Package 'psychotools'

May 11, 2024

Title Psychometric Modeling Infrastructure

Version 0.7-4

Date 2024-05-11

Depends R (>= 3.6.0)

Imports graphics, grDevices, stats, utils

Suggests Formula, likert, mirt, multcomp, sandwich

Description Infrastructure for psychometric modeling such as data classes (for item response data and paired comparisons), basic model fitting functions (for Bradley-Terry, Rasch, parametric logistic IRT, generalized partial credit, rating scale, multinomial processing tree models), extractor functions for different types of parameters (item, person, threshold, discrimination, guessing, upper asymptotes), unified inference and visualizations, and various datasets for illustration. Intended as a common lightweight and efficient toolbox for psychometric modeling and a common building block for fitting psychometric mixture models in package ``psychomix'' and trees based on psychometric models in package ``psychotree''.

License GPL-2 | GPL-3

NeedsCompilation yes

Author Achim Zeileis [aut, cre] (<https://orcid.org/0000-0003-0918-3766>),
Carolin Strobl [aut] (<https://orcid.org/0000-0003-0952-3230>),
Florian Wickelmaier [aut],
Basil Komboz [aut],
Julia Kopf [aut],
Lennart Schneider [aut] (<https://orcid.org/0000-0003-4152-5308>),
Rudolf Debelak [aut] (<https://orcid.org/0000-0001-8900-2106>)

Maintainer Achim Zeileis <Achim.Zeileis@R-project.org>

Repository CRAN

Date/Publication 2024-05-11 08:40:02 UTC

R topics documented:

---

anchor *Anchor Methods for the Detection of Uniform DIF the Rasch Model*

Description

The anchor function provides a variety of anchor methods for the detection of uniform differential item functioning (DIF) in the Rasch model between two pre-specified groups. These methods can be divided in an anchor class that determines characteristics of the anchor method and an anchor selection that determines the ranking order of candidate anchor items. The aim of the anchor function is to provide anchor items for DIF testing, e.g. with anchortest.

Usage

```
anchor(object, ...)
## Default S3 method:
anchor(object, object2,
  class = c("constant", "forward"), select = NULL,
  length = NULL, range = c(0.1, 0.8), ...)
## S3 method for class 'formula'
anchor(formula, data = NULL, subset = NULL,
  na.action = NULL, weights = NULL, model = raschmodel, ...)
```

Arguments

object, object2	Fitted model objects of class "raschmodel" estimated via conditional maximum likelihood using raschmodel.
...	further arguments passed over to an internal call of anchor.default in the formula method. In the default method, these additional arguments are currently not being used.
class	character. Available anchor classes are the constant anchor class implying a constant anchor length defined by length and the iterative forward anchor class, for an overview see Kopf et al. (2015a).
select	character. Several anchor selection strategies are available: "MTT", "MPT", "MT", "MP", "NST", "AO", "AOP" based on Kopf et al. (2015b) as well as "Gini", "CLF", "GiniT", "CLFT" based on Strobl et al. (2021). The latter four can only be combined with class = "constant" and length = 1. The default is select = "Gini" unless either length > 1 where "MPT" is used or class = "forward" where "MTT" is used. For more details see below.

length	integer. It pre-defines a maximum anchor length. Per default, the forward anchor grows up to the proportion of currently presumed DIF-free items specified in range and the constant anchor class selects one anchor item, unless an explicit limiting number is defined in length by the user.
range	numeric vector of length 2. The first element is the percentage of first anchor candidates to be excluded for consideration when the forward anchor class is used and the second element determines a percentage of currently presumed DIF-free items up to which the anchor from the forward anchor class is allowed to grow.
formula	formula of type y ~ x where y specifies a matrix of dichotomous item responses and x the grouping variable, e.g., gender, for which DIF should be tested for.
data	a data frame containing the variables of the specified formula.
subset	logical expression indicating elements or rows to keep: missing values are taken as false.
na.action	a function which indicates what should happen when the data contain missing values (NAs).
weights	an optional vector of weights (interpreted as case weights).
model	an IRT model fitting function with a suitable itempar method, by default raschmodel.

Details

The anchor methods provided consist of an anchor class that determines characteristics of the anchor and an anchor selection that determines the ranking order of candidate anchor items.

In the constant anchor class, the anchor length is pre-defined by the user within the argument length, defaulting to a length of one. In contrast, the iterative forward class starts with a single anchor item and includes items in the anchor as long as the anchor length is shorter than a certain percentage of the number of items that do not display statistically significant DIF (default: 0.8). Furthermore, a percentage of first anchor candidates is excluded from consideration (default: 0.1) and the user is allowed to set a maximum number of anchor items using the argument length. A detailed description of the anchor classes can be found in Kopf et al. (2015a).

In more recent work Strobl et al. (2021) suggest a simpler yet powerful anchor method based on inequality criteria like the Gini coefficient. A similar approach based on the component loss function (CLF) was suggested by Muthén & Asparouhov (2014). These criteria can be shown to attain their optimium for a single-anchor, thus correponding to a constant class of length 1. Due to the simple structure in combination with good empirical performance the Gini-based selection was made the default in version 0.7-0 of the package.

Both anchor classes require an explicit anchor selection strategy (as opposed to the all-other anchor class which is therefore not included in the function anchor). The anchor selection strategy determines the ranking order of candidate anchor items. In case of two groups, each item $j, j = 1, \ldots, k$ (where k denotes the number of items in the test) obtains a criterion value c_j that is defined by the anchor selection strategy. The ranking order is determined by the rank of the criterion value $\text{rank}(c_j)$.

The criterion values c_j for item j from the different anchor selection strategies are provided in the following equations: d_j denotes the difference of the item parameters, t_j the corresponding test statistic, and p_j the resulting p-values. In all cases, the anchor items are given in parentheses.

Furthermore, $\mathrm{Gini}(\cdot)$ denotes the Gini inequality index, $\mathrm{CLF}(\cdot)$ the component loss function (sum of square root values), $1(\cdot)$ the indicator function, $\lceil 0.5 \cdot k \rceil$ the empirical 50% quantile, and A_{purified} the anchor after purification steps. More detailed descriptions are available in Strobl et al. (2021) and Kopf et al. (2015b).

Gini selection (of item parameter differences) by Strobl et al. (2021):

$$c_j^{\mathrm{Gini}} = -\mathrm{Gini}(\{|d_1(j)|, \ldots, |d_k(j)|\})$$

GiniT selection (of test statistics) similar to Strobl et al. (2021):

$$c_j^{\mathrm{GiniT}} = -\mathrm{Gini}(\{|t_1(j)|, \ldots, |t_k(j)|\})$$

CLF selection (of item parameter differences) by Muthén & Asparouhov (2014):

$$c_j^{\mathrm{CLF}} = \mathrm{CLF}(\{|d_1(j)|, \ldots, |d_k(j)|\})$$

CLFT selection (of test statistics) similar to Muthén & Asparouhov (2014):

$$c_j^{\mathrm{CLFT}} = \mathrm{CLF}(\{|t_1(j)|, \ldots, |t_k(j)|\})$$

All-other selection by Woods (2009), here abbreviated AO:

$$c_j^{\mathrm{AO}} = |t_j(\{1, \ldots, k\} \setminus j)|$$

All-other purified selection by Wang et al. (2012), here abbreviated AOP:

$$c_j^{\mathrm{AOP}} = |t_j(A_{\mathrm{purified}})|$$

Number of significant threshold selection based on Wang et al. (2004), here abbreviated NST:

$$c_j^{\mathrm{NST}} = \sum_{l \in \{1,\ldots,k\} \setminus j} 1\{p_j(\{l\}) \leq \alpha\})|$$

Mean test statistic selection by Shih et al. (2009), here abbreviated MT:

$$c_j^{\mathrm{MT}} = \frac{1}{k-1} \sum_{l \in \{1,\ldots,k\} \setminus j} |t_j(\{l\})|$$

Mean p-value selection by Kopf et al. (2015b), here abbreviated MP:

$$c_j^{\mathrm{MP}} = -\frac{1}{k-1} \sum_{l \in \{1,\ldots,k\} \setminus j} p_j(\{l\})$$

Mean test statistic threshold selection by Kopf et al. (2015b), here abbreviated MTT:

$$c_j^{\mathrm{MTT}} = \sum_{l \in \{1,\ldots,k\} \setminus j} 1\left\{ |t_j(\{l\})| > \left(\left| \frac{1}{k-1} \sum_{l \in \{1,\ldots,k\} \setminus j} t_j(\{l\}) \right| \right)_{(\lceil 0.5 \cdot k \rceil)} \right\}$$

Mean p-value threshold selection by Kopf et al. (2015b), here abbreviated MPT:

$$c_j^{\mathrm{MPT}} = -\sum_{l \in \{1,\ldots,k\} \setminus j} 1\left\{ p_j(\{l\}) > \left(\frac{1}{k-1} \sum_{l \in \{1,\ldots,k\} \setminus j} p_j(\{l\}) \right)_{(\lceil 0.5 \cdot k \rceil)} \right\}$$

Kopf et al. (2015b) recommend to combine the `class = "constant"` with `select = "MPT"` and the `class = "forward"` with `select = "MTT"`, respectively.

The `all-other` anchor class (that assumes that DIF is balanced i.e. no group has an advantage in the test) is here not considered as explicit anchor selection and, thus, not included in the `anchor` function (but in the `anchortest` function). Note that the `all-other` anchor class requires strong prior knowledge that DIF is balanced.

Value

An object of class `anchor`, i.e. a list including

`anchor_items`	the integer index for the selected anchor items
`ranking_order`	a ranking order (integer index) of the candidate anchor items by their criterion values
`criteria`	the criterion values obtained in the anchor selection for each item (unsorted)

References

Kopf J, Zeileis A, Strobl C (2015a). A Framework for Anchor Methods and an Iterative Forward Approach for DIF Detection. *Applied Psychological Measurement*, **39**(2), 83–103. doi:10.1177/0146621614544195

Kopf J, Zeileis A, Strobl C (2015b). Anchor Selection Strategies for DIF Analysis: Review, Assessment, and New Approaches. *Educational and Psychological Measurement*, **75**(1), 22–56. doi:10.1177/0013164414529792

Muthén B, Asparouhov T (2014). IRT Studies of Many Groups: The Alignment Method. *Frontiers in Psychology*, **5**, 978. doi:10.3389/fpsyg.2014.00978

Shih CL, Wang WC (2009). Differential Item Functioning Detection Using the Multiple Indicators, Multiple Causes Method with a Pure Short Anchor. *Applied Psychological Measurement*, **33**(3), 184–199.

Strobl C, Kopf J, Kohler L, von Oertzen T, Zeileis A (2021). Anchor Point Selection: Scale Alignment Based on an Inequality Criterion. *Applied Psychological Measurement*, **45**(3), 214–230. doi:10.1177/0146621621990743

Wang WC (2004). Effects of Anchor Item Methods on the Detection of Differential Item Functioning within the Family of Rasch Models. *Journal of Experimental Education*, **72**(3), 221–261.

Wang WC, Shih CL, Sun GW (2012). The DIF-Free-then-DIF Strategy for the Assessment of Differential Item Functioning. *Educational and Psychological Measurement*, **72**(4), 687–708.

Woods C (2009). Empirical Selection of Anchors for Tests of Differential Item Functioning. *Applied Psychological Measurement*, **33**(1), 42–57.

See Also

anchortest

Examples

```
## Verbal aggression data
data("VerbalAggression", package = "psychotools")

## Gini anchor (Strobl et al. 2021) for gender DIF in the self-to-blame situations
anchor(resp2[, 1:12] ~ gender , data = VerbalAggression)

## alternatively: based on fitted raschmodel objects
raschmodels <- with(VerbalAggression, lapply(levels(gender), function(i)
  raschmodel(resp2[gender == i, 1:12])))
anchor(raschmodels[[1]], raschmodels[[2]])

if(requireNamespace("multcomp")) {

## four anchor items from constant anchor class using MPT-selection (Kopf et al. 2015b)
anchor(object = raschmodels[[1]], object2 = raschmodels[[2]],
  class = "constant", select = "MPT", length = 4)

## iterative forward anchor class using MTT-selection (Kopf et al. 2015b)
set.seed(1)
fanchor <- anchor(object = raschmodels[[1]], object2 = raschmodels[[2]],
  class = "forward", select = "MTT", range = c(0.05, 1))
fanchor

## the same using the formula interface
set.seed(1)
fanchor2 <- anchor(resp2[, 1:12] ~ gender , data = VerbalAggression,
  class = "forward", select = "MTT", range = c(0.05, 1))

## criteria really the same?
all.equal(fanchor$criteria, fanchor2$criteria, check.attributes = FALSE)
}
```

---

anchortest *Anchor methods for the detection of uniform DIF in the Rasch model*

Description

The anchortest function provides a Wald test (see, e.g., Glas, Verhelst, 1995) for the detection of uniform differential item functioning (DIF) in the Rasch model between two pre-specified groups. A variety of anchor methods is available to build a common scale necessary for the comparison of the item parameters in the Rasch model.

Usage

```
anchortest(object, ...)
## Default S3 method:
anchortest(object, object2,
  class = c("constant", "forward", "all-other", "fixed"), select = NULL,
  test = TRUE, adjust = "none", length = NULL, range = c(0.1, 0.8), ...)
## S3 method for class 'formula'
anchortest(formula, data = NULL, subset = NULL,
  na.action = NULL, weights = NULL, model = raschmodel, ...)
```

Arguments

`object, object2`	Fitted model objects of class "raschmodel" estimated via conditional maximum likelihood using `raschmodel`.
`...`	further arguments passed over to an internal call of `anchor.default` in the formula method. In the default method, these additional arguments are currently not being used.
`class`	character. Available anchor classes are the `constant` anchor class implying a constant anchor length defined by `length`, the iterative `forward` anchor class that iteratively includes items in the anchor and the `all-other` anchor class, for an overview see Kopf et al. (2015a). Additionally, the class can be `fixed`, then `select` needs to be the numeric index of the fixed selected anchor items.
`select`	character or numeric. Several anchor selection strategies are available, for details see `anchor`. Alternatively, for `class = "fixed"`, `select` needs to be the numeric index of the fixed selected anchor items. Defaults are set such that `class = "constant"` is combined with `select = "Gini"` while `class = "forward"` is combined with `select = "MTT"`. And if `select` is numeric, then `class = "fixed"` is used.
`test`	logical. Should the Wald test be returned for the intended anchor method as final DIF test?
`adjust`	character. Should the final DIF test be adjusted for multiple testing? For the type of adjustment, see `summary.glht` and `p.adjust`.
`length`	integer. It pre-defines a maximum anchor length. Per default, the `forward` anchor grows up to the proportion of currently presumed DIF-free items specified in `range` and the `constant` anchor class selects four anchor items, unless an explicit limiting number is defined in `length` by the user.
`range`	numeric vector of length 2. The first element is the percentage of first anchor candidates to be excluded for consideration when the `forward` anchor class is used and the second element determines a percentage of currently presumed DIF-free items up to which the anchor from the `forward` anchor class is allowed to grow.
`formula`	formula of type `y ~ x` where `y` specifies a matrix of dichotomous item responses and `x` the grouping variable, e.g., gender, for which DIF should be tested for.
`data`	a data frame containing the variables of the specified `formula`.

subset	logical expression indicating elements or rows to keep: missing values are taken as false.
na.action	a function which indicates what should happen when the data contain missing values (NAs).
weights	an optional vector of weights (interpreted as case weights).
model	an IRT model fitting function with a suitable itempar method, by default raschmodel.

Details

To conduct the Wald test (see, e.g., Glas, Verhelst, 1995) for uniform DIF in the Rasch model, the user needs to specify an anchor method. The anchor methods can be divided in an anchor class that determines characteristics of the anchor method and an anchor selection that determines the ranking order of candidate anchor items.

Explicit anchor selection strategies are used in the constant anchor class and in the iterative forward anchor class, for a detailed description see anchor. Since $k - 1$ parameters are free in the estimation, only $k - 1$ estimated standard errors result. Thus, the first anchor item obtains no DIF test result and we report $k - 1$ test results. This decision is applied only to those methods that rely on an explicit anchor selection strategy.

In the constant anchor class, the anchor length is pre-defined by the user within the argument length. The default is a single anchor item. The iterative forward class starts with a single anchor item and includes items in the anchor as long as the anchor length is shorter than a certain percentage of the number of items that do not display statistically significant DIF. The default proportion is set to 0.8 in the argument range. Alternatively, the user is allowed to set a maximum number of anchor items using the argument length. Both anchor classes require an explicit anchor selection strategy as opposed to the all-other anchor class.

The all-other anchor class is here not considered as explicit anchor selection and, thus, only included in the anchortest function. For the all-other anchor class, the strategy is set to "none", since all items except for the item currently studied for DIF are used as anchor. Thus, no explicit anchor selection strategy is required and we report k test results. Note that the all-other anchor class requires strong prior knowledge that DIF is balanced.

See Strobl et al. (2021) and Kopf et al. (2015ab) for a detailed introduction. For convenience a trivial "fixed" anchor class is provided where the selected anchor is given directly (e.g., as chosen by a practitioner or by some other anchor selection method).

Value

An object of class anchor, i.e. a list including

anchor_items	the anchor items for DIF analysis.
ranking_order	a ranking order of candidate anchor items.
criteria	the criterion values obtained by the respective anchor selection.
anchored_item_parameters	the anchored item parameters using the anchor items.
anchored_covariances	the anchored covariance matrices using the anchor items.
final_tests	the final Wald test for uniform DIF detection if intended.

References

Glas CAW, Verhelst ND (1995). "Testing the Rasch Model." In Fischer GH, Molenaar IW (eds.), *Rasch Models: Foundations, Recent Developments, and Applications*, chapter 5. Springer-Verlag, New York.

Kopf J, Zeileis A, Strobl C (2015a). A Framework for Anchor Methods and an Iterative Forward Approach for DIF Detection. *Applied Psychological Measurement*, **39**(2), 83–103. doi:10.1177/0146621614544195

Kopf J, Zeileis A, Strobl C (2015b). Anchor Selection Strategies for DIF Analysis: Review, Assessment, and New Approaches. *Educational and Psychological Measurement*, **75**(1), 22–56. doi:10.1177/0013164414529792

Strobl C, Kopf J, Kohler L, von Oertzen T, Zeileis A (2021). Anchor Point Selection: Scale Alignment Based on an Inequality Criterion. *Applied Psychological Measurement*, **45**(3), 214–230. doi:10.1177/0146621621990743

Wang WC (2004). Effects of Anchor Item Methods on the Detection of Differential Item Functioning within the Family of Rasch Models. *Journal of Experimental Education*, **72**(3), 221–261.

Woods C (2009). Empirical Selection of Anchors for Tests of Differential Item Functioning. *Applied Psychological Measurement*, **33**(1), 42–57.

See Also

anchor

Examples

```
if(requireNamespace("multcomp")) {

o <- options(digits = 4)

## Verbal aggression data
data("VerbalAggression", package = "psychotools")

## Rasch model for the self-to-blame situations; gender DIF test
raschmodels <- with(VerbalAggression, lapply(levels(gender), function(i)
  raschmodel(resp2[gender == i, 1:12])))

## single anchor from Gini selection (default)
gini1 <- anchortest(object = raschmodels[[1]], object2 = raschmodels[[2]])
gini1
summary(gini1)

## four anchor items from constant anchor class using MPT selection
const1 <- anchortest(object = raschmodels[[1]], object2 = raschmodels[[2]],
  class = "constant", select = "MPT", length = 4)
const1
summary(const1)

## iterative forward anchor class using MTT selection
set.seed(1)
forw1 <- anchortest(object = raschmodels[[1]], object2 = raschmodels[[2]],
```

```
  class = "forward", select = "MTT", test = TRUE,
  adjust = "none", range = c(0.05,1))
forw1

## DIF test with fixed given anchor (arbitrarily selected to be items 1 and 2)
anchortest(object = raschmodels[[1]], object2 = raschmodels[[2]], select = 1:2)

options(digits = o$digits)
}
```

`as.list.itemresp` *Coercing Item Response Data*

Description

Coercing `"itemresp"` data objects to other classes.

Usage

```
  ## S3 method for class 'itemresp'
as.list(x, items = NULL, mscale = TRUE, df = FALSE, ...)
```

Arguments

`x`	an object of class `"itemresp"`.
`items`	character, integer, or logical for subsetting the items.
`mscale`	logical. Should the measurement scale labels be used for creating factor levels? If `FALSE`, the values 0, 1, ... are used.
`df`	logical. Should a data frame of factors be returned? If `FALSE`, a plain list of factors is returned.
`...`	currently not used.

Details

The `as.list` method coerces item response data to a list (or data frame) of factors with factor levels either taken from the `mscale(x)` or as the values 0, 1, ...

The `as.data.frame` method returns a data frame with a single column of class `"itemresp"`.

Furthermore, `as.matrix`, `as.integer`, `as.double` all return a matrix with the item responses coded as values 0, 1, ...

The `as.character` method simply calls `format.itemresp`.

`is.itemresp` can be used to check wether a given object is of class `"itemresp"`.

See Also

`itemresp`

Examples

```
## item responses from binary matrix
x <- cbind(c(1, 0, 1, 0), c(1, 0, 0, 0), c(0, 1, 1, 1))
xi <- itemresp(x)
## change mscale
mscale(xi) <- c("-", "+")
xi

## coercion to list of factors with levels taken from mscale
as.list(xi)
## same but levels taken as integers 0, 1
as.list(xi, mscale = FALSE)
## only for first two items
as.list(xi, items = 1:2)
## result as data.frame
as.list(xi, df = TRUE)

## data frame with single itemresp column
as.data.frame(xi)

## integer matrix
as.matrix(xi)

## character vector
as.character(xi)

## check class of xi
is.itemresp(xi)
```

`btmodel` *Bradley-Terry Model Fitting Function*

Description

`btmodel` is a basic fitting function for simple Bradley-Terry models.

