Package 'easyRasch2'

June 8, 2026

Title Psychometric Analysis with Rasch Measurement Theory

Version 0.8.0

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, Yen's Q3 local-dependence statistic, the largest residual-PCA eigenvalue, and ordinal CFA fit indices. 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

RoxygenNote 7.3.3

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

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

Depends R (>= 4.1.0)

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

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

Config/testthat/edition 3

VignetteBuilder knitr

NeedsCompilation no

Author Magnus Johansson [aut, cre] (ORCID:
<https://orcid.org/0000-0003-1669-592X>),
Nicklas Korsell [ctb] (PCM simulation code),
Mirka Henninger [ctb] (ORCID: <https://orcid.org/0000-0003-4676-2361>,
MH / partial-gamma effect-size and ETS-classification algorithms in
dif_tree.R, adapted under MIT licence from the raschtreeMH and
effecttree packages),
Jan Radek [ctb] (ORCID: <https://orcid.org/0009-0003-8842-9206>,
partial-gamma effect-size and ETS-classification algorithms in
dif_tree.R, adapted under MIT licence from the effecttree package)

Maintainer Magnus Johansson <pgmj@pm.me>

Repository CRAN

Date/Publication 2026-06-08 17:50:12 UTC

Contents

---

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, 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.
`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).

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." (a star-string indicator from `iarm::partgam_DIF()`). When `cutoff` is provided, additional columns "Gamma low", "Gamma high", and "Flagged" are included.
- 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.

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

See Also

RMdifGammaCutoff

Examples

```
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)
cutoff_res <- RMdifGammaCutoff(sim_data, dif_var = dif_group,
                                  iterations = 100, parallel = FALSE,
                                  seed = 42)
RMdifGamma(sim_data, dif_group, cutoff = cutoff_res)
```

RMdifGammaCutoff *Simulation-Based Partial Gamma DIF Cutoff Determination*

Description

Uses parametric bootstrap simulation to determine appropriate cutoff values for partial gamma DIF analysis via partgam_DIF. Under a correctly fitting Rasch model where the DIF variable is unrelated to item responses (i.e., no true DIF), this function generates the expected distribution of absolute partial gamma values per item, providing empirical critical values.

Usage

```
RMdifGammaCutoff(
  data,
  dif_var,
  iterations = 250,
  parallel = TRUE,
  n_cores = NULL,
  verbose = FALSE,
  seed = NULL,
  cutoff_method = "hdci",
  hdci_width = 0.99
)
```

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). The actual group labels are used to determine the number of groups and their relative sizes; during simulation, respondents are randomly assigned to groups with the same proportions, so there is no true DIF by construction.
iterations	Integer. Number of simulation iterations (default 250).
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.
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 0.99 (99\ cutoff_method = "quantile".

Details

For each simulation iteration the function:

1. Resamples person parameters (thetas) with replacement from ML estimates.
2. Simulates item response data under a Rasch model (dichotomous via psychotools::rrm() or polytomous via an internal partial credit simulator).
3. Creates a random DIF variable by sampling group labels with the same proportions as the observed dif_var, so there is **no true DIF** by construction.
4. Computes partial gamma DIF statistics via iarm::partgam_DIF().

The distribution of partial gamma values across iterations provides empirical critical values per item. Values from real data that fall outside these bounds suggest DIF that exceeds what would be expected by chance under a correctly fitting Rasch model. Failed iterations (e.g., due to convergence issues or degenerate data) are silently discarded.

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

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`, 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.

`sample_n` Number of complete cases used.

`sample_summary` Summary statistics of estimated person parameters.

`item_names` Character vector of item names from data.

`dif_group_sizes` Named integer vector of group sizes used in the simulation (matches proportions in the observed `dif_var`).

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

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

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

See Also

`partgam_DIF`

Examples

```
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
```

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 via partgam_DIF.

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

Value

A ggplot object.

