Package: tppcm (via r-universe)

October 29, 2026

Type Package

Title Step Discriminations for the Two-Parameter Partial Credit Model on Top of 'TAM'

Version 0.1.0.9000

Description In the two-parameter partial credit model (TPPCM) of Yu (1991) <https://hdl.handle.net/2142/21459>, every step of a polytomous item has its own discrimination. Estimation is left to existing software: 'TAM' (tam.mml.3pl() with a step design matrix, tam.mml.2pl(), tam.mml()) and, for the bilinear product form and non-negativity constraints, 'sirt' (xxirt()). The package adds what these packages do not provide: standard errors of step discriminations based on the full observed information, step-level Wald tests, and score tests with modification indices and expected parameter changes for moving down the model ladder saturated TPPCM -> product form -> generalized partial credit model -> partial credit model, limited-information fit statistics (the M2 statistic, root mean square error of approximation, standardized root mean squared residual, comparative fit index and Tucker-Lewis index; Maydeu-Olivares and Joe (2006) <doi:10.1007/s11336-005-1295-9>) for any step design, step contributions to the item information, and tests of differential step functioning across groups: per-step Lagrange multiplier tests from one multigroup fit (Glas (1998) <https://www3.stat.sinica.edu.tw/statistica/j8n3/j8n32/j8n32.htm>), with the generalized score test of Boos (1992) <doi:10.1080/00031305.1992.10475921>, likelihood ratio tests with configural, metric, scalar and partial multigroup designs, score-based tests (Merkle and Zeileis (2013) <doi:10.1007/s11336-012-9302-4>) and model-based trees (Zeileis, Hothorn and Hornik (2008) <doi:10.1198/106186008X319331>). The generalized partial credit model of Muraki (1992) <doi:10.1177/014662169201600206> is the special case with equal steps within items.

License GPL (>= 2)

URL https://github.com/jiewenTsai/tppcm

BugReports https://github.com/jiewenTsai/tppcm/issues

Encoding UTF-8

Language en-US

Depends R (>= 4.1), TAM

Imports CDM, graphics, MASS, sirt, stats

Suggests knitr, mirt, numDeriv, partykit, psychotools, psychotree, rmarkdown, strucchange, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Repository https://jiewentsai.r-universe.dev

Date/Publication 2026-10-28 16:35:15 UTC

RemoteUrl https://github.com/jiewentsai/tppcm

RemoteRef HEAD

RemoteSha 713bf87f23d811d211b0136ee490da70dd3a6cdc

Contents

dif_test . . . 3
dsf . . . 6
dsf_design . . . 8
get_parts . . . 10
irt_pars . . . 12
item_test . . . 13
m2 . . . 14
mi . . . 15
score_test . . . 16
sim_tppcm . . . 17
step_info . . . 18
tppcm . . . 19
tppcmtree . . . 21
wald_test . . . 23
xxirt_tppcm . . . 24
xxirt_vcov . . . 25

Index **26**

dif_test *Score (LM) tests of differential step and item functioning*

Description

Tests whether item parameters differ between groups, or vary with a person covariate, using only the fitted model (no refitting). The main use is the per-step DSF screen on a multigroup fit:

```
fit <- tam.mml.3pl(dat, E = tppcm(dat), group = g, est.variance = FALSE)
dif_test(fit, g, by = "step")
```

which tests, for every category boundary, its discrimination (CBD) and location jointly across the groups: the Lagrange multiplier (score) test for DIF of Glas (1998), with 2 df per group contrast. The units of flagged steps (`"item:step"`) can be passed to `dsf(fit, items = )` as they are, to tell CBD from location DSF by likelihood ratio tests. See tppcm-package for the parameter names (CBD, CBL, IBD).

Usage

```
dif_test(
  x,
  covariate,
  parm = c("disc", "diff"),
  by = c("param", "step", "item", "block"),
  functional = NULL,
  param = c("irt", "si"),
  info = c("robust", "louis", "opg"),
  adjust = "holm"
)
```

Arguments

`x`	A fitted model (see `get_parts()`) or a `get_parts` object.
`covariate`	Person covariate in the order of the fitted responses (no missing values), usually the grouping of a multigroup fit. A factor (or character, logical) gives the LM test, an ordered factor the weighted maximum LM test (`"maxLMo"`), a numeric covariate the Cramer-von Mises type test (`"CvM"`).
`parm`	Parameters to test: `c("disc", "diff")` (default, both blocks), `"disc"` (or `"slope"`; slope block), `"diff"` (or `"int"`; location block), or column names of `get_parts(fit, "estfun")` (`get_parts(fit, "estfun.si")` with `param = "si"`). `"all"` is accepted for `c("disc", "diff")`.
`by`	Testing unit: every parameter (`"param"`), every step (`"step"`: discrimination and location jointly, units named `"item:step"`; needs both blocks and a TPPCM step design), every item (`"item"`) or all selected parameters together (`"block"`).

functional	NULL (chosen from the covariate type), one of "LM", "maxLMo", "DM", "CvM", "maxLM", or a **strucchange** functional. Not used by the split LM test of multigroup fits.
param	Parameterization of the tested locations: "irt" (disc, diff: kernel disc * (theta - diff)) or "si" (slope, int: kernel slope * theta + int). by = "step" does not depend on it.
info	Information used by the split LM test of multigroup fits: "robust" (default), "louis" or "opg"; see Details. Ignored otherwise.
adjust	Multiple-testing adjustment of p_adj over all rows, a method of stats::p.adjust() ("none" for none).

Details

Multigroup fits (recommended). Fit the model with the grouping (tam.mml.3pl(..., group = g, est.variance = FALSE)): the latent means and variances of groups 2, ..., G are then estimated and only item-level differences remain in the scores. For a factor covariate the test is the score test of the model in which the tested parameters are split by the levels of the covariate, with npar * (levels - 1) degrees of freedom. By default (info = "robust") it is the generalized score test of Boos (1992): the observed (Louis) information of each level as the Hessian and the outer product of the case-wise scores in the middle, which keeps its level when the model does not fit exactly (as with most rating-scale data; Falk & Monroe, 2018; Guastadisegni et al., 2021). In simulations of per-step tests (two groups, 300 or 600 persons per group, with and without local dependence) the robust test kept its level in every condition; it is slightly conservative with 300 per group (about .03), and with the size equalized it was as powerful as the likelihood ratio test. info = "louis" (model-based) agrees closely with the likelihood ratio test without refitting and shares its slight excess of rejections in small samples; info = "opg" is liberal and is for comparison only.

