Package ‘PP’

May 13, 2017

Type Package
Title Person Parameter Estimation
Version 0.6.1
Date 2017-05-11
Description The PP package includes estimation of (MLE, WLE, MAP, EAP, ROBUST) person parameters for the 1,2,3,4-PL model and the GPCM (generalized partial credit model). The parameters are estimated under the assumption that the item parameters are known and fixed. The package is useful e.g. in the case that items from an item pool / item bank with known item parameters are administered to a new population of test-takers and an ability estimation for every test-taker is needed.

URL https://github.com/manuelreif/PP
License GPL-3
LazyLoad yes
LazyData true
VignetteBuilder knitr
Depends R (>= 3.0)
Suggests testthat, roxygen2, knitr, eRm, data.table, prettydoc
Imports Rcpp (>= 0.11.2)
LinkingTo Rcpp
RoxygenNote 6.0.1
NeedsCompilation yes
Author Reif Manuel [cre, aut],
Steinfeld Jan [aut]
Maintainer Reif Manuel <manuel.reif@gmail.com>
Repository CRAN
Date/Publication 2017-05-12 22:26:50 UTC
findmodel

Create a model-type vector template

Description

This is a small helper function which creates a vector template quick and easily for the PPall() function. Modify this template as you like.

Usage

findmodel(thres)

Arguments

thres A numeric matrix which contains the threshold parameter for each item. NA is allowed - in fact expected!

Details

This function tries to guess the model which was applied to each item by using the matrix of threshold parameters. It only discriminates between GPCM and 4-PL model, and returns a character vector of length equal to the number of items, that contains "GPCM" or "4PL" entries depending on the structure of the thres matrix.

Author(s)

Manuel Reif

See Also

PPall
Examples

# some threshold parameters
THRES <- matrix(c(-2,-1.23,1.11,3.48,1
 ,2,-1,-0.2,0.5,1.3,-0.8,1.5),nrow=2)

# slopes
sl <- c(0.5,1,1.5,1.1,1,0.98)

THRESx <- THRES
THRESx[2,1:3] <- NA

# for the 4PL item the estimated parameters are submitted,
# for the GPCM items the lower asymptote = 0
# and the upper asymptote = 1.
la <- c(0.02,0.1,0,0,0)
uu <- c(0.97,0.91,1,1,1,1)

awmatrix <- matrix(c(1,0,1,0,1,1,1,0,1
 ,2,0,0,0,0,0,0,0,0,1
 ,1,2,2,1,1,1,0,0,1),byrow=TRUE,nrow=5)

# create model2est
# this function tries to help finding the appropriate
# model by inspecting the THRESx.
model2est <- findmodel(THRESx)

# MLE
respmixed_mle <- PPall(respm=awmatrix,thres=THRESx,
 slopes=sl,lowerA=la, upperA=uu,type = “mle”,
 model2est=model2est)

# WLE
respmixed_wle <- PPall(respm=awmatrix,thres=THRESx,
 slopes=sl,lowerA=la, upperA=uu,type = “wle”,
 model2est=model2est)

# MAP estimation
respmixed_map <- PPall(respm=awmatrix,thres=THRESx,
 slopes=sl,lowerA=la, upperA=uu,type = “map”,
 model2est=model2est)

# EAP estimation
respmixed_eap <- PPall(respm=awmatrix,thres=THRESx,
 slopes=sl,lowerA=la, upperA=uu,type = “eap”,
 model2est=model2est)

# Robust estimation
respmixed_rob <- PPall(respm=awmatrix,thres=THRESx,
 slopes=sl,lowerA=la, upperA=uu,type = “robust”,
 model2est=model2est)
JKpp

# summary to summarize the results
summary(respmixed_mle)
summary(respmixed_wle)
summary(respmixed_map)
summary(respmixed_eap)
summary(respmixed_rob)

fourpl_df  Simulated data set

Description
This dataset contains data simulated with the `sim_4pl()` function.

Format
A data.frame with 60 rows and 14 columns.

Source
simulation

See Also
sim_4pl

JKpp  Run a jackknife

Description
This function uses a jackknife approach to compute person parameters. The jackknife ability measure is based on primarily estimated models (`PP_4pl()`, `PP_gpcm()` or `PPall()`) - so the function is applied on the estimation objects, and jackknifed ability measures are returned.

Usage

JKpp(estobj, ...)

## S3 method for class 'fourpl'
JKpp(estobj, cmeth = "mean", maxsteps = 500,
     exac = 0.001, fullmat = FALSE, ctrl = list(), ...)

## S3 method for class 'gpcm'
JKpp(estobj, cmeth = "mean", maxsteps = 500, exac = 0.001,
Arguments

estobj An object which originates from using PP_gpcm(), PP_4pl() or PPall().
... More input.
cmeth Choose the centering method, to summarize the n jackknife results to one single
ability estimate. There are three valid entries: "mean", "median" and "AMT"
(see Details for further description).
maxsteps The maximum number of steps the NR Algorithm will take.
exac How accurate are the estimates supposed to be? Default is 0.001.
fullmat Default = FALSE. If TRUE, the function returns the whole jackknife matrix,
which is the basis for the jackknife estimator.
ctrl More controls
x An object of class jk which is the result of using the JKpp() function
object An object of class jk which is the result of using the JKpp() function
nrowmax When printing the matrix of estimates - how many rows should be shown? De-
default = 15.

Details

Please use the Jackknife Standard-Error output with caution! It is implemented as suggested in
Wainer and Wright (1980), but the results seem a bit strange, because the JK-SE is supposed to
overestimate the SE compared to the MLE-SE. Actually, in all examples an underestimation of the
SE was observed compared to the MLE/WLE-SE!

AMT-robustified jackknife: When choosing cmeth = AMT, the jackknife ability subsample es-
timates and the original supplied ability estimate are combined to a single jackknife-ability value
by the Sine M-estimator. The AMT (or Sine M-estimator) is one of the winners in the Princeton
Robustness Study of 1972. To get a better idea how the estimation process works, take a closer look
to the paper which is mentioned below (Wainer & Wright, 1980).

