Package: easyRasch2 (via r-universe)

October 7, 2026

Title Psychometric Analysis with Rasch Measurement Theory

Version 1.4.0

Description Streamlines reproducible Rasch measurement theory analyses for ordinal item-response data, combining estimation routines from 'eRm', 'psychotools', 'mirt', 'iarm', and 'lavaan' with consistent diagnostic, plotting, and reporting layers. Covers the four basic psychometric criteria summarised by Christensen et al. (2021) <doi:10.1111/sms.13908> -- unidimensionality, local independence, ordered response category thresholds, and invariance across subgroups -- together with item fit, targeting, reliability, category functioning, and descriptive item-response plots. A distinguishing feature is the use of simulation-based critical values to replace rule-of-thumb cutoffs for conditional infit mean-square, item-rest-score gamma, Yen's Q3 local-dependence statistic, the largest residual-PCA eigenvalue, ordinal CFA fit indices, and partial-gamma DIF and local-dependence coefficients, optionally augmented with multiplicity-corrected bootstrap p-values. Outputs are knitr::kable() tables and 'ggplot2' figures suitable for direct inclusion in 'Quarto' and 'R Markdown' reports.

License GPL (>= 3)

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

URL https://github.com/pgmj/easyRasch2, https://pgmj.github.io/easyRasch2/

BugReports https://github.com/pgmj/easyRasch2/issues

Depends R (>= 4.1.0)

Imports graphics, grDevices, knitr, mirt, psychotools (>= 0.7-3), stats, utils, rlang

Suggests difR, eRm, geomtextpath, ggdist, ggtext, iarm, mirai, ggplot2 (>= 3.4.0), partykit, psychotree, stablelearner, testthat (>= 3.0.0), rmarkdown, patchwork, scales, mice, ggrepel, lavaan, withr

Config/testthat/edition 3

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake make libuv1-dev libx11-dev

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

Date/Publication 2026-10-06 10:49:51 UTC

RemoteUrl https://github.com/pgmj/easyrasch2

RemoteRef HEAD

RemoteSha b45f3a39d9be7a3a6a2bfe9e62708353c6bf9440

Contents

phq9 . . . 3
RMdifGamma . . . 4
RMdifGammaCutoff . . . 8
RMdifGammaPlot . . . 11
RMdifLR . . . 12
RMdifTree . . . 14
RMdimCFA . . . 19
RMdimCFACutoff . . . 21
RMdimCFAPlot . . . 24
RMdimMartinLof . . . 25
RMdimMartinLofResiduals . . . 29
RMdimResidualPCA . . . 31
RMdimResidualPCACutoff . . . 34
RMitemCatProb . . . 36
RMitemHierarchy . . . 39
RMitemICCPlot . . . 41
RMitemInfit . . . 44
RMitemInfitCutoff . . . 48
RMitemInfitCutoffMI . . . 51
RMitemInfitMI . . . 53
RMitemInfitPlot . . . 56
RMitemParameters . . . 58
RMitemRestscore . . . 60
RMitemRestscoreBoot . . . 64
RMitemRestscoreCutoff . . . 67
RMitemRestscorePlot . . . 70
RMlocdepGamma . . . 71
RMlocdepGammaCutoff . . . 74
RMlocdepGammaPlot . . . 77

RMlocdepQ3 . . . 79
RMlocdepQ3Cutoff . . . 82
RMlocdepQ3Plot . . . 85
RMpersonChange . . . 87
RMpersonFit . . . 93
RMpersonParameters . . . 96
RMplotBar . . . 99
RMplotStackedbar . . . 101
RMplotTile . . . 103
RMreliability . . . 106
RMreliabilityCurve . . . 109
RMretestSD . . . 113
RMscoreSE . . . 116
RMtargeting . . . 118
RMUreliability . . . 121

Index **123**

---

phq9 *PHQ-9 Depression Screener (NHANES Subsample)*

Description

A processed subsample of the Patient Health Questionnaire 9-item (PHQ-9) depression screener from the U.S. National Health and Nutrition Examination Survey (NHANES), September 2024 release. Six hundred respondents were drawn at random from the cycle's PHQ-9 module subject to having complete responses on all nine items, while retaining a realistic share of respondents with a sum-score of zero (n = 8) so that floor behaviour can be illustrated in a Rasch analysis.

Usage

```
phq9
```

Format

A data frame with 600 rows and 12 variables:

q1 Little interest or pleasure in doing things. Integer 0–3.

q2 Feeling down, depressed, or hopeless. Integer 0–3.

q3 Trouble falling/staying asleep, or sleeping too much. Integer 0–3.

q4 Feeling tired or having little energy. Integer 0–3.

q5 Poor appetite or overeating. Integer 0–3.

q6 Feeling bad about yourself — or that you are a failure or have let yourself or your family down. Integer 0–3.

q7 Trouble concentrating on things, such as reading the newspaper or watching television. Integer 0–3.

q8 Moving or speaking so slowly that other people could have noticed — or the opposite, being so fidgety or restless that you have been moving around a lot more than usual. Integer 0–3.

q9 Thoughts that you would be better off dead, or of hurting yourself in some way. Integer 0–3.

gender Self-reported gender, factor with levels `"Female"` and `"Male"` (31 respondents with missing values).

age Age in years (integer, range 15–85).

edu Highest educational attainment, factor with levels `"Elementary School"`, `"High school"`, `"University"`.

Each PHQ-9 item uses a four-point ordinal response scale, scored 0 (`"Not at all"`), 1 (`"Several days"`), 2 (`"More than half the days"`) and 3 (`"Nearly every day"`).

Details

The dataset is a *processed subsample* intended for teaching and for the package's worked example; it should not be treated as a canonical NHANES microdata file. Users wishing to validate against NCHS-published figures should download the original public-use microdata directly from the NHANES website (see *Source*).

Source

U.S. Centers for Disease Control and Prevention, National Center for Health Statistics. *National Health and Nutrition Examination Survey*, September 2024 release. https://wwwn.cdc.gov/nchs/nhanes/search/datapage.aspx?Component=Questionnaire&CycleBeginYear=2024. NHANES data are released to the public domain by the U.S. federal government (https://www.cdc.gov/nchs/policy/data-release-policy.html).

References

Kroenke, K., Spitzer, R. L., & Williams, J. B. W. (2001). The PHQ-9: Validity of a brief depression severity measure. *Journal of General Internal Medicine*, 16(9), 606–613. doi:10.1046/j.1525-1497.2001.016009606.x

Examples

```
data(phq9)
str(phq9)
summary(rowSums(phq9[, 1:9]))
```

`RMdifGamma` *Partial Gamma DIF Analysis*

Description

Computes partial gamma coefficients for Differential Item Functioning (DIF) using `iarm::partgam_DIF()`. Each item is tested for association with a single categorical DIF variable, controlling for the total score.

Usage

```
RMdifGamma(
  data,
  dif_var,
  cutoff = NULL,
  p_value = NULL,
  correction = c("fwer", "fdr_bh", "fdr_by", "none"),
  alpha = 0.05,
  output = "kable"
)
```

Arguments

data	A data.frame or matrix of item responses. Items must be scored starting at 0 (non-negative integers). Missing values (NA) are allowed, but at least one complete case must exist after combining data and dif_var.
dif_var	A vector (factor or character) of the same length as nrow(data), representing the grouping variable for DIF analysis.
cutoff	Optional. Default NULL (no cutoff applied). Can be: • The return value of RMdifGammaCutoff (a list with $item_cutoffs): the data.frame is extracted automatically and simulation metadata is included in the kable caption. • The $item_cutoffs data.frame from RMdifGammaCutoff directly: must have columns Item, gamma_low, gamma_high. When provided, adds columns Gamma_low, Gamma_high, and Flagged (logical; TRUE when the observed partial gamma falls outside the credible range) to the result.
p_value	Logical or NULL. When TRUE, adds two-sided bootstrap p-values (p_gamma, padj_gamma) comparing each item's observed partial gamma against its simulated null distribution, and flagged reflects padj_gamma < alpha instead of the credible range. The asymptotic BH-adjusted p-value and star columns from iarm::partgam_DIF() are **dropped** in this mode (two p-value families in one table would invite double-reading); the simulated gamma_low / gamma_high band is kept as the effect-size reference. TRUE requires the **full** RMdifGammaCutoff object as cutoff (it carries the simulated distributions in $results). NULL, the default, means TRUE when cutoff is that full object and FALSE otherwise, so calling with no cutoff keeps the asymptotic table unchanged. Pass FALSE to flag against the interval instead. The interval makes a poor decision rule, since its width sets a family-wise error rate of 1 - width^k over all k items at once (Johansson, 2026).
correction	Character. Multiplicity correction for the bootstrap p-values: "fwer" (default; Westfall-Young studentised-max step-down), "fdr_bh", "fdr_by", or "none". Ignored when p_value = FALSE.
alpha	Numeric in (0, 1). Significance level used to flag items on the corrected p-value. Default 0.05. Ignored when p_value = FALSE.
output	Character string controlling the return value. Either "kable" (default) for a formatted knitr::kable() table, or "dataframe" for the underlying data.frame.

Details

Partial gamma (Bjorner et al., 1998) measures the association between item response and an exogenous grouping variable, controlling for the total score. Values near 0 indicate no DIF. Recommended interpretive thresholds (Bjorner et al., 1998):

- **No or negligible DIF**: gamma within $[-0.21, 0.21]$, *or* gamma not significantly different from 0.
- **Slight to moderate DIF**: gamma within $[-0.31, 0.31]$ (and outside $[-0.21, 0.21]$), *or* not significantly outside $[-0.21, 0.21]$.
- **Moderate to large DIF**: gamma outside $[-0.31, 0.31]$, **and** significantly outside $[-0.21, 0.21]$.

The `iarm` package must be installed (it is in Suggests, not Imports). The asymptotic adjusted p-value is a Benjamini-Hochberg adjustment over the items of the p-values from `iarm::partgam_DIF()`, applied by easyRasch2. iarm's own adjusted column is a Bonferroni correction whatever method is named, because iarm adjusts one p-value at a time, so it is not used.

Bootstrap p-values. When `p_value = TRUE`, each item's observed partial gamma is compared against its simulated null distribution (from `cutoff$results`, simulated with no DIF by construction). The per-item statistic is the residual studentised by the bootstrap mean and SD; the marginal p-value is the two-sided Monte-Carlo p-value `(1 + #\{|t*| >= |t|\}) / (B + 1)`, so it can be no smaller than `1 / (B + 1)`. The test is two-sided because DIF has no privileged direction: an item can favour either group. This differs from `RMlocdepGamma()`, which tests one-sided for excess positive local dependence. `correction = "fwer"` uses the Westfall-Young studentised-max step-down, which exploits the bootstrap dependence among items (Ferreira, 2024); it is mildly liberal below 400 iterations in `RMdifGammaCutoff()` (a message is issued below that). Unlike the asymptotic p-values from `iarm::partgam_DIF()`, these are calibrated against the *simulated Rasch null* rather than the asymptotic SE; they are model-conditional and sample-size-sensitive, and are reported alongside the simulated effect-size band, not in place of it. The observed partial gamma they test is computed by the same internal routine as the simulated values, which reproduces `iarm::partgam_DIF()` exactly.

Value

- If `output = "kable"`: a `knitr_kable` object with columns "Item", "Partial gamma", "SE", "Lower CI", "Upper CI", "Adj. p-value (BH)", and "p-value sign." (stars for the BH-adjusted p-value, as `iarm` shows them). When `cutoff` is provided, additional columns "Gamma low", "Gamma high", and "Flagged" are included. With `p_value = TRUE`, the asymptotic p-value columns are replaced by bootstrap "p" and "p (adj)".
- If `output = "dataframe"`: a data.frame with columns `Item`, `gamma`, `se`, `lower`, `upper`, `padj_bh`, `Significance`. When `cutoff` is provided, columns `gamma_low`, `gamma_high`, and `flagged` are also included. With `p_value = TRUE`, `padj_bh` and `Significance` are replaced by `p_gamma` and `padj_gamma`.

Multiple comparisons

The marginal p-value controls the error rate of a *single* comparison: for one item (or item pair) decided on in advance it is the relevant value. But scanning all k comparisons and flagging whichever fall below `alpha` tests k hypotheses at once, so the chance of at least one false flag inflates to roughly

$1-(1-\alpha)^k$ (e.g. about 34% for $k = 8$ at `alpha = 0.05`) – even when every marginal p-value is correctly calibrated. The corrected (adjusted) p-value controls this: `correction = "fwer"` bounds the probability of *any* false flag (strict, lower power), while `"fdr_bh"` / `"fdr_by"` bound the expected *proportion* of false flags among those raised (a more lenient middle ground). Rule of thumb: use the marginal p-value for a single pre-specified comparison, and a corrected p-value when screening the whole table – the usual workflow.

References

Bjorner, J. B., Kreiner, S., Ware, J. E., Damsgaard, M. T., & Bech, P. (1998). Differential item functioning in the Danish translation of the SF-36. *Journal of Clinical Epidemiology, 51*(11), 1189–1202. doi:10.1016/S08954356(98)001115

Ferreira, J. A. (2024). Methods of testing a 'small' or 'moderate' number of hypotheses simultaneously. *Journal of Statistical Theory and Practice, 19*(6). doi:10.1007/s42519024004124

Westfall, P. H., & Young, S. S. (1993). *Resampling-Based Multiple Testing*. Wiley.

Johansson, M. (2026). Simulation-based cutoffs for conditional item fit in Rasch models: Iterations, multiplicity correction, and decision stability. *PsyArXiv*. doi:10.31234/osf.io/7pqz4_v2

See Also

`RMdifGammaCutoff`

Examples

```
if (requireNamespace("iarm", quietly = TRUE)) {
  set.seed(42)
  sim_data <- as.data.frame(
    matrix(sample(0:1, 200 * 10, replace = TRUE), nrow = 200, ncol = 10)
  )
  colnames(sim_data) <- paste0("Item", 1:10)
  dif_group <- factor(sample(c("A", "B"), 200, replace = TRUE))

  # Default kable output
  RMdifGamma(sim_data, dif_group)

  # Return as data.frame
  RMdifGamma(sim_data, dif_group, output = "dataframe")

  # Simulation-based cutoffs (100 Monte-Carlo iterations)
  if (requireNamespace("ggdist", quietly = TRUE)) {
    cutoff_res <- RMdifGammaCutoff(sim_data, dif_var = dif_group,
                                   iterations = 100, parallel = FALSE,
                                   seed = 42)
    RMdifGamma(sim_data, dif_group, cutoff = cutoff_res)

    # The full cutoff object switches on bootstrap p-values with the
    # family-wise (Westfall-Young) correction; 100 iterations keeps the
    # example quick, use the default 400 or more in real analyses
    RMdifGamma(sim_data, dif_group, cutoff = cutoff_res,
               output = "dataframe")
```

```
  }
}
```

`RMdifGammaCutoff` *Simulation-Based Partial Gamma DIF Cutoff Determination*

Description

Simulates the distribution of the partial gamma DIF coefficient for each item when there is no DIF, to give empirical critical values and the simulated null behind the bootstrap p-values in `RMdifGamma`. Every simulated dataset keeps each respondent's observed group membership and total score, so the null reflects the observed group sizes and any difference between the groups' latent distributions.

Usage

```
RMdifGammaCutoff(
  data,
  dif_var,
  iterations = 400,
  parallel = TRUE,
  n_cores = NULL,
  verbose = FALSE,
  seed = NULL,
  cutoff_method = "hdci",
  hdci_width = 0.95,
  dgp = c("conditional", "permutation")
)
```

Arguments

`data`	A data.frame or matrix of item responses. Items must be scored starting at 0 (non-negative integers). Only complete cases (rows without any `NA`) are used.
`dif_var`	A vector (factor, character, or integer) defining group membership for DIF analysis. Must have the same length as `nrow(data)`. Group membership is held fixed in every simulated dataset, so the group sizes and each group's distribution of total scores are those observed.
`iterations`	Integer. Number of simulation iterations (default 400).
`parallel`	Logical. Use parallel processing via `mirai` if available (default `TRUE`).
`n_cores`	Integer or `NULL`. Number of parallel workers. When `NULL`, `getOption("mc.cores")` is checked first. If neither is set and `parallel = TRUE`, a warning is issued and execution falls back to sequential (single core) processing.
`verbose`	Logical. Show a progress bar (default `FALSE`).
`seed`	Integer or `NULL`. Random seed for reproducibility. See easyRasch2-reproducibility for what this guarantees and how it interacts with `parallel`.

`cutoff_method` Character string specifying how cutoff intervals are computed. Either `"hdci"` (default) for the Highest Density Interval via `ggdist::hdci()`, or `"quantile"` for the 2.5th/97.5th percentiles via `stats::quantile()`.

`hdci_width` Numeric. Width of the HDCI when `cutoff_method = "hdci"`. Default is `0.95` (95\ Ignored when `cutoff_method = "quantile"`.

`dgp` Character. How the null datasets are generated. `"conditional"` (default) draws each respondent's response pattern from the Rasch conditional distribution given their observed total score, with the CML item thresholds fixed, as the conditional option of `RMitemRestscoreCutoff` does. `"permutation"` keeps the observed responses and permutes group membership among respondents with the same total score, the Monte Carlo form of an exact conditional test of item-group independence given the score (Kreiner, 1987). It needs no item parameters, keeps any misfit elsewhere in the observed responses, and is 25 to 60 times faster per iteration. Both held the nominal error rate in simulation, including when the groups differed by 1 SD.

Details

Partial gamma conditions on the total score, and under the Rasch model the response to an item is independent of group membership given the total score. Both data-generating processes preserve each respondent's group and total score, so the simulated null has the observed distribution of groups across score strata. An earlier version assigned simulated respondents to groups at random, which gave both groups the same latent distribution. When a minority group differed from the majority by 1 SD, that null was too narrow and flagged at least one item in about 14 percent of datasets with no DIF, against a nominal 5 percent.

For each iteration the function draws one null dataset (see `dgp`) and computes partial gamma for every item. The computation reproduces `iarm::partgam_DIF()` exactly but is vectorised, so `iarm` is not needed here. Iterations that fail are discarded and reported through `actual_iterations`.

The conditional data-generating process uses CML item thresholds from `psychotools::pcmodel()` (a dichotomous item is a 2-category PCM). The group order follows the levels of `dif_var` (alphabetical for character vectors), which sets the sign of gamma as in `iarm::partgam_DIF()`.

Parallel processing is provided by the `mirai` package (optional). Install it with `install.packages("mirai")` to enable parallelisation.

Value

A list with components:

`results` data.frame with columns `iteration`, `Item`, and `gamma` (one row per item per successful iteration).

`item_cutoffs` data.frame with per-item cutoff summaries: `Item`, `gamma_low`, `gamma_high`. Bounds are computed using the method specified by `cutoff_method`.

`actual_iterations` Number of successful iterations.

`requested_iterations` Number of iterations asked for.

`sample_n` Number of complete cases used.

`sample_n_total` Number of respondents in the raw input data, before removing rows with `NA` in `data` or `dif_var`.

`sample_has_na` Logical. Whether `data` or `dif_var` contained any missing values.

`sample_summary` Summary statistics of the WLE person locations, or `NULL` when the model could not be fitted with `dgp = "permutation"`.

`item_names` Character vector of item names from data.

`dif_group_sizes` Integer vector of group sizes, held fixed in every simulated dataset.

`cutoff_method` The method used to compute cutoffs (`"hdci"` or `"quantile"`).

`hdci_width` The HDCI width used (only meaningful when `cutoff_method = "hdci"`).

`dgp` The data-generating process used.

References

Bjorner, J. B., Kreiner, S., Ware, J. E., Damsgaard, M. T., & Bech, P. (1998). Differential item functioning in the Danish translation of the SF-36. *Journal of Clinical Epidemiology, 51*(11), 1189–1202. doi:10.1016/S08954356(98)001115

Henninger, M., Radek, J., Debelak, R., & Strobl, C. (2025). Partial credit trees meet the partial gamma coefficient for quantifying DIF and DSF in polytomous items. *Behaviormetrika, 52*, 221–257. doi:10.1007/s41237024002523

Kreiner, S. (1987). Analysis of multidimensional contingency tables by exact conditional tests: Techniques and strategies. *Scandinavian Journal of Statistics, 14*(2), 97–112.

See Also

`partgam_DIF`, `RMdifGamma`, `RMdifGammaPlot`

Examples

```
if (requireNamespace("ggdist", quietly = TRUE)) {
  set.seed(42)
  sim_data <- as.data.frame(
    matrix(sample(0:1, 200 * 10, replace = TRUE), nrow = 200, ncol = 10)
  )
  colnames(sim_data) <- paste0("Item", 1:10)
  dif_sex <- sample(c("male", "female"), 200, replace = TRUE)

  # Run 100 iterations sequentially for a quick demo
  cutoff_res <- RMdifGammaCutoff(sim_data, dif_var = dif_sex,
                                 iterations = 100, parallel = FALSE,
                                 seed = 42)
  cutoff_res$item_cutoffs

  # The permutation null needs no item parameters and is much faster
  perm_res <- RMdifGammaCutoff(sim_data, dif_var = dif_sex,
                               iterations = 100, parallel = FALSE,
                               seed = 42, dgp = "permutation")
  perm_res$item_cutoffs
}
```

RMdifGammaPlot *Plot Distribution of Simulated Partial Gamma DIF Values*

Description

Visualises the distribution of simulation-based partial gamma DIF values from RMdifGammaCutoff, optionally overlaying observed partial gamma values computed from real data with the same routine the simulation uses, which reproduces partgam_DIF exactly.

Usage

```
RMdifGammaPlot(simfit, data, dif_var)
```

Arguments

simfit	The return value of RMdifGammaCutoff (a list with components results, item_cutoffs, actual_iterations, sample_n, and item_names).
data	Optional. A data.frame or matrix of item responses for computing and overlaying observed partial gamma values. Items must be scored starting at 0 (non-negative integers). When provided, the plot includes orange diamond markers for the observed partial gamma alongside the simulated distribution, plus segment summaries from the cutoff intervals.
dif_var	Required when data is supplied. A vector (factor, character, or integer) defining group membership for the DIF analysis. Must have the same length as nrow(data).

Details

Uses ggdist::stat_dotsinterval() (when data is not supplied) or ggdist::stat_dots() (when data is supplied) with point_interval = "median_hdci" and .width = c(0.66, 0.95, 0.99).

When data is **not** supplied, the function plots the simulated partial gamma distributions as dotinterval plots using ggdist::stat_dotsinterval() with median and Highest Density Continuous Interval (HDCI) summaries.

When data **is** supplied (along with dif_var), the function:

1. Computes observed partial gamma values (identical to iarm::partgam_DIF()).
2. Overlays observed gamma values as orange diamond markers on the simulated distributions.
3. Shows per-item cutoff intervals (from simfit$item_cutoffs) as black line segments, with thicker segments for the 66\ black dots for the median.

The ggplot2 and ggdist packages must be installed (they are in Suggests, not Imports).

Value

A ggplot object.

See Also

RMdifGammaCutoff, RMdifGamma

Examples

```
if (requireNamespace("ggdist", quietly = TRUE) &&
    requireNamespace("ggplot2", quietly = TRUE)) {
  set.seed(42)
  sim_data <- as.data.frame(
    matrix(sample(0:1, 200 * 10, replace = TRUE), nrow = 200, ncol = 10)
  )
  colnames(sim_data) <- paste0("Item", 1:10)
  dif_group <- factor(sample(c("A", "B"), 200, replace = TRUE))

  # Run simulation
  cutoff_res <- RMdifGammaCutoff(sim_data, dif_var = dif_group,
                                 iterations = 100, parallel = FALSE,
                                 seed = 42)

  # Simulated distribution only
  RMdifGammaPlot(cutoff_res)

  # With observed partial gamma overlaid
  RMdifGammaPlot(cutoff_res, data = sim_data, dif_var = dif_group)
}
```

RMdifLR *DIF analysis via Andersen's likelihood-ratio test*

Description

Splits a Rasch model by an external grouping variable using `eRm::LRtest()` and reports per-group item locations (or per-group threshold locations) together with their standard errors. A single function replaces the four legacy helpers (`RIdifTableLR`, `RIdifThreshTblLR`, `RIdifFigureLR`, `RIdifThreshFigLR`) by exposing the two underlying axes – `level` (item or threshold) and `output` (data.frame, kable, or ggplot) – as arguments. The same data preparation pipeline feeds all six combinations.

Usage

```
RMdifLR(
  data,
  dif_var,
  model = c("auto", "PCM", "RM"),
  level = c("item", "threshold"),
  output = c("kable", "dataframe", "ggplot"),
  cutoff = 0.5,
```

```
  conf = 0.95,
  sort = FALSE
)
```

Arguments

data	A data.frame or matrix of item responses (non-negative integers, 0-based). One column per item, one row per person. Person IDs and grouping variables must not be included – pass the grouping variable separately via dif_var.
dif_var	Vector of length nrow(data) (factor, character, or numeric) defining the DIF grouping variable. Coerced to factor; unused levels are dropped. Rows where dif_var is NA are dropped with a message. Must result in at least 2 groups after cleaning.
model	One of "auto" (default), "PCM", or "RM". "auto" fits eRm::RM() when the data are dichotomous (max response = 1) and eRm::PCM() otherwise. "RM" errors on polytomous data.
level	One of "item" (default) or "threshold". "item" reports each item's mean threshold location per group; "threshold" reports each individual threshold per group. For dichotomous (RM) data the two views are equivalent (one threshold per item).
output	One of "kable" (default), "dataframe", or "ggplot". The data.frame view always carries a Flagged logical column; the kable view bolds flagged rows; the ggplot view shows confidence intervals.
cutoff	Numeric or NULL. Threshold (in logits) for the Flagged column: a row is flagged when MaxDiff > cutoff, where MaxDiff is the difference between the largest and smallest per-group location for that item (or threshold). Set to NULL to suppress flagging. Default 0.5.
conf	Numeric in (0, 1). Confidence level used for the ggplot error bars. Default 0.95.
sort	Logical. kable output only: sort rows by MaxDiff (descending). Default FALSE.

Details

The Partial Credit Model (PCM) is fitted by default for polytomous data and the dichotomous Rasch Model (RM) is fitted when all responses are 0/1; this can be overridden via model.

For the data.frame and kable outputs, locations are reported on the centred eRm parameterisation returned by eRm::thresholds(). Per-group fits come from eRm::LRtest(..., splitcr = dif_var); the unsplit fit (All column) is the model fitted to the full dataset. The Andersen LR statistic, df, and p-value reported as the lr_test attribute / caption come directly from LRtest()'s return value.

cell_spec()-style HTML cell colouring used in the legacy easyRasch package has been dropped in favour of a logical Flagged column (and bold rendering in the kable output), so the kable renders correctly in HTML, LaTeX, and pipe/markdown.

Value

A data.frame, a knitr_kable object, or a ggplot object, depending on output.

The data.frame has one row per item (`level = "item"`) or per item x threshold (`level = "threshold"`), with columns `Item` (and `Threshold` at threshold level), one numeric column per group level, an `All` column for the unsplit fit, `MaxDiff`, `Flagged` (when `cutoff` is non-`NULL`), and matching `SE_*` columns.

The Andersen LR test result is attached as `attr(result, "lr_test")` on the data.frame, in the kable footnote, and in the ggplot caption (LR χ^2, df, p-value).

Examples

```
if (requireNamespace("eRm", quietly = TRUE)) {
  set.seed(1)
  data("pcmdat2", package = "eRm")
  grp <- factor(sample(c("A", "B"), nrow(pcmdat2), replace = TRUE))

  # Default: kable of per-group item locations
  RMdifLR(pcmdat2, dif_var = grp)

  # ggplot panel of item locations with 95% CIs
  RMdifLR(pcmdat2, dif_var = grp, output = "ggplot")

  # Threshold-level kable, sorted by MaxDiff
  RMdifLR(pcmdat2, dif_var = grp, level = "threshold", sort = TRUE)

  # Tidy data.frame for downstream use
  df <- RMdifLR(pcmdat2, dif_var = grp, output = "dataframe")
  attr(df, "lr_test")
  df[df$Flagged, ]
}
```

`RMdifTree` *Tree-based DIF analysis with effect-size classification*

Description

Detects Differential Item Functioning (DIF) by recursively splitting the sample on one or more covariates using `psychotree::raschtree()` (dichotomous data) or `psychotree::pctree()` (polytomous data), then computes per-split effect-size measures for every item – the Mantel-Haenszel odds-ratio (Holland & Thayer, 1986), on the Delta scale developed at the Educational Testing Service (ETS; Zwick, 2012), for dichotomous data, or the partial gamma coefficient for polytomous data – and classifies them into ETS A/B/C categories (Bjorner et al., 1998). Optionally, an iterative purification step (Henninger et al., 2025) is applied, and tree stability across resamples can be assessed via `stablelearner::stabletree()` (Philipp et al., 2018).

Usage

```
RMdifTree(
  data,
```

```
  covariates,
  model = c("auto", "PCM", "RM"),
  effect_size = c("auto", "MH", "pgamma"),
  purification = c("none", "iterative"),
  p_adj = c("none", "fdr", "bonferroni"),
  thresholds = c(0.21, 0.31),
  alpha = 0.05,
  prune_negligible = FALSE,
  stability = FALSE,
  stability_B = 100L,
  stability_sampler = c("subsampling", "bootstrap"),
  min_n_per_level = 20L,
  on_rescale = c("message", "warning", "stop"),
  output = c("kable", "dataframe", "tree", "plot", "list"),
  ...
)
```

Arguments

`data`	A data.frame or matrix of item responses (non-negative integers, 0-based). One column per item. Items only – pass covariates separately via `covariates`.
`covariates`	A data.frame with `nrow(data)` rows giving the DIF covariates. Columns may be numeric (continuous), factor, ordered factor, or logical. At least one column is required.
`model`	One of `"auto"` (default), `"PCM"`, or `"RM"`. `"auto"` picks RM when the data are dichotomous (max response = 1) and PCM otherwise.
`effect_size`	One of `"auto"` (default), `"MH"` (Mantel-Haenszel), or `"pgamma"` (partial gamma). `"auto"` uses MH for RM and partial gamma for PCM.
`purification`	`"none"` (default) or `"iterative"`. Iterative purification recomputes the effect size while excluding items already classified as DIF from the matching score (Bjorner et al., 1998).
`p_adj`	Multiple-testing adjustment for the partial-gamma classification: `"none"` (default), `"fdr"` (Benjamini & Hochberg), or `"bonferroni"`. Ignored for MH (which uses the ETS test directly).
`thresholds`	Numeric length-2 vector with the partial-gamma B/C boundaries (default `c(0.21, 0.31)`). Ignored for MH (the ETS Delta-scale boundaries 1.0 / 1.5 are used).
`alpha`	Significance level for the A/C tests. Default `0.05`.
`prune_negligible`	Logical. If `TRUE`, splits whose every item is classified as A are pruned from the tree. Default `FALSE`.
`stability`	Logical. If `TRUE`, runs `stablelearner::stabletree()` on the fitted tree to assess variable-selection and cutpoint stability across resamples. Default `FALSE`.
`stability_B`	Integer. Number of resamples for stability assessment. Default `100`.
`stability_sampler`	One of `"subsampling"` (default) or `"bootstrap"`. Subsampling is recommended for tree stability because bootstrap resamples can drop levels of categorical covariates, breaking the model fit on small subsamples.

`min_n_per_level`
: Integer. Minimum count required for any factor level in `covariates`. Default `20`. Set to `0` to disable.

`on_rescale`
: One of `"message"` (default), `"warning"`, or `"stop"`. Controls how the function reports the situation in which one or more items have no responses in their lowest category within a terminal node of the tree (psychotree silently rescales those items per node, which leaves the per-node item parameters on a different metric). *The MH and partial-gamma effect sizes reported here are computed on the raw responses and are not affected.* The diagnostic concerns only the per-node item parameters that are visible via `plot(tree)`.

`output`
: One of `"kable"` (default), `"dataframe"`, `"tree"`, `"plot"`, or `"list"`. `"kable"` renders the per-split per-item effect-size table as a `knitr::kable()`; `"dataframe"` returns the underlying tidy data.frame; `"tree"` returns the augmented partykit tree object; `"plot"` returns the partykit tree plot with item names on the terminal-node x-axis (`tp_args = list(names = TRUE)`); `"list"` returns both views from a single fit, as `list(table = <data.frame>, tree = <tree object>)`. For full control over the plot, use `output = "tree"` and call `plot()` on the result with your own `tp_args`. When `stability = TRUE`, the stability summary is attached as `attr(result, "stability")` (a small data.frame) and `attr(result, "stability_kable")` (pre-rendered kable).

`...`
: Additional arguments forwarded to `psychotree::raschtree()` or `psychotree::pctree()`, e.g. `minsize`, `alpha` for the parameter-instability test. Note: this `alpha` (the tree-fitting argument) is independent of the `alpha` for ETS classification above; pass it via `....`

Details

Continuous covariates (e.g., age in years) and interactions among multiple covariates are handled natively by the model-based recursive-partitioning machinery of `partykit` / `psychotree`.

ETS Delta-scale Mantel-Haenszel. The MH common odds ratio $\hat{\alpha}_{MH}$ is mapped to the ETS Delta scale via $\Delta_{MH} = -2.35 \log \hat{\alpha}_{MH}$. The classification rules (Holland & Thayer, 1988; Zwick, 2012) are:

- Class **A**: $|\Delta_{MH}| < 1$ or test of $\Delta_{MH} = 0$ not rejected at `alpha`.
- Class **C**: $|\Delta_{MH}| \geq 1.5$ and test of $|\Delta_{MH}| \leq 1$ rejected at `alpha`.
- Class **B**: otherwise.

easyRasch2 uses the sign convention of `effecttree`: positive Δ indicates that the item is more difficult for the second (reference) group.

Partial gamma. `iarm::partgam_DIF()` provides the gamma estimate and SE per item. ETS-style classification follows Bjorner et al. (1998): A if $|\gamma| < 0.21$ or the test of $\gamma = 0$ is not rejected at `alpha`; C if $|\gamma| > 0.31$ and the test of $|\gamma| \leq 0.21$ is rejected at `alpha`; B otherwise.

