Package 'rasch'

July 30, 2026

Type Package

Title Pairwise Conditional Rasch Measurement Analysis and Diagnostics

Version 1.11.7

Description Pairwise conditional maximum likelihood estimation of dichotomous and polytomous Rasch models (partial credit and rating scale) after Andrich and Luo (2003) and Zwinderman (1995) <doi:10.1177/014662169501900406>, with standard errors from a Godambe sandwich estimator. An optional alternative estimator reparameterises each item's thresholds as Andrich's (1978 <doi:10.1007/BF02293814>, 1985) orthogonal-polynomial principal components (location, spread, skewness, and kurtosis; Pedler 1987), exact for items with up to 3 thresholds and a smoothed reduced-rank model for items with more, useful when some categories are sparsely populated. Person measures are Warm's (1989) <doi:10.1007/BF02294627> weighted likelihood estimates, computed per missing-data pattern. The diagnostic suite follows the conventions set out in Andrich and Marais (2019) <doi:10.1007/978-981-13-7496-8>: the log-of-mean-square fit residual with apportioned degrees of freedom (and its natural form), infit and outfit, the item-trait interaction chi-square over automatically sized class intervals with its per-interval detail table, the class-interval ANOVA item-fit F, the person separation index with and without extremes and the item separation index, Cronbach's alpha, summary distribution statistics with skewness and kurtosis, targeting, the score-to-measure table with maximum likelihood and geometric extreme-score extrapolation options, test information, threshold and category diagnostics, residual principal-components dimensionality testing, local dependence by residual correlation, and differential item functioning by two-way residual analysis of variance over any number of person factors, factor-at-a-time (the full two-way table with partial eta-squared effect sizes) or as a full factorial with interaction precedence, Tukey HSD post-hoc comparisons on significant group terms and interaction cells, false-discovery-rate or familywise adjustment, and DIF magnitudes in logits by resolved-item locations with a

practical-significance criterion. Violations of
independence are quantified, not just flagged: the magnitude of
response dependence between two items by the resolution method of
Andrich and Kreiner (2010) <doi:10.1177/0146621609360202> (polytomous
form Andrich, Humphry and Marais 2012 <doi:10.1177/0146621612441858>), the spread-
parameter least-upper-bound screen (Andrich 1985),
and the magnitude of multidimensionality (latent subscale correlation
and common-variance proportion) from Andrich's (2016) two-calculation
reliability comparison. A likelihood-ratio test of the partial credit
against the rating parameterisation is reported both raw, as
conventionally displayed, and with a first-order composite-likelihood calibration
(Kent 1982 <doi:10.1093/biomet/69.1.19>) from the Godambe matrices. Also
included: anchored estimation for test equating (individual threshold
and average item-location anchors), common-item equating tests and
plots, item splitting to resolve invariance violations, tailored
analysis for guessing with the four-step anchored comparison (Andrich,
Marais and Humphry 2012 <doi:10.3102/1076998611411914>), classical test theory compan-
ion statistics,
racked and stacked reshaping for repeated measurements, model comparison
by composite-likelihood information criteria whose penalty is the
Godambe effective parameter count (Varin and Vidoni 2005
<doi:10.1093/biomet/92.3.519>; Gao and Song 2010
<doi:10.1198/jasa.2010.tm09414>), absorbing the pairwise over-
counting that a nominal AIC or BIC
would ignore, the many-facet
Rasch model (Linacre 1989) for rated long-format data with facet
severities, fit, and optional item-by-facet interactions, subtest
formation for locally dependent items, multiple-choice scoring against
a key with double keying and polytomous option scoring of informative
distractors (Andrich and Styles 2011, with an evidence-based rescoring
proposal), rest-measure distractor analysis and option curves, the Guttman
scalogram with the coefficient of reproducibility, the
Bradley-Terry-Luce model for paired comparisons (Bradley and Terry
1952 <doi:10.1093/biomet/39.3-
4.324>; Luce 1959) as the conditional form of the dichotomous Rasch model
(Andrich 1978), estimated by the same conventions with judge-clustered
sandwich errors and judge fit diagnostics, and the first software implementation of
the extended frame of reference model (Humphry 2005; Humphry and Andrich
2008), in which the unit of the latent scale differs across item-set by
person-group frames: group units are estimated by person-free
within-frame pairwise conditioning and set units by error-corrected
person linking, all reported in a common arbitrary unit; its
paired-comparison form estimates judge-panel and object-set units
with the linking identified from cross-set comparisons alone. A modern
'shiny' interface and a one-call exporter for every table and plot are
included.
Implemented from published measurement theory in base R, with no
dependence on other estimation engines.

URL https://drjoshmcgrane.github.io/rasch/,
https://github.com/drjoshmcgrane/rasch

BugReports https://github.com/drjoshmcgrane/rasch/issues

License MIT + file LICENSE

Encoding UTF-8

Imports stats, graphics, grDevices, utils

Suggests testthat (>= 3.0.0), shiny, bslib, DT, bsicons, knitr, rmarkdown, eRm, sirt, psychotools

VignetteBuilder knitr

RoxygenNote 7.3.3

Config/testthat/edition 3

NeedsCompilation no

Author Josh McGrane [aut, cre]

Maintainer Josh McGrane <drjoshmcgrane@gmail.com>

Repository CRAN

Date/Publication 2026-07-30 12:40:03 UTC

Contents

`btl` *Fit the Bradley-Terry-Luce model to paired comparisons*

Description

Estimates object locations from paired-comparison data by conditional maximum likelihood. The Bradley-Terry-Luce model is the conditional form of the dichotomous Rasch model – within an item pair, given one correct response, the Rasch probability that it was the easier item is exactly of BTL form – so it belongs to the same measurement family and is estimated by the same conventions as the rest of the package: Newton-Raphson on the person-free likelihood, locations identified by the sum-zero constraint, and Godambe sandwich standard errors, clustered by judge when a judge column is given (so repeated comparisons by the same judge need not be independent). Objects that win or lose every comparison have no finite estimate and are removed with a note, exactly as extreme persons are set aside in a Rasch calibration; the comparison graph must remain connected.

Usage

```
btl(
  data,
  object_a,
  object_b,
  winner = NULL,
  response = NULL,
  margin = NULL,
```

```
  judge = NULL,
  count = NULL,
  order = NULL,
  position = FALSE,
  anchors = NULL,
  ties = c("drop", "half", "error"),
  thresholds = c("free", "pc"),
  maxit = 60,
  tol = 1e-08
)
```

Arguments

`data`	A data frame with one comparison per row.
`object_a`, `object_b`	Names of the columns holding the two objects compared.
`winner`	Name of the column holding the winner of each row: its value must equal the row's `object_a` or `object_b` entry; `"tie"` or `"draw"` marks a tie; anything else (including blanks) is treated as missing and dropped with a note. Ignored when `response` is given.
`response`	Optional name of a column holding a graded preference for `object_a` over `object_b` – an ORDERED factor (`factor(..., ordered = TRUE)`, levels worst to best for `object_a`) or integer scores `0..m`; a plain factor is refused, since its alphabetical level order would silently define (and can reverse) the response scale. Fits the adjacent-categories ordinal extension of BTL (Tutz 1986; Agresti 1992): a partial-credit structure on the difference of locations with thresholds constrained symmetric, `tau_k = -tau_(m+1-k)`, so the model is invariant to presentation order. Two categories reproduce BTL exactly; three give the Davidson (1970) ties model.
`margin`	Optional name of a column holding the extent of the win ("a little", "much", ...), as an ordered factor or increasing values; combined with `winner` it assembles the graded response without any orientation bookkeeping ("B by much" means the same thing whichever column B sits in). Winner values matching neither object are ties and form the middle category.
`judge`	Optional name of a judge column; enables the judge fit table and clusters the sandwich standard errors by judge.
`count`	Optional name of a column of replication counts (a row standing for several identical comparisons).
`order`	Optional name of a column giving each judge's judgment sequence (timestamps or ranks; requires `judge`). Adds the within-judge dependence analysis: an exposure effect (the advantage, in logits, of an object the judge has seen before over one they have not) and a carry-over effect (the pull of the judge's own earlier verdicts on the same object – response dependence in the sense of Marais and Andrich 2008), estimated jointly with the locations and reported in `dependence`. Incompatible with `ties = "half"`.

`position` Logical: when `TRUE`, `object_a` is taken as the first-presented (left) object of every comparison and a first-position advantage is estimated – a single coefficient, in logits, added to every comparison's location difference, the pure positional form of the Davidson and Beaver (1977) within-pair order-effect device. It is reported in `dependence` with `effect = "position"` (every comparison is informative, so `n_informative` is the total weighted comparison count) and estimated jointly with the locations, alongside the exposure and carry-over effects when `order` is also given. Identification comes from triangle closure (K >= 3), so the constant oriented covariate is estimable even when each pair has a fixed orientation, though weakly. Note that `ties = "half"` duplicates rows in the same orientation, so the first position stays well defined.

`anchors` Optional named numeric vector for equating: names are object names, values are fixed locations in logits. The named objects are held exactly at those locations and the remaining objects are estimated freely with no sum-zero constraint – the origin and scale come from the anchors, exactly as an anchored `rasch` calibration works. Anchored objects report a standard error of zero (their location is a constant, not an estimate). An anchored object that is undefeated or winless is an error, not silently removed as a free boundary object would be.

`ties` How to treat ties in the dichotomous analysis: `"drop"` (default, removed with a note), `"half"` (half a win each way, a common pragmatic device – flagged in the notes because the halves are not independent Bernoulli trials), or `"error"`. With graded responses, code ties as a middle category instead.

`thresholds` `"free"` (default) estimates every symmetric threshold parameter; `"pc"` pools them to the spread (linear) principal component – the symmetric case of the principal-component threshold structure, whose even skewness component is structurally zero here – so thinly used categories borrow strength from every response. Both modes report the component decomposition.

`maxit`, `tol` Newton-Raphson iteration cap and convergence tolerance.

Details

Fit is reported at three levels, mirroring the Rasch diagnostics. Per object and (when given) per judge: the log-of-mean-square fit residual of Andrich and Marais (2019, ch. 23) over their comparisons, with apportioned degrees of freedom – an erratic judge or an object of inconsistent quality shows exactly as an erratic person or misfitting item does. Per pair: the classical goodness-of-fit table comparing observed and expected win proportions, with the total chi-square on (pairs used) minus (free location parameters) degrees of freedom. The object separation index is the analogue of the PSI: the proportion of observed location variance not due to error. Anchored objects enter it with their fixed locations and zero error – they are separated with certainty by construction.

Value

A list of class `"rasch_btl"`: `objects` (location, se, comparisons, wins – or the graded `score` – infit and outfit mean squares, fit residual and its df), `pairs` (per pair: n, observed and expected win proportions – or mean graded responses – standardised residual, chi-square component – the pair chi-squares treat comparisons as independent and are descriptive under judge clustering; the object and judge fit residuals and the clustered standard errors carry the robust inference), `judges`

(when given: per judge n, infit, outfit, fit residual, df), `total_chisq`, `total_df`, `total_p`, the object separation index `osi`, `loglik`, `cl` (the composite-likelihood information ingredients used by `compare_fits`: the Godambe effective parameter count and the independent-unit count), convergence details, and `notes`. Graded fits add `thresholds` (the symmetric threshold estimates with standard errors), `m`, and `categories`. With an `order` column the within-judge `dependence` effects table carries an `n_informative` count, and `dependence_data` holds every comparison with its per-comparison exposure and carry-over covariates (see `plot_btl_dependence`).

References

Bradley, R. A. and Terry, M. E. (1952). Rank analysis of incomplete block designs: I. The method of paired comparisons. Biometrika, 39, 324-345. Luce, R. D. (1959). Individual Choice Behavior. Wiley. Andrich, D. (1978). Relationships between the Thurstone and Rasch approaches to item scaling. Applied Psychological Measurement, 2, 451-462.

Tutz, G. (1986). Bradley-Terry-Luce models with an ordered response. Journal of Mathematical Psychology, 30(3), 306-316. Agresti, A. (1992). Analysis of ordinal paired comparison data. Journal of the Royal Statistical Society C, 41(2), 287-297. Davidson, R. R. (1970). On extending the Bradley-Terry model to accommodate ties in paired comparison experiments. Journal of the American Statistical Association, 65(329), 317-328.

Davidson, R. R., & Beaver, R. J. (1977). On extending the Bradley-Terry model to incorporate within-pair order effects. Biometrics, 33(4), 693-702.

Examples

```
set.seed(1)
beta <- c(A = -1, B = -0.3, C = 0.4, D = 0.9)
pairs <- t(combn(names(beta), 2))
d <- data.frame(a = rep(pairs[, 1], each = 30),
                b = rep(pairs[, 2], each = 30))
p <- plogis(beta[d$a] - beta[d$b])
d$win <- ifelse(runif(nrow(d)) < p, d$a, d$b)
btl(d, object_a = "a", object_b = "b", winner = "win")
```

---

btl_dif *DIF analysis for paired comparisons*

Description

Tests whether objects function differently for identifiable groups of judges. One judge factor is analysed on its own; several factors are modelled jointly – with main effects by default and factor-by-factor interactions optional – exactly as `dif_anova` treats person factors. For each object the standardised residuals of its comparisons, oriented to the object, are analysed by the judge factor(s) crossed with opponent-strength bands: a term is uniform DIF, its crossing with the band non-uniform DIF, and a significant higher-order group term supersedes the lower-order group terms built from a subset of its factors. Each term flagged for uniform DIF and not superseded is then resolved – the object split into one copy per cell of the term's factors inside a joint refit – and the differences between the resolved locations reported in logits with judge-clustered Wald tests and the

practical-significance flag, mirroring dif_size. Fits with within-judge dependence effects (order) keep those effects in the residual moments and in the refits, so dependence is not mistaken for judge-group DIF; count-weighted comparisons enter all tests with their weights.

Usage

```
btl_dif(
  fit,
  factors,
  objects = NULL,
  effects = c("main", "factorial"),
  p_adjust = "BH",
  alpha = 0.05,
  flag_logits = 0.5,
  min_n = 20,
  maxit = 60,
  tol = 1e-08
)
```

Arguments

fit	An object from btl.
factors	A judge factor, or a named list of them, each either one value per row of fit$comparisons or a vector named by judge.
objects	Objects to test; all by default.
effects	"main" (default) models several factors additively (each factor's main effect and its band interaction); "factorial" also crosses the factors with one another.
p_adjust	Multiplicity adjustment across objects within each term; the resolved-size probabilities are adjusted in one pool over all objects, terms, and cell pairs.
alpha	Significance level for adjusted probabilities.
flag_logits	Absolute resolved difference flagged as practically significant.
min_n	Term cells with fewer comparisons involving the object are dropped from its resolution, with a note.
maxit, tol	Newton controls for the resolution refits.

Details

The screening ANOVA treats JUDGES as the independent units: residuals are aggregated to one weighted mean per judge (per opponent band) and tested in a split-plot design with the judge as the error unit – group terms between judges, band terms and their interactions within. Testing judge-level factors against comparison-level residuals would pseudo-replicate (a null simulation with judge heterogeneity and arbitrary groups falsely flagged uniform DIF in 6 of 10 datasets); the judge-level design is calibrated, and its power grows with the number of judges per group, not the number of comparisons. Each factor level needs at least two judges, and an object at least four judges overall, to be testable.

Each object is resolved against the other objects' common locations. When several objects carry real DIF, resolving them one at a time can spread a large effect onto clean objects as compensating,

opposite-signed artificial DIF (Andrich & Hagquist 2012, 2015); read large flags on several objects together with that hazard in mind, and prefer resolving the largest effect first and re-running.

Value

A list of class `"rasch_btl_dif"`: `summary` (one row per object and group term with the uniform F, adjusted p and partial eta-squared – the term itself – the non-uniform ones – the term crossed with the opponent band – plus `uniform_DIF`, `nonuniform_DIF` and `superseded` flags); `terms` (the full per-object analysis-of-variance table); `levels` (resolved location and SE per object, term and cell); `sizes` (per object, term and cell pair: difference in logits, SE, z, adjusted p, significance and practical flags); `effects`, `factors`, and `notes`.

References

Andrich, D., & Hagquist, C. (2012). Real and artificial differential item functioning. *Journal of Educational and Behavioral Statistics*, 37(3), 387-416.

Dittrich, R., Hatzinger, R., & Katzenbeisser, W. (1998). Modelling the effect of subject-specific covariates in paired comparison studies with an application to university rankings. *Journal of the Royal Statistical Society C*, 47(4), 511-525.

Examples

```
set.seed(1)
beta <- c(A = -1, B = -0.3, C = 0.4, D = 0.9)
pr <- t(combn(names(beta), 2))
d <- data.frame(a = rep(pr[, 1], each = 60), b = rep(pr[, 2], each = 60),
                judge = sample(sprintf("J%02d", 1:12), 360, TRUE))
shift <- ifelse(d$judge %in% sprintf("J%02d", 1:6) & d$a == "C", 0.9,
         ifelse(d$judge %in% sprintf("J%02d", 1:6) & d$b == "C", -0.9, 0))
p <- plogis(beta[d$a] - beta[d$b] + shift)
d$win <- ifelse(runif(nrow(d)) < p, d$a, d$b)
f <- btl(d, "a", "b", winner = "win", judge = "judge")
grp <- setNames(rep(c("g1", "g2"), each = 6), sprintf("J%02d", 1:12))
btl_dif(f, grp, objects = "C")
```

`btl_dimensionality` *Residual dimensionality of paired comparisons*

Description

The residual-PCA analogue for paired comparisons. The fitted model predicts how often each object should beat each other from their locations; the object-by-object matrix of departures from that prediction (on the log-odds scale) is *skew-symmetric*, so its structure decomposes into rotational planes – Gower's (1977) bimensions – rather than the ordinary components of a symmetric residual-correlation matrix. A dominant leading bimension is a coherent "swirl" in the residuals (A over-beats B, B over-beats C, C over-beats A): a second attribute steering some contests. A flat spectrum is noise: the single scale suffices. The leading bimension is judged against a reference built by simulating unidimensional data from the fitted model with the observed pair counts (a parametric bootstrap, as in `plot_scree`); an observed strength above the reference is structure the one-dimensional

model does not explain. For graded fits the residual log-odds are taken on the points-proportion scale, whose model mean is not exactly `plogis(beta_i - beta_j)`; the simulated reference carries the same construction, so the test stays calibrated (verified mildly conservative on model-true graded data) rather than anticonservative. Likewise, when the fit carries within-judge dependence effects (`order`), the reference is simulated sequentially through each judge's comparisons WITH the fitted exposure and carry-over coefficients: order effects push the marginal pair rates around in a structured way, and a reference without them would read that structure as a second attribute. The price is power: carry-over and a judge-camp second attribute are partially confounded (both appear as consistent within-judge deviation), so with `order` modelled the test is conservative about attributing the ambiguous share to a second dimension. The reference simulates from the point estimates without refitting each replicate, so it carries sampling noise in the responses but not estimation noise in the parameters – adequate for the screening use here, slightly liberal in tiny designs.

Usage

```
btl_dimensionality(fit, reps = 50L)
```

Arguments

`fit`	A paired-comparison fit from `btl`.
`reps`	Model-simulated replicates for the noise reference.

Value

A list of class `"rasch_btl_dim"`: `bimensions` (per bimension: strength and share of residual size; the reference mean, 95th percentile, and the clears-the-reference flag are reported for the leading bimension and `NA` for the rest); `coords` (each object's position in the leading bimension plane, for the residual map); `leading_structured` (whether bimension 1 clears its reference); `residual_matrix`; and `notes`.

References

Gower, J. C. (1977). The analysis of asymmetry and orthogonality. In J. R. Barra et al. (Eds.), *Recent Developments in Statistics* (pp. 109-123). North-Holland.

Examples

```
set.seed(1); objs <- LETTERS[1:6]; beta <- setNames(seq(-1.5, 1.5, len = 6), objs)
pr <- t(utils::combn(objs, 2))
d <- data.frame(a = rep(pr[, 1], each = 30), b = rep(pr[, 2], each = 30))
d$win <- ifelse(runif(nrow(d)) < plogis(beta[d$a] - beta[d$b]), d$a, d$b)
btl_dimensionality(btl(d, "a", "b", "win"), reps = 20)
```

`btl_efrm` *Fit the extended frame of reference model for paired comparisons*

Description

Estimates object locations from paired comparisons when the unit of the latent scale differs across frames – judge-panel by object-set cells – an extension of Humphry's (2005) extended frame of reference model to the Bradley-Terry-Luce family. Objects are partitioned into sets and judges into panels. Each object has a common-scale value `v_k = alpha_s beta_k + kappa_s`, with `beta_k` the within-set calibration location, `alpha_s > 0` the set unit and `kappa_s` the set origin; a comparison judged in panel g carries the panel unit `phi_g`. Within a set the comparison logit is `phi_g (beta_a - beta_b)` (the set origin cancels and the set unit is confounded with the spread of `beta`, so neither is identified within a set); across sets it is `phi_g (v_a - v_b)`, which places the sets on one common scale.