Author(s)

Manuel Reif
References


See Also

PP_gpcm, PP_4pl, PPall

Examples

############################################################### Jackknife###############################################################

### 4 PL model ######

### data creation ############

set.seed(1623)
# intercepts
diffpar <- seq(-3,3,length=12)
# slope parameters
sl <- round(runif(12,0.5,1.5),2)
la <- round(runif(12,0,0.25),2)
ua <- round(runif(12,0.8,1),2)

# response matrix
abpar <- rnorm(10,0,0.7)
awm <- sim_4pl(beta = diffpar, alpha = sl,lowerA = la,upperA = ua,theta = abpar)

### 1PL model ######

# MLE
res1plmle <- PP_4pl(respm = awm,thres = diffpar,type = "mle")
# WLE
res1plwle <- PP_4pl(respm = awm,thres = diffpar,type = "wle")
# MAP estimation
res1plmap <- PP_4pl(respm = awm,thres = diffpar,type = "map")
# EAP estimation
res1pleap <- PP_4pl(respm = awm,thres = diffpar,type = "eap")
# robust estimation
res1plrob <- PP_4pl(respm = awm,thres = diffpar,type = "robust")

### centering method = mean
res_jk1 <- JKpp(res1plmle)
res_jk2 <- JKpp(res1plwle)
res_jk3 <- JKpp(res1plmap)
res_jk4 <- JKpp(res1plrob)
res_jk5 <- JKpp(res1pleap)

summary(res_jk1)

### centering method = median
res_jk1a <- JKpp(res1plmle,cmeth = "median")
res_jk2a <- JKpp(res1plwle,cmeth = "median")
res_jk3a <- JKpp(res1plmap,cmeth = "median")

summary(res_jk2a)

## centering method = AMT
res_jk1b <- JKpp(res1plmle,cmeth = "AMT")
res_jk2b <- JKpp(res1plwle,cmeth = "AMT")
res_jk3b <- JKpp(res1plmap,cmeth = "AMT")

summary(res_jk3b)

## 2PL model ######
# MLE
res2plmle <- PP_4pl(respm = awm,thres = diffpar, slopes = sl,type = "mle")
# WLE
res2plwle <- PP_4pl(respm = awm,thres = diffpar, slopes = sl,type = "wle")
# MAP estimation
res2plmap <- PP_4pl(respm = awm,thres = diffpar, slopes = sl,type = "map")
# EAP estimation
res2pleap <- PP_4pl(respm = awm,thres = diffpar,slopes = sl,type = "eap")
# robust estimation
res2plrob <- PP_4pl(respm = awm,thres = diffpar,slopes = sl,type = "robust")

res_jk6 <- JKpp(res2plmle)
res_jk7 <- JKpp(res2plwle)
res_jk8 <- JKpp(res2plmap)
res_jk9 <- JKpp(res2pleap)
res_jk10 <- JKpp(res2plrob)

### GPCM model #######
# some threshold parameters
THRES <- matrix(c(-2,-1.23,1.11,3.48,1
 ,2,-1,-0.2,0.5,1.3,-0.8,1.5),nrow=2)
# slopes
sl <- c(0.5,1.5,1,1,0.98)
awmatrix <- matrix(c(1,0,2,0,1,1,0,0,1,2,0,0,0,0,0,0,0,1,1,2,2,1,1,1,0,0,1),byrow=TRUE,nrow=5)

### PCM model #######
# MLE
respcmnlmle <- PP_gpcm(respm = awmatrix,thres = THRES,
                        slopes = rep(1,ncol(THRES)),type = "mle")
# WLE
respcmwlle <- PP_gpcm(respm = awmatrix,thres = THRES,
                        slopes = rep(1,ncol(THRES)),type = "wle")
# MAP estimation
respcmmap <- PP_gpcm(respm = awmatrix,thres = THRES,
slopes = rep(1, ncol(THRES)), type = "map")

res_jk11 <- JKpp(respcmmlme)
res_jk12 <- JKpp(respcmwle)
res_jk13 <- JKpp(respcmmap)

### GPCM/4-PL mixed model ######

THRES <- matrix(c(-2,-1.23,-1.11,3.48,1,
                   2,-1,-0.2,0.5,1.3,-0.8,1.5), nrow=2)

sl <- c(0.5,1,1.5,1,1,1,0.98)

THRESx <- THRES
THRESx[2,1:3] <- NA

# for the 4PL item the estimated parameters are submitted,
# for the GPCM items the lower asymptote = 0
# and the upper asymptote = 1.
la <- c(0.02,0.1,0,0,0,0)
u <- c(0.97,0.91,1,1,1,1)

awmatrix <- matrix(c(1,0,1,0,1,1,1,0,0,1,
                      2,0,0,1,0,0,0,0,1,
                      0,2,2,1,1,1,0,0,1), byrow=TRUE, nrow=5)

# create model2est
# this function tries to help finding the appropriate
# model by inspecting the THRESx.
model2est <- findmodel(THRESx)

# MLE estimation
respmixed_mle <- PPall(respm = awmatrix,
                       thres = THRESx,
                       slopes = sl,
                       lowerA = la,
                       upperA = u,
                       type = "mle",
                       model2est = model2est)

# WLE estimation
respmixed_wle <- PPall(respm = awmatrix,
                       thres = THRESx,
                       slopes = sl,
                       lowerA = la,
                       upperA = u,
                       type = "wle",
                       model2est = model2est)

res_jk114 <- JKpp(respmixed_mle)
res_jk115 <- JKpp(respmixed_wle)
Description

Compute several person fit statistic for the 1-PL, 2-PL, 3-PL, 4-PL and PCM.

Usage

\[
Pfit(respm, pp, fitindices, SE = \text{FALSE})
\]

```r
## S3 method for class 'gpcm'
Pfit(respm, pp, fitindices = c("infit", "outfit"),
     SE = \text{FALSE})
```

Arguments

- `respm`: numeric response matrix
- `pp`: object of the class fourpl with estimated person parameter
- `fitindices`: character vector of desired person fit statistics `c("lz", "lzstar", "infit", "outfit")`
- `SE`: logical: if true standard errors are computed using jackknife method

Details

Please note that currently only the likelihood based LZ-Index (Drasgow, Levine, and Williams, 1985) and LZ*-Index (Snijders, 2001) are implemented. Also the INFIT-OUTFIT (Wright and Masters, 1982, 1990) statistic as well as the polytomous version of INFIT-OUTFIT are supported. Other person fit statistics will be added soon.

The calculation of the person fit statistics requires the numeric response-matrix as well as an object of the fourpl-class. So first you should estimate the person parameter and afterwards calculate the person fit statistics. You could also use our PPass-function to estimate the person parameter and calculate the desired person fit simultaneously. It is possible to calculate several person fit statistics at once, you only have to specify them in a vector.

For the Partial Credit model we currently support the infit-outfit statistic. Please submit also the numeric response-matrix as well as the estimated person parameter with an gpcm-class.