See Also

RMdifGammaCutoff, RMdifGamma

Examples

```
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("ggplot", "kable", "dataframe"),
  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 "ggplot" (default), "kable", or "dataframe". 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

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

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

# Threshold-level kable, sorted by MaxDiff
RMdifLR(pcmdat2, dif_var = grp,
        level = "threshold", output = "kable", 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 (in the ETS Delta scale) for dichotomous data, or the partial gamma coefficient for polytomous data – and classifies them into ETS A/B/C categories. Optionally, an iterative purification step (Holland & Thayer, 1988; Bjorner et al., 1998) is applied, and tree stability across resamples can be assessed via `stablelearner::stabletree()`.

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"),
  ...
)
```

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"`, or `"plot"`. `"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)`). 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.

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.

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

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

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
  }
}
```

`RMdimCFACutoff` *Posterior-predictive CFA fit-index cutoffs under PCM unidimensionality*

Description

Tests whether the observed one-factor categorical-CFA fit indices (CFI, RMSEA, SRMR) are consistent with the data being generated by a unidimensional Rasch / Partial Credit Model. The simulation generates `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 the three fit indices are recorded. The result is a parametric-bootstrap null distribution against which the observed CFA fit can be compared, one-sided in the unfavourable direction for each index (CFI from below; RMSEA and SRMR from above).

Usage

```
RMdimCFACutoff(
  data,
  iterations = 250L,
  percentile = 99,
  output = c("kable", "list"),
  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 one-sided cutoff. Default 99. Higher = stricter: • For CFI (higher = better fit), the cutoff is the (100 - percentile)-th percentile of the simulated null distribution; observed values *below* this are flagged. • For RMSEA and SRMR (lower = better fit), the cutoff is the percentile-th percentile; observed values *above* this are flagged.
output	Character. "kable" (default) returns a formatted knitr::kable() summary of observed values vs cutoffs, with the full result list attached as attr(., "result"). "list" returns the result list directly.
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.
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

At the default 99th-percentile cutoff, an item is flagged when the observed value lies in the worst 1\ distribution.

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; the Yuan-Bentler mean-variance-adjusted "robust" variants are sometimes NA at small n and would produce holes in the simulated null distribution. SRMR is reported unchanged. For percentile-based comparison the binding requirement is that the same metric is computed for observed and simulated iterations, which both variants satisfy.

Why a null distribution. A perfectly PCM-unidimensional dataset will typically not yield CFA fit indices at their ideal values (CFI = 1, RMSEA = 0). Two reasons: (1) PCM uses a logistic threshold

structure while WLSMV uses a probit-link via the polychoric correlation matrix, so there is a small built-in metric mismatch even under correct unidimensionality; (2) finite samples produce sampling variability in the polychoric correlations. The simulated distribution captures both. Comparing observed to this distribution is more honest than rule-of-thumb cutoffs (CFI > 0.95, RMSEA < 0.06, SRMR < 0.08) which were derived under continuous-data ML and do not transfer cleanly to ordinal WLSMV.

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

Companion functions. See `RMdimCFAPlot` for a faceted visualisation of the observed value against each simulated distribution. This test complements `RMdimResidualPCA` (which identifies *which* items deviate) and `RMdimMartinLof` (which tests a specific hypothesised partition).

Value

If `output = "kable"`, a `knitr_kable` object summarising each index against its cutoff, with the full result list as `attr(., "result")`. If `output = "list"`, that list directly. The list has components:

`observed` Named numeric vector of observed CFA fit indices (`cfi.scaled`, `rmsea.scaled`, `srmr`).

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

`percentile` Numeric: the strictness setting used.

`cutoffs` Named numeric vector (`cfi`, `rmsea`, `srmr`) of simulated cutoffs at the chosen percentile.

`flagged` Named logical vector indicating whether each observed index falls outside its cutoff in the unfavourable direction.

`actual_iterations` Number of successful MC iterations.

`sample_n` Number of complete cases used.