Which unit. by = "step" is the screen: the joint test of a step has a clean null (the boundary is the same in all groups) and does not depend on the parameterization (CBD and CBL, or CBD and IBD, give the same statistic); it is computed in the slope-intercept form, so a step with discrimination near 0 keeps both degrees of freedom. With by = "param" the 1-df disc row of a step tests its CBD with its location (diff, the CBL) held equal across groups (with param = "si": its intercept, the IBD), so location DSF leaks into it: in a simulation with 600 persons per group, a CBL shift of 0.5 in one step was flagged by its disc row in 16.5% of the replications. Use dsf() to decompose a flagged step instead. by = "item" and "block" test all selected parameters of an item or of the test jointly.

Impact is not DIF. A single-group fit assumes one latent distribution for everybody. When the covariate is related to the latent trait (groups that differ in ability, a numeric covariate correlated with it), the scores of all parameters move with the covariate and most items are flagged. Fit the multigroup model (group =) or the latent regression (Y =) with that covariate first and test on that fit.

Single-group fits and other covariates. Without groups in the fit (or for a numeric or ordered covariate) the case-wise scores (get_parts(fit, "estfun")) are ordered by the covariate and aggregated with a functional from **strucchange** (Merkle & Zeileis, 2013; Strobl, Kopf & Zeileis, 2015). Each tested unit is replaced by its efficient scores (residuals from the regression on all other parameters, outer-product information), so DIF in the other parameters does not leak into the test.

Value

Data frame (class `tppcm_table`) in the order of the items, with `unit` (parameter name, `"item:step"`, item name or `"all"`), `npar` (number of parameters tested), `statistic`, `df` (chi-square degrees of freedom of the LM tests, `npar * (levels - 1)`; `NA` for the other functionals), `p` and `p_adj`.

References

Boos, D. D. (1992). On generalized score tests. *The American Statistician, 46*, 327-333.

Falk, C. F., & Monroe, S. (2018). On Lagrange multiplier tests in multidimensional item response theory: Information matrices and model misspecification. *Educational and Psychological Measurement, 78*, 653-678.

Glas, C. A. W. (1998). Detection of differential item functioning using Lagrange multiplier tests. *Statistica Sinica, 8*, 647-667.

Guastadisegni, L., Cagnone, S., Moustaki, I., & Vasdekis, V. (2021). Use of the Lagrange multiplier test for assessing measurement invariance under model misspecification. *Educational and Psychological Measurement*. doi:10.1177/00131644211020355

Merkle, E. C., & Zeileis, A. (2013). Tests of measurement invariance without subgroups: A generalization of classical methods. *Psychometrika, 78*, 59-82.

Strobl, C., Kopf, J., & Zeileis, A. (2015). Rasch trees: A new method for detecting differential item functioning in the Rasch model. *Psychometrika, 80*, 289-316.

See Also

`dsf()` for likelihood ratio tests of single step parameters, `dsf_design()` for configural, metric and scalar models, `tppcmtree()` for unknown groupings.

Examples

```
set.seed(1)
a <- matrix(c(1, 1.5, 1.2), 5, 3, byrow = TRUE)
b <- matrix(c(-1, 0, 1), 5, 3, byrow = TRUE)
a2 <- a; a2[3, 2] <- 2.5                      # DSF in the CBD of item 3, step 2
dat <- rbind(sim_tppcm(600, a, b),
             sim_tppcm(600, a2, b, theta = rnorm(600, 0.3, 0.8)))   # impact
g <- rep(c("ref", "foc"), each = 600)
fit <- tam.mml.3pl(dat, E = tppcm(dat), group = g, est.variance = FALSE,
                   verbose = FALSE)
st <- dif_test(fit, g, by = "step")           # screen: every step, 2 df
st
st$unit[st$p_adj < 0.05]                      # flagged steps, e.g. for dsf()
dif_test(fit, g, by = "item")                 # every item jointly
```

dsf *Likelihood ratio tests of differential step functioning*

Description

Partial invariance search at the level of category boundaries, in the way of `group.equal` and `group.partial` in lavaan. The baseline model is the multigroup TPPCM with the step parameters in `group.equal` equal across groups (except the steps in `group.partial`); each restricted parameter is then released one at a time and compared with the baseline by a likelihood ratio test (df = number of groups - 1). The latent means and variances of groups 2, ..., G are estimated as far as the baseline identifies them, and all other steps serve as anchors. Use it to decompose the steps flagged by the joint screen `dif_test(fit, g, by = "step")`: `dsf(fit, items = c("I2:2", "I5:1"))`.

Usage

```
dsf(
  fit,
  group.equal = c("cbd", "ibd"),
  group.partial = NULL,
  items = NULL,
  steps = NULL,
  adjust = "holm",
  control = NULL,
  verbose = TRUE
)
```

Arguments

`fit`	A multigroup `TAM::tam.mml.3pl()` fit with a step design `E` from `tppcm()` and `group =` , e.g. `tam.mml.3pl(dat, E = tppcm(dat), group = g, est.variance = FALSE)`.
`group.equal`	`c("cbd", "ibd")` (default) or `"cbd"`; `"cbl"` may be used for `"ibd"`.
`group.partial`	Steps already released in the baseline, as in `dsf_design()`; they are not tested.
`items`	Items (names) or single steps (`"item:step"`, as in `group.partial` and the units of `dif_test(by = "step")`) to test; default all.
`steps`	Steps to test within `items` (numbers); default all.
`adjust`	Multiple-testing adjustment of `p_adj` over all rows, a method of `stats::p.adjust()` (`"none"` for none).
`control`	TAM `control` list for the refits; default the fit's control with `conv <= 1e-6`, `convD <= 1e-5` and `maxiter >= 3000`.
`verbose`	Print progress.

Details

- `group.equal = c("cbd", "ibd")` (default, scalar baseline): every step gives two tests. The IBD (location) test releases the IBD of the step. The CBD test releases the CBD of the step with its location free, i.e. it compares the model with CBD and IBD of the step released against the model with only the IBD released, so that it does not depend on the origin of the scale and location DSF does not leak into it. The two LR statistics of a step add up to the joint 2-df LR test of the step.
- `group.equal = "cbd"` (metric baseline, all IBDs free): one CBD test per step, as in testing loadings at the metric stage.

`"cbl"` may be written for `"ibd"`: the location rows then report the CBL (CBL = -IBD / CBD) instead of the IBD; with equal CBDs the two restrictions are the same. As in invariance testing, interpret the location of a step only when its CBD is invariant. A significant test can also reflect DSF in other steps, which are assumed invariant (anchor contamination); with many tests use `p_adj`. See tppcm-package for CBD, CBL and IBD.

Only the data, the groups, the within-group step design (`design` or `index` of `tppcm()`) and the TAM control settings are taken from `fit`; the baseline is refitted with `dsf_design()`. Every test refits the model once or twice, so restrict `items` to the steps of interest for long tests.

Value

Data frame (class `tppcm_table`) with one row per test: `item`, `step`, `par` (`"cbd"`, `"ibd"` or `"cbl"`), the estimate in every group from the model with the parameter released, `LR`, `df`, `p`, `p_adj`. Attributes `"impact"` (latent means and variances of the baseline) and `"loglik0"`.