Value

list of person-fits for each person-fit statistic
- the list of person-fits contains the calculated person-fit (like \text{Iz}, \text{Izstar}) and also additional information like p-value or standard error if desired.
- the additional information is provided after the short form of the personfit
- \text{Iz} (Iz)
- \text{Izstar} (Izs)
• infit the mean-square statistic (in)
• outfit the mean-square statistic (ou)
• _unst: unstandardised
• _se: standard error
• _t: t-value
• _chisq: $\chi^2$-value
• _df: degrees of freedom
• _pv: p-value

Author(s)
Jan Steinfeld

References
Mousavi, A. & Cui, Y. Evaluate the performance of and of person fit: A simulation study.
See Also

PPall, PP_4pl, PPass

Examples

Pfit

### data creation

```r
set.seed(1337)

# intercepts
diffpar <- seq(-3, 3, length = 15)

# slope parameters
sl <- round(rnorm(15, 0.5, 1.5), 2)
la <- round(rnorm(15, 0.0, 0.25), 2)
uu <- round(rnorm(15, 0.8, 1), 2)

# response matrix
awm <- matrix(sample(0:1, 100 * 15, replace = TRUE), ncol = 15)
```

# 1PL model
```
res1plmle <- PP_4pl(respm = awm, thres = diffpar, type = "mle")
res1plwle <- PP_4pl(respm = awm, thres = diffpar, type = "wle")
res1plmap <- PP_4pl(respm = awm, thres = diffpar, type = "map")
```

# LZx-Index
```
Pfit(respm = awm, pp = res1plwle, fitindices = "lzstar")
Pfit(respm = awm, pp = res1plmle, fitindices = "lzstar")
Pfit(respm = awm, pp = res1plmap, fitindices = "lzstar")
```

# 2PL model
```
res2plmle <- PP_4pl(respm = awm, thres = diffpar, slopes = sl, type = "mle")
res2plwle <- PP_4pl(respm = awm, thres = diffpar, slopes = sl, type = "wle")
```
Estimation of Person Parameters and calculation of Person Fit for the 1,2,3,4-PL model and the GPCM.

Pfit(respm=awm,pp=res2plwle,fitindices="lzstar")
Pfit(respm=awm,pp=res2plmle,fitindices="lzstar")
## LZ* - Index combined with Infit-Outfit ####
Pfit(respm=awm,pp=res2plwle,fitindices=c("lzstar","infit","outfit"))
# ---------------------------------------------------------------

# 3PL model #####
# -------------------------------
# MLE
res3plmle <- PP_4pl(respm = awm,thres = diffpar,
slopes = sl,lowerA = 1a,type = "mle")
# WLE
res3plwle <- PP_4pl(respm = awm,thres = diffpar,
slopes = sl,lowerA = 1a,type = "wle")
## LZ* - Index #####
Pfit(respm=awm,pp=res3plwle,fitindices="lzstar")
Pfit(respm=awm,pp=res3plmle,fitindices="lzstar")
## LZ* - Index combined with Infit-Outfit ####
Pfit(respm=awm,pp=res3plwle,fitindices=c("lzstar","infit","outfit"))
# ---------------------------------------------------------------

# 4PL model #####
# -----------------------------------------------
# MLE
res4plmle <- PP_4pl(respm = awm,thres = diffpar,
slopes = sl,lowerA = 1a,upperA=ua,type = "mle")
# WLE
res4plwle <- PP_4pl(respm = awm,thres = diffpar,
slopes = sl,lowerA = 1a,upperA=ua,type = "wle")
## LZ* - Index #####
Pfit(respm=awm,pp=res4plwle,fitindices="lzstar")
Pfit(respm=awm,pp=res4plmle,fitindices="lzstar")
## LZ* - Index combined with Infit-Outfit ####
Pfit(respm=awm,pp=res4plwle,fitindices=c("lzstar","infit","outfit"))
# ---------------------------------------------------------------
Description

PP-package has been developed to easily compute ML, WL (Warm 1989), MAP, EAP and robust estimates of person parameters for a given response matrix and given item parameters of the 1,2,3,4-PL model (Birnbaum 1968, Barton & Lord 1981) and the GPCM (Muraki 1992). It provides c++ routines which makes estimation of parameters very fast. Additional some methods to calculate the person fit are provided like the infit-outfit-, lz- and lz*-idex. Read the vignettes for getting started with this package.

Author(s)

Manuel Reif and Jan Steinfeld

References


See Also

PPass, PP_gpcm, PP_4pl, PPall, Pfit

Examples

```r
set.seed(1522)
# intercepts
difflpar <- seq(-3,3,length=12)
# slope parameters
```
sl <- round(runif(12,0.5,1.5),2)
la <- round(runif(12,0,0.25),2)
ua <- round(runif(12,0.8,1),2)

# response matrix
awm <- matrix(sample(0:1,10*12,replace=TRUE),ncol=12)
# MLE estimation
res3plmle <- PP_4pl(respm = awm,thres = diffpar, slopes = sl,lowerA = la,type = "mle")
# WLE estimation
res3plwle <- PP_4pl(respm = awm,thres = diffpar, slopes = sl,lowerA = la,type = "wle")
# MAP estimation
res3plmap <- PP_4pl(respm = awm,thres = diffpar, slopes = sl,lowerA = la,type = "map")

# calculate person fit
res3plmlefit <- Pfit(respm=awm,pp=res3plmle,fitindices=c("infit","outfit"))

# or estimate person parameter and calculate person fit in one step
out <- PPass(respdf = data.frame(awm),thres = diffpar,items="all",mod=c("1PL"),fitindices= c("1z","1zstar","infit","outfit"))

---

PPall

**Estimate Person Parameters**

**Description**

Compute person parameters for the 1,2,3,4-PL model and for the GPCM. Choose between ML, WL, MAP, EAP and robust estimation. Use this function if 4-PL items and GPCM items are mixed for each person.

**Usage**

PPall(respm, thres, slopes, lowerA, upperA, theta_start = NULL, mu = NULL, sigma2 = NULL, type = "wle", model2est = 100, exac = 0.001, H = 1, ctrl = list())

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

### S3 method for class 'ppeo'
summary(object, nrowmax = 15, ...)

**Arguments**

- **respm**
  - An integer matrix, which contains the examinees responses. A persons x items matrix is expected.

- **thres**
  - A numeric matrix which contains the threshold parameter for each item. If the first row of the matrix is not set to zero (only zeroes in the first row) - then a row-vector with zeroes is added by default.
slopes A numeric vector, which contains the slope parameters for each item - one parameter per item is expected.

lowerA A numeric vector, which contains the lower asymptote parameters (kind of guessing parameter) for each item. In the case of polytomous items, the value must be 0.

upperA numeric vector, which contains the upper asymptote parameters for each item. In the case of polytomous items, the value must be 1.

theta_start A vector which contains a starting value for each person. Currently this is necessary to supply, but soon it will be set automatically if nothing is committed.

mu A numeric vector of location parameters for each person in case of MAP estimation. If nothing is submitted this is set to 0 for each person for MAP estimation.

sigma2 A numeric vector of variance parameters for each person in case of MAP or EAP estimation. If nothing is submitted this is set to 1 for each person for MAP estimation.

type Which maximization should be applied? There are five valid entries possible: "mle", "wle", "map", "eap" and "robust". To choose between the methods, or just to get a deeper understanding the papers mentioned below are quite helpful. The default is "wle" which is a good choice in many cases.

model2est A character vector with length equal to the number of submitted items. It defines itemwise the response model under which the item parameter was estimated. There are 2 valid inputs up to now: "GPCM" and "4PL".

maxsteps The maximum number of steps the NR algorithm will take. Default = 100.

exac How accurate are the estimates supposed to be? Default is 0.001.