Usage

```
btl_efrm(
  data,
  object_a,
  object_b,
  winner,
  judge,
  panels,
  object_sets,
  response = NULL,
  ties = c("drop", "error"),
  min_link = 20,
  se_method = c("bootstrap", "judge_bootstrap", "conditional"),
  boot_reps = 60,
  maxit = 60,
  tol = 1e-08
)
```

Arguments

`data`	A data frame with one comparison per row.
`object_a, object_b`	Names of the columns holding the two compared objects.
`winner`	Name of the column holding the winner; its value must equal the row's `object_a` or `object_b` entry, `"tie"` or `"draw"` marks a tie, and anything else is treated as missing.
`judge`	Name of the judge column (clusters the stage-one standard errors and defines the panels when `panels` is a judge attribute).
`panels`	Either the name of a judge-attribute column in `data` or a named vector mapping judge to panel.

object_sets	A named list mapping set names to character vectors of object names; every compared object must belong to exactly one set.
response	Not supported: this first implementation fits dichotomous winner data only. Supplying it raises an informative error.
ties	"drop" (default, removed with a note) or "error".
min_link	Minimum number of cross-set comparisons a set pair must supply to be used for linking; sets not reachable from the reference set through sufficient cross-set pairs raise an error.
se_method	"bootstrap" (the default): a parametric bootstrap; "judge_bootstrap": judges resampled with replacement within panels, carrying within-judge dependence; of the ENTIRE two-stage pipeline – winners resampled from the fitted probabilities, both stages refitted boot_reps times – so the reported standard errors carry every source of sampling variability, including the stage-one uncertainty that flows into the linking. "conditional": the fast analytic errors, exact for beta and phi but conditional on stage one for alpha and kappa, which they therefore UNDERSTATE (verified by simulation); for quick inspection only.
boot_reps	Bootstrap replicates for se_method = "bootstrap".
maxit, tol	Newton iteration cap and convergence tolerance.

Details

One modelling convention deserves plain statement. Rewritten with a single latent value per object, the model says within-set-s contests are judged at discrimination phi_g / alpha_s while every cross-set contest is judged at exactly phi_g: the common scale is *defined* as the scale of between-frame judgement. That is an assumption about how discriminal dispersion behaves when unlike objects meet (a Thurstonian question with no assumption-free answer), not a consequence of the theory; the per-frame fit residuals in frames are where a violation would show.

Estimation follows rasch_efrm in two conditional stages. In stage one the within-set comparisons of each set, pooled over panels, fit the bilinear model logit = rho_{gs} (b_a - b_b) with b sum-zero and the most-used panel fixed at rho = 1; the ratios rho_{gs} = phi_g / phi_{ref(s)} estimate the panel units up to each set's reference panel and are reconciled across sets by a precision-weighted least squares over the panel-by-set linking graph, then normalised to geometric mean one. In stage two the cross-set comparisons are a low-dimensional maximum likelihood in (log alpha, kappa) for the non-reference sets, with alpha_1 = 1 and kappa_1 = 0 fixing the reference set (the first, alphabetically).

The set units are identified here WITHIN the conditional framework: the cross-set linking uses only comparison outcomes and makes no distributional assumption about the objects. This is the substantive difference from the persons-by-items EFRM, whose item-set units can only be identified from the person side – that is, from the distribution of the persons over the linked sets. The paired-comparison design supplies its own conditional link, so no such distributional step is needed. See Humphry (2005) and Humphry and Andrich (2008) for the theory of the unit, and Thurstone (1927) and David (1988) for the varying-discriminal-dispersion lineage from which the frame-dependent unit descends. The paired-comparison form is this package's extension of Humphry's model.

The ESTIMATES are always the staged conditional estimator: the within-frame calibrations are invariant to the linking data (the frame-of-reference property), a deliberate trade of some efficiency for invariance, exactly as anchored equating trades. Inference defaults to a parametric bootstrap of the

whole pipeline (`se_method = "bootstrap"`): winners are resampled from the fitted probabilities and both stages refitted, so the standard errors of `phi`, `alpha`, `kappa`, `beta` and the common-scale `v` carry every stage's sampling variability – verified by simulation to restore nominal coverage where the conditional errors cover at a third of their nominal rate on linked designs. The `"conditional"` option keeps the fast analytic errors (judge-clustered sandwich for stage one, inverse observed information for stage two, conditional on stage one) for quick inspection; its `alpha` and `kappa` errors understate, and the fit says so.

Two honesty notes on the bootstrap. The default is model-based: replicates are drawn as independent Bernoulli outcomes at the fitted probabilities, which is self-consistent (the model has no judge parameter) but does not carry extra-model dependence within judges. When judges plausibly carry idiosyncratic preferences across their comparisons, use `se_method = "judge_bootstrap"`: judges are redrawn with replacement within each panel and the whole pipeline is refitted, so the errors carry both the stage-one uncertainty and the within-judge dependence (it needs enough judges per panel to resample stably; failed replicates are counted and reported). And a parameter that reaches its boundary in some replicates (a set unit driven to zero when a resampled within-set order flips against the cross-set evidence, the signature of a two-object set with a near-even internal pair) has no normal sampling distribution: its standard error is reported as `NA` and a note names the parameter and the boundary count rather than manufacturing a number. Relatedly, a set whose within-set contests are all near-even (or all one-sided) carries no stable information about the panel-unit ratios; such sets are screened out of the `phi` reconciliation, refit with the panel units held at the reconciled `phi` (which the frame model says apply to them regardless), and named in a note.

A single set (`S = 1`) reduces the model to panel units alone; stage two is skipped and the print states the panel-units model. When additionally `G = 1` the fit reduces exactly to `btl` on the same data. The equal-unit (single-unit) comparison refits plain `btl` on all comparisons pooled and reports the descriptive composite log-likelihood difference against the frames model; because that comparison is a composite likelihood, the inference on the units is carried by the Wald tests on `log phi_g` and `log alpha_s` in `phi_table` and `alpha_table`.

Value

An object of class `"rasch_btl_efrm"`: `objects` (object, set, `beta_set` and its standard error, common-scale `v` and its standard error), `phi_table` (panel units with Wald tests against `log phi = 0`), `alpha_table` and `kappa_table` (set units and origins with Wald tests; the reference set carries `alpha = 1`, `kappa = 0` with no standard error), `frames` (panel by set: unit `rho = phi alpha`, comparison count, pooled fit residual), `equal_unit` (the descriptive single-unit comparison), `n_cross` (cross-set comparison counts per set pair), `notes` and `converged`.

References

Bradley, R. A. and Terry, M. E. (1952). Rank analysis of incomplete block designs: I. The method of paired comparisons. Biometrika, 39, 324-345.

David, H. A. (1988). The Method of Paired Comparisons (2nd ed.). Griffin.

Humphry, S. M. (2005). Maintaining a common arbitrary unit in social measurement. PhD thesis, Murdoch University.

Humphry, S. M. and Andrich, D. (2008). Understanding the unit in the Rasch model. Journal of Applied Measurement, 9(3), 249-264.

Luce, R. D. (1959). Individual Choice Behavior. Wiley.

Thurstone, L. L. (1927). A law of comparative judgment. Psychological Review, 34, 273-286.

Examples

```
d <- simulate_btl_efrm(n_objects_per_set = 6, n_sets = 2, n_panels = 2,
                       set_units = c(1, 1.4), set_origins = c(0, 0.8),
                       seed = 1)
fit <- btl_efrm(d, "object_a", "object_b", winner = "winner",
                judge = "judge", panels = "panel",
                object_sets = attr(d, "truth")$object_sets)
fit$alpha_table
```

---

btl_equate *Equate two paired-comparison calibrations through their common objects*

Description

Compares the object locations of a Bradley-Terry-Luce fit with those of a second fit (or a banked table of object locations), matched by object name. The use case is standards maintenance and comparative judgement across panels or years: the same scripts, performances, or products are judged by different panels – or by one panel in successive years – and a common set of anchor objects is carried through so that the two rounds land on a single scale.

Usage

```
btl_equate(fit1, fit2, alpha = 0.05, p_adjust = "holm")
```

Arguments

fit1	A fitted object from btl: the calibration whose scale (origin) the equating targets.
fit2	A second btl fit, or a bank: a data frame with columns object, location, and optionally se.
alpha	Significance level for the (multiplicity-adjusted) drift tests.
p_adjust	Multiple-comparison adjustment across the common objects, passed to p.adjust (default "holm").

Details

Each calibration is identified by the sum-zero constraint, but the two constraints are imposed over *different* object sets, so the origins do not coincide even when the shared objects are unchanged: each scale is centred on the mean of a different collection. A scale shift between the two origins is therefore estimated by the precision-weighted mean difference over the common objects, and each common object is then tested against the shifted identity line. A flagged object shows drift – a script the two panels valued differently, or a standard that moved between years – and weakens the

equating link; the surviving objects carry the second calibration's whole scale onto the first (`loc2 + shift`).

The single shift presumes the drifting objects are a *minority*. When most of the common objects have genuinely moved, the precision-weighted shift is pulled toward the movers and the drift tests can invert – the stable anchors flag as the apparent drifters. Read wholesale flagging (several objects, one direction) as a contaminated link, not as evidence about the individual objects; equate through a vetted anchor subset instead. The `shift_se` accounts for the covariance of the location estimates within each calibration (each is sum-zero constrained, so its locations are not independent).

Value

A list of class `"rasch_btl_equate"`: the comparison `table` (per common object: object, both locations and standard errors, their `difference`, the `shifted_difference` against the estimated origin, the pooled `se_diff`, `t`, raw and adjusted `p`, and the `drifting` flag); the estimated `shift` and its `shift_se`; `equated`, the second calibration's full object table re-expressed on `fit1`'s scale; the number of common objects `n_common`; `alpha`; `p_adjust`; and `notes`.

References

Bramley, T. (2007). Paired comparison methods. In P. Newton, J. Baird, H. Goldstein, H. Patrick, & P. Tymms (Eds.), *Techniques for monitoring the comparability of examination standards* (pp. 246-294). London: Qualifications and Curriculum Authority.

Examples

```
set.seed(1)
beta <- setNames(seq(-2, 2, length.out = 8), paste0("O", 1:8))
sim <- function(objs) {
  pr <- t(utils::combn(objs, 2))
  d <- data.frame(a = rep(pr[, 1], each = 40), b = rep(pr[, 2], each = 40))
  d$win <- ifelse(runif(nrow(d)) < plogis(beta[d$a] - beta[d$b]), d$a, d$b)
  btl(d, "a", "b", "win")
}
eq <- btl_equate(sim(paste0("O", 1:7)), sim(paste0("O", 2:8)))
eq$table
```

`btl_information` *Information and targeting of a paired-comparison design*

Description

The paired-comparison analogue of the test-information function. The Fisher information a single comparison carries about the location difference `d = beta_a - beta_b` is, in this exponential family, the variance of its score – `P(1 - P)` for the dichotomous choice and the graded response variance `V` for the ordinal extension (the score is the sufficient statistic for `d`, so its variance is the information). Weighted by each comparison's replication count and summed over the comparisons the design actually contains, this gives a *design information* for every object: how much the observed comparisons pin its location down, the counterpart of an item's contribution to test information. Because

the information peaks at gap zero and falls away with the location gap, near-neighbour contests are the informative ones.

Usage

```
btl_information(fit)
```

Arguments

`fit`	A paired-comparison fit from `btl`.

Details

The design information inverts to `se_naive = 1 / sqrt(information)`: the error the object's comparisons would give if its location were the ONLY free parameter – a single-parameter lower bound, useful for reading which objects the design serves well. It is not the model's standard error even with independent comparisons (every location is estimated jointly with the others, and the fit's own `se` is additionally the judge-clustered Godambe sandwich), so `se` sits above `se_naive` as a rule; treat their ratio as descriptive, not as a clustering test.

Value

A list of class `"rasch_btl_info"`: `objects` (per object: `location`, the fit's `se`, `n_comparisons`, the design `information`, and `se_naive`); `pairs` (per observed pair: `n`, the mean location `gap`, and the pair's `information`); `comparisons` (per comparison: the signed `gap`, `weight`, and the single-comparison `information`); the scalar `total` information; `m`; the `clustered` flag; and `notes`.

References

Pollitt, A. (2012). The method of adaptive comparative judgement. *Assessment in Education*, 19(3), 281-300.

See Also

`plot_btl_targeting`, `btl_next_pairs`

Examples

```
set.seed(1)
beta <- c(A = -1, B = -0.3, C = 0.4, D = 0.9)
pr <- t(combn(names(beta), 2))
d <- data.frame(a = rep(pr[, 1], each = 30), b = rep(pr[, 2], each = 30))
d$win <- ifelse(runif(nrow(d)) < plogis(beta[d$a] - beta[d$b]), d$a, d$b)
btl_information(btl(d, "a", "b", "win"))
```

`btl_next_pairs` *Recommend the next informative comparisons (adaptive step)*

Description

The adaptive comparative judgement step of Pollitt (2012): rank candidate object pairs by the information one additional comparison would carry at the current estimates. That information peaks when the two objects are close in location, so at equal measurement the recommender favours near-neighbour contests. With `weight_se = TRUE` (the default) each pair's priority is the one-step reduction in TOTAL location variance that one added comparison of the pair would deliver, from a rank-one (Sherman-Morrison) update of the fit's stored covariance with the comparison's information on the contrast, so pairs of poorly measured (and correlated) objects are promoted. The update formula is exact for a model-based information matrix; applied to the sandwich covariance it is a scoring device, consistent with the ranking-heuristic status described below.

Usage

```
btl_next_pairs(fit, n = 10, weight_se = TRUE)
```

Arguments

`fit`	A paired-comparison fit from `btl`.
`n`	Number of pairs to return. The priority is a greedy one-step RANKING heuristic: it plugs the judge-clustered sandwich covariance into an information-update formula that is exact only for a model-based information matrix, so the ranking orders candidate pairs sensibly but the implied variance reductions are not exact sandwich updates. Treat the ordering, not the magnitudes, as the output.
`weight_se`	Rank by the one-step total-variance reduction (default `TRUE`; falls back to `expected_information * (se_a^2 + se_b^2)` if the fit carries no covariance). When `FALSE`, pairs are ranked by expected information alone (pure closeness).

Details

Two honest cautions. This is a *greedy* rule that scores each pair on its own immediate one-step gain (an A-optimality step at the current estimates, taking the clustered covariance as the state); it is not a full optimal design and can be beaten by one that plans several comparisons jointly. And adaptive selection is known to inflate a separation (scale) reliability computed naively afterwards, because the design concentrates comparisons where they shrink the errors most: report reliability from an independent or non-adaptive subset, or treat an adaptive reliability as an upper bound (Bramley 2015).

Value

A data frame of the top `n` candidate pairs, each oriented to its stronger object: `object_a`, `object_b`, the location `gap`, `n_existing` (replications already observed for the pair), `expected_information` (of one new comparison), and `priority`. Sorted by `priority` (or by `expected_information` when `weight_se = FALSE`).

References

Pollitt, A. (2012). The method of adaptive comparative judgement. *Assessment in Education*, 19(3), 281-300. Bramley, T. (2015). Investigating the reliability of Adaptive Comparative Judgment. *Cambridge Assessment Research Report*.

See Also

btl_information, plot_btl_targeting

Examples

```
set.seed(1)
beta <- c(A = -1, B = -0.3, C = 0.4, D = 0.9)
pr <- t(combn(names(beta), 2))
d <- data.frame(a = rep(pr[, 1], each = 30), b = rep(pr[, 2], each = 30))
d$win <- ifelse(runif(nrow(d)) < plogis(beta[d$a] - beta[d$b]), d$a, d$b)
btl_next_pairs(btl(d, "a", "b", "win"), n = 5)
```

---

btl_transitivity *Transitivity of paired comparisons*

Description

The single-dimension analogue for paired comparisons of the unidimensionality question. A Bradley-Terry-Luce scale implies that preferences stack into one consistent order: if A beats B and B beats C then A should beat C. A *circular triad* (A beats B, B beats C, C beats A) is a local contradiction, like rock-paper-scissors. A few are sampling noise; many, systematically, mean the comparisons are not being driven by a single attribute. The rate of circular triads is compared with the value expected from pure guessing (one quarter of triples), and, when every pair has been compared, Kendall's coefficient of consistency is reported (Kendall & Babington Smith 1940). With judges, each judge's own consistency is reported too, flagging judges whose choices approach chance.

Usage

```
btl_transitivity(fit, min_triples = 5L)
```

Arguments

fit	A paired-comparison fit from btl.
min_triples	A judge is reported only if this many complete triples (all three pairs judged) are available.

Value

A list of class "rasch_btl_transitivity": summary (one row: objects, pairs compared, complete triples, circular triads, the circular rate, the chance rate 0.25, the consistency index 1 - rate/0.25, and Kendall's zeta when the design is a complete round-robin with no exactly-tied pair – NA otherwise); objects (each object's circular-triad involvement); judges (per-judge consistency, when judges exist); and notes.

References

Kendall, M. G., & Babington Smith, B. (1940). On the method of paired comparisons. *Biometrika*, 31(3/4), 324-345.

Examples

```
set.seed(1); objs <- LETTERS[1:6]; beta <- setNames(seq(-1.5, 1.5, len = 6), objs)
pr <- t(utils::combn(objs, 2))
d <- data.frame(a = rep(pr[, 1], each = 20), b = rep(pr[, 2], each = 20))
d$win <- ifelse(runif(nrow(d)) < plogis(beta[d$a] - beta[d$b]), d$a, d$b)
btl_transitivity(btl(d, "a", "b", "win"))
```

---

chisq_detail *Class-interval detail for one item's chi-square test of fit*

Description

The per-class-interval breakdown behind an item's item-trait chi-square, as dissected in Andrich and Marais (2019, ch. 13): for every class interval the size, the maximum and mean person location, the standardised residual between observed and expected interval means, its squared chi-square component, the observed and expected means (OM, EV), the sample-size-free effect size ES = (OM - EV)/sqrt(mean V), and per response category the observed proportion (OBS.P), the mean model probability (EST.P), and the observed conditional threshold proportion (OBS.T), the proportion scoring k among those scoring k - 1 or k.

Usage

```
chisq_detail(fit, item)
```

Arguments

`fit`	A fitted object from `rasch`.
`item`	Item name or index.

Value

A list with `item`, `location`, the `intervals` data frame, the `categories` data frame, the whole-sample observed mean `ave`, and the item's total `chisq`, `df`, and `p`. Intervals with fewer than 2 responders are shown but carry no chi-square contribution (`used = FALSE`), matching the item-trait computation.

Examples

```
set.seed(1)
d <- seq(-1.5, 1.5, length.out = 6)
X <- matrix(rbinom(400 * 6, 1, plogis(outer(rnorm(400), d, "-"))), 400, 6)
colnames(X) <- paste0("I", 1:6)
chisq_detail(rasch(X), "I3")$intervals
```

combine_items *Combine items into subtests and re-analyse*

Description

Forms one polytomous super-item from each nominated group of items (its score is the member sum; missing if any member is missing), keeps all other items as they are, and refits the model with the same settings. The usual treatment for item pairs flagged by `residual_correlations`.

Usage

```
combine_items(fit, groups, model = "PCM")
```

Arguments

`fit`	A fitted object from `rasch`.
`groups`	A list of character vectors, each naming two or more items to combine; a single vector is also accepted.
`model`	Model for the re-analysis; defaults to `"PCM"`, which is almost always required because subtests change the maximum scores.

Value

A new `rasch` fit on the combined structure, with the combinations recorded in its notes.

Examples

```
set.seed(1); Np <- 500; L <- 8
d <- seq(-2, 2, length.out = L)
X <- matrix(rbinom(Np * L, 1, plogis(outer(rnorm(Np), d, "-"))), Np, L)
colnames(X) <- paste0("I", 1:L)
X[, 5] <- ifelse(runif(Np) < 0.9, X[, 4], X[, 5])   # dependent pair
fit <- rasch(X)
fit2 <- combine_items(fit, list(c("I4", "I5")))
fit2$items$item
```

compare_fits *Compare fitted Rasch models*

Description

Builds a comparison table for two or more fits from `rasch`, `rasch_mfrm`, `rasch_efrm`, or (all together) `btl`. For fits of the same response data (identical item columns, maximum scores, and number of persons) the pairwise conditional log-likelihoods share their conditional information, and twice the difference from the reference fit is reported with the difference in parameter counts; this is descriptive (composite likelihood), and most meaningful for nested structures such as RSM inside PCM.

