Package ‘tirt’

August 20, 2026

Title Testlet Item Response Theory

Version 0.4.0

Description Implementation of Testlet and Item Response Theory.
A light-version yet comprehensive and streamlined framework for psychometric analysis using unidimensional and multidimensional Item Response Theory (IRT; Baker & Kim (2004) <doi:10.1201/9781482276725>) and Testlet Response Theory (TRT; Wainer et al., (2007) <doi:10.1017/CBO9780511618765>).
Designed for researchers, this package supports the estimation of item and person parameters for a wide variety of models, including binary (i.e., Rasch, 2-Parameter Logistic, 3-Parameter Logistic) and polytomous (Partial Credit Model, Generalized Partial Credit Model, Graded Response Model) formats. It also supports the estimation of Testlet models (Rasch Testlet, 2-Parameter Logistic Testlet, 3-Parameter Logistic Testlet, Bifactor, Partial Credit Model Testlet, Graded Response), allowing users to account for local item dependence in bundled items. A key feature is the specialized support for combination use and joint estimation of item response model and testlet response model in one calibration.
Beyond standard estimation via Marginal Maximum Likelihood with Expectation-Maximization (EM) or Joint Maximum Likelihood, the package also offers Bayesian estimation using priors with maximum a posteriori (MAP) method for unidimensional item response theory models.
It also provides functions for scale linking and equating (Mean-Mean, Mean-Sigma, Stocking-Lord) to ensure comparability across mixed-format test forms. It also facilitates fixed-parameter calibration, enabling users to estimate person abilities with known item parameters or vice versa, which is essential for pre-equating studies and item bank maintenance. Comprehensive data simulation functions are included to generate synthetic datasets with complex structures, including mixed-model blocks and specific testlet effects, aiding in methodological research and study design validation. Researchers can try multiple simulation situations.
A suite of post-estimation tools is also provided, including item and test information functions with the conditional standard error of measurement, summed-score to scale-score conversion tables (expected a posteriori, weighted likelihood, and maximum likelihood), person-fit and item-fit statistics, local dependence diagnostics (Yen's Q3), differential item functioning (M-H and logistic regression), reliability coefficients, test characteristic curves, and mixture (latent-class) item response models.

License GPL-3

Encoding UTF-8

LazyData true

LazyDataCompression xz

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports dplyr, tidyr, purrr, gtools, stats, utils

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Jiawei Xiong [aut, cre],
Cheng Tang [ctb],
Qidi Liu [ctb]

Maintainer Jiawei Xiong <jiawei.xiong@uga.edu>

Repository CRAN

Date/Publication 2026-08-20 08:10:02 UTC

Contents

binary_irt 3
dif 5
ela1 6
ela2 7
ela3 8
ela3_testmap 8
equate_irt 9
fixed_item 12
fix_person 14
irt_trt 16
item_fit 19
item_info 20
ld_stats 22
mirt_binary 23
mixed_irt 25
mixture_irt 28
person_fit 29
polytomous_irt 31
reliability 34
score_table 35
sim_irt 37
sim_mirt 38
sim_tirt 40
sim_trt 42
tcc 44
test_info 45

trt_binary 46
trt_poly 49

Index **52**

`binary_irt` *Binary (Dichotomous) Item Response Theory Estimation Using Likelihood or Bayesian*

Description

Estimates item and person parameters for binary item response models using either Marginal Maximum Likelihood or Joint Maximum Likelihood. Now supports flexible prior distributions for Bayesian estimation (MAP estimation).

Usage

```
binary_irt(data, model = "2PL", method = "EM", control = list())
```

Arguments

`data`	A N x J data.frame of dichotomous responses (0/1).
`model`	String. "Rasch", "2PL" (2-Parameter Logistic), or "3PL" (3-Parameter Logistic).
`method`	String. "EM" (Marginal Maximum Likelihood via Expectation-Maximization) or "MLE" (Joint Maximum Likelihood). However, using Bayesian will override the likelihood estimation.
`control`	A `list` of control parameters for the estimation algorithm:

- `max_iter`: Maximum number of EM iterations (default = 100).
- `converge_tol`: Convergence criterion for parameter change (default = 1e-4).
- `theta_range`: Numeric vector of length 2 specifying the integration grid bounds (default = c(-4, 4)).
- `quad_points`: Number of quadrature points (default = 21).
- `verbose`: Logical; if `TRUE`, prints progress to console.
- `prior`: A list specifying prior distributions or fixed values for item parameters. Default is NULL (no priors). For Rasch: `list(b = value)` or `list(b = function(x) dnorm(x, 0, 1, log=TRUE))`. For 2PL: `list(a = ..., b = ...)`. For 3PL: `list(a = ..., b = ..., g = ...)`. Each parameter can be: - A single numeric value (applied to all items) - A numeric vector of length k (item-specific fixed values) - A function returning log-density (applied to all items) - A list of length k with functions or values (item-specific)