H In case type = "robust" a Huber ability estimate is performed, and H modulates how fast the downweighting takes place (for more details read Schuster & Yuan 2011).

ctrl More controls:

- killdupli: Should duplicated response pattern be removed for estimation (estimation is faster)? This is especially reasonable in case of a large number of examinees and a small number of items. Use this option with caution (for map and eap), because persons with different mu and sigma2 will have different ability estimates despite they responded identically. Default value is FALSE.
- skipcheck: Default = FALSE. If TRUE data matrix and arguments are not checked - this saves time e.g. when you use this function for simulations.

x an object of class gpcm4pl which is the result of using the PPall() function

... just some points.

object An object of class gpcm4pl which is the result of using the PPall() function

nrowmax When printing the matrix of estimates - how many rows should be shown? Default = 15.
Details
For a test with both: dichotomous and polytomous items which have been scaled under 1/2/3/4-PL
model or the GPCM, use this function to estimate the person ability parameters. You have to define
the appropriate model for each item.

Please note, that robust estimation with (Huber ability estimate) polytomous items is still experi-
mental!

Value
The function returns a list with the estimation results and pretty much everything which has been
submitted to fit the model. The estimation results can be found in OBJ$resPP. The core result is a
number_of_persons x 2 matrix, which contains the ability estimate and the SE for each submitted
person.

Author(s)
Manuel Reif

References
Baker, Frank B., and Kim, Seock-Ho (2004). Item Response Theory - Parameter Estimation Tech-
niques. CRC-Press.
Response Model.
Psychological Measurement, 16, 159-176.
chological Measurement, 17, 351-363.
Samejima, Fumiko (1993). The bias function of the maximum likelihood estimate of ability for the 
dichotomous response level. Psychometrika, 58, 195-209.
Samejima, Fumiko (1993). An approximation of the bias function of the maximum likelihood esti-
mate of a latent variable for the general case where the item responses are discrete. Psychometrika, 
58, 119-138.
Journal of Educational and Behavioral Statistics, 36(6), 720-735.
Wang, S. and Wang, T. (2001). Precision of Warm’s Weighted Likelihood Estimates for a Polyto-
Psychometrika, 54, 427-450.
of the slip correction in the four parameter logistic models with computerized adaptive testing. 
Applied Psychological Measurement, 36, 75-87.
See Also

PPass, PP_gpcm, PP_4pl, JKpp, PV

Examples

```
# some threshold parameters
THRES <- matrix(c(-2,-1.23,1.11,3.48,1
                  ,2,-1,-0.2,0.5,1.3,-0.8,1.5),nrow=2)

# slopes
sl <- c(0.5,1.5,1.1,1,0.98)

THRESx <- THRES
THRESx[2,1:3] <- NA

# for the 4PL item the estimated parameters are submitted,
# for the GPCM items the lower asymptote = 0
# and the upper asymptote = 1.
la <- c(0.02,0.1,0,0,0)
uu <- c(0.97,0.91,1,1,1)

awmatrix <- matrix(c(1,0,1,0,1,1,1,0,0,1
                      ,2,0,0,0,0,0,0,0,1,0,1,1,1,1,0,0,0),byrow=TRUE,nrow=5)

# create model2est
# this function tries to help finding the appropriate
# model by inspecting the THRESx.
model2est <- findmodel(THRESx)

# MLE
respmixed_mle <- PPall(respm = awmatrix, thres = THRESx,
                        slopes = sl,lowerA = la, upperA=uu,type = "mle",
                        model2est=model2est)

# WLE
respmixed_wle <- PPall(respm = awmatrix, thres = THRESx,
                        slopes = sl,lowerA = la, upperA=uu,type = "wle",
                        model2est=model2est)

# MAP estimation
respmixed_map <- PPall(respm = awmatrix, thres = THRESx,
                        slopes = sl,lowerA = la, upperA=uu,type = "map",
                        model2est=model2est)

# EAP estimation
respmixed_eap <- PPall(respm = awmatrix, thres = THRESx,
                        slopes = sl,lowerA = la, upperA=uu,type = "eap",
                        model2est=model2est)

# Robust estimation
respmixed_rob <- PPall(respm = awmatrix, thres = THRESx,
```
slopes = sl, lowerA = la, upperA=ua, type = "robust",
model2est=model2est)

# summary to summarize the results
summary(respmixed_mle)
summary(respmixed_wle)
summary(respmixed_map)
summary(respmixed_eap)
summary(respmixed_rob)

---

**PPass**  
*Person Assessment function*

**Description**

Estimate Person Parameters and calculate Person Fit in one step to gain response pattern assessment. Submit a data.frame which contains item responses, or an fitted model (Rasch Model and Partial Credit Model are supported) of the *erm* package.

**Usage**

```r
PPass(...)
```

## Default S3 method:
```r
PPass(respdf, items = "all", mod = c("1PL", "2PL", "3PL",
"4PL", "PCM", "GPCM", "MIXED"), fitindices = c("lz", "lzstar", "infit",
"outfit"), ...)
```

## S3 method for class 'Rm'
```r
PPass(RMobj, fitindices = c("lz", "lzstar", "infit", "outfit"), ...
```

**Arguments**

- **respdf** A data.frame which contains the items, and perhaps other informations. Each row is a person related response pattern. Each column denotes a variable.
- **items** A numeric (integer) vector which indicates the positions of the items in the data.frame *(respdf)*. If `items = 'all'`, **all columns** are treated as items.
- **mod** Choose your data generating model. This argument switches between the three person parameter estimating functions PP_4p1, PP_gpcm and PPall.
- **fitindices** A character vector which denotes the fit indices to compute.
- **RMobj** A fitted Rasch Model *(RM())* object which stems from the *erm* package.
- **...** Submit arguments to the underlying functions: PP_4p1, PP_gpcm and PPall (see documentation files) for person parameter estimation.
**PPass**

**Details**

PPass fuses Person Parameter estimation and Person Fit computation into a single function.

