Package 'MiCT'

August 9, 2026

Type Package

Title Minimal Important Change and Threshold Estimation

Version 2.0.0

Description Provides methods for estimating minimal important change (MIC) and interpretation thresholds for multi-item questionnaires and single-item continuous or ordinal measures. Methods include predictive modelling, adjusted predictive modelling, improved adjusted predictive modelling using anchor reliability, confirmatory factor analysis for anchor reliability, longitudinal confirmatory factor analysis for MIC estimation, longitudinal confirmatory factor analysis-based MIC estimation for single-item measures, and confirmatory factor analysis-based threshold estimation for single-item and multi-item measures. Implemented methods include those developed by Terluin et al. (2015) <doi:10.1016/j.jclinepi.2015.03.015>, Terluin et al. (2017) <doi:10.1016/j.jclinepi.2016.12.015>, Terluin et al. (2022) <doi:10.1016/j.jclinepi.2022.04.018>, Terluin et al. (2023) <doi:10.1007/s11136-023-03355-8>, Terluin et al. (2024) <doi:10.1007/s11136-023-03577-w>, Terluin et al. (2024) <doi:10.1007/s11136-024-03763-4>, and Terluin et al. (2026) <doi:10.1007/s11136-025-04134-3>.

Depends R (>= 4.1.0)

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Imports lavaan, MASS, mirt, pROC

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

URL https://github.com/yhpua/MiCT, https://yhpua.github.io/MiCT/

BugReports https://github.com/yhpua/MiCT/issues

NeedsCompilation no

Author Iris Eekhout [aut] (Original package author),
Berend Terluin [aut],
Yong-Hao Pua [aut, cre]

Maintainer Yong-Hao Pua <gmspuyh@duke-nus.edu.sg>

Repository CRAN

Date/Publication 2026-08-09 08:30:08 UTC

Contents

`equalize_levels` *Equalize Paired Item Response Levels*

Description

`equalize_levels()` prepares paired Time 1 and Time 2 PROM items for longitudinal CFA by ensuring that each item has the same observed response levels at both time points. Sparse or mismatched response categories are collapsed consistently within each item pair.

Usage

```
equalize_levels(
  data,
  pair_by = c("position", "suffix"),
  t1_suffix = NULL,
  t2_suffix = NULL,
  min_resp = 5L,
```

```
  verbose = TRUE,
  print_tables = TRUE
)
```

Arguments

`data`	A data frame containing paired item variables. Do not include the transition rating variable.
`pair_by`	Pairing method. `"position"` assumes the first half of columns are Time 1 items and the second half are Time 2 items. `"suffix"` detects item pairs using `t1_suffix` and `t2_suffix`.
`t1_suffix`	Regex suffix identifying Time 1 items when `pair_by = "suffix"`. Use `""` when Time 1 items have no suffix.
`t2_suffix`	Regex suffix identifying Time 2 items when `pair_by = "suffix"`.
`min_resp`	Integer. Minimum number of responses required in each response category at both time points. Categories with fewer responses are collapsed.
`verbose`	Logical. If `TRUE`, prints progress messages.
`print_tables`	Logical. If `TRUE`, prints before/after frequency tables only for item pairs that required collapsing or equalization.

Details

Item pairs can be detected either by column position or by suffix patterns. The function does not shift or rescale item scores; it only collapses and equalizes response levels within paired items.

Value

A list with components:

data The processed data frame, ordered as T1 items followed by T2 items.

pair_map A data frame describing matched T1/T2 item pairs.

collapsed_items A data frame listing item pairs that required collapsing/equalization.

before_tables Frequency tables before collapsing for affected item pairs.

after_tables Frequency tables after collapsing for affected item pairs.

mappings Category mappings for affected item pairs.

summary A data frame summarizing all item pairs.

Examples

```
dat <- data.frame(
  item1 = c(1, 1, 2, 2, 3, 3, 3, 4),
  item2 = c(1, 2, 2, 3, 3, 3, 4, 4),
  item1.1 = c(1, 2, 2, 2, 3, 4, 4, 4),
  item2.1 = c(1, 1, 2, 2, 3, 4, 4, 4)
)

out <- equalize_levels(
```

```
  data = dat,
  pair_by = "suffix",
  t1_suffix = "",
  t2_suffix = "\\.1",
  min_resp = 2,
  verbose = FALSE,
  print_tables = FALSE
)
```

`example` *Simulated longitudinal MIC example dataset*

Description

A simulated dataset containing responses to a 10-item patient-reported outcome measure at two time points for 1000 subjects, together with a binary transition rating anchor.

Usage

```
example
```

Format

A data frame with 1000 rows and 21 variables:

v1_1 Item 1 at Time 1.

v1_2 Item 2 at Time 1.

v1_3 Item 3 at Time 1.

v1_4 Item 4 at Time 1.

v1_5 Item 5 at Time 1.

v1_6 Item 6 at Time 1.

v1_7 Item 7 at Time 1.

v1_8 Item 8 at Time 1.

v1_9 Item 9 at Time 1.

v1_10 Item 10 at Time 1.

v2_1 Item 1 at Time 2.

v2_2 Item 2 at Time 2.

v2_3 Item 3 at Time 2.

v2_4 Item 4 at Time 2.

v2_5 Item 5 at Time 2.

v2_6 Item 6 at Time 2.

v2_7 Item 7 at Time 2.

v2_8 Item 8 at Time 2.

v2_9 Item 9 at Time 2.

v2_10 Item 10 at Time 2.

trat Binary transition rating anchor, coded 0/1.

Details

Each item has four ordered response categories scored 0, 1, 2, and 3. Therefore, the summed score at each time point ranges from 0 to 30.

The Time 1 items are named `v1_1` to `v1_10`, and the Time 2 items are named `v2_1` to `v2_10`. The variable `trat` is a dichotomous transition rating indicator coded 0/1.

This dataset is useful for demonstrating MIC estimation functions such as `mic_roc()`, `mic_pred()`, `mic_adjust()`, `mic_iapm()`, `tr_reliability()`, and `mic_lcfa()`.

Source

Simulated example data generated for demonstrating MIC package functions. See `data-raw/R/example.R`.