Usage

```
btmodel(y, weights = NULL, type = c("loglin", "logit"), ref = NULL,
  undecided = NULL, position = NULL, start = NULL, vcov = TRUE, estfun =
  FALSE, ...)
```

Arguments

`y`	paircomp object with the response.
`weights`	an optional vector of weights (interpreted as case weights).
`type`	character. Should an auxiliary log-linear Poisson model or logistic binomial be employed for estimation? The latter is not available if undecided effects are estimated.

ref	character or numeric. Which object parameter should be the reference category, i.e., constrained to zero?
undecided	logical. Should an undecided parameter be estimated?
position	logical. Should a position effect be estimated?
start	numeric. Starting values when calling glm.fit.
vcov	logical. Should the estimated variance-covariance be included in the fitted model object?
estfun	logical. Should the empirical estimating functions (score/gradient contributions) be included in the fitted model object?
...	further arguments passed to functions.

Details

btmodel provides a basic fitting function for Bradley-Terry models, intended as a building block for fitting Bradley-Terry trees and Bradley-Terry mixtures in the **psychotree** package, respectively. While btmodel is intended for individual paired-comparison data, the **eba** package provides functions for aggregate data.

btmodel returns an object of class "btmodel" for which several basic methods are available, including print, plot, summary, coef, vcov, logLik, estfun and worth.

Value

btmodel returns an S3 object of class "btmodel", i.e., a list with components as follows.

y	paircomp object with the response
coefficients	estimated parameters on log-scale (without the first parameter which is always constrained to be 0),
vcov	covariance matrix of the parameters in the model,
loglik	log-likelihood of the fitted model,
df	number of estimated parameters,
weights	the weights used (if any),
n	number of observations (with non-zero weights),
type	character for model type (see above),
ref	character for reference category (see above),
undecided	logical for estimation of undecided parameter (see above),
position	logical for estimation of position effect (see above),
labels	character labels of the objects compared,
estfun	empirical estimating function (also known as scores or gradient contributions).

See Also

pcmodel, gpcmodel, rsmodel, raschmodel, nplmodel, the **eba** package

Examples

```
o <- options(digits = 4)

## data
data("GermanParties2009", package = "psychotools")

## Bradley-Terry model
bt <- btmodel(GermanParties2009$preference)
summary(bt)
plot(bt)

options(digits = o$digits)
```

ConspiracistBeliefs2016

Generic Conspiracist Beliefs Scale (2016 Data)

Description

Responses of 2449 persons to 15 five-point likert-rated items (0 = disagree to 4 = agree) measuring belief in conspiracy theories as well as responses on 2 covariates.

Usage

```
data("ConspiracistBeliefs2016", package = "psychotools")
```

Format

A data frame containing 2449 observations on 3 variables.

resp Item response matrix with 15 items (see details below).

area Factor coding the area one lived in as a child (`"rural"`, `"suburban"`, `"urban"`).

gender Factor coding gender (`"male"`, `"female"`, `"other"`).

Details

The Open Source Psychometrics Project published this dataset collected online in 2016. Persons responded to the Generic Conspiracist Beliefs (GCB) Scale (Brotherton, French & Pickering, 2013) as well as other additional questions primarily for personal amusement. At the end of the test but before the results were displayed, users were asked if they would allow their responses to be saved for research. Only users who agreed are part of this dataset. Individuals with age lower than 13 years were not recorded. Moreover, two persons stating their age to be 5555 years or higher as well as 44 persons with missing data in `area` or `gender` were excluded from this dataset. The 15 items of the GCB Scale are:

Q1: The government is involved in the murder of innocent citizens and/or well-known public figures, and keeps this a secre
Q2: The power held by heads of state is second to that of small unknown groups who really control world politics.
Q3: Secret organizations communicate with extraterrestrials, but keep this fact from the public.

Q4: The spread of certain viruses and/or diseases is the result of the deliberate, concealed efforts of some organization.
Q5: Groups of scientists manipulate, fabricate, or suppress evidence in order to deceive the public.
Q6: The government permits or perpetrates acts of terrorism on its own soil, disguising its involvement.
Q7: A small, secret group of people is responsible for making all major world decisions, such as going to war.
Q8: Evidence of alien contact is being concealed from the public.
Q9: Technology with mind-control capacities is used on people without their knowledge.
Q10: New and advanced technology which would harm current industry is being suppressed.
Q11: The government uses people as patsies to hide its involvement in criminal activity.
Q12: Certain significant events have been the result of the activity of a small group who secretly manipulate world events.
Q13: Some UFO sightings and rumors are planned or staged in order to distract the public from real alien contact.
Q14: Experiments involving new drugs or technologies are routinely carried out on the public without their knowledge or co
Q15: A lot of important information is deliberately concealed from the public out of self-interest.

Additional information can be found online (see below) via inspecting the codebook contained in 'GCBS.zip'.

Source

https://openpsychometrics.org/_rawdata/.

References

Brotherton R, French CC, Pickering AD (2013). Measuring Belief in Conspiracy Theories: The Generic Conspiracist Beliefs Scale. *Frontiers in Psychology*, **4**, 279.

Open Source Psychometrics Project (2016). Data From: The Generic Conspiracist Beliefs Scale [Dataset]. Retrieved from https://openpsychometrics.org/_rawdata/.

See Also

gpcmodel

Examples

```
## overview
data("ConspiracistBeliefs2016", package = "psychotools")
str(ConspiracistBeliefs2016)

## response
plot(itemresp(ConspiracistBeliefs2016$resp))
## covariates
summary(ConspiracistBeliefs2016[, -1])
```

covariates *Extract/Set Covariates*

Description

A generic function for extracting/setting covariates for an object.

Usage

```
covariates(object, ...)
covariates(object) <- value
```

Arguments

object	an object.
...	arguments passed to methods.
value	an object.

Examples

```
## method for "paircomp" data
pc <- paircomp(rbind(
  c(1,  1,  1), # a > b, a > c, b > c
  c(1,  1, -1), # a > b, a > c, b < c
  c(1, -1, -1), # a > b, a < c, b < c
  c(1,  1,  1)))
covariates(pc)
covariates(pc) <- data.frame(foo = factor(c(1, 2, 2), labels = c("foo", "bar")))
covariates(pc)
```

curveplot *Response Curve Plots for IRT Models*

Description

Base graphics plotting function for response curve plot visualization of IRT models.

Usage

```
curveplot(object, ref = NULL, items = NULL, names = NULL,
  layout = NULL, xlim = NULL, ylim = c(0, 1), col = NULL,
  lty = NULL, main = NULL, xlab = "Latent trait",
  ylab = "Probability", add = FALSE, ...)
```

Arguments

object	a fitted model object of class "raschmodel", "rsmodel", "pcmodel", "nplmodel" or "gpcmodel".
ref	argument passed over to internal calls of predict.
items	character or numeric, specifying the items for which response curves should be visualized.
names	character, specifying labels for the items.
layout	matrix, specifying how the response curve plots of different items should be arranged.
xlim, ylim	numeric, specifying the x and y axis limits.
col	character, specifying the colors of the response curve lines. The length of col should be the maximum number of available categories.
lty	numeric, specifying the line type of the response curve lines. The length of lty should either be one or the maximum number of available categories. In the first case, a single line type is used for all category response curves. In the latter case, separate line types for each category response curve are used.
main	character, specifying the overall title of the plot.
xlab, ylab	character, specifying the x and y axis labels.
add	logical. If TRUE, new response curves are added to an existing plot. Only possible when a single item is visualized.
...	further arguments passed to internal calls of matplot.

Details

The response curve plot visualization illustrates the predicted probabilities as a function of the ability parameter θ under a certain IRT model. This type of visualization is sometimes also called item/category operating curves or item/category characteristic curves.

See Also

regionplot, profileplot, infoplot, piplot

Examples

```
## load verbal aggression data
data("VerbalAggression", package = "psychotools")

## fit Rasch, rating scale and partial credit model to verbal aggression data
rmmod <- raschmodel(VerbalAggression$resp2)
rsmod <- rsmodel(VerbalAggression$resp)
pcmod <- pcmodel(VerbalAggression$resp)

## curve plots of the dichotomous RM
plot(rmmod, type = "curves")

## curve plots under the RSM for the first six items of the data set
```

```
plot(rsmod, type = "curves", items = 1:6)

## curve plots under the PCM for the first six items of the data set with
## custom labels
plot(pcmod, type = "curves", items = 1:6, names = paste("Item", 1:6))

## compare the predicted probabilities under the RSM and the PCM for a single
## item
plot(rsmod, type = "curves", item = 1)
plot(pcmod, type = "curves", item = 1, lty = 2, add = TRUE)
legend(x = "topleft", y = 1.0, legend = c("RSM", "PCM"), lty = 1:2, bty = "n")


if(requireNamespace("mirt")) {
## fit 2PL and generaliced partial credit model to verbal aggression data
twoplmod <- nplmodel(VerbalAggression$resp2)
gpcmod <- gpcmodel(VerbalAggression$resp)

## curve plots of the dichotomous 2PL
plot(twoplmod, type = "curves", xlim = c(-6, 6))

## curve plots under the GPCM for the first six items of the data set
plot(gpcmod, type = "curves", items = 1:6, xlim = c(-6, 6))
}
```

`discrpar` *Extract Discrimination Parameters of Item Response Models*

Description

A class and generic function for representing and extracting the discrimination parameters of a given item response model.

Usage

```
  discrpar(object, ...)
  ## S3 method for class 'raschmodel'
discrpar(object, ref = NULL, alias = TRUE, vcov = TRUE, ...)
  ## S3 method for class 'rsmodel'
discrpar(object, ref = NULL, alias = TRUE, vcov = TRUE, ...)
  ## S3 method for class 'pcmodel'
discrpar(object, ref = NULL, alias = TRUE, vcov = TRUE, ...)
  ## S3 method for class 'nplmodel'
discrpar(object, ref = NULL, alias = TRUE, vcov = TRUE, ...)
  ## S3 method for class 'gpcmodel'
discrpar(object, ref = NULL, alias = TRUE, vcov = TRUE, ...)
```

Arguments

object	a fitted model object whose discrimination parameters should be extracted.
ref	a restriction to be used. Not used for models estimated via CML as the discrimination parameters are fixed to 1 in raschmodels, rsmodels and pcmodels. For models estimated via MML (nplmodels and gpcmodels), the parameters are by default identified via the distributional parameters of the person parameters (mean and variance of a normal distribution). Nevertheless, a restriction on the ratio scale can be applied.
alias	logical. If TRUE (the default), the aliased parameters are included in the return vector (and in the variance-covariance matrix if vcov = TRUE). If FALSE, these parameters are removed. For raschmodels, rsmodels and pcmodels where all discrimination parameters are fixed to 1, this means that an empty numeric vector and an empty variance-covariance matrix is returned if alias is FALSE.
vcov	logical. If TRUE (the default), the variance-covariance matrix of the discrimination parameters is attached as attribute vcov.
...	further arguments which are currently not used.

Details

discrpar is both, a class to represent discrimination parameters of item response models as well as a generic function. The generic function can be used to extract the discrimination parameters of a given item response model.

For objects of class discrpar, several methods to standard generic functions exist: print, coef, vcov. coef and vcov can be used to extract the discrimination parameters and their variance-covariance matrix without additional attributes.

Value

A named vector with discrimination parameters of class discrpar and additional attributes model (the model name), ref (the items or parameters used as restriction/for normalization), alias (either TRUE or a named numeric vector with the aliased parameters not included in the return value), and vcov (the estimated and adjusted variance-covariance matrix).

See Also

personpar, itempar, threshpar, guesspar, upperpar

Examples

```
o <- options(digits = 4)

## load verbal aggression data
data("VerbalAggression", package = "psychotools")

## fit Rasch model to verbal aggression data
rmod <- raschmodel(VerbalAggression$resp2)

## extract the discrimination parameters
```

```
dp1 <- discrpar(rmod)

## extract the standard errors
sqrt(diag(vcov(dp1)))

if(requireNamespace("mirt")) {
## fit 2PL to verbal aggression data
twoplmod <- nplmodel(VerbalAggression$resp2)

## extract the discrimination parameters
dp2 <- discrpar(twoplmod)

## this time with the first discrimination parameter being the reference
discrpar(twoplmod, ref = 1)

## extract the standard errors
sqrt(diag(vcov(dp2)))
}

options(digits = o$digits)
```

`elementary_symmetric_functions` *Calculation of the Elementary Symmetric Functions and Their Derivatives*

Description

Calculation of `elementary_symmetric_functions` (ESFs), their first and, in the case of dichotomous items, second derivatives with sum or difference algorithm for the Rasch, rating scale and partial credit model.

Usage

```
elementary_symmetric_functions(par, order = 0L, log = TRUE,
  diff = FALSE, engine = NULL)
```

Arguments

`par`	numeric vector or a list. Either a vector of item difficulty parameters of dichotomous items (Rasch model) or a list of item-category parameters of polytomous items (rating scale and partial credit model).
`order`	integer between 0 and 2, specifying up to which derivative the ESFs should be calculated. Please note, second order derivatives are currently only possible for dichtomous items in an R implementation `engine == "R"`.
`log`	logical. Are the parameters given in `par` on log scale? Primarily used for internal recursive calls of `elementary_symmetric_functions`.

diff	logical. Should the first and second derivatives (if requested) of the ESFs calculated with sum (FALSE) or difference algorithm (TRUE).
engine	character, either "C" or "R". If the former, a C implementation is used to calculcate the ESFs and their derivatives, otherwise ("R") pure R code is used.

Details

Depending on the type of par, the elementary symmetric functions for dichotomous (par is a numeric vector) or polytomous items (par is a list) are calculated.

For dichotomous items, the summation and difference algorithm published in Liou (1994) is used. For calculating the second order derivatives, the equations proposed by Jansens (1984) are employed.

For polytomous items, the summation and difference algorithm published by Fischer and Pococny (1994) is used (see also Fischer and Pococny, 1995).

Value

elementary_symmetric_function returns a list of length 1 + order.

If order = 0, then the first (and only) element is a numeric vector with the ESFs of order 0 to the maximum score possible with the given parameters.

If order = 1, the second element of the list contains a matrix, with the rows corresponding to the possible scores and the columns corresponding to the derivatives with respect to the i-th parameter of par.

For dichotomous items and order = 2, the third element of the list contains an array with the second derivatives with respect to every possible combination of two parameters given in par. The rows of the individual matrices still correspond to the possibles scores (orders) starting from zero.

References

Liou M (1994). More on the Computation of Higher-Order Derivatives of the Elementary Symmetric Functions in the Rasch Model. *Applied Psychological Measurement*, **18**, 53–62.

Jansen PGW (1984). Computing the Second-Order Derivatives of the Symmetric Functions in the Rasch Model. *Kwantitatieve Methoden*, **13**, 131–147.

Fischer GH, and Ponocny I (1994). An Extension of the Partial Credit Model with an Application to the Measurement of Change. *Psychometrika*, **59**(2), 177–192.

Fischer GH, and Ponocny I (1995). "Extended Rating Scale and Partial Credit Models for Assessing Change." In Fischer GH, and Molenaar IW (eds.). *Rasch Models: Foundations, Recent Developments, and Applications.*

Examples

```
## zero and first order derivatives of 100 dichotomous items
di <- rnorm(100)
system.time(esfC <- elementary_symmetric_functions(di, order = 1))

## again with R implementation
```

```
system.time(esfR <- elementary_symmetric_functions(di, order = 1,
engine = "R"))

## are the results equal?
all.equal(esfC, esfR)


## calculate zero and first order elementary symmetric functions
## for 10 polytomous items with three categories each.
pi <- split(rnorm(20), rep(1:10, each = 2))
x <- elementary_symmetric_functions(pi)

## use difference algorithm instead and compare results
y <- elementary_symmetric_functions(pi, diff = TRUE)
all.equal(x, y)
```

`FirstNames` *Popularity of First Names*

Description

Preferences of 192 respondents choosing among six boys names with respect to their popularity.

Usage

```
data("FirstNames")
```

Format

A data frame containing 192 observations on 11 variables.

preference Paired comparison of class `paircomp`. All 15 pairwise choices among six boys names: Tim, Lucas, Michael, Robin, Benedikt, and Julius.

ordered.pref Ordered paired comparison of class `paircomp`. Same as `preference`, but within-pair order is recognized.

gender Factor coding gender.

age Integer. Age of the respondents in years.

education Ordered factor. Level of education: 1 Hauptschule with degree (Secondary General School), 2 and 3 Realschule without and with degree (Intermediate Secondary School), 4 and 5 Gymnasium without and with degree (High School), 6 and 7 Studium without and with degree (University).

children Integer. Number of children.

state Factor. State of Germany where participant grew up.

state.reg Factor. The region (south, north-west, east) each state belongs to.

fname Factor. Participant's fist name(s). (Umlaute in Jörg and Jürgen have been transliterated to Joerg and Juergen for portability of the data.)

interviewer Factor. Interviewer id.

gender.int Factor coding interviewer's gender.

Details

A survey was conducted at the Department of Psychology, Universität Tübingen, in June 2009. The sample was stratified by gender and age (younger versus older than 30 years) with 48 participants in each group. The interviewers were Psychology Master's students who collected the data for course credits.

Participants were presented with 15 pairs of boys names in random order. On each trial, their task was to choose the name they would rather give to their own child. The pairs of boys names were read to the participants one at a time. A given participant compared each pair in one order only, hence the NA's in `ordered.pref`.

The names were selected to fall within the upper (Tim, Lucas), mid (Michael, Robin) and lower (Benedikt, Julius) range of the top 100 of the most popular boys names in Germany in the years from 1990 to 1999 (https://www.beliebte-vornamen.de/3778-1990er-jahre.htm). The names have either front (e, i) or back (o, u) vowels in the stressed syllables. Phonology of the name and attractiveness of a person have been shown to be related (Perfors, 2004; Hartung et al., 2009).

References

Hartung F, Klenovsak D, Santiago dos Santos L, Strobl C, Zaefferer D (2009). Are Tims Hot and Toms Not? Probing the Effect of Sound Symbolism on Perception of Facial Attractiveness. Presented at the *31th Annual Meeting of the Cognitive Science Society*, July 27–August 1, Amsterdam, The Netherlands.

Perfors A (2004). What's in a Name? The Effect of Sound Symbolism on Perception of Facial Attractiveness. Presented at the *26th Annual Meeting of the Cognitive Science Society*, August 5–7, Chicago, USA.

See Also

`paircomp`

Examples

```
data("FirstNames", package = "psychotools")
summary(FirstNames$preference)
covariates(FirstNames$preference)
```

`GermanParties2009` *Choice among German Political Parties*

Description

Preferences of 192 respondents choosing among five German political parties and abstention from voting.

Usage

```
data("GermanParties2009")
```

Format

A data frame containing 192 observations on 6 variables.

preference Paired comparison of class `paircomp`. All 15 pairwise choices among five German parties and abstention from voting.

ordered.pref Ordered paired comparison of class `paircomp`. Same as `preference`, but within-pair order is recognized.

gender Factor coding gender.

age Integer. Age of the respondents in years.

education Ordered factor. Level of education: 1 no degree, 2 Hauptschule (Secondary General School), 3 Realschule (Intermediate Secondary School), 4 Gymnasium (High School), 5 Studium (University)

crisis Factor. Do you feel affected by the economic crisis?

interviewer Factor. Interviewer id.

Details

A survey was conducted at the Department of Psychology, Universität Tübingen, in June 2009, three months before the German election. The sample was stratified by gender and age (younger versus older than 30 years) with 48 participants in each group.

The parties to be compared were Die Linke (socialists), Die Grünen (ecologists), SPD (social democrats), CDU/CSU (conservatives), and FDP (liberals). In addition, there was the option of abstaining from voting (coded as `none`).

Participants were presented with 15 pairs of options in random order. On each trial, their task was to choose the party they would rather vote for at an election for the German parliament. A given participant compared each pair in one order only, hence the NA's in `ordered.pref`.

In order to minimize response biases, the pairs of options were read to the participants one at a time. Participants made their choices by crossing either "First Option" or "Second Option" on an anonymous response sheet.

The interviewers were Psychology Master's students who collected the data for course credits. Since they mainly interviewed people they knew, the results are not representative of the political opinions in Germany. As far as the winner of the survey (Die Grünen) is concerned, however, the results agree with the outcome of the election for the Tübingen voters.

The results of the election on September 27, 2009 (number of so-called Zweitstimmen in percent) were:

	Germany	Tübingen
Die Linke	11.9	8.5
Die Grünen	10.7	27.9
SPD	23.0	21.1
CDU/CSU	33.8	23.0
FDP	14.6	13.9
Others	6.0	5.7

The voter turnout was 70.8 percent in Germany and 80.5 percent in Tübingen.

See Also

paircomp

Examples

```
data("GermanParties2009", package = "psychotools")
summary(GermanParties2009$preference)
```

---

gpcmodel *Generalized Partial Credit Model Fitting Function*

Description

gpcmodel is a basic fitting function for generalized partial credit models providing a wrapper around mirt and multipleGroup relying on marginal maximum likelihood (MML) estimation via the standard EM algorithm.

Usage

```
gpcmodel(y, weights = NULL, impact = NULL, type = c("GPCM", "PCM"),
  grouppars = FALSE, vcov = TRUE, nullcats = "downcode",
  start = NULL, method = "BFGS", maxit = 500, reltol = 1e-5, ...)
```

Arguments

y	item response object that can be coerced (via as.matrix) to a numeric matrix with scores 0, 1, ... Typically, either already a matrix, data frame, or dedicated object of class itemresp.
weights	an optional vector of weights (interpreted as case weights)
.	
impact	an optional factor allowing for grouping the subjects (rows). If specified, a multiple-group model is fitted to account for impact (see details below). By default, no impact is modelled, i.e., a single-group model is used.
type	character string, specifying the type of model to be estimated. In addition to the default GPCM (generalized partial credit model) it is also possible to estimate a standard PCM (partial credit model) by marginal maximum likelihood (MML).
grouppars	logical. Should the estimated distributional group parameters of a multiple group model be included in the model parameters?
vcov	logical or character specifying the type of variance-covariance matrix (if any) computed for the final model. The default vcov = TRUE corresponds to vcov = "Oakes", see mirt for further options. If set to vcov = FALSE (or vcov = "none"), vcov() will return a matrix of NAs only.

`nullcats` character string, specifying how items with null categories (i.e., categories not observed) should be treated. Currently only `"downcode"` is available, i.e., all categories above a null category are shifted down to close the observed gap(s).

`start` an optional vector or list of starting values (see examples below).

`method, maxit, reltol` control parameters for the optimizer employed by `mirt` for the EM algorithm.

`...` further arguments passed to `mirt` or `multipleGroup`, respectively.