`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

`RMdimCFAPlot`, `RMdimResidualPCA`, `RMdimMartinLof`

Examples

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

  # Few iterations for a fast example; use 250+ in real analyses
  result <- RMdimCFACutoff(raschdat1[, 1:8], iterations = 50,
                           parallel = FALSE, seed = 1, output = "list")
  result$observed
  result$flagged

  if (requireNamespace("ggplot2", quietly = TRUE)) {
    RMdimCFAPlot(result)
  }
}
```

`RMdimCFAPlot` *Plot observed CFA fit indices against the simulated null distribution*

Description

Faceted histograms of the simulated CFI, RMSEA, and SRMR distributions (one panel per index), with the observed value overlaid as a vertical line. The line is coloured red when the observed value falls outside the chosen percentile of the simulation in the unfavourable direction (CFI from below; RMSEA / SRMR from above), and grey otherwise.

Usage

```
RMdimCFAPlot(cutoff_res, percentile = NULL)
```

Arguments

`cutoff_res`	The list returned by `RMdimCFACutoff` with `output = "list"`, or the kable returned with `output = "kable"` (the underlying list is read from `attr(., "result")`).
`percentile`	Numeric in (50, 100) or `NULL`. When supplied, the cutoff and the flagged status are recomputed at this percentile from the simulated distribution stored in `cutoff_res` (no re-simulation needed). When `NULL` (default), the percentile that was used by the original `RMdimCFACutoff()` call is reused.

Value

A `ggplot` object.

See Also

`RMdimCFACutoff`

Examples

```
if (requireNamespace("lavaan", quietly = TRUE) &&
    requireNamespace("ggplot2", quietly = TRUE)) {
  data("raschdat1", package = "eRm")
  res <- RMdimCFACutoff(raschdat1[, 1:8], iterations = 50,
                    parallel = FALSE, seed = 1, output = "list")
  RMdimCFAPlot(res)                # use the percentile from RMdimCFACutoff()
  RMdimCFAPlot(res, percentile = 95) # override (no re-simulation needed)
}
```

RMdimMartinLof *Martin-Lof 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). Rows with any `NA` are dropped.
`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.

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. For dichotomous data the fast algorithm of Christensen & Kreiner (2007, p. 23) is used (sample without replacement weighted by item easinesses). For polytomous data the recursive γ-function approach is used, with each item's response sampled conditional on the remaining items' joint score distribution (computed via `psychotools::elementary_symmetric_functions()`).

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.

Value

A list with components:

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

`p_value` Monte Carlo p-value with `(n_exceed + 1) / (n + 1)` correction.

`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.

`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

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

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

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). Rows with any `NA` are dropped.
`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.

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., 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
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 (`eRm::RM()` for dichotomous data, `eRm::PCM()` for polytomous data — chosen automatically), extracts standardized residuals via `eRm::itemfit()$st.res`, 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, 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.
`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 `"loadings"` for a ggplot of PC1 loadings against item locations (similar in spirit to the loadings-by-location plot used in `easyRasch::RIloadLoc`).

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 returned by `eRm::itemfit()$st.res`, which are `(observed - expected) / sqrt(var)` under the Rasch model. 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 computed as the per-item mean of Andrich thresholds for polytomous data (PCM) or as `-beta` for dichotomous data (RM).

The variance partition follows Linacre's CML/MLE convention: per-item observed variance is compared to per-item *expected* variance under the fitted model, summed across items. Expected scores are computed from MLE person locations (via `eRm::person.parameter()`) and the CML item parameters from `eRm::RM()` / `eRm::PCM()`. Persons with extreme raw scores (no finite MLE theta) are excluded from the partition, matching the sample used by `eRm::SepRel()` and the PCA itself.

Value

- If `output = "kable"`: a `knitr_kable` object with columns Component, Eigenvalue, Proportion of variance (and `Flagged` when `cutoff` is provided). The caption gives the variance partition (% of total observed variance explained by measures vs. unexplained), the model fitted, sample size, and cutoff metadata if applicable.
- If `output = "dataframe"`: a data.frame with columns `Component`, `Eigenvalue`, `Proportion_of_variance` (and `Flagged` when `cutoff` is provided). 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 = "loadings"`: 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
RMdimResidualPCA(dat, output = "loadings")

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

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.

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).

Per iteration: theta values are sampled with replacement from the WLE/MLE estimates derived from the fitted full-sample model; response data are simulated under the model (`psychotools::rrm` for dichotomous, partial-credit simulator for polytomous); the model is refitted with `eRm::RM()` / `eRm::PCM()`; standardized residuals are extracted via `eRm::itemfit()$st.res`; `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. Polytomous items are fitted with the Partial Credit Model via `eRm::PCM()`; dichotomous items are fitted with the Rasch model via `eRm::RM()`. 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 $\delta_{i,1}, \ldots, \delta_{i,K_i}$ from `eRm::thresholds()`, 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 PCM thresholds returned by `eRm::thresholds()` 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

```
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) for all items in `data`. Each panel shows the model-expected ICC together with average observed item scores within class intervals (or for every possible total score, with `method = "score"`). When a `dif_var` is supplied, observed averages are computed separately per group, so each panel becomes a visual DIF check.