Rule-of-thumb caveat. The A/B/C boundaries (Delta 1.0 / 1.5; gamma 0.21 / 0.31) are conventions carried over from large-scale educational testing, not values calibrated to the sample and items at hand – in contrast to the simulation-based cutoffs used elsewhere in this package – so the classification is best read as a rough magnitude guide rather than a calibrated test. For a sample-calibrated

partial-gamma DIF test, see `RMdifGamma()` with an `RMdifGammaCutoff()` object (optionally with `p_value = TRUE`).

Continuous and interaction effects. The recursive partitioning step automatically handles continuous covariates (the parameter-instability test searches for the optimal cutpoint) and multivariate interactions (later splits are conditional on earlier ones). To assess sensitivity of variable selection and cutpoints to the particular sample, set `stability = TRUE`.

Stability assessment. When `stability = TRUE`, the same tree-fitting call is replayed on `stability_B` resamples of the data. The returned `stabletree` object reports, per covariate, the proportion of resamples in which it was selected at any split, and (for continuous covariates) the empirical distribution of cutpoints. Stability is independent of effect-size classification – a stable split with a negligible effect is still negligible, and a large effect on an unstable split should be interpreted with care. Cost is roughly `stability_B` times the original fitting time.

Acknowledgement. The effect-size machinery – MH and partial gamma per node, iterative purification, ETS A/B/C classification – follows the implementations by Henninger & Radek in the GitHub packages *raschtreeMH* and *effecttree*. The relevant code has been adapted here under the MIT licence (see file header).

Value

Depending on `output`:

`"kable"` A `knitr_kable` object with one row per (split node x item), grouped by node via `pack_rows`-style section headers. The caption summarises model, effect-size measure, purification, and (if requested) stability.

`"dataframe"` A data.frame with one row per (split node x item): columns `NodeID`, `Split` (human-readable description of the split), `Variable`, `Direction`, `Item`, `EffectSize`, `SE`, `Class` (A/B/C), `Flagged` (TRUE for B or C), `n_left`, `n_right`.

`"tree"` The fitted partykit tree object with class `c("RMdifTree", ...)`, with effect-size results stored at `tree$info$effectsize`.

`"plot"` A plotted partykit tree.

`"list"` A list with both views from a single fit: `table` (the `"dataframe"` result, with its attributes) and `tree` (the `"tree"` result).

Stability results (when `stability = TRUE`) are attached to the return value as `attr(result, "stability")` (data.frame) and `attr(result, "stability_kable")` (pre-rendered kable).

References

Bjorner, J. B., Kreiner, S., Ware, J. E., Damsgaard, M. T., & Bech, P. (1998). Differential item functioning in the Danish translation of the SF-36. *Journal of Clinical Epidemiology, 51*(11), 1189-1202. doi:10.1016/S08954356(98)001115

Henninger, M., Debelak, R., & Strobl, C. (2023). A new stopping criterion for Rasch trees based on the Mantel-Haenszel effect size measure for DIF. *Educational and Psychological Measurement, 83*, 181-212. doi:10.1177/00131644221077135

Henninger, M., Radek, J., Debelak, R., & Strobl, C. (2025). Partial credit trees meet the partial gamma coefficient for quantifying DIF and DSF in polytomous items. *Behaviormetrika, 52*, 221-257. doi:10.1007/s41237024002523

Holland, P. W., & Thayer, D. T. (1986). Differential item performance and the Mantel-Haenszel procedure. *ETS Research Report Series*(2). doi:10.1002/j.23308516.1986.tb00186.x

Philipp, M., Rusch, T., Hornik, K., & Strobl, C. (2018). Measuring the stability of results from supervised statistical learning. *Journal of Computational and Graphical Statistics, 27*, 685-700. doi:10.1080/10618600.2018.1473779

Asamoah, N. A. B., Turner, R. C., Lo, W.-J., Crawford, B. L., & Jozkowski, K. N. (2025). Impacts of DIF Item Balance and Effect Size Incorporation With the Rasch Tree. *Educational and Psychological Measurement*. doi:10.1177/00131644251370605

Zwick, R. (2012). A review of ETS differential item functioning assessment procedures: Flagging rules, minimum sample size requirements, and criterion refinement. *ETS Research Report Series*(1). doi:10.1002/j.23338504.2012.tb02290.x

See Also

RMdifLR, RMdifGamma

Examples

```
if (requireNamespace("psychotree", quietly = TRUE) &&
    requireNamespace("partykit", quietly = TRUE) &&
    requireNamespace("difR", quietly = TRUE) &&
    requireNamespace("iarm", quietly = TRUE)) {
  data("DIFSimPC", package = "psychotree")

  items <- as.data.frame(as.matrix(DIFSimPC$resp))
  covs  <- DIFSimPC[, c("age", "gender", "motivation")]

  # Default: kable of per-split effect sizes
  RMdifTree(items, covariates = covs)

  # Tidy data.frame -- one row per (split node x item)
  df <- RMdifTree(items, covariates = covs, output = "dataframe")
  df[df$Flagged, ]

  # Tree object (for plotting via partykit::plot)
  tree <- RMdifTree(items, covariates = covs, output = "tree")
  plot(tree)

  # Stability assessment refits the tree on B resamples.
  # (use more resamples, e.g. 100+, in real analyses)
  if (requireNamespace("stablelearner", quietly = TRUE)) {
    kbl <- RMdifTree(items, covariates = covs,
                     purification = "iterative", p_adj = "fdr",
                     stability = TRUE, stability_B = 25)
    kbl                           # main effect-size kable
    attr(kbl, "stability_kable") # pre-rendered stability kable
    attr(kbl, "stability")       # raw stability data.frame
  }
}
```

RMdimCFA *Observed one-factor CFA fit and loadings vs a simulated reference*

Description

Fits the observed one-factor categorical CFA to `data` and compares its fit indices and per-item standardized loadings against the simulated null distribution produced by `RMdimCFACutoff`. Returns a list of two tables: model-fit indices and per-item loadings, each with the observed value, the expected reference from the simulation, and a flag.

Usage

```
RMdimCFA(
  data,
  cutoff,
  p_value = FALSE,
  correction = c("fwer", "fdr_bh", "fdr_by", "none"),
  alpha = 0.05,
  output = c("kable", "dataframe")
)
```

Arguments

`data`	A data.frame or matrix of item responses (non-negative integers, 0-based), the same items used for the cutoff simulation.
`cutoff`	The list returned by `RMdimCFACutoff`. Required: observed CFA fit indices are not interpretable without the simulated reference, so the function errors if it is missing.
`p_value`	Logical. When `TRUE`, adds bootstrap p-values from the simulated null distributions: one-sided in the unfavourable direction for the fit indices (CFI low; RMSEA / SRMR high), two-sided for the per-item loadings. The `Flagged` columns then reflect `padj < alpha` instead of the percentile cutoffs. The fit indices and the loadings are corrected as two separate families. Default `FALSE`.
`correction`	Character. Multiplicity correction for the p-values: `"fwer"` (default; Westfall-Young studentised-max step-down), `"fdr_bh"` (Benjamini-Hochberg), `"fdr_by"` (Benjamini-Yekutieli), or `"none"`. Ignored when `p_value = FALSE`.
`alpha`	Numeric in (0, 1). Significance level used to flag comparisons on the corrected p-value. Default `0.05`. Ignored when `p_value = FALSE`.
`output`	Character. `"kable"` (default) returns each table as a `knitr::kable()`; `"dataframe"` returns plain data.frames.

Details

Bootstrap p-values. The per-comparison statistic is the residual studentised by the bootstrap mean and SD. Marginal p-values are Monte-Carlo, `(1 + count) / (B + 1)`, so they can be no smaller than `1 / (B + 1)`. `correction = "fwer"` uses the Westfall-Young studentised-max step-down, which

exploits the bootstrap dependence among the statistics (Ferreira, 2024); it is liberal when the simulation is small, so at least 1000 `iterations` in `RMdimCFACutoff()` are recommended (a warning is issued below that). These p-values are model-conditional and sample-size-sensitive and are reported alongside the simulated expected ranges, not in place of them.

Value

A named list with two elements, `fit` and `loadings`:

`fit` CFI / RMSEA / SRMR with columns `Index`, `Observed`, `Cutoff`, `Direction`, `Flagged` (one-sided, in the unfavourable direction). With `p_value = TRUE`, columns `p` and `padj` are added and `Flagged` reflects `padj < alpha`.

`loadings` One row per item with columns `Item`, `Observed`, `Expected_low`, `Expected_high`, `Flagged` (`"below"` / `"above"` / `""`). With `p_value = TRUE`, columns `p_loading` and `padj_loading` are added and `Flagged` reflects `padj_loading < alpha` (direction from the sign of the deviation from the simulated mean).

Each element is a `knitr_kable` (when `output = "kable"`) or a data.frame (when `output = "dataframe"`).

Multiple comparisons

The marginal p-value controls the error rate of a *single* comparison: for one item (or item pair) decided on in advance it is the relevant value. But scanning all k comparisons and flagging whichever fall below `alpha` tests k hypotheses at once, so the chance of at least one false flag inflates to roughly $1-(1-\alpha)^k$ (e.g. about 34% for $k = 8$ at `alpha = 0.05`) – even when every marginal p-value is correctly calibrated. The corrected (adjusted) p-value controls this: `correction = "fwer"` bounds the probability of *any* false flag (strict, lower power), while `"fdr_bh"` / `"fdr_by"` bound the expected *proportion* of false flags among those raised (a more lenient middle ground). Rule of thumb: use the marginal p-value for a single pre-specified comparison, and a corrected p-value when screening the whole table – the usual workflow.

References

Ferreira, J. A. (2024). Methods of testing a 'small' or 'moderate' number of hypotheses simultaneously. *Journal of Statistical Theory and Practice, 19*(6). doi:10.1007/s42519024004124

Westfall, P. H., & Young, S. S. (1993). *Resampling-Based Multiple Testing*. Wiley.

See Also

`RMdimCFACutoff`, `RMdimCFAPlot`

Examples

```
if (requireNamespace("lavaan", quietly = TRUE) &&
    requireNamespace("eRm", quietly = TRUE)) {
  data("raschdat1", package = "eRm")
  sim <- RMdimCFACutoff(raschdat1[, 1:8], iterations = 50,
                        parallel = FALSE, seed = 1)
  tabs <- RMdimCFA(raschdat1[, 1:8], cutoff = sim)
  tabs$fit
```

```
  tabs$loadings

  # Bootstrap p-values with family-wise (Westfall-Young) correction
  # (use iterations >= 1000 in real analyses for stable p-values)
  RMdimCFA(raschdat1[, 1:8], cutoff = sim, p_value = TRUE)
}
```

`RMdimCFACutoff` *Simulated null distribution for one-factor CFA fit and loadings under PCM unidimensionality*

Description

Generates a parametric-bootstrap null distribution against which observed one-factor categorical-CFA results can be compared. The simulation draws `iterations` datasets from the fitted PCM (or RM, for dichotomous data) using the observed item parameters and a resampled person distribution; each simulated dataset is fitted with `lavaan::cfa(..., ordered = TRUE, estimator = "WLSMV")` and both the three fit indices (CFI, RMSEA, SRMR) and the per-item standardized factor loadings are recorded. Because the simulated data satisfy the PCM unidimensional assumption by construction, the resulting distributions are the "expected" reference for what a correctly fitting unidimensional model produces at this sample size and item structure.

Usage

```
RMdimCFACutoff(
  data,
  iterations = 250L,
  percentile = 99,
  output = c("list", "kable"),
  parallel = TRUE,
  n_cores = NULL,
  verbose = FALSE,
  seed = NULL,
  estimator = "WLSMV"
)
```

Arguments

`data`	A data.frame or matrix of item responses (non-negative integers, 0-based). One column per item, one row per person.
`iterations`	Integer. Number of parametric-bootstrap iterations. Default `250`.
`percentile`	Numeric in (50, 100). The strictness of the cutoffs. Default `99`. Fit indices use a one-sided cutoff in the unfavourable direction (CFI from below; RMSEA and SRMR from above); standardized loadings use a two-sided central interval covering `percentile`% of the simulated distribution (an item is flagged when

	its observed loading falls in the outer `100 - percentile`%, split across the two tails).
`output`	Character. Only `"list"` (the default) is supported and the function always returns the simulation object. `"kable"` is retained only to raise an informative error: tables now come from `RMdimCFA`.
`parallel`	Logical. If `TRUE` (default), uses parallel processing via `mirai`. Falls back to sequential if `mirai` is not installed or `n_cores` cannot be resolved.
`n_cores`	Integer or `NULL`. Number of parallel workers. When `NULL`, `getOption("mc.cores")` is consulted; if neither is set, sequential is used.
`verbose`	Logical. Show a progress bar (default `FALSE`).
`seed`	Integer or `NULL`. Master seed for reproducibility. See easyRasch2-reproducibility for what this guarantees and how it interacts with `parallel`.
`estimator`	Character. The lavaan estimator passed to `lavaan::cfa()`. Default `"WLSMV"`. Other limited-information estimators that produce robust/scaled fit indices (e.g., `"DWLS"`, `"ULSMV"`) are also accepted; full-information ML is rejected (incompatible with `ordered = TRUE`).

Details

This function only generates the simulated reference. To obtain the observed-vs-expected tables, pass its result to `RMdimCFA`; for the figures, pass it to `RMdimCFAPlot`.

Generative model. The data-generating process for each simulated dataset is the PCM (or RM) fitted to the observed data, with persons drawn from the empirical theta distribution (resampled with replacement). This means the simulated data perfectly satisfy the PCM unidimensional assumption.

Estimation model. The CFA on each simulated dataset uses a single-factor model with all items as ordinal indicators (`F1 =~ I1 + I2 + ...`), fitted with `WLSMV` by default. Reported CFI / RMSEA are the Satorra-Bentler-scaled variants (`cfi.scaled`, `rmsea.scaled`) for consistency across iterations; SRMR is reported unchanged. Standardized loadings are the `est.std` of the `=~` paths from `lavaan::standardizedSolution()`.

Why a null distribution. A perfectly PCM-unidimensional dataset will typically not yield CFA fit indices at their ideal values (CFI = 1, RMSEA = 0), nor identical loadings across items: PCM uses a logistic threshold structure while WLSMV uses a probit link via the polychoric correlation matrix, and finite samples add sampling variability. The simulated distributions capture both, giving a more honest reference than rule-of-thumb cutoffs derived under continuous-data ML.

Iteration failures. Some simulated datasets cause WLSMV to fail (non-positive-definite polychoric matrix, boundary thresholds, empty categories). Failed iterations are dropped; `actual_iterations` reflects the number that succeeded.

Value

A list (the simulation object), with components:

`simulated` data.frame with one row per successful iteration and columns `iteration`, `cfi`, `rmsea`, `srmr`.

`simulated_loadings` data.frame with one row per successful iteration: an `iteration` column followed by one column per item holding the simulated standardized loading.

`percentile` Numeric: the strictness setting used.

`cutoffs` Named numeric vector (`cfi`, `rmsea`, `srmr`) of one-sided fit-index cutoffs at the chosen percentile.

`loading_cutoffs` data.frame `Item`, `low`, `high` — the two-sided expected loading interval per item.

`actual_iterations` Number of successful MC iterations.

`sample_n` Number of complete cases used.

`sample_n_total` Number of respondents in the raw input data, before the complete-case filter.

`sample_has_na` Logical. Whether the raw input data contained any missing values.

`n_items` Number of items.

`item_names` Character vector of item names.

`is_polytomous` Logical: was a PCM (vs RM) fitted?

`estimator` The lavaan estimator used.

References

Yuan, K.-H., & Bentler, P. M. (2000). Three likelihood-based methods for mean and covariance structure analysis with nonnormal missing data. *Sociological Methodology, 30*(1), 165-200. doi:10.1111/00811750.00078

Rosseel, Y. (2012). lavaan: An R Package for Structural Equation Modeling. *Journal of Statistical Software, 48*(2), 1-36. doi:10.18637/jss.v048.i02

See Also

RMdimCFA, RMdimCFAPlot, RMdimResidualPCA, RMdimMartinLof

Examples

```
if (requireNamespace("lavaan", quietly = TRUE) &&
    requireNamespace("eRm", quietly = TRUE)) {
  data("raschdat1", package = "eRm")

  # Few iterations for a fast example; use 250+ in real analyses
  sim <- RMdimCFACutoff(raschdat1[, 1:8], iterations = 50,
                        parallel = FALSE, seed = 1)

  # Observed-vs-expected tables
  RMdimCFA(raschdat1[, 1:8], cutoff = sim)

  if (requireNamespace("ggplot2", quietly = TRUE)) {
    plots <- RMdimCFAPlot(sim, data = raschdat1[, 1:8])
    plots$loadings
    plots$fit
  }
}
```

RMdimCFAPlot *Plot observed CFA fit and loadings against the simulated null*

Description

Returns two figures (in a list) comparing the observed one-factor CFA to the simulated null distribution from RMdimCFACutoff: a per-item standardized-loadings plot (observed marker against each item's simulated distribution and expected range, in the style of RMitemInfitPlot), and a faceted plot of the CFI / RMSEA / SRMR distributions with the observed value overlaid.

Usage

```
RMdimCFAPlot(simfit, data, percentile = NULL)
```

Arguments

simfit	The list returned by RMdimCFACutoff.
data	The item-response data the CFA was run on (the same items used for the cutoff). Required: the observed values are computed from it.
percentile	Numeric in (50, 100) or NULL. When supplied, the cutoffs and flags are recomputed at this percentile from the stored simulated distributions (no re-simulation). When NULL (default), the percentile from the original RMdimCFACutoff() call is reused.

Value

A named list of two ggplot objects:

loadings Per-item standardized loadings: simulated distribution (dots), expected interval, and the observed loading as a diamond (red when flagged).

fit Faceted CFI / RMSEA / SRMR simulated distributions with the observed value and cutoff overlaid.

See Also

RMdimCFACutoff, RMdimCFA

Examples

```
if (requireNamespace("lavaan", quietly = TRUE) &&
    requireNamespace("ggplot2", quietly = TRUE) &&
    requireNamespace("eRm", quietly = TRUE)) {
  data("raschdat1", package = "eRm")
  sim <- RMdimCFACutoff(raschdat1[, 1:8], iterations = 50,
                        parallel = FALSE, seed = 1)
  plots <- RMdimCFAPlot(sim, data = raschdat1[, 1:8])
  plots$loadings
  plots$fit
```

```
}
```

`RMdimMartinLof` *Martin-Löf Test of Unidimensionality*

Description

Likelihood-ratio test of unidimensionality against an *a priori* specified multidimensional alternative, generalised to polytomous Rasch / partial credit models (Christensen, Bjorner, Kreiner, & Petersen, 2002). The p-value is obtained by parametric-bootstrap (Monte Carlo) sampling under the unidimensional null, following Christensen & Kreiner (2007), because the asymptotic chi-square approximation is biased toward conservatism for realistic sample sizes – especially with polytomous items, where the degrees of freedom can be very large.

Usage

```
RMdimMartinLof(
  data,
  partition,
  iterations = 1000L,
  stopping = c("none", "sequential"),
  h = 50L,
  alpha = 0.05,
  parallel = TRUE,
  n_cores = NULL,
  verbose = FALSE,
  seed = NULL
)
```

Arguments

`data`	A data.frame or matrix of item responses (0-based, non-negative integers). Complete cases only: rows with any `NA` on the items named in `partition` are dropped. Missingness on items outside `partition` is ignored.
`partition`	The hypothesised partition of items into subscales. One of: • a **list** of column-name or column-index vectors, e.g. `list(c("I1","I2","I3"), c("I4","I5","I6"))`; • a **vector** of length `ncol(data)` indicating each item's subscale (factor, character, or integer), e.g. `c(1,1,1,2,2,2)`. Each subscale must contain at least two items. Subscales must not overlap; items not assigned to any subscale are dropped with a warning.
`iterations`	Integer. Maximum number of Monte Carlo iterations (default `1000`).

`stopping`	Character. `"none"` (default) runs all iterations. `"sequential"` uses Besag & Clifford's (1991) sequential rule: stop as soon as `h` simulated statistics have exceeded the observed value. The sequential strategy substantially reduces compute time when H0 holds but cannot be parallelised.
`h`	Integer. Sequential-stopping count threshold (default `50`). Ignored when `stopping = "none"`.
`alpha`	Numeric in (0, 1). Nominal significance level used only for the `rejected` flag in the result; default `0.05`.
`parallel`	Logical. Use parallel processing via `mirai` (default `TRUE`). Ignored when `stopping = "sequential"`.
`n_cores`	Integer or `NULL`. Number of parallel workers. When `NULL`, `getOption("mc.cores")` is checked first; if neither is set, falls back to sequential with a warning.
`verbose`	Logical. Show a progress bar (default `FALSE`).
`seed`	Integer or `NULL`. Random seed for reproducibility. Items are processed internally in a fixed (alphabetical) order, so the same seed reproduces the same p-value regardless of how the data's columns are arranged and regardless of the `parallel` setting. See easyRasch2-reproducibility for what this guarantees and how it interacts with `parallel`.

Details

This is **not** a routine screening tool. The test requires an *a priori* partition of items into subscales; using it post-hoc on, e.g., the partition suggested by `RMdimResidualPCA()`'s PC1 sign would inflate the Type-I error rate. Both source papers state this explicitly.

Test statistic. With items partitioned into D subscales, total score t and subscores $(t_1, \ldots, t_D)$ (Christensen et al. 2002, eq. 22):

$$T = 2\Big[\sum_{t_1,\ldots,t_D} n_{t_1,\ldots,t_D} \log(n_{t_1,\ldots,t_D}/N) - \sum_t n_t \log(n_t/N) - \ell_C(\hat{\epsilon}) + \sum_d \ell_C(\hat{\epsilon}^{(d)})\Big]$$

where ℓ_C is the conditional log-likelihood and the $\hat{\epsilon}^{(d)}$ are CML estimates on the d-th subscale alone. CML fits use `psychotools::raschmodel()` (RM) or `psychotools::pcmodel()` (PCM) for speed.

Monte Carlo sampling under H0. Following Christensen & Kreiner (2007): (a) sample N total scores from the empirical score distribution n_t/N; (b) for each sampled score, sample an item-response vector from the conditional distribution $p(x \mid t, \hat{\epsilon})$ given by eq. 4 of the paper. Step (b) uses the recursive γ-function algorithm (Andersen, 1995, eq. 15.22 and 15.27) for dichotomous and polytomous items alike: items are filled in from the last backwards, each conditional on the score still to be distributed over the items before it. The nested $\gamma^{(1)}, \ldots, \gamma^{(n)}$ depend only on the item parameters, which are fixed across the run, so the recursion is evaluated once rather than per person.

The faster dichotomous shortcut described by Christensen & Kreiner (2007, p. 23), choosing s items one at a time with probabilities $\phi_i / \sum_{\text{remaining}} \phi$, is deliberately *not* used: successive sampling of that kind follows a Wallenius-type distribution, whereas the Rasch conditional distribution weights a set of items by the *product* of their easinesses. See the note on validation below.

Iterations that fail (e.g., simulated dataset has an empty category for an item) are silently dropped.

Item parameters are estimated once on the observed data and held fixed across MC iterations. Christensen & Kreiner (2007) use the extended likelihood function (Tjur, 1982) with the empirical score distribution as a non-parametric estimate of the latent distribution, so no distributional assumption about θ is needed.

Missing data. Complete cases only. The statistic is built from the joint table of subscores, so a respondent needs a defined subscore on every subscale; there is no partial-missingness path, and the SAS macro this implementation was validated against has the same requirement. Rows with `NA` are dropped *after* items outside `partition` have been removed, so missingness on items the test does not use costs no respondents. `sample_n` reports the respondents analysed and `sample_n_total` the raw input rows.

Validation. The statistic, the conditional log-likelihoods, the conditional sampler and the expected counts and residuals of `RMdimMartinLofResiduals` have been checked against the `pml` SAS macro (Christensen, 2004), kindly shared by Karl Bang Christensen. The statistic and the residual table agree with the macro to numerical precision. The sampler follows the macro's recursion rather than the dichotomous shortcut published in Christensen & Kreiner (2007).

Value

A list with components:

`T_obs` Observed Martin-Löf likelihood-ratio statistic.

`p_value` Monte Carlo p-value with `(n_exceed + 1) / (n + 1)` correction. The attainable p-values are `k / (n + 1)` for `k = 1, ..., n + 1`, so the p-value's resolution is limited by the number of iterations: with 100 iterations the smallest attainable value is `1/101 = 0.0099`. A reported p-value *equal to the floor* (see `p_value_floor`) means no simulated statistic reached the observed one and should be read as "p < floor" – the true p-value may be much smaller; increase `iterations` for finer resolution.

`p_value_floor` The smallest attainable p-value, `1 / (actual_iterations + 1)`.

`actual_iterations` Number of successful MC iterations completed.

`rejected` Logical: is `p_value < alpha`?

`partition` Normalised partition (list of integer indices).

`n_subscales` Number of subscales.

`is_polytomous` Whether a PCM was fitted.

`sample_n` Number of complete cases analysed, counted over the items in `partition` only.

`sample_n_total` Number of respondents in the raw input data, before the complete-case filter.

`sample_has_na` Logical. Whether the partitioned items contained any missing values.

`n_items` Number of items.

`stopping` The stopping strategy used.

`h` The sequential-stopping count, or `NA` for `stopping = "none"`.

`T_rep` Numeric vector of successful MC test statistics.

`wle_scores` data.frame with one row per person and one column per subscale (`subscale_1_wle`, ..., `subscale_D_wle`), giving Warm's Weighted Likelihood Estimate of theta from a CML fit on each subscale alone. Persons whose subscore equals the minimum or maximum on a subscale produce non-finite WLEs (`Inf` / `-Inf`) and are excluded from `wle_correlation` pairwise.

`wle_correlation` data.frame of pairwise Pearson correlations between subscale WLEs, with columns `subscale_a`, `subscale_b`, `r`, `ci_lower`, `ci_upper` (95% CI from `stats::cor.test`), `p_value`, and `n` (number of persons with finite WLEs on both subscales). One row per pair; for D = 2, a single row. Useful as an effect-size companion to `p_value` – a rejected test with `r` near 1 indicates a small effect; `r` clearly below 1 indicates substantive multidimensionality.

References

Andersen, E. B. (1995). Polytomous Rasch models and their estimation. In G. H. Fischer & I. W. Molenaar (Eds.), *Rasch models: Foundations, recent developments, and applications* (pp. 271-291). Springer-Verlag.

Besag, J., & Clifford, P. (1991). Sequential Monte Carlo p-values. *Biometrika, 78*(2), 301-304. doi:10.1093/biomet/78.2.301

Christensen, K. B. (2004). *pml: A SAS macro for testing unidimensionality in polytomous Rasch models* (Technical note). National Institute of Occupational Health, Denmark, and Department of Biostatistics, University of Copenhagen.

Christensen, K. B., Bjorner, J. B., Kreiner, S., & Petersen, J. H. (2002). Testing unidimensionality in polytomous Rasch models. *Psychometrika, 67*(4), 563-574. doi:10.1007/BF02295132

Christensen, K. B., & Kreiner, S. (2007). A Monte Carlo approach to unidimensionality testing in polytomous Rasch models. *Applied Psychological Measurement, 31*(1), 20-30. doi:10.1177/0146621605286204

See Also

`RMdimResidualPCA`, `RMdimResidualPCACutoff`

Examples

```
set.seed(1)
# Build 2-dimensional polytomous data: 4 items per subscale, 5 categories
n     <- 400
theta1 <- rnorm(n)
theta2 <- 0.6 * theta1 + sqrt(1 - 0.6^2) * rnorm(n)
make_pcm <- function(theta, n_items, taus) {
  sapply(seq_len(n_items), function(j) {
    # ... toy simulation here
    sample(0:4, n, replace = TRUE)
  })
}
dat <- cbind(make_pcm(theta1, 4, NULL), make_pcm(theta2, 4, NULL))
colnames(dat) <- paste0("I", 1:8)

# Few iterations for a fast example; use 1000+ in real analyses
RMdimMartinLof(dat,
            partition = list(c("I1","I2","I3","I4"),
                             c("I5","I6","I7","I8")),
            iterations = 100, parallel = FALSE, seed = 1)

# Sequential stopping: stop as soon as h = 25 simulated statistics exceed
# the observed one (cuts compute time under H0).
```

```
RMdimMartinLof(dat,
            partition = c(1,1,1,1,2,2,2,2),
            iterations = 200, stopping = "sequential", h = 25,
            seed = 1)
```

`RMdimMartinLofResiduals` *Standardised Residuals from the Joint Subscore Distribution*

Description

Diagnostic accompanying `RMdimMartinLof`: per-cell standardised residuals from the joint distribution of subscores under unidimensionality (Christensen, Bjorner, Kreiner, & Petersen, 2002, eq. 13). Useful for identifying *where* a partition deviates from the unidimensional null rather than just whether it does (which `RMdimMartinLof()` answers).

Usage

```
RMdimMartinLofResiduals(
  data,
  partition,
  output = c("kable", "dataframe", "ggplot"),
  flag_threshold = 2,
  color_by = c("residual", "n"),
  color_limits = NULL,
  min_expected = NULL
)
```

Arguments

`data`	A data.frame or matrix of item responses (0-based, non-negative integers). Complete cases only: rows with any `NA` on the items named in `partition` are dropped. Missingness on items outside `partition` is ignored.
`partition`	Same format as in `RMdimMartinLof`: a list of item-name/index vectors, or a length-`ncol(data)` vector of group labels. Each subscale must contain at least 2 items.
`output`	Character. `"kable"` (default) for a 2-D pipe-format residual table when D = 2 (long-format kable when D > 2), `"dataframe"` for the underlying long-format data.frame, or `"ggplot"` for a diverging-fill heatmap (with `facet_wrap` over `t3` for D = 3, error for D > 3).
`flag_threshold`	Numeric. Cells with `|residual| > flag_threshold` are flagged: marked as `**bold**` in the kable, shown in the `flagged` column of the dataframe. Default 2.

`color_by` Character. For `output = "ggplot"`, what the tile fill colour encodes. `"residual"` (default) – diverging red-white-blue scale centred at 0, the most directly diagnostic. `"n"` – sequential blue scale on the observed cell count, useful for spotting whether large-magnitude residuals are driven by sparse cells. Either way, the numeric residual is printed inside each cell.

`color_limits` Numeric length-2 vector or `NULL`. Caps the colour scale for `output = "ggplot"`. Default `c(-5, 5)` when `color_by = "residual"` (sparse-cell residuals from `(o-e)/sqrt(n*p*(1-p))` can be enormous when expected counts are tiny; capping the scale stops outliers from compressing the rest of the plot). The unclipped residual values still appear as cell labels. `NULL` for `color_by = "n"`, which uses the natural data range.

`min_expected` Numeric or `NULL`. If set, cells with `expected < min_expected` have their residual set to `NA` (they appear grey in the heatmap and are not flagged). Default `NULL` (no filtering). Setting `min_expected = 1` (or `5`) is the analogue of the Cochran rule for sparse-cell chi-square contributions and removes residuals whose asymptotic standard normal approximation is unreliable.

Details

For each cell of the joint subscore table (indexed by $(t_1, \ldots, t_D)$), the conditional probability under H0 given the total score $t = \sum_d t_d$ is

$$p(t_1, \ldots, t_D \mid t) = \prod_d \gamma_{t_d}^{(d)} / \gamma_t,$$

the expected count is $e = n_t \cdot p$, and the residual is $(o - e)/\sqrt{n_t \cdot p \cdot (1 - p)}$. CML estimates from the unidimensional model are used for the γ-functions.

Reading the table (D = 2): under the unidimensional null, residuals should be patternless and roughly N(0, 1). Multidimensionality with positively correlated dimensions typically shows up as **positive** residuals at the corners of each antidiagonal (high `t1` + low `t2`, low `t1` + high `t2`) and **negative** residuals near the antidiagonal centre (matched subscores). Negatively correlated dimensions show positive residuals at the table corners (high/low and low/high) and negative residuals at high/high and low/low. See Christensen et al. (2002, section7) for a worked example.

Cells where the total score has no observed cases (`n_t = 0`) are uninformative and are dropped from the output.

Complete cases only, as in `RMdimMartinLof`, and rows with `NA` are dropped only after items outside `partition` have been removed. The reported `n = X of Y respondents` counts respondents complete on the partitioned items against the raw input rows.

Expected counts and residuals agree to numerical precision with the `pml` SAS macro (Christensen, 2004), kindly shared by Karl Bang Christensen. The one deliberate difference is at the extreme totals `t = 0` and `t = max`, where the conditional probability is 1 and the residual is undefined: the macro prints 0, this function returns `NA`.

Value

- `output = "kable"`: a `knitr_kable` object. For D = 2, a wide table with rows = `t1`, columns = `t2`, cells = standardised residual (`**bold**` if flagged, em-dash if NA). For D > 2, long-format with one row per cell.

- `output = "dataframe"`: a long-format data.frame with columns `t1`, ..., `tD`, `total`, `observed`, `expected`, `residual`, `flagged`.
- `output = "ggplot"`: a `geom_tile()` heatmap (D = 2 or 3 only).

References

Christensen, K. B. (2004). *pml: A SAS macro for testing unidimensionality in polytomous Rasch models* (Technical note). National Institute of Occupational Health, Denmark, and Department of Biostatistics, University of Copenhagen.