Details

`gpcmodel` provides a basic fitting function for generalized partial credit models (GPCMs) providing a wrapper around `mirt` (and `multipleGroup`, respectively) relying on MML estimation via the standard EM algorithm (Bock & Aitkin, 1981). Models are estimated under the slope/intercept parametrization, see e.g. Chalmers (2012). The probability of person i falling into category x_{ij} of item j out of all categories p_j is modelled as:

$$P(X_{ij} = x_{ij}|\theta_i, a_j, \boldsymbol{d_j}) = \frac{\exp\left(x_{ij}a_j\theta_i + d_{jx_{ij}}\right)}{\sum_{k=0}^{p_j} \exp\left(ka_j\theta_i + d_{jk}\right)}$$

Note that all d_{j0} are fixed at 0. A reparametrization of the intercepts to the classical IRT parametrization, see e.g. Muraki (1992), is provided via `threshpar`.

If an optional `impact` variable is supplied, a multiple-group model of the following form is being fitted: Item parameters are fixed to be equal across the whole sample. For the first group of the `impact` variable the person parameters are fixed to follow the standard normal distribution. In the remaining `impact` groups, the distributional parameters (mean and variance of a normal distribution) of the person parameters are estimated freely. See e.g. Baker & Kim (2004, Chapter 11), Debelak & Strobl (2019), or Schneider et al. (2022) for further details. To improve convergence of the model fitting algorithm, the first level of the `impact` variable should always correspond to the largest group. If this is not the case, levels are re-ordered internally.

If `grouppars` is set to `TRUE` the freely estimated distributional group parameters (if any) are returned as part of the model parameters.

Instead of the default GPCM, a standard partial credit model (PCM) can also be estimated via MML by setting `type = "PCM"`. In this case all slopes are restricted to be equal across all items.

`gpcmodel` returns an object of class `"gpcmodel"` for which several basic methods are available, including `print`, `plot`, `summary`, `coef`, `vcov`, `logLik`, `estfun`, `discrpar`, `itempar`, `threshpar`, and `personpar`.

Value

`gpcmodel` returns an S3 object of class `"gpcmodel"`, i.e., a list of the following components:

`coefficients` estimated model parameters in slope/intercept parametrization,

`vcov` covariance matrix of the model parameters,

`data` modified data, used for model-fitting, i.e., centralized so that the first category is zero for all items, treated null categories as specified via argument `"nullcats"` and without observations with zero weight,

items	logical vector of length ncol(y), indicating which items were used during estimation,
categories	list of length ncol(y), containing integer vectors starting from one to the number of categories minus one per item,
n	number of observations (with non-zero weights),
n_org	original number of observations in y,
weights	the weights used (if any),
na	logical indicating whether the data contain NAs,
nullcats	currently always NULL as eventual items with null categories are handled via "downcode",
impact	either NULL or the supplied impact variable with the levels reordered in decreasing order (if this has not been the case prior to fitting the model),
loglik	log-likelihood of the fitted model,
df	number of estimated (more precisely, returned) model parameters,
code	convergence code from mirt,
iterations	number of iterations used by mirt,
reltol	convergence threshold passed to mirt,
grouppars	the logical grouppars value,
type	the type of model restriction specified,
mirt	the mirt object fitted internally,
call	original function call.

References

Baker FB, Kim SH (2004). *Item Response Theory: Parameter Estimation Techniques*. Chapman & Hall/CRC, Boca Raton.

Bock RD, Aitkin M (1981). Marginal Maximum Likelihood Estimation of Item Parameters: Application of an EM Algorithm. *Psychometrika*, **46**(4), 443–459.

Chalmers RP (2012). mirt: A Multidimensional Item Response Theory Package for the R Environment. *Journal of Statistical Software*, **48**(6), 1–29. doi:10.18637/jss.v048.i06

Debelak R, Strobl C (2019). Investigating Measurement Invariance by Means of Parameter Instability Tests for 2PL and 3PL Models. *Educational and Psychological Measurement*, **79**(2), 385–398. doi:10.1177/0013164418777784

Muraki E (1992). A Generalized Partial Credit Model: Application of an EM Algorithm. *Applied Psychological Measurement*, **16**(2), 159–176.

Schneider L, Strobl C, Zeileis A, Debelak R (2022). An R Toolbox for Score-Based Measurement Invariance Tests in IRT Models. *Behavior Research Methods*, forthcoming. doi:10.3758/s13428-021016890

See Also

pcmodel, rsmodel, nplmodel, raschmodel, btmodel

Examples

```
if(requireNamespace("mirt")) {

o <- options(digits = 4)

## mathematics 101 exam results
data("MathExam14W", package = "psychotools")

## generalized partial credit model
gpcm <- gpcmodel(y = MathExam14W$credit)
summary(gpcm)

## how to specify starting values as a vector of model parameters
st <- coef(gpcm)
gpcm <- gpcmodel(y = MathExam14W$credit, start = st)
## or a list containing a vector of slopes and a list of intercept vectors
## itemwise
set.seed(0)
st <- list(a = rlnorm(13, 0, 0.0625), d = replicate(13, rnorm(2, 0, 1), FALSE))
gpcm <- gpcmodel(y = MathExam14W$credit, start = st)

## visualizations
plot(gpcm, type = "profile")
plot(gpcm, type = "regions")
plot(gpcm, type = "piplot")
plot(gpcm, type = "curves", xlim = c(-6, 6))
plot(gpcm, type = "information", xlim = c(-6, 6))
## visualizing the IRT parametrization
plot(gpcm, type = "curves", xlim = c(-6, 6), items = 1)
abline(v = threshpar(gpcm)[[1]])
abline(v = itempar(gpcm)[1], lty = 2)

options(digits = o$digits)
}
```

---

guesspar *Extract Guessing Parameters of Item Response Models*

Description

A class and generic function for representing and extracting the so-called guessing parameters of a given item response model.

Usage

```
  guesspar(object, ...)
  ## S3 method for class 'raschmodel'
guesspar(object, alias = TRUE, vcov = TRUE, ...)
  ## S3 method for class 'rsmodel'
```

```
guesspar(object, alias = TRUE, vcov = TRUE, ...)
  ## S3 method for class 'pcmodel'
guesspar(object, alias = TRUE, vcov = TRUE, ...)
  ## S3 method for class 'nplmodel'
guesspar(object, alias = TRUE, logit = FALSE, vcov = TRUE, ...)
  ## S3 method for class 'gpcmodel'
guesspar(object, alias = TRUE, vcov = TRUE, ...)
```

Arguments

object	a fitted model object whose guessing parameters should be extracted.
alias	logical. If TRUE (the default), the aliased parameters are included in the return vector (and in the variance-covariance matrix if vcov = TRUE). If FALSE, these parameters are removed. For raschmodels, rsmodels, pcmodels and gpcmodels, where all guessing parameters are fixed to 0, this means that an empty numeric vector and an empty variance-covariace matrix is returned if alias is FALSE.
logit	logical. If a nplmodel of type "3PL" or "4PL" model has been fit, the guessing parameters were estimated on the logit scale. If logit = FALSE, these estimates and the variance-covariance (if requested) are retransformed using the logistic function and the delta method.
vcov	logical. If TRUE (the default), the variance-covariance matrix of the guessing parameters is attached as attribute vcov.
...	further arguments which are currently not used.

Details

guesspar is both, a class to represent guessing parameters of item response models as well as a generic function. The generic function can be used to extract the guessing parameters of a given item response model.

For objects of class guesspar, several methods to standard generic functions exist: print, coef, vcov. coef and vcov can be used to extract the guessing parameters and their variance-covariance matrix without additional attributes.

Value

A named vector with guessing parameters of class guesspar and additional attributes model (the model name), alias (either TRUE or a named numeric vector with the aliased parameters not included in the return value), logit (indicating whether the estimates are on the logit scale or not), and vcov (the estimated and adjusted variance-covariance matrix).

See Also

personpar, itempar, threshpar, discrpar, upperpar

Examples

```
if(requireNamespace("mirt")) {

o <- options(digits = 3)

## load simulated data
data("Sim3PL", package = "psychotools")

## fit 2PL to data simulated under the 3PL
twoplmod <- nplmodel(Sim3PL$resp)

## extract the guessing parameters (all fixed at 0)
gp1 <- guesspar(twoplmod)

## fit 3PL to data simulated under the 3PL
threeplmod <- nplmodel(Sim3PL$resp, type = "3PL")

## extract the guessing parameters
gp2 <- guesspar(threeplmod)

## extract the standard errors
sqrt(diag(vcov(gp2)))

## extract the guessing parameters on the logit scale
gp2_logit <- guesspar(threeplmod, logit = TRUE)

## along with the delta transformed standard errors
sqrt(diag(vcov(gp2_logit)))

options(digits = o$digits)
}
```

---

infoplot *Information Plots for IRT Models*

Description

Base graphics plotting function for information plot visualization of IRT models.

Usage

```
infoplot(object, what = c("categories", "items", "test"),
  ref = NULL, items = NULL, names = NULL, layout = NULL, xlim = NULL,
  ylim = NULL, col = NULL, lty = NULL, lwd = NULL, main = NULL, legend = TRUE,
  xlab = "Latent trait", ylab = "Information", add = FALSE, ...)
```

Arguments

object	a fitted model object of class "raschmodel", "rsmodel", "pcmodel", "nplmodel" or "gpcmodel".
what	character, specifying the type of information to visualize.
ref	argument passed over to internal calls of predict.
items	character or numeric, specifying the items for which information curves should be visualized.
names	character, specifying labels for the items.
layout	matrix, specifying how the item or category information curves of different items should be arranged. If null and what is set to "items", the item information curves are overlayed within a single plot.
xlim, ylim	numeric, specifying the x and y axis limits.
col	character, specifying the colors of the test, item or category information curves.
lty	numeric, specifying the line type of the information curves.
lwd	numeric, specifying the line width of the information curves.
main	character, specifying the overall title of the plot.
legend	logical, specifying if a legend is drawn when multiple item information curves are overlayed. The labels in the legend correspond to the item names (which can be specified in the argument names).
xlab, ylab	character, specifying the x and y axis labels.
add	logical. If TRUE, new information curves are added to an existing plot. Only possible for a test or a single item information curve.
...	further arguments passed to internal calls of matplot.

Details

The information plot visualization illustrates the test, item or category information as a function of the ability parameter θ under a certain IRT model. Further details on the computation of the displayed information can be found on the help page of the function predict.pcmodel.

See Also

curveplot, regionplot, profileplot, piplot

Examples

```
## load verbal aggression data
data("VerbalAggression", package = "psychotools")

## fit Rasch and partial credit model to verbal aggression data
rmmod <- raschmodel(VerbalAggression$resp2)
pcmod <- pcmodel(VerbalAggression$resp)

## category information plots for all items under the dichotomous RM
plot(rmmod, type = "information", what = "categories")
```

```
## category information plots for all items under the PCM
plot(pcmod, type = "information", what = "categories")

## overlayed item information plots for the first six items of the
## data set under the PCM
plot(pcmod, type = "information", what = "items", items = 1:6)

## a comparison of the item information for the first six items under the
## dichotomous RM and the PCM
plot(pcmod, type = "information", what = "items", items = 1:6,
  xlim = c(-5, 5))
plot(rmmod, type = "information", what = "items", items = 1:6,
  lty = 2, add = TRUE)
legend(x = "topright", legend = c("PCM", "RM"), lty = 1:2, bty = "n")

## a comparison of the test information based on all items of the
## data set under the dichotomous RM and the PCM
plot(pcmod, type = "information", what = "test", items = 1:6, xlim = c(-5, 5))
plot(rmmod, type = "information", what = "test", items = 1:6, lty = 2,
  add = TRUE)
legend(x = "topright", legend = c("PCM", "RM"), lty = 1:2, bty = "n")

if(requireNamespace("mirt")) {
## fit 2PL to verbal aggression data
twoplmod <- nplmodel(VerbalAggression$resp2)

## category information plots for all items under the dichotomous 2PL
plot(twoplmod, type = "information", what = "categories")
}
```

`itempar` *Extract Item Parameters of Item Response Models*

Description

A class and generic function for representing and extracting the item parameters of a given item response model.

Usage

```
  itempar(object, ...)
  ## S3 method for class 'raschmodel'
itempar(object, ref = NULL, alias = TRUE, vcov = TRUE, ...)
  ## S3 method for class 'rsmodel'
itempar(object, ref = NULL, alias = TRUE, vcov = TRUE, ...)
  ## S3 method for class 'pcmodel'
itempar(object, ref = NULL, alias = TRUE, vcov = TRUE, ...)
  ## S3 method for class 'nplmodel'
```

```
itempar(object, ref = NULL, alias = TRUE, vcov = TRUE, ...)
  ## S3 method for class 'gpcmodel'
itempar(object, ref = NULL, alias = TRUE, vcov = TRUE, ...)
  ## S3 method for class 'btmodel'
itempar(object, ref = NULL, alias = TRUE, vcov = TRUE, log = FALSE, ...)
```

Arguments

object	a fitted model or tree object whose item parameters should be extracted.
ref	a vector of labels or position indices of item parameters or a contrast matrix which should be used as restriction/for normalization. If NULL (the default) for all models except models estimated via MML, all items are used (sum zero restriction). For models estimated via MML (nplmodels and gpcmodels), the parameters are by default identified via the distributional parameters of the person parameters (mean and variance of a normal distribution). Nevertheless, a restriction on the interval scale can be applied.
alias	logical. If TRUE (the default), the aliased parameter is included in the return vector (and in the variance-covariance matrix if vcov = TRUE). If FALSE, it is removed. If the restriction given in ref depends on several parameters, the first parameter of the restriction specified is (arbitrarily) chosen to be removed if alias is FALSE.
vcov	logical. If TRUE (the default), the (transformed) variance-covariance matrix of the item parameters is attached as attribute vcov. If FALSE, an NA-matrix is attached.
log	logical. Whether to return the estimated model parameters on the logit (TRUE) or preference scale (FALSE).
...	further arguments which are currently not used.

Details

itempar is both, a class to represent item parameters of item response models as well as a generic function. The generic function can be used to extract the item parameters of a given item response model.

For Rasch models and n-parameter logistic models, itempar returns the estimated item difficulty parameters $\hat{\beta}_j$ under the restriction specified in argument ref. For rating scale models, itempar returns computed item location parameters $\hat{\beta}_j$ under the restriction specified in argument ref. These are computed from the estimated item-specific parameters $\hat{\xi}_j$ (who mark the location of the first category of an item on the latent theta axis). For partial credit models and generalized partial credit models, itempar returns 'mean' absolute item threshold parameters, $\hat{\beta}_j = \frac{1}{p_j} \sum_{k=1}^{p_j} \hat{\delta}_{jk}$, i.e., a single parameter per item is returned which results as the mean of the absolute item threshold parameters $\hat{\delta}_{jk}$ of this item. Based upon these 'mean' absolute item threshold parameters $\hat{\beta}_j$, the restriction specified in argument ref is applied. For all models, the variance-covariance matrix of the returned item parameters is adjusted according to the multivariate delta rule.

For objects of class itempar, several methods to standard generic functions exist: print, coef, vcov. coef and vcov can be used to extract the estimated calculated item parameters and their variance-covariance matrix without additional attributes. Based on this Wald tests or confidence intervals can be easily computed, e.g., via confint.

Two-sample item-wise Wald tests for DIF in the item parameters can be carried out using the function anchortest.

Value

A named vector with item parameters of class itempar and additional attributes model (the model name), ref (the items or parameters used as restriction/for normalization), alias (either FALSE or a named character vector with the removed aliased parameter, and vcov (the adjusted covariance matrix of the estimates if vcov = TRUE or an NA-matrix otherwise).

See Also

personpar, threshpar, discrpar, guesspar, upperpar

Examples

```
o <- options(digits = 4)

## load verbal aggression data
data("VerbalAggression", package = "psychotools")

## fit a Rasch model to dichotomized verbal aggression data
raschmod <- raschmodel(VerbalAggression$resp2)

## extract item parameters with sum zero or use last two items as anchor
ip1 <- itempar(raschmod)
ip2a <- itempar(raschmod, ref = 23:24) # with position indices
ip2b <- itempar(raschmod, ref = c("S4WantShout", "S4DoShout")) # with item label

ip1
ip2a

all.equal(ip2a, ip2b)

## extract vcov
vc1 <- vcov(ip1)
vc2 <- vcov(ip2a)

## adjusted standard errors,
## smaller with more items used as anchors
sqrt(diag(vc1))
sqrt(diag(vc2))

## Wald confidence intervals
confint(ip1)
confint(ip2a)

options(digits = o$digits)
```

`itemresp` *Data Structure for Item Response Data*

Description

A class for representing data from questionnaires along with methods for many generic functions.

Usage

```
itemresp(data, mscale = NULL, labels = NULL, names = NULL)
```

Arguments

`data`	matrix or data frame. A matrix or data frame with integer values or factors where the rows correspond to subjects and the columns to items. See below for details.
`mscale`	integer or character. A list of vectors (either integer or character) giving the measurement scale. See below for details. By default guessed from `data`.
`labels`	character. A vector of character labels for the items. By default, the column names of `data` are used or, if these are not available, the string `"item"` along with numbers 1, 2, ... is used.
`names`	character. A vector of names (or IDs) for the subjects. By default, no subject names are used.

Details

`itemresp` is designed for item response data of n subjects for k items.

The item responses should be coded in a matrix `data` with n rows (subjects) and k columns (items). Alternatively, `data` can be a data frame with n rows (subjects) and k variables (items), which can be either factors or integer valued vectors.

`mscale` provides the underlying measurement scale either as integer or character vector(s). If all items are measured on the same scale, `mscale` can be a vector. Alternatively, it can be provided as a named list of vectors for each item. If the list contains one unnamed element, this element will be used as the measurement scale for items that have not been named. Integers or characters not present in `mscale` but in `data` will be replaced by `NA`. All items must be measured with at least 2 categories. By default, `mscale` is set to the full range of observed values for all integer items (see example below) and the corresponding levels for all factor items in `data`.

Methods to standard generic functions include: `str`, `length` (number of subjects), `dim` (number of subjects and items), `is.na` (only `TRUE` if all item responses are `NA` for a subject), `print` (see `print.itemresp` for details), `summary` and `plot` (see `summary.itemresp` for details), subsetting via `[` and `subset` (see `subset.itemresp` for details), `is.itemresp` and various coercion functions to other classes (see `as.list.itemresp` for details).

Extracting/replacing properties is available through: `labels` for the item labels, `mscale` for the measurement scale, `names` for subject names/IDs.

Value

itemresp returns an object of class "itemresp" which is a matrix (data transformed to integers 0, 1, ...) plus an attribute "mscale" as a named list for each item (after being checked and potentially suitably coerced or transformed to all integer or all character).

See Also

print.itemresp, summary.itemresp, as.list.itemresp, subset.itemresp

Examples

```
## binary responses to three items, coded as matrix
x <- cbind(c(1, 0, 1, 0), c(1, 0, 0, 0), c(0, 1, 1, 1))
## transformed to itemresp object
xi <- itemresp(x)

## printing (see also ?print.itemresp)
print(xi)
print(xi, labels = TRUE)

## subsetting/indexing (see also ?subset.itemresp)
xi[2]
xi[c(TRUE, TRUE, FALSE, FALSE)]
subset(xi, items = 1:2)
dim(xi)
length(xi)

## summary/visualization (see also ?summary.itemresp)
summary(xi)
plot(xi)

## query/set measurement scale labels
## extract mscale (tries to collapse to vector)
mscale(xi)
## extract as list
mscale(xi, simplify = FALSE)
## replacement by list
mscale(xi) <- list(item1 = c("no", "yes"),
  item2 = c("nay", "yae"), item3 = c("-", "+"))
xi
mscale(xi)
## replacement with partially named list plus default
mscale(xi) <- list(item1 = c("n", "y"), 0:1)
mscale(xi)
## replacement by vector (if number of categories constant)
mscale(xi) <- c("-", "+")
mscale(xi, simplify = FALSE)

## query/set item labels and subject names
labels(xi)
labels(xi) <- c("i1", "i2", "i3")
names(xi)
```

```
names(xi) <- c("John", "Joan", "Jen", "Jim")
print(xi, labels = TRUE)

## coercion (see also ?as.list.itemresp)
## to integer matrix
as.matrix(xi)
## to data frame with single itemresp column
as.data.frame(xi)
## to list of factors
as.list(xi)
## to data frame with factors
as.list(xi, df = TRUE)


## polytomous responses with missing values and unequal number of
## categories in a data frame
d <- data.frame(
  q1 = c(-2, 1, -1, 0, NA, 1, NA),
  q2 = c(3, 5, 2, 5, NA, 2, 3),
  q3 = factor(c(1, 2, 1, 2, NA, 3, 2), levels = 1:3,
    labels = c("disagree", "neutral", "agree")))
di <- itemresp(d)
di

## auto-completion of mscale: full range (-2, ..., 2) for q1, starting
## from smallest observed (negative) value (-2) to the same (positive)
## value (2), full (positive) range for q2, starting from smallest
## observed value (2) to largest observed value (5), missing category of
## 4 is detected, for q3 given factor levels are used
mscale(di)

## set mscale for q2 and add category 1, q1 and q3 are auto-completed:
di <- itemresp(d, mscale = list(q2 = 1:5))

## is.na.itemresp - only true for observation 5 (all missing)
is.na(di)

## illustration for larger data set
data("VerbalAggression", package = "psychotools")
r <- itemresp(VerbalAggression$resp[, 1:12])
str(r)
head(r)
plot(r)
summary(r)
prop.table(summary(r), 1)

## dichotomize response
r2 <- r
mscale(r2) <- c(0, 1, 1)
plot(r2)

## transform to "likert" package
if(require("likert")) {
```

```
lik <- likert(as.data.frame(as.list(r)))
lik
plot(lik)
}
```

`labels<-` *Set Labels*

Description

A generic function for setting labels for an object.

Usage

```
labels(object) <- value
```

Arguments

`object`	an object.
`value`	an object.

Examples

```
## method for "paircomp" data
pc <- paircomp(rbind(
  c(1,  1,  1), # a > b, a > c, b > c
  c(1,  1, -1), # a > b, a > c, b < c
  c(1, -1, -1), # a > b, a < c, b < c
  c(1,  1,  1)))
labels(pc)
labels(pc) <- c("ah", "be", "ce")
pc
```

`MathExam14W` *Mathematics 101 Exam Results*

Description

Responses of 729 students to 13 items in a written exam of introductory mathematics along with several covariates.