References

Stark, S., Chernyshenko, O. S., & Drasgow, F. (2006). Detecting differential item functioning with confirmatory factor analysis and item response theory: Toward a unified strategy. *Journal of Applied Psychology, 91*, 1292-1306.

See Also

`dif_test()` for the score-based screen without refitting; `dsf_design()` for configural, metric, scalar and partial models.

Examples

```
set.seed(1)
a <- matrix(c(1, 1.5, 1.2), 4, 3, byrow = TRUE)
b <- matrix(c(-1, 0, 1), 4, 3, byrow = TRUE)
a2 <- a; a2[2, 2] <- 2.5
dat <- rbind(sim_tppcm(500, disc = a, diff = b),
             sim_tppcm(500, disc = a2, diff = b, theta = rnorm(500, 0.3, 0.8)))
g <- rep(c("ref", "foc"), each = 500)
fit <- TAM::tam.mml.3pl(dat, E = tppcm(dat), group = g, est.variance = FALSE,
                        verbose = FALSE)
dif_test(fit, g, by = "step")                # screen: joint 2-df test per step
dsf(fit, items = c("I2:2", "I3:2"), verbose = FALSE)          # decompose single steps
dsf(fit, group.equal = "cbd", items = "I2", verbose = FALSE)  # metric baseline
```

dsf_design *Multigroup TPPCM design with group.equal and group.partial*

Description

Builds virtual items and the arguments of `TAM::tam.mml.3pl()` for a multigroup TPPCM in which the step parameters are constrained across groups in the way of the `group.equal` and `group.partial` arguments of lavaan. Every item is split into one virtual item per group; a person answers only the virtual items of their own group.

Usage

```
dsf_design(
  dat,
  group,
  group.equal = c("cbd", "ibd"),
  group.partial = NULL,
  design = ~item * step,
  index = NULL,
  K = NULL
)
```

Arguments

`dat`	Data frame or matrix of item responses coded `0, 1, ..., K_i - 1`.
`group`	Vector of group memberships, one per row of `dat`, without missing values.
`group.equal`	`NULL`, `"cbd"`, or `c("cbd", "ibd")` (default); `"cbl"` may be used for `"ibd"`.
`group.partial`	Steps released from `group.equal`: a list with elements `cbd` and/or `ibd` (or `cbl`), or a character vector for all restrictions.
`design`	Structure of the step discriminations within each group, a linear design formula of `tppcm()`: `~ item * step` (default), `~ item`, `~ step` or `~ 1`.
`index`	Optional items x steps matrix of discrimination indices within a group, as in `tppcm()`; overrides `design`.
`K`	Number of categories per item, as in `tppcm()`.

Details

The step parameters are named as in the CBD literature (see tppcm-package): `"cbd"` (category boundary discrimination, the step discrimination or slope, cf. loadings) and `"ibd"` (intercept boundary difference, the step intercept, cf. intercepts or thresholds). With the CBDs of a step equal, equal IBDs and equal CBLs (category boundary locations, CBL = -IBD / CBD) are the same restriction, so `"cbl"` is accepted for `"ibd"`.

- `group.equal = NULL`: configural model (each group its own parameters; the latent mean and variance of every group are fixed at 0 and 1).
- `group.equal = "cbd"`: boundary-level metric model (CBDs equal, IBDs free; latent variances of groups 2, ..., G free, means fixed).
- `group.equal = c("cbd", "ibd")` (default): boundary-level scalar model (latent means and variances of groups 2, ..., G free).

`group.partial` releases single steps from these restrictions, as `list(cbd = "S03:2", ibd = c("S03:2", "S05"))` (`"item:step"` or an item name for all its steps); a character vector releases the steps from all restrictions in `group.equal`. Releasing the CBD of a step while its IBD is kept equal makes the boundary curves of the groups cross at `theta = 0` (the mean of the reference group), which depends on the origin of the scale; release both, or use `"cbl"`, in which case the IBD of a step with a released CBD is released as well (with unequal CBDs, equal CBLs are not a linear restriction).

The reference group is `sort(unique(group))[1]`, as in TAM. The identification of the latent means and variances is returned as `variance.fixed` and `beta.fixed`; pass both to `TAM::tam.mml.3pl()` together with `group`.

Value

A list of class `tppcm_dsf_design` with `resp` (virtual items, named `item_group`), `A`, `E`, `variance.fixed`, `beta.fixed` (arguments of `TAM::tam.mml.3pl()`), `index` and `index_int` (virtual items x steps parameter numbers of the CBDs and IBDs), `groups`, `equal`, `partial`.

See Also

`dsf()` for likelihood ratio tests step by step.

Examples

```
set.seed(1)
a <- matrix(c(1, 1.5, 1.2), 5, 3, byrow = TRUE)
b <- matrix(c(-1, 0, 1), 5, 3, byrow = TRUE)
g <- rep(1:2, each = 500)
dat <- rbind(sim_tppcm(500, disc = a, diff = b),
             sim_tppcm(500, disc = a, diff = b, theta = rnorm(500, 0.3, 0.8)))
fit_mg <- function(d) TAM::tam.mml.3pl(d$resp, A = d$A, E = d$E, group = g,
  variance.fixed = d$variance.fixed, beta.fixed = d$beta.fixed, verbose = FALSE)
configural <- fit_mg(dsf_design(dat, g, group.equal = NULL))
metric     <- fit_mg(dsf_design(dat, g, group.equal = "cbd"))
scalar     <- fit_mg(dsf_design(dat, g, group.equal = c("cbd", "ibd")))
anova(configural, metric)
anova(metric, scalar)
partial <- fit_mg(dsf_design(dat, g, group.partial = list(cbd = "I2:2", ibd = "I2:2")))
anova(scalar, partial)
```

`get_parts` *Components of a fitted model*

Description

Extracts the mathematical components behind the package functions from a fitted model, in the spirit of `lme4::getME()`, `lavaan::lavInspect()` and `mirt::extract.mirt()`. Without `what` it returns all components as one object that every function of the package accepts in place of the fit, so they are computed only once. Nothing is re-estimated: everything follows from the fitted item response functions, the posterior and the responses stored in the fit.

Usage

```
get_parts(x, what = NULL, model = NULL, fixed = NULL)
```

Arguments

`x`	A fitted model (see table) or the object returned by `get_parts()`.
`what`	`NULL` (all components) or one or more names from the table; several names give a named list.
`model`	Fitted model type; detected from `x` when `NULL`. One of `"E"`, `"tppcm"`, `"gpcm"`, `"pcm"`, `"pcm_fixed"`, `"rank1"`.
`fixed`	For `tam.mml.3pl` fits: indices of fixed slope parameters (default: those with `se.gammaslope == 0`).