Value

A list containing:

- `item_params`: A data frame of estimated item parameters (discrimination, difficulty, guessing) and their standard errors.
- `person_params`: A data frame of estimated person abilities (theta) and standard errors.
- `model_fit`: A data frame containing fit statistics such as Akaike's Information Criterion (AIC), the Bayesian Information Criterion (BIC), and Log-Likelihood.
- `settings`: A list of control parameters used in the estimation.

Examples

```
# # Simulate data
set.seed(123)
N <- 500; J <- 10
true_theta <- rnorm(N)
true_b <- seq(-2, 2, length.out=J)
true_a <- runif(J, 0.8, 1.2)
data_mat <- matrix(NA, N, J)
for(i in 1:N) {
  p <- 1 / (1 + exp(-true_a * (true_theta[i] - true_b)))
  data_mat[i,] <- rbinom(J, 1, p)
}
df <- as.data.frame(data_mat)
names(df) <- paste0("Q", 1:J)

# # Run Function without prior
res <- binary_irt(df, model="2PL", method="EM")


# # Run Function with prior (function-based)
res_prior <- binary_irt(df, model="2PL", method="EM",
                        control=list(prior=list(
                          a = function(x) dlnorm(x, 0, 0.5, log=TRUE),
                          b = function(x) dnorm(x, 0, 2, log=TRUE)
                        )))

# # Run Function with fixed value prior
res_fixed <- binary_irt(df, model="2PL", method="EM",
                        control=list(prior=list(
                          a = 1,  # Fix all discrimination prior to 1
                          b = function(x) dnorm(x, 0, 2, log=TRUE)
                        )))

# View Results
head(res$item_params)
head(res$person_params)
print(res$model_fit)
# --- Example 2: With Package Data ---
data("ela1", package = "tirt")
# Subset the first 30 columns (must use the object name 'data_binary')
```

```
df <- ela1[, 1:30]
# Run Function on package data
real_res <- binary_irt(df, model="2PL", method="EM")
head(real_res$item_params)
```

`dif` *Differential Item Functioning (Mantel-Haenszel and Logistic Regression)*

Description

Detects Differential Item Functioning (DIF) for dichotomous items, that is, items that behave differently for two groups of examinees (for example, a reference and a focal group) after matching on ability. Two widely used approaches are provided: the Mantel-Haenszel (1959) procedure, with the ETS delta effect size and A/B/C flagging scheme (Holland & Thayer, 1988), and the logistic regression approach (Swaminathan & Rogers, 1990), which separates uniform and non-uniform DIF.

Usage

```
dif(data, group, focal = NULL, match = NULL, purify = FALSE)
```

Arguments

`data`	A data frame of dichotomous (0/1) item responses (rows = persons, columns = items).
`group`	A vector of length equal to the number of rows in `data` giving the group membership of each person. It must have exactly two distinct non-missing values.
`focal`	Optional. The value of `group` that identifies the focal group. If `NULL` (default), the second value (in sorted order) is used as the focal group and the first as the reference group.
`match`	Optional numeric vector of matching scores (length equal to the number of rows in `data`). If `NULL` (default), the total number- correct score across all items is used.
`purify`	Logical. If `TRUE`, the matching total score is computed as the rest score (total minus the studied item) separately for each item, which avoids contaminating the match with the item under study (default `FALSE`).

Value

A data frame with one row per item and the columns:

- `item`: the item name.
- `MH_chisq`, `MH_p`: the Mantel-Haenszel chi-square (with continuity correction) and its p-value.
- `MH_OR`: the Mantel-Haenszel common odds ratio.

- `MH_delta`: the ETS delta effect size, $-2.35\ln(OR)$.
- `ETS_class`: the ETS DIF classification, `"A"` (negligible), `"B"` (moderate), or `"C"` (large).
- `LR_chisq`, `LR_p`: the 2-degree-of-freedom logistic regression test for combined (uniform + non-uniform) DIF.
- `LR_uniform_p`: p-value for uniform DIF (group main effect).
- `LR_nonuniform_p`: p-value for non-uniform DIF (group-by-match interaction).

References

Holland, P. W., & Thayer, D. T. (1988). Differential item performance and the Mantel-Haenszel procedure. In H. Wainer & H. I. Braun (Eds.), *Test validity* (pp. 129-145). Erlbaum.