**Value**

The original data.frame and

- The Person Parameter estimates incl. Standard Errors (2 columns)
- Person Fit Indices you chose (1 or more)

**Author(s)**

Manuel Reif, Jan Steinfeld

**See Also**

`PP_4pl, PP_gpcm, PPall, Pfit`

**Examples**

```r
library(eRm)

### real data ############

data(pp_amt)

d <- pp_amt$daten_amt

rd_res <- PPass(respdf = d,
    items = 8:ncol(d),
    mod="1PL",
    thres = pp_amt$betas[,2],
    fitindices = "lz")

head(rd_res)

### RM - eRm
my_data <- eRm::sim.rasch(200, 12)
my_rm <- eRm::RM(my_data)

res_pp1 <- PPass(my_rm)

### PCM - eRm
set.seed(2751)

THRES <- matrix(c(-2,-1.23,1.11,3.48,1,
    2,-1,-0.2,0.5,1.3,-0.8,1.5),nrow=2)

THRES <- rbind(THRES, c(-0.2,NA,NA,NA,NA))
```
sl <- rep(1,6)
THRESx <- rbind(0,THRES)
THETA <- rnorm(200)
simdat_gpcm <- sim_gpcm(thres = THRESx, alpha = sl, theta = THETA)

my_pcm <- eRm::PCM(simdat_gpcm)

res_pp2 <- PPass(my_pcm)

## = = = = = = = = = = = 1PL model

set.seed(1337)

# intercepts
diffpar <- seq(-3,3,length=15)
# slope parameters
sl <- round(runif(15,0.5,1.5),2)
la <- round(runif(15,0,0.25),2)
uu <- round(runif(15,0.8,1),2)

# response matrix
awm <- matrix(sample(0:1,100*15,replace=TRUE),ncol=15)
awm <- as.data.frame(awm)
# estimate person parameter

# estimate person parameter and person fit
out <- PPass(respdf = awm, thres = diffpar, items="all",
            mod=c("1PL"), fitindices= c("lz","lzstar","infit","outfit"))

# show first rows
head(out)

---

**PP_4pl**  
**Estimate Person Parameters for the 4-PL model**

**Description**

Compute Person Parameters for the 1/2/3/4-PL model and choose between five common estimation techniques: ML, WL, MAP, EAP and a robust estimation. All item parameters are treated as fixed.

**Usage**

```r
PP_4pl(respm, thres, slopes = NULL, lowerA = NULL, upperA = NULL,
        theta_start = NULL, mu = NULL, sigma2 = NULL, type = "wle",
        maxsteps = 40, exac = 0.001, H = 1, ctrl = list())
```
Arguments

- **respm**: An integer matrix, which contains the examinees responses. A persons x items matrix is expected.

- **thres**: A numeric vector or a numeric matrix which contains the threshold parameter (also known as difficulty parameter or beta parameter) for each item. If a matrix is submitted, the first row must contain only zeroes!

- **slopes**: A numeric vector, which contains the slope parameters for each item - one parameter per item is expected.

- **lowerA**: A numeric vector, which contains the lower asymptote parameters (kind of guessing parameter) for each item.

- **upperA**: A numeric vector, which contains the upper asymptote parameters for each item.

- **theta_start**: A vector which contains a starting value for each person. Currently this is necessary to supply, but soon it will be set automatically if nothing is committed.

- **mu**: A numeric vector of location parameters for each person in case of MAP or EAP estimation. If nothing is submitted this is set to 0 for each person for MAP estimation.

- **sigma2**: A numeric vector of variance parameters for each person in case of MAP or EAP estimation. If nothing is submitted this is set to 1 for each person for MAP estimation.

- **type**: Which maximization should be applied? There are five valid entries possible: "mle", "wle", "map", "eap" and "robust". To choose between the methods, or just to get a deeper understanding the papers mentioned below are quite helpful. The default is "wle" which is a good choice in many cases.

- **maxsteps**: The maximum number of steps the NR Algorithm will take. Default = 100.

- **exac**: How accurate are the estimates supposed to be? Default is 0.001.

- **H**: In case type = "robust" a Huber ability estimate is performed, and H modulates how fast the downweighting takes place (for more Details read Schuster & Yuan 2011).

- **ctrl**: more controls:
  - **killdupli**: Should duplicated response pattern be removed for estimation (estimation is faster)? This is especially reasonable in case of a large number of examinees and a small number of items. Use this option with caution (for map and eap), because persons with different mu and sigma2 will have different ability estimates despite they responded identically. Default value is FALSE.
  - **skipcheck**: Default = FALSE. If TRUE data matrix and arguments are not checked - this saves time e.g. when you use this function for simulations.

Details

With this function you can estimate:

- **1-PL model** (Rasch model) by submitting: the data matrix, item difficulties and **nothing else**, since the 1-PL model is merely a 4-PL model with: any slope = 1, any lower asymptote = 0 and any upper asymptote = 1!
• **2-PL model** by submitting: the data matrix, item difficulties and slope parameters. Lower and upper asymptotes are automatically set to 0 and 1 respectively.

• **3-PL model** by submitting anything except the upper asymptote parameters

• **4-PL model** → submit all parameters ...

The probability function of the 4-PL model is:

\[
P(x_{ij} = 1 | \hat{\alpha}_i, \hat{\beta}_i, \hat{\gamma}_i, \hat{\delta}_i, \theta_j) = \frac{\hat{\gamma}_i + (\hat{\delta}_i - \hat{\gamma}_i) \exp(\hat{\alpha}_i(\theta_j - \hat{\beta}_i))}{1 + \exp(\hat{\alpha}_i(\theta_j - \hat{\beta}_i))}
\]

In our case \(\theta\) is to be estimated, and the four item parameters are assumed as fixed (usually these are estimates of a former scaling procedure).

The 3-PL model is the same, except that \(\delta_i = 1\), \(\forall i\).

In the 2-PL model \(\delta_i = 1\), \(\gamma_i = 0\), \(\forall i\).

In the 1-PL model \(\delta_i = 1\), \(\gamma_i = 0\), \(\alpha_i = 1\), \(\forall i\).

The robust estimation method, applies a Huber-type estimator (Schuster & Yuan, 2011), which downweights responses to items which provide little information for the ability estimation. First a residuum is estimated and on this basis, the weight for each observation is computed.

residuum:

\[
r_i = \alpha_i(\theta - \beta_i)
\]

weight:

\[
w(r_i) = \begin{cases} 1 & \text{if } |r_i| \leq H \\ H/|r_i| & \text{if } |r_i| > H \end{cases}
\]

Value

The function returns a list with the estimation results and pretty much everything which has been submitted to fit the model. The estimation results can be found in OBJ$resPP. The core result is a number_of_persons x 2 matrix, which contains the ability estimate and the SE for each submitted person.