Christensen, K. B., Bjorner, J. B., Kreiner, S., & Petersen, J. H. (2002). Testing unidimensionality in polytomous Rasch models. *Psychometrika, 67*(4), 563-574. doi:10.1007/BF02295132

See Also

RMdimMartinLof

Examples

```
set.seed(1)
dat <- as.data.frame(matrix(sample(0:1, 400 * 8, replace = TRUE),
                            nrow = 400, ncol = 8))
colnames(dat) <- paste0("I", 1:8)

# Wide kable table for D = 2
RMdimMartinLofResiduals(dat,
                    partition = list(c("I1","I2","I3","I4"),
                                     c("I5","I6","I7","I8")))

# Heatmap
if (requireNamespace("ggplot2", quietly = TRUE)) {
  RMdimMartinLofResiduals(dat,
                       partition = c(1,1,1,1,2,2,2,2),
                       output = "ggplot")
}

# Underlying data.frame for custom analysis
df <- RMdimMartinLofResiduals(dat,
                           partition = c(1,1,1,1,2,2,2,2),
                           output = "dataframe")
df[df$flagged, ]
```

`RMdimResidualPCA` *PCA of Standardized Rasch Residuals*

Description

Fits a Rasch model by CML via `psychotools` (a dichotomous item is a 2-category PCM), extracts the standardized residuals $(x - E)/\sqrt{Var}$ at WLE person locations, and runs an unrotated principal-component analysis on those residuals via `stats::prcomp()`. The function reports the top `n_components` eigenvalues and their proportions of unexplained variance, and optionally compares the first-contrast eigenvalue against a simulation-based bound from `RMdimResidualPCACutoff`.

Usage

```
RMdimResidualPCA(
  data,
  cutoff = NULL,
  p_value = FALSE,
  n_components = 5L,
  output = "kable"
)
```

Arguments

`data`	A data.frame or matrix of item responses. Items must be scored starting at 0 (non-negative integers). Rows with any `NA` are dropped before PCA, since `prcomp()` does not accept missing values.
`cutoff`	Optional. The list returned by `RMdimResidualPCACutoff` (its `suggested_cutoff` is used), or a single numeric value to use as the cutoff directly. When provided, the result includes a `Flagged` column (logical: is the eigenvalue above the simulated bound?) and the kable caption notes the cutoff.
`p_value`	Logical. When `TRUE`, adds a one-sided bootstrap p-value for the first-contrast eigenvalue: the proportion of simulated first-contrast eigenvalues at least as large as the observed one, `(1 + #\{lambda* >= lambda\}) / (B + 1)`. Requires the **full** `RMdimResidualPCACutoff` object as `cutoff` (it carries the simulated eigenvalues in `$results`); a bare numeric cutoff is not sufficient. The simulated null is the distribution of the *largest* eigenvalue, so the p-value applies to PC1 only (`NA` for the other components). This is a single test — no multiplicity correction is involved. Default `FALSE`.
`n_components`	Integer. Number of eigenvalues to report. Capped at the number of items. Default 5.
`output`	Character. `"kable"` (default) for a formatted `knitr::kable()` table, `"dataframe"` for the underlying data.frame, or `"ggplot"` for a ggplot of PC1 loadings against item locations. `"loadings"` is accepted as a backward-compatible alias for `"ggplot"`.

Details

Rule-of-thumb thresholds for the first-contrast eigenvalue (e.g., the "> 2" heuristic occasionally cited from Winsteps documentation) are not reliable indicators of multidimensionality; the first-contrast eigenvalue under a correctly fitting unidimensional model varies systematically with sample size, test length, and item-parameter spread. Empirical (simulated) bounds tailored to the data

structure should be used instead — see RMdimResidualPCACutoff, and Chou & Wang (2010) for the underlying simulation argument.

The PCA is performed on the standardized residuals $(x - E)/\sqrt{Var}$ from the shared CML/WLE engine (CML item parameters via psychotools, WLE person locations). The reported eigenvalues are unrotated; rotation is appropriate for *interpreting* a multidimensional solution but obscures the dominant first contrast that dimensionality assessment is concerned with.

Item locations on the loadings plot are the per-item mean of the CML Andrich thresholds.

The variance partition follows Linacre's convention: per-item observed variance is compared to per-item *expected* variance under the fitted model, summed across items. Expected scores are computed from the CML item parameters and WLE person locations. WLE is finite at extreme scores, so all persons are retained (the previous MLE partition dropped extreme-score cases).

Bootstrap p-value. When p_value = TRUE, the observed first-contrast eigenvalue is compared against the simulated null distribution of largest eigenvalues (from cutoff$results), giving the one-sided Monte-Carlo p-value (1 + #\{lambda* >= lambda\}) / (B + 1). Because the maximum eigenvalue is a single family-wise statistic, no multiplicity correction applies. The p-value is model-conditional and sample-size-sensitive; it is reported alongside the simulated cutoff, not in place of it, and can be no smaller than 1 / (B + 1).

Value

- If output = "kable": a knitr_kable object with columns Component, Eigenvalue, Proportion of variance (and Flagged when cutoff is provided; p when p_value = TRUE). The caption gives the variance partition (% of total observed variance explained by measures vs. unexplained), the model fitted, sample size, and cutoff/p-value metadata if applicable.
- If output = "dataframe": a data.frame with columns Component, Eigenvalue, Proportion_of_variance (and Flagged when cutoff is provided; p when p_value = TRUE, non-NA for PC1 only). The variance partition is attached as the "variance_partition" attribute — a list with elements total, explained, unexplained, pct_explained, pct_unexplained, n_persons. Access via attr(result, "variance_partition").
- If output = "ggplot": a ggplot showing each item's PC1 loading on the x-axis and Rasch item location on the y-axis, with dashed reference lines at zero, and the variance partition in the figure caption. Item names are labelled via ggrepel::geom_text_repel() when ggrepel is installed; otherwise plain geom_text().

References

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. doi:10.1177/0013164410379322

See Also

RMdimResidualPCACutoff

Examples

```
set.seed(1)
dat <- as.data.frame(
```

```
  matrix(sample(0:1, 200 * 12, replace = TRUE), nrow = 200, ncol = 12)
)
colnames(dat) <- paste0("I", 1:12)

# Default kable output
RMdimResidualPCA(dat)

# PC1 loadings vs item location plot
if (requireNamespace("ggplot2", quietly = TRUE) &&
    requireNamespace("ggrepel", quietly = TRUE)) {
  RMdimResidualPCA(dat, output = "ggplot")
}

# Simulation-based cutoff (use 250+ iterations in real analyses)
bound <- RMdimResidualPCACutoff(dat, iterations = 50, parallel = FALSE, seed = 1)
RMdimResidualPCA(dat, cutoff = bound)

# With the one-sided bootstrap p-value for the first contrast
RMdimResidualPCA(dat, cutoff = bound, p_value = TRUE)
```

`RMdimResidualPCACutoff` *Simulation-based Cutoff for First-Contrast Eigenvalue*

Description

Parametric bootstrap producing an empirical upper percentile for the largest eigenvalue from a PCA of standardized residuals under a correctly fitting Rasch model. For each iteration the function resamples theta values from the data-based estimates, simulates response data, refits the appropriate Rasch model, and extracts the first-contrast eigenvalue. Several upper-tail percentiles of the resulting distribution are returned; the 99th percentile is reported as the suggested cutoff (matching the convention used by `RMlocdepQ3Cutoff`).

Usage

```
RMdimResidualPCACutoff(
  data,
  iterations = 250,
  parallel = TRUE,
  n_cores = NULL,
  verbose = FALSE,
  seed = NULL
)
```

Arguments

`data`	A data.frame or matrix of item responses (0-based, non-negative integers).

iterations	Integer. Number of simulation iterations. Default 250.
parallel	Logical. Use parallel processing via mirai. Default TRUE.
n_cores	Integer or NULL. Number of parallel workers. When NULL, getOption("mc.cores") is checked first; if neither is set, parallel = TRUE falls back to sequential with a warning.
verbose	Logical. Show a progress bar. Default FALSE.
seed	Integer or NULL. Random seed for reproducibility. See easyRasch2-reproducibility for what this guarantees and how it interacts with parallel.

Details

Rule-of-thumb cutoffs for the first-contrast eigenvalue depend strongly on sample size, test length, and item-parameter spread; simulation-based cutoffs tailored to the data are more defensible (Chou & Wang, 2010).

The generating model uses CML item parameters (psychotools) and WLE person locations. Per iteration: theta values are sampled with replacement from the WLE estimates; response data are simulated under the model (psychotools::rrm for dichotomous, partial-credit simulator for polytomous); CML/WLE standardized residuals are computed (the same engine as the observed analysis); prcomp() is run; the first-contrast eigenvalue is recorded.

Iterations that fail (e.g., due to a degenerate simulated dataset where some category isn't represented) are silently dropped. The iarm package is **not** required.

Value

A list with components:

results data.frame: iteration, eigenvalue.

p95, p99, p995, p999 Empirical percentiles of eigenvalue.

max The largest simulated eigenvalue.

suggested_cutoff The 99th percentile (p99) — pass this list back into RMdimResidualPCA via cutoff = , or use suggested_cutoff directly.

suggested_cutoff_percentile The percentile used for suggested_cutoff, currently always 99.

actual_iterations Number of successful iterations.

sample_n Number of complete cases used.

item_names Character vector of item names.

References

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. doi:10.1177/0013164410379322

See Also

RMdimResidualPCA

Examples

```
set.seed(1)
dat <- as.data.frame(
  matrix(sample(0:1, 200 * 12, replace = TRUE), nrow = 200, ncol = 12)
)
colnames(dat) <- paste0("I", 1:12)

# Few iterations for a fast example; use 250+ in real analyses
bound <- RMdimResidualPCACutoff(dat, iterations = 50, parallel = FALSE, seed = 1)
bound$suggested_cutoff

RMdimResidualPCA(dat, cutoff = bound)
```

RMitemCatProb *Item Category Probability Curves*

Description

Plots model-implied response-category probability curves for each item as a function of the latent trait θ. Item parameters are estimated by conditional maximum likelihood via `psychotools::pcmodel()` (a dichotomous item is a 2-category PCM). Each item gets its own facet panel, with one curve per response category coloured from low to high using the viridis palette. Comparable in scope to `eRm::plotICC()` and `mirt`'s trace plots, with a `ggplot2` / viridis output and optional descriptive labels for items and categories.

Usage

```
RMitemCatProb(
  data,
  item_labels = NULL,
  category_labels = NULL,
  theta_range = NULL,
  n_points = 200L,
  viridis_option = "D",
  viridis_end = 0.95,
  facet_ncol = NULL,
  label_wrap = 25L,
  line_width = 0.9,
  font = "sans",
  output = c("ggplot", "dataframe"),
  label_curves = c("legend", "path"),
  item = NULL,
  text_size = 4
)
```

Arguments

data	A data.frame or matrix of item responses. Items must be scored starting at 0 (non-negative integers). Missing values (NA) are allowed — the model fit handles them.
item_labels	Optional character vector of descriptive item labels (facet strip titles). Must be the same length as ncol(data). If NULL (the default), column names are used.
category_labels	Optional character vector of labels for the response categories (legend). Must be the same length as the number of categories spanning from 0 to the maximum observed value. If NULL (the default), numeric category values are used.
theta_range	Numeric length 2. Range of the latent trait θ (logits) plotted along the x-axis. When NULL (default) the range is c(-4, 4) for label_curves = "legend" and a wider c(-5, 5) for label_curves = "path" — the wider range gives geomtextpath more horizontal room to place per-curve labels at their peaks. Pass an explicit c(lo, hi) to override.
n_points	Integer. Number of evenly-spaced θ values at which the curves are evaluated. Default 200L.
viridis_option	Character. Viridis palette identifier. One of "A" through "H". Default "D" (viridis green).
viridis_end	Numeric in (0, 1]. Upper end of the viridis palette range; lower values keep the palette inside its mid-tones (avoids the very bright yellow at 1.0). Default 0.95.
facet_ncol	Optional integer. Number of columns in the facet layout. Default NULL (ggplot2::facet_wrap() auto-layout).
label_wrap	Integer. Characters per line for facet-strip label wrapping. Default 25L.
line_width	Numeric. Line width for the probability curves. Default 0.9.
font	Character. Font family for all text. Default "sans".
output	Character. Either "ggplot" (default) for a ggplot object, or "dataframe" for the underlying long-format probability table.
label_curves	Character. How response categories are identified. "legend" (default) draws lines and a separate colour legend with one swatch per category — the standard multi-item presentation. "path" uses geomtextpath::geom_textpath() to write each category's label *along* its own curve (a la classic IRT trace plots) and suppresses the legend; valid for **a single item at a time** because narrow facets do not give path labels enough horizontal room to render legibly. The model is still fit on the full multi-item data (PCM thresholds require multi-item input for CML estimation); the item argument then selects which item's curves to plot. Requires the optional geomtextpath package.
item	Character or integer. Used only when label_curves = "path". Either an item name (matching a column in data) or a column index, identifying which item's category curves to plot. Required for path mode.
text_size	Numeric. Used only when label_curves = "path". Size of the path labels in mm. Default 4.

Details

For each polytomous item i with response categories $0, 1, \ldots, K_i$ and threshold parameters $\delta_{i,1}, \ldots, \delta_{i,K_i}$ (CML, `psychotools`), the PCM category probability is

$$P(X_i = k \mid \theta) = \frac{\exp(\sum_{j=1}^{k}(\theta - \delta_{i,j}))}{\sum_{k'=0}^{K_i}\exp(\sum_{j=1}^{k'}(\theta - \delta_{i,j}))},$$

with the empty sum (when $k = 0$) taken as zero. For dichotomous items the function fits a Rasch model and treats the item difficulty $\delta_i = -\beta_i$ as the single threshold, recovering the standard two-category logistic ICC.

The colour mapping uses `scale_color_viridis_c()` against the integer category value, so the natural ordering of response categories is preserved visually — low categories at one end of the palette, high categories at the other. When `category_labels` is provided, the legend uses those labels (e.g., "Never" / "Sometimes" / "Often") while the colour mapping stays on the integer category value.

Items with fewer response categories than the maximum (e.g., an otherwise four-category scale with one three-category item) contribute only the categories they actually have to their own facet — the y-axis still spans $[0, 1]$.

Value

- If `output = "ggplot"`: a ggplot2::ggplot object, one facet per item.
- If `output = "dataframe"`: a long-format data.frame with columns `Item` (factor in column order), `Category` (integer), and `Theta`, `Probability` (numeric), one row per item × category × theta gridpoint.

See Also

`RMitemICCPlot()` for conditional ICCs binned by total score, `RMitemHierarchy()` for item threshold locations on the logit scale.

Examples

```
if (requireNamespace("eRm", quietly = TRUE) &&
    requireNamespace("ggplot2", quietly = TRUE)) {
  data(pcmdat2, package = "eRm")

  # Default plot
  RMitemCatProb(pcmdat2)

  # Custom item and category labels
  RMitemCatProb(
    pcmdat2,
    item_labels     = c("Mood", "Sleep", "Appetite", "Energy"),
    category_labels = c("Never", "Sometimes", "Often")
  )

  # Underlying probability data
```

```
  df <- RMitemCatProb(pcmdat2, output = "dataframe")
  head(df)

  # Single-item plot with labels written along each curve
  # (classic IRT trace-plot style). Model is still fit on all four
  # items; `item` picks which one to plot.
  if (requireNamespace("geomtextpath", quietly = TRUE)) {
    RMitemCatProb(
      pcmdat2,
      category_labels = c("Never", "Sometimes", "Often"),
      label_curves    = "path",
      item            = "I1"
    )
  }
}
```

`RMitemHierarchy` *Item-Threshold Hierarchy Plot*

Description

Visualises item and threshold locations on the logit scale for a Partial Credit Model. Items are sorted by their (mean-threshold) location; each item shows its location as a black diamond and its individual thresholds as coloured dots with confidence-interval error bars.

Usage

```
RMitemHierarchy(
  data,
  show_numbers = TRUE,
  sem_multiplier = 1.405,
  item_labels = NULL,
  output = c("ggplot", "dataframe")
)
```

Arguments

`data`	A data.frame or matrix of polytomous item responses (non-negative integers, 0-based, max value > 1). One column per item, one row per person.
`show_numbers`	Logical. When `TRUE` (default), prints the numerical item location and threshold values next to the points on the plot. When `FALSE`, only the threshold labels (`T1`, `T2`, ...) are shown.
`sem_multiplier`	Numeric multiplier for the threshold SE used to draw the error bars. Default `1.405` (84% CI). Common alternatives: `1.96` (95% CI), `2.576` (99% CI).

item_labels	Optional character vector of length ncol(data) providing descriptive labels for items. Default NULL uses the column names of data. Labels are appended to the column names on the y-axis (name - label) and wrapped at 36 characters via base-R strwrap().
output	One of "ggplot" (default) – a faceted hierarchy figure – or "dataframe" – the long-format underlying data with one row per (item × threshold).

Details

Threshold locations are centred at the grand mean of all thresholds across items, so the dashed reference line at 0 represents the mean threshold location across the scale.

Confidence intervals around thresholds are 84% by default (sem_multiplier = 1.405), following Payton, Greenstone, & Schenker (2003) – non-overlap of 84% intervals approximately corresponds to a two-sample significance test at $\alpha = 0.05$. Use sem_multiplier = 1.96 for 95% intervals.

Polytomous only. Dichotomous items have a single threshold that coincides with the item location; the hierarchy plot is visually degenerate in that case. For dichotomous data use RMtargeting() or RMscoreSE() instead.

Centring convention. The CML PCM thresholds (estimated via psychotools::pcmodel()) are shifted so that their grand mean is zero; each item's location is then the mean of its centred thresholds. The dashed horizontal reference line on the plot marks this zero – i.e., the average threshold across all items.

Value

Either a ggplot (default) or a data.frame with columns Item, ItemLabel, Threshold, ThresholdLocation, ThresholdSE, and ItemLocation (the per-item mean of the centred thresholds).

References

Payton, M. E., Greenstone, M. H., & Schenker, N. (2003). Overlapping confidence intervals or standard error intervals: What do they mean in terms of statistical significance? *Journal of Insect Science, 3*(34), 1-6. doi:10.1093/jis/3.1.34

See Also

RMtargeting(), RMscoreSE(), RMitemICCPlot()

Examples

```
if (requireNamespace("eRm", quietly = TRUE)) {
  data("pcmdat2", package = "eRm")
  RMitemHierarchy(pcmdat2)

  # 95% CI instead of 84%
  RMitemHierarchy(pcmdat2, sem_multiplier = 1.96)

  # Underlying data.frame
  RMitemHierarchy(pcmdat2, output = "dataframe")
}
```

RMitemICCPlot *Conditional Item Characteristic Curves*

Description

Faceted panel of Conditional Item Characteristic Curves (cICCs), one per item. Each panel shows the model-expected conditional item score $E[X_i \mid R = r]$ against the total score r, together with the average observed item score within class intervals of the total score and their confidence intervals. When a `dif_var` is supplied, observed averages are computed separately per group, turning each panel into a visual DIF check, and the partial-gamma DIF magnitude is reported per item.

Usage

```
RMitemICCPlot(
  data,
  dif_var = NULL,
  method = c("quantile", "width", "score", "manual", "cut"),
  class_intervals = 4,
  score_breaks = NULL,
  ci = TRUE,
  error_band = FALSE,
  conf_level = 0.95,
  min_n = 8L,
  items = NULL,
  output = c("patchwork", "list")
)
```

Arguments

`data`	A data.frame or matrix of item responses (non-negative integers, 0-based). One column per item, one row per person.
`dif_var`	Optional vector of length `nrow(data)` (factor or coercible to factor) defining a DIF grouping variable. When supplied, observed averages are shown separately per group and the partial-gamma DIF coefficient (`iarm::partgam_DIF()`, as in `RMdifGamma()`) is annotated per panel. Default `NULL` (no DIF).
`method`	How the observed means are grouped along the total score (see *Class intervals* in Details). One of `"quantile"` (default): `class_intervals` groups of approximately equal numbers of respondents; `"width"`: `class_intervals` equal-width intervals over the observed total-score range; `"score"`: every observed total score is its own group; or `"manual"`: groups defined by `score_breaks`. `"cut"` is accepted as a legacy alias for `"quantile"`.
`class_intervals`	Integer >= 2. Number of class intervals for `method = "quantile"` and `method = "width"`. Default `4`. Ignored (without an error) by `method = "score"` and `method = "manual"`.

score_breaks	Integer vector for method = "manual": the total scores at which a new group *starts*, in increasing order. For example, score_breaks = c(3, 6, 8) on a 0-9 scale defines the groups 0-2, 3-5, 6-7, and 8-9 (the endpoints are added automatically). Serves the same purpose as lower.groups in RASCHplot's CICCplot(), without the leading zero. Default NULL.
ci	Logical. Draw confidence intervals (error bars) on the observed class-interval means. Default TRUE.
error_band	Logical. Add a shaded band around the model-expected curve: the model's interval for the *observed mean* at each total score, $E \pm z\sqrt{\mathrm{Var}/n_r}$ (n_r = number of respondents at that total score), as in RASCHplot's CICCplot. Complementary to ci, not a replacement – see Details. Default FALSE.
conf_level	Numeric in (0, 1). Confidence level for the observed error bars and the model band. Default 0.95.
min_n	Integer. A group-by-interval cell needs at least this many respondents to contribute an observed point + CI; sparser cells are skipped. Default 8 (the package's per-cell stability floor).
items	Optional character or integer vector selecting which items to plot. The model is always fitted on all items; only the rendering is filtered. Default NULL (all items).
output	One of "patchwork" (default) – composite patchwork figure – or "list" – a named list of per-item ggplot objects.

Details

The model curve and its conditional variance come from the shared CML engine (CML item thresholds via psychotools and the exact conditional distribution of each item score given the total score, via elementary symmetric functions). The conditional-ICC approach follows Buchardt, Christensen & Jensen (2023) and their RASCHplot package; the implementation here is native to easyRasch2's CML/WLE engine.

Conditioning. The expected curve is the exact conditional expectation of the item score given the total score (it accounts for the item being part of the total), not a marginal ICC.

Class intervals. Following Buchardt et al. (2023), the empirical item means can be shown for each value of the total score or for each value of the *grouped* total score; all grouping happens on the total-score scale (never on an estimated latent score). Four grouping rules are available. method = "quantile" (the default) splits the observed total scores into class_intervals bins of approximately equal numbers of respondents, using common boundaries so DIF groups share the x-axis – the same approach as iarm::ICCplot()'s class intervals (Hmisc::cut2(g = ...)). method = "width" instead splits the observed score range into class_intervals intervals of equal width, which reads naturally on the score axis but can leave sparse intervals (pruned by min_n). method = "score" uses every observed total score as its own group – the maximal-resolution display, sensible for short scales, with min_n pruning thinly-populated scores. method = "manual" places the group boundaries exactly where you say via score_breaks (RASCHplot's lower.groups concept). Each bin contributes one observed point at its mean total score. If the score distribution is too sparse to form class_intervals distinct bins, the quantile and width methods fall back to score-level points.

The figure caption names the grouping that was actually used, which is not always the one requested. Quantile grouping forms fewer bins than asked for when total scores tie at the boundaries, equal-width and manual grouping can define intervals no respondent falls into, and both binned methods

fall back to score level on a degenerate score distribution. The caption reports the realised number of groups, how many of them contain respondents when that is fewer, and any fall back.

Confidence intervals. Observed error bars use the normal approximation $\bar{x}_l \pm z\sqrt{\mathrm{var}(x_l)/n_l}$ within each (group, interval) cell, clamped to the item's score range; cells with fewer than `min_n` respondents are dropped. In DIF mode this makes sparse group differences visibly uncertain rather than over-interpreted.

The model band (`error_band`). The shaded band is drawn around the model-expected curve at $E_{ri} \pm z\sqrt{V_{ri}/n_r}$, where E_{ri} and V_{ri} are the exact conditional mean and variance of the item score given total score r (model quantities), and n_r is the *observed* number of respondents at that total score. If the Rasch model is true, this is where the empirical mean item score at total score r should fall, given how many people actually sit at that score; an observed diamond outside the band is localized graphical misfit. The band is deliberately the standard error of the *mean* ($\sqrt{V/n_r}$), not the far wider individual-response SD band, which would flag nothing. Its width therefore tracks the data: narrow where many respondents sit (a strict test), wide in sparse score regions (a lenient one), with gaps where $n_r = 0$ and collapsing to the curve at the deterministic extremes (total score 0 and the maximum). It is complementary to, not a replacement for, the `ci` error bars: the band is the model-implied uncertainty of the observed mean per raw score; the error bars are the sample-based uncertainty of the observed group means.

DIF magnitude. The annotated partial gamma (Bjorner et al., 1998) is the association between the item score and the group conditional on the rest score – a non-parametric effect size, complementary to the total-score-conditional visual. It is the same statistic as `RMdifGamma()`.

Value

Either a `patchwork`/`ggplot` composite (default) or a named list of per-item `ggplot` objects (`output = "list"`). In DIF mode the per-item partial-gamma table is attached as `attr(., "dif_gamma")`.

References

Buchardt, A.-S., Christensen, K. B., & Jensen, S. N. (2023). Visualizing Rasch item fit using conditional item characteristic curves in R. *Psychological Test and Assessment Modeling, 65*(2), 206-219.

Andersen, E. B. (1995). Polytomous Rasch models and their estimation. In G. H. Fischer & I. W. Molenaar (Eds.), *Rasch Models: Foundations, Recent Developments, and Applications* (pp. 271-291). Springer. (Conditional expectation of item scores given the total score; formula 15.22.)

Bjorner, J. B., Kreiner, S., Ware, J. E., Damsgaard, M. T., & Bech, P. (1998). Differential item functioning in the Danish translation of the SF-36. *Journal of Clinical Epidemiology, 51*(11), 1189-1202.

See Also

`RMdifGamma`, `RMitemRestscore`

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE) &&
    requireNamespace("patchwork", quietly = TRUE) &&
    requireNamespace("iarm", quietly = TRUE)) {
```

```
  set.seed(42)
  sim_data <- as.data.frame(
    matrix(sample(0:2, 200 * 4, replace = TRUE), nrow = 200, ncol = 4)
  )
  colnames(sim_data) <- paste0("Item", 1:4)

  # Default: quantile groups (~equal numbers of respondents per group)
  RMitemICCPlot(sim_data)

  # Equal-width intervals on the total-score scale, with the model band
  RMitemICCPlot(sim_data, method = "width", error_band = TRUE)

  # Every total score as its own group (short scales)
  RMitemICCPlot(sim_data, method = "score")

  # Manual grouping: new groups start at total scores 3, 5, and 7
  RMitemICCPlot(sim_data, method = "manual", score_breaks = c(3, 5, 7))
}
```

`RMitemInfit` *Conditional Item Infit MSQ*

Description

Computes conditional infit mean-square (MSQ) statistics for each item using `iarm::out_infit()`, enriched with item locations relative to the sample mean person location.

Usage

```
RMitemInfit(
  data,
  cutoff = NULL,
  p_value = NULL,
  correction = c("fwer", "fdr_bh", "fdr_by", "none"),
  alpha = 0.05,
  output = "kable",
  sort,
  statistic = "infit"
)
```

Arguments

`data`	A data.frame or matrix of item responses. Items must be scored starting at 0 (non-negative integers). Missing values (`NA`) are allowed, but at least one complete case (row with no `NA`) must be present.
`cutoff`	Optional. Default `NULL` (no cutoff applied, behaviour is identical to the current version). Can be:

- The return value of RMitemInfitCutoff (a list with $item_cutoffs): the data.frame is extracted automatically and the number of simulation iterations, cutoff_method, and hdci_width are included in the kable caption.
- The $item_cutoffs data.frame from RMitemInfitCutoff directly: must have columns Item, infit_low, and infit_high (or outfit_low and outfit_high when statistic = "outfit"). When provided, adds columns Infit_low, Infit_high, and Flagged to the result. Flagged labels the misfit direction: "overfit" (infit below the range), "underfit" (above the range), or "" (within range, no misfit).

p_value — Logical or NULL. Whether to compute bootstrap p-values from the simulated null distribution and flag on them.

NULL (the default) means **use them when they are available**: TRUE when cutoff is the full RMitemInfitCutoff object, which carries the per-item simulated values in its $results element, and FALSE when cutoff is NULL or the summarised $item_cutoffs data.frame, neither of which can support a p-value. Explicit TRUE with an insufficient cutoff is an error rather than a silent downgrade.

With FALSE, items are flagged against the interval instead, and a one-time message reports the family-wise error rate that implies. The interval describes where a fitting item's statistic is expected to fall. Using it as a decision rule tests all k items at once, so its width sets a family-wise error rate of 1 - width^k, which is 37% for the default 95% interval over nine items. The corrected p-value targets alpha directly and needs far fewer iterations to do it (Johansson, 2026).

Up to and including version 1.1.1 the default was FALSE, so Flagged came from the interval. Existing scripts that pass the full cutoff object will now flag on the corrected p-value instead and may report different items.

correction — Character. Multiple-comparison correction applied across items when p_value = TRUE: "fwer" (default) for the Westfall-Young studentised-max step-down (family-wise error rate), "fdr_bh" / "fdr_by" for Benjamini-Hochberg / Benjamini-Yekutieli false discovery rate control, or "none" for uncorrected per-item p-values.

alpha — Numeric in (0, 1). Significance level for the Flagged column when p_value = TRUE (an item is flagged when its corrected p-value is below alpha). Default 0.05.

output — Character string controlling the return value. Either "kable" (default) for a formatted knitr::kable() table, or "dataframe" for the underlying data.frame.

sort — Optional character string. When sort matches statistic ("infit" by default), rows are sorted by the reported MSQ column in descending order before output.

statistic — Character string. Which conditional fit statistic the table reports: "infit" (default) or "outfit". Both come from the same iarm::out_infit() call and the same parametric bootstrap, so the choice only selects which one is displayed and tested. Column names, cutoff bounds, p-values and Flagged all follow it, and when p_value = TRUE the multiplicity correction is applied across items *within* the selected statistic (the family is the k items, not $2k$ item-by-statistic combinations). Testing both statistics on the same items and reporting whichever flags is a larger family than either call corrects for.

Details

Infit MSQ is a weighted fit statistic that emphasises deviations near the item location. Values close to 1.0 indicate good fit. Values substantially above 1.0 suggest underfit (unexpected responses), while values substantially below 1.0 suggest overfit (overly predictable responses). The definition of "substantially" depends on several factors such as sample size, and needs to be determined by simulation using `RMitemInfitCutoff`. There is no general rule-of-thumb value that is correct.

Conditional infit MSQ statistics are computed via `iarm::out_infit()`, which uses the conditional distribution of the sufficient statistics (Müller, 2020). Only complete cases (rows without any `NA`) are used in the conditional fit calculation.

Item parameters are estimated by conditional maximum likelihood via `psychotools::pcmodel()` (a dichotomous item is a 2-category PCM); the conditional infit/outfit MSQ comes from `iarm::out_infit()` and is invariant to the estimation engine. Per-item average locations are the means of the CML thresholds, and the person-location reference is the mean of the Warm WLE estimates.

Relative item location is defined as the item's average location minus the sample mean person location, providing a measure of item targeting.

The `iarm` package must be installed (it is in Suggests, not Imports).

Bootstrap p-values. When `p_value = TRUE`, each item's observed infit (or outfit, per `statistic`) is compared against its simulated null distribution (from `cutoff$results`). The per-item statistic is the residual studentised by the bootstrap mean and SD – deliberately the empirical SD rather than the Wilson-Hilferty / ZSTD transform, which is uninformative for conditional MSQ (Müller, 2020). The marginal p-value is the two-sided Monte-Carlo p-value `(1 + #{|t*| >= |t|}) / (B + 1)`. For `correction = "fwer"` the family-wise adjustment uses the Westfall-Young studentised-max step-down, which exploits the bootstrap dependence among items and is more powerful than Bonferroni/Holm (Ferreira, 2024); its validity rests on subset pivotality. `"fdr_bh"`/`"fdr_by"` apply Benjamini-Hochberg / Benjamini- Yekutieli instead. These are model-conditional, sample-size-sensitive p-values and are reported alongside the simulated effect-size band, not in place of it. p-values can be no smaller than `1 / (B + 1)`. Below 400 `iterations` in `RMitemInfitCutoff()` the studentised-max (FWER) correction is mildly liberal, because the bootstrap mean and SD used for studentisation are too noisy, and a message says so. From 400 iterations the family-wise error rate is at the nominal level, and more iterations improve reproducibility instead: two analysts using different seeds disagree about at least one item roughly 10 percent of the time at 400 iterations against 4 percent at 2000 (Johansson, 2026). The table caption notes this below 1000 iterations, so use 1000 to 2000 for a final analysis. The marginal p-values are well calibrated even at a few hundred iterations.

Value

- If `output = "kable"`: a `knitr_kable` object (plain text table via `format = "pipe"`) with columns "Item", "Infit MSQ", and "Relative location", and a caption showing the number of complete cases. When `cutoff` is provided, columns "Infit low", "Infit high", and "Flagged" are also included, and the caption notes the simulation-based cutoffs.
- If `output = "dataframe"`: a data.frame with columns `Item`, `Infit_MSQ`, and `Relative_location`. When `cutoff` is provided, columns `Infit_low`, `Infit_high`, and `Flagged` are also included (inserted after `Infit_MSQ`, before `Relative_location`). `Flagged` is a character column with values `"overfit"`, `"underfit"`, or `""` (not the previous logical). When `p_value = TRUE`, columns `p_infit` (marginal two-sided p-value) and `padj_infit` (corrected p-value) are added

and `Flagged` reflects items with `padj_infit < alpha` (direction from `Infit_MSQ` relative to 1).

With `statistic = "outfit"` the layout is identical and the column names follow the statistic: `Outfit_MSQ`, `Outfit_low`, `Outfit_high`, `p_outfit`, `padj_outfit`.

Multiple comparisons

The marginal p-value controls the error rate of a *single* comparison: for one item (or item pair) decided on in advance it is the relevant value. But scanning all k comparisons and flagging whichever fall below `alpha` tests k hypotheses at once, so the chance of at least one false flag inflates to roughly $1-(1-\alpha)^k$ (e.g. about 34% for $k = 8$ at `alpha = 0.05`) – even when every marginal p-value is correctly calibrated. The corrected (adjusted) p-value controls this: `correction = "fwer"` bounds the probability of *any* false flag (strict, lower power), while `"fdr_bh"` / `"fdr_by"` bound the expected *proportion* of false flags among those raised (a more lenient middle ground). Rule of thumb: use the marginal p-value for a single pre-specified comparison, and a corrected p-value when screening the whole table – the usual workflow.

Flags depend on the other items

Item infit and item-restscore both judge each item against expectations from a Rasch model fitted to all items, so misfitting items change what is expected of the others. A flag in one direction can therefore produce flags in the other. Overfitting items can push fitting items into the underfit range, and underfitting items, for example from a second dimension, can make fitting items look overfit. How large the effect is depends on how strong the misfit is and how many items share it. In simulation with seven polytomous items, two overfitting items led to about 6 percent of the fitting items being flagged as underfit, and one or two mildly multidimensional items to 1 to 2 percent being flagged as overfit. When items are flagged in both directions, consider removing the clearest misfit and testing again before interpreting the rest.

References

Müller, M. (2020). Item fit statistics for Rasch analysis: Can we trust them? *Journal of Statistical Distributions and Applications*, 7(5). doi:10.1186/s40488020001087

Ferreira, J. A. (2024). Methods of testing a 'small' or 'moderate' number of hypotheses simultaneously. *Journal of Statistical Theory and Practice, 19*(6). doi:10.1007/s42519024004124

Westfall, P. H., & Young, S. S. (1993). *Resampling-Based Multiple Testing*. Wiley.

Johansson, M. (2026). Simulation-based cutoffs for conditional item fit in Rasch models: Iterations, multiplicity correction, and decision stability. *PsyArXiv*. doi:10.31234/osf.io/7pqz4_v2

See Also

`RMitemInfitCutoff`

Examples