Usage

```
data("MathExam14W")
```

Format

A data frame containing 729 observations on 9 variables.

solved Item response matrix (of class `itemresp`) with values 1/0 coding solved correctly/other.

credits Item response matrix (of class `itemresp`) with values 2/1/0 coding solved correctly/incorrectly/not attempted.

nsolved Integer. The number of items solved correctly.

tests Integer. The number of online test exercises solved correctly prior to the written exam.

gender Factor indicating gender.

study Factor indicating two different types of business/economics degrees. Either the 3-year bachelor program (571) or the 4-year diploma program (155).

semester Integer. The number of semesters enrolled in the given university program.

attempt Factor. The number of times the course/exam has been attempted (including the current attempt).

group Factor indicating whether the students were in the first or second batch (with somewhat different items) in the exam.

Details

The data provides individual end-term exam results from a Mathematics 101 course for first-year business and economics students at Universität Innsbruck. The format of the course comprised biweekly online tests (26 numeric exercises, conducted in OpenOLAT) and a written exam at the end of the semester (13 single-choice exercises with five answer alternatives). The course covers basics of analysis, linear algebra, financial mathematics, and probability calculus (where the latter is not assessed in this exam).

In this exam, 729 students participated (out of 941 registered in the course). To avoid cheating, all students received items with essentially the same questions but different numbers (using the exams infrastructure of Zeileis et al. 2014). Also, due to the large number of students two groups of students had to be formed which received partially different items. The items which differed (namely 1, 5, 6, 7, 8, 9, 11, 12) varied in the setup/story, but not in the mathematical skills needed to solve the exercises. Prior to the exam, the students could select themselves either into the first group (early in the morning) or the second group (starting immediately after the end of the first group).

Correctly solved items yield 100 percent of the associated points. Items without correct solution can either be unanswered (0 percent) or receive an incorrect answer (minus 25 percent) to discourage random guessing. In the examples below, the items are mostly only considered as binary. Typically, students with 8 out of 13 correct answers passed the course.

Source

Department of Statistics, Universität Innsbruck

References

Zeileis A, Umlauf N, Leisch F (2014). Flexible Generation of E-Learning Exams in R: Moodle Quizzes, OLAT Assessments, and Beyond. *Journal of Statistical Software*, **58**(1), 1–36. doi:10.18637/jss.v058.i01

See Also

itemresp, raschmodel, pcmodel, anchortest

Examples

```
## load data and exclude extreme scorers
data("MathExam14W", package = "psychotools")
MathExam14W <- transform(MathExam14W,
  points = 2 * nsolved - 0.5 * rowSums(credits == 1)
)
me <- subset(MathExam14W, nsolved > 0 & nsolved < 13)


## item response data:
## solved (correct/other) or credits (correct/incorrect/not attempted)
par(mfrow = c(1, 2))
plot(me$solved)
plot(me$credits)

## PCA
pr <- prcomp(me$solved, scale = TRUE)
names(pr$sdev) <- 1:10
plot(pr, main = "", xlab = "Number of components")
biplot(pr, col = c("transparent", "black"), main = "",
  xlim = c(-0.065, 0.005), ylim = c(-0.04, 0.065))


## points achieved (and 50% threshold)
par(mfrow = c(1, 1))
hist(MathExam14W$points, breaks = -4:13 * 2 + 0.5,
  col = "lightgray", main = "", xlab = "Points")
abline(v = 12.5, lwd = 2, col = 2)


## Rasch and partial credit model
ram <- raschmodel(me$solved)
pcm <- pcmodel(me$credits)

## various types of graphics displays
plot(ram, type = "profile")
plot(pcm, type = "profile", add = TRUE, col = "blue")
plot(ram, type = "piplot")
plot(pcm, type = "piplot")
plot(ram, type = "region")
plot(pcm, type = "region")
plot(ram, type = "curves")
plot(pcm, type = "curves")



## test for differential item function with automatic anchoring
## passing vs. not passing students
```

```
at1 <- anchortest(solved ~ factor(nsolved <= 7), data = me,
  adjust = "single-step")
at1
plot(at1$final_tests)
## -> "good" students discriminate somewhat more
## (quad/payflow/lagrange are slightly more difficult)

## group 1 vs. group 2
at2 <- anchortest(solved ~ group, data = me, adjust = "single-step")
at2
plot(at2$final_tests)
## -> quad/payflow/planning easier for group 1
## -> hesse slightly easier for group 2

## bring out differences between groups 1 and 2
## by (anchored) item difficulty profiles
ram1 <- raschmodel(subset(me, group == "1")$solved)
ram2 <- raschmodel(subset(me, group == "2")$solved)
plot(ram1, parg = list(ref = at2$anchor_items), ylim = c(-2, 3))
plot(ram2, parg = list(ref = at2$anchor_items), add = TRUE, col = "blue")
legend("topleft", c("Group 1", "Group 2"), pch = 21,
  pt.bg = c("lightgray", "blue"), bty = "n")
```

`MemoryDeficits` *Memory Deficits in Psychiatric Patients*

Description

Response frequencies of 96 patients who took part in a pair-clustering experiment to assess their memory deficits.

Usage

```
data("MemoryDeficits")
```

Format

A data frame containing 576 observations on 7 variables.

ID Participant ID.

group Factor with four levels specifying patient or control group of participant.

trial Trial number from 1 to 6.

E1 Number of pairs recalled adjacently.

E2 Number of pairs recalled non-adjacently.

E3 Number of single pair members recalled.

E4 Number of non-recalled pairs.

Details

Riefer, Knapp, Batchelder, Bamber and Manifold (2002) report a study on memory deficits in schizophrenic (n = 29) and organic alcoholic (n = 21) patients who were compared to two matched control groups (n = 25, n = 21). Participants were presented with 20 pairs of semantically related words. In a later memory test, they freely recalled the presented words. This procedure was repeated for a total of six study and test trials. Responses were classified into four categories: both words in a pair are recalled adjacently (E1) or non-adjacently (E2), one word in a pair is recalled (E3), neither word in a pair is recalled (E4).

Source

The data were made available by William H. Batchelder.

References

Riefer DM, Knapp BR, Batchelder WH, Bamber D, Manifold V (2002). Cognitive Psychometrics: Assessing Storage and Retrieval Deficits in Special Populations with Multinomial Processing Tree Models. *Psychological Assessment*, **14**, 184–201.

Examples

```
data("MemoryDeficits", package = "psychotools")
aggregate(cbind(E1, E2, E3, E4) ~ trial + group, MemoryDeficits, sum)
```

---

mptmodel *Multinomial Processing Tree (MPT) Model Fitting Function*

Description

`mptmodel` is a basic fitting function for multinomial processing tree (MPT) models.

Usage

```
mptmodel(y, weights = NULL, spec, treeid = NULL,
  optimargs = list(control = list(reltol = .Machine$double.eps^(1/1.2),
                                  maxit = 1000),
                   init = NULL),
  start = NULL, vcov = TRUE, estfun = FALSE, ...)
```

Arguments

`y`	matrix of response frequencies.
`weights`	an optional vector of weights (interpreted as case weights).
`spec`	an object of class `mptspec`: typically result of a call to `mptspec`. A symbolic description of the model to be fitted.
`treeid`	a factor that identifies each tree in a joint multinomial model.

optimargs	a list of arguments passed to the optimization function (optim).
start	a vector of starting values for the parameter estimates between zero and one.
vcov	logical. Should the estimated variance-covariance be included in the fitted model object?
estfun	logical. Should the empirical estimating functions (score/gradient contributions) be included in the fitted model object?
...	further arguments passed to functions.

Details

mptmodel provides a basic fitting function for multinomial processing tree (MPT) models, intended as a building block for fitting MPT trees in the **psychotree** package. While mptmodel is intended for individual response frequencies, the **mpt** package provides functions for aggregate data.

MPT models are specified using the mptspec function. See the documentation in the **mpt** package for details.

mptmodel returns an object of class "mptmodel" for which several basic methods are available, including print, plot, summary, coef, vcov, logLik, estfun and predict.

Value

mptmodel returns an S3 object of class "mptmodel", i.e., a list with components as follows:

y	a matrix with the response frequencies,
coefficients	estimated parameters (for extraction, the coef function is preferred),
loglik	log-likelihood of the fitted model,
npar	number of estimated parameters,
weights	the weights used (if any),
nobs	number of observations (with non-zero weights),
ysum	the aggregate response frequencies,
fitted, goodness.of.fit, ...	see mpt in the **mpt** package.

See Also

btmodel, pcmodel, gpcmodel, rsmodel, raschmodel, nplmodel, mptspec, the **mpt** package

Examples

```
o <- options(digits = 4)

## data
data("SourceMonitoring", package = "psychotools")

## source-monitoring MPT model
mpt1 <- mptmodel(SourceMonitoring$y, spec = mptspec("SourceMon"))
summary(mpt1)
```

```
plot(mpt1)

options(digits = o$digits)
```

---

mscale *Extract/Replace Measurement Scale*

Description

Generic functions for extracting and replacing the measurement scale from an object.

Usage

```
mscale(object, ...)
mscale(object) <- value
```

Arguments

object	an object.
...	arguments passed to methods.
value	an object describing the measurement scale.

Examples

```
## methods for "paircomp" data
pc <- paircomp(rbind(
  c(2,  1,  0),
  c(1,  1, -1),
  c(1, -2, -1),
  c(0,  0,  0)))
pc

## extract
mscale(pc)

## replace (collapse to >/=/< scale)
mscale(pc) <- sign(mscale(pc))
pc


## similar for "itemresp" data
ir <- itemresp(cbind(
  c(-1, 0, 1, 1, 0),
  c(0, 1, 2, 1, 2),
  c(1, 2, 1, 1, 3)))
ir

## extract
mscale(ir)
```

```
## replace (single scale for all items)
mscale(ir) <- 1:3
ir
```

---

nplmodel *Parametric Logistic Model (n-PL) Fitting Function*

Description

nplmodel is a basic fitting function for n-PL type parametric logistic IRT models (2PL, 3PL, 3PLu, 4PL, Rasch/1PL), providing a wrapper around mirt and multipleGroup relying on marginal maximum likelihood (MML) estimation via the standard EM algorithm.

Usage

```
nplmodel(y, weights = NULL, impact = NULL,
  type = c("2PL", "3PL", "3PLu", "4PL", "1PL", "RM"),
  grouppars = FALSE, vcov = TRUE,
  start = NULL, method = "BFGS", maxit = 500, reltol = 1e-5, ...)
```

Arguments

y	item response object that can be coerced (via as.matrix) to a numeric matrix with scores 0, 1. Typically, either already a matrix, data frame, or dedicated object of class itemresp.
weights	an optional vector of weights (interpreted as case weights).
impact	an optional factor allowing for grouping the subjects (rows). If specified, a multiple-group model is fitted to account for impact (see details below). By default, no impact is modelled, i.e., a single-group model is used.
type	character string, specifying the type of parametric logistic IRT model to be estimated (see details below).
grouppars	logical. Should the estimated distributional group parameters of a multiple group model be included in the model parameters?
vcov	logical or character specifying the type of variance-covariance matrix (if any) computed for the final model. The default vcov = TRUE corresponds to vcov = "Oakes", see mirt for further options. If set to vcov = FALSE (or vcov = "none"), vcov() will return a matrix of NAs only.
start	an optional vector or list of starting values (see examples below).
method, maxit, reltol	control parameters for the optimizer employed by mirt for the EM algorithm.
...	further arguments passed to mirt or multipleGroup, respectively.

Details

nplmodel (plmodel for backward compatibility with earlier **psychotools** versions) provides a basic fitting function for n-PL type parametric logistic IRT models (2PL, 3PL, 3PLu, 4PL, Rasch/1PL) providing a wrapper around mirt and multipleGroup relying on MML estimation via the standard EM algorithm (Bock & Aitkin, 1981). Models are estimated under the slope/intercept parametrization, see e.g. Chalmers (2012). The probability of person i 'solving' item j is modelled as:

$$P(X_{ij} = 1|\theta_i, a_j, d_j, g_j, u_j) = g_j + \frac{(u_j - g_j)}{1 + \exp\left(-(a_j\theta_i + d_j)\right)}$$

A reparametrization of the intercepts to the classical IRT parametrization, $b_j = -\frac{d_j}{a_j}$, is provided via the corresponding itempar method.

If an optional impact variable is supplied, a multiple-group model of the following form is being fitted: Item parameters are fixed to be equal across the whole sample. For the first group of the impact variable the person parameters are fixed to follow the standard normal distribution. In the remaining impact groups, the distributional parameters (mean and variance of a normal distribution) of the person parameters are estimated freely. See e.g. Baker & Kim (2004, Chapter 11), Debelak & Strobl (2019), or Schneider et al. (2022) for further details. To improve convergence of the model fitting algorithm, the first level of the impact variable should always correspond to the largest group. If this is not the case, levels are re-ordered internally.

If grouppars is set to TRUE the freely estimated distributional group parameters (if any) are returned as part of the model parameters.

By default, type is set to "2PL". Therefore, all so-called guessing parameters are fixed at 0 and all upper asymptotes are fixed at 1. "3PL" results in all upper asymptotes being fixed at 1 and "3PLu" results in all all guessing parameters being fixed at 0. "4PL" results in a full estimated model as specified above. Finally, if type is set to "1PL" (or equivalently "RM"), an MML-estimated Rasch model is being fitted. This means that all slopes are restricted to be equal across all items, all guessing parameters are fixed at 0 and all upper asymptotes are fixed at 1.

Note that internally, the so-called guessing parameters and upper asymptotes are estimated on the logit scale (see also mirt). Therefore, most of the basic methods below include a logit argument, which can be set to TRUE or FALSE allowing for a retransformation of the estimates and their variance-covariance matrix (if requested) using the logistic function and the delta method if logit = FALSE.

nplmodel returns an object of class "nplmodel" for which several basic methods are available, including print, plot, summary, coef, vcov, logLik, estfun, discrpar, itempar, threshpar, guesspar, upperpar, and personpar.

Finally, if type is set to "1PL", a Rasch model is estimated. Here, a common slope parameter is estimated for all items, whereas the person parameters are assumed to follow a standard normal distribution. Please note that this variant of the Rasch model differs from the one used by mirt, which sets all slope parameters to 1, and estimates the variance of the person parameters instead. Both variants are mathematically equivalent and hence should lead to the same intercept parameter estimates. For numerical reasons, nplmodel and mirt can lead to slightly different item parameter estimates, though, under their respective default settings, in particular when some items are very easy or very difficult and the common slope parameter is large. A distinct advantage of the variant used by nplmodel is that it allows a direct comparison of the slope and intercept parameters with that estimated in more complex IRT models, such as the 2PL model.

Value

`nplmodel` returns an S3 object of class `"nplmodel"`, i.e., a list of the following components:

`coefficients`	estimated model parameters in slope/intercept parametrization,
`vcov`	covariance matrix of the model parameters,
`data`	modified data, used for model-fitting, i.e., without observations with zero weight,
`items`	logical vector of length `ncol(y)`, indicating which items were used during estimation,
`n`	number of observations (with non-zero weights),
`n_org`	original number of observations in `y`,
`weights`	the weights used (if any),
`na`	logical indicating whether the data contain `NA`s,
`impact`	either `NULL` or the supplied `impact` variable with the levels reordered in decreasing order (if this has not been the case prior to fitting the model),
`loglik`	log-likelihood of the fitted model,
`df`	number of estimated (more precisely, returned) model parameters,
`code`	convergence code from `mirt`,
`iterations`	number of iterations used by `mirt`,
`reltol`	convergence threshold passed to `mirt`,
`grouppars`	the logical `grouppars` value,
`type`	the `type` of model restriction specified,
`mirt`	the `mirt` object fitted internally,
`call`	original function call.

References

Baker FB, Kim SH (2004). *Item Response Theory: Parameter Estimation Techniques*. Chapman & Hall/CRC, Boca Raton.

Bock RD, Aitkin M (1981). Marginal Maximum Likelihood Estimation of Item Parameters: Application of an EM Algorithm. *Psychometrika*, **46**(4), 443–459.

Chalmers RP (2012). mirt: A Multidimensional Item Response Theory Package for the R Environment. *Journal of Statistical Software*, **48**(6), 1–29. doi:10.18637/jss.v048.i06

Debelak R, Strobl C (2019). Investigating Measurement Invariance by Means of Parameter Instability Tests for 2PL and 3PL Models. *Educational and Psychological Measurement*, **79**(2), 385–398. doi:10.1177/0013164418777784

Schneider L, Strobl C, Zeileis A, Debelak R (2022). An R Toolbox for Score-Based Measurement Invariance Tests in IRT Models. *Behavior Research Methods*, forthcoming. doi:10.3758/s13428-021016890

See Also

`raschmodel`, `gpcmodel`, `rsmodel`, `pcmodel`, `btmodel`

Examples

```
if(requireNamespace("mirt")) {

o <- options(digits = 4)

## mathematics 101 exam results
data("MathExam14W", package = "psychotools")

## 2PL
twopl <- nplmodel(y = MathExam14W$solved)
summary(twopl)

## how to specify starting values as a vector of model parameters
st <- coef(twopl)
twopl <- nplmodel(y = MathExam14W$solved, start = st)
## or a list containing a vector of slopes and a vector of intercepts
set.seed(0)
st <- list(a = rlnorm(13, 0, 0.0625), d = rnorm(13, 0, 1))
twopl <- nplmodel(y = MathExam14W$solved, start = st)

## visualizations
plot(twopl, type = "profile")
plot(twopl, type = "regions")
plot(twopl, type = "piplot")
plot(twopl, type = "curves", xlim = c(-6, 6))
plot(twopl, type = "information", xlim = c(-6, 6))
## visualizing the IRT parametrization
plot(twopl, type = "curves", xlim = c(-6, 6), items = 1)
abline(v = itempar(twopl)[1])
abline(h = 0.5, lty = 2)

## 2PL accounting for gender impact
table(MathExam14W$gender)
mtwopl <- nplmodel(y = MathExam14W$solved, impact = MathExam14W$gender,
  grouppars = TRUE)
summary(mtwopl)
plot(mtwopl, type = "piplot")
## specifying starting values as a vector of model parameters, note that in
## this example impact is being modelled and therefore grouppars must be TRUE
## to get all model parameters
st <- coef(mtwopl)
mtwopl <- nplmodel(y = MathExam14W$solved, impact = MathExam14W$gender,
  start = st)
## or a list containing a vector of slopes, a vector of intercepts and a vector
## of means and a vector of variances as the distributional group parameters
set.seed(1)
st <- list(a = rlnorm(13, 0, 0.0625), d = rnorm(13, 0, 1), m = 0, v = 1)
mtwopl <- nplmodel(y = MathExam14W$solved, impact = MathExam14W$gender,
  start = st)

## MML estimated Rasch model (1PL)
rm <- nplmodel(y = MathExam14W$solved, type = "1PL")
```

```
summary(rm)

options(digits = o$digits)
}
```

`PairClustering` *Pair Clustering Data in Klauer (2006)*

Description

Response frequencies of 63 participants who took part in a pair-clustering experiment.

Usage

```
data("PairClustering")
```

Format

A data frame containing 126 observations on 8 variables.

ID Participant ID.

trial Trial number, 1 or 2.

E1 Number of pairs recalled adjacently.

E2 Number of pairs recalled non-adjacently.

E3 Number of single pair members recalled.

E4 Number of non-recalled pairs.

F1 Number of recalled singleton words.

F2 Number of non-recalled singleton words.

Details

Klauer (2006) reports a pair-clustering experiment with 63 participants, who were presented with ten pairs of related words and five unrelated singleton words. In a later memory test, they freely recalled the presented words. This procedure was repeated for two study and test trials. For pairs, responses were classified into four categories: both words in a pair are recalled adjacently (E1) or non-adjacently (E2), one word in a pair is recalled (E3), neither word in a pair is recalled (E4); for singletons, into two categories: word recalled (F1), word not recalled (F2).

Source

Stahl C, Klauer KC (2007). HMMTree: A Computer Program for Latent-Class Hierarchical Multinomial Processing Tree Models. *Behavior Research Methods*, **39**, 267–273.

References

Klauer KC (2006). Hierarchical Multinomial Processing Tree Models: A Latent-Class Approach. *Psychometrika*, **71**, 1–31.

Examples

```
data("PairClustering", package = "psychotools")
aggregate(cbind(E1, E2, E3, E4, F1, F2) ~ trial, PairClustering, sum)
```

`paircomp` *Data Structure for Paired Comparisons*

Description

A class for representing data from paired comparison experiments along with methods for many generic functions.

Usage

```
paircomp(data,
  labels = NULL, mscale = NULL, ordered = FALSE, covariates = NULL)
```

Arguments

`data`	matrix. A matrix with integer values where the rows correspond to subjects and the columns to paired comparisons between objects. See below for details.
`labels`	character. A vector of character labels for the objects. By default a suitable number of `letters` is used.
`mscale`	integer. A vector of integers giving the measurement scale. See below for details. By default guessed from `data`.
`ordered`	logical. Does `data` contain both orderings of each comparison?
`covariates`	data.frame. An optional data.frame with object covariates, i.e., it must have the same number of rows as the length of `labels`. May be `NULL` (default).

Details

`paircomp` is designed for holding paired comparisons of k objects measured for n subjects.

The comparisons should be coded in an integer matrix `data` with n rows (subjects) and $\binom{k}{2}$ columns (unless `ordered = TRUE`, see below). The columns must be ordered so that objects are sequentially compared with all previous objects, i.e.: 1:2, 1:3, 2:3, 1:4, 2:4, 3:4, etc. Each column represents the results of a comparison for two particular objects. Positive values signal that the first object was preferred, negative values that the second was preferred, zero signals no preference. Larger absolute values signal stronger preference.

`mscale` provides the underlying measurement scale. It must be a symmetric sequence of integers of type `(-i):i` where `i` must be at least `1`. However, it may exclude `0` (i.e., forced choice).

If `ordered = TRUE`, the order of comparison matters and thus `data` is assumed to have twice as many columns. The second half of columns then corresponds to the comparisons 2:1, 3:1, 3:2, 4:1, 4:2, 4:3, etc.

Value

paircomp returns an object of class "paircomp" which is a matrix (essentially data) with all remaining arguments of paircomp as attributes (after being checked and potentially suitably coerced or transformed).