See Also

`simdat()` for generating new simulated datasets with user-specified simulation parameters.

`mic_roc()`, `mic_pred()`, `mic_adjust()`, and `mic_iapm()` for predictive modeling and adjusted predictive modeling MIC estimation.

`tr_reliability()` for estimating transition rating reliability.

`mic_lcfa()` for LCFA-based MIC estimation.

Examples

```
data(example)

nitems <- 10
example$score_t1 <- rowSums(example[, paste0("v1_", 1:nitems)])
example$score_t2 <- rowSums(example[, paste0("v2_", 1:nitems)])
example$change <- example$score_t2 - example$score_t1

head(example)

mic_roc(
  data = example,
  x = "score_t1",
  y = "score_t2",
  tr = "trat",
  nboot = 0
)
```

`lcfa_model` *Generate lavaan syntax for longitudinal CFA MIC estimation*

Description

`lcfa_model()` generates lavaan model syntax for estimating anchor-based minimal important change (MIC) using longitudinal confirmatory factor analysis.

Usage

```
lcfa_model(
  data,
  trt,
  pair_by = c("position", "suffix"),
  t1_suffix = NULL,
  t2_suffix = NULL,
  pair_map = NULL,
  factor_t1 = "F1",
  factor_t2 = "F2",
  loading_prefix = "a",
  threshold_prefix = "b",
  trt_loading_t1 = "f1",
  trt_loading_t2 = "f2",
  trt_threshold_label = "thr.trt",
  psb_label = "psb",
  mic_label = "b_param",
  correlated_errors = TRUE,
  threshold_invariance = TRUE,
  include_residual_variances_t2 = TRUE,
  include_factor_structure = TRUE,
  include_comments = TRUE,
  print_model = TRUE
)
```

Arguments

`data`	A data frame containing paired Time 1 and Time 2 PROM items and a transition rating variable.
`trt`	Character. Name of the transition rating variable.
`pair_by`	Pairing method. `"position"` assumes the first half of item columns are Time 1 items and the second half are Time 2 items. `"suffix"` detects item pairs using `t1_suffix` and `t2_suffix`.
`t1_suffix`	regex suffix identifying Time 1 items when `pair_by = "suffix"`. Use `""` when Time 1 items have no suffix.
`t2_suffix`	regex suffix identifying Time 2 items when `pair_by = "suffix"`.

`pair_map` Optional data frame describing item pairs, typically from `equalize_levels()$pair_map`. If supplied, it takes precedence over `pair_by`, `t1_suffix`, and `t2_suffix`.

`factor_t1` Character. Name of the Time 1 latent factor.

`factor_t2` Character. Name of the Time 2 latent factor.

`loading_prefix` Character. Prefix for equality-constrained item loading labels.

`threshold_prefix` Character. Prefix for equality-constrained item threshold labels.

`trt_loading_t1` Character. Label for the transition rating loading on the Time 1 factor.

`trt_loading_t2` Character. Label for the transition rating loading on the Time 2 factor.

`trt_threshold_label` Character. Label for the transition rating threshold.

`psb_label` Character. Name of the defined present-state-bias parameter.

`mic_label` Character. Name of the defined MIC parameter on the latent theta scale.

`correlated_errors` Logical. If `TRUE`, add correlated residuals between corresponding Time 1 and Time 2 items.

`threshold_invariance` Logical. If `TRUE`, constrain thresholds equal across Time 1 and Time 2 within each item pair.

`include_residual_variances_t2` Logical. If `TRUE`, frees residual variances of Time 2 items using `item_t2 ~~ NA*item_t2`.

`include_factor_structure` Logical. If `TRUE`, adds factor variances, covariance, and latent means/intercepts sections.

`include_comments` Logical. If `TRUE`, include section comments in the generated lavaan syntax.

`print_model` Logical. If `TRUE`, prints the generated lavaan syntax for easy inspection.

Details

The generated model is intended for ordinal PROM items and a binary transition rating item. Item thresholds are constrained equal across Time 1 and Time 2 within each item pair. The number of thresholds is determined automatically from the observed response levels in the supplied data.

Item pairs can be detected either by column position or by suffix patterns, using the same logic as `equalize_levels()`.

The model defines:

```
psb := (f1/f2) + 1
b_param := thr.trt/f2
```

Value

An object of class `lcfa_model`, invisibly. The generated lavaan syntax can be accessed using `$model`.

Examples

```
set.seed(123)
sim <- simdat(N = 100)
dat <- sim$datw

mod <- lcfa_model(
  data = dat[, c(sim$item_names$t1_items,
                 sim$item_names$t2_items,
                 "trat")],
  trt = "trat",
  pair_by = "suffix",
  t1_suffix = "",
  t2_suffix = "\\.1",
  print_model = FALSE
)

mod$model
```

`mic_adjust` *Predicted Minimal Important Change (MIC)*

Description

Minimal Important Change (MIC) obtained as predicted value from a logistic regression model.

Usage

```
mic_adjust(data = NULL, x, y, tr, reliability)
```

Arguments

`data`	data.frame that holds the scores at two time points and the transition rate in separate columns.
`x`	vector with score at time point 1, or column name in the data if `!is.null(data)`
`y`	vector with score at time point 1, or column name in the data if `!is.null(data)`
`tr`	vector with transition rates (perceived change), or column name in the data if `!is.null(data)`
`reliability`	the reliability for the transition score. Can be computed with the `tr_reliability()` function.

Details

`mic_adjust()` is a focused function for estimating the adjusted predictive modeling-based MIC. For an all-in-one workflow that returns the predictive MIC, adjusted predictive MIC, and improved adjusted predictive MIC with optional bootstrap confidence intervals, see `mic_iapm()`.

Value

vector with the adjusted MIC value

See Also

mic_iapm(), tr_reliability()

Examples

```
data(example)
nitems <- 10
example$x <- rowSums(example[,1:nitems])                # sumscore T1
example$y <- rowSums(example[,(nitems+1):(2*nitems)])   # sumscore T2
mic_adjust(x = example$x, y = example$y, tr = example$trat, reliability = 0.5)
mic_adjust(data = example, x = "x", y = "y", tr = "trat", reliability = 0.5)


# For predictive, adjusted, and improved adjusted MICs in one workflow,
# see mic_iapm().
```

---

mic_iapm *Estimate Predictive Modeling-Based MICs and Thresholds*

Description

Estimates (i) predictive modeling-based, (ii) adjusted predictive modeling-based, and (iii) improved adjusted predictive modeling-based minimal important change (MIC) estimates, with optional bootstrap confidence intervals. `mic_iapm()` can also be used to estimate the interpretation threshold of a predictor.