```
if (requireNamespace("iarm", quietly = TRUE)) {
  # Simulate binary item response data (5 items, 40 persons)
  set.seed(42)
  sim_data <- as.data.frame(
```

```
  matrix(sample(0:1, 40 * 5, replace = TRUE), nrow = 40, ncol = 5)
)
colnames(sim_data) <- paste0("Item", 1:5)

# Default kable output
RMitemInfit(sim_data)

# Sorted by infit MSQ descending
RMitemInfit(sim_data, sort = "infit")

# Return as data.frame for further processing
df <- RMitemInfit(sim_data, output = "dataframe")

# Simulation-based cutoffs (100 Monte-Carlo iterations)
if (requireNamespace("ggdist", quietly = TRUE)) {
  cutoff_res <- RMitemInfitCutoff(sim_data, iterations = 100,
                                  parallel = FALSE, seed = 42)
  RMitemInfit(sim_data, cutoff = cutoff_res)
  RMitemInfit(sim_data, cutoff = cutoff_res, output = "dataframe")

  # Bootstrap p-values with family-wise (Westfall-Young) correction
  # (use iterations >= 1000 in real analyses for stable p-values)
  RMitemInfit(sim_data, cutoff = cutoff_res, p_value = TRUE,
              output = "dataframe")

  # The same for conditional outfit MSQ
  RMitemInfit(sim_data, cutoff = cutoff_res, p_value = TRUE,
              statistic = "outfit", output = "dataframe")
}
}
```

RMitemInfitCutoff *Simulation-Based Infit MSQ Cutoff Determination*

Description

Uses parametric bootstrap simulation to determine appropriate cutoff values for RMitemInfit. This function simulates data from a correctly fitting Rasch model that mimics your data and returns per-item empirical cutoffs.

Usage

```
RMitemInfitCutoff(
  data,
  iterations = 400,
  parallel = TRUE,
  n_cores = NULL,
  verbose = FALSE,
```

```
  seed = NULL,
  cutoff_method = "hdci",
  hdci_width = 0.95,
  dgp = c("resample", "conditional")
)
```

Arguments

data	A data.frame or matrix of item responses. Items must be scored starting at 0 (non-negative integers). Only complete cases (rows without any NA) are used.
iterations	Integer. Number of simulation iterations (default 400).
parallel	Logical. Use parallel processing via mirai if available (default TRUE).
n_cores	Integer or NULL. Number of parallel workers. When NULL, getOption("mc.cores") is checked first. If neither is set and parallel = TRUE, a warning is issued and execution falls back to sequential (single core) processing.
verbose	Logical. Show a progress bar (default FALSE).
seed	Integer or NULL. Random seed for reproducibility. See easyRasch2-reproducibility for what this guarantees and how it interacts with parallel.
cutoff_method	Character string specifying how cutoff intervals are computed. Either "hdci" (default) for the Highest Density Interval via ggdist::hdci(), or "quantile" for the 2.5th/97.5th percentiles via stats::quantile().
hdci_width	Numeric. Width of the HDCI when cutoff_method = "hdci". Default is 0.95 (95% HDCI). Ignored when cutoff_method = "quantile". The interval is a **description** of where a fitting item's statistic is expected to fall, not a decision rule. Flagging every item outside a width-w interval tests all k items at once, so the family-wise error rate is 1 - w^k, which is 37% for a 95% interval over nine items. Decisions should come from the corrected p-value instead (RMitemInfit with p_value = NULL and the full object returned here). The default was 0.999 up to and including version 1.1.1, which needs roughly 5000 iterations before the interval reaches its stated width; 0.95 reaches it by about 1000, so the band shown to readers means close to what it says (Johansson, 2026).
dgp	Character. Data-generating process for the parametric bootstrap. "resample" (default) resamples WLE person locations with replacement and simulates responses under the model (a *marginal* null). "conditional" simulates each respondent's pattern from the exact Rasch conditional distribution given their observed total score, item parameters fixed (a *conditional* null). Because the conditional infit/outfit statistic is itself conditional on the total score, "conditional" is its naturally matched null. **Experimental.**

Details

The generating model is CML item parameters (via psychotools) with WLE person locations. For each iteration a dataset is simulated under the chosen dgp, the model is refitted by CML (psychotools::pcmodel(), which handles dichotomous and polytomous data and is accepted by

iarm), and conditional infit and outfit MSQ are computed via iarm::out_infit(). The distribution of these statistics across iterations provides empirical critical values per item. Failed iterations (e.g., degenerate simulated data) are silently discarded.

Parallel processing is provided by the mirai package (optional). Install it with install.packages("mirai") to enable parallelisation.

The iarm package must be installed (it is in Suggests, not Imports).

Value

A list with components:

results data.frame with columns iteration, Item, InfitMSQ, OutfitMSQ (one row per item per successful iteration).

item_cutoffs data.frame with per-item cutoff summaries: Item, infit_low, infit_high, outfit_low, outfit_high. Bounds are computed using the method specified by cutoff_method.

actual_iterations Number of successful iterations. Everything downstream rests on this rather than on iterations, so it is the number to report.

requested_iterations The iterations argument, kept so callers can tell how many simulated datasets were discarded.

sample_n Number of complete cases used.

sample_n_total Number of respondents in the raw input data, before the complete-case filter.

sample_has_na Logical. Whether the raw input data contained any missing values.

sample_summary Summary statistics of estimated person parameters.

item_names Character vector of item names from data.

cutoff_method The method used to compute cutoffs ("hdci" or "quantile").

hdci_width The HDCI width used (only meaningful when cutoff_method = "hdci").

dgp The data-generating process used ("resample" or "conditional").

References

Johansson, M. (2025). Detecting item misfit in Rasch models. *Educational Methods & Psychometrics, 3*(18). doi:10.61186/emp.2025.5

Johansson, M. (2026). Simulation-based cutoffs for conditional item fit in Rasch models: Iterations, multiplicity correction, and decision stability. *PsyArXiv*. doi:10.31234/osf.io/7pqz4_v2

See Also

RMitemInfit

Examples

```
if (requireNamespace("iarm", quietly = TRUE) &&
    requireNamespace("ggdist", quietly = TRUE)) {
  set.seed(42)
  sim_data <- as.data.frame(
    matrix(sample(0:1, 200 * 10, replace = TRUE), nrow = 200, ncol = 10)
```

```
  )
  colnames(sim_data) <- paste0("Item", 1:10)

  # Run 100 iterations sequentially for a quick demo
  cutoff_res <- RMitemInfitCutoff(sim_data, iterations = 100,
                                  parallel = FALSE, seed = 42)
  cutoff_res$item_cutoffs

  # Use the cutoffs in RMitemInfit()
  RMitemInfit(sim_data)
}
```

`RMitemInfitCutoffMI` *Simulation-Based Infit MSQ Cutoff Determination for Multiply Imputed Data*

Description

Extends `RMitemInfitCutoff` to work with multiply imputed datasets produced by the `mice` package. Runs the parametric bootstrap simulation on each imputed dataset and stacks the resulting distributions, so that the final cutoff intervals reflect both sampling variability and imputation uncertainty.

Usage

```
RMitemInfitCutoffMI(
  mids_object,
  iterations = 500,
  parallel = TRUE,
  n_cores = NULL,
  verbose = FALSE,
  seed = NULL,
  cutoff_method = "hdci",
  hdci_width = 0.999
)
```

Arguments

`mids_object`	A `mids` object (multiply imputed dataset) as returned by `mice::mice()`. Each completed dataset must contain only the item response columns to be analysed (i.e., no ID or grouping variables). Items must be scored starting at 0 (non-negative integers).
`iterations`	Integer. Total number of simulation iterations to run across all imputations. These are distributed approximately evenly across the `m` imputed datasets (default 500).
`parallel`	Logical. Use parallel processing via `mirai` within each imputed dataset (default `TRUE`). Passed to `RMitemInfitCutoff`.

n_cores	Integer or NULL. Number of parallel workers. Passed to RMitemInfitCutoff.
verbose	Logical. Show progress messages (default FALSE).
seed	Integer or NULL. Master random seed for reproducibility. A unique per-imputation seed is derived from this value. See easyRasch2-reproducibility for what this guarantees and how it interacts with parallel.
cutoff_method	Character string specifying how cutoff intervals are computed from the stacked distribution. Either "hdci" (default) for the Highest Density Interval via ggdist::hdci(), or "quantile" for the 2.5th/97.5th percentiles via stats::quantile().
hdci_width	Numeric. Width of the HDCI when cutoff_method = "hdci". Default is 0.999 (99.9% HDCI). Ignored when cutoff_method = "quantile".

Details

The function completes each of the m imputed datasets via mice::complete(), then calls RMitemInfitCutoff on each one. The total number of iterations is split approximately evenly across imputations (i.e., each imputed dataset receives ceiling(iterations / m) or floor(iterations / m) iterations). The per-imputation simulation results are stacked into a single distribution from which cutoff intervals are computed, naturally incorporating imputation uncertainty.

Imputed datasets that cause model convergence failures are dropped with a warning. If all imputations fail, the function stops with an error.

The mice package must be installed (it is in Suggests, not Imports).

Value

A list with the same structure as RMitemInfitCutoff, so that the result can be passed directly to RMitemInfit, RMitemInfitMI, and RMitemInfitPlot:

results data.frame with columns iteration, imputation, Item, InfitMSQ, OutfitMSQ — the stacked simulation results from all imputed datasets.

item_cutoffs data.frame with per-item cutoff summaries: Item, infit_low, infit_high, outfit_low, outfit_high. Computed from the stacked distribution.

actual_iterations Total number of successful iterations across all imputations.

sample_n Number of rows (respondents) per imputed dataset.

sample_summary Summary statistics of estimated person parameters from the first imputed dataset.

item_names Character vector of item names.

cutoff_method The method used to compute cutoffs.

hdci_width The HDCI width used.

n_imputations Number of imputed datasets used.

iterations_per_imputation Integer vector of requested iterations per imputed dataset.

actual_iterations_per_imputation Integer vector of successful iterations per imputed dataset.

Interval width under multiple imputation

hdci_width defaults to 0.999 here, not to the 0.95 used by RMitemInfitCutoff. RMitemInfitMI has no corrected-p-value path yet, so under imputation the interval is still the decision rule and keeps a width chosen for that job. Combining bootstrap p-values across imputations is planned but needs its own calibration, since Johansson (2026) covers complete data only.

See Also

RMitemInfitCutoff, RMitemInfitMI, RMitemInfitPlot

Examples

```
if (requireNamespace("mice", quietly = TRUE) &&
    requireNamespace("iarm", quietly = TRUE) &&
    requireNamespace("ggdist", quietly = TRUE)) {
  # Create example data with ~10% MCAR missingness
  set.seed(42)
  mat <- matrix(sample(0:1, 200 * 8, replace = TRUE), nrow = 200, ncol = 8)
  mat[sample(length(mat), round(0.10 * length(mat)))] <- NA
  sim_data <- as.data.frame(mat)
  colnames(sim_data) <- paste0("Item", 1:8)

  # mice's ordinal method (`polr`) requires the items to be ordered
  # factors, so code them as such before imputing. RMitemInfitCutoffMI()
  # converts the completed factors back to numeric internally.
  sim_data[] <- lapply(sim_data, function(x) factor(x, ordered = TRUE))

  # Impute (use more imputations, e.g. m = 5+, in real analyses)
  imp <- mice::mice(sim_data, m = 2, method = "polr", seed = 123,
                    printFlag = FALSE)

  # Compute simulation-based cutoffs across imputations
  # (use more iterations, e.g. 250+, in real analyses)
  cutoff_mi <- RMitemInfitCutoffMI(imp, iterations = 50, parallel = FALSE,
                                   seed = 42)
  cutoff_mi$item_cutoffs

  # Use with RMitemInfitMI()
  RMitemInfitMI(imp, cutoff = cutoff_mi)
}
```

RMitemInfitMI *Conditional Item Infit MSQ for Multiply Imputed Data*

Description

Extends RMitemInfit to work with multiply imputed datasets produced by the mice package. Computes conditional infit MSQ on each imputed dataset and pools the results using Rubin's rules.

Usage

```
RMitemInfitMI(mids_object, cutoff = NULL, output = "kable", sort)
```

Arguments

mids_object	A mids object (multiply imputed dataset) as returned by mice::mice(). Each completed dataset must contain only the item response columns to be analysed (i.e., no ID or grouping variables). Items must be scored starting at 0 (non-negative integers).
cutoff	Optional. Default NULL (no cutoff applied). Can be: • The return value of RMitemInfitCutoff or RMitemInfitCutoffMI (a list with $item_cutoffs): the data.frame is extracted automatically and meta-data is included in the kable caption. • The $item_cutoffs data.frame directly: must have columns Item, infit_low, and infit_high. When provided, adds columns Infit_low, Infit_high, and Flagged to the result. Flagged is a character column labelling the misfit direction: "overfit" (pooled infit below the range), "underfit" (above), or "" (within range).
output	Character string controlling the return value. Either "kable" (default) for a formatted knitr::kable() table, or "dataframe" for the underlying data.frame.
sort	Optional character string. When sort = "infit", rows are sorted by Infit_MSQ in descending order before output.

Details

For each of the m imputed datasets, the function:

1. Fits a Rasch model by CML via psychotools::pcmodel() (a dichotomous item is a 2-category partial credit model), consistent with RMitemInfit and the rest of the package.
2. Computes conditional infit MSQ and its standard error via iarm::out_infit().
3. Computes item locations (mean of the grand-mean-centred CML Andrich thresholds) and the mean WLE person location.

The per-imputation estimates are then pooled using Rubin's rules:

Pooled MSQ The mean of the m infit MSQ point estimates.

Within-imputation variance The mean of the m squared standard errors.

Between-imputation variance The sample variance of the m point estimates.

Total variance Within + (1 + 1/m) * Between.

Pooled SE The square root of the total variance.

Relative item location is the mean of per-imputation relative locations (item location minus sample mean person location).

Caveat on the pooled SE. The within-imputation variance is the squared conditional infit SE from iarm::out_infit(). Müller (2020) showed that this asymptotic SE is an unreliable measure of uncertainty for the conditional infit statistic; Rubin's pooled SE inherits that limitation, so the Infit_SE/Infit SE column should be read as an approximate indication of imputation-related variability rather than a trustworthy inferential standard error. For item misfit decisions, prefer the simulation-based cutoffs from RMitemInfitCutoffMI.

Imputed datasets that cause model convergence failures are dropped with a warning. If all imputations fail, the function stops with an error. At least two successful imputations are required to estimate between-imputation variance.

The `mice` and `iarm` packages must be installed (they are in Suggests, not Imports).

Value

- If `output = "kable"`: a `knitr_kable` object (plain text table via `format = "pipe"`) with columns "Item", "Infit MSQ", "Infit SE", "Relative location", and a caption noting the number of imputations and complete cases. When `cutoff` is provided, columns "Infit low", "Infit high", and "Flagged" are also included.
- If `output = "dataframe"`: a data.frame with columns `Item`, `Infit_MSQ`, `Infit_SE`, and `Relative_location`. When `cutoff` is provided, columns `Infit_low`, `Infit_high`, and `Flagged` are also included (inserted after `Infit_SE`, before `Relative_location`). `Flagged` is a character column (`"overfit"` / `"underfit"` / `""`), not the previous logical.

Flagging differs from the complete-data function

`RMitemInfit` flags on the Westfall-Young corrected p-value by default. This function has no p-value path and flags against the interval, so the two are not directly comparable. Combining bootstrap p-values across imputations is planned but needs calibration of its own, since Johansson (2026) studied complete data. Until then `RMitemInfitCutoffMI` keeps `hdci_width = 0.999`, a width suited to a decision rule, and the family-wise error rate of that rule is `1 - 0.999^k` over `k` items.

References

Müller, M. (2020). Item fit statistics for Rasch analysis: Can we trust them? *Journal of Statistical Distributions and Applications*, 7(5). doi:10.1186/s40488020001087

See Also

`RMitemInfit`, `RMitemInfitCutoffMI`

Examples

```
if (requireNamespace("mice", quietly = TRUE) &&
    requireNamespace("iarm", quietly = TRUE) &&
    requireNamespace("ggdist", quietly = TRUE)) {
  # Create example data with ~10% MCAR missingness
  set.seed(42)
  mat <- matrix(sample(0:1, 200 * 8, replace = TRUE), nrow = 200, ncol = 8)
  mat[sample(length(mat), round(0.10 * length(mat)))] <- NA
  sim_data <- as.data.frame(mat)
  colnames(sim_data) <- paste0("Item", 1:8)

  # mice's ordinal method (`polr`) requires the items to be ordered
  # factors, so code them as such before imputing. RMitemInfitMI()
  # converts the completed factors back to numeric internally.
  sim_data[] <- lapply(sim_data, function(x) factor(x, ordered = TRUE))
```

```
  # Impute (use more imputations, e.g. m = 5+, in real analyses)
  imp <- mice::mice(sim_data, m = 2, method = "polr", seed = 123,
                    printFlag = FALSE)

  # Pooled infit table (no cutoffs)
  RMitemInfitMI(imp)

  # With simulation-based cutoffs
  # (use more iterations, e.g. 250+, in real analyses)
  cutoff_mi <- RMitemInfitCutoffMI(imp, iterations = 50, parallel = FALSE,
                                   seed = 42)
  RMitemInfitMI(imp, cutoff = cutoff_mi)

  # As data.frame
  df <- RMitemInfitMI(imp, cutoff = cutoff_mi, output = "dataframe")
}
```

`RMitemInfitPlot` *Plot Distribution of Simulated Infit and Outfit MSQ Values*

Description

Visualises the distribution of simulation-based conditional item fit values from `RMitemInfitCutoff`, optionally overlaying observed item fit from the original data.

Usage

```
RMitemInfitPlot(simfit, data, statistic = "infit")

RMitemInfitCutoffPlot(...)
```

Arguments

`simfit`	The return value of `RMitemInfitCutoff` (a list with components `results`, `item_cutoffs`, `actual_iterations`, `sample_n`, and `item_names`).
`data`	Optional. A data.frame or matrix of item responses for computing and overlaying observed conditional item fit values. Items must be scored starting at 0 (non-negative integers). When provided, the plot includes orange diamond markers for the observed infit/outfit MSQ alongside the simulated distribution, plus segment summaries from the cutoff intervals.
`statistic`	Character string. Either `"infit"` (default) to show only the infit panel, `"outfit"` to show only the outfit panel, or `"both"` to show infit and outfit side by side (requires the `patchwork` package when `data` is supplied).
`...`	Arguments passed on to `RMitemInfitPlot()`.

Details

Uses `ggdist::stat_dotsinterval()` (when `data` is not supplied) or `ggdist::stat_dots()` (when `data` is supplied) with `point_interval = "median_hdci"`. The outer `.width` follows `simfit$hdci_width`, so the shaded interval matches the one `RMitemInfit()` tabulates.

When `data` is **not** supplied, the function plots the simulated MSQ distributions as dot-interval plots using `ggdist::stat_dotsinterval()` with median and Highest Density Continuous Interval (HDCI) summaries, faceted by statistic (InfitMSQ / OutfitMSQ).

When `data` **is** supplied, the function:

1. Fits a Rasch / Partial Credit model by CML via `psychotools::pcmodel()` (a dichotomous item is a 2-category PCM) and computes observed conditional infit and outfit MSQ via `iarm::out_infit()`.
2. Overlays observed fit values as orange diamond markers on the simulated distributions.
3. Shows per-item cutoff intervals (from `simfit$item_cutoffs`) as black line segments, with thicker segments for the 66% HDCI range and black dots for the median.

The `ggplot2`, `ggdist`, and optionally `patchwork` packages must be installed (they are in Suggests, not Imports).

`RMitemInfitCutoffPlot()` is a deprecated alias for `RMitemInfitPlot()`, retained for backward compatibility with code written against easyRasch2 0.8.0. It warns and forwards to `RMitemInfitPlot()`.

Value

A `ggplot` object (or a `patchwork` object when `statistic = "both"` and `data` is supplied).

See Also

`RMitemInfitCutoff`, `RMitemInfit`

Examples

```
if (requireNamespace("iarm", quietly = TRUE) &&
    requireNamespace("ggdist", quietly = TRUE) &&
    requireNamespace("ggplot2", quietly = TRUE)) {
  set.seed(42)
  sim_data <- as.data.frame(
    matrix(sample(0:1, 200 * 10, replace = TRUE), nrow = 200, ncol = 10)
  )
  colnames(sim_data) <- paste0("Item", 1:10)

  # Run simulation
  cutoff_res <- RMitemInfitCutoff(sim_data, iterations = 100,
                                  parallel = FALSE, seed = 42)

  # Simulated distribution only (infit + outfit faceted)
  RMitemInfitPlot(cutoff_res)

  # With observed fit overlaid (infit only, the default)
  RMitemInfitPlot(cutoff_res, data = sim_data)
```

```
  # Both infit and outfit panels side by side
  if (requireNamespace("patchwork", quietly = TRUE)) {
    RMitemInfitPlot(cutoff_res, data = sim_data, statistic = "both")
  }
}
```

`RMitemParameters` *Item Parameters for a Rasch / Partial Credit Model*

Description

Estimates item difficulty (dichotomous) or item-category threshold (polytomous) parameters and returns them in long or wide format, with optional standard errors and Wald confidence intervals. Item parameters are estimated by conditional maximum likelihood (CML, via **psychotools**) by default, with marginal maximum likelihood (MML, via **mirt**) available for sparse data where CML can be unstable.

Usage

```
RMitemParameters(
  data,
  estimator = c("CML", "MML"),
  format = c("long", "wide"),
  se = TRUE,
  ci_level = 0.95,
  center = TRUE,
  output = c("kable", "dataframe", "file"),
  filename = NULL
)
```

Arguments

`data`	A data.frame or matrix of item responses. Items must be scored starting at 0 (non-negative integers). Missing values (`NA`) are allowed; both estimators handle them.
`estimator`	Character. `"CML"` (default) estimates item parameters with `psychotools::pcmodel()` (a dichotomous item is a 2-category PCM); `"MML"` uses `mirt::mirt()` with `itemtype = "Rasch"`. CML is preferred for Rasch measurement; MML can be more robust when data are sparse. When `estimator = "CML"` and sparse response categories are detected, a warning suggests switching to MML.
`format`	Character. `"long"` (default) returns one row per item (dichotomous) or per item-threshold (polytomous); `"wide"` returns one row per item with threshold columns `t1`, `t2`, ... plus a mean `location` column.
`se`	Logical. If `TRUE` (default), standard-error and confidence-interval columns are added.

ci_level	Numeric in (0, 1). Confidence level for the Wald interval (estimate +/- z * SE). Default 0.95.
center	Logical. If TRUE (default), all thresholds are shifted so their grand mean is zero, the usual Rasch identification. CML estimates are already centred; the shift mainly affects the MML path, keeping the two estimators on a common scale.
output	Character. "kable" (default) for a formatted knitr::kable() table, "dataframe" for the underlying data.frame, or "file" to write that data.frame to a CSV at filename (the data.frame is also returned invisibly).
filename	Character. Path to the CSV file to write when output = "file". Required in that case; ignored otherwise.

Details

Items are detected as dichotomous (maximum score 1) or polytomous (maximum score > 1), and the Rasch or Partial Credit model is chosen accordingly. Thresholds are reported as Andrich thresholds (the person locations at which adjacent response categories are equally probable) on the logit difficulty scale, matching RMtargeting().

Standard errors. For the CML path, threshold SEs are the square roots of the diagonal of the threshold-parameter covariance from psychotools::threshpar(vcov = TRUE). For the MML path, SEs come from the **mirt** parameter covariance, propagated by the delta method through the linear threshold map. Confidence intervals are Wald intervals and are symmetric on the logit scale.

Value

For output = "dataframe", a data.frame. In **long** format the columns are item, threshold (integer; 1 for dichotomous items), location, and – when se = TRUE – se, ci_lower, ci_upper. In **wide** format the columns are item, the threshold locations (t1, t2, ... or location for dichotomous items), and a mean location; when se = TRUE, matching se_t1, se_t2, ... columns are appended. For output = "kable", the same content as a knitr_kable object.

References

Andrich, D. (1978). A rating formulation for ordered response categories. *Psychometrika, 43*(4), 561-573. doi:10.1007/BF02293814

Mair, P., & Hatzinger, R. (2007). Extended Rasch modeling: The eRm package for the application of IRT models in R. *Journal of Statistical Software, 20*(9), 1-20. doi:10.18637/jss.v020.i09

See Also

RMpersonParameters(), RMscoreSE(), RMtargeting()

Examples

```
set.seed(1)
poly <- as.data.frame(
  matrix(sample(0:2, 250 * 5, replace = TRUE), nrow = 250, ncol = 5)
)
colnames(poly) <- paste0("Item", 1:5)
```

```
# Default: long-format kable with SE and 95% CI
RMitemParameters(poly)

# Wide format, point estimates only
RMitemParameters(poly, format = "wide", se = FALSE, output = "dataframe")

# Dichotomous data
dich <- as.data.frame(
  matrix(sample(0:1, 250 * 6, replace = TRUE), nrow = 250, ncol = 6)
)
colnames(dich) <- paste0("Item", 1:6)
RMitemParameters(dich, output = "dataframe")

# Write the parameter table to a CSV (also returned invisibly)
RMitemParameters(poly, output = "file",
                 filename = tempfile(fileext = ".csv"))
```

`RMitemRestscore` *Item Restscore Analysis*

Description

Computes observed and model-expected item-restscore correlations using `iarm::item_restscore()`, and enriches the output with the absolute difference between observed and expected values, item average locations, and item locations relative to the sample mean person location.

Usage

```
RMitemRestscore(
  data,
  cutoff = NULL,
  p_value = NULL,
  correction = c("fwer", "fdr_bh", "fdr_by", "none"),
  alpha = 0.05,
  output = "kable",
  sort,
  p_adj = "BH"
)
```

Arguments

`data`	A data.frame or matrix of item responses. Items must be scored starting at 0 (non-negative integers). Missing values (`NA`) are allowed, but at least one complete case (row with no `NA`) must be present.
`cutoff`	Optional. Default `NULL`, which tests each item against the asymptotic normal reference distribution from `iarm::item_restscore()`, adjusted by `p_adj`. Can be:

- The return value of RMitemRestscoreCutoff: the parametric bootstrap null distribution of the observed minus expected gamma. Adds the columns Diff_low and Diff_high (the descriptive interval), and, with p_value resolving to TRUE, p_restscore and padj_restscore.
- The $item_cutoffs data.frame from RMitemRestscoreCutoff directly (columns Item, diff_low, diff_high). Items are then flagged against the interval.

With a cutoff, the asymptotic p_adjusted column is dropped and p_adj is ignored.

p_value — Logical or NULL. Whether to compute bootstrap p-values from the simulated null distribution and flag on them.

NULL (the default) means **use them when they are available**: TRUE when cutoff is the full RMitemRestscoreCutoff object, which carries the per-item simulated values in its $results element, and FALSE otherwise. Explicit TRUE without the full object is an error rather than a silent downgrade. With FALSE and a cutoff, items are flagged against the interval instead, and a one-time message reports the family-wise error rate that implies.

correction — Character. Multiple-comparison correction applied across items when p_value = TRUE: "fwer" (default) for the Westfall-Young studentised-max step-down (family-wise error rate), "fdr_bh" / "fdr_by" for Benjamini-Hochberg / Benjamini-Yekutieli false discovery rate control, or "none" for uncorrected per-item p-values.

alpha — Numeric in (0, 1). Significance level for the Flagged column when p_value = TRUE. Default 0.05.

output — Character string controlling the return value. Either "kable" (default) for a formatted knitr::kable() table, or "dataframe" for the underlying data.frame.

sort — Optional character string. When sort = "diff", rows are sorted by the absolute magnitude of Difference in descending order, so that both over- and underfitting items appear near the top.

p_adj — Character string specifying the p-value adjustment method passed to iarm::item_restscore(). Default "BH" (Benjamini-Hochberg); use "none" for unadjusted p-values. Run ?stats::p.adjust for the list of available methods. Only used when cutoff = NULL.

Details

Item-restscore correlations using Goodman-Kruskal's gamma (Kreiner, 2011) measure the association between a person's score on a single item and their total score on the remaining items (the "restscore"). Under a correctly fitting Rasch model, observed and model-expected correlations should agree closely.

Item parameters are estimated by conditional maximum likelihood via psychotools::pcmodel() (a dichotomous item is a 2-category PCM); the item-restscore statistic itself comes from iarm::item_restscore() and is conditional on the total score, so it is invariant to the estimation engine. Per-item average locations are the means of the CML thresholds, and the person-location reference is the mean of the Warm WLE estimates.

Relative item location is defined as the item's average location minus the sample mean person location, providing a measure of item targeting.

The `iarm` package must be installed (it is in Suggests, not Imports).

The asymptotic p-value is miscalibrated. Without a `cutoff`, each item is tested with `(observed - expected) / SE` against the standard normal, where the SE is that of the observed gamma alone. The expected gamma is estimated from the same data and correlates with the observed one, so the SE is too large for the difference. The difference is also biased upwards in small samples. Under a true Rasch model the test therefore flags too many items as overfit in small samples and too few as underfit at every sample size, and the adjustment chosen in `p_adj` does not correct either. In simulation, 20 dichotomous items at n = 150 gave at least one BH-flagged item in 13 percent of datasets (30 percent when 1.5 logits off target), almost all of them overfit, while 9 polytomous items at n = 1000 gave about 1 percent. Pass the object from `RMitemRestscoreCutoff()` as `cutoff` for p-values from a parametric bootstrap null instead.

Bootstrap p-values. When `p_value = TRUE`, each item's observed `Difference` is compared against its simulated null distribution (from `cutoff$results`), studentised by the bootstrap mean and SD. Because the bootstrap mean is subtracted, the small-sample upward bias of the difference is removed, and because the bootstrap SD is used, the correlation between observed and expected gamma is accounted for. The marginal p-value is the two-sided Monte-Carlo p-value `(1 + #{|t*| >= |t|}) / (B + 1)`, and `correction = "fwer"` applies the Westfall-Young studentised-max step-down across items (Ferreira, 2024).

The two directions do not get equal shares of alpha. Gamma is bounded at 1, so the null of `Difference` is left-skewed and a two-sided test on `|t|` rejects more often in the long (underfit) tail. In simulation under a true Rasch model the total rate was nominal, but the per-item marginal rate was about 3 percent for underfit and 2 percent for overfit with 20 dichotomous items at n = 150 and 1.5 logits off target, narrowing toward equal shares with polytomous items and larger samples. With `correction = "fwer"`, over four such conditions, at least one item was flagged as underfit in 3.7 percent of datasets and as overfit in 1.6 percent, 5.0 percent in total. An equal-tailed test, with alpha/2 for each direction, was evaluated and not adopted: it detected 2 to 3 percentage points fewer underfitting items and no more overfitting ones.

Reproducibility. The bootstrap p-values depend on the simulated null, so two analyses of the same data with different seeds can disagree about items near the decision boundary. In simulation, in conditions chosen to include such items, two seeds disagreed about the flag of at least one item in about 10 percent of analyses at 400 iterations and about 6 percent at 1000 with `correction = "fwer"`, and in 13 and 11 percent with `correction = "fdr_bh"`. Set `seed` in `RMitemRestscoreCutoff()` for a reproducible analysis, and use 1000 or more iterations for a final one.

The direction in `Flagged` follows the studentised value, not the sign of `Difference`. At small sample sizes the null mean of `Difference` is slightly positive, so an item can be flagged `"underfit"` while its `Difference` is still just above zero. This is rare.

Value

- If `output = "kable"`: a `knitr_kable` object (plain text table via `format = "pipe"`) with columns for item name, observed and expected restscore correlations, the signed difference (observed minus expected), adjusted p-value, the `Flagged` misfit label, and item location relative to the sample mean person location.
- If `output = "dataframe"`: a data.frame with columns `Item`, `Observed`, `Expected`, `Difference`,

p_adjusted, Flagged, and Relative_location. Flagged is "overfit" (observed above expected, adj. p < .05), "underfit" (below, adj. p < .05), or "" (not flagged).

With a cutoff, p_adjusted is replaced by Diff_low and Diff_high, followed by p_restscore (marginal two-sided bootstrap p-value) and padj_restscore (corrected p-value) when p_value resolves to TRUE. Flagged then reflects padj_restscore < alpha, or the interval when p_value = FALSE.

The Difference column is signed (observed minus expected): *positive* values indicate that the item correlates more strongly with the rest-score than the Rasch model predicts (over-discrimination / *overfit*, often associated with local dependence), and *negative* values indicate weaker-than-expected association (under-discrimination / *underfit*, often associated with multidimensionality or noise).

Multiple comparisons

The marginal p-value controls the error rate of a *single* comparison: for one item (or item pair) decided on in advance it is the relevant value. But scanning all k comparisons and flagging whichever fall below alpha tests k hypotheses at once, so the chance of at least one false flag inflates to roughly $1-(1-\alpha)^k$ (e.g. about 34% for $k = 8$ at alpha = 0.05) – even when every marginal p-value is correctly calibrated. The corrected (adjusted) p-value controls this: correction = "fwer" bounds the probability of *any* false flag (strict, lower power), while "fdr_bh" / "fdr_by" bound the expected *proportion* of false flags among those raised (a more lenient middle ground). Rule of thumb: use the marginal p-value for a single pre-specified comparison, and a corrected p-value when screening the whole table – the usual workflow.

Flags depend on the other items

Item infit and item-restscore both judge each item against expectations from a Rasch model fitted to all items, so misfitting items change what is expected of the others. A flag in one direction can therefore produce flags in the other. Overfitting items can push fitting items into the underfit range, and underfitting items, for example from a second dimension, can make fitting items look overfit. How large the effect is depends on how strong the misfit is and how many items share it. In simulation with seven polytomous items, two overfitting items led to about 6 percent of the fitting items being flagged as underfit, and one or two mildly multidimensional items to 1 to 2 percent being flagged as overfit. When items are flagged in both directions, consider removing the clearest misfit and testing again before interpreting the rest.

References

Kreiner, S. (2011). A Note on Item–Restscore Association in Rasch Models. *Applied Psychological Measurement, 35*(7), 557–561. doi:10.1177/0146621611410227

Ferreira, J. A. (2024). Methods of testing a 'small' or 'moderate' number of hypotheses simultaneously. *Journal of Statistical Theory and Practice, 19*(6). doi:10.1007/s42519024004124

See Also

RMitemRestscoreCutoff, RMitemRestscorePlot

Examples