Usage

```
compare_fits(..., reference = 1)
```

Arguments

`...`	Two or more fitted objects, ideally named (`compare_fits(PCM = f1, RSM = f2)`). Either all Rasch-family fits or all `btl` fits; for `btl`, fits of the same comparison data (same objects, comparisons, and judges) support the likelihood columns – e.g. free versus principal-component thresholds, with and without a position effect or within-judge dependence.
`reference`	Index or name of the reference fit for the log-likelihood difference; defaults to the first.

Details

The calibrated comparison is carried by the composite-likelihood information criteria `cl_aic` (Varin and Vidoni 2005) and `cl_bic` (Gao and Song 2010): $-2\,cl + c \cdot tr(H^{-1}J)$, with $c = 2$ or $\log n$. Because every response enters every pair its item forms, the pairwise log-likelihood over-counts the data; the effective parameter count $tr(H^{-1}J)$ from the Godambe matrices – the same quantity whose eigenvalues calibrate `lr_test` – absorbs exactly that over-counting, where the nominal parameter count would not. n counts independent units: persons contributing at least one informative pair, or judges for paired-comparison fits (count-weighted comparisons when unclustered). Smaller is better; the criteria are valid across models of the same data whether or not they nest, and are `NA` (with the reason in the printed note) for MFRM and EFRM fits, which do not carry their Godambe matrices.

Across different data preparations (subtests, splits, facet or frame structures) the likelihoods are not comparable and the calibration-free columns carry the comparison: total item-trait chi-square per degree of freedom, item and person fit residual SDs (ideal 1), PSI, and alpha (OSI for paired comparisons).

Value

A data frame with one row per fit: label, model, persons, items (judges, objects, comparisons for `btl`), parameters, log-likelihood, `eff_params`, `cl_aic`, `cl_bic`, comparability with the reference, `two_delta_ll` and `delta_parameters` (same-data fits only), chi-square per df, fit residual SDs, PSI, and alpha (OSI for `btl`).

Examples

```
set.seed(1)
simP <- function(th, tau) { x <- 0:length(tau); p <- exp(x * th - c(0, cumsum(tau))); p / sum(p) }
th <- rnorm(400)
X <- sapply(seq(-1, 1, length.out = 6), function(b)
  sapply(th, function(t) sample(0:3, 1, prob = simP(t, b + c(-0.8, 0, 0.8)))))
colnames(X) <- paste0("R", 1:6)
compare_fits(PCM = rasch(X, model = "PCM"), RSM = rasch(X, model = "RSM"))
```

ctt_table *Traditional (classical test theory) statistics*

Description

The classical companion table conventionally reported alongside a Rasch analysis (Andrich and Marais 2019, chs. 3-5), on complete cases only: per item the facility (mean score over maximum), the item-total and corrected item-rest correlations, the discrimination index DI = PRU - PRL (mean proportion-of-maximum in the upper third of total scores minus the lower third), and alpha if the item is deleted; plus coefficient alpha, the raw-score mean, SD, and the classical standard error of measurement $s\sqrt{1-\alpha}$, which unlike the Rasch SE is one value for all persons.

Usage

```
ctt_table(fit)
```

Arguments

`fit`	A fitted object from `rasch`.

Value

A list of class `"rasch_ctt"`: the per-item `table` (`item`, `n`, `min`, `max`, `facility`, `item_total`, `item_rest`, `di`, `alpha_drop`), and the scalars `alpha`, `n` (complete cases), `mean`, `sd`, and `sem`.

Examples

```
set.seed(1)
d <- seq(-2, 2, length.out = 8)
X <- matrix(rbinom(400 * 8, 1, plogis(outer(rnorm(400), d, "-"))), 400, 8)
colnames(X) <- paste0("I", 1:8)
ctt_table(rasch(X))
```

dependence_magnitude *Estimate the magnitude of response dependence between two items*

Description

Quantifies how strongly a dependent item's response follows an independent item's response, in logits, by the resolution method of Andrich and Kreiner (2010; polytomous generalisation Andrich, Humphry and Marais 2012), by resolution of the dependent item. The dependent item is resolved into one item per category of the independent item (each carrying the responses of the persons who gave that category), both original items are removed, and the model refitted. Under dependence of magnitude d, threshold k of the resolved item for category x_i is shifted by $-d$ when $k \leq x_i$ and $+d$ otherwise, so each threshold yields $\hat{d}_k = (\hat{\delta}_{ji(k)}(x_i = k-1) - \hat{\delta}_{ji(k)}(x_i = k))/2$ and $\hat{d}$ is their mean (eq. 24.7 of Andrich and Marais 2019). Because the resolved items are answered by disjoint persons, the estimates are independent, and the standard error pools the threshold variances: $\hat{\sigma}^2_k = (\hat{\sigma}^2_{(k)(k-1)} + \hat{\sigma}^2_{(k)(k)})/4$, with $V[\hat{d}] = \hat{\bar{\sigma}}^2_k/m$ (eqs. 24.9-24.11).

Usage

```
dependence_magnitude(fit, dependent, independent)
```

Arguments

`fit`	A fitted object from `rasch`.
`dependent, independent`	Item names or indices: the item hypothesised to depend, and the item it depends on. Both must share the same maximum score (the formalisation requires it).

Value

A list of class `"rasch_dependence"`: the estimate `d`, its `se`, `z` and `p` for the hypothesis $d = 0$, the per-threshold table `thresholds` (columns `k`, `delta_lo`, `delta_hi`, `d_k`, `se_k`), and the resolved `refit`.

References

Andrich, D. and Kreiner, S. (2010). Quantifying response dependence between two dichotomous items using the Rasch model. Applied Psychological Measurement, 34, 181-192. Andrich, D., Humphry, S. M. and Marais, I. (2012). Quantifying local, response dependence between two polytomous items using the Rasch model. Applied Psychological Measurement, 36, 309-324.

Examples

```
set.seed(1); N <- 700
d0 <- seq(-1.5, 1.5, length.out = 8)
X <- matrix(rbinom(N * 8, 1, plogis(outer(rnorm(N), d0, "-"))), N, 8)
X[, 5] <- ifelse(runif(N) < 0.75, X[, 4], X[, 5])   # I5 follows I4
colnames(X) <- paste0("I", 1:8)
dependence_magnitude(rasch(X), dependent = "I5", independent = "I4")
```

`dif_anova` *Differential item functioning by residual analysis of variance*

Description

For each item the standardised residuals are analysed by the nominated person factor(s) crossed with the trait class interval. A term not involving the class interval is uniform DIF; a term crossing it is non-uniform DIF (Hagquist and Marais 2019, ch. 16). With one factor this is a one-way analysis, `z ~ g * ci`. With several factors they are modelled jointly – the statistically correct treatment, rather than one factor at a time – with main effects by default (`z ~ (f1 + f2 + ...) * ci`); set `effects = "factorial"` to add the factor-by-factor interactions (`z ~ (f1 * f2 * ...) * ci`). When interactions are fitted, a significant one supersedes the lower-order terms built from its variables, recorded in the `superseded` column; interpret the highest-order significant terms.

Usage

```
dif_anova(
  fit,
  factors = NULL,
  n_groups = NULL,
  p_adjust = "BH",
  alpha = 0.05,
  effects = c("main", "factorial"),
  sizes = FALSE,
  id = NULL,
  within = NULL
)
```

Arguments

`fit`	A fitted object from `rasch`.
`factors`	A vector (one factor), a data frame of person factors, or a character vector naming factor columns nominated in the fit. Defaults to every factor stored in the fit.
`n_groups`	Number of trait class intervals. By default set from the smallest factor-combination cell so every interval-by-cell count keeps about 30 expected responses (between 2 and 10 intervals); the value used is returned in `n_groups`.
`p_adjust`	Multiplicity adjustment across items within each term; default `"BH"`.
`alpha`	Significance level applied to the adjusted probabilities.
`effects`	`"main"` (default) models several factors additively (each factor's main effect and its class-interval interaction, but no factor-by-factor terms); `"factorial"` also crosses the factors with each other. Immaterial with a single factor.
`sizes`	Also compute DIF magnitudes in logits (`dif_size`) for every significant, non-superseded group term: the item is resolved by the term's levels (interaction terms by their cells) and all pairwise location differences are returned with Holm familywise adjustment and the practical-significance flag. Each size involves a re-analysis, so this costs one refit per flagged item-term.
`id`, `within`	Person identifier and within-subject factor names for stacked repeated-measures designs. When any factor is within-subject the model becomes a mixed (split-plot) analysis of variance – the class interval taken at the person level, the within factors carrying a person error stratum – so their terms are tested validly. Auto-detected from the fit's person identifier.

Details

Probabilities are adjusted across items within each term (Benjamini-Hochberg by default). Tukey HSD comparisons are returned for each significant, non-superseded group term. Sums of squares are sequential (factors in the order given, class interval last).

Value

A list with `summary`, the compact reading of the analysis (one row per item and group term with the uniform F, adjusted p, and partial eta-squared – the term itself – and the non-uniform ones – the term crossed with class interval – plus `uniform_DIF`, `nonuniform_DIF` and `superseded` flags); `terms`, the complete per-item analysis of variance table (term, df, sum of squares, mean square, F, partial eta-squared, raw and adjusted p, significance, supersession, including the residual row); and `tukey` (per item, term, and level comparison: difference, 95 per cent interval, and Tukey-adjusted p), plus the `alpha` and adjustment used. Tukey comparisons are reported for significant, non-superseded group terms except two-level main effects, where the F test is already the only comparison. With `sizes = TRUE`, `sizes` holds the logit DIF magnitudes per item, term, and level pair (two-level main effects included, since the single difference is exactly the DIF size).

Examples

```
set.seed(1); n <- 800
d <- seq(-1.5, 1.5, length.out = 6)
g1 <- rep(c("a", "b"), each = n / 2)
g2 <- rep(c("x", "y"), times = n / 2)
sh <- matrix(0, n, 6); sh[g1 == "b", 2] <- 0.8
X <- matrix(rbinom(n * 6, 1, plogis(outer(rnorm(n), d, "-") - sh)), n, 6)
colnames(X) <- paste0("I", 1:6)
fit <- rasch(data.frame(X, g1 = g1, g2 = g2), factors = c("g1", "g2"))
dif_anova(fit)$summary
```

---

dif_contrasts *Planned DIF contrasts derived from the factor structure*

Description

The confirmatory alternative to exhaustive post-hoc comparison: instead of every pair of design cells, a small family of one-degree-of-freedom questions is tested, so familywise control costs little power (Maxwell and Delaney 2004, ch. 5). By default the family is derived from the structure of the factors themselves – a two-level factor contributes its difference; an ordered factor (declared ordered, or with numeric levels such as ages or waves) contributes its linear and quadratic trends; a nominal factor contributes all pairs when it has up to four levels and each-level-against-the-rest otherwise; and every pair of factors with a leading contrast (a difference or a linear trend) contributes the product interaction. Print the returned object to see the family in words before reading the results; a family endorsed in advance of the results is what makes the contrasts planned.

Usage

```
dif_contrasts(
  fit,
  factors = NULL,
  items = NULL,
  within = NULL,
  id = NULL,
```

```
  contrasts = "auto",
  p_adjust = "holm",
  alpha = 0.05,
  flag_logits = 0.5,
  min_n = 20
)
```

Arguments

fit	A fitted object from rasch.
factors	A data frame of person factors, a character vector naming factors nominated in the fit, or a single grouping vector. Defaults to every factor stored in the fit.
items	Item names or indices to test; all items by default.
within	Names of factors that vary within person (for example time). Detected automatically when id is supplied and a factor varies within an id.
id	Person identifier with one entry per row, or the name of a nominated factor holding it; required for stacked designs where the same person occupies several rows.
contrasts	"auto" (derive the family from the factor structure) or a named list of numeric cell-weight vectors, each named by the design-cell labels (factor levels joined by ":"). Weights are rescaled so the positive and negative parts each sum to one.
p_adjust	Familywise adjustment over the whole family (items by contrasts); default "holm".
alpha	Significance level for the adjusted probabilities.
flag_logits	Absolute estimate flagged as practically significant.
min_n	Cells with fewer responders to an item are dropped from that item's resolution, with a note.

Details

Each contrast is estimated in logits from resolved item locations (the item split into one copy per design cell and the model refitted, as in dif_size), with cell weights scaled so every estimate is a difference between two weighted averages – directly comparable to the practical-significance criterion. Because resolution is used, magnitudes are read from a calibration in which compensating artificial DIF has been removed (Andrich and Hagquist 2015).

When id shows that persons repeat across rows (a stacked repeated-measures design), between-row independence fails and the usual tests would be invalid. Significance is then computed from person-level scores of the standardised residuals: a within-subject contrast (for example a trend over time) becomes one contrast score per person, tested against zero; a between-subjects contrast is tested on person-mean residuals; and a between-by-within interaction tests the person contrast scores across the between groups. Logit estimates are still reported from the resolved locations; their standard errors treat rows as independent and are conservative for within-subject differences.

Value

A list of class `"rasch_dif_contrasts"`: `table` (one row per item and contrast: estimate in logits, SE, statistic, df where a t test was used, raw and adjusted p, 95 per cent interval, `significant`, `practical`, `within`), `family` (the derived questions with their cell weights), the settings, and any `notes`.

References

Maxwell, S. E., & Delaney, H. D. (2004). *Designing Experiments and Analyzing Data* (2nd ed.). Mahwah, NJ: Erlbaum.

Andrich, D., & Hagquist, C. (2015). Real and artificial differential item functioning in polytomous items. *Educational and Psychological Measurement*, 75(2), 185-207.

Hagquist, C., & Andrich, D. (2017). Recent advances in analysis of differential item functioning in health research using the Rasch model. *Health and Quality of Life Outcomes*, 15, 181.

Examples

```
set.seed(1); n <- 600
d <- seq(-2, 2, length.out = 8); g <- rep(c("a", "b"), each = n / 2)
sh <- matrix(0, n, 8); sh[g == "b", 3] <- 0.8
X <- matrix(rbinom(n * 8, 1, plogis(outer(rnorm(n), d, "-") - sh)), n, 8)
colnames(X) <- paste0("I", 1:8)
fit <- rasch(data.frame(X, grp = g), factors = "grp")
dif_contrasts(fit, items = c("I3", "I5"))
```

---

dif_size *DIF magnitude in logits with pairwise comparisons*

Description

Quantifies differential item functioning on the measurement scale itself, where practical significance is judged: the item is resolved into one copy per group (or per cell of a factor combination), the model is refitted, and the distance between the resolved locations is the DIF size in logits (Andrich & Marais 2019, ch. 16: a simulated shift of 0.71 was recovered as 0.75 by exactly this method). Every pair of levels is compared with a Wald test using the full sandwich covariance of the resolved locations (for a between-person factor the persons behind different levels are disjoint, but the shared calibration of the other items still couples the estimates, so the covariance is used rather than assumed zero), with familywise adjustment over the pairs. For a within-person factor – the same persons behind several levels, as in a stacked repeated-measures design – the sandwich carries no person clustering, so the standard errors are conservative; a note says so, and `dif_contrasts` handles that case with person-level differencing. Differences at least `flag_logits` in absolute size are flagged as practically significant; half a logit is a common working criterion, to be weighed against the test's targeting and purpose.

Usage

```
dif_size(
  fit,
  item,
  by,
  p_adjust = "holm",
  alpha = 0.05,
  flag_logits = 0.5,
  min_n = 20
)
```

Arguments

fit	A fitted object from rasch.
item	Item name or index.
by	One or more person-factor names nominated in the fit (several names give interaction cells), or a grouping vector/data frame with one entry per person.
p_adjust	Familywise adjustment over the pairwise comparisons; default "holm".
alpha	Significance level for the adjusted probabilities.
flag_logits	Absolute difference flagged as practically significant.
min_n	Levels with fewer responders to the item are dropped (their resolved locations would be too unstable to compare), with a note.

Details

For an interaction, supply several factor names: levels are then the factor-combination cells, which is the post-hoc follow-up to a significant factor-by-factor term in dif_anova.

Value

A list of class "rasch_dif_size": levels (resolved location and SE per level, with its n), pairs (per comparison: difference in logits, SE, z, raw and adjusted p, 95 per cent interval, significant, practical), the settings, and any notes.

Examples

```
set.seed(1); n <- 600
d <- seq(-2, 2, length.out = 8); g <- rep(c("a", "b"), each = n / 2)
sh <- matrix(0, n, 8); sh[g == "b", 3] <- 0.8
X <- matrix(rbinom(n * 8, 1, plogis(outer(rnorm(n), d, "-") - sh)), n, 8)
colnames(X) <- paste0("I", 1:8)
fit <- rasch(data.frame(X, grp = g), factors = "grp")
dif_size(fit, "I3", by = "grp")
```

`dimensionality_magnitude` *Magnitude of multidimensionality from a subtest analysis*

Description

Estimates how strongly two or more hypothesised subscales measure distinct traits, by Andrich's (2016) comparison of two reliability calculations: one treating all items as independent (which inflates reliability under multidimensionality) and one on the subtest analysis in which each subscale is combined into a single polytomous super-item (which absorbs the unique subscale variance). Under the bifactor formalisation $\beta_{ns} = \beta_n + c\,\beta'_{ns}$ (Marais and Andrich 2008), with S subscales of K items,

$$c^2 = S\,(r_1/r_2 - 1)\,\frac{SK-1}{S(K-1)},$$

the latent correlation between subscales is $\rho = 1/(1+c^2)$, and $A = S/(S+c^2)$ is the proportion of common (non-unique, non-error) variance. Both the person separation index and coefficient alpha versions are reported (Andrich and Marais 2019, ch. 24).

Usage

```
dimensionality_magnitude(fit, subtests)
```

Arguments

`fit`	A fitted object from `rasch`.
`subtests`	A list of character vectors assigning *every* item of the fit to one subscale (at least two subscales of two or more items). Unequal subscale sizes use their mean as K.

Value

A list of class `"rasch_dim_magnitude"`: the comparison `table` (rows PSI and alpha; columns `run1`, `subtest`, `c2`, `c`, `rho`, `A`), the subtest `refit`, and the design constants `S` and `K`.

References

Andrich, D. (2016). Components of variance of scales with a bifactor structure from two calculations of coefficient alpha. Educational Measurement: Issues and Practice, 35(4), 25-30.

Examples

```
set.seed(1); N <- 500
common <- rnorm(N); u1 <- rnorm(N); u2 <- rnorm(N)
d <- rep(seq(-1, 1, length.out = 5), 2)
X <- sapply(1:10, function(i) rbinom(N, 1,
  plogis(common + 0.8 * (if (i <= 5) u1 else u2) - d[i])))
colnames(X) <- paste0("I", 1:10)
```

```
fit <- rasch(X)
dimensionality_magnitude(fit,
  list(paste0("I", 1:5), paste0("I", 6:10)))$table
```

`dimensionality_test` *Residual-component test of unidimensionality*

Description

Estimates each person separately on two item subsets and compares the two estimates with a per-person t-test (Smith 2002). By default the subsets are defined by the sign of a residual-component loading (the first by default; any leading component may be chosen); they can also be nominated manually (for example, by content). Under unidimensionality and local independence the two subset estimates are independent given the person location, so `t = (theta_A - theta_B) / sqrt(se_A^2 + se_B^2)` is approximately standard normal and about `alpha` of the tests should reach significance. Persons with an extreme score on either subset are excluded (their weighted-likelihood estimates are most biased there). The proportion of significant tests is reported with an exact (Clopper-Pearson) binomial confidence interval; a lower bound above `alpha` signals multidimensionality.

Usage

```
dimensionality_test(
  fit,
  alpha = 0.05,
  items_positive = NULL,
  items_negative = NULL,
  component = 1
)
```

Arguments

`fit`	A fitted object from `rasch`.
`alpha`	Nominal significance level for the per-person t-tests.
`items_positive, items_negative`	Optional character vectors naming the two item subsets; both must be given (disjoint, at least two items each), otherwise the sign of a residual component defines the split.
`component`	Which residual principal component's loading sign defines the default split (ignored when subsets are named). Default the first component.

Value

A list with the proportion of significant tests, its exact confidence interval, the sample sizes (`n` used, `n_excluded_extreme`), the item split and its source, a `multidimensional` verdict, and `paired_t`, the paired t-test of the two subset means (the group-level comparison, which requires pairing because both estimates come from the same persons).