Usage

```
mic_iapm(
  mypred,
  anchor,
  mydata,
  anchor_reliability = NULL,
  nboot = 0,
  report_every = 100,
  verbose = FALSE,
  max_attempts = nboot * 5
)
```

Arguments

mypred	Character string; name of the column containing the change score or predictor score.
anchor	Character string; name of the column containing the binary anchor. The anchor must be binary and coded as 0/1 or TRUE/FALSE.
mydata	Data frame with the change score or predictor score and the anchor in separate columns.
anchor_reliability	Optional anchor reliability. Can be either a single numeric value between 0 and 1, or an object returned by `tr_reliability()`. If supplied, the improved adjusted predictive modeling-based MIC is also calculated.
nboot	Integer; number of bootstrap samples for estimating 95% confidence intervals. Bootstrapping is performed only when `nboot >= 100`.
report_every	Integer. The interval at which bootstrap progress should be printed.
verbose	Logical. If `TRUE`, progress messages are printed.
max_attempts	Integer; maximum number of bootstrap attempts. This avoids an infinite loop when many bootstrap samples fail.

Details

For reproducible bootstrap confidence intervals, call `set.seed()` before calling `mic_iapm()`.

Value

A `mic_iapm` object containing:

mic_pm Predictive modeling-based MIC.

mic_apm Adjusted predictive modeling-based MIC.

mic_iapm Improved adjusted predictive modeling-based MIC, returned only when `anchor_reliability` is supplied.

mic_pm_ci Bootstrap confidence interval for `mic_pm`, if requested.

mic_apm_ci Bootstrap confidence interval for `mic_apm`, if requested.

mic_iapm_ci Bootstrap confidence interval for `mic_iapm`, if requested and `anchor_reliability` is supplied.

mic_ci Matrix of available MIC estimates and confidence intervals.

anchor_reliability Anchor reliability used in the improved adjusted predictive modeling calculation.

nboot Requested number of bootstrap samples.

n_successful_boot Number of successful bootstrap samples.

References

Terluin B, Eekhout I, Terwee CB, de Vet HCW. Minimal important change (MIC) based on a predictive modeling approach was more precise than MIC based on receiver operating characteristic analysis. J Clin Epidemiol. 2015;68(12):1388-1396. doi:10.1016/j.jclinepi.2015.03.015

Terluin B, Eekhout I, Terwee CB. The anchor-based minimal important change, based on receiver operating characteristic analysis or predictive modeling, may need to be adjusted for the proportion of improved patients. J Clin Epidemiol. 2017;83:90-100. doi:10.1016/j.jclinepi.2016.12.015

Terluin B, Eekhout I, Terwee CB. Improved adjusted minimal important change took reliability of transition ratings into account. J Clin Epidemiol. 2022;148:48-53. doi:10.1016/j.jclinepi.2022.04.018

See Also

`tr_reliability()`

Examples

```
set.seed(123)
sim <- simdat(N = 200, add_change = TRUE)
dat <- sim$datw

mic_iapm(
  mypred = "change",
  anchor = "trat",
  mydata = dat,
  anchor_reliability = sim$truth$observed_rel_trt,
  nboot = 200
)
```

---

mic_lcfa *Estimate Present-State Bias and Anchor-Based MIC Using Longitudinal Confirmatory Factor Analysis*

Description

`mic_lcfa()` estimates present-state bias and anchor-based minimal important change (MIC) using a longitudinal confirmatory factor analysis (LCFA) model.

Usage

```
mic_lcfa(
  mydat,
  trt = NULL,
  model = NULL,
  trt_cut = 1,
  auto_equalize = TRUE,
  pair_by = c("position", "suffix"),
```

```
  t1_suffix = NULL,
  t2_suffix = NULL,
  min_resp = 5L,
  B = 0L,
  report_every = 100L,
  N_ets = 5000L,
  score_method = c("irt", "cfa", "both"),
  return_prepared = FALSE,
  print_model = TRUE,
  verbose = TRUE
)
```

Arguments

mydat	Data frame containing paired Time 1 and Time 2 PROM items and one transition rating variable.
trt	Character. Name of the transition rating variable. If NULL, the final column of mydat is used.
model	Optional lavaan model syntax. If NULL, the model is generated automatically using lcfa_model().
trt_cut	Numeric. Cutpoint used to dichotomize the transition rating variable when it has more than two observed values. Values greater than or equal to trt_cut are coded as 1, and lower values are coded as 0. If the transition rating is already binary, the larger value is coded as 1 and trt_cut is ignored.
auto_equalize	Logical. If TRUE, item levels are equalized using equalize_levels().
pair_by	Character. Pairing method passed to equalize_levels() and lcfa_model(). "position" assumes Time 1 items followed by Time 2 items. "suffix" detects pairs using t1_suffix and t2_suffix.
t1_suffix	Regex suffix identifying Time 1 items when pair_by = "suffix".
t2_suffix	Regex suffix identifying Time 2 items when pair_by = "suffix".
min_resp	Integer. Minimum number of responses required in each response category before collapsing.
B	Integer. Number of bootstrap samples. Bootstrap confidence intervals are computed only if B >= 100.
report_every	Integer. Progress message interval during bootstrapping.
N_ets	Integer. Number of simulated theta values used for the IRT expected test score calculation. This argument is not used by the closed-form CFA method.
score_method	Character. Method used to map the latent MIC to the observed PROM score metric. Options are "irt", "cfa", and "both". "irt" uses mirt::expected.test(), "cfa" uses the closed-form CFA/probit expected-score method, and "both" returns both mappings.
return_prepared	Logical. If TRUE, returns prepared data, generated model, fitted LCFA object, and bootstrap details.
print_model	Logical. If TRUE, prints the generated lavaan model.
verbose	Logical. If TRUE, prints progress messages.