```
if (requireNamespace("iarm", quietly = TRUE)) {
  # Simulate binary item response data (8 items, 200 persons)
  set.seed(42)
  sim_data <- as.data.frame(
    matrix(sample(0:1, 200 * 8, replace = TRUE), nrow = 200, ncol = 8)
  )
  colnames(sim_data) <- paste0("Item", 1:8)

  # Default kable output
  RMitemRestscore(sim_data)

  # Sorted by absolute difference
  RMitemRestscore(sim_data, sort = "diff")

  # Return as data.frame for further processing
  df <- RMitemRestscore(sim_data, output = "dataframe")

  # Bootstrap null distribution, flagging on Westfall-Young corrected
  # p-values (use 1000 or more iterations in a final analysis)
  if (requireNamespace("ggdist", quietly = TRUE)) {
    cutoff_res <- RMitemRestscoreCutoff(sim_data, iterations = 100,
                                        parallel = FALSE, seed = 42)
    RMitemRestscore(sim_data, cutoff = cutoff_res)
  }
}
```

`RMitemRestscoreBoot` *Bootstrap Item-Restscore Misfit Detection*

Description

Non-parametric bootstrap of item-restscore fit using `iarm::item_restscore()`. For each iteration, a sample of size `samplesize` is drawn from `data` with replacement, the appropriate Rasch model is refitted, and item-restscore results are classified as `"overfit"`, `"underfit"`, or `"no misfit"` based on the BH-adjusted p-value (< .05) and the sign of `expected - observed`. The function returns the percentage of iterations in which each item is flagged.

Usage

```
RMitemRestscoreBoot(
  data,
  iterations = 200,
  samplesize = 600,
  parallel = TRUE,
  n_cores = NULL,
  cutoff = 5,
  verbose = FALSE,
```

```
  seed = NULL,
  output = "kable"
)
```

Arguments

data	A data.frame or matrix of item responses. Items must be scored starting at 0 (non-negative integers).
iterations	Integer. Number of bootstrap samples (default 200).
samplesize	Integer. Size of each bootstrap sample (default 600). Must not exceed `nrow(data)`.
parallel	Logical. Use parallel processing via `mirai` if available (default `TRUE`).
n_cores	Integer or `NULL`. Number of parallel workers. When `NULL`, `getOption("mc.cores")` is checked first. If neither is set and `parallel = TRUE`, a warning is issued and execution falls back to sequential processing.
cutoff	Numeric. Items flagged in fewer than this percentage of iterations are excluded from the kable output (default 5). Has no effect when `output = "dataframe"` or `output = "raw"`.
verbose	Logical. Show a progress bar (default `FALSE`).
seed	Integer or `NULL`. Random seed for reproducibility. See easyRasch2-reproducibility for what this guarantees and how it interacts with `parallel`.
output	Either `"kable"` (default) for a formatted `knitr::kable()` table, `"dataframe"` for the per-item summary data.frame, or `"raw"` for the per-iteration long data.frame (useful for custom plotting).

Details

Each iteration uses the asymptotic test from `iarm::item_restscore()` with Benjamini-Hochberg adjustment, so the result inherits that test's calibration at the subsample size. Under a true Rasch model it flags too many items as overfit in small or mistargeted samples of dichotomous items, too few items of any kind in large polytomous samples, and too few underfitting items at every sample size (see `RMitemRestscore()`). The percentage is therefore how often that test flags an item in samples of size `samplesize`, not the probability that the item misfits, and it changes with `samplesize`. A low underfit percentage is weak evidence of fit. For a test of item fit, use `RMitemRestscoreCutoff()` with `RMitemRestscore()`, whose bootstrap p-values are calibrated at the observed sample size. This function is best read as a description of how stable the asymptotic flags are across resamples.

The full-sample model is fitted by CML via `psychotools::pcmodel()` (a dichotomous item is a 2-category partial credit model), and item locations (mean of the grand-mean-centred CML Andrich thresholds) and the mean WLE person location are computed from it — consistent with `RMitemRestscore` and the rest of the package. Each bootstrap iteration draws a sample of size `samplesize` with replacement and refits via `psychotools::pcmodel(..., hessian = FALSE)` for speed; `iarm::item_restscore()` accepts the fitted model.

Conditional infit MSQ (computed once on the full sample via `iarm::out_infit()`) and relative item locations are reported alongside the bootstrap percentages for context. The item-restscore classification and the infit statistic are conditional and engine-invariant; only the relative-item location shifts slightly relative to the previous `eRm` implementation, because it now uses the WLE person mean (finite at extreme scores) rather than the `eRm` MLE mean.

Iterations that fail (e.g., due to convergence issues on a degenerate bootstrap sample) are silently discarded; the caption / `actual_iterations` reflects only successful runs.

Parallel processing is provided by the `mirai` package (optional). Install it with `install.packages("mirai")` to enable parallelisation.

The `iarm` package must be installed (it is in Suggests, not Imports).

Value

- If `output = "kable"`: a `knitr_kable` object listing items flagged in more than `cutoff`% of iterations, with columns Item, Item-restscore result, % of iterations, Conditional MSQ infit, and Relative item location, and a caption noting iteration count, bootstrap size, and number of complete cases.
- If `output = "dataframe"`: a data.frame with one row per item × classification combination (`Item`, `item_restscore`, `n`, `percent`, `Infit_MSQ`, `Relative_location`), including `"no misfit"` rows.
- If `output = "raw"`: a long data.frame with one row per item × successful iteration (`iteration`, `Item`, `item_restscore`, `diff`, `diff_abs`), where `diff = expected - observed`.

References

Kreiner, S. (2011). A Note on Item-Restscore Association in Rasch Models. *Applied Psychological Measurement, 35*(7), 557-561. doi:10.1177/0146621611410227

See Also

RMitemRestscore, RMitemInfit

Examples

```
if (requireNamespace("iarm", quietly = TRUE)) {
set.seed(42)
sim_data <- as.data.frame(
  matrix(sample(0:1, 400 * 8, replace = TRUE), nrow = 400, ncol = 8)
)
colnames(sim_data) <- paste0("Item", 1:8)

# Few iterations for a fast example; use 100+ in real analyses
# Default kable output (only items flagged > cutoff%)
RMitemRestscoreBoot(sim_data, iterations = 50, samplesize = 300,
                parallel = FALSE, seed = 1)

# Per-item summary data.frame (all classifications, including "no misfit")
summary_df <- RMitemRestscoreBoot(sim_data, iterations = 50, samplesize = 300,
                              parallel = FALSE, seed = 1,
                              output = "dataframe")

# Per-iteration long data for custom plotting
raw_df <- RMitemRestscoreBoot(sim_data, iterations = 50, samplesize = 300,
                          parallel = FALSE, seed = 1, output = "raw")
```

```
# Distribution of (expected - observed) across iterations, per item
if (requireNamespace("ggplot2", quietly = TRUE)) {
  library(ggplot2)
  ggplot(raw_df, aes(x = Item, y = diff)) +
    geom_hline(yintercept = 0, linetype = "dashed", colour = "grey50") +
    geom_violin(fill = "grey90", colour = NA) +
    geom_jitter(aes(colour = item_restscore),
                width = 0.15, alpha = 0.5, size = 0.8) +
    scale_colour_manual(values = c("overfit"   = "#377eb8",
                                   "underfit"  = "#e41a1c",
                                   "no misfit" = "grey60")) +
    labs(y = "Expected - observed restscore correlation",
         colour = NULL) +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))
}
}
```

`RMitemRestscoreCutoff` *Simulation-Based Item-Restscore Null Distribution*

Description

Uses parametric bootstrap simulation to build the null distribution of the item-restscore statistic for `RMitemRestscore`. This function simulates data from a correctly fitting Rasch model that mimics your data and returns, per item, the simulated difference between observed and expected item-restscore gamma.

Usage

```
RMitemRestscoreCutoff(
  data,
  iterations = 400,
  parallel = TRUE,
  n_cores = NULL,
  verbose = FALSE,
  seed = NULL,
  cutoff_method = "hdci",
  hdci_width = 0.95,
  dgp = c("conditional", "resample")
)
```

Arguments

`data`	A data.frame or matrix of item responses. Items must be scored starting at 0 (non-negative integers). Only complete cases (rows without any `NA`) are used.
`iterations`	Integer. Number of simulation iterations (default 400).

parallel	Logical. Use parallel processing via mirai if available (default TRUE).
n_cores	Integer or NULL. Number of parallel workers. When NULL, getOption("mc.cores") is checked first. If neither is set and parallel = TRUE, a warning is issued and execution falls back to sequential (single core) processing.
verbose	Logical. Show a progress bar (default FALSE).
seed	Integer or NULL. Random seed for reproducibility. See easyRasch2-reproducibility for what this guarantees and how it interacts with parallel.
cutoff_method	Character string specifying how the intervals are computed. Either "hdci" (default) for the Highest Density Interval via ggdist::hdci(), or "quantile" for the 2.5th/97.5th percentiles via stats::quantile().
hdci_width	Numeric. Width of the HDCI when cutoff_method = "hdci". Default is 0.95 (95% HDCI). Ignored when cutoff_method = "quantile". The interval is a **description** of where a fitting item's difference is expected to fall, not a decision rule. Flagging every item outside a width-w interval tests all k items at once, so the family-wise error rate is 1 - w^k. Decisions should come from the corrected p-value instead (RMitemRestscore with p_value = NULL and the full object returned here).
dgp	Character. Data-generating process for the parametric bootstrap. "conditional" (default) simulates each respondent's pattern from the exact Rasch conditional distribution given their observed total score, item parameters fixed (a *conditional* null). The expected gamma is computed from the observed score distribution, which the conditional null holds fixed. "resample" resamples WLE person locations with replacement and simulates responses under the model (a *marginal* null). In simulation under a true Rasch model, the conditional null gave a family-wise rate of 5.0 percent with correction = "fwer" and the resample null 6.4 percent, pooled over four designs. The conditional null takes about 1.2 to 1.6 times as long. **Experimental.**

Details

The asymptotic test in iarm::item_restscore() divides the observed minus expected gamma by the standard error of the observed gamma alone. The expected gamma is estimated from the same data and correlates with the observed one, so that standard error is too large for the difference, and the difference is also biased upwards in small samples. Under a true Rasch model the resulting test is liberal for dichotomous items in small or mistargeted samples and conservative for polytomous items, in every case flagging too few underfitting items. The bootstrap null replaces the asymptotic reference distribution and absorbs both problems.

The generating model is CML item parameters (via psychotools) with WLE person locations. For each iteration a dataset is simulated under the chosen dgp, the model is **refitted** by CML (psychotools::pcmodel()), and the observed and expected item-restscore gamma are computed as in iarm::item_restscore(), by a faster internal routine that skips the standard errors and the rounding of the printed values. The refit matters: the expected gamma varies from sample to sample because the thresholds do, and holding them fixed would reproduce the problem the bootstrap exists to solve. Failed iterations (e.g., degenerate simulated data) are silently discarded.

Parallel processing is provided by the mirai package (optional). Install it with install.packages("mirai") to enable parallelisation.

The iarm package must be installed (it is in Suggests, not Imports).

Value

A list with components:

`results` data.frame with columns `iteration`, `Item`, `Observed`, `Expected`, `Difference` (one row per item per successful iteration). `Difference` is `Observed - Expected`, the statistic `RMitemRestscore` tests.

`item_cutoffs` data.frame with per-item interval bounds for the difference: `Item`, `diff_low`, `diff_high`. Bounds are computed using the method specified by `cutoff_method`.

`actual_iterations` Number of successful iterations. Everything downstream rests on this rather than on `iterations`, so it is the number to report.

`requested_iterations` The `iterations` argument, kept so callers can tell how many simulated datasets were discarded.

`sample_n` Number of complete cases used.

`sample_n_total` Number of respondents in the raw input data, before the complete-case filter.

`sample_has_na` Logical. Whether the raw input data contained any missing values.

`sample_summary` Summary statistics of estimated person parameters.

`item_names` Character vector of item names from data.

`cutoff_method` The method used to compute the intervals (`"hdci"` or `"quantile"`).

`hdci_width` The HDCI width used (only meaningful when `cutoff_method = "hdci"`).

`dgp` The data-generating process used (`"resample"` or `"conditional"`).

References

Kreiner, S. (2011). A Note on Item-Restscore Association in Rasch Models. *Applied Psychological Measurement, 35*(7), 557-561. doi:10.1177/0146621611410227

Johansson, M. (2026). Simulation-based cutoffs for conditional item fit in Rasch models: Iterations, multiplicity correction, and decision stability. *PsyArXiv*. doi:10.31234/osf.io/7pqz4_v2

See Also

`RMitemRestscore`, `RMitemRestscorePlot`

Examples

```
if (requireNamespace("iarm", quietly = TRUE) &&
    requireNamespace("ggdist", quietly = TRUE)) {
  set.seed(42)
  sim_data <- as.data.frame(
    matrix(sample(0:1, 200 * 10, replace = TRUE), nrow = 200, ncol = 10)
  )
  colnames(sim_data) <- paste0("Item", 1:10)

  # Run 100 iterations sequentially for a quick demo
  cutoff_res <- RMitemRestscoreCutoff(sim_data, iterations = 100,
                                      parallel = FALSE, seed = 42)
  cutoff_res$item_cutoffs
```

```
  # Flag on bootstrap p-values in RMitemRestscore()
  RMitemRestscore(sim_data, cutoff = cutoff_res)
}
```

`RMitemRestscorePlot` *Plot the Simulated Item-Restscore Null Distribution*

Description

Visualises the per-item null distribution of the observed minus expected item-restscore gamma from `RMitemRestscoreCutoff`, optionally overlaying the observed differences from the original data.

Usage

```
RMitemRestscorePlot(simfit, data)
```

Arguments

`simfit`	The return value of `RMitemRestscoreCutoff` (a list with components `results`, `item_cutoffs`, `actual_iterations`, `sample_n`, and `item_names`).
`data`	Optional. A data.frame or matrix of item responses for computing and overlaying the observed item-restscore differences. Items must be scored starting at 0 (non-negative integers). When provided, the plot includes orange diamond markers for the observed difference alongside the simulated distribution, plus segment summaries of the intervals.

Details

Uses `ggdist::stat_dotsinterval()` (when `data` is not supplied) or `ggdist::stat_dots()` (when `data` is supplied) with `point_interval = "median_hdci"`. The outer `.width` follows `simfit$hdci_width`, so the shaded interval matches the one `RMitemRestscore()` tabulates.

The x-axis is the difference between observed and model-expected gamma, the statistic `RMitemRestscore` tests. A dashed line marks zero. The simulated distributions are generally not centred on zero in small samples, which is one of the reasons the asymptotic test is miscalibrated (see `RMitemRestscoreCutoff`). Positive values indicate over-discrimination (overfit), negative values under-discrimination (underfit).

When `data` **is** supplied, the observed differences are computed with `RMitemRestscore` and overlaid as orange diamonds, with per-item intervals drawn as black line segments (thicker for the 66% range) and black dots for the simulated median.

The `ggplot2` and `ggdist` packages must be installed (they are in Suggests, not Imports).

Value

A `ggplot` object.

See Also

RMitemRestscoreCutoff, RMitemRestscore

Examples

```
if (requireNamespace("iarm", quietly = TRUE) &&
    requireNamespace("ggdist", quietly = TRUE) &&
    requireNamespace("ggplot2", quietly = TRUE)) {
  set.seed(42)
  sim_data <- as.data.frame(
    matrix(sample(0:1, 200 * 10, replace = TRUE), nrow = 200, ncol = 10)
  )
  colnames(sim_data) <- paste0("Item", 1:10)

  cutoff_res <- RMitemRestscoreCutoff(sim_data, iterations = 100,
                                      parallel = FALSE, seed = 42)

  # Simulated distribution only
  RMitemRestscorePlot(cutoff_res)

  # With the observed differences overlaid
  RMitemRestscorePlot(cutoff_res, data = sim_data)
}
```

RMlocdepGamma *Partial Gamma Local Dependence Analysis*

Description

Computes partial gamma coefficients for Local Dependence (LD) assessment using `iarm::partgam_LD()`. Each pair of items is tested for residual association, controlling for the rest score (total score minus one of the items in the pair).

Usage

```
RMlocdepGamma(
  data,
  cutoff = NULL,
  p_value = NULL,
  correction = c("fwer", "fdr_bh", "fdr_by", "none"),
  alpha = 0.05,
  output = "kable",
  n_pairs = NULL
)
```

Arguments

data	A data.frame or matrix of item responses. Items must be scored starting at 0 (non-negative integers). Missing values (NA) are allowed, but at least one complete case must exist.
cutoff	Optional. Default NULL (no cutoff applied). Can be: • The return value of RMlocdepGammaCutoff (a list with $pair_cutoffs): the data.frame is extracted automatically and simulation metadata is included in the kable caption. • The $pair_cutoffs data.frame from RMlocdepGammaCutoff directly: must have columns Item1, Item2, gamma_low, gamma_high. When provided, adds columns Gamma_low, Gamma_high, and Flagged (logical; TRUE when the observed partial gamma falls outside the credible range) to the result.
p_value	Logical or NULL. When TRUE, adds one-sided bootstrap p-values for *excess positive* local dependence (p_gamma, padj_gamma), matching the p_value semantics of RMlocdepQ3, and flagged reflects padj_gamma < alpha (positive deviations only) instead of the credible range. One test per item pair: the p-value is computed on gamma_pair, the larger of the two conditioning directions, and repeated in the direction-2 table for the same pair. The asymptotic adjusted p-value and star columns from iarm::partgam_LD() are **dropped** in this mode, and the simulated gamma_low / gamma_high band is kept as the effect-size reference. NULL, the default, means TRUE when cutoff is the **full** RMlocdepGammaCutoff object (it carries the simulated distributions in $results) and FALSE otherwise, so calling with no cutoff keeps the asymptotic table unchanged. Pass FALSE for the pre-1.2.0 behaviour. The interval makes a poor decision rule, since its width sets a family-wise error rate of 1 - width^m over all m pairs at once (Johansson, 2026).
correction	Character. Multiplicity correction for the bootstrap p-values, applied over the family of all item pairs (before any n_pairs display filter): "fwer" (default; Westfall-Young studentised-max step-down), "fdr_bh", "fdr_by", or "none". Ignored when p_value = FALSE.
alpha	Numeric in (0, 1). Significance level used to flag pairs on the corrected p-value. Default 0.05. Ignored when p_value = FALSE.
output	Character string controlling the return value. Either "kable" (default) for a formatted knitr::kable() table, or "dataframe" for the underlying data.frame.
n_pairs	Optional positive integer. When supplied, only the n_pairs item pairs with the largest absolute partial-gamma values (i.e., strongest residual dependence in either direction) are retained per rest-score direction, sorted by |gamma| descending. When NULL (default), all pairs are returned in iarm's native ordering. Values larger than the total number of pairs are silently capped at that total.

Details

Partial gamma (Christensen, Kreiner & Mesbah, 2013) measures the residual association between pairs of items after controlling for the rest score (total score minus one item). Because it matters which item is subtracted, calculations are done for each pair in both directions, yielding two data.frames.

Values near 0 indicate no local dependence. Large positive values suggest positive LD (items share variance beyond the latent trait), while large negative values suggest negative LD.

The `iarm` package must be installed (it is in Suggests, not Imports). The asymptotic adjusted p-value is a Benjamini-Hochberg adjustment of the p-values from `iarm::partgam_LD()` over all $k(k-1)$ tests in both directions, applied by easyRasch2. iarm's own adjusted column is a Bonferroni correction whatever method is named, because iarm adjusts one p-value at a time, so it is not used.

Bootstrap p-values. When `p_value = TRUE`, each pair is tested once, on the larger of its two rest-score directions (the `gamma_pair` column), against its simulated null distribution (from `cutoff$results`, simulated under local independence and built from that same maximum). The per-pair statistic is the residual studentised by the bootstrap mean and SD; the marginal p-value is the one-sided Monte-Carlo p-value `(1 + #\{t* >= t\}) / (B + 1)` for excess *positive* LD (redundancy, the diagnostic target — matching `RMlocdepQ3`), so it can be no smaller than `1 / (B + 1)`. The band still shows both bounds for reference. `correction = "fwer"` uses the Westfall-Young studentised-max step-down over the family of all pairs, which exploits the bootstrap dependence among them (Ferreira, 2024); it is liberal when the simulation is small, so at least 1000 `iterations` in `RMlocdepGammaCutoff()` are recommended (a warning is issued below that). Unlike the asymptotic p-values from `iarm::partgam_LD()`, these are calibrated against the *simulated Rasch null* rather than the asymptotic SE; they are model-conditional and sample-size-sensitive, and are reported alongside the simulated effect-size band, not in place of it.

Value

- If `output = "kable"`: an object of class `"RMlocdepGamma"`. Internally a list with two `knitr_kable` elements, `$direction1` and `$direction2`. In both, the rest score is the total score minus Item 2 (the second column); the two elements list each item pair in the two possible orders, so together they cover both rest-score directions for every pair. Each has columns "Item 1", "Item 2", "Partial gamma", "Adj. p-value (BH)", and "p-value sign." (stars for the BH-adjusted p-value, as `iarm` shows them). When `cutoff` is provided, additional columns "Gamma low", "Gamma high", and "Flagged" are included.

  The object has custom `print()` and `knitr::knit_print()` methods: in the R console it prints the two tables stacked vertically; in a Quarto / R Markdown chunk it renders as two distinct pipe tables. Access the individual tables explicitly as `result$direction1` and `result$direction2` if needed.
- If `output = "dataframe"`: a named list of two data.frames (`$direction1`, `$direction2`) with columns `Item1`, `Item2`, `gamma`, `se`, `lower`, `upper` (95% Wald CI), `padj_bh`, `Significance`. When `cutoff` is provided, columns `gamma_low`, `gamma_high`, and `flagged` are also included. With `p_value = TRUE`, `padj_bh` and `Significance` are replaced by `p_gamma` and `padj_gamma` (identical for a pair in both directions).

Multiple comparisons

The marginal p-value controls the error rate of a *single* comparison: for one item (or item pair) decided on in advance it is the relevant value. But scanning all *k* comparisons and flagging whichever fall below `alpha` tests *k* hypotheses at once, so the chance of at least one false flag inflates to roughly $1-(1-\alpha)^k$ (e.g. about 34% for *k* = 8 at `alpha = 0.05`) – even when every marginal p-value is correctly calibrated. The corrected (adjusted) p-value controls this: `correction = "fwer"` bounds the probability of *any* false flag (strict, lower power), while `"fdr_bh"` / `"fdr_by"` bound the expected *proportion* of false flags among those raised (a more lenient middle ground). Rule of thumb: use

the marginal p-value for a single pre-specified comparison, and a corrected p-value when screening the whole table – the usual workflow.

References

Christensen, K. B., Kreiner, S. & Mesbah, M. (Eds.) (2013). *Rasch Models in Health*, pp. 133–135. ISTE & Wiley. doi:10.1002/9781118574454

Ferreira, J. A. (2024). Methods of testing a 'small' or 'moderate' number of hypotheses simultaneously. *Journal of Statistical Theory and Practice, 19*(6). doi:10.1007/s42519024004124

Westfall, P. H., & Young, S. S. (1993). *Resampling-Based Multiple Testing*. Wiley.

See Also

RMlocdepGammaCutoff, RMlocdepGammaPlot

Examples

```
if (requireNamespace("iarm", quietly = TRUE)) {
  set.seed(42)
  sim_data <- as.data.frame(
    matrix(sample(0:1, 200 * 10, replace = TRUE), nrow = 200, ncol = 10)
  )
  colnames(sim_data) <- paste0("Item", 1:10)

  # Default kable output
  RMlocdepGamma(sim_data)

  # Return as data.frame list
  RMlocdepGamma(sim_data, output = "dataframe")

  # Simulation-based cutoffs (slow): 100+ Monte-Carlo iterations
  if (requireNamespace("ggdist", quietly = TRUE)) {
    cutoff_res <- RMlocdepGammaCutoff(sim_data, iterations = 100,
                                      parallel = FALSE, seed = 42)
    RMlocdepGamma(sim_data, cutoff = cutoff_res)

    # Bootstrap p-values with family-wise (Westfall-Young) correction
    # (use iterations >= 1000 in real analyses for stable p-values)
    RMlocdepGamma(sim_data, cutoff = cutoff_res, p_value = TRUE,
                  output = "dataframe")
  }
}
```

RMlocdepGammaCutoff *Simulation-Based Partial Gamma LD Cutoff Determination*

Description

Uses parametric bootstrap simulation to determine appropriate cutoff values for partial gamma Local Dependence analysis via partgam_LD. Under a correctly fitting Rasch model where items are locally independent, this function generates the expected distribution of partial gamma values per item pair, providing empirical critical values.

Usage

```
RMlocdepGammaCutoff(
  data,
  iterations = 400,
  parallel = TRUE,
  n_cores = NULL,
  verbose = FALSE,
  seed = NULL,
  cutoff_method = "hdci",
  hdci_width = 0.95
)
```

Arguments

data	A data.frame or matrix of item responses. Items must be scored starting at 0 (non-negative integers). Only complete cases (rows without any NA) are used.
iterations	Integer. Number of simulation iterations (default 400, was 250 before 1.2.0). 400 is the calibrated floor for the Westfall-Young correction (Johansson, 2026) and the count a 95\ converge. Use 1000 to 2000 for a final analysis.
parallel	Logical. Use parallel processing via mirai if available (default TRUE).
n_cores	Integer or NULL. Number of parallel workers. When NULL, getOption("mc.cores") is checked first. If neither is set and parallel = TRUE, a warning is issued and execution falls back to sequential (single core) processing.
verbose	Logical. Show a progress bar (default FALSE).
seed	Integer or NULL. Random seed for reproducibility. See easyRasch2-reproducibility for what this guarantees and how it interacts with parallel.
cutoff_method	Character string specifying how cutoff intervals are computed. Either "hdci" (default) for the Highest Density Interval via ggdist::hdci(), or "quantile" for the 2.5th/97.5th percentiles via stats::quantile().
hdci_width	Numeric. Width of the HDCI when cutoff_method = "hdci". Default is 0.95 (95\ describes where a fitting pair's coefficient is expected to fall and is no longer the default decision rule, so the width is chosen to converge at the default iteration count rather than to imply an error rate. Ignored when cutoff_method = "quantile".

Details

For each simulation iteration the function:

1. Resamples person parameters (thetas) with replacement from the WLE person locations.

2. Simulates item response data under a Rasch model (dichotomous via `psychotools::rrm()` or polytomous via an internal partial credit simulator).
3. Computes partial gamma for every item pair in both rest-score directions and keeps the larger. The coefficients are identical to those of `iarm::partgam_LD()`, but are computed by a vectorised internal, since `iarm` also derives the asymptotic standard error and confidence interval that a simulated null does not need and costs roughly two orders of magnitude more per iteration.

Because the data are simulated under the Rasch model, items are locally independent by construction. The distribution of partial gamma values across iterations provides empirical critical values per item pair. Values from real data that fall outside these bounds suggest local dependence that exceeds what would be expected by chance. Failed iterations (e.g., due to convergence issues or degenerate data) are silently discarded.

The generating model uses CML item thresholds via `psychotools::pcmodel()` (a dichotomous item is a 2-category PCM) and WLE person locations, consistent with the rest of the package; responses are simulated with `psychotools::rrm()` (dichotomous) or an internal partial credit score simulator (polytomous).

Parallel processing is provided by the `mirai` package (optional). Install it with `install.packages("mirai")` to enable parallelisation.

The `iarm` package must be installed (it is in Suggests, not Imports).

Value

A list with components:

`results` data.frame with columns `iteration`, `Item1`, `Item2`, and `gamma` (one row per item pair per successful iteration). `gamma` is the larger of the pair's two rest-score directions, the statistic `RMlocdepGamma()` tests. Rows are keyed by the sorted pair order, with `Item1` before `Item2` in column order.

`pair_cutoffs` data.frame with per-pair cutoff summaries: `Item1`, `Item2`, `gamma_low`, `gamma_high`. Bounds are computed using the method specified by `cutoff_method`.

`actual_iterations` Number of successful iterations.

`sample_n` Number of complete cases used.

`sample_n_total` Number of respondents in the raw input data, before the complete-case filter.

`sample_has_na` Logical. Whether the raw input data contained any missing values.

`sample_summary` Summary statistics of estimated person parameters.

`item_names` Character vector of item names from data.

`cutoff_method` The method used to compute cutoffs (`"hdci"` or `"quantile"`).

`hdci_width` The HDCI width used (only meaningful when `cutoff_method = "hdci"`).

References

Christensen, K. B., Kreiner, S. & Mesbah, M. (Eds.) (2013). *Rasch Models in Health*, pp. 133–135. ISTE & Wiley. doi:10.1002/9781118574454

See Also

partgam_LD, RMlocdepGamma, RMlocdepGammaPlot

Examples

```
if (requireNamespace("iarm", quietly = TRUE) &&
    requireNamespace("ggdist", quietly = TRUE)) {
  set.seed(42)
  sim_data <- as.data.frame(
    matrix(sample(0:1, 200 * 10, replace = TRUE), nrow = 200, ncol = 10)
  )
  colnames(sim_data) <- paste0("Item", 1:10)

  # Run 100 iterations sequentially for a quick demo
  cutoff_res <- RMlocdepGammaCutoff(sim_data, iterations = 100,
                                    parallel = FALSE, seed = 42)
  cutoff_res$pair_cutoffs
}
```

RMlocdepGammaPlot *Plot Distribution of Simulated Partial Gamma LD Values*

Description

Visualises the distribution of simulation-based partial gamma LD values from RMlocdepGammaCutoff, optionally overlaying observed partial gamma values computed from real data via partgam_LD.

Usage

```
RMlocdepGammaPlot(simfit, data, items = NULL, n_pairs = NULL)
```

Arguments

simfit	The return value of RMlocdepGammaCutoff (a list with components results, pair_cutoffs, actual_iterations, sample_n, and item_names).
data	Optional. A data.frame or matrix of item responses for computing and overlaying observed partial gamma values. Items must be scored starting at 0 (non-negative integers). When provided, the plot includes orange diamond markers for the observed partial gamma alongside the simulated distribution, plus segment summaries from the cutoff intervals.
items	Optional character vector of item names to include in the plot. Only item pairs where **both** items are in this vector will be shown. When NULL (default), all item pairs are plotted.

`n_pairs` Optional positive integer. When supplied, only the `n_pairs` item pairs with the largest absolute partial gamma values are plotted, sorted by `|gamma|` descending. When `data` is supplied, the ranking uses the *observed* partial gammas (the diamonds you actually want to interpret); otherwise it falls back to the per-pair median of the simulated distributions. Applied *after* the `items` filter when both are supplied. Values larger than the number of available pairs are silently capped.

Details

Uses `ggdist::stat_dotsinterval()` (when `data` is not supplied) or `ggdist::stat_dots()` (when `data` is supplied) with `point_interval = "median_hdci"` and `.width = c(0.66, 0.95, 0.99)`.

The plot shows one row per item pair (labelled as "Item1 - Item2"). The observed overlay is the larger of the pair's two rest-score directions, matching the statistic `RMlocdepGamma()` tests and the simulated null.

When `data` is **not** supplied, the function plots the simulated partial gamma distributions as dot-interval plots using `ggdist::stat_dotsinterval()` with median and Highest Density Continuous Interval (HDCI) summaries.

When `data` **is** supplied, the function:

1. Computes observed partial gamma values via `iarm::partgam_LD()`.
2. Overlays observed gamma values as orange diamond markers on the simulated distributions.
3. Shows per-pair cutoff intervals (from `simfit$pair_cutoffs`) as black line segments, with thicker segments for the 66\ black dots for the median.

The `ggplot2`, `ggdist`, and optionally `iarm` packages must be installed (they are in Suggests, not Imports).

Value

A `ggplot` object.

See Also

`RMlocdepGammaCutoff`, `RMlocdepGamma`

Examples

