL, prob.theta = NULL,
  adjfac = rep(1, nrow(likelihood)), extreme.item = 5, target.EAP.rel = NULL,
  min_tuning = 0.2, max_tuning = 3, maxiter = 100, conv = 1e-04,
  trait.normal = TRUE)
```

**Arguments**

<code>likelihood</code>	A matrix containing the individual likelihood $L(\mathbf{X}_p \theta)$ or an object of class <code>IRT.likelihood</code> .
<code>theta</code>	Optional vector of (unidimensional) $\theta$ values
<code>prob.theta</code>	Optional vector of probabilities of $\theta$ trait distribution
<code>adjfac</code>	Vector with individual adjustment factors of the standard deviations of the likelihood
<code>extreme.item</code>	Item difficulties of two extreme pseudo items which are added as additional observed data to the likelihood. A large number (e.g. <code>extreme.item=15</code> ) leaves the likelihood almost unaffected. See also Mislevy (1990).
<code>target.EAP.rel</code>	Target EAP reliability. An additional tuning parameter is estimated which adjusts the likelihood to obtain a pre-specified reliability.
<code>min_tuning</code>	Minimum value of tuning parameter (if <code>! is.null(target.EAP.rel)</code> )
<code>max_tuning</code>	Maximum value of tuning parameter (if <code>! is.null(target.EAP.rel)</code> )
<code>maxiter</code>	Maximum number of iterations (if <code>! is.null(target.EAP.rel)</code> )
<code>conv</code>	Convergence criterion (if <code>! is.null(target.EAP.rel)</code> )
<code>trait.normal</code>	Optional logical indicating whether the trait distribution should be normally distributed (if <code>! is.null(target.EAP.rel)</code> ).

**Value**

Object of class `IRT.likelihood`.

**Author(s)**

Alexander Robitzsch

**References**

Mislevy, R. (1990). Scaling procedures. In E. Johnson & R. Zwick (Eds.), *Focusing the new design: The NAEP 1988 technical report* (ETS RR 19-20). Princeton, NJ: Educational Testing Service.

**See Also**

[IRT.likelihood](#) (CDM), [TAM::tam.latreg](#)

**Examples**

```
## Not run:
#####
# EXAMPLE 1: Adjustment of the likelihood | data.read
#####

library(TAM)
data(data.read)
dat <- data.read

# define theta grid
theta.k <- seq(-6,6,len=41)

**** Model 1: fit Rasch model in TAM
mod1 <- TAM::tam.mml( dat , control=list( nodes=theta.k ) )
summary(mod1)

**** Model 2: fit Rasch copula model
testlets <- substring( colnames(dat) , 1 , 1 )
mod2 <- rasch.copula2( dat , itemcluster=testlets , theta.k=theta.k )
summary(mod2)

# model comparison
IRT.compareModels( mod1 , mod2 )

# extract EAP reliabilities
rel1 <- mod1$EAP.rel
rel2 <- mod2$EAP.Rel
# variance inflation factor
vif <- (1-rel2) / (1-rel1)
## > vif
## [1] 1.211644

# extract individual likelihood
like1 <- IRT.likelihood( mod1 )
# adjust likelihood from Model 1 to obtain a target EAP reliability of .599
like1b <- likelihood.adjustment( like1 , target.EAP.rel = .599 )

# compare estimated latent regressions
lmod1a <- TAM::tam.latreg( like1 , Y = NULL )
lmod1b <- TAM::tam.latreg( like1b , Y = NULL )
summary(lmod1a)
summary(lmod1b)

## End(Not run)
```

linking.haberman

*Linking in the 2PL/Generalized Partial Credit Model***Description**

This function does the linking of several studies which are calibrated using the 2PL or the generalized item response model according to Haberman (2009). This method is a generalization of log-mean-mean linking from two to several studies.

**Usage**

```
linking.haberman(itempars, personpars, conv = 1e-05, maxiter = 1000,
  progress = TRUE)
```

**Arguments**

itempars	A data frame with four or five columns. The first four columns contain in the order: study name, item name, $a$ parameter, $b$ parameter. The fifth column is an optional weight for every item and every study.
personpars	A list with vectors (e.g. EAPs or WLEs) or data frames (e.g. plausible values) containing person parameters which should be transformed. If a data frame in each list entry has se or SE (standard error) in a column name, then the corresponding column is only multiplied by $A_t$ . If a column is labeled as pid (person ID), then it is left untransformed.
conv	Convergence criterion.
maxiter	Maximum number of iterations.
progress	An optional logical indicating whether computational progress should be displayed.

**Details**

For  $t = 1, \dots, T$  studies, item difficulties  $b_{it}$  and item slopes  $a_{it}$  are available. For dichotomous responses, these parameters are defined by the 2PL response equation

$$\text{logit}P(X_{pi} = 1|\theta_p) = a_i(\theta_p - b_i)$$

while for polytomous responses the generalized partial credit model holds

$$\log \frac{P(X_{pi} = k|\theta_p)}{P(X_{pi} = k-1|\theta_p)} = a_i(\theta_p - b_i + d_{ik})$$

The parameters  $\{a_{it}, b_{it}\}$  of all items and studies are linearly transformed using equations  $a_{it} \approx a_i/A_t$  and  $b_{it} \cdot A_t \approx B_t + b_i$ . For identification reasons we define  $A_1 = 1$  and  $B_1 = 0$ .

The optimization function (which is a least squares criterion; see Haberman 2009) seeks the transformation parameters  $A_t$  and  $B_t$  with an alternating least squares method. Note that every item  $i$  and every study  $t$  can be weighted (specified in the fifth column of `itempars`).

Effect sizes of invariance are calculated as R-squared measures of explained item slopes and intercepts after linking in comparison to item parameters across groups (Asparouhov & Muthen, 2014).

**Value**

A list with following entries

transf.pars	Data frame with transformation parameters $A_t$ and $B_t$
transf.personpars	Data frame with linear transformation functions for person parameters
joint.itempars	Estimated joint item parameters $a_i$ and $b_i$
a.trans	Transformed $a_{it}$ parameters
b.trans	Transformed $b_{it}$ parameters
a.orig	Original $a_{it}$ parameters
b.orig	Original $b_{it}$ parameters
personpars	Transformed person parameters
es.invariance	Effect size measures of invariance, separately for item slopes and intercepts. In the rows, $R^2$ and $\sqrt{1 - R^2}$ are displayed.

**Author(s)**

Alexander Robitzsch

**References**

- Asparouhov, T., & Muthen, B. (2014). Multiple-group factor analysis alignment. *Structural Equation Modeling*, **21**, 1-14.
- Haberman, S. J. (2009). *Linking parameter estimates derived from an item response model through separate calibrations*. ETS Research Report ETS RR-09-40. Princeton, ETS.

**See Also**

See the **plink** package for a wide diversity of linking methods.

Mean-mean linking, Stocking-Lord and Haebara linking in the generalized logistic item response model can be conducted with [equating.rasch](#).

For more general linking functions than the Haberman method see [invariance.alignment](#).

**Examples**

```
#####
# EXAMPLE 1: Item parameters data.pars1.rasch and data.pars1.2pl
#####

# Model 1: Linking three studies calibrated by the Rasch model
data(data.pars1.rasch)
mod1 <- linking.haberman( itempars=data.pars1.rasch )

# Model 1b: Linking these studies but weigh these studies by
#   proportion weights 3 : 0.5 : 1 (see below).
#   All weights are the same for each item but they could also
```

```

# be item specific.
itempars <- data.pars1.rasch
itempars$wgt <- 1
itempars[ itempars$study == "study1", "wgt" ] <- 3
itempars[ itempars$study == "study2", "wgt" ] <- .5
mod1b <- linking.haberman( itempars=itempars )

# Model 2: Linking three studies calibrated by the 2PL model
data(data.pars1.2pl)
mod2 <- linking.haberman( itempars=data.pars1.2pl )

## Not run:
#####
# EXAMPLE 2: Linking longitudinal data
#####
data(data.long)

#*****
# Model 1: Scaling with the 1PL model

# scaling at T1
dat1 <- data.long[ , grep("T1" , colnames(data.long) ) ]
resT1 <- rasch.mm12( dat1 )
itempartable1 <- data.frame( "study"="T1" , resT1$item[ , c("item" , "a" , "b" ) ] )
# scaling at T2
dat2 <- data.long[ , grep("T2" , colnames(data.long) ) ]
resT2 <- rasch.mm12( dat2 )
summary(resT2)
itempartable2 <- data.frame( "study"="T2" , resT2$item[ , c("item" , "a" , "b" ) ] )
itempartable <- rbind( itempartable1 , itempartable2 )
itempartable[,2] <- substring( itempartable[,2] , 1 , 2 )
# estimate linking parameters
mod1 <- linking.haberman( itempars= itempartable )

#*****
# Model 2: Scaling with the 2PL model

# scaling at T1
dat1 <- data.long[ , grep("T1" , colnames(data.long) ) ]
resT1 <- rasch.mm12( dat1 , est.a=1:6 )
itempartable1 <- data.frame( "study"="T1" , resT1$item[ , c("item" , "a" , "b" ) ] )

# scaling at T2
dat2 <- data.long[ , grep("T2" , colnames(data.long) ) ]
resT2 <- rasch.mm12( dat2 , est.a=1:6 )
summary(resT2)
itempartable2 <- data.frame( "study"="T2" , resT2$item[ , c("item" , "a" , "b" ) ] )
itempartable <- rbind( itempartable1 , itempartable2 )
itempartable[,2] <- substring( itempartable[,2] , 1 , 2 )
# estimate linking parameters
mod2 <- linking.haberman( itempars= itempartable )

#####

```

```

# SIMULATED EXAMPLE 3: 2 Studies - 1PL and 2PL linking
#####
set.seed(789)
I <- 20 # number of items
N <- 2000 # number of persons
# define item parameters
b <- seq( -1.5 , 1.5 , length=I )
# simulate data
dat1 <- sim.raschtype( rnorm( N , mean=0,sd=1 ) , b=b )
dat2 <- sim.raschtype( rnorm( N , mean=0.5,sd=1.50 ) , b=b )

*** Model 1: 1PL
# 1PL Study 1
mod1 <- rasch.mml2( dat1 , est.a= rep(1,I) )
summary(mod1)
# 1PL Study 2
mod2 <- rasch.mml2( dat2 , est.a= rep(1,I) )
summary(mod2)

# collect item parameters
dfr1 <- data.frame( "study1" , mod1$item$item , mod1$item$a , mod1$item$b )
dfr2 <- data.frame( "study2" , mod2$item$item , mod2$item$a , mod2$item$b )
colnames(dfr2) <- colnames(dfr1) <- c("study" , "item" , "a" , "b" )
itempars <- rbind( dfr1 , dfr2 )

# Haberman linking
linkhab1 <- linking.haberman(itempars=itempars)
## Transformation parameters (Haberman linking)
##   study   At   Bt
## 1 study1 1.000 0.000
## 2 study2 1.465 -0.512
##
## Linear transformation for item parameters a and b
##   study  A_a  A_b  B_b
## 1 study1 1.000 1.000 0.000
## 2 study2 0.682 1.465 -0.512
##
## Linear transformation for person parameters theta
##   study  A_theta  B_theta
## 1 study1  1.000  0.000
## 2 study2  1.465  0.512
##
## R-Squared Measures of Invariance
##           slopes intercepts
## R2          1      0.9979
## sqrtU2       0      0.0456

*** Model 2: 2PL
# 2PL Study 1
mod1 <- rasch.mml2( dat1 , est.a= 1:I )
summary(mod1)
# 2PL Study 2
mod2 <- rasch.mml2( dat2 , est.a= 1:I )

```

```

summary(mod2)

# collect item parameters
dfr1 <- data.frame( "study1" , mod1$item$item , mod1$item$a , mod1$item$b )
dfr2 <- data.frame( "study2" , mod2$item$item , mod2$item$a , mod2$item$b )
colnames(dfr2) <- colnames(dfr1) <- c("study" , "item" , "a" , "b" )
itempars <- rbind( dfr1 , dfr2 )

# Haberman linking
linkhab2 <- linking.haberman(itempars=itempars)
## Transformation parameters (Haberman linking)
##   study   At   Bt
## 1 study1 1.000 0.000
## 2 study2 1.468 -0.515
##
## Linear transformation for item parameters a and b
##   study  A_a  A_b  B_b
## 1 study1 1.000 1.000 0.000
## 2 study2 0.681 1.468 -0.515
##
## Linear transformation for person parameters theta
##   study  A_theta  B_theta
## 1 study1   1.000   0.000
## 2 study2   1.468   0.515
##
## R-Squared Measures of Invariance
##           slopes intercepts
## R2      0.9984      0.9980
## sqrtU2 0.0397      0.0443

#####
# SIMULATED EXAMPLE 4: 3 Studies - 1PL and 2PL linking
#####
set.seed(789)
I <- 20      # number of items
N <- 1500    # number of persons
# define item parameters
b <- seq( -1.5 , 1.5 , length=I )
# simulate data
dat1 <- sim.raschtype( rnorm( N , mean=0,sd=1 ) , b=b )
dat2 <- sim.raschtype( rnorm( N , mean=0.5,sd=1.50 ) , b=b )
dat3 <- sim.raschtype( rnorm( N , mean=-.2,sd=.8 ) , b=b )
# set some items to non-administered
dat3 <- dat3[ , -c(1,4) ]
dat2 <- dat2[ , -c(1,2,3) ]

### Model 1: 1PL in sirt
# 1PL Study 1
mod1 <- rasch.mml2( dat1 , est.a= rep(1,ncol(dat1)) )
summary(mod1)
# 1PL Study 2
mod2 <- rasch.mml2( dat2 , est.a= rep(1,ncol(dat2)) )
summary(mod2)

```

```

# 1PL Study 3
mod3 <- rasch.mml2( dat3 , est.a= rep(1,ncol(dat3)) )
summary(mod3)

# collect item parameters
dfr1 <- data.frame( "study1" , mod1$item$item , mod1$item$a , mod1$item$b )
dfr2 <- data.frame( "study2" , mod2$item$item , mod2$item$a , mod2$item$b )
dfr3 <- data.frame( "study3" , mod3$item$item , mod3$item$a , mod3$item$b )
colnames(dfr3) <- colnames(dfr2) <- colnames(dfr1) <- c("study" , "item" , "a" , "b" )
itempars <- rbind( dfr1 , dfr2 , dfr3 )

# use person parameters
personpars <- list( mod1$person[ , c("EAP","SE.EAP") ] , mod2$person[ , c("EAP","SE.EAP") ] ,
  mod3$person[ , c("EAP","SE.EAP") ] )

# Haberman linking
linkhab1 <- linking.haberman(itempars=itempars , personpars=personpars)
# compare item parameters
round( cbind( linkhab1$joint.itempars[,-1], linkhab1$b.trans )[1:5,] , 3 )
##      aj      bj study1 study2 study3
## I0001 0.998 -1.427 -1.427      NA      NA
## I0002 0.998 -1.290 -1.324      NA -1.256
## I0003 0.998 -1.140 -1.068      NA -1.212
## I0004 0.998 -0.986 -1.003 -0.969      NA
## I0005 0.998 -0.869 -0.809 -0.872 -0.926

# summary of person parameters of second study
round( psych::describe( linkhab1$personpars[[2]] ) , 2 )
## var    n mean  sd median trimmed mad  min max range skew kurtosis
## EAP    1 1500 0.45 1.36  0.41  0.47 1.52 -2.61 3.25 5.86 -0.08 -0.62
## SE.EAP 2 1500 0.57 0.09  0.53  0.56 0.04  0.49 0.84 0.35 1.47  1.56
##      se
## EAP   0.04
## SE.EAP 0.00

**** Model 2: 2PL in TAM
library(TAM)
# 2PL Study 1
mod1 <- TAM::tam.mml.2pl( resp=dat1 , irtmodel="2PL" )
pvmod1 <- TAM::tam.pv(mod1, ntheta=300 , normal.approx=TRUE) # draw plausible values
summary(mod1)
# 2PL Study 2
mod2 <- TAM::tam.mml.2pl( resp=dat2 , irtmodel="2PL" )
pvmod2 <- TAM::tam.pv(mod2, ntheta=300 , normal.approx=TRUE)
summary(mod2)
# 2PL Study 3
mod3 <- TAM::tam.mml.2pl( resp=dat3 , irtmodel="2PL" )
pvmod3 <- TAM::tam.pv(mod3, ntheta=300 , normal.approx=TRUE)
summary(mod3)

# collect item parameters
#!! Note that in TAM the parametrization is a*theta - b while linking.haberman
#!! needs the parametrization a*(theta-b)

```



```

dfr1 <- data.frame( "study1" , mod1$item$item , mod1$B[,2,1] , mod1$xsi$xsi / mod1$B[,2,1] )
dfr2 <- data.frame( "study2" , mod2$item$item , mod2$B[,2,1] , mod2$xsi$xsi / mod2$B[,2,1] )
dfr3 <- data.frame( "study3" , mod3$item$item , mod3$B[,2,1] , mod3$xsi$xsi / mod3$B[,2,1] )
colnames(dfr3) <- colnames(dfr2) <- colnames(dfr1) <- c("study" , "item" , "a" , "b" )
itempars <- rbind( dfr1 , dfr2 , dfr3 )

# define list containing person parameters
personpars <- list( pvmod1$pv[,-1] , pvmod2$pv[,-1] , pvmod3$pv[,-1] )

# Haberman linking
linkhab2 <- linking.haberman(itempars=itempars,personpars=personpars)
## Linear transformation for person parameters theta
## study A_theta B_theta
## 1 study1 1.000 0.000
## 2 study2 1.485 0.465
## 3 study3 0.786 -0.192

# extract transformed person parameters
personpars.trans <- linkhab2$personpars

## End(Not run)

```

---

linking.robust

*Robust Linking of Item Intercepts*


---

## Description

This function implements a robust alternative of mean-mean linking which employs trimmed means instead of means. The linking constant is calculated for varying trimming parameters  $k$ .

## Usage

```

linking.robust(itempars)

## S3 method for class 'linking.robust'
summary(object,...)

## S3 method for class 'linking.robust'
plot(x, ...)

```

## Arguments

itempars	Data frame of item parameters (item intercepts). The first column contains the item label, the 2nd and 3rd columns item parameters of two studies.
object	Object of class linking.robust
x	Object of class linking.robust
...	Further arguments to be passed

**Value**

A list with following entries

ind.kopt	Index for optimal scale parameter
kopt	Optimal scale parameter
meanpars.kopt	Linking constant for optimal scale parameter
se.kopt	Standard error for linking constant obtained with optimal scale parameter
meanpars	Linking constant dependent on the scale parameter
se	Standard error of the linking constant dependent on the scale parameter
sd	DIF standard deviation (non-robust estimate)
mad	DIF standard deviation (robust estimate using the MAD measure)
pars	Original item parameters
k.robust	Used vector of scale parameters
I	Number of items
itempars	Used data frame of item parameters

**Author(s)**

Alexander Robitzsch

**See Also**

Other functions for linking: [linking.haberman](#), [equating.rasch](#)

See also the **plink** package.

**Examples**

```
#####
# EXAMPLE 1: Linking data.si03
#####

data(data.si03)
res1 <- linking.robust( itempars=data.si03 )
summary(res1)
## Number of items = 27
## Optimal trimming parameter k = 8 | non-robust parameter k = 0
## Linking constant = -0.0345 | non-robust estimate = -0.056
## Standard error = 0.0186 | non-robust estimate = 0.027
## DIF SD: MAD = 0.0771 (robust) | SD = 0.1405 (non-robust)
plot(res1)

## Not run:
#####
# EXAMPLE 2: Linking PISA item parameters data.pisaPars
#####

data(data.pisaPars)
```

```

# Linking with items
res2 <- linking.robust( data.pisaPars[ , c(1,3,4)] )
summary(res2)
  ## Optimal trimming parameter k = 0 | non-robust parameter k = 0
  ## Linking constant = -0.0883 | non-robust estimate = -0.0883
  ## Standard error = 0.0297 | non-robust estimate = 0.0297
  ## DIF SD: MAD = 0.1824 (robust) | SD = 0.1487 (non-robust)
## -> no trimming is necessary for reducing the standard error
plot(res2)

#####
# SIMULATED EXAMPLE 3: Linking with simulated item parameters containing outliers
#####

# simulate some parameters
I <- 38
set.seed(18785)
itempars <- data.frame("item" = paste0("I",1:I) )
itempars$study1 <- rnorm( I , mean = .3 , sd =1.4 )
# simulate DIF effects plus some outliers
bdif <- rnorm(I,mean=.4,sd=.09)+(runif(I)>.9)* rep( 1*c(-1,1)+.4 , each=I/2 )
itempars$study2 <- itempars$study1 + bdif

# robust linking
res <- linking.robust( itempars )
summary(res)
  ## Number of items = 38
  ## Optimal trimming parameter k = 12 | non-robust parameter k = 0
  ## Linking constant = -0.4285 | non-robust estimate = -0.5727
  ## Standard error = 0.0218 | non-robust estimate = 0.0913
  ## DIF SD: MAD = 0.1186 (robust) | SD = 0.5628 (non-robust)
## -> substantial differences of estimated linking constants in this case of
## deviations from normality of item parameters
plot(res)

## End(Not run)

```

## Description

This function estimates the least squares distance method of cognitive validation (Dimitrov, 2007; Dimitrov & Atanasov, 2012) which assumes a multiplicative relationship of attribute response probabilities to explain item response probabilities. The function also estimates the classical linear logistic test model (LLTM; Fischer, 1973) which assumes a linear relationship for item difficulties in the Rasch model.

**Usage**

```
lsdm(data, Qmatrix, theta=qnorm(seq(5e-04,0.9995,len=100)),
      quant.list=c(0.5,0.65,0.8), b=NULL, a=rep(1,nrow(Qmatrix)),
      c=rep(0,nrow(Qmatrix)) )

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

**Arguments**

data	An $I \times L$ matrix of dichotomous item responses. The data consists of $I$ item response functions (parametrically or nonparametrically estimated) which are evaluated at a discrete grid of $L$ theta values (person parameters) and are specified in the argument theta.
Qmatrix	An $I \times K$ matrix where the allocation of items to attributes is coded. Values of zero and one and all values between zero and one are permitted. There must not be any items with only zero Q-matrix entries in a row.
theta	The discrete grid points where item response functions are evaluated for doing the LSDM method.
quant.list	A vector of quantiles where attribute response functions are evaluated.
b	An optional vector of item difficulties. If it is specified, then no data input is necessary.
a	An optional vector of item discriminations.
c	An optional vector of guessing parameters.
object	Object of class lsdm
...	Further arguments to be passed

**Details**

The least squares distance method (LSDM; Dimitrov 2007) is based on the assumption that estimated item response functions  $P(X_i = 1|\theta)$  can be decomposed in a multiplicative way (in the implemented conjunctive model):

$$P(X_i = 1|\theta) = \prod_{k=1}^K [P(A_k = 1|\theta)]^{q_{ik}}$$

where  $P(A_k = 1|\theta)$  are attribute response functions and  $q_{ik}$  are entries of the Q-matrix. Note that the multiplicative form can be rewritten by taking the logarithm

$$\log P(X_i = 1|\theta) = \sum_{k=1}^K q_{ik} \log[P(A_k = 1|\theta)]$$

Evaluation item and attribute response functions on a grid of  $\theta$  values and collecting these values in matrices  $\mathbf{L} = \{\log P(X_i = 1|\theta)\}$ ,  $\mathbf{Q} = \{q_{ik}\}$  and  $\mathbf{X} = \{\log P(A_k = 1|\theta)\}$  leads to a least squares problem of the form  $\mathbf{L} \approx \mathbf{Q}\mathbf{X}$  with the restriction of positive  $\mathbf{X}$  matrix entries. This least

squares problem is a linear inequality constrained model which is solved by making use of the **ic.infer** package (Groemping, 2010).

After fitting the attribute response functions, empirical item-attribute discriminations  $w_{ik}$  are calculated as the approximation of the following equation

$$\log P(X_i = 1|\theta) = \sum_{k=1}^K w_{ik} q_{ik} \log[P(A_k = 1|\theta)]$$

### Value

A list with following entries

mean.mad.lsdm0	Mean of <i>MAD</i> statistics for LSDM
mean.mad.lltm	Mean of <i>MAD</i> statistics for LLTM
attr.curves	Estimated attribute response curves evaluated at theta
attr.pars	Estimated attribute parameters for LSDM and LLTM
data.fitted	LSDM-fitted item reponse functions evaluated at theta
theta	Grid of ability distributions at which functions are evaluated
item	Item statistics (p value, <i>MAD</i> , ...)
data	Estimated or fixed item reponse functions evaluated at theta
Qmatrix	Used Q-matrix
lltm	Model output of LLTM (1m values)
W	Matrix with empirical item-attribute discriminations

### Note

This function needs the **ic.infer** package.

### Author(s)

Alexander Robitzsch

### References

- DiBello, L. V., Roussos, L. A., & Stout, W. F. (2007). Review of cognitively diagnostic assessment and a summary of psychometric models. In C. R. Rao and S. Sinharay (Eds.), *Handbook of Statistics*, Vol. 26 (pp. 979-1030). Amsterdam: Elsevier.
- Dimitrov, D. M. (2007). Least squares distance method of cognitive validation and analysis for binary items using their item response theory parameters. *Applied Psychological Measurement*, **31**, 367-387.
- Dimitrov, D. M., & Atanasov, D. V. (2012). Conjunctive and disjunctive extensions of the least squares distance model of cognitive diagnosis. *Educational and Psychological Measurement*, **72**, 120-138.
- Fischer, G. H. (1973). The linear logistic test model as an instrument in educational research. *Acta Psychologica*, **37**, 359-374.

Groemping, U. (2010). Inference with linear equality and inequality constraints using R: The package **ic.infer**. *Journal of Statistical Software*, **33**(10), 1-31.

Sonnleitner, P. (2008). Using the LLTM to evaluate an item-generating system for reading comprehension. *Psychology Science*, **50**, 345-362.

### See Also

Get a summary of the LSDM analysis with [summary.lsdm](#).

See the **CDM** package for the estimation of related cognitive diagnostic models (DiBello, Roussos & Stout, 2007).

### Examples

```
#####
# EXAMPLE 1: DATA FISCHER (see Dimitrov, 2007)
#####

# item difficulties
b <- c( 0.171,-1.626,-0.729,0.137,0.037,-0.787,-1.322,-0.216,1.802,
        0.476,1.19,-0.768,0.275,-0.846,0.213,0.306,0.796,0.089,
        0.398,-0.887,0.888,0.953,-1.496,0.905,-0.332,-0.435,0.346,
        -0.182,0.906)

# read Q-matrix
Qmatrix <- c( 1,1,0,1,0,0,0,0,1,0,1,0,0,0,0,0,1,0,0,1,0,0,0,0,
             1,0,1,1,0,0,0,0,1,0,0,1,0,0,0,0,0,1,0,0,1,1,0,0,1,0,1,0,0,0,0,
             1,0,1,0,1,1,0,0,1,0,1,1,0,1,0,0,1,0,0,1,0,0,1,0,1,1,1,0,0,0,0,
             1,0,0,1,0,0,1,0,1,0,0,1,0,0,1,0,1,0,0,0,1,0,1,1,0,1,0,1,1,0,
             1,0,1,1,0,0,1,0,1,0,0,1,0,0,0,1,1,0,1,1,0,0,0,1,1,0,0,1,0,0,0,1,
             0,1,0,0,0,1,0,1,1,1,0,1,0,1,1,0,0,1,0,1,0,0,1,1,0,0,1,0,0,0,0,
             1,0,0,1,1,0,0,0,1,1,0,1,0,0,0,0,1,0,1,1,0,0,0,1,0,1,1,0,0,0,0,
             1,1,0,1,0,0,0,0,1,0,1,1,1,0,0,0,1,0,1,1,0,1,0,0,0,1,0,0,0,0,
             1,1,0,1,0,0,0,0,1,0,1,1,1,0,0,0 )

Qmatrix <- matrix( Qmatrix , nrow=29, byrow=TRUE )
colnames(Qmatrix) <- paste("A",1:8,sep="")
rownames(Qmatrix) <- paste("Item",1:29,sep="")

# Perform a LSDM analysis
lsdm.res <- lsdm( b = b, Qmatrix = Qmatrix )
summary(lsdm.res)
## Model Fit
## Model Fit LSDM - Mean MAD: 0.071 Median MAD: 0.07
## Model Fit LLTM - Mean MAD: 0.079 Median MAD: 0.063 R^2= 0.615
## .....
## Attribute Parameters
## N.Items b.2PL a.2PL b.1PL eta.LLTM se.LLTM pval.LLTM
## A1 27 -2.101 1.615 -2.664 -1.168 0.404 0.009
## A2 8 -3.736 3.335 -5.491 -0.645 0.284 0.034
## A3 12 -5.491 0.360 -2.685 -0.013 0.284 0.963
## A4 22 -0.081 0.744 -0.059 1.495 0.350 0.000
## A5 7 -2.306 0.580 -1.622 0.243 0.301 0.428
## A6 10 -1.946 0.542 -1.306 0.447 0.243 0.080
## A7 5 -4.247 1.283 -4.799 -0.147 0.316 0.646
## A8 5 -2.670 0.663 -2.065 0.077 0.310 0.806
```

```

## [...]

#####
# EXAMPLE 2 DATA HENNING (see Dimitrov, 2007)
#####

# item difficulties
b <- c(-2.03,-1.29,-1.03,-1.58,0.59,-1.65,2.22,-1.46,2.58,-0.66)
# item slopes
a <- c(0.6,0.81,0.75,0.81,0.62,0.75,0.54,0.65,0.75,0.54)
# define Q-matrix
Qmatrix <- c(1,0,0,0,0,1,0,0,0,0,1,0,1,0,0,1,0,0,0,1,1,0,0,
            0,0,0,1,0,0,1,0,0,0,1,0,0,0,0,1,1,1,0,1,0,0 )
Qmatrix <- matrix( Qmatrix , nrow=10, byrow=TRUE )
colnames(Qmatrix) <- paste("A",1:5,sep="")
rownames(Qmatrix) <- paste("Item",1:10,sep="")

# LSDM analysis
lsdm.res <- lsdm( b = b, a=a , Qmatrix = Qmatrix )
summary(lsdm.res)
## Model Fit LSDM - Mean MAD: 0.061 Median MAD: 0.06
## Model Fit LLTM - Mean MAD: 0.069 Median MAD: 0.069 R^2= 0.902
## .....
## Attribute Parameters
## N.Items b.2PL a.2PL b.1PL eta.LLTM se.LLTM pval.LLTM
## A1 2 -2.727 0.786 -2.367 -1.592 0.478 0.021
## A2 5 -2.099 0.794 -1.834 -0.934 0.295 0.025
## A3 2 -0.763 0.401 -0.397 1.260 0.507 0.056
## A4 4 -1.459 0.638 -1.108 -0.738 0.309 0.062
## A5 2 2.410 0.509 1.564 2.673 0.451 0.002
## [...]
##
## Discrimination Parameters
##
## A1 A2 A3 A4 A5
## Item1 1.723 NA NA NA NA
## Item2 NA 1.615 NA NA NA
## Item3 NA 0.650 NA 0.798 NA
## Item4 NA 1.367 NA NA NA
## Item5 NA 1.001 1.26 NA NA
## Item6 NA NA NA 0.866 NA
## Item7 NA 0.697 NA NA 0.891
## Item8 NA NA NA 0.997 NA
## Item9 NA NA NA 1.312 1.074
## Item10 1.000 NA 0.74 NA NA

## Not run:
#####
# EXAMPLE 3: PISA reading (data.pisaRead)
# using nonparametrically estimated item response functions
#####

data(data.pisaRead)

```

```

# response data
dat <- data.pisaRead$data
dat <- dat[ , substring( colnames(dat),1,1)=="R" ]
# define Q-matrix
pars <- data.pisaRead$item
Qmatrix <- data.frame( "A0" = 1*(pars$itemFormat=="MC" ) ,
                      "A1" = 1*(pars$itemFormat=="CR" ) )

# start with estimating the 1PL in order to get person parameters
mod <- rasch.mml2( dat )
theta <- wle.rasch( dat=dat , b = mod$item$b )$theta
# Nonparametric estimation of item response functions
mod2 <- np.dich( dat=dat , theta=theta , thetagrid = seq(-3,3,len=100) )

# LSDM analysis
lsdm.res <- lsdm( data=mod2$estimate , Qmatrix=Qmatrix , theta=mod2$theta )
summary(lsdm.res)
## Model Fit
## Model Fit LSDM - Mean MAD: 0.215 Median MAD: 0.151
## Model Fit LLTM - Mean MAD: 0.193 Median MAD: 0.119 R^2= 0.285
## .....
## Attribute Parameter
## N.Items b.2PL a.2PL b.1PL eta.LLTM se.LLTM pval.LLTM
## A0 5 1.326 0.705 1.289 -0.455 0.965 0.648
## A1 7 -1.271 1.073 -1.281 -1.585 0.816 0.081

#####
# EXAMPLE 4: Fraction subtraction dataset
#####

data( data.fraction1 , package="CDM")
data <- data.fraction1$data
q.matrix <- data.fraction1$q.matrix

#****
# Model 1: 2PL estimation
mod1 <- rasch.mml2( data , est.a=1:nrow(q.matrix) )

# LSDM analysis
lsdm.res1 <- lsdm( b=mod1$item$b , a=mod1$item$a , Qmatrix=q.matrix )
summary(lsdm.res1)
##
## Model Fit LSDM - Mean MAD: 0.076 Median MAD: 0.055
## Model Fit LLTM - Mean MAD: 0.153 Median MAD: 0.155 R^2= 0.801
## .....
## Attribute Parameter
## N.Items b.2PL a.2PL b.1PL eta.LLTM se.LLTM pval.LLTM
## QT1 14 -0.741 2.991 -1.115 -0.815 0.217 0.004
## QT2 8 -80.387 0.031 -4.806 -0.025 0.262 0.925
## QT3 12 -2.461 0.711 -2.006 -0.362 0.268 0.207
## QT4 9 -0.019 3.873 -0.100 1.465 0.345 0.002
## QT5 3 -3.062 0.375 -1.481 0.254 0.280 0.387

```



```

#####
#****
# Model 2: 1PL estimation
mod2 <- rasch.mml2( data )

# LSDM analysis
lsdm.res2 <- lsdm( b=mod1$item$b , Qmatrix=q.matrix )
summary(lsdm.res2)
##
## Model Fit LSDM - Mean MAD: 0.046 Median MAD: 0.03
## Model Fit LLTM - Mean MAD: 0.041 Median MAD: 0.042 R^2= 0.772

#####
# EXAMPLE 5: Dataset LLTM Sonnleitner Reading Comprehension (Sonnleitner, 2008)
#####

# item difficulties Table 7, p. 355 (Sonnleitner, 2008)
b <- c(-1.0189,1.6754,-1.0842,-.4457,-1.9419,-1.1513,2.0871,2.4874,-1.659,-1.197,-1.2437,
        2.1537,.3301,-.5181,-1.3024,-.8248,-.0278,1.3279,2.1454,-1.55,1.4277,.3301)
b <- b[-21] # remove Item 21

# Q-matrix Table 9 , p. 357 (Sonnleitner, 2008)
Qmatrix <- scan()
  1 0 0 0 0 0 7 4 0 0 0 0 0 1 0 0 0 0 0 5 1 0 0 0 0 1 1 0 1 0 0 0 0 9 1 0 1 0
  1 1 1 0 0 0 0 0 5 2 0 1 0 1 1 0 0 1 0 0 7 5 1 1 0 1 1 0 0 0 0 0 7 3 0 0 0
  0 1 0 0 0 0 0 2 6 1 0 0 0 0 0 0 0 0 0 2 6 1 0 0 0 1 0 0 0 0 0 1 7 4 1 0 0
  0 1 0 0 0 0 0 0 6 2 1 1 0 0 1 0 0 0 1 0 7 3 1 0 0 0 1 0 0 0 0 0 5 1 0 0 0
  0 0 0 0 0 0 1 0 4 1 0 0 0 1 0 0 0 0 0 0 0 6 1 0 1 1 0 0 1 0 0 0 0 6 3 0 1 1
  0 0 0 1 0 0 1 7 5 0 0 1 0 1 0 0 0 0 1 2 2 0 0 1 0 1 1 0 0 0 1 4 1 0 0 1
  0 1 0 0 1 0 0 5 1 0 0 1 0 1 0 0 0 0 1 7 2 0 0 1 0 0 0 0 0 1 0 5 1 0 0 1

Qmatrix <- matrix( as.numeric(Qmatrix) , nrow=21 , ncol=12 , byrow=TRUE )
colnames(Qmatrix) <- scan( what="character" , nlines=1)
  pc ic ier inc iui igc ch nro ncro td a t

# divide Q-matrix entries by maximum in each column
Qmatrix <- round(Qmatrix / matrix(apply(Qmatrix,2,max),21,12,byrow=TRUE) ,3)
# LSDM analysis
res <- lsdm( b=b , Qmatrix=Qmatrix )
summary(res)
##
## Model Fit LSDM - Mean MAD: 0.217 Median MAD: 0.178
## Model Fit LLTM - Mean MAD: 0.087 Median MAD: 0.062 R^2= 0.785

## End(Not run)

```

## Description

Local structural equation models (LSEM) are structural equation models (SEM) which are evaluated for each value of a pre-defined moderator variable (Hildenbrandt, Wilhelm, & Robitzsch, 2009). Like in nonparametric regression models, observations near a focal point - at which the model is evaluated - obtain higher weights, far distant observations obtain lower weight. The LSEM can be specified by making use of **lavaan** syntax. It is also possible to specify a discretized version of LSEM which means that values of the moderator are grouped into a small number of groups. The LSEM can be tested by employing a permutation test, see [lsem.permutationTest](#). The function `lsem.MGM.stepfunctions` outputs stepwise functions for a multiple group model evaluated at a grid of focal points of the moderator, specified in `moderator.grid`.

## Usage

```
lsem.estimate(data, moderator, moderator.grid, lavmodel, type="LSEM", h = 1.1,
  residualize=TRUE, fit_measures = c("rmsea", "cfi", "tli", "gfi", "srmr"),
  eps = 1e-08, verbose = TRUE, ...)

## S3 method for class 'lsem'
summary(object, file=NULL, digits=3, ...)

## S3 method for class 'lsem'
plot(x, parindex=NULL, ask=TRUE, ci = TRUE, lintrend = TRUE,
  parsummary = TRUE, ylim=NULL, xlab=NULL, ylab=NULL, main=NULL,
  digits=3, ...)

lsem.MGM.stepfunctions( object, moderator.grid )
```

## Arguments

<code>data</code>	Data frame
<code>moderator</code>	Variable name of the moderator
<code>moderator.grid</code>	Focal points at which the LSEM should be evaluated. If <code>type="MGM"</code> , breaks are defined in this vector.
<code>lavmodel</code>	Specified SEM in <b>lavaan</b> . The function <a href="#">sem (lavaan)</a> is used.
<code>type</code>	Type of estimated model. The default is <code>type="LSEM"</code> which means that a local structural equation model is estimated. A multiple group model with a discretized moderator as the grouping variable can be estimated with <code>type="MGM"</code> . In this case, the breaks must be defined in <code>moderator.grid</code> .
<code>h</code>	Bandwidth factor
<code>residualize</code>	Logical indicating whether a residualization should be applied.
<code>fit_measures</code>	Vector with names of fit measures following the labels in <b>lavaan</b>
<code>eps</code>	Minimum number for weights
<code>verbose</code>	Optional logical printing information about computation progress.
<code>object</code>	Object of class <code>lsem</code>
<code>file</code>	A file name in which the summary output will be written.

digits	Number of digits.
x	Object of class lsem.
parindex	Vector of indices for parameters in plot function.
ask	A logical which asks for changing the graphic for each parameter.
ci	Logical indicating whether confidence intervals should be plotted.
lintrend	Logical indicating whether a linear trend should be plotted.
parsummary	Logical indicating whether a parameter summary should be displayed.
ylim	Plot parameter ylim. Can be a list, see Examples.
xlab	Plot parameter xlab. Can be a vector.
ylab	Plot parameter ylab. Can be a vector.
main	Plot parameter main. Can be a vector.
...	Further arguments to be passed to <code>lavaan::sem</code> .

### Value

List with following entries

parameters	Data frame with all parameters estimated at focal points of moderator
weights	Data frame with weights at each focal point
bw	Used bandwidth
h	Used bandwidth factor
N	Sample size
moderator.density	Estimated frequencies and effective sample size for moderator at focal points
moderator.stat	Descriptive statistics for moderator
moderator	Variable name of moderator
moderator.grid	Used grid of focal points for moderator
moderator.grouped	Data frame with informations about grouping of moderator if type="MGM".
residualized.intercepts	Estimated intercept functions used for residualization.
lavmodel	Used lavaan model
data	Used data frame, possibly residualized if residualize=TRUE

### Author(s)

Alexander Robitzsch, Oliver Luedtke, Andrea Hildebrandt

### References

Hildebrandt, A., Wilhelm, O., & Robitzsch, A. (2009). Complementary and competing factor analytic approaches for the investigation of measurement invariance. *Review of Psychology*, **16**, 87-102.

**See Also**

[lsem.permutationTest](#)

**Examples**

```
## Not run:
#####
# EXAMPLE 1: data.lsem01 | Age differentiation
#####

data(data.lsem01)
dat <- data.lsem01

# specify lavaan model
lavmodel <- "
    F =~ v1+v2+v3+v4+v5
    F ~~ 1*F"

# define grid of moderator variable age
moderator.grid <- seq(4,23,1)

#####
### Model 1: estimate LSEM with bandwidth 2
mod1 <- lsem.estimate( dat , moderator="age" , moderator.grid=moderator.grid ,
                      lavmodel=lavmodel , h=2 , std.lv=TRUE)
summary(mod1)
plot(mod1 , parindex=1:5)

# perform permutation test for Model 1
pmod1 <- lsem.permutationTest( mod1 , B=10 )
    # only for illustrative purposes the number of permutations B is set
    # to a low number of 10
summary(pmod1)
plot(pmod1, type="global")

#####
### Model 2: estimate multiple group model with 4 age groups

# define breaks for age groups
moderator.grid <- seq( 3.5 , 23.5 , len=5) # 4 groups
# estimate model
mod2 <- lsem.estimate( dat , moderator="age" , moderator.grid=moderator.grid ,
                      lavmodel=lavmodel , type="MGM" , std.lv=TRUE)
summary(mod2)

# output step functions
smod2 <- lsem.MGM.stepfunctions( object=mod2 , moderator.grid=seq(4,23,1) )
str(smod2)

## End(Not run)
```

---

lsem.permutationTest *Permutation Test for a Local Structural Equation Model*

---

### Description

Performs a permutation test for testing the hypothesis that model parameter are independent of a moderator variable (see Hildebrandt, Wilhelm, & Robitzsch, 2009).

### Usage

```
lsem.permutationTest(lsem.object, B = 1000, residualize = TRUE, verbose = TRUE)
```

```
## S3 method for class 'lsem.permutationTest'
summary(object, file=NULL, digits=3, ...)
```

```
## S3 method for class 'lsem.permutationTest'
plot(x, type = "global", stattype = "SD",
     parindex = NULL, sig_add = TRUE, sig_level = 0.05, sig_pch=17, nonsig_pch=2,
     sig_cex = 1, sig_lab = "p value", stat_lab = "Test statistic",
     moderator_lab = NULL, digits = 3, title = NULL, parlabels = NULL,
     ask = TRUE, ...)
```

### Arguments

lsem.object	Fitted object of class lsem with <a href="#">lsem.estimate</a>
B	Number of permutation samples
residualize	Optional logical indicating whether residualization of the moderator should be performed for each permutation sample.
verbose	Optional logical printing information about computation progress.
object	Object of class lsem
file	A file name in which the summary output will be written.
digits	Number of digits.
...	Further arguments to be passed.
x	Object of class lsem
type	Type of the statistic to be plotted. If type="global", a global test will be displayed. If type="pointwise" for each value at the focal point (defined in <code>moderator.grid</code> ) are calculated.
stattype	Type of test statistics. Can be MAD (mean absolute deviation), SD (standard deviation) or <code>lin_slo</code> (linear slope).
parindex	Vector of indices of selected parameters.
sig_add	Logical indicating whether significance values (p values) should be displayed.
sig_level	Significance level.
sig_pch	Point symbol for significant values.

nonsig_pch	Point symbol for non-significant values.
sig_cex	Point size for graphic displaying p values
sig_lab	Label for significance value (p value).
stat_lab	Label of y axis for graphic with pointwise test statistic
moderator_lab	Label of the moderator.
title	Title of the plot. Can be a vector.
parlabels	Labels of the parameters. Can be a vector.
ask	A logical which asks for changing the graphic for each parameter.

### Value

List with following entries

teststat	Data frame with global test statistics. The statistics are SD, MAD and lin_slo with their corresponding p values.
parameters_pointwise_test	Data frame with pointwise test statistics.
parameters	Original parameters.
parameters	Parameters in permutation samples.
parameters_summary	Original parameter summary.
parameters_summary_M	Mean of each parameter in permutation sample.
parameters_summary_SD	Standard deviation (SD) statistic in permutation slope.
parameters_summary_MAD	Mean absolute deviation (MAD) statistic in permutation sample.
parameters_summary_MAD	Linear slope parameter in permutation sample.

### Author(s)

Alexander Robitzsch, Oliver Luedtke, Andrea Hildebrandt

### References

Hildebrandt, A., Wilhelm, O., & Robitzsch, A. (2009). Complementary and competing factor analytic approaches for the investigation of measurement invariance. *Review of Psychology*, **16**, 87-102.

### See Also

For Examples see [lsem.estimate](#).

---

 marginal.truescore.reliability

*True-Score Reliability for Dichotomous Data*


---

### Description

This function computes the marginal true-score reliability for dichotomous data (Dimitrov, 2003; May & Nicewander, 1994) for the four-parameter logistic item response model (see [rasch.mm12](#) for details regarding this IRT model).

### Usage

```
marginal.truescore.reliability(b, a=1+0*b ,c=0*b ,d=1+0*b,
  mean.traits=0, sd.traits=1, theta.k=seq(-6,6,len=200) )
```

### Arguments

b	Vector of item difficulties
a	Vector of item discriminations
c	Vector of guessing parameters
d	Vector of upper asymptotes
mean.traits	Mean of trait distribution
sd.traits	Standard deviation of trait distribution
theta.k	Grid at which the trait distribution should be evaluated

### Value

A list with following entries:

rel.test	Reliability of the test
item	True score variance (sig2.true, error variance (sig2.error) and item reliability (rel.item). Expected proportions correct are in the column pi.
pi	Average proportion correct for all items and persons
sig2.tau	True score variance $\sigma_{\tau}^2$ (calculated by the formula in May & Nicewander, 1994)
sig2.error	Error variance $\sigma_e^2$

### Author(s)

Alexander Robitzsch

### References

Dimitrov, D. (2003). Marginal true-score measures and reliability for binary items as a function of their IRT parameters. *Applied Psychological Measurement*, **27**, 440-458.

May, K., & Nicewander, W. A. (1994). Reliability and information functions for percentile ranks. *Journal of Educational Measurement*, **31**, 313-325.

**See Also**

See [greenyang.reliability](#) for calculating the reliability for multidimensional measures.

**Examples**

```
#####
# EXAMPLE 1: Dimitrov (2003) Table 1 - 2PL model
#####

# item discriminations
a <- 1.7*c(0.449,0.402,0.232,0.240,0.610,0.551,0.371,0.321,0.403,0.434,0.459,
          0.410,0.302,0.343,0.225,0.215,0.487,0.608,0.341,0.465)
# item difficulties
b <- c( -2.554,-2.161,-1.551,-1.226,-0.127,-0.855,-0.568,-0.277,-0.017,
        0.294,0.532,0.773,1.004,1.250,1.562,1.385,2.312,2.650,2.712,3.000 )

marginal.truescore.reliability( b=b , a =a )
## Reliability= 0.606

#####
# EXAMPLE 2: Dimitrov (2003) Table 2
# 3PL model: Poetry items (4 items)
#####

# slopes, difficulties and guessing parameters
a <- 1.7*c(1.169,0.724,0.554,0.706 )
b <- c(0.468,-1.541,-0.042,0.698 )
c <- c(0.159,0.211,0.197,0.177 )

res <- marginal.truescore.reliability( b=b , a =a , c=c)
## Reliability= 0.403
## > round( res$item , 3 )
## item pi sig2.tau sig2.error rel.item
## 1 1 0.463 0.063 0.186 0.252
## 2 2 0.855 0.017 0.107 0.135
## 3 3 0.605 0.026 0.213 0.107
## 4 4 0.459 0.032 0.216 0.130

#####
# EXAMPLE 3: Reading Data
#####
data( data.read)

###
# Model 1: 1PL
mod <- rasch.mml2( data.read )
marginal.truescore.reliability( b=mod$item$b )
## Reliability= 0.653

###
# Model 2: 2PL
mod <- rasch.mml2( data.read , est.a=1:12 )
```



```

marginal.truescore.reliability( b=mod$item$b , a=mod$item$a)
  ## Reliability= 0.696

## Not run:
# compare results with Cronbach's alpha and McDonald's omega
# posing a 'wrong model' for normally distributed data
library(psych)
psych::omega(dat , nfactors=1) # 1 factor
  ## Omega_h for 1 factor is not meaningful, just omega_t
  ## Omega
  ## Call: omega(m = dat, nfactors = 1)
  ## Alpha:          0.69
  ## G.6:            0.7
  ## Omega Hierarchical: 0.66
  ## Omega H asymptotic: 0.95
  ## Omega Total      0.69

##! Note that alpha in psych is the standardized one.

## End(Not run)

```

---

matrixfunctions.sirt *Some Matrix Functions*

---

## Description

Some matrix functions which are written in **Rcpp** for speed reasons.

## Usage

```

rowMaxs.sirt(matr) # rowwise maximum

rowMins.sirt(matr) # rowwise minimum

rowCumsums.sirt(matr) # rowwise cumulative sum

colCumsums.sirt(matr) # columnwise cumulative sum

rowIntervalIndex.sirt(matr,rn) # first index in row nn when matr(nn,zz) > rn(nn)

rowKSmallest.sirt(matr , K , break.ties=TRUE) # k smallest elements in a row
rowKSmallest2.sirt(matr , K )

```

## Arguments

matr	A numeric matrix
rn	A vector, usually a random number in applications
K	An integer indicating the number of smallest elements to be extracted
break.ties	A logical which indicates if ties are randomly broken. The default is TRUE.

## Details

The function `rowIntervalIndex.sirt` searches for all rows  $n$  the first index  $i$  for which  $\text{matr}(n, i) > \text{rn}(n)$  holds.

The functions `rowKSmallest.sirt` and `rowKSmallest2.sirt` extract the  $K$  smallest entries in a matrix row. For small numbers of  $K$  the function `rowKSmallest2.sirt` is the faster one.

## Value

The output of `rowMaxs.sirt` is a list with the elements `maxval` (rowwise maximum values) and `maxind` (rowwise maximum indices). The output of `rowMins.sirt` contains corresponding minimum values with entries `minval` and `minind`.

The output of `rowKSmallest.sirt` are two matrices: `smallval` contains the  $K$  smallest values whereas `smallind` contains the  $K$  smallest indices.

## Author(s)

Alexander Robitzsch

The **Rcpp** code for `rowCumsums.sirt` is copied from code of Romain Francois (<http://lists.r-forge.r-project.org/pipermail/rcpp-devel/2010-October/001198.html>).

## See Also

For other matrix functions see the **matrixStats** package.

## Examples

```
#####
# EXAMPLE 1: a small toy example (I)
#####
set.seed(789)
N1 <- 10 ; N2 <- 4
M1 <- round( matrix( runif(N1*N2) , nrow=N1 , ncol=N2) , 1 )

rowMaxs.sirt(M1)      # rowwise maximum
rowMins.sirt(M1)     # rowwise minimum
rowCumsums.sirt(M1)  # rowwise cumulative sum

# row index for exceeding a certain threshold value
matr <- M1
matr <- matr / rowSums( matr )
matr <- rowCumsums.sirt( matr )
rn <- runif(N1) # generate random numbers
rowIntervalIndex.sirt(matr,rn)

# select the two smallest values
rowKSmallest.sirt(matr=M1 , K=2)
rowKSmallest2.sirt(matr=M1 , K=2)
```

---

mcmc.2pno	<i>MCMC Estimation of the Two-Parameter Normal Ogive Item Response Model</i>
-----------	--

---

### Description

This function estimates the Two-Parameter normal ogive item response model by MCMC sampling (Johnson & Albert, 1999, p. 195ff.).

### Usage

```
mcmc.2pno(dat, weights=NULL, burnin = 500, iter = 1000, N.sampvalues = 1000,
          progress.iter = 50, save.theta = FALSE)
```

### Arguments

dat	Data frame with dichotomous item responses
weights	An optional vector with student sample weights
burnin	Number of burnin iterations
iter	Total number of iterations
N.sampvalues	Maximum number of sampled values to save
progress.iter	Display progress every progress.iter-th iteration. If no progress display is wanted, then choose progress.iter larger than iter.
save.theta	Should theta values be saved?

### Details

The two-parameter normal ogive item response model with a probit link function is defined by

$$P(X_{pi} = 1|\theta_p) = \Phi(a_i\theta_p - b_i) \quad , \quad \theta_p \sim N(0, 1)$$

Note that in this implementation non-informative priors for the item parameters are chosen (Johnson & Albert, 1999, p. 195ff.).

### Value

A list of class `mcmc.sirt` with following entries:

mcmcobj	Object of class <code>mcmc.list</code>
summary.mcmcobj	Summary of the <code>mcmcobj</code> object. In this summary the Rhat statistic and the mode estimate MAP is included. The variable <code>PercSEratio</code> indicates the proportion of the Monte Carlo standard error in relation to the total standard deviation of the posterior distribution.
burnin	Number of burnin iterations

iter	Total number of iterations
a.chain	Sampled values of $a_i$ parameters
b.chain	Sampled values of $b_i$ parameters
theta.chain	Sampled values of $\theta_p$ parameters
deviance.chain	Sampled values of Deviance values
EAP.rel	EAP reliability
person	Data frame with EAP person parameter estimates for $\theta_p$ and their corresponding posterior standard deviations
dat	Used data frame
weights	Used student weights
...	Further values

**Author(s)**

Alexander Robitzsch

**References**

Johnson, V. E., & Albert, J. H. (1999). *Ordinal Data Modeling*. New York: Springer.

**See Also**

S3 methods: [summary.mcmc.sirt](#), [plot.mcmc.sirt](#)

For estimating the 2PL model with marginal maximum likelihood see [rasch.mml2](#) or [smirt](#).

A hierarchical version of this model can be estimated with [mcmc.2pnoh](#).

**Examples**

```
## Not run:
#####
# EXAMPLE 1: Dataset Reading
#####
data(data.read)
# estimate 2PNO with MCMC with 3000 iterations and 500 burn-in iterations
mod <- mcmc.2pno( dat=data.read , iter=3000 , burnin=500 )
# plot MCMC chains
plot( mod$mcmcobj , ask=TRUE )
# write sampled chains into codafile
mcmclist2coda( mod$mcmcobj , name = "dataread_2pno" )
# summary
summary(mod)

#####
# SIMULATED EXAMPLE 2
#####
# simulate data
N <- 1000
I <- 10
```

```

b <- seq( -1.5 , 1.5 , len=I )
a <- rep( c(1,2) , I/2 )
theta1 <- rnorm(N)
dat <- sim.raschtype( theta=theta1 , fixed.a =a , b=b )

####
# Model 1: estimate model without weights
mod1 <- mcmc.2pno( dat , iter= 1500 , burnin=500)
mod1$summary.mcmcobj
plot( mod1$mcmcobj , ask=TRUE )

####
# Model 2: estimate model with weights
# define weights
weights <- c( rep( 5 , N/4 ) , rep( .2 , 3/4*N ) )
mod2 <- mcmc.2pno( dat , weights=weights , iter= 1500 , burnin=500)
mod1$summary.mcmcobj

## End(Not run)

```

---

mcmc.2pno.ml

*Random Item Response Model / Multilevel IRT Model*


---

## Description

This function enables the estimation of random item models and multilevel (or hierarchical) IRT models (Chaimongkol, Huffer & Kamata, 2007; Fox & Verhagen, 2010; van den Noortgate, de Boeck & Meulders, 2003; Asparouhov & Muthen, 2012; Muthen & Asparouhov, 2013, 2014). Dichotomous response data is supported using a probit link. Normally distributed responses can also be analyzed. See Details for a description of the implemented item response models.