Swaminathan, H., & Rogers, H. J. (1990). Detecting differential item functioning using logistic regression procedures. *Journal of Educational Measurement, 27*(4), 361-370.

Examples

```
set.seed(123)
sim <- sim_irt(n_people = 600,
               item_structure = list(list(model = "2PL", n_items = 10)))
resp <- sim$resp

# Create two groups and plant DIF in item 3 (harder for group "B")
grp <- rep(c("A", "B"), each = 300)
flip <- grp == "B" & resp[[3]] == 1
resp[[3]][flip] <- rbinom(sum(flip), 1, 0.6)

dif_res <- dif(resp, group = grp)
dif_res
```

`ela1` *Mixed-Format English Language Arts (ELA) Assessment Data (Form 1)*

Description

A dataset containing binary and polytomous responses for demonstration.

Usage

```
ela1
```

Format

A data frame with 52417 rows and 47 columns:

- `ITEM1` - `ITEM30`: Binary responses (0 = Incorrect, 1 = Correct).
- `ITEM31` - `ITEM45`: Polytomous responses (scored 0-5).
- `THETA`: Latent ability estimates.
- `COVARIATE`: Person-level background variable.

Source

Tang, C., Xiong, J., & Engelhard, G. (2025). Identification of writing strategies in educational assessments with an unsupervised learning measurement framework. *Education Sciences*, 15(7), 912. doi:10.3390/educsci15070912

Examples

```
data(ela1)
head(ela1)
```

---

ela2 *Mixed-Format English Language Arts (ELA) Assessment Data (Form 2)*

Description

A smaller dataset containing item responses.

Usage

```
ela2
```

Format

A data frame with columns representing item responses.

- `ITEM1` - `ITEM7`: Binary responses (0 = Incorrect, 1 = Correct).
- `ITEM8`: Polytomous response (scored 0-2).
- `ITEM9`: Polytomous response (scored 0-5).
- `ITEM10`: Polytomous response (scored 0-5).

Source

Tang, C., Xiong, J., & Engelhard, G. (2025). Identification of writing strategies in educational assessments with an unsupervised learning measurement framework. *Education Sciences*, 15(7), 912. doi:10.3390/educsci15070912

Examples

```
data(ela2)
head(ela2)
```

ela3 — *Large-Scale Mixed-Format English Language Arts (ELA) Assessment Data (Form 3)*

Description

A long format dataset containing binary and polytomous responses for demonstration.

Usage

```
ela3
```

Format

A data frame with 2,434,185 observations and 5 variables:

- STUDENTID: Unique student identifier.
- FORMID: Unique test form identifier.
- ITEMID: Unique item identifier, where _T1 indicates trait 1 and _T2 indicates trait 2.
- SEQ: Position of the item in a form.
- SCORE: Dichotomously and polytomously scored item responses (0,1,2...).

Source

Tang, C., Xiong, J., & Engelhard, G. (2025). Identification of writing strategies in educational assessments with an unsupervised learning measurement framework. *Education Sciences*, 15(7), 912. doi:10.3390/educsci15070912

Examples

```
data(ela3)
head(ela3)
```

ela3_testmap — *Large-Scale Mixed-Format English Language Arts (ELA) Assessment Data Testmap (Form 3 Testmap)*

Description

Test map for the long format ela3 response data

Usage

```
ela3_testmap
```

Format

A data frame with 328 rows and 5 variables:

- FORMID: Unique test form identifier.
- SEQ: Position of the item in a form.
- ITEMID: Unique item identifier, where _T1 indicates trait 1 and _T2 indicates trait 2.
- TYPE: Type of an item, where OP indicates operational items and FT indicates field-test items.
- MAX_SCORE: Max score of that item.

Source

Tang, C., Xiong, J., & Engelhard, G. (2025). Identification of writing strategies in educational assessments with an unsupervised learning measurement framework. *Education Sciences*, 15(7), 912. doi:10.3390/educsci15070912

Examples

```
data(ela3_testmap)
head(ela3_testmap)
```

---

equate_irt *Item Response Theory Equating / Linking*

Description

Conducts item response theory scale linking using Mean-Mean, Mean-Sigma, and Stocking-Lord methods. Supports mixed formats of both dichotomous and polytomous models. Automatically detects anchor items and validates model consistency.

Usage

```
equate_irt(base_params, new_params, person_params = NULL, methods = NULL)
```

Arguments

base_params	Data frame of reference item parameters (Form X).
new_params	Data frame of new item parameters to be transformed (Form Y).
person_params	(Optional) Data frame of person parameters from Form Y.
methods	Character vector. Options: "Mean-Mean", "