```
if (requireNamespace("iarm", quietly = TRUE) &&
    requireNamespace("ggdist", quietly = TRUE) &&
    requireNamespace("ggplot2", quietly = TRUE)) {
  set.seed(42)
  sim_data <- as.data.frame(
    matrix(sample(0:1, 200 * 10, replace = TRUE), nrow = 200, ncol = 10)
  )
  colnames(sim_data) <- paste0("Item", 1:10)

  # Run simulation
  cutoff_res <- RMlocdepGammaCutoff(sim_data, iterations = 100,
```

```
                                    parallel = FALSE, seed = 42)

  # Simulated distribution only
  RMlocdepGammaPlot(cutoff_res)

  # With observed partial gamma overlaid
  RMlocdepGammaPlot(cutoff_res, data = sim_data)

  # Plot only a subset of items
  RMlocdepGammaPlot(cutoff_res, data = sim_data,
                    items = c("Item1", "Item2", "Item3"))
}
```

RMlocdepQ3 *Q_3 Residual Correlations for Local Dependence Assessment*

Description

Computes Yen's Q_3 residual correlations between item pairs. By default the Rasch model is fitted by conditional maximum likelihood with WLE person locations (`estimator = "CML"`); marginal ML via `mirt` is available with `estimator = "MML"`. High correlations (above the dynamic cut-off) indicate potential local dependence between items. See `RMlocdepQ3Cutoff` for how to determine the appropriate dynamic cut-off for your data.

Usage

```
RMlocdepQ3(
  data,
  cutoff = NULL,
  output = "kable",
  n_pairs = NULL,
  p_value = NULL,
  correction = c("fwer", "fdr_bh", "fdr_by", "none"),
  alpha = 0.05,
  estimator = c("CML", "MML")
)
```

Arguments

`data`	A data.frame or matrix of item responses. Items must be scored starting at 0 (non-negative integers). Missing values (`NA`) are allowed.
`cutoff`	Optional. `NULL` (default) returns the raw Q_3matrix. A single numeric value returns the Q_3 matrix with a global dynamic cut-off (the value added to the mean off-diagonal Q_3). The **full list** returned by `RMlocdepQ3Cutoff` returns a *list of two tables* (see Value): the cut-off matrix plus a per-pair table.
`output`	Character string controlling the return value. Either `"kable"` (default) for formatted `knitr::kable()` table(s), or `"dataframe"` for the underlying data.frame(s).

`n_pairs` Integer or `NULL` (default). When the full cutoff object is supplied, limits the per-pair table to the `n_pairs` pairs with the largest departure from their expected range. `NULL` shows all pairs.

`p_value` Logical or `NULL`. When `TRUE` the per-pair table reports one-sided bootstrap p-values (`p_q3`, `padj_q3`) and flags above pairs (only) on `padj_q3 < alpha` instead of on the expected range. `NULL`, the default, means `TRUE` when `cutoff` is the full `RMlocdepQ3Cutoff` object and `FALSE` otherwise, so a numeric cutoff or no cutoff keeps the interval. Pass `FALSE` for the pre-1.2.0 behaviour. The interval is a description of where a pair's Q_3 is expected to fall and makes a poor decision rule, since its width sets a family-wise error rate of `1 - width^m` over all m pairs at once, and pairs grow quadratically in items (Johansson, 2026).

`correction` Character. Multiple-comparison correction across item pairs when `p_value = TRUE`: `"fwer"` (default) for the Westfall-Young studentised-max step-down, `"fdr_bh"` / `"fdr_by"` for Benjamini-Hochberg / Benjamini-Yekutieli, or `"none"`.

`alpha` Numeric in (0, 1). Significance level for `Flagged` when `p_value = TRUE`. Default `0.05`.

`estimator` Character. Estimation engine for the Q_3 residual correlations. `"CML"` (default) fits item parameters by conditional maximum likelihood (`psychotools`) and person locations by Warm's weighted likelihood (WLE) – true to the Rasch tradition and finite at extreme scores. `"MML"` uses the marginal-ML / EAP engine (`mirt`), retained for backward compatibility. The estimator must match the one used by `RMlocdepQ3Cutoff`; when the full cutoff object is supplied, its stored estimator takes precedence (a mismatching `estimator` argument is overridden with a warning).

Details

The Q_3 statistic (Yen, 1984) is the correlation between residuals of pairs of items after accounting for the latent trait. Under local independence, Q_3 values are expected to be around $-1/(k-1)$ where k is the number of items. When `cutoff` is supplied, the dynamic cut-off is the mean of all off-diagonal Q_3 values plus `cutoff`, following the approach of Christensen et al. (2017). Use `RMlocdepQ3Cutoff` to obtain a simulation-based cutoff recommendation.

Q_3 is the column-wise correlation matrix of the model standardized residuals $(x - E)/\sqrt{Var}$. By default (`estimator = "CML"`) item parameters are estimated by conditional maximum likelihood (`psychotools`) and person locations by Warm's weighted likelihood; `estimator = "MML"` instead uses `mirt`'s marginal-ML model and its built-in Q_3 residuals. The two estimators give very similar Q_3 values (off-diagonal correlation typically > 0.95); what matters for inference is that the observed Q_3 and the simulated cut-off in `RMlocdepQ3Cutoff` use the *same* estimator, which the functions enforce.

Two views of local dependence. Given the full `RMlocdepQ3Cutoff` object, two complementary tables are returned. The `$matrix` applies a single *global* cut-off (the Christensen et al. approach: the 99th percentile of the simulated max-minus-mean Q_3) – a family-wise "is there any local dependence" overview. The `$pairs` table is the per-comparison view: each observed Q_3 against its own simulated expected range (the `Low`/`High` bounds), so individual dependent pairs can be read off and ranked.

Bootstrap p-values. When `p_value = TRUE`, the `$pairs` table also tests each observed Q_3 against its simulated null (from `cutoff$pair_results`) with a one-sided (upper-tail) test for excess local

dependence. The pair statistic is studentised by the bootstrap mean and SD; the marginal p-value is `(1 + #{Q3* >= Q3}) / (B + 1)`, and `correction` applies the family-wise (Westfall-Young step-down) or FDR adjustment across the $k(k-1)/2$ pairs. As for item fit, the family-wise correction is liberal below 400 `iterations` in `RMlocdepQ3Cutoff()`, which is reported once per session, and between 400 and 1000 the table caption notes that decisions remain somewhat seed-dependent. A false discovery rate correction needs far more iterations than the family-wise one, because a bootstrap p-value cannot fall below `1/(B + 1)` and Benjamini-Hochberg compares the smallest against `alpha/m`. Over the 36 pairs of a nine-item scale that takes 720 iterations, and over the 190 pairs of a twenty-item scale 3799, against 19 for Westfall-Young whatever the number of pairs.

Value

With `cutoff = NULL` or a bare numeric cut-off, a single object (`kable` or data.frame) holding the lower triangle of the Q_3 matrix; a numeric cut-off adds a `Flagged` row flag and a caption describing the dynamic cut-off.

With the **full** `RMlocdepQ3Cutoff()` **object**, a named list of two:

`$matrix` the Q_3 lower-triangle matrix with the *global* dynamic cut-off (mean off-diagonal Q_3 + suggested cut-off), as above.

`$pairs` one row per item pair: `Item1`, `Item2`, `Observed` (Q_3), `Low/High` (the per-pair expected range, i.e. the simulated bounds), and `Flagged` – `"above"` (Q_3 above the upper bound, indicating local dependence), `"below"` (below the lower bound), or `""`. Sorted by absolute departure from the per-pair simulated median and truncated to `n_pairs`. With `p_value = TRUE`, columns `p_q3` and `padj_q3` are added and `Flagged` reflects `padj_q3 < alpha` and **only flags** `"above"`.

The Q_3 tile heatmap that earlier versions returned as `$plot` is now produced by `RMlocdepQ3Plot` (as its `$matrix` element), so the table and plot outputs share the same `$matrix`/`$pairs` structure.

Multiple comparisons

The marginal p-value controls the error rate of a *single* comparison: for one item (or item pair) decided on in advance it is the relevant value. But scanning all *k* comparisons and flagging whichever fall below `alpha` tests *k* hypotheses at once, so the chance of at least one false flag inflates to roughly $1-(1-\alpha)^k$ (e.g. about 34% for *k* = 8 at `alpha = 0.05`) – even when every marginal p-value is correctly calibrated. The corrected (adjusted) p-value controls this: `correction = "fwer"` bounds the probability of *any* false flag (strict, lower power), while `"fdr_bh"` / `"fdr_by"` bound the expected *proportion* of false flags among those raised (a more lenient middle ground). Rule of thumb: use the marginal p-value for a single pre-specified comparison, and a corrected p-value when screening the whole table – the usual workflow.

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. doi:10.1177/014662168400800201

Christensen, K. B., Makransky, G., & Horton, M. (2017). Critical values for Yen's Q_3: Identification of local dependence in the Rasch model. *Applied Psychological Measurement, 41*(3), 178–194. doi:10.1177/0146621616677520

Ferreira, J. A. (2024). Methods of testing a 'small' or 'moderate' number of hypotheses simultaneously. *Journal of Statistical Theory and Practice, 19*(6). doi:10.1007/s42519024004124

Johansson, M. (2026). Simulation-based cutoffs for conditional item fit in Rasch models: Iterations, multiplicity correction, and decision stability. *PsyArXiv*. doi:10.31234/osf.io/7pqz4_v2

Examples

```
# Simulate binary item response data (10 items, 200 persons)
set.seed(42)
sim_data <- as.data.frame(
  matrix(sample(0:1, 200 * 10, replace = TRUE), nrow = 200, ncol = 10)
)
colnames(sim_data) <- paste0("Item", 1:10)

# Raw Q3 matrix (no cutoff)
RMlocdepQ3(sim_data)

# Get the underlying data.frame
q3_df <- RMlocdepQ3(sim_data, output = "dataframe")

# Simulation-based cutoff (use 500+ iterations in real analyses)
if (requireNamespace("ggdist", quietly = TRUE)) {
  cutoff_res <- RMlocdepQ3Cutoff(sim_data, iterations = 50, parallel = FALSE)

  # Bare numeric cutoff -> just the matrix
  RMlocdepQ3(sim_data, cutoff = cutoff_res$suggested_cutoff)

  # Full object -> list of two tables: $matrix and $pairs
  res <- RMlocdepQ3(sim_data, cutoff = cutoff_res, output = "dataframe")
  res$pairs

  # Top 5 pairs, with bootstrap p-values (use iterations >= 1000 in practice)
  RMlocdepQ3(sim_data, cutoff = cutoff_res, n_pairs = 5, p_value = TRUE,
             output = "dataframe")$pairs
}
```

`RMlocdepQ3Cutoff` *Simulation-Based Q_3 Cutoff Determination*

Description

Uses parametric bootstrap simulation to determine an appropriate cutoff value for `RMlocdepQ3`. Under a correctly fitting Rasch model, Q_3 residuals have an unknown distribution; this function simulates that distribution and returns empirical percentiles.

Usage

```
RMlocdepQ3Cutoff(
  data,
  iterations = 400,
  parallel = TRUE,
  n_cores = NULL,
  verbose = FALSE,
  seed = NULL,
  cutoff_method = "hdci",
  hdci_width = 0.95,
  estimator = c("CML", "MML"),
  dgp = c("resample", "conditional")
)
```

Arguments

data	A data.frame or matrix of item responses. Items must be scored starting at 0 (non-negative integers).
iterations	Integer. Number of simulation iterations (default 400). 400 is the calibrated floor for the Westfall-Young correction (Johansson, 2026) and the count a 95\ 1000 to 2000 for a final analysis.
parallel	Logical. Use parallel processing via `mirai` if available (default `TRUE`).
n_cores	Integer or `NULL`. Number of parallel workers. When `NULL`, `getOption("mc.cores")` is checked first. If neither is set and `parallel = TRUE`, a warning is issued and execution falls back to sequential (single core) processing.
verbose	Logical. Show a progress bar (default `FALSE`).
seed	Integer or `NULL`. Random seed for reproducibility. See easyRasch2-reproducibility for what this guarantees and how it interacts with `parallel`.
cutoff_method	Character. Method used to compute per-pair Q_3 credible intervals in `pair_cutoffs`. One of `"hdci"` (the default, Highest Density Continuous Interval via `ggdist::hdci()`) or `"quantile"` (symmetric 2.5th / 97.5th percentiles). Only affects `pair_cutoffs`; the global `$suggested_cutoff` (99th percentile of `max(Q3) - mean(Q3)`) is unaffected.
hdci_width	Numeric in (0, 1). Width of the HDCI when `cutoff_method = "hdci"`. Default `0.95`, was `0.99` before 1.2.0. The interval describes where a fitting pair's Q_3 is expected to fall and is no longer the default decision rule, so the width is chosen to converge at the default iteration count rather than to imply an error rate. Ignored when `cutoff_method = "quantile"`.
estimator	Character. Estimation engine for the simulated Q_3 values, passed through to the per-iteration computation. `"CML"` (default) uses CML item parameters and WLE person locations; `"MML"` uses `mirt`. This must match the `estimator` later given to RMlocdepQ3; the value is stored in the returned object's `$estimator` and reused automatically.
dgp	Character. Data-generating process for the parametric bootstrap. `"resample"` (default) draws person locations by resampling the WLE estimates with replacement and simulates responses under the model – a *marginal* null. Each resampled respondent keeps their pattern of missing responses, so incomplete data are

simulated as incomplete and the pairwise-complete Q_3 of the null rests on as many respondents per pair as the observed one. `"conditional"` instead simulates each respondent's pattern from the exact Rasch conditional distribution given their observed total score (and answered items), with item parameters fixed – a *conditional* null that fixes the score margin and needs no latent distribution, avoiding the over-dispersion of resampled point estimates. The two give different cut-offs; see the package's comparison study. **Experimental.**

Details

The generating model is fitted once: CML item parameters (via `psychotools`) and WLE person locations. For each simulation iteration, those WLE thetas are resampled with replacement, response data are simulated under the Rasch / Partial Credit model, the model is refitted, and Q_3 residuals are computed under `estimator`. The distribution of `max(Q3) - mean(Q3)` across iterations provides empirical critical values. Failed iterations (e.g., due to convergence issues) are silently discarded.

Supports both **dichotomous** data (simulated via `psychotools::rrm()`) and **polytomous** data (via an internal partial credit score simulator).

Parallel processing is provided by the `mirai` package (optional). Install it with `install.packages("mirai")` to enable parallelisation.

Value

A list with components:

`results` data.frame with columns `mean`, `max`, `diff` (one row per successful iteration).

`pair_results` Long data.frame with columns `Item1`, `Item2`, `Q3`, `iteration` — one row per item pair per successful iteration. Used by `RMlocdepQ3Plot`.

`pair_cutoffs` data.frame with per-pair cutoff summaries: `Item1`, `Item2`, `Q3_low`, `Q3_high`. Boundaries are computed via the method specified by `cutoff_method`.

`actual_iterations` Number of successful iterations.

`sample_n` Number of persons used: respondents with no responses at all (all-`NA` rows) are dropped, as in `RMlocdepQ3`; incomplete response patterns are retained.

`sample_n_total` Number of respondents in the raw input data, before dropping all-`NA` rows.

`sample_has_na` Logical. Whether the data contained any missing values.

`sample_summary` Summary statistics of estimated person parameters.

`item_names` Character vector of item names from `data`.

`max_diff`, `sd_diff` Max and SD of the `diff` distribution.

`p95`, `p99`, `p995`, `p999` Empirical percentiles of `diff`.

`suggested_cutoff` The 99th percentile (`p99`) — recommended scalar cutoff for `RMlocdepQ3`.

`cutoff_method` The method used for `pair_cutoffs` (`"hdci"` or `"quantile"`).

`hdci_width` The HDCI width used (only meaningful when `cutoff_method = "hdci"`).

`estimator` The estimator used for the simulated Q_3 (`"CML"` or `"MML"`); reused by `RMlocdepQ3` and `RMlocdepQ3Plot`.

`dgp` The data-generating process used (`"resample"` or `"conditional"`).

See Also

RMlocdepQ3

Examples

```
if (requireNamespace("ggdist", quietly = TRUE)) {
  set.seed(42)
  sim_data <- as.data.frame(
    matrix(sample(0:1, 200 * 10, replace = TRUE), nrow = 200, ncol = 10)
  )
  colnames(sim_data) <- paste0("Item", 1:10)

  # Few iterations for a fast example; use 500+ in real analyses
  cutoff_res <- RMlocdepQ3Cutoff(sim_data, iterations = 50, parallel = FALSE,
                                 seed = 42)
  cutoff_res$suggested_cutoff  # 99th percentile

  # Use the cutoff in RMlocdepQ3()
  RMlocdepQ3(sim_data, cutoff = cutoff_res$suggested_cutoff)
}
```

RMlocdepQ3Plot *Plot Distribution of Simulated Q_3 Residual Correlations*

Description

Visualises the distribution of simulation-based Yen's Q_3 residual correlations per item pair from RMlocdepQ3Cutoff, optionally overlaying observed Q_3 values computed from real data via `mirt::residuals(..., type = "Q3")`.

Usage

```
RMlocdepQ3Plot(simfit, data, items = NULL, n_pairs = NULL)
```

Arguments

simfit	The return value of RMlocdepQ3Cutoff (a list with components `pair_results`, `pair_cutoffs`, `actual_iterations`, `sample_n`, and `item_names`).
data	Optional. A data.frame or matrix of item responses for computing and overlaying observed Q_3 values. Items must be scored starting at 0 (non-negative integers). When provided, the plot includes orange diamond markers for the observed Q_3 alongside the simulated distribution, plus segment summaries from the cutoff intervals.
items	Optional character vector of item names to include in the plot. Only item pairs where **both** items are in this vector will be shown. When `NULL` (default), all item pairs are plotted.

`n_pairs` Optional positive integer. When supplied, only the `n_pairs` item pairs with the largest **deviation** from the simulated null are plotted, sorted by `|observed Q3 - median(simulated Q3 p` descending when `data` is supplied, or by `|median(simulated Q3 per pair)|` otherwise. Applied *after* the `items` filter when both are supplied. Values larger than the number of available pairs are silently capped.

Details

Uses `ggdist::stat_dotsinterval()` (when `data` is not supplied) or `ggdist::stat_dots()` (when `data` is supplied) with `point_interval = "median_hdci"` and `.width = c(0.66, 0.95, 0.99)`.

The `$pairs` plot shows one row per item pair (labelled as "Item1 - Item2"). Only the upper triangle of the Q_3 matrix is plotted (pairs are unordered under symmetric Q_3, unlike partial gamma which is direction-dependent).

When `data` is **not** supplied, the function plots the simulated Q3 distributions as dot-interval plots using `ggdist::stat_dotsinterval()` with median and Highest Density Continuous Interval (HDCI) summaries.

When `data` **is** supplied, the function:

1. Computes observed Q_3 residual correlations under the same estimator used to build `simfit` (its `$estimator`: CML/WLE by default, or MML via `mirt`).
2. Overlays observed Q_3 values as orange diamond markers on the simulated distributions.
3. Shows per-pair cutoff intervals (from `simfit$pair_cutoffs`) as black line segments, with thicker segments for the 66\ interval and black dots for the median.

The `ggplot2`, `ggdist`, `mirt`, and `scales` packages must be installed (most are in Suggests, not Imports).

Value

A named list of two `ggplot` objects (mirroring the `$matrix` / `$pairs` structure of `RMlocdepQ3`'s table output):

`$pairs` the per-pair plot described below (always returned).

`$matrix` a lower-triangle tile heatmap of the **observed** Q_3 matrix, with pairs above the global dynamic cut-off outlined. This needs the observed data, so it is `NULL` (with a message) when `data` is not supplied. When `items` is given, the heatmap is subset to those items; `n_pairs` does not apply to it.

See Also

`RMlocdepQ3`, `RMlocdepQ3Cutoff`, `RMlocdepGammaPlot`

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE) &&
    requireNamespace("ggdist", quietly = TRUE)) {
  set.seed(42)
  sim_data <- as.data.frame(
    matrix(sample(0:1, 200 * 10, replace = TRUE), nrow = 200, ncol = 10)
```

```
  )
  colnames(sim_data) <- paste0("Item", 1:10)

  # Run simulation (use more iterations, e.g. 500+, in real analyses)
  cutoff_res <- RMlocdepQ3Cutoff(sim_data, iterations = 50,
                                 parallel = FALSE, seed = 42)

  # Simulated distribution only
  RMlocdepQ3Plot(cutoff_res)

  # With observed Q3 overlaid
  RMlocdepQ3Plot(cutoff_res, data = sim_data)

  # Top 10 pairs by departure from null
  RMlocdepQ3Plot(cutoff_res, data = sim_data, n_pairs = 10)
}
```

RMpersonChange *Single-subject change between two occasions*

Description

Tests, for each respondent, whether their person location moved between two occasions by more than measurement error allows. The statistic is the Rasch change index, $RCI = (\hat{\theta}_2 - \hat{\theta}_1)/SE_{diff}$, referred by default to its exact null distribution rather than a normal one.

Usage

```
RMpersonChange(
  data_t1,
  data_t2,
  id = NULL,
  anchor = "stack",
  item_params = NULL,
  method = "WLE",
  estimator = "CML",
  null = "measurement",
  retest_sd = NULL,
  critical = "exact",
  alpha = 0.05,
  direction = "two.sided",
  conditional_crit = FALSE,
  sim_iter = 1000,
  parallel = TRUE,
  n_cores = NULL,
  seed = NULL,
  theta_range = c(-10, 10),
```

```
  verbose = FALSE,
  output = "dataframe"
)
```

Arguments

data_t1, data_t2	Data.frames or matrices of item responses at the two occasions. Same items, in the same order, one row per respondent, with row `i` of each being the same person. Items must be scored from 0.
id	Optional vector of respondent identifiers, length `nrow(data_t1)`. Defaults to row numbers.
anchor	Character. Where the item calibration comes from: `"stack"` (default, both occasions calibrated together), `"t1"`, or `"t2"`. Ignored when `item_params` is supplied.
item_params	Optional pre-specified item parameters, either a named list of Andrich-threshold vectors or the long-format data.frame from `RMitemParameters()`, as in `RMpersonParameters()`. **Required for a single respondent**, since item parameters cannot be estimated from one person. Supplied thresholds are used as given and are not re-centred.
method	Character. Person-location estimator, `"WLE"` (default) or `"EAP"`.
estimator	Character. How item parameters are estimated when `item_params` is `NULL`: `"CML"` (default) or `"MML"`.
null	Character. Which null hypothesis is tested. `"measurement"` (default) treats the response process as the only source of variation. `"retest"` additionally treats occasion-to-occasion fluctuation as noise and requires `retest_sd`. See Details, and note that the two answer different questions.
retest_sd	Numeric, or `NULL` (default). The **per-occasion** SD, in logits, of a respondent's occasion-specific deviation. Required when `null = "retest"` and not accepted otherwise. Estimate it with `RMretestSD()`.
critical	How the critical values are obtained. `"exact"` (default) enumerates the null distribution, `"simulate"` estimates it by Monte Carlo, and a positive number is used as a symmetric cutoff on the RCI referred to a normal null (for example `1.96`).
alpha	Numeric in (0, 1). Error rate for a single respondent. Default `0.05`.
direction	Character. `"two.sided"` (default), or `"increase"` / `"decrease"` for a one-sided test. These name the direction of θ; whether an increase is an improvement depends on how the scale is oriented.
conditional_crit	Logical. If `TRUE`, critical values are computed separately for each respondent at their own location (enumerated or simulated, per `critical`) rather than pooled across the sample. Default `FALSE`. Ignored when `critical` is numeric.
sim_iter	Integer. Simulation iterations when `critical = "simulate"`. Default `1000`.
parallel	Logical. Use `mirai` for the simulation if available. Default `TRUE`. Only used when `critical = "simulate"`.

n_cores	Integer or NULL. Parallel workers. When NULL, getOption("mc.cores") is checked first; if neither is set, the simulation runs sequentially. Only used when critical = "simulate".
seed	Integer or NULL. Random seed. See easyRasch2-reproducibility. Only used when critical = "simulate".
theta_range	Numeric length 2. Search range for the WLE root and bounds for the EAP grid. Default c(-10, 10).
verbose	Logical. Print a progress bar for the simulation. Default FALSE. Only used when critical = "simulate".
output	Character. "dataframe" (default), "kable", or "ggplot".

Details

Which null is tested. Following Zumbo (2026), the estimand is fixed by what counts as systematic and what counts as residual, and that choice has to be made before an estimator is picked. Two are available here:

- null = "measurement", the default. The respondent's θ is fixed and identical at both occasions and only the response process is random, so $SE^2_{diff} = SE^2_1 + SE^2_2$. This asks whether the two estimates differ by more than responding alone would produce.
- null = "retest". Occasion-to-occasion fluctuation that is not change in the construct (state, mood, recall, practice) also counts as noise, so $SE^2_{diff} = SE^2_1 + SE^2_2 + 2\sigma^2_{retest}$. The factor of 2 is there because retest_sd is a per-occasion SD and both occasions carry one.

The default is the narrower of the two, so it flags more change than a retest-based null would. It is the only one computable from two response vectors alone, which is why it is the default, but a significant result under it is a necessary condition for change rather than evidence of it. retest_sd_tip reports, for each flagged respondent, how large σ_{retest} would have to be to overturn their result.

Caronni et al. (2026) compute $1.96\sqrt{SE^2_A + SE^2_B}$ from the Rasch score-to-measure map and call it a minimal detectable change. That is the null = "measurement" quantity, whereas the classical MDC is built from a test-retest ICC and is the null = "retest" quantity. The two are not interchangeable, which is why the null in force is named in every caption here.

Regression to the mean is present under both nulls and is addressed by neither. A regression-based reliable change index (Maassen, 2004) is out of scope for a single-subject function, since it needs population parameters the individual case does not supply.

Why 1.96 is the wrong cutoff. The sum score is a sufficient statistic, so $\hat{\theta}$ takes only as many values as there are scores, and $SE(\hat{\theta})$ is a deterministic function of the score rather than a constant. The denominator of the RCI is therefore not independent of its numerator: a respondent who moves to an extreme score produces a large change and a large standard error together, and the ratio is damped. Both tails of the null are pulled in, so the critical value sits below 1.96, and the shortfall grows as the scale shortens. Enumerated values for one family of items with four response categories: 1.62 at four items, 1.70 at six, 1.79 at ten, 1.85 at twenty, 1.92 at forty. Response categories matter much less than item count here, and almost all of their effect is the step from dichotomous to polytomous: at six items the value moves from 1.55 with two categories to 1.72 with three and only 1.78 with seven, while reliability over the same range climbs from about .60 to about .91. Referring the RCI to a normal null is conservative, and materially so on short scales.

How the null is obtained. `critical = "exact"` enumerates it. Because the score is sufficient, $\hat{\theta}(r)$ and $SE(r)$ are deterministic lookups and $P(r \mid \theta)$ follows from the Lord-Wingersky recursion, so the null is a discrete distribution over score pairs with no Monte Carlo error. Respondents are grouped by their pair of answered-item sets, so incomplete data is enumerated within each pattern. Under `null = "retest"` the occasion deviations are integrated out by quadrature. `critical = "simulate"` estimates the same distribution by drawing response vectors instead, which is slower and noisier but makes no use of sufficiency.

Quantiles are taken from the **signed** RCI, so `crit_lower` and `crit_upper` are reported separately. Under the null the two occasions are exchangeable, which makes the RCI symmetric about zero however skewed $\hat{\theta}$ itself is, and the two bounds then agree in magnitude. They part company when exchangeability fails, most commonly when the occasions differ in which items were answered.

The pooled critical value is a property of the sample as well as the items, because it averages over where the respondents sit. Targeting matters more than the shape of the distribution: skewing the sample barely moves it, while pushing the same sample off target until 40 percent of respondents score at an extreme moved it from 1.70 to 1.52 in one check.

What the RCI measures, and what it does not. The RCI answers whether a change is distinguishable from the null. It is not a measure of how much someone changed, and the two orderings genuinely differ. Its value mixes two ingredients that it cannot separate: how far the respondent moved, and how precisely each of their two positions was pinned down. On a scale of six items with four response categories the standard error runs from 0.49 in the middle to 1.45 at the boundary, a factor of three within one instrument. A respondent going from the minimum to the maximum score moves 7.24 logits and scores RCI 3.53, while one going from 3 to 15 moves 3.11 logits and scores 3.59. The first traversed the whole scale, the second less than half of it, and the second has the larger statistic. Both statements are correct about their own question.

In a group comparison every case shares one standard error, so ordering by the test statistic matches ordering by effect. Here the scale factor changes from respondent to respondent, and the output is read one person at a time, which is where the misreading does damage. Read `change` for magnitude, in logits, and `rci` only for the decision. Do not rank respondents by `rci`, and do not compare RCIs across instruments or across regions of one scale as though they were a common currency.

Replacing 1.96 fixes the reference distribution, not the estimand. An exact or simulated critical value under `null = "measurement"` is a well calibrated answer to the measurement-error question, not to the retest question.

Calibration and its precondition. `anchor = "stack"` calibrates on both occasions row-bound together. It uses all the data and puts both occasions on one metric by construction, at the cost of assuming the items behave the same way on both occasions. When they do not, the compromise calibration pulls the occasions toward each other and shrinks apparent change. Test that precondition first with the package's DIF tools, using occasion as the grouping variable on the stacked data. `anchor = "t1"` measures follow-up on the baseline metric and keeps post-treatment data out of it, at the cost of halving the calibration sample. `"t2"` is the mirror.

Note that stacking puts each respondent in the data twice, so the item parameters' own standard errors are optimistic. That does not propagate into the person standard errors, which treat the thresholds as fixed.

Conditional independence between occasions. Both nulls assume that, given θ, the two response vectors are independent. That is a conditional statement, and it does not rule out the two estimates being strongly correlated, which they are, since both measure the same person. What it rules out

is carry-over: a time-2 answer depending on the time-1 answer to the same item beyond what θ explains. Administering the occasions separately does not deliver it.

Carry-over does not make this test liberal. It makes it conservative, because positive carry-over shrinks the spread of the RCI below what the enumerated null assumes. The damage is in the direction of the verdicts. Under the model of Marais (2009), which shifts the time-2 location of an item by $(1 - 2x_{j1})d$, the observed change acquires an outward drift that is zero at the item mean, positive above it and negative below it. On twenty items with four categories and no true change whatever, the simulated mean observed change reached +0.46 logits at $\theta = 2.5$ and -0.47 at $\theta = -2.5$ when $d = 1.5$. Two consequences follow. False positives stop dividing evenly by direction, reaching 97 percent "increase" among respondents above the items. And a uniform true change is read as a gradient in baseline severity: a real improvement of 0.50 logits for everyone came out as 0.14 for respondents below the items and 0.55 for those above. `null = "retest"` does not address this, since it adds symmetric variance while the drift is a shift in the conditional mean.

Screen for it before relying on a change verdict. Bind the two occasions side by side into one analysis of 2k items, then compare the k same-item pairs against a simulated Q3 cutoff, centred on the other cross-occasion pairs:

```
rack <- cbind(data_t1, setNames(data_t2, paste0(names(data_t2), "_t2")))
k    <- ncol(data_t1)
ct   <- RMlocdepQ3Cutoff(rack)$suggested_cutoff
q3   <- as.matrix(RMlocdepQ3(rack, output = "dataframe"))
cr   <- q3[(k + 1):(2 * k), seq_len(k)]
same <- q3[cbind(k + seq_len(k), seq_len(k))] -
          mean(cr[row(cr) != col(cr)], na.rm = TRUE)
sum(same > as.numeric(ct))
```

Binding side by side rather than stacking is what makes the correlation computable, since `anchor = "stack"` puts the two occasions in different rows. In simulation this flags no pair when there is no carry-over and catches $d = 0.5$ in all 25 runs at `n = 500` on twenty items. Real change does not set it off: a uniform shift is absorbed into the time-2 item locations, and change that varies between respondents lifts the within-occasion pairs rather than the same-item ones.

A flag means the change verdicts should not be trusted, and in particular that any apparent relationship between baseline and change may be an artifact. The package offers no correction for it.

`retest_sd_tip`**.** For a respondent whose change is flagged, this is the per-occasion retest SD that would bring their RCI back to the critical value, $\sigma^2_{tip} = (\text{change}^2/c^2 - SE_1^2 - SE_2^2)/2$. It is `NA` for respondents not flagged, for whom the question does not arise. The critical value is held fixed at the one in force, which is exact when `critical` is numeric and an approximation otherwise, since an exact or simulated critical value drifts toward ± 1.96 as the added normal component smooths the discreteness.

Value

- If `output = "dataframe"`: one row per respondent, with columns `id`, `sum_t1`, `sum_t2`, `theta_t1`, `se_t1`, `theta_t2`, `se_t2`, `extreme_t1`, `extreme_t2`, `change`, `se_diff`, `rci`, `p_value`, `crit_lower`, `crit_upper`, `change_class`, and `retest_sd_tip`. Attributes `null`, `retest_sd`, `anchor`, `alpha`, `direction`, `critical` and `sim_iter` (`NA` unless simulated) record the analysis.

- If `output = "kable"`: the same content as a `knitr_kable`.
- If `output = "ggplot"`: occasion 1 against occasion 2, with the no-change band and points coloured by `change_class`.

References

Caronni, A., et al. (2026). Improving single-subject change assessment: deriving the minimal detectable change of questionnaires' ordinal scores from the Rasch analysis measures.

Jacobson, N. S., & Truax, P. (1991). Clinical significance: A statistical approach to defining meaningful change in psychotherapy research. *Journal of Consulting and Clinical Psychology, 59*(1), 12-19. doi:10.1037/0022006X.59.1.12

Marais, I. (2009). Response dependence and the measurement of change. *Journal of Applied Measurement, 10*(1), 17-29.