Details

The input data should contain paired Time 1 and Time 2 PROM items plus one transition rating variable. Item pairs may be identified by column position or by suffixes. By default, paired item levels are first equalized using `equalize_levels()`, and lavaan syntax is generated using `lcfa_model()`.

The LCFA model estimates the MIC on the latent-change scale. This latent MIC is then mapped onto the observed PROM summed-score metric using an expected score function.

The LCFA model uses both Time 1 and Time 2 item responses, together with the transition rating, to estimate the latent MIC and present-state bias. The latent MIC is then mapped onto the observed PROM summed-score metric.

1. at the baseline latent trait distribution, `theta ~ N(0, 1)`;
2. at the shifted latent trait distribution, `theta ~ N(MIC.theta, 1)`.

Two expected-score mappings are available through `score_method`:

- `"irt"` uses an IRT-based expected test score method via `mirt::expected.test()`. This follows the published implementation of Terluin et al. (2024).
- `"cfa"` uses a closed-form CFA probit expected-score method based on lavaan-estimated item loadings and thresholds. For each item, the method computes marginal category probabilities under `theta ~ N(mu, 1)`, first with `mu = 0` and then with `mu = MIC.theta`. These probabilities are used to obtain expected item scores, which are summed across items to obtain the expected PROM score.
- `"both"` computes expected scores using both irt and cfa methods

Under the CFA probit method, the closed-form marginal cumulative probability for item threshold `tau_k` is based on the latent response model `Y* = lambda * theta + error`. If `theta ~ N(mu, 1)` and `error ~ N(0, 1)`, then `Y* ~ N(lambda * mu, lambda^2 + 1)`. Therefore, marginal category probabilities can be computed without simulating theta values or fitting an additional IRT model.

Value

A `mic_lcfa` object with elements including:

- `psb`: estimated present-state bias;
- `MIC.theta`: MIC on the latent-change scale;
- `MIC.ets`: MIC on the observed PROM summed-score metric;
- `MIC.ets.irt`: IRT-based expected-score MIC, if requested;
- `MIC.ets.cfa`: CFA-based expected-score MIC, if requested;
- `MIC_CI`: bootstrap confidence interval, if requested;
- `nboot`: number of successful bootstrap estimates.

If `return_prepared = TRUE`, the returned object also contains the prepared data, generated lavaan model, fitted LCFA object, and bootstrap values.

References

Terluin B, Trigg A, Fromy P, Schuller W, Terwee CB, Bjorner JB. Estimating anchor-based minimal important change using longitudinal confirmatory factor analysis. Qual Life Res. 2024;33:963-973. doi:10.1007/s11136-023-03577-w

Terluin B, Fromy P, Trigg A, Terwee CB, Bjorner JB. Effect of present state bias on minimal important change estimates: a simulation study. Quality of Life Research. 2024. doi:10.1007/s11136-024-03763-4

Terluin B, Griffiths P, Trigg A, Terwee CB, Bjorner JB. Present state bias in transition ratings was accurately estimated in simulated and real data. J Clin Epidemiol. 2022;143:128-136. doi:10.1016/j.jclinepi.2021.12.024

See Also

lcfa_model(), equalize_levels(), simdat()

Examples

```
set.seed(123)
sim <- simdat(N = 500, add_change = TRUE)
dat <- sim$datw

mydat <- dat[, c(
  sim$item_names$t1_items,
  sim$item_names$t2_items,
  "trat"
)]


out_both <- mic_lcfa(
  mydat = mydat,
  trt = "trat",
  trt_cut = 1,
  auto_equalize = TRUE,
  pair_by = "suffix",
  t1_suffix = "",
  t2_suffix = "\\.1",
  min_resp = 5,
  score_method = "both",
  B = 0,
  print_model = FALSE,
  verbose = FALSE
)

out_both$MIC.ets
```

`mic_pred` *Predicted Minimal Important Change (MIC)*

Description

Minimal Important Change (MIC) obtained as predicted value from a logistic regression model.

Usage

```
mic_pred(data = NULL, x, y, tr)
```

Arguments

`data`	data.frame that holds the scores at two time points and the transition rate in separate columns.
`x`	vector with score at time point 1, or column name in the data if `!is.null(data)`
`y`	vector with score at time point 1, or column name in the data if `!is.null(data)`
`tr`	vector with transition rates (perceived change), or column name in the data if `!is.null(data)`

Details

`mic_pred()` is a focused function for estimating the predictive modeling-based MIC. For an all-in-one workflow that returns the predictive MIC, adjusted predictive MIC, and improved adjusted predictive MIC with optional bootstrap confidence intervals, see `mic_iapm()`.

Value

vector with the predicted MIC value

See Also

`mic_iapm()`

Examples

```
data(example)
nitems <- 10
example$x <- rowSums(example[,1:nitems])                # sumscore T1
example$y <- rowSums(example[,(nitems+1):(2*nitems)])   # sumscore T2
mic_pred(x = example$x, y = example$y, tr = example$trat)
mic_pred(data = example, x = example$x, y = example$y, tr = example$trat)
mic_pred(data = example, x = "x", y = "y", tr = "trat")

# For predictive, adjusted, and improved adjusted MICs in one workflow,
# see mic_iapm().
```

`mic_roc` *ROC-Based Minimal Important Change*

Description

Estimates the minimal important change (MIC) using receiver operating characteristic (ROC) analysis. The MIC is estimated as the change-score threshold that optimizes the Youden criterion.

Usage

```
mic_roc(
  data = NULL,
  x,
  y,
  tr,
  nboot = 0,
  report_every = 100,
  verbose = FALSE,
  max_attempts = nboot * 5
)
```

Arguments

`data`	Optional data frame containing the Time 1 score, Time 2 score, and binary anchor.
`x`	Numeric vector of scores at Time 1, or character name of the Time 1 score column in `data`.
`y`	Numeric vector of scores at Time 2, or character name of the Time 2 score column in `data`.
`tr`	Binary anchor vector, or character name of the anchor column in `data`. The anchor must be coded as 0/1 or TRUE/FALSE.
`nboot`	Integer. Number of bootstrap samples for estimating the 95% confidence interval. Bootstrapping is performed only when `nboot >= 100`.
`report_every`	Integer. The interval at which bootstrap progress should be printed.
`verbose`	Logical. If `TRUE`, progress messages are printed.
`max_attempts`	Integer. Maximum number of bootstrap attempts. This avoids an infinite loop when many bootstrap samples fail.