Usage

```
RMitemICCPlot(
  data,
  class_intervals = 4L,
  method = c("cut", "score"),
  dif_var = 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.
`class_intervals`	Integer. Number of class intervals used to group respondents when `method = "cut"`. Default 4. Each class interval needs enough respondents to compute a stable average observed item score; with small samples or many possible total scores, lower this. Ignored when `method = "score"`.
`method`	One of `"cut"` (default) or `"score"`. `"cut"` groups respondents into `class_intervals` adjacent bins; `"score"` uses every possible total raw score.
`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. Default `NULL` (no DIF).
`output`	One of `"patchwork"` (default) – composite patchwork figure – or `"list"` – a named list of per-item `ggplot` objects.

Details

Wraps `iarm::ICCplot()` (Santiago, Mueller, & Müller, 2022) for the per-item computation and composes the panels with `patchwork::wrap_plots()`.

Class intervals vs every total score. With `method = "cut"` (the default) respondents are divided into `class_intervals` bins of roughly equal size based on their total raw score; each bin's average observed item score is plotted as a point. With `method = "score"`, the average observed item score is computed for every observed total raw score. The cut method is the conventional Rasch-analysis default and works well at moderate sample sizes; the score method is more granular but can be unstable when raw-score totals are sparsely populated.

DIF mode. With `dif_var` supplied, `iarm::ICCplot()` computes average observed item scores separately for each level of the grouping variable. Each item's panel then shows the model-expected ICC (one black line) plus one set of observed points per DIF group. Stark visual divergence between groups suggests DIF.

Dependencies. Requires `iarm` for the per-item curve, and `patchwork` + `ggplot2` for the figure composition. All three are in Suggests with runtime guards.

Value

Either a `ggplot` / `patchwork` composite (default), or a list of `ggplot` objects (`output = "list"`), one per item.

References

Mueller, M., & Santiago, P. H. R. (2022). *iarm: Item Analysis in Rasch Models*. R package version 0.4.3. https://CRAN.R-project.org/package=iarm

See Also

iarm::ICCplot(), RMtargeting(), RMplotTile()

Examples

```
data("pcmdat2", package = "eRm")
# NB: with `method = "cut"` the quantile-based binning of total scores
#     may collapse to fewer bins than requested when the score
#     distribution is peaked. `pcmdat2` happens to accept 3 or 5 but
#     not 4. See `?iarm::ICCplot`; `method = "score"` avoids the issue.
RMitemICCPlot(pcmdat2, class_intervals = 5)

# With DIF grouping
set.seed(1)
grp <- factor(sample(c("A", "B"), nrow(pcmdat2), replace = TRUE))
RMitemICCPlot(pcmdat2, class_intervals = 5, dif_var = grp)
```

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, output = "kable", sort)
```

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. When provided, adds columns Infit_low, Infit_high, and Flagged (logical; TRUE when Infit_MSQ falls outside the credible range) to the result.

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

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.

For **dichotomous** data (maximum score = 1), a Rasch model is fitted via eRm::RM(). Item locations are the negative beta parameters. Person locations are estimated via eRm::person.parameter().

For **polytomous** data (maximum score > 1), a Partial Credit Model is fitted via eRm::PCM(). Item average locations are taken from the "Location" column of the threshold parameter table returned by eRm::thresholds(); if that column is absent, row means of the threshold columns are used instead. Person locations are estimated via eRm::person.parameter().

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).

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).