Details

Supported fits (the model is detected automatically):

Fit	`model`	`formula`
`TAM::tam.mml.3pl()` with `tppcm()`	`"E"` (linear step design; fixed steps detected)	detected from `E`: `~ 1`, `~ item`, `~ s`
`TAM::tam.mml.2pl()`, `irtmodel = "2PL"`	`"tppcm"` (saturated TPPCM = NRM)	`~ item * step`
`TAM::tam.mml.2pl()`, `irtmodel = "GPCM"`	`"gpcm"`	`~ item`
`TAM::tam.mml()`	`"pcm"` (free latent variance)	`~ 1`
`xxirt_tppcm()`	`"tppcm"`, `"gpcm"`, `"rank1"`	its `design`

Components (`what`):

`what`	Content
`"disc"`, `"diff"`, `"int"`	items x steps matrices: discriminations, difficulties (`disc * (theta - diff)`), in
`"est"`, `"se"`	slope parameters of the fitted model (e.g. item slopes, α, γ) and their standard e
`"vcov"`	covariance of the fitted model's parameters
`"par"`, `"partable"`	saturated parameters (per item $a_1..a_L, d_1..d_L, d = a \cdot diff$) and their descriptio
`"vcov.sat"`, `"se.sat"`	implied covariance and standard errors of the saturated parameters
`"information"`, `"information.opg"`	observed (Louis) and outer-product information of the saturated parameters

"gradient", "scores"	score vector and case-wise scores of the saturated parameters
"estfun", "estfun.si"	case-wise scores of the fitted slopes and the step difficulties (disc, diff) or step
"jacobian", "curvature"	Jacobian of the fitted model in the saturated space and its curvature term
"posterior", "nodes", "prior", "probs"	posterior, quadrature nodes, latent distribution at the nodes, item response proba
"data", "nobs", "loglik", "npar"	responses, number of persons, log-likelihood, number of parameters
"model", "formula"	type of fit, and its design formula (see tppcm(); NULL for a custom design)
"groups", "design"	latent means and variances, design array E

The fitted model is a map $h(\vartheta)$ into the saturated parameters with Jacobian J; its observed information is $J'IJ - \sum_k g_k \nabla^2 h_k$, where I is the Louis information and g the score of the saturated model. The second term is non-zero only for the product form. The latent trait distribution is held fixed at its estimate.

Any unidimensional fit whose item response functions are of the divide-by-total form $\log P_k/P_{k-1} = a_k\theta - d_k$ is accepted, including binary items (one step), several groups, person weights (pweights; m2() and dif_test() then refuse) and latent regression (with a warning; regression coefficients are treated as fixed, m2() refuses). Fits with guessing parameters, many-facet models (tam.mml.mfr()), JML fits and multidimensional fits are refused.

For TAM fits the location parameters follow TAM's design matrix A, so restricted thresholds (e.g. tam.mml(irtmodel = "RSM")) and fixed xsi are taken into account. Multidimensional fits are not supported. With several groups (group = in TAM), the free latent means and variances of groups 2, ..., G are part of the parameter vector: their scores and information are included, the item standard errors account for them, and what = "groups" reports them with standard errors. Treating them as fixed would understate the item standard errors by up to 40% on real data.

Value

Without what, an object of class tppcm_parts with methods for print(), coef() (the slope parameters of the fitted model, as what = "est"), vcov(), logLik() and nobs(); otherwise the component(s). The internal elements of the object are not the components of the table: use get_parts(x, what), not x$what.

Examples

```
data(data.Students, package = "CDM")
sc <- data.Students[, c("sc1", "sc2", "sc3", "sc4")]
fit <- tam.mml.3pl(sc, E = tppcm(sc), est.variance = FALSE, verbose = FALSE)
get_parts(fit, "disc")                # step discriminations
get_parts(fit, c("loglik", "npar"))
x <- get_parts(fit)                   # all components, computed once
dim(get_parts(x, "information"))
irt_pars(x); wald_test(x)            # reuse without recomputing
```

irt_pars *Item and group parameters in a compact table*

Description

A clean report of a fitted TPPCM-family model, in the spirit of `coef(mod, IRTpars = TRUE, simplify = TRUE)` in mirt: one row per item, one column per parameter, standard errors in a matching table, and the latent mean and variance.

Usage

```
irt_pars(
  x,
  IRTpars = TRUE,
  se = TRUE,
  fit = TRUE,
  long = FALSE,
  level = 0.95,
  restricted = FALSE
)
```

Arguments

`x`	A fitted model (see `get_parts()`) or a `get_parts` object.
`IRTpars`	Report discriminations and difficulties (default) or slopes and intercepts.
`se`	Include standard errors.
`fit`	Include model fit: log-likelihood, number of parameters, AIC, BIC and the limited-information statistics of `m2()` (M2, RMSEA2 with confidence interval, SRMSR, CFI, TLI). M2 needs a single group; otherwise only the information criteria are shown.
`long`	Return a long table (one row per parameter) with standard errors and confidence limits instead of the compact report.
`level`	Confidence level for `long = TRUE` or `restricted = TRUE`.
`restricted`	Return the parameters of the fitted model instead, e.g. the item slopes of the GPCM or α_i, γ_l of the product form, with standard errors and confidence limits.

Details

With `IRTpars = TRUE` the step kernel is `disc * (theta - diff)`: `disc` is the step discrimination and `diff` the step difficulty (Masters' step difficulty; TAM's `beta + tau`; mirt's `b` with `IRTpars = TRUE`; `psychotools::threshpar()`). With `IRTpars = FALSE` it is `slope * theta + int`, where the step intercepts `int` cumulate to the category intercepts of mirt / psychotools (`d_k`), which equal `-AXsi_` in TAM. In the terms of the category boundary literature `disc` (= `slope`) is the CBD, `diff` the CBL and `int` the IBD (see tppcm-package). Standard errors come from the observed information of the fitted model (delta method for `diff`), not from TAM's `se.gammaslope`/`se.xsi`.

A step discrimination that is numerically 0 (an estimate on the bound of `xxirt_tppcm()`) or practically flat (`|disc| < 0.1`) makes the step difficulty `diff = -int / disc` meaningless; for such steps `diff` and its SE are left blank and the step is listed below the table. The slope-intercept form (`IRTpars = FALSE`) is well behaved for them.

With `fit = TRUE` the header includes `m2()`, which takes a few seconds for long tests (e.g. 24 items); use `fit = FALSE` to skip it.

Value

Object of class `irt_pars`: a list with `items` (estimates, items x parameters), `se` (standard errors or `NULL`), `group` (latent mean and variance per group), `model`, `label`, `IRTpars`, `fit` (the header) and `at_bound`, `near_zero` (names of step discriminations at or near 0). `as.data.frame()` gives the long table. With `long = TRUE` or `restricted = TRUE`, a data frame (class `tppcm_table`).