Details

Optional bootstrap confidence intervals can be requested by setting `nboot >= 100`.

For reproducible bootstrap confidence intervals, call `set.seed()` before calling `mic_roc()`.

Value

A `mic_roc` object containing the ROC-based MIC and, optionally, bootstrap confidence intervals.

Examples

```
data(example)

nitems <- 10

example$score_t1 <- rowSums(example[, paste0("v1_", seq_len(nitems))])
example$score_t2 <- rowSums(example[, paste0("v2_", seq_len(nitems))])

mic_roc(
  data = example,
  x = "score_t1",
  y = "score_t2",
  tr = "trat",
  nboot = 0
)


set.seed(123)

mic_roc(
  data = example,
  x = "score_t1",
  y = "score_t2",
  tr = "trat",
  nboot = 500
)
```

`mim_threshold` *Estimate a CFA-based threshold for a multi-item questionnaire*

Description

`mim_threshold()` estimates an anchor-based interpretation threshold for a multi-item measure (MIM) using a one-factor CFA model with an anchor/transition rating item.

Usage

```
mim_threshold(
  mydata,
  var_formula,
  item_discretize = FALSE,
  item_levels = 10L,
  require_all_item_levels = FALSE,
  anchor_cut = 1,
  B = 0L,
  report_every = 50L,
  factor_name = "F1",
```

```
  item_suffix = "_ord",
  anchor_suffix = "_bin",
  std.lv = TRUE,
  parameterization = "theta",
  verbose = TRUE,
  ...
)
```

Arguments

mydata	Data frame.
var_formula	Formula of the form anchor ~ item1 + item2 + ... Use anchor ~ . to use all variables except the anchor as items.
item_discretize	Logical. If TRUE, discretize each item using var_discretize(). If FALSE, use the item values as observed ordinal categories.
item_levels	Number of levels to use when item_discretize = TRUE. Must be between 2 and 12.
require_all_item_levels	Logical. Passed to var_discretize().
anchor_cut	Cutpoint for binarizing the anchor if it has more than two unique non-missing values. Values >= anchor_cut are coded 1.
B	Number of bootstrap samples. Bootstrap CI is computed only if B >= 100.
report_every	During bootstrapping, print progress every report_every attempted fits.
factor_name	Name of the latent factor.
item_suffix	Suffix appended to item names when items are discretized or internally recoded.
anchor_suffix	Suffix appended to the anchor variable name if the anchor has more than two unique values and is dichotomized.
std.lv	Passed to lavaan::cfa().
parameterization	Passed to lavaan::cfa().
verbose	If TRUE, print item names and generated lavaan model.
...	Additional arguments passed to lavaan::cfa().

Details

This technique is based on Terluin et al (2023) and Terluin et al (2024). Briefly, the mim_threshold() function:

1. fits a one-factor CFA model with the MIM items and an anchor item,
2. computes theta_star, the latent factor value where the anchor, threshold occurs,
3. maps theta_star to each item using CFA-implied category probabilities,
4. multiplies item category probabilities by item values or bin midpoints, and
5. sums the item-level expected values to obtain the MIM threshold.

Value

A mim_threshold object. Additional details can be retrieved with mim_threshold_details().

References

Terluin, B., Koopman, J.E., Hoogendam, L. et al. Estimating meaningful thresholds for multi-item questionnaires using item response theory. Qual Life Res 32, 1819–1830 (2023)

Terluin, B., Trigg, A., Fromy, P. et al. Estimating anchor-based minimal important change using longitudinal confirmatory factor analysis. Qual Life Res 33, 963–973 (2024).

Examples

```
set.seed(123)
sim <- simdat(N = 500)
dat <- sim$datw
t1_items <- sim$item_names$t1_items

dat_t1 <- dat[, c(t1_items, "trat")]

out <- mim_threshold(
  mydata = dat_t1,
  var_formula = trat ~ .,
  B = 0
)

out
```

---

mim_threshold_details *Extract hidden details from a mim_threshold object*

Description

Extract hidden details from a mim_threshold object

Usage

```
mim_threshold_details(x)
```

Arguments

x	A mim_threshold object.

Value

A list containing hidden details.

simdat *Simulate Longitudinal PROM Data and a Binary Anchor*

Description

simdat() simulates Time 1 and Time 2 item responses for a 10-item patient-reported outcome measure (PROM) and a binary anchor / transition rating from an item response theory context. Each item has four ordered response categories scored 0, 1, 2, and 3, so the total PROM score ranges from 0 to 30 at each time point.

Usage

```
simdat(
  N = 2000,
  mn_imic = 0.37425,
  sd_imic = 0.05,
  cor_t1_change = -0.5,
  mean_tetch = 0.3,
  sd_tetch = 1,
  rel_trt = 0.7,
  return_latent = TRUE,
  add_change = FALSE
)
```

Arguments

N	Integer. Sample size for simulation.
mn_imic	Numeric. Mean individual minimal important change (MIC) on the latent theta-change scale. For context, mn_imic = 0.5 corresponds approximately to a raw-score MIC of about 2.8 points, while mn_imic = 0.37425 corresponds approximately to a raw-score MIC of about 2.5 points on the 0-30 PROM scale.
sd_imic	Numeric. Standard deviation of individual MICs on the latent theta-change scale.
cor_t1_change	Numeric. Correlation between baseline theta and latent change.
mean_tetch	Numeric. Mean latent change.
sd_tetch	Numeric. Standard deviation of latent change.
rel_trt	Numeric. Target reliability of perceived change used to generate the binary anchor / transition rating.
return_latent	Logical. If TRUE, returns latent variables and item parameters in the output object.
add_change	Logical. If TRUE, adds change = score_t2 - score_t1 to the returned data frame. Defaults to FALSE.