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

RMitemInfitCutoff

Examples

```
# 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)
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")
```

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 = 250,
  parallel = TRUE,
  n_cores = NULL,
  verbose = FALSE,
  seed = NULL,
  cutoff_method = "hdci",
  hdci_width = 0.999
)
```

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 250).
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.
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 0.999 (99.9% HDCI). Ignored when cutoff_method = "quantile".

Details

For each simulation iteration, person parameters (thetas) are resampled with replacement from ML estimates, response data are simulated under the Rasch model, the model is refitted, and conditional infit and outfit MSQ statistics are computed via iarm::out_infit(). The distribution of these statistics across iterations provides empirical critical values per item. Failed iterations (e.g., due to convergence issues or degenerate data) are silently discarded.

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

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.

sample_n Number of complete cases used.

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").

See Also

RMitemInfit

Examples

```
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.
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 RMitemInfitCutoffPlot:

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.

See Also

RMitemInfitCutoff, RMitemInfitMI, RMitemInfitCutoffPlot

Examples

```
if (requireNamespace("mice", quietly = TRUE)) {
  # Create example data with missing values
  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)
  # Introduce ~10% MCAR missingness
  sim_data[sample(length(sim_data), 0.10 * length(sim_data))] <- NA

  # 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)
}
```

RMitemInfitCutoffPlot *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

```
RMitemInfitCutoffPlot(simfit, data, output = "infit")
```

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.
output	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).

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, 0.999).

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 model (eRm::RM() for dichotomous data or eRm::PCM() for polytomous data) 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).

Value

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

See Also

RMitemInfitCutoff, RMitemInfit

Examples

```
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)
RMitemInfitCutoffPlot(cutoff_res)

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

# Both infit and outfit panels side by side
RMitemInfitCutoffPlot(cutoff_res, data = sim_data, output = "both")
```

`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 `$item_cutoffs`): the data.frame is extracted automatically and metadata is included in the kable caption. • The `$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.
`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 (`eRm::RM()` for dichotomous data or `eRm::PCM()` for polytomous data).
2. Computes conditional infit MSQ and its standard error via `iarm::out_infit()`.
3. Computes item and person locations.

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).

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`).

See Also

`RMitemInfit`, `RMitemInfitCutoffMI`

Examples

```
if (requireNamespace("mice", quietly = TRUE)) {
  # Create example data with missing values
  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)
  sim_data[sample(length(sim_data), 0.10 * length(sim_data))] <- NA

  # 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")
}
```

`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, 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.
`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). See `stats::p.adjust()` for available methods.

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.

For **dichotomous** data (maximum score = 1), a Rasch model is fitted via `eRm::RM()`. Item locations are the negative beta parameters. Person locations are estimated via `eRm::person.parameter()`.

For **polytomous** data (maximum score > 1), a Partial Credit Model is fitted via `eRm::PCM()`. Item average locations are the row-means of the threshold parameter table returned by `eRm::thresholds()`. Person locations are estimated via `eRm::person.parameter()`.

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).

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, significance level, item location, and item location relative to the sample mean person location.
- If `output = "dataframe"`: a data.frame with columns `Item`, `Observed`, `Expected`, `Difference`, `p_adjusted`, `Significance`, `Location`, and `Relative_location`.

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).

References

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

Examples

```
# 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")
```

`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.
`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

Useful with large samples, where the asymptotic test underlying `RMitemRestscore` can flag items that are not practically misfitting; bootstrapping gives a more nuanced view of the probability of an item actually being misfit.

For **dichotomous** data (maximum score = 1), the full-sample model is fitted via `eRm::RM()` and bootstrap iterations refit via `eRm::RM()` with `se = FALSE` for speed.

For **polytomous** data (maximum score > 1), the full-sample model is fitted via `eRm::PCM()` (so item locations can be obtained from `eRm::thresholds()`) and bootstrap iterations refit via `psychotools::pcmodel(..., hessian = FALSE)` for speed. `iarm::item_restscore()` accepts both model classes.

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.

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

```
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))
}
```

`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, 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.

`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).

Value

- If `output = "kable"`: an object of class `"RMlocdepGamma"`. Internally a list with two `knitr_kable` elements — `$direction1` (rest score = total - Item2) and `$direction2` (rest score = total - Item1) — each with columns "Item 1", "Item 2", "Partial gamma", "Adj. p-value (BH)", and "p-value sign." (a star-string indicator from `iarm::partgam_LD()`). 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`, `padj_bh`, `Significance`. When `cutoff` is provided, columns `gamma_low`, `gamma_high`, and `flagged` are also included.