Examples

```
set.seed(1)
dat <- sim_tppcm(800, disc = outer(c(0.8, 1, 1.2, 1.5), c(1, 1.5, 0.7)),
                 diff = matrix(c(-1, 0, 1), 4, 3, byrow = TRUE))
irt_pars(tam.mml.2pl(dat, irtmodel = "GPCM", verbose = FALSE))
fit <- tam.mml.3pl(dat, E = tppcm(dat), est.variance = FALSE, verbose = FALSE)
irt_pars(fit)
irt_pars(fit, long = TRUE)       # long format with confidence limits
fit$se.gammaslope                # TAM's standard errors: too small
```

`item_test` *Item-level score tests*

Description

For each item, the score test for freeing all of its step discriminations relative to the model of `x` (e.g. whether the item deviates from the GPCM).

Usage

```
item_test(x, info = c("louis", "opg"), adjust = "holm")
```

Arguments

`x`	A fitted model (see `get_parts()`) or a `get_parts` object.
`info`	`"louis"` (observed information) or `"opg"`.
`adjust`	Multiple-testing adjustment of `p_adj` over all rows, a method of `stats::p.adjust()` (`"none"` for none).

Value

Data frame (class `tppcm_table`) with `item`, `statistic`, `df`, `p`, `p_adj`.

Examples

```
data(data.Students, package = "CDM")
sc <- data.Students[, c("sc1", "sc2", "sc3", "sc4")]
fit <- tam.mml.2pl(sc, irtmodel = "GPCM", verbose = FALSE)
item_test(fit)                          # which items deviate from the GPCM
```

---

m2 *Limited-information goodness of fit: M2, RMSEA2, SRMSR, CFI, TLI*

Description

Computes the limited-information statistic of Maydeu-Olivares and Joe (2006) for any fit that `get_parts()` accepts, including step designs and the product form, which other software cannot express. The derivatives of the margins are taken with respect to the saturated parameters and mapped to the fitted model with the Jacobian of `get_parts()`.

Usage

```
m2(x, type = c("auto", "M2", "C2", "M2*"), CI = 0.9)
```

Arguments

`x`	A fitted model (see `get_parts()`) or the object returned by `get_parts()`.
`type`	`"auto"` (`"M2"` with up to 2000 margins, otherwise `"C2"`), `"M2"`, `"C2"` or `"M2*"`. The computations use all univariate and bivariate margins, so the number of margins is limited to 5000 (about 30 items with 4 categories).
`CI`	Coverage of the RMSEA confidence interval.

Details

- `"M2"`: all univariate and bivariate margins (categories 1, ..., K-1).
- `"C2"`: univariate margins and bivariate cross-products $E(X_i X_j)$ (Cai & Monroe, 2014).
- `"M2*"`: means $E(X_i)$ and cross-products $E(X_i X_j)$.

The reduced statistics have few degrees of freedom; for the saturated TPPCM with few items they are not identified (`df <= 0`), so the full `"M2"` is preferred when it is affordable. CFI and TLI compare against the independence model (items independent, free category proportions). SRMSR is the root mean squared difference between observed and model-implied Pearson correlations of the item scores.

Missing responses. The observed margins use the available cases and their covariance is scaled by the numbers of persons observed on each pair of margins. This is valid only when responses are missing completely at random (MCAR), e.g. planned missingness by booklet. When non-response is related to ability or to the responses themselves (omitted items, not reached items), the statistic is invalid: in a simulation with missingness depending on the response to one item, M2 rejected the true model in every replication. Score it or analyze complete cases in that situation. N in RMSEA is the number of persons.

Under misfit "C2" can differ from mirt's C2 by 10-20% with the same log-likelihood: both use the same margins and derivatives, but this function uses the exact model-implied covariance matrix of the margins (Maydeu-Olivares & Joe, 2006), whereas mirt estimates parts of it from the sample. Under the true model the two agree.

Value

Object of class tppcm_m2: a one-row data frame with M2, df, p, RMSEA, RMSEA_lower, RMSEA_upper, SRMSR, CFI, TLI.

References

Maydeu-Olivares, A., & Joe, H. (2006). Limited information goodness-of-fit testing in multidimensional contingency tables. *Psychometrika, 71*, 713-732.

Cai, L., & Monroe, S. (2014). A new statistic for evaluating item response theory models for ordinal data. CRESST Report 839.

Examples

```
set.seed(1)
dat <- sim_tppcm(1000, disc = outer(c(0.8, 1, 1.2, 1.5, 1), c(1, 1.5, 0.7)),
                 diff = matrix(c(-1, 0, 1), 5, 3, byrow = TRUE))
m2(tam.mml.2pl(dat, irtmodel = "GPCM", verbose = FALSE))
m2(tam.mml.3pl(dat, E = tppcm(dat), est.variance = FALSE, verbose = FALSE))
```

---

mi — *Modification indices for single step discriminations*

Description

For each step discrimination that is restricted in x, the 1-df score test for freeing it (MI) and its expected parameter change (EPC). The EPC is the first-order (one Newton step) approximation to the change of the estimate when the step is freed; when the MI is large it can overshoot the actual change noticeably.

Usage

```
mi(x, pars = NULL, info = c("louis", "opg"), adjust = "holm")
```

Arguments

x	A fitted model (see get_parts()) or a get_parts object.
pars	Names of step discriminations (default: all, e.g. "I1_disc2").
info	"louis" (observed information) or "opg".
adjust	Multiple-testing adjustment of p_adj over all rows, a method of stats::p.adjust() ("none" for none).

Details

For an item with two steps, freeing one step is the same 1-df test as freeing the other (the two rows have equal MI and EPCs of opposite sign).

Value

Data frame (class `tppcm_table`) with `par`, `MI`, `df`, `p`, `p_adj`, `EPC` (expected change of the step discrimination), sorted by `MI`.

Examples

```
data(data.Students, package = "CDM")
sc <- data.Students[, c("sc1", "sc2", "sc3", "sc4")]
fit <- tam.mml.2pl(sc, irtmodel = "GPCM", verbose = FSALSE)
mi(fit)                                  # which steps deviate from the GPCM
mi(fit, pars = c("sc1_disc3", "sc2_disc3"))
```

---

score_test — *Score test against a less restricted TPPCM*

Description

Tests the restricted model of `x` against `against` without fitting the alternative. With extra directions D,

$$S = g_D'\big(D'ID - D'IJ(J'IJ)^{-1}J'ID\big)^{-1} g_D,$$

where g_D is the score in the directions D adjusted for the restricted parameters and $J'IJ$ includes the curvature term. Against the saturated model $S = g'I^{-1}g$.