## Usage

```

mcmc.2pno.ml(dat, group, link="logit" , est.b.M = "h", est.b.Var = "n",
  est.a.M = "f", est.a.Var = "n", burnin = 500, iter = 1000,
  N.sampvalues = 1000, progress.iter = 50, prior.sigma2 = c(1, 0.4),
  prior.sigma.b = c(1, 1), prior.sigma.a = c(1, 1), prior.omega.b = c(1, 1),
  prior.omega.a = c(1, 0.4) , sigma.b.init=.3 )

```

## Arguments

dat	Data frame with item responses.
group	Vector of group identifiers (e.g. classes, schools or countries)
link	Link function. Choices are "logit" for dichotomous data and "normal" for data under normal distribution assumptions
est.b.M	Estimation type of $b_i$ parameters: n - non-hierarchical prior distribution, i.e. $\omega_b$ is set to a very high value and is not estimated

	h - hierarchical prior distribution with estimated distribution parameters $\mu_b$ and $\omega_b$
est.b.Var	Estimation type of standard deviations of item difficulties $b_i$ . n – no estimation of the item variance, i.e. $\sigma_{b,i}$ is assumed to be zero i – item-specific standard deviation of item difficulties j – a joint standard deviation of all item difficulties is estimated, i.e. $\sigma_{b,1} = \dots = \sigma_{b,I} = \sigma_b$
est.a.M	Estimation type of $a_i$ parameters: f - no estimation of item slopes, i.e all item slopes $a_i$ are fixed at one n - non-hierarchical prior distribution, i.e. $\omega_a = 0$ h - hierarchical prior distribution with estimated distribution parameter $\omega_a$
est.a.Var	Estimation type of standard deviations of item slopes $a_i$ . n – no estimation of the item variance i – item-specific standard deviation of item slopes j – a joint standard deviation of all item slopes is estimated, i.e. $\sigma_{a,1} = \dots = \sigma_{a,I} = \sigma_a$
burnin	Number of burnin iterations
iter	Total number of iterations
N.sampvalues	Maximum number of sampled values to save
progress.iter	Display progress every progress.iter-th iteration. If no progress display is wanted, then choose progress.iter larger than iter.
prior.sigma2	Prior for Level 2 standard deviation $\sigma_{L2}$
prior.sigma.b	Priors for item difficulty standard deviations $\sigma_{b,i}$
prior.sigma.a	Priors for item difficulty standard deviations $\sigma_{a,i}$
prior.omega.b	Prior for $\omega_b$
prior.omega.a	Prior for $\omega_a$
sigma.b.init	Initial standard deviation for $\sigma_{b,i}$ parameters

## Details

For dichotomous item responses (link="logit") of persons  $p$  in group  $j$  on item  $i$ , the probability of a correct response is defined as

$$P(X_{pji} = 1 | \theta_{pj}) = \Phi(a_{ij}\theta_{pj} - b_{ij})$$

The ability  $\theta_{pj}$  is decomposed into a Level 1 and a Level 2 effect

$$\theta_{pj} = u_j + e_{pj} \quad , \quad u_j \sim N(0, \sigma_{L2}^2) \quad , \quad e_{pj} \sim N(0, \sigma_{L1}^2)$$

In a multilevel IRT model (or a random item model), item parameters are allowed to vary across groups:

$$b_{ij} \sim N(b_i, \sigma_{b,i}^2) \quad , \quad a_{ij} \sim N(a_i, \sigma_{a,i}^2)$$

In a hierarchical IRT model, a hierarchical distribution of the (main) item parameters is assumed

$$b_i \sim N(\mu_b, \omega_b^2) \quad , \quad a_i \sim N(1, \omega_a^2)$$

Note that for identification purposes, the mean of all item slopes  $a_i$  is set to one. Using the arguments `est.b.M`, `est.b.Var`, `est.a.M` and `est.a.Var` defines which variance components should be estimated.

For normally distributed item responses (`link="normal"`), the model equations remain the same except the item response model which is now written as

$$X_{pji} = a_{ij}\theta_{pj} - b_{ij} + \varepsilon_{pji} \quad , \quad \varepsilon_{pji} \sim N(0, \sigma_{res,i}^2)$$

## Value

A list of class `mcmc.sirt` with following entries:

<code>mcmcobj</code>	Object of class <code>mcmc.list</code>
<code>summary.mcmcobj</code>	Summary of the <code>mcmcobj</code> object. In this summary the Rhat statistic and the mode estimate MAP is included. The variable <code>PercSEratio</code> indicates the proportion of the Monte Carlo standard error in relation to the total standard deviation of the posterior distribution.
<code>ic</code>	Information criteria (DIC)
<code>burnin</code>	Number of burnin iterations
<code>iter</code>	Total number of iterations
<code>theta.chain</code>	Sampled values of $\theta_{pj}$ parameters
<code>theta.chain</code>	Sampled values of $u_j$ parameters
<code>deviance.chain</code>	Sampled values of Deviance values
<code>EAP.rel</code>	EAP reliability
<code>person</code>	Data frame with EAP person parameter estimates for $\theta_{pj}$ and their corresponding posterior standard deviations
<code>dat</code>	Used data frame
<code>...</code>	Further values

## Author(s)

Alexander Robitzsch

## References

- Asparouhov, T. & Muthen, B. (2012). General random effect latent variable modeling: Random subjects, items, contexts, and parameters. [http://www.statmodel.com/papers\\_date.shtml](http://www.statmodel.com/papers_date.shtml).
- Chaimongkol, S., Huffer, F. W., & Kamata, A. (2007). An explanatory differential item functioning (DIF) model by the WinBUGS 1.4. *Songklanakarin Journal of Science and Technology*, **29**, 449-458.
- Fox, J.-P., & Verhagen, A.-J. (2010). Random item effects modeling for cross-national survey data. In E. Davidov, P. Schmidt, & J. Billiet (Eds.), *Cross-cultural Analysis: Methods and Applications* (pp. 467-488), London: Routledge Academic.
- Muthen, B. & Asparouhov, T. (2013). New methods for the study of measurement invariance with many groups. [http://www.statmodel.com/papers\\_date.shtml](http://www.statmodel.com/papers_date.shtml)

Muthen, B. & Asparouhov, T. (2014). Item response modeling in Mplus: A multi-dimensional, multi-level, and multi-timepoint example. In W. Linden & R. Hambleton (2014). *Handbook of item response theory: Models, statistical tools, and applications*. [http://www.statmodel.com/papers\\_date.shtml](http://www.statmodel.com/papers_date.shtml)

van den Noortgate, W., De Boeck, P., & Meulders, M. (2003). Cross-classification multilevel logistic models in psychometrics. *Journal of Educational and Behavioral Statistics*, **28**, 369-386.

### See Also

S3 methods: [summary.mcmc.sirt](#), [plot.mcmc.sirt](#)

For MCMC estimation of three-parameter (testlet) models see [mcmc.3pno.testlet](#).

See also the **MLIRT** package (<http://www.jean-paulfox.com>).

For more flexible estimation of multilevel IRT models see the **MCMCglmm** and **lme4** packages.

### Examples

```
## Not run:
#####
# EXAMPLE 1: Dataset Multilevel data.ml1 - dichotomous items
#####
data(data.ml1)
dat <- data.ml1[,-1]
group <- data.ml1$group
# just for a try use a very small number of iterations
burnin <- 50 ; iter <- 100

####
# Model 1: 1PNO with no cluster item effects
mod1 <- mcmc.2pno.ml( dat , group , est.b.Var="n" , burnin=burnin , iter=iter )
summary(mod1) # summary
plot(mod1,layout=2,ask=TRUE) # plot results
# write results to coda file
mcmcList2coda( mod1$mcmcobj , name = "data.ml1_mod1" )

####
# Model 2: 1PNO with cluster item effects of item difficulties
mod2 <- mcmc.2pno.ml( dat , group , est.b.Var="i" , burnin=burnin , iter=iter )
summary(mod2)
plot(mod2, ask=TRUE , layout=2 )

####
# Model 3: 2PNO with cluster item effects of item difficulties but
#           jointly estimated item slopes
mod3 <- mcmc.2pno.ml( dat , group , est.b.Var="i" , est.a.M="h" ,
                     burnin=burnin , iter=iter )
summary(mod3)

####
# Model 4: 2PNO with cluster item effects of item difficulties and
#           jointly estimated item slopes and a jointly estimated SD
mod4 <- mcmc.2pno.ml( dat , group , est.b.Var="i" , est.a.M="h" ,
```



```

        est.a.Var="j" , burnin=burnin , iter=iter )
summary(mod4)

#####
# EXAMPLE 2: Dataset Multilevel data.ml2 - polytomous items
#           assuming a normal distribution for polytomous items
#####
data(data.ml2)
dat <- data.ml2[,-1]
group <- data.ml2$group
# set iterations for all examples (too few!!)
burnin <- 100 ; iter <- 500

***
# Model 1: no intercept variance, no slopes
mod1 <- mcmc.2pno.ml( dat=dat , group=group , est.b.Var="n" ,
                    burnin=burnin , iter=iter , link="normal" , progress.iter=20 )
summary(mod1)

***
# Model 2a: itemwise intercept variance, no slopes
mod2a <- mcmc.2pno.ml( dat=dat , group=group , est.b.Var="i" ,
                    burnin=burnin , iter=iter , link="normal" , progress.iter=20 )
summary(mod2a)

***
# Model 2b: homogeneous intercept variance, no slopes
mod2b <- mcmc.2pno.ml( dat=dat , group=group , est.b.Var="j" ,
                    burnin=burnin , iter=iter , link="normal" , progress.iter=20 )
summary(mod2b)

***
# Model 3: intercept variance and slope variances
#           hierarchical item and slope parameters
mod3 <- mcmc.2pno.ml( dat=dat , group=group ,
                    est.b.M="h" , est.b.Var="i" , est.a.M="h" , est.a.Var="i" ,
                    burnin=burnin , iter=iter , link="normal" , progress.iter=20 )
summary(mod3)

#####
# SIMULATED EXAMPLE 3: Simulated random effects model | dichotomous items
#####
set.seed(7698)

*** model parameters
sig2.lev2 <- .3^2 # theta level 2 variance
sig2.lev1 <- .8^2 # theta level 1 variance
G <- 100         # number of groups
n <- 20         # number of persons within a group
I <- 12         # number of items
*** simulate theta
theta2 <- rnorm( G , sd = sqrt(sig2.lev2) )
theta1 <- rnorm( n*G , sd = sqrt(sig2.lev1) )

```

```

theta <- theta1 + rep( theta2 , each=n )
**** item difficulties
b <- seq( -2 , 2 , len=I )
**** define group identifier
group <- 1000 + rep(1:G , each=n )
**** SD of group specific difficulties for items 3 and 5
sigma.item <- rep(0,I)
sigma.item[c(3,5)] <- 1
**** simulate group specific item difficulties
b.class <- sapply( sigma.item , FUN = function(sii){ rnorm( G , sd = sii ) } )
b.class <- b.class[ rep( 1:G ,each=n ) , ]
b <- matrix( b , n*G , I , byrow=TRUE ) + b.class
**** simulate item responses
m1 <- pnorm( theta - b )
dat <- 1 * ( m1 > matrix( runif( n*G*I ) , n*G , I ) )

**** estimate model
mod <- mcmc.2pno.ml( dat , group=group , burnin=burnin , iter=iter ,
                    est.b.M="n" , est.b.Var="i" , progress.iter=20)
summary(mod)
plot(mod , layout=2 , ask=TRUE )

## End(Not run)

```

---

mcmc.2pnoh

---

*MCMC Estimation of the Hierarchical IRT Model for Criterion-Referenced Measurement*


---

## Description

This function estimates the hierarchical IRT model for criterion-referenced measurement which is based on a two-parameter normal ogive response function (Janssen, Tuerlinckx, Meulders & de Boeck, 2000).

## Usage

```

mcmc.2pnoh(dat, itemgroups , prob.mastery=c(.5,.8) , weights=NULL ,
           burnin = 500, iter = 1000, N.sampvalues = 1000,
           progress.iter = 50, prior.variance=c(1,1) , save.theta = FALSE)

```

## Arguments

dat	Data frame with dichotomous item responses
itemgroups	Vector with characters or integers which define the criterion to which an item is associated.
prob.mastery	Probability levels which define nonmastery, transition and mastery stage (see Details)
weights	An optional vector with student sample weights

burnin	Number of burnin iterations
iter	Total number of iterations
N.sampvalues	Maximum number of sampled values to save
progress.iter	Display progress every progress.iter-th iteration. If no progress display is wanted, then choose progress.iter larger than iter.
prior.variance	Scale parameter of the inverse gamma distribution for the $\sigma^2$ and $\nu^2$ item variance parameters
save.theta	Should theta values be saved?

### Details

The hierarchical IRT model for criterion-referenced measurement (Janssen et al., 2000) assumes that every item  $i$  intends to measure a criterion  $k$ . The item response function is defined as

$$P(X_{pik} = 1|\theta_p) = \Phi[\alpha_{ik}(\theta_p - \beta_{ik})] \quad , \quad \theta_p \sim N(0, 1)$$

Item parameters  $(\alpha_{ik}, \beta_{ik})$  are hierarchically modelled, i.e.

$$\beta_{ik} \sim N(\xi_k, \sigma^2) \quad \text{and} \quad \alpha_{ik} \sim N(\omega_k, \nu^2)$$

In the `mcmc.list` output object, also the derived parameters  $d_{ik} = \alpha_{ik}\beta_{ik}$  and  $\tau_k = \xi_k\omega_k$  are calculated. Mastery and nonmastery probabilities are based on a reference item  $Y_k$  of criterion  $k$  and a response function

$$P(Y_{pk} = 1|\theta_p) = \Phi[\omega_k(\theta_p - \xi_k)] \quad , \quad \theta_p \sim N(0, 1)$$

With known item parameters and person parameters, response probabilities of criterion  $k$  are calculated. If a response probability of criterion  $k$  is larger than `prob.mastery[2]`, then a student is defined as a master. If this probability is smaller than `prob.mastery[1]`, then a student is a nonmaster. In all other cases, students are in a transition stage.

In the `mcmcobj` output object, the parameters `d[i]` are defined by  $d_{ik} = \alpha_{ik} \cdot \beta_{ik}$  while `tau[k]` are defined by  $\tau_k = \xi_k \cdot \omega_k$ .

### Value

A list of class `mcmc.sirt` with following entries:

<code>mcmcobj</code>	Object of class <code>mcmc.list</code>
<code>summary.mcmcobj</code>	Summary of the <code>mcmcobj</code> object. In this summary the Rhat statistic and the mode estimate MAP is included. The variable <code>PercSEratio</code> indicates the proportion of the Monte Carlo standard error in relation to the total standard deviation of the posterior distribution.
<code>burnin</code>	Number of burnin iterations
<code>iter</code>	Total number of iterations
<code>alpha.chain</code>	Sampled values of $\alpha_{ik}$ parameters
<code>beta.chain</code>	Sampled values of $\beta_{ik}$ parameters

xi.chain	Sampled values of $\xi_k$ parameters
omega.chain	Sampled values of $\omega_k$ parameters
sigma.chain	Sampled values of $\sigma$ parameter
nu.chain	Sampled values of $\nu$ parameter
theta.chain	Sampled values of $\theta_p$ parameters
deviance.chain	Sampled values of Deviance values
EAP.rel	EAP reliability
person	Data frame with EAP person parameter estimates for $\theta_p$ and their corresponding posterior standard deviations
dat	Used data frame
weights	Used student weights
...	Further values

### Author(s)

Alexander Robitzsch

### References

Janssen, R., Tuerlinckx, F., Meulders, M., & de Boeck, P. (2000). A hierarchical IRT model for criterion-referenced measurement. *Journal of Educational and Behavioral Statistics*, **25**, 285-306.

### See Also

S3 methods: [summary.mcmc.sirt](#), [plot.mcmc.sirt](#)

The two-parameter normal ogive model can be estimated with [mcmc.2pno](#).

### Examples

```
## Not run:
#####
# SIMULATED EXAMPLE 1: Simulated data according to Janssen et al. (2000, Table 2)
#####

N <- 1000
Ik <- c(4,6,8,5,9,6,8,6,5)
xi.k <- c( -.89 , -1.13 , -1.23 , .06 , -1.41 , -.66 , -1.09 , .57 , -2.44)
omega.k <- c(.98 , .91 , .76 , .74 , .71 , .80 , .79 , .82 , .54)

# select 4 attributes
K <- 4
Ik <- Ik[1:K] ; xi.k <- xi.k[1:K] ; omega.k <- omega.k[1:K]
sig2 <- 3.02
nu2 <- .09
I <- sum(Ik)
b <- rep( xi.k , Ik ) + rnorm(I , sd = sqrt(sig2) )
a <- rep( omega.k , Ik ) + rnorm(I , sd = sqrt(nu2) )
```

```

theta1 <- rnorm(N)
t1 <- rep(1,N)
p1 <- pnorm( outer(t1,a) * ( theta1 - outer(t1,b) ) )
dat <- 1 * ( p1 > runif(N*I) )
itemgroups <- rep( paste0("A" , 1:K ) , Ik )

# estimate model
mod <- mcmc.2pnoh(dat , itemgroups , burnin=200 , iter=1000 )
# summary
summary(mod)
# plot
plot(mod$mcmcobj , ask=TRUE)
# write coda files
mcmclist2coda( mod$mcmcobj , name = "simul_2pnoh" )

## End(Not run)

```

---

mcmc.3pno.testlet      *3PNO Testlet Model*

---

## Description

This function estimates the 3PNO testlet model (Wang, Bradlow & Wainer, 2002, 2007) by Markov Chain Monte Carlo methods (Glas, 2012).

## Usage

```

mcmc.3pno.testlet(dat, testlets = rep(NA, ncol(dat)),
  weights = NULL, est.slope = TRUE, est.guess = TRUE, guess.prior = NULL,
  testlet.variance.prior = c(1, 0.2), burnin = 500, iter = 1000,
  N.sampvalues = 1000, progress.iter = 50, save.theta = FALSE)

```

## Arguments

<code>dat</code>	Data frame with dichotomous item responses for $N$ persons and $I$ items
<code>testlets</code>	An integer or character vector which indicates the allocation of items to testlets. Same entries corresponds to same testlets. If an entry is NA, then this item does not belong to any testlet.
<code>weights</code>	An optional vector with student sample weights
<code>est.slope</code>	Should item slopes be estimated? The default is TRUE.
<code>est.guess</code>	Should guessing parameters be estimated? The default is TRUE.
<code>guess.prior</code>	A vector of length two or a matrix with $I$ items and two columns which defines the beta prior distribution of guessing parameters. The default is a non-informative prior, i.e. the Beta(1,1) distribution.

testlet.variance.prior	A vector of length two which defines the (joint) prior for testlet variances assuming an inverse chi-squared distribution. The first entry is the effective sample size of the prior while the second entry defines the prior variance of the testlet. The default of $c(1, .2)$ means that the prior sample size is 1 and the prior testlet variance is .2.
burnin	Number of burnin iterations
iter	Number of iterations
N.sampvalues	Maximum number of sampled values to save
progress.iter	Display progress every progress.iter-th iteration. If no progress display is wanted, then choose progress.iter larger than iter.
save.theta	Should theta values be saved?

### Details

The testlet response model for person  $p$  at item  $i$  is defined as

$$P(X_{pi} = 1) = c_i + (1 - c_i)\Phi(a_i\theta_p + \gamma_{p,t(i)} + b_i) \quad , \quad \theta_p \sim N(0, 1), \gamma_{p,t(i)} \sim N(0, \sigma_t^2)$$

In case of `est.slope=FALSE`, all item slopes  $a_i$  are set to 1. Then a variance  $\sigma^2$  of the  $\theta_p$  distribution is estimated which is called the Rasch testlet model in the literature (Wang & Wilson, 2005).

In case of `est.guess=FALSE`, all guessing parameters  $c_i$  are set to 0.

After fitting the testlet model, marginal item parameters are calculated (integrating out testlet effects  $\gamma_{p,t(i)}$ ) according the defining response equation

$$P(X_{pi} = 1) = c_i + (1 - c_i)\Phi(a_i^*\theta_p + b_i^*)$$

### Value

A list of class `mcmc.sirt` with following entries:

mcmcobj	Object of class <code>mcmc.list</code> containing item parameters ( <code>b_marg</code> and <code>a_marg</code> denote marginal item parameters) and person parameters (if requested)
summary.mcmcobj	Summary of the <code>mcmcobj</code> object. In this summary the <code>Rhat</code> statistic and the mode estimate <code>MAP</code> is included. The variable <code>PercSERatio</code> indicates the proportion of the Monte Carlo standard error in relation to the total standard deviation of the posterior distribution.
ic	Information criteria (DIC)
burnin	Number of burnin iterations
iter	Total number of iterations
theta.chain	Sampled values of $\theta_p$ parameters
deviance.chain	Sampled values of deviance values
EAP.rel	EAP reliability
person	Data frame with EAP person parameter estimates for $\theta_p$ and their corresponding posterior standard deviations and for all testlet effects

dat	Used data frame
weights	Used student weights
...	Further values

**Author(s)**

Alexander Robitzsch

**References**

Glas, C. A. W. (2012). *Estimating and testing the extended testlet model*. LSAC Research Report Series, RR 12-03.

Wainer, H., Bradlow, E. T., & Wang, X. (2007). *Testlet response theory and its applications*. Cambridge: Cambridge University Press.

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

Wang, X., Bradlow, E. T., & Wainer, H. (2002). A general Bayesian model for testlets: Theory and applications. *Applied Psychological Measurement*, **26**, 109-128.

**See Also**

S3 methods: [summary.mcmc.sirt](#), [plot.mcmc.sirt](#)

**Examples**

```
## Not run:
#####
# EXAMPLE 1: Dataset Reading
#####
data(data.read)
dat <- data.read
I <- ncol(dat)

# set burnin and total number of iterations here (CHANGE THIS!)
burnin <- 200
iter <- 500

###
# Model 1: 1PNO model
mod1 <- mcmc.3pno.testlet( dat , est.slope=FALSE , est.guess=FALSE ,
                          burnin=burnin, iter=iter )
summary(mod1)
plot(mod1,ask=TRUE) # plot MCMC chains in coda style
plot(mod1,ask=TRUE , layout=2) # plot MCMC output in different layout

###
# Model 2: 3PNO model with Beta(5,17) prior for guessing parameters
mod2 <- mcmc.3pno.testlet( dat , guess.prior=c(5,17) ,
                          burnin=burnin, iter=iter )
summary(mod2)
```

```

****
# Model 3: Rasch (1PNO) testlet model
testlets <- substring( colnames(dat) , 1 , 1 )
mod3 <- mcmc.3pno.testlet( dat , testlets=testlets , est.slope=FALSE ,
                          est.guess=FALSE , burnin=burnin, iter=iter )
summary(mod3)

****
# Model 4: 3PNO testlet model with (almost) fixed guessing parameters .25
mod4 <- mcmc.3pno.testlet( dat , guess.prior=1000*c(25,75) , testlets=testlets ,
                          burnin=burnin, iter=iter )
summary(mod4)
plot(mod4, ask=TRUE, layout=2)

#####
# SIMULATED EXAMPLE 2: Simulated data according to the Rasch testlet model
#####
set.seed(678)

N <- 3000 # number of persons
I <- 4    # number of items per testlet
TT <- 3   # number of testlets

ITT <- I*TT
b <- round( rnorm( ITT , mean=0 , sd = 1 ) , 2 )
sd0 <- 1 # sd trait
sdt <- seq( 0 , 2 , len=TT ) # sd testlets
sdt <- sdt

# simulate theta
theta <- rnorm( N , sd = sd0 )
# simulate testlets
ut <- matrix(0,nrow=N , ncol=TT )
for (tt in 1:TT){ ut[,tt] <- rnorm( N , sd = sdt[tt] ) }
ut <- ut[ , rep(1:TT,each=I) ]
# calculate response probability
prob <- matrix( pnorm( theta + ut + matrix( b , nrow=N , ncol=ITT ,
byrow=TRUE ) ) , N, ITT)
Y <- (matrix( runif(N*ITT) , N , ITT) < prob )*1
colMeans(Y)

# define testlets
testlets <- rep(1:TT , each=I )

burnin <- 300
iter <- 1000

****
# Model 1: 1PNO model (without testlet structure)
mod1 <- mcmc.3pno.testlet( dat=Y , est.slope=FALSE , est.guess=FALSE ,
                          burnin=burnin, iter=iter , testlets= testlets )
summary(mod1)

```



```

summ1 <- mod1$summary.mcmcobj
# compare item parameters
cbind( b , summ1[ grep("b" , summ1$parameter ) , "Mean" ] )
# Testlet standard deviations
cbind( sdt , summ1[ grep("sigma\\.testlet" , summ1$parameter ) , "Mean" ] )

###
# Model 2: 1PNO model (without testlet structure)
mod2 <- mcmc.3pno.testlet( dat=Y , est.slope=TRUE , est.guess=FALSE ,
                          burnin=burnin, iter=iter , testlets= testlets )
summary(mod2)

summ2 <- mod2$summary.mcmcobj
# compare item parameters
cbind( b , summ2[ grep("b\[", summ2$parameter ) , "Mean" ] )
# item discriminations
cbind( sd0 , summ2[ grep("a\[", summ2$parameter ) , "Mean" ] )
# Testlet standard deviations
cbind( sdt , summ2[ grep("sigma\\.testlet" , summ2$parameter ) , "Mean" ] )

#####
# SIMULATED EXAMPLE 3: Simulated data according to the 2PNO testlet model
#####
set.seed(678)

N <- 3000   # number of persons
I <- 3      # number of items per testlet
TT <- 5     # number of testlets

ITT <- I*TT
b <- round( rnorm( ITT , mean=0 , sd = 1 ) , 2 )
a <- round( runif( ITT , 0.5 , 2 ) ,2)
sdt <- seq( 0 , 2 , len=TT ) # sd testlets
sdt <- sdt

# simulate theta
theta <- rnorm( N , sd = sd0 )
# simulate testlets
ut <- matrix(0,nrow=N , ncol=TT )
for (tt in 1:TT){ ut[,tt] <- rnorm( N , sd = sdt[tt] ) }
ut <- ut[ , rep(1:TT,each=I) ]
# calculate response probability
bM <- matrix( b , nrow=N , ncol=ITT , byrow=TRUE )
aM <- matrix( a , nrow=N , ncol=ITT , byrow=TRUE )
prob <- matrix( pnorm( aM*theta + ut + bM ) , N, ITT)
Y <- (matrix( runif(N*ITT) , N , ITT) < prob )*1
colMeans(Y)

# define testlets
testlets <- rep(1:TT , each=I )

burnin <- 500

```

```

iter <- 1500

****
# Model 1: 2PNO model
mod1 <- mcmc.3pno.testlet( dat=Y , est.slope=TRUE , est.guess=FALSE ,
                          burnin=burnin, iter=iter , testlets= testlets )
summary(mod1)

summ1 <- mod1$summary.mcmcobj
# compare item parameters
cbind( b , summ1[ grep("b" , summ1$parameter ) , "Mean" ] )
# item discriminations
cbind( a , summ1[ grep("a\[", summ1$parameter ) , "Mean" ] )
# Testlet standard deviations
cbind( sdt , summ1[ grep("sigma\\.testlet" , summ1$parameter ) , "Mean" ] )

## End(Not run)

```

---

mcmc.list.descriptives

*Computation of Descriptive Statistics for a mcmc.list Object*

---

## Description

Computation of descriptive statistics, Rhat convergence statistic and MAP for a `mcmc.list` object. The Rhat statistic is computed by splitting one Monte Carlo chain into three segments of equal length. The MAP is the mode estimate of the posterior distribution which is approximated by the mode of the kernel density estimate.

## Usage

```
mcmc.list.descriptives( mcmcobj , quantiles=c(.025,.05,.1,.5,.9,.95,.975) )
```

## Arguments

<code>mcmcobj</code>	Object of class <code>mcmc.list</code>
<code>quantiles</code>	Quantiles to be calculated for all parameters

## Value

A data frame with descriptive statistics for all parameters in the `mcmc.list` object.

## Author(s)

Alexander Robitzsch

## See Also

See [mcmclist2coda](#) for writing an object of class `mcmc.list` into a coda file (see also the `coda` package).

**Examples**

```

## Not run:
miceadds::library_install("coda")
miceadds::library_install("R2WinBUGS")

#####
# EXAMPLE 1: Logistic regression
#####

*****
# (1) simulate data
set.seed(8765)
N <- 500
x1 <- rnorm(N)
x2 <- rnorm(N)
y <- 1*( plogis( -.6 + .7*x1 + 1.1 *x2 ) > runif(N) )

*****
# (2) estimate logistic regression with glm
mod <- glm( y ~ x1 + x2 , family="binomial" )
summary(mod)

*****
# (3) estimate model with rcppbugs package
b <- rcppbugs::mcmc.normal(rnorm(3),mu=0,tau=0.0001)
y.hat <- rcppbugs::deterministic(function(x1,x2,b) { plogis( b[1] + b[2]*x1 + b[3]*x2 ) },
                                x1 , x2 , b)
y.lik <- rcppbugs::mcmc.bernoulli( y , p = y.hat, observed = TRUE)
m <- rcppbugs::create.model(b, y.hat, y.lik)

**** estimate model in rcppbugs; 5000 iterations, 1000 burnin iterations
ans <- rcppbugs::run.model(m, iterations=5000, burn=1000, adapt=1000, thin=5)
print(rcppbugs::get.ar(ans)) # get acceptance rate
print(apply(ans[["b"]],2,mean)) # get means of posterior

**** convert rcppbugs into mcmcobj object
mcmcobj <- data.frame( ans$b )
colnames(mcmcobj) <- paste0("b",1:3)
mcmcobj <- as.matrix(mcmcobj)
class(mcmcobj) <- "mcmc"
attr(mcmcobj, "mcpair") <- c( 1 , nrow(mcmcobj) , 1 )
mcmcobj <- as.mcmc.list( mcmcobj )

# plot results
plot(mcmcobj)

# summary
summ1 <- mcmc.list.descriptives( mcmcobj )
summ1

## End(Not run)

```

---

mcmclist2coda

*Write Coda File from an Object of Class mcmc.list*


---

### Description

This function writes a coda file from an object of class `mcmc.list`. Note that only first entry (i.e. one chain) will be processed.

### Usage

```
mcmclist2coda(mcmclist, name, coda.digits = 5)
```

### Arguments

<code>mcmclist</code>	An object of class <code>mcmc.list</code> .
<code>name</code>	Name of the coda file to be written
<code>coda.digits</code>	Number of digits after decimal in the coda file

### Value

The coda file and a corresponding index file are written into the working directory.

### Author(s)

Alexander Robitzsch

### Examples

```
## Not run:
#####
# EXAMPLE 1: MCMC estimation 2PNO dataset Reading
#####

data(data.read)
# estimate 2PNO with MCMC with 3000 iterations and 500 burn-in iterations
mod <- mcmc.2pno( dat=data.read , iter=3000 , burnin=500 )
# plot MCMC chains
plot( mod$mcmcobj , ask=TRUE )
# write sampled chains into codafile
mcmclist2coda( mod$mcmcobj , name = "dataread_2p1" )

## End(Not run)
```

---

md.pattern.sirt	<i>Response Pattern in a Binary Matrix</i>
-----------------	--

---

**Description**

Computes different statistics of the response pattern in a binary matrix.

**Usage**

```
md.pattern.sirt(dat)
```

**Arguments**

dat	A binary data matrix
-----	----------------------

**Value**

A list with following entries:

dat	Original dataset
dat.resp1	Indices for responses of 1's
dat.resp0	Indices for responses of 0's
resp_patt	Vector of response patterns
unique_resp_patt	Unique response patterns
unique_resp_patt_freq	Frequencies of unique response patterns
unique_resp_patt_firstobs	First observation in original dataset dat of a unique response pattern
freq1	Frequencies of 1's
freq0	Frequencies of 0's
dat.ordered	Dataset according to response patterns

**Author(s)**

Alexander Robitzsch

**See Also**

See also the `md.pattern` function in the **mice** package.

## Examples

```
#####
# SIMULATED EXAMPLE 1
#####
set.seed(7654)
N <- 21      # number of rows
I <- 4       # number of columns
dat <- matrix( 1*(runif(N*I) > .3 ) , N, I )
res <- md.pattern.sirt(dat)
# plot of response patterns
res$dat.ordered
image( z=t(res$dat.ordered) , y =1:N , x=1:I , xlab="Items" , ylab="Persons")
# 0's are yellow and 1's are red

#####
# EXAMPLE 2: Item response patterns for dataset data.read
#####

data(data.read)
dat <- data.read ; N <- nrow(dat) ; I <- ncol(dat)
# order items according to p values
dat <- dat[ , order(colMeans(dat , na.rm=TRUE )) ]
# analyzing response pattern
res <- md.pattern.sirt(dat)
res$dat.ordered
image( z=t(res$dat.ordered) , y =1:N , x=1:I , xlab="Items" , ylab="Persons")
```

---

mirt.specify.partable *Specify or modify a Parameter Table in mirt*

---

## Description

Specify or modify a parameter table in **mirt**.

## Usage

```
mirt.specify.partable(mirt.partable, parlist, verbose=TRUE)
```

## Arguments

mirt.partable	Parameter table in <b>mirt</b> package
parlist	List of parameters which are used for specification in the parameter table. See Examples.
verbose	An optional logical indicating whether the some warnings should be printed.

## Value

A modified parameter table

**Author(s)**

Alexander Robitzsch, Phil Chalmers

**Examples**

```
#####
# EXAMPLE 1: Modifying a parameter table for single group
#####

library(mirt)
data(LSAT7,package="mirt")
data <- mirt::expand.table(LSAT7)

mirt.partable <- mirt::mirt(data, 1, pars = "values")
colnames(mirt.partable)
## > colnames(mirt.partable) [1] 'group' 'item' 'class' 'name' 'parnum' 'value'
## 'lbound' 'ubound' 'est' 'prior.type' 'prior_1' 'prior_2'

# specify some values of item parameters
value <- data.frame(d = c(0.7, -1, NA), a1 = c(1, 1.2, 1.3), g = c(NA, 0.25, 0.25))
rownames(value) <- c("Item.1", "Item.4", "Item.3")

# fix some item paramters
est1 <- data.frame(d = c(TRUE, NA), a1 = c(FALSE, TRUE))
rownames(est1) <- c("Item.4", "Item.3")

# estimate all guessing parameters
est2 <- data.frame(g = rep(TRUE, 5))
rownames(est2) <- colnames(data)

# prior distributions
prior.type <- data.frame(g = rep("norm", 4))
rownames(prior.type) <- c("Item.1", "Item.2", "Item.4", "Item.5")
prior_1 <- data.frame(g = rep(-1.38, 4))
rownames(prior_1) <- c("Item.1", "Item.2", "Item.4", "Item.5")
prior_2 <- data.frame(g = rep(0.5, 4))
rownames(prior_2) <- c("Item.1", "Item.2", "Item.4", "Item.5")

# misspecify some entries
rownames(prior_2)[c(3,2)] <- c("A", "B")
rownames(est1)[2] <- c("B")

# define complete list with parameter specification
parlist <- list(value = value, est = est1, est = est2, prior.type = prior.type,
               prior_1 = prior_1, prior_2 = prior_2)

# modify parameter table
mirt.specify.partable(mirt.partable, parlist)
```

## Description

Some functions for wrapping with the **mirt** package.

## Usage

```
# extract coefficients
mirt.wrapper.coef(mirt.obj)

# extract posterior, likelihood, ...
mirt.wrapper.posterior(mirt.obj,weights=NULL)
## S3 method for class 'SingleGroupClass'
IRT.likelihood(object, ...)
## S3 method for class 'SingleGroupClass'
IRT.posterior(object, ...)

# S3 method for extracting item response functions
## S3 method for class 'SingleGroupClass'
IRT.irfprob(object, ...)

# compute factor scores
mirt.wrapper.fscores(mirt.obj,weights=NULL)

# convenience function for itemplot
mirt.wrapper.itemplot( mirt.obj , ask=TRUE , ...)
```

## Arguments

mirt.obj	A fitted model in <b>mirt</b> package
object	A fitted object in <b>mirt</b> package of class <code>ConfirmatoryClass</code> or <code>ExploratoryClass</code> .
weights	Optional vector of student weights
ask	Optional logical indicating whether each new plot should be confirmed.
...	Further arguments to be passed.

## Details

The function `mirt.wrapper.coef` collects all item parameters in a data frame.

The function `mirt.wrapper.posterior` extracts the individual likelihood, individual likelihood and expected counts. This function does not yet cover the case of multiple groups.

The function `mirt.wrapper.fscores` computes factor scores EAP, MAP and MLE. The factor scores are computed on the discrete grid of latent traits (contrary to the computation in `mirt`) as specified in `mirt.obj@Theta`. This function does also not work for multiple groups.

The function `mirt.wrapper.itemplot` displays all item plots after each other.



**Value**

Function `mirt.wrapper.coef` – List with entries

`coef`                Data frame with item parameters  
`GroupPars`        Data frame or list with distribution parameters

Function `mirt.wrapper.posterior` – List with entries

`theta.k`            Grid of theta points  
`pi.k`                Trait distribution on `theta.k`  
`f.yi.qk`            Individual likelihood  
`f.qk.yi`            Individual posterior  
`n.ik`                Expected counts  
`data`                Used dataset

Function `mirt.wrapper.fscores` – List with entries

`person`            Data frame with person parameter estimates (factor scores) EAP, MAP and MLE for all dimensions.  
`EAP.rel`            EAP reliabilities

**Examples for the mirt Package**

1. Latent class analysis ([data.read](#), Model 7)
2. Mixed Rasch model ([data.read](#), Model 8)
3. Located unidimensional and multidimensional latent class models / Multidimensional latent class IRT models ([data.read](#), Model 12; [rasch.mirtlc](#), Example 4)
4. Multidimensional IRT model with discrete latent traits ([data.read](#), Model 13)
5. DINA model ([data.read](#), Model 14; [data.dcm](#), **CDM**, Model 1m)
6. Unidimensional IRT model with non-normal distribution ([data.read](#), Model 15)
7. Grade of membership model ([gom.em](#), Example 2)
8. Rasch copula model ([rasch.copula2](#), Example 5)
9. Additive GDINA model ([data.dcm](#), **CDM**, Model 6m)
10. Longitudinal Rasch model ([data.long](#), Model 3)
11. Normally distributed residuals ([data.big5](#), Example 1, Model 5)
12. Nedelsky model ([nedelsky.irf](#), Examples 1, 2)
13. Beta item response model ([brm.irf](#), Example 1)

**Author(s)**

Alexander Robitzsch

**See Also**

See the **mirt** package on CRAN <http://cran.r-project.org/package=mirt> and on GitHub <https://github.com/philchalmers/mirt>.

See <https://groups.google.com/forum/#!forum/mirt-package> for discussion about the **mirt** package.

See for the main estimation functions in **mirt**: `mirt::mirt`, `mirt::multipleGroup` and `mirt::bfactor`.

See `mirt::coef-method` for extracting coefficients.

See `mirt::mod2values` for collecting parameter values in a mirt parameter table.

See `lavaan2mirt` for converting lavaan syntax to mirt syntax.

See `tam2mirt` for converting fitted tam models into mirt objects.

See also `CDM::IRT.likelihood`, `CDM::IRT.posterior` and `CDM::IRT.irfprob` for general extractor functions.

**Examples**

```
## Not run:
# A development version can be installed from GitHub
if (FALSE){ # default is set to FALSE, use the installed version
  library(devtools)
  devtools::install_github("philchalmers/mirt")
}
# now, load mirt
library(mirt)

#####
# EXAMPLE 1: Extracting item parameters and posterior LSAT data
#####

data(LSAT7, package="mirt")
data <- mirt::expand.table(LSAT7)

###* Model 1: 3PL model for item 5 only, other items 2PL
mod1 <- mirt::mirt(data, 1, itemtype=c("2PL", "2PL", "2PL", "2PL", "3PL") , verbose=TRUE)
print(mod1)
summary(mod1)
# extracting coefficients
coef(mod1)
mirt.wrapper.coef(mod1)$coef
# extract parameter values in mirt
mirt::mod2values(mod1)
# extract posterior
post1 <- mirt.wrapper.posterior(mod1)
# extract item response functions
probs1 <- IRT.irfprob( mod1 )
str(probs1)
# extract individual likelihood
likemod1 <- IRT.likelihood( mod1 )
str(likemod1)
```

```

# extract individual posterior
postmod1 <- IRT.posterior( mod1 )
str(postmod1)

### Model 2: Confirmatory model with two factors
cmodel <- mirt::mirt.model("
  F1 = 1,4,5
  F2 = 2,3
")
mod2 <- mirt::mirt(data, cmodel , verbose=TRUE)
print(mod2)
summary(mod2)
# extract coefficients
coef(mod2)
mirt.wrapper.coef(mod2)$coef
# extract posterior
post2 <- mirt.wrapper.posterior(mod2)

#####
# EXAMPLE 2: Extracting item parameters and posterior for differering
#           number of response catagories | Dataset Science
#####

data(Science,package="mirt")
library(psych)
psych::describe(Science)

# modify dataset
dat <- Science
dat[ dat[,1] > 3 ,1] <- 3
psych::describe(dat)

# estimate generalized partial credit model
mod1 <- mirt::mirt(dat, 1, itemtype="gpcm")
print(mod1)
# extract coefficients
coef(mod1)
mirt.wrapper.coef(mod1)$coef
# extract posterior
post1 <- mirt.wrapper.posterior(mod1)

#####
# SIMULATED EXAMPLE 3: Multiple group model; simulated dataset from
#           mirt package
#####

### simulate data (copy from the multipleGroup manual site in mirt package)
set.seed(1234)
a <- matrix(c(abs(rnorm(5,1,.3)), rep(0,15),abs(rnorm(5,1,.3))),
            rep(0,15),abs(rnorm(5,1,.3))), 15, 3)
d <- matrix(rnorm(15,0,.7),ncol=1)
mu <- c(-.4, -.7, .1)
sigma <- matrix(c(1.21,.297,1.232,.297,.81,.252,1.232,.252,1.96),3,3)

```

```

itemtype <- rep("dich", nrow(a))
N <- 1000
dataset1 <- mirt::simdata(a, d, N, itemtype)
dataset2 <- mirt::simdata(a, d, N, itemtype, mu = mu, sigma = sigma)
dat <- rbind(dataset1, dataset2)
group <- c(rep("D1", N), rep("D2", N))

#group models
model <- mirt::mirt.model("
  F1 = 1-5
  F2 = 6-10
  F3 = 11-15
  ")

# separate analysis
mod_configural <- multipleGroup(dat, model, group = group , verbose=TRUE)
mirt.wrapper.coef(mod_configural)

# equal slopes (metric invariance)
mod_metric <- multipleGroup(dat, model, group = group, invariance=c("slopes") ,
  verbose=TRUE)
mirt.wrapper.coef(mod_metric)

# equal slopes and intercepts (scalar invariance)
mod_scalar <- multipleGroup(dat, model, group = group,
  invariance=c("slopes", "intercepts", "free_means", "free_varcov"), verbose=TRUE)
mirt.wrapper.coef(mod_scalar)

# full constraint
mod_fullconstrain <- multipleGroup(dat, model, group = group,
  invariance=c("slopes", "intercepts") , verbose=TRUE )
mirt.wrapper.coef(mod_fullconstrain)

#####
# EXAMPLE 4: Nonlinear item response model
#####

data(data.read)
dat <- data.read
# specify mirt model with some interactions
mirtmodel <- mirt.model("
  A = 1-4
  B = 5-8
  C = 9-12
  (A*B) = 4,8
  (C*C) = 9
  (A*B*C) = 12
  ")
# estimate model
res <- mirt::mirt( dat , mirtmodel , verbose=TRUE , technical=list(NCYCLES=3) )
# look at estimated parameters
mirt.wrapper.coef(res)
coef(res)

```

```

mirt::mod2values(res)
# model specification
res@model

#####
# EXAMPLE 5: Extracting factor scores
#####

data(data.read)
dat <- data.read
# define lavaan model and convert syntax to mirt
lavamodel <- "
  A =~ a*A1+a*A2+1.3*A3+A4      # set loading of A3 to 1.3
  B =~ B1+1*B2+b3*B3+B4
  C =~ c*C1+C2+c*C3+C4
  A1 | da*t1
  A3 | da*t1
  C4 | dg*t1
  B1 | 0*t1
  B3 | -1.4*t1                 # fix item threshold of B3 to -1.4
  A ~~ B                       # estimate covariance between A and B
  A ~~ .6 * C                  # fix covariance to .6
  B ~~ B                       # estimate variance of B
  A ~ .5*1                     # set mean of A to .5
  B ~ 1                         # estimate mean of B
"

res <- lavaan2mirt( dat , lavamodel , verbose=TRUE , technical=list(NCYCLES=3) )
# estimated coefficients
mirt.wrapper.coef(res$mirt)
# extract factor scores
fres <- mirt.wrapper.fscores(res$mirt)
# look at factor scores
head( round(fres$person,2))
##      case      M EAP.Var1 SE.EAP.Var1 EAP.Var2 SE.EAP.Var2 EAP.Var3 SE.EAP.Var3 MLE.Var1
##  1  1 0.92      1.26      0.67      1.61      0.60      0.05      0.69      2.65
##  2  2 0.58      0.06      0.59      1.14      0.55     -0.80      0.56      0.00
##  3  3 0.83      0.86      0.66      1.15      0.55      0.48      0.74      0.53
##  4  4 1.00      1.52      0.67      1.57      0.60      0.73      0.76      2.65
##  5  5 0.50     -0.13      0.58      0.85      0.48     -0.82      0.55     -0.53
##  6  6 0.75      0.41      0.63      1.09      0.54      0.27      0.71      0.00
##      MLE.Var2 MLE.Var3 MAP.Var1 MAP.Var2 MAP.Var3
##  1      2.65     -0.53      1.06      1.59      0.00
##  2      1.06     -1.06      0.00      1.06     -1.06
##  3      1.06      2.65      1.06      1.06      0.53
##  4      2.65      2.65      1.59      1.59      0.53
##  5      0.53     -1.06     -0.53      0.53     -1.06
##  6      1.06      2.65      0.53      1.06      0.00

# EAP reliabilities
round(fres$EAP.rel,3)
##      Var1 Var2 Var3
##  0.574 0.452 0.541

## End(Not run)

```

---

mle.pcm.group	<i>Maximum Likelihood Estimation of Person or Group Parameters in the Generalized Partial Credit Model</i>
---------------	--

---

### Description

This function estimates person or group parameters in the partial credit model (see Details).

### Usage

```
mle.pcm.group(dat, b, a = rep(1, ncol(dat)), group = NULL,
              pid = NULL, adj_eps = 0.3, conv = 1e-04, maxiter = 30)
```

### Arguments

dat	A numeric $N \times I$ matrix
b	Matrix with item thresholds
a	Vector of item slopes
group	Vector of group identifiers
pid	Vector of person identifiers
adj_eps	Numeric value which is used in $\varepsilon$ adjustment of the likelihood. A value of zero (or a very small $\varepsilon > 0$ ) corresponds to the usual maximum likelihood estimate.
conv	Convergence criterion
maxiter	Maximum number of iterations

### Details

It is assumed that the generalized partial credit model holds. In case one estimates a person parameter  $\theta_p$ , the log-likelihood is maximized and the following estimating equation results: (see Penfield & Bergeron, 2005):

$$0 = (\log L)' = \sum_i a_i \cdot [\tilde{x}_{pi} - E(X_{pi}|\theta_p)]$$

where  $E(X_{pi}|\theta_p)$  denotes the expected item response conditionally on  $\theta_p$ .

With the method of  $\varepsilon$ -adjustment (Bertoli-Barsotti & Punzo, 2012; Bertoli-Barsotti, Lando & Punzo, 2014), the observed item responses  $x_{pi}$  are transformed such that no perfect scores arise and bias is reduced. If  $S_p$  is the sum score of person  $p$  and  $M_p$  the maximum score of this person, then the transformed sum scores  $\tilde{S}_p$  are

$$\tilde{S}_p = \varepsilon + \frac{M_p - 2\varepsilon}{M_p} S_p$$

However, the adjustment is directly conducted on item responses to also handle the case of the generalized partial credit model with item slope parameters different from 1.

In case one estimates a group parameter  $\theta_g$ , the following estimating equation is used:

$$0 = (\log L)' = \sum_p \sum_i a_i \cdot [\tilde{x}_{pgi} - E(X_{pgi}|\theta_g)]$$

where persons  $p$  are nested within a group  $g$ . The  $\varepsilon$ -adjustment is then performed at the group level, not at the individual level.

### Value

A list with following entries:

person	Data frame with person or group parameters
data_adjeps	Modified dataset according to the $\varepsilon$ adjustment.

### Author(s)

Alexander Robitzsch

### References

Bertoli-Barsotti, L., & Punzo, A. (2012). Comparison of two bias reduction techniques for the Rasch model. *Electronic Journal of Applied Statistical Analysis*, **5**, 360-366.

Bertoli-Barsotti, L., Lando, T., & Punzo, A. (2014). Estimating a Rasch Model via fuzzy empirical probability functions. In D. Vicari, A. Okada, G. Ragozini & C. Weihs (Eds.). *Analysis and Modeling of Complex Data in Behavioural and Social Sciences*, Springer.

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.

### Examples

```
## Not run:
#####
# EXAMPLE 1: Estimation of a group parameter for only one item per group
#####

data(data.si01)
dat <- data.si01
# item parameter estimation (partial credit model) in TAM
library(TAM)
mod <- TAM::tam.mml( dat[,2:3] , irtmodel="PCM")
# extract item difficulties
b <- matrix( mod$xisi$xisi , nrow=2 , byrow=TRUE )
# groupwise estimation
res1 <- mle.pcm.group( dat[,2:3] , b=b , group=dat$idgroup )
# individual estimation
res2 <- mle.pcm.group( dat[,2:3] , b=b )

#####
# EXAMPLE 2: Data Reading data.read
#####

data(data.read)
# estimate Rasch model
mod <- rasch.mml2( data.read )
score <- rowSums( data.read )
```

```

data.read <- data.read[ order(score) , ]
score <- score[ order(score) ]
# compare different epsilon-adjustments
res30 <- mle.pcm.group( data.read , b = matrix( mod$item$b , 12 , 1 ) ,
  adj_eps=.3 )$person
res10 <- mle.pcm.group( data.read , b = matrix( mod$item$b , 12 , 1 ) ,
  adj_eps=.1 )$person
res05 <- mle.pcm.group( data.read , b = matrix( mod$item$b , 12 , 1 ) ,
  adj_eps=.05 )$person
# plot different scorings
plot( score , res05$theta , type="l" , xlab="Raw score" , ylab=expression(theta[epsilon]),
  main="Scoring with different epsilon-adjustments")
lines( score , res10$theta , col=2 , lty=2 )
lines( score , res30$theta , col=4 , lty=3 )

## End(Not run)

```

---

modelfit.sirt

*Assessing Model Fit and Local Dependence by Comparing Observed  
and Expected Item Pair Correlations*


---

## Description

This function computes several measures of absolute model fit and local dependence indices for dichotomous item responses which are based on comparing observed and expected frequencies of item pairs (Chen, de la Torre & Zhang, 2013; see [modelfit.cor](http://modelfit.cor) for more details).

## Usage

```
modelfit.sirt(object)
```

```
modelfit.cor.poly( data , probs , theta.k , f.qk.yi)
```

## Arguments

object	An object generated by <a href="#">rasch.mml2</a> , <a href="#">rasch.mirtlc</a> , <a href="#">rasch.pm13</a> ( <a href="#">rasch.pm12</a> ), <a href="#">smirt</a> , <a href="#">R2noharm</a> , <a href="#">noharm.sirt</a> , <a href="#">gom.em</a> , <a href="#">TAM::tam.mml</a> , <a href="#">TAM::tam.mml.2pl</a> , <a href="#">TAM::tam.fa</a> , <a href="#">mirt::mirt</a>
data	Dataset with polytomous item responses
probs	Item response probabilities at grid theta.k
theta.k	Grid of theta vector
f.qk.yi	Individual posterior



**Value**

A list with following entries:

modelfit	<p>Model fit statistics:</p> <p>MADcor: mean of absolute deviations in observed and expected correlations (DiBello et al., 2007)</p> <p>SRMSR: standardized mean square root of squared residuals (Maydeu-Olivares, 2013; Maydeu-Olivares &amp; Joe, 2014)</p> <p>MX2: Mean of <math>\chi^2</math> statistics of all item pairs (Chen &amp; Thissen, 1997)</p> <p>MADRESIDCOV: Mean of absolute deviations of residual covariances (McDonald &amp; Mok, 1995)</p> <p>MADQ3: Mean of absolute values of <math>Q_3</math> statistic (Yen, 1984)</p> <p>MADaQ3: Mean of absolute values of centered <math>Q_3</math> statistic</p>
itempairs	Fit of every item pair

**Note**

The function `modelfit.cor.poly` is just a wrapper to `TAM::tam.modelfit` in the **TAM** package.

**Author(s)**

Alexander Robitzsch

**References**

- Chen, W., & Thissen, D. (1997). Local dependence indexes for item pairs using item response theory. *Journal of Educational and Behavioral Statistics*, **22**, 265-289.
- DiBello, L. V., Roussos, L. A., & Stout, W. F. (2007) Review of cognitively diagnostic assessment and a summary of psychometric models. In C. R. Rao and S. Sinharay (Eds.), *Handbook of Statistics*, Vol. 26 (pp. 979–1030). Amsterdam: Elsevier.
- Maydeu-Olivares, A. (2013). Goodness-of-fit assessment of item response theory models (with discussion). *Measurement: Interdisciplinary Research and Perspectives*, **11**, 71-137.
- Maydeu-Olivares, A., & Joe, H. (2014). Assessing approximate fit in categorical data analysis. *Multivariate Behavioral Research*, **49**, 305-328.
- McDonald, R. P., & Mok, M. M.-C. (1995). Goodness of fit in item response models. *Multivariate Behavioral Research*, **30**, 23-40.
- Yen, W. M. (1984). Effects of local item dependence on the fit and equating performance of the three-parameter logistic model. *Applied Psychological Measurement*, **8**, 125-145.

**See Also**

Supported classes: `rasch.mml2`, `rasch.mirtlc`, `rasch.pml3` (`rasch.pml2`), `smirt`, `R2noharm`, `noharm.sirt`, `gom.em`, `TAM::tam.mml`, `TAM::tam.mml.2pl`, `TAM::tam.fa`, `mirt::mirt`

For more details on fit statistics of this function see `CDM::modelfit.cor`.

## Examples

```
## Not run:
#####
# EXAMPLE 1: Reading data
#####
data(data.read)
dat <- data.read
I <- ncol(dat)

#### Model 1: Rasch model
mod1 <- rasch.mml2(dat)
fmod1 <- modelfit.sirt( mod1 )
summary(fmod1)

#### Model 1b: Rasch model in TAM package
library(TAM)
mod1b <- TAM::tam.mml(dat)
fmod1b <- modelfit.sirt( mod1b )
summary(fmod1b)

#### Model 2: Rasch model with smoothed distribution
mod2 <- rasch.mml2( dat , distribution.trait="smooth3" )
fmod2 <- modelfit.sirt( mod2 )
summary(fmod2 )

#### Model 3: 2PL model
mod3 <- rasch.mml2( dat , distribution.trait="normal" , est.a=1:I )
fmod3 <- modelfit.sirt( mod3 )
summary(fmod3 )

#### Model 3: 2PL model in TAM package
mod3b <- TAM::tam.mml.2pl( dat )
fmod3b <- modelfit.sirt(mod3b)
summary(fmod3b)
# model fit in TAM package
tmod3b <- TAM::tam.modelfit(mod3b)
summary(tmod3b)
# model fit in mirt package
library(mirt)
mmod3b <- tam2mirt(mod3b) # convert to mirt object
mirt::M2(mmod3b$mirt) # global fit statistic
mirt::residuals( mmod3b$mirt , type="LD" ) # local dependence statistics

#### Model 4: 3PL model with equal guessing parameter
mod4 <- TAM::rasch.mml2( dat , distribution.trait="smooth3", est.a=1:I, est.c=rep(1,I) )
fmod4 <- modelfit.sirt( mod4 )
summary(fmod4 )

#### Model 5: Latent class model with 2 classes
mod5 <- rasch.mirtlc( dat , Nclasses=2 )
fmod5 <- modelfit.sirt( mod5 )
summary(fmod5 )
```

```

**** Model 6: Rasch latent class model with 3 classes
mod6 <- rasch.mirtlc( dat , Nclasses=3 , modeltype="MLC1", mmliter=100)
fmod6 <- modelfit.sirt( mod6 )
summary(fmod6 )

**** Model 7: PML estimation
mod7 <- rasch.pml3( dat )
fmod7 <- modelfit.sirt( mod7 )
summary(fmod7 )

**** Model 8: PML estimation
# Modelling error correlations:
# joint residual correlations for each item cluster
error.corr <- diag(1,ncol(dat))
itemcluster <- rep( 1:4 ,each=3 )
for ( ii in 1:3){
  ind.ii <- which( itemcluster == ii )
  error.corr[ ind.ii , ind.ii ] <- ii
}
mod8 <- rasch.pml3( dat , error.corr = error.corr )
fmod8 <- modelfit.sirt( mod8 )
summary(fmod8 )

**** Model 9: 1PL in smirt
Qmatrix <- matrix( 1 , nrow=I , ncol=1 )
mod9 <- smirt( dat , Qmatrix=Qmatrix )
fmod9 <- modelfit.sirt( mod9 )
summary(fmod9 )

**** Model 10: 3-dimensional Rasch model in NOHARM
noharm.path <- "c:/NOHARM"
Q <- matrix( 0 , nrow=12 , ncol=3 )
Q[ cbind(1:12 , rep(1:3,each=4) ) ] <- 1
rownames(Q) <- colnames(dat)
colnames(Q) <- c("A","B","C")
# covariance matrix
P.pattern <- matrix( 1 , ncol=3 , nrow=3 )
P.init <- 0.8+0*P.pattern
diag(P.init) <- 1
# loading matrix
F.pattern <- 0*Q
F.init <- Q
# estimate model
mod10 <- R2noharm( dat = dat , model.type="CFA" , F.pattern = F.pattern ,
  F.init = F.init , P.pattern = P.pattern , P.init = P.init ,
  writename = "ex4e" , noharm.path = noharm.path , dec = "," )
fmod10 <- modelfit.sirt( mod10 )
summary(fmod10)

**** Model 11: Rasch model in mirt package
library(mirt)
mod11 <- mirt::mirt(dat , 1, itemtype="Rasch",verbose=TRUE)

```

```
fmod11 <- modelfit.sirt( mod11 )
summary(fmod11)
# model fit in mirt package
mirt::M2(mod11)
mirt::residuals(mod11)

## End(Not run)
```

---

monoreg.rowwise

*Monotone Regression for Rows or Columns in a Matrix*

---

### Description

Monotone (isotone) regression for rows (`monoreg.rowwise`) or columns (`monoreg.colwise`) in a matrix.