Examples

```
set.seed(1)
d <- seq(-2, 2, length.out = 8)
X <- matrix(rbinom(500 * 8, 1, plogis(outer(rnorm(500), d, "-"))), 500, 8)
colnames(X) <- paste0("I", 1:8)
dimensionality_test(rasch(X))$multidimensional
```

`distractor_analysis` *Distractor analysis for multiple-choice items*

Description

For every keyed item and response option: the count and proportion choosing it, the mean location of those persons, and the point-biserial correlation between choosing the option and the person measure. Locations and correlations use the rest measure (the person estimate from the other items), so the analysed item cannot credit its own takers. The keyed option should attract the ablest persons and carry the only positive point-biserial; a distractor whose takers are abler than the keyed option's (with at least `min_n` takers) is flagged as a possible miskey.

Usage

```
distractor_analysis(fit, items = NULL, min_n = 10)
```

Arguments

`fit`	A fitted object from `rasch` run with a `key`.
`items`	Optional subset of item names; defaults to every keyed item.
`min_n`	Minimum takers for an option to be eligible for the miskey flag.

Value

A data frame with one row per item-option: `item`, `option`, its assigned `score`, `keyed` (full credit), `n`, `prop`, `mean_location`, `point_biserial`, and `flag`.

Examples

```
set.seed(1); Np <- 400
th <- rnorm(Np)
raw <- sapply(seq(-1, 1, length.out = 6), function(d) {
  ok <- rbinom(Np, 1, plogis(th - d))
  ifelse(ok == 1, "A", sample(c("B", "C", "D"), Np, replace = TRUE))
})
colnames(raw) <- paste0("M", 1:6)
fit <- rasch(raw, key = setNames(rep("A", 6), colnames(raw)))
head(distractor_analysis(fit))
```

`distractor_rescore` *Propose polytomous option scores from the distractor evidence*

Description

Multiple-choice items can be rescored polytomously so that a distractor carrying information about the trait receives partial credit (Andrich and Styles 2011). This function proposes such a scoring from the rest-measure distractor analysis: within each keyed item, a distractor qualifies for credit when it attracts at least `min_n` takers, its takers' mean rest location exceeds that of the uncredited distractors by more than `z` standard errors of the difference, and it remains below the keyed option. Qualifying distractors are ranked by mean location and scored 1, 2, ... below the keyed option's top score. The result is a proposal for substantive review, not an automatic decision: inspect `plot_distractors` and the item content, edit as needed, then refit with `rasch(raw_data, key = proposal$option_scores)`.

Usage

```
distractor_rescore(fit, items = NULL, min_n = 20, z = 1.96)
```

Arguments

`fit`	A fitted object from `rasch` run with a `key`.
`items`	Optional subset of keyed item names.
`min_n`	Minimum takers for a distractor to be considered.
`z`	Required separation, in standard errors, between a credited distractor and the uncredited ones.

Value

A list of class `"rasch_rescore"`: `option_scores`, a data frame (`item`, `option`, `score`) ready for `rasch(key = )` and covering every observed option of the examined items, and `evidence`, the distractor analysis with the proposed scores and the separation z per option.

References

Andrich, D. and Styles, I. (2011). Distractors with information in multiple choice items: A rationale based on the Rasch model. Journal of Applied Measurement, 12, 67-95.

Examples

```
set.seed(1); Np <- 600
th <- rnorm(Np)
raw <- sapply(seq(-0.5, 0.5, length.out = 4), function(d) {
  x <- vapply(th, function(b) sample(0:2, 1,
    prob = item_moments(b, c(d - 0.7, d + 0.7))$P), 0L)
  c("D", "B", "A")[x + 1]   # B is an informative distractor
})
```

```
colnames(raw) <- paste0("M", 1:4)
fit <- rasch(raw, key = setNames(rep("A", 4), colnames(raw)))
pr <- distractor_rescore(fit)
pr$option_scores
```

`equate_tests` *Equate two test calibrations through their common items*

Description

Compares the item locations of a fit with those of a second fit (or a reference table such as an item bank), matched by item name. A scale shift between the two origins is estimated by the precision-weighted mean difference, and each common item is then tested against the shifted identity line; flagged items show drift and weaken the equating link.

Usage

```
equate_tests(fit, reference, shift = c("mean", "none"))
```

Arguments

`fit`	A fitted object from `rasch`.
`reference`	A second `rasch` fit, or a data frame with columns `item`, `location`, and optionally `se`.
`shift`	`"mean"` (default) allows a scale shift between the two analyses; `"none"` compares raw locations, appropriate when both analyses are already on a shared (anchored) scale.

Value

A list with the comparison `table` (locations, standard errors, difference, t, raw and BH-adjusted p, drift flag), the estimated `shift`, the location `correlation`, the root mean square difference after shifting (`rmsd`), and the number of common items `n`.

Examples

```
set.seed(1); d <- seq(-1.5, 1.5, length.out = 8)
mk <- function() {
  X <- matrix(rbinom(400 * 8, 1, plogis(outer(rnorm(400), d, "-"))), 400, 8)
  colnames(X) <- paste0("I", 1:8); rasch(X)
}
eq <- equate_tests(mk(), mk())
eq$table
```

fit_summary_table *Test-of-fit summary as a table*

Description

The headline fit statistics of a calibration – model, estimation, total item-trait chi-square, the item and person fit-residual moments, fit-location correlations, chi-square flag count, and disordered thresholds – as a two-column table suitable for saving and reporting.

Usage

```
fit_summary_table(fit)
```

Arguments

fit	A fitted object from rasch, or a paired-comparison fit from btl (which reports its own headline set: convergence, pairwise chi-square, object separation, thresholds structure, and dependence effects when estimated).

Value

A data frame with columns statistic and value.

Examples

```
set.seed(1)
d <- seq(-2, 2, length.out = 6)
X <- matrix(rbinom(300 * 6, 1, plogis(outer(rnorm(300), d, "-"))), 300, 6)
colnames(X) <- paste0("I", 1:6)
fit_summary_table(rasch(X))
```

guttman_table *Guttman-ordered response matrix and reproducibility*

Description

Orders persons by location (descending) and items by location (ascending), the Guttman scalogram arrangement, and computes the coefficient of reproducibility against the deterministic Guttman pattern implied by each person's total score.

Usage

```
guttman_table(fit)
```

Arguments

fit	A fitted object from rasch.

Value

A list with the ordered score matrix `matrix` (persons by items, row and column names carrying the ID and item labels), the person and item orderings, and the coefficient of reproducibility `CR`.

Examples

```
set.seed(1)
d <- seq(-2, 2, length.out = 6)
X <- matrix(rbinom(200 * 6, 1, plogis(outer(rnorm(200), d, "-"))), 200, 6)
colnames(X) <- paste0("I", 1:6)
guttman_table(rasch(X))$CR
```

---

item_moments *Category-score moments for a polytomous item*

Description

Category-score moments for a polytomous item

Usage

```
item_moments(theta, tau_i, disc = 1)
```

Arguments

`theta`	Person location, in logits.
`tau_i`	Numeric vector of the item's threshold parameters.
`disc`	Discrimination (frame unit) multiplier on the exponent; 1 for the ordinary Rasch model.

Value

A list with category probabilities `P`, expected score `E`, variance `V`, and third and fourth central moments `mu3`, `mu4`.

Examples

```
item_moments(0.5, c(-1, 0, 1))
```

`judge_pair_surprise` *Unexpected judgements of one judge, pair by pair*

Description

The comparison-level companion of `judge_surprise`. Each pair the judge met is oriented to its stronger object (higher consensus location) and given a standardised residual: `z < 0` means the stronger object won less than its lead predicts – the judge backed the underdog. A matchup is an unexpected judgement when `z` falls at or below `-flag_z` and the pair was seen at least `min_n` times, i.e. the judge favoured the weaker object further than sampling noise explains.

Usage

```
judge_pair_surprise(fit, judge, min_n = 1L, flag_z = 1.96)
```

Arguments

`fit`	A paired-comparison fit from `btl` with judges.
`judge`	The judge to profile.
`min_n`	Pairs met fewer times are shown but never flagged.
`flag_z`	Absolute residual at or beyond which an upset is flagged.

Value

A list of class `"rasch_btl_judge_pairs"`: `pairs` (per matchup: the stronger and weaker object and their locations, the location `gap`, times met `n`, residual `z`, the `net_winner`, and the `surprise` flag); `all_locations`; the `judge` and settings.

Examples

```
set.seed(1); objs <- LETTERS[1:6]; beta <- setNames(seq(-1.5, 1.5, len = 6), objs)
pr <- t(utils::combn(objs, 2))
d <- data.frame(a = rep(pr[, 1], each = 12), b = rep(pr[, 2], each = 12))
d$judge <- sample(paste0("J", 1:5), nrow(d), TRUE)
d$win <- ifelse(runif(nrow(d)) < plogis(beta[d$a] - beta[d$b]), d$a, d$b)
judge_pair_surprise(btl(d, "a", "b", "win", judge = "judge"), "J1")
```

`judge_surprise` *Unexpected judgements of one judge*

Description

The paired-comparison counterpart of the kidmap. A judge has no ability to condition on, so the reference is the consensus object scale (the pooled locations). For the nominated judge, each object it met is given a standardised residual oriented to the object – how much more (`z > 0`, over-rated) or less (`z < 0`, under-rated) that judge favoured it than its consensus location predicts. A surprise is an object the judge treated against its standing: a strong object under-rated, or a weak object over-rated (residual opposite in sign to the location), beyond `flag_z` and seen at least `min_n` times.

Usage

```
judge_surprise(fit, judge, min_n = 2L, flag_z = 1.96)
```

Arguments

`fit`	A paired-comparison fit from `btl` with judges.
`judge`	The judge to profile (a value of the fit's judge column).
`min_n`	Objects met fewer times are shown but never flagged.
`flag_z`	Absolute residual at or beyond which a contrary judgement is flagged unexpected.

Value

A list of class `"rasch_btl_judge"`: `objects` (per object met: location, times met `n`, residual `z`, `surprise` flag and its `type`); `all_locations` (every object, for orientation); the `judge` and settings.

Examples

```
set.seed(1); objs <- LETTERS[1:6]; beta <- setNames(seq(-1.5, 1.5, len = 6), objs)
pr <- t(utils::combn(objs, 2))
d <- data.frame(a = rep(pr[, 1], each = 12), b = rep(pr[, 2], each = 12))
d$judge <- sample(paste0("J", 1:5), nrow(d), TRUE)
d$win <- ifelse(runif(nrow(d)) < plogis(beta[d$a] - beta[d$b]), d$a, d$b)
judge_surprise(btl(d, "a", "b", "win", judge = "judge"), "J1")
```

`lr_test` *Likelihood-ratio test of the partial credit against the rating scale model*

Description

A likelihood-ratio test in the tradition of Andersen (1973): an unrestricted (partial credit) analysis is compared with the rating re-parameterisation of the same model on the same data. Twice the difference in the pairwise conditional log-likelihoods is referred to a chi-square on the difference in the number of threshold parameters. A non-significant outcome supports adopting the simpler rating parameterisation.

Usage

```
lr_test(fit, maxit = 60, tol = 1e-08)
```

Arguments

`fit`	A `"PCM"` fit from `rasch` with equal maximum scores across items (the rating parameterisation requires them).
`maxit`, `tol`	Passed to the rating-scale refit.

Details

The likelihood here is the pairwise composite likelihood, not a full likelihood, and twice its difference is not chi-square distributed: each response enters every pair its item forms, so the raw statistic is inflated. Two statistics are therefore reported. `chisq` is the raw composite value with its naive `p`, the conventional display. The limiting law of the raw statistic is $\sum_j \lambda_j \chi^2_1$ (Kent 1982; Varin, Reid and Firth 2011) with λ_j the eigenvalues of $(C'H^{-1}C)^{-1}\,C'H^{-1}JH^{-1}C$ over the r constrained directions C (the part of the partial-credit threshold space outside the rating subspace), estimated from the same Godambe H and J matrices that supply the sandwich standard errors; matching the mean gives `chisq_adj` $= rW/\sum_j \lambda_j$ on r degrees of freedom. Use `p_adj` for inference; the naive `p` is severely anticonservative and kept only for comparability with conventional software displays.

Value

A list of class `"rasch_lr"`: raw `chisq`, `df`, `p` (the conventional display); adjusted `chisq_adj`, `p_adj`, and the eigenvalues `lambda`; the two log-likelihoods; and the rating-scale refit (`fit_rsm`).

References

Kent, J. T. (1982). Robust properties of likelihood ratio tests. Biometrika, 69, 19-27. Varin, C., Reid, N. and Firth, D. (2011). An overview of composite likelihood methods. Statistica Sinica, 21, 5-42.

Examples

```
set.seed(1)
tau <- c(-0.7, 0.7)
X <- sapply(seq(-1, 1, length.out = 6), function(d) vapply(rnorm(300),
  function(b) sample(0:2, 1, prob = item_moments(b, tau + d)$P), 0L))
colnames(X) <- paste0("Q", 1:6)
lr_test(rasch(X, model = "PCM"))
```

`pcml` *Estimate Rasch thresholds by pairwise conditional maximum likelihood*

Description

Maximises the pairwise conditional likelihood, in which the person parameter cancels within every item pair, by Newton-Raphson (Andrich and Luo 2003; Zwinderman 1995). The partial credit model estimates every threshold freely; the rating scale model constrains `tau_ik = delta_i + kappa_k` through the design matrix.

Usage

```
pcml(X, model = c("PCM", "RSM"), anchors = NULL, maxit = 60, tol = 1e-08)
```

Arguments

`X`	Persons-by-items integer score matrix (categories from 0). Missing values are handled by pairwise deletion, so linked booklet designs and random missingness estimate without imputation; the item-pair graph must be connected (some person answering items in both of any two blocks), otherwise relative locations between blocks are unidentified and the fit stops with an error naming the blocks – unless `anchors` fix an item in every block, the disjoint-form equating case.
`model`	`"PCM"` or `"RSM"`.
`anchors`	Optional anchor table for equating: a data frame with columns `item` (name or column index), `k`, and `tau` (the fixed value). A numeric `k` fixes that single threshold (individual anchoring); `k = NA` fixes the item's mean location at `tau` while its thresholds remain free (average anchoring). The remaining parameters are estimated on the anchored scale and no recentring is applied. PCM only.
`maxit`, `tol`	Newton-Raphson iteration cap and convergence tolerance.

Value

A list with the threshold table `thr` (columns `id`, `item`, `k`, `tau`, `se`, `anchored`, and `weak` – `TRUE` for a threshold adjacent to a category with fewer than 3 responses, whose estimate can run toward a boundary while the ridged covariance understates the error; its `se` is reported as `NA` and a note names the item and category), the threshold covariance matrix `cov_tau`, the pairwise conditional log-likelihood, the iteration count, a convergence flag, `notes`, and the max-score vector `m`.

Examples

```
set.seed(1)
d <- seq(-1.5, 1.5, length.out = 6)
X <- matrix(rbinom(400 * 6, 1, plogis(outer(rnorm(400), d, "-"))), 400, 6)
colnames(X) <- paste0("I", 1:6)
pcml(X)$thr
# anchor two items at fixed values (equating)
pcml(X, anchors = data.frame(item = c("I1", "I6"), k = 1, tau = c(-1.5, 1.5)))$thr
```

---

pcml_pc *Estimate Rasch thresholds via the Andrich principal-components reparameterisation*

Description

An optional alternative to pcml's free-threshold estimation, useful when some response categories are sparsely populated. Each item's thresholds are re-expressed as up to four orthogonal polynomial components in the category score: location, spread, skewness, and kurtosis (Andrich 1978, 1985; Pedler 1987). Location is always estimated; spread, skewness, and kurtosis are added in turn as an item's number of thresholds and n_components allow. Estimation uses the same pairwise conditional likelihood as pcml (Andrich and Luo 2003), so it inherits the same missing-data handling and sandwich standard errors. The component family stops at the quartic (kurtosis) term, so the reparameterisation is exact, matching pcml's free partial credit thresholds and log-likelihood, only while every item has at most 3 thresholds (4 categories); from 4 thresholds on pcml_pc is necessarily a reduced-rank smoothing of the thresholds to a polynomial trend across categories, however large n_components is set, trading flexibility for the stability that comes from pooling information across all of an item's categories – useful when a category has low or zero frequency.

Usage

```
pcml_pc(X, n_components = 4, maxit = 60, tol = 1e-08)
```

Arguments

X	Persons-by-items integer score matrix (categories from 0). Missing values are handled by pairwise deletion.
n_components	Maximum number of components per item: 1 (location only) up to 4 (location, spread, skewness, kurtosis; the highest derived by Pedler 1987). Capped per item at its own number of thresholds, and further wherever a component would be collinear with lower-order ones for that item's threshold count (kurtosis is unidentified, and dropped, at exactly 4 thresholds).
maxit, tol	Newton-Raphson iteration cap and convergence tolerance.

Value

A list with the threshold table `thr` (columns `id`, `item`, `k`, `tau`, `se`), the component table `components` (one row per item, with `location`, `spread`, `skewness`, `kurtosis` and their standard errors, `NA` where an item's rank does not support that component), the threshold covariance matrix `cov_tau`, the pairwise conditional log-likelihood, the iteration count, a convergence flag, and the max-score vector `m`.

Examples

```
set.seed(1)
d <- seq(-1.5, 1.5, length.out = 6)
X <- matrix(rbinom(400 * 6, 1, plogis(outer(rnorm(400), d, "-"))), 400, 6)
colnames(X) <- paste0("I", 1:6)
pcml_pc(X)$components
```

`person_extrapolated` *Person measures with extrapolated extreme scores*

Description

Extreme persons (zero or maximum raw score on their observed items) are excluded from calibration, but they cannot be left out of group comparisons; Andrich and Marais (2019, ch. 10) therefore describe an extrapolated measure for them, continuing the growth of the score-to-score differences so the last difference is the geometric mean of its neighbours (see `score_table`). This helper applies the same rule to the person table: for each missing-data pattern with extreme persons, the score-to-measure conversion over that pattern's items is extrapolated at its ends, and the extreme persons receive the extrapolated location with the standard error $1/\sqrt{I(\theta)}$ evaluated there. Non-extreme persons keep their estimates unchanged. The extrapolation continues the Warm (weighted likelihood) conversion, matching the package's person estimates.

Usage

```
person_extrapolated(fit)
```

Arguments

`fit`	A fitted object from `rasch` (equal discriminations; not EFRM).

Value

The fit's person table with two added columns, `theta_extrapolated` and `se_extrapolated`: equal to `theta` and `se` for non-extreme persons, extrapolated for extreme persons. Patterns with fewer than three interior scores cannot be extrapolated and keep their Warm values.

Examples

```
set.seed(1)
d <- seq(-2, 2, length.out = 8)
X <- matrix(rbinom(300 * 8, 1, plogis(outer(rnorm(300, 0, 2), d, "-"))), 300, 8)
colnames(X) <- paste0("I", 1:8)
fit <- rasch(X)
pe <- person_extrapolated(fit)
head(pe[pe$extreme, c("theta", "theta_extrapolated", "se", "se_extrapolated")])
```

`person_wle` *Warm's weighted likelihood estimates by raw score*

Description

Computes the weighted likelihood estimate (WLE) of person location for every possible raw score on a set of items, with standard errors. WLE estimates are finite at the extreme (zero and maximum) scores, unlike the maximum likelihood estimate.

Usage

```
person_wle(tau_list, disc = 1)
```

Arguments

`tau_list`	List of per-item threshold vectors.
`disc`	Common discrimination (frame unit) of the items; with a constant discrimination the raw score remains sufficient.

Value

A list with `theta` and `se`, each named by raw score.

Examples

```
person_wle(list(c(-1, 0), c(-0.5, 0.5), c(0, 1)))
```

plot_btl *Plot Bradley-Terry-Luce object locations*

Description

Caterpillar plot of the object locations with 95 per cent error bars, misfitting objects highlighted, in the package's house style.

Usage

```
plot_btl(fit, band = 2.5)
```

Arguments

fit	An object from btl.
band	Absolute fit-residual value beyond which an object is highlighted.

Value

Called for its plotting side effect; invisibly NULL.