Usage

```
score_test(x, against = ~item * step, D = NULL, info = c("louis", "opg"))
```

Arguments

`x`	A fitted model (see `get_parts()`) or a `get_parts` object.
`against`	Alternative model as a design formula (see `tppcm()`); must contain the model of `x`: `~ item * step` (default, saturated), `~ item + step` (Rank-1, $\alpha_i\gamma_l$), `~ item` (GPCM, α_i) or `~ step` (γ_l, shared by items). The models form a hierarchy: `~ 1` is contained in `~ item` and `~ step`, both are contained in `~ item + step`, and that in `~ item * step`. Thus the GPCM against `~ item + step` tests for a step effect, Rank-1 against `~ item * step` for an item x step interaction.
`D`	Optional matrix of extra directions in the saturated space (rows as in `x$parindex`); overrides `against`.
`info`	`"louis"` (observed information) or `"opg"`.

Value

Object of class `tppcm_test` with `statistic`, `df`, `p.value`, `epc`.

Examples

```
set.seed(1)
dat <- sim_tppcm(800, disc = outer(c(0.8, 1, 1.2, 1.5), c(1, 1.5, 0.7)),
                 diff = matrix(c(-1, 0, 1), 4, 3, byrow = TRUE))
fit <- tam.mml.2pl(dat, irtmodel = "GPCM", verbose = FALSE)
score_test(fit)                  # GPCM vs saturated TPPCM
score_test(fit, against = ~ item + step)   # GPCM vs Rank-1: step effect?
item_test(fit)
mi(fit)
```

---

sim_tppcm — *Simulate item responses from a TPPCM*

Description

Simulate item responses from a TPPCM

Usage

```
sim_tppcm(N, disc, diff, theta = stats::rnorm(N))
```

Arguments

`N`	Number of persons.
`disc`	Items x steps matrix of step discriminations a_{il}. Trailing `NA`s give items with fewer steps; a vector gives binary items.
`diff`	Items x steps matrix of step difficulties b_{il}; the kernel of step l is $a_{il}(\theta - b_{il})$. Same shape as `disc`.
`theta`	Person parameters (default standard normal).

Value

Data frame of responses coded `0, ..., K-1`, columns `I1, I2, ...`.

Examples

```
dat <- sim_tppcm(200, disc = matrix(1, 3, 2), diff = matrix(c(-0.5, 0.5), 3, 2, byrow = TRUE))
```

`step_info` *Step, item and test information*

Description

Fisher information of a TPPCM-family fit, split into the contributions of the steps. With the item score on the step scale $A_i(X) = \sum_{l \le X} a_{il}$, the item information is $I_i(\theta) = \mathrm{Var}(A_i(X) \mid \theta)$ and step l contributes

$$c_{il}(\theta) = a_{il} \sum_m a_{im} \{S_{\max(l,m)} - S_l S_m\}, \quad S_l = P(X_i \ge l \mid \theta),$$

so that $\sum_l c_{il} = I_i$; the contributions are non-negative when all $a_{il} \ge 0$. For the PCM (all $a = 1$) this is the usual item information, the variance of the item score.

Usage

```
step_info(x, theta = seq(-4, 4, by = 0.05))
```

Arguments

`x`	A fitted model (see `get_parts()`) or a `get_parts` object.
`theta`	Trait values, on the scale of the fit.

Value

Object of class `step_info` with `theta`, `step` (array theta x items x steps), `item` (theta x items) and `test`. `plot()` draws the test, item or step information.

Examples

```
set.seed(1)
dat <- sim_tppcm(1000, disc = outer(c(0.8, 1, 1.2, 1.5, 1), c(1, 1.5, 0.7)),
                 diff = matrix(c(-1, 0, 1), 5, 3, byrow = TRUE))
fit <- tam.mml.3pl(dat, E = tppcm(dat), est.variance = FALSE, verbose = FALSE)
inf <- step_info(fit)
inf
plot(inf)                # test and item information
plot(inf, item = "I4")   # step contributions of one item
```

tppcm *Step design for the TPPCM and its submodels*

Description

Builds the slope design array E for `TAM::tam.mml.3pl()` so that the slope parameters (gammaslope) are the step discriminations (CBDs) a_{il}, restricted as described by `design`. The category slope of category k is $\sum_{l \le k} a_{il}$.

Usage

```
tppcm(dat, design = ~item * step, index = NULL, K = NULL)
```

Arguments

`dat`	Data frame or matrix of item responses coded `0, 1, ..., K_i - 1`. Items may have different numbers of categories; every category from 0 to the item's maximum must be observed.
`design`	One-sided formula in the terms `1`, `item`, `step` and `item:step` describing the step discriminations (see the section above): `~ 1`, `~ item`, `~ step` or `~ item * step` (default). Ignored when `index` is given.
`index`	Optional integer matrix (items x steps, steps up to the largest number of steps) of parameter indices `1, ..., H`. Entries for steps an item does not have are ignored (use `NA`). Indices are renumbered to 1, ..., H in order. Overrides `design`.
`K`	Number of categories per item (one number for all items, or one per item). Defaults to the observed maximum + 1 of each item; give it when a subsample may lack the highest categories (e.g. in tree nodes).

Value

An array of dimension items x categories x 1 x H with attributes `"index"` (items x steps parameter numbers) and `"design"` (the design formula, `NULL` for a custom `index`).

The design formula (an item x step layout on the log scale)

`design` describes the step discriminations of the items as a two-way item x step layout, read on the log scale like a two-way ANOVA:

$$\log a_{il} = \mu + \alpha_i + \gamma_l + (\alpha\gamma)_{il}.$$

`design`	Model	a_{il}	Fitted with
`~ 1`	PCM (common slope)	a	`tam.mml.3pl(E = tppcm(dat, design = ~ 1))`
`~ item`	GPCM	α_i	`tppcm(dat, design = ~ item)`
`~ step`	step model	γ_l, common to all items	`tppcm(dat, design = ~ step)`
`~ item + step`	Rank-1 (product form)	$\alpha_i\gamma_l$	`xxirt_tppcm()` (not linear)

~ item * step	saturated TPPCM (default)	a_{il}	tppcm(dat)

~ item:step and ~ item + step + item:step are the same model as ~ item * step. The models form a hierarchy: ~ 1 is contained in ~ item and ~ step, both are contained in ~ item + step, which is contained in ~ item * step. Read as main effects and an interaction:

- the item x step interaction means that items differ in the *shape* of their step discriminations: the ratios $a_{il}/a_{il'}$ differ between items (saturated TPPCM);
- Rank-1 (main effects only): every item has the same shape γ_l, scaled to its own level α_i;
- GPCM (~ item): the shape is flat, every step of an item has the item's discrimination;
- ~ step: the same shape and the same level for all items.