### Usage

```
monoreg.rowwise(yM, wM)
```

```
monoreg.colwise(yM, wM)
```

### Arguments

`yM` Matrix with dependent variable for the regression. Values are assumed to be sorted.

`wM` Matrix with weights for every entry in the `yM` matrix.

### Value

Matrix with fitted values

### Note

This function is used for fitting the ISOP model (see [isop.dich](#)).

### Author(s)

Alexander Robitzsch

The `monoreg` function from the **fdrtool** package is simply extended to handle matrix input.

### See Also

See also the `monoreg` function from the **fdrtool** package.

**Examples**

```

y <- c(22.5 , 23.33 , 20.83 , 24.25 )
w <- c( 3,3,3,2)
# define matrix input
yM <- matrix( 0 , nrow=2 , ncol=4 )
wM <- yM
yM[1,] <- yM[2,] <- y
wM[1,] <- w
wM[2,] <- c(1,3,4, 3 )

# fit rowwise monotone regression
monoreg.rowwise( yM , wM )
# compare results with monoreg function from fdrtool package
## Not run:
miceadds::library_install("fdrtool")
fdrtool::monoreg(x=yM[1,] , w=wM[1,])$yf
fdrtool::monoreg(x=yM[2,] , w=wM[2,])$yf

## End(Not run)

```

---

nedelsky-methods

*Functions for the Nedelsky Model*


---

**Description**

Functions for simulating and estimating the Nedelsky model (Bechger et al., 2003, 2005). `nedelsky.sim` can be used for simulating the model, `nedelsky.irf` computes the item response function and can be used for example when estimating the Nedelsky model in the **mirt** package.

**Usage**

```

# simulating the Nedelsky model
nedelsky.sim(theta, b, a = NULL, tau = NULL)

# creating latent responses of the Nedelsky model
nedelsky.latresp(K)

# computing the item response function of the Nedelsky model
nedelsky.irf(Theta, K, b, a, tau, combis, thdim = 1)

```

**Arguments**

theta	Unidimensional ability (theta)
b	Matrix of category difficulties
a	Vector of item discriminations
tau	Category attractivity parameters $\tau$ (see Bechger et al., 2005)
K	(Maximum) Number of distractors of the used multiple choice items

Theta	Theta vector. Note that the Nedelsky model can be only specified as models with between item dimensionality (defined in thdim).
combis	Latent response classes as produced by nedelsky.latresp.
thdim	Theta dimension at which the item loads

### Details

Assume that for item  $i$  there exists  $K + 1$  categories  $0, 1, \dots, K$ . The category 0 denotes the correct alternative. The Nedelsky model assumes that a respondent eliminates all distractors which are thought to be incorrect and guesses the solution from the remaining alternatives. This means, that for item  $i$ ,  $K$  latent variables  $S_{ik}$  are defined which indicate whether alternative  $k$  has been correctly identified as a distractor. By definition, the correct alternative is never been judged as wrong by the respondent.

Formally, the Nedelsky model assumes a 2PL model for eliminating each of the distractors

$$P(S_{ik} = 1|\theta) = \text{invlogit}[a_i(\theta - b_{ik})]$$

where  $\theta$  is the person ability and  $b_{ik}$  are distractor difficulties.

The guessing process of the Nedelsky model is defined as

$$P(X_i = j|\theta, S_{i1}, \dots, S_{iK}) = \frac{(1 - S_{ij})\tau_{ij}}{\sum_{k=0}^K [(1 - S_{ik})\tau_{ik}]}$$

where  $\tau_{ij}$  are attractivity parameters of alternative  $j$ . By definition  $\tau_{i0}$  is set to 1. By default, all attractivity parameters are set to 1.

### Author(s)

Alexander Robitzsch

### References

Bechger, T. M., Maris, G., Verstralen, H. H. F. M., & Verhelst, N. D. (2003). *The Nedelsky model for multiple-choice items*. CITO Research Report, 2003-5.

Bechger, T. M., Maris, G., Verstralen, H. H. F. M., & Verhelst, N. D. (2005). The Nedelsky model for multiple-choice items. In L. van der Ark, M. Croon, & Sijtsma, K. (Eds.). *New developments in categorical data analysis for the social and behavioral sciences*, pp. 187-206. Mahwah, Lawrence Erlbaum.

### Examples

```
## Not run:
#####
# SIMULATED EXAMPLE 1: Simulated data according to the Nedelsky model
#####

*** simulate data
set.seed(123)
I <- 20      # number of items
```

```

b <- matrix(NA,I,ncol=3)
b[,1] <- -.5 + runif( I , -.75 , .75 )
b[,2] <- -1.5 + runif( I , -.75 , .75 )
b[,3] <- -2.5 + runif( I , -.75 , .75 )
K <- 3          # number of distractors
N <- 2000       # number of persons
# apply simulation function
dat <- nedelsky.sim( theta = rnorm(N,sd=1.2) , b=b )

*** latent response patterns
K <- 3
combis <- nedelsky.latresp(K=3)

*** defining the Nedelky item response function for estimation in mirt
par <- c( 3 , rep(-1,K) , 1 , rep(1,K+1) ,1)
names(par) <- c("K" , paste0("b",1:K) , "a" , paste0("tau" , 0:K) ,"thdim")
est <- c( FALSE , rep(TRUE,K) , rep(FALSE , K+1 + 2 ) )
names(est) <- names(par)
nedelsky.icc <- function( par , Theta , ncat ){
  K <- par[1]
  b <- par[ 1:K + 1]
  a <- par[ K+2]
  tau <- par[1:(K+1) + (K+2) ]
  thdim <- par[ K+2+K+1 +1 ]
  probs <- nedelsky.irf( Theta , K=K , b=b , a=a , tau=tau , combis ,
                        thdim=thdim )$probs
  return(probs)
}
name <- "nedelsky"
# create item response function
nedelsky.itemfct <- mirt::createItem(name, par=par, est=est, P=nedelsky.icc)

*** define model in mirt
mirtmodel <- mirt::mirt.model("
  F1 = 1-2θ
  COV = F1*F1
  # define some prior distributions
  PRIOR = (1-2θ,b1 ,norm,-1,2),(1-2θ,b2 ,norm,-1,2),
          (1-2θ,b3 ,norm,-1,2)
" )

itemtype <- rep("nedelsky" , I )
customItems <- list("nedelsky"= nedelsky.itemfct)
# define parameters to be estimated
mod1.pars <- mirt::mirt(dat, mirtmodel , itemtype=itemtype ,
                      customItems=customItems, pars = "values")
# estimate model
mod1 <- mirt::mirt(dat,mirtmodel , itemtype=itemtype , customItems=customItems,
                  pars = mod1.pars , verbose=TRUE )
# model summaries
print(mod1)
summary(mod1)
mirt.wrapper.coef( mod1 )$coef

```

```

mirt.wrapper.itemplot(mod1 ,ask=TRUE)

#####
# EXAMPLE 2: Multiple choice dataset data.si06
#####

data(data.si06)
dat <- data.si06

###* create latent responses
combis <- nedelsky.latresp(K)
I <- ncol(dat)
###* define item response function
K <- 3
par <- c( 3 , rep(-1,K) , 1 , rep(1,K+1) ,1)
names(par) <- c("K" , paste0("b",1:K) , "a" , paste0("tau" , 0:K) ,"thdim")
est <- c( FALSE , rep(TRUE,K) , rep(FALSE , K+1 + 2 ) )
names(est) <- names(par)
nedelsky.icc <- function( par , Theta , ncat ){
  K <- par[1]
  b <- par[ 1:K + 1]
  a <- par[ K+2]
  tau <- par[1:(K+1) + (K+2) ]
  thdim <- par[ K+2+K+1 +1 ]
  probs <- nedelsky.irf( Theta , K=K , b=b , a=a , tau=tau , combis ,
    thdim=thdim )$probs
  return(probs)
}
name <- "nedelsky"
# create item response function
nedelsky.itemfct <- mirt::createItem(name, par=par, est=est, P=nedelsky.icc)

###* define model in mirt
mirtmodel <- mirt::mirt.model("
  F1 = 1-14
  COV = F1*F1
  PRIOR = (1-14,b1 ,norm,-1,2),(1-14,b2 ,norm,-1,2),
    (1-14,b3 ,norm,-1,2)
" )

itemtype <- rep("nedelsky" , I )
customItems <- list("nedelsky"= nedelsky.itemfct)
# define parameters to be estimated
mod1.pars <- mirt::mirt(dat, mirtmodel , itemtype=itemtype ,
  customItems=customItems, pars = "values")

###* estimate model
mod1 <- mirt::mirt(dat,mirtmodel , itemtype=itemtype , customItems=customItems,
  pars = mod1.pars , verbose=TRUE )

###* summaries
print(mod1)
summary(mod1)
mirt.wrapper.coef( mod1 )$coef

```



```
mirt.wrapper.itemplot(mod1 ,ask=TRUE)

## End(Not run)
```

noharm.sirt

*NOHARM Model in R***Description**

The function is an R implementation of the normal ogive harmonic analysis robust method (the NOHARM model; McDonald, 1997). Exploratory and confirmatory multidimensional item response models for dichotomous data using the probit link function can be estimated. Lower asymptotes (guessing parameters) and upper asymptotes (One minus slipping parameters) can be provided as fixed values.

**Usage**

```
noharm.sirt(dat, weights = NULL, Fval = NULL, Fpatt = NULL, Pval = NULL,
  Ppatt = NULL, Psival = NULL, Psipatt = NULL, dimensions = NULL,
  lower = rep(0, ncol(dat)), upper = rep(1, ncol(dat)), wgtm = NULL, modesttype=1,
  pos.loading=FALSE , pos.variance = FALSE , pos.residcorr = FALSE ,
  maxiter = 1000, conv = 10^(-6), increment.factor = 1.01)
```

```
## S3 method for class 'noharm.sirt'
summary(object, logfile=NULL , ...)
```

**Arguments**

dat	Matrix of dichotomous item responses. This matrix contain missing data (indicated by NA) but missingness is assumed to be missing completely at random (MCAR).
weights	Optional vector of student weights.
Fval	Initial or fixed values of the loading matrix $F$ .
Fpatt	Pattern matrix of the loading matrix $F$ . If elements should be estimated, then an entry of 1 must be included in the pattern matrix.
Pval	Initial or fixed values for the covariance matrix $P$ .
Ppatt	Pattern matrix for the covariance matrix $P$ .
Psival	Initial or fixed values for the matrix of residual correlations $\Psi$ .
Psipatt	Pattern matrix for the matrix of residual correlations $\Psi$ .
dimensions	Number of dimensions if an exploratory factor analysis should be estimated.
lower	Fixed vector of lower asymptotes $c_i$ .
upper	Fixed vector of upper asymptotes $d_i$ .
wgtm	Matrix with positive entries which indicates by a positive entry which item pairs should be used for estimation.

modesttype	Estimation type. modesttype=1 refers to the NOHARM approximation, while modesttype=2 refers to the estimation based on tetrachoric correlations.
pos.loading	An optional logical indicating whether all entries in the loading matrix $\mathbf{F}$ should be positive
pos.variance	An optional logical indicating whether all variances (i.e. diagonal entries in $\mathbf{P}$ ) should be positive
pos.residcorr	An optional logical indicating whether all entries in the matrix of residual correlations $\Psi$ should be positive
maxiter	Maximum number of iterations
conv	Convergence criterion for parameters
increment.factor	Numeric value larger than 1 which controls the size of increments in increasing iteration numbers. With a larger value, the increments are heavier penalized.
object	Object of class noharm.sirt
logfile	String indicating a file name for summary.
...	Further arguments to be passed. For plot.noharm.sirt, additional arguments for <code>semPlot::semPaths</code> (from <code>semPlot</code> package) can be specified.

## Details

The NOHARM item response model follows the response equation

$$P(X_{pi} = 1 | \boldsymbol{\theta}_p) = c_i + (d_i - c_i) \Phi(f_{i0} + f_{i1}\theta_{p1} + \dots + f_{iD}\theta_{pD})$$

for item responses  $X_{pi}$  of person  $p$  on item  $i$ ,  $\mathbf{F} = (f_{id})$  is a loading matrix and  $\mathbf{P}$  the covariance matrix of  $\boldsymbol{\theta}_p$ . The lower asymptotes  $c_i$  and upper asymptotes  $d_i$  must be provided as fixed values. The response equation can be equivalently written by introducing a latent continuous item response  $X_{pi}^*$

$$X_{pi}^* = f_{i0} + f_{i1}\theta_{p1} + \dots + f_{iD}\theta_{pD} + e_{pi}$$

with a standard normally distributed residuals  $e_{pi}$ . These residuals have a correlation matrix  $\Psi$  with ones in the diagonal. In this R implementation of the NOHARM model, correlations between residuals are allowed.

See References for more details about estimation.

## Value

A list. The most important entries are

tanaka	Tanaka fit statistic
rmsr	RMSR fit statistic
N.itempair	Sample size per item pair
pm	Product moment matrix
wgtm	Matrix of weights for each item pair
sumwgtm	Sum of lower triangle matrix wgtm

lower	Lower asymptotes
upper	Upper asymptotes
residuals	Residual matrix from approximation of the pm matrix
final.constants	Final constants
factor.cor	Covariance matrix
thresholds	Threshold parameters
uniquenesses	Uniquenesses
loadings	Matrix of standardized factor loadings (delta parametrization)
loadings.theta	Matrix of factor loadings $F$ (theta parametrization)
residcorr	Matrix of residual correlations
Nobs	Number of observations
Nitems	Number of items
Fpatt	Pattern loading matrix for $F$
Ppatt	Pattern loading matrix for $P$
Psipatt	Pattern loading matrix for $\Psi$
dat	Used dataset
dimensions	Number of dimensions
iter	Number of iterations
Nestpars	Number of estimated parameters
chisquare	Statistic $\chi^2$
df	Degrees of freedom
chisquare_df	Ratio $\chi^2/df$
rmsea	RMSEA statistic
p.chisquare	Significance for $\chi^2$ statistic
omega.rel	Reliability of the sum score according to Green and Yang (2009)

**Author(s)**

Alexander Robitzsch

**References**

- Fraser, C., & McDonald, R. P. (1988). NOHARM: Least squares item factor analysis. *Multivariate Behavioral Research*, **23**, 267-269.
- Fraser, C., & McDonald, R. P. (2012). *NOHARM 4 Manual*. <http://noharm.niagararesearch.ca/nh4man/nhman.html>.
- Green, S. B., & Yang, Y. (2009). Reliability of summed item scores using structural equation modeling: An alternative to coefficient alpha. *Psychometrika*, **74**, 155-167.
- McDonald, R. P. (1982a). Linear versus models in item response theory. *Applied Psychological Measurement*, **6**, 379-396.
- McDonald, R. P. (1982b). *Unidimensional and multidimensional models for item response theory*. I.R.T., C.A.T. conference, Minneapolis, 1982, Proceedings.
- McDonald, R. P. (1997). Normal-ogive multidimensional model. In W. van der Linden & R. K. Hambleton (1997): *Handbook of modern item response theory* (pp. 257-269). New York: Springer.

## See Also

EAP person parameter estimates can be obtained by [R2noharm.EAP](#).

Model fit can be assessed by [modelfit.sirt](#).

See [R2noharm](#) for running the NOHARM software from within R.

See Fraser and McDonald (2012) for an implementation of the NOHARM model which is available as freeware (<http://noharm.niagararesearch.ca/>).

## Examples

```
#####
# SIMULATED EXAMPLE 1: Two-dimensional IRT model with 10 items
#####

**** data simulation
set.seed(9776)
N <- 3400 # sample size
# define difficulties
f0 <- c( .5 , .25 , -.25 , -.5 , 0 , -.5 , -.25 , .25 , .5 , 0 )
I <- length(f0)
# define loadings
f1 <- matrix( 0 , I , 2 )
f1[ 1:5,1] <- c(.8,.7,.6,.5 , .5)
f1[ 6:10 ,2] <- c(.8,.7,.6,.5 , .5 )
# covariance matrix
Pval <- matrix( c(1,.5,.5,1) , 2 , 2 )
# simulate theta
library(MASS)
theta <- MASS::mvrnorm(N , mu=c(0,0) , Sigma = Pval )
# simulate item responses
dat <- matrix( NA , N , I )
for (ii in 1:I){ # ii <- 1
  dat[,ii] <- 1*(pnorm(f0[ii]+theta[,1]*f1[ii,1]+theta[,2]*f1[ii,2])>runif(N))
}
colnames(dat) <- paste0("I" , 1:I)

**** Model 1: Two-dimensional CFA with estimated item loadings
# define pattern matrices
Pval <- .3+0*Pval
Ppatt <- 1*(Pval>0)
diag(Ppatt) <- 0
diag(Pval) <- 1
Fval <- .7 * ( f1>0)
Fpatt <- 1 * ( Fval > 0 )
# estimate model
mod1 <- noharm.sirt( dat=dat , Ppatt=Ppatt,Fpatt=Fpatt , Fval=Fval , Pval=Pval )
summary(mod1)
# EAP ability estimates
pmod1 <- R2noharm.EAP(mod1 , theta.k = seq(-4,4,len=10) )
# model fit
summary( modelfit.sirt( mod1 ))
```

```

# estimate model based on tetrachoric correlations
mod1b <- noharm.sirt( dat=dat , Ppatt=Ppatt,Fpatt=Fpatt , Fval=Fval , Pval=Pval ,
                    modesttype=2)
summary(mod1b)

## Not run:
##### Model 2: Two-dimensional CFA with correlated residuals
# define pattern matrix for residual correlation
Pspipatt <- 0*diag(I)
Pspipatt[1,2] <- 1
Psival <- 0*Pspipatt
# estimate model
mod2 <- noharm.sirt( dat=dat , Ppatt=Ppatt,Fpatt=Fpatt , Fval=Fval , Pval=Pval ,
                    Psival=Psival , Pspipatt=Pspipatt )
summary(mod2)

##### Model 3: Two-dimensional Rasch model
# pattern matrices
Fval <- matrix(0,10,2)
Fval[1:5,1] <- Fval[6:10,2] <- 1
Fpatt <- 0*Fval
Ppatt <- Pval <- matrix(1,2,2)
Pval[1,2] <- Pval[2,1] <- 0
# estimate model
mod3 <- noharm.sirt( dat=dat , Ppatt=Ppatt,Fpatt=Fpatt , Fval=Fval , Pval=Pval )
summary(mod3)
# model fit
summary( modelfit.sirt( mod3 ))

*** compare fit with NOHARM
noharm.path <- "c:/NOHARM"
P.pattern <- Ppatt ; P.init <- Pval
F.pattern <- Fpatt ; F.init <- Fval
mod3b <- R2noharm( dat = dat , model.type="CFA" ,
                  F.pattern = F.pattern , F.init = F.init , P.pattern = P.pattern ,
                  P.init = P.init , writename = "example_sim_2dim_rasch" ,
                  noharm.path = noharm.path , dec = "," )
summary(mod3b)

#####
# EXAMPLE 2: data.read
#####

data(data.read)
dat <- data.read
I <- ncol(dat)

##### Model 1: Unidimensional Rasch model
Fpatt <- matrix( 0 , I , 1 )
Fval <- 1 + 0*Fpatt
Ppatt <- Pval <- matrix(1,1,1)
# estimate model
mod1 <- noharm.sirt( dat=dat , Ppatt=Ppatt,Fpatt=Fpatt , Fval=Fval , Pval=Pval )

```

```

summary(mod1)
plot(mod1) # semPaths plot

##### Model 2: Rasch model in which item pairs within a testlet are excluded
wgtm <- matrix( 1 , I , I )
wgtm[1:4,1:4] <- wgtm[5:8,5:8] <- wgtm[ 9:12, 9:12] <- 0
# estimation
mod2 <- noharm.sirt( dat=dat , Ppatt=Ppatt,Fpatt=Fpatt , Fval=Fval , Pval=Pval , wgtm=wgtm)
summary(mod2)

##### Model 3: Rasch model with correlated residuals
Psipatt <- Psival <- 0*diag(I)
Psipatt[1:4,1:4] <- Psipatt[5:8,5:8] <- Psipatt[ 9:12, 9:12] <- 1
diag(Psipatt) <- 0
Psival <- .6*(Psipatt>0)
# estimation
mod3 <- noharm.sirt( dat=dat , Ppatt=Ppatt,Fpatt=Fpatt , Fval=Fval , Pval=Pval ,
                    Psival=Psival , Psipatt=Psipatt )
summary(mod3)
# allow only positive residual correlations
mod3b <- noharm.sirt( dat=dat , Ppatt=Ppatt,Fpatt=Fpatt , Fval=Fval , Pval=Pval ,
                    Psival=Psival , Psipatt=Psipatt , pos.residcorr=TRUE)
summary(mod3b)

##### Model 4: Rasch testlet model
Fval <- Fpatt <- matrix( 0 , I , 4 )
Fval[,1] <- Fval[1:4,2] <- Fval[5:8,3] <- Fval[9:12,4 ] <- 1
Ppatt <- Pval <- diag(4)
colnames(Ppatt) <- c("g" , "A" , "B" ,"C")
Pval <- .5*Pval
# estimation
mod4 <- noharm.sirt( dat=dat , Ppatt=Ppatt,Fpatt=Fpatt , Fval=Fval , Pval=Pval )
summary(mod4)
# allow only positive variance entries
mod4b <- noharm.sirt( dat=dat , Ppatt=Ppatt,Fpatt=Fpatt , Fval=Fval , Pval=Pval ,
                    pos.variance=TRUE )
summary(mod4b)

##### Model 5: Bifactor model
Fval <- matrix( 0 , I , 4 )
Fval[,1] <- Fval[1:4,2] <- Fval[5:8,3] <- Fval[9:12,4 ] <- .6
Fpatt <- 1 * ( Fval > 0 )
Pval <- diag(4)
Ppatt <- 0*Pval
colnames(Ppatt) <- c("g" , "A" , "B" ,"C")
# estimation
mod5 <- noharm.sirt( dat=dat , Ppatt=Ppatt,Fpatt=Fpatt , Fval=Fval , Pval=Pval )
summary(mod5)
# allow only positive loadings
mod5b <- noharm.sirt( dat=dat , Ppatt=Ppatt,Fpatt=Fpatt , Fval=Fval , Pval=Pval ,
                    pos.loading=TRUE )
summary(mod5b)
summary( modelfit.sirt(mod5b))

```

```

##### Model 6: 3-dimensional Rasch model
Fval <- matrix( 0 , I , 3 )
Fval[1:4,1] <- Fval[5:8,2] <- Fval[9:12,3 ] <- 1
Fpatt <- 0*Fval
Pval <- .6*diag(3)
diag(Pval) <- 1
Ppatt <- 1+0*Pval
colnames(Ppatt) <- c("A", "B", "C")
# estimation
mod6 <- noharm.sirt( dat=dat , Ppatt=Ppatt,Fpatt=Fpatt , Fval=Fval , Pval=Pval )
summary(mod6)
summary( modelfit.sirt(mod6) ) # model fit

##### Model 7: 3-dimensional 2PL model
Fval <- matrix( 0 , I , 3 )
Fval[1:4,1] <- Fval[5:8,2] <- Fval[9:12,3 ] <- 1
Fpatt <- Fval
Pval <- .6*diag(3)
diag(Pval) <- 1
Ppatt <- 1+0*Pval
diag(Ppatt) <- 0
colnames(Ppatt) <- c("A", "B", "C")
# estimation
mod7 <- noharm.sirt( dat=dat , Ppatt=Ppatt,Fpatt=Fpatt , Fval=Fval , Pval=Pval )
summary(mod7)
summary( modelfit.sirt(mod7) )

##### Model 8: Exploratory factor analysis with 3 dimensions
# estimation
mod8 <- noharm.sirt( dat=dat , dimensions=3 )
summary(mod8)

## End(Not run)

```

**Description**

This function does nonparametric item response function estimation (Ramsay, 1991).

**Usage**

```
np.dich(dat, theta, thetagrid, progress = FALSE, bwscale = 1.1,
        method = "normal")
```

**Arguments**

dat	An $N \times I$ data frame of dichotomous item responses
theta	Estimated theta values, for example weighted likelihood estimates from <a href="#">wle.rasch</a>
thetagrid	A vector of theta values where the nonparametric item response functions shall be evaluated.
progress	Display progress?
bwscale	The bandwidth parameter $h$ is calculated by the formula $h = \text{bwscale} \cdot N^{-1/5}$
method	The default normal performs kernel regression with untransformed item responses. The method binomial uses nonparametric logistic regression implemented in the <b>sm</b> library.

**Value**

A list with following entries

dat	Original data frame
thetagrid	Vector of theta values at which the item response functions are evaluated
theta	Used theta values as person parameter estimates
estimate	Estimated item response functions
...	

**Author(s)**

Alexander Robitzsch

**References**

Ramsay, J. O. (1991). Kernel smoothing approaches to nonparametric item characteristic curve estimation. *Psychometrika*, **56**, 611-630.

**Examples**

```
#####
# EXAMPLE 1: Reading dataset
#####
data( data.read )
dat <- data.read

# estimate Rasch model
mod <- rasch.mml2( dat )
# WLE estimation
wle1 <- wle.rasch( dat=dat , b =mod$item$b )$theta
# nonparametric function estimation
np1 <- np.dich( dat=dat, theta= wle1, thetagrid = seq(-2.5 , 2.5 , len=100 ) )
print( str(np1))
# plot nonparametric item response curves
plot( np1 , b = mod$item$b )
```



---

pbivnorm2

*Cumulative Function for the Bivariate Normal Distribution*

---

### Description

This function evaluates the bivariate normal distribution  $\Phi_2(x, y; \rho)$  assuming zero means and unit variances. It uses a simple approximation by Cox and Wermuth (1991) with corrected formulas in Hong (1999).

### Usage

```
pbivnorm2(x, y, rho)
```

### Arguments

x	Vector of $x$ coordinates
y	Vector of $y$ coordinates
rho	Vector of correlations between random normal variates

### Value

Vector of probabilities

### Note

The function is less precise for correlations near 1 or -1.

### Author(s)

Alexander Robitzsch

### References

Cox, D. R., & Wermuth, N. (1991). A simple approximation for bivariate and trivariate normal integrals. *International Statistical Review*, **59**, 263-269.

Hong, H. P. (1999). An approximation to bivariate and trivariate normal integrals. *Engineering and Environmental Systems*, **16**, 115-127.

### See Also

See also the pbivnorm function in the **pbivnorm** package.

**Examples**

```

library(pbivnorm)
# define input
x <- c(0, 0, .5, 1, 1)
y <- c(0, -.5, 1, 3, .5)
rho <- c(.2, .8, -.4, .6, .5)
# compare pbivnorm2 and pbivnorm functions
pbiv2 <- pbivnorm2( x = x, y = y, rho = rho )
pbiv <- pbivnorm::pbivnorm( x, y, rho = rho )
max( abs(pbiv-pbiv2) )
## [1] 0.0030626
round( cbind( x, y, rho, pbiv, pbiv2 ), 4 )
##           x   y rho  pbiv pbiv2
## [1,] 0.0  0.0 0.2 0.2820 0.2821
## [2,] 0.0 -0.5 0.8 0.2778 0.2747
## [3,] 0.5  1.0 -0.4 0.5514 0.5514
## [4,] 1.0  3.0  0.6 0.8412 0.8412
## [5,] 1.0  0.5  0.5 0.6303 0.6304

```

pcm.conversion

*Conversion of the Parameterization of the Partial Credit Model***Description**

Converts a parameterization of the partial credit model (see Details).

**Usage**

```
pcm.conversion(b)
```

**Arguments**

**b** Matrix of item-category-wise intercepts  $b_{ik}$  (see Details).

**Details**

Assume that the input matrix **b** containing parameters  $b_{ik}$  is defined according to the following parametrization of the partial credit model

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

if item  $i$  possesses  $K_i$  categories. The transformed parameterization is defined as

$$b_{ik} = k\delta_i + \sum_{v=1}^k \tau_{iv} \quad \text{with} \quad \sum_{k=1}^{K_i} \tau_{ik} = 0$$

The function `pcm.conversion` has the  $\delta$  and  $\tau$  parameters as values. The  $\delta$  parameter is simply  $\delta_i = b_{iK_i}/K_i$ .

**Value**

List with the following entries

delta	Vector of $\delta$ parameters
tau	Matrix of $\tau$ parameters

**Author(s)**

Alexander Robitzsch

**Examples**

```
## Not run:
#####
# EXAMPLE 1: Transformation PCM for data.mg
#####

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

**** Model 1: estimate partial credit model in parameterization "PCM"
mod1a <- TAM::tam.mml( dat , irtmodel="PCM")
# use parameterization "PCM2"
mod1b <- TAM::tam.mml( dat , irtmodel="PCM2")
summary(mod1a)
summary(mod1b)

# convert parameterization of Model 1a into parameterization of Model 1b
b <- mod1a$item[ , c("AXsi_.Cat1", "AXsi_.Cat2", "AXsi_.Cat3") ]
# compare results
pcm.conversion(b)
mod1b$xsi

## End(Not run)
```

---

pcm.fit

*Item and Person Fit Statistics for the Partial Credit Model*

---

**Description**

Computes item and person fit statistics in the partial credit model (Wright & Masters, 1990). The rating scale model is accomodated as a particular partial credit model (see Example 3).

**Usage**

```
pcm.fit(b, theta, dat)
```

**Arguments**

b	Matrix with item category parameters (see Examples)
theta	Vector with estimated person parameters
dat	Dataset with item responses

**Value**

A list with entries	
itemfit	Item fit statistics
personfit	Person fit statistics

**References**

Wright, B. D., & Masters, G. N. (1990). Computation of outfit and infit statistics. *Rasch Measurement Transactions*, **3:4**, 84-85.

**See Also**

See also `personfit.stat` for person fit statistics for dichotomous item responses. See also the **PerFit** package for further person fit statistics.

Item fit in other R packages: `itemfit` (**eRm**), `tam.fit` (**TAM**), `itemfit` (**mirt**), `item.fit` (**ltm**),

Person fit in other R packages: `itemfit` (**eRm**), `itemfit` (**mirt**), `person.fit` (**ltm**),

See `pcm.conversion` for conversions of different parametrizations of the partial credit model.

**Examples**

```
## Not run:
#####
# EXAMPLE 1: Partial credit model
#####

data(data.Students,package="CDM")
dat <- data.Students
# select items
items <- c(paste0("sc" , 1:4 ) , paste0("mj" , 1:4 ) )
dat <- dat[,items]
dat <- dat[ rowSums( 1 - is.na(dat) ) > 0 , ]

**** Model 1a: Partial credit model in TAM
# estimate model
mod1a <- TAM::tam.mml( resp=dat )
summary(mod1a)
# estimate person parameters
wle1a <- TAM::tam.wle(mod1a)
# extract item parameters
b1 <- - mod1a$AXsi[ , -1 ]
# parametrization in xsi parameters
b2 <- matrix( mod1a$xsi$xsi , ncol=3 , byrow=TRUE )
```

```

# convert b2 to b1
b1b <- 0*b1
b1b[,1] <- b2[,1]
b1b[,2] <- rowSums( b2[,1:2] )
b1b[,3] <- rowSums( b2[,1:3] )
# assess fit
fit1a <- pcm.fit(b=b1, theta=wle1a$theta , dat)
fit1a$item

#####
# EXAMPLE 2: Rasch model
#####

data(data.read)
dat <- data.read

###* Rasch model in TAM
# estimate model
mod <- TAM::tam.mml( resp=dat )
summary(mod)
# estimate person parameters
wle <- TAM::tam.wle(mod)
# extract item parameters
b1 <- - mod$AXsi[ , -1 ]
# assess fit
fit1a <- pcm.fit(b=b1, theta=wle$theta , dat)
fit1a$item

#####
# EXAMPLE 3: Rating scale model
#####

data(data.Students,package="CDM")
dat <- data.Students
items <- paste0("sc" , 1:4 )
dat <- dat[,items]
dat <- dat[ rowSums( 1 - is.na(dat) ) > 0 , ]

###* Model 1: Rating scale model in TAM
# estimate model
mod1 <- tam.mml( resp=dat , irtmodel="RSM")
summary(mod1)
# estimate person parameters
wle1 <- tam.wle(mod1)
# extract item parameters
b1 <- - mod1a$AXsi[ , -1 ]
# fit statistic
pcm.fit(b=b1, theta=wle1$theta, dat)

## End(Not run)

```

---

person.parameter.rasch.copula

*Person Parameter Estimation of the Rasch Copula Model (Braeken, 2011)*

---

### Description

Ability estimates as maximum likelihood estimates (MLE) are provided by the Rasch copula model.

### Usage

```
person.parameter.rasch.copula(raschcopula.object, numdiff.parm = 0.001,
  conv.parm = 0.001, maxiter = 20, stepwidth = 1,
  print.summary = TRUE, ...)
```

### Arguments

raschcopula.object	Object which is generated by the <code>coderasch.copula2</code> function.
numdiff.parm	Parameter $h$ for numerical differentiation
conv.parm	Convergence criterion
maxiter	Maximum number of iterations
stepwidth	Maximal increment in iterations
print.summary	Print summary?
...	Further arguments to be passed

### Value

A list with following entries

person	Estimated person parameters
se.inflat	Inflation of individual standard errors due to local dependence
theta.table	Ability estimates for each unique response pattern
pattern.in.data	Item response pattern
summary.theta.table	Summary statistics of person parameter estimates

### Author(s)

Alexander Robitzsch

### See Also

See [rasch.copula2](#) for estimating Rasch copula models.

## Examples

```
#####
# EXAMPLE 1: Reading Data
#####

data(data.read)
dat <- data.read

# define item cluster
itemcluster <- rep( 1:3 , each = 4 )
mod1 <- rasch.copula2( dat , itemcluster = itemcluster )
summary(mod1)

# person parameter estimation under the Rasch copula model
pmod1 <- person.parameter.rasch.copula(raschcopula.object = mod1 )
## Mean percentage standard error inflation
## missing.pattern Mperc.seinflat
## 1          1          6.35

## Not run:
#####
# SIMULATED EXAMPLE 2: 12 items nested within 3 item clusters (testlets)
# Cluster 1 -> Items 1-4; Cluster 2 -> Items 6-9; Cluster 3 -> Items 10-12
#####

set.seed(967)
I <- 12                # number of items
n <- 450               # number of persons
b <- seq(-2,2, len=I)  # item difficulties
b <- sample(b)         # sample item difficulties
theta <- rnorm( n , sd = 1 )  # person abilities
# itemcluster
itemcluster <- rep(0,I)
itemcluster[ 1:4 ] <- 1
itemcluster[ 6:9 ] <- 2
itemcluster[ 10:12 ] <- 3
# residual correlations
rho <- c( .35 , .25 , .30 )

# simulate data
dat <- sim.rasch.dep( theta , b , itemcluster , rho )
colnames(dat) <- paste("I" , seq(1,ncol(dat)) , sep="")

# estimate Rasch copula model
mod1 <- rasch.copula2( dat , itemcluster = itemcluster )
summary(mod1)

# person parameter estimation under the Rasch copula model
pmod1 <- person.parameter.rasch.copula(raschcopula.object = mod1 )
## Mean percentage standard error inflation
## missing.pattern Mperc.seinflat
## 1          1          10.48
```

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

---

```
personfit.stat      Person Fit Statistics for the Rasch Model
```

---

### Description

This function collects some person fit statistics for the Rasch model (Karabatsos, 2003; Meijer & Sijtsma, 2001).

### Usage

```
personfit.stat(dat, abil, b)
```

### Arguments

dat	An $N \times I$ data frame of dichotomous item responses
abil	An ability estimate, e.g. the WLE
b	Estimated item difficulty

### Value

A data frame with following columns (see Meijer & Sijtsma 2001 for a review of different person fit statistics):

case	Case index
abil	Ability estimate <i>abil</i>
mean	Person mean of correctly solved items
caution	Caution index
depend	Dependability index
ECI1	<i>ECI1</i>
ECI2	<i>ECI2</i>
ECI3	<i>ECI3</i>
ECI4	<i>ECI4</i>
ECI5	<i>ECI5</i>
ECI6	<i>ECI6</i>
l0	Fit statistic $l_0$
lz	Fit statistic $l_z$
outfit	Person outfit statistic
infit	Person infit statistic
rpbis	Point biserial correlation of item responses and item $p$ values
rpbis.itemdiff	Point biserial correlation of item responses and item difficulties $b$
U3	Fit statistic $U_3$



**Author(s)**

Alexander Robitzsch

**References**

Karabatsos, G. (2003). Comparing the aberrant response detection performance of thirty-six person-fit statistics. *Applied Measurement in Education*, **16**, 277-298.

Meijer, R. R., & Sijtsma, K. (2001). Methodology review: Evaluating person fit. *Applied Psychological Measurement*, **25**, 107-135.

**See Also**

See [pcm.fit](#) for person fit in the partial credit model.

See the **irtProb** and **PerFit** packages for person fit statistics and person response curves and functions included in other packages: [mirt::personfit](#), [eRm::personfit](#) and [ltm::person.fit](#).

**Examples**

```
#####
# EXAMPLE 1: Person fit Reading Data
#####

data(data.read)
dat <- data.read

# estimate Rasch model
mod <- rasch.mml2( dat )
# WLE
wle1 <- wle.rasch( dat,b=mod$item$b )$theta
b <- mod$item$b # item difficulty

# evaluate person fit
pf1 <- personfit.stat( dat = dat , abil=wle1 , b=b)

## Not run:
# dimensional analysis of person fit statistics
x0 <- na.omit(pf1[ , -c(1:3) ])
factanal( x=x0 , factors=2 , rotation="promax" )
## Loadings:
##
##          Factor1 Factor2
## caution      0.914
## depend       0.293  0.750
## ECI1         0.869  0.160
## ECI2         0.869  0.162
## ECI3         1.011
## ECI4         1.159 -0.269
## ECI5         1.012
## ECI6         0.879  0.130
## l0           0.409 -1.255
## lz          -0.504 -0.529
## outfit       0.297  0.702
```

```

## infit          0.362  0.695
## rpbis          -1.014
## rpbis.itemdiff 1.032
## U3             0.735  0.309
##
## Factor Correlations:
##           Factor1 Factor2
## Factor1   1.000  -0.727
## Factor2  -0.727   1.000
##
## End(Not run)

```

---

pgenlogis	<i>Calculation of Probabilities and Moments for the Generalized Logistic Item Response Model</i>
-----------	--

---

### Description

Calculation of probabilities and moments for the generalized logistic item response model (Stukel, 1988).

### Usage

```
pgenlogis(x, alpha1 = 0, alpha2 = 0)
```

```
genlogis.moments(alpha1, alpha2)
```

### Arguments

x	Vector
alpha1	Upper tail parameter $\alpha_1$ in the generalized logistic item response model. The default is 0.
alpha2	Lower tail parameter $\alpha_2$ parameter in the generalized logistic item response model. The default is 0.

### Details

The class of generalized logistic link functions contain the most important link functions using the specifications (Stukel, 1988):

- logistic link function  $L$ :

$$L(x) \approx G_{(\alpha_1=0, \alpha_2=0)}[x]$$

- probit link function  $\Phi$ :

$$\Phi(x) \approx G_{(\alpha_1=0.165, \alpha_2=0.165)}[1.47x]$$

- loglog link function  $H$ :

$$H(x) \approx G_{(\alpha_1=-0.037, \alpha_2=0.62)}[-0.39 + 1.20x - 0.007x^2]$$

- cloglog link function  $H$ :

$$H(x) \approx G_{(\alpha_1=0.62, \alpha_2=-0.037)}[0.54 + 1.64x + 0.28x^2 + 0.046x^3]$$

## Value

Vector of probabilities or moments

## Author(s)

Alexander Robitzsch

## References

Stukel, T. A. (1988). Generalized logistic models. *Journal of the American Statistical Association*, **83**, 426-431.

## Examples

```
pgenlogis( x=c(-.3 , 0 , .25 , 1 ) , alpha1=0 , alpha2= .6 )
## [1] 0.4185580 0.5000000 0.5621765 0.7310586

#####
# compare link functions
x <- seq( -3 , 3 , .1 )

###
# logistic link
y <- pgenlogis( x , alpha1=0 , alpha2=0 )
plot( x , plogis(x) , type="l" , main="Logistic Link" , lwd=2)
points( x , y , pch=1 , col=2 )

###
# probit link
round( genlogis.moments( alpha1=.165 , alpha2=.165 ) , 3 )
## M SD Var
## 0.000 1.472 2.167
# SD of generalized logistic link function is 1.472
y <- pgenlogis( x * 1.47 , alpha1=.165 , alpha2=.165 )
plot( x , pnorm(x) , type="l" , main="Probit Link" , lwd=2)
points( x , y , pch=1 , col=2 )

###
# loglog link
y <- pgenlogis( -.39 + 1.20*x -.007*x^2 , alpha1=-.037 , alpha2=.62 )
plot( x , exp( - exp( -x ) ) , type="l" , main="Loglog Link" , lwd=2,
      ylab="loglog(x) = exp(-exp(-x))" )
points( x , y , pch=17 , col=2 )
```

```

****
# cloglog link
y <- pgenlogis( .54+1.64*x +.28*x^2 + .046*x^3 , alpha1=.062 , alpha2=-.037 )
plot( x , 1-exp( - exp(x) ) , type="l" , main="Cloglog Link" , lwd=2,
      ylab="loglog(x) = 1-exp(-exp(x))" )
points( x , y , pch=17 , col=2 )

```

---

plausible.value.imputation.raschtype

*Plausible Value Imputation in Generalized Logistic Item Response Model*

---

### Description

This function performs unidimensional plausible value imputation (Adams & Wu, 2007; Mislevy, 1991).

### Usage

```

plausible.value.imputation.raschtype(data=NULL, f.yi.qk=NULL, X,
  Z=NULL, beta0=rep(0, ncol(X)), sig0=1, b=rep(1, ncol(X)),
  a=rep(1, length(b)), c=rep(0, length(b)), d=1+0*b,
  alpha1=0, alpha2=0, theta.list=seq(-5, 5, len=50),
  cluster=NULL, iter, burnin, nplausible=1, printprogress=TRUE)

```

### Arguments

data	An $N \times I$ data frame of dichotomous responses
f.yi.qk	An optional matrix which contains the individual likelihood. This matrix is produced by <a href="#">rasch.mm12</a> or <a href="#">rasch.copula2</a> . The use of this argument allows the estimation of the latent regression model independent of the parameters of the used item response model.
X	A matrix of individual covariates for the latent regression of $\theta$ on $X$
Z	A matrix of individual covariates for the regression of individual residual variances on $Z$
beta0	Initial vector of regression coefficients
sig0	Initial vector of coefficients for the variance heterogeneity model
b	Vector of item difficulties. It must not be provided if the individual likelihood f.yi.qk is specified.
a	Optional vector of item slopes
c	Optional vector of lower item asymptotes
d	Optional vector of upper item asymptotes
alpha1	Parameter $\alpha_1$ in generalized item response model

alpha2	Parameter $\alpha_2$ in generalized item response model
theta.list	Vector of theta values at which the ability distribution should be evaluated
cluster	Cluster identifier (e.g. schools or classes) for including theta means in the plausible imputation.
iter	Number of iterations
burnin	Number of burn-in iterations for plausible value imputation
nplausible	Number of plausible values
printprogress	A logical indicated whether iteration progress should be displayed at the console.

### Details

Plausible values are drawn from the latent regression model with heterogeneous variances:

$$\theta_p = X_p\beta + \epsilon_p \quad , \quad \epsilon_p \sim N(0, \sigma_p^2) \quad , \quad \log(\sigma_p) = Z_p\gamma + \nu_p$$

### Value

A list with following entries:

coefs.X	Sampled regression coefficients for covariates $X$
coefs.Z	Sampled coefficients for modeling variance heterogeneity for covariates $Z$
pvdrows	Matrix with drawn plausible values
posterior	Posterior distribution from last iteration
EAP	Individual EAP estimate
SE.EAP	Standard error of the EAP estimate
pv.indexes	Index of iterations for which plausible values were drawn

### Author(s)

Alexander Robitzsch

### References

- Adams, R., & Wu, M. (2007). The mixed-coefficients multinomial logit model: A generalized form of the Rasch model. In M. von Davier & C. H. Carstensen: *Multivariate and Mixture Distribution Rasch Models: Extensions and Applications* (pp. 57-76). New York: Springer.
- Mislevy, R. J. (1991). Randomization-based inference about latent variables from complex samples. *Psychometrika*, **56**, 177-196.

### See Also

For estimating the latent regression model see [latent.regression.em.raschtype](#).

## Examples

```
#####
# SIMULATED EXAMPLE 1: Rasch model with covariates
#####

set.seed(899)
I <- 21      # number of items
b <- seq(-2,2, len=I) # item difficulties
n <- 2000    # number of students

# simulate theta and covariates
theta <- rnorm( n )
x <- .7 * theta + rnorm( n , .5 )
y <- .2 * x + .3*theta + rnorm( n , .4 )
dfr <- data.frame( theta , 1 , x , y )

# simulate Rasch model
dat1 <- sim.raschtype( theta = theta , b = b )

# Plausible value draws
pv1 <- plausible.value.imputation.raschtype(data=dat1 , X=dfr[,-1] , b = b ,
      nplausible=3 , iter=10 , burnin=5)
# estimate linear regression based on first plausible value
mod1 <- lm( pv1$pvdraws[,1] ~ x+y )
summary(mod1)
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.27755    0.02121  -13.09  <2e-16 ***
## x           0.40483    0.01640   24.69  <2e-16 ***
## y           0.20307    0.01822   11.15  <2e-16 ***

# true regression estimate
summary( lm( theta ~ x + y ) )
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.27821    0.01984  -14.02  <2e-16 ***
## x           0.40747    0.01534   26.56  <2e-16 ***
## y           0.18189    0.01704   10.67  <2e-16 ***

## Not run:
#####
# SIMULATED EXAMPLE 2: Classical test theory, homogeneous regression variance
#####

set.seed(899)
n <- 3000    # number of students
x <- round( runif( n , 0 , 1 ) )
y <- rnorm(n)
# simulate true score theta
theta <- .4*x + .5 * y + rnorm(n)
# simulate observed score by adding measurement error
sig.e <- rep( sqrt(.40) , n )
theta_obs <- theta + rnorm( n , sd=sig.e)
```

```

# define theta grid for evaluation of density
theta.list <- mean(theta_obs) + sd(theta_obs) * seq( - 5 , 5 , length=21)
# compute individual likelihood
f.yi.qk <- dnorm( outer( theta_obs , theta.list , "-" ) / sig.e )
f.yi.qk <- f.yi.qk / rowSums(f.yi.qk)
# define covariates
X <- cbind( 1 , x , y )
# draw plausible values
mod2 <- plausible.value.imputation.raschtype( f.yi.qk =f.yi.qk ,
      theta.list=theta.list , X=X , iter=10 , burnin=5)

# linear regression
mod1 <- lm( mod2$pvdraws[,1] ~ x+y )
summary(mod1)
  ##           Estimate Std. Error t value Pr(>|t|)
  ## (Intercept) -0.01393    0.02655  -0.525    0.6
  ## x            0.35686    0.03739   9.544 <2e-16 ***
  ## y            0.53759    0.01872  28.718 <2e-16 ***

# true regression model
summary( lm( theta ~ x + y ) )
  ##           Estimate Std. Error t value Pr(>|t|)
  ## (Intercept)  0.002931    0.026171   0.112   0.911
  ## x            0.359954    0.036864   9.764 <2e-16 ***
  ## y            0.509073    0.018456  27.584 <2e-16 ***

#####
# SIMULATED EXAMPLE 3: Classical test theory, heterogeneous regression variance
#####

set.seed(899)
n <- 5000      # number of students
x <- round( runif( n , 0 , 1 ) )
y <- rnorm(n)
# simulate true score theta
theta <- .4*x + .5 * y + rnorm(n) * ( 1 - .4 * x )
# simulate observed score by adding measurement error
sig.e <- rep( sqrt(.40) , n )
theta_obs <- theta + rnorm( n , sd=sig.e)

# define theta grid for evaluation of density
theta.list <- mean(theta_obs) + sd(theta_obs) * seq( - 5 , 5 , length=21)
# compute individual likelihood
f.yi.qk <- dnorm( outer( theta_obs , theta.list , "-" ) / sig.e )
f.yi.qk <- f.yi.qk / rowSums(f.yi.qk)
# define covariates
X <- cbind( 1 , x , y )
# draw plausible values (assuming variance homogeneity)
mod3a <- plausible.value.imputation.raschtype( f.yi.qk =f.yi.qk ,
      theta.list=theta.list , X=X , iter=10 , burnin=5)
# draw plausible values (assuming variance heterogeneity)
# -> include predictor Z

```

```

mod3b <- plausible.value.imputation.raschtype( f.yi.qk =f.yi.qk ,
      theta.list=theta.list , X=X , Z=X , iter=10 , burnin=5)

# investigate variance of theta conditional on x
res3 <- sapply( 0:1 , FUN = function(vv){
  c( var(theta[x==vv]), var(mod3b$pvdraw[x==vv,1]),
    var(mod3a$pvdraw[x==vv,1]))})
rownames(res3) <- c("true" , "pv(hetero)" , "pv(homog)" )
colnames(res3) <- c("x=0","x=1")
## > round( res3 , 2 )
##           x=0  x=1
## true       1.30 0.58
## pv(hetero) 1.29 0.55
## pv(homog)  1.06 0.77
## -> assuming heteroscedastic variances recovers true conditional variance

## End(Not run)

```

---

plot.mcmc.sirt

*Plot Function for Objects of Class mcmc.sirt*


---

## Description

Plot function for objects of class `mcmc.sirt`. These objects are generated by: [mcmc.2pno](#), [mcmc.2pnoh](#), [mcmc.3pno.testlet](#), [mcmc.2pno.ml](#)

## Usage

```

## S3 method for class 'mcmc.sirt'
plot( x, layout = 1, confllevel = 0.9, round.summ = 3,
      lag.max = .1 , col.smooth = "red", lwd.smooth = 2, col.ci = "orange",
      cex.summ = 1, ask = FALSE, ...)

```

## Arguments

<code>x</code>	Object of class <code>mcmc.sirt</code>
<code>layout</code>	Layout type. <code>layout=1</code> is the standard coda plot output, <code>layout=2</code> gives a slightly different display.
<code>confllevel</code>	Confidence level (only applies to <code>layout=2</code> )
<code>round.summ</code>	Number of digits to be rounded in summary (only applies to <code>layout=2</code> )
<code>lag.max</code>	Maximum lag for autocorrelation plot (only applies to <code>layout=2</code> ). The default of <code>.1</code> means that it is set to 1/10 of the number of iterations.
<code>col.smooth</code>	Color of smooth trend in traceplot (only applies to <code>layout=2</code> )
<code>lwd.smooth</code>	Line type of smooth trend in traceplot (only applies to <code>layout=2</code> )
<code>col.ci</code>	Color for displaying confidence interval (only applies to <code>layout=2</code> )
<code>cex.summ</code>	Cex size for descriptive summary (only applies to <code>layout=2</code> )
<code>ask</code>	Ask for a new plot (only applies to <code>layout=2</code> )
<code>...</code>	Further arguments to be passed



**Author(s)**

Alexander Robitzsch

**See Also**[mcmc.2pno](#), [mcmc.2pnoh](#), [mcmc.3pno](#), [testlet](#), [mcmc.2pno.ml](#)

plot.np.dich

*Plot Method for Object of Class np.dich***Description**

This function plots nonparametric item response functions estimated with `dich.np`.

**Usage**

```
## S3 method for class 'np.dich'
plot(x, b, infit = NULL, outfit = NULL,
     nsize = 100, askplot = TRUE, progress = TRUE, bands = FALSE,
     plot.b = FALSE, shade = FALSE, shadecol = "burlywood1" , ...)
```

**Arguments**

<code>x</code>	Object of class <a href="#">np.dich</a>
<code>b</code>	Estimated item difficulty (threshold)
<code>infit</code>	Infit (optional)
<code>outfit</code>	Outfit (optional)
<code>nsize</code>	XXX
<code>askplot</code>	Ask for new plot?
<code>progress</code>	Display progress?
<code>bands</code>	Draw confidence bands?
<code>plot.b</code>	Plot difficulty parameter?
<code>shade</code>	Shade curves?
<code>shadecol</code>	Shade color
<code>...</code>	Further arguments to be passed

**Author(s)**

Alexander Robitzsch

**See Also**

For examples see [np.dich](#).

---

`polychoric2`*Polychoric Correlation*

---

**Description**

This function estimates the polychoric correlation coefficient using maximum likelihood estimation (Olsson, 1979).

**Usage**

```
polychoric2(dat, maxiter = 100, cor.smooth = TRUE)
```

**Arguments**

<code>dat</code>	A dataset with integer values
<code>maxiter</code>	Maximum number of iterations
<code>cor.smooth</code>	An optional logical indicating whether the polychoric correlation matrix should be smooth to ensure positive definiteness.

**Value**

A list with following entries

<code>tau</code>	Matrix of thresholds
<code>rho</code>	Polychoric correlation matrix
<code>Nobs</code>	Sample size for every item pair
<code>maxcat</code>	Maximum number of categories per item

**Author(s)**

Alexander Robitzsch

**References**

Olsson, U. (1979). Maximum likelihood estimation of the polychoric correlation coefficient. *Psychometrika*, **44**, 443-460.

**See Also**

See the `polychoric` function in the **psych** package.

For estimating tetrachoric correlations see [tetrachoric2](#).

**Examples**

```
#####
# EXAMPLE 1: data.Students | activity scale
#####

data(data.Students, package="CDM")
dat <- data.Students[ , paste0("act" , 1:5 ) ]

# tetrachoric correlation from psych package
library(psych)
t0 <- psych::polychoric( dat )$rho
# Olsson method (maximum likelihood estimation)
t1 <- polychoric2( dat )$rho
# maximum absolute difference
max( abs( t0 - t1 ) )
## [1] 0.006914892
```

---

prmse.subscores.scales

*Proportional Reduction of Mean Squared Error (PRMSE) for Subscale Scores*

---

**Description**

This function estimates the proportional reduction of mean squared error (PRMSE) according to Haberman (Haberman 2008; Haberman, Sinharay & Puhan, 2008).

**Usage**

```
prmse.subscores.scales(data, subscale)
```

**Arguments**

data	An $N \times I$ data frame of item responses
subscale	Vector of labels corresponding to subscales

**Value**

Matrix with columns corresponding to subscales  
The symbol  $X$  denotes the subscale and  $Z$  the whole scale (see also in the Examples section for the structure of this matrix).

**Author(s)**

Alexander Robitzsch

## References

- Haberman, S. J. (2008). When can subscores have value? *Journal of Educational and Behavioral Statistics*, **33**, 204-229.
- Haberman, S., Sinharay, S., & Puhon, G. (2008). Reporting subscores for institutions. *British Journal of Mathematical and Statistical Psychology*, **62**, 79-95.

## Examples

```
#####
# EXAMPLE 1: PRMSE Reading data data.read
#####

data( data.read )
p1 <- prmse.subscores.scales(data=data.read,
                             subscale = substring( colnames(data.read) , 1 ,1 ) )
print( p1 , digits= 3 )
##           A           B           C
## N       328.000 328.000 328.000
## nX       4.000  4.000  4.000
## M.X      2.616  2.811  3.253
## Var.X    1.381  1.059  1.107
## SD.X     1.175  1.029  1.052
## alpha.X  0.545  0.381  0.640
## [...]
## nZ       12.000 12.000 12.000
## M.Z      8.680  8.680  8.680
## Var.Z    5.668  5.668  5.668
## SD.Z     2.381  2.381  2.381
## alpha.Z  0.677  0.677  0.677
## [...]
## cor.TX_Z 0.799  0.835  0.684
## rmse.X   0.585  0.500  0.505
## rmse.Z   0.522  0.350  0.614
## rmse.XZ  0.495  0.350  0.478
## prmse.X  0.545  0.381  0.640
## prmse.Z  0.638  0.697  0.468
## prmse.XZ 0.674  0.697  0.677
#-> Scales A and B do not have lower RMSEA,
#   but for scale C the RMSE is smaller than the RMSE of a
#   prediction based on a whole scale.
```

---

 prob.guttman

*Probabilistic Guttman Model*


---

## Description

This function estimates the probabilistic Guttman model which is a special case of an ordered latent trait model (Hanson, 2000; Proctor, 1970).