Examples

```
set.seed(1)
beta <- c(A = -1, B = -0.3, C = 0.4, D = 0.9)
pairs <- t(combn(names(beta), 2))
d <- data.frame(a = rep(pairs[, 1], each = 30),
                b = rep(pairs[, 2], each = 30))
p <- plogis(beta[d$a] - beta[d$b])
d$win <- ifelse(runif(nrow(d)) < p, d$a, d$b)
plot_btl(btl(d, "a", "b", "win"))
```

plot_btl_categories *Plot graded-comparison category curves*

Description

For a graded paired-comparison fit, the probability of each response category as a function of the location difference beta_a - beta_b, with the symmetric threshold structure marked. The display is the paired-comparison counterpart of the category probability curves of a polytomous item.

Usage

```
plot_btl_categories(fit, grid = seq(-4, 4, 0.05))
```

Arguments

fit	A graded fit from btl (with response).
grid	Difference grid, in logits.

Value

Called for its plotting side effect; invisibly NULL.

Examples

```
set.seed(1)
beta <- c(A = -1, B = -0.3, C = 0.4, D = 0.9)
pr <- t(combn(names(beta), 2))
d <- data.frame(a = rep(pr[, 1], each = 40), b = rep(pr[, 2], each = 40))
P <- vapply(seq_len(nrow(d)), function(r)
  item_moments(beta[d$a[r]] - beta[d$b[r]], c(-1, 0, 1))$P, numeric(4))
d$grade <- apply(P, 2, function(p) sample(0:3, 1, prob = p))
plot_btl_categories(btl(d, "a", "b", response = "grade"))
```

---

plot_btl_dependence *Plot a within-judge dependence effect*

Description

The graphical display of a paired-comparison dependence effect, the counterpart of the DIF characteristic curve. For every comparison the departure of the observed response from what the object locations alone predict is taken, and the contribution of the *other* dependence effect is removed (a partial-residual display); these departures are then averaged in bins of the effect's own history covariate and plotted against it, with the model's fitted contribution overlaid. Observed points that rise with the covariate along the fitted line are the effect the coefficient summarises; a flat, scattered cloud means the estimate rests on little. Only the informative comparisons (a non-zero covariate: the two objects' histories differ) carry the effect, and the count in each bin is printed so a thin exposure tail is visible.

Usage

```
plot_btl_dependence(fit, effect = c("exposure", "carry_over"), bins = 6)
```

Arguments

fit	An object from btl fitted with an order column, so fit$dependence_data is present.
effect	Which effect to display: "exposure" (the seen-before advantage, the default) or "carry_over" (response dependence).
bins	Number of covariate bins for the continuous carry-over display; exposure takes its three natural levels (-1, 0, +1).

Value

Called for its plotting side effect; invisibly a data frame of the binned covariate value, observed and fitted departure, and bin count.

References

Davidson, R. R., & Beaver, R. J. (1977). On extending the Bradley-Terry model to incorporate within-pair order effects. *Biometrics*, 33(4), 693-702.

Examples

```
set.seed(1)
beta <- c(A = -0.8, B = -0.2, C = 0.4, D = 0.9)
pr <- t(combn(names(beta), 2))
d <- data.frame(a = rep(pr[, 1], each = 40), b = rep(pr[, 2], each = 40))
d$judge <- sample(sprintf("J%02d", 1:8), nrow(d), TRUE)
d <- d[order(d$judge), ]; d$t <- ave(seq_len(nrow(d)), d$judge, FUN = seq_along)
d$win <- ifelse(runif(nrow(d)) < plogis(beta[d$a] - beta[d$b]), d$a, d$b)
f <- btl(d, "a", "b", winner = "win", judge = "judge", order = "t")
plot_btl_dependence(f, "carry_over")
```

---

plot_btl_dim_map *Residual map of the leading paired-comparison bimension*

Description

Objects placed in the leading bimension plane. Reading round the swirl, an object sits "upstream" of those it over-beats relative to the fitted locations; a clear rotational arrangement is the second attribute, a formless blob near the origin is noise. Point size grows with the object's location on the primary scale.

Usage

```
plot_btl_dim_map(x, ...)
```

Arguments

x	A "rasch_btl_dim" object.
...	Unused.

Value

Called for its plotting side effect.

plot_btl_equate *Plot a paired-comparison equating comparison*

Description

Scatter of the two calibrations' common-object locations with the shifted identity line, per-object 95 per cent error bars, and a dotted guide band at the average pooled precision; objects that drift (after the multiplicity adjustment) are highlighted and labelled. The counterpart of `plot_equate` for Bradley-Terry-Luce scales.

Usage

```
plot_btl_equate(fit1, fit2, ...)
```

Arguments

`fit1`	A fitted object from `btl`.
`fit2`	A second `btl` fit, or a bank data frame with columns `object`, `location`, and optionally `se`.
`...`	Passed to `btl_equate` (e.g. `alpha`, `p_adjust`).

Value

Called for its plotting side effect; invisibly the `btl_equate` result.

Examples

```
set.seed(1)
beta <- setNames(seq(-2, 2, length.out = 8), paste0("O", 1:8))
sim <- function(objs) {
  pr <- t(utils::combn(objs, 2))
  d <- data.frame(a = rep(pr[, 1], each = 40), b = rep(pr[, 2], each = 40))
  d$win <- ifelse(runif(nrow(d)) < plogis(beta[d$a] - beta[d$b]), d$a, d$b)
  btl(d, "a", "b", "win")
}
plot_btl_equate(sim(paste0("O", 1:7)), sim(paste0("O", 2:8)))
```

plot_btl_icc *Plot an object characteristic curve*

Description

The paired-comparison counterpart of the item characteristic curve: the model expected response for one object as a function of opponent location (the win probability, or the expected graded response), with the observed mean response against each opponent overlaid at that opponent\'s estimated location. Observed points shrink in toward the curve as the model holds; an object of inconsistent quality shows points straying from it, exactly as a misfitting item does.

Usage

```
plot_btl_icc(fit, object, group = NULL, grid = NULL, min_n = 10)
```

Arguments

fit	An object from btl.
object	Object name.
group	Optional judge grouping for a DIF overlay: either one value per comparison row of fit$comparisons or a vector named by judge. Observed means are then drawn separately per group, as plot_icc draws person groups.
grid	Opponent-location grid, in logits.
min_n	An opponent's observed point is drawn only when the object (or, in the grouped display, that judge group) met it at least this many times; sparser pairs from incomplete or unbalanced designs are omitted.

Value

Called for its plotting side effect; invisibly the names of the opponents drawn (the ungrouped display), or NULL for the grouped display.

Examples

```
set.seed(1)
beta <- c(A = -1, B = -0.3, C = 0.4, D = 0.9)
pr <- t(combn(names(beta), 2))
d <- data.frame(a = rep(pr[, 1], each = 30), b = rep(pr[, 2], each = 30))
p <- plogis(beta[d$a] - beta[d$b])
d$win <- ifelse(runif(nrow(d)) < p, d$a, d$b)
plot_btl_icc(btl(d, "a", "b", winner = "win"), "C")
```

---

plot_btl_judge_map *Unexpected-judgement map for one judge (pair level)*

Description

The judge counterpart of the kidmap, drawn matchup by matchup. Each pair the judge met is a segment on the consensus location axis, spanning its two objects, positioned horizontally by how surprising the verdict was: at zero (the dashed line, inside the shaded band) the stronger object won as its lead predicts; to the left the judge backed the underdog. A filled dot marks the object the judge's verdict favoured, hollow the other – so an upset is a red segment on the left with its filled dot at the lower end. The rug marks every object's location.

Usage

```
plot_btl_judge_map(fit, judge, min_n = 1L, flag_z = 1.96, ...)
```

Arguments

fit	A paired-comparison fit from btl with judges.
judge	The judge to map.
min_n, flag_z	Passed to judge_pair_surprise.
...	Unused.

Value

Called for its plotting side effect; invisibly the rasch_btl_judge_pairs object.

---

plot_btl_scree *Scree of paired-comparison residual bimensions*

Description

Bimension strengths against the model-simulated noise reference (its mean and 95th percentile band). A leading bar clearing the band is structured residual dependence – a likely second attribute.

Usage

```
plot_btl_scree(x, ...)
```

Arguments

x	A "rasch_btl_dim" object.
...	Unused.

Value

Called for its plotting side effect.

---

plot_btl_targeting *Targeting plot for a paired-comparison design*

Description

The paired-comparison counterpart of a test-information display. Every object is a dot at its location (x) and its design information (y, the pooled Fisher information of the comparisons it took part in), the dot sized by how many comparisons that is. A reference curve, read on the right axis, traces the information a single *new* comparison would carry against an opponent at each location – anchored at the centre of the scale, so it peaks at gap zero and falls away with the gap. The curve is the visual explanation of why an adaptive design chases near-neighbour contests: information is bought most cheaply where the two objects are close, and a well-targeted design lifts the low dots by pairing their objects against opponents near them.

Usage

```
plot_btl_targeting(fit, grid = NULL)
```

Arguments

fit	A paired-comparison fit from btl.
grid	Optional location grid for the reference curve.

Value

Called for its plotting side effect; invisibly NULL.

See Also

btl_information, btl_next_pairs

Examples

```
set.seed(1)
beta <- c(A = -1, B = -0.3, C = 0.4, D = 0.9)
pr <- t(combn(names(beta), 2))
d <- data.frame(a = rep(pr[, 1], each = 30), b = rep(pr[, 2], each = 30))
d$win <- ifelse(runif(nrow(d)) < plogis(beta[d$a] - beta[d$b]), d$a, d$b)
plot_btl_targeting(btl(d, "a", "b", "win"))
```

---

plot_btl_transitivity *Consistency plot for paired-comparison transitivity*

Description

With by = "judge" (the default when judges exist), plots each judge's consistency – one minus the circular-triad rate over the chance rate – as a dot against the chance line at zero: the individual-judge lens, a judge-fit analogue. With by = "object", plots each object's circular-triad involvement instead: the structural lens, showing which objects sit in the most contradictions.

Usage

```
plot_btl_transitivity(x, by = c("auto", "judge", "object"), ...)
```

Arguments

x	A "rasch_btl_transitivity" object.
by	"auto" (judges if present, else objects), "judge", or "object".
...	Unused.

Value

Called for its plotting side effect.

plot_btl_units *Plot the frame units of a paired-comparison EFRM fit*

Description

Caterpillar plot of the estimated units on the log scale: one row per panel unit `phi_g` and one per set unit `alpha_s`, with 95 per cent intervals, the reference (unit one) marked, mirroring `plot_frames` in the package's house style.

Usage

```
plot_btl_units(fit)
```

Arguments

`fit`	A fitted object from `btl_efrm`.

Value

Called for its plotting side effect; invisibly `NULL`.

Examples

```
# see ?btl_efrm for a complete simulated example
```

plot_catfreq *Plot category frequencies*

Description

Observed response distribution over the categories of one item.

Usage

```
plot_catfreq(fit, item)
```

Arguments

`fit`	A fitted object from `rasch`.
`item`	Item name or column index.

Value

Called for its plotting side effect; invisibly `NULL`.

Examples

```
set.seed(1)
simP <- function(th, t) { x <- 0:length(t); p <- exp(x * th - c(0, cumsum(t))); p / sum(p) }
th <- rnorm(400)
X <- sapply(1:4, function(i) sapply(th, function(t) sample(0:3, 1, prob = simP(t, c(-1, 0, 1)))))
colnames(X) <- sprintf("P%02d", 1:4)
plot_catfreq(rasch(X), "P01")
```

`plot_ccc` *Plot category probability curves*

Description

Plot category probability curves

Usage

```
plot_ccc(
  fit,
  item,
  grid = seq(-6, 6, 0.05),
  observed = FALSE,
  n_groups = fit$n_groups
)
```

Arguments

`fit`	A fitted object from `rasch`.
`item`	Item name or column index.
`grid`	Logit grid over which to draw the curves.
`observed`	Overlay the observed category proportions per class interval (Andrich and Marais 2019, ch. 20).
`n_groups`	Class intervals for the observed points.

Value

Called for its plotting side effect; invisibly `NULL`.

Examples

```
set.seed(1)
simP <- function(th, t) { x <- 0:length(t); p <- exp(x * th - c(0, cumsum(t))); p / sum(p) }
th <- rnorm(400)
X <- sapply(1:4, function(i) sapply(th, function(t) sample(0:3, 1, prob = simP(t, c(-1, 0, 1)))))
colnames(X) <- sprintf("P%02d", 1:4)
plot_ccc(rasch(X), "P01", observed = TRUE)
```

plot_distractors *Plot multiple-choice option curves*

Description

The proportion choosing each response option across class intervals of the rest measure (the person estimate from the other items), with the keyed option drawn solid and bold. The keyed option should rise with the trait and every distractor should fall; a rising distractor is the graphical signature of a miskey or an ambiguous option.

Usage

```
plot_distractors(fit, item, n_groups = fit$n_groups)
```

Arguments

fit	A fitted object from rasch run with a key.
item	Keyed item name.
n_groups	Number of class intervals.

Value

Called for its plotting side effect; invisibly NULL.

Examples

```
set.seed(1); Np <- 400
th <- rnorm(Np)
raw <- sapply(seq(-1, 1, length.out = 6), function(d) {
  ok <- rbinom(Np, 1, plogis(th - d))
  ifelse(ok == 1, "A", sample(c("B", "C", "D"), Np, replace = TRUE))
})
colnames(raw) <- paste0("M", 1:6)
fit <- rasch(raw, key = setNames(rep("A", 6), colnames(raw)))
plot_distractors(fit, "M3")
```

plot_equate *Plot a test-equating comparison*

Description

Scatter of the two calibrations' common-item locations with the shifted identity line and per-item 95 per cent bands; drifting items (BH-adjusted) are highlighted and labelled.

Usage

```
plot_equate(fit, reference, shift = c("mean", "none"))
```

Arguments

fit	A fitted object from rasch.
reference	A second rasch fit, or a data frame with columns item, location, and optionally se.
shift	Passed to equate_tests.

Value

Called for its plotting side effect; invisibly the equate_tests result.

Examples

```
set.seed(1); d <- seq(-1.5, 1.5, length.out = 8)
mk <- function() {
  X <- matrix(rbinom(400 * 8, 1, plogis(outer(rnorm(400), d, "-"))), 400, 8)
  colnames(X) <- paste0("I", 1:8); rasch(X)
}
plot_equate(mk(), mk())
```

---

plot_facets *Plot facet severities*

Description

Caterpillar plot of the severity of each level of a facet from a many-facet analysis, with 95 per cent error bars; levels with pooled fit residuals beyond the band are highlighted.

Usage

```
plot_facets(fit, facet = NULL, band = 2.5)
```

Arguments

fit	A fitted object from rasch_mfrm.
facet	Facet name; defaults to the first facet.
band	Fit residual band beyond which a level is highlighted.

Value

Called for its plotting side effect; invisibly NULL.

Examples

```
set.seed(1)
simP <- function(th, tau) { x <- 0:length(tau); p <- exp(x * th - c(0, cumsum(tau))); p / sum(p) }
persons <- sprintf("P%03d", 1:120); raters <- paste0("R", 1:4)
th <- setNames(rnorm(120, 0, 1.3), persons)
rho <- setNames(c(-0.6, -0.2, 0.2, 0.6), raters)
tau <- list(A = c(-1, 1), B = c(-0.5, 1.2), C = c(-1.2, 0.4))
d <- expand.grid(person = persons, item = names(tau), rater = raters,
                 stringsAsFactors = FALSE)
d$score <- mapply(function(p, i, r)
  sample(0:2, 1, prob = simP(th[p], tau[[i]] + rho[r])), d$person, d$item, d$rater)
plot_facets(rasch_mfrm(d, "person", "item", "score", facets = "rater"))
```

`plot_frames` *Plot frame units*

Description

Caterpillar plot of the frame units `rho_sg` = `alpha_s` `phi_g` on the log scale, grouped by item set and coloured by person group, with 95 per cent error bars; frames with pooled fit residuals beyond the band are highlighted.

Usage

```
plot_frames(fit, band = 2.5)
```

Arguments

`fit`	A fitted object from `rasch_efrm`.
band	Pooled fit residual band beyond which a frame is highlighted.

Value

Called for its plotting side effect; invisibly `NULL`.

Examples

```
# see ?rasch_efrm for a complete simulated example
```

plot_guttman *Plot the Guttman scalogram*

Description

Displays the Guttman-ordered response matrix as a heatmap (dark for high categories), with persons sorted by location down the rows and items by location across the columns. The coefficient of reproducibility is shown in the subtitle.

Usage

```
plot_guttman(fit, max_persons = 80)
```

Arguments

fit	A fitted object from rasch.
max_persons	Persons are thinned to at most this many evenly spaced rows for legibility on large samples.

Value

Called for its plotting side effect; invisibly NULL.

Examples

```
set.seed(1)
d <- seq(-2, 2, length.out = 6)
X <- matrix(rbinom(200 * 6, 1, plogis(outer(rnorm(200), d, "-"))), 200, 6)
colnames(X) <- paste0("I", 1:6)
plot_guttman(rasch(X))
```

plot_icc *Plot an item characteristic curve*

Description

Draws the model expected-score curve with observed class-interval means overlaid. With group supplied, observed means are drawn separately per group, the conventional graphical DIF display.

Usage

```
plot_icc(fit, item, group = NULL, n_groups = NULL, grid = seq(-5, 5, 0.05))
```

Arguments

fit	A fitted object from rasch.
item	Item name or column index.
group	Optional person grouping vector, or one or more names of factors nominated in the fit, for a DIF overlay; several names give the factor-combination cells (the factorial display).
n_groups	Number of class intervals for the observed means; by default the fit's own count, or – with a group overlay – a count adapted to keep the smallest group's interval cells adequately filled.
grid	Logit grid over which to draw the model curve.

Value

Called for its plotting side effect; invisibly NULL.

Examples

```
set.seed(1)
d <- seq(-2, 2, length.out = 6)
X <- matrix(rbinom(400 * 6, 1, plogis(outer(rnorm(400), d, "-"))), 400, 6)
colnames(X) <- sprintf("I%02d", 1:6)
plot_icc(rasch(X), "I03")
```

---

plot_icc_frames *Plot an item's characteristic curves across frames*

Description

The signature display of the extended frame of reference model: the model expected-score curve of one underlying item drawn once per person group (curves fan with the group units), with observed class-interval means per group overlaid.

Usage

```
plot_icc_frames(fit, item, n_groups = fit$n_groups, grid = seq(-5, 5, 0.05))
```

Arguments

fit	A fitted object from rasch_efrm.
item	Underlying item name.
n_groups	Number of class intervals for the observed means.
grid	Logit grid.

Value

Called for its plotting side effect; invisibly NULL.

Examples

```
# see ?rasch_efrm for a complete simulated example
```

`plot_item_map` *Plot the item map (location against fit residual)*

Description

Items plotted by location and fit residual, with the conventional acceptance band at +/- 2.5 and misfitting items labelled.

Usage

```
plot_item_map(fit, band = 2.5)
```

Arguments

`fit`	A fitted object from `rasch`.
`band`	Fit residual acceptance band.

Value

Called for its plotting side effect; invisibly `NULL`.

Examples

```
set.seed(1)
d <- seq(-2, 2, length.out = 6)
X <- matrix(rbinom(400 * 6, 1, plogis(outer(rnorm(400), d, "-"))), 400, 6)
colnames(X) <- paste0("I", 1:6)
plot_item_map(rasch(X))
```

`plot_kidmap` *Plot a kidmap*

Description

The person diagnostic map (Wright, Mead and Ludlow 1980): item thresholds the person achieved to the right of a vertical logit axis and thresholds not achieved to the left, with the person's location drawn as a dashed line inside its confidence band. Achieved thresholds above the band (unexpected successes) and unachieved thresholds below it (unexpected failures) are highlighted; a clean response pattern shows achieved thresholds below the band and unachieved ones above it.

Usage

```
plot_kidmap(
  fit,
  person,
  level = 0.95,
  bins = 35,
  xlim = NULL,
  cex_labels = 0.8
)
```

Arguments

fit	A fitted object from rasch.
person	Row number of the person, or an ID matching fit$person$id.
level	Confidence level of the band around the person location used to mark unexpected responses.
bins	Number of vertical bins used to stack the threshold labels.
xlim	Optional logit range; thresholds outside it are omitted.
cex_labels	Character expansion for the threshold labels.

Value

Called for its plotting side effect; invisibly NULL.

References

Wright, B. D., Mead, R. J., & Ludlow, L. H. (1980). *KIDMAP: person-by-item interaction mapping* (Research Memorandum No. 29). Chicago: University of Chicago, MESA Psychometric Laboratory.

Examples

```
set.seed(1)
d <- seq(-2, 2, length.out = 12)
X <- matrix(rbinom(300 * 12, 1, plogis(outer(rnorm(300), d, "-"))), 300, 12)
colnames(X) <- paste0("I", 1:12)
plot_kidmap(rasch(X), person = 1)
```

---

plot_pca *Plot residual principal-component loadings*

Description

Residual-component loadings against item location; opposing clusters at top and bottom suggest a further dimension. Any leading component may be shown, not only the first.