See Also

subset.paircomp, print.paircomp

Examples

```
## a simple paired comparison
pc <- paircomp(rbind(
  c(1,  1,  1), # a > b, a > c, b > c
  c(1,  1, -1), # a > b, a > c, b < c
  c(1, -1, -1), # a > b, a < c, b < c
  c(1,  1,  1)))

## basic methods
pc
str(pc)
summary(pc)
pc[2:3]
c(pc[2], pc[c(1, 4)])

## methods to extract/set attributes
labels(pc)
labels(pc) <- c("ah", "be", "ce")
pc
mscale(pc)
covariates(pc)
covariates(pc) <- data.frame(foo = factor(c(1, 2, 2), labels = c("foo", "bar")))
covariates(pc)
names(pc)
names(pc) <- LETTERS[1:4]
pc

## reorder() and subset() both select a subset of
## objects and/or reorders the objects
reorder(pc, c("ce", "ah"))


## include paircomp object in a data.frame
## (i.e., with subject covariates)
dat <- data.frame(
  x = rnorm(4),
  y = factor(c(1, 2, 1, 1), labels = c("hansi", "beppi")))
dat$pc <- pc
dat


## formatting with long(er) labels and extended scale
```

```
pc2 <- paircomp(rbind(
  c(4,  1,  0),
  c(1,  2, -1),
  c(1, -2, -1),
  c(0,  0,  -3)),
  labels = c("Nordrhein-Westfalen", "Schleswig-Holstein", "Baden-Wuerttemberg"))
## default: abbreviate
print(pc2)
print(pc2, abbreviate = FALSE)
print(pc2, abbreviate = FALSE, width = FALSE)


## paired comparisons with object covariates
pc3 <- paircomp(rbind(
  c(2,  1,  0),
  c(1,  1, -1),
  c(1, -2, -1),
  c(0,  0,  0)),
  labels = c("New York", "Rio", "Tokyo"),
  covariates = data.frame(hemisphere = factor(c(1, 2, 1), labels = c("North", "South"))))
covariates(pc3)
```

---

pcmodel *Partial Credit Model Fitting Function*

Description

pcmodel is a basic fitting function for partial credit models.

Usage

```
pcmodel(y, weights = NULL, nullcats = c("keep", "downcode", "ignore"),
  start = NULL, reltol = 1e-10, deriv = c("sum", "diff"),
  hessian = TRUE, maxit = 100L, full = TRUE, ...)
```

Arguments

y	item response object that can be coerced (via as.matrix) to a numeric matrix with scores 0, 1, ... Typically, either already a matrix, data frame, or dedicated object of class itemresp.
weights	an optional vector of weights (interpreted as case weights).
deriv	character. If "sum" (the default), the first derivatives of the elementary symmetric functions are calculated with the sum algorithm. Otherwise ("diff") the difference algorithm (faster but numerically unstable) is used.
nullcats	character string, specifying how items with null categories (i.e., categories not observed) should be treated (see details below).
start	an optional vector of starting values.

`hessian` logical. Should the Hessian of the final model be computed? If set to `FALSE`, the `vcov` method can only return `NA`s and consequently no standard errors or tests are available in the `summary`.

`reltol, maxit, ...` further arguments passed to `optim`.

`full` logical. Should a full model object be returned? If set to `FALSE`, no variance-covariance matrix and no matrix of estimating functions are computed.

Details

`pcmodel` provides a basic fitting function for partial credit models, intended as a building block for fitting partial credit trees. It estimates the partial credit model suggested by Masters (1982) under the cumulative threshold parameterization, i.e., the item-category parameters $\eta_{jk} = \sum_{\ell=1}^{k} \delta_{jk}$ are estimated by the the function `pcmodel`.

Null categories, i.e., categories which have not been used, can be problematic when estimating a partial credit model. Several strategies have been suggested to cope with null categories. `pcmodel` allows to select from three possible strategies via the argument `nullcats`. If `nullcats` is set to `"keep"` (the default), the strategy suggested by Wilson & Masters (1993) is used to handle null categories. That basically means that the integrity of the response framework is maintained, i.e., no category scores are changed. This is not the case, when `nullcats` is set to `"downcode"`. Then all categories above a null category are shifted down to close the existing gap. In both cases (`"keep"` and `"downcode"`) the number of estimated parameters is reduced by the number of null categories. When `nullcats` is set to `"ignore"`, these are literally ignored and a threshold parameter is estimated during the optimization nevertheless. This strategy is used by the related package **eRm** when fitting partial credit models via `eRm::PCM`.

`pcmodel` returns an object of class `"pcmodel"` for which several basic methods are available, including `print`, `plot`, `summary`, `coef`, `vcov`, `logLik`, `discrpar`, `itempar`, `estfun`, `threshpar`, and `personpar`.

Value

`pcmodel` returns an S3 object of class `"pcmodel"`, i.e., a list the following components:

`coefficients` a named vector of estimated item-category parameters (without the first item-category parameter which is constrained to 0),

`vcov` covariance matrix of the parameters in the model,

`data` modified data, used for model-fitting, i.e., cleaned for items without variance, centralized so that the first category is zero for all items, treated null categories as specified via argument `"nullcats"` and without observations with zero weight. Be careful, this is different than for objects of class `"raschmodel"` or `"btmodel"`, where `data` contains the *original* data,

`items` logical vector of length `ncol(dat)`, indicating which items have variance (`TRUE`), i.e., are identified and have been used for the estimation or not (`FALSE`),

`categories` list of length `ncol(y)`, containing integer vectors starting from one to the number of categories minus one per item,

`n` number of observations (with non-zero weights),

n_org	original number of observations in y,
weights	the weights used (if any),
na	logical indicating whether the data contain NAs,
nullcats	either NULL or, if there have been null categories, a list of length ncol(y) with logical vectors specifying which categories are null categories (TRUE) or not (FALSE),
esf	list of elementary symmetric functions and their derivatives for estimated parameters,
loglik	log-likelihood of the fitted model,
df	number of estimated parameters,
code	convergence code from optim,
iterations	number of iterations used by optim,
reltol	tolerance passed to optim,
call	original function call.

References

Masters GN (1992). A Rasch Model for Partial Credit Scoring. *Psychometrika*, **47**(2), 149–174.

Wilson M, Masters GN (1993). The Partial Credit Model and Null Categories. *Psychometrika*, **58**(1), 87–99.

See Also

gpcmodel, rsmodel, raschmodel, nplmodel, btmodel

Examples

```
o <- options(digits = 4)

## Verbal aggression data
data("VerbalAggression", package = "psychotools")

## Partial credit model for the other-to-blame situations
pcm <- pcmodel(VerbalAggression$resp[, 1:12])
summary(pcm)

## visualizations
plot(pcm, type = "profile")
plot(pcm, type = "regions")
plot(pcm, type = "piplot")
plot(pcm, type = "curves")
plot(pcm, type = "information")

## Get data of situation 1 ('A bus fails to
## stop for me') and induce a null category in item 2.
pcd <- VerbalAggression$resp[, 1:6, drop = FALSE]
pcd[pcd[, 2] == 1, 2] <- NA
```

```
## fit pcm to these data, comparing downcoding and keeping strategy
pcm_va_keep  <- pcmodel(pcd, nullcats = "keep")
pcm_va_down  <- pcmodel(pcd, nullcats = "downcode")

plot(x = coef(pcm_va_keep), y = coef(pcm_va_down),
     xlab = "Threshold Parameters (Keeping)",
     ylab = "Threshold Parameters (Downcoding)",
     main = "Comparison of two null category strategies (I2 with null category)",
     pch = rep(as.character(1:6), each = 2)[-3])
abline(b = 1, a = 0)

options(digits = o$digits)
```

---

personpar *Extract Person Parameters of Item Response Models*

Description

A class and generic function for representing and estimating the person parameters of a given item response model.

Usage

```
personpar(object, ...)
## S3 method for class 'raschmodel'
personpar(object, personwise = FALSE, ref = NULL,
  vcov = TRUE, interval = NULL, tol = 1e-8, ...)
## S3 method for class 'rsmodel'
personpar(object, personwise = FALSE, ref = NULL,
  vcov = TRUE, interval = NULL, tol = 1e-8, ...)
## S3 method for class 'pcmodel'
personpar(object, personwise = FALSE, ref = NULL,
  vcov = TRUE, interval = NULL, tol = 1e-8, ...)
## S3 method for class 'nplmodel'
personpar(object, personwise = FALSE, vcov = TRUE,
  interval = NULL, tol = 1e-6, method = "EAP", ...)
## S3 method for class 'gpcmodel'
personpar(object, personwise = FALSE, vcov = TRUE,
  interval = NULL, tol = 1e-6, method = "EAP", ...)
```

Arguments

object	a fitted model object for which person parameters should be returned/estimated.
personwise	logical. Should the distributional parameters of the latent person ability distribution be computed (default) or the person-wise (individual) person parameters? See below for details.

ref	a vector of labels or position indices of item parameters or a contrast matrix which should be used as restriction/for normalization. This argument will be passed over to internal calls of itempar.
vcov	logical. Should a covariance matrix be returned/estimated for the person parameter estimates? See also details below.
interval	numeric vector of length two, specifying an interval for uniroot or fscores to calculate the person parameter estimates.
tol	numeric tolerance passed to uniroot or fscores.
method	type of estimation method being passed to fscores.
...	further arguments which are passed to optim in case of vcov = TRUE or fscores.

Details

personpar is both a class to represent person parameters of item response models as well as a generic function. The generic function can be used to return/estimate the person parameters of a given item response model.

By default, the function personpar() reports the distribution parameters of the assumed person ability distribution. For models estimated by marginal maximum likelihood estimation (MML) this is the mean/variance of the underlying normal distribution, whereas for models estimated by conditional maximum likelihood estimation (CML) this is a discrete distribution with one estimation for each observed raw score in the data.

Alternatively, when setting personwise = TRUE, the person parameter for each person/subject in the underlying data set can be extracted. In the CML case, this simply computes the raw score for each person and then extracts the corresponding person parameter. In the MML case, this necessitates (numerically) integrating out the individual person parameters (also known as factor scores or latent trait estimates) based on the underlying normal distribution.

More specifically, the following algorithms are employed for obtaining the distributional person parameters:

- In the MML case – i.e., for nplmodels and gpcmodels – the distributional parameters are already part of the model specification. In a single-group specification and in the reference group of a multi-group specification the mean/variance parameters are fixed to 0/1. In the multi-group case the remaining mean/variance parameters were already estimated along with all other model parameters and simply need to be extracted. Analogously, the corresponding variance-covariance matrix just needs to be extracted and has zero covariances in the cells corresponding to fixed parameters.
- In the CML case – i.e., raschmodels, rsmodels, and pcmodels – the distributional parameters are estimated via uniroot() with the estimation equations given by Hoijtink & Boomsma (1995) as well as Andersen (1995). This approach is fast and estimates for all possible raw scores are available. If the covariance matrix of the estimated person parameters is requested (vcov = TRUE), an additional call of optim is employed to obtain the Hessian numerically. With this approach, person parameters are available only for observed raw scores.

As already explained above, obtaining the person-wise (individual) person paremeters (or ability estimates or factor scores) is straightforward in the CML case. In the MML case, fscores is used, see Chalmers (2012) for further details. If personwise = TRUE, the associated variance-covariance

matrix is not provided and simply a matrix with NAs is returned. (The same is done for vcov = FALSE.)

For objects of class personpar, several methods to standard generic functions exist: print, coef, vcov. coef and vcov can be used to extract the person parameters and covariance matrix without additional attributes. Based on this Wald tests or confidence intervals can be easily computed, e.g., via confint.

Value

A named vector with person parmeters of class personpar and additional attributes "model" (the model name), "vcov" (the covariance matrix of the estimates if vcov = TRUE or an NA-matrix otherwise) and "type" (the type of the parameters, depending on personwise).

References

Andersen EB (1995). Polytomous Rasch Models and Their Estimation. In Fischer GH, Molenaar IW (eds.). *Rasch Models: Foundations, Recent Developments, and Applications*. Springer, New York.

Chalmers RP (2012). mirt: A Multidimensional Item Response Theory Package for the R Environment. *Journal of Statistical Software*, **48**(6), 1–29.

Hoijtink H, Boomsma A (1995). On Person Parameter Estimation in the Dichotomous Rasch Model. In Fischer GH, Molenaar IW (eds.). *Rasch Models: Foundations, Recent Developments, and Applications*. Springer, New York.

See Also

itempar, threshpar, discrpar, guesspar, upperpar

Examples

```
o <- options(digits = 3)

## load verbal aggression data
data("VerbalAggression", package = "psychotools")

## fit a Rasch model to dichotomized verbal aggression data and
ram <- raschmodel(VerbalAggression$resp2)

## extract person parameters
## (= parameters of the underlying ability distribution)
rap <- personpar(ram)
rap

## extract variance-covariance matrix and standard errors
vc <- vcov(rap)
sqrt(diag(vc))

## Wald confidence intervals
confint(rap)
```

```
## now match each person to person parameter with the corresponding raw score
personpar(ram, personwise = TRUE)[1:6]

## person parameters for RSM/PCM fitted to original polytomous data
rsm <- rsmodel(VerbalAggression$resp)
pcm <- pcmodel(VerbalAggression$resp)
cbind(personpar(rsm, vcov = FALSE), personpar(pcm, vcov = FALSE))

if(requireNamespace("mirt")) {
## fit a 2PL accounting for gender impact and
twoplm <- nplmodel(VerbalAggression$resp2, impact = VerbalAggression$gender)

## extract the person parameters
## (= mean/variance parameters from the normal ability distribution)
twoplp <- personpar(twoplm)
twoplp

## extract the standard errors
sqrt(diag(vcov(twoplp)))

## Wald confidence intervals
confint(twoplp)

## now look at the individual person parameters
## (integrated out over the normal ability distribution)
personpar(twoplm, personwise = TRUE)[1:6]
}

options(digits = o$digits)
```

`piplot` *Person-Item Plots for IRT Models*

Description

Base graphics plotting function for person-item plot visualization of IRT models.

Usage

```
piplot(object, pcol = NULL, histogram = TRUE, ref = NULL, items = NULL,
  xlim = NULL, names = NULL, labels = TRUE, main = "Person-Item Plot",
  xlab = "Latent trait", abbreviate = FALSE, cex.axis = 0.8, cex.text = 0.5,
  cex.points = 1.5, grid = TRUE, ...)
```

Arguments

`object`	a fitted model object of class `"raschmodel"`, `"rsmodel"`, `"pcmodel"`, `"nplmodel"` or `"gpcmodel"`.
`pcol`	optional character (vector), specifying the color(s) used for the person parameter plot.

histogram	logical. For models estimated via MML (nplmodels and gpcmodels), should a histogram of the person-wise (individual) person parameters be drawn additionally to the normal distribution density of the person parameters?
ref	argument passed over to internal calls of threshpar and itempar.
items	character or numeric, specifying the items which should be visualized in the person-item plot.
xlim	numeric, specifying the x axis limits.
names	character, specifying labels for the items.
labels	logical, whether to draw the number of the threshold as text below the threshold.
main	character, specifying the overall title of the plot.
xlab	character, specifying the x axis labels.
abbreviate	logical or numeric, specifying whether object names are to be abbreviated. If numeric, this controls the length of the abbreviation.
cex.axis	numeric, the magnification to be used for the axis notation relative to the current setting of cex.
cex.text	numeric, the magnification to be used for the symbols relative to the current setting of cex.
cex.points	numeric, the magnification to be used for the points relative to the current setting of cex.
grid	logical or color specification of horizontal grid lines. If set to FALSE or "transparent" grid lines can be suppressed.
...	further arguments passed to internal calls of lines, points and text.

Details

The person-item plot visualization illustrates the distribution of the person parameters against the absolute item threshold parameters under a certain data set and IRT model. For models estimated via MML (nplmodels and gpcmodels), the normal distribution density of the person parameters is drawn. If histogram is set to TRUE (the default), a histogram of the person-wise (individual) person parameters is drawn additionally. If a multiple group model has been fitted by supplying an impact variable, multiple person parameter plots are drawn, each corresponding to a specific level of this variable.

See Also

curveplot, regionplot, profileplot, infoplot

Examples

```
## load verbal agression data
data("VerbalAggression", package = "psychotools")

## fit partial credit model to verbal aggression data
pcmod <- pcmodel(VerbalAggression$resp)

## create a person-item plot visualization of the fitted PCM
```

```
plot(pcmod, type = "piplot")

## just visualize the first six items and the person parameter plot
plot(pcmod, type = "piplot", items = 1:6, pcol = "lightblue")


if(requireNamespace("mirt")) {
## fit generalized partial credit model to verbal aggression data
gpcmod <- gpcmodel(VerbalAggression$resp)

## create a person-item plot visualization of the fitted GPCM
plot(gpcmod, type = "piplot")

## turn off the histogram and grid
plot(gpcmod, type = "piplot", histogram = FALSE, grid = FALSE)

## fit GPCM to verbal aggression data accounting for gender impact
mgpcmod <- gpcmodel(VerbalAggression$resp, impact = VerbalAggression$gender)

## create a person-item plot visualization of the fitted GPCM
plot(mgpcmod, type = "piplot", pcol = c("darkgreen", "darkorange"))
}
```

`plot.btmodel` *Visualizing Bradley-Terry Models*

Description

Base graphics plotting function for Bradley-Terry models.

Usage

```
## S3 method for class 'btmodel'
plot(x, worth = TRUE, index = TRUE, names = TRUE,
  ref = TRUE, abbreviate = FALSE, type = NULL, lty = NULL,
  xlab = "Objects", ylab = NULL, ...)
```

Arguments

`x`	an object of class `"btmodel"`.
`worth`	logical. Should worth parameters (or alternatively coefficients on log-scale) be displayed?
`index`	logical. Should different indexes for different items be used?
`names`	logical. Should the names for the objects be displayed?
`ref`	logical. Should a horizontal line for the reference level be drawn? Alternatively, `ref` can also be numeric or character to employ a reference level different from that stored in `x`.

abbreviate	logical or numeric. Should object names be abbreviated? If numeric this controls the length of the abbreviation.
type	plot type. Default is "b" if index is TRUE.
lty	line type.
xlab, ylab	x and y axis labels.
...	further arguments passed to plot.

See Also

btmodel

Examples

```
## data
data("GermanParties2009", package = "psychotools")

## Bradley-Terry model
bt <- btmodel(GermanParties2009$preference)
plot(bt)
plot(bt, worth = FALSE)
plot(bt, index = FALSE)
```

---

plot.paircomp *Plotting Paired Comparison Data*

Description

Plotting the frequency table from "paircomp" data.

Usage

```
  ## S3 method for class 'paircomp'
plot(x, off = 0.05,
  xlab = "Proportion of comparisons", ylab = "", tol.xlab = 0.05,
  abbreviate = TRUE, hue = NULL, chroma = 40, luminance = 80,
  xlim = c(0, 1), ylim = NULL, xaxs = "i", yaxs = "i", ...)
```

Arguments

x	an object of class "paircomp".
off	numeric. Offset between segments on the y-axis.
xlab, ylab	character. Axis labels.
tol.xlab	numeric. convenience tolerance parameter for x-axis annotation. If the distance between two labels drops under this threshold, they are plotted equidistantly.

abbreviate	logical or integer. Should object labels be abbreviated? Alternative an integer with the desired abbreviation length. The default is some heuristic based on the length of the labels.
hue	numeric. A vector of hues in [0, 360], recycled to the number of objects compared in x. A sequential palette is computed for each hue, see below.
chroma	numeric. Maximum chroma in the palette.
luminance	numeric. Minimum (and maximum) luminance in the palette. If omitted, the maximum is set to 95.
xlim, ylim, xaxs, yaxs, ...	graphical arguments passed to plot.

Details

The plot method creates a frequency table (using summary) and visualizes this using a sort of spine plot with HCL-based diverging palettes. See Zeileis, Hornik, Murrell (2009) for the underlying ideas.

References

Zeileis A, Hornik K, Murrell P (2009), Escaping RGBland: Selecting Colors for Statistical Graphics. *Computational Statistics & Data Analysis*, **53**, 3259-3270. doi:10.1016/j.csda.2008.11.033

See Also

paircomp

Examples

```
data("GermanParties2009", package = "psychotools")
par(mar = c(5, 6, 3, 6))
plot(GermanParties2009$preference, abbreviate = FALSE)
```

---

plot.raschmodel *Visualizing IRT Models*

Description

Base graphics plotting function for various IRT models.

Usage

```
## S3 method for class 'raschmodel'
plot(x,
  type = c("profile", "curves", "regions", "information", "piplot"), ...)
## S3 method for class 'rsmodel'
plot(x,
  type = c("regions", "profile", "curves", "information", "piplot"), ...)
```

```
## S3 method for class 'pcmodel'
plot(x,
  type = c("regions", "profile", "curves", "information", "piplot"), ...)
## S3 method for class 'nplmodel'
plot(x,
  type = c("regions", "profile", "curves", "information", "piplot"), ...)
## S3 method for class 'gpcmodel'
plot(x,
  type = c("regions", "profile", "curves", "information", "piplot"), ...)
```

Arguments

x	a fitted model object of class "raschmodel", "rsmodel", "pcmodel", "nplmodel" or "gpcmodel".
type	character, specifying the type of plot to create.
...	further arguments passed over to the specific plotting function.

See Also

curveplot, regionplot, profileplot, infoplot, piplot

---

predict.pcmodel *Predict Methods for Item Response Models*

Description

Prediction of (cumulated) response probabilities and responses based on fitted item response models.

Usage

```
## S3 method for class 'pcmodel'
predict(object, newdata = NULL, type = c("probability",
  "cumprobability", "mode", "median", "mean", "category-information",
  "item-information", "test-information"), ref = NULL, ...)
```

Arguments

object	a fitted model object whose item parameters should be used for prediction.
newdata	an optional (possibly named) vector of person parameters used for prediction. If NULL (the default), the person parameters of the subjects used to fit the model in object are used.
type	character of length one which determines the type of prediction (see details below).
ref	arguments passed over to internal calls of itempar or threshpar. Not used for models estimated via MML.
...	further arguments which are currently not used.

Details

Depending on the value of `type` either probabilities, responses or some form of information under the model specified in `object` are returned:

If `type` is `"probability"`, the category response probabilities are returned.

If `type` is `"cumprobability"`, the cumulated category response probabilities are returned, i.e., $P(X_{ij} \geq k)$ with k corresponding to the categories of item j.