Accordingly the score test of the GPCM against Rank-1 (score_test(gpcm_fit, against = ~ item + step)) asks whether there is a step effect, and the test of Rank-1 against the saturated model (score_test(rank1_fit)) whether there is an item x step interaction. Rank-1 is multiplicative, so it cannot be written as a linear design for TAM; tppcm() stops and points to xxirt_tppcm(). For the four linear models the log scale does not matter (the same equality restrictions on a_{il}), and tppcm() returns the design array. The log scale presumes $a_{il} > 0$: Yu (1991) restricted the step discriminations to be positive, TAM does not impose it (xxirt_tppcm() does, with lower0 = TRUE).

The same item x step language describes the locations in TAM's and ConQuest's facet formulas: formulaA = ~ item + step is the rating scale model (step parameters common to all items) and ~ item + item:step the partial credit model. design = ~ step is the rating-scale idea applied to the discriminations.

Custom restrictions

Any other linear restriction on the step discriminations is written into index: steps with the same index share one parameter. index takes precedence over design. To fix a step, use the gammaslope.fixed argument of TAM::tam.mml.3pl() with its index. Do not use gammaslope.constr.V for equality constraints (it does not reach the maximum likelihood estimate) and do not use userfct.gammaslope to impose $a_{il} \geq 0$ (truncation is not the constrained estimate); use xxirt_tppcm() for bounds.

Convergence of tam.mml.3pl() on real rating scales

The saturated design (~ item * step) is cumulative: a_{il} enters the slopes of categories $l, \ldots, K_i - 1$, so the steps of an item are strongly correlated in the likelihood. TAM updates every slope with its own gradient and second derivative (no cross-derivatives), which overshoots when this correlation is strong: with 5 or more steps per item and sparse low categories (e.g. Likert items with few responses in the lowest categories) the deviance rises from iteration to iteration, the slopes drift to the bound gammaslope.max (9.99) and maxiter is exhausted. Such a fit is not an estimate; do not report it.

The remedy is to damp the M-step: max.increment = 0.1 caps the change of every parameter per iteration (TAM's default is 1), and the fit then needs more iterations (hundreds to a few thousand, a few seconds). For the saturated design on real data, use this call as the standard one:

```
tam.mml.3pl(dat, E = tppcm(dat), est.variance = FALSE,
            control = list(maxiter = 3000, max.increment = 0.1))
```

Check the fit against TAM's nominal response model, which is the saturated TPPCM in another parameterization and converges without tuning: the log-likelihood of `tam.mml.2pl(dat, irtmodel = "2PL")` must agree. If the NRM log-likelihood is higher, the damped fit sits in a local optimum. Then either use the NRM fit itself (all functions of the package accept it) or, as a suggestion, start the damped fit from the NRM solution, e.g. `gammaslope = as.vector(t(get_parts(m_nrm)$a))` and `xsi.inits = cbind(seq_along(m_nrm$xsi$xsi), m_nrm$xsi$xsi)` when all items have the same number of categories (check the result against the NRM log-likelihood). Quasi-Monte Carlo integration (`QMC = TRUE`) does not help: the problem is the step length of the M-step, not the quadrature. Restricted designs (`~ item`, `~ step`, `~ 1`, an `index`) have fewer and less correlated slopes and converge with the defaults.

References

Yu, M.-N. (1991). *A two-parameter partial credit model* (Doctoral dissertation). University of Illinois at Urbana-Champaign.

See Also

`irt_pars()` for the estimates with correct standard errors; `xxirt_tppcm()` for Rank-1; tppcm-package for an overview and `vignette("tppcm-tutorial")`.

Examples

```
set.seed(1)
dat <- sim_tppcm(500, disc = matrix(c(1, 1.5, 0.7), 4, 3, byrow = TRUE),
                      diff = matrix(c(-1, 0, 1), 4, 3, byrow = TRUE))
fit <- function(E) TAM::tam.mml.3pl(dat, E = E, est.variance = FALSE, verbose = FALSE)
m_sat  <- fit(tppcm(dat))                       # saturated: ~ item * step
m_gpcm <- fit(tppcm(dat, design = ~ item))      # GPCM
m_step <- fit(tppcm(dat, design = ~ step))      # step model
m_pcm  <- fit(tppcm(dat, design = ~ 1))         # PCM (common slope)
score_test(m_gpcm, against = ~ item + step)     # step effect?
# items 1-2 saturated, items 3-4 GPCM
idx <- rbind(1:3, 4:6, rep(7, 3), rep(8, 3))
m_mix <- fit(tppcm(dat, index = idx))
```

`tppcmtree` *TPPCM trees: model-based recursive partitioning of step discriminations*

Description

Fits a TPPCM-family model with `TAM::tam.mml.3pl()` in every node and splits the sample along covariates where the item parameters are unstable (Zeileis, Hothorn & Hornik, 2008), using `partykit::mob()`. With `parm = "disc"` (or `"slope"`) only instability of the slopes drives the splits; with `parm = "diff"` (or `"int"`) only instability of the step difficulties (intercepts); with `c("disc", "diff")` (default) both.

Usage

```
tppcmtree(
  formula,
  data,
  design = ~item * step,
  parm = c("disc", "diff"),
  param = c("irt", "si"),
  orthogonal = TRUE,
  impact = NULL,
  ...
)
```

Arguments

`formula`	`resp ~ z1 + z2 + ...`, where `resp` is a numeric matrix of item responses stored as one column of `data` (`data$resp <- as.matrix(items)`) and `z` are partitioning covariates.
`data`	Data frame.
`design`	Node model, a linear design formula of `tppcm()`: `~ item * step` (default), `~ item`, `~ step` or `~ 1`.
`parm`	Parameters used in the instability tests: `c("disc", "diff")` (default, both blocks), the slope block (`"disc"` or `"slope"`) or the location block (`"diff"` or `"int"`). (`"all"` is accepted for both.)
`param`	Parameterization of the scores: `"irt"` (`disc`, `diff`) or `"si"` (`slope`, `int`); see `get_parts(fit, "estfun")`.
`orthogonal`	For a slope (location) block test: replace the tested scores by efficient scores, i.e. residuals from their regression on the other block ($s_a - s_b \hat{I}_{bb}^{-1} \hat{I}_{ba}$ with the outer-product information). Without this, `mob()` selects components of the jointly decorrelated process and instability in one block leaks into the test of the other (e.g. location DIF triggers slope splits).
`impact`	`NULL`, or the name of a factor in `data` (or a factor of length `nrow(data)`) whose latent means and variances are estimated in every node.
`...`	Passed to `partykit::mob_control()` (e.g. `alpha`, `minsize`, `maxdepth`, `bonferroni`).

Details

Node models are the linear designs of `tppcm()`: `~ item * step` (step discriminations), `~ item` (item discriminations, the GPCM; compare `psychotree::gpcmtree()`), `~ step` (one discrimination per

step, common to the items) and ~ 1 (one common slope; compare psychotree::pctree()). With the trait fixed at $N(0,1)$, the common slope of the ~ 1 node plays the role of the latent standard deviation, so for ~ 1 use parm = "diff" to look for DIF rather than for differences in trait variance.