Maassen, G. H. (2004). The standard error in the Jacobson and Truax Reliable Change Index. *Journal of Clinical and Experimental Neuropsychology, 26*(5), 643-657. doi:10.1080/13803390409609791

Zumbo, B. D. (2026). Conditional standard error of measurement as an estimand of individual score precision. *Psychometrika*. doi:10.1017/psy.2026.10141

See Also

`RMretestSD()`, `RMpersonParameters()`, `RMreliabilityCurve()`, `RMscoreSE()`

Examples

```
set.seed(1)
thr <- lapply(seq(-1.2, 1.2, length.out = 6), function(b) b + c(-0.8, 0, 0.8))
theta <- rnorm(80, 0, 1.4)
t1 <- as.data.frame(easyRasch2:::sim_partial_score(thr, theta))
t2 <- as.data.frame(easyRasch2:::sim_partial_score(thr, theta + 0.8))
colnames(t1) <- colnames(t2) <- paste0("I", 1:6)

RMpersonChange(t1, t2)

# A single respondent needs an external calibration
names(thr) <- paste0("I", 1:6)
RMpersonChange(t1[1, ], t2[1, ], item_params = thr)

# Monte Carlo alternative, for comparison
RMpersonChange(t1, t2, critical = "simulate", sim_iter = 200,
               parallel = FALSE, seed = 1)
```

`RMpersonFit` *Person-Fit Statistics for a Rasch / Partial Credit Model*

Description

Computes per-respondent person-fit statistics and resampling-based p-values. Three statistics are reported: conditional **infit** and **outfit** mean-squares (MSQ) – the magnitude (effect-size) measures familiar from Rasch analysis – and the standardized log-likelihood statistic **lz**. The conditional MSQ statistics use response probabilities conditional on the total score, which require no person estimate and are therefore unbiased (Kreiner & Christensen, 2011); they are computed on each person's observed response pattern, so partial missingness is handled directly.

Usage

```
RMpersonFit(
  data,
  statistics = c("infit", "outfit", "lz"),
  estimator = c("CML", "MML"),
  theta_method = c("WLE", "EAP"),
  iterations = 500L,
  flag_alpha = 0.05,
  flag = c("both", "underfit"),
  zstd = FALSE,
  parallel = FALSE,
  n_cores = NULL,
  seed = NULL,
  output = c("kable", "dataframe", "ggplot", "list")
)
```

Arguments

`data`	A data.frame or matrix of item responses. Items must be scored starting at 0 (non-negative integers). Missing values (`NA`) are allowed.
`statistics`	Character vector. Which statistics to report; any of `"infit"`, `"outfit"`, `"lz"`. Defaults to all three.
`estimator`	Character. How item parameters are estimated: `"CML"` (default) or `"MML"` (via **mirt**).
`theta_method`	Character. Person-location estimator used for the lz statistic: `"WLE"` (default) or `"EAP"`. Ignored if `"lz"` is not requested.
`iterations`	Integer. Number of Monte-Carlo replications per person. `0` skips resampling and returns statistics only. Default `500`.
`flag_alpha`	Numeric in (0, 1). Significance level for the `flagged` column and the plot highlighting. Default `0.05`.

`flag` Character. Which misfit direction drives flagging for the MSQ statistics: `"both"` (default) flags either direction with a two-sided p-value; `"underfit"` flags only underfit (MSQ > 1, noisy responding) with a one-sided upper p-value, which is more powerful for detecting it and ignores (benign) overfit. The reported `p_infit` / `p_outfit` follow this choice. lz is one-sided regardless. Overfit can occasionally be suspicious (e.g. fabricated or copied response patterns), so `"both"` is the default.

`zstd` Logical. If `TRUE`, also report the Wilson-Hilferty standardized (ZSTD) versions of the MSQ statistics. These are provided only as a familiar cross-person comparability metric and are **not** used for inference (see Müller, 2020). Default `FALSE`.

`parallel, n_cores` Logical / integer. Parallelise the resampling across persons via **mirai** when available. Default sequential. When `n_cores` is `NULL`, `getOption("mc.cores")` is used, and two workers when that is unset too.

`seed` Optional integer for reproducible resampling. See easyRasch2-reproducibility for what this guarantees and how it interacts with `parallel`.

`output` Character. `"kable"` (default), `"dataframe"`, `"ggplot"`, or `"list"`. For `"ggplot"`, a **named list** of person-fit maps is returned – one per requested statistic (e.g. `$infit`, `$outfit`, `$lz`) – each plotting the statistic against person location, with respondents flagged by *that* statistic highlighted and a caption reporting the assessed sample size and the proportion flagged. For the MSQ maps the flagged count is split into underfit (MSQ > 1, noisy/erratic responding) and overfit (MSQ < 1, overly deterministic responding).

Details

Statistical significance is assessed by Monte-Carlo resampling under the fitted model rather than by an assumed asymptotic distribution. The asymptotic null distributions of MSQ and lz are known to be unreliable – the Wilson-Hilferty (ZSTD) transformation of MSQ in particular adds nothing once conditional estimation is used (Müller, 2020) – so resampling provides the valid reference (Sinharay, 2016).

Conditional MSQ. For person v with answered items A_v and total score r_v, the conditional residual is $z_{vi} = (x_{vi} - E(X_{vi} \mid R_v = r_v))/\sqrt{\mathrm{Var}(X_{vi} \mid R_v = r_v)}$, with conditional moments obtained from elementary symmetric functions of the answered items' parameters. Outfit is the unweighted mean of z_{vi}^2; infit is the information-weighted mean. Because the conditional moments use only item parameters, no (biased) person estimate enters the residual.

Interpreting MSQ direction. Values near 1 indicate fit. MSQ > 1 (*underfit*) means the responses are noisier / more erratic than the model expects – the validity-relevant direction, indicating careless or aberrant responding. MSQ < 1 (*overfit*) means responses are overly deterministic (Guttman-like); usually benign for score validity, though a suspiciously perfect pattern can occasionally signal copied or fabricated data. Use `flag = "underfit"` to flag only the former. lz is one-sided: low (negative) values flag the aberrant (underfit-like) direction.

lz. The standardized log-likelihood of the response pattern evaluated at the estimated person location (Drasgow, Levine & Williams, 1985); small (negative) values indicate misfit. Because the location is estimated rather than known, the variance of lz is below 1 and its asymptotic standard-normal null is invalid, which makes a naive test conservative (Snijders, 2001; Sinharay, 2016).

Instead of applying the analytic lz* standardization – derived by Snijders (2001) for dichotomous items and extended to polytomous / mixed-format items by Sinharay (2016, 2026) – easyRasch2 obtains the reference distribution by resampling with the person location **re-estimated for every simulated pattern** (see **Resampling**). This reproduces the ability-estimation effect and is the resampling analogue of lz*, so the resampled p-value is well calibrated.

Resampling. Each statistic is referenced against the null that matches it, removing the need to choose a scheme. The conditional MSQ statistics use patterns sampled **conditional on the person's total score** – the exact Rasch-native null, requiring no person estimate and fully consistent with the (conditional) statistic. lz, which is defined at the estimated location, uses patterns **simulated at that location and then scored with the location re-estimated from each simulated pattern**, so the null carries the same ability-estimation effect as the observed lz (for a Rasch / PCM model the location is a function of the total score, so this re-estimation is a cheap score-based lookup). Both schemes are computed per person over the items actually answered, so partial missingness is handled by either. The p-value is the proportion of the `iterations` replicates at least as extreme as the observed value. This follows the resampling-based person-fit approach (Sinharay, 2016) and the bootstrap recommendation of Müller (2020).

Value

For `output = "dataframe"`, a data.frame with one row per respondent (input order): `id`, `n_answered`, `sum_score`, the requested statistics (`infit_msq`, `outfit_msq`, `lz`), their resampled p-values (`p_infit`, `p_outfit`, `p_lz`) when `iterations > 0`, `flagged`, and – if `zstd = TRUE` – `infit_zstd`, `outfit_zstd`. The `flagged` column is `TRUE` when the marginal (uncorrected) p-value of **any** requested statistic is below `flag_alpha` for that respondent; it is therefore a per-person screening flag, not corrected for the number of respondents tested (so under fit expect about `flag_alpha` of respondents to be flagged by chance). Each `output = "ggplot"` map instead colours and counts respondents flagged by its **own** statistic alone. Extreme scorers (minimum or maximum possible given the items they answered) receive `NA` statistics and are not assessed. For `output = "kable"` the same content as a `knitr_kable`; for `output = "ggplot"` the named list of maps described under that argument; for `output = "list"`, a list with both views from a single computation: `fit` (the data.frame) and `plots` (the named list of maps).

References

de la Torre, J., & Deng, W. (2008). Improving person-fit assessment by correcting the ability estimate and its reference distribution. *Journal of Educational Measurement, 45*(2), 159-177. doi:10.1111/j.17453984.2008.00058.x

Drasgow, F., Levine, M. V., & Williams, E. A. (1985). Appropriateness measurement with polychotomous item response models and standardized indices. *British Journal of Mathematical and Statistical Psychology, 38*(1), 67-86. doi:10.1111/j.20448317.1985.tb00817.x

Kreiner, S., & Christensen, K. B. (2011). Exact evaluation of bias in Rasch model residuals. *Advances in Mathematics Research, 12*, 19-40.

Müller, M. (2020). Item fit statistics for Rasch analysis: can we trust them? *Journal of Statistical Distributions and Applications, 7*(5). doi:10.1186/s40488020001087

Sinharay, S. (2016). Assessment of person fit using resampling-based approaches. *Journal of Educational Measurement, 53*(1), 63-85. doi:10.1111/jedm.12101

Sinharay, S. (2016). Asymptotically correct standardization of person-fit statistics beyond dichotomous items. *Psychometrika, 81*(4), 992-1013. doi:10.1007/s113360159465x

Sinharay, S. (2026). Refining the asymptotically correct standardization of person-fit statistics for mixed-format tests. *British Journal of Mathematical and Statistical Psychology*. doi:10.1111/bmsp.70049

Snijders, T. A. B. (2001). Asymptotic null distribution of person fit statistics with estimated person parameter. *Psychometrika, 66*(3), 331-342. doi:10.1007/BF02294440

See Also

RMpersonParameters(), RMitemInfit(), RMitemParameters()

Examples

```
set.seed(1)
dat <- as.data.frame(
  matrix(sample(0:2, 200 * 8, replace = TRUE), nrow = 200, ncol = 8)
)
colnames(dat) <- paste0("Item", 1:8)

# Conditional infit/outfit MSQ + lz with resampled p-values
RMpersonFit(dat, iterations = 200, output = "dataframe") |> head()

# Person-fit maps: a named list with one plot per statistic
if (requireNamespace("ggplot2", quietly = TRUE)) {
  plots <- RMpersonFit(dat, iterations = 200, output = "ggplot")
  plots$infit    # infit map (each plot's caption reports the % flagged)
}

# Flag only underfit (noisy responding), the validity-relevant direction
RMpersonFit(dat, iterations = 200, flag = "underfit",
            output = "dataframe") |> head()
```

RMpersonParameters *Person Locations for a Rasch / Partial Credit Model*

Description

Estimates a person location (theta) and its standard error of measurement (SEM) for every respondent. Estimation is performed directly on each person's observed response pattern, so partial missingness is handled correctly: two respondents with the same sum score on different subsets of items receive different estimates. This avoids the sum-score lookup used by RMscoreSE(), which assumes complete data.

Usage

```
RMpersonParameters(
  data,
  method = c("WLE", "EAP"),
  item_params = NULL,
  estimator = c("CML", "MML"),
  theta_range = c(-10, 10),
  prior_mean = 0,
  prior_sd = NULL,
  output = c("kable", "dataframe", "ggplot", "file"),
  filename = NULL
)
```

Arguments

data	A data.frame or matrix of item responses. Items must be scored starting at 0 (non-negative integers). Missing values (NA) are allowed.
method	Character. "WLE" (default) for Warm's Weighted Likelihood Estimate (lower bias than MLE; Warm, 1989), or "EAP" for the Expected A Posteriori estimate under a normal prior.
item_params	Optional pre-specified item parameters, e.g. for anchoring or equating. Either a named list of Andrich-threshold vectors (one per item) or the long-format data.frame returned by RMitemParameters(). When NULL (default) item parameters are estimated from data.
estimator	Character. How item parameters are estimated when item_params is NULL: "CML" (default, via **psychotools**) or "MML" (via **mirt**). Ignored when item_params is supplied.
theta_range	Numeric length 2. Search range for the WLE root and bounds for the EAP quadrature grid. Default c(-10, 10). Only if the WLE root falls outside this range is the boundary returned.
prior_mean	Numeric. Mean of the normal prior used by method = "EAP". Default 0, the natural choice when item parameters are centred at mean difficulty zero. Ignored for WLE.
prior_sd	Numeric or NULL. Standard deviation of the normal prior used by method = "EAP". When NULL (default) it is estimated from the data by marginal maximum likelihood, holding the item parameters fixed. Supply a number (e.g. 1 for a standard N(0, 1) prior, or a value carried over from a reference sample for equating) to fix it. Ignored for WLE.
output	Character. "kable" (default) for a formatted knitr::kable() table, "dataframe" for the underlying data.frame, "ggplot" for a histogram of the estimated person locations, or "file" to write the data.frame to a CSV at filename (the data.frame is also returned invisibly).
filename	Character. Path to the CSV file to write when output = "file". Required in that case; ignored otherwise.

Details

Item parameters are obtained once (by CML or MML, or taken from `item_params`) and treated as fixed. Each person's theta is then estimated from the items they answered.

WLE solves Warm's weighted-likelihood score equation `l'(theta) + J(theta) / (2 I(theta)) = 0` by bracketed root finding, where the bias-correction term `J / (2 I)` keeps the equation solvable at the boundaries. The SEM is `1 / sqrt(I(theta))`. Unlike the MLE, Warm's estimator therefore yields *finite* locations for the minimum and maximum possible scores (a sensible step beyond the next-most extreme score rather than +/-Inf; cf. Warm, 1989), matching established implementations such as **catR** and **TAM**. Such scores are still marked in the `extreme` column because they are extrapolated and carry large standard errors. Only if the root lies outside `theta_range` is the boundary returned with `NA` SEM.

EAP integrates the pattern likelihood against a normal prior over a quadrature grid; the estimate is the posterior mean and the SEM is the posterior standard deviation. Unlike WLE, EAP is finite at extreme scores because the prior shrinks them inward, at the cost of depending on the assumed prior. The prior actually used – including the marginal-ML estimate of `prior_sd` when it is left `NULL` – is reported in the kable caption and attached to the result as `attr(result, "prior")`.

Value

For `output = "dataframe"`, a data.frame with one row per respondent (in input order; respondents with no responses are dropped with a message) and columns `theta` (the person location), `sem` (standard error of measurement), `sum_score`, `n_answered` (number of non-missing responses), and `extreme` (logical: a minimum or maximum possible score given the items answered). For `output = "kable"`, the same content as a `knitr_kable` object. For `output = "ggplot"`, a histogram of `theta`.

References

Warm, T. A. (1989). Weighted likelihood estimation of ability in item response theory. *Psychometrika, 54*(3), 427-450. doi:10.1007/BF02294627

Bock, R. D., & Mislevy, R. J. (1982). Adaptive EAP estimation of ability in a microcomputer environment. *Applied Psychological Measurement, 6*(4), 431-444. doi:10.1177/014662168200600405

Magis, D. (2015). A note on weighted likelihood and Jeffreys modal estimation of proficiency levels in polytomous item response models. *Psychometrika, 80*(1), 200-204. doi:10.1007/s11336-01393785

See Also

`RMitemParameters()`, `RMscoreSE()`, `RMreliability()`

Examples

```
set.seed(1)
dat <- as.data.frame(
  matrix(sample(0:2, 200 * 6, replace = TRUE), nrow = 200, ncol = 6)
)
colnames(dat) <- paste0("Item", 1:6)
# Introduce some missingness
```

```
dat[cbind(sample(200, 30), sample(6, 30, replace = TRUE))] <- NA

# Default: WLE person locations
RMpersonParameters(dat, output = "dataframe") |> head()

# EAP with a data-estimated prior SD
eap <- RMpersonParameters(dat, method = "EAP", output = "dataframe")
attr(eap, "prior")

# EAP with a fixed N(0, 1) prior
RMpersonParameters(dat, method = "EAP", prior_sd = 1, output = "dataframe") |>
  head()

# Write the person-location table to a CSV (also returned invisibly)
RMpersonParameters(dat, output = "file",
                   filename = tempfile(fileext = ".csv"))
```

RMplotBar *Item Response Distribution Bar Chart*

Description

Creates a faceted bar chart showing the response distribution for each item, with counts and percentages displayed on each bar. Each item gets its own panel, with response categories on the x-axis and percentage of responses on the y-axis. This is a descriptive data visualization tool intended for use before model fitting.

Usage

```
RMplotBar(
  data,
  item_labels = NULL,
  category_labels = NULL,
  ncol = 1L,
  label_wrap = 25L,
  text_y = 6,
  viridis_option = "A",
  viridis_end = 0.9,
  font = "sans"
)
```

Arguments

data	A data.frame in wide format containing only the item response columns. Each column is one item, each row is one person. All columns must be numeric (integer-valued). Response categories may be coded starting from 0 or 1. Do not include person IDs, grouping variables, or other non-item columns.

item_labels	An optional character vector of descriptive labels for the items (facet strips). Must be the same length as ncol(data). If NULL (the default), column names are used. Labels are displayed as "column_name - label".
category_labels	An optional character vector of labels for the response categories (x-axis). Must be the same length as the number of response categories spanning from the minimum to the maximum observed value. If NULL (the default), numeric category values are used.
ncol	Integer. Number of columns in the faceted layout. Default is 1L.
label_wrap	Integer. Number of characters per line in facet strip labels before wrapping. Default is 25L.
text_y	Numeric. Vertical position (in percent units) for the count labels on each bar. Adjust upward if bars are tall. Default is 6.
viridis_option	Character. Viridis palette option for the count-text colour. One of "A" through "H". Default is "A".
viridis_end	Numeric in $[0, 1]$. End point of the viridis colour scale for count text. Adjust if text is hard to read against the bar colours. Default is 0.9.
font	Character. Font family for all text. Default is "sans".

Details

Each item is displayed as a separate facet panel with the item label in the strip on the left side. Bars are coloured by response category using the viridis palette. Each bar shows the count (n = X) as text.

The plot caption reports the sample in the standard n = X respondents (policy) form; item-level NAs are retained – each bar counts the non-missing responses for that item.

Input requirements:

- All columns must be numeric (integer-valued).
- The data frame must contain at least 2 columns (items) and at least 1 row (person).

Value

A ggplot2::ggplot object.

See Also

RMplotStackedbar(), RMplotTile()

Examples

```
if (requireNamespace("eRm", quietly = TRUE) &&
    requireNamespace("ggplot2", quietly = TRUE)) {
  data(pcmdat2, package = "eRm")

  # Basic response distribution plot
  RMplotBar(pcmdat2)
```

```
  # With custom item labels
  RMplotBar(
    pcmdat2,
    item_labels = c("Mood", "Sleep", "Appetite", "Energy")
  )

  # Two-column layout with wrapped labels
  RMplotBar(
    pcmdat2,
    item_labels = c(
      "General mood and emotional wellbeing",
      "Quality of sleep at night",
      "Appetite and eating habits",
      "Overall energy level during the day"
    ),
    ncol = 2, label_wrap = 20
  )

  # With custom category labels
  RMplotBar(
    pcmdat2,
    category_labels = c("Never", "Sometimes", "Often")
  )
}
```

RMplotStackedbar *Stacked Bar Chart of Item Response Distributions*

Description

Creates a horizontal stacked bar chart showing the response distribution for all items. Each bar represents one item, with segments coloured by response category. Counts are displayed as text labels within each segment. This is a descriptive data visualization tool intended for use before model fitting.

Usage

```
RMplotStackedbar(
  data,
  item_labels = NULL,
  category_labels = NULL,
  show_n = TRUE,
  show_percent = FALSE,
  text_color = "sienna1",
  text_size = 3,
  min_label_n = 0L,
  viridis_option = "D",
  viridis_end = 0.99,
```

```
  title = "Item responses"
)
```

Arguments

data	A data.frame in wide format containing only the item response columns. Each column is one item, each row is one person. All columns must be numeric (integer-valued). Response categories may be coded starting from 0 or 1. Do not include person IDs, grouping variables, or other non-item columns.
item_labels	An optional character vector of descriptive labels for the items (y-axis). Must be the same length as `ncol(data)`. If `NULL` (the default), column names are used.
category_labels	An optional character vector of labels for the response categories (legend). Must be the same length as the number of response categories spanning from the minimum to the maximum observed value, ordered from lowest to highest category. If `NULL` (the default), numeric category values are used.
show_n	Logical. If `TRUE` (the default), the count of responses is displayed as a text label inside each bar segment.
show_percent	Logical. If `TRUE`, the percentage of responses is displayed instead of (or in addition to) counts. Default is `FALSE`.
text_color	Character. Colour for the count/percentage labels. Default is `"sienna1"`.
text_size	Numeric. Size of the count/percentage labels. Default is `3`.
min_label_n	Integer. Minimum count required for a label to be displayed within a bar segment. Segments with fewer responses are left unlabelled to avoid clutter. Default is `0L` (all segments labelled).
viridis_option	Character. Viridis palette option. One of `"A"` through `"H"`. Default is `"D"`.
viridis_end	Numeric in $[0, 1]$. End point of the viridis colour scale. Default is `0.99`.
title	Character. Plot title. Default is `"Item responses"`.

Details

Items are displayed on the y-axis in the same order as the columns in `data` (first column at the top). Each bar is divided into segments representing response categories, with the lowest category on the left and the highest on the right. The total bar length equals the number of non-missing responses for that item.

Categories with zero responses still appear in the legend but produce no visible bar segment, which helps identify gaps in the response distribution.

The plot caption reports the sample in the standard `n = X respondents (policy)` form; item-level `NA`s are retained – each bar counts the non-missing responses for that item.

Input requirements:

- All columns must be numeric (integer-valued).
- The data frame must contain at least 2 columns (items) and at least 1 row (person).

Value

A ggplot2::ggplot object.

See Also

RMplotBar(), RMplotTile()

Examples

```
if (requireNamespace("eRm", quietly = TRUE) &&
    requireNamespace("ggplot2", quietly = TRUE)) {
  data(pcmdat2, package = "eRm")

  # Basic stacked bar chart
  RMplotStackedbar(pcmdat2)

  # With custom item and category labels
  RMplotStackedbar(
    pcmdat2,
    item_labels     = c("Mood", "Sleep", "Appetite", "Energy"),
    category_labels = c("Never", "Sometimes", "Often")
  )

  # Show percentages, suppress small segments
  RMplotStackedbar(
    pcmdat2,
    show_percent = TRUE,
    show_n       = FALSE,
    min_label_n  = 5
  )
}
```

RMplotTile *Tile Plot of Item Response Distributions*

Description

Creates a tile (heat map) plot showing the distribution of responses across all items and response categories. Each cell displays the count (or percentage) of responses, with optional conditional highlighting for cells with low counts. Optional faceting by a grouping variable is provided for inspecting subgroup response distributions before DIF analyses – particularly useful for spotting empty categories or under-represented subgroups before fitting Rasch models per group.

Usage

```
RMplotTile(
  data,
  group = NULL,
```

```
  cutoff = 10,
  highlight = TRUE,
  percent = FALSE,
  text_color = "orange",
  text_size = 4,
  zero_fill = "white",
  item_labels = NULL,
  category_labels = NULL,
  group_labels = NULL,
  facet_ncol = NULL,
  output = c("ggplot", "dataframe")
)
```

Arguments

data	A data.frame in wide format containing only the item response columns. Each column is one item, each row is one person. All columns must be numeric (integer-valued). Response categories may be coded starting from 0 or 1. Do not include person IDs, grouping variables, or other non-item columns – supply the grouping variable separately via group.
group	Optional vector of length nrow(data) (factor, character, or numeric) defining a grouping variable. When provided, the plot is faceted by group and counts / percentages are computed within each group. Default NULL (no faceting). Persons with NA group are excluded, with a message() reporting how many rows were dropped.
cutoff	Integer. Cells with counts below this value are highlighted (when highlight = TRUE). Default 10.
highlight	Logical. If TRUE (default), cell labels with counts below cutoff are displayed in red. This includes empty cells (n = 0), useful for identifying gaps in the response distribution.
percent	Logical. If TRUE, cell labels show percentages instead of raw counts. Percentages are computed within item (and within group, when group is supplied). Default FALSE.
text_color	Character. Colour for non-highlighted cell labels. Default "orange".
text_size	Numeric. Size of the cell labels (passed to ggplot2::geom_text()). Default 4, matching RMitemCatProb().
zero_fill	Colour for cells with zero responses, or NULL to keep them on the fill scale. Default "white": empty cells are shown unfilled with a light grey outline, so the *absence* of data reads as absence rather than as the darkest end of the colour scale (where n = 0 is nearly indistinguishable from n = 1).
item_labels	Optional character vector of descriptive labels for the items (y-axis), same length as ncol(data). Default NULL uses the column names.
category_labels	Optional character vector of labels for the response categories (x-axis), same length as the number of categories spanning min to max observed value. Default NULL uses the numeric category values.

group_labels	Optional character vector of length nlevels(as.factor(group)) to override the displayed facet labels. Order corresponds to levels(as.factor(group)).
facet_ncol	Integer or NULL. Number of columns in the facet grid when group is supplied. Default NULL (ggplot2 chooses).
output	Character. "ggplot" (default) returns the plot; "dataframe" returns the underlying per-cell counts (one row per item x category, plus group when supplied).

Details

Adapted from easyRaschBayes::plot_tile() and extended with the group parameter for faceted display.

Items are placed on the y-axis (in the same order as the columns of data, top to bottom) and response categories on the x-axis. Cell shading represents the count of responses (darker = more responses). Categories with zero responses are explicitly shown (n = 0), which helps identify gaps in the response distribution – one of the primary purposes of the plot, especially before DIF analyses where under-represented categories within a subgroup can break model fitting on that subgroup.

When group is supplied, percentages and the highlight cutoff are applied within each group, so a cell labelled "5" in the group-A facet contains the count for group A only.

The plot caption reports the sample in the standard n = X of Y respondents (policy) form: rows with NA group are dropped (and counted in Y only), while item-level NAs are retained – each cell simply counts the non-missing responses for that item.

Value

Either a ggplot object or a data.frame, depending on output.

Examples

```
if (requireNamespace("eRm", quietly = TRUE)) {
  data("pcmdat2", package = "eRm")

  # Basic tile plot
  RMplotTile(pcmdat2)

  # With percentages
  RMplotTile(pcmdat2, percent = TRUE)

  # Faceted by an external grouping variable
  set.seed(1)
  grp <- sample(c("A", "B"), nrow(pcmdat2), replace = TRUE)
  RMplotTile(pcmdat2, group = grp)

  # With custom labels and tighter cutoff
  RMplotTile(pcmdat2,
             group = grp,
             group_labels = c("Female", "Male"),
             cutoff = 5,
             facet_ncol = 2)
```

```
  # Underlying counts as a data.frame
  RMplotTile(pcmdat2, group = grp, output = "dataframe")
}
```

`RMreliability` *Reliability metrics for a Rasch model*

Description

Computes three reliability indices for a Rasch / partial credit model: the Person Separation Index (PSI) – WLE-based separation reliability – the marginal reliability (native, CML test information integrated over the estimated normal latent density), and Relative Measurement Uncertainty (RMU) via `RMUreliability()` applied to plausible values from `mirt::fscores()`.

Usage

```
RMreliability(
  data,
  conf_int = 0.95,
  draws = 1000,
  rmu_iter = 50,
  estim = "WLE",
  boot = FALSE,
  boot_iter = 200,
  parallel = TRUE,
  n_cores = NULL,
  seed = NULL,
  verbose = FALSE,
  theta_range = c(-10, 10),
  output = "kable"
)
```

Arguments

`data`	A data.frame or matrix of item responses. Items must be scored starting at 0 (non-negative integers).
`conf_int`	Numeric in (0, 1). HDCI width for both bootstrap CIs and RMU. Default `0.95`.
`draws`	Integer. Number of plausible-value draws drawn from the mirt model for the RMU calculation. Default `1000`. More gives a more stable RMU; computational cost is mostly linear.
`rmu_iter`	Integer. Number of times `RMUreliability()` is repeated on the same set of plausible-value draws (each repetition uses a fresh random column split). Estimates are averaged across repetitions to stabilise against split-induced variability. Default `50`.

estim	Character. Theta estimator used by mirt::fscores() for the RMU plausible-value seed. One of "WLE" (default), "EAP", "MAP", "ML". Plausible draws themselves are produced by Metropolis-Hastings. (PSI and marginal reliability are computed natively and do not use this.)
boot	Logical. If TRUE, run a non-parametric bootstrap to obtain CIs for PSI and Marginal reliability. Default FALSE.
boot_iter	Integer. Number of bootstrap iterations when boot = TRUE. Default 200.
parallel	Logical. Use parallel processing via mirai for the bootstrap if available. Default TRUE.
n_cores	Integer or NULL. Number of parallel workers. When NULL, getOption("mc.cores") is checked first; if neither is set, bootstrapping falls back to sequential.
seed	Integer or NULL. Master random seed for reproducibility. See easyRasch2-reproducibility for what this guarantees and how it interacts with parallel.
verbose	Logical. Print progress messages and a progress bar for the bootstrap. Default FALSE.
theta_range	Numeric length-2 vector. Theta limits passed to mirt::fscores(). Default c(-10, 10).
output	Character. "kable" (default) for a formatted knitr::kable() table, or "dataframe" for the underlying data.frame.

Details

Confidence intervals for **PSI** and **Marginal** reliability are obtained by non-parametric bootstrap (resampling respondents; all three indices are recomputed natively per resample, no model is refitted by mirt). The RMU interval is the HDCI of correlations across plausible-value draws, averaged over rmu_iter random splits of the draws.

Marginal reliability is the latent-density-weighted mean of the conditional reliability curve, $\int \sigma^2/(\sigma^2 + 1/I(\theta))\, g(\theta)\, d\theta$, where the test information $I(\theta)$ is summed from the CML item parameters and g is the estimated normal latent density $N(\mu, \sigma^2)$, **both** parameters coming from marginal ML with the items held fixed. Integrating over the estimated latent distribution, rather than the $N(0, 1)$ assumed by mirt::marginal_rxx(), keeps it correct on the Rasch logit scale, where σ is typically well above 1 and the $N(0, 1)$ assumption underestimates reliability.

Through version 1.3.0 μ was held at 0 rather than estimated. That is an assumption, not a consequence of centring the item thresholds, which fixes the *item* mean and says nothing about where the respondents sit. A normal density pinned to 0 can only reach an off-target sample by widening, so σ absorbed the mistargeting and marginal reliability **rose** as targeting worsened. See the note under Value in RMreliabilityCurve() for the size of the effect.

It is the model-based complement to the sample-based PSI, and the two are now the same coefficient by two routes: PSI divides by the observed spread of the WLE estimates, marginal reliability by the fitted latent density. A large gap between them therefore does flag an off-target or non-normal sample, and since 1.3.1 both move in the same direction when targeting worsens rather than apart.

How this relates to the published coefficients. Three things are worth being explicit about, because the reported value is not any of the coefficients the literature names.

It is the mean of a curve, not a ratio of averages. The classical latent-scale reliability is a single variance ratio, $\sigma^2/(\sigma^2 + \overline{SEM^2})$ (Milanzi et al., 2015, section 3.2, where the error variance is a

constant). What is reported here averages the ratio over the latent density instead. Since $\sigma^2/(\sigma^2+x)$ is convex in x, the mean of the curve is the **larger** of the two by Jensen's inequality. The choice was made on accuracy, not convention: averaging the curve tracked the exact expected-sum-score reliability more closely than the ratio of averages in both arms of an internal replication (mean absolute error 0.015 against 0.018 for binary data, 0.010 against 0.015 for polytomous).

It is a latent-scale quantity checked against a manifest criterion. Milanzi et al. (2015) argue that interest usually lies in the reliability of observed scores rather than latent ones, and that latent coefficients run consistently higher. This one was therefore validated against the exact expected-sum-score reliability, $Var(\mu)/(Var(\mu)+Var(\epsilon))$, rather than assumed to match it, and tracked it to within about 0.02 across the conditions tested. It remains a latent-scale coefficient that approximates a manifest one, not a manifest coefficient.

Milanzi et al. recommend something this package does not implement. Their conclusion favours Taylor-series manifest reliability measures, which approximate $Var(\mu)$ and $Var(\epsilon)$ on the observed-score scale directly rather than working from test information. In the same internal replication those were the most accurate of the estimators compared (0.012). They are cited here for their criticism of the subtractive coefficient, which this package acted on, and not for their remedy, which it has not adopted.

The superseded subtractive coefficient is Milanzi et al.'s equation (12), attributed there to Lord (1980), and is still available as the `marginal_green` attribute of `RMreliabilityCurve()`. Through version 1.2.0 this row used Green's subtractive $1 - \overline{1/I(\theta)}/\sigma^2$ instead, under which much of the PSI-to-marginal gap was an artefact of the differing formulas rather than a property of the sample. See `dev/TODO-reliability-form.md`.

PSI is the WLE-based separation reliability, $1 - \overline{SEM^2}/\mathrm{Var}(\hat{\theta})$, computed from CML item thresholds (`psychotools`) and Warm's WLE person locations / analytic SEMs. Respondents with extreme (min/max) raw scores are excluded – their boundary estimates would inflate the person variance and overstate reliability. (Earlier versions used `eRm::SepRel()` with MLE; the values can differ, most noticeably for scales with many extreme scorers, e.g. dichotomous items.)

RMU is from Bignardi, Kievit, & Bürkner (2025), modified here to use mirt plausible values rather than fully Bayesian posterior draws (see Mislevy, 1991, for the plausible-values framework).

Marginal reliability here is the subtractive Green/Lord coefficient. Milanzi et al. (2015) show that this form can fall below zero when the average error variance exceeds the trait variance, which happens with few items or a narrow sample, and it is floored at 0 above. `RMreliabilityCurve()` reports the same quantity in the bounded ratio form alongside this one, so the two can be compared directly. See `dev/TODO-reliability-form.md` for the open question of which form this row should use.