If `type` is `"mode"`, the most probable category response for a given subject and item is returned.

If `type` is `"median"`, the first category k where $P(X_{ij} = k) \geq 0.5$ is returned.

If `type` is `"mean"`, the rounded expected category response, i.e., $E(X_{ij}|\theta_i)$, is returned.

If `type` is `"category-information"`, the item-category information as suggested by Bock (1972) is returned.

If `type` is `"item-information"`, the item information as suggested by Samejima (1974) is returned.

If `type` is `"test-information"`, the sum over the individual item information values is returned.

Value

A (possibly named) numeric matrix with rows corresponding to subjects and columns corresponding to the whole test, the single items or categories. The exact content depends on the value of `type` (see details above).

References

Bock RD (1972). Estimating Item Parameters and Latent Ability When Responses Are Scored in Two or More Nominal Categories. *Psychometrika*, **37**(1), 29–51.

Samejima F (1974). Normal Ogive Model on the Continuous Response Level in the Multidimensional Latent Space. *Psychometrika*, **39**(1), 111–121.

See Also

The help page of the generic function `predict` and other predict methods (e.g., `predict.lm`, `predict.glm`, ...)

Examples

```
o <- options(digits = 4)

## load verbal aggression data
data("VerbalAggression", package = "psychotools")

## fit a partial credit model to first ten items
pcmod <- pcmodel(VerbalAggression$resp[, 1:10])

## predicted response probabilities for each subject and category (the default)
head(predict(pcmod), 3)

## predicted mode (most probable category) for certain subjects whose person
## parameters are given via argument "newdata"
```

```
predict(pcmod, type = "mode",
  newdata = c("Sarah" = 1.2, "Michael" = 0.1, "Arnd" = -0.8))

## rounded expected category value for the same subjects
predict(pcmod, type = "mean",
  newdata = c("Sarah" = 1.2, "Michael" = 0.1, "Arnd" = -0.8))

## in the Rasch model mode, mean and median are the same
raschmod <- raschmodel(VerbalAggression$resp2[, 1:10])
med <- predict(raschmod, type = "median")
mn <- predict(raschmod, type = "mean")
mod <- predict(raschmod, type = "mode")

head(med, 3)

all.equal(med, mn)
all.equal(mod, mn)

options(digits = o$digits)
```

`print.itemresp` *Formatting Item Response Data*

Description

Fine control for formatting and printing `"itemresp"` data objects.

Usage

```
  ## S3 method for class 'itemresp'
format(x, sep = c(",", ":"), brackets = TRUE,
    abbreviate = NULL, mscale = TRUE, labels = FALSE,
    width = getOption("width") - 7L, ...)
  ## S3 method for class 'itemresp'
print(x, quote = FALSE, ...)
```

Arguments

`x`	an object of class `"itemresp"`.
`sep`	character. A character of length 2 (otherwise expanded/reduced) for separating responses and items, respectively.
`brackets`	logical or character. Either a logical (Should brackets be wrapped around all responses for a single subject?) or a character of length 2 with opening and ending symbol.
`abbreviate`	logical or integer. Should scale labels be abbreviated? Alternatively, an integer with the desired abbreviation length. The default is some heuristic based on the length of the labels.

mscale	logical. Should mscale values be used for printing? If FALSE, integers 0, 1, ... are used.
labels	logical. Should item labels be displayed?
width	integer or logical. Maximal width of the string for a subject. If FALSE no maximal width is set.
...	arguments passed to other functions.
quote	logical. Should quotes be printed?

Details

The print method just calls format (passing on all further arguments) and then prints the resulting string.

See Also

itemresp

Examples

```
## item responses from binary matrix
x <- cbind(c(1, 0, 1, 0), c(1, 0, 0, 0), c(0, 1, 1, 1))
xi <- itemresp(x)
## change mscale
mscale(xi) <- c("-", "+")
xi

## flexible formatting
## no/other brackets
print(xi, brackets = FALSE)
print(xi, brackets = c(">>", "<<"))

## include item labels (with different separators)
print(xi, labels = TRUE)
print(xi, labels = TRUE, sep = c(" | ", ": "))

## handling longer mscale categories
mscale(xi) <- c("disagree", "agree")
print(xi)
print(xi, mscale = FALSE)
print(xi, abbreviate = FALSE)
print(xi, abbreviate = FALSE, width = 23)
print(xi, abbreviate = 2)
```

print.paircomp *Formatting Paired Comparison Data*

Description

Fine control for formatting and printing objects of "paircomp" data.

Usage

```
  ## S3 method for class 'paircomp'
format(x, sep = ", ", brackets = TRUE,
    abbreviate = NULL, width = getOption("width") - 7, ...)
  ## S3 method for class 'paircomp'
print(x, quote = FALSE, ...)
```

Arguments

x	an object of class "paircomp".
sep	character. A character for separating comparisons within subjects.
brackets	logical or character. Either a logical (Should brackets be wrapped around all comparisons for a single subject?) or a character of length two with opening and ending symbol.
abbreviate	logical or integer. Should object labels be abbreviated? Alternative an integer with the desired abbreviation length. The default is some heuristic based on the length of the labels.
width	integer or logical. Maximal width of the string for a subject. If FALSE no maximal width is set.
...	arguments passed to other functions.
quote	logical. Should quotes be printed?

Details

The print method just calls format (passing on all further arguments) and then prints the resulting string.

See Also

paircomp

Examples

```
pc2 <- paircomp(rbind(
  c(4,  1,  0),
  c(1,  2, -1),
  c(1, -2, -1),
  c(0,  0,  -3)),
```

```
  labels = c("New York", "Rio", "Tokyo"))

print(pc2)
print(pc2, abbreviate = FALSE)
print(pc2, abbreviate = FALSE, width = 10)
```

---

profileplot *Profile Plots for IRT Models*

Description

Base graphics plotting function for profile plot visualization of IRT models.

Usage

```
profileplot(object,
  what = c("items", "thresholds", "discriminations", "guessings", "uppers"),
 parg = list(type = NULL, ref = NULL, alias = TRUE, logit = FALSE), index = TRUE,
  names = TRUE, main = NULL, abbreviate = FALSE, ref = TRUE,
  col = "lightgray", border = "black", pch = NULL, cex = 1,
  refcol = "lightgray", linecol = "black", lty = 2, ylim = NULL,
  xlab = NULL, ylab = NULL, add = FALSE, srt = 45, adj = c(1.1, 1.1),
  axes = TRUE, ...)
```

Arguments

object	a fitted model object of class "raschmodel", "rsmodel", "pcmodel", "nplmodel" or "gpcmodel".
what	character, specifying the type of parameters to be plotted.
parg	list of arguments passed over to internal calls of itempar, threshpar, discrpar, guesspar, or upperpar.
index	logical, should different indexes for different items be used?
names	logical or character. If TRUE, the names of the items are displayed on the x-axis. If FALSE, numbers of items are shown. Alternatively a character vector of the same length as the number of items can be supplied.
main	character, specifying the overall title of the plot.
abbreviate	logical or numeric, specifying whether object names are to be abbreviated. If numeric this controls the length of the abbreviation.
ref	logical, whether to draw a horizontal line for the reference level. Only takes effect if argument what is "items" or "discriminations".
col, border, pch, cex	graphical appearance of plotting symbols. Can be of the same length as the number of items, i.e., a different graphical appearance is used for each item. If what = "thresholds", col and pch can be matrices with a number of columns equal to the number of threshold parameters per item resulting in different symbols and colors used for different threshold parameter profiles.

refcol	character, specifying the line color for the reference line (if ref is set to TRUE).
linecol	character or numeric, specifying the line color to be used for the profiles.
lty	numeric, specifying the line type for the profiles.
ylim	numeric, specifying the y axis limits.
xlab, ylab	character, specifying the x and y axis labels.
add	logical. If TRUE, new plotted profiles are added to an existing plot.
srt, adj	numeric. Angle (srt) and adjustment (adj) in case names (rather than numbers) are used as x-axis labels. These are passed to text.
axes	logical. Should axes be drawn?
...	further arguments passed over to plot.

Details

The profile plot visualization illustrates profiles of specific estimated parameters under a certain IRT model.

See Also

curveplot, regionplot, infoplot, piplot

Examples

```
## load verbal aggression data
data("VerbalAggression", package = "psychotools")

## fit Rasch, rating scale and partial credit model to verbal aggression data
rmmod <- raschmodel(VerbalAggression$resp2)
rsmod <- rsmodel(VerbalAggression$resp)
pcmod <- pcmodel(VerbalAggression$resp)

## profile plots of the item parameters of the three fitted IRT models
plot(rmmod, type = "profile", what = "items", col = 4)
plot(rsmod, type = "profile", what = "items", col = 2, add = TRUE)
plot(pcmod, type = "profile", what = "items", col = 3, add = TRUE)
legend(x = "topleft", legend = c("RM", "RSM", "PCM"), col = 1,
  bg = c(4, 2, 3), pch = 21, bty = "n")

## profile plots of the threshold parameters of type "mode"
plot(rmmod, type = "profile", what = "thresholds", parg = list(type = "mode"))
plot(rsmod, type = "profile", what = "thresholds", parg = list(type = "mode"))
plot(pcmod, type = "profile", what = "thresholds", parg = list(type = "mode"))

## profile plot of the discrimination parameters of the dichotomous RM
plot(rmmod, type = "profile", what = "discrimination")


if(requireNamespace("mirt")) {
## fit 2PL and generalized partial credit model to verbal aggression data
twoplmod <- nplmodel(VerbalAggression$resp2)
```

```
gpcmod <- gpcmodel(VerbalAggression$resp)

## profile plot of the discrimination parameters of a dichotomous 2PL
plot(twoplmod, type = "profile", what = "discrimination")

## profile plot of the item parameters of the 2PL and GPCM
plot(twoplmod, type = "profile", what = "items", col = 4)
plot(gpcmod, type = "profile", what = "items", col = 2, add = TRUE)
}
```

---

raschmodel *Rasch Model Fitting Function*

Description

raschmodel is a basic fitting function for simple Rasch models.

Usage

```
raschmodel(y, weights = NULL, start = NULL, reltol = 1e-10,
  deriv = c("sum", "diff", "numeric"), hessian = TRUE,
  maxit = 100L, full = TRUE, gradtol = reltol, iterlim = maxit, ...)
```

Arguments

y	item response object that can be coerced (via as.matrix) to a binary 0/1 matrix (e.g., from class itemresp.
weights	an optional vector of weights (interpreted as case weights).
start	an optional vector of starting values.
deriv	character. Which type of derivatives should be used for computing gradient and Hessian matrix? Analytical with sum algorithm ("sum"), analytical with difference algorithm ("diff", faster but numerically unstable), or numerical.
hessian	logical. Should the Hessian of the final model be computed? If set to FALSE, the vcov method can only return NAs and consequently no standard errors or tests are available in the summary.
reltol, maxit, ...	further arguments passed to optim.
full	logical. Should a full model object be returned? If set to FALSE, no variance-covariance matrix and no matrix of estimating functions are computed.
gradtol, iterlim	numeric. For backward compatibility with previous versions these arguments are mapped to reltol and maxit, respectively.

Details

raschmodel provides a basic fitting function for simple Rasch models, intended as a building block for fitting Rasch trees and Rasch mixtures in the **psychotree** and **psychomix** packages, respectively.

raschmodel returns an object of class "raschmodel" for which several basic methods are available, including print, plot, summary, coef, vcov, logLik, estfun, discrpar, itempar, threshpar, and personpar.

Value

raschmodel returns an S3 object of class "raschmodel", i.e., a list with the following components:

coefficients	estimated item difficulty parameters (without first item parameter which is always constrained to be 0),
vcov	covariance matrix of the parameters in the model,
loglik	log-likelihood of the fitted model,
df	number of estimated parameters,
data	the original data supplied (excluding columns without variance),
weights	the weights used (if any),
n	number of observations (with non-zero weights),
items	status indicator (0, 0/1, 1) of all original items,
na	logical indicating whether the data contains NAs,
elementary_symmetric_functions	List of elementary symmetric functions for estimated parameters (up to order 2; or 1 in case of numeric derivatives),
code	convergence code from optim,
iterations	number of iterations used by optim,
reltol	tolerance passed to optim,
deriv	type of derivatives used for computing gradient and Hessian matrix,
call	original function call.

See Also

nplmodel, pcmodel, rsmodel, gpcmodel, btmodel

Examples

```
o <- options(digits = 4)

## Verbal aggression data
data("VerbalAggression", package = "psychotools")

## Rasch model for the other-to-blame situations
m <- raschmodel(VerbalAggression$resp2[, 1:12])
## IGNORE_RDIFF_BEGIN
summary(m)
```

```
## IGNORE_RDIFF_END

## visualizations
plot(m, type = "profile")
plot(m, type = "regions")
plot(m, type = "curves")
plot(m, type = "information")
plot(m, type = "piplot")

options(digits = o$digits)
```

`regionplot` *Region Plots for IRT Models*

Description

Base graphics plotting function for region plot visualization of IRT models.

Usage

```
regionplot(object, parg = list(type = NULL, ref = NULL, alias = TRUE),
  names = TRUE, main = NULL, xlab = "", ylab = "Latent trait", ylim = NULL,
  off = 0.1, col = NULL, linecol = 2, srt = 45, adj = c(1.1, 1.1),
  axes = TRUE, ...)
```

Arguments

`object`	a fitted model object of class `"raschmodel"`, `"rsmodel"`, `"pcmodel"`, `"nplmodel"` or `"gpcmodel"`.
`parg`	list of arguments passed over to internal calls of `threshpar`. See the help page of `threshpar` for more details.
`names`	logical or character. If `TRUE`, the names of the items are displayed on the x-axis. If `FALSE`, numbers of items are shown. Alternatively a character vector of the same length as the number of items can be supplied.
`main`	character, specifying the overall title of the plot.
`xlab, ylab`	character, specifying the x and y axis labels.
`ylim`	numeric, specifying the y axis limits.
`off`	numeric, the distance (in scale units) between two item rectangles.
`col`	character, list or function, specifying the colors of the regions. Either a single vector with k color names, a list with m elements and each element is a character vector with color names for the regions of item j or a color-generating function like, e.g., `gray.colors`, which is then directly used to create the color names.
`linecol`	color for lines indicating "hidden" categories.
`srt, adj`	numeric. Angle (`srt`) and adjustment (`adj`) in case names (rather than numbers) are used as x-axis labels. These are passed to `text`.
`axes`	logical. Should axes be drawn?
`...`	further arguments passed to `plot`.

Details

The region plot visualization implemented here was already used by Van der Linden and Hambleton (1997) in the context of IRT and has been called "effect plots" by Fox & Hong (2009). In our implementation, these plots show, dependent on the chosen type of threshold parameters, different regions for the categories of an item over the theta axis. If `type` is set to `"modus"`, the cutpoints correspond to the threshold parameters and the rectangles mark the theta regions where a category is the single most probable category chosen with a certain value of the latent trait. If `type` is set to `"median"`, the cutpoints correspond to the point on the theta axis, where the cumulative probability to score in category k or higher is 0.5, i.e., $P(X_{ij} \geq k) = 0.5$. If set to `"mean"`, the cutpoints correspond to the point on the theta axis where the expected score $E(X_{ij})$ is exactly between two categories, e.g., 0.5 for a dichotomous item.

If `type` is set to `"mode"` and there are unordered threshold parameters, the location of the original threshold parameters are indicated by red dashed lines.

References

Fox J, Hong J (2009). Effect Displays in R for Multinomial and Proportional-Odds Logit Models: Extensions to the effects Package. *Journal of Statistical Software*, **32**(1), 1–24.

Van der Linden WJ, Hambleton RK (1997). *Handbook of Modern Item Response Theory*. Springer, New York.

See Also

`curveplot`, `profileplot`, `infoplot`, `piplot`

Examples

```
## load verbal aggression data
data("VerbalAggression", package = "psychotools")

## fit a Partial credit model to the items of the first other-to-blame
## situation: "A bus fails to stop for me"
pcm <- pcmodel(VerbalAggression$resp[, 1:6])

## a region plot with modus as cutpoint and custom labels
lab <- paste(rep(c("Curse", "Scold", "Shout"), each = 2),
  rep(c("Want", "Do"), 3 ), sep = "-")
plot(pcm, type = "regions", names = lab)

## compare the cutpoints (with ylim specified manually)
opar <- par(no.readonly = TRUE)
ylim <- c(-2, 2)
layout(matrix(1:3, ncol = 1))
plot(pcm, type = "regions", parg = list(type = "mode"),
  main = "Modus as Cutpoint", ylim = ylim)
plot(pcm, type = "regions", parg = list(type = "median"),
  main = "Median as Cutpoint", ylim = ylim)
plot(pcm, type = "regions", parg = list(type = "mean"),
  main = "Mean as Cutpoint", ylim = ylim)
par(opar)
```

```
## PCM for full verbal aggression data set
pcm_va <- pcmodel(VerbalAggression$resp)
plot(pcm_va, type = "regions")

if(requireNamespace("mirt")) {
## generalized partial credit model for full verbal aggression data set
gpcm_va <- gpcmodel(VerbalAggression$resp)
plot(gpcm_va, type = "regions")
}
```

---

rgpcm *Simulate Data under a Generalized Partial Credit Model*

Description

rgpcm simulates IRT data under a generalized partial credit model.

Usage

```
rgpcm(theta, a, b, nullcats = FALSE, return_setting = TRUE)
```

Arguments

theta	numeric vector of person parameters. Can also be a list, then a list of length length(theta) is returned, containing multiple simulated data sets.
a	list of numerics of item discrimination parameters.
b	list of numeric vectors of item threshold parameters.
nullcats	logical. Should null categories be allowed?
return_setting	logical. Should a list containing slots of "a", "b", and "theta", as well as the simulated data matrix "data" be returned (default) or only the simulated data matrix?

Value

rgpcm returns either a list of the following components:

a	list of numerics of item discrimination parameters used,
b	list of numeric vectors of item threshold parameters used,
theta	numeric vector of person parameters used,
data	numeric matrix containing the simulated data,

or (if return_setting = FALSE) only the numeric matrix containing the simulated data.

See Also

rpcm, rrsm, rpl, rrm

Examples

```
set.seed(1)
## item responses under a GPCM (generalized partial credit model) from
## 6 persons with three different person parameters
## 8 items with different combinations of two or three threshold parameters
## and corresponding discrimination parameters
ppar <- rep(-1:1, each = 2)
tpar <- rep(list(-2:0, -1:1, 0:1, 0:2), each = 2)
dpar <- rep(list(1, 2), each = 4)
sim <- rgpcm(theta = ppar, a = dpar, b = tpar)

## simulated item response data along with setting parameters
sim

## print and plot corresponding item response object
iresp <- itemresp(sim$data)
iresp
plot(iresp)
```

---

rpcm *Simulate Data under a Partial Credit Model*

Description

rpcm simulates IRT data under a partial credit model.

Usage

```
rpcm(theta, delta, nullcats = FALSE, return_setting = TRUE)
```

Arguments

theta	numeric vector of person parameters. Can also be a list, then a list of length length(theta) is returned, containing multiple simulated data sets.
delta	list of numeric vectors of item threshold parameters.
nullcats	logical. Should null categories be allowed?
return_setting	logical. Should a list containing slots of "delta", and "theta", as well as the simulated data matrix "data" be returned (default) or only the simulated data matrix?

Value

rpcm returns either a list of the following components:

delta	list of numeric vectors of item threshold parameters used,
theta	numeric vector of person parameters used,
data	numeric matrix containing the simulated data,

or (if return_setting = FALSE) only the numeric matrix containing the simulated data.

See Also

rgpcm, rrsm, rrm, rpl

Examples

```
set.seed(1)
## item responses under a partial credit model (PCM) with
## 6 persons with three different person parameters
## 8 items with different combinations of two or three threshold parameters
ppar <- rep(-1:1, each = 2)
tpar <- rep(list(-2:0, -1:1, 0:1, 0:2), each = 2)
sim <- rpcm(theta = ppar, delta = tpar)

## simulated item response data along with setting parameters
sim

## print and plot corresponding item response object
iresp <- itemresp(sim$data)
iresp
plot(iresp)
```

---

rpl *Simulate Data under a Parametric Logistic IRT Model*

Description

rpl simulates IRT data under a parametric logistic IRT model of type "2PL", "3PL", "3PLu", "4PL", and "Rasch/1PL".

Usage

```
rpl(theta, a = NULL, b, g = NULL, u = NULL, return_setting = TRUE)
```

Arguments

theta	numeric vector of person parameters. Can also be a list, then a list of length length(theta) is returned, containing multiple simulated data matrices.
a	numeric vector of item discrimination parameters. If NULL, by default set to a vector of ones of length length(b).
b	numeric vector of item difficulty parameters.
g	numeric vector of so-called item guessing parameters. If NULL, by default set to a vector of zeroes of length length(b).
u	numeric vector of item upper asymptote parameters. If NULL, by default set to a vector of ones of length length(b).
return_setting	logical. Should a list containing slots of "a", "b", "g", "u", and "theta", as well as the simulated data matrix "data" be returned (default) or only the simulated data matrix.

Value

rpl returns either a list of the following components:

a	numeric vector of item discrimination parameters used,
b	numeric vector of item difficulty parameters used,
g	numeric vector of item guessing parameters used,
u	numeric vector of item upper asymptote parameters used,
theta	numeric vector of person parameters used,
data	numeric matrix containing the simulated data,

or (if return_setting = FALSE) only the numeric matrix containing the simulated data.

See Also

rrm, rgpcm, rpcm, rrsm

Examples

```
set.seed(1)
## item responses under a 2PL (two-parameter logistic) model from
## 6 persons with three different person parameters
## 9 increasingly difficult items and corresponding discrimination parameters
## no guessing (= 0) and upper asymptote 1
ppar <- rep(c(-2, 0, 2), each = 2)
ipar <- seq(-2, 2, by = 0.5)
dpar <- rep(c(0.5, 1, 1.5), each = 3)
sim <- rpl(theta = ppar, a = dpar, b = ipar)

## simulated item response data along with setting parameters
sim

## print and plot corresponding item response object
iresp <- itemresp(sim$data)
iresp
plot(iresp)
```

---

rrm *Simulate Data under a Rasch model*

Description

rrm simulates IRT data under a Rasch model.