Usage

```
plot_pca(fit, component = 1)
```

Arguments

fit	A fitted object from rasch.
component	Which residual principal component to plot (default the first component).

Value

Called for its plotting side effect; invisibly NULL.

Examples

```
set.seed(1)
d <- seq(-2, 2, length.out = 6)
X <- matrix(rbinom(400 * 6, 1, plogis(outer(rnorm(400), d, "-"))), 400, 6)
colnames(X) <- paste0("I", 1:6)
plot_pca(rasch(X))
```

---

plot_pca_biplot *Biplot of the first two residual components*

Description

Item loadings on the first two residual principal components – the pair that usually carries any interpretable second dimension – plotted against one another on equal (isometric) axes. Items far from the origin with opposing signs on PC1 mark a possible contrast, and PC2 separates them further. Point colour follows the sign of the PC1 loading, the split the unidimensionality t-test uses by default.

Usage

```
plot_pca_biplot(fit)
```

Arguments

fit	A fitted object from rasch.

Value

Called for its plotting side effect; invisibly NULL.

Examples

```
set.seed(1)
d <- seq(-2, 2, length.out = 8)
X <- matrix(rbinom(500 * 8, 1, plogis(outer(rnorm(500), d, "-"))), 500, 8)
colnames(X) <- paste0("I", 1:8)
plot_pca_biplot(rasch(X))
```

plot_pcc *Plot a person characteristic curve*

Description

The person characteristic curve: the probability of success as a function of item location at the person's estimated measure, with the person's observed responses overlaid, grouped into item-difficulty intervals (proportion of maximum score per interval). Erratic responding (for example lucky guessing on hard items by a low-proficiency person) shows as observed points far from the curve, complementing the person fit residual.

Usage

```
plot_pcc(fit, person, n_groups = 5, grid = seq(-5, 5, 0.05))
```

Arguments

fit	A fitted object from rasch.
person	Row number of the person, or an ID matching fit$person$id.
n_groups	Number of item-difficulty intervals for the observed points (capped by the number of observed items).
grid	Item-location grid over which to draw the curve.

Value

Called for its plotting side effect; invisibly NULL.

Examples

```
set.seed(1)
d <- seq(-2, 2, length.out = 12)
X <- matrix(rbinom(300 * 12, 1, plogis(outer(rnorm(300), d, "-"))), 300, 12)
colnames(X) <- paste0("I", 1:12)
plot_pcc(rasch(X), person = 1)
```

plot_person_fit *Plot person fit*

Description

Person locations against person fit residuals with the +/- 2.5 band; persons beyond the band respond erratically (positive) or too deterministically (negative).

Usage

```
plot_person_fit(fit, band = 2.5)
```

Arguments

fit	A fitted object from rasch.
band	Fit residual acceptance band.

Value

Called for its plotting side effect; invisibly NULL.

Examples

```
set.seed(1)
d <- seq(-2, 2, length.out = 6)
X <- matrix(rbinom(400 * 6, 1, plogis(outer(rnorm(400), d, "-"))), 400, 6)
colnames(X) <- paste0("I", 1:6)
plot_person_fit(rasch(X))
```

---

plot_pimap *Plot the person-item threshold distribution*

Description

The targeting display: the person location distribution above the axis and the item threshold distribution mirrored below it, on a shared logit scale.

Usage

```
plot_pimap(fit, bins = 35, xlim = NULL)
```

Arguments

fit	A fitted object from rasch.
bins	Number of histogram bins.
xlim	Optional logit range for the shared scale; persons and thresholds outside it are omitted.

Value

Called for its plotting side effect; invisibly NULL.

Examples

```
set.seed(1)
d <- seq(-2, 2, length.out = 6)
X <- matrix(rbinom(400 * 6, 1, plogis(outer(rnorm(400), d, "-"))), 400, 6)
colnames(X) <- paste0("I", 1:6)
plot_pimap(rasch(X))
```

plot_recovery *Recovery scatter of planted against recovered parameters*

Description

One true-versus-estimated panel per parameter type, with the identity line and the correlation and RMSE.

Usage

```
plot_recovery(x, ...)
```

Arguments

x	A "rasch_recovery" object.
...	Unused.

Value

Called for its plotting side effect.

plot_resid_cor *Plot the residual-correlation heatmap*

Description

Only the lower triangle is drawn – the matrix is symmetric, so each pair is shown once. With `stat = "q3star"` (the default) cells are coloured by Q3* – each pair's residual correlation minus the average off-diagonal correlation – so white marks the value expected under local independence and warm colour marks dependence; with `stat = "q3"` the raw residual correlation is coloured, white at zero. The scale saturates at `cap` rather than the +/-1 of an ordinary correlation: a residual correlation seldom reaches even 0.5 under a fitting model (the conventional flag is Q3* above 0.2; Christensen, Makransky and Horton 2017), so the colour is spent where the values actually discriminate.

Usage

```
plot_resid_cor(fit, stat = c("q3star", "q3"), cap = 0.5)
```

Arguments

fit	A fitted object from rasch.
stat	Which statistic to colour: "q3star" (adjusted Q3, the default) or "q3" (the raw residual correlation).
cap	Value at which the colour saturates (default 0.5).

Value

Called for its plotting side effect; invisibly `NULL`.

Examples

```
set.seed(1)
d <- seq(-2, 2, length.out = 6)
X <- matrix(rbinom(400 * 6, 1, plogis(outer(rnorm(400), d, "-"))), 400, 6)
colnames(X) <- paste0("I", 1:6)
plot_resid_cor(rasch(X))
```

`plot_resid_dist` *Plot the fit residual distribution*

Description

A histogram of the item or person fit residuals – the log-transformed statistic or its untransformed natural form – against the standard normal density they should approximate under fit (Andrich and Marais 2019, ch. 15). The natural residual is visibly skewed (that is why the log transform is reported); both are available.

Usage

```
plot_resid_dist(
  fit,
  what = c("items", "persons"),
  statistic = c("fit_resid", "natural"),
  bins = 25
)
```

Arguments

`fit`	A fitted object from `rasch`.
`what`	`"items"` or `"persons"`.
`statistic`	`"fit_resid"` (log-transformed, default) or `"natural"`.
`bins`	Number of histogram bins.

Value

Called for its plotting side effect; invisibly `NULL`.

Examples

```
set.seed(1)
d <- seq(-2, 2, length.out = 10)
X <- matrix(rbinom(400 * 10, 1, plogis(outer(rnorm(400), d, "-"))), 400, 10)
colnames(X) <- paste0("I", 1:10)
plot_resid_dist(rasch(X), what = "persons")
```

`plot_scree` *Scree plot of the residual components with parallel analysis*

Description

Eigenvalues of the residual correlation matrix for the leading components, with a model-simulated parallel-analysis reference: responses are simulated from the calibrated model (observed missingness kept), every person is re-estimated, and the residual eigenvalues recomputed. Because estimating the person locations couples the residuals within a person, this reference sits above the classical random-normal one and is calibrated under the fitted model (Raiche 2005; Chou & Wang 2010). Observed eigenvalues above the reference suggest structure beyond what the model itself produces.

Usage

```
plot_scree(fit, n_components = 10, parallel = TRUE, reps = 20)
```

Arguments

`fit`	A fitted object from `rasch`.
`n_components`	Number of leading components to display.
`parallel`	Draw the parallel-analysis reference line.
`reps`	Model-simulated replicates for the reference.

Value

Called for its plotting side effect; invisibly the eigen table.

References

Raiche, G. (2005). Critical eigenvalue sizes in standardized residual principal components analysis. *Rasch Measurement Transactions*, 19(1), 1012.

Chou, Y.-T., & Wang, W.-C. (2010). Checking dimensionality in item response models with principal component analysis on standardized residuals. *Educational and Psychological Measurement*, 70(5), 717-731.

Examples

```
set.seed(1)
d <- seq(-2, 2, length.out = 8)
X <- matrix(rbinom(500 * 8, 1, plogis(outer(rnorm(500), d, "-"))), 500, 8)
colnames(X) <- paste0("I", 1:8)
plot_scree(rasch(X))
```

plot_tcc *Plot the test characteristic curve*

Description

Expected total score against person location for the whole instrument.

Usage

```
plot_tcc(fit, grid = seq(-6, 6, 0.05))
```

Arguments

fit	A fitted object from rasch.
grid	Logit grid.

Value

Called for its plotting side effect; invisibly NULL.

Examples

```
set.seed(1)
d <- seq(-2, 2, length.out = 6)
X <- matrix(rbinom(400 * 6, 1, plogis(outer(rnorm(400), d, "-"))), 400, 6)
colnames(X) <- paste0("I", 1:6)
plot_tcc(rasch(X))
```

plot_threshold_map *Plot the threshold map*

Description

Each item's threshold locations on a common logit scale, ordered by item location, with disordered thresholds highlighted.

Usage

```
plot_threshold_map(fit, order_by_location = TRUE)
```

Arguments

fit	A fitted object from rasch.
order_by_location	Order items by their location (the default) rather than their original sequence.

Value

Called for its plotting side effect; invisibly NULL.

Examples

```
set.seed(1)
d <- seq(-2, 2, length.out = 6)
X <- matrix(rbinom(400 * 6, 1, plogis(outer(rnorm(400), d, "-"))), 400, 6)
colnames(X) <- paste0("I", 1:6)
plot_threshold_map(rasch(X))
```

---

plot_threshold_prob *Plot threshold probability curves*

Description

Conditional probability of success at each threshold, P(X = k | X = k - 1 or k), a logistic ogive crossing 0.5 at the threshold location. Disordered thresholds are immediately visible as out-of-sequence ogives. With observed = TRUE the observed conditional proportions per class interval are overlaid, the direct check on whether each threshold discriminates (and hence whether collapsing categories could ever be justified; Andrich and Marais 2019, ch. 22).

Usage

```
plot_threshold_prob(
  fit,
  item,
  grid = seq(-6, 6, 0.05),
  observed = FALSE,
  n_groups = fit$n_groups
)
```

Arguments

fit	A fitted object from rasch.
item	Item name or column index.
grid	Logit grid over which to draw the curves.
observed	Overlay the observed conditional threshold proportions per class interval.
n_groups	Class intervals for the observed points.

Value

Called for its plotting side effect; invisibly NULL.

Examples

```
set.seed(1)
simP <- function(th, t) { x <- 0:length(t); p <- exp(x * th - c(0, cumsum(t))); p / sum(p) }
th <- rnorm(400)
X <- sapply(1:4, function(i) sapply(th, function(t) sample(0:3, 1, prob = simP(t, c(-1, 0, 1)))))
colnames(X) <- sprintf("P%02d", 1:4)
plot_threshold_prob(rasch(X), "P01")
```

`plot_tif` *Plot the test information function*

Description

Test information across the logit scale with the standard error of measurement overlaid on a second axis.

Usage

```
plot_tif(fit, grid = seq(-6, 6, 0.05))
```

Arguments

`fit`	A fitted object from `rasch`.
`grid`	Logit grid.

Value

Called for its plotting side effect; invisibly `NULL`.

Examples

```
set.seed(1)
d <- seq(-2, 2, length.out = 6)
X <- matrix(rbinom(400 * 6, 1, plogis(outer(rnorm(400), d, "-"))), 400, 6)
colnames(X) <- paste0("I", 1:6)
plot_tif(rasch(X))
```

`plot_wright` *Plot a Wright map*

Description

The conventional vertical person-item map (Wright and Stone 1979): the person distribution to the left of a shared logit axis and the item thresholds, labelled by item (and threshold number for polytomous items), stacked to its right.

Usage

```
plot_wright(fit, bins = 35, xlim = NULL, cex_labels = 0.8)
```

Arguments

`fit`	A fitted object from `rasch`.
`bins`	Number of bins for the person distribution and the threshold label rows.
`xlim`	Optional logit range for the shared scale; persons and thresholds outside it are omitted.
`cex_labels`	Character expansion for the threshold labels.

Value

Called for its plotting side effect; invisibly `NULL`.

References

Wright, B. D., & Stone, M. H. (1979). *Best Test Design*. Chicago: MESA Press.

Examples

```
set.seed(1)
d <- seq(-2, 2, length.out = 6)
X <- matrix(rbinom(400 * 6, 1, plogis(outer(rnorm(400), d, "-"))), 400, 6)
colnames(X) <- paste0("I", 1:6)
plot_wright(rasch(X))
```

rack_data *Reshape repeated measurements for racked or stacked analysis*

Description

Repeated measurements (the same persons and items at two or more time points) enter a Rasch analysis in one of two designs (Andrich & Marais 2019, ch. 26). *Racking* keeps one row per person and duplicates the items per time point (columns `item@time`), so change over time shows in the item estimates. *Stacking* keeps one column per item and duplicates the persons per time point (rows `person@time`), so change shows in the person estimates and DIF of items over time can be examined with `time` as a person factor.

Usage

```
rack_data(data, person, time, items)

stack_data(data, person, time, items)
```

Arguments

`data`	A long data frame with one measurement per row.
`person`, `time`	Names of the person and time-point columns.
`items`	Character vector naming the item columns.

Value

`rack_data`: a wide data frame with one row per person and `length(items) * n_times` item columns. `stack_data`: a data frame with one row per person-time (`id` column), the original item columns, and `time` as a factor column for DIF analysis.

Examples

```
d <- data.frame(pid = rep(1:100, 2), t = rep(1:2, each = 100),
                Q1 = rbinom(200, 1, 0.6), Q2 = rbinom(200, 1, 0.5))
racked <- rack_data(d, person = "pid", time = "t", items = c("Q1", "Q2"))
names(racked)
stacked <- stack_data(d, person = "pid", time = "t", items = c("Q1", "Q2"))
head(stacked)
```

rasch *Fit and diagnose a Rasch model by pairwise conditional estimation*

Description

Runs a complete Rasch analysis: Andrich and Luo pairwise conditional maximum likelihood item estimation (see pcml), Warm weighted likelihood person estimates per missing-data pattern, item and person fit residuals (the log-of-mean-square statistic of Andrich and Marais 2019, ch. 23, with its untransformed natural form and degrees of freedom), infit and outfit, the item-trait interaction chi-square and the class-interval ANOVA item-fit F, the person separation index with and without extremes, Cronbach's alpha, targeting, threshold diagnostics, and the score-to-measure table.

Usage

```
rasch(
  data,
  model = c("PCM", "RSM"),
  id = NULL,
  factors = NULL,
  items = NULL,
  n_groups = NULL,
  adjust_N = NA,
  anchors = NULL,
  na_codes = -1,
  key = NULL,
  pc_components = NULL,
  maxit = 60,
  tol = 1e-08
)
```

Arguments

data	Persons-by-items integer score matrix (categories from 0), or a data frame also containing ID and person-factor columns. Missing values are allowed.
model	Either "PCM" (partial credit) or "RSM" (rating scale).
id	Optional name of an ID column in data, or a vector of IDs; carried through to the person estimates.
factors	Optional character vector of person-factor column names in data (for DIF analysis), or a data frame of factors.
items	Optional character vector naming the item columns; by default every column not named in id or factors.
n_groups	Number of class intervals for the item-trait chi-square and ANOVA item fit. The default NULL applies the rule of Andrich and Marais (2019, ch. 15): as many intervals of at least 50 non-extreme persons as the sample allows, at most 10, at least 2. The resolved value is stored in fit$n_groups.

`adjust_N` Optional reference sample size; if supplied, item-trait chi-squares are rescaled to this size (a sample-size adjustment for the sensitivity of the chi-square to large samples). The scaling is proportional and global – every item's chi-square is multiplied by `adjust_N` over the number of classified persons – so an item answered by a subset of persons keeps its proportionally smaller share of the notional sample rather than being inflated to the full `adjust_N`.

`anchors` Optional anchor table for equating: a data frame with columns `item`, `k`, and `tau` fixing nominated thresholds at known values; see `pcml`. With anchors in place the scale origin comes from the anchors, so person measures are directly comparable across separately analysed datasets.

`na_codes` Values to read as missing. Defaults to `-1`, the conventional missing-response code; any negative score is also treated as missing, since valid category scores start at zero.

`key` Optional multiple-choice scoring key, in any of three forms. (1) A named vector or data frame with columns `item` and `key` naming each item's correct option: scored 0/1 (case-insensitive after trimming; blanks become missing). (2) Double keying: several correct options separated by `"/"` (for example `"A/C"`), all scoring 1. (3) Polytomous option scoring (Andrich and Styles 2011): a data frame with columns `item`, `option`, and `score` assigning an integer score to every credited option (unlisted options score 0), so informative distractors receive partial credit and the item is fitted as polytomous; see `distractor_rescore` for an evidence-based proposal. Raw responses are retained in `fit$mc` for `distractor_analysis` and `plot_distractors`.

`pc_components` `NULL` (default) estimates every PCM threshold freely. An integer 1 to 4 instead estimates each item's thresholds through the Andrich principal-components reparameterisation (see `pcml_pc`): 1 = location only, 2 = + spread (the dispersion model of Andrich 1982), 3 = + skewness, 4 = + kurtosis (the full principal-components model; Pedler 1987). Useful when some categories are sparsely populated; the component estimates are returned in `fit$est$components`. PCM only, and not combinable with anchors.

`maxit`, `tol` Newton-Raphson iteration cap and convergence tolerance of the pairwise conditional estimation.

Details

The fit residual follows Andrich and Marais (2019, ch. 23) exactly: standardised residuals are squared and summed over each item's (person's) observed cells among non-extreme persons, compared with the summed cell degrees of freedom (the model-testing degrees of freedom, cells minus estimated parameters, apportioned equally over cells), and symmetrised by the log-of-mean-square transform $f(\ln Y^2 - \ln f)/\sqrt{V[Y^2]}$ with model-based variance $V[Y^2] = \sum(C_4/V^2 - 1)$. Values are approximately N(0,1) under fit; the conventional flagging value is 2.5 (Andrich and Marais 2019, ch. 15). Negative values indicate over-discrimination (Guttman-like responses), positive values under-discrimination.

A calibration note that applies to the whole test-of-fit suite: the item-trait chi-square evaluates unconditional residuals at estimated person measures and refers the sum of correlated item statistics to a chi-square with summed degrees of freedom, following the convention of Andrich and Marais

(2019); the class-interval ANOVA F and the Wilson-Hilferty-style transformations are approximations of the same kind. Null simulation with this package shows rejection rates near but not exactly at the nominal level (conservative in the settings examined), and the direction can vary with design. These statistics are therefore approximate, convention-faithful diagnostics for ordering and flagging misfit – not exactly calibrated hypothesis tests; where exact calibration matters, use the simulation tools (sim_replicate) to build parametric-bootstrap reference distributions for the observed design.

Value

An object of class "rasch": a list with the item summary (items), thresholds (with standard errors), the person table (person, including ID and factors), the score table, residuals, reliability (psi, psi_noext, the item separation index isi, alpha), targeting, item-trait statistics (item_trait, item_anova), the summary distribution block (summary_stats: location and fit residual mean/SD/skewness/kurtosis, fit-location correlations, and the cell degrees-of-freedom factor), threshold diagnostics, and estimation details (est).

Examples

```
set.seed(1)
d <- seq(-2, 2, length.out = 8)
X <- matrix(rbinom(500 * 8, 1, plogis(outer(rnorm(500), d, "-"))), 500, 8)
colnames(X) <- paste0("I", 1:8)
fit <- rasch(X, model = "PCM")
fit$items
fit$psi$PSI
```

---

rasch_efrm *Fit the extended frame of reference model*

Description

Estimates Humphry's extended frame of reference model, in which the unit of the latent scale differs across frames (item-set by person-group cells): the response model is P(X = x) prop exp(rho_sg (x theta - sum delta)) with rho_sg = alpha_s phi_g. Within frames the partial credit model holds in the frame's natural unit, so item thresholds and the person group units phi are estimated by within-frame pairwise conditional maximum likelihood (the person parameter cancels; Andrich and Luo 2003), jointly across frames through the sets shared by several groups. Item-set units alpha and set locations are then estimated from persons common to pairs of sets, using error-corrected true-score variances, and reconciled over the linking graph by weighted least squares. Everything is reported in a common arbitrary unit, and the returned object is also a full rasch fit at the item-by-group level, so the package's diagnostic tables and plots apply.