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

RMlocdepGammaCutoff, RMlocdepGammaPlot

Examples

```
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
cutoff_res <- RMlocdepGammaCutoff(sim_data, iterations = 100, parallel = FALSE,
                            seed = 42)
RMlocdepGamma(sim_data, cutoff = cutoff_res)
```

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 = 250,
  parallel = TRUE,
  n_cores = NULL,
  verbose = FALSE,
  seed = NULL,
  cutoff_method = "hdci",
  hdci_width = 0.99
)
```

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 250).
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.
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 0.99 (99\ cutoff_method = "quantile".

Details

For each simulation iteration the function:

1. Resamples person parameters (thetas) with replacement from ML estimates.
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 LD statistics via iarm::partgam_LD().

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.

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

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). Contains results from direction 1 only (rest score = total - Item2), which is the conventional direction.

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_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

```
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"). Only direction 1 (rest score = total - Item2) is plotted, matching the convention used in the simulation.

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, 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

```
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 *Q3 Residual Correlations for Local Dependence Assessment*

Description

Computes Yen's Q3 residual correlations between item pairs using a Rasch model fitted via Marginal Maximum Likelihood (MML) in `mirt`. 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")
```

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. Either a single numeric value (added to the mean off-diagonal Q3 correlation to produce the dynamic cut-off threshold) or the full list returned by `RMlocdepQ3Cutoff` (from which `$suggested_cutoff` is extracted automatically). When `NULL` (default), the raw Q3 residual correlation matrix is returned without any dynamic cut-off applied.
`output`	Character string controlling the return value. Either `"kable"` (default) for a formatted `knitr::kable()` table, or `"dataframe"` for the underlying numeric data.frame.

Details

The Q3 statistic (Yen, 1984) is the correlation between residuals of pairs of items after accounting for the latent trait. Under local independence, Q3 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 Q3 values plus `cutoff`, following the approach of Christensen et al. (2017). Use `RMlocdepQ3Cutoff` to obtain a simulation-based cutoff recommendation.

`mirt` is used for model fitting here because Q3 requires model-based expected responses, which are most readily available from MML estimation.

Value

- If `output = "kable"`: a `knitr_kable` object showing the lower triangle of the Q3 correlation matrix. When `cutoff` is provided, a footnote describing the dynamic cut-off is included and an extra `above_cutoff` column marks rows containing at least one value above the threshold.
- If `output = "dataframe"`: a data.frame (rounded to 2 decimal places) with the lower triangle of the Q3 correlation matrix; the upper triangle and diagonal are set to `NA`. When `cutoff` is provided, an additional logical column `above_cutoff` indicates whether the row contains any value exceeding the dynamic cut-off.

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 Q3: Identification of local dependence in the Rasch model. *Applied Psychological Measurement, 41*(3), 178–194. doi:10.1177/0146621616677520

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)
  RMlocdepQ3(sim_data, cutoff = cutoff_res$suggested_cutoff)
}
```

RMlocdepQ3Cutoff *Simulation-Based Q3 Cutoff Determination*

Description

Uses parametric bootstrap simulation to determine an appropriate cutoff value for RMlocdepQ3. Under a correctly fitting Rasch model, Q3 residuals have an unknown distribution; this function simulates that distribution and returns empirical percentiles.

Usage

```
RMlocdepQ3Cutoff(
  data,
  iterations = 500,
  parallel = TRUE,
  n_cores = NULL,
  verbose = FALSE,
  seed = NULL,
  cutoff_method = "hdci",
  hdci_width = 0.99
)
```

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 500).
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.
cutoff_method	Character. Method used to compute per-pair Q3 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 0.99. Ignored when cutoff_method = "quantile".

Details

For each simulation iteration, person parameters (thetas) are resampled with replacement from ML estimates, response data are simulated under the Rasch model, a `mirt` Rasch model is fitted to the simulated data, and Q3 residuals are extracted. 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 (via `eRm::RM()` and `psychotools::rrm()`) and **polytomous** data (via `eRm::PCM()` and 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 in the original data.

`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"`).

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 Q3 Residual Correlations*

Description

Visualises the distribution of simulation-based Yen's Q3 residual correlations per item pair from RMlocdepQ3Cutoff, optionally overlaying observed Q3 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 Q3 values. Items must be scored starting at 0 (non-negative integers). When provided, the plot includes orange diamond markers for the observed Q3 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 plot shows one row per item pair (labelled as "Item1 - Item2"). Only the upper triangle of the Q3 matrix is plotted (pairs are unordered under symmetric Q3, 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. Fits a Rasch model to `data` via `mirt::mirt()` and extracts observed Q3 residual correlations.
2. Overlays observed Q3 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 `ggplot` object.

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)
}
```