**Author(s)**

Manuel Reif

**References**


See Also

PPass, PPall, PP_gpcm, JKpp, PV

Examples

########################################################### 4 PL ###########################################################

### real data #######

data(pp_amt)

d <- as.matrix(pp_amt$daten_amt[,-(1:7)])

rd_res <- PP_4pl(respm = d, thres = pp_amt$betas[,2], type = "wle")
summary(rd_res)

### fake data #######

# smaller ... faster

set.seed(1522)
# intercepts
diffpar <- seq(-3,3,length=12)
# slope parameters
sl <- round(runif(12,0.5,1.5),2)
lv <- round(runif(12,0,0.25),2)
uv <- round(runif(12,0.8,1),2)
# response matrix
awm <- matrix(sample(0:1,10*12,replace=TRUE),ncol=12)
## 1PL model ######

`# MLE`
`res1plmle <- PP_4pl(respm = awm, thres = diffpar, type = "mle")`

`# WLE`
`res1plwle <- PP_4pl(respm = awm, thres = diffpar, type = "wle")`

`# MAP estimation`
`res1plmap <- PP_4pl(respm = awm, thres = diffpar, type = "map")`

`# EAP estimation`
`res1pleap <- PP_4pl(respm = awm, thres = diffpar, type = "eap")`

`# robust estimation`
`res1plrob <- PP_4pl(respm = awm, thres = diffpar, type = "robust")`

`# summarize results`
`summary(res1plmle)`
`summary(res1plwle)`
`summary(res1plmap)`

## 2PL model ######

`# MLE`
`res2plmle <- PP_4pl(respm = awm, thres = diffpar, slopes = sl, type = "mle")`

`# WLE`
`res2plwle <- PP_4pl(respm = awm, thres = diffpar, slopes = sl, type = "wle")`

`# MAP estimation`
`res2plmap <- PP_4pl(respm = awm, thres = diffpar, slopes = sl, type = "map")`

`# EAP estimation`
`res2pleap <- PP_4pl(respm = awm, thres = diffpar, slopes = sl, type = "eap")`

`# robust estimation`
`res2plrob <- PP_4pl(respm = awm, thres = diffpar, slopes = sl, type = "robust")`

## 3PL model ######

`# MLE`
`res3plmle <- PP_4pl(respm = awm, thres = diffpar, slopes = sl, lowerA = la, type = "mle")`

`# WLE`
`res3plwle <- PP_4pl(respm = awm, thres = diffpar, slopes = sl, lowerA = la, type = "wle")`

`# MAP estimation`
`res3plmap <- PP_4pl(respm = awm, thres = diffpar, slopes = sl, lowerA = la, type = "map")`

`# EAP estimation`
`res3pleap <- PP_4pl(respm = awm, thres = diffpar, slopes = sl, lowerA = la, type = "eap")`

## 4PL model ######
Adaptive Matrices Test data

### Description

This dataset contains real data from the 'Adaptive Matrices Test' (AMT), which is a computer-administered test. This power test assesses logical reasoning as an indicator of general intelligence. The ability to identify regularities and draw logical conclusions is a very good predictor of long-term success at work. The dataset is sparse, because the test tailors a specific set of items for each examinee’s ability level. (More information about adaptive testing: https://en.wikipedia.org/wiki/Computerized_adaptive_testing)

### Format

A list with two data.frames. The first data.frame ‘daten_amt’ contains 298 columns and 710 rows. Each row contains responses from on examinee. The second data.frame ‘betas’ contains the difficulty parameter (1PL) (These parameters came with the raw-score extraction.).
Details

The data are provided from the University of Vienna, Faculty of Psychology, Department of Psychological Assessment. Thanks to Schuhfried https://www.schuhfried.at/test/AMT.

Source

Division of Psychological Assessment and Applied Psychometrics, Faculty of Psychology, University of Vienna

• ID: id of person
• AGE: age in years (with ages below 18 and above 34 are collapsed)
• TE_GA:
  – TE: self-assessment. To pass a psychological assessment course, the students have to complete several hours self assessment on a bunch of tests, to get familiar with them.
  – GA: testing for an assessment report. The students also have to test other people (not psychologists nor psychology students) in order to write an assessment report.
• FORM: There are several different versions of this test, which differ in test length, duration etc . . .
• TIME1: start time
• TIME2: end time
• REL: reliability for each person
• i: items

References


See Also

PPass

---

**PP_gpcm**

*Estimate Person Parameters for the GPCM*

Description

Compute person parameters for the GPCM and choose between five common estimation techniques: ML, WL, MAP, EAP and a robust estimation. All item parameters are treated as fixed.

Usage

```
PP_gpcm(respm, thres, slopes, theta_start = NULL, mu = NULL, sigma2 = NULL, type = "wle", maxsteps = 100, exac = 0.001, H = 1, ctrl = list())
```
Arguments

respm An integer matrix, which contains the examinees responses. A persons x items matrix is expected.

thres A numeric matrix which contains the threshold parameter for each item. If the first row of the matrix is not set to zero (only zeroes in the first row) - then a row-vector with zeroes is added by default.

slopes A numeric vector, which contains the slope parameters for each item - one parameter per item is expected.

theta_start A vector which contains a starting value for each person. Currently this is necessary to supply, but soon it will be set automatically if nothing is committed.

mu A numeric vector of location parameters for each person in case of MAP or EAP estimation. If nothing is submitted this is set to 0 for each person for MAP estimation.

sigma2 A numeric vector of variance parameters for each person in case of MAP or EAP estimation. If nothing is submitted this is set to 1 for each person for MAP estimation.

type Which maximization should be applied? There are five valid entries possible: "mle", "wle", "map", "eap" and "robust". To choose between the methods, or just to get a deeper understanding the papers mentioned below are quite helpful. The default is "wle" which is a good choice in many cases.

maxsteps The maximum number of steps the NR Algorithm will take. Default = 100.

exac How accurate are the estimates supposed to be? Default is 0.001.

H In case type = "robust" a Huber ability estimate is performed, and H modulates how fast the downweighting takes place (for more Details read Schuster & Yuan 2011).

ctrl more controls

• killdupli: Should duplicated response pattern be removed for estimation (estimation is faster)? This is especially resonable in case of a large number of examinees and a small number of items. Use this option with caution (for map and eap), because persons with different mu and sigma2 will have different ability estimates despite they responded identically. Default value is FALSE.

• skipcheck: Default = FALSE. If TRUE data matrix and arguments are not checked - this saves time e.g. when you use this function for simulations.

Details

Please note, that robust estimation with (Huber ability estimate) polytomous items is still experimental!

The probability choosing the k-th category is as follows:

\[
P(x_{ij} = k|\hat{\alpha}_i, \hat{\beta}_{iv}, \theta_j) = \frac{exp(\sum_{v=0}^{(k-1)} \hat{\alpha}_i(\theta_j - \hat{\beta}_{iv}))}{\sum_{v=0}^{m_i-1} exp(\sum_{c=0}^{c} \hat{\alpha}_i(\theta_j - \hat{\beta}_{iv})))}
\]
In our case $\theta$ is to be estimated. The item parameters are assumed as fixed (usually these are estimates of a former scaling procedure).