Impact is not DIF. The latent trait is $N(0,1)$ in every node, so a covariate related to ability (groups that differ in their mean) is split on even without DIF: the difficulties of all items shift together. impact names a grouping variable whose latent means and variances are estimated in every node (tam.mml.3pl(group =), as impact in psychotools::gpcmodel()); splits on that variable then reflect item differences only. Conditional-maximum-likelihood trees such as psychotree::pctree() do not have this problem for the PCM.

Rows with missing item responses are kept (TAM uses the observed responses); the partitioning covariates must be complete. The node model needs every category of every item in every node; otherwise the fit stops with a message. Increase minsize or use a less demanding node model.

Value

A modelparty object (class tppcmtree). coef() gives the node-wise parameters; print() shows the splits and the number of persons per node.

Examples

```
data("VerbalAggression", package = "psychotools")
tr <- tppcmtree(resp ~ gender + anger, data = VerbalAggression,
                design = ~ 1, parm = "diff", minsize = 50)
tr
head(t(coef(tr)))
```

---

wald_test *Wald test of equal step discriminations within items*

Description

For each item, tests $a_{i1} = \cdots = a_{iL}$ (the item follows the GPCM) with CDM::WaldTest(). Items whose steps are already constrained equal in x are skipped.

Usage

```
wald_test(x, adjust = "holm")
```

Arguments

x	A fitted model (see get_parts()) or a get_parts object in which the tested steps are free.
adjust	Multiple-testing adjustment of p_adj over all rows, a method of stats::p.adjust() ("none" for none).

Value

Data frame (class `tppcm_table`) with `item`, `statistic`, `df`, `p`, `p_adj`.

Examples

```
data(data.Students, package = "CDM")
sc <- data.Students[, c("sc1", "sc2", "sc3", "sc4")]
fit <- tam.mml.3pl(sc, E = tppcm(sc), est.variance = FALSE, verbose = FALSE)
wald_test(fit)                          # a_i1 = a_i2 = a_i3 for each item?
```

---

xxirt_tppcm — *Fit TPPCM-family models with sirt::xxirt()*

Description

Use this for what TAM cannot estimate exactly: Rank-1, the multiplicative product form (`design = ~ item + step`: $a_{il} = \alpha_i \gamma_l$, $\gamma_1 = 1$, γ shared across items; see `tppcm()` for the item x step view of the designs) and the bound $a_{il} \geq 0$. The saturated model (`~ item * step`) and the GPCM (`~ item`) are also available, e.g. for simulations; fit `~ 1` and `~ step` with `TAM::tam.mml.3pl()` and `tppcm()`.

Usage

```
xxirt_tppcm(
  dat,
  design = ~item * step,
  Theta = matrix(seq(-6, 6, length.out = 21), ncol = 1),
  lower0 = TRUE,
  ...
)
```

Arguments

`dat`	Item responses coded `0, ..., K-1`, same `K` for all items.
`design`	`~ item * step` (default, saturated), `~ item + step` (Rank-1) or `~ item` (GPCM); see `tppcm()`.
`Theta`	Quadrature nodes (matrix with one column).
`lower0`	Impose $a_{il} \geq 0$ (and $\alpha, \gamma \geq 0$).
`...`	Passed to `sirt::xxirt()` (e.g. `conv`, `maxit`). Item parameters are stored in slope-intercept form: step `l` has kernel `a_l * theta + int_l` (`alpha * g_l * theta + int_l` for the product form). Use `irt_pars()` for discriminations and difficulties.

Details

The latent trait is $N(0, 1)$. The default nodes are those of TAM (`seq(-6, 6, len = 21)`), so log-likelihoods can be compared with TAM fits of the same data (e.g. with `CDM::IRT.compareModels()`; use its `LRtest` part, not `anova()`).

Value

An `xxirt` object (with the additional class `xxirt_tppcm`, whose `print()` method shows a short summary instead of the whole list). Use `xxirt_vcov()` for its covariance matrix and `irt_pars()`, `score_test()` and the other tools of this package.

Examples

```
set.seed(1)
dat <- sim_tppcm(500, disc = outer(c(0.8, 1, 1.2, 1.5), c(1, 1.5, 0.7)),
                      diff = matrix(c(-1, 0, 1), 4, 3, byrow = TRUE))
fit <- xxirt_tppcm(dat, design = ~ item + step)       # Rank-1
fit
irt_pars(fit, restricted = TRUE)                       # alpha_i, gamma_l
score_test(fit)                                        # item x step interaction?
```

---

xxirt_vcov — *Covariance matrix of an xxirt fit with shared or fixed parameters*

Description

`vcov()` for `xxirt` objects (sirt 4.2-133) fails with "the condition has length > 1" when parameters are shared across rows of the parameter table or fixed. This recomputes the numerical Hessian without the shortcut that causes the error.

Usage

```
xxirt_vcov(x)
```

Arguments

`x`	An `xxirt` object.

Value

Inverse of the negative Hessian of the log-likelihood.

Examples

```
data(data.Students, package = "CDM")
sc <- as.matrix(data.Students[, c("sc1", "sc2", "sc3", "sc4")])
fit <- xxirt_tppcm(sc, design = ~ item + step)
fit
round(sqrt(diag(xxirt_vcov(fit))), 3)
```

Index

dif_test, 3
dif_test(), *7*, *11*
dsf, 6
dsf(), *4*, *5*, *9*
dsf_design, 8
dsf_design(), *5–7*

get_parts, 10
get_parts(), *3*, *12–16*, *18*, *23*

irt_pars, 12
irt_pars(), *21*, *24*, *25*
item_test, 13

m2, 14
m2(), *11–13*
mi, 15

partykit::mob(), *22*
partykit::mob_control(), *22*
psychotools::gpcmodel(), *23*
psychotree::gpcmtree(), *22*
psychotree::pctree(), *23*

score_test, 16
score_test(), *25*
sim_tppcm, 17
sirt::xxirt(), *24*
stats::p.adjust(), *4*, *6*, *13*, *15*, *23*
step_info, 18

TAM::tam.mml(), *10*
TAM::tam.mml.2pl(), *10*
TAM::tam.mml.3pl(), *6*, *8–10*, *19*, *20*, *22*, *24*
tppcm, 19
tppcm(), *6–8*, *10*, *11*, *16*, *22*, *24*
tppcm-package, *3*, *7*, *8*, *12*, *21*
tppcmtree, 21
tppcmtree(), *5*, *11*

wald_test, 23

xxirt_tppcm, 24
xxirt_tppcm(), *10*, *19–21*
xxirt_vcov, 25
xxirt_vcov(), *25*