Usage

```
rrm(theta, beta, return_setting = TRUE)
```

Arguments

theta	numeric vector of person parameters. Can also be a list, then a list of length length(theta) is returned, containing multiple simulated data matrices.
beta	numeric vector of item difficulty parameters.
return_setting	logical. Should a list containing slots of "beta", and "theta", as well as the simulated data matrix "data" be returned (default) or only the simulated data matrix.

Value

rrm returns either a list of the following components:

beta	numeric vector of item difficulty parameters used,
theta	numeric vector of person parameters used,
data	numeric matrix containing the simulated data,

or (if return_setting = FALSE) only the numeric matrix containing the simulated data.

See Also

rpl, rpcm, rrsm, rgpcm

Examples

```
set.seed(1)
## item responses under a Rasch model from
## 6 persons with three different person parameters
## 9 increasingly difficult items
ppar <- rep(-1:1, each = 2)
ipar <- seq(-2, 2, by = 0.5)
sim <- rrm(theta = ppar, beta = ipar)

## simulated item response data along with setting parameters
sim

## print and plot corresponding item response object
iresp <- itemresp(sim$data)
iresp
plot(iresp)
```

---

rrsm *Simulate Data under a Rating Scale Model*

Description

rrsm simulates IRT data under a rating scale model.

Usage

```
rrsm(theta, beta, tau, nullcats = FALSE, return_setting = TRUE)
```

Arguments

theta	numeric vector of person parameters. Can also be a list, then a list of length length(theta) is returned, containing multiple simulated data sets.
beta	numeric vector of item difficulty parameters.
tau	numeric vector of threshold parameters.
nullcats	logical. Should null categories be allowed?
return_setting	logical. Should a list containing slots of "beta", "tau", and "theta", as well as the simulated data matrix "data" be returned (default) or only the simulated data matrix?

Value

rrsm returns either a list of the following components:

beta	numeric vector of item difficulty parameters used,
tau	numeric vector of threshold parameters used,
theta	numeric vector (or list) of person parameters used,
data	numeric matrix containing the simulated data,

or (if return_setting = FALSE) only the numeric matrix containing the simulated data.

See Also

rpcm, rgpcm, rrm, rpl

Examples

```
set.seed(1)
## item responses under a rating scale model (RSM) with
## 6 persons with three different person parameters
## 9 increasingly difficult items
## 3 different threshold parameters
ppar <- rep(-1:1, each = 2)
ipar <- seq(-2, 2, by = 0.5)
tpar <- 0:2
sim <- rrsm(theta = ppar, beta = ipar, tau = tpar)

## simulated item response data along with setting parameters
sim

## print and plot corresponding item response object
iresp <- itemresp(sim$data)
iresp
plot(iresp)
```

rsmodel *Rating Scale Model Fitting Function*

Description

rsmodel is a basic fitting function for rating scale models.

Usage

```
rsmodel(y, weights = NULL, start = NULL, reltol = 1e-10,
  deriv = c("sum", "diff"), hessian = TRUE,
  maxit = 100L, full = TRUE, ...)
```

Arguments

y	item response object that can be coerced (via as.matrix) to a numeric matrix with scores 0, 1, ... Typically, either already a matrix, data frame, or dedicated object of class itemresp.
weights	an optional vector of weights (interpreted as case weights).
deriv	character. If "sum" (the default), the first derivatives of the elementary symmetric functions are calculated with the sum algorithm. Otherwise ("diff") the difference algorithm (faster but numerically unstable) is used.
start	an optional vector of starting values.
hessian	logical. Should the Hessian of the final model be computed? If set to FALSE, the vcov method can only return NAs and consequently no standard errors or tests are available in the summary.
reltol, maxit, ...	further arguments passed to optim.
full	logical. Should a full model object be returned? If set to FALSE, no variance-covariance matrix and no matrix of estimating functions are computed.

Details

rsmodel provides a basic fitting function for rating scales models, intended as a building block for fitting rating scale trees. It estimates the rating scale model in the parametrization suggested by Andrich (1978), i.e., item-specific parameters ξ_j who mark the location of the first absolute threshold of an item on the theta axis and cumulative relative threshold parameters κ_k are estimated by the function rsmodel.

rsmodel returns an object of class "rsmodel" (and class "pcmodel") for which several basic methods are available, including print, plot, summary, coef, vcov, logLik, discrpar, estfun, itempar, threshpar, and personpar.

Value

`rsmodel` returns an S3 object of class `"rsmodel"`, i.e., a list with the following components:

`coefficients`	a named vector of estimated item-specific parameters (without the first item parameter which is constrained to 0) and estimated cumulative relative treshold parameters (again without first threshold parameter which is also constrained to 0),
`vcov`	covariance matrix of the parameters in the model,
`data`	modified data, used for model-fitting, i.e., cleaned for items without variance, centralized so that the first category is zero for all items and without observations with zero weight. Be careful, this is different than for objects of class `"raschmodel"` or `"btmodel"`, where `data` contains the *original* data,
`items`	logical vector of length `ncol(y)`, which indicates which items have variance (`TRUE`), i.e., are identified and have been used for the estimation or not (`FALSE`),
`categories`	integer vector of length `ncol(y)`, which contains the number of categories minus one per item,
`n`	number of observations (with non-zero weights),
`n_org`	original number of observations in `y`,
`weights`	the weights used (if any),
`na`	logical indicating whether the data contains NAs,
`esf`	list of elementary symmetric functions and their derivatives for estimated parameters,
`loglik`	log-likelihood of the fitted model,
`df`	number of estimated parameters,
`code`	convergence code from `optim`,
`iterations`	number of iterations used by `optim`,
`reltol`	tolerance passed to `optim`,
`call`	original function call.

References

Andrich D (1978). Application of a Psychometric Rating Model to Ordered Categories Which Are Scored with Successive Integers. *Psychometrika*, **2**(4), 581–594.

See Also

`pcmodel`, `gpcmodel`, `raschmodel`, `nplmodel`, `btmodel`

Examples

```
o <- options(digits = 4)

## Verbal aggression data
data("VerbalAggression", package = "psychotools")
```

```
## Rating scale model for the other-to-blame situations
rsm <- rsmodel(VerbalAggression$resp[, 1:12])
summary(rsm)

## visualizations
plot(rsm, type = "profile")
plot(rsm, type = "regions")
plot(rsm, type = "curves")
plot(rsm, type = "information")
plot(rsm, type = "piplot")

options(digits = o$digits)
```

Sim3PL *Simulated Data for fitting a 3PL and 3PLu*

Description

Simulated responses of 10000 persons to 10 dichotomous items under two different simulation conditions.

Usage

```
data("Sim3PL", package = "psychotools")
```

Format

A data frame containing 10000 observations on 2 variables.

resp Item response matrix with 10 items (see details below).

resp2 Item response matrix with 10 items (see details below).

Details

Data were simulated under the 3PL (`resp`) and 3PLu (`resp2`) (see `nplmodel`). For the 3PL scenario, the random number generator's seed was set to 277. For the 3PLu scenario, the random number generator's seed was set to 167. Person parameters θ_i of 10000 persons were drawn from the standard normal distribution. Item difficulties b_j of 10 items (under the classical IRT parametrization) were drawn from the standard normal distribution. Item discrimination parameters a_j were drawn from a log-normal distribution with a mean of 0 and a variance of 0.0625 on the log scale. For the 3PL, guessing parameters g_j were drawn from a uniform distribution with a lower limit of 0.1 and an upper limit of 0.2. For the 3PLu, upper asymptote parameters u_j were drawn from a uniform distribution with a lower limit of 0.8 and an upper limit of 0.9. In both scenarios, a 10000 x 10 matrix based on realizations of a uniform distribution with a lower limit of 0 and an upper limit of 1 was generated and compared to a 10000 x 10 matrix based on the probability function under the respective model. If the probability of person i solving item j exceeded the corresponding realization of the uniform distribution, this cell of the matrix was set to 1, e.g., person i solved item j.

See Also

nplmodel

Examples

```
## overview
data("Sim3PL", package = "psychotools")
str(Sim3PL)

## data generation
M <- 10000
N <- 10

## 3PL scenario
set.seed(277)
theta <- rnorm(M, 0, 1)
a <- rlnorm(N, 0, 0.25)
b <- rnorm(N, 0, 1)
g <- runif(N, 0.1, 0.2)
u <- rep(1, N)
probs <- matrix(g, M, N, byrow = TRUE) + matrix(u - g, M, N, byrow = TRUE) *
  plogis(matrix(a, M, N, byrow = TRUE) * outer(theta, b, "-"))
resp <- (probs > matrix(runif(M * N, 0, 1), M, N)) + 0
all.equal(resp, Sim3PL$resp, check.attributes = FALSE)

## 3PLu scenario
set.seed(167)
theta <- rnorm(M, 0, 1)
a <- rlnorm(N, 0, 0.25)
b <- rnorm(N, 0, 1)
g <- rep(0, N)
u <- runif(N, 0.8, 0.9)
probs <- matrix(g, M, N, byrow = TRUE) + matrix(u - g, M, N, byrow = TRUE) *
  plogis(matrix(a, M, N, byrow = TRUE) * outer(theta, b, "-"))
resp2 <- (probs > matrix(runif(M * N, 0, 1), M, N)) + 0
all.equal(resp2, Sim3PL$resp2, check.attributes = FALSE)
```

SoundQuality *Quality of Multichannel Reproduced Sound*

Description

Paired comparison judgments of 40 selected listeners with respect to eight audio reproduction modes and four types of music.

Usage

```
data("SoundQuality")
```

Format

A data frame containing 783 observations on 6 variables.

id Factor. Listener ID.

time Factor. Listening experiment before or after elicitation and scaling of more specific auditory attributes.

progmat Factor. The program material: Beethoven, Rachmaninov, Steely Dan, Sting.

repet The repetition within each time point.

session The experimental session coding the presentation order of the program material.

preference Paired comparison of class `paircomp`. Preferences for all 28 paired comparisons from 8 audio reproduction modes: Mono, Phantom Mono, Stereo, Wide-Angle Stereo, 4-channel Matrix, 5-channel Upmix 1, 5-channel Upmix 2, and 5-channel Original.

Details

The data were collected within a series of experiments conducted at the Sound Quality Research Unit (SQRU), Department of Acoustics, Aalborg University, Denmark, between September 2004 and March 2005.

The results of scaling listener preference and spatial and timbral auditory attributes are reported in Choisel and Wickelmaier (2007).

Details about the loudspeaker setup and calibration are given in Choisel and Wickelmaier (2006).

The attribute elicitation procedure is described in Wickelmaier and Ellermeier (2007) and in Choisel and Wickelmaier (2006).

The selection of listeners for the experiments is described in Wickelmaier and Choisel (2005).

An extended version of this data set, including judgments on spatial and timbral auditory attributes and including listener variables, is available via `data("soundquality", package = "eba")`.

References

Choisel S, Wickelmaier F (2006). Extraction of Auditory Features and Elicitation of Attributes for the Assessment of Multichannel Reproduced Sound. *Journal of the Audio Engineering Society*, **54**(9), 815–826.

Choisel S, Wickelmaier F (2007). Evaluation of Multichannel Reproduced Sound: Scaling Auditory Attributes Underlying Listener Preference. *Journal of the Acoustical Society of America*, **121**(1), 388–400. doi:10.1121/1.2385043

Wickelmaier F, Choisel S (2005). Selecting Participants for Listening Tests of Multichannel Reproduced Sound. Presented at the AES 118th Convention, May 28–31, Barcelona, Spain, convention paper 6483.

Wickelmaier F, Ellermeier W (2007). Deriving Auditory Features from Triadic Comparisons. *Perception & Psychophysics*, **69**(2), 287–297. doi:10.3758/BF03193750

See Also

`paircomp`

Examples

```
data("SoundQuality", package = "psychotools")
summary(SoundQuality$preference)
ftable(xtabs(~ time + repet + progmat, data = SoundQuality))
```

SourceMonitoring *Performance in a Source-Monitoring Experiment*

Description

Response frequencies of 128 participants who took part in a source-monitoring experiment with two sources.

Usage

```
data("SourceMonitoring")
```

Format

A data frame containing 128 observations on four components.

sources Factor. Sources A and B.

age Integer. Age of the respondents in years.

gender Factor coding gender.

y Matrix containing the response frequencies. The column names indicate the nine response categories:

a.a	Number of source A items judged to be of source A.
a.b	Number of source A items judged to be of source B.
a.n	Number of source A items judged to be new.
b.a	Number of source B items judged to be of source A.
b.b	Number of source B items judged to be of source B.
b.n	Number of source B items judged to be new.
n.a	Number of new items judged to be of source A.
n.b	Number of new items judged to be of source B.
n.n	Number of new items judged to be new.

Details

In a source-monitoring experiment with two sources, participants study items from two different sources, A and B. The final memory test consists of A and B items along with new distractor items, N. Participants are required to classify each item as A, B, or N.

In an experiment at the Department of Psychology, University of Tuebingen (Wickelmaier & Zeileis, 2013, 2018), two source conditions were used in the study phase: Half of the subjects had to read items either quietly (source A = think) or aloud (source B = say). The other half had to write items down (source A = write) or read them aloud (source B = say).

The data were analyzed using the multinomial processing tree model of source monitoring (Batchelder & Riefer, 1990).

Source

Wickelmaier F, Zeileis A (2013). A First Implementation of Recursive Partitioning for Multinomial Processing Tree Models. Presented at the Psychoco 2013 International Workshop on Psychometric Computing, February 14–15, Zurich, Switzerland.

References

Batchelder WH, Riefer DM (1990). Multinomial Processing Tree Models of Source Monitoring. *Psychological Review*, **97**, 548–564.

Wickelmaier F, Zeileis A (2018). Using Recursive Partitioning to Account for Parameter Heterogeneity in Multinomial Processing Tree Models. *Behavior Research Methods*, **50**(3), 1217–1233. doi:10.3758/s134280170937z

Examples

```
data("SourceMonitoring", package = "psychotools")
xtabs(~ gender + I(age >= 30) + sources, SourceMonitoring)
```

StereotypeThreat *Stereotype Threat in Dutch Differential Aptitude Test*

Description

Cross-section data from Differential Aptitude Test (DAT) among Dutch highschool students, along with experimental conditions pertaining to stereotype threat.

Usage

```
data("StereotypeThreat")
```

Format

A data frame containing 295 observations on 11 variables.

condition Factor indicating experimental condition: `"control"` or stereotype `"threat"`, for details see below.

ethnicity Factor coding ethnicity: Dutch `"majority"` or `"minority"`.

numerical Number of items solved in numerical ability subtest (out of 14 complicated mathematical items).

abstract Number of items solved in abstract reasoning subtest (out of 18 items with a logical sequence of diagrams).

verbal Number of items solved in verbal reasoning subtest (out of 16 verbal analogy items).

gender Factor indicating gender.

age Age in years.

vintelligence Numerical coding of the value of one's own intelligence. Answer to: How important is your intelligence for you? Range is from very important (5) to unimportant (1).

vgrades Numerical coding of the value of getting good grades. Answer to: How much do you value getting good school grades? Range is from a lot of value (5) to not so much value (1).

vprejudice Numerical coding of the answer to: Do you think that people of your group are prejudiced against? Range is from certainly (5) to not at all (1).

gpa Numerical grade point average on 10-point scale (with 10 being the best grade). It has 57 missing values as some schools were either unwilling to share the data or did not provide it timely enough.

Details

The data are taken from Study 1 of Wicherts et al. (2005) and have been used to study stereotype threat on intelligence test performance among Dutch highschool students.

On average, Dutch minority students attain lower educational levels compared to Dutch majority students and studies have shown that minority students are often viewed as less smart/educated. Conversely, minorities often feel discriminated against in scholastic domains.

Wicherts et al. (2005) administered an intelligence test consisting of three subtests (for numerical ability, abstract reasoning, and verbal reasoning) and varied the amount of stereotype threat related to ethnic minorities by changing the presentation of the test. In the `"threat"` condition, the questions were declared to be part of an intelligence test and also an ethnicity questionnaire was conducted prior to the DAT. In the `"control"` condition, intelligence was not mentioned and no ethnicity questionnaire was conducted.

The variables `numerical`, `abstract`, and `verbal` can be used to assess ability/intelligence. And the `vintelligence`, `vgrades`, `vprejudice`, and `gpa` variables capture identification with the scholastic domain.

See Wicherts et al. (2005) for details.

Source

Provided by Jelte M. Wicherts.

References

Wicherts JM, Conor VD, Hessen DJ (2005). Stereotype Threat and Group Differences in Test Performance: A Question of Measurement Invariance. *Journal of Personality and Social Psychology*, **89**(5), 696-716.

Examples

```
## Data: Load and include/order wrt group variable
data("StereotypeThreat", package = "psychotools")
StereotypeThreat <- transform(StereotypeThreat, group = interaction(ethnicity, condition))
StereotypeThreat <- StereotypeThreat[order(StereotypeThreat$group),]
```

```
## Exploratory analysis (Table 2, p. 703)
tab2 <- with(StereotypeThreat, rbind(
   "#"         = tapply(numerical, group, length),
   "Numerical" = tapply(numerical, group, mean),
   "         " = tapply(numerical, group, sd),
   "Abstract " = tapply(abstract,  group, mean),
   "         " = tapply(abstract,  group, sd),
   "Verbal   " = tapply(verbal,    group, mean),
   "         " = tapply(verbal,    group, sd)))
round(tab2, digits = 2)

## Corresponding boxplots
plot(numerical ~ group, data = StereotypeThreat)
plot(abstract  ~ group, data = StereotypeThreat)
plot(verbal    ~ group, data = StereotypeThreat)

## MANOVA (p. 703)
m <- lm(cbind(numerical, abstract, verbal) ~ ethnicity * condition, data = StereotypeThreat)
anova(m, update(m, . ~ . - ethnicity:condition))
## corresponding univariate results
printCoefmat(t(sapply(summary(m),
  function(x) x$coefficients["ethnicityminority:conditionthreat", ])))

## MGCFA (Table 3, p. 704)
## can be replicated using package lavaan
## Not run:
## convenience function for multi-group CFA on this data
mgcfa <- function(model, ...) cfa(model, data = StereotypeThreat,
  group = "group", likelihood = "wishart", start = "simple", ...)

## list of all 9 models
m <- vector("list", length = 9)
names(m) <- c("m2", "m2a", "m3", "m3a", "m4", "m5", "m5a", "m5b", "m6")

## Step 2: Fix loadings across groups
f <- 'ability =~ abstract + verbal + numerical'
m$m2 <- mgcfa(f, group.equal = "loadings")

## Step 2a: Free numerical loading in group 4 (minority.threat)
f <- 'ability =~ abstract + verbal + c(l1, l1, l1, l4) * numerical'
m$m2a <- mgcfa(f, group.equal = "loadings")

## Step 3: Fix variances across groups
m$m3 <- mgcfa(f, group.equal = c("loadings", "residuals"))

## Step 3a: Free numerical variance in group 4
f <- c(f, 'numerical ~~ c(e1, e1, e1, e4) * numerical')
m$m3a <- mgcfa(f, group.equal = c("loadings", "residuals"))

## Step 4: Fix latent variances within conditions
f <- c(f, 'ability ~~ c(vmaj, vmin, vmaj, vmin) * ability')
m$m4 <- mgcfa(f, group.equal = c("loadings", "residuals"))
```

```
## Step 5: Fix certain means, free others
f <- c(f, 'numerical ~ c(na1, na1, na1, na4) * 1')
m$m5 <- mgcfa(f, group.equal = c("loadings", "residuals", "intercepts"))

## Step 5a: Free ability mean in group majority.control
f <- c(f, 'abstract ~ c(ar1, ar2, ar2, ar2) * 1')
m$m5a <- mgcfa(f, group.equal = c("loadings", "residuals", "intercepts"))

## Step 5b: Free also ability mean in group minority.control
f <- c(f[1:4], 'abstract ~ c(ar1, ar2, ar3, ar3) * 1')
m$m5b <- mgcfa(f, group.equal = c("loadings", "residuals", "intercepts"))

## Step 6: Different latent mean structure
f <- c(f, 'ability ~  c(maj, min, maj, min) * 1 + c(0, NA, 0, NA) * 1')
m$m6 <- mgcfa(f, group.equal = c("loadings", "residuals", "intercepts"))

## Extract measures of fit
tab <- t(sapply(m, fitMeasures, c("chisq", "df", "pvalue", "rmsea", "cfi")))
tab <- rbind("1" = c(0, 0, 1, 0, 1), tab)
tab <- cbind(tab,
  delta_chisq = c(NA, abs(diff(tab[, "chisq"]))),
  delta_df = c(NA, diff(tab[, "df"])))
tab <- cbind(tab, "pvalue2" = pchisq(tab[, "delta_chisq"],
  abs(tab[, "delta_df"]), lower.tail = FALSE))
tab <- tab[, c(2, 1, 3, 7, 6, 8, 4, 5)]
round(tab, digits = 3)

## End(Not run)
```

`subset.itemresp` *Subsetting Item Response Data*

Description

Subsetting and combining `"itemresp"` data objects.

Usage

```
  ## S3 method for class 'itemresp'
subset(x, items = NULL, subjects = NULL, ...)
```

Arguments

`x`	an object of class `"itemresp"`.
`items`	character, integer, or logical for subsetting the items.
`subjects`	character, integer, or logical for subsetting the subjects.
`...`	currently not used.

Details

The subset method selects subsets of items and/or subjects in item response data. Alternatively, the [method can be used with the row index corresponding to subjects and the column index corresponding to items.

The c method can be used to combine item response data from different subjects for the same items

The merge method can be used to combine item response data from the same subjects for different items.

See Also

itemresp

Examples

```
## binary responses to three items, coded as matrix
x <- cbind(c(1, 0, 1, 0), c(1, 0, 0, 0), c(0, 1, 1, 1))
xi <- itemresp(x)

## subsetting/indexing
xi[2]
xi[-(3:4)]
xi[c(TRUE, TRUE, FALSE, FALSE)]
subset(xi, items = 1:2) # or xi[, 1:2]
subset(xi, items = -2, subjects = 2:3)

## combine two itemresp vectors for different subjects but the same items
xi12 <- xi[1:2]
xi34 <- xi[3:4]
c(xi12, xi34)

## combine two itemresp vectors for the same subjects but different items
## polytomous responses in a data frame
d <- data.frame(q1 = c(-2, 1, -1, 0), q2 = factor(c(1, 3, 1, 3),
  levels = 1:3, labels = c("disagree", "neutral", "agree")))
di <-itemresp(d)
merge(xi, di)

## if subjects have names/IDs, these are used for merging
names(xi) <- c("John", "Joan", "Jen", "Jim")
names(di) <-         c("Joan", "Jen", "Jim", "Jo")
merge(xi, di)
merge(xi, di, all = TRUE)
```

---

subset.paircomp *Subsetting/Reordering Paired Comparison Data*

Description

Selection of subsets of objects to be compared and/or reordering of objects in "paircomp" data.