Details

The returned data frame includes item-level responses, the binary anchor `trat`, the Time 1 summed PROM score `score_t1`, and the Time 2 summed PROM score `score_t2`. If `add_change = TRUE`, the observed change score `change = score_t2 - score_t1` is also added.

For reproducible simulations, call `set.seed()` before calling `simdat()`.

The R code is adapted from supplementary materials of Terluin et al. Qual Life Res. 2024;33:963-973.

Value

A list containing:

settings Simulation settings.

item_names Names of Time 1 items, Time 2 items, and anchor.

truth Truth / diagnostic quantities, including `target_rel_trt` and `observed_rel_trt`.

datw The simulated wide-format data frame.

If `return_latent = TRUE`, the output also includes item parameters, latent variables, perceived change, and individual MICs.

Examples

```
set.seed(123)
sim <- simdat(N = 200, add_change = TRUE)

names(sim$datw)
sim$truth
```

---

sim_mic_lcfa *Longitudinal confirmatory factor analysis-based MIC for a Single-Item Measure*

Description

`sim_mic_lcfa()` estimates the minimal important change (MIC) for a single-item measure (SIM) using longitudinal confirmatory factor analysis (LCFA) with an auxiliary variable. The SIM is measured at Time 1 and Time 2, an auxiliary variable is measured at Time 1 and Time 2, and a transition rating is included as an anchor.

Usage

```
sim_mic_lcfa(
  mydata,
  sim,
  aux,
  trt,
  trt_cut = 1,
  sim_levels = 10L,
  sim_discretize = c("auto", "yes", "no"),
  min_resp = 1L,
  aux_ordered = FALSE,
  B = 0L,
  report_every = 50L,
  add_lmodel = NULL,
  print_model = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

mydata	Data frame.
sim	Character vector of length 2. Names of the SIM variable at Time 1 and Time 2.
aux	Character vector of length 2. Names of the auxiliary variable at Time 1 and Time 2.
trt	Character. Name of the transition rating / anchor variable.
trt_cut	Numeric. Cutpoint used to binarize trt if it has more than two observed values. Values greater than or equal to trt_cut are coded 1. If trt already has exactly two observed values, the larger value is coded as 1 and trt_cut is ignored.
sim_levels	Integer. Number of ordered levels used when discretizing the SIM with var_discretize(). Must be between 2 and 12. Default is 10.
sim_discretize	Character. One of "auto", "yes", or "no". "auto" applies var_discretize() only when the pooled SIM has more than 12 observed levels. "yes" always applies var_discretize(). "no" never discretizes and errors if the pooled SIM has more than 12 observed levels.
min_resp	Integer. Minimum response count per category used by equalize_levels() when equalizing SIM response levels across time.
aux_ordered	Logical. If TRUE, treats the auxiliary variables as ordered indicators in lavaan. Auxiliary variable loadings and thresholds are freely estimated over time.
B	Integer. Number of nonparametric bootstrap samples. Bootstrap confidence intervals are computed only when B >= 100.For reproducible simulations or bootstrap confidence intervals, call set.seed() before calling sim_mic_lcfa().
report_every	Integer. Print bootstrap progress every report_every attempted bootstrap fits.
add_lmodel	Optional lavaan syntax appended to the generated model.
print_model	Logical. If TRUE, prints the generated lavaan model.

verbose	Logical. If TRUE, prints progress messages.
...	Additional arguments passed to lavaan::cfa().

Details

If the pooled SIM values across Time 1 and Time 2 have 12 or fewer observed levels, the original SIM values are used as ordered categories. If the pooled SIM values have more than 12 observed levels, the SIM is discretized using var_discretize() with common equal-width bins across Time 1 and Time 2. In both cases, SIM response levels are equalized across time before fitting the LCFA model.

The SIM loading and SIM thresholds are constrained equal over time so that the Time 1 and Time 2 latent factors are on the same measurement scale.

The latent MIC is estimated as MIC.theta = tau_trt / f2, where tau_trt is the transition-rating threshold and f2 is the transition-rating loading on the Time 2 factor.

The SIM-scale MIC is first calculated on the transformed SIM scale as sqrt(var(SIM_T1_transformed) * Rel_SIM_T1) * MIC.theta, where Rel_SIM_T1 is the model R-squared of the Time 1 SIM. If var_discretize() is used, the transformed-scale MIC is then back-transformed to the original SIM metric using the equal-width bin width.

Continuous SIMs with many distinct values are not used directly as ordered indicators. When discretization is needed, sim_mic_lcfa() applies var_discretize() to the pooled Time 1 and Time 2 SIM values. This ensures that the same equal-width binning rule is applied at both time points.

The LCFA model is fitted to the discretized/equalized ordered SIM variables. Therefore, the reliability of the SIM, Rel_SIM_T1, is the reliability of the transformed SIM used in the CFA model. For this reason, the SIM MIC is first computed on the transformed SIM scale:

MIC_SIM_transformed = sqrt(var(SIM_T1_transformed) * Rel_SIM_T1) * MIC.theta.

If discretization was used, this transformed-scale MIC is then converted back to the original SIM metric using:

MIC_SIM_original = MIC_SIM_transformed * backtransform_factor,

where backtransform_factor is the equal-width bin width, calculated as the pooled original SIM range divided by the number of requested discretized levels. If no discretization is used, backtransform_factor = 1.

Value

A sim_mic_lcfa object with elements including:

- MIC.theta: MIC on the latent-change scale;
- MIC.sim.transformed: MIC on the transformed SIM scale;
- MIC.sim: MIC on the original SIM metric;
- rel_SIM1: model R-squared / reliability of SIM at Time 1;
- rel_trt: model R-squared / reliability of the transition rating;
- psb: estimated present-state bias;
- ci: bootstrap confidence intervals, if requested;
- sim_prepared: information about discretization, equalization, and back-transformation.

References

Terluin B, Pua YH, Fromy P, Trigg A, van der Zwaard B, Bjorner JB. Estimating the minimal important change of single-item measures using the adjusted predictive modeling method or the longitudinal confirmatory factor analysis method. Quality of Life Research. 2026. doi:10.1007/s11136-025-04134-3

Terluin B, Trigg A, Fromy P, Schuller W, Terwee CB, Bjorner JB. Estimating anchor-based minimal important change using longitudinal confirmatory factor analysis. Qual Life Res. 2024;33:963-973. doi:10.1007/s11136-023-03577-w

See Also

`var_discretize()`, `equalize_levels()`, `mic_lcfa()`, `simdat()`

Examples