The model simplifies to the Partial Credit Model by setting $\alpha_i = 1, \forall i$.

**Value**

The function returns a list with the estimation results and pretty much everything which has been submitted to fit the model. The estimation results can be found in `OBJ$resPP`. The core result is a number_of_persons x 2 matrix, which contains the ability estimate and the SE for each submitted person.

**Author(s)**

Manuel Reif

**References**


**See Also**

PPass, PPall, PP_4pl, JKpp, PV

**Examples**

```r
# some threshold parameters
THRES <- matrix(c(-2,-1.23,1.11,3.48,1,
                ,2,-1,-0.2,0.5,1.3,-0.8,1.5),nrow=2)
```
# slopes
sl <- c(0.5,1,1.5,1,1,1,0.98)
awmatrix <- matrix(c(1,0,2,0,1,1,1,0,0,1,2,0,0,0,0,0,0,0,0,0,0,1,2,1,1,1,0,0,1), byrow=TRUE, nrow=5)

## GPCM model ######

# MLE
resgpcmmlle <- PP_gpcm(respm = awmatrix, thres = THRES, slopes = sl, type = "mle")
# WLE
resgpcmwle <- PP_gpcm(respm = awmatrix, thres = THRES, slopes = sl, type = "wle")
# MAP estimation
resgpcmmap <- PP_gpcm(respm = awmatrix, thres = THRES, slopes = sl, type = "map")
# EAP estimation
resgpcmmap <- PP_gpcm(respm = awmatrix, thres = THRES, slopes = sl, type = "eap")
# robust estimation
resgpcmrob <- PP_gpcm(respm = awmatrix, thres = THRES, slopes = sl, type = "robust")

## PCM model #######

# MLE
respcmmlle <- PP_gpcm(respm = awmatrix, thres = THRES, slopes = rep(1,ncol(THRES)), type = "mle")
# WLE
respcmwle <- PP_gpcm(respm = awmatrix, thres = THRES, slopes = rep(1,ncol(THRES)), type = "wle")
# MAP estimation
respcmmap <- PP_gpcm(respm = awmatrix, thres = THRES, slopes = rep(1,ncol(THRES)),
                     type = "map")
# EAP estimation
respcmmap <- PP_gpcm(respm = awmatrix, thres = THRES, slopes = rep(1,ncol(THRES)),
                     type = "eap")

### with different number of categories ###

THRES <- matrix(c(-2,-1.23,1,1,1.3,4.8,1.2,-1,0.2,0.5,1.3,-0.8,1.5), nrow=2)
THRES1 <- rbind(THRES, c(NA,NA,NA,NA,1,7,1))

awmatrix1 <- matrix(c(1,0,2,0,1,3,1,0,0,1,3,1,0,0,0,0,1,2,2,1,1,1,0,0,1), byrow=TRUE, nrow=5)

# MLE estimation
respcmmlle1 <- PP_gpcm(respm = awmatrix1, thres = THRES1, slopes = sl, type = "mle")
# WLE estimation
respcmwle1 <- PP_gpcm(respm = awmatrix, thres = THRES1, slopes = sl, type = "wle")
# MAP estimation
respcmmap1 <- PP_gpcm(respm = awmatrix1, thres = THRES1, slopes = sl, type = "map")
# EAP estimation
respcmmap1 <- PP_gpcm(respm = awmatrix1, thres = THRES1, slopes = sl, type = "eap")
Description

This function draws npv plausible values for each person from their posterior density.

Usage

PV(estobj, ...)

## S3 method for class 'fourpl'
PV(estobj, npv = 10, approx = TRUE, thinning = 6,
   burnin = 10, mult = 2, ...)

## S3 method for class 'gpcm'
PV(estobj, npv = 10, approx = TRUE, thinning = 6,
   burnin = 10, mult = 2, ...)

## S3 method for class 'gpcm4pl'
PV(estobj, npv = 10, approx = TRUE, thinning = 6,
   burnin = 10, mult = 2, ...)

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

## S3 method for class 'pv'
summary(object, nrowmax = 15, ...)

Arguments

estobj An object which originates from using PP_4pl(), PP_gpcm() or PP_all(). EAP estimation is strongly recommended (type = "eap"), when plausible values are drawn afterwards, because the EAP estimate is used as starting point for the MH algorithm.

... More arguments

npv The number of (effectively returned) plausible values - default is 10.

approx Whether a normal approximation N(mu, sigma2) is used to draw the plausible values. Default = TRUE. If FALSE a Metropolitan-Hastings-Algorithm will draw the values.

thinning A numeric vector of length = 1. If approx = FALSE, a Metropolitan-Hastings-Algorithm draws the plausible values. To avoid autocorrelation, thinning takes every kth value as effective plausible value. The default is 6 (every 6th value is taken), which works appropriately in almost all cases here (with default settings).

burnin How many draws should be discarded at the chains beginning? Default is 10 - and this seems reasonable high (probably 5 will be enough as well), because starting point is the EAP.

mult Multiplication constant (default = 2). Use this parameter to vary the width of the proposal distribution - which is N(theta_v, mult*SE_eap) - when a MH-Algorithm is applied. So the constant quantifies the width in terms of multiples
of the EAP standard error. 2 works fine with the default thinning. If the supplied value is large, thinning can take lower values without causing autocorrelation.

x

object

nrowmax

Value

The function returns a list which main element is pvdraws. This is a matrix of size number_of_persons x npv - so if 10 plausible values are requested for 100 persons, a 100x10 matrix is returned.

Author(s)

Manuel Reif

References


See Also

PP_gpcm, PP_4pl, JKpp

Examples

# Plausible values

### 4 PL model ####

### data creation ####

set.seed(1522)

# intercepts
diffpar <- seq(-3,3,length=12)

# slope parameters
sl <- round(runif(12,0.5,1.5),2)
l1 <- round(runif(12,0,0.25),2)
ua <- round(runif(12,0.8,1),2)

# response matrix
awm <- matrix(sample(0:1,10*12,replace=TRUE),ncol=12)
# EAP estimation - 2pl model
res2pleap <- PP_4pl(respm = awm, thres = diffpar, slopes = sl, type = "eap")

# draw 10 plausible values
res_pv <- PV(res2pleap)
summary(res_pv)

# draw 10 plausible values - use a metropolitan hastings algorithm
res_pv2 <- PV(res2pleap, approx = FALSE)
summary(res_pv2)

# ------- check the PVs

# -- autocorrelation?
autocor <- function(acv)
{
  cor(acv[-1], acv[-length(acv)])
}

res_pvac <- PV(res2pleap, approx = FALSE, npv = 200)