Usage

```
rasch_efrm(
  data,
  item_sets,
```

```
  groups,
  id = NULL,
  factors = NULL,
  items = NULL,
  n_groups = NULL,
  adjust_N = NA,
  na_codes = -1,
  maxit = 50,
  tol = 1e-07,
  min_link_persons = 30,
  se_method = c("hybrid", "bootstrap"),
  boot_reps = NULL
)
```

Arguments

`data`	Persons-by-items data (matrix or data frame, like `rasch`), plus a person-group column.
`item_sets`	A named list mapping set names to item-column names, or a named character vector mapping item names to set names. Items not mentioned form their own set `"(rest)"` when a list is given.
`groups`	Name of the person-group column in `data`, or a vector with one entry per person.
`id`, `factors`, `items`, `n_groups`, `adjust_N`, `na_codes`	As in `rasch`.
`maxit`, `tol`	Outer iteration cap and convergence tolerance of the bilinear pairwise stage.
`min_link_persons`	Minimum number of common persons required for a set pair to contribute to the unit linking.
`se_method`	`"hybrid"` (sandwich + linking bootstrap + delta propagation; fast, default) or `"bootstrap"` (full person bootstrap of all stages).
`boot_reps`	Bootstrap replicates; defaults to 300 for the linking bootstrap and 200 for the full bootstrap.

Details

Humphry (2005) states the model for dichotomous responses. The polytomous form fitted here, with the frame unit multiplying the whole exponent over the item's partial-credit thresholds, is this package's extension of that statement. It is the form characterised by preserving the two properties the model's logic rests on: the partial credit model holds within every frame in the frame's natural unit (so the pairwise conditional cancellation remains valid), and the weighted score remains sufficient for the person parameter. It reduces exactly to the dichotomous model when items are scored 0/1 and to the ordinary partial credit model when all units equal one. One interpretive consequence: category widths in natural units scale with the frame unit, so a high-unit frame makes proportionally sharper category distinctions; frame-level fit and the per-frame category curves are where a violation of this would appear.

Estimation order: the within-frame pairwise stage establishes the centred set thresholds and the person-group units `phi`; the person-side linking stage then establishes the item-set units `alpha` and

set locations. The reported item parameters and all person measures are computed only after every unit is established: item thresholds are mapped into the common arbitrary unit using `alpha` and the set locations, and person measures are weighted-score weighted likelihood estimates evaluated under the final units `rho = alpha * phi`. The per-frame person estimates used inside the linking stage are interim quantities for the unit ratios only and are discarded. The within-frame stage needs no re-estimation once `alpha` is known, because the pairwise likelihood is invariant to the within-set rescaling that `alpha` represents; the units' own uncertainty is reported in `alpha_table` and `phi_table` and folded into the common-unit standard errors as described below.

Standard errors: under `se_method = "hybrid"` (default) the group units carry sandwich standard errors from the pairwise stage, the set units carry person-bootstrap standard errors from the linking stage, and the unit uncertainty is propagated into the common-unit threshold and item standard errors by the delta method (treating the item-side and person-side information as independent). Under `se_method = "bootstrap"` all stages are re-estimated on `boot_reps` person resamples and every standard error and the threshold covariance come from the replicate spread; slower, but captures all cross-dependencies jointly.

Relation to Humphry (2005, sections 5.3 and 5.4): the two-stage architecture implemented here operationalises the estimation approach proposed in section 5.3 of the thesis (conditional estimation within frames, with the item-set units obtained through person estimates from linked sets), and retains the thesis's error-variance correction (its equation 2.29, after Andrich 1982) and its transformation of standard errors into the common unit by the inverse discrimination. The standard errors themselves go further than the thesis's section 5.4, which inverts each diagonal element of the joint-likelihood information separately and therefore conditions on the remaining parameters, including the person locations, being treated as known. Here full covariance matrices are used throughout; the item-side covariance carries the Godambe sandwich correction required for a pairwise composite likelihood; the unit uncertainty that the thesis's transformation treats as fixed is propagated into the common-unit parameters; and resampling replaces analytic plug-in variances for the person-side linking stage.

Measurement-theoretic status: within every frame the model is strictly Rasch, with person-free item comparisons by conditioning. Across frames it is an argued extension of the theory of the unit (Humphry 2005; Humphry and Andrich 2008): on this account the unit was always a frame-dependent empirical property that the ordinary model leaves implicit, and the extension makes it explicit; the orthodox reading of Rasch measurement contests this, and applied reports should present it as an extension rather than settled doctrine. Two concessions are intrinsic to the model rather than to this implementation: the item-set units are identified only from the person side (their conditional identification is impossible, as documented above), so that step uses distributional information; and person measures rest on weighted-score sufficiency with estimated weights, whose uncertainty is propagated rather than ignored.

Value

An object of classes `"rasch_efrm"` and `"rasch"`. In addition to the standard components (computed over item-by-group virtual columns with the frame units carried in `disc`), it has `frames` (one row per frame: units, origin, pooled fit), `phi_table`, `alpha_table`, `set_table`, `item_arbitrary` and `thresholds_arbitrary` (the structural parameters in the common unit), `score_curves` (per-group score-to-measure curves, replacing the raw-score table), `efrm_vs_rasch` (fit comparison against the equal-unit model on the same conditional information), and `linking` (the linking evidence).

Examples

```
set.seed(1); Np <- 400
simP <- function(th, tau, r) { x <- 0:length(tau)
  p <- exp(r * (x * th - c(0, cumsum(tau)))); p / sum(p) }
grp <- rep(c("A", "B"), each = Np / 2)
phi <- c(A = 0.8, B = 1.25)
d <- seq(-1.5, 1.5, length.out = 10)
X <- sapply(seq_along(d), function(i) sapply(seq_len(Np), function(n)
  sample(0:1, 1, prob = simP(rnorm(Np)[n], d[i], phi[grp[n]]))))
colnames(X) <- sprintf("I%02d", seq_along(d))
fit <- rasch_efrm(data.frame(X, grp = grp), item_sets = list(core = colnames(X)),
                  groups = "grp")
fit$phi_table
```

`rasch_mfrm` *Fit a many-facet Rasch model*

Description

Estimates the many-facet Rasch model (Linacre 1989) for long-format data in which each row is one scored response carrying a person, an item, a score, and one or more facet levels (for example the rater). Every item-by-facet combination becomes a virtual item whose thresholds are the item's thresholds shifted by the facet severities, and the whole structure is estimated in one pass of the pairwise conditional likelihood, in which the person parameter cancels. Facet severities are reported with standard errors and pooled fit statistics; the returned object is also a full `rasch` fit at the virtual-item level, so every diagnostic table and plot in the package applies to it.

Usage

```
rasch_mfrm(
  data,
  person,
  item = NULL,
  score = NULL,
  facets,
  items = NULL,
  n_groups = NULL,
  adjust_N = NA,
  na_codes = -1,
  interaction = NULL,
  maxit = 60,
  tol = 1e-08
)
```

Arguments

data	Long-format data frame.
person	Name of the person identifier column.
item	Name of the item column.
score	Name of the integer score column (categories from 0; gaps are collapsed per item with a note).
facets	Character vector naming one or more facet columns (for example a rater column).
items	Optional character vector of item score columns for data in wide format: one row per person-by-facet combination (for example one row per script per rater) with one column per item or criterion. The long form (item + score) remains available for data where the facet varies within items.
n_groups	Number of class intervals for the item-trait chi-square; NULL (the default) applies the class-interval rule of Andrich and Marais (2019, ch. 15) (at least 50 non-extreme persons per interval, at most 10 intervals, at least 2).
adjust_N	Optional reference sample size for the chi-square.
na_codes	Score values to read as missing (default -1); any negative score is also treated as missing.
interaction	Optional name of one facet to interact with the items (interactive facet mode). Adds item-by-facet terms gamma[item, level] with double sum-to-zero constraints on top of the additive severities, so each level may be more or less severe on particular items; estimates are returned in interaction_effects. The interactive model remains in the Rasch class (all discriminations equal one and the parameters are additive), but a significant interaction qualifies specific objectivity in practice: comparisons of the interacting facet's levels become item-dependent, which is itself the substantive finding.
maxit, tol	Newton-Raphson iteration cap and convergence tolerance.

Value

An object of classes "rasch_mfrm" and "rasch". In addition to every component of a rasch fit (computed over the virtual items), it carries facet_effects (per facet: level, severity, standard error, observation count, pooled fit), item_effects (underlying item locations and pooled fit), item_thresholds (the structural delta_ik with standard errors), and facet_spec. Two fit residuals are reported per facet level and per underlying item. fit_resid is the facet-margin statistic of the published three-facet fit tables (Andrich and Marais 2019, ch. 26 and app. C), the mean of the constituent virtual items' fit residuals; it weighs each virtual item equally, so an erratic level shows the average of its per-item misfit. fit_resid_pooled is the log-of-mean-square statistic summed over the margin's observed cells of non-extreme persons, with its degrees of freedom in df_fit; it weighs each response equally and is the more powerful statistic when misfit is spread evenly over the level's cells.

Examples

```
set.seed(1)
```

```
simP <- function(th, tau) { x <- 0:length(tau); p <- exp(x * th - c(0, cumsum(tau))); p / sum(p) }
persons <- sprintf("P%03d", 1:120); raters <- paste0("R", 1:4)
th <- setNames(rnorm(120, 0, 1.3), persons)
rho <- setNames(c(-0.6, -0.2, 0.2, 0.6), raters)
tau <- list(A = c(-1, 1), B = c(-0.5, 1.2), C = c(-1.2, 0.4))
d <- expand.grid(person = persons, item = names(tau), rater = raters,
                 stringsAsFactors = FALSE)
d$score <- mapply(function(p, i, r)
  sample(0:2, 1, prob = simP(th[p], tau[[i]] + rho[r])), d$person, d$item, d$rater)
fit <- rasch_mfrm(d, person = "person", item = "item", score = "score",
                  facets = "rater")
fit$facet_effects$rater
```

`report_html` *Write a self-contained HTML report of a Rasch analysis*

Description

Builds a single portable HTML file containing the complete analysis: the summary statistics, every diagnostic table, and every test-level plot embedded as an image, styled for reading and sharing. The file has no external dependencies, so it can be e-mailed or archived as the record of an analysis.

Usage

```
report_html(fit, file, title = "Rasch measurement analysis", dpi = 150)
```

Arguments

`fit`	A fitted object from `rasch`.
`file`	Path of the HTML file to write.
`title`	Report title.
`dpi`	Resolution of the embedded plots.

Value

Invisibly, `file`.

Examples

```
set.seed(1)
d <- seq(-2, 2, length.out = 6)
X <- matrix(rbinom(300 * 6, 1, plogis(outer(rnorm(300), d, "-"))), 300, 6)
colnames(X) <- paste0("I", 1:6)
out <- file.path(tempdir(), "report.html")
report_html(rasch(X), out)
```

`residual_correlations` *Residual correlations for local dependence (Yen's Q3)*

Description

The pairwise correlations of the standardised response residuals are Yen's (1984) Q3 statistics. Under unidimensionality and local independence the off-diagonal values sit near `-1/(L-1)`; large positive values flag local dependence between item pairs. Following Christensen, Makransky and Horton (2017), each Q3 is also reported relative to the average off-diagonal value (`q3_star`), and a pair is flagged when that excess passes `flag`.

Usage

```
residual_correlations(fit, flag = 0.2)
```

Arguments

`fit`	A fitted object from `rasch`.
`flag`	Excess above the average off-diagonal Q3 at which a pair is flagged as dependent.

Value

A list with the Q3 `matrix`, the adjusted-Q3 `star_matrix` (each Q3 less the average off-diagonal value, diagonal empty), the `average` off-diagonal value, `pairs` (every item pair with `q3`, `q3_star` and a `flagged` indicator, sorted by `q3`), and the subset of `flagged` pairs.

References

Yen, W. M. (1984). Effects of local item dependence on the fit and equating performance of the three-parameter logistic model. *Applied Psychological Measurement*, 8(2), 125-145.

Christensen, K. B., Makransky, G., & Horton, M. (2017). Critical values for Yen's Q3: identification of local dependence in the Rasch model using residual correlations. *Applied Psychological Measurement*, 41(3), 178-194.

Examples

```
set.seed(1)
d <- seq(-2, 2, length.out = 8)
X <- matrix(rbinom(500 * 8, 1, plogis(outer(rnorm(500), d, "-"))), 500, 8)
colnames(X) <- paste0("I", 1:8)
residual_correlations(rasch(X))$average
```

residual_pca *Principal components of the residual correlations*

Description

The first residual component (PC1) carries any second dimension; items with opposing loadings define the split used by the unidimensionality t-test. Loadings for the leading components and the eigenvalue table support inspection beyond the first component.

Usage

```
residual_pca(fit, n_components = 10)
```

Arguments

fit	A fitted object from rasch.
n_components	Number of leading components to return loadings and eigenvalue rows for (capped at the number of items).

Value

A list with the residual eigenvalues, their proportions, the first-component loadings (sorted), the loadings_matrix for the leading components, the eigen_table (component, eigenvalue, proportion, cumulative), and the first_eigenvalue.

Examples

```
set.seed(1)
d <- seq(-2, 2, length.out = 8)
X <- matrix(rbinom(500 * 8, 1, plogis(outer(rnorm(500), d, "-"))), 500, 8)
colnames(X) <- paste0("I", 1:8)
residual_pca(rasch(X))$first_eigen
```

resolve_dif *Resolve differential item functioning by iterative item splitting*

Description

Splits DIF items one at a time, largest effect first, refitting after each split, until no item shows significant DIF (or the anchor set would fall too low). Splitting the item with the largest real DIF first removes the artificial DIF it induces on other items (Andrich & Hagquist 2012, 2015), so the procedure resolves genuine DIF without chasing the artificial DIF that a single simultaneous pass would flag. Each split resolves the item into one copy per group (or per factor-combination cell for an interaction), with independent locations and thresholds, so both uniform and non-uniform DIF are resolved together.

Usage

```
resolve_dif(
  fit,
  factors = NULL,
  alpha = 0.05,
  p_adjust = "BH",
  min_anchors = NULL,
  max_splits = NULL
)
```

Arguments

fit	A fitted object from rasch carrying person factors.
factors	Person factors to test, as in dif_anova; defaults to every nominated factor.
alpha	Significance level for the adjusted probabilities.
p_adjust	Multiplicity adjustment across items each round.
min_anchors	Minimum number of original items to leave unsplit; the procedure stops before the anchor set falls below this (pervasive DIF is not artificial DIF). Default max(3, items / 4).
max_splits	Hard cap on the number of splits. Default: the number of items.

Value

A list of class "rasch_resolve_dif": the final resolved fit, the splits performed (order, item, factor, partial eta-squared, DIF magnitude in logits), the stopped reason, and the residual dif table for the final fit.

References

Andrich, D., & Hagquist, C. (2012). Real and artificial differential item functioning. *Journal of Educational and Behavioral Statistics*, 37(3), 387-416.

Examples

```
set.seed(1); n <- 600
d <- seq(-2, 2, length.out = 8); g <- rep(c("a", "b"), each = n / 2)
sh <- matrix(0, n, 8); sh[g == "b", 3] <- 1.2      # one strong DIF item
X <- matrix(rbinom(n * 8, 1, plogis(outer(rnorm(n), d, "-") - sh)), n, 8)
colnames(X) <- paste0("I", 1:8)
fit <- rasch(data.frame(X, grp = g), factors = "grp")
resolve_dif(fit)$splits
```

run_app *Launch the rasch graphical interface*

Description

Opens the Shiny application: data upload with ID, person-factor, and item column nomination; the full analysis; interactive tables and plots; and one-click export of every table and plot.

Usage

```
run_app(...)
```

Arguments

`...`	Passed to `shiny::runApp`.

Value

Called for its side effect of launching the app.

Examples

```
if (interactive()) run_app()
```

save_item_plots *Save a plot for every item*

Description

Writes one plot per item – the item characteristic curve, category probability curves, threshold probability curves, or category frequencies – to a single multi-page PDF or a ZIP archive of PNGs, chosen by the extension of `file`.

Usage

```
save_item_plots(
  fit,
  what = c("icc", "ccc", "tpc", "cfreq"),
  file,
  items = NULL,
  n_groups = fit$n_groups,
  grid = seq(-5, 5, 0.05),
  observed = TRUE,
  width = 8,
  height = 5.5,
  dpi = 300
)
```

Arguments

fit	A fitted object from rasch.
what	Which plot: "icc", "ccc", "tpc", or "cfreq".
file	Output path ending in .pdf (one page per item) or .zip (one PNG per item).
items	Item names or indices; all items by default.
n_groups	Class intervals for observed overlays.
grid	Logit grid for the curves.
observed	Overlay observed proportions on the category and threshold probability curves.
width, height, dpi	Device size in inches and PNG resolution.

Value

Invisibly, the output path.

Examples

```
set.seed(1)
d <- seq(-2, 2, length.out = 6)
X <- matrix(rbinom(400 * 6, 1, plogis(outer(rnorm(400), d, "-"))), 400, 6)
colnames(X) <- paste0("I", 1:6)
f <- rasch(X)
save_item_plots(f, "icc", file.path(tempdir(), "icc_all.pdf"))
```

---

save_outputs *Save every output of a Rasch analysis to a folder*

Description

Writes all tables (item statistics, thresholds with standard errors, person estimates including ID and factors, the score-to-measure table, residual correlations, principal-component loadings, category frequencies, and DIF results for every nominated factor) as CSV; every plot, including the per-item characteristic, category, threshold, and frequency plots, as PNG and optionally PDF; and a plain-text analysis summary.

Usage

```
save_outputs(
  fit,
  dir,
  formats = c("png", "pdf"),
  width = 9,
  height = 6,
  dpi = 300,
  item_plots = TRUE
)
```

Arguments

`fit`	A fitted object from `rasch`.
`dir`	Output directory; created if absent.
`formats`	Plot formats, any of `"png"` and `"pdf"`.
`width, height`	Plot size in inches.
`dpi`	PNG resolution; the default 300 is publication quality.
`item_plots`	Also write the per-item plot set (one ICC, category curve, threshold curve, and frequency chart per item).

Value

Invisibly, the vector of files written.

Examples

```
set.seed(1)
d <- seq(-2, 2, length.out = 6)
X <- matrix(rbinom(300 * 6, 1, plogis(outer(rnorm(300), d, "-"))), 300, 6)
colnames(X) <- paste0("I", 1:6)
out <- file.path(tempdir(), "rasch-out")
save_outputs(rasch(X), out, formats = "png", item_plots = FALSE)
```

---

save_person_plots *Save a kidmap for every person*

Description

Writes one kidmap (`plot_kidmap`) per person to a single multi-page PDF or a ZIP archive of PNGs, chosen by the extension of `file`. Persons without a location estimate are skipped.

Usage

```
save_person_plots(
  fit,
  file,
  persons = NULL,
  level = 0.95,
  width = 8,
  height = 6,
  dpi = 300
)
```

Arguments

`fit`	A fitted object from `rasch`.
`file`	Output path ending in `.pdf` (one page per person) or `.zip` (one PNG per person).
`persons`	Row numbers or IDs; all estimated persons by default.
`level`	Confidence level of the band marking unexpected responses.
`width, height, dpi`	Device size in inches and PNG resolution.

Value

Invisibly, the output path.

Examples

```
set.seed(1)
d <- seq(-2, 2, length.out = 6)
X <- matrix(rbinom(60 * 6, 1, plogis(outer(rnorm(60), d, "-"))), 60, 6)
colnames(X) <- paste0("I", 1:6)
f <- rasch(X)
save_person_plots(f, file.path(tempdir(), "kidmaps.pdf"), persons = 1:5)
```

`score_table` *Raw score to measure conversion table*

Description

The score-to-logit conversion for complete responders: every possible raw score with its location, standard error, and the frequency and cumulative percentage of complete responders at that score (the complete-data estimates table of Andrich and Marais 2019, ch. 10).

Usage

```
score_table(
  fit,
  method = c("wle", "mle"),
  extremes = c("model", "extrapolated")
)
```

Arguments

`fit`	A fitted object from `rasch`.
`method`	`"wle"` (Warm, default) or `"mle"`.
`extremes`	`"model"` keeps the estimator's own extreme-score values (`NA` for MLE); `"extrapolated"` applies the geometric extrapolation.

Details

Two estimators are available. `"wle"` (the default) is Warm's weighted likelihood estimate, finite at the extreme scores. `"mle"` is the plain maximum likelihood estimate, infinite at the extremes. `extremes = "extrapolated"` replaces the extreme-score entries by the geometric extrapolation described in Andrich and Marais (2019, ch. 10): successive score-to-score differences grow towards the extremes, so the last difference is continued geometrically – the extrapolated top difference d solves $b = \sqrt{ad}$ where a, b are the two preceding differences (equivalently $d = b^2/a$), and symmetrically at zero. The standard error at an extrapolated location is $1/\sqrt{I(\theta)}$ evaluated there. With `method = "wle"` the extrapolation replaces the finite Warm estimates at the extremes, giving the extrapolated form of the conversion table from a WLE analysis.