```
set.seed(123)
sim <- simdat(N = 500, add_change = TRUE)
dat <- sim$datw

# Create a toy single-item measure from several items
dat$sim_t1 <- rowSums(dat[, paste0("item", 1:8), drop = FALSE])
dat$sim_t2 <- rowSums(dat[, paste0("item", 1:8, ".1"), drop = FALSE])

out <- sim_mic_lcfa(
  mydata = dat,
  sim = c("sim_t1", "sim_t2"),
  aux = c("item9", "item9.1"),
  trt = "trat",
  sim_discretize = "auto",
  sim_levels = 10,
  aux_ordered = TRUE,
  B = 0,
  print_model = TRUE
)

out

# Inspect how the SIM was prepared
out$sim_prepared$used_discretization
out$sim_prepared$backtransform_factor
out$sim_prepared$original_levels
```

`sim_threshold` *Estimate a Confirmatory Factor Analysis-based interpretation threshold for a single-item measure*

Description

sim_threshold() estimates an interpretation threshold for a continuous or ordinal single-item measure using a confirmatory factor analysis (CFA) approach developed by Terluin et al (2026)

Usage

```
sim_threshold(
  mydata,
  var_formula,
  sim_levels = 10L,
  ordered = NULL,
  add_lmodel = NULL,
  tr_cut = 1,
  B = 0L,
  report_every = 50L,
  require_all_sim_levels = FALSE,
  factor_name = "F1",
  sim_suffix = "_ord",
  std.lv = TRUE,
  parameterization = "theta",
  verbose = FALSE,
  ...
)
```

Arguments

mydata	Data frame.
var_formula	Formula of the form tr ~ sim + aux1 + aux2 + ... The left-hand side is the transition or anchor variable. The first right-hand side variable is the continuous SIM. Remaining right-hand side variables are auxiliary CFA indicators.
sim_levels	Number of equal-width levels for discretizing the SIM. Must be between 2 and 12.
ordered	Additional auxiliary variables to treat as ordered in lavaan.
add_lmodel	Optional lavaan syntax appended to the generated model.
tr_cut	Cutpoint for binarizing the transition variable when it has more than two unique non-missing values. Values >= tr_cut are coded 1. If the transition variable already has exactly two unique values, tr_cut is ignored.
B	Number of bootstrap samples. Bootstrap CI is computed only if B >= 100.
report_every	During bootstrapping, print progress every report_every attempted fits.
require_all_sim_levels	Passed to var_discretize().
factor_name	Name of the latent factor.
sim_suffix	Suffix appended to the SIM variable name after discretization.
std.lv	Passed to lavaan::cfa().

parameterization
: Passed to `lavaan::cfa()`.

verbose
: If `TRUE`, print generated lavaan model.

...
: Additional arguments passed to `lavaan::cfa()`.

Details

The `sim_threshold()` function:

1. discretizes a continuous SIM into equal-width ordered categories;
2. fits a one-factor CFA model with an anchor transition item;
3. computes `theta_star`, the latent factor value where the anchor threshold occurs;
4. maps `theta_star` back to the original SIM scale using CFA-implied category probabilities and bin midpoints.

Value

A `sim_threshold` object. The printed output is compact. Additional details can be retrieved with `sim_threshold_details()`.

References

Terluin B, Pua YH, Fromy P, Trigg A, van der Zwaard B, Bjorner JB. Estimating the minimal important change of single-item measures using the adjusted predictive modeling method or the longitudinal confirmatory factor analysis method. Quality of Life Research. 2026. doi:10.1007/s11136-025-04134-3

Examples

```
set.seed(123)
sim <- simdat(N = 500)
dat <- sim$datw
t1_items <- sim$item_names$t1_items

dat_t1 <- dat[, c(t1_items, "trat")]

dat_sim <- dat_t1
dat_sim$sim8 <- rowSums(dat_sim[, t1_items[1:8], drop = FALSE])
dat_sim$aux9 <- dat_sim[[t1_items[9]]]
dat_sim$aux10 <- dat_sim[[t1_items[10]]]
dat_sim <- dat_sim[, c("sim8", "aux9", "aux10", "trat")]

out <- sim_threshold(
  mydata = dat_sim,
  var_formula = trat ~ sim8 + aux9 + aux10,
  sim_levels = 10,
  ordered = c("aux9", "aux10"),
  B = 0
)
```

```
out
```

`sim_threshold_details` *Extract details from a SIM threshold object*

Description

`sim_threshold_details()` extracts additional details from an object returned by `sim_threshold()`, including the fitted lavaan model, CFA data, latent anchor location, model-implied category probabilities, and bootstrap results.

Usage

```
sim_threshold_details(x)
```

Arguments

`x`	A `sim_threshold` object.

Value

A list of additional model details.

`tr_reliability` *Estimate anchor reliability using CFA*

Description

`tr_reliability()` estimates the reliability of an anchor or transition rating item using confirmatory factor analysis. The reliability estimate is the R-squared value of the anchor item from the fitted CFA model.

Usage

```
tr_reliability(
  data,
  model = NULL,
  anchor = NULL,
  xsec = FALSE,
  pair_by = c("position", "suffix"),
  t1_suffix = NULL,
  t2_suffix = NULL,
  factor_names = NULL,
  item_type = NULL,
```