**Usage**

```

prob.guttman(dat, pid = NULL, guess.equal = FALSE, slip.equal = FALSE,
             itemlevel = NULL, conv1 = 0.001, glob.conv = 0.001, mmliter = 500)

## S3 method for class 'prob.guttman'
summary(object,...)

## S3 method for class 'prob.guttman'
anova(object,...)

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

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

## S3 method for class 'prob.guttman'
IRT.likelihood(object,...)

## S3 method for class 'prob.guttman'
IRT.posterior(object,...)

```

**Arguments**

dat	An $N \times I$ data frame of dichotomous item responses
pid	Optional vector of person identifiers
guess.equal	Should the same guessing parameters for all the items estimated?
slip.equal	Should the same slipping parameters for all the items estimated?
itemlevel	A vector of item levels of the Guttman scale for each item. If there are $K$ different item levels, then the Guttman scale possesses $K$ ordered trait values.
conv1	Convergence criterion for item parameters
glob.conv	Global convergence criterion for the deviance
mmliter	Maximum number of iterations
object	Object of class prob.guttman
...	Further arguments to be passed

**Value**

An object of class prob.guttman

person	Estimated person parameters
item	Estimated item parameters
theta.k	Ability levels
trait	Estimated trait distribution
ic	Information criteria

deviance	Deviance
iter	Number of iterations
itemdesign	Specified allocation of items to trait levels

**Author(s)**

Alexander Robitzsch

**References**

Hanson, B. (2000). *IRT parameter estimation using the EM algorithm*. Technical Report.

Proctor, C. H. (1970). A probabilistic formulation and statistical analysis for Guttman scaling. *Psychometrika*, **35**, 73-78.

**Examples**

```
#####
# EXAMPLE 1: Dataset Reading
#####
data(data.read)
dat <- data.read

###
# Model 1: estimate probabilistic Guttman model
mod1 <- prob.guttman( dat )
summary(mod1)

###
# Model 2: probabilistic Guttman model with equal guessing and slipping parameters
mod2 <- prob.guttman( dat , guess.equal=TRUE , slip.equal=TRUE)
summary(mod2)

###
# Model 3: Guttman model with three a priori specified item levels
itemlevel <- rep(1,12)
itemlevel[ c(2,5,8,10,12) ] <- 2
itemlevel[ c(3,4,6) ] <- 3
mod3 <- prob.guttman( dat , itemlevel=itemlevel )
summary(mod3)

## Not run:
###
# Model3m: estimate Model 3 in mirt

library(mirt)
# define four ordered latent classes
Theta <- scan(nlines=1)
  0 0 0   1 0 0   1 1 0   1 1 1
Theta <- matrix( Theta , nrow=4 , ncol=3,byrow=TRUE)

# define mirt model
```

```

I <- ncol(dat) # I = 12
mirtmodel <- mirt::mirt.model("
  # specify factors for each item level
  C1 = 1,7,9,11
  C2 = 2,5,8,10,12
  C3 = 3,4,6
")
# get initial parameter values
mod.pars <- mirt::mirt(dat, model=mirtmodel , pars = "values")
# redefine initial parameter values
mod.pars[ mod.pars$name == "d" ,"value" ] <- -1
mod.pars[ mod.pars$name %in% paste0("a",1:3) & mod.pars$est == "value" ] <- 2
mod.pars
# define prior for latent class analysis
lca_prior <- function(Theta,Etable){
  # number of latent Theta classes
  TP <- nrow(Theta)
  # prior in initial iteration
  if ( is.null(Etable) ){ prior <- rep( 1/TP , TP ) }
  # process Etable (this is correct for datasets without missing data)
  if ( ! is.null(Etable) ){
    # sum over correct and incorrect expected responses
    prior <- ( rowSums(Etable[ , seq(1,2*I,2)]) + rowSums(Etable[,seq(2,2*I,2)]) )/I
  }
  prior <- prior / sum(prior)
  return(prior)
}
# estimate model in mirt
mod3m <- mirt::mirt(dat, mirtmodel , pars = mod.pars , verbose=TRUE ,
  technical = list( customTheta=Theta , customPriorFun = lca_prior ) )
# correct number of estimated parameters
mod3m@nest <- as.integer(sum(mod.pars$est) + nrow(Theta)-1 )
# extract log-likelihood and compute AIC and BIC
mod3m@logLik
( AIC <- -2*mod3m@logLik+2*mod3m@nest )
( BIC <- -2*mod3m@logLik+log(mod3m@Data$N)*mod3m@nest )
# compare with information criteria from prob.guttman
mod3$ic
# model fit in mirt
mirt::M2(mod3m)
# extract coefficients
( cmod3m <- mirt.wrapper.coef(mod3m) )
# compare estimated distributions
round( cbind( "sirt" = mod3$trait$prob , "mirt" = mod3m@Prior[[1]] ) , 5 )
##      sirt  mirt
## [1,] 0.13709 0.13765
## [2,] 0.30266 0.30303
## [3,] 0.15239 0.15085
## [4,] 0.40786 0.40846
# compare estimated item parameters
ipars <- data.frame( "guess.sirt" = mod3$item$guess ,
  "guess.mirt" = plogis( cmod3m$coef$d ) )
ipars$slip.sirt <- mod3$item$slip

```

```

ipars$slip.mirt <- 1-plogis( rowSums(cmod3m$coef[ , c("a1","a2","a3","d") ] ) )
round( ipars , 4 )
##      guess.sirt guess.mirt slip.sirt slip.mirt
## 1      0.7810      0.7804      0.1383      0.1382
## 2      0.4513      0.4517      0.0373      0.0368
## 3      0.3203      0.3200      0.0747      0.0751
## 4      0.3009      0.3007      0.3082      0.3087
## 5      0.5776      0.5779      0.1800      0.1798
## 6      0.3758      0.3759      0.3047      0.3051
## 7      0.7262      0.7259      0.0625      0.0623
## [...]

###
# Model 4: Monotone item response function estimated in mirt

# define four ordered latent classes
Theta <- scan(nlines=1)
      0 0 0      1 0 0      1 1 0      1 1 1
Theta <- matrix( Theta , nrow=4 , ncol=3,byrow=TRUE)

# define mirt model
I <- ncol(dat) # I = 12
mirtmodel <- mirt::mirt.model("
      # specify factors for each item level
      C1 = 1-12
      C2 = 1-12
      C3 = 1-12
      ")

# get initial parameter values
mod.pars <- mirt::mirt(dat, model=mirtmodel , pars = "values")
# redefine initial parameter values
mod.pars[ mod.pars$name == "d" ,"value" ] <- -1
mod.pars[ mod.pars$name %in% paste0("a",1:3) & mod.pars$est      ,"value" ] <- .6
# set lower bound to zero to ensure monotonicity
mod.pars[ mod.pars$name %in% paste0("a",1:3)      ,"lbound" ] <- 0
mod.pars
# estimate model in mirt
mod4 <- mirt::mirt(dat, mirtmodel , pars = mod.pars , verbose=TRUE ,
      technical = list( customTheta=Theta , customPriorFun = lca_prior ) )
# correct number of estimated parameters
mod4@nest <- as.integer(sum(mod.pars$est) + nrow(Theta)-1 )
# extract coefficients
cmod4 <- mirt.wrapper.coef(mod4)
cmod4
# compute item response functions
cmod4c <- cmod4$coef[ , c("d" , "a1" , "a2" , "a3" ) ]
probs4 <- t( apply( cmod4c , 1 , FUN = function(l1){
      plogis(cumsum(as.numeric(l1))) } ) )
matplot( 1:4 , t(probs4) , type="b" , pch=1:I)

## End(Not run)

```



**Description**

This function calculates integration nodes based on the multivariate standard normal distribution.

**Usage**

```
qmc.nodes(snodes, ndim)
```

**Arguments**

snodes	Number of integration nodes
ndim	Number of dimensions

**Value**

theta	A matrix of integration points
-------	--------------------------------

**Note**

This function uses the **sfsmisc** package.

**Author(s)**

Alexander Robitzsch

**References**

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

**Examples**

```
## some toy examples

# 5 nodes on one dimension
qmc.nodes( snodes=5 , ndim=1 )
##           [,1]
## [1,]  0.0000000
## [2,] -0.3863753
## [3,]  0.8409238
## [4,] -0.8426682
## [5,]  0.3850568

# 7 nodes on two dimensions
qmc.nodes( snodes =7 , ndim=2 )
##           [,1]           [,2]
```

```
## [1,] 0.00000000 -0.43072730
## [2,] -0.38637529 0.79736332
## [3,] 0.84092380 -1.73230641
## [4,] -0.84266815 -0.03840544
## [5,] 0.38505683 1.51466109
## [6,] -0.00122394 -0.86704605
## [7,] 1.35539115 0.33491073
```

---

R2conquest

*Running ConQuest From Within R*


---

### Description

The function R2conquest runs the IRT software ConQuest (Wu, Adams, Wilson & Haldane, 2007) from within R.

Other functions are utility functions for reading item parameters, plausible values or person-item maps.

### Usage

```
R2conquest(dat, path.conquest, conquest.name="console", converge=0.001,
  deviancechange=1e-04, iter=800, nodes=20, minnode=-6 , maxnode=6,
  show.conquestoutput=FALSE, name="rasch", pid=1:(nrow(dat)), wgt=NULL, X=NULL,
  set.constraints=NULL, model="item", regression=NULL,
  itemcodes=seq(0,max(dat,na.rm=TRUE)), constraints=NULL, digits=5, onl syntax=FALSE,
  qmatrix=NULL, import.regression=NULL, anchor.regression=NULL,
  anchor.covariance=NULL, pv=TRUE, designmatrix=NULL, only.calibration=FALSE,
  init_parameters=NULL, n_plausible=10, persons.elim=TRUE, est.wle=TRUE,
  save.bat=TRUE , use.bat=FALSE , read.output=TRUE , ignore.pid=FALSE)
```

```
rasch.conquest(dat, path.conquest, conquest.name="console", converge=0.001,
  deviancechange=1e-04, iter=800, nodes=20, minnode=-6 , maxnode=6,
  show.conquestoutput=FALSE, name="rasch", pid=1:(nrow(dat)), wgt=NULL, X=NULL,
  set.constraints=NULL, model="item", regression=NULL,
  itemcodes=seq(0,max(dat,na.rm=TRUE)), constraints=NULL, digits=5, onl syntax=FALSE,
  qmatrix=NULL, import.regression=NULL, anchor.regression=NULL,
  anchor.covariance=NULL, pv=TRUE, designmatrix=NULL, only.calibration=FALSE,
  init_parameters=NULL, n_plausible=10, persons.elim=TRUE, est.wle=TRUE,
  save.bat=TRUE , use.bat=FALSE , read.output=TRUE , ignore.pid=FALSE)
```

```
## S3 method for class 'R2conquest'
summary(object, ...)
## S3 method for class 'rasch.conquest'
summary(object, ...)
```

```
# read all terms in a show file or only some terms
read.show(showfile)
```

```

read.show.term(showfile, term)

# read regression parameters in a show file
read.show.regression(showfile)

# read unidimensional plausible values form a pv file
read.pv(pvfile, npv = 5)
# read multidimensional plausible values
read.multidimpv(pvfile, ndim, npv = 5)

# read person-item map
read.pimap(showfile)

```

### Arguments

<code>dat</code>	Data frame of item responses
<code>path.conquest</code>	Directory where the ConQuest executable file is located
<code>conquest.name</code>	Name of the ConQuest executable.
<code>converge</code>	Maximal change in parameters
<code>deviancechange</code>	Maximal change in deviance
<code>iter</code>	Maximum number of iterations
<code>nodes</code>	Number of nodes for integration
<code>minnode</code>	Minimum value of discrete grid of $\theta$ nodes
<code>maxnode</code>	Maximum value of discrete grid of $\theta$ nodes
<code>show.conquestoutput</code>	Show ConQuest run log file on console?
<code>name</code>	Name of the output files. The default is 'rasch'.
<code>pid</code>	Person identifier
<code>wgt</code>	Vector of person weights
<code>X</code>	Matrix of covariates for the latent regression model (e.g. gender, socioeconomic status, ..) or for the item design (e.g. raters, booklets, ...)
<code>set.constraints</code>	This is the <code>set.constraints</code> in ConQuest. It can be "cases" (constraint for persons), "items" or "none"
<code>model</code>	Definition model statement. It can be for example "item+item*step" or "item+booklet+rater"
<code>regression</code>	The ConQuest regression statement (for example "gender+status")
<code>itemcodes</code>	Vector of valid codes for item responses. E.g. for partial credit data with at most 3 points it must be <code>c(0, 1, 2, 3)</code> .
<code>constraints</code>	Matrix of item parameter constraints. 1st column: Item names, 2nd column: Item parameters. It only works correctly for dichotomous data.
<code>digits</code>	Number of digits for covariates in the latent regression model
<code>onlysyntax</code>	Should only be ConQuest syntax generated?

<code>qmatrix</code>	Matrix of item loadings on dimensions in a multidimensional IRT model
<code>import.regression</code>	Name of an file with initial covariance parameters (follow the ConQuest specification rules!)
<code>anchor.regression</code>	Name of an file with anchored regression parameters
<code>anchor.covariance</code>	Name of an file with anchored covariance parameters (follow the ConQuest specification rules!)
<code>pv</code>	Draw plausible values?
<code>designmatrix</code>	Design matrix for item parameters (see the ConQuest manual)
<code>only.calibration</code>	Estimate only item parameters and not person parameters (no WLEs or plausible values are estimated)?
<code>init_parameters</code>	Name of an file with initial item parameters (follow the ConQuest specification rules!)
<code>n_plausible</code>	Number of plausible values
<code>persons.elim</code>	Eliminate persons with only missing item responses?
<code>est.wle</code>	Estimate weighted likelihood estimate?
<code>save.bat</code>	Save bat file?
<code>use.bat</code>	Run ConQuest from within R due a direct call via the system command ( <code>use.bat=FALSE</code> ) or via a system call of a bat file in the working directory ( <code>use.bat=TRUE</code> )
<code>read.output</code>	Should ConQuest output files be processed? Default is TRUE.
<code>ignore.pid</code>	Logical indicating whether person identifiers ( <code>pid</code> ) should be processed in ConQuest input syntax.
<code>object</code>	Object of class R2conquest
<code>showfile</code>	A ConQuest show file ( <code>shw</code> file)
<code>term</code>	Name of the term to be extracted in the show file
<code>pvfile</code>	File with plausible values
<code>ndim</code>	Number of dimensions
<code>npv</code>	Number of plausible values
<code>...</code>	Further arguments to be passed

### Details

Consult the ConQuest manual (Wu et al., 2007) for specification details.

**Value**

A list with several entries

<code>item</code>	Data frame with item parameters and item statistics
<code>person</code>	Data frame with person parameters
<code>shw.itemparameter</code>	ConQuest output table for item parameters
<code>shw.regrparameter</code>	ConQuest output table for regression parameters
<code>...</code>	More values

**Author(s)**

Alexander Robitzsch

**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 also the **eat** package (<http://r-forge.r-project.org/projects/eat/>) for elaborate functionality of using ConQuest from within R.

See also the **TAM** package for similar (and even extended) functionality for specifying item response models.

**Examples**

```
## Not run:
# define ConQuest path
path.conquest <- "C:/Conquest/"

#####
# EXAMPLE 1: Dichotomous data (data.pisaMath)
#####
library(sirt)
data(data.pisaMath)
dat <- data.pisaMath$data

# select items
items <- colnames(dat)[ which( substring( colnames(dat) , 1 , 1)=="M" ) ]

###
# Model 11: Rasch model
mod11 <- R2conquest(dat=dat[,items] , path.conquest=path.conquest ,
                    pid=dat$idstud , name="mod11")
summary(mod11)
# read show file
```

```

shw11 <- read.show( "mod11.shw" )
# read person-item map
pi11 <- read.pimap(showfile="mod11.shw")

####
# Model 12: Rasch model with fixed item difficulties (from Model 1)
mod12 <- R2conquest(dat=dat[,items] , path.conquest=path.conquest ,
                    pid=dat$idstud , constraints = mod11$item[ , c("item","itemdiff")] ,
                    name="mod12")
summary(mod12)

####
# Model 13: Latent regression model with predictors female, hisei and migra
mod13a <- R2conquest(dat=dat[,items] , path.conquest=path.conquest ,
                    pid=dat$idstud , X = dat[ , c("female" , "hisei" , "migra") ] ,
                    name="mod13a")
summary(mod13a)

# latent regression with a subset of predictors
mod13b <- R2conquest(dat=dat[,items] , path.conquest=path.conquest ,
                    pid=dat$idstud , X = dat[ , c("female" , "hisei" , "migra") ] ,
                    regression= "hisei migra" , name="mod13b")

####
# Model 14: Differential item functioning (female)
mod14 <- R2conquest(dat=dat[,items] , path.conquest=path.conquest ,
                    pid=dat$idstud , X = dat[ , c("female") ] , drop=FALSE] ,
                    model="item+female+item*female" , regression="" , name="mod14")

#####
# EXAMPLE 2: Polytomous data (data.Students)
#####
library(CDM)
data(data.Students)
dat <- data.Students

# select items
items <- grep.vec( "act" , colnames(dat) )$x

####
# Model 21: Partial credit model
mod21 <- R2conquest(dat=dat[,items] , path.conquest=path.conquest ,
                    model="item+item*step" , name="mod21")

####
# Model 22: Rating scale model
mod22 <- R2conquest(dat=dat[,items] , path.conquest=path.conquest ,
                    model="item+step" , name="mod22")

####
# Model 23: Multidimensional model
items <- grep.vec( c("act" , "sc" ) , colnames(dat) , "OR" )$x
qmatrix <- matrix( 0 , nrow=length(items) , 2 )

```

```

qmatrix[1:5,1] <- 1
qmatrix[6:9,2] <- 1
mod23 <- R2conquest(dat=dat[,items] , path.conquest=path.conquest ,
                    model="item+item*step" , qmatrix=qmatrix , name="mod23")

#####
# EXAMPLE 3: Multi facet models (data.ratings1)
#####
library(sirt)
data(data.ratings1)
dat <- data.ratings1

items <- paste0("k",1:5)

# use numeric rater ID's
raters <- as.numeric( substring( paste( dat$rater ) , 3 ) )

###
# Model 31: Rater model 'item+item*step+rater'
mod31 <- R2conquest(dat=dat[,items] , path.conquest=path.conquest ,
                    itemcodes= 0:3 , model="item+item*step+rater" ,
                    pid=dat$idstud , X=data.frame("rater"=raters) ,
                    regression="" , name="mod31")

###
# Model 32: Rater model 'item+item*step+rater+item*rater'
mod32 <- R2conquest(dat=dat[,items] , path.conquest=path.conquest ,
                    model="item+item*step+rater+item*rater" ,
                    pid=dat$idstud , X=data.frame("rater"=raters) ,
                    regression="" , name="mod32")

## End(Not run)

```

---

## Description

This function enables the estimation of a NOHARM analysis (Fraser & McDonald, 1988; McDonald, 1982a, 1982b, 1997) from within R. NOHARM estimates a compensatory multidimensional factor analysis for dichotomous response data. Arguments of this function strictly follow the rules of the NOHARM manual (see Fraser & McDonald, 2012).

## Usage

```

R2noharm(dat=NULL,pm=NULL , n=NULL , model.type, weights=NULL , dimensions = NULL,
          guesses = NULL , noharm.path, F.pattern = NULL, F.init = NULL,
          P.pattern = NULL, P.init = NULL, digits.pm = 4, writename = NULL,
          display.fit = 5, dec = ".", display = TRUE)

```

```
## S3 method for class 'R2noharm'
summary(object, logfile=NULL , ...)
```

### Arguments

<code>dat</code>	An $N \times I$ data frame of item responses for $N$ subjects and $I$ items
<code>pm</code>	A matrix or a vector containing product-moment correlations
<code>n</code>	Sample size. This value must only be included if <code>pm</code> is provided.
<code>model.type</code>	Can be "EFA" (exploratory factor analysis) or "CFA" (confirmatory factor analysis).
<code>weights</code>	Optional vector of student weights
<code>dimensions</code>	Number of dimensions in exploratory factor analysis
<code>guesses</code>	An optional vector of fixed guessing parameters of length $I$ . In case of the default NULL, all guessing parameters are set to zero.
<code>noharm.path</code>	Local path where the NOHARM 4 command line 64-bit version is located.
<code>F.pattern</code>	Pattern matrix for $F$ ( $I \times D$ )
<code>F.init</code>	Initial matrix for $F$ ( $I \times D$ )
<code>P.pattern</code>	Pattern matrix for $P$ ( $D \times D$ )
<code>P.init</code>	Initial matrix for $P$ ( $D \times D$ )
<code>digits.pm</code>	Number of digits after decimal separator which are used for estimation
<code>writename</code>	Name for NOHARM input and output files
<code>display.fit</code>	How many digits (after decimal separator) should be used for printing results on the R console?
<code>dec</code>	Decimal separator (". " or ", ")
<code>display</code>	Display output?
<code>object</code>	Object of class R2noharm
<code>logfile</code>	File name if the summary should be sinked into a file
<code>...</code>	Further arguments to be passed

### Details

NOHARM estimates a multidimensional compensatory item response model with the probit link function  $\Phi$ . For item responses  $X_{pi}$  of person  $p$  on item  $i$  the model equation is defined as

$$P(X_{pi} = 1 | \theta_p) = c_i + (1 - c_i) \Phi(f_{i0} + f_{i1}\theta_{p1} + \dots + f_{iD}\theta_{pD})$$

where  $F = (f_{id})$  is a loading matrix and  $P$  the covariance matrix of  $\theta_p$ . The guessing parameters  $c_i$  must be provided as fixed values.

For the definition of  $F$  and  $P$  matrices, please consult the NOHARM manual.

This function needs the 64-bit command line version which can be downloaded at <http://noharm.niagararesearch.ca/nh4cldl.f>



**Value**

A list with following entries

tanaka	Tanaka index
rmsr	RMSR statistic
N.itempair	Sample sizes of pairwise item observations
pm	Product moment matrix
weights	Used student weights
guesses	Fixed guessing parameters
residuals	Residual covariance matrix
final.constants	Vector of final constants
thresholds	Threshold parameters
uniquenesses	Item uniquenesses
loadings.theta	Matrix of loadings in theta parametrization (common factor parametrization)
factor.cor	Covariance matrix of factors
difficulties	Item difficulties (for unidimensional models)
discriminations	Item discriminations (for unidimensional models)
loadings	Loading matrix (latent trait parametrization)
model.type	Used model type
Nobs	Number of observations
Nitems	Number of items
modtype	Model type according to the NOHARM specification (see NOHARM manual)
F.init	Initial loading matrix for $F$
F.pattern	Pattern loading matrix for $F$
P.init	Initial covariance matrix for $P$
P.pattern	Pattern covariance matrix for $P$
dat	Original data frame
systemtime	System time
noharm.path	Used NOHARM directory
digits.pm	Number of digits in product moment matrix
dec	Used decimal symbol
display.fit	Number of digits for fit display
dimensions	Number of dimensions
chisquare	Statistic $\chi^2$
Nestpars	Number of estimated parameters
df	Degrees of freedom
chisquare_df	Ratio $\chi^2/df$
rmsea	RMSEA statistic
p.chisquare	Significance for $\chi^2$ statistic

**Note**

Possible errors often occur due to wrong dec specification.

**Author(s)**

Alexander Robitzsch

**References**

- Fraser, C., & McDonald, R. P. (1988). NOHARM: Least squares item factor analysis. *Multivariate Behavioral Research*, **23**, 267-269.
- Fraser, C., & McDonald, R. P. (2012). *NOHARM 4 Manual*. <http://noharm.niagararesearch.ca/nh4man/nhman.html>.
- McDonald, R. P. (1982a). Linear versus models in item response theory. *Applied Psychological Measurement*, **6**, 379-396.
- McDonald, R. P. (1982b). *Unidimensional and multidimensional models for item response theory*. I.R.T., C.A.T. conference, Minneapolis, 1982, Proceedings.
- McDonald, R. P. (1997). Normal-ogive multidimensional model. In W. van der Linden & R. K. Hambleton (1997): *Handbook of modern item response theory* (pp. 257-269). New York: Springer.

**See Also**

- For estimating standard errors see [R2noharm.jackknife](#).
- For EAP person parameter estimates see [R2noharm.EAP](#).
- For an R implementation of the NOHARM model see [noharm.sirt](#).

**Examples**

```
## Not run:
#####
# EXAMPLE 1: Data data.noharm18 with 18 items
#####

# load data
data( data.noharm18 )
dat <- data.noharm18
I <- ncol(dat) # number of items

# locate noharm.path
noharm.path <- "c:/NOHARM"

#####
# Model 1: 1-dimensional Rasch model (1-PL model)

# estimate one factor variance
P.pattern <- matrix( 1 , ncol=1 , nrow=1 )
P.init <- P.pattern
# fix all entries in the loading matrix to 1
F.pattern <- matrix( 0 , nrow=I , ncol=1 )
F.init <- 1 + 0*F.pattern      #
```

```

# estimate model
mod <- R2noharm( dat = dat , model.type="CFA" ,
                F.pattern = F.pattern , F.init = F.init , P.pattern = P.pattern ,
                P.init = P.init , writename = "ex1__1dim_1pl" ,
                noharm.path = noharm.path , dec = "," )
# summary
summary(mod , logfile="ex1__1dim_1pl__SUMMARY")
# jackknife
jmod <- R2noharm.jackknife( mod , jackunits = 20 )
summary(jmod, logfile="ex1__1dim_1pl__JACKKNIFE")
# compute factor scores (EAPs)
emod <- R2noharm.EAP(mod)

#*****-----
# Model 1b: Include student weights in estimation
N <- nrow(dat)
weights <- runif( N , 1 , 5 )
mod1b <- R2noharm( dat = dat , model.type="CFA" , weights=weights ,
                  F.pattern = F.pattern , F.init = F.init , P.pattern = P.pattern ,
                  P.init = P.init , writename = "ex1__1dim_1pl_w" ,
                  noharm.path = noharm.path , dec = "," )
summary(mod1b)

#*****
# Model 2: 1-dimensional 2PL Model

# set trait variance equal to 1
P.pattern <- matrix( 0 , ncol=1 , nrow=1 )
P.init <- 1+0*P.pattern
# loading matrix
F.pattern <- matrix( 1 , nrow=I , ncol=1 )
F.init <- 1 + 0*F.pattern

mod <- R2noharm( dat = dat , model.type="CFA" ,
                F.pattern = F.pattern , F.init = F.init , P.pattern = P.pattern ,
                P.init = P.init , writename = "ex2__1dim_2pl" ,
                noharm.path = noharm.path , dec = "," )

summary(mod)
jmod <- R2noharm.jackknife( mod , jackunits = 20 )
summary(jmod)

#*****
# Model 3: 1-dimensional 3PL Model with fixed guessing parameters

# set trait variance equal to 1
P.pattern <- matrix( 0 , ncol=1 , nrow=1 )
P.init <- 1+0*P.pattern
# loading matrix
F.pattern <- matrix( 1 , nrow=I , ncol=1 )
F.init <- 1 + 0*F.pattern      #
# fix guessing parameters equal to .2 (for all items)
guesses <- rep( .1 , I )

```

```

mod <- R2noharm( dat = dat , model.type="CFA" ,
  F.pattern = F.pattern , F.init = F.init , P.pattern = P.pattern ,
  P.init = P.init , guesses = guesses ,
  writename = "ex3__1dim_3pl" , noharm.path = noharm.path , dec="," )
summary(mod)
jmod <- R2noharm.jackknife( mod , jackunits = 20 )
summary(jmod)

#*****
# Model 4: 3-dimensional Rasch model

# estimate one factor variance
P.pattern <- matrix( 1 , ncol=3 , nrow=3 )
P.init <- .8*P.pattern
diag(P.init) <- 1
# fix all entries in the loading matrix to 1
F.init <- F.pattern <- matrix( 0 , nrow=I , ncol=3 )
F.init[1:6,1] <- 1
F.init[7:12,2] <- 1
F.init[13:18,3] <- 1

mod <- R2noharm( dat = dat , model.type="CFA" ,
  F.pattern = F.pattern , F.init = F.init , P.pattern = P.pattern ,
  P.init = P.init , writename = "ex4__3dim_1pl" ,
  noharm.path = noharm.path , dec =" ," )
# write output from R console in a file
summary(mod , logfile="ex4__3dim_1pl__SUMMARY.Rout")

jmod <- R2noharm.jackknife( mod , jackunits = 20 )
summary(jmod)

# extract factor scores
emod <- R2noharm.EAP(mod)

#*****
# Model 5: 3-dimensional 2PL model

# estimate one factor variance
P.pattern <- matrix( 1 , ncol=3 , nrow=3 )
P.init <- .8*P.pattern
diag(P.init) <- 0
# fix all entries in the loading matrix to 1
F.pattern <- matrix( 0 , nrow=I , ncol=3 )
F.pattern[1:6,1] <- 1
F.pattern[7:12,2] <- 1
F.pattern[13:18,3] <- 1
F.init <- F.pattern

mod <- R2noharm( dat = dat , model.type="CFA" ,
  F.pattern = F.pattern , F.init = F.init , P.pattern = P.pattern ,
  P.init = P.init , writename = "ex5__3dim_2pl" ,
  noharm.path = noharm.path , dec = "," )

```

```

summary(mod)
# use 50 jackknife units with 4 persons within a unit
jmod <- R2noharm.jackknife( mod , jackunits = seq( 1:50 , each = 4 ) )
summary(jmod)

#####
# Model 6: Exploratory Factor Analysis with 3 factors

mod <- R2noharm( dat = dat , model.type="EFA" , dimensions = 3 ,
                writename = "ex6__3dim_efa" , noharm.path = noharm.path , dec = " ,")
summary(mod)

jmod <- R2noharm.jackknife( mod , jackunits = 20 )

#####
# EXAMPLE 2: NOHARM manual Example A
#####

# See NOHARM manual: http://noharm.niagararesearch.ca/nh4man/nhman.html
# The following text and data is copied from this manual.
#
# In the first example, we demonstrate how to prepare the input for a 2-dimensional
# model using exploratory analysis. Data from a 9 item test were collected from
# 200 students and the 9x9 product-moment matrix of the responses was computed.
#
# Our hypothesis is for a 2-dimensional model with no guessing,
# i.e., all guesses are equal to zero. However, because we are unsure of any
# particular pattern for matrix F, we wish to prescribe an exploratory analysis, i.e.,
# set EX = 1. Also, we will content ourselves with letting the program supply all
# initial values.
#
# We would like both the sample product-moment matrix and the residual matrix to
# be included in the output.

# scan product-moment matrix copied from the NOHARM manual
pm <- scan()
  0.8967
  0.2278 0.2356
  0.6857 0.2061 0.7459
  0.8146 0.2310 0.6873 0.8905
  0.4505 0.1147 0.3729 0.4443 0.5000
  0.7860 0.2080 0.6542 0.7791 0.4624 0.8723
  0.2614 0.0612 0.2140 0.2554 0.1914 0.2800 0.2907
  0.7549 0.1878 0.6236 0.7465 0.4505 0.7590 0.2756 0.8442
  0.6191 0.1588 0.5131 0.6116 0.3845 0.6302 0.2454 0.6129 0.6879

ex2 <- R2noharm( pm= pm , n =200 , model.type="EFA" , dimensions=2 ,
                noharm.path=noharm.path , writename="ex2_noharmExA" , dec=" ,")
summary(ex2)

#####
# EXAMPLE 3: NOHARM manual Example B
#####

```

```

# See NOHARM manual: http://noharm.niagararesearch.ca/nh4man/nhman.html
# The following text and data is copied from this manual.

# Suppose we have the product-moment matrix of data from 125 students on 9 items.
# Our hypothesis is for 2 dimensions with simple structure. In this case,
# items 1 to 5 have coefficients of theta which are to be estimated for one
# latent trait but are to be fixed at zero for the other one.
# For the latent trait for which items 1 to 5 have zero coefficients,
# items 6 to 9 have coefficients which are to be estimated. For the other
# latent trait, items 6 to 9 will have zero coefficients.
# We also wish to estimate the correlation between the latent traits,
# so we prescribe P as a 2x2 correlation matrix.
#
# Our hypothesis prescribes that there was no guessing involved, i.e.,
# all guesses are equal to zero. For demonstration purposes,
# let us not have the program print out the sample product-moment matrix.
# Also let us not supply any starting values but, rather, use the defaults
# supplied by the program.

pm <- scan()
  0.930
  0.762 0.797
  0.541 0.496 0.560
  0.352 0.321 0.261 0.366
  0.205 0.181 0.149 0.110 0.214
  0.858 0.747 0.521 0.336 0.203 0.918
  0.773 0.667 0.465 0.308 0.184 0.775 0.820
  0.547 0.474 0.347 0.233 0.132 0.563 0.524 0.579
  0.329 0.290 0.190 0.140 0.087 0.333 0.308 0.252 0.348

I <- 9 # number of items
# define loading matrix
F.pattern <- matrix(0,I,2)
F.pattern[1:5,1] <- 1
F.pattern[6:9,2] <- 1
F.init <- F.pattern
# define covariance matrix
P.pattern <- matrix(1,2,2)
diag(P.pattern) <- 0
P.init <- 1+P.pattern

ex3 <- R2noharm( pm=pm , n=125, , model.type="CFA" ,
  F.pattern = F.pattern , F.init = F.init , P.pattern = P.pattern ,
  P.init = P.init , writename = "ex3_noharmExB" ,
  noharm.path = noharm.path , dec = "," )
summary(ex3)

#####
# EXAMPLE 4: NOHARM manual Example C
#####

data(data.noharmExC)

```

```

# See NOHARM manual: http://noharm.niagararesearch.ca/nh4man/nhman.html
# The following text and data is copied from this manual.

# In this example, suppose that from 300 respondents we have item
# responses scored dichotomously, 1 or 0, for 8 items.
#
# Our hypothesis is for a unidimensional model where all eight items
# have coefficients of theta which are to be estimated.
# Suppose that since the items were multiple choice with 5 options each,
# we set the fixed guesses all to 0.2 (not necessarily good reasoning!)
#
# Let's supply initial values for the coefficients of theta (F matrix)
# as .75 for items 1 to 4 and .6 for items 5 to 8.

I <- 8
guesses <- rep(.2,I)
F.pattern <- matrix(1,I,1)
F.init <- F.pattern
F.init[1:4,1] <- .75
F.init[5:8,1] <- .6
P.pattern <- matrix(0,1,1)
P.init <- 1 + 0 * P.pattern

ex4 <- R2noharm( dat=data.noharmExC , , model.type="CFA" ,
                guesses=guesses , F.pattern = F.pattern , F.init = F.init ,
                P.pattern = P.pattern, P.init = P.init , writename = "ex3_noharmExC" ,
                noharm.path = noharm.path , dec ="," )
summary(ex4)

# modify F pattern matrix
# f11 = f51 (since both have equal pattern values of 2),
# f21 = f61 (since both have equal pattern values of 3),
# f31 = f71 (since both have equal pattern values of 4),
# f41 = f81 (since both have equal pattern values of 5).
F.pattern[ c(1,5) ] <- 2
F.pattern[ c(2,6) ] <- 3
F.pattern[ c(3,7) ] <- 4
F.pattern[ c(4,8) ] <- 5
F.init <- .5+0*F.init

ex4a <- R2noharm( dat=data.noharmExC , , model.type="CFA" ,
                guesses=guesses , F.pattern = F.pattern , F.init = F.init ,
                P.pattern = P.pattern, P.init = P.init , writename = "ex3_noharmExC1" ,
                noharm.path = noharm.path , dec ="," )
summary(ex4a)

## End(Not run)

```

**Description**

This function performs EAP factor score estimation of an item response model estimated with NOHARM.

**Usage**

```
R2noharm.EAP(noharmobj, theta.k = seq(-6, 6, len = 21) , print.output=TRUE)
```

**Arguments**

noharmobj	Object of class R2noharm or noharm.sirt
theta.k	Vector of discretized theta values on which the posterior is evaluated. This vector applies to all dimensions.
print.output	An optional logical indicating whether output should be displayed at the console

**Value**

A list with following entries

person	Data frame of person parameter EAP estimates and their corresponding standard errors
theta	Grid of multidimensional theta values where the posterior is evaluated.
posterior	Individual posterior distribution evaluated at theta
like	Individual likelihood
EAP.rel	EAP reliabilities of all dimensions
probs	Item response probabilities evaluated at theta

**Author(s)**

Alexander Robitzsch

**See Also**

For examples see [R2noharm](#) and [noharm.sirt](#).

---

R2noharm.jackknife      *Jackknife Estimation of NOHARM Analysis*

---

**Description**

This function performs a jackknife estimation of NOHARM analysis to get standard errors based on a replication method (see Christoffersson, 1977).



**Usage**

```
R2noharm.jackknife(object, jackunits = NULL)
```

```
## S3 method for class 'R2noharm.jackknife'
summary(object, logfile=NULL , ...)
```

**Arguments**

object	Object of class R2noharm
jackunits	A vector of integers or a number. If it is a number, then it refers to the number of jackknife units. If it is a vector of integers, then this vector defines the allocation of persons jackknife units. Integers corresponds to row indexes in the data set.
logfile	File name if the summary should be sinked into a file
...	Further arguments to be passed

**Value**

A list of lists with following entries:

partable	Data frame with parameters
se.pars	List of estimated standard errors for all parameter estimates: tanaka.stat, rmsr.stat, rmsea.stat, chisquare_df.stat, thresholds.stat, final.constants.stat, uniquenesses.stat, factor.cor.stat, loadings.stat, loadings.theta.stat
jackknife.pars	List with obtained results by jackknifing for all parameters: j.tanaka, j.rmsr, rmsea, chisquare_df, j.pm, j.thresholds, j.factor.cor, j.loadings, j.loadings.theta
u.jackunits	Unique jackknife elements

**Author(s)**

Alexander Robitzsch

**References**

Christoffersson, A. (1977). Two-step weighted least squares factor analysis of dichotomized variables. *Psychometrika*, **42**, 433-438.

**See Also**

[R2noharm](#)

---

 rasch.copula2

 Multidimensional IRT Copula Model
 

---

### Description

This function handles local dependence by specifying copulas for residuals in multidimensional item response models for dichotomous item responses (Braeken, 2011; Braeken, Tuerlinckx & de Boeck, 2007). Estimation is allowed for item difficulties, item slopes and a generalized logistic link function (Stukel, 1988).

The function `rasch.copula3` allows the estimation of multidimensional models while `rasch.copula2` only handles unidimensional models.

### Usage

```
rasch.copula2(dat, itemcluster, copula.type = "bound.mixt" ,
  progress = TRUE, mmliter = 1000, delta = NULL,
  theta.k = seq(-4, 4, len = 21), alpha1 = 0, alpha2 = 0,
  numdiff.parm = 1e-06, est.b = seq(1, ncol(dat)),
  est.a = rep(1, ncol(dat)), est.delta = NULL, b.init = NULL , a.init = NULL ,
  est.alpha = FALSE, glob.conv = 0.0001, alpha.conv = 1e-04, conv1 = 0.001,
  dev.crit=.2 , increment.factor=1.01)
```

```
rasch.copula3(dat, itemcluster, dims=NULL , copula.type = "bound.mixt" ,
  progress = TRUE, mmliter = 1000, delta = NULL,
  theta.k = seq(-4, 4, len = 21), alpha1 = 0, alpha2 = 0,
  numdiff.parm = 1e-06, est.b = seq(1, ncol(dat)),
  est.a = rep(1, ncol(dat)), est.delta = NULL, b.init = NULL , a.init = NULL ,
  est.alpha = FALSE, glob.conv = 0.0001, alpha.conv = 1e-04, conv1 = 0.001,
  dev.crit=.2 , rho.init=.5 , increment.factor=1.01)
```

```
## S3 method for class 'rasch.copula2'
summary(object,...)
```

```
## S3 method for class 'rasch.copula3'
summary(object,...)
```

```
## S3 method for class 'rasch.copula2'
anova(object,...)
```

```
## S3 method for class 'rasch.copula3'
anova(object,...)
```

```
## S3 method for class 'rasch.copula2'
logLik(object,...)
```

```
## S3 method for class 'rasch.copula3'
logLik(object,...)
```

```
## S3 method for class 'rasch.copula2'
```

```

IRT.likelihood(object,...)
## S3 method for class 'rasch.copula3'
IRT.likelihood(object,...)

## S3 method for class 'rasch.copula2'
IRT.posterior(object,...)
## S3 method for class 'rasch.copula3'
IRT.posterior(object,...)

```

### Arguments

<code>dat</code>	An $N \times I$ data frame. Cases with only missing responses are removed from the analysis.
<code>itemcluster</code>	An integer vector of length $I$ (number of items). Items with the same integers define a joint item cluster of (positively) locally dependent items. Values of zero indicate that the corresponding item is not included in any item cluster of dependent responses.
<code>dims</code>	A vector indicating to which dimension an item is allocated. The default is that all items load on the first dimension.
<code>copula.type</code>	A character or a vector containing one of the following copula types: <code>bound.mixt</code> (boundary mixture copula), <code>cook.johnson</code> (Cook-Johnson copula) or <code>frank</code> (Frank copula) (see Braeken, 2011). The vector <code>copula.type</code> must match the number of different itemclusters. For every itemcluster, a different copula type may be specified (see Examples).
<code>progress</code>	Print progress? Default is TRUE.
<code>mmliter</code>	Maximum number of iterations.
<code>delta</code>	An optional vector of starting values for the dependency parameter <code>delta</code> .
<code>theta.k</code>	Discretized trait distribution
<code>alpha1</code>	<code>alpha1</code> parameter in the generalized logistic item response model (Stukel, 1988). The default is 0 which leads together with <code>alpha2=0</code> to the logistic link function.
<code>alpha2</code>	<code>alpha2</code> parameter in the generalized logistic item response model
<code>numdiff.parm</code>	Parameter for numerical differentiation
<code>est.b</code>	Integer vector of item difficulties to be estimated
<code>est.a</code>	Integer vector of item discriminations to be estimated
<code>est.delta</code>	Integer vector of length <code>length(itemcluster)</code> . Nonzero integers correspond to <code>delta</code> parameters which are estimated. Equal integers indicate parameter equality constraints.
<code>b.init</code>	Initial $b$ parameters
<code>a.init</code>	Initial $a$ parameters
<code>est.alpha</code>	Should both alpha parameters be estimated? Default is FALSE.
<code>glob.conv</code>	Convergence criterion for all parameters
<code>alpha.conv</code>	Maximal change in alpha parameters for convergence
<code>conv1</code>	Maximal change in item parameters for convergence

<code>dev.crit</code>	Maximal change in the deviance. Default is .2.
<code>rho.init</code>	Initial value for off-diagonal elements in correlation matrix
<code>increment.factor</code>	A numeric value larger than one which controls the size of increments in iterations. To stabilize convergence, choose values 1.05 or 1.1 in some situations.
<code>object</code>	Object of class <code>rasch.copula2</code> or <code>rasch.copula3</code>
<code>...</code>	Further arguments to be passed

**Value**

A list with following entries

<code>N.itemclusters</code>	Number of item clusters
<code>item</code>	Estimated item parameters
<code>iter</code>	Number of iterations
<code>dev</code>	Deviance
<code>delta</code>	Estimated dependency parameters $\delta$
<code>b</code>	Estimated item difficulties
<code>a</code>	Estimated item slopes
<code>mu</code>	Mean
<code>sigma</code>	Standard deviation
<code>alpha1</code>	Parameter $\alpha_1$ in the generalized item response model
<code>alpha2</code>	Parameter $\alpha_2$ in the generalized item response model
<code>ic</code>	Information criteria
<code>theta.k</code>	Discretized ability distribution
<code>pi.k</code>	Fixed $\theta$ distribution
<code>deviance</code>	Deviance
<code>pattern</code>	Item response patterns with frequencies and posterior distribution
<code>person</code>	Data frame with person parameters
<code>datalist</code>	List of generated data frames during estimation
<code>EAP.rel</code>	Reliability of the EAP
<code>copula.type</code>	Type of copula
<code>summary.delta</code>	Summary for estimated $\delta$ parameters
<code>f.qk.yi</code>	Individual posterior
<code>f.yi.qk</code>	Individual likelihood
<code>...</code>	

**Author(s)**

Alexander Robitzsch

## References

- Braeken, J. (2011). A boundary mixture approach to violations of conditional independence. *Psychometrika*, **76**, 57-76.
- Braeken, J., & Tuerlinckx, F. (2009). Investigating latent constructs with item response models: A MATLAB IRTm toolbox. *Behavior Research Methods*, **41**, 1127-1137.
- Braeken, J., Tuerlinckx, F., & De Boeck, P. (2007). Copulas for residual dependencies. *Psychometrika*, **72**, 393-411.
- Stukel, T. A. (1988). Generalized logistic models. *Journal of the American Statistical Association*, **83**, 426-431.

## See Also

For a summary see [summary.rasch.copula2](#).

For simulating locally dependent item responses see [sim.rasch.dep](#).

Person parameters estimates are obtained by [person.parameter.rasch.copula](#).

See [rasch.mml2](#) for the generalized logistic link function.

See also Braeken and Tuerlinckx (2009) for alternative (and more expanded) copula models implemented in the MATLAB software. See <http://ppw.kuleuven.be/okp/software/irtm/>.

## Examples

```
#####
# EXAMPLE 1: Reading Data
#####

data(data.read)
dat <- data.read

# define item clusters
itemcluster <- rep( 1:3 , each=4 )

# estimate Copula model
mod1 <- rasch.copula2( dat=dat , itemcluster=itemcluster)

# estimate Rasch model
mod2 <- rasch.copula2( dat=dat , itemcluster=itemcluster ,
                      delta=rep(0,3) , est.delta=rep(0,3) )
summary(mod1)
summary(mod2)

## Not run:
#####
# SIMULATED EXAMPLE 2: 11 items nested within 2 item clusters (testlets)
#   with 2 resp. 3 dependent and 6 independent items
#####

set.seed(5698)
I <- 11                                # number of items
```

```

n <- 3000                                # number of persons
b <- seq(-2,2, len=I)                    # item difficulties
theta <- rnorm( n , sd = 1 )             # person abilities
# define item clusters
itemcluster <- rep(0,I)
itemcluster[ c(3,5) ] <- 1
itemcluster[ c(2,4,9) ] <- 2
# residual correlations
rho <- c( .7 , .5 )

# simulate data
dat <- sim.rasch.dep( theta , b , itemcluster , rho )
colnames(dat) <- paste("I" , seq(1,ncol(dat)) , sep="")

# estimate Rasch copula model
mod1 <- rasch.copula2( dat , itemcluster = itemcluster )
summary(mod1)

# both item clusters have Cook-Johnson copula as dependency
mod1c <- rasch.copula2( dat , itemcluster = itemcluster ,
  copula.type = "cook.johnson" )
summary(mod1c)

# first item boundary mixture and second item Cook-Johnson copula
mod1d <- rasch.copula2( dat , itemcluster = itemcluster ,
  copula.type = c( "bound.mixt" , "cook.johnson" ) )
summary(mod1d)

# compare result with Rasch model estimation in rasch.copula2
# delta must be set to zero
mod2 <- rasch.copula2( dat , itemcluster = itemcluster , delta = c(0,0) ,
  est.delta = c(0,0) )
summary(mod2)

#####
# SIMULATED EXAMPLE 3: 12 items nested within 3 item clusters (testlets)
# Cluster 1 -> Items 1-4; Cluster 2 -> Items 6-9; Cluster 3 -> Items 10-12
#####

set.seed(967)
I <- 12                                # number of items
n <- 450                                # number of persons
b <- seq(-2,2, len=I)                    # item difficulties
b <- sample(b)                          # sample item difficulties
theta <- rnorm( n , sd = 1 )             # person abilities
# itemcluster
itemcluster <- rep(0,I)
itemcluster[ 1:4 ] <- 1
itemcluster[ 6:9 ] <- 2
itemcluster[ 10:12 ] <- 3
# residual correlations
rho <- c( .35 , .25 , .30 )

```

```

# simulate data
dat <- sim.rasch.dep( theta , b , itemcluster , rho )
colnames(dat) <- paste("I" , seq(1,ncol(dat)) , sep="")

# estimate Rasch copula model
mod1 <- rasch.copula2( dat , itemcluster = itemcluster )
summary(mod1)

# person parameter estimation assuming the Rasch copula model
pmod1 <- person.parameter.rasch.copula(raschcopula.object = mod1 )

# Rasch model estimation
mod2 <- rasch.copula2( dat , itemcluster = itemcluster ,
                      delta = rep(0,3) , est.delta = rep(0,3) )
summary(mod1)
summary(mod2)

#####
# SIMULATED EXAMPLE 4: Two-dimensional copula model
#####

set.seed(5698)
I <- 9
n <- 1500                      # number of persons
b <- seq(-2,2, len=I)          # item difficulties
theta0 <- rnorm( n , sd = sqrt( .6 ) )

*** Dimension 1
theta <- theta0 + rnorm( n , sd = sqrt( .4 ) )      # person abilities
# itemcluster
itemcluster <- rep(0,I)
itemcluster[ c(3,5) ] <- 1
itemcluster[ c(2,4,9) ] <- 2
itemcluster1 <- itemcluster
# residual correlations
rho <- c( .7 , .5 )
# simulate data
dat <- sim.rasch.dep( theta , b , itemcluster , rho )
colnames(dat) <- paste("A" , seq(1,ncol(dat)) , sep="")
dat1 <- dat
# estimate model of dimension 1
mod0a <- rasch.copula2( dat1 , itemcluster = itemcluster1)
summary(mod0a)

*** Dimension 2
theta <- theta0 + rnorm( n , sd = sqrt( .8 ) )      # person abilities
# itemcluster
itemcluster <- rep(0,I)
itemcluster[ c(3,7,8) ] <- 1
itemcluster[ c(4,6) ] <- 2
itemcluster2 <- itemcluster
# residual correlations
rho <- c( .2, .4 )

```

```

# simulate data
dat <- sim.rasch.dep( theta , b , itemcluster , rho )
colnames(dat) <- paste("B" , seq(1,ncol(dat)) , sep="")
dat2 <- dat
# estimate model of dimension 2
mod0b <- rasch.copula2( dat2 , itemcluster = itemcluster2)
summary(mod0b)

# both dimensions
dat <- cbind( dat1 , dat2 )
itemcluster2 <- ifelse( itemcluster2 > 0 , itemcluster2 +2 , 0 )
itemcluster <- c( itemcluster1 , itemcluster2 )
dims <- rep( 1:2 , each=I)

# estimate two-dimensional copula model
mod1 <- rasch.copula3( dat , itemcluster=itemcluster , dims=dims , est.a=dims ,
                      theta.k = seq(-5,5,len=15) )
summary(mod1)

#####
# SIMULATED EXAMPLE 5: Subset of data Simulated Example 2
#####

set.seed(5698)
I <- 11                # number of items
n <- 3000              # number of persons
b <- seq(-2,2, len=I)  # item difficulties
theta <- rnorm( n , sd = 1.3 ) # person abilities
# define item clusters
itemcluster <- rep(0,I)
itemcluster[ c(3,5) ] <- 1
itemcluster[ c(2,4,9) ] <- 2
# residual correlations
rho <- c( .7 , .5 )
# simulate data
dat <- sim.rasch.dep( theta , b , itemcluster , rho )
colnames(dat) <- paste("I" , seq(1,ncol(dat)) , sep="")

# select subdataset with only one dependent item cluster
item.sel <- scan( what="character" , nlines=1 )
      I1 I6 I7 I8 I10 I11 I3 I5
dat1 <- dat[,item.sel]

#####
#*** Model 1a: estimate Copula model in sirt
itemcluster <- rep(0,8)
itemcluster[ c(7,8) ] <- 1
mod1a <- rasch.copula2( dat3 , itemcluster=itemcluster )
summary(mod1a)

#####
#*** Model 1b: estimate Copula model in mirt
library(mirt)

```



```

*** redefine dataset for estimation in mirt
dat2 <- dat1[ , itemcluster == 0 ]
dat2 <- as.data.frame(dat2)
# combine items 3 and 5
dat2$C35 <- dat1[,"I3"] + 2*dat1[,"I5"]
table( dat2$C35 , paste0( dat1[,"I3"],dat1[,"I5"] ) )
#* define mirt model
mirtmodel <- mirt::mirt.model("
  F = 1-7
  CONSTRAIN = (1-7,a1)
  " )
#-- Copula function with two dependent items
# define item category function for pseudo-items like C35
P.copula2 <- function(par,Theta, ncat){
  b1 <- par[1]
  b2 <- par[2]
  a1 <- par[3]
  ldelta <- par[4]
  P1 <- plogis( a1*(Theta - b1 ) )
  P2 <- plogis( a1*(Theta - b2 ) )
  Q1 <- 1-P1
  Q2 <- 1-P2
  # define vector-wise minimum function
  minf2 <- function( x1 , x2 ){
    ifelse( x1 < x2 , x1 , x2 )
  }
  # distribution under independence
  F00 <- Q1*Q2
  F10 <- Q1*Q2 + P1*Q2
  F01 <- Q1*Q2 + Q1*P2
  F11 <- 1+0*Q1
  F_ind <- c(F00,F10,F01,F11)
  # distribution under maximal dependence
  F00 <- minf2(Q1,Q2)
  F10 <- Q2          # = minf2(1,Q2)
  F01 <- Q1          # = minf2(Q1,1)
  F11 <- 1+0*Q1     # = minf2(1,1)
  F_dep <- c(F00,F10,F01,F11)
  # compute mixture distribution
  delta <- plogis(ldelta)
  F_tot <- (1-delta)*F_ind + delta * F_dep
  # recalculate probabilities of mixture distribution
  L1 <- length(Q1)
  v1 <- 1:L1
  F00 <- F_tot[v1]
  F10 <- F_tot[v1+L1]
  F01 <- F_tot[v1+2*L1]
  F11 <- F_tot[v1+3*L1]
  P00 <- F00
  P10 <- F10 - F00
  P01 <- F01 - F00
  P11 <- 1 - F10 - F01 + F00
  prob_tot <- c( P00 , P10 , P01 , P11 )

```

```

    return(prob_tot)
  }
# create item
copula2 <- mirt::createItem(name="copula2", par=c(b1 = 0 , b2 = 0.2 , a1=1 , ldelta=0) ,
  est=c(TRUE,TRUE,TRUE,TRUE) , P=P.copula2 ,
  lbound=c(-Inf,-Inf,0,-Inf) , ubound=c(Inf,Inf,Inf,Inf) )
# define item types
itemtype <- c( rep("2PL",6), "copula2" )
customItems <- list("copula2"=copula2)
# parameter table
mod.pars <- mirt::mirt(dat2, 1, itemtype=itemtype,
  customItems=customItems, pars = 'values')
# estimate model
mod1b <- mirt::mirt(dat2, mirtmodel , itemtype=itemtype , customItems=customItems,
  verbose = TRUE , pars=mod.pars ,
  technical=list(customTheta=as.matrix(seq(-4,4,len=21)) ) )
# estimated coefficients
cmod <- sirt::mirt.wrapper.coef(mod)$coef

# compare common item discrimination
round( c("sirt"=mod1a$item$a[1] , "mirt"=cmod$a1[1] ) , 4 )
##   sirt  mirt
##  1.2845 1.2862
# compare delta parameter
round( c("sirt"=mod1a$item$delta[7] , "mirt"= plogis( cmod$ldelta[7] ) ) , 4 )
##   sirt  mirt
##  0.6298 0.6297
# compare thresholds a*b
dfr <- cbind( "sirt"=mod1a$item$thresh ,
  "mirt"= c(- cmod$d[-7],cmod$b1[7]*cmod$a1[1] , cmod$b2[7]*cmod$a1[1]))
round(dfr,4)
##           sirt    mirt
## [1,] -1.9236 -1.9231
## [2,] -0.0565 -0.0562
## [3,]  0.3993  0.3996
## [4,]  0.8058  0.8061
## [5,]  1.5293  1.5295
## [6,]  1.9569  1.9572
## [7,] -1.1414 -1.1404
## [8,] -0.4005 -0.3996

## End(Not run)

```

## Description

This function performs the eigenvector approach to estimate item parameters which is based on a pairwise estimation approach (Gardner & Engelhard, 2002). No assumption about person parame-

ters is required for item parameter estimation. Statistical inference is performed by Jackknifing. If a group identifier is provided, tests for differential item functioning are performed.