# independent draws - so there cannot be any systematic autocorrelation when
# approx = TRUE. So this acts as a kind of benchmark for the MH-Alg.
res_pvac2 <- PV(res2pleap, approx = TRUE, npv = 200)

apply(res_pvac$pvdraws, 1, autocor)
apply(res_pvac2$pvdraws, 1, autocor)

# -- autocorrelation distr?

apply(res_pvac$pvdraws, 1, quantile)
apply(res_pvac2$pvdraws, 1, quantile)

### GPCM model ######

# some threshold parameters
THRES <- matrix(c(-2,-1.23,1.11,3.48,1
      ,2,-1,-0.2,0.5,1.3,-0.8,1.5), nrow=2)
# slopes
sl <- c(0.5,1,1.5,1,1,0.98)
awmatrix <- matrix(c(1,0.2,0,1,1,0,0.1
      ,2,0,0,0,0,0,0,0,0,1,2,1,1,1,0,0,1), byrow=TRUE, nrow=5)

# EAP estimation
resgpcmeap <- PP_gpcm(respm = awmatrix, thres = THRES, slopes = sl, type = "eap")
res_gpcmpv <- PV(resgpcmeap,approx = FALSE,npv = 20)

### GPCM and 4PL model ######

# some threshold parameters
THRES <- matrix(c(-2,-1.23,1.11,3.48,1
                 ,2,-1,-0.2,0.5,1.3,-0.8,1.5),nrow=2)
# slopes
sl <- c(0.5,1.5,1.1,1,0.98)

THRESx <- THRES
THRESx[2,1:3] <- NA

# for the 4PL item the estimated parameters are submitted, # for the GPCM items the lower asymptote = 0 # and the upper asymptote = 1.
la <- c(0.02,0.1,0,0,0)
uu <- c(0.97,0.91,1,1,1)

awmatrix <- matrix(c(1,0,1,0,1,1,1,0,0,1
                      ,2,0,0,0,0,0,0,0,0,1
                      ,1,2,2,1,1,1,0,0,1,0),byrow=TRUE,nrow=5)

model2est <- findmodel(THRESx)

# EAP estimation
respcmeap1 <- PPall(respm = awmatrix,thres = THRESx,
                      slopes = sl,lowerA = la, upperA=uu, type = "eap",
                      model2est=model2est)

res_mixedpv_1 <- PV(respcmeap1,approx = FALSE,npv = 200)

# rowMeans of plausible values should approximate the EAPs
rowMeans(res_mixedpv_1$pvdraws)
# EAPs
respcmeap1

# show the quantiles of the empirical distribution
apply(res_mixedpv_1$pvdraws,1,quantile)
Description

This function returns a dichotomous matrix of simulated responses under given item and person parameters.

Usage

sim_4pl(beta, alpha, lowerA, upperA, theta)

Arguments

- **beta**: A numeric vector which contains the difficulty parameters for each item.
- **alpha**: A numeric vector, which contains the slope parameters for each item.
- **lowerA**: A numeric vector, which contains the lower asymptote parameters (kind of guessing parameter) for each item.
- **upperA**: numeric vector, which contains the upper asymptote parameters for each item.
- **theta**: A numeric vector which contains person parameters.

Author(s)

Manuel Reif

See Also

sim_gpcm, PP_4pl

Examples

```
# simulate 4PL
set.seed(1700)

# simulate 1-PL model
thetas <- c(0.231, -1.313, 1.772, 1.601, 1.733, -2.001, 0.443, 3.111, -4.156)
s1 <- c(1, 1.1, 0.9, 0.89, 1.5, 1.1, 1)
sla <- c(0, 0, 0.2, 0.15, 0.04, 0, 0.21)
sua <- c(0, 0.98, 0.97, 1, 1, 0.97)

simdat1pl <- sim_4pl(beta=seq(-4, 4, length.out=7), alpha=rep(1, 7),
                      lowerA=rep(0, 7), upperA=rep(1, 7), theta=thetas)

head(simdat1pl)

# simulate 2-PL model
simdat2pl <- sim_4pl(beta=seq(-4, 4, length.out=7), alpha=s1,
                     lowerA=rep(0, 7), upperA=rep(1, 7), theta=thetas)
```
Simulate data for the gpcm model

Description

This function returns an integer matrix of simulated responses under given item and person parameters.

Usage

sim_gpcm(thres, alpha, theta)

Arguments

- **thres**: An numeric matrix which contains threshold parameters for each item. The first row must contain zeroes only!
- **alpha**: A numeric vector, which contains the slope parameters - one parameter per item is expected.
- **theta**: A numeric vector which contains person parameters.

See Also

sim_4pl, PP_gpcm
Examples

```
set.seed(1750)

THRES <- matrix(c(-2,-1.23,1.11,3.48,1,2,-1,-0.2,0.5,1.3,-0.8,1.5),nrow=2)

# slopes
sl <- c(0.5,1,1.5,1.1,1,0.98)

THRESx <- rbind(0,THRES)

THETA <- rnorm(100)

simdat_gpcm <- sim_gpcm(thres = THRESx, alpha = sl, theta = THETA)

head(simdat_gpcm)

### simulate with a different number of categories

THRES1 <- rbind(THRESx,c(NA,NA,NA,NA,1.7,1))

THRES1 # last 2 items have +1 category

simdat_gpcm2 <- sim_gpcm(thres = THRES1, alpha = sl, theta = THETA)

head(simdat_gpcm2)

# check the maximum category
apply(simdat_gpcm2,2,max)
```
Index

*Topic 4pl
  PP_4pl, 20

*Topic GPCM
  PP_gpcm, 26

*Topic Infit-Outfit
  Pfit, 9

*Topic LZ-Index,
  Pfit, 9

*Topic Parameters,
  PP_4pl, 20
  PP_gpcm, 26

*Topic Parameters
  findmodel, 2

*Topic Person
  findmodel, 2
  Pfit, 9
  PP_4pl, 20
  PP_gpcm, 26

*Topic fit,
  Pfit, 9

findmodel, 2
fourpl_df, 4

JKpp, 4, 17, 23, 28, 31

Pfit, 9, 13, 19
PP, 12
PP-package (PP), 12
PP_4pl, 6, 11, 13, 17, 19, 20, 28, 31, 34
pp_amt, 25
PP_gpcm, 6, 13, 17, 19, 23, 26, 31, 35
PPall, 2, 6, 11, 13, 14, 19, 23, 28
PPass, 11, 13, 17, 18, 23, 26, 28
print.jk (JKpp), 4
print.ppeo (PPall), 14
print.pv (PV), 29
PV, 17, 23, 28, 29

sim_4pl, 4, 33, 35

sim_gpcm, 34, 35
summary.jk (JKpp), 4
summary.ppeo (PPall), 14
summary.pv (PV), 29