Usage

```
  ## S3 method for class 'paircomp'
reorder(x, labels, ...)
  ## S3 method for class 'paircomp'
subset(x, subset, select, ...)
```

Arguments

x	an object of class "paircomp".
labels, select	character or integer. Either a vector of (at least two) elements of labels(x) or an integer with their position. Partial string matching is enabled.
subset	currently not implemented. (Should be a specification of subsets of subjects.)
...	currently not used.

Details

The subset method currently just calls the reorder method.

See Also

paircomp

Examples

```
pc <- paircomp(rbind(
  c(1,  1,  1), # a > b, a > c, b > c
  c(1,  1, -1), # a > b, a > c, b < c
  c(1, -1, -1), # a > b, a < c, b < c
  c(1,  1,  1)))
reorder(pc, c("c", "a"))
```

---

summary.itemresp *Summarizing and Visualizing Item Response Data*

Description

Summarizing and visualizing "itemresp" data objects.

Usage

```
  ## S3 method for class 'itemresp'
summary(object, items = NULL, abbreviate = FALSE,
    mscale = TRUE, simplify = TRUE, sep = " ", ...)
  ## S3 method for class 'itemresp'
plot(x, xlab = "", ylab = "", items = NULL,
    abbreviate = FALSE, mscale = TRUE, sep = "\n", off = 2, axes = TRUE,
    names = TRUE, srt = 45, adj = c(1.1, 1.1), ...)
```

Arguments

object, x	an object of class "itemresp".
items	character or integer for subsetting the items to be summarized/visualized. By default, all items are used.
abbreviate	logical or integer. Should scale labels be abbreviated? Alternatively, an integer with the desired abbreviation length. The default is some heuristic based on the length of the labels.
mscale	logical. Should mscale values be used for printing/plotting? If FALSE, integers 0, 1, ... are used.
simplify	logical. Should the summary table be collapsed into a matrix or returned as a list?
sep	character. A character for separating item labels from their corresponding scale labels (if any).
xlab, ylab, off, axes, ...	arguments passed to spineplot.
names	logical or character. If TRUE, the names of the items are displayed on the x-axis. If FALSE, numbers of items are shown. Alternatively a character vector of the same length as the number of items can be supplied.
srt, adj	numeric. Angle (srt) and adjustment (adj) in case names (rather than numbers) are used as x-axis labels. These are passed to text.

Details

The plot method essentially just calls summary (passing on most further arguments) and then visualizes the result as a spineplot.

See Also

itemresp, spineplot

Examples

```
## summary/visualization for verbal aggression data
data("VerbalAggression", package = "psychotools")
r <- itemresp(VerbalAggression$resp[, 1:6])
mscale(r) <- c("no", "perhaps", "yes")
summary(r)
plot(r)

## modify formatting of mscale
summary(r, abbreviate = 1)
summary(r, mscale = FALSE)

## illustration for varying mscale across items
## merge with additional random binary response
b <- itemresp(rep(c(-1, 1), length.out = length(r)),
  mscale = c(-1, 1), labels = "Dummy")
rb <- merge(r[, 1:2], b)
```

```
head(rb, 2)
## summary has NAs for non-existent response categories
summary(rb)
summary(rb, mscale = FALSE)
plot(rb, srt = 25)
plot(rb, mscale = FALSE)
```

---

threshpar *Extract Threshold Parameters of Item Response Models*

Description

A class and generic function for representing and extracting the item threshold parameters of a given item response model.

Usage

```
  threshpar(object, ...)
  ## S3 method for class 'raschmodel'
threshpar(object, type = c("mode", "median", "mean"),
    ref = NULL, alias = TRUE, relative = FALSE, cumulative = FALSE, vcov = TRUE,
    ...)
  ## S3 method for class 'rsmodel'
threshpar(object, type = c("mode", "median", "mean"),
    ref = NULL, alias = TRUE, relative = FALSE, cumulative = FALSE, vcov = TRUE,
    ...)
  ## S3 method for class 'pcmodel'
threshpar(object, type = c("mode", "median", "mean"),
    ref = NULL, alias = TRUE, relative = FALSE, cumulative = FALSE, vcov = TRUE,
    ...)
  ## S3 method for class 'nplmodel'
threshpar(object, type = c("mode", "median", "mean"),
    ref = NULL, alias = TRUE, relative = FALSE, cumulative = FALSE, vcov = TRUE,
    ...)
  ## S3 method for class 'gpcmodel'
threshpar(object, type = c("mode", "median", "mean"),
    ref = NULL, alias = TRUE, relative = FALSE, cumulative = FALSE, vcov = TRUE,
    ...)
```

Arguments

object	a fitted model object whose threshold parameters should be extracted.
type	character of length one which determines the type of threshold parameters to return (see details below).
ref	a vector of labels or position indices of (relative) threshold parameters or a contrast matrix which should be used as restriction/for normalization. For partial credit models, argument ref can also be a list of contrasts. If NULL (the default),

for all models except models etimated via MML, the relative threshold parameters are centered around their item-specific means and the absolute threshold parameters are centered around their global mean. For models estimated via MML (nplmodels and gpcmodels), the parameters are by default identified via the distributional parameters of the person parameters (mean and variance of a normal distribution). Nevertheless, a restriction on the interval scale can be applied.

`alias` logical. If TRUE (the default), the aliased parameter is included in the return vector (and in the variance-covariance matrix if vcov = TRUE). If FALSE, it is removed. If the restriction given in `ref` depends on several parameters, the first parameter of the restriction specified is (arbitrarily) chosen to be removed if `alias` is FALSE.

`relative` logical. If set to FALSE (default), absolute item threshold parameters are returned. If set to TRUE, relative item threshold parameters with the contrast specified in argument `ref` are returned.

`cumulative` logical. If set to TRUE, cumulative threshold parameters are returned. These correspond to the cumulative sum over the absolute or relative item threshold parameters (after the restriction given in argument `ref` has been applied).

`vcov` logical. If TRUE (the default), the (transformed) variance-covariance matrix of the (relative) threshold parameters is attached as attribute vcov. If FALSE, a NA-matrix is attached.

`...` further arguments which are currently not used.

Details

`threshpar` is both, a class to represent threshold parameters of item response models as well as a generic function. The generic function can be used to extract the threshold parameters of a given item response model.

For objects of class `threshpar`, methods to standard generic functions `print` and `coef` can be used to print and extract the threshold parameters.

Depending on argument `type`, different item threshold parameters are returned. For `type = "mode"`, the returned item threshold parameters correspond to the location on the theta axis where the probability of category k equals the probability of category $k-1$. For Rasch and partial credit models, item threshold parameters of this type correspond directly to the estimated absolute item threshold parameters of these models. For `type = "median"`, the returned item threshold parameters correspond to the location on the theta axis where the probability of choosing category k or higher, i.e., $P(X_{ij} >= k)$, equals 0.5. For `type = "mean"`, the returned absolute item threshold parameters correspond to the location on the theta axis where the expected category response is in the middle between two categories, i.e. 0.5, 1.5, ... An illustration of these threshold parameters can be found on page 104 in Masters & Wright (1995).

Value

A named list with item threshold parameters of class `threshpar` and additional attributes `model` (the model name), `type` (the type of item threshold parameters returned, see details above), `ref` (the items or parameters used as restriction/for normalization), `relative` (whether relative or absolute item threshold parameters are returned), `cumulative` (whether the cumulative item threshold

parameters are returned), `alias` (either `FALSE` or a named character vector or list with the removed aliased parameters), and `vcov` (the estimated and adjusted variance-covariance matrix).

References

Masters GN, Wright BD (1997). The Partial Credit Model. In Van der Linden WJ, Hambleton RK (eds.). *Handbook of Modern Item Response Theory*. Springer, New York.

See Also

`personpar`, `itempar`, `discrpar`, `guesspar`, `upperpar`

Examples

```
o <- options(digits = 4)

## load verbal aggression data
data("VerbalAggression", package = "psychotools")

## fit a rasch model to dichotomized verbal aggression data
raschmod <- raschmodel(VerbalAggression$resp2)

## extract threshold parameters with sum zero restriction
tr <- threshpar(raschmod)
tr

## compare to item parameters (again with sum zero restriction)
ip <- itempar(raschmod)
ip

all.equal(coef(tr), coef(ip))

## rating scale model example
rsmod <- rsmodel(VerbalAggression$resp)
trmod <- threshpar(rsmod, type = "mode")
trmed <- threshpar(rsmod, type = "median")
trmn <- threshpar(rsmod, type = "mean")

## compare different types of threshold parameters
cbind("Mode" = coef(trmod, type = "vector"),
 "Median" = coef(trmod, type = "vector"),
 "Mean" = coef(trmn, type = "vector"))

if(requireNamespace("mirt")) {
## fit a partial credit model and a generalized partial credit model
pcmod <- pcmodel(VerbalAggression$resp)
gpcmod <- gpcmodel(VerbalAggression$resp)

## extract the threshold parameters with different default restrictions and
## therefore incompareable scales
tp <- threshpar(pcmod)
tg <- threshpar(gpcmod)
```

```
plot(unlist(tp), unlist(tg), xlab = "PCM", ylab = "GPCM")
abline(a = 0, b = 1)

## extract the threshold parameters with the first as the reference leading
## to a compareable scale visualizing the differences due to different
## discrimination parameters
tp <- threshpar(pcmod, ref = 1)
tg <- threshpar(gpcmod, ref = 1)
plot(unlist(tp), unlist(tg), xlab = "PCM", ylab = "GPCM")
abline(a = 0, b = 1)

options(digits = o$digits)
}
```

---

upperpar *Extract Upper Asymptote Parameters of Item Response Models*

Description

A class and generic function for representing and extracting the upper asymptote parameters of a given item response model.

Usage

```
  upperpar(object, ...)
  ## S3 method for class 'raschmodel'
upperpar(object, alias = TRUE, vcov = TRUE, ...)
  ## S3 method for class 'rsmodel'
upperpar(object, alias = TRUE, vcov = TRUE, ...)
  ## S3 method for class 'pcmodel'
upperpar(object, alias = TRUE, vcov = TRUE, ...)
  ## S3 method for class 'nplmodel'
upperpar(object, alias = TRUE, logit = FALSE, vcov = TRUE, ...)
  ## S3 method for class 'gpcmodel'
upperpar(object, alias = TRUE, vcov = TRUE, ...)
```

Arguments

object	a fitted model object whose upper asymptote parameters should be extracted.
alias	logical. If TRUE (the default), the aliased parameters are included in the return vector (and in the variance-covariance matrix if vcov = TRUE). If FALSE, these parameters are removed. For raschmodels, rsmodels, pcmodels and gpcmodels, where all upper asymptote parameters are fixed to 1, this means that an empty numeric vector and an empty variance-covariace matrix is returned if alias is FALSE.
logit	logical. If a nplmodel of type "3PLu" or "4PL" model has been fit, the upper asymptote parameters were estimated on the logit scale. If logit = FALSE, these estimates and the variance-covariance (if requested) are retransformed using the logistic function and the delta method.

vcov	logical. If TRUE (the default), the variance-covariance matrix of the upper asymptote parameters is attached as attribute vcov.
...	further arguments which are currently not used.

Details

upperpar is both, a class to represent upper asymptote parameters of item response models as well as a generic function. The generic function can be used to extract the upper asymptote parameters of a given item response model.

For objects of class upperpar, several methods to standard generic functions exist: print, coef, vcov. coef and vcov can be used to extract the upper asymptote parameters and their variance-covariance matrix without additional attributes.

Value

A named vector with upper asymptote parameters of class upperpar and additional attributes model (the model name), alias (either TRUE or a named numeric vector with the aliased parameters not included in the return value), logit (indicating whether the estimates are on the logit scale or not), and vcov (the estimated and adjusted variance-covariance matrix).

See Also

personpar, itempar, threshpar, discrpar, guesspar

Examples

```
if(requireNamespace("mirt")) {

o <- options(digits = 3)

## load simulated data
data("Sim3PL", package = "psychotools")

## fit 2PL to data simulated under the 3PLu
twoplmod <- nplmodel(Sim3PL$resp2)

## extract the upper asymptote parameters (all fixed at 1)
up1 <- upperpar(twoplmod)

## fit 3PLu to data simulated under the 3PLu
threeplmodu <- nplmodel(Sim3PL$resp2, type = "3PLu")

## extract the upper asymptote parameters
up2 <- upperpar(threeplmodu)

## extract the standard errors
sqrt(diag(vcov(up2)))

## extract the upper asymptote parameters on the logit scale
up2_logit <- upperpar(threeplmodu, logit = TRUE)
```

```
## along with the delta transformed standard errors
sqrt(diag(vcov(up2_logit)))

options(digits = o$digits)
}
```

VerbalAggression *Situation-Response Questionnaire on Verbal Aggression*

Description

Responses of 316 subjects to 24 items describing possible reactions to 4 different frustrating situations.

Usage

```
data("VerbalAggression")
```

Format

A data frame containing 316 observations on 4 variables.

resp Item response matrix with values 0/1/2 coding no/perhaps/yes, respectively.

resp2 Dichotomized item response matrix with perhaps/yes merged to 1.

gender Factor coding gender.

anger Trait anger, assessed by the Dutch adaptation of the state-trait anger scale (STAS).

Details

The 24 items are constructed by factorial combination of four different frustrating situations (see below), three possible verbally aggressive responses (curse, scold, shout), and two behavioural models (want, do). The four situations are

S1: A bus fails to stop for me.
S2: I miss a train because a clerk gave me faulty information.
S3: The grocery store closes just as I am about to enter.
S4: The operator disconnects me when I used up my last 10 cents for a call.

Note that the first two situations are other-to-blame situations, and the latter two are self-to-blame situations.

The subjects were 316 first-year psychology students from a university in the Dutch speaking part of Belgium. Participation was a partial fulfillment of the requirement to participate in research. The sample consists of 73 males and 243 females, reflecting the gender proportion among psychology students. The average age was 18.4.

Source

Online materials accompanying De Boeck and Wilson (2004).

References

De Boeck, P., Wilson, M. (eds) (2004). Explanatory Item Response Models: A Generalized Linear and Nonlinear Approach. New York: Springer-Verlag.

Smits, D.J.M., De Boeck, P., Vansteelandt, K. (2004). The Inhibition of Verbally Aggressive Behaviour *European Journal of Personality*, **18**, 537-555. doi:10.1002/per.529

See Also

raschmodel

Examples

```
data("VerbalAggression", package = "psychotools")

## Rasch model for the self-to-blame situations
m <- raschmodel(VerbalAggression$resp2[, 1:12])
plot(m)

## IGNORE_RDIFF_BEGIN
summary(m)
## IGNORE_RDIFF_END
```

---

worth *Extract Worth Parameters*

Description

Generic functions for extracting worth parameters from paired comparison models.

Usage

```
worth(object, ...)
```

Arguments

object	an object.
...	arguments passed to methods.

Details

Since version 0.3-0, calls to worth are internally passed over to itempar.

See Also

btmodel, raschmodel

Examples

```
o <- options(digits = 4)

## data
data("GermanParties2009", package = "psychotools")

## Bradley-Terry model
bt <- btmodel(GermanParties2009$preference)

## worth parameters
worth(bt)

## or
itempar(bt)

options(digits = o$digits)
```

YouthGratitude *Measuring Gratitude in Youth*

Description

Cross-section data on several gratitude scales for children and adolescents.

Usage

```
data("YouthGratitude")
```

Format

A data frame containing 1405 observations on 28 variables.

id Integer person ID.

age Age in years (10–19 years).

agegroup Factor coding of age with levels "10-11", "12-13", "14", "15", "16", "17-19".

losd_1 Life has been good to me.

losd_2 There never seems to be enough to go around, and I never seem to get my share. (Reverse scored.)

losd_3 I really don't think that I've gotten all the good things that I deserve in life. (Reverse scored.)

losd_4 More bad things have happened to me in my life than I deserve. (Reverse scored.)

losd_5 Because of what I've gone through in my life, I really feel like the world owes me something. (Reverse scored.)

losd_6 For some reason I never seem to get the advantages that others get. (Reverse scored.)

sa_1 Oftentimes I have been overwhelmed at the beauty of nature.

sa_2 Every Fall I really enjoy watching the leaves change colors.

sa_3 I think that it's important to 'Stop and smell the roses.'

sa_4 I think that it's important to pause often to 'count my blessings.'

sa_5 I think it's important to enjoy the simple things in life.

sa_6 I think it's important to appreciate each day that you are alive.

ao_1 I couldn't have gotten where I am today without the help of many people.

ao_2 Although I think it's important to feel good about your accomplishments, I think that it's also important to remember how others have contributed to my accomplishments.

ao_3 Although I'm basically in control of my life, I can't help but think about all those who have supported me and helped me along the way.

ao_4 I feel deeply appreciative for the things others have done for me in my life.

gq6_1 I have so much in life to be thankful for.

gq6_2 If I had to list everything that I felt thankful for, it would be a very long list.

gq6_3 When I look at the world, I don't see much to be thankful for.

gq6_4 I am thankful to a wide variety of people. (Reverse scored.)

gq6_5 As I get older I find myself more able to appreciate the people, events, and situations that have been part of my life history.

gq6_6 Long amounts of time can go by before I feel gratitude to something or someone. (Reverse scored.)

gac_1 Grateful.

gac_2 Thankful.

gac_3 Appreciative.

Details

The gratitude scales employed are:
GRAT: Gratitude, Resentment, Appreciation Test (1–9).
Short form with subscales LOSD (lack of a sense of deprivation), SA (simple appreciation), and AO (appreciation for others).
GQ-6: Gratitude Questionnaire-6 (1–7).
GAC: Gratitude Adjective Checklist (1–5).

The item `losd_1` has been omitted from all analyses in Froh et al. (2011) because it loaded lowly on all factors. Hence `losd_1` is not listed in Table B1 of Froh et al. (2011). Instead, the remaining items are labeled `losd_1` to `losd_5`.

Source

Provided by Jeff Froh and Jinyan Fan.

References

Froh JJ, Fan J, Emmons RA, Bono G, Huebner ES, Watkins P (2011). Measuring Gratitude in Youth: Assessing the Psychometric Properties of Adult Gratitude Scales in Children and Adolescents. *Psychological Assessment*, **23**(2), 311–324.

Examples

```
data("YouthGratitude", package = "psychotools")
summary(YouthGratitude)

## modeling can be carried out using package lavaan
## Not run:
## remove cases with 'imputed' values (not in 1, ..., 9)
yg <- YouthGratitude[apply(YouthGratitude[, 4:28], 1, function(x) all(x

## GQ-6
gq6_congeneric <- cfa(
  'f1 =~ gq6_1 + gq6_2 + gq6_3 + gq6_4 + gq6_5',
  data = yg, group = "agegroup", meanstructure = TRUE)
gq6_tauequivalent <- cfa(
  'f1 =~ gq6_1 + gq6_2 + gq6_3 + gq6_4 + gq6_5',
  data = yg, group = "agegroup", meanstructure = TRUE,
  group.equal = "loadings")
gq6_parallel <- cfa(
  'f1 =~ gq6_1 + gq6_2 + gq6_3 + gq6_4 + gq6_5',
  data = yg, group = "agegroup", meanstructure = TRUE,
  group.equal = c("loadings", "residuals", "lv.variances"))
anova(gq6_congeneric, gq6_tauequivalent, gq6_parallel)
t(sapply(
  list(gq6_congeneric, gq6_tauequivalent, gq6_parallel),
  function(m) fitMeasures(m)[c("chisq", "df", "cfi", "srmr")]
))

## GAC
gac_congeneric <- cfa(
  'f1 =~ gac_1 + gac_2 + gac_3',
  data = yg, group = "agegroup", meanstructure = TRUE)
gac_tauequivalent <- cfa(
  'f1 =~ gac_1 + gac_2 + gac_3',
  data = yg, group = "agegroup", meanstructure = TRUE,
  group.equal = "loadings")
gac_parallel <- cfa(
  'f1 =~ gac_1 + gac_2 + gac_3',
  data = yg, group = "agegroup", meanstructure = TRUE,
  group.equal = c("loadings", "residuals", "lv.variances"))
anova(gac_congeneric, gac_tauequivalent, gac_parallel)
t(sapply(
  list(gac_congeneric, gac_tauequivalent, gac_parallel),
  function(m) fitMeasures(m)[c("chisq", "df", "cfi", "srmr")]
))

## GRAT
grat_congeneric <- cfa(
  'f1 =~ losd_2 + losd_3 + losd_4 + losd_5 + losd_6
   f2 =~ sa_1 + sa_2 + sa_3 + sa_4 + sa_5 + sa_6
   f3 =~ ao_1 + ao_2 + ao_3 + ao_4',
  data = yg, group = "agegroup", meanstructure = TRUE)
grat_tauequivalent <- cfa(
```

```
  'f1 =~ losd_2 + losd_3 + losd_4 + losd_5 + losd_6
   f2 =~ sa_1 + sa_2 + sa_3 + sa_4 + sa_5 + sa_6
   f3 =~ ao_1 + ao_2 + ao_3 + ao_4',
  data = yg, group = "agegroup", meanstructure = TRUE,
  group.equal = "loadings")
grat_parallel <- cfa(
  'f1 =~ losd_2 + losd_3 + losd_4 + losd_5 + losd_6
   f2 =~ sa_1 + sa_2 + sa_3 + sa_4 + sa_5 + sa_6
   f3 =~ ao_1 + ao_2 + ao_3 + ao_4',
  data = yg, group = "agegroup", meanstructure = TRUE,
  group.equal = c("loadings", "residuals", "lv.variances"))
anova(grat_congeneric, grat_tauequivalent, grat_parallel)
t(sapply(
  list(grat_congeneric, grat_tauequivalent, grat_parallel),
  function(m) fitMeasures(m)[c("chisq", "df", "cfi", "srmr")]
))

## End(Not run)
```

Index