### Usage

```
rasch.evm.pcm(dat, jackunits = 20, weights = NULL, pid = NULL ,
             group=NULL , powB = 2, adj_eps = 0.3, progress = TRUE )
```

```
## S3 method for class 'rasch.evm.pcm'
summary(object,...)
```

```
## S3 method for class 'rasch.evm.pcm'
coef(object,...)
```

```
## S3 method for class 'rasch.evm.pcm'
vcov(object,...)
```

### Arguments

dat	Data frame with dichotomous or polytomous item responses
jackunits	A number of Jackknife units (if an integer is provided as the argument value) or a vector in which the Jackknife units are already defined.
weights	Optional vector of sample weights
pid	Optional vector of person identifiers
group	Optional vector of group identifiers. In this case, item parameters are group wise estimated and tests for differential item functioning are performed.
powB	Power created in $B$ matrix which is the basis of parameter estimation
adj_eps	Adjustment parameter for person parameter estimation (see <a href="#">mle.pcm.group</a> )
progress	An optional logical indicating whether progress should be displayed
object	Object of class <code>rasch.evm.pcm</code>
...	Further arguments to be passed

### Value

A list with following entries

item	Data frame with item parameters. The item parameter estimate is denoted by <code>est</code> while a Jackknife bias-corrected estimate is <code>est_jack</code> . The Jackknife standard error is <code>se</code> .
b	Item threshold parameters
person	Data frame with person parameters obtained (MLE)
B	Paired comparison matrix
D	Transformed paired comparison matrix
coef	Vector of estimated coefficients
vcov	Covariance matrix of estimated item parameters

JJ	Number of jackknife units
JJadj	Reduced number of jackknife units
powB	Used power of comparison matrix $B$
maxK	Maximum number of categories per item
G	Number of groups
desc	Some descriptives
difstats	Statistics for differential item functioning if group is provided as an argument

### Author(s)

Alexander Robitzsch

### References

- Choppin, B. (1985). A fully conditional estimation procedure for Rasch Model parameters. *Evaluation in Education*, **9**, 29-42.
- Garner, M., & Engelhard, G. J. (2002). An eigenvector method for estimating item parameters of the dichotomous and polytomous Rasch models. *Journal of Applied Measurement*, **3**, 107-128.
- Wang, J., & Engelhard, G. (2014). A pairwise algorithm in R for rater-mediated assessments. *Rasch Measurement Transactions*, **28(1)**, 1457-1459.

### See Also

See the **pairwise** package for the alternative row averaging approach of Choppin (1985) and Wang and Engelhard (2014) for an alternative R implementation.

### Examples

```
#####
# EXAMPLE 1: Dataset Liking for Science
#####

data(data.liking.science)
dat <- data.liking.science

# estimate partial credit model using 10 Jackknife units
mod1 <- rasch.evm.pcm( dat , jackunits=10 )
summary(mod1)

## Not run:
# compare results with TAM
library(TAM)
mod2 <- TAM::tam.mml( dat )
r1 <- mod2$xi$xi
r1 <- r1 - mean(r1)
# item parameters are similar
dfr <- data.frame( "b_TAM"=r1 , mod1$item[,c( "est","est_jack") ] )
round( dfr , 3 )
##      b_TAM      est est_jack
```

```

## 1 -2.496 -2.599 -2.511
## 2  0.687  0.824  1.030
## 3 -0.871 -0.975 -0.943
## 4 -0.360 -0.320 -0.131
## 5 -0.833 -0.970 -0.856
## 6  1.298  1.617  1.444
## 7  0.476  0.465  0.646
## 8  2.808  3.194  3.439
## 9  1.611  1.460  1.433
## 10 2.396  1.230  1.095
## [...]

# partial credit model in eRm package
miceadds::library_install("eRm")
mod3 <- eRm::PCM(X=dat)
summary(mod3)
eRm::plotINFO(mod3)      # plot item and test information
eRm::plotICC(mod3)      # plot ICCs
eRm::plotPImap(mod3)    # plot person-item maps

#####
# EXAMPLE 2: Garner and Engelhard (2002) toy example dichotomous data
#####

dat <- scan()
  1 0 1 1   1 1 0 0   1 0 0 0   0 1 1 1   1 1 1 0
  1 1 0 1   1 1 1 1   1 0 1 0   1 1 1 1   1 1 0 0

dat <- matrix( dat , 10 , 4 , byrow=TRUE)
colnames(dat) <- paste0("I" , 1:4 )

# estimate Rasch model with no jackknifing
mod1 <- rasch.evm.pcm( dat , jackunits=0 )

# paired comparison matrix
mod1$B
##          I1_Cat1 I2_Cat1 I3_Cat1 I4_Cat1
## I1_Cat1         0         3         4         5
## I2_Cat1         1         0         3         3
## I3_Cat1         1         2         0         2
## I4_Cat1         1         1         1         0

#####
# EXAMPLE 3: Garner and Engelhard (2002) toy example polytomous data
#####

dat <- scan()
  2 2 1 1 1   2 1 2 0 0   1 0 0 0 0   0 1 1 2 0   1 2 2 1 1
  2 2 0 2 1   2 2 1 1 0   1 0 1 0 0   2 1 2 2 2   2 1 0 0 1

dat <- matrix( dat , 10 , 5 , byrow=TRUE)
colnames(dat) <- paste0("I" , 1:5 )

```

```

# estimate partial credit model with no jackknifing
mod1 <- rasch.evm.pcm( dat , jackunits=0 , powB=3 )

# paired comparison matrix
mod1$B
##          I1_Cat1 I1_Cat2 I2_Cat1 I2_Cat2 I3_Cat1 I3_Cat2 I4_Cat1 I4_Cat2 I5_Cat1 I5_Cat2
## I1_Cat1      0      0      2      0      1      1      2      1      2      1
## I1_Cat2      0      0      0      3      2      2      2      2      2      3
## I2_Cat1      1      0      0      0      1      1      2      0      2      1
## I2_Cat2      0      1      0      0      1      2      0      3      1      3
## I3_Cat1      1      1      1      1      0      0      1      2      3      1
## I3_Cat2      0      1      0      2      0      0      1      1      1      1
## I4_Cat1      0      1      0      0      0      2      0      0      1      2
## I4_Cat2      1      0      0      2      1      1      0      0      1      1
## I5_Cat1      0      1      0      1      2      1      1      2      0      0
## I5_Cat2      0      0      0      1      0      0      0      0      0      0

#####
# EXAMPLE 4: Partial credit model for dataset data.mg from CDM package
#####

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

**** Model 1: estimate partial credit model
mod1 <- rasch.evm.pcm( dat )
# item parameters
round( mod1$b , 3 )
##          Cat1   Cat2   Cat3
## I1  -1.537    NA    NA
## I2  -2.360    NA    NA
## I3  -0.574    NA    NA
## I4  -0.971 -2.086    NA
## I5  -0.104  0.201    NA
## I6   0.470  0.806    NA
## I7  -1.027  0.756  1.969
## I8   0.897    NA    NA
## I9   0.766    NA    NA
## I10  0.069    NA    NA
## I11 -1.122  1.159  2.689

**** Model 2: estimate PCM with pairwise package
miceadds::library_install("pairwise")
mod2 <- pairwise::pair(daten=dat)
summary(mod2)
plot(mod2)
# compute standard errors
semod2 <- pairwise::pairSE(daten=dat, nsample = 20)
semod2

#####
# EXAMPLE 5: Differential item functioning for dataset data.mg

```

```
#####

library(CDM)
data(data.mg,package="CDM")
dat <- data.mg[ data.mg$group %in% c(2,3,11) , ]
# define items
items <- paste0("I",1:11)
# estimate model
mod1 <- rasch.evm.pcm( dat[,items] , weights= dat$weight , group= dat$group )
summary(mod1)

#####
# SIMULATED EXAMPLE 6: Differential item functioning for Rasch model
#####

# simulate some data
set.seed(9776)
N <- 1000 # number of persons
I <- 10 # number of items
# simulate data for first group
b <- seq(-1.5,1.5,len=I)
dat1 <- sim.raschtype( rnorm(N) , b )
# simulate data for second group
b1 <- b
b1[4] <- b1[4] + .5 # introduce DIF for fourth item
dat2 <- sim.raschtype( rnorm(N,mean=.3) , b1 )
dat <- rbind(dat1 , dat2 )
group <- rep( 1:2 , each=N )
# estimate model
mod1 <- rasch.evm.pcm( dat , group= group )
summary(mod1)

## End(Not run)
```

---

 rasch.jml

---

*Joint Maximum Likelihood (JML) Estimation of the Rasch Model*


---

## Description

This function estimates the Rasch model using joint maximum likelihood estimation (Lincare, 1994). The PROX algorithm (Lincare, 1994) is used for the generation of starting values of item parameters.

## Usage

```
rasch.jml(dat, method = "MLE", b.init = NULL, constraints = NULL, weights = NULL,
  glob.conv = 10^(-6), conv1 = 1e-05, conv2 = 0.001, progress = TRUE,
  bsteps = 4, thetasteps = 2, wle.adj = 0, jmliter = 100, prox = TRUE ,
  proxiter = 30, proxconv = 0.01, dp=NULL , theta.init = NULL , calc.fit=TRUE)
```

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

### Arguments

<code>dat</code>	An $N \times I$ data frame of dichotomous item responses where $N$ indicates the number of persons and $I$ the number of items
<code>method</code>	Method for estimating person parameters during JML iterations. MLE is maximum likelihood estimation (where person with perfect scores are deleted from analysis). WLE uses weighted likelihood estimation (Warm, 1989) for person parameter estimation. Default is MLE.
<code>b.init</code>	Initial values of item difficulties
<code>constraints</code>	Optional matrix or data.frame with two columns. First column is an integer of item indexes or item names ( <code>colnames(dat)</code> ) which shall be fixed during estimation. The second column is the corresponding item difficulty.
<code>weights</code>	Person sample weights. Default is NULL, i.e. all persons in the sample are equally weighted.
<code>glob.conv</code>	Global convergence criterion with respect to the log-likelihood function
<code>conv1</code>	Convergence criterion for estimation of item parameters
<code>conv2</code>	Convergence criterion for estimation of person parameters
<code>progress</code>	Display progress? Default is TRUE
<code>b.steps</code>	Number of steps for b parameter estimation
<code>theta.steps</code>	Number of steps for theta parameter estimation
<code>wle.adj</code>	Score adjustment for WLE estimation
<code>jml.iter</code>	Number of maximal iterations during JML estimation
<code>prox</code>	Should the PROX algorithm (see <a href="#">rasch.prox</a> ) be used as initial estimations? Default is TRUE.
<code>prox.iter</code>	Number of maximal PROX iterations
<code>prox.conv</code>	Convergence criterion for PROX iterations
<code>dp</code>	Object created from data preparation function ( <code>.data.prep</code> ) which could be created in earlier JML runs. Default is NULL.
<code>theta.init</code>	Initial person parameter estimate
<code>calc.fit</code>	Should itemfit being calculated?
<code>object</code>	Object of class <code>rasch.jml</code>
<code>...</code>	Further arguments to be passed

### Details

The estimation is known to have a bias in item parameters for a fixed (finite) number of items. In literature (Lincare, 1994), a simple bias correction formula is proposed and included in the value `item$itemdiff.correction` in this function. If  $I$  denotes the number of items, then the correction factor is  $\frac{I-1}{I}$ .



**Value**

A list with following entries

item	Estimated item parameters
person	Estimated person parameters
method	Person parameter estimation method
dat	Original data frame
deviance	Deviance
data.proc	Processed data frames excluding persons with extreme scores
dp	Value of data preparation (it is used in the function <a href="#">rasch.jml.jackknife1</a> )

**Author(s)**

Alexander Robitzsch

**References**

- Linacre, J. M. (1994). *Many-Facet Rasch Measurement*. Chicago: MESA Press.
- Warm, T. A. (1989). Weighted likelihood estimation of ability in the item response theory. *Psychometrika*, **54**, 427-450.

**See Also**

Get a summary with [summary.rasch.jml](#).

See [rasch.prox](#) for the PROX algorithm as initial iterations.

For a bias correction of the JML method try [rasch.jml.jackknife1](#).

See also marginal maximum likelihood estimation with [rasch.mml2](#) or the R package **ltm**.

**Examples**

```
#####
# SIMULATED EXAMPLE 1: Simulated data from the Rasch model
#####
set.seed(789)
N <- 500 # number of persons
I <- 11 # number of items
b <- seq(-2, 2, length=I)
dat <- sim.raschtype(rnorm(N), b)
colnames(dat) <- paste("I", 1:I, sep="")

# JML estimation of the Rasch model
mod1 <- rasch.jml(dat)
summary(mod1)

# MML estimation with rasch.mml2 function
mod2 <- rasch.mml2(dat)
summary(mod2)
```

```

# Pairwise method of Fischer
mod3 <- rasch.pairwise( dat )
summary(mod3)

# JML estimation in TAM
library(TAM)
mod4 <- TAM::tam.jml2( resp=dat )

#*****
# item parameter constraints in JML estimation
# fix item difficulties: b[4]=-0.76 and b[6]= 0.10
constraints <- matrix( cbind( 4 , -0.76 ,
                             6 , 0.10 ) ,
                      ncol=2 , byrow=TRUE )
mod4 <- rasch.jml( dat , constraints = constraints )
summary(mod4)
# For constrained item parameters, it is not obvious
# how to calculate a 'right correction' of item parameter bias

```

---

rasch.jml.biascorr      *Bias Correction of Item Parameters for Joint Maximum Likelihood Estimation in the Rasch model*

---

## Description

This function computes an analytical bias correction for the Rasch model according to the method of Arellano and Hahn (2007).

## Usage

```
rasch.jml.biascorr(jmlobj, itemfac=NULL)
```

## Arguments

jmlobj	An object which is the output of the <a href="#">rasch.jml</a> function
itemfac	Number of items which are used for bias correction. By default it is the average number of item responses per person.

## Value

A list with following entries

b.biascorr	Matrix of item difficulty estimates. The column b.analytcorr1 contains item difficulties by analytical bias correction of Method 1 in Arellano and Hahn (2007) whereas b.analytcorr2 corresponds to Method 2.
b.bias1	Estimated bias by Method 1
b.bias2	Estimated bias by Method 2
itemfac	Number of items which are used as the factor for bias correction

**Author(s)**

Alexander Robitzsch

**References**

Arellano, M., & Hahn, J. (2007). Understanding bias in nonlinear panel models: Some recent developments. In R. Blundell, W. Newey & T. Persson (Eds.): *Advances in Economics and Econometrics, Ninth World Congress*, Cambridge University Press.

**See Also**

See [rasch.jml.jackknife1](#) for bias correction based on Jackknife.

**Examples**

```
#####
# EXAMPLE 1: Dataset Reading
#####
data(data.read)
dat <- data( data.read )

# estimate Rasch model
mod <- rasch.jml( data.read )

# JML with analytical bias correction
res1 <- rasch.jml.biascorr( jmlobj=mod )
print( res1$b.biascorr , digits= 3 )
##      b.JML b.JMLcorr b.analytcorr1 b.analytcorr2
## 1 -2.0086 -1.8412 -1.908 -1.922
## 2 -1.1121 -1.0194 -1.078 -1.088
## 3 -0.0718 -0.0658 -0.150 -0.127
## 4  0.5457  0.5002  0.393  0.431
## 5 -0.9504 -0.8712 -0.937 -0.936
## [...]
```

---

rasch.jml.jackknife1 *Jackknifing the IRT Model Estimated by Joint Maximum Likelihood (JML)*

---

**Description**

Jackknife estimation is an alternative to other ad hoc proposed methods for bias correction (Hahn & Newey, 2004).

**Usage**

```
rasch.jml.jackknife1(jmlobj)
```

**Arguments**

jmlobj            Output of rasch.jml

**Details**

Note that items are used for jackknifing (Hahn & Newey, 2004). By default, all  $I$  items in the data frame are used as jackknife units.

**Value**

A list with following entries

item            A data frame with item parameters

- b. JML: Item difficulty from JML estimation
- b. JMLcorr: Item difficulty from JML estimation by applying the correction factor  $(I - 1)/I$
- b. jack: Item difficulty from Jackknife estimation
- b. jackse: Standard error of Jackknife estimation for item difficulties. Note that this parameter refer to the standard error with respect to item sampling
- b. JMLse: Standard error for item difficulties obtained from JML estimation

jack.itemdiff   A matrix containing all item difficulties obtained by Jackknife

**Author(s)**

Alexander Robitzsch

**References**

Hahn, J., & Newey, W. (2004). Jackknife and analytical bias reduction for nonlinear panel models. *Econometrica*, **72**, 1295-1319.

**See Also**

For JML estimation [rasch.jml](#).

For analytical bias correction methods see [rasch.jml.biascorr](#).

**Examples**

```
#####
# SIMULATED EXAMPLE 1: Simulated data from the Rasch model
#####
set.seed(7655)
N <- 5000    # number of persons
I <- 11     # number of items
b <- seq(-2 , 2 , length=I )
dat <- sim.raschtype( rnorm( N ) , b )
colnames(dat) <- paste( "I" , 1:I , sep="")

# estimate the Rasch model with JML
```

```

mod <- rasch.jml( dat )
summary(mod)

# re-estimate the Rasch model using Jackknife
mod2 <- rasch.jml.jackknife1( mod )
##
## Joint Maximum Likelihood Estimation
## Jackknife Estimation
## 11 Jackknife Units are used
## |-----PROGRESS-----|
## |-----|
##
##          N      p  b.JML b.JMLcorr b.jack b.jackse b.JMLse
## I1  4929 0.853 -2.345  -2.131 -2.078  0.079  0.045
## I2  4929 0.786 -1.749  -1.590 -1.541  0.075  0.039
## I3  4929 0.723 -1.298  -1.180 -1.144  0.065  0.036
## I4  4929 0.657 -0.887  -0.806 -0.782  0.059  0.035
## I5  4929 0.576 -0.420  -0.382 -0.367  0.055  0.033
## I6  4929 0.492  0.041  0.038  0.043  0.054  0.033
## I7  4929 0.409  0.502  0.457  0.447  0.056  0.034
## I8  4929 0.333  0.939  0.854  0.842  0.058  0.035
## I9  4929 0.264  1.383  1.257  1.229  0.065  0.037
## I10 4929 0.210  1.778  1.617  1.578  0.071  0.040
## I11 4929 0.154  2.266  2.060  2.011  0.077  0.044
#-> Item parameters obtained by jackknife seem to be acceptable.

```

---

rasch.mirtlc

*Multidimensional Latent Class 1PL and 2PL Model*


---

## Description

This function estimates the multidimensional latent class Rasch (1PL) and 2PL model (Bartolucci, 2007; Bartolucci, Montanari & Pandolfi, 2012) for dichotomous data which emerges from the original latent class model (Goodman, 1974) and a multidimensional IRT model.

## Usage

```

rasch.mirtlc(dat, Nclasses=NULL, modeltype="LC", dimensions=NULL ,
  group=NULL, weights=rep(1,nrow(dat)), theta.k=NULL, ref.item=NULL ,
  distribution.trait= FALSE , range.b =c(-8,8), range.a=c(.2 , 6 ) ,
  progress=TRUE, glob.conv=10^(-5), conv1=10^(-5), mmliter=1000,
  mstep.maxit=3, seed=0, nstarts=1 , fac.iter=.35)

```

```

## S3 method for class 'rasch.mirtlc'
summary(object,...)

```

```

## S3 method for class 'rasch.mirtlc'
anova(object,...)

```

```

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

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

## S3 method for class 'rasch.mirtlc'
IRT.likelihood(object,...)

## S3 method for class 'rasch.mirtlc'
IRT.posterior(object,...)

## S3 method for class 'rasch.mirtlc'
IRT.modelfit(object,...)

## S3 method for class 'IRT.modelfit.rasch.mirtlc'
summary(object,...)

```

### Arguments

dat	An $N \times I$ data frame
Nclasses	Number of latent classes. If the trait vector (or matrix) theta.k is specified, then Nclasses is set to the dimension of theta.k.
modeltype	Modeltype. LC is the latent class model of Goodman (1974). MLC1 is the multi-dimensional latent class Rasch model with item discrimination parameter of 1. MLC2 allows for the estimation of item discriminations.
dimensions	Vector of dimension integers which allocate items to dimensions.
group	A group identifier for multiple group estimation
weights	Vector of sample weights
theta.k	A grid of theta values can be specified if theta should not be estimated. In the one-dimensional case, it must be a vector, in the $D$ -dimensional case it must be a matrix of dimension $D$ .
ref.item	An optional vector of integers which indicate the items whose intercept and slope are fixed at 0 and 1, respectively.
distribution.trait	A type of the assumed theta distribution can be specified. One alternative is normal for the normal distribution assumption. The options smooth2, smooth3 and smooth4 use the log-linear smoothing of Xu and von Davier (2008) to smooth the distribution up to two, three or four moments, respectively. This function only works in unidimensional models. If a different string is provided as an input (e.g. no), then no smoothing is conducted.
range.b	Range of item difficulties which are allowed for estimation
range.a	Range of item slopes which are allowed for estimation
progress	Display progress? Default is TRUE.

glob.conv	Global relative deviance convergence criterion
conv1	Item parameter convergence criterion
mmliter	Maximum number of iterations
mstep.maxit	Maximum number of iterations within an M step
seed	Set random seed for latent class estimation. A seed can be specified. If the seed is negative, then the function will generate a random seed.
nstarts	If a positive integer is provided, then a nstarts starts with different starting values are conducted.
fac.iter	A parameter between 0 and 1 to control the maximum increment in each iteration. The larger the parameter the more increments will become smaller from iteration to iteration.
object	Object of class rasch.mirtlc
...	Further arguments to be passed

### Details

The multidimensional latent class Rasch model (Bartolucci, 2007) is an item response model which combines ideas from latent class analysis and item response models with continuous variables. With `modeltype="MLC2"` the following  $D$ -dimensional item response model is estimated

$$\text{logit}P(X_{pi} = 1|\theta_p) = a_i\theta_{pcd} - b_i$$

Besides the item thresholds  $b_i$  and item slopes  $a_i$ , for a prespecified number of latent classes  $c = 1, \dots, C$  a set of  $C$   $D$ -dimensional  $\{\theta_{cd}\}_{cd}$  vectors are estimated. These vectors represent the locations of latent classes. If the user provides a grid of theta distribution `theta.k` as an argument in `rasch.mirtlc`, then the ability distribution is fixed.

In the unidimensional Rasch model with  $I$  items,  $(I + 1)/2$  (if  $I$  odd) or  $I/2 + 1$  (if  $I$  even) trait location parameters are identified (see De Leeuw & Verhelst, 1986; Lindsay et al., 1991; for a review see Formann, 2007).

### Value

A list with following entries

pjk	Item probabilities evaluated at discretized ability distribution
rprobs	Item response probabilities like in pjk, but for each item category
pi.k	Estimated trait distribution
theta.k	Discretized ability distribution
item	Estimated item parameters
trait	Estimated ability distribution (theta.k and pi.k)
mean.trait	Estimated mean of ability distribution
sd.trait	Estimated standard deviation of ability distribution
skewness.trait	Estimated skewness of ability distribution
cor.trait	Estimated correlation between abilities (only applies for multidimensional models)

ic	Information criteria
D	Number of dimensions
G	Number of groups
deviance	Deviance
ll	Log-likelihood
Nclasses	Number of classes
modeltype	Used model type
estep.res	Result from E step: $f.qk.yi$ is the individual posterior, $f.yi.qk$ is the individual likelihood
dat	Original data frame
devL	Vector of deviances if multiple random starts were conducted
seedL	Vector of seed if multiple random starts were conducted
iter	Number of iterations

### Note

For the estimation of latent class models, rerunning the model with different starting values (different random seeds) is recommended.

For fixed theta estimation in the multidimensional case, large vectors are generated during estimation leading to memory overflow in R.

### Author(s)

Alexander Robitzsch

### References

- Bartolucci, F. (2007). A class of multidimensional IRT models for testing unidimensionality and clustering items. *Psychometrika*, *72*, 141-157.
- Bartolucci, F., Montanari, G. E., & Pandolfi, S. (2012). Dimensionality of the latent structure and item selection via latent class multidimensional IRT models. *Psychometrika*, *77*, 782-802.
- De Leeuw, J., & Verhelst, N. (1986). Maximum likelihood estimation in generalized Rasch models. *Journal of Educational and Behavioral Statistics*, *11*, 183-196.
- 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: *Multivariate and Mixture Distribution Rasch Models* (pp. 177-189). Springer: New York.
- Goodman, L. A. (1974). Exploratory latent structure analysis using both identifiable and unidentifiable models. *Biometrika*, *61*, 215-231.
- Lindsay, B., Clogg, C. C., & Grego, J. (1991). Semiparametric estimation in the Rasch model and related exponential response models, including a simple latent class model for item analysis. *Journal of the American Statistical Association*, *86*, 96-107.
- Xu, X., & von Davier, M. (2008). *Fitting the structured general diagnostic model to NAEP data*. ETS Research Report ETS RR-08-27. Princeton, ETS.



**See Also**

See also the `CDM::gdm` function in the **CDM** package.

For an assessment of global model fit see `modelfit.sirt`.

The estimation of the multidimensional latent class item response model for polytomous data can be conducted in the **MultiLCIRT** package. Latent class analysis can be carried out with **poLCA** and **randomLCA** packages.

**Examples**

```
#####
# EXAMPLE 1: Reading data
#####
data( data.read )
dat <- data.read

#*****
# latent class models

# latent class model with 1 class
mod1 <- rasch.mirtlc( dat , Nclasses = 1 )
summary(mod1)

# latent class model with 2 classes
mod2 <- rasch.mirtlc( dat , Nclasses = 2 )
summary(mod2)

## Not run:
# latent class model with 3 classes
mod3 <- rasch.mirtlc( dat , Nclasses = 3 , seed = - 30)
summary(mod3)

# extract individual likelihood
lmod3 <- IRT.likelihood(mod3)
str(lmod3)
# extract likelihood value
logLik(mod3)
# extract item response functions
IRT.irfprob(mod3)

# compare models 1, 2 and 3
anova(mod2,mod3)
IRT.compareModels(mod1,mod2,mod3)
# avbsolute and relative model fit
smod2 <- IRT.modelfit(mod2)
smod3 <- IRT.modelfit(mod3)
summary(smod2)
IRT.compareModels(smod2,smod3)

# latent class model with 4 classes and 3 starts with different seeds
mod4 <- rasch.mirtlc( dat , Nclasses = 4 ,seed= -30 , nstarts=3 )
# display different solutions
```

```

sort(mod4$devL)
summary(mod4)

# latent class multiple group model
# define group identifier
group <- rep( 1 , nrow(dat))
group[ 1:150 ] <- 2
mod5 <- rasch.mirtlc( dat , Nclasses = 3 , group = group )
summary(mod5)

#*****
# Unidimensional IRT models with ordered trait

# 1PL model with 3 classes
mod11 <- rasch.mirtlc( dat , Nclasses = 3 , modeltype="MLC1" , mmliter=30)
summary(mod11)

# 1PL model with 11 classes
mod12 <- rasch.mirtlc( dat , Nclasses = 11 ,modeltype="MLC1", mmliter=30)
summary(mod12)

# 1PL model with 11 classes and fixed specified theta values
mod13 <- rasch.mirtlc( dat , modeltype="MLC1" ,
                      theta.k = seq( -4 , 4 , len=11 ) , mmliter=100)
summary(mod13)

# 1PL model with fixed theta values and normal distribution
mod14 <- rasch.mirtlc( dat , modeltype="MLC1" , mmliter=30 ,
                      theta.k = seq( -4 , 4 , len=11 ) , distribution.trait="normal")
summary(mod14)

# 1PL model with a smoothed trait distribution (up to 3 moments)
mod15 <- rasch.mirtlc( dat , modeltype="MLC1" , mmliter=30 ,
                      theta.k = seq( -4, 4 , len=11 ) , distribution.trait="smooth3")
summary(mod15)

# 2PL with 3 classes
mod16 <- rasch.mirtlc( dat , Nclasses=3 , modeltype="MLC2" , mmliter=30 )
summary(mod16)

# 2PL with fixed theta and smoothed distribution
mod17 <- rasch.mirtlc( dat , theta.k=seq(-4,4,len=12) , mmliter=30 ,
                      modeltype="MLC2" , distribution.trait="smooth4" )
summary(mod17)

# 1PL multiple group model with 8 classes
# define group identifier
group <- rep( 1 , nrow(dat))
group[ 1:150 ] <- 2
mod21 <- rasch.mirtlc( dat , Nclasses = 8 , modeltype="MLC1" , group=group )
summary(mod21)

#*****

```

```

# multidimensional latent class IRT models

# define vector of dimensions
dimensions <- rep( 1:3 , each = 4 )

# 3-dimensional model with 8 classes and seed 145
mod31 <- rasch.mirtlc( dat , Nclasses = 8 , mmliter=30 ,
                      modeltype="MLC1" , seed = 145 , dimensions = dimensions )
summary(mod31)

# try the model above with different starting values
mod31s <- rasch.mirtlc( dat , Nclasses = 8 ,
                      modeltype="MLC1" , seed = -30 , nstarts=30 , dimensions = dimensions )
summary(mod31s)

# estimation with fixed theta vectors
# => 4^3 = 216 classes
theta.k <- seq(-4 , 4 , len=6 )
theta.k <- as.matrix( expand.grid( theta.k , theta.k , theta.k ) )
mod32 <- rasch.mirtlc( dat , dimensions=dimensions ,
                      theta.k= theta.k , modeltype="MLC1" )
summary(mod32)

# 3-dimensional 2PL model
mod33 <- rasch.mirtlc( dat , dimensions=dimensions , theta.k= theta.k , modeltype="MLC2")
summary(mod33)

#####
# SIMULATED EXAMPLE 2: Skew trait distribution
#####
set.seed(789)
N <- 1000 # number of persons
I <- 20 # number of items
theta <- sqrt( exp( rnorm( N ) ) )
theta <- theta - mean(theta)
# calculate skewness of theta distribution
mean( theta^3 ) / sd(theta)^3
# simulate item responses
dat <- sim.raschtype( theta , b=seq(-2,2,len=I ) )

# normal distribution
mod1 <- rasch.mirtlc( dat , theta.k=seq(-4,4,len=15) , modeltype="MLC1",
                    distribution.trait="normal" , mmliter=30)

# allow for skew distribution with smoothed distribution
mod2 <- rasch.mirtlc( dat , theta.k=seq(-4,4,len=15) , modeltype="MLC1",
                    distribution.trait="smooth3" , mmliter=30)

# nonparametric distribution
mod3 <- rasch.mirtlc( dat , theta.k=seq(-4,4,len=15) , modeltype="MLC1", mmliter=30)

summary(mod1)
summary(mod2)

```

```

summary(mod3)

#####
# EXAMPLE 3: Stouffer-Toby dataset data.si02 with 5 items
#####

data(dat.si02)
dat <- data.si02$data
weights <- data.si02$weights # extract weights

# Model 1: 2 classes Rasch model
mod1 <- rasch.mirtlc( dat , Nclasses=2 , modeltype="MLC1" , weights = weights ,
                    ref.item = 4 , nstarts=5)
summary(mod1)

# Model 2: 3 classes Rasch model: not all parameters are identified
mod2 <- rasch.mirtlc( dat , Nclasses=3 , modeltype="MLC1" , weights = weights ,
                    ref.item = 4 , nstarts=5)
summary(mod2)

# Model 3: Latent class model with 2 classes
mod3 <- rasch.mirtlc( dat , Nclasses=2 , modeltype="LC" , weights = weights , nstarts=5)
summary(mod3)

# Model 4: Rasch model with normal distribution
mod4 <- rasch.mirtlc( dat , modeltype="MLC1" , weights=weights ,
                    theta.k = seq( -6 , 6 , len=21 ) , distribution.trait="normal" , ref.item=4)
summary(mod4)

## End(Not run)

#####
# SIMULATED EXAMPLE 4: 5 classes, 3 dimensions and 27 items
#####

set.seed(979)
I <- 9
N <- 5000
b <- seq( - 1.5, 1.5 , len=I)
b <- rep(b,3)
# define class locations
theta.k <- c(-3.0, -4.1, -2.8 , 1.7 , 2.3 , 1.8 ,
            0.2 , 0.4 , -0.1 , 2.6 , 0.1, -0.9, -1.1 , -0.7 , 0.9 )

Nclasses <- 5
theta.k0 <- theta.k <- matrix( theta.k , Nclasses , 3 , byrow=TRUE )
pi.k <- c(.20,.25,.25,.10,.15)
theta <- theta.k[ rep( 1:Nclasses , round(N*pi.k) ) , ]
dimensions <- rep( 1:3 , each=I)
# simulate item responses
dat <- matrix( NA , nrow=N , ncol=I*3)
for (ii in 1:(3*I) ){
  dat[,ii] <- 1 * ( runif(N) < plogis( theta[, dimensions[ii] ] - b[ ii] ) )
}

```

```

    }
  colnames(dat) <- paste0( rep( LETTERS[1:3] , each=I ) , 1:(3*I) )

  # estimate model
  mod1 <- rasch.mirtlc( dat , Nclasses=Nclasses , dimensions=dimensions , modeltype="MLC1" ,
    ref.item= c(5,14,23) , glob.conv=.0005, conv1=.0005)

  round( cbind( mod1$theta.k , mod1$pi.k ) , 3 )
  ##          [,1] [,2] [,3] [,4]
  ## [1,] -2.776 -3.791 -2.667 0.250
  ## [2,] -0.989 -0.605 0.957 0.151
  ## [3,] 0.332 0.418 -0.046 0.246
  ## [4,] 2.601 0.171 -0.854 0.101
  ## [5,] 1.791 2.330 1.844 0.252
  cbind( theta.k , pi.k )
  ##          pi.k
  ## [1,] -3.0 -4.1 -2.8 0.20
  ## [2,] 1.7 2.3 1.8 0.25
  ## [3,] 0.2 0.4 -0.1 0.25
  ## [4,] 2.6 0.1 -0.9 0.10
  ## [5,] -1.1 -0.7 0.9 0.15

  # plot class locations
  plot( 1:3 , mod1$theta.k[1,] , xlim=c(1,3) , ylim=c(-5,3) , col=1 , pch=1 , type="n" ,
    axes=FALSE, xlab="Dimension" , ylab="Location")
  axis(1 , 1:3 ) ; axis(2) ; axis(4)
  for (cc in 1:Nclasses){ # cc <- 1
    lines(1:3, mod1$theta.k[cc,] , col=cc , lty=cc )
    points(1:3, mod1$theta.k[cc,] , col=cc , pch =cc )
  }

  ## Not run:
  #-----
  # estimate model with gdm function in CDM package
  library(CDM)
  # define Q-matrix
  Qmatrix <- matrix(0,3*I,3)
  Qmatrix[ cbind( 1:(3*I) , rep(1:3 , each=I) ) ] <- 1

  set.seed(9176)
  # random starting values for theta locations
  theta.k <- matrix( 2*rnorm(5*3) , 5 , 3 )
  colnames(theta.k) <- c("Dim1","Dim2","Dim3")
  # try possibly different starting values

  # estimate model in CDM
  b.constraint <- cbind( c(5,14,23) , 1 , 0 )
  mod2 <- CDM::gdm( dat , theta.k = theta.k , b.constraint=b.constraint, skillspace="est",
    irtmodel="1PL", Qmatrix=Qmatrix)
  summary(mod2)

  #-----
  # estimate model with MultiLCIRT package

```

```

miceadds::library_install("MultiLCIRT")

# define matrix to allocate each item to one dimension
multi1 <- matrix( 1:(3*I) , nrow=3 , byrow=TRUE )
# define reference items in item-dimension allocation matrix
multi1[ 1 , c(1,5) ] <- c(5,1)
multi1[ 2 , c(10,14) - 9 ] <- c(14,9)
multi1[ 3 , c(19,23) - 18 ] <- c(23,19)

# Rasch model with 5 latent classes (random start: start=1)
mod3 <- MultiLCIRT::est_multi_poly(S=dat,k=5, # k=5 ability levels
  start=1,link=1,multi=multi1,tol=10^-5 ,
  output=TRUE , disp=TRUE , fort=TRUE)
# estimated location points and class probabilities in MultiLCIRT
cbind( t( mod3$Th ) , mod3$piv )
# compare results with rasch.mirtlc
cbind( mod1$theta.k , mod1$pi.k )
# simulated data parameters
cbind( theta.k , pi.k )

#---
# estimate model with customized input in mirt
library(mirt)
#-- define Theta design matrix for 5 classes
Theta <- diag(5)
Theta <- cbind( Theta , Theta , Theta )
r1 <- rownames(Theta) <- paste0("C",1:5)
colnames(Theta) <- c( paste0(r1 , "D1") , paste0(r1 , "D2") , paste0(r1 , "D3") )
##      C1D1 C2D1 C3D1 C4D1 C5D1 C1D2 C2D2 C3D2 C4D2 C5D2 C1D3 C2D3 C3D3 C4D3 C5D3
## C1    1    0    0    0    0    1    0    0    0    0    1    0    0    0    0
## C2    0    1    0    0    0    0    1    0    0    0    0    1    0    0    0
## C3    0    0    1    0    0    0    0    1    0    0    0    0    1    0    0
## C4    0    0    0    1    0    0    0    0    1    0    0    0    0    1    0
## C5    0    0    0    0    1    0    0    0    0    1    0    0    0    0    1
#-- define mirt model
I <- ncol(dat) # I = 27
mirtmodel <- mirt::mirt.model("
  C1D1 = 1-9 \n C2D1 = 1-9 \n C3D1 = 1-9 \n C4D1 = 1-9 \n C5D1 = 1-9
  C1D2 = 10-18 \n C2D2 = 10-18 \n C3D2 = 10-18 \n C4D2 = 10-18 \n C5D2 = 10-18
  C1D3 = 19-27 \n C2D3 = 19-27 \n C3D3 = 19-27 \n C4D3 = 19-27 \n C5D3 = 19-27
  CONSTRAIN = (1-9,a1),(1-9,a2),(1-9,a3),(1-9,a4),(1-9,a5),
              (10-18,a6),(10-18,a7),(10-18,a8),(10-18,a9),(10-18,a10),
              (19-27,a11),(19-27,a12),(19-27,a13),(19-27,a14),(19-27,a15)
  ")
#-- get initial parameter values
mod.pars <- mirt::mirt(dat, model=mirtmodel , pars = "values")
#-- redefine initial parameter values
# set all d parameters initially to zero
ind <- which( ( mod.pars$name == "d" ) )
mod.pars[ ind , "value" ] <- 0
# fix item difficulties of reference items to zero
mod.pars[ ind[ c(5,14,23) ] , "est" ] <- FALSE
mod.pars[ind,]

```

```

# initial item parameters of cluster locations (a1,...,a15)
ind <- which( ( mod.pars$name %in% paste0("a", c(1,6,11) ) ) & ( mod.pars$est ) )
mod.pars[ind,"value"] <- -2
ind <- which( ( mod.pars$name %in% paste0("a", c(1,6,11)+1 ) ) & ( mod.pars$est ) )
mod.pars[ind,"value"] <- -1
ind <- which( ( mod.pars$name %in% paste0("a", c(1,6,11)+2 ) ) & ( mod.pars$est ) )
mod.pars[ind,"value"] <- 0
ind <- which( ( mod.pars$name %in% paste0("a", c(1,6,11)+3 ) ) & ( mod.pars$est ) )
mod.pars[ind,"value"] <- 1
ind <- which( ( mod.pars$name %in% paste0("a", c(1,6,11)+4 ) ) & ( mod.pars$est ) )
mod.pars[ind,"value"] <- 0
#-- define prior for latent class analysis
lca_prior <- function(Theta,Etable){
  TP <- nrow(Theta)
  if ( is.null(Etable) ){ prior <- rep( 1/TP , TP ) }
  if ( ! is.null(Etable) ){
    prior <- ( rowSums(Etable[ , seq(1,2*I,2)]) + rowSums(Etable[,seq(2,2*I,2)]) )/I
  }
  prior <- prior / sum(prior)
  return(prior)
}

#-- estimate model in mirt
mod4 <- mirt::mirt(dat, mirtmodel , pars = mod.pars , verbose=TRUE ,
  technical = list( customTheta=Theta , customPriorFun = lca_prior ,
    MAXQUAD = 1E20) )
# correct number of estimated parameters
mod4@nest <- as.integer(sum(mod.pars$est) + nrow(Theta)-1 )
# extract coefficients
# source.all(pfsirt)
cmod4 <- mirt.wrapper.coef(mod4)

# estimated item difficulties
dfr <- data.frame( "sim"=b , "mirt"=-cmod4$coef$d , "sirt"=mod1$item$thresh )
round( dfr , 4 )
##      sim      mirt      sirt
##  1 -1.500 -1.3782 -1.3382
##  2 -1.125 -1.0059 -0.9774
##  3 -0.750 -0.6157 -0.6016
##  4 -0.375 -0.2099 -0.2060
##  5  0.000  0.0000  0.0000
##  6  0.375  0.5085  0.4984
##  7  0.750  0.8661  0.8504
##  8  1.125  1.3079  1.2847
##  9  1.500  1.5891  1.5620
##  [...]

#-- reordering estimated latent clusters to make solutions comparable
#* extract estimated cluster locations from sirt
order.sirt <- c(1,5,3,4,2) # sort(order.sirt)
round(mod1$trait[,1:3],3)
dfr <- data.frame( "sim"=theta.k , mod1$trait[order.sirt,1:3] )
colnames(dfr)[4:6] <- paste0("sirt",1:3)

```

```

#* extract estimated cluster locations from mirt
c4 <- cmod4$coef[ , paste0("a",1:15) ]
c4 <- apply( c4 ,2 , FUN = function(ll){ ll[ ll!= 0 ][1] } )
trait.loc <- matrix(c4,5,3)
order.mirt <- c(1,4,3,5,2) # sort(order.mirt)
dfr <- cbind( dfr , trait.loc[ order.mirt , ] )
colnames(dfr)[7:9] <- paste0("mirt",1:3)
# compare estimated cluster locations
round(dfr,3)
##      sim.1 sim.2 sim.3 sirt1 sirt2 sirt3 mirt1 mirt2 mirt3
## 1 -3.0 -4.1 -2.8 -2.776 -3.791 -2.667 -2.856 -4.023 -2.741
## 5 1.7 2.3 1.8 1.791 2.330 1.844 1.817 2.373 1.869
## 3 0.2 0.4 -0.1 0.332 0.418 -0.046 0.349 0.421 -0.051
## 4 2.6 0.1 -0.9 2.601 0.171 -0.854 2.695 0.166 -0.876
## 2 -1.1 -0.7 0.9 -0.989 -0.605 0.957 -1.009 -0.618 0.962
#* compare estimated cluster sizes
dfr <- data.frame( "sim" = pi.k , "sirt"=mod1$pi.k[order.sirt,1] ,
                  "mirt"=mod4@Prior[[1]][ order.mirt] )
round(dfr,4)
##      sim sirt mirt
## 1 0.20 0.2502 0.2500
## 2 0.25 0.2522 0.2511
## 3 0.25 0.2458 0.2494
## 4 0.10 0.1011 0.0986
## 5 0.15 0.1507 0.1509

#####
# EXAMPLE 5: Dataset data.si04 from Bartolucci et al. (2012)
#####

data(data.si04)

# define reference items
ref.item <- c(7,17,25,44,64)
dimensions <- data.si04$itempars$dim

# estimate a Rasch latent class with 9 classes
mod1 <- rasch.mirtlc( data.si04$data , Nclasses=9 , dimensions=dimensions , modeltype="MLC1" ,
                    ref.item=ref.item , glob.conv=.005 , conv1=.005 , nstarts=1 , mmliter=200 )

# compare estimated distribution with simulated distribution
round( cbind( mod1$theta.k , mod1$pi.k ) , 4 ) # estimated
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,] -3.6043 -5.1323 -5.3022 -6.8255 -4.3611 0.1341
## [2,] 0.2083 -2.7422 -2.8754 -5.3416 -2.5085 0.1573
## [3,] -2.8641 -4.0272 -5.0580 -0.0340 -0.9113 0.1163
## [4,] -0.3575 -2.0081 -1.7431 1.2992 -0.1616 0.0751
## [5,] 2.9329 0.3662 -1.6516 -3.0284 0.1844 0.1285
## [6,] 1.5092 -2.0461 -4.3093 1.0481 1.0806 0.1094
## [7,] 3.9899 3.1955 -4.0010 1.8879 2.2988 0.1460
## [8,] 4.3062 0.7080 -1.2324 1.4351 2.0893 0.1332
## [9,] 5.0855 4.1214 -0.9141 2.2744 1.5314 0.0000

```



```

round(d2,4) # simulated
##      class      A      B      C      D      E      pi
## [1,]      1 -3.832 -5.399 -5.793 -7.042 -4.511 0.1323
## [2,]      2 -2.899 -4.217 -5.310 -0.055 -0.915 0.1162
## [3,]      3 -0.376 -2.137 -1.847  1.273 -0.078 0.0752
## [4,]      4  0.208 -2.934 -3.011 -5.526 -2.511 0.1583
## [5,]      5  1.536 -2.137 -4.606  1.045  1.143 0.1092
## [6,]      6  2.042 -0.573 -0.404 -4.331 -1.044 0.0471
## [7,]      7  3.853  0.841 -2.993 -2.746  0.803 0.0822
## [8,]      8  4.204  3.296 -4.328  1.892  2.419 0.1453
## [9,]      9  4.466  0.700 -1.334  1.439  2.161 0.1343

## End(Not run)

```

---

```
rasch.mml2
```

*Estimation of the Generalized Logistic Item Response Model, Ramsay's Quotient Model, Nonparametric Item Response Model, Pseudo-Likelihood Estimation and a Missing Data Item Response Model*

---

## Description

This function employs marginal maximum likelihood estimation of item response models for dichotomous data. First, the Rasch type model (generalized item response model) can be estimated. The generalized logistic link function (Stukel, 1988) can be estimated or fixed for conducting IRT with different link functions than the logistic one. The Four-Parameter logistic item response model is a special case of this model (Loken & Rulison, 2010). Second, Ramsay's quotient model (Ramsay, 1989) can be estimated by specifying `irtmodel="ramsay.qm"`. Third, quite general item response functions can be estimated in a nonparametric framework (Rossi, Wang & Ramsay, 2002). Fourth, pseudo-likelihood estimation for fractional item responses can be conducted for Rasch type models. Fifth, a simple two-dimensional missing data item response model (`irtmodel='missing1'`; Mislevy & Wu, 1996) can be estimated.

See Details for more explanations.

## Usage

```

rasch.mml2( dat , theta.k=seq(-6,6,len=21) , group=NULL , weights=NULL ,
  constraints=NULL , glob.conv=10^(-5) , parm.conv=10^(-4) , mitermax=4 ,
  mmliter=1000 , progress=TRUE , fixed.a=rep(1,ncol(dat)) ,
  fixed.c=rep(0,ncol(dat)) , fixed.d=rep(1,ncol(dat)) ,
  fixed.K=rep(3,ncol(dat)) , b.init=NULL , est.a=NULL , est.b=NULL ,
  est.c=NULL , est.d=NULL , min.b=-99 , max.b=99 , min.a=-99 , max.a = 99 ,
  min.c=0 , max.c=1 , min.d=0 , max.d=1 , est.K=NULL , min.K=1 , max.K=20 ,
  beta.init = NULL , min.beta = -8 , pid=1:(nrow(dat)) , trait.weights=NULL ,
  center.trait=TRUE , center.b=FALSE , alpha1=0 , alpha2=0 , est.alpha=FALSE ,
  equal.alpha=FALSE , designmatrix=NULL , alpha.conv=parm.conv , numdiff.parm=0.00001 ,
  numdiff.alpha.parm= numdiff.parm , distribution.trait="normal" , Qmatrix=NULL ,
  variance.fixed=NULL , variance.init=NULL ,
  mu.fixed=cbind(seq(1,ncol(Qmatrix)),rep(0,ncol(Qmatrix))) ,

```

```

    irtmodel="raschtype" , npformula=NULL , npirt.monotone=TRUE ,
    use.freqpatt = is.null(group) , delta.miss=0 , est.delta=rep(NA,ncol(dat)) ,
    ... )

## S3 method for class 'rasch.mml'
summary(object,...)

## S3 method for class 'rasch.mml'
plot(x,items=NULL, xlim=NULL, main=NULL, ...)

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

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

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

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

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

## S3 method for class 'rasch.mml'
IRT.modelfit(object,...)

## S3 method for class 'IRT.modelfit.rasch.mml'
summary(object,...)

```

### Arguments

dat	An $N \times I$ data frame of dichotomous item responses. For the missing data item response model ( <code>irtmodel='missing1'</code> ), code item responses by 9 which should be treated by the missing data model. Other missing responses can be coded by NA.
theta.k	Optional vector of discretized theta values. For multidimensional IRT models with $D$ dimensions, it is a matrix with $D$ columns.
group	Vector of integers with group identifiers in multiple group estimation. The multiple group does not work for <code>irtmodel="missing1"</code> .
weights	Optional vector of person weights (sample weights).
constraints	Constraints on b parameters (item difficulties). It must be a matrix with two columns: the first column contains item names, the second column fixed parameter values.
glob.conv	Convergence criterion for deviance
parm.conv	Convergence criterion for item parameters

mitermax	Maximum number of iterations in M step. This argument does only apply for the estimation of the $b$ parameters.
mmliter	Maximum number of iterations
progress	Should progress be displayed at the console?
fixed.a	Fixed or initial $a$ parameters
fixed.c	Fixed or initial $c$ parameters
fixed.d	Fixed or initial $d$ parameters
fixed.K	Fixed or initial $K$ parameters in Ramsay's quotient model.
b.init	Initial $b$ parameters
est.a	Vector of integers which indicate which $a$ parameters should be estimated. Equal integers correspond to the same estimated parameters.
est.b	Vector of integers which indicate which $b$ parameters should be estimated. Equal integers correspond to the same estimated parameters.
est.c	Vector of integers which indicate which $c$ parameters should be estimated. Equal integers correspond to the same estimated parameters.
est.d	Vector of integers which indicate which $d$ parameters should be estimated. Equal integers correspond to the same estimated parameters.
min.b	Minimal $b$ parameter to be estimated
max.b	Maximal $b$ parameter to be estimated
min.a	Minimal $a$ parameter to be estimated
max.a	Maximal $a$ parameter to be estimated
min.c	Minimal $c$ parameter to be estimated
max.c	Maximal $c$ parameter to be estimated
min.d	Minimal $d$ parameter to be estimated
max.d	Maximal $d$ parameter to be estimated
est.K	Vector of integers which indicate which $K$ parameters should be estimated. Equal integers correspond to the same estimated parameters.
min.K	Minimal $K$ parameter to be estimated
max.K	Maximal $K$ parameter to be estimated
beta.init	Optional vector of initial $\beta$ parameters
min.beta	Minimum $\beta$ parameter to be estimated.
pid	Optional vector of person identifiers
trait.weights	Optional vector of trait weights for a fixing the trait distribution.
center.trait	Should the trait distribution be centered
center.b	An optional logical indicating whether $b$ parameters should be centered at each dimension
alpha1	Fixed or initial $\alpha_1$ parameter
alpha2	Fixed or initial $\alpha_2$ parameter
est.alpha	Should $\alpha$ parameters be estimated?

<code>equal.alpha</code>	Estimate $\alpha$ parameters under the assumption $\alpha_1 = \alpha_2$ ?
<code>designmatrix</code>	Design matrix for item difficulties $b$ to estimate linear logistic test models
<code>alpha.conv</code>	Convergence criterion for $\alpha$ parameter
<code>numdiff.parm</code>	Parameter for numerical differentiation
<code>numdiff.alpha.parm</code>	Parameter for numerical differentiation for $\alpha$ parameter
<code>distribution.trait</code>	Assumed trait distribution. The default is the normal distribution ("normal"). Log-linear smoothing of the trait distribution is also possible ("smooth2", "smooth3" or "smooth4" for smoothing up to 2, 3 or 4 moments respectively).
<code>Qmatrix</code>	The Q-matrix
<code>variance.fixed</code>	Matrix for fixing covariance matrix (See Examples)
<code>variance.init</code>	Optional initial covariance matrix
<code>mu.fixed</code>	Matrix for fixing mean vector (See Examples)
<code>irtmodel</code>	Specify estimable IRT models: <code>raschtype</code> (Rasch type model), <code>ramsay.qm</code> (Ramsay's quotient model), <code>npirt</code> (Nonparametric item response model). If <code>npirt</code> is used as the argument for <code>irtmodel</code> , the argument <code>npformula</code> specifies different item response functions in the R formula framework (like " $y \sim I(\theta^2)$ "); see Examples). For estimating the missing data item response model, use <code>irtmodel='missing1'</code> .
<code>npformula</code>	A string or a vector which contains R formula objects for specifying the item response function. For example, " $y \sim \theta$ " is the specification of the 2PL model (see Details). If <code>irtmodel="npirt"</code> and <code>npformula</code> is not specified, then an unrestricted item response functions on the grid of $\theta$ values is estimated.
<code>npirt.monotone</code>	Should nonparametrically estimated item response functions be monotone? The default is TRUE. This function applies only to <code>irtmodel='npirt'</code> and <code>npformula=NULL</code> .
<code>use.freqpatt</code>	A logical if frequencies of pattern should be used or not. The default is <code>is.null(group)</code> . This means that for single group analyses, frequency patterns are used but not for multiple groups. If data processing times are large, then <code>use.freqpatt=FALSE</code> is recommended.
<code>delta.miss</code>	Missingness parameter $\delta$ quantifying the meaning of responding to an item between the two extremes of ignoring missing responses and setting all missing responses to incorrect
<code>est.delta</code>	Vector with indices indicating the $\delta$ parameters to be estimated if <code>irtmodel="missing1"</code> .
<code>object</code>	Object of class <code>rasch.mml</code>
<code>x</code>	Object of class <code>rasch.mml</code>
<code>items</code>	Vector of integer or item names which should be plotted
<code>xlim</code>	Specification for <code>xlim</code> in plot
<code>main</code>	Title of the plot
<code>...</code>	Further arguments to be passed

**Details**

The item response function of the generalized item response model (`irtmodel="raschtype"`; Stukel, 1988) can be written as

$$P(X_{pi} = 1|\theta_{pd}) = c_i + (d_i - c_i)g_{\alpha_1, \alpha_2}[a_i(\theta_{pd} - b_i)]$$

where  $g$  is the generalized logistic link function depending on parameters  $\alpha_1$  and  $\alpha_2$ .

For the most important link functions the specifications are (Stukel, 1988):

logistic link function:  $\alpha_1 = 0$  and  $\alpha_2 = 0$

probit link function:  $\alpha_1 = 0.165$  and  $\alpha_2 = 0.165$

loglog link function:  $\alpha_1 = -0.037$  and  $\alpha_2 = 0.62$

cloglog link function:  $\alpha_1 = 0.62$  and  $\alpha_2 = -0.037$

See [pgenlogis](#) for exact transformation formulas of the mentioned link functions.

A  $D$ -dimensional model can also be specified but only allows for between item dimensionality (one item loads on one and only dimension). Setting  $c_i = 0$ ,  $d_i = 1$  and  $a_i = 1$  for all items  $i$ , an additive item response model

$$P(X_{pi} = 1|\theta_p) = g_{\alpha_1, \alpha_2}(\theta_p - b_i)$$

is estimated.

Ramsay's quotient model (`irtmodel="qm.ramsay"`) uses the item response function

$$P(X_{pi} = 1|\theta_p) = \frac{\exp(\theta_p/b_i)}{K_i + \exp(\theta_p/b_i)}$$

Quite general unidimensional item response models can be estimated in a nonparametric framework (`irtmodel="npirt"`). The response functions are a linear combination of transformed  $\theta$  values

$$\text{logit}[P(X_{pi} = 1|\theta_p)] = Y_\theta \beta$$

Where  $Y_\theta$  is a design matrix of  $\theta$  and  $\beta$  are item parameters to be estimated. The formula  $Y_\theta \beta$  can be specified in the R formula framework (see Simulated Example 3, Model 3c).

Pseudo-likelihood estimation can be conducted for fractional item response data as input (i.e. some item response  $x_{pi}$  do have values between 0 and 1). Then the pseudo-likelihood  $L_p$  for person  $p$  is defined as

$$L_p = \prod_i P_i(\theta_p)^{x_{pi}} [1 - P_i(\theta_p)]^{(1-x_{pi})}$$

Note that for dichotomous responses this term corresponds to the ordinary likelihood. See Example 7.

A special two-dimensional missing data item response model (`irtmodel="missing1"`) is implemented according to Mislavy and Wu (1996). Besides an unidimensional ability  $\theta_p$ , an individual response propensity  $\xi_p$  is proposed. We define item responses  $X_{pi}$  and response indicators  $R_{pi}$  indicating whether item responses  $X_{pi}$  are observed or not. Denoting the logistic function by  $L$ , the item response model for ability is defined as

$$P(X_{pi} = 1|\theta_p, \xi_p) = P(X_{pi} = 1|\theta_p) = L(\theta_p - b_i)$$

We also define a measurement model for response indicators  $R_{pi}$  which depends on the item response  $X_{pi}$  itself:

$$P(R_{pi} = 1|X_{pi} = k, \theta_p, \xi_p) = P(R_{pi} = 1|X_{pi} = k, \xi_p) = L[\xi_p - \beta_i - k\delta_i] \quad \text{for } k = 0, 1$$

If  $\delta_i = 0$ , then the probability of responding to an item is independent of the incompletely observed item  $X_{pi}$  which is an item response model with nonignorable missings (Holman & Glas, 2005; see also Pohl, Graefe & Rose, 2014). If  $\delta_i$  is a large negative number (e.g.  $\delta = -100$ ), then it follows  $P(R_{pi} = 1|X_{pi} = 1, \theta_p, \xi_p) = 1$  and as a consequence it holds that  $P(X_{pi} = 1|R_{pi} = 0, \theta_p, \xi_p) = 0$ , which is equivalent to treating all missing item responses as incorrect. The missingness parameter  $\delta$  can be specified by the user and studied as a sensitivity analysis under different missing not at random assumptions or can be estimated by choosing `est.delta=TRUE`.

## Value

A list with following entries

<code>dat</code>	Original data frame
<code>item</code>	Estimated item parameters in the generalized item response model
<code>item2</code>	Estimated item parameters for Ramsay's quotient model
<code>trait.distr</code>	Discretized ability distribution points and probabilities
<code>mean.trait</code>	Estimated mean vector
<code>sd.trait</code>	Estimated standard deviations
<code>skewness.trait</code>	Estimated skewnesses
<code>deviance</code>	Deviance
<code>pjk</code>	Estimated probabilities of item correct evaluated at <code>theta.k</code>
<code>rprobs</code>	Item response probabilities like in <code>pjk</code> , but slightly extended to accomodate all categories
<code>person</code>	Person parameter estimates: mode (MAP) and mean (EAP) of the posterior distribution
<code>pid</code>	Person identifier
<code>ability.est.pattern</code>	Response pattern estimates
<code>f.qk.yi</code>	Individual posterior distribution
<code>f.yi.qk</code>	Individual likelihood
<code>fixed.a</code>	Estimated $a$ parameters
<code>fixed.c</code>	Estimated $c$ parameters
<code>G</code>	Number of groups
<code>alpha1</code>	Estimated $\alpha_1$ parameter in generalized logistic item response model
<code>alpha2</code>	Estimated $\alpha_2$ parameter in generalized logistic item response model
<code>se.b</code>	Standard error of $b$ parameter in generalized logistic model or Ramsay's quotient model
<code>se.a</code>	Standard error of $a$ parameter in generalized logistic model

se.c	Standard error of $c$ parameter in generalized logistic model
se.d	Standard error of $d$ parameter in generalized logistic model
se.alpha	Standard error of $\alpha$ parameter in generalized logistic model
se.K	Standard error of $K$ parameter in Ramsay's quotient model
iter	Number of iterations
reliability	EAP reliability
irtmodel	Type of estimated item response model
D	Number of dimensions
mu	Mean vector (for multidimensional models)
Sigma.cov	Covariance matrix (for multidimensional models)
theta.k	Grid of discretized ability distributions
trait.weights	Fixed vector of probabilities for the ability distribution
pi.k	Trait distribution
ic	Information criteria
esttype	Estimation type: 11 (Log-Likelihood), pseudol1 (Pseudo-Log-Likelihood)
...	

### Note

Multiple group estimation is not possible for Ramsay's quotient model and multidimensional models.

### Author(s)

Alexander Robitzsch

### References

- Holman, R., & Glas, C. A. (2005). Modelling non-ignorable missing-data mechanisms with item response theory models. *British Journal of Mathematical and Statistical Psychology*, **58**, 1-17.
- Loken, E., & Rulison, K. L. (2010). Estimation of a four-parameter item response theory model. *British Journal of Mathematical and Statistical Psychology*, **63**, 509-525.
- Mislevy, R. J., & Wu, P. K. (1996). *Missing Responses and IRT Ability Estimation: Omits, Choice, Time Limits, and Adaptive Testing*. ETS Research Report ETS RR-96-30. Princeton, ETS.
- Pohl, S., Graefe, L., & Rose, N. (2014). Dealing with omitted and not-reached items in competence tests evaluating approaches accounting for missing responses in item response theory models. *Educational and Psychological Measurement*, **74**, 423-452.
- Ramsay, J. O. (1989). A comparison of three simple test theory models. *Psychometrika*, **54**, 487-499.
- Rossi, N., Wang, X., & Ramsay, J. O. (2002). Nonparametric item response function estimates with the EM algorithm. *Journal of Educational and Behavioral Statistics*, **27**, 291-317.
- Stukel, T. A. (1988). Generalized logistic models. *Journal of the American Statistical Association*, **83**, 426-431.

van der Maas, H. J. L., Molenaar, D., Maris, G., Kievit, R. A., & Borsboom, D. (2011). Cognitive psychology meets psychometric theory: On the relation between process models for decision making and latent variable models for individual differences. *Psychological Review*, **318**, 339-356.

### See Also

Simulate the generalized logistic Rasch model with [sim.raschtype](#).

Simulate Ramsay's quotient model with [sim.qm.ramsay](#).

Simulate locally dependent item response data using [sim.rasch.dep](#).

For an assessment of global model fit see [modelfit.sirt](#).

See [CDM::itemfit.sx2](#) for item fit statistics.

### Examples

```
#####
# EXAMPLE 1: Reading dataset
#####

data(data.read)
dat <- data.read
I <- ncol(dat) # number of items

# Rasch model
mod1 <- rasch.mml2( dat )
summary(mod1)
plot( mod1 ) # plot all items
# title 'Rasch model', display curves from -3 to 3 only for items 1, 5 and 8
plot(mod1, main="Rasch model Items 1, 5 and 8", xlim=c(-3,3) , items=c(1,5,8) )

# Rasch model with constraints on item difficulties
# set item parameters of A1 and C3 equal to -2
constraints <- data.frame( c("A1","C3") , c(-2,-2) )
mod1a <- rasch.mml2( dat , constraints=constraints)
summary(mod1a)

# estimate equal item parameters for 1st and 11th item
est.b <- 1:I
est.b[11] <- 1
mod1b <- rasch.mml2( dat , est.b = est.b )
summary(mod1b)

# estimate Rasch model with skew trait distribution
mod1c <- rasch.mml2( dat , distribution.trait="smooth3")
summary(mod1c)

# 2PL model
mod2 <- rasch.mml2( dat , est.a = 1:I )
summary(mod2)
plot(mod2) # plot 2PL item response curves

# extract individual likelihood
```





```

summary(mod9)

#####
# EXAMPLE 2: Longitudinal data
#####

data(data.long)
dat <- data.long[,-1]

# define Q loading matrix
Qmatrix <- matrix( 0 , 12 , 2 )
Qmatrix[1:6,1] <- 1 # T1 items
Qmatrix[7:12,2] <- 1 # T2 items

# define restrictions on item difficulties
est.b <- c(1,2,3,4,5,6, 3,4,5,6,7,8)
mu.fixed <- cbind(1,0)
# set first mean to 0 for identification reasons

# Model 1: 2-dimensional Rasch model
mod1 <- rasch.mml2( dat , Qmatrix=Qmatrix , miterstep=4,
  est.b = est.b , mu.fixed = mu.fixed , mmliter=30 )
summary(mod1)
plot(mod1)
## Plot function is only applicable for unidimensional models

## End(Not run)

#####
# SIMULATED EXAMPLE 3
# one group, estimation of alpha parameter in the generalized logistic model
#####

# simulate theta values
set.seed(786)
N <- 1000 # number of persons
theta <- rnorm( N , sd =1.5 ) # N persons with SD 1.5
b <- seq( -2 , 2 , len=15)

# simulate data
dat <- sim.raschtype( theta = theta , b = b , alpha1 = 0 , alpha2 = -0.3 )

# estimating alpha parameters
mod1 <- rasch.mml2( dat , est.alpha = TRUE , mmliter=30 )
summary(mod1)
plot(mod1)

## Not run:
# fixed alpha parameters
mod1b <- rasch.mml2( dat , est.alpha = FALSE , alpha1=0 , alpha2=-.3 )
summary(mod1b)

# estimation with equal alpha parameters

```

```

mod1c <- rasch.mml2( dat , est.alpha = TRUE , equal.alpha=TRUE )
summary(mod1c)

# Ramsay QM
mod2a <- rasch.mml2( dat , irtmodel ="ramsay.qm" )
summary(mod2a)

## End(Not run)

# Ramsay QM with estimated K parameters
mod2b <- rasch.mml2( dat , irtmodel ="ramsay.qm" , est.K=1:15 , mmliter=30)
summary(mod2b)
plot(mod2b)

## Not run:
# nonparametric estimation of monotone item response curves
mod3a <- rasch.mml2( dat , irtmodel ="npirt" , mmliter =100 ,
                    theta.k = seq( -3 , 3 , len=10) ) # evaluations at 10 theta grid points
# nonparametric ICC of first 4 items
round( t(mod3a$pjk)[1:4,] , 3 )
summary(mod3a)
plot(mod3a)

# nonparametric IRT estimation without monotonicity assumption
mod3b <- rasch.mml2( dat , irtmodel ="npirt" , mmliter =10 ,
                    theta.k = seq( -3 , 3 , len=10) , npirt.monotone=FALSE)
plot(mod3b)

# B-Spline estimation of ICCs
library(splines)
mod3c <- rasch.mml2( dat , irtmodel ="npirt" ,
                    npformula = "y~bs(theta,df=3)" , theta.k = seq(-3,3,len=15) )
summary(mod3c)
round( t(mod3c$pjk)[1:6,] , 3 )
plot(mod3c)

# estimation of quadratic item response functions: ~ theta + I( theta^2)
mod3d <- rasch.mml2( dat , irtmodel ="npirt" ,
                    npformula = "y~theta + I(theta^2)" )
summary(mod3d)
plot(mod3d)

# estimation of a stepwise ICC function
# ICCs are constant on the theta domains: [-Inf,-1], [-1,1], [1,Inf]
mod3e <- rasch.mml2( dat , irtmodel ="npirt" ,
                    npformula = "y~I(theta>-1 )+I(theta>1)" )
summary(mod3e)
plot(mod3e , xlim=c(-2.5,2.5) )

# 2PL model
mod4 <- rasch.mml2( dat , est.a=1:15)
summary(mod4)

```

```
#####
# SIMULATED EXAMPLE 4
# two groups, estimation of generalized logistic model
#####

# simulate generalized logistic Rasch model in two groups
set.seed(8765)
N1 <- 1000      # N1=1000 persons in group 1
N2 <- 500      # N2= 500 persons in group 2
dat1 <- sim.raschtype( theta = rnorm( N1 , sd = 1.5 ) , b = b ,
alpha1 = -0.3 , alpha2=0)
dat2 <- sim.raschtype( theta = rnorm( N2 , mean=-.5 , sd =.75) ,
b = b , alpha1 = -0.3 , alpha2=0)
dat1 <- rbind( dat1 , dat2 )
group <- c( rep(1,N1) , rep(2,N2))

mod1 <- rasch.mml2( dat1 , parm.conv=.0001 , group=group , est.alpha = TRUE )
summary(mod1)

#####
# SIMULATED EXAMPLE 5: Multidimensional model
#####

***
# (1) simulate data
set.seed(785)
library(mvtnorm)
N <- 500
theta <- mvtnorm::rmvnorm( N,mean=c(0,0) , sigma=matrix( c(1.45,.5,.5,1.7) , 2 , 2 ))
I <- 10
# 10 items load on the first dimension
p1 <- plogis( outer( theta[,1] , seq( -2 , 2 , len=I ) , "-" ) )
resp1 <- 1 * ( p1 > matrix( runif( N*I ) , nrow=N , ncol=I ) )
# 10 items load on the second dimension
p1 <- plogis( outer( theta[,2] , seq( -2 , 2 , len=I ) , "-" ) )
resp2 <- 1 * ( p1 > matrix( runif( N*I ) , nrow=N , ncol=I ) )
#Combine the two sets of items into one response matrix
resp <- cbind(resp1,resp2)
colnames(resp) <- paste("I" , 1:(2*I), sep="")
dat <- resp

# define Q-matrix
Qmatrix <- matrix( 0 , 2*I , 2 )
Qmatrix[1:I,1] <- 1
Qmatrix[1:I+I,2] <- 1

***
# (2) estimation of models
# 2-dimensional Rasch model
mod1 <- rasch.mml2( dat , Qmatrix=Qmatrix )
summary(mod1)

# 2-dimensional 2PL model
```

```

mod2 <- rasch.mml2( dat , Qmatrix=Qmatrix , est.a = 1:(2*I) )
summary(mod2)

# estimation with some fixed variances and covariances
# set variance of 1st dimension to 1 and
# covariance to zero
variance.fixed <- matrix( cbind(c(1,1) , c(1,2) , c(1,0)) ,
                          byrow=FALSE , ncol= 3 )
mod3 <- rasch.mml2( dat , Qmatrix=Qmatrix , variance.fixed = variance.fixed )
summary(mod3)

# constraints on item difficulties
# useful for example in longitudinal linking
est.b <- c( 1:I , 1:I )
# equal indices correspond to equally estimated item parameters
mu.fixed <- cbind( 1 , 0 )
mod4 <- rasch.mml2( dat, Qmatrix=Qmatrix, est.b = est.b , mu.fixed = mu.fixed )
summary(mod4)

#####
# SIMULATED EXAMPLE 6: Two booklets with same items but with item context effects.
# Therefore, item slopes and item difficulties are assumed to be shifted in the
# second design group.
#####

***
# simulate data
set.seed(987)
I <- 10 # number of items
# define person design groups 1 and 2
n1 <- 700
n2 <- 1500
# item difficulties group 1
b1 <- seq(-1.5,1.5,length=I)
# item slopes group 1
a1 <- rep(1, I)
# simulate data group 1
dat1 <- sim.raschtype( rnorm(n1) , b=b1 , fixed.a=a1 )
colnames(dat1) <- paste0("I" , 1:I , "des1" )
# group 2
b2 <- b1 - .15
a2 <- 1.1*a1
# Item parameters are slightly transformed in the second group
# compared to the first group. This indicates possible item context effects.

# simulate data group 2
dat2 <- sim.raschtype( rnorm(n2) , b=b2 , fixed.a=a2 )
colnames(dat2) <- paste0("I" , 1:I , "des2" )
# define joint dataset
dat <- matrix( NA , nrow=n1+n2 , ncol=2*I)
colnames(dat) <- c( colnames(dat1) , colnames(dat2) )
dat[ 1:n1 , 1:I ] <- dat1
dat[ n1 + 1:n2 , I + 1:I ] <- dat2

```

```

# define group identifier
group <- c( rep(1,n1) , rep(2,n2) )

####
# Model 1: Rasch model two groups
itemindex <- rep( 1:I , 2 )
mod1 <- rasch.mml2( dat , group=group , est.b=itemindex )
summary(mod1)

####
# Model 2: two item slope groups and designmatrix for intercepts
designmatrix <- matrix( 0 , 2*I , I+1)
designmatrix[ ( 1:I )+ I,1:I] <- designmatrix[1:I ,1:I] <- diag(I)
designmatrix[ ( 1:I )+ I,I+1] <- 1
mod2 <- rasch.mml2( dat , est.a=rep(1:2,each=I) , designmatrix=designmatrix )
summary(mod2)

#####
# EXAMPLE 7: PIRLS dataset with missing responses
#####

data(data.pirlsmissing)
items <- grep( "R31" , colnames(data.pirlsmissing) , value=TRUE )
I <- length(items)
dat <- data.pirlsmissing

####
# Model 1: recode missing responses as missing (missing are ignorable)

# data recoding
dat1 <- dat
dat1[ dat1 == 9 ] <- NA
# estimate Rasch model
mod1 <- rasch.mml2( dat1[,items] , weights= dat$studwgt , group=dat$country )
summary(mod1)
## Mean= 0 0.341 -0.134 0.219
## SD= 1.142 1.166 1.197 0.959

####
# Model 2: recode missing responses as wrong

# data recoding
dat2 <- dat
dat2[ dat2 == 9 ] <- 0
# estimate Rasch model
mod2 <- rasch.mml2( dat2[,items] , weights= dat$studwgt , group=dat$country )
summary(mod2)
## Mean= 0 0.413 -0.172 0.446
## SD= 1.199 1.263 1.32 0.996

####
# Model 3: recode missing responses as rho * P_i( theta ) and
# apply pseudo-log-likelihood estimation

```

```

# Missing item responses are predicted by the model implied probability
# P_i( theta ) where theta is the ability estimate when ignoring missings (Model 1)
# and rho is an adjustment parameter. rho=0 is equivalent to Model 2 (treating
# missing as wrong) and rho=1 is equivalent to Model 1 (treating missing as ignorable).

# data recoding
dat3 <- dat
# simulate theta estimate from posterior distribution
theta <- rnorm( nrow(dat3) , mean = mod1$person$EAP , sd=mod1$person$SE.EAP )
rho <- .3 # define a rho parameter value of .3
for (ii in items){
  ind <- which( dat[,ii] == 9 )
  dat3[ind,ii] <- rho*plogis( theta[ind] - mod1$item$b[ which( items == ii ) ] )
}

# estimate Rasch model
mod3 <- rasch.mml2( dat3[,items] , weights= dat$studwgt , group=dat$country )
summary(mod3)
## Mean= 0 0.392 -0.153 0.38
## SD= 1.154 1.209 1.246 0.973

#####
# Model 4: simulate missing responses as rho * P_i( theta )
# The definition is the same as in Model 3. But it is now assumed
# that the missing responses are 'latent responses'.
set.seed(789)

# data recoding
dat4 <- dat
# simulate theta estimate from posterior distribution
theta <- rnorm( nrow(dat4) , mean = mod1$person$EAP , sd=mod1$person$SE.EAP )
rho <- .3 # define a rho parameter value of .3
for (ii in items){
  ind <- which( dat[,ii] == 9 )
  p3 <- rho*plogis( theta[ind] - mod1$item$b[ which( items == ii ) ] )
  dat4[ ind , ii ] <- 1*(runif( length(ind) , 0 , 1 ) < p3)
}

# estimate Rasch model
mod4 <- rasch.mml2( dat4[,items] , weights= dat$studwgt , group=dat$country )
summary(mod4)
## Mean= 0 0.396 -0.156 0.382
## SD= 1.16 1.216 1.253 0.979

#####
# Model 5: recode missing responses for multiple choice items with four alternatives
# to 1/4 and apply pseudo-log-likelihood estimation.
# Missings for constructed response items are treated as incorrect.

# data recoding
dat5 <- dat
items_mc <- items[ substring( items , 7,7) == "M" ]
items_cr <- items[ substring( items , 7,7) == "C" ]

```

```

for (ii in items_mc){
  ind <- which( dat[,ii] == 9 )
  dat5[ind,ii] <- 1/4
}
for (ii in items_cr){
  ind <- which( dat[,ii] == 9 )
  dat5[ind,ii] <- 0
}

# estimate Rasch model
mod5 <- rasch.mml2( dat5[,items] , weights= dat$studwgt , group=dat$country )
summary(mod5)
## Mean= 0 0.411 -0.165 0.435
## SD= 1.19 1.245 1.293 0.995

**** For the following analyses, we ignore sample weights and the
# country grouping.
data(data.pirlsmissing)
items <- grep( "R31" , colnames(data.pirlsmissing) , value=TRUE )
dat <- data.pirlsmissing
dat1 <- dat
dat1[ dat1 == 9 ] <- 0

**** Model 6: estimate item difficulties assuming incorrect missing data treatment
mod6 <- rasch.mml2( dat1[,items] , mmliter=50 )
summary(mod6)

**** Model 7: reestimate model with constrained item difficulties
I <- length(items)
constraints <- cbind( 1:I , mod6$item$b )
mod7 <- rasch.mml2( dat1[,items] , constraints=constraints , mmliter=50 )
summary(mod7)

**** Model 8: score all missings responses as missing items
dat2 <- dat[,items]
dat2[ dat2 == 9 ] <- NA
mod8 <- rasch.mml2( dat2 , constraints=constraints , mmliter=50 , mu.fixed=NULL )
summary(mod8)

**** Model 9: estimate missing data model 'missing1' assuming a missingness
# parameter delta.miss of zero
dat2 <- dat[,items] # note that missing item responses must be defined by 9
mod9 <- rasch.mml2( dat2 , constraints=constraints , irtmodel="missing1" ,
  theta.k=seq(-5,5,len=10) , delta.miss= 0 , mitermax= 4 , mmliter=200 ,
  mu.fixed=NULL )
summary(mod9)

**** Model 10: estimate missing data model with a large negative missing delta parameter
# => This model is equivalent to treating missing responses as wrong
mod10 <- rasch.mml2( dat2 , constraints=constraints , irtmodel="missing1" ,
  theta.k=seq(-5 , 5 , len=10) , delta.miss= -10 , mitermax= 4 , mmliter=200 ,
  mu.fixed=NULL )
summary(mod10)

```



```

**** Model 11: choose a missingness delta parameter of -1
mod11 <- rasch.mml2( dat2 , constraints=constraints , irtmodel="missing1" ,
                    theta.k=seq(-5 , 5 , len=10) , delta.miss= -1 , mitermax= 4 ,
                    mmliter=200 , mu.fixed=NULL )
summary(mod11)

**** Model 12: estimate joint delta parameter
mod12 <- rasch.mml2( dat2 , irtmodel="missing1" , mu.fixed = cbind( c(1,2) , 0 ) ,
                    theta.k=seq(-8 , 8 , len=10) , delta.miss= 0 , mitermax= 4 ,
                    mmliter=30 , est.delta=rep(1,I) )
summary(mod12)

**** Model 13: estimate delta parameter in item groups defined by item format
est.delta <- 1 + 1 * ( substring( colnames(dat2),7,7 ) == "M" )
mod13 <- rasch.mml2( dat2 , irtmodel="missing1" , mu.fixed = cbind( c(1,2) , 0 ) ,
                    theta.k=seq(-8 , 8 , len=10) , delta.miss= 0 , mitermax= 4 ,
                    mmliter=30 , est.delta=est.delta )
summary(mod13)

**** Model 14: estimate item specific delta parameter
mod14 <- rasch.mml2( dat2 , irtmodel="missing1" , mu.fixed = cbind( c(1,2) , 0 ) ,
                    theta.k=seq(-8 , 8 , len=10) , delta.miss= 0 , mitermax= 4 ,
                    mmliter=30 , est.delta= 1:I )
summary(mod14)

#####
# EXAMPLE 8: Comparison of different models for polytomous data
#####

data(data.Students, package="CDM")
head(data.Students)
dat <- data.Students[ , paste0("act",1:5) ]
I <- ncol(dat)

*****
**** Model 1: Partial Credit Model (PCM)

**** Model 1a: PCM in TAM
mod1a <- TAM::tam.mml( dat )
summary(mod1a)

**** Model 1b: PCM in sirt
mod1b <- rm.facets( dat )
summary(mod1b)

**** Model 1c: PCM in mirt
mod1c <- mirt::mirt( dat , 1 , itemtype = rep("Rasch",I) , verbose=TRUE )
print(mod1c)

*****
**** Model 2: Sequential Model (SM): Equal Loadings

```

```

#### Model 2a: SM in sirt
dat1 <- CDM::sequential.items(dat)
resp <- dat1$dat.expand
iteminfo <- dat1$iteminfo
# fit model
mod2a <- rasch.mml2( resp )
summary(mod2a)

#####
#### Model 3: Sequential Model (SM): Different Loadings

#### Model 3a: SM in sirt
mod3a <- rasch.mml2( resp , est.a= iteminfo$itemindex )
summary(mod3a)

#####
#### Model 4: Generalized partial credit model (GPCM)

#### Model 4a: GPCM in TAM
mod4a <- TAM::tam.mml.2pl( dat , irtmodel="GPCM")
summary(mod4a)

#####
#### Model 5: Graded response model (GRM)

#### Model 5a: GRM in mirt
mod5a <- mirt::mirt( dat , 1 , itemtype= rep("graded",I) , verbose=TRUE)
print(mod5a)

# model comparison
logLik(mod1a);logLik(mod1b);mod1c@logLik # PCM
logLik(mod2a) # SM (Rasch)
logLik(mod3a) # SM (GPCM)
logLik(mod4a) # GPCM
mod5a@logLik # GRM

## End(Not run)

```

---

 rasch.pairwise

*Pairwise Estimation Method of the Rasch Model*


---

## Description

This function estimates the Rasch model with a minimum chi square estimation method (cited in Fischer, 2007, p. 544) which is a pairwise conditional likelihood estimation approach.

## Usage

```

rasch.pairwise(dat, conv = 1e-04, maxiter = 3000, progress = TRUE,
              b.init = NULL)

```

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

### Arguments

dat	An $N \times I$ data frame of dichotomous item responses
conv	Convergence criterion
maxiter	Maximum number of iterations
progress	Display iteration progress?
b.init	An optional vector of length $I$ of item difficulties
object	Object of class <code>rasch.pairwise</code>
...	Further arguments to be passed

### Value

An object of class `rasch.pairwise` with following entries

b	Item difficulties
eps	Exponentiated item difficulties, i.e. $\text{eps}=\exp(-b)$
iter	Number of iterations
conv	Convergence criterion
dat	Original data frame
freq.ij	Frequency table of all item pairs
item	Summary table of item parameters

### Author(s)

Alexander Robitzsch

### References

Fischer, G. H. (2007). Rasch models. In C. R. Rao and S. Sinharay (Eds.), *Handbook of Statistics*, Vol. 26 (pp. 515-585). Amsterdam: Elsevier.

### See Also

See [summary.rasch.pairwise](#) for a summary.

A slightly different implementation of this conditional pairwise method is implemented in [rasch.pairwise.itemcluster](#).

Pairwise marginal likelihood estimation (also labeled as pseudolikelihood estimation) can be conducted with [rasch.pml3](#).

**Examples**

```
#####
# EXAMPLE 1: Reading data set | pairwise estimation Rasch model
#####

data(data.read)
mod <- rasch.pairwise( data.read )
summary(mod)
```

---

```
rasch.pairwise.itemcluster
```

*Pairwise Estimation of the Rasch Model for Locally Dependent Items*

---

**Description**

This function uses pairwise conditional likelihood estimation for estimating item parameters in the Rasch model.

**Usage**

```
rasch.pairwise.itemcluster(dat, itemcluster = NULL, b.fixed=NULL ,
  conv = 1e-05, maxiter = 3000, progress = TRUE, b.init = NULL)
```

**Arguments**

<code>dat</code>	An $N \times I$ data frame. Missing responses are allowed and must be recoded as NA.
<code>itemcluster</code>	Optional integer vector of itemcluster (see Examples). Different integers correspond to different item clusters. No item cluster is set as default.
<code>b.fixed</code>	Matrix for fixing item parameters. The first columns contains the item (number or name), the second column the parameter to be fixed.
<code>conv</code>	Convergence criterion in maximal absolute parameter change
<code>maxiter</code>	Maximal number of iterations
<code>progress</code>	A logical which displays progress. Default is TRUE.
<code>b.init</code>	Vector of initial item difficulty estimates. Default is NULL.

**Details**

This is an adaptation of the algorithm of van der Linden and Eggen (1986). Only item pairs of different item clusters are taken into account for item difficulty estimation. Therefore, the problem of locally dependent items within each itemcluster is (almost) eliminated (see Examples below) because contributions of local dependencies do not appear in the pairwise likelihood terms. In detail, the estimation rests on observed frequency tables of items  $i$  and  $j$  and therefore on conditional probabilities

$$\frac{P(X_i = x, X_j = y)}{P(X_i + X_j = 1)} \quad \text{with } x, y = 0, 1 \quad \text{and } x + y = 1$$

If for some item pair  $(i, j)$  a higher positive (or negative) correlation is expected (i.e. deviation from local dependence), then this pair is removed from estimation. Clearly, there is a loss in precision but item parameters can be less biased.

**Value**

Object of class `rasch.pairwise` with elements

- `b`                      Vector of item difficulties
- `item`                   Data frame of item paramters ( $N, p$  and item difficulty)

**Note**

No standard errors are provided by this function. Use resampling methods for conducting statistical inference.

Formulas for asymptotic standard errors of this pairwise estimation method are described in Zwinderman (1995).

**Author(s)**

Alexander Robitzsch

**References**

van der Linden, W. J., & Eggen, T. J. H. M. (1986). *An empirical Bayes approach to item banking*. Research Report 86-6, University of Twente.

Zwinderman, A. H. (1995). Pairwise parameter estimation in Rasch models. *Applied Psychological Measurement*, **19**, 369-375.

**See Also**

[rasch.pairwise](#), [summary.rasch.pairwise](#),  
 Pairwise marginal likelihood estimation (also labeled as pseudolikelihood estimation) can be conducted with [rasch.pml3](#).  
 Other estimation methods are implemented in [rasch.copula2](#) or [rasch.mml2](#).  
 For simulation of locally dependent data see [sim.rasch.dep](#).

**Examples**

```
#####
# EXAMPLE 1: Example with locally dependent items
#      12 Items: Cluster 1 -> Items 1,...,4
#              Cluster 2 -> Items 6,...,9
#              Cluster 3 -> Items 10,11,12
#####

set.seed(7896)
I <- 12           # number of items
n <- 5000        # number of persons
```

```

b <- seq(-2,2, len=I)           # item difficulties
bsamp <- b <- sample(b)         # sample item difficulties
theta <- rnorm( n , sd = 1 )    # person abilities
# itemcluster
itemcluster <- rep(0,I)
itemcluster[ 1:4 ] <- 1
itemcluster[ 6:9 ] <- 2
itemcluster[ 10:12 ] <- 3
# residual correlations
rho <- c( .55 , .25 , .45 )

# simulate data
dat <- sim.rasch.dep( theta , b , itemcluster , rho )
colnames(dat) <- paste("I" , seq(1,ncol(dat)) , sep="")

# estimation with pairwise Rasch model
mod3 <- rasch.pairwise( dat )
summary(mod3)

# use item cluster in rasch pairwise estimation
mod <- rasch.pairwise.itemcluster( dat = dat , itemcluster = itemcluster )
summary(mod)

## Not run:
# Rasch MML estimation
mod4 <- rasch.mml2( dat )
summary(mod4)

# Rasch Copula estimation
mod5 <- rasch.copula2( dat , itemcluster = itemcluster )
summary(mod5)

# compare different item parameter estimates
M1 <- cbind( "true.b"=bsamp , "b.rasch" = mod4$item$b ,
            "b.rasch.copula" = mod5$item$thresh ,
            "b.rasch.pairwise" = mod3$b ,
            "b.rasch.pairwise.cluster" = mod$b
          )
# center item difficulties
M1 <- scale( M1 , scale=FALSE )
round( M1 , 3 )
round( apply( M1 , 2 , sd ) , 3 )

# Below the output of the example is presented.
# It is surprising that the rasch.pairwise.itemcluster is pretty close
# to the estimate in the Rasch copula model.

## > M1 <- scale( M1 , scale=F )
## > round( M1 , 3 )
##      true.b b.rasch b.rasch.copula b.rasch.pairwise b.rasch.pairwise.cluster
## I1  0.545  0.561      0.526      0.628      0.524
## I2 -0.182 -0.168     -0.174     -0.121     -0.156
## I3 -0.909 -0.957     -0.867     -0.971     -0.899

```

```

## I4 -1.636 -1.726 -1.625 -1.765 -1.611
## I5 1.636 1.751 1.648 1.694 1.649
## I6 0.909 0.892 0.836 0.898 0.827
## I7 -2.000 -2.134 -2.020 -2.051 -2.000
## I8 -1.273 -1.355 -1.252 -1.303 -1.271
## I9 -0.545 -0.637 -0.589 -0.581 -0.598
## I10 1.273 1.378 1.252 1.308 1.276
## I11 0.182 0.241 0.226 0.109 0.232
## I12 2.000 2.155 2.039 2.154 2.026
## > round( apply( M1 , 2 , sd ) , 3 )
## true.b b.rasch b.rasch.copula
## 1.311 1.398 1.310
## b.rasch.pairwise b.rasch.pairwise.cluster
## 1.373 1.310
## End(Not run)

# set item parameters of first item to 0 and of second item to -0.7
b.fixed <- cbind( c(1,2) , c(0,-.7) )
mod5 <- rasch.pairwise.itemcluster( dat = dat , b.fixed=b.fixed,
itemcluster = itemcluster )
# difference between estimations 'mod' and 'mod5'
dfr <- cbind( mod5$item$b , mod$item$b )
plot( mod5$item$b , mod$item$b , pch=16)
apply( dfr , 1 , diff )

```

---

rasch.pml2

*Pairwise Marginal Likelihood Estimation for the Probit Rasch Model*


---

## Description

This function estimates unidimensional 1PL and 2PL models with the probit link using pairwise marginal maximum likelihood estimation (PMML; Renard, Molenberghs & Geys, 2004). Item pairs within an itemcluster can be excluded from the pairwise likelihood (argument `itemcluster`). The other alternative is to model a residual error structure with itemclusters (argument `error.corr`).

## Usage

```

rasch.pml3(dat, est.b = seq(1, ncol(dat)), est.a=rep(0,ncol(dat)) ,
est.sigma = TRUE, itemcluster = NULL,
weight = rep(1, nrow(dat)), numdiff.parm = 0.001,
b.init = NULL, a.init=NULL , sigma.init = NULL,
error.corr = 0*diag( 1 , ncol(dat) ) ,
err.constraintM=NULL , err.constraintV=NULL ,
glob.conv = 10^(-6), conv1 = 10^(-4), pmliter = 300, progress = TRUE,
use.maxincrement=TRUE )

rasch.pml2(dat, est.b = seq(1, ncol(dat)), est.a=rep(0,ncol(dat)) ,
est.sigma = TRUE, itemcluster = NULL,
weight = rep(1, nrow(dat)), numdiff.parm = 0.001,

```

```

b.init = NULL, a.init=NULL , sigma.init = NULL,
error.corr = 0*diag( 1 , ncol(dat) ) ,
err.constraintM=NULL , err.constraintV=NULL ,
glob.conv = 10^(-6), conv1 = 10^(-4), pmliter = 300, progress = TRUE)

```

```

## S3 method for class 'rasch.pml'
summary(object,...)

```

## Arguments

<code>dat</code>	An $N \times I$ data frame of dichotomous item responses
<code>est.b</code>	Vector of integers of length $I$ . Same integers mean that the corresponding items do have the same item difficulty $b$ . Entries of 0 mean fixing item parameters to values specified in <code>b.init</code> .
<code>est.a</code>	Vector of integers of length $I$ . Same integers mean that the corresponding items do have the same item slope $a$ . Entries of 0 mean fixing item parameters to values specified in <code>a.init</code> .
<code>est.sigma</code>	Should sigma (the trait standard deviation) be estimated? The default is TRUE.
<code>itemcluster</code>	Optional vector of length $I$ of integers which indicates itemclusters. Same integers correspond to the same itemcluster. An entry of 0 correspond to an item which is not included in any itemcluster.
<code>weight</code>	Optional vector of person weights
<code>numdiff.parm</code>	Step parameter for numerical differentiation
<code>b.init</code>	Initial or fixed item difficulty
<code>a.init</code>	Initial or fixed item slopes
<code>sigma.init</code>	Initial or fixed trait standard deviation
<code>error.corr</code>	An optional $I \times I$ integer matrix which defines the estimation of residual correlations. Entries of zero indicate that the corresponding residual correlation should not be estimated. Integers which differ from zero indicate correlations to be estimated. All entries with an equal integer are estimated by the same residual correlation. The default of <code>error.corr</code> is a diagonal matrix which means that no residual correlation is estimated. If <code>error.corr</code> deviates from this default, then the argument <code>itemcluster</code> is set to NULL. If some error correlations are estimated, then no itempairs in <code>itemcluster</code> can be excluded from the pairwise modeling.
<code>err.constraintM</code>	An optional $P \times L$ matrix where $P$ denotes the number of item pairs in pseudolikelihood estimation and $L$ is the number of linear constraints for residual correlations (see Details).
<code>err.constraintV</code>	An optional $L \times 1$ matrix with specified values for linear constraints on residual correlations (see Details).
<code>glob.conv</code>	Global convergence criterion
<code>conv1</code>	Convergence criterion for model parameters



pmliter	Maximum number of iterations
progress	Display progress?
use.maxincrement	Optional logical whether increments in slope parameters should be controlled in size in iterations. The default is TRUE.
object	Object of class <code>rasch.pml</code>
...	Further arguments to be passed

### Details

The probit item response model can be estimated with this function:

$$P(X_{pi} = 1|\theta_p) = \Phi(a_i\theta_p - b_i) \quad , \quad \theta_p \sim N(0, \sigma^2)$$

where  $\Phi$  denotes the normal distribution function. This model can also be expressed as a latent variable model which assumes a latent response tendency  $X_{pi}^*$  which is equal to 1 if  $X_{pi} > -b_i$  and otherwise zero. If  $\epsilon_{pi}$  is standard normally distributed, then

$$X_{pi}^* = a_i\theta_p - b_i + \epsilon_{pi}$$

An arbitrary pattern of residual correlations between  $\epsilon_{pi}$  and  $\epsilon_{pj}$  for item pairs  $i$  and  $j$  can be imposed using the `error.corr` argument.

Linear constraints  $Me = v$  on residual correlations  $e = Cov(\epsilon_{pi}, \epsilon_{pj})_{ij}$  (in a vectorized form) can be specified using the arguments `err.constraintM` (matrix  $M$ ) and `err.constraintV` (vector  $v$ ). The estimation is described in Neuhaus (1996).

For the pseudo likelihood information criterion (PLIC) see Stanford and Raftery (2002).

### Value

A list with following entries:

item	Data frame with estimated item parameters
iter	Number of iterations
deviance	Pseudolikelihood multiplied by minus 2
b	Estimated item difficulties
sigma	Estimated standard deviation
dat	Original dataset
ic	Data frame with information criteria (sample size, number of estimated parameters, pseudolikelihood information criterion PLIC)
link	Used link function (only probit is permitted)
itempairs	Estimated statistics of item pairs
error.corr	Estimated error correlation matrix
eps.corr	Vectorized error correlation matrix
omega.rel	Reliability of the sum score according to Green and Yang (2009). If some item pairs are excluded in the estimation, the residual correlation for these item pairs is assumed to be zero.
...	

**Note**

This function needs the **combinat** library.

rasch.pml2 uses the pbivnorm function from the **pbivnorm** package but rasch.pml3 uses the [pbivnorm2](#) from this package (**sirt**) and is somewhat faster.

**Author(s)**

Alexander Robitzsch

**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.

Neuhauser, W. (1996). Optimal estimation under linear constraints. *Astin Bulletin*, **26**, 233-245.

Renard, D., Molenberghs, G., & Geys, H. (2004). A pairwise likelihood approach to estimation in multilevel probit models. *Computational Statistics & Data Analysis*, **44**, 649-667.

Stanford, D. C., & Raftery, A. E. (2002). Approximate Bayes factors for image segmentation: The pseudolikelihood information criterion (PLIC). *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **24**, 1517-1520.

**See Also**

Get a summary of rasch.pml2 with [summary.rasch.pml](#).

For simulation of locally dependent items see [sim.rasch.dep](#).

For pairwise conditional likelihood estimation see [rasch.pairwise](#) or [rasch.pairwise.itemcluster](#).

For an assessment of global model fit see [modelfit.sirt](#).

**Examples**

```
#####
# EXAMPLE 1: Reading data set
#####

data(data.read)
dat <- data.read

#####
# Model 1: Rasch model with PML estimation
mod1 <- rasch.pml3( dat )
summary(mod1)

#####
# Model 2: Excluding item pairs with local dependence
#           from bivariate composite likelihood
itemcluster <- rep( 1:3 , each=4)
mod2 <- rasch.pml3( dat , itemcluster = itemcluster )
summary(mod2)
```

```

## Not run:
#####
# Model 3: Modelling error correlations:
#      joint residual correlations for each itemcluster
error.corr <- diag(1,ncol(dat))
for ( ii in 1:3){
  ind.ii <- which( itemcluster == ii )
  error.corr[ ind.ii , ind.ii ] <- ii
}
# estimate the model with error correlations
mod3 <- rasch.pml3( dat , error.corr = error.corr )
summary(mod3)

#####
# Model 4: model separate residual correlations
I <- ncol(error.corr)
error.corr1 <- matrix( 1:(I*I) , ncol= I )
error.corr <- error.corr1 * ( error.corr > 0 )
# estimate the model with error correlations
mod4 <- rasch.pml3( dat , error.corr = error.corr )
summary(mod4)

#####
# Model 5: assume equal item difficulties:
# b_1 = b_7 and b_2 = b_12
# fix item difficulty of the 6th item to .1
est.b <- 1:I
est.b[7] <- 1; est.b[12] <- 2 ; est.b[6] <- 0
b.init <- rep( 0, I ) ; b.init[6] <- .1
mod5 <- rasch.pml3( dat , est.b =est.b , b.init=b.init)
summary(mod5)

#####
# Model 6: estimate three item slope groups
est.a <- rep(1:3 , each=4 )
mod6 <- rasch.pml3( dat , est.a =est.a , est.sigma=0)
summary(mod6)

#####
# EXAMPLE 2: PISA reading
#####

data(data.pisaRead)
dat <- data.pisaRead$data

# select items
dat <- dat[ , substring(colnames(dat),1,1)=="R" ]

#####
# Model 1: Rasch model with PML estimation
mod1 <- rasch.pml3( as.matrix(dat) )
## Trait SD (Logit Link) : 1.419

```

```

#####
# Model 2: Model correlations within testlets
error.corr <- diag(1,ncol(dat))
testlets <- paste( data.pisaRead$item$testlet )
itemcluster <- match( testlets , unique(testlets) )
for ( ii in 1:(length(unique(testlets))) ){
  ind.ii <- which( itemcluster == ii )
  error.corr[ ind.ii , ind.ii ] <- ii
}
# estimate the model with error correlations
mod2 <- rasch.pml3( dat , error.corr = error.corr )
## Trait SD (Logit Link) : 1.384

#####
# Model 3: model separate residual correlations
I <- ncol(error.corr)
error.corr1 <- matrix( 1:(I*I) , ncol= I )
error.corr <- error.corr1 * ( error.corr > 0 )
# estimate the model with error correlations
mod3 <- rasch.pml3( dat , error.corr = error.corr )
## Trait SD (Logit Link) : 1.384

#####
# SIMULATED EXAMPLE 3: 10 locally independent items
#####

#####
# simulate some data
set.seed(554)
N <- 500 # persons
I <- 10 # items
theta <- rnorm(N,sd=1.3 ) # trait SD of 1.3
b <- seq(-2 , 2 , length=I) # item difficulties

# simulate data from the Rasch model
dat <- sim.raschtype( theta = theta , b = b )

# estimation with rasch.pml and probit link
mod1 <- rasch.pml3( dat )
summary(mod1)

# estimation with rasch.mml2 function
mod2 <- rasch.mml3( dat )

# estimate item parameters for groups with five item parameters each
est.b <- rep( 1:(I/2) , each=2 )
mod3 <- rasch.pml3( dat , est.b=est.b )
summary(mod3)

# compare parameter estimates
summary(mod1)
summary(mod2)
summary(mod3)

```

```
#####
# SIMULATED EXAMPLE 4: 11 items and 2 item clusters with 2 and 3 items
#####

set.seed(5698)
I <- 11                # number of items
n <- 5000              # number of persons
b <- seq(-2,2, len=I)  # item difficulties
theta <- rnorm( n , sd = 1 )  # person abilities
# itemcluster
itemcluster <- rep(0,I)
itemcluster[c(3,5)] <- 1
itemcluster[c(2,4,9)] <- 2
# residual correlations
rho <- c( .7 , .5 )

# simulate data (under the logit link)
dat <- sim.rasch.dep( theta , b , itemcluster , rho )
colnames(dat) <- paste("I" , seq(1,ncol(dat)) , sep="")

####
# Model 1: estimation using the Rasch model (with probit link)
mod1 <- rasch.pml3( dat )
####
# Model 2: estimation when pairs of locally dependent items are eliminated
mod2 <- rasch.pml3( dat , itemcluster=itemcluster)

####
# Model 3: Positive correlations within testlets
est.corr <- diag( 1 , I )
est.corr[ c(3,5) , c(3,5) ] <- 2
est.corr[ c(2,4,9) , c(2,4,9) ] <- 3
mod3 <- rasch.pml3( dat , error.corr=est.corr )

####
# Model 4: Negative correlations between testlets
est.corr <- diag( 1 , I )
est.corr[ c(3,5) , c(2,4,9) ] <- 2
est.corr[ c(2,4,9) , c(3,5) ] <- 2
mod4 <- rasch.pml3( dat , error.corr=est.corr )

####
# Model 5: sum constraint of zero within and between testlets
est.corr <- matrix( 1:(I*I) , I , I )
cluster2 <- c(2,4,9)
est.corr[ setdiff( 1:I , c(cluster2)) , ] <- 0
est.corr[ , setdiff( 1:I , c(cluster2)) ] <- 0
# define an error constraint matrix
itempairs0 <- mod4$itempairs
IP <- nrow(itempairs0)
err.constraint <- matrix( 0 , IP , 1 )
err.constraint[ ( itempairs0$item1 %in% cluster2 )
```

```

      & ( itempairs0$item2 %in% cluster2 ) , 1 ] <- 1
# set sum of error covariances to 1.2
err.constraintV <- matrix(3*.4,1,1)

mod5 <- rasch.pml3( dat , error.corr=est.corr ,
  err.constraintM=err.constraint, err.constraintV=err.constraintV)

#####
#****
# Model 6: Constraint on sum of all correlations
est.corr <- matrix( 1:(I*I) , I , I )
# define an error constraint matrix
itempairs0 <- mod4$itempairs
IP <- nrow(itempairs0)
# define two side conditions
err.constraint <- matrix( 0 , IP , 2 )
err.constraintV <- matrix( 0 , 2 , 1)
# sum of all correlations is zero
err.constraint[ , 1 ] <- 1
err.constraintV[1,1] <- 0
# sum of items cluster c(1,2,3) is 0
cluster2 <- c(1,2,3)
err.constraint[ ( itempairs0$item1 %in% cluster2 )
  & ( itempairs0$item2 %in% cluster2 ) , 2 ] <- 1
err.constraintV[2,1] <- 0

mod6 <- rasch.pml3( dat , error.corr=est.corr ,
  err.constraintM=err.constraint, err.constraintV=err.constraintV)
summary(mod6)

#####
# SIMULATED EXAMPLE 5: 10 Items: Cluster 1 -> Items 1,2
# Cluster 2 -> Items 3,4,5; Cluster 3 -> Items 7,8,9
#####

set.seed(7650)
I <- 10 # number of items
n <- 5000 # number of persons
b <- seq(-2,2, len=I) # item difficulties
bsamp <- b <- sample(b) # sample item difficulties
theta <- rnorm( n , sd = 1 ) # person abilities
# define itemcluster
itemcluster <- rep(0,I)
itemcluster[ 1:2 ] <- 1
itemcluster[ 3:5 ] <- 2
itemcluster[ 7:9 ] <- 3
# define residual correlations
rho <- c( .55 , .35 , .45)

# simulate data
dat <- sim.rasch.dep( theta , b , itemcluster , rho )
colnames(dat) <- paste("I" , seq(1,ncol(dat)) , sep="")

###

```

```

# Model 1: residual correlation (equal within item clusters)
# define a matrix of integers for estimating error correlations
error.corr <- diag(1,ncol(dat))
for ( ii in 1:3){
  ind.ii <- which( itemcluster == ii )
  error.corr[ ind.ii , ind.ii ] <- ii
}
# estimate the model
mod1 <- rasch.pml3( dat , error.corr = error.corr )

####
# Model 2: residual correlation (different within item clusters)
# define again a matrix of integers for estimating error correlations
error.corr <- diag(1,ncol(dat))
for ( ii in 1:3){
  ind.ii <- which( itemcluster == ii )
  error.corr[ ind.ii , ind.ii ] <- ii
}
I <- ncol(error.corr)
error.corr1 <- matrix( 1:(I*I) , ncol= I )
error.corr <- error.corr1 * ( error.corr > 0 )
# estimate the model
mod2 <- rasch.pml3( dat , error.corr = error.corr )

####
# Model 3: eliminate item pairs within itemclusters for PML estimation
mod3 <- rasch.pml3( dat , itemcluster = itemcluster )

####
# Model 4: Rasch model ignoring dependency
mod4 <- rasch.pml3( dat )

# compare different models
summary(mod1)
summary(mod2)
summary(mod3)
summary(mod4)

## End(Not run)

```

## Description

This function estimates the Rasch model using the PROX algorithm (cited in Wright & Stone, 1999).

**Usage**

```
rasch.prox(dat, dat.resp = 1 - is.na(dat), freq=rep(1,nrow(dat)) ,
           conv = 0.001, maxiter = 30, progress = FALSE)
```

**Arguments**

dat	An $N \times I$ data frame of dichotomous response data. NAs are not allowed and must be indicated by zero entries in the response indicator matrix <code>dat.resp</code> .
dat.resp	An $N \times I$ indicator data frame of nonmissing item responses.
freq	A vector of frequencies (or weights) of all rows in data frame <code>dat</code> .
conv	Convergence criterion for item parameters
maxiter	Maximum number of iterations
progress	Display progress?

**Value**

A list with following entries

b	Estimated item difficulties
theta	Estimated person abilities
iter	Number of iterations
sigma.i	Item standard deviations
sigma.n	Person standard deviations

**Author(s)**

Alexander Robitzsch

**References**

Wright, B., & Stone, W. (1999). *Measurement Essentials*. Wilmington: Wide Range.

**Examples**

```
#####
# EXAMPLE 1: PROX data.read
#####

data(data.read)
mod <- rasch.prox( data.read )
mod$b      # item difficulties
```



---

 rasch.va

*Estimation of the Rasch Model with Variational Approximation*


---

**Description**

This function estimates the Rasch model by the estimation method of variational approximation (Rijmen & Vomlel, 2008).

**Usage**

```
rasch.va(dat, globconv = 0.001, maxiter = 1000)
```

**Arguments**

dat	Data frame with dichotomous item responses
globconv	Covergence criterion for item parameters
maxiter	Maximal number of iterations

**Value**

A list with following entries:

sig	Standard deviation of the trait
item	Data frame with item parameters
xsi.ij	Data frame with variational parameters $\xi_{ij}$
mu.i	Vector with individual means $\mu_i$
sigma2.i	Vector with individual variances $\sigma_i^2$

**Author(s)**

Alexander Robitzsch

**References**

Rijmen, F., & Vomlel, J. (2008). Assessing the performance of variational methods for mixed logistic regression models. *Journal of Statistical Computation and Simulation*, **78**, 765-779.

**Examples**

```
#####
# SIMULATED EXAMPLE 1: Rasch model
#####
set.seed(8706)
N <- 5000
I <- 20
dat <- sim.raschtype( rnorm(N,sd=1.3) , b= seq(-2,2,len=I) )
```

```

# estimation via variational approximation
mod1 <- rasch.va(dat)

# estimation via marginal maximum likelihood
mod2 <- rasch.mml2(dat)

# estimation via joint maximum likelihood
mod3 <- rasch.jml(dat)

# compare sigma
round( c( mod1$sig , mod2$sd.trait ) , 3 )
## [1] 1.222 1.314

# compare b
round( cbind( mod1$item$b , mod2$item$b , mod3$item$itemdiff ) , 3 )
##      [,1] [,2] [,3]
## [1,] -1.898 -1.967 -2.090
## [2,] -1.776 -1.841 -1.954
## [3,] -1.561 -1.618 -1.715
## [4,] -1.326 -1.375 -1.455
## [5,] -1.121 -1.163 -1.228

```

---

reliability.nonlinearSEM

*Estimation of Reliability for Confirmatory Factor Analyses Based on  
Dichotomous Data*

---

## Description

This function estimates a model based reliability using confirmatory factor analysis (Green & Yang, 2009).

## Usage

```
reliability.nonlinearSEM(facloadings, thresh, resid.cov = NULL , cor.factors = NULL)
```

## Arguments

facloadings	Matrix of factor loadings
thresh	Vector of thresholds
resid.cov	Matrix of residual covariances
cor.factors	Optional matrix of covariances (correlations) between factors. The default is a diagonal matrix with variances of 1.

## Value

A list. The reliability is the list element `omega.rel`

**Note**

This function needs the **mvtnorm** package.

**Author(s)**

Alexander Robitzsch

**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.

**See Also**

This function is used in [greenyang.reliability](#).

**Examples**

```
#####
# EXAMPLE 1: Reading data set
#####
data(data.read)
dat <- data.read
I <- ncol(dat)

# define item clusters
itemcluster <- rep( 1:3 , each=4)
error.corr <- diag(1,ncol(dat))
for ( ii in 1:3){
  ind.ii <- which( itemcluster == ii )
  error.corr[ ind.ii , ind.ii ] <- ii
}
# estimate the model with error correlations
mod1 <- rasch.pml3( dat , error.corr = error.corr)
summary(mod1)

# extract item parameters
thresh <- - matrix( mod1$item$a * mod1$item$b , I , 1 )
A <- matrix( mod1$item$a * mod1$item$sigma , I , 1 )
# extract estimated correlation matrix
corM <- mod1$eps.corrM
# compute standardized factor loadings
facA <- 1 / sqrt( A^2 + 1 )
resvar <- 1 - facA^2
covM <- outer( sqrt(resvar[,1]) , sqrt(resvar[,1]) ) * corM
facloadings <- A * facA

# estimate reliability
rel1 <- reliability.nonlinearSEM( facloadings =facloadings , thresh =thresh ,
  resid.cov=covM)
rel1$omega.rel
```

rm.facets

*Rater Facets Models with Item/Rater Intercepts and Slopes***Description**

This function estimates the unidimensional rater facets model (Lincare, 1994) and an extension to slopes (see Details). The estimation is conducted by an EM algorithm employing marginal maximum likelihood.

**Usage**

```
rm.facets(dat, pid=NULL, rater=NULL, Qmatrix=NULL, theta.k=seq(-9, 9, len=30),
  est.b.rater=TRUE, est.a.item=FALSE, est.a.rater=FALSE, est.mean=FALSE ,
  tau.item.fixed=NULL , a.item.fixed=NULL , b.rater.fixed=NULL , a.rater.fixed=NULL ,
  max.b.increment=1, numdiff.parm=0.00001, maxdevchange=0.1, globconv=0.001,
  maxiter=1000, msteps=4, mstepconv=0.001)
```

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

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

```
## S3 method for class 'rm.facets'
logLik(object,...)
```

```
## S3 method for class 'rm.facets'
IRT.irfprob(object,...)
```

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

```
## S3 method for class 'rm.facets'
IRT.likelihood(object,...)
```

```
## S3 method for class 'rm.facets'
IRT.posterior(object,...)
```

```
## S3 method for class 'rm.facets'
IRT.modelfit(object,...)
```

```
## S3 method for class 'IRT.modelfit.rm.facets'
summary(object,...)
```

**Arguments**

**dat** Original data frame. Ratings on variables must be in rows, i.e. every row corresponds to a person-rater combination.

pid	Person identifier.
rater	Rater identifier
Qmatrix	An optional Q-matrix. If this matrix is not provided, then by default the ordinary scoring of categories (from 0 to the maximum score of $K$ ) is used.
theta.k	A grid of theta values for the ability distribution.
est.b.rater	Should the rater severities $b_r$ be estimated?
est.a.item	Should the item slopes $a_i$ be estimated?
est.a.rater	Should the rater slopes $a_r$ be estimated?
est.mean	Optional logical indicating whether the mean of the trait distribution should be estimated.
tau.item.fixed	Matrix with fixed $\tau$ parameters. Non-fixed parameters must be declared by NA values.
a.item.fixed	Vector with fixed item discriminations
b.rater.fixed	Vector with fixed rater intercept parameters
a.rater.fixed	Vector with fixed rater discrimination parameters
max.b.increment	Maximum increment of item parameters during estimation
numdiff.parm	Numerical differentiation step width
maxdevchange	Maximum relative deviance change as a convergence criterion
globconv	Maximum parameter change
maxiter	Maximum number of iterations
msteps	Maximum number of iterations during an M step
mstepconv	Convergence criterion in an M step
object	Object of class <code>rm.facets</code>
type	Factor score estimation method. Factor score types "EAP", "MLE" and "WLE" are supported.
...	Further arguments to be passed

### Details

This function models ratings  $X_{pri}$  for person  $p$ , rater  $r$  and item  $i$  and category  $k$

$$P(X_{pri} = k | \theta_p) \propto \exp(a_i a_r q_{ik} \theta_p - q_{ik} b_r - \tau_{ik}) \quad , \quad \theta_p \sim N(0, \sigma^2)$$

By default, the scores in the  $Q$  matrix are  $q_{ik} = k$ . Item slopes  $a_i$  and rater slopes  $a_r$  are standardized such that their product equals one, i.e.  $\prod_i a_i = \prod_r a_r = 1$ .

### Value

A list with following entries:

deviance	Deviance
ic	Information criteria and number of parameters

item	Data frame with item parameters
rater	Data frame with rater parameters
person	Data frame with person parameters: EAP and corresponding standard errors
EAP.rel	EAP reliability
mu	Mean of the trait distribution
sigma	Standard deviation of the trait distribution
theta.k	Grid of theta values
pi.k	Fitted distribution at theta.k values
tau.item	Item parameters $\tau_{ik}$
se.tau.item	Standard error of item parameters $\tau_{ik}$
a.item	Item slopes $a_i$
se.a.item	Standard error of item slopes $a_i$
delta.item	Delta item parameter. See <a href="#">pcm.conversion</a> .
b.rater	Rater severity parameter $b_r$
se.b.rater	Standard error of rater severity parameter $b_r$
a.rater	Rater slope parameter $a_r$
se.a.rater	Standard error of rater slope parameter $a_r$
f.yi.qk	Individual likelihood
f.qk.yi	Individual posterior distribution
probs	Item probabilities at grid theta.k
n.ik	Expected counts
maxK	Maximum number of categories
procdat	Processed data
iter	Number of iterations
ipars.dat2	Item parameters for expanded dataset dat2
...	Further values

### Note

If the trait standard deviation sigma strongly differs from 1, then a user should investigate the sensitivity of results using different theta integration points theta.k.

### Author(s)

Alexander Robitzsch

### References

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

**See Also**

See also the **TAM** package for the estimation of more complicated facet models.

See [rm.sdt](#) for estimating a hierarchical rater model.

**Examples**

```
#####
# EXAMPLE 1: Partial Credit Model and Generalized partial credit model
#           5 items and 1 rater
#####
data(data.ratings1)
dat <- data.ratings1

# select rater db01
dat <- dat[ paste(dat$rater) == "db01" , ]

# Model 1: Partial Credit Model
mod1 <- rm.facets( dat[ , paste0( "k",1:5) ] , pid=dat$idstud , maxiter=15)

# Model 2: Generalized Partial Credit Model
mod2 <- rm.facets( dat[ , paste0( "k",1:5) ] , pid=dat$idstud ,
                  est.a.item=TRUE , maxiter=15)

summary(mod1)
summary(mod2)

## Not run:
#####
# EXAMPLE 2: Facets Model: 5 items, 7 raters
#####

data(data.ratings1)
dat <- data.ratings1
maxit <- 15 # maximum number of iterations, increase it in applications!

# Model 1: Partial Credit Model: no rater effects
mod1 <- rm.facets( dat[ , paste0( "k",1:5) ] , rater=dat$rater ,
                  pid=dat$idstud , est.b.rater=FALSE , maxiter=maxit)

# Model 2: Partial Credit Model: intercept rater effects
mod2 <- rm.facets( dat[ , paste0( "k",1:5) ] , rater=dat$rater ,
                  pid=dat$idstud , maxiter=maxit)

# extract individual likelihood
lmod1 <- IRT.likelihood(mod1)
str(lmod1)
# likelihood value
logLik(mod1)
# extract item response functions
pmod1 <- IRT.irfprob(mod1)
str(pmod1)
# model comparison
```

```

anova(mod1,mod2)
# absolute and relative model fit
smo1 <- IRT.modelfit(mod1)
summary(smo1)
smo2 <- IRT.modelfit(mod2)
summary(smo2)
IRT.compareModels( smo1 , smo2 )
# extract factor scores (EAP is the default)
IRT.factor.scores(mod2)
# extract WLEs
IRT.factor.scores(mod2 , type="WLE")

# Model 2a: compare results with TAM package
# Results should be similar to Model 2
library(TAM)
mod2a <- TAM::tam.mml.mfr( resp= dat[ , paste0( "k",1:5) ] ,
                          facets= dat[ , "rater" , drop=FALSE] ,
                          pid= dat$pid , formulaA = ~ item*step + rater )

# Model 2b: Partial Credit Model: some fixed parameters
# fix rater parameters for raters 1, 4 and 5
b.rater.fixed <- rep(NA,7)
b.rater.fixed[ c(1,4,5) ] <- c(1,-.8,0) # fixed parameters
# fix item parameters of first and second item
tau.item.fixed <- round( mod2$tau.item , 1 ) # use parameters from mod2
tau.item.fixed[ 3:5 , ] <- NA # free item parameters of items 3, 4 and 5
mod2b <- rm.facets( dat[ , paste0( "k",1:5) ] , rater=dat$rater ,
                  b.rater.fixed=b.rater.fixed , tau.item.fixed=tau.item.fixed ,
                  est.mean = TRUE , pid=dat$idstud , maxiter=maxit)
summary(mod2b)

# Model 3: estimated rater slopes
mod3 <- rm.facets( dat[ , paste0( "k",1:5) ] , rater=dat$rater ,
                  est.a.rater=TRUE , maxiter=maxit)

# Model 4: estimated item slopes
mod4 <- rm.facets( dat[ , paste0( "k",1:5) ] , rater=dat$rater ,
                  pid=dat$idstud , est.a.item=TRUE , maxiter=maxit)

# Model 5: estimated rater and item slopes
mod5 <- rm.facets( dat[ , paste0( "k",1:5) ] , rater=dat$rater ,
                  pid=dat$idstud , est.a.rater=TRUE , est.a.item=TRUE , maxiter=maxit)
summary(mod1)
summary(mod2)
summary(mod2a)
summary(mod3)
summary(mod4)
summary(mod5)

# Model 5a: Some fixed parameters in Model 5
# fix rater b parameters for raters 1, 4 and 5
b.rater.fixed <- rep(NA,7)
b.rater.fixed[ c(1,4,5) ] <- c(1,-.8,0)

```



```

# fix rater a parameters for first four raters
a.rater.fixed <- rep(NA,7)
a.rater.fixed[ c(1,2,3,4) ] <- c(1.1,0.9,.85,1)
# fix item b parameters of first item
tau.item.fixed <- matrix( NA , nrow=5 , ncol=3 )
tau.item.fixed[ 1 , ] <- c(-2,-1.5 , 1 )
# fix item a parameters
a.item.fixed <- rep(NA,5)
a.item.fixed[ 1:4 ] <- 1
# estimate model
mod5a <- rm.facets( dat[ , paste0( "k",1:5) ] , rater=dat$rater ,
  pid=dat$idstud , est.a.rater=TRUE , est.a.item=TRUE ,
  tau.item.fixed=tau.item.fixed , b.rater.fixed=b.rater.fixed ,
  a.rater.fixed=a.rater.fixed , a.item.fixed=a.item.fixed ,
  est.mean=TRUE , maxiter=maxit)
summary(mod5a)

## End(Not run)

```

---

rm.sdt

*Hierarchical Rater Model Based on Signal Detection Theory (HRM-SDT)*


---

## Description

This function estimates a version of the hierarchical rater model (HRM) based on signal detection theory (HRM-SDT; DeCarlo, 2005; DeCarlo, Kim & Johnson, 2011).

## Usage

```

rm.sdt(dat, pid, rater, Qmatrix = NULL, theta.k = seq(-9, 9, len = 30),
  est.a.item = FALSE, est.c.rater = "n", est.d.rater = "n", est.mean=FALSE ,
  skillspace="normal" , tau.item.fixed = NULL , a.item.fixed = NULL ,
  d.min = 0.5, d.max = 100, d.start = 3, max.increment = 1, numdiff.parm = 0.00001,
  maxdevchange = 0.1, globconv = .001, maxiter = 1000, msteps = 4, mstepconv = 0.001)

## S3 method for class 'rm.sdt'
summary(object,...)

## S3 method for class 'rm.sdt'
plot(x, ask=TRUE, ...)

## S3 method for class 'rm.sdt'
anova(object,...)

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

```

```

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

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

## S3 method for class 'rm.sdt'
IRT.likelihood(object,...)

## S3 method for class 'rm.sdt'
IRT.posterior(object,...)

## S3 method for class 'rm.sdt'
IRT.modelfit(object,...)

## S3 method for class 'IRT.modelfit.rm.sdt'
summary(object,...)

```

### Arguments

dat	Original data frame. Ratings on variables must be in rows, i.e. every row corresponds to a person-rater combination.
pid	Person identifier.
rater	Rater identifier.
Qmatrix	An optional Q-matrix. If this matrix is not provided, then by default the ordinary scoring of categories (from 0 to the maximum score of $K$ ) is used.
theta.k	A grid of theta values for the ability distribution.
est.a.item	Should item parameters $a_i$ be estimated?
est.c.rater	Type of estimation for item-rater parameters $c_{ir}$ in the signal detection model. Options are 'n' (no estimation), 'e' (set all parameters equal to each other), 'i' (item wise estimation), 'r' (rater wise estimation) and 'a' (all parameters are estimated independently from each other).
est.d.rater	Type of estimation of $d$ parameters. Options are the same as in est.c.rater.
est.mean	Optional logical indicating whether the mean of the trait distribution should be estimated.
skillspace	Specified $\theta$ distribution type. It can be "normal" or "discrete". In the latter case, all probabilities of the distribution are separately estimated.
tau.item.fixed	Optional matrix with three columns specifying fixed $\tau$ parameters. The first two columns denote item and category indices, the third the fixed value. See Example 3.
a.item.fixed	Optional matrix with two columns specifying fixed $a$ parameters. First column: Item index. Second column: Fixed $a$ parameter.
d.min	Minimal $d$ parameter to be estimated
d.max	Maximal $d$ parameter to be estimated
d.start	Starting value of $d$ parameters

max.increment	Maximum increment of item parameters during estimation
numdiff.parm	Numerical differentiation step width
maxdevchange	Maximum relative deviance change as a convergence criterion
globconv	Maximum parameter change
maxiter	Maximum number of iterations
msteps	Maximum number of iterations during an M step
mstepconv	Convergence criterion in an M step
object	Object of class rm.sdt
x	Object of class rm.sdt
ask	Optional logical indicating whether a new plot should be asked for.
type	Factor score estimation method. Up to now, only type="EAP" is supported.
...	Further arguments to be passed

**Details**

The specification of the model follows DeCarlo et al. (2011). The second level models the ideal rating (latent response)  $\eta = 0, \dots, K$  of person  $p$  on item  $i$

$$P(\eta_{pi} = \eta | \theta_p) \propto \exp(a_i q_{ik} \theta_p - \tau_{ik})$$

At the first level, the ratings  $X_{pir}$  for person  $p$  on item  $i$  and rater  $r$  are modelled as a signal detection model

$$P(X_{pir} \leq k | \eta_{pi}) = G(c_{irk} - d_{ir} \eta_{pi})$$

where  $G$  is the logistic distribution function and the categories are  $k = 1, \dots, K + 1$ . Note that the item response model can be equivalently written as

$$P(X_{pir} \geq k | \eta_{pi}) = G(d_{ir} \eta_{pi} - c_{irk})$$

The thresholds  $c_{irk}$  can be further restricted to  $c_{irk} = c_k$  (est.c.rater='e'),  $c_{irk} = c_{ik}$  (est.c.rater='i') or  $c_{irk} = c_{ir}$  (est.c.rater='r'). The same holds for rater precision parameters  $d_{ir}$ .

**Value**

A list with following entries:

deviance	Deviance
ic	Information criteria and number of parameters
item	Data frame with item parameters. The columns N and M denote the number of observed ratings and the observed mean of all ratings, respectively. In addition to item parameters $\tau_{ik}$ and $a_i$ , the mean for the latent response (latM) is computed as $E(\eta_i) = \sum_p P(\theta_p) q_{ik} P(\eta_i = k   \theta_p)$ which provides an item parameter at the original metric of ratings. The latent standard deviation (latSD) is computed in the same manner.
rater	Data frame with rater parameters. Transformed $c$ parameters (c_x.trans) are computed as $c_{irk} / (d_{ir})$ .

person	Data frame with person parameters: EAP and corresponding standard errors
EAP.rel	EAP reliability
EAP.rel	EAP reliability
mu	Mean of the trait distribution
sigma	Standard deviation of the trait distribution
tau.item	Item parameters $\tau_{ik}$
se.tau.item	Standard error of item parameters $\tau_{ik}$
a.item	Item slopes $a_i$
se.a.item	Standard error of item slopes $a_i$
c.rater	Rater parameters $c_{irk}$
se.c.rater	Standard error of rater severity parameter $c_{irk}$
d.rater	Rater slope parameter $d_{ir}$
se.d.rater	Standard error of rater slope parameter $d_{ir}$
f.yi.qk	Individual likelihood
f.qk.yi	Individual posterior distribution
probs	Item probabilities at grid theta.k. Note that these probabilities are calculated on the pseudo items $i \times r$ , i.e. the interaction of item and rater.
prob.item	Probabilities $P(\eta_i = \eta \theta)$ of latent item responses evaluated at theta grid $\theta_p$ .
n.ik	Expected counts
pi.k	Estimated trait distribution $P(\theta_p)$ .
maxK	Maximum number of categories
procdata	Processed data
iter	Number of iterations
...	Further values

### Author(s)

Alexander Robitzsch

### References

- DeCarlo, L. T. (2005). A model of rater behavior in essay grading based on signal detection theory. *Journal of Educational Measurement*, **42**, 53-76.
- DeCarlo, L. T. (2010). *Studies of a latent-class signal-detection model for constructed response scoring II: Incomplete and hierarchical designs*. ETS Research Report ETS RR-10-08. Princeton NJ: ETS.
- DeCarlo, T., Kim, Y., & Johnson, M. S. (2011). A hierarchical rater model for constructed responses, with a signal detection rater model. *Journal of Educational Measurement*, **48**, 333-356.

### See Also

The facets rater model can be estimated with [rm.facets](#).

## Examples

```
#####
# EXAMPLE 1: Hierarchical rater model (HRM-SDT) data.ratings1
#####
data(data.ratings1)
dat <- data.ratings1

## Not run:
# Model 1: Partial Credit Model: no rater effects
mod1 <- rm.sdt( dat[ , paste0( "k",1:5) ] , rater=dat$rater ,
               pid=dat$idstud , est.c.rater="n" , est.d.rater="n" , maxiter=15)
summary(mod1)

# Model 2: Generalized Partial Credit Model: no rater effects
mod2 <- rm.sdt( dat[ , paste0( "k",1:5) ] , rater=dat$rater ,
               pid=dat$idstud , est.c.rater="n" , est.d.rater="n" ,
               est.a.item =TRUE , d.start=100 , maxiter=15)
summary(mod2)

## End(Not run)

# Model 3: Equal effects in SDT
mod3 <- rm.sdt( dat[ , paste0( "k",1:5) ] , rater=dat$rater ,
               pid=dat$idstud , est.c.rater="e" , est.d.rater="e" , maxiter=15)
summary(mod3)

## Not run:
# Model 4: Rater effects in SDT
mod4 <- rm.sdt( dat[ , paste0( "k",1:5) ] , rater=dat$rater ,
               pid=dat$idstud , est.c.rater="r" , est.d.rater="r" , maxiter=15)
summary(mod4)

#####
# EXAMPLE 2: HRM-SDT data.ratings3
#####

data(data.ratings3)
dat <- data.ratings3
dat <- dat[ dat$rater < 814 , ]
psych::describe(dat)

# Model 1: item- and rater-specific effects
mod1 <- rm.sdt( dat[ , paste0( "crit",c(2:4)) ] , rater=dat$rater ,
               pid=dat$idstud , est.c.rater="a" , est.d.rater="a" , maxiter=10)
summary(mod1)
plot(mod1)

# Model 2: Differing number of categories per variable
mod2 <- rm.sdt( dat[ , paste0( "crit",c(2:4,6)) ] , rater=dat$rater ,
               pid=dat$idstud , est.c.rater="a" , est.d.rater="a" , maxiter=10)
summary(mod2)
plot(mod2)
```

```
#####
# EXAMPLE 3: Hierarchical rater model with discrete skill spaces
#####

data(data.ratings3)
dat <- data.ratings3
dat <- dat[ dat$rater < 814 , ]
psych::describe(dat)

# Model 1: Discrete theta skill space with values of 0,1,2 and 3
mod1 <- rm.sdt( dat[ , paste0( "crit",c(2:4)) ] , theta.k = 0:3 , rater=dat$rater ,
               pid=dat$idstud , est.c.rater="a" , est.d.rater="a" , skillspace="discrete" ,
               maxiter=20)
summary(mod1)
plot(mod1)

# Model 2: Modelling of one item by using a discrete skill space and
#         fixed item parameters

# fixed tau and a parameters
tau.item.fixed <- cbind( 1, 1:3, 100*cumsum( c( 0.5, 1.5, 2.5)) )
a.item.fixed <- cbind( 1, 100 )
# fit HRM-SDT
mod2 <- rm.sdt( dat[ , "crit2" , drop=FALSE] , theta.k = 0:3 , rater=dat$rater ,
               tau.item.fixed=tau.item.fixed ,a.item.fixed=a.item.fixed, pid=dat$idstud,
               est.c.rater="a", est.d.rater="a", skillspace="discrete", maxiter=20)
summary(mod2)
plot(mod2)

## End(Not run)
```

---

sia.sirt

*Statistical Implicative Analysis (SIA)*


---

## Description

This function is a simplified implementation of statistical implicative analysis (Gras & Kuntz, 2008) which aims at deriving implications  $X_i \rightarrow X_j$ . This means that solving item  $i$  implies solving item  $j$ .

## Usage

```
sia.sirt(dat, significance = 0.85)
```

## Arguments

dat	Data frame with dichotomous item responses
significance	Minimum implicative probability for inclusion of an arrow in the graph. The probability can be interpreted as a kind of significance level, i.e. higher probabilities indicate more probable implications.

## Details

The test statistic for selection an implicative relation follows Gras and Kuntz (2008). Transitive arrows (implications) are removed from the graph. If some implications are symmetric, then only the more probable implication will be retained.

## Value

A list with following entries

<code>adj.matrix</code>	Adjacency matrix of the graph. Transitive and symmetric implications (arrows) have been removed.
<code>adj.pot</code>	Adjacency matrix including all potencies, i.e. all direct and indirect paths from item $i$ to item $j$ .
<code>adj.matrix.trans</code>	Adjacency matrix including transitive arrows.
<code>desc</code>	List with descriptive statistics of the graph.
<code>desc.item</code>	Descriptive statistics for each item.
<code>impl.int</code>	Implication intensity (probability) as the basis for deciding the significance of an arrow
<code>impl.t</code>	Corresponding $t$ values of <code>impl.int</code>
<code>impl.significance</code>	Corresponding $p$ values (significancies) of <code>impl.int</code>
<code>conf.loev</code>	Confidence according to Loevinger (see Gras & Kuntz, 2008). This values are just conditional probabilities $P(X_j = 1 X_i = 1)$ .
<code>graph.matr</code>	Matrix containing all arrows. Can be used for example for the <b>Rgraphviz</b> package.
<code>graph.edges</code>	Vector containing all edges of the graph, e.g. for the <b>Rgraphviz</b> package.
<code>igraph.matr</code>	Matrix containing all arrows for the <b>igraph</b> package.
<code>igraph.obj</code>	An object of the graph for the <b>igraph</b> package.

## Note

For an implementation of statistical implicative analysis in the C.H.I.C. (Classification Hierarchique, Implicative et Cohesitive) software.

See <http://www.aradm.eu/contenu/logiciel-d-analyse-de-donnees-chic>.

## Author(s)

Alexander Robitzsch

## References

Gras, R., & Kuntz, P. (2008). An overview of the statistical implicative analysis (SIA) development. In R. Gras, E. Suzuki, F. Guillet, & F. Spagnolo (Eds.). *Statistical Implicative Analysis* (pp. 11-40). Springer, Berlin Heidelberg.

## See Also

See also the **IsingFit** package for calculating a graph for dichotomous item responses using the Ising model.

## Examples

```
#####
# EXAMPLE 1: SIA for data.read
#####

data(data.read)
dat <- data.read

res <- sia.sirt(dat , significance=.85 )

###* plot results with igraph package
library(igraph)
plot( res$igraph.obj ) # , vertex.shape="rectangle" , vertex.size=30 )

###* plot results with qgraph package
library(qgraph)
qgraph::qgraph( res$adj.matrix )

## Not run:
###* plot results with Rgraphviz package
# Rgraphviz can only be obtained from Bioconductor
# If it should be downloaded, select TRUE for the following lines
if (FALSE){
  source("http://bioconductor.org/biocLite.R")
  biocLite("Rgraphviz")
}
# define graph
grmatrix <- res$graph.matr
res.graph <- new("graphNEL", nodes= res$graph.edges , edgemode="directed")
# add edges
RR <- nrow(grmatrix)
for (rr in 1:RR){
  res.graph <- Rgraphviz::addEdge(grmatrix[rr,1], grmatrix[rr,2], res.graph , 1)
}
# define cex sizes and shapes
V <- length(res$graph.edges)
size2 <- rep(16,V)
shape2 <- rep("rectangle" , V )
names(shape2) <- names(size2) <- res$graph.edges
# plot graph
Rgraphviz::plot( res.graph, nodeAttrs =list("fontsize" = size2 , "shape" = shape2) )

## End(Not run)
```



---

`sim.qm.ramsay`*Simulate from Ramsay's Quotient Model*

---

**Description**

This function simulates dichotomous item response data according to Ramsay's quotient model (Ramsay, 1989).

**Usage**

```
sim.qm.ramsay(theta, b, K)
```

**Arguments**

theta	Vector of of length $N$ person parameters (must be positive!)
b	Vector of length $I$ of item difficulties (must be positive)
K	Vector of length $I$ of guessing parameters (must be positive)

**Details**

Ramsay's quotient model (Ramsay, 1989) is defined by the equation

$$P(X_{pi} = 1|\theta_p) = \frac{\exp(\theta_p/b_i)}{K_i + \exp(\theta_p/b_i)}$$

**Value**

An  $N \times I$  data frame with dichotomous item responses.

**Author(s)**

Alexander Robitzsch

**References**

Ramsay, J. O. (1989). A comparison of three simple test theory models. *Psychometrika*, **54**, 487-499.

van der Maas, H. J. L., Molenaar, D., Maris, G., Kievit, R. A., & Borsboom, D. (2011). Cognitive psychology meets psychometric theory: On the relation between process models for decision making and latent variable models for individual differences. *Psychological Review*, **318**, 339-356.

**See Also**

See [rasch.mml2](#) for estimating Ramsay's quotient model.

See [sim.raschtype](#) for simulating response data from the generalized logistic item response model.

**Examples**

```
#####
# SIMULATED EXAMPLE 1: Estimate Ramsay Quotient Model with rasch.mml2
#####

set.seed(657)
# simulate data according to the Ramsay model
N <- 1000      # persons
I <- 11        # items
theta <- exp( rnorm( N ) ) # person ability
b <- exp( seq(-2,2,len=I) ) # item difficulty
K <- rep( 3 , I )      # K parameter (=> guessing)

# apply simulation function
dat <- sim.qm.ramsay( theta , b , K )

###
# analysis
mmliter <- 50      # maximum number of iterations
I <- ncol(dat)
fixed.K <- rep( 3 , I )

# Ramsay QM with fixed K parameter (K=3 in fixed.K specification)
mod1 <- rasch.mml2( dat , mmliter = mmliter , irtmodel = "ramsay.qm" ,
                  fixed.K = fixed.K )
summary(mod1)

# Ramsay QM with joint estimated K parameters
mod2 <- rasch.mml2( dat , mmliter = mmliter , irtmodel = "ramsay.qm" ,
                  est.K = rep(1,I) )
summary(mod2)

## Not run:
# Ramsay QM with itemwise estimated K parameters
mod3 <- rasch.mml2( dat , mmliter = mmliter , irtmodel = "ramsay.qm" ,
                  est.K = 1:I )
summary(mod3)

# Rasch model
mod4 <- rasch.mml2( dat )
summary(mod4)

# generalized logistic model
mod5 <- rasch.mml2( dat , est.alpha = TRUE , mmliter=mmliter)
summary(mod5)

# 2PL model
mod6 <- rasch.mml2( dat , est.a = rep(1,I) )
summary(mod6)

# Difficulty + Guessing (b+c) Model
mod7 <- rasch.mml2( dat , est.c = rep(1,I) )
```

```

summary(mod7)

# estimate separate guessing (c) parameters
mod8 <- rasch.mml2( dat , est.c = 1:I )
summary(mod8)

*** estimate Model 1 with user defined function in mirt package

# create user defined function for Ramsay's quotient model
name <- 'ramsayqm'
par <- c("K" = 3 , "b" = 1 )
est <- c(TRUE, TRUE)
P.ramsay <- function(par,Theta){
  eps <- .01
  K <- par[1]
  b <- par[2]
  num <- exp( exp( Theta[,1] ) / b )
  denom <- K + num
  P1 <- num / denom
  P1 <- eps + ( 1 - 2*eps ) * P1
  cbind(1-P1, P1)
}

# create item response function
ramsayqm <- mirt::createItem(name, par=par, est=est, P=P.ramsay)
# define parameters to be estimated
mod1m.pars <- mirt::mirt(dat, 1, rep( "ramsayqm",I) ,
  customItems=list("ramsayqm"=ramsayqm), pars = "values")
mod1m.pars[ mod1m.pars$name == "K" , "est" ] <- FALSE
# define Theta design matrix
Theta <- matrix( seq(-3,3,len=10) , ncol=1)
# estimate model
mod1m <- mirt::mirt(dat, 1, rep( "ramsayqm",I) , customItems=list("ramsayqm"=ramsayqm),
  pars = mod1m.pars , verbose=TRUE ,
  technical = list( customTheta=Theta , NCYCLES=50)
)
print(mod1m)
summary(mod1m)
cmod1m <- mirt.wrapper.coef( mod1m )$coef
# compare simulated and estimated values
dfr <- cbind( b , cmod1m$b , exp(mod1$item$b ) )
colnames(dfr) <- c("simulated" , "mirt" , "sirt_rasch.mml2")
round( dfr , 2 )
##      simulated mirt sirt_rasch.mml2
## [1,]      0.14 0.11      0.11
## [2,]      0.20 0.17      0.18
## [3,]      0.30 0.27      0.29
## [4,]      0.45 0.42      0.43
## [5,]      0.67 0.65      0.67
## [6,]      1.00 1.00      1.01
## [7,]      1.49 1.53      1.54
## [8,]      2.23 2.21      2.21
## [9,]      3.32 3.00      2.98

```

```
##[10,]      4.95 5.22          5.09
##[11,]      7.39 5.62          5.51

## End(Not run)
```

---

sim.rasch.dep

*Simulation of the Rasch Model with Locally Dependent Responses*


---

### Description

This function simulates dichotomous item responses where for some itemclusters residual correlations can be defined.

### Usage

```
sim.rasch.dep(theta, b, itemcluster, rho)
```

### Arguments

theta	Vector of person abilities of length $N$
b	Vector of item difficulties of length $I$
itemcluster	Vector of integers (including 0) of length $I$ . Different integers correspond to different itemclusters.
rho	Vector of residual correlations. The length of vector must be equal to the number of itemclusters.

### Value

An  $N \times I$  data frame of dichotomous item responses.

### Note

The specification of the simulation models follows a marginal interpretation of the latent trait. Local dependencies are only interpreted as nuisance and not of substantive interest. If local dependencies should be substantively interpreted, a testlet model seems preferable (see [mcmc.3pno.testlet](#)).

### Author(s)

Alexander Robitzsch

### See Also

To simulate the generalized logistic item response model see [sim.raschtype](#). Ramsay's quotient model can be simulated using [sim.qm.ramsay](#).

Marginal item response models for locally dependent item responses can be estimated with [rasch.copula2](#), [rasch.pairwise](#) or [rasch.pairwise.itemcluster](#).

## Examples

```
#####
# SIMULATED EXAMPLE 1: 11 Items: 2 itemclusters with 2 resp. 3 dependent items
#           and 6 independent items
#####

set.seed(7654)
I <- 11                # number of items
n <- 1500              # number of persons
b <- seq(-2,2, len=I)  # item difficulties
theta <- rnorm( n , sd = 1 )  # person abilities
# itemcluster
itemcluster <- rep(0,I)
itemcluster[ c(3,5) ] <- 1
itemcluster[ c(2,4,9) ] <- 2
# residual correlations
rho <- c( .7 , .5 )

# simulate data
dat <- sim.rasch.dep( theta , b , itemcluster , rho )
colnames(dat) <- paste("I" , seq(1,ncol(dat)) , sep="")

# estimate Rasch copula model
mod1 <- rasch.copula2( dat , itemcluster = itemcluster )
summary(mod1)

# compare result with Rasch model estimation in rasch.copula
# delta must be set to zero
mod2 <- rasch.copula2( dat , itemcluster = itemcluster , delta = c(0,0) ,
                      est.delta = c(0,0) )
summary(mod2)

# estimate Rasch model with rasch.mml2 function
mod3 <- rasch.mml2( dat )
summary(mod3)

## Not run:
#####
# SIMULATED EXAMPLE 2: 12 Items: Cluster 1 -> Items 1,...,4;
#           Cluster 2 -> Items 6,...,9; Cluster 3 -> Items 10,11,12
#####

set.seed(7896)
I <- 12                # number of items
n <- 450               # number of persons
b <- seq(-2,2, len=I)  # item difficulties
b <- sample(b)         # sample item difficulties
theta <- rnorm( n , sd = 1 )  # person abilities
# itemcluster
itemcluster <- rep(0,I)
itemcluster[ 1:4 ] <- 1
itemcluster[ 6:9 ] <- 2
```

```

itemcluster[ 10:12 ] <- 3
# residual correlations
rho <- c( .55 , .25 , .45 )

# simulate data
dat <- sim.rasch.dep( theta , b , itemcluster , rho )
colnames(dat) <- paste("I" , seq(1,ncol(dat)) , sep="")

# estimate Rasch copula model
mod1 <- rasch.copula2( dat , itemcluster = itemcluster , numdiff.parm = .001 )
summary(mod1)

# Rasch model estimation
mod2 <- rasch.copula2( dat , itemcluster = itemcluster ,
                      delta = rep(0,3) , est.delta = rep(0,3) )
summary(mod2)

# estimation with pairwise Rasch model
mod3 <- rasch.pairwise( dat )
summary(mod3)

## End(Not run)

```

---

sim.raschtype

*Simulate from Generalized Logistic Item Response Model*


---

## Description

This function simulates dichotomous item responses from a generalized logistic item response model (Stukel, 1988). The four-parameter logistic item response model (Loken & Rulison, 2010) is a special case. See [rasch.mm12](#) for more details.

## Usage

```

sim.raschtype(theta, b, alpha1 = 0, alpha2 = 0, fixed.a = NULL,
              fixed.c = NULL, fixed.d = NULL)

```

## Arguments

theta	Unidimensional ability vector $\theta$
b	Vector of item difficulties $b$
alpha1	Parameter $\alpha_1$ in generalized logistic link function
alpha2	Parameter $\alpha_2$ in generalized logistic link function
fixed.a	Vector of item slopes $a$
fixed.c	Vector of lower item asymptotes $c$
fixed.d	Vector of lower item asymptotes $d$

## Details

The class of generalized logistic link functions contain the most important link functions using the specifications (Stukel, 1988):

logistic link function:  $\alpha_1 = 0$  and  $\alpha_2 = 0$

probit link function:  $\alpha_1 = 0.165$  and  $\alpha_2 = 0.165$

loglog link function:  $\alpha_1 = -0.037$  and  $\alpha_2 = 0.62$

cloglog link function:  $\alpha_1 = 0.62$  and  $\alpha_2 = -0.037$

See [pgenlogis](#) for exact transformation formulas of the mentioned link functions.

## Author(s)

Alexander Robitzsch

## References

Loken, E., & Rulison, K. L. (2010). Estimation of a four-parameter item response theory model. *British Journal of Mathematical and Statistical Psychology*, **63**, 509-525.

Stukel, T. A. (1988). Generalized logistic models. *Journal of the American Statistical Association*, **83**, 426-431.

## See Also

[rasch.mml2](#), [pgenlogis](#)

## Examples

```
# Simulation of data from a Rasch model (alpha_1 = alpha_2 = 0)
N <- 500 # number of persons
I <- 11 # number of items
b <- seq(-2, 2, length=I)
dat <- sim.raschtype( rnorm(N), b )
colnames(dat) <- paste("I", 1:I, sep="")
```

---

smirt

*Multidimensional Noncompensatory, Compensatory and Partially Compensatory Item Response Model*

---

## Description

This function estimates the noncompensatory and compensatory multidimensional item response model (Bolt & Lall, 2003; Reckase, 2009) as well as the partially compensatory item response model (Spray et al., 1990) for dichotomous data.

**Usage**

```

smirt(dat, Qmatrix, irtmodel="noncomp" , est.b = NULL, est.a = NULL,
      est.c = NULL, est.d = NULL, est.mu.i=NULL , b.init = NULL, a.init = NULL,
      c.init = NULL, d.init=NULL, mu.i.init=NULL , Sigma.init=NULL ,
      theta.k=seq(-6,6,len=20), theta.kDES=NULL,
      qmcnodes=0 , mu.fixed = NULL, variance.fixed = NULL, est.corr = FALSE,
      max.increment = 1, increment.factor=1, numdiff.parm = 0.0001,
      maxdevchange = 0.1, globconv = 0.001, maxiter = 1000, msteps = 4,
      mstepconv = 0.001)

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

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

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

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

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

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

## S3 method for class 'smirt'
IRT.modelfit(object,...)

## S3 method for class 'IRT.modelfit.smirt'
summary(object,...)

```

**Arguments**

<code>dat</code>	Data frame with dichotomous item responses
<code>Qmatrix</code>	The Q-matrix which specifies the loadings to be estimated
<code>irtmodel</code>	The item response model. Options are the noncompensatory model ("noncomp"), the compensatory model ("comp") and the partially compensatory model ("partcomp"). See Details for more explanations.
<code>est.b</code>	An integer matrix (if <code>irtmodel="noncomp"</code> ) or integer vector (if <code>irtmodel="comp"</code> ) for <i>b</i> parameters to be estimated
<code>est.a</code>	An integer matrix for <i>a</i> parameters to be estimated. If <code>est.a="2PL"</code> , then all item loadings will be estimated and the variances are set to one (and therefore <code>est.corr=TRUE</code> ).
<code>est.c</code>	An integer vector for <i>c</i> parameters to be estimated



<code>est.d</code>	An integer vector for $d$ parameters to be estimated
<code>est.mu.i</code>	An integer vector for $\mu_i$ parameters to be estimated
<code>b.init</code>	Initial $b$ coefficients. For <code>irtmodel="noncomp"</code> it must be a matrix, for <code>irtmodel="comp"</code> it is a vector.
<code>a.init</code>	Initial $a$ coefficients arranged in a matrix
<code>c.init</code>	Initial $c$ coefficients
<code>d.init</code>	Initial $d$ coefficients
<code>mu.i.init</code>	Initial $d$ coefficients
<code>Sigma.init</code>	Initial covariance matrix $\Sigma$
<code>theta.k</code>	Vector of discretized trait distribution. This vector is expanded in all dimensions by using the <code>expand.grid</code> function. If a user specifies a design matrix <code>theta.kDES</code> of transformed $\theta_p$ values (see Details and Examples), then <code>theta.k</code> must be a matrix, too.
<code>theta.kDES</code>	An optional design matrix. This matrix will differ from the ordinary theta grid in case of nonlinear item response models.
<code>qmcnodes</code>	Number of integration nodes for quasi Monte Carlo integration (see Pan & Thompson, 2007). Integration points are obtained by using the function <code>qmc.nodes</code> . Note that when using quasi Monte Carlo nodes, no theta design matrix <code>theta.kDES</code> can be specified. See Simulated Example 1, Model 11.
<code>mu.fixed</code>	Matrix with fixed entries in the mean vector. By default, all means are set to zero.
<code>variance.fixed</code>	Matrix (with rows and three columns) with fixed entries in the covariance matrix (see Examples). The entry $c_{kd}$ of the covariance between dimensions $k$ and $d$ is set to $c_0$ iff <code>variance.fixed</code> has a row with a $k$ in the first column, a $d$ in the second column and the value $c_0$ in the third column.
<code>est.corr</code>	Should only a correlation matrix instead of a covariance matrix be estimated?
<code>max.increment</code>	Maximum increment
<code>increment.factor</code>	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 <code>iter</code> is defined as <code>max.increment*increment.factor^(-iter)</code> where <code>max.increment=1</code> . Using a value larger than 1 helps to reach convergence in some non-converging analyses (use values of 1.01, 1.02 or even 1.05). See also Simulated Example 1 Model 2a.
<code>numdiff.parm</code>	Numerical differentiation parameter
<code>maxdevchange</code>	Convergence criterion for change in relative deviance
<code>globconv</code>	Global convergence criterion for parameter change
<code>maxiter</code>	Maximum number of iterations
<code>msteps</code>	Number of iterations within a M step
<code>mstepconv</code>	Convergence criterion within a M step
<code>object</code>	Object of class <code>smirt</code>
<code>...</code>	Further arguments to be passed

## Details

The noncompensatory item response model (`irtmodel="noncomp"`; e.g. Bolt & Lall, 2003) is defined as

$$P(X_{pi} = 1|\boldsymbol{\theta}_p) = c_i + (d_i - c_i) \prod_l \text{invlogit}(a_{il}q_{il}\theta_{pl} - b_{il})$$

where  $i, p, l$  denote items, persons and dimensions respectively.

The compensatory item response model (`irtmodel="comp"`) is defined by

$$P(X_{pi} = 1|\boldsymbol{\theta}_p) = c_i + (d_i - c_i) \text{invlogit}\left(\sum_l a_{il}q_{il}\theta_{pl} - b_{il}\right)$$

Using a design matrix `theta.kDES` the model can be made even more general in a model which is linear in item parameters

$$P(X_{pi} = 1|\boldsymbol{\theta}_p) = c_i + (d_i - c_i) \text{invlogit}\left(\sum_l a_{il}q_{il}t_l(\boldsymbol{\theta}_p) - b_{il}\right)$$

with known functions  $t_l$  of the trait vector  $\boldsymbol{\theta}_p$ . Fixed values of the functions  $t_l$  are specified in the  $\boldsymbol{\theta}_p$  design matrix `theta.kDES`.

The partially compensatory item response model (`irtmodel="partcomp"`) is defined by

$$P(X_{pi} = 1|\boldsymbol{\theta}_p) = c_i + (d_i - c_i) \frac{\exp\left(\sum_l (a_{il}q_{il}\theta_{pl} - b_{il})\right)}{\mu_i \prod_l (1 + \exp(a_{il}q_{il}\theta_{pl} - b_{il})) + (1 - \mu_i)(1 + \exp(\sum_l (a_{il}q_{il}\theta_{pl} - b_{il})))}$$

with item parameters  $\mu_i$  indicating the degree of compensatory.  $\mu_i = 1$  indicates a noncompensatory model while  $\mu_i = 0$  indicates a (fully) compensatory model.

The models are estimated by an EM algorithm employing marginal maximum likelihood.

## Value

A list with following entries:

<code>deviance</code>	Deviance
<code>ic</code>	Information criteria
<code>item</code>	Data frame with item parameters
<code>person</code>	Data frame with person parameters. It includes the person mean of all item responses ( <code>M</code> ; percentage correct of all non-missing items), the EAP estimate and its corresponding standard error for all dimensions (EAP and <code>SE.EAP</code> ) and the maximum likelihood estimate as well as the mode of the posterior distribution (MLE and MAP).
<code>EAP.rel</code>	EAP reliability
<code>mean.trait</code>	Means of trait
<code>sd.trait</code>	Standard deviations of trait
<code>Sigma</code>	Trait covariance matrix
<code>cor.trait</code>	Trait correlation matrix
<code>b</code>	Matrix (vector) of $b$ parameters

se.b	Matrix (vector) of standard errors $b$ parameters
a	Matrix of $a$ parameters
se.a	Matrix of standard errors of $a$ parameters
c	Vector of $c$ parameters
se.c	Vector of standard errors of $c$ parameters
d	Vector of $d$ parameters
se.d	Vector of standard errors of $d$ parameters
mu.i	Vector of $\mu_i$ parameters
se.mu.i	Vector of standard errors of $\mu_i$ parameters
f.yi.qk	Individual likelihood
f.qk.yi	Individual posterior
probs	Probabilities of item response functions evaluated at theta.k
n.ik	Expected counts
iter	Number of iterations
dat2	Processed data set
dat2.resp	Data set of response indicators
I	Number of items
D	Number of dimensions
K	Maximum item response score
theta.k	Used theta integration grid
pi.k	Distribution function evaluated at theta.k
irtmodel	Used IRT model
Qmatrix	Used Q-matrix

**Author(s)**

Alexander Robitzsch

**References**

- Bolt, D. M., & Lall, V. F. (2003). Estimation of compensatory and noncompensatory multidimensional item response models using Markov chain Monte Carlo. *Applied Psychological Measurement*, *27*, 395-414.
- Pan, J., & Thompson, R. (2007). Quasi-Monte Carlo estimation in generalized linear mixed models. *Computational Statistics & Data Analysis*, *51*, 5765-5775.
- Reckase, M.D. (2009). *Multidimensional Item Response Theory*. New York: Springer.
- Spray, J. A., Davey, T. C., Reckase, M. D., Ackerman, T. A., & Carlson, J. E. (1990). *Comparison of two logistic multidimensional item response theory models*. ACT Research Report No. ACT-RR-ONR-90-8.

**See Also**

Other multidimensional IRT models can also be estimated with [rasch.mml2](#) and [rasch.mirtlc](#).

See [itemfit.sx2](#) (CDM) for item fit statistics.

See also the **mirt** and **TAM** packages for estimation of multidimensional models.

**Examples**

```
#####
## SIMULATED EXAMPLE 1: Noncompensatory and compensatory IRT models
#####
set.seed(997)

# (1) simulate data from a two-dimensional noncompensatory
#     item response model
#     -> increase number of iterations in all models!

N <- 1000 # number of persons
I <- 10 # number of items
theta0 <- rnorm( N , sd= 1 )
theta1 <- theta0 + rnorm(N , sd = .7 )
theta2 <- theta0 + rnorm(N , sd = .7 )
Q <- matrix( 1 , nrow=I,ncol=2 )
Q[ 1:(I/2) , 2 ] <- 0
Q[ I,1] <- 0
b <- matrix( rnorm( I*2 ) , I , 2 )
a <- matrix( 1 , I , 2 )

# simulate data
prob <- dat <- matrix(0 , nrow=N , ncol=I )
for (ii in 1:I){
  prob[,ii] <- ( plogis( theta1 - b[ii,1] ) )^Q[ii,1]
  prob[,ii] <- prob[,ii] * ( plogis( theta2 - b[ii,2] ) )^Q[ii,2]
}
dat[ prob > matrix(runif( N*I),N,I) ] <- 1
colnames(dat) <- paste0("I",1:I)

####
# Model 1: Noncompensatory 1PL model
mod1 <- smirt(dat, Qmatrix=Q , maxiter=10 ) # change number of iterations
summary(mod1)

## Not run:
####
# Model 2: Noncompensatory 2PL model
mod2 <- smirt(dat,Qmatrix=Q , est.a="2PL" , maxiter=15 )
summary(mod2)

# Model 2a: avoid convergence problems with increment.factor
mod2a <- smirt(dat,Qmatrix=Q , est.a="2PL" , maxiter=30 , increment.factor=1.03)
summary(mod2a)
```

```

****
# Model 3: some fixed c and d parameters different from zero or one
c.init <- rep(0,I)
c.init[ c(3,7)] <- .2
d.init <- rep(1,I)
d.init[c(4,8)] <- .95
mod3 <- smirt( dat , Qmatrix=Q , c.init=c.init , d.init=d.init )
summary(mod3)

****
# Model 4: some estimated c and d parameters (in parameter groups)
est.c <- c.init <- rep(0,I)
c.estpars <- c(3,6,7)
c.init[ c.estpars ] <- .2
est.c[c.estpars] <- 1
est.d <- rep(0,I)
d.init <- rep(1,I)
d.estpars <- c(6,9)
d.init[ d.estpars ] <- .95
est.d[ d.estpars ] <- d.estpars # different d parameters
mod4 <- smirt(dat,Qmatrix=Q , est.c=est.c , c.init=c.init ,
              est.d=est.d , d.init=d.init )
summary(mod4)

****
# Model 5: Unidimensional 1PL model
Qmatrix <- matrix( 1 , nrow=I , ncol=1 )
mod5 <- smirt( dat , Qmatrix=Qmatrix )
summary(mod5)

****
# Model 6: Unidimensional 2PL model
mod6 <- smirt( dat , Qmatrix=Qmatrix , est.a="2PL" )
summary(mod6)

****
# Model 7: Compensatory model with between item dimensionality
# Note that the data is simulated under the noncompensatory condition
# Therefore Model 7 should have a worse model fit than Model 1
Q1 <- Q
Q1[ 6:10 , 1] <- 0
mod7 <- smirt(dat,Qmatrix=Q1 , irtmodel="comp" , maxiter=30)
summary(mod7)

****
# Model 8: Compensatory model with within item dimensionality
# assuming zero correlation between dimensions
variance.fixed <- as.matrix( cbind( 1,2,0) )
# set the covariance between the first and second dimension to zero
mod8 <- smirt(dat,Qmatrix=Q , irtmodel="comp" , variance.fixed=variance.fixed ,
              maxiter=30)
summary(mod8)

```

```

****
# Model 8b: 2PL model with starting values for a and b parameters
b.init <- rep(0,10) # set all item difficulties initially to zero
# b.init <- NULL
a.init <- Q        # initialize a.init with Q-matrix
# provide starting values for slopes of first three items on Dimension 1
a.init[1:3,1] <- c( .55 , .32 , 1.3)

mod8b <- smirt(dat,Qmatrix=Q , irtmodel="comp" , variance.fixed=variance.fixed ,
               b.init=b.init , a.init=a.init , maxiter=20 , est.a="2PL" )
summary(mod8b)

****
# Model 9: Unidimensional model with quadratic item response functions
# define theta
theta.k <- seq( - 6 , 6 , len=15 )
theta.k <- as.matrix( theta.k , ncol=1 )
# define design matrix
theta.kDES <- cbind( theta.k[,1] , theta.k[,1]^2 )
# define Q-matrix
Qmatrix <- matrix( 0 , I , 2 )
Qmatrix[,1] <- 1
Qmatrix[ c(3,6,7) , 2 ] <- 1
colnames(Qmatrix) <- c("F1" , "F1sq" )
# estimate model
mod9 <- smirt(dat,Qmatrix=Qmatrix , maxiter=50 , irtmodel="comp" ,
              theta.k=theta.k , theta.kDES=theta.kDES , est.a="2PL" )
summary(mod9)

****
# Model 10: Two-dimensional item response model with latent interaction
#           between dimensions
theta.k <- seq( - 6 , 6 , len=15 )
theta.k <- expand.grid( theta.k , theta.k ) # expand theta to 2 dimensions
# define design matrix
theta.kDES <- cbind( theta.k , theta.k[,1]*theta.k[,2] )
# define Q-matrix
Qmatrix <- matrix( 0 , I , 3 )
Qmatrix[,1] <- 1
Qmatrix[ 6:10 , c(2,3) ] <- 1
colnames(Qmatrix) <- c("F1" , "F2" , "F1iF2" )
# estimate model
mod10 <- smirt(dat,Qmatrix=Qmatrix , irtmodel="comp" , theta.k=theta.k ,
               theta.kDES= theta.kDES , est.a="2PL" )
summary(mod10)

*****
# Model 11: Example Quasi Monte Carlo integration
Qmatrix <- matrix( 1 , I , 1 )
mod11 <- smirt( dat , irtmodel="comp" , Qmatrix=Qmatrix , qmcnodes=1000 )
summary(mod11)

#####

```

```

## EXAMPLE 2: Dataset Reading data.read
##           Multidimensional models for dichotomous data
#####

data(data.read)
dat <- data.read
I <- ncol(dat)  # number of items

****
# Model 1: 3-dimensional 2PL model

# define Q-matrix
Qmatrix <- matrix(0,nrow=I,ncol=3)
Qmatrix[1:4,1] <- 1
Qmatrix[5:8,2] <- 1
Qmatrix[9:12,3] <- 1

# estimate model
mod1 <- smirt( dat , Qmatrix=Qmatrix , irtmodel="comp" , est.a="2PL" ,
              qmcnodes=1000 , maxiter=20)
summary(mod1)

****
# Model 2: 3-dimensional Rasch model
mod2 <- smirt( dat , Qmatrix=Qmatrix , irtmodel="comp" ,
              qmcnodes=1000 , maxiter=20)
summary(mod2)

****
# Model 3: 3-dimensional 2PL model with uncorrelated dimensions
# fix entries in variance matrix
variance.fixed <- cbind( c(1,1,2) , c(2,3,3) , 0 )
# set the following covariances to zero: cov[1,2]=cov[1,3]=cov[2,3]=0

# estimate model
mod3 <- smirt( dat , Qmatrix=Qmatrix , irtmodel="comp" , est.a="2PL" ,
              variance.fixed=variance.fixed , qmcnodes=1000 , maxiter=20)
summary(mod3)

****
# Model 4: Bifactor model with one general factor (g) and
#           uncorrelated specific factors

# define a new Q-matrix
Qmatrix1 <- cbind( 1 , Qmatrix )
# uncorrelated factors
variance.fixed <- cbind( c(1,1,1,2,2,3) , c(2,3,4,3,4,4) , 0 )
# The first dimension refers to the general factors while the other
# dimensions refer to the specific factors.
# The specification means that:
# Cov[1,2]=Cov[1,3]=Cov[1,4]=Cov[2,3]=Cov[2,4]=Cov[3,4]=0

# estimate model

```

```

mod4 <- smirt( dat , Qmatrix=Qmatrix1 , irtmodel="comp" , est.a="2PL" ,
              variance.fixed=variance.fixed , qmcnodes=1000 , maxiter=20)
summary(mod4)

#####
## SIMULATED EXAMPLE 3: Partially compensatory model
#####

**** simulate data
set.seed(7656)
I <- 10 # number of items
N <- 2000 # number of subjects
Q <- matrix( 0 , 3*I,2) # Q-matrix
Q[1:I,1] <- 1
Q[1:I + I ,2] <- 1
Q[1:I + 2*I ,1:2] <- 1
b <- matrix( runif( 3*I *2, -2 , 2 ) , nrow=3*I , 2 )
b <- b*Q
b <- round( b , 2 )
mui <- rep(0,3*I)
mui[ seq(2*I+1 , 3*I) ] <- 0.65
# generate data
dat <- matrix( NA , N , 3*I )
colnames(dat) <- paste0("It" , 1:(3*I) )
# simulate item responses
library(MASS)
theta <- MASS::mvrnorm(N , mu=c(0,0) , Sigma = matrix( c( 1.2 , .6,.6,1.6) ,2 , 2 ) )
for (ii in 1:(3*I)){
  # define probability
  tmp1 <- exp( theta[,1] * Q[ii,1] - b[ii,1] + theta[,2] * Q[ii,2] - b[ii,2] )
  # non-compensatory model
  nco1 <- ( 1 + exp( theta[,1] * Q[ii,1] - b[ii,1] ) ) *
          ( 1 + exp( theta[,2] * Q[ii,2] - b[ii,2] ) )
  co1 <- ( 1 + tmp1 )
  p1 <- tmp1 / ( mui[ii] * nco1 + ( 1 - mui[ii] )*co1 )
  dat[,ii] <- 1 * ( runif(N) < p1 )
}

*** Model 1: Joint mu.i parameter for all items
est.mu.i <- rep(0,3*I)
est.mu.i[ seq(2*I+1,3*I)] <- 1
mod1 <- smirt( dat , Qmatrix = Q , irtmodel = "partcomp" , est.mu.i=est.mu.i)
summary(mod1)

*** Model 2: Separate mu.i parameter for all items
est.mu.i[ seq(2*I+1,3*I)] <- 1:I
mod2 <- smirt( dat , Qmatrix = Q , irtmodel = "partcomp" , est.mu.i=est.mu.i)
summary(mod2)

## End(Not run)

```



---

 stratified.cronbach.alpha

*Stratified Cronbach's Alpha*


---

### Description

This function computes the stratified Cronbach's Alpha for composite scales (Cronbach, Schoenemann & McKie, 1965; Meyer, 2010).

### Usage

```
stratified.cronbach.alpha(data, itemstrata=NULL)
```

### Arguments

data	An $N \times I$ data frame
itemstrata	A matrix with two columns defining the item stratification. The first column contains the item names, the second column the item stratification label (these can be integers). The default NULL does only compute Cronbach's Alpha for the whole scale.

### Author(s)

Alexander Robitzsch

### References

Cronbach, L.J., Schoenemann, P., & McKie, D. (1965). Alpha coefficient for stratified-parallel tests. *Educational and Psychological Measurement*, **25**, 291-312.

Meyer, P. (2010). *Reliability*. Cambridge: Oxford University Press.

### Examples

```
#####
# EXAMPLE 1: data.read
#####

data( data.read )
dat <- data.read
I <- ncol(dat)

# apply function without defining item strata
stratified.cronbach.alpha( data.read )

# define item strata
itemstrata <- cbind( colnames(dat) , substring( colnames(dat) , 1 ,1 ) )
stratified.cronbach.alpha( data.read , itemstrata=itemstrata )
## scale I alpha mean.tot var.tot alpha.stratified
```

```

## 1 total 12 0.677      8.680  5.668          0.703
## 2   A   4 0.545      2.616  1.381          NA
## 3   B   4 0.381      2.811  1.059          NA
## 4   C   4 0.640      3.253  1.107          NA

## Not run:
#####
# reliability analysis in psych package
library(psych)
# Cronbach's alpha and item discriminations
psych::alpha( dat )
# McDonald's omega
psych::omega(dat , nfactors=1)    # 1 factor
##   Alpha:                0.69
##   Omega Total           0.69
## => Note that alpha in this function is the standardized Cronbach's
##   alpha, i.e. alpha computed for standardized variables.
psych::omega(dat , nfactors=2)    # 2 factors
##   Omega Total           0.72
psych::omega(dat , nfactors=3)    # 3 factors
##   Omega Total           0.74

## End(Not run)

```

---

summary.mcmc.sirt      *Summary Method for Objects of Class mcmc.sirt*

---

## Description

S3 method to summarize objects of class `mcmc.sirt`. This object is generated by following functions: [mcmc.2pno](#), [mcmc.2pnoh](#), [mcmc.3pno.testlet](#), [mcmc.2pno.ml](#)

## Usage

```
## S3 method for class 'mcmc.sirt'
summary(object,digits=3,...)
```

## Arguments

<code>object</code>	Object of class <code>mcmc.sirt</code>
<code>digits</code>	Number of digits after decimal
<code>...</code>	Further arguments to be passed

## Author(s)

Alexander Robitzsch

## See Also

[mcmc.2pno](#), [mcmc.2pnoh](#), [mcmc.3pno.testlet](#), [mcmc.2pno.ml](#)

---

 tam2mirt

---

*Converting a fitted TAM Object into a mirt Object*


---

### Description

Converts a fitted TAM object into a mirt object. As a by-product, lavaan syntax is generated which can be used with [lavaan2mirt](#) for re-estimating the model in the **mirt** package. Up to now, only single group models are supported. There must not exist background covariates (no latent regression models!).

### Usage

```
tam2mirt(tamobj)
```

### Arguments

tamobj            Object of class `TAM::tam.mml`

### Value

A list with following entries

mirt	Object generated by mirt function if <code>est.mirt=TRUE</code>
mirt.model	Generated mirt model
mirt.syntax	Generated mirt syntax
mirt.pars	Generated parameter specifications in mirt
lavaan.model	Used lavaan model transformed by <code>lavaanify</code> function
dat	Used dataset. If necessary, only items used in the model are included in the dataset.
lavaan.syntax.fixed	Generated lavaan syntax with fixed parameter estimates.
lavaan.syntax.freed	Generated lavaan syntax with freed parameters for estimation.

### Author(s)

Alexander Robitzsch

### See Also

See [mirt.wrapper](#) for convenience wrapper functions for `mirt` objects.

See [lavaan2mirt](#) for converting lavaan syntax to mirt syntax.

## Examples

```
## Not run:
library(TAM)
library(mirt)

#####
# EXAMPLE 1: Estimations in TAM for data.read dataset
#####

data(data.read)
dat <- data.read

*****
*** Model 1: Rasch model
*****

# estimation in TAM package
mod <- TAM::tam.mml( dat )
summary(mod)
# conversion to mirt
res <- tam2mirt(mod)
# generated lavaan syntax
cat(res$lavaan.syntax.fixed)
cat(res$lavaan.syntax.freed)
# extract object of class mirt
mres <- res$mirt
# print and parameter values
print(mres)
mirt::mod2values(mres)
# model fit
mirt::M2(mres)
# residual statistics
mirt::residuals(mres , type="Q3")
mirt::residuals(mres , type="LD")
# item fit
mirt::itemfit(mres)
# person fit
mirt::personfit(mres)
# compute several types of factor scores (quite slow)
f1 <- mirt::fscores(mres, method='WLE',response.pattern=dat[1:10,])
# method = MAP and EAP also possible
# item plot
mirt::itemplot(mres,"A3") # item A3
mirt::itemplot(mres,4) # fourth item
# some more plots
plot(mres,type="info")
plot(mres,type="score")
plot(mres,type="trace")
# compare estimates with estimated Rasch model in mirt
mres1 <- mirt::mirt(dat,1,"Rasch" )
print(mres1)
mirt.wrapper.coef(mres1)
```

```

*****
*** Model 2: 2PL model
*****

# estimation in TAM
mod <- TAM::tam.mml.2pl( dat )
summary(mod)
# conversion to mirt
res <- tam2mirt(mod)
mres <- res$mirt
# lavaan syntax
cat(res$lavaan.syntax.fixed)
cat(res$lavaan.syntax.freed)
# parameter estimates
print(mres)
mod2values(mres)
mres@nest # number of estimated parameters
# some plots
plot(mres,type="info")
plot(mres,type="score")
plot(mres,type="trace")
# model fit
mirt::M2(mres)
# residual statistics
mirt::residuals(mres , type="Q3")
mirt::residuals(mres , type="LD")
# item fit
mirt::itemfit(mres)

*****
*** Model 3: 3-dimensional Rasch model
*****

# define Q-matrix
Q <- matrix( 0 , nrow=12 , ncol=3 )
Q[ cbind(1:12 , rep(1:3,each=4) ) ] <- 1
rownames(Q) <- colnames(dat)
colnames(Q) <- c("A","B","C")
# estimation in TAM
mod <- TAM::tam.mml( resp=dat , Q=Q , control=list(snodes=1000,maxiter=30) )
summary(mod)
# mirt conversion
res <- tam2mirt(mod)
mres <- res$mirt
# mirt syntax
cat(res$mirt.syntax)
## Dim01=1,2,3,4
## Dim02=5,6,7,8
## Dim03=9,10,11,12
## COV = Dim01*Dim01,Dim02*Dim02,Dim03*Dim03,Dim01*Dim02,Dim01*Dim03,Dim02*Dim03
## MEAN = Dim01,Dim02,Dim03
# lavaan syntax

```

```

cat(res$lavaan.syntax.freed)
## Dim01 =~ 1*A1+1*A2+1*A3+1*A4
## Dim02 =~ 1*B1+1*B2+1*B3+1*B4
## Dim03 =~ 1*C1+1*C2+1*C3+1*C4
## A1 | t1_1*t1
## A2 | t1_2*t1
## A3 | t1_3*t1
## A4 | t1_4*t1
## B1 | t1_5*t1
## B2 | t1_6*t1
## B3 | t1_7*t1
## B4 | t1_8*t1
## C1 | t1_9*t1
## C2 | t1_10*t1
## C3 | t1_11*t1
## C4 | t1_12*t1
## Dim01 ~ 0*1
## Dim02 ~ 0*1
## Dim03 ~ 0*1
## Dim01 ~~ Cov_11*Dim01
## Dim02 ~~ Cov_22*Dim02
## Dim03 ~~ Cov_33*Dim03
## Dim01 ~~ Cov_12*Dim02
## Dim01 ~~ Cov_13*Dim03
## Dim02 ~~ Cov_23*Dim03
# model fit
mirt::M2(mres)
# residual statistics
residuals(mres,type="LD")
# item fit
mirt::itemfit(mres)

#*****
#*** Model 4: 3-dimensional 2PL model
#*****

# estimation in TAM
mod <- TAM::tam.mml.2pl( resp=dat , Q=Q , control=list(snodes=1000,maxiter=30) )
summary(mod)
# mirt conversion
res <- tam2mirt(mod)
mres <- res$mirt
# generated lavaan syntax
cat(res$lavaan.syntax.fixed)
cat(res$lavaan.syntax.freed)
# write lavaan syntax on disk
sink( "mod4_lav_freed.txt" , split=TRUE )
cat(res$lavaan.syntax.freed)
sink()
# some statistics from mirt
print(mres)
summary(mres)
mirt::M2(mres)

```

```

mirt::residuals(mres)
mirt::itemfit(mres)

# estimate mirt model by using the generated lavaan syntax with freed parameters
res2 <- lavaan2mirt( dat , res$lavaan.syntax.freed ,
                    technical=list(NCYCLES=3) , verbose=TRUE)
                    # use only few cycles for illustrational purposes
mirt.wrapper.coef(res2$mirt)
summary(res2$mirt)
print(res2$mirt)

#####
# EXAMPLE 4: mirt conversions for polytomous dataset data.big5
#####

data(data.big5)
# select some items
items <- c( grep( "0" , colnames(data.big5) , value=TRUE )[1:6] ,
           grep( "N" , colnames(data.big5) , value=TRUE )[1:4] )
# 03 08 013 018 023 028 N1 N6 N11 N16
dat <- data.big5[ , items ]
library(psych)
psych::describe(dat)

library(TAM)
#####
*** Model 1: Partial credit model in TAM
mod1 <- TAM::tam.mml( dat[,1:6] )
summary(mod1)
# convert to mirt object
mmod1 <- tam2mirt( mod1 )
rmod1 <- mmod1$mirt
# coefficients in mirt
coef(rmod1)
mirt.wrapper.coef(rmod1)
# model fit
mirt::M2(rmod1)
# item fit
mirt::itemfit(rmod1)
# plots
plot(rmod1,type="trace")
plot(rmod1, type = "trace", which.items = 1:4 )
mirt::itemplot(rmod1,"03")

#####
*** Model 2: Generalized partial credit model in TAM
mod2 <- TAM::tam.mml.2pl( dat[,1:6] , irtmodel="GPCM" )
summary(mod2)
# convert to mirt object
mmod2 <- tam2mirt( mod2 )
rmod2 <- mmod2$mirt
# coefficients in mirt
mirt.wrapper.coef(rmod2)

```

```
# model fit
mirt::M2(rmod2)
# item fit
mirt::itemfit(rmod2)

## End(Not run)
```

---

testlet.marginalized *Marginal Item Parameters from a Testlet (Bifactor) Model*

---

### Description

This function computes marginal item parameters of a general factor if item parameters from a testlet (bifactor) model are provided as an input (see Details).

### Usage

```
testlet.marginalized(tam.fa.obj=NULL ,a1=NULL, d1=NULL, testlet=NULL,
  a.testlet=NULL, var.testlet=NULL)
```

### Arguments

tam.fa.obj	Optional object of class tam.fa generated by <code>TAM::tam.fa</code> from the <b>TAM</b> package.
a1	Vector of item discriminations of general factor
d1	Vector of item intercepts of general factor
testlet	Integer vector of testlet (bifactor) identifiers (must be integers between 1 to $T$ ).
a.testlet	Vector of testlet (bifactor) item discriminations
var.testlet	Vector of testlet (bifactor) variances

### Details

A testlet (bifactor) model is assumed to be estimated:

$$P(X_{pit} = 1 | \theta_p, u_{pt}) = \text{invlogit}(a_{i1}\theta_p + a_t u_{pt} - d_i)$$

with  $\text{Var}(u_{pt}) = \sigma_t^2$ . This multidimensional item response model with locally independent items is equivalent to a unidimensional IRT model with locally dependent items (Ip, 2010). Marginal item parameters  $a_i^*$  and  $d_i^*$  are obtained according to the response equation

$$P(X_{pit} = 1 | \theta_p^*) = \text{invlogit}(a_i^* \theta_p^* - d_i^*)$$

Calculation details can be found in Ip (2010).

### Value

A data frame containing all input item parameters and marginal item intercept  $d_i^*$  (d1\_marg) and marginal item slope  $a_i^*$  (a1\_marg).



**Author(s)**

Alexander Robitzsch

**References**

Ip, E. H. (2010). Empirically indistinguishable multidimensional IRT and locally dependent unidimensional item response models. *British Journal of Mathematical and Statistical Psychology*, **63**, 395-416.

**See Also**

For estimating a testlet (bifactor) model see [TAM: :tam.fa](#).

**Examples**

```
#####
# EXAMPLE 1: Small numeric example for Rasch testlet model
#####

# Rasch testlet model with 9 items contained into 3 testlets
# the third testlet has essentially no dependence and therefore
# no testlet variance
testlet <- rep( 1:3 , each=3 )
a1 <- rep(1 , 9 ) # item slopes first dimension
d1 <- rep( c(-1.25,0,1.5) , 3 ) # item intercepts
a.testlet <- rep( 1 , 9 ) # item slopes testlets
var.testlet <- c( .8 , .2 , 0 ) # testlet variances

# apply function
res <- testlet.marginalized( a1=a1 , d1=d1 , testlet=testlet ,
                           a.testlet=a.testlet , var.testlet=var.testlet )
round( res , 2 )
##   item testlet a1    d1 a.testlet var.testlet a1_marg d1_marg
##  1  1      1  1 -1.25      1      0.8  0.89 -1.11
##  2  2      1  1  0.00      1      0.8  0.89  0.00
##  3  3      1  1  1.50      1      0.8  0.89  1.33
##  4  4      2  1 -1.25      1      0.2  0.97 -1.21
##  5  5      2  1  0.00      1      0.2  0.97  0.00
##  6  6      2  1  1.50      1      0.2  0.97  1.45
##  7  7      3  1 -1.25      1      0.0  1.00 -1.25
##  8  8      3  1  0.00      1      0.0  1.00  0.00
##  9  9      3  1  1.50      1      0.0  1.00  1.50

## Not run:
#####
# EXAMPLE 2: Dataset reading
#####

library(TAM)
data(data.read)
resp <- data.read
maxiter <- 100
```

```

# Model 1: Rasch testlet model with 3 testlets
dims <- substring( colnames(resp),1,1 ) # define dimensions
mod1 <- TAM::tam.fa( resp=resp , irtmodel="bifactor1" , dims=dims ,
                    control=list(maxiter=maxiter) )
# marginal item parameters
res1 <- testlet.marginalized( mod1 )

####
# Model 2: estimate bifactor model but assume that items 3 and 5 do not load on
#           specific factors
dims1 <- dims
dims1[c(3,5)] <- NA
mod2 <- TAM::tam.fa( resp=resp , irtmodel="bifactor2" , dims=dims1 ,
                    control=list(maxiter=maxiter) )
res2 <- testlet.marginalized( mod2 )
res2

## End(Not run)

```

---

testlet.yen.q3

*Q<sub>3</sub> Statistic of Yen (1984) for Testlets*


---

## Description

This function calculates the average  $Q_3$  statistic (Yen, 1984) within and between testlets.

## Usage

```
testlet.yen.q3(q3.res, testlet.matrix)
```

## Arguments

`q3.res` An object generated by [yen.q3](#)

`testlet.matrix` A matrix with two columns. The first column contains names of the testlets and the second names of the items. See the examples for the definition of such matrices.

## Value

A list with following entries

`testlet.q3` Data frame with average  $Q_3$  statistics within testlets

`testlet.q3.korr` Matrix of average  $Q_3$  statistics within and between testlets

## Author(s)

Alexander Robitzsch

## References

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.

## See Also

For estimating all  $Q_3$  statistics between item pairs use [yen.q3](#).

## Examples

```
#####
# EXAMPLE 1: data.read. The 12 items are arranged in 4 testlets
#####
data(data.read)

# estimate the Rasch model
mod <- rasch.mm12( data.read)
mod$item

# estimate WLEs
mod.wle <- wle.rasch( dat = data.read , b = mod$item$b )

# Yen's Q3 statistic
mod.q3 <- yen.q3( dat = data.read , theta = mod.wle$theta , b = mod$item$b )

# Yen's Q3 statistic with testlets
items <- colnames(data.read)
testlet.matrix <- cbind( substring( items,1,1) , items )
mod.testletq3 <- testlet.yen.q3( q3.res=mod.q3,testlet.matrix=testlet.matrix)
mod.testletq3
```

---

tetrachoric2

*Tetrachoric Correlation Matrix*

---

## Description

This function estimates a tetrachoric correlation matrix according to the maximum likelihood estimation of Olsson (Olsson, 1979; method="O1"), the Tucker method (Method 2 of Froemel, 1971; method="Tu") and Divgi (1979, method="Di"). In addition, an alternative non-iterative approximation of Bonnett and Price (2005; method="Bo") is provided.

## Usage

```
tetrachoric2(dat, method="O1" , delta = 0.007, maxit = 1000000, cor.smooth=TRUE,
  progress=TRUE)
```

**Arguments**

dat	A data frame of dichotomous response
method	Computation method for calculating the tetrachoric correlation. The ML method is method="Ol" (which is the default), the Tucker method is method="Tu", the Divgi method is method="Di" the method of Bonett and Price (2005) is method="Bo".
delta	The step parameter. It is set by default to $2^{-7}$ which is approximately .007.
maxit	Maximum number of iterations.
cor.smooth	Should smoothing of the tetrachoric correlation matrix be performed to ensure positive definiteness? Choosing cor.smooth=TRUE, the function cor.smooth from the <b>psych</b> package is used for obtaining a positive definite tetrachoric correlation matrix.
progress	Display progress? Default is TRUE.

**Value**

A list with following entries

tau	Item thresholds
rho	Tetrachoric correlation matrix

**Author(s)**

Alexander Robitzsch

The code is adapted from an R script of Cengiz Zopluoglu. See <http://sites.education.miami.edu/zopluoglu/software-programs/>.

**References**

- Bonett, D. G., & Price, R. M. (2005). Inferential methods for the tetrachoric correlation coefficient. *Journal of Educational and Behavioral Statistics*, **30**, 213-225.
- Divgi, D. R. (1979). Calculation of the tetrachoric correlation coefficient. *Psychometrika*, **44**, 169-172.
- Froemel, E. C. (1971). A comparison of computer routines for the calculation of the tetrachoric correlation coefficient. *Psychometrika*, **36**, 165-174.
- Olsson, U. (1979). Maximum likelihood estimation of the polychoric correlation coefficient. *Psychometrika*, **44**, 443-460.

**See Also**

See also the tetrachoric function in the **psych** package and the function tet in the **irtoys** package. See [polychoric2](#) for estimating polychoric correlations.

**Examples**

```
#####
# EXAMPLE 1: data.read
#####

data(data.read)

# tetrachoric correlation from psych package
library(psych)
t0 <- psych::tetrachoric( data.read )$rho
# Olsson method (maximum likelihood estimation)
t1 <- tetrachoric2( data.read )$rho
# Divgi method
t2 <- tetrachoric2( data.read , method="Di" )$rho
# Tucker method
t3 <- tetrachoric2( data.read , method="Tu" )$rho
# Bonett method
t4 <- tetrachoric2( data.read , method="Bo" )$rho

# maximum absolute deviation ML method
max( abs( t0 - t1 ) )
## [1] 0.008224986
# mean absolute deviation Divgi method
max( abs( t0 - t2 ) )
## [1] 0.1766688
# mean absolute deviation Tucker method
max( abs( t0 - t3 ) )
## [1] 0.1766292
# mean absolute deviation Bonett method
max( abs( t0 - t4 ) )
## [1] 0.05695522
```

truescore.irt

*Conversion of Trait Scores  $\theta$  into True Scores  $\tau(\theta)$* **Description**

This function computes the true score  $\tau = \tau(\theta) = \sum_{i=1}^I P_i(\theta)$  in a unidimensional item response model with  $I$  items. In addition, it also transforms conditional standard errors if they are provided.

**Usage**

```
truescore.irt(A, B, c = NULL, d = NULL, theta = seq(-3, 3, len = 21),
             error = NULL, pid = NULL, h = 0.001)
```

**Arguments**

A Matrix or vector of item slopes. See Examples for polytomous responses.

B	Matrix or vector of item intercepts. Note that the entries in B refer to item intercepts and not to item difficulties.
c	Optional vector of guessing parameters
d	Optional vector of slipping parameters
theta	Vector of trait values
error	Optional vector of standard errors of trait
pid	Optional vector of person identifiers
h	Numerical differentiation parameter

### Details

In addition, the function  $\pi(\theta) = \frac{1}{I} \cdot \tau(\theta)$  of the expected percent score is approximated by a logistic function

$$\pi(\theta) \approx l + (u - l) \cdot \text{invlogit}(a\theta + b)$$

### Value

A data frame with following columns:

truescore	True scores $\tau = \tau(\theta)$
truescore.error	Standard errors of true scores
percscore	Expected correct scores which is $\tau$ divided by the maximum true score
percscore.error	Standard errors of expected correct scores
lower	The $l$ parameter
upper	The $u$ parameter
a	The $a$ parameter
b	The $b$ parameter

### Author(s)

Alexander Robitzsch

### Examples

```
#####
# EXAMPLE 1: Dataset with mixed dichotomous and polytomous responses
#####

data(data.mixed1)
dat <- data.mixed1

#####
#****
# Model 1: Partial credit model
# estimate model with TAM package
library(TAM)
```

```

mod1 <- TAM::tam.mml( dat )
# estimate person parameter estimates
wmod1 <- TAM::tam.wle( mod1 )
wmod1 <- wmod1[ order(wmod1$theta) , ]
# extract item parameters
A <- mod1$B[,-1,1]
B <- mod1$AXsi[,-1]
# person parameters and standard errors
theta <- wmod1$theta
error <- wmod1$error

# estimate true score transformation
dfr <- truescore.irt( A=A , B=B , theta=theta , error=error )

# plot different person parameter estimates and standard errors
par(mfrow=c(2,2))
plot( theta , dfr$truescore , pch=16 , cex=.6 , xlab=expression(theta) , type="l",
      ylab=expression(paste( tau , "(" , theta , ")" )) , main="True Score Transformation" )
plot( theta , dfr$percscore , pch=16 , cex=.6 , xlab=expression(theta) , type="l",
      ylab=expression(paste( pi , "(" , theta , ")" )) , main="Percent Score Transformation" )
points( theta , dfr$lower + (dfr$upper-dfr$lower)*plogis(dfr$a*theta+dfr$b) , col=2 , lty=2)
plot( theta , error , pch=16 , cex=.6 , xlab=expression(theta) , type="l",
      ylab=expression(paste("SE(",theta , ")" )) , main="Standard Error Theta" )
plot( dfr$truescore , dfr$truescore.error , pch=16 , cex=.6 , xlab=expression(tau) ,
      ylab=expression(paste("SE(",tau , ")" )) , main="Standard Error True Score Tau" ,
      type="l")
par(mfrow=c(1,1))

## Not run:
#####
# Model 2: Generalized partial credit model
mod2 <- TAM::tam.mml.2pl( dat , irtmodel="GPCM")
# estimate person parameter estimates
wmod2 <- TAM::tam.wle( mod2 )
# extract item parameters
A <- mod2$B[,-1,1]
B <- mod2$AXsi[,-1]
# person parameters and standard errors
theta <- wmod2$theta
error <- wmod2$error
# estimate true score transformation
dfr <- truescore.irt( A=A , B=B , theta=theta , error=error )

#####
# EXAMPLE 2: Dataset Reading data.read
#####
data(data.read)

#####
# Model 1: estimate difficulty + guessing model
mod1 <- rasch.mml2( data.read , fixed.c = rep(.25,12) )
mod1$person <- mod1$person[ order( mod1$person$EAP) , ]
# person parameters and standard errors

```

```

theta <- mod1$person$EAP
error <- mod1$person$SE.EAP
A <- rep(1,12)
B <- - mod1$item$b
c <- rep(.25,12)
# estimate true score transformation
dfr <- truescore.irt( A=A , B=B , theta=theta , error=error ,c=c)

plot( theta , dfr$percscore , pch=16 , cex=.6 , xlab=expression(theta) , type="l",
      ylab=expression(paste( pi , "(" , theta , ")" )) , main="Percent Score Transformation" )
points( theta , dfr$lower + (dfr$upper-dfr$lower)*plogis(dfr$a*theta+dfr$b) , col=2 , lty=2)

#####
# Model 2: Rasch model
mod2 <- rasch.mml2( data.read )
# person parameters and standard errors
theta <- mod2$person$EAP
error <- mod2$person$SE.EAP
A <- rep(1,12)
B <- - mod2$item$b
# estimate true score transformation
dfr <- truescore.irt( A=A , B=B , theta=theta , error=error )

## End(Not run)

```

---

 unidim.test.csn

*Test for Unidimensionality of CSN*


---

## Description

This function tests whether item covariances given the sum score are non-positive (CSN; see Junker 1993), i.e. for items  $i$  and  $j$  it holds that

$$\text{Cov}(X_i, X_j | X^+) \leq 0$$

Note that this function only works for dichotomous data.

## Usage

```
unidim.test.csn(dat, RR = 400, prop.perm = 0.75, progress = TRUE)
```

## Arguments

dat	Data frame with dichotomous item responses. All persons with (some) missing responses are removed.
RR	Number of permutations used for statistical testing
prop.perm	A positive value indicating the amount of permutation in an existing permuted data set
progress	An optional logical indicating whether computation progress should be displayed



## Details

For each item pair  $(i, j)$  and a each sum score group  $k$  a conditional covariance  $r(i, j|k)$  is calculated. Then, the test statistic for CSN is

$$h = \sum_{k=1}^{I-1} \frac{n_k}{n} \max_{i,j} r(i, j|k)$$

where  $n_k$  is the number of persons in score group  $k$ . "Large values" of  $h$  are not in agreement with the null hypothesis of non-positivity of conditional covariances.

The distribution of the test statistic  $h$  under the null hypothesis is empirically obtained by column wise permutation of items within all score groups. In the population, this procedure corresponds to conditional covariances of zero. See de Gooijer and Yuan (2011) for more details.

## Value

A list with following entries

stat	Value of the statistic
stat_perm	Distribution of statistic under $H_0$ of permuted dataset
p	The corresponding p value of the statistic
H0_quantiles	Quantiles of the statistic under permutation (the null hypothesis $H_0$ )

## Author(s)

Alexander Robitzsch

## References

De Gooijer, J. G., & Yuan, A. (2011). Some exact tests for manifest properties of latent trait models. *Computational Statistics and Data Analysis*, **55**, 34-44.

Junker, B.W. (1993). Conditional association, essential independence, and monotone unidimensional item response models. *Annals of Statistics*, **21**, 1359-1378.

## Examples

```
#####
# EXAMPLE 1: Dataset data.read
#####

data(data.read)
dat <- data.read
set.seed(778)
res <- unidim.test.csn( dat )
## CSN Statistic = 0.04737 , p = 0.02

## Not run:
#####
# SIMULATED EXAMPLE 2: CSN statistic for two-dimensional simulated data
#####
```

```

set.seed(775)
N <- 2000
I <- 30 # number of items
rho <- .60 # correlation between 2 dimensions
t0 <- rnorm(N)
t1 <- sqrt(rho)*t0 + sqrt(1-rho)*rnorm(N)
t2 <- sqrt(rho)*t0 + sqrt(1-rho)*rnorm(N)
dat1 <- sim.raschtype(t1 , b=seq(-1.5,1.5,length=I/2) )
dat2 <- sim.raschtype(t2 , b=seq(-1.5,1.5,length=I/2) )
dat <- as.matrix(cbind( dat1 , dat2) )
res <- unidim.test.csn( dat )
## CSN Statistic = 0.06056 , p = 0.02

## End(Not run)

```

---

wle.rasch

*Weighted Likelihood Estimation of Person Abilities*


---

## Description

This function computes weighted likelihood estimates for dichotomous responses based on the Rasch model (Warm, 1989).

## Usage

```

wle.rasch(dat, dat.resp = NULL, b, itemweights = 1 + 0 * b,
  theta = rep(0, nrow(dat)), conv = 0.001, maxit = 200,
  wle.adj=0 , progress=FALSE)

```

## Arguments

dat	An $N \times I$ data frame of dichotomous item responses
dat.resp	Optional data frame with dichotomous response indicators
b	Vector of length $I$ with fixed item difficulties
itemweights	Optional vector of fixed item discriminations
theta	Optional vector of initial person parameter estimates
conv	Convergence criterion
maxit	Maximal number of iterations
wle.adj	Constant for WLE adjustment
progress	Display progress?

**Value**

A list with following entries

theta	Estimated weighted likelihood estimate
dat.resp	Data frame with dichotomous response indicators. A one indicates an observed response, a zero a missing response. See also dat.resp in the list of arguments of this function.
p.ia	Matrix with expected item response, i.e. the probabilities $P(X_{pi} = 1 \theta_p) = \text{invlogit}(\theta_p - b_i)$ .
wle	WLE reliability (Adams, 2005)

**Author(s)**

Alexander Robitzsch

**References**

- Adams, R. J. (2005). Reliability as a measurement design effect. *Studies in Educational Evaluation*, **31**, 162-172.
- Warm, T. A. (1989). Weighted likelihood estimation of ability in item response theory. *Psychometrika*, **54**, 427-450.

**See Also**

For standard errors of weighted likelihood estimates estimated via jackknife see [wle.rasch.jackknife](#).

For a joint estimation of item and person parameters see the joint maximum likelihood estimation method in [rasch.jml](#).

**Examples**

```
#####
# EXAMPLE 1: Dataset Reading
#####
data(data.read)

# estimate the Rasch model
mod <- rasch.mml2(data.read)
mod$item

# estimate WLEs
mod.wle <- wle.rasch( dat = data.read , b = mod$item$b )
```

---

wle.rasch.jackknife     *Standard Error Estimation of WLE by Jackknifing*

---

### Description

This function calculates standard errors of WLEs (Warm, 1989) for stratified item designs and item designs with testlets for the Rasch model.

### Usage

```
wle.rasch.jackknife(dat, b, itemweights = 1 + 0 * b, pid = NULL,
  testlet = NULL, stratum = NULL, size.itempop = NULL)
```

### Arguments

<code>dat</code>	An $N \times I$ data frame of item responses
<code>b</code>	Vector of item difficulties
<code>itemweights</code>	Weights for items, i.e. fixed item discriminations
<code>pid</code>	Person identifier
<code>testlet</code>	A vector of length $I$ which defines which item belongs to which testlet. If some items does not belong to any testlet, then define separate testlet labels for these single items.
<code>stratum</code>	Item stratum
<code>size.itempop</code>	Number of items in an item stratum of the finite item population.

### Details

The idea of Jackknife in item response models can be found in Wainer and Wright (1980).

### Value

A list with following entries:

<code>wle</code>	Data frame with some estimated statistics. The column <code>wle</code> is the WLE and <code>wle.jackse</code> its corresponding standard error estimated by jackknife.
<code>wle.rel</code>	WLE reliability (Adams, 2005)

### Author(s)

Alexander Robitzsch

## References

- Adams, R. J. (2005). Reliability as a measurement design effect. *Studies in Educational Evaluation*, **31**, 162-172.
- Gershunskaya, J., Jiang, J., & Lahiri, P. (2009). Resampling methods in surveys. In D. Pfeiffermann and C.R. Rao (Eds.). *Handbook of Statistics 29B; Sample Surveys: Inference and Analysis* (pp. 121-151). Amsterdam: North Holland.
- Wainer, H., & Wright, B. D. (1980). Robust estimation of ability in the Rasch model. *Psychometrika*, **45**, 373-391.
- Warm, T. A. (1989). Weighted likelihood estimation of ability in item response theory. *Psychometrika*, **54**, 427-450.

## See Also

[wle.rasch](#)

## Examples

```
#####
# EXAMPLE 1: Dataset Reading
#####
data(data.read)
dat <- data.read

# estimation of the Rasch model
res <- rasch.mml2( dat , parm.conv = .001)

# WLE estimation
wle1 <- wle.rasch(dat, b = res$item$thresh )

# simple jackknife WLE estimation
wle2 <- wle.rasch.jackknife(dat, b =res$item$thresh )
## WLE Reliability = 0.651

# SE(WLE) for testlets A, B and C
wle3 <- wle.rasch.jackknife(dat, b =res$item$thresh ,
  testlet = substring( colnames(dat),1,1) )
## WLE Reliability = 0.572

# SE(WLE) for item strata A,B, C
wle4 <- wle.rasch.jackknife(dat, b =res$item$thresh ,
  stratum = substring( colnames(dat),1,1) )
## WLE Reliability = 0.683

# SE (WLE) for finite item strata
# A (10 items) , B (7 items) , C (4 items -> no sampling error)
# in every stratum 4 items were sampled
size.itempop <- c(10,7,4)
names(size.itempop) <- c("A","B","C")
wle5 <- wle.rasch.jackknife(dat, b =res$item$thresh ,
  stratum = substring( colnames(dat),1,1) ,
```

```

      size.itempop = size.itempop )
## Stratum A (Mean) Correction Factor 0.6
## Stratum B (Mean) Correction Factor 0.42857
## Stratum C (Mean) Correction Factor 0
## WLE Reliability = 0.876

# compare different estimated standard errors
a2 <- aggregate( wle2$wle$wle.jackse , list( wle2$wle$wle) , mean )
colnames(a2) <- c("wle" , "se.simple")
a2$se.testlet <- aggregate( wle3$wle$wle.jackse , list( wle3$wle$wle) , mean )[,2]
a2$se.strata <- aggregate( wle4$wle$wle.jackse , list( wle4$wle$wle) , mean )[,2]
a2$se.finitepop.strata <- aggregate( wle5$wle$wle.jackse ,
list( wle5$wle$wle) , mean )[,2]
round( a2 , 3 )
## > round( a2 , 3 )
##      wle se.simple se.testlet se.strata se.finitepop.strata
## 1  -5.085   0.440   0.649   0.331   0.138
## 2  -3.114   0.865   1.519   0.632   0.379
## 3  -2.585   0.790   0.849   0.751   0.495
## 4  -2.133   0.715   1.177   0.546   0.319
## 5  -1.721   0.597   0.767   0.527   0.317
## 6  -1.330   0.633   0.623   0.617   0.377
## 7  -0.942   0.631   0.643   0.604   0.365
## 8  -0.541   0.655   0.678   0.617   0.384
## 9  -0.104   0.671   0.646   0.659   0.434
## 10  0.406   0.771   0.706   0.751   0.461
## 11  1.080   1.118   0.893   1.076   0.630
## 12  2.332   0.400   0.631   0.272   0.195

```

---

 yen.q3

*Estimation of the  $Q_3$  Statistic (Yen, 1984)*


---

## Description

This function estimates the  $Q_3$  statistic according to Yen (1984). The statistic  $Q_3$  is calculated for every item pair  $(i, j)$  which is the correlation between item residuals after fitting the Rasch model.

## Usage

```
yen.q3(dat, theta, b, progress=TRUE)
```

## Arguments

dat	An $N \times I$ data frame of dichotomous item responses
theta	Vector of length $N$ of person parameter estimates (e.g. obtained from <a href="#">wle.rasch</a> )
b	Vector of length $I$ (e.g. obtained from <a href="#">rasch.mml2</a> )
progress	Should iteration progress be displayed?

**Value**

A list with following entries

q3.matrix	An $I \times I$ matrix of $Q_3$ statistics
q3.long	Just the q3.matrix in long matrix format where every row corresponds to an item pair
expected	An $N \times I$ matrix of expected probabilities by the Rasch model
residual	An $N \times I$ matrix of residuals obtained after fitting the Rasch model
Q3.stat	Vector with descriptive statistics of $Q_3$

**Author(s)**

Alexander Robitzsch

**References**

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.

**See Also**

For the estimation of the average  $Q_3$  statistic within testlets see [testlet.yen.q3](#).

For modelling testlet effects see [mcmc.3pno.testlet](#).

For handling local dependencies in IRT models see [rasch.copula2](#), [rasch.pml3](#) or [rasch.pairwise.itemcluster](#).

**Examples**

```
#####
# EXAMPLE 1: data.read. The 12 items are arranged in 4 testlets
#####
data(data.read)

# estimate the Rasch model
mod <- rasch.mml2( data.read)
# estimate WLEs
mod.wle <- wle.rasch( dat = data.read , b = mod$item$b )
# calculate Yen's Q3 statistic
mod.q3 <- yen.q3( dat = data.read , theta = mod.wle$theta , b = mod$item$b )
##   Yen's Q3 Statistic based on an estimated theta score
##   *** 12 Items | 66 item pairs
##   *** Q3 Descriptives
##           M      SD   Min   10%   25%   50%   75%   90%   Max
##   -0.085  0.110 -0.261 -0.194 -0.152 -0.107 -0.051  0.041  0.412

# plot Q3 statistics
I <- ncol(data.read)
image( 1:I , 1:I , mod.q3$q3.matrix , col=gray( 1 - (0:32)/32) ,
      xlab="Item" , ylab="Item")
```

```
abline(v=c(5,9)) # borders for testlets
abline(h=c(5,9))

## Not run:
# obtain Q3 statistic from modelfit.sirt function which is based on the
# posterior distribution of theta and not on observed values
fitmod <- modelfit.sirt( mod )
# extract Q3 statistic
q3stat <- fit$itempairs$Q3
## > summary(q3stat)
##      Min. 1st Qu.  Median      Mean 3rd Qu.      Max.
## -0.21760 -0.11590 -0.07280 -0.05545 -0.01220  0.44710
## > sd(q3stat)
## [1] 0.1101451

## End(Not run)
```



# Index

- \*Topic **ADISOP model**
  - fit.isop, 121
  - isop, 154
- \*Topic **Alignment**
  - invariance.alignment, 141
- \*Topic **Approximate Invariance**
  - invariance.alignment, 141
- \*Topic **Belief function**
  - fuzdiscr, 126
- \*Topic **Beta item response model**
  - brm-Methods, 14
- \*Topic **Bifactor model**
  - testlet.marginalized, 392
- \*Topic **Bivariate normal distribution**
  - pbivnorm2, 249
- \*Topic **Bradley-Terry model**
  - btm, 18
- \*Topic **Classification accuracy**
  - class.accuracy.rasch, 25
- \*Topic **Clustering**
  - fuzcluster, 123
- \*Topic **ConQuest**
  - R2conquest, 274
- \*Topic **DETECT**
  - ccov.np, 24
  - conf.detect, 26
  - detect.index, 101
  - expl.detect, 116
- \*Topic **DIF variance**
  - dif.variance, 105
- \*Topic **Differential item functioning (DIF)**
  - dif.logistic.regression, 102
  - dif.strata.variance, 104
  - dif.variance, 105
- \*Topic **Dirichlet distribution**
  - dirichlet.mle, 106
  - dirichlet.simul, 109
- \*Topic **Eigenvalues**
  - eigenvalues.manymatrices, 110
  - eigenvalues.sirt, 112
- \*Topic **Eigenvector method**
  - rasch.evm.pcm, 298
- \*Topic **Equating**
  - equating.rasch, 113
  - equating.rasch.jackknife, 115
  - linking.haberman, 179
- \*Topic **Facets model**
  - rm.facets, 356
- \*Topic **Factor scores**
  - R2noharm.EAP, 287
- \*Topic **Functional unidimensional item response model**
  - f1d.irt, 117
- \*Topic **Fuzzy data**
  - fuzcluster, 123
  - fuzdiscr, 126
- \*Topic **Generalized logistic item response model**
  - latent.regression.em.raschtype, 161
  - pgenlogis, 258
- \*Topic **Grade of membership model**
  - gom.em, 129
  - gom.jml, 137
- \*Topic **Graphical modeling**
  - sia.sirt, 366
- \*Topic **Group parameters**
  - mle.pcm.group, 230
- \*Topic **IRT copula models**
  - person.parameter.rasch.copula, 254
  - rasch.copula2, 290
- \*Topic **ISOP model**
  - fit.isop, 121
  - isop, 154
  - isop.scoring, 158
  - isop.test, 160
- \*Topic **Isotone regression**

- monoreg.rowwise, 236
- \*Topic **Item fit**
  - pcm.fit, 251
- \*Topic **Joint maximum likelihood (JML)**
  - rasch.jml, 303
  - rasch.jml.biascorr, 306
  - rasch.jml.jackknife1, 307
- \*Topic **LSEM**
  - lsem.estimate, 193
  - lsem.permutationTest, 197
- \*Topic **Latent class model**
  - lc.2raters, 174
- \*Topic **Latent regression model**
  - latent.regression.em.raschtype, 161
- \*Topic **Least Squares Distance Method (LSDM)**
  - lsdm, 187
- \*Topic **Likelihood adjustment**
  - likelihood.adjustment, 176
- \*Topic **Linking**
  - equating.rasch, 113
  - equating.rasch.jackknife, 115
  - invariance.alignment, 141
  - linking.haberman, 179
  - linking.robust, 185
- \*Topic **Local dependence**
  - rasch.copula2, 290
  - rasch.pairwise.itemcluster, 340
  - rasch.pml2, 343
  - testlet.yen.q3, 394
  - yen.q3, 406
- \*Topic **Marginal maximum likelihood (MML)**
  - rasch.mml2, 321
- \*Topic **Markov Chain Monte Carlo (MCMC)**
  - mcmc.2pno, 203
  - mcmc.2pno.ml, 205
  - mcmc.2pnoh, 210
  - mcmc.3pno.testlet, 213
- \*Topic **Matrix utilities**
  - matrixfunctions.sirt, 201
- \*Topic **Minchi method**
  - rasch.pairwise, 338
- \*Topic **Model fit**
  - modelfit.sirt, 232
- \*Topic **Monotone regression**
  - monoreg.rowwise, 236
- \*Topic **Multidimensional item response model**
  - smirt, 375
- \*Topic **Multidimensional latent class Rasch model**
  - rasch.mirtlc, 309
- \*Topic **Multilevel DIF**
  - mcmc.2pno.ml, 205
- \*Topic **Multilevel item response model**
  - mcmc.2pno.ml, 205
- \*Topic **NOHARM**
  - noharm.sirt, 241
  - R2noharm, 279
  - R2noharm.EAP, 287
  - R2noharm.jackknife, 288
- \*Topic **Nedelsky model**
  - nedelsky-methods, 237
- \*Topic **Nonparametric IRT**
  - isop, 154
  - isop.scoring, 158
  - isop.test, 160
- \*Topic **Nonparametric item response theory**
  - np.dich, 247
  - plot.np.dich, 265
  - rasch.mml2, 321
- \*Topic **PROX algorithm**
  - rasch.prox, 351
- \*Topic **Pairwise conditional maximum likelihood (PCML)**
  - rasch.pairwise, 338
  - rasch.pairwise.itemcluster, 340
- \*Topic **Pairwise estimation**
  - rasch.evm.pcm, 298
- \*Topic **Pairwise marginal maximum likelihood (PMML)**
  - rasch.pml2, 343
- \*Topic **Partial credit model**
  - pcm.conversion, 250
- \*Topic **Person fit**
  - pcm.fit, 251
  - personfit.stat, 256
- \*Topic **Person parameter estimation**
  - person.parameter.rasch.copula, 254
  - wle.rasch, 402
- \*Topic **Person parameters**

- IRT.mle, 151
- mle.pcm.group, 230
- \*Topic **Plausible values**
  - latent.regression.em.raschtype, 161
  - plausible.value.imputation.raschtype, 260
- \*Topic **Polychoric correlation**
  - polychoric2, 266
- \*Topic **Probabilistic Guttman model**
  - prob.guttman, 268
- \*Topic **Proportional reduction of mean squared error (PRMSE)**
  - prmse.subscores.scales, 267
- \*Topic **Pseudo-likelihood estimation**
  - rasch.mml2, 321
- \*Topic **Q3**
  - testlet.yen.q3, 394
  - yen.q3, 406
- \*Topic **Quasi Monte Carlo integration**
  - qmc.nodes, 273
- \*Topic **R utilities**
  - automatic.recode, 13
  - data.wide2long, 99
- \*Topic **Ramsay's quotient model**
  - rasch.mml2, 321
- \*Topic **Rasch grade of membership model**
  - gom.em, 129
- \*Topic **Rater model**
  - lc.2raters, 174
  - rm.facets, 356
  - rm.sdt, 361
- \*Topic **Reliability**
  - greenyang.reliability, 139
  - marginal.truescore.reliability, 199
  - reliability.nonlinearSEM, 354
  - stratified.cronbach.alpha, 385
- \*Topic **Robust linking**
  - linking.robust, 185
- \*Topic **Signal detection model**
  - rm.sdt, 361
- \*Topic **Simulating IRT models**
  - sim.qm.ramsay, 369
  - sim.rasch.dep, 372
  - sim.raschtype, 374
- \*Topic **Statistical implicative analysis**
  - sia.sirt, 366
- \*Topic **TAM**
  - tam2mirt, 387
- \*Topic **Test for unidimensionality**
  - unidim.test.csn, 400
- \*Topic **Testlet model**
  - mcmc.3pno.testlet, 213
  - testlet.marginalized, 392
- \*Topic **Testlets**
  - mcmc.3pno.testlet, 213
  - testlet.yen.q3, 394
  - yen.q3, 406
- \*Topic **Tetrachoric correlation**
  - polychoric2, 266
  - tetrachoric2, 395
- \*Topic **True scores**
  - truescore.irt, 397
- \*Topic **Utilities**
  - categorize, 22
  - md.pattern.sirt, 221
- \*Topic **Variational approximation**
  - rasch.va, 353
- \*Topic **Weighted likelihood estimation (WLE)**
  - wle.rasch, 402
  - wle.rasch.jackknife, 404
- \*Topic **anova**
  - rasch.copula2, 290
- \*Topic **coda**
  - mcmc.list2coda, 220
- \*Topic **coef**
  - rasch.evm.pcm, 298
- \*Topic **datasets**
  - data.activity.itempars, 30
  - data.big5, 31
  - data.bs, 35
  - data.eid, 38
  - data.ess2005, 45
  - data.g308, 46
  - data.inv4gr, 48
  - data.liking.science, 49
  - data.long, 49
  - data.lsem, 54
  - data.math, 54
  - data.mcdonald, 55
  - data.mixed1, 59
  - data.ml, 60

- data.noharm, [61](#)
- data.parsl.rasch, [62](#)
- data.pirlsmissing, [63](#)
- data.pisaMath, [64](#)
- data.pisaPars, [65](#)
- data.pisaRead, [65](#)
- data.pw, [66](#)
- data.ratings, [67](#)
- data.raw1, [68](#)
- data.read, [68](#)
- data.reck, [88](#)
- data.sirt, [94](#)
- data.timss, [97](#)
- data.timss07.G8.RUS, [98](#)
- \*Topic **lavaan**
  - lavaan2mirt, [167](#)
- \*Topic **logLik**
  - rasch.copula2, [290](#)
- \*Topic **mirt**
  - lavaan2mirt, [167](#)
  - mirt.specify.partable, [222](#)
  - mirt.wrapper, [224](#)
  - tam2mirt, [387](#)
- \*Topic **package**
  - sirt-package, [4](#)
- \*Topic **plot**
  - isop, [154](#)
  - linking.robust, [185](#)
  - plot.mcmc.sirt, [264](#)
  - plot.np.dich, [265](#)
  - rasch.mml2, [321](#)
  - rm.sdt, [361](#)
- \*Topic **summary**
  - btm, [18](#)
  - fuzcluster, [123](#)
  - gom.em, [129](#)
  - invariance.alignment, [141](#)
  - isop, [154](#)
  - isop.test, [160](#)
  - latent.regression.em.raschtype, [161](#)
  - lc.2raters, [174](#)
  - linking.robust, [185](#)
  - lsdm, [187](#)
  - noharm.sirt, [241](#)
  - prob.guttman, [268](#)
  - R2conquest, [274](#)
  - R2noharm, [279](#)
  - R2noharm.jackknife, [288](#)
  - rasch.copula2, [290](#)
  - rasch.evm.pcm, [298](#)
  - rasch.jml, [303](#)
  - rasch.mirtlc, [309](#)
  - rasch.mml2, [321](#)
  - rasch.pairwise, [338](#)
  - rasch.pml2, [343](#)
  - rm.facets, [356](#)
  - rm.sdt, [361](#)
  - smirt, [375](#)
  - summary.mcmc.sirt, [386](#)
- \*Topic **vcov**
  - rasch.evm.pcm, [298](#)
- anova.gom (gom.em), [129](#)
- anova.prob.guttman (prob.guttman), [268](#)
- anova.rasch.copula2 (rasch.copula2), [290](#)
- anova.rasch.copula3 (rasch.copula2), [290](#)
- anova.rasch.mirtlc (rasch.mirtlc), [309](#)
- anova.rasch.mml (rasch.mml2), [321](#)
- anova.rm.facets (rm.facets), [356](#)
- anova.rm.sdt (rm.sdt), [361](#)
- anova.smirt (smirt), [375](#)
- automatic.recode, [13](#)
- brm-Methods, [14](#)
- brm.irf, [225](#)
- brm.irf (brm-Methods), [14](#)
- brm.sim (brm-Methods), [14](#)
- btm, [18](#)
- categorize, [22](#)
- ccov.np, [24](#), [27](#), [101](#)
- CDM: :gdm, [313](#)
- CDM: :IRT.irfprob, [226](#)
- CDM: :IRT.likelihood, [226](#)
- CDM: :IRT.posterior, [226](#)
- CDM: :itemfit.sx2, [328](#)
- CDM: :modelfit.cor, [233](#)
- class.accuracy.rasch, [25](#)
- coef.rasch.evm.pcm (rasch.evm.pcm), [298](#)
- colCumsums.sirt (matrixfunctions.sirt), [201](#)
- conf.detect, [7](#), [24](#), [26](#), [102](#), [117](#)
- data.activity.itempars, [30](#)
- data.big5, [31](#), [225](#)
- data.bs, [35](#)

- data.bs07a (data.bs), 35
- data.dcm, 225
- data.eid, 38
- data.ess2005, 45
- data.g308, 46
- data.inv4gr, 48
- data.liking.science, 49
- data.long, 49, 225
- data.lsem, 54
- data.lsem01 (data.lsem), 54
- data.math, 54
- data.mcdonald, 55
- data.mixed1, 59
- data.ml, 60
- data.ml1 (data.ml), 60
- data.ml2 (data.ml), 60
- data.noharm, 61
- data.noharm18 (data.noharm), 61
- data.noharmExC (data.noharm), 61
- data.pars1.2p1 (data.pars1.rasch), 62
- data.pars1.rasch, 62
- data.pirlsmissing, 63
- data.pisaMath, 64
- data.pisaPars, 65
- data.pisaRead, 65
- data.pw, 66
- data.pw01 (data.pw), 66
- data.ratings, 67
- data.ratings1 (data.ratings), 67
- data.ratings2 (data.ratings), 67
- data.ratings3 (data.ratings), 67
- data.raw1, 68
- data.read, 68, 225
- data.reck, 88
- data.reck21 (data.reck), 88
- data.reck61DAT1 (data.reck), 88
- data.reck61DAT2 (data.reck), 88
- data.reck73C1a (data.reck), 88
- data.reck73C1b (data.reck), 88
- data.reck75C2 (data.reck), 88
- data.reck78ExA (data.reck), 88
- data.reck79ExB (data.reck), 88
- data.si01 (data.sirt), 94
- data.si02 (data.sirt), 94
- data.si03 (data.sirt), 94
- data.si04 (data.sirt), 94
- data.si05 (data.sirt), 94
- data.si06 (data.sirt), 94
- data.sirt, 94
- data.timss, 97
- data.timss07.G8.RUS, 98
- data.wide2long, 99
- decategorize (categorize), 22
- detect.index, 101
- dif.logistic.regression, 102, 105, 106
- dif.strata.variance, 103, 104
- dif.variance, 103, 105
- dirichlet.mle, 106
- dirichlet.simul, 107, 109
- eigenvalues.manymatrices, 110
- eigenvalues.sirt, 112
- equating.rasch, 7, 113, 115, 180, 186
- equating.rasch.jackknife, 114, 115
- eRm: :personfit, 257
- expl.detect, 24, 116
- f1d.irt, 7, 117, 140
- fit.adisop, 154, 156, 157
- fit.adisop (fit.isop), 121
- fit.isop, 121, 156, 157, 159
- fuzcluster, 123
- fuzdiscr, 124, 126
- genlogis.moments (pgenlogis), 258
- gom.em, 6, 129, 225, 232, 233
- gom.jml, 133, 137
- greenyang.reliability, 7, 119, 139, 200, 355
- invariance.alignment, 7, 31, 114, 141, 180
- IRT.factor.scores.rm.facets (rm.facets), 356
- IRT.factor.scores.rm.sdt (rm.sdt), 361
- IRT.irfprob.gom (gom.em), 129
- IRT.irfprob.prob.guttman (prob.guttman), 268
- IRT.irfprob.rasch.mirtlc (rasch.mirtlc), 309
- IRT.irfprob.rasch.mml (rasch.mml2), 321
- IRT.irfprob.rm.facets (rm.facets), 356
- IRT.irfprob.rm.sdt (rm.sdt), 361
- IRT.irfprob.SingleGroupClass (mirt.wrapper), 224
- IRT.irfprob.smirt (smirt), 375
- IRT.likelihood, 178
- IRT.likelihood.gom (gom.em), 129

- IRT.likelihood.prob.guttman  
(prob.guttman), 268
- IRT.likelihood.rasch.copula2  
(rasch.copula2), 290
- IRT.likelihood.rasch.copula3  
(rasch.copula2), 290
- IRT.likelihood.rasch.mirtlc  
(rasch.mirtlc), 309
- IRT.likelihood.rasch.mml (rasch.mml2),  
321
- IRT.likelihood.rm.facets (rm.facets),  
356
- IRT.likelihood.rm.sdt (rm.sdt), 361
- IRT.likelihood.SingleGroupClass  
(mirt.wrapper), 224
- IRT.likelihood.smirt (smirt), 375
- IRT.mle, 151
- IRT.modelfit.gom (gom.em), 129
- IRT.modelfit.rasch.mirtlc  
(rasch.mirtlc), 309
- IRT.modelfit.rasch.mml (rasch.mml2), 321
- IRT.modelfit.rm.facets (rm.facets), 356
- IRT.modelfit.rm.sdt (rm.sdt), 361
- IRT.modelfit.smirt (smirt), 375
- IRT.posterior.gom (gom.em), 129
- IRT.posterior.prob.guttman  
(prob.guttman), 268
- IRT.posterior.rasch.copula2  
(rasch.copula2), 290
- IRT.posterior.rasch.copula3  
(rasch.copula2), 290
- IRT.posterior.rasch.mirtlc  
(rasch.mirtlc), 309
- IRT.posterior.rasch.mml (rasch.mml2),  
321
- IRT.posterior.rm.facets (rm.facets), 356
- IRT.posterior.rm.sdt (rm.sdt), 361
- IRT.posterior.SingleGroupClass  
(mirt.wrapper), 224
- IRT.posterior.smirt (smirt), 375
- isop, 153
- isop.dich, 7, 121, 122, 159, 161, 236
- isop.poly, 7, 161
- isop.scoring, 7, 155, 157, 158
- isop.test, 157, 160
- item.fit, 252
- itemfit, 252
- itemfit.sx2, 380
- latent.regression.em.normal  
(latent.regression.em.raschtype),  
161
- latent.regression.em.raschtype, 161,  
261
- lavaan2mirt, 167, 226, 387
- lavaan::lavaanify, 168
- lavaan::sem, 195
- lc.2raters, 6, 174
- likelihood.adjustment, 176
- linking.haberman, 7, 62, 114, 143, 179, 186
- linking.robust, 7, 114, 185
- logLik.gom (gom.em), 129
- logLik.prob.guttman (prob.guttman), 268
- logLik.rasch.copula2 (rasch.copula2),  
290
- logLik.rasch.copula3 (rasch.copula2),  
290
- logLik.rasch.mirtlc (rasch.mirtlc), 309
- logLik.rasch.mml (rasch.mml2), 321
- logLik.rm.facets (rm.facets), 356
- logLik.rm.sdt (rm.sdt), 361
- logLik.smirt (smirt), 375
- lsdm, 7, 187
- lsem.estimate, 7, 193, 197, 198
- lsem.MGM.stepfunctions (lsem.estimate),  
193
- lsem.permutationTest, 194, 196, 197
- ltm::person.fit, 257
- marginal.truescore.reliability, 7, 199
- matrixfunctions.sirt, 201
- mcmc.2pno, 7, 203, 212, 264, 265, 386
- mcmc.2pno.ml, 6, 7, 60, 143, 205, 264, 265,  
386
- mcmc.2pnoh, 7, 204, 210, 264, 265, 386
- mcmc.3pno.testlet, 7, 208, 213, 264, 265,  
372, 386, 407
- mcmc.list.descriptives, 218
- mcmlist2coda, 218, 220
- md.pattern.sirt, 221
- mirt, 168, 387
- mirt.specify.partable, 222
- mirt.wrapper, 168, 224, 387
- mirt::bfactor, 226
- mirt::mirt, 167, 168, 226, 232, 233
- mirt::mod2values, 226
- mirt::multipleGroup, 226
- mirt::personfit, 257

- mle.pcm.group, 230, 299
- modelfit.cor, 232
- modelfit.cor.poly (modelfit.sirt), 232
- modelfit.sirt, 232, 244, 313, 328, 346
- monoreg.colwise (monoreg.rowwise), 236
- monoreg.rowwise, 236
  
- nedelsky-methods, 237
- nedelsky.irf, 225
- nedelsky.irf (nedelsky-methods), 237
- nedelsky.latresp (nedelsky-methods), 237
- nedelsky.sim (nedelsky-methods), 237
- noharm.sirt, 7, 232, 233, 241, 282, 288
- np.dich, 6, 247, 265
  
- pbivnorm2, 249, 346
- pcm.conversion, 250, 252, 358
- pcm.fit, 251, 257
- person.fit, 252
- person.parameter.rasch.copula, 254, 293
- personfit.stat, 7, 256
- pgenlogis, 258, 325, 375
- plausible.value.imputation.raschtype, 164, 260
- plot.invariance.alignment (invariance.alignment), 141
- plot.isop (isop), 154
- plot.linking.robust (linking.robust), 185
- plot.lsem (lsem.estimate), 193
- plot.lsem.permutationTest (lsem.permutationTest), 197
- plot.mcmc.sirt, 204, 208, 212, 215, 264
- plot.np.dich, 265
- plot.rasch.mml (rasch.mml2), 321
- plot.rm.sdt (rm.sdt), 361
- polychoric2, 266, 396
- prmse.subscores.scales, 267
- prob.guttman, 7, 268
  
- qmc.nodes, 273, 377
  
- R2conquest, 274
- R2noharm, 7, 61, 232, 233, 244, 279, 288, 289
- R2noharm.EAP, 244, 282, 287
- R2noharm.jackknife, 282, 288
- rasch.conquest (R2conquest), 274
- rasch.copula2, 162, 225, 254, 260, 290, 341, 372, 407
- rasch.copula3, 6
- rasch.copula3 (rasch.copula2), 290
- rasch.evm.pcm, 7, 103, 298
- rasch.jml, 6, 303, 306, 308, 403
- rasch.jml.biascorr, 6, 306, 308
- rasch.jml.jackknife1, 6, 305, 307, 307
- rasch.mirtlc, 6, 225, 232, 233, 309, 380
- rasch.mml2, 6, 162, 199, 204, 232, 233, 260, 293, 305, 321, 341, 369, 374, 375, 380, 406
- rasch.pairwise, 6, 338, 341, 346, 372
- rasch.pairwise.itemcluster, 6, 339, 340, 346, 372, 407
- rasch.pml2, 232, 233, 343
- rasch.pml3, 7, 232, 233, 339, 341, 407
- rasch.pml3 (rasch.pml2), 343
- rasch.prox, 304, 305, 351
- rasch.va, 7, 353
- read.multidimpv (R2conquest), 274
- read.pimap (R2conquest), 274
- read.pv (R2conquest), 274
- read.show (R2conquest), 274
- reliability.nonlinearSEM, 140, 354
- rm.facets, 6, 176, 356, 364
- rm.sdt, 6, 176, 359, 361
- rowCumsums.sirt (matrixfunctions.sirt), 201
- rowIntervalIndex.sirt (matrixfunctions.sirt), 201
- rowKSmallest.sirt (matrixfunctions.sirt), 201
- rowKSmallest2.sirt (matrixfunctions.sirt), 201
- rowMaxs.sirt (matrixfunctions.sirt), 201
- rowMins.sirt (matrixfunctions.sirt), 201
  
- sem, 194
- semPlot::semPaths, 242
- sia.sirt, 366
- sim.qm.ramsay, 328, 369, 372
- sim.rasch.dep, 293, 328, 341, 346, 372
- sim.raschtype, 328, 369, 372, 374
- sirt (sirt-package), 4
- sirt-package, 4
- smirt, 6, 204, 232, 233, 375
- stratified.cronbach.alpha, 385
- summary.btm (btm), 18
- summary.fuzcluster (fuzcluster), 123
- summary.gom, 138

- summary.gom (gom.em), 129
- summary.invariance.alignment  
(invariance.alignment), 141
- summary.IRT.modelfit.gom (gom.em), 129
- summary.IRT.modelfit.rasch.mirtlc  
(rasch.mirtlc), 309
- summary.IRT.modelfit.rasch.mml  
(rasch.mm12), 321
- summary.IRT.modelfit.rm.facets  
(rm.facets), 356
- summary.IRT.modelfit.rm.sdt (rm.sdt),  
361
- summary.IRT.modelfit.smirt (smirt), 375
- summary.isop (isop), 154
- summary.isop.test (isop.test), 160
- summary.latent.regression  
(latent.regression.em.raschtype),  
161
- summary.lc.2raters (lc.2raters), 174
- summary.linking.robust  
(linking.robust), 185
- summary.lsdm, 190
- summary.lsdm (lsdm), 187
- summary.lsem (lsem.estimate), 193
- summary.lsem.permutationTest  
(lsem.permutationTest), 197
- summary.mcmc.sirt, 204, 208, 212, 215, 386
- summary.noharm.sirt (noharm.sirt), 241
- summary.prob.guttman (prob.guttman), 268
- summary.R2conquest (R2conquest), 274
- summary.R2noharm (R2noharm), 279
- summary.R2noharm.jackknife  
(R2noharm.jackknife), 288
- summary.rasch.conquest (R2conquest), 274
- summary.rasch.copula2, 293
- summary.rasch.copula2 (rasch.copula2),  
290
- summary.rasch.copula3 (rasch.copula2),  
290
- summary.rasch.evm.pcm (rasch.evm.pcm),  
298
- summary.rasch.jml, 305
- summary.rasch.jml (rasch.jml), 303
- summary.rasch.mirtlc (rasch.mirtlc), 309
- summary.rasch.mml (rasch.mm12), 321
- summary.rasch.pairwise, 339, 341
- summary.rasch.pairwise  
(rasch.pairwise), 338
- summary.rasch.pml, 346
- summary.rasch.pml (rasch.pml2), 343
- summary.rm.facets (rm.facets), 356
- summary.rm.sdt (rm.sdt), 361
- summary.smirt (smirt), 375
- tam.fa, 119
- tam.fit, 252
- tam2mirt, 168, 226, 387
- TAM: :lavaanify.IRT, 167, 168
- TAM: :tam.fa, 232, 233, 392, 393
- TAM: :tam.latreg, 178
- TAM: :tam.mml, 232, 233, 387
- TAM: :tam.mml.2pl, 232, 233
- TAM: :tam.mml.3pl, 15
- TAM: :tam.modelfit, 233
- testlet.marginalized, 392
- testlet.yen.q3, 394, 407
- tetrachoric2, 119, 266, 395
- truescore.irt, 397
- unidim.test.csn, 400
- vcov.rasch.evm.pcm (rasch.evm.pcm), 298
- wle.rasch, 248, 402, 405, 406
- wle.rasch.jackknife, 7, 403, 404
- yen.q3, 394, 395, 406