Bootstrap iterations that fail to converge are silently dropped.

Value

- If `output = "kable"`: a `knitr_kable` object with one row per metric.
- If `output = "dataframe"`: a data.frame with columns `metric`, `estimate`, `lower`, `upper`, `notes`.

References

Bignardi, G., Kievit, R., & Bürkner, P. C. (2025). A general method for estimating reliability using Bayesian Measurement Uncertainty. *PsyArXiv*. doi:10.31234/osf.io/h54k8_v1

Green, B. F., Bock, R. D., Humphreys, L. G., Linn, R. L., & Reckase, M. D. (1984). Technical Guidelines for Assessing Computerized Adaptive Tests. *Journal of Educational Measurement, 21*(4), 347–360. doi:10.1111/j.17453984.1984.tb01039.x

Mislevy, R. J. (1991). Randomization-Based Inference about Latent Variables from Complex Samples. *Psychometrika, 56*(2), 177-196. doi:10.1007/BF02294457

Adams, R. J. (2005). Reliability as a measurement design effect. *Studies in Educational Evaluation, 31*(2), 162-172. doi:10.1016/j.stueduc.2005.05.008

Milanzi, E., Molenberghs, G., Alonso, A., Verbeke, G., & De Boeck, P. (2015). Reliability measures in item response theory: Manifest versus latent correlation functions. *British Journal of Mathematical and Statistical Psychology, 68*(1), 43-64. doi:10.1111/bmsp.12033

See Also

`RMUreliability()`, `RMreliabilityCurve()`

Examples

```
if (requireNamespace("ggdist", quietly = TRUE) &&
    requireNamespace("eRm", quietly = TRUE)) {
  set.seed(1)
  RMreliability(eRm::raschdat1[, 1:20], draws = 1000)

  # Bootstrap CI for PSI and Marginal
  # (use more bootstrap iterations, e.g. 200+, in real analyses)
  RMreliability(eRm::raschdat1[, 1:20], draws = 1000,
                boot = TRUE, boot_iter = 25, parallel = FALSE, seed = 42)
}
```

RMreliabilityCurve *Conditional measurement precision across the latent scale*

Description

Plots (or tabulates) how precisely a scale measures at each point of the latent continuum, rather than collapsing precision to the single number reported by `RMreliability()`. The curve is the conditional standard error of measurement $SEM(\theta) = 1/\sqrt{I(\theta)}$, the test information $I(\theta)$ it comes from, or the conditional reliability derived from either.

Usage

```
RMreliabilityCurve(
  data,
  statistic = "sem",
  method = "WLE",
  benchmark = NULL,
  reference = "marginal",
```

```
  items = NULL,
  item_params = NULL,
  boot = FALSE,
  boot_iter = 200,
  conf_int = 0.95,
  parallel = TRUE,
  n_cores = NULL,
  seed = NULL,
  show_density = TRUE,
  theta_range = NULL,
  n_nodes = 161L,
  verbose = FALSE,
  output = "ggplot"
)
```

Arguments

data	A data.frame or matrix of item responses. Items must be scored starting at 0 (non-negative integers). Missing values (NA) are allowed.
statistic	Character. The quantity on the y-axis: "sem" (default, the conditional standard error in logits), "information" (test information), or "reliability" (see Details for the formula used, and why). This is the argument that chooses what the figure shows, not method.
method	Character. Person-location estimator, "WLE" (default) or "EAP". **This does not change the curve**, which is always built from test information. It governs only the respondent locations behind the density overlay and the benchmark percentage, and is ignored when show_density = FALSE and benchmark = NULL.
benchmark	Numeric in (0, 1) or NULL (default). When supplied, the region of the scale whose conditional reliability reaches benchmark is shaded, and the percentage of respondents located inside it is reported. There is deliberately no default value; see Details.
reference	Character. Horizontal line showing the flat summary that a single coefficient implies: "marginal" (default) or "none".
items	Optional character vector of column names, or numeric column indices, selecting a subset of items. Useful for comparing a short form against the full scale.
item_params	Optional pre-specified item parameters, used in place of the thresholds estimated from data. Either a named list of Andrich threshold vectors or a long-format data.frame from RMitemParameters(), the same two forms RMpersonParameters() accepts. Supply this to draw the curve from anchored or previously published parameters. As in RMpersonParameters(), supplied thresholds are used as given and are **not** re-centred, so their origin defines the origin of the theta axis.
boot	Logical. If TRUE, add a bootstrap confidence band to the curve by resampling respondents. Default FALSE.
boot_iter	Integer. Bootstrap iterations when boot = TRUE. Default 200.
conf_int	Numeric in (0, 1). HDCI width for the bootstrap band. Default 0.95.
parallel	Logical. Use mirai for the bootstrap if available. Default TRUE.

n_cores	Integer or NULL. Number of parallel workers. When NULL, getOption("mc.cores") is checked first; if neither is set, the bootstrap falls back to sequential.
seed	Integer or NULL. Random seed for the bootstrap. See easyRasch2-reproducibility.
show_density	Logical. Draw the respondent location distribution as a background band. Default TRUE.
theta_range	Numeric length-2 vector, or NULL (default) for $\pm 3\sigma$.
n_nodes	Integer. Number of points on the theta grid. Default 161.
verbose	Logical. Print a progress bar for the bootstrap. Default FALSE.
output	Character. "ggplot" (default), "dataframe" for the curve, or "kable" for a summary table.

Details

What the curve is. Test information $I(\theta) = \sum_i \mathrm{Var}_i(\mathrm{score} \mid \theta)$ is summed from the CML item thresholds (psychotools), and the conditional standard error follows as $1/\sqrt{I(\theta)}$. Both are properties of the item set alone. The respondent distribution is drawn behind the curve because precision only matters where people actually are, but it does not enter the curve.

Conditional reliability, and which formula. Two definitions circulate:

$$\rho(\theta) = \frac{\sigma^2}{\sigma^2 + SEM(\theta)^2} \qquad \text{and} \qquad \rho(\theta) = 1 - \frac{SEM(\theta)^2}{\sigma^2}$$

with σ the latent SD estimated by marginal maximum likelihood, jointly with the latent mean μ (see the latent_mean attribute). This function uses the first, the ratio form, for three reasons. It is bounded in (0, 1), whereas the subtractive form returns negative values whenever $SEM(\theta) > \sigma$, which is common at the floor of a skewed scale and at both tails of a short one. It treats σ^2 as true-score variance, which is what the MML estimate of the latent SD is. And it tracks the reliability of the observed scores far more closely. Milanzi et al. (2015) report the subtractive coefficient falling to -0.120 against a true value of 0.480 for a 1PL with $\sigma^2 = 0.25$. Replicating their design (dev/milanzi_check.R, 1000 replications per cell) gives a mean absolute error against the exact expected-sum-score reliability of 0.22 for the subtractive form and 0.018 for the ratio form, with the subtractive form negative in 99.7% of replications of their lowest-variance cell. For polytomous Rasch data at 5 to 20 items the corresponding errors are 0.18 and 0.015.

The advantage is concentrated where it matters rather than uniform. Where information is high relative to the trait variance the two forms are close and either can be nearer the truth. The gap comes from the low-information cases, where the subtractive form does not merely lose accuracy but leaves the (0, 1) interval altogether.

marginal_ratio is the latent-density-weighted mean of the reliability curve rather than the ratio formed from the averaged error variance. The two differ by Jensen's inequality, and the curve mean is the more accurate of the pair (mean absolute error 0.015 against 0.018 for binary data, 0.010 against 0.015 for polytomous).

RMreliability() reports the same ratio-form coefficient in its "Marginal (curve mean)" row, so the scalar and the curve agree. The superseded subtractive value is still returned as marginal_green for comparison with easyRasch2 1.2.0 and earlier, and with mirt::marginal_rxx() and similar software.

Limits. The information-based standard error is asymptotic, and is only an approximation for short scales; Milanzi et al. (2015) find the worst behaviour with six items. The curve describes the precision of the person location estimate on the logit scale, which is not the same quantity as the reliability of a raw sum score, and reliability computed on a latent scale is consistently higher than its manifest counterpart. Use `RMscoreSE()` for the raw-score view.

No default benchmark. `benchmark` is `NULL` unless asked for, in keeping with the package's treatment of fixed rules of thumb. McNeish and Dumas (2025), whose respondent-weighted summary this borrows, are explicit that their own interpretive percentages are heuristic and should not be used as cutoffs.

Value

- If `output = "ggplot"`: a `ggplot` object.
- If `output = "dataframe"`: a data.frame with one row per grid point and columns `theta`, `information`, `sem`, `reliability`, plus `<statistic>_lower` / `<statistic>_upper` band columns when `boot = TRUE`. Attributes carry the summary quantities: `sigma`, `marginal_ratio` (the latent-density-weighted mean of the reliability curve, matching `RMreliability()`), `marginal_green` (the superseded subtractive coefficient), `sem_average` (root mean error variance), `latent_mean`, `benchmark`, `benchmark_range`, `benchmark_percent` and `n_not_estimable`.

  `latent_mean` and `sigma` are estimated together. Before 1.3.1 the mean was held at 0 and σ absorbed any distance between the sample and the item locations: on simulated data with a true SD of 0.90, shifting the sample 2 logits off target returned 2.02 rather than 0.96, and pushed marginal reliability from 0.81 up to 0.90 when it should have fallen to 0.73. Well-targeted samples are unaffected.
- If `output = "kable"`: a `knitr_kable` summary table of those quantities.

References

Green, B. F., Bock, R. D., Humphreys, L. G., Linn, R. L., & Reckase, M. D. (1984). Technical Guidelines for Assessing Computerized Adaptive Tests. *Journal of Educational Measurement, 21*(4), 347-360. doi:10.1111/j.17453984.1984.tb01039.x

McNeish, D., & Dumas, D. (2025). Reliability representativeness: How well does coefficient alpha summarize reliability across the score distribution? *Behavior Research Methods, 57*(3), 93. doi:10.3758/s13428025026118

Milanzi, E., Molenberghs, G., Alonso, A., Verbeke, G., & De Boeck, P. (2015). Reliability measures in item response theory: Manifest versus latent correlation functions. *British Journal of Mathematical and Statistical Psychology, 68*(1), 43-64. doi:10.1111/bmsp.12033

See Also

`RMreliability()`, `RMscoreSE()`, `RMtargeting()`

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  # Conditional SEM across the scale
  RMreliabilityCurve(phq9[, 1:9])
```

```
  # Conditional reliability, with the region reaching .8 shaded
  RMreliabilityCurve(phq9[, 1:9], statistic = "reliability",
                     benchmark = 0.8)
}

# Summary quantities
RMreliabilityCurve(phq9[, 1:9], benchmark = 0.8, output = "kable")
```

`RMretestSD` *Occasion-to-occasion SD from a test-retest study*

Description

Estimates the per-occasion SD, in logits, of a respondent's occasion-specific deviation: the variation between two administrations that is neither measurement error nor change in the construct. It is the `retest_sd` argument of `RMpersonChange()` when `null = "retest"`.

Usage

```
RMretestSD(
  data_t1,
  data_t2,
  anchor = "stack",
  item_params = NULL,
  method = "WLE",
  estimator = "CML",
  sim_iter = 500,
  parallel = TRUE,
  n_cores = NULL,
  verbose = FALSE,
  boot = TRUE,
  boot_iter = 500,
  conf_int = 0.95,
  seed = NULL,
  theta_range = c(-10, 10),
  output = "kable"
)
```

Arguments

`data_t1`, `data_t2`
: Data.frames or matrices of item responses from a test-retest study, same items in the same order, row `i` being the same respondent at both administrations.

`anchor`, `item_params`, `method`, `estimator`, `theta_range`
: As in `RMpersonChange()`. Use the same settings there as here, since the estimate is defined relative to the measurement-error variance the model implies.

`sim_iter`	Integer. Iterations used to simulate the measurement-error component. Default `500`.
`parallel`	Logical. Use `mirai` for the simulation if available. Default `TRUE`.
`n_cores`	Integer or `NULL`. Parallel workers. When `NULL`, `getOption("mc.cores")` is checked first.
`verbose`	Logical. Progress bar for the simulation. Default `FALSE`.
`boot`	Logical. Bootstrap a confidence interval by resampling respondents. Default `TRUE`.
`boot_iter`	Integer. Bootstrap iterations. Default `500`.
`conf_int`	Numeric in (0, 1). HDCI width. Default `0.95`.
`seed`	Integer or `NULL`. Random seed.
`output`	Character. `"kable"` (default) or `"dataframe"`.

Details

Writing $\hat{\theta}_{it} = \theta_i + u_{it} + e_{it}$ with $u_{it} \sim N(0, \sigma^2_{retest})$ the occasion deviation and e_{it} the measurement error, the observed change has variance $2\sigma^2_{retest} + SE^2_{i1} + SE^2_{i2}$. So

$$\hat{\sigma}^2_{retest} = \frac{\mathrm{Var}(d) - \mathrm{Var}_{err}(d)}{2}$$

How the error term is obtained. The obvious choice for $\mathrm{Var}_{err}(d)$ is $\overline{SE^2_1 + SE^2_2}$ from the information-based standard errors, and that is what Caronni et al. (2026) and the classical formulations use. It is badly biased on short scales, because the information standard error is asymptotic and overstates the real sampling variance of $\hat{\theta}$ when items are few, so the subtraction removes too much. Simulation over a range of scale lengths, at a true occasion variance of 0.09 and n = 2000:

items	asymptotic	simulated
6	-0.023	0.072
12	0.053	0.093
20	0.086	0.099

Across eight replications at n = 1000 the simulated version is close to unbiased on a six-item scale (mean 0.002 against a truth of 0, and 0.081 against a truth of 0.09), with a replication SD near 0.015.

This function therefore obtains $\mathrm{Var}_{err}(d)$ by simulation: respondents are held at their observed locations, two response vectors are drawn with no occasion variance, and the variance of the resulting change is averaged over `sim_iter` iterations. The observed missingness pattern is carried into the simulated responses. `var_error` in the output is that simulated quantity, not the mean of the squared standard errors.

The retest interval has to be long enough for recall to fade and short enough that no real change occurs. That is an assumption about the design and the data cannot check it.

The estimate can be negative. It is a variance obtained by subtraction, so when the observed change is tighter than the measurement model says it should be, the difference goes below zero. A negative value is reported as such rather than floored, because flooring hides the signal. It points at

standard errors overstated by misfit or local dependence, at regression toward a common value over the interval, or at sampling noise. `sd` is `NA` whenever `variance` is negative.

It is a variance estimated from `n` differences, so its own sampling error is roughly $\sigma^2\sqrt{2/(n-1)}$. A few dozen respondents will not pin it down, and the bootstrap interval is the honest way to see that. The bootstrap resamples respondents and recomputes $\mathrm{Var}(d)$ only, holding the simulated error term at its full-sample value, since that term is model-implied and its Monte Carlo error is small beside the sampling error of the change variance.

Converting a published test-retest coefficient. A published test-retest ICC or Pearson correlation can be converted, but it is not itself the quantity wanted here, and neither is a published test-retest SEM ($SD\sqrt{1-ICC}$), which already contains measurement error. Since $\hat{\theta} = \theta + u + e$ gives $r = \mathrm{Var}(\theta)/\mathrm{Var}(\hat{\theta})$,

$$\hat{\sigma}^2_{retest} = (1-r)\,\mathrm{Var}(\hat{\theta}) - \mathrm{Var}_{err}(d)/2$$

with $\mathrm{Var}(\hat{\theta})$ the observed variance of the person estimates and $\mathrm{Var}_{err}(d)$ the simulated error term this function reports as `var_error`. Checked against known truth at n = 3000, true occasion variance 0.09:

items	r (logit)	r (sum score)	from logit r	from sum-score r
6	.808	.829	0.094	0.046
12	.873	.890	0.084	0.048
20	.903	.913	0.088	0.067

Three conditions have to hold, and the first is usually violated:

- **The coefficient must be on the logit metric.** The score-to-logit map is nonlinear, so a coefficient computed on sum scores runs consistently higher than the same coefficient on logits. After subtraction the residual is small, so that modest gap does real damage: on a six-item scale, substituting the sum-score value roughly halves the recovered occasion variance. The error has a known direction, understating occasion noise and leaving `RMpersonChange()` too permissive. Nearly all published coefficients are on sum scores.
- **The published sample must resemble yours.** A retest coefficient rises with the heterogeneity of the people measured, so it is a property of that sample rather than of the instrument. Combining someone else's `r` with your $\mathrm{Var}(\hat{\theta})$ assumes the structure transfers.
- **Pearson and an agreement ICC are not interchangeable.** Pearson ignores a systematic mean shift between occasions and so reports stability that practice or drift has removed. ICC(A,1) penalises it and is the closer choice here.

The conversion also needs the simulated error term, which this function produces only when given two occasions of raw data. Converting from a published coefficient and a single administration is not implemented. When that is the situation you are in, `retest_sd_tip` from `RMpersonChange()` answers the practical question from the other end: it reports how much occasion noise each flagged result would tolerate, which you can weigh against a published coefficient without converting anything.

Value

A one-row data.frame (or its `knitr_kable`) with `variance`, `sd`, `lower`, `upper`, `var_change` (the observed variance of the change) and `var_error` (the simulated measurement-error variance of the change), plus `n`.

See Also

`RMpersonChange()`

Examples

```
set.seed(1)
thr <- lapply(seq(-1.2, 1.2, length.out = 6), function(b) b + c(-0.8, 0, 0.8))
theta <- rnorm(200, 0, 1.4)
# a true occasion SD of 0.3 logits
r1 <- as.data.frame(easyRasch2:::sim_partial_score(thr, theta + rnorm(200, 0, 0.3)))
r2 <- as.data.frame(easyRasch2:::sim_partial_score(thr, theta + rnorm(200, 0, 0.3)))
colnames(r1) <- colnames(r2) <- paste0("I", 1:6)
RMretestSD(r1, r2, sim_iter = 100, parallel = FALSE, boot = FALSE)
```

RMscoreSE *Raw-Score to Logit Score Transformation Table*

Description

For a given set of items, returns the score-to-theta lookup that maps each possible raw sum score to a person-location estimate (in logits) and its standard error. Useful when reporting a scale's measurement properties or converting raw totals to interval-scaled scores for downstream analysis.

Usage

```
RMscoreSE(
  data,
  method = "WLE",
  output = "kable",
  ci_multiplier = 1.96,
  point_size = 3,
  error_width = 0.5,
  theta_range = c(-10, 10)
)
```

Arguments

`data`	A data.frame or matrix of item responses. Items must be scored starting at 0 (non-negative integers). Missing values (`NA`) are allowed; the underlying model fit handles them.

method	Character string. Either "WLE" (default) for Warm's Weighted Likelihood Estimator computed from a CML-fitted Rasch / Partial Credit Model via psychotools, or "EAP" for Expected A Posteriori sum-score estimates from an MML-fitted model via mirt.
output	Character string controlling the return value: "kable" (default) for a formatted knitr::kable() table, "dataframe" for the underlying data.frame, or "ggplot" for a ggplot2 figure showing each raw score's logit estimate with ci_multiplier-scaled error bars.
ci_multiplier	Numeric. Multiplier applied to the standard error to draw error bars on the figure. Default 1.96 (≈95% CI under a Gaussian approximation). Ignored when output != "ggplot".
point_size	Numeric. Point size for the figure. Default 3.
error_width	Numeric. Cap width for error bars on the figure. Default 0.5.
theta_range	Numeric length 2. Theta search range used for boundary raw scores under WLE estimation. Default c(-10, 10). Ignored when method = "EAP".

Details

The function automatically detects whether the data is dichotomous (max score 1) or polytomous (max score > 1) and selects the appropriate Rasch / Partial Credit model.

method = "WLE" fits the model by CML with psychotools::pcmodel(), centres the item thresholds to grand-mean-zero, and solves Warm's weighted-likelihood equation for each raw score with the same engine used by RMpersonParameters(); the two functions therefore report identical locations and standard errors. Warm's bias correction yields finite locations even at the minimum and maximum scores (only a root outside theta_range is clamped to the boundary with NA SE). The reported logit_se is the information-based standard error 1 / sqrt(I(theta)), matching catR, TAM and most Rasch software.

method = "EAP" fits the model with mirt::mirt(..., itemtype = "Rasch") (MML) and obtains sum-score-based EAP estimates and posterior SDs via mirt::fscores(method = "EAPsum", full.scores = FALSE, full.scores.SE = TRUE). EAP estimates are finite at all score boundaries (the prior shrinks them inward), but they depend on the assumed normal prior on theta. Item parameters from MML differ slightly from the CML values used by the WLE path; for well-behaved data the difference is small.

Value

- If output = "kable": a knitr_kable object with columns "Ordinal sum score", "Logit score", and "Logit std.error", and a caption noting the estimation method.
- If output = "dataframe": a data.frame with columns raw_score, logit_score, and logit_se (one row per possible raw sum score from 0 to the theoretical maximum).
- If output = "ggplot": a ggplot object — points at each (logit_score, raw_score) with horizontal error bars at ± ci_multiplier × logit_se.

References

Warm, T. A. (1989). Weighted likelihood estimation of ability in item response theory. *Psychometrika, 54*(3), 427-450. doi:10.1007/BF02294627

Bock, R. D., & Mislevy, R. J. (1982). Adaptive EAP estimation of ability in a microcomputer environment. *Applied Psychological Measurement, 6*(4), 431-444. doi:10.1177/014662168200600405

Examples

```
set.seed(42)
sim_data <- as.data.frame(
  matrix(sample(0:3, 200 * 6, replace = TRUE), nrow = 200, ncol = 6)
)
colnames(sim_data) <- paste0("Item", 1:6)

# Default kable output, WLE
RMscoreSE(sim_data)

# Underlying data.frame
RMscoreSE(sim_data, output = "dataframe")

# ggplot figure
if (requireNamespace("ggplot2", quietly = TRUE)) {
  RMscoreSE(sim_data, output = "ggplot")
}

# EAP via mirt
RMscoreSE(sim_data, method = "EAP")
```

`RMtargeting` *Person-Item Targeting Plot (Wright Map)*

Description

Produces a three-panel targeting plot with a shared logit scale x-axis:

1. **Top**: Histogram of person location estimates, with a reference line for the mean (or median) and shading for ±1 SD (or ±1 MAD).
2. **Middle**: Inverted histogram of item threshold locations, with the same summary annotations.
3. **Bottom**: one bar per item, either partitioned into response-category bands (`panel = "categories"`, the default) or drawn as a dot-and-whisker plot of the individual thresholds (`panel = "thresholds"`).

Usage

```
RMtargeting(
  data,
  panel = c("categories", "thresholds"),
  robust = FALSE,
  sort_items = c("data", "location"),
  bins,
  xlim = c(-4, 4),
  ci_level = 0.95,
```

```
  category_labels = NULL,
  person_fill = "#0072B2",
  threshold_fill = "#D55E00",
  viridis_option = "G",
  viridis_begin = 0.9,
  viridis_end = 0.2,
  row_gap = NULL,
  height_ratios = c(3, 2, 5),
  output = "patchwork"
)
```

Arguments

`data`	A data.frame or matrix of item responses. Items must be scored starting at 0 (non-negative integers). Missing values (`NA`) are allowed.
`panel`	Character string selecting the bottom panel. `"categories"` (the default) draws each item as a bar partitioned into response-category bands, with the threshold estimates and their confidence intervals below it. `"thresholds"` draws the dot-and-whisker plot of item thresholds that was the only option before version 1.3.0.
`robust`	Logical. If `FALSE` (the default), histogram annotations use mean ± SD. If `TRUE`, median ± MAD is used instead.
`sort_items`	Character string controlling item ordering on the y-axis of the bottom panel. `"data"` (the default) preserves the column order in `data` (first item at top). `"location"` sorts items by their average threshold location (easiest at top, hardest at bottom).
`bins`	Integer. Number of bins for both histograms. Default is number of unique scores divided by 2 (rounded up), but no less than 11.
`xlim`	Numeric vector of length 2. Initial lower and upper limits for the shared x-axis. Automatically expanded if any person or item threshold values fall outside these limits.
`ci_level`	Numeric. Confidence level for the item threshold error bars. Default is `0.95` (95% CI). Set to `NULL` to hide error bars.
`category_labels`	Optional character vector of labels for the response categories, in ascending order and one per category. Used for the legend of the `"categories"` panel. Default `NULL` uses the category scores.
`person_fill`	Fill colour for the person histogram. Default `"#0072B2"` (blue).
`threshold_fill`	Fill colour for the item threshold histogram, and for the dot-and-whisker panel. Default `"#D55E00"` (vermillion).
`viridis_option`	Character. Viridis palette option for the category bands. Default `"G"` (mako).
`viridis_begin, viridis_end`	Numeric in $[0, 1]$. Start and end points of the viridis palette for the category bands. Defaults `0.9` and `0.2`, which runs the palette from light to dark so that higher categories are darker.

row_gap	Numeric. Vertical spacing between item rows in the "categories" panel. Default NULL uses 1, widened to 1.18 when at least one category collapses, so that its label has room above the bar.
height_ratios	Numeric vector of length 3 specifying the relative heights of the top (person), middle (threshold), and bottom (dot-whisker) panels. Default c(3, 2, 5).
output	Character string. "patchwork" (the default) returns the combined patchwork plot. "list" returns a named list of the three ggplot objects (p1, p2, p3) for further customisation.

Details

Together, the top and middle panels form a back-to-back histogram that makes it easy to assess whether the test is well-targeted to the sample. The bottom panel places the items on the same scale, so the category bands show which response is the most likely one at the locations where the persons actually sit.

Estimation method selection. The function checks whether any item response category has fewer than 3 observations. If all categories have at least 3 responses, item threshold locations and their standard errors are estimated via Conditional Maximum Likelihood (CML) using psychotools::pcmodel() (a dichotomous item is a 2-category PCM). If any category has fewer than 3 responses, the function falls back to Marginal Maximum Likelihood (MML) estimation via mirt::mirt() with itemtype = "Rasch" and SE = TRUE, which is more numerically stable under sparse-category conditions. A message is emitted when the MML fallback is used.

In both cases, item threshold locations are centered (shifted so the grand mean of all thresholds equals zero).

Person estimates are obtained by Warm's weighted likelihood (WLE) from the fitted item thresholds, consistent with the rest of the package. WLE is finite at extreme scores, so all-zero and perfect responders are located rather than dropped.

Confidence intervals for item thresholds are based on Wald-type intervals: threshold estimate ± z × SE, where z is the standard normal quantile corresponding to ci_level.

Category bands. With panel = "categories", each band spans the locations at which its response category is the most likely response. When an item's thresholds are ordered these boundaries are the Andrich thresholds themselves. When they are not, the disordered run is pooled by averaging and the categories it skips over, which are never the most likely response at any location, collapse to a red tick labelled with the category number. Red arrows below the bar give the size of each threshold reversal in logits. Ordered thresholds therefore leave no red marks at all.

The two outer bands are open-ended and fade towards the panel edge, since the lowest and highest categories have no outer boundary.

The ggplot2 and patchwork packages must be installed (they are in Suggests, not Imports).

Value

- If output = "patchwork": a patchwork object (combined ggplot).
- If output = "list": a named list with elements p1 (person histogram), p2 (threshold histogram), and p3 (the bottom panel selected by panel).

References

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

See Also

psychotools::pcmodel(), mirt::mirt()

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE) &&
    requireNamespace("patchwork", quietly = TRUE)) {
  # Polytomous example
  set.seed(42)
  sim_data <- as.data.frame(
    matrix(sample(0:3, 200 * 8, replace = TRUE), nrow = 200, ncol = 8)
  )
  colnames(sim_data) <- paste0("Item", 1:8)

  # Default: category bands, mean/SD, data order, 95% CI
  RMtargeting(sim_data)

  # Category bands with labels
  RMtargeting(sim_data, category_labels = c("Never", "Sometimes",
                                            "Often", "Always"))

  # The dot-and-whisker panel
  RMtargeting(sim_data, panel = "thresholds")

  # Robust (median/MAD), sorted by location, 84% CI
  RMtargeting(sim_data, robust = TRUE, sort_items = "location",
              ci_level = 0.84)

  # Get list of sub-plots for customisation
  plots <- RMtargeting(sim_data, output = "list")
  plots$p1 + ggplot2::ggtitle("My custom title")

  # Dichotomous example
  sim_bin <- as.data.frame(
    matrix(sample(0:1, 200 * 10, replace = TRUE), nrow = 200, ncol = 10)
  )
  colnames(sim_bin) <- paste0("Item", 1:10)
  RMtargeting(sim_bin)
}
```

RMUreliability *Relative Measurement Uncertainty (RMU)*

Description

Bayesian-style reliability estimate (Bignardi, Kievit & Bürkner, 2025) computed from a matrix of posterior or plausible-value draws. The columns of `input_draws` are split at random into two halves; reliability is the Pearson correlation across persons of paired columns from the two halves, summarised across pairs as a posterior mean with HDCI.

Usage

```
RMUreliability(input_draws, level = 0.95, verbose = FALSE)
```

Arguments

`input_draws`	Numeric matrix or data.frame of draws. Rows are subjects; columns are draws. Must have at least two columns; ideally many.
`level`	Numeric in (0, 1). Width of the HDCI returned. Default `0.95`.
`verbose`	Logical. Print summary information about the input. Default `FALSE`.

Details

Adapted (with permission, GPL-2/3) from `rmu()`, formerly `reliability()`, in the 'gbtoolbox' package, https://github.com/giac01/gbtoolbox.

The function silently returns 0 for any column pair where either side has zero variance (the correlation is undefined there).

Requires the `ggdist` package (Suggests).

Value

A 1-row data.frame with columns `rmu_estimate`, `hdci_lowerbound`, `hdci_upperbound`, plus the `.width`/`.point`/`.interval` metadata columns added by `ggdist::mean_hdci()`.

References

Bignardi, G., Kievit, R., & Bürkner, P. C. (2025). A general method for estimating reliability using Bayesian Measurement Uncertainty. *PsyArXiv*. doi:10.31234/osf.io/h54k8_v1

See Also

`RMreliability()`

Index

∗ **datasets**
phq9, 3

easyRasch2-reproducibility, *8*, *22*, *26*, *35*, *49*, *52*, *65*, *68*, *75*, *83*, *89*, *94*, *107*, *111*

ggplot2::ggplot, *38*, *100*, *103*

knitr::kable(), *59*, *97*

mirt::mirt(), *58*, *121*

partgam_DIF, *10, 11*
partgam_LD, *75*, *77*
phq9, 3
psychotools::pcmodel(), *58*, *121*

RMdifGamma, 4, *8*, *10*, *12*, *18*, *43*
RMdifGamma(), *17*, *41*, *43*
RMdifGammaCutoff, *5*, *7*, 8, *11, 12*
RMdifGammaCutoff(), *6*, *17*
RMdifGammaPlot, *10*, 11
RMdifLR, 12, *18*
RMdifTree, 14
RMdimCFA, 19, *22–24*
RMdimCFACutoff, *19, 20*, 21, *24*
RMdimCFACutoff(), *20*
RMdimCFAPlot, *20*, *22, 23*, 24
RMdimMartinLof, *23*, 25, *29–31*
RMdimMartinLofResiduals, *27*, 29
RMdimResidualPCA, *23*, *28*, 31, *35*
RMdimResidualPCACutoff, *28*, *32, 33*, 34
RMitemCatProb, 36
RMitemCatProb(), *104*
RMitemHierarchy, 39
RMitemHierarchy(), *38*
RMitemICCPlot, 41
RMitemICCPlot(), *38*, *40*
RMitemInfit, 44, *48–50*, *52–55*, *57*, *66*
RMitemInfit(), *57*, *96*
RMitemInfitCutoff, *45–47*, 48, *51–54*, *56, 57*
RMitemInfitCutoff(), *46*
RMitemInfitCutoffMI, 51, *54, 55*
RMitemInfitCutoffPlot *(*RMitemInfitPlot*)*, 56
RMitemInfitMI, *52*, 53, *53*
RMitemInfitPlot, *24*, *52, 53*, 56
RMitemParameters, 58
RMitemParameters(), *88*, *96–98*, *110*
RMitemRestscore, *43*, 60, *65–71*
RMitemRestscore(), *65*, *70*
RMitemRestscoreBoot, 64
RMitemRestscoreCutoff, *9*, *61*, *63*, 67, *70, 71*
RMitemRestscoreCutoff(), *62*, *65*
RMitemRestscorePlot, *63*, *69*, 70
RMlocdepGamma, 71, *77, 78*
RMlocdepGamma(), *6*
RMlocdepGammaCutoff, *72*, 74, *74*, *77, 78*
RMlocdepGammaCutoff(), *73*
RMlocdepGammaPlot, *74*, 77, *77*, *86*
RMlocdepQ3, *72, 73*, 79, *82–86*
RMlocdepQ3Cutoff, *34*, *79, 80*, 82, *85, 86*
RMlocdepQ3Cutoff(), *81*
RMlocdepQ3Plot, *81*, *84*, 85
RMpersonChange, 87
RMpersonChange(), *113*, *115, 116*
RMpersonFit, 93
RMpersonParameters, 96
RMpersonParameters(), *59*, *88*, *92*, *96*, *110*, *117*
RMplotBar, 99
RMplotBar(), *103*
RMplotStackedbar, 101
RMplotStackedbar(), *100*
RMplotTile, 103
RMplotTile(), *100*, *103*
RMreliability, 106
RMreliability(), *98*, *109*, *111, 112*, *122*

RMreliabilityCurve, 109
RMreliabilityCurve(), *92*, *107–109*
RMretestSD, 113
RMretestSD(), *88*, *92*
RMscoreSE, 116
RMscoreSE(), *40*, *59*, *92*, *96*, *98*, *112*
RMtargeting, 118
RMtargeting(), *40*, *59*, *112*
RMUreliability, 121
RMUreliability(), *106*, *109*