Value

A data frame with `score`, `theta`, `se`, `freq`, `cum_pct` (omitted when no complete responders exist), and `extrapolated`; `NULL` for fits without a common raw-score metric (EFRM).

Examples

```
set.seed(1)
d <- seq(-1.5, 1.5, length.out = 6)
X <- matrix(rbinom(300 * 6, 1, plogis(outer(rnorm(300), d, "-"))), 300, 6)
colnames(X) <- paste0("I", 1:6)
score_table(rasch(X), method = "mle", extremes = "extrapolated")
```

---

simulate_btl *Simulate paired-comparison (BTL) data with dial-in misfit*

Description

Generates dichotomous or graded paired comparisons from the Bradley-Terry-Luce model, with optional departures each of which a paired-comparison diagnostic is built to detect. The result is a data frame ready for `btl`, with the truth attached.

Usage

```
simulate_btl(
  n_objects = 8,
  n_judges = 12,
  reps_per_pair = 25,
  model = c("dichotomous", "graded"),
  n_categories = 4,
  object_sd = 1,
  second_attribute = NULL,
  erratic_judges = 0,
  dependence = NULL,
  seed = NULL
)
```

Arguments

n_objects, n_judges	Objects to scale and judges comparing them.
reps_per_pair	Comparisons made of each object pair.
model	"dichotomous" (a winner) or "graded" (a rated margin in n_categories categories).
n_categories	Categories for the graded model.
object_sd	Spread of the object locations (evenly spaced, sum-zero).
second_attribute	NULL, or list(rho=): half the judges rank by a second object attribute correlated rho with the first – genuine multidimensionality. Feeds btl_dimensionality and btl_transitivity.
erratic_judges	Proportion of judges who choose at random. Feeds the judge fit residual, btl_transitivity consistency, and judge_surprise.
dependence	NULL, or list(exposure=, carry_over=): within-judge order effects (a seen-before advantage and a pull from the judge's own earlier verdicts). Adds an order column. Feeds the dependence effects of btl.
seed	Optional RNG seed.

Value

A data frame of class "rasch_sim": object_a, object_b, winner (or response when graded), judge, and order when dependence is planted; with attr(x, "truth").

Examples

```
d <- simulate_btl(8, 12, erratic_judges = 0.15, seed = 1)
bt <- btl(d, "object_a", "object_b", winner = "winner", judge = "judge")
bt$judges          # the erratic judges carry large fit residuals
```

---

simulate_btl_efrm *Simulate paired-comparison EFRM data with differing frame units*

Description

Generates dichotomous paired comparisons whose latent unit differs across judge-panel by object-set frames – the paired-comparison extension of the extended frame of reference model (Humphry 2005) fitted by btl_efrm. Objects in set s have a within-set calibration location beta; their common-scale value is v = alpha_s beta + kappa_s. A comparison judged in panel g carries the panel unit phi_g: within a set the comparison logit is phi_g (beta_a - beta_b), across sets it is phi_g (v_a - v_b). The planted panel units, set units and origins are recovered by btl_efrm.

Usage

```
simulate_btl_efrm(
  n_objects_per_set = 8,
  n_sets = 2,
  n_judges_per_panel = 6,
  n_panels = 2,
  reps_within = 20,
  reps_cross = 20,
  panel_units = NULL,
  set_units = NULL,
  set_origins = NULL,
  object_sd = 1,
  seed = NULL
)
```

Arguments

n_objects_per_set, n_sets	Objects in each set and number of sets.
n_judges_per_panel, n_panels	Judges in each panel and number of panels.
reps_within	Replications of each within-set object pair.
reps_cross	Replications of each cross-set object pair.
panel_units	Panel units phi (length n_panels); the default is all one, and any supplied vector is rescaled to geometric mean one.
set_units	Set units alpha (length n_sets); the default is all one, and alpha_1 is forced to one (the reference set).
set_origins	Set origins kappa (length n_sets); the default is all zero, and kappa_1 is forced to zero.
object_sd	Spread of the within-set calibration locations.
seed	Optional RNG seed.

Value

A data frame of class "rasch_sim" with columns object_a, object_b, winner, judge and panel, and attr(x, "truth") holding the common-scale values v, the per-set beta, the units phi, alpha, kappa, and the object_sets map to pass to btl_efrm.

Examples

```
d <- simulate_btl_efrm(6, 2, set_units = c(1, 1.4), seed = 1)
bt <- btl_efrm(d, "object_a", "object_b", winner = "winner",
               judge = "judge", panels = "panel",
               object_sets = attr(d, "truth")$object_sets)
bt$alpha_table   # recovers the ~1.4 set unit
```

`simulate_efrm` *Simulate extended frame-of-reference data with differing units*

Description

Generates data whose latent unit differs across item-set by person-group frames (Humphry 2005): a person in group g responding to an item in set s does so at the frame unit rho = alpha_set * phi_group scaling the whole exponent. The planted set- and group-unit ratios are recovered by `rasch_efrm`.

Usage

```
simulate_efrm(
  n_per_group = 300,
  items_per_set = 8,
  n_sets = 2,
  n_groups = 2,
  set_unit_ratio = 1.3,
  group_unit_ratio = 1,
  theta_sd = 1.3,
  seed = NULL
)
```

Arguments

`n_per_group`	Persons in each group.
`items_per_set`	Items in each set.
`n_sets, n_groups`	Numbers of item sets and person groups.
`set_unit_ratio, group_unit_ratio`	Geometric span of the set and group units across their levels (1 = equal units, i.e. an ordinary Rasch fit).
`theta_sd`	Spread of person ability.
`seed`	Optional RNG seed.

Value

A wide data frame of class `"rasch_sim"` (`id`, item columns, `group`) with `attr(x, "truth")$item_sets` the set map to pass to `rasch_efrm`.

Examples

```
d <- simulate_efrm(300, 8, set_unit_ratio = 1.3, seed = 1)
tr <- attr(d, "truth")
ef <- rasch_efrm(d, item_sets = tr$item_sets, groups = "group")
ef$alpha_table   # recovers the ~1.3 set-unit ratio
```

`simulate_mfrm` *Simulate many-facet (rated) data with dial-in misfit*

Description

Generates ratings from the many-facet Rasch model (Linacre 1989): every rater rates every person on every item, from person ability, item difficulty, and rater severity. Departures each feed an MFRM diagnostic.

Usage

```
simulate_mfrm(
  n_persons = 80,
  n_items = 5,
  n_raters = 6,
  n_categories = 4,
  theta_sd = 1.2,
  item_sd = 1,
  rater_severity_sd = 0.6,
  erratic_raters = 0,
  interaction = NULL,
  halo = 0,
  seed = NULL
)
```

Arguments

`n_persons`, `n_items`, `n_raters`	Facet sizes (fully crossed).
`n_categories`	Rating categories.
`theta_sd`, `item_sd`	Spread of person ability and item difficulty.
`rater_severity_sd`	Spread of rater severities (the core facet; recovered in `facet_effects`).
`erratic_raters`	Proportion of raters who rate at random (feeds the rater fit residual).
`interaction`	`NULL`, or `list(rater=, item=, bias=)`: one rater is unusually harsh (positive) or lenient (negative) on one item. Feeds the item-by-rater interaction (fit with `interaction = `).
`halo`	Proportion of raters showing a halo effect: they rate by the person's overall level and barely differentiate items (feeds the rater fit residual and the item-by-rater interaction).
`seed`	Optional RNG seed.

Value

A long data frame of class `"rasch_sim"` (`person`, `item`, `rater`, `score`) ready for `rasch_mfrm`, with the truth attached.

Examples

```
d <- simulate_mfrm(60, 5, 6, rater_severity_sd = 0.8, seed = 1)
mf <- rasch_mfrm(d, person = "person", item = "item", score = "score",
                 facets = "rater")
cor(mf$facet_effects$rater$severity, attr(d, "truth")$severity)  # recovered
```

`simulate_rasch` *Simulate person-by-item Rasch data with dial-in misfit*

Description

Generates dichotomous or polytomous (partial credit / rating scale) data from the Rasch model, with optional, individually controllable departures from it – each of which the package's matching diagnostic is built to detect. The result is a data frame ready for `rasch`, with the true parameters attached as `attr(x, "truth")`.

Usage

```
simulate_rasch(
  n_persons = 500,
  n_items = 20,
  model = c("dichotomous", "PCM", "RSM"),
  n_categories = 3,
  theta_mean = 0,
  theta_sd = 1,
  theta_dist = "normal",
  difficulty = c(-2.5, 2.5),
  threshold_spread = 1.2,
  discrimination = 1,
  guessing = 0,
  second_dim = NULL,
  dependence = NULL,
  dif = NULL,
  careless = 0,
  response_style = NULL,
  speeded = 0,
  disordered = NULL,
  n_groups = 1,
  missing = 0,
  seed = NULL
)
```

Arguments

`n_persons`, `n_items`
: Sample size and test length.

`model` "dichotomous", "PCM", or "RSM". Under "RSM" every item shares one category-threshold pattern (items differ by location only); under "PCM" each item's threshold spacings and span are drawn afresh, as the partial credit model allows.

`n_categories` Response categories for polytomous models (>= 3).

`theta_mean, theta_sd, theta_dist`
Person distribution: mean, SD, and shape ("normal", "uniform", "skew", "bimodal").

`difficulty` Two numbers giving the item-location range (evenly spaced), or a length-`n_items` vector of locations.

`threshold_spread`
Half-range of the category thresholds about each item location (polytomous).

`discrimination` Scalar or length-`n_items`: the slope of each item. Values above 1 over-discriminate (Guttman-like, negative fit residual); below 1 under-discriminate (noisy, positive residual). Feeds infit/outfit and the item-fit F.

`guessing` Scalar or length-`n_items` lower asymptote (dichotomous): low-ability persons answer correctly by chance. Feeds `tailored_analysis`.

`second_dim` NULL, or `list(items=, rho=)`: the named items load on a second trait correlated `rho` with the first. Feeds `dimensionality_test`.

`dependence` NULL, or `list(pairs=, strength=)`: each pair's second item responds partly to the first (response dependence). Feeds `residual_correlations` / `dependence_magnitude`.

`dif` NULL, or `list(items=, uniform=, nonuniform=)`: the named items function differently for the last person group – a location shift (`uniform`) and/or a slope change (`nonuniform`). Needs `n_groups` >= 2. Feeds `dif_anova` / `dif_size`.

`careless` Proportion of persons who answer at random (person misfit; feeds person infit/outfit).

`response_style` NULL, or `list(type=, prop=, strength=)` with `type` "extreme" or "middle": a proportion `prop` of persons favour the end (or middle) categories regardless of the trait, with distortion `strength` (default 1.6) on the log-probability scale (polytomous; feeds the category diagnostics and person fit).

`speeded` Proportion not-reached at the last item: a growing tail of missing responses over the final items, as under time pressure (feeds the item statistics and the missingness pattern).

`disordered` NULL or item names/indices given disordered thresholds (polytomous; feeds the threshold diagnostics).

`n_groups` Number of equal person groups (a `group` factor column is added when > 1, for DIF).

`missing` Proportion of responses set missing (completely at random).

`seed` Optional RNG seed.

Value

A data frame of class "rasch_sim" (item columns `I01`..., an `id` column, and a `group` column when grouped), with `attr(x, "truth")` holding the generating parameters and the planted departures.

Examples

```
# a clean scale with one over-discriminating item and one DIF item
d <- simulate_rasch(400, 12, discrimination = c(3, rep(1, 11)),
                    dif = list(items = "I06", uniform = 1), n_groups = 2,
                    seed = 1)
fit <- rasch(d, id = "id", factors = "group")
fit$items[c("item", "infit_ms", "outfit_ms")]   # item 1 misfits
dif_anova(fit)$summary                           # item 6 flags
```

---

sim_recovery *Parameter recovery of a fit against the simulation truth*

Description

Compares the parameters recovered by a fit with the ones a `simulate_*` function planted (carried on the data as `attr(sim, "truth")`): item difficulties and person abilities for a Rasch fit, object locations for a paired-comparison fit, rater severities (with item and person measures) for a many-facet fit, and the set units for a frames fit. Locations are mean-centred before comparison, since the model identifies them only up to an origin.

Usage

```
sim_recovery(fit, sim)
```

Arguments

`fit`	A fit of the simulated data (`rasch`, `btl`, `rasch_mfrm`, or `rasch_efrm`).
`sim`	The simulated data (from a `simulate_*` function).

Value

A list of class `"rasch_recovery"`: `summary` (per parameter type: n, correlation, RMSE, bias) and `pieces` (the true and estimated values behind each).

Examples

```
d <- simulate_rasch(500, 12, seed = 1)
sim_recovery(rasch(d, id = "id"), d)$summary
```

sim_replicate *Replicate a simulation for Monte Carlo studies*

Description

Calls one of the `simulate_*` functions `n` times with successive seeds, returning the datasets as a list – for power, Type-I, or parameter-recovery studies.

Usage

```
sim_replicate(FUN, n, ..., seed = NULL)
```

Arguments

`FUN`	A simulator, e.g. `simulate_rasch`.
`n`	Number of datasets.
`...`	Arguments passed to `FUN` (the same each replicate).
`seed`	Seed of the first replicate (each subsequent one increments it).

Value

A list of class `"rasch_sim_batch"`, one simulated dataset per element.

Examples

```
# 20 datasets with a planted DIF item; how often is it flagged?
batch <- sim_replicate(simulate_rasch, 20, n_persons = 400, n_items = 10,
                       dif = list(items = "I05", uniform = 0.8), n_groups = 2,
                       seed = 1)
mean(vapply(batch, function(d)
  dif_anova(rasch(d, id = "id", factors = "group"))$summary$uniform_DIF[5], TRUE))
```

split_items *Split items by a person factor to resolve DIF*

Description

Replaces each nominated item with one item per level of a person factor, each carrying that level's responses only (other levels missing). Every group then receives its own item location, which resolves the invariance violation flagged by `dif_anova`; the distance between the split locations estimates the DIF size. The model is refitted with the same settings.

Usage

```
split_items(fit, items, by)
```

Arguments

`fit`	A fitted object from `rasch`.
`items`	Character vector naming the item(s) to split.
`by`	The name of a person factor nominated in the fit, or a grouping vector with one entry per person.

Value

A new `rasch` fit in which each split item appears as `"item (level)"`, with the splits recorded in its notes.

Examples

```
set.seed(1); n <- 600
d <- seq(-2, 2, length.out = 8); g <- rep(c("a", "b"), each = n / 2)
sh <- matrix(0, n, 8); sh[g == "b", 3] <- 1
X <- matrix(rbinom(n * 8, 1, plogis(outer(rnorm(n), d, "-") - sh)), n, 8)
colnames(X) <- paste0("I", 1:8)
fit <- rasch(data.frame(X, grp = g), factors = "grp")
fit2 <- split_items(fit, "I3", by = "grp")
fit2$items$item
```

`spread_test` *Spread-parameter test for dependence within subtests*

Description

Andrich's (1985) least-upper-bound screen: the spread component λ of a polytomous item (half the distance between successive thresholds in the principal-components parameterisation, estimated here by `pcml_pc`) cannot fall below the value implied by the binomial distribution when the item is a subtest of equally difficult, independent dichotomous items; different difficulties only raise it. A spread estimate below the bound therefore indicates response dependence among the members (Andrich and Marais 2019, Table 24.1). Typically applied after `combine_items`, whose super-items are exactly such subtests.

Usage

```
spread_test(fit, maxit = 60, tol = 1e-08)
```

Arguments

`fit`	A fitted object from `rasch`.
`maxit, tol`	Passed to the `pcml_pc` refit.

Value

A data frame with one row per polytomous item: `item`, `m`, the `spread` estimate and its `se`, the bound `lub` (available for maximum scores 2 to 8), `z` = (spread - lub)/se, and `dependent` = spread below the bound. Dichotomous items carry no spread and are omitted.

Examples

```
set.seed(1); N <- 600
d0 <- seq(-1.5, 1.5, length.out = 8)
X <- matrix(rbinom(N * 8, 1, plogis(outer(rnorm(N), d0, "-"))), N, 8)
X[, 5] <- X[, 4]; X[, 6] <- X[, 4]                # a dependent triple
colnames(X) <- paste0("I", 1:8)
fit2 <- combine_items(rasch(X), list(c("I4", "I5", "I6"), c("I1", "I2", "I3")))
spread_test(fit2)
```

---

tailored_analysis *Tailored analysis for guessing*

Description

Runs the four-step tailored procedure of Andrich, Marais and Humphry (2012) on a dichotomous analysis. Step 1 is the supplied fit. Step 2 (tailored) sets to missing every observed response whose modelled probability of success, at the step-1 person and item estimates, is below `chance`, and re-estimates items and persons. Step 3 (origin-equated) re-analyses the *original* data with the mean location of the anchor items fixed at their tailored values by average anchoring, so the two calibrations share an origin. Step 4 (all-anchored) fixes every item at its tailored difficulty and re-estimates persons on the original data. Guessing is indicated when difficult items are estimated harder in the tailored analysis than in the origin-equated one; the comparison table and `plot_equate` on the two calibrations show it directly.

Usage

```
tailored_analysis(fit, chance = 0.25, anchor_items = NULL)
```

Arguments

`fit`	A dichotomous fit from `rasch`.
`chance`	The guessing floor: the probability of success by chance (1/number of options; default 0.25).
`anchor_items`	Items whose mean location fixes the common origin in step 3. The default takes the third of the test (at least two items) least affected by tailoring – fewest responses removed, ties broken towards the easier tailored location – which are the easy items the procedure trusts.

Value

A list of class "rasch_tailored": tailored, origin_equated, and anchored fits, the comparison table (initial, tailored, origin-equated locations, the tailored-minus-equated shift, and its z), the number of responses removed, and the anchor items used.

References

Waller, M. I. (1989). Modeling guessing behavior: A comparison of two IRT models. Applied Psychological Measurement, 13, 233-243. Andrich, D., Marais, I. and Humphry, S. (2012). Using a theorem by Andersen and the dichotomous Rasch model to assess the presence of random guessing in multiple choice items. Journal of Educational and Behavioral Statistics, 37, 417-442.

Examples

```
set.seed(1); N <- 800
d <- seq(-2, 2.5, length.out = 10); th <- rnorm(N)
P <- plogis(outer(th, d, "-"))
P <- 0.25 + 0.75 * P            # uniform guessing floor
X <- matrix(rbinom(N * 10, 1, P), N, 10)
colnames(X) <- paste0("I", 1:10)
ta <- tailored_analysis(rasch(X), chance = 0.25)
ta$table
```

---

targeting_table *Targeting and reliability summary as a table*

Description

The person and item location moments (with and without extreme persons), the threshold range and coverage, and the reliability indices – PSI, PSI without extremes, item separation, and coefficient alpha – as a two-column table suitable for saving and reporting.

Usage

```
targeting_table(fit)
```

Arguments

fit	A fitted object from rasch.

Value

A data frame with columns statistic and value.

Examples

```
set.seed(1)
d <- seq(-2, 2, length.out = 6)
X <- matrix(rbinom(300 * 6, 1, plogis(outer(rnorm(300), d, "-"))), 300, 6)
colnames(X) <- paste0("I", 1:6)
targeting_table(rasch(X))
```

`test_information` *Test information function*

Description

Fisher information of the whole test over a grid of person locations, with the corresponding standard error of measurement.

Usage

```
test_information(fit, grid = seq(-6, 6, by = 0.1))
```

Arguments

`fit`	A fitted object from `rasch`.
`grid`	Logit grid over which to evaluate the information.

Value

A data frame with `theta`, `info`, and `sem`.

Examples

```
set.seed(1)
d <- seq(-1.5, 1.5, length.out = 6)
X <- matrix(rbinom(300 * 6, 1, plogis(outer(rnorm(300), d, "-"))), 300, 6)
colnames(X) <- paste0("I", 1:6)
head(test_information(rasch(X)))
```

`threshold_index` *Enumerate item-category thresholds*

Description

Builds the index mapping each item-category threshold to a global id, given the maximum score of each item.

Usage

```
threshold_index(m)
```

Arguments

`m`	Integer vector of maximum scores per item (1 for dichotomous items).

Value

A data frame with columns `id`, `item`, and `k` (the within-item threshold number).

Examples

```
threshold_index(c(1, 3, 2))
```

Index