```
  continuous_items = NULL,
  modification = TRUE,
  mi_cut = 0.3,
  complete_cases = TRUE,
  print_model = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

data	A data frame containing items and an anchor variable.
model	Optional lavaan model syntax. If NULL, syntax is generated automatically using tr_reliability_model().
anchor	Character. Name of the anchor variable. If NULL, the final column of data is treated as the anchor.
xsec	Logical. If TRUE, estimate cross-sectional anchor reliability. If FALSE, estimate longitudinal anchor / transition-rating reliability.
pair_by	Pairing method for longitudinal data. "position" assumes Time 1 items followed by Time 2 items. "suffix" detects pairs using t1_suffix and t2_suffix.
t1_suffix	Regex suffix identifying Time 1 items when pair_by = "suffix".
t2_suffix	Regex suffix identifying Time 2 items when pair_by = "suffix".
factor_names	Optional character vector of factor name(s). If NULL, defaults to "F1" when xsec = TRUE, and c("F1", "F2") when xsec = FALSE.
item_type	Optional character. Type of items used as indicators. If NULL, defaults to "ordinal" unless continuous_items is supplied, in which case it defaults to "mixed". "ordinal" treats all items and the anchor as ordered categorical variables. "continuous" treats all items as continuous while still treating the anchor as ordered. "mixed" treats all items as ordered except those listed in continuous_items; the anchor is always treated as ordered.
continuous_items	Optional character vector of item names to treat as continuous. If supplied and item_type = NULL, item_type is automatically set to "mixed". Do not include the anchor variable; it is always treated as ordered.
modification	Logical. If TRUE, return modification indices with sepc.lv > mi_cut.
mi_cut	Numeric. Cutoff for standardized latent-variable modification indices.
complete_cases	Logical. If TRUE, retain only complete cases before fitting the model.
print_model	Logical. If TRUE, print the generated lavaan syntax.
verbose	Logical. If TRUE, print progress messages.
...	Additional arguments passed to lavaan::cfa().

Details

This function follows the CFA approach for estimating transition-rating reliability described by Griffiths et al. (2022). For longitudinal data, Time 1 items load on the first factor, Time 2 items load on the second factor, and the anchor loads on both factors. Residuals of corresponding items across time-points are allowed to correlate. No constraints are placed on loadings or thresholds.

For cross-sectional data, a one-factor CFA model is used, with the anchor item included as an indicator of the latent factor.

Value

An object of class `tr_reliability`.

References

Griffiths P, Terluin B, Trigg A, Schuller W, Bjorner JB. A confirmatory factor analysis approach was found to accurately estimate the reliability of transition ratings. J Clin Epidemiol. 2022;141:36-45. doi:10.1016/j.jclinepi.2021.08.029

See Also

`tr_reliability_model()`

Examples

```
set.seed(123)
sim <- simdat(N = 300)
dat <- sim$datw

rel <- tr_reliability(
  data = dat[, c(sim$item_names$t1_items,
                 sim$item_names$t2_items,
                 "trat")],
  anchor = "trat",
  xsec = FALSE,
  pair_by = "suffix",
  t1_suffix = "",
  t2_suffix = "\\.1",
  item_type = "ordinal",
  modification = FALSE,
  print_model = FALSE
)

rel
```

`tr_reliability_model` *Generate CFA syntax for anchor reliability*

Description

`tr_reliability_model()` generates lavaan syntax for estimating the reliability of an anchor or transition rating item using confirmatory factor analysis.

Usage

```
tr_reliability_model(
  data,
  anchor = NULL,
  xsec = FALSE,
  pair_by = c("position", "suffix"),
  t1_suffix = NULL,
  t2_suffix = NULL,
  factor_names = NULL,
  print_model = TRUE
)
```

Arguments

`data`	A data frame containing items and an anchor variable.
`anchor`	Character. Name of the anchor variable. If `NULL`, the final column of `data` is treated as the anchor.
`xsec`	Logical. If `TRUE`, generate a cross-sectional one-factor model. If `FALSE`, generate a longitudinal two-factor model.
`pair_by`	Pairing method for longitudinal data. `"position"` assumes Time 1 items followed by Time 2 items. `"suffix"` detects pairs using `t1_suffix` and `t2_suffix`.
`t1_suffix`	Regex suffix identifying Time 1 items when `pair_by = "suffix"`. Use `""` when Time 1 items have no suffix.
`t2_suffix`	Regex suffix identifying Time 2 items when `pair_by = "suffix"`.
`factor_names`	Optional character vector of factor name(s). If `NULL`, defaults to `"F1"` when `xsec = TRUE`, and `c("F1", "F2")` when `xsec = FALSE`.
`print_model`	Logical. If `TRUE`, print the generated lavaan syntax.

Details

The generated model follows the approach described by Griffiths et al. J Clin Epidemiol. 2022;141:36-45. No constraints are placed on loadings or thresholds. For longitudinal data, residuals of corresponding items across time-points are allowed to correlate to account for item non-independence.

For cross-sectional data, a one-factor model is generated. For longitudinal data, a two-factor model is generated, with the anchor loading on both Time 1 and Time 2 factors.

Value

An object of class `tr_reliability_model`. The lavaan syntax can be accessed using `$model`.

Examples

```
set.seed(123)
sim <- simdat(N = 200)
dat <- sim$datw

tr_reliability_model(
  data = dat[, c(sim$item_names$t1_items,
                 sim$item_names$t2_items,
                 "trat")],
  anchor = "trat",
  xsec = FALSE,
  pair_by = "suffix",
  t1_suffix = "",
  t2_suffix = "\\.1"
)
```

---

var_discretize *Discretize a continuous variable into equal-width ordered categories*

Description

`var_discretize()` converts a continuous numeric vector into equal-width integer categories. It is intended for creating ordered indicators for lavaan-based CFA workflows. To be used with `sim_threshold_cfa`

Usage

```
var_discretize(
  x,
  n_levels = 10L,
  min_level = 1L,
  require_all_levels = FALSE,
  warn_unused = TRUE
)
```

Arguments

`x`	Numeric vector to discretize.
`n_levels`	Integer. Number of equal-width categories to create. Must be between 2 and 12.
`min_level`	Integer. Lowest score value. Defaults to `1L`.
`require_all_levels`	Logical. If `TRUE`, stop when the discretized variable does not use all requested levels.
`warn_unused`	Logical. If `TRUE`, warn when some levels are unused. Ignored when `require_all_levels = TRUE`.

Details

Missing values are allowed. They are ignored when computing the discretization range and are preserved as `NA_integer_` in the returned score vector.

Value

A list containing the discretized score, bin midpoints, bin edges, used levels, and unused levels.

Examples

```
x <- 0:24

out <- var_discretize(x, n_levels = 10)

out$score
out$midpoints
```

Index