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.

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.

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",
  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.

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".
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.

Value

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

Examples

```
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) via `eRm::SepRel()`, empirical reliability via `mirt::empirical_rxx()`, 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 empirical reliability and the plausible-value seed. One of "WLE" (default), "EAP", "MAP", "ML". Plausible draws themselves are produced by Metropolis-Hastings.
boot	Logical. If TRUE, run a non-parametric bootstrap to obtain CIs for PSI and Empirical 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.
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 **Empirical** reliability are obtained by non-parametric bootstrap (refitting both the eRm and mirt models on each resample). The RMU interval is the HDCI of correlations across plausible-value draws, averaged over rmu_iter random splits of the draws.

This is a CRAN-friendly replacement for easyRasch::RIreliability(). The TAM-based plausible-values path has been removed; mirt is the only MML backend.

For **dichotomous** data (max = 1), mirt::mirt(..., itemtype = "Rasch") and eRm::RM() are fitted. For **polytomous** data, the same itemtype is used in mirt and eRm::PCM() in eRm.

PSI is computed by eRm::SepRel() on the person-parameter object from eRm::person.parameter(). Note that PSI excludes respondents with extreme (min/max) raw scores by construction.

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).

Bootstrap iterations that fail to converge in either mirt or eRm 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

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

See Also

RMUreliability()

Examples

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

  # Bootstrap CI for PSI and Empirical
  # (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)
}
```

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 eRm, 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 with eRm::RM() or eRm::PCM() (CML) and applies Warm's Weighted Likelihood correction for the bias inherent in MLE-based person estimates at the score boundaries. Score-boundary estimates are obtained via the theta_range search; if a boundary cannot be solved within that range it is returned as Inf / -Inf with NA SE. This path uses lightly patched copies of the iarm 0.4.x person-estimation code so that boundary scores converge cleanly across the full theta range.

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
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**: Dot-and-whisker plot of individual item thresholds with confidence intervals based on threshold standard errors.

Usage

```
RMtargeting(
  data,
  robust = FALSE,
  sort_items = c("data", "location"),
  bins,
```

```
  xlim = c(-4, 4),
  ci_level = 0.95,
  person_fill = "#0072B2",
  threshold_fill = "#D55E00",
  height_ratios = c(3, 2, 5),
  output = "figure"
)
```

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.
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 plus one, but no less than 15.
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.
person_fill	Fill colour for the person histogram. Default "#0072B2" (blue).
threshold_fill	Fill colour for the item threshold histogram. Default "#D55E00" (vermillion).
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. "figure" (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.

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 eRm::RM() (dichotomous) or eRm::PCM() (polytomous), with SEs from eRm::thresholds(). 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 via Maximum Likelihood (ML) from `eRm::person.parameter()`, which uses spline interpolation to extrapolate location estimates for persons with extreme scores (all-zero or perfect). Persons for whom the spline interpolation fails receive `NA` and are excluded from the histogram.

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`.

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

Value

- If `output = "figure"`: a `patchwork` object (combined `ggplot`).
- If `output = "list"`: a named list with elements `p1` (person histogram), `p2` (threshold histogram), and `p3` (item threshold dot-whisker plot).

References

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

See Also

`eRm::PCM()`, `eRm::RM()`, `mirt::mirt()`, `eRm::person.parameter()`, `eRm::thresholds()`

Examples

```
# 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: mean/SD, data order, 95% CI
RMtargeting(sim_data)

# 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 https://github.com/giac01/gbtoolbox/blob/main/R/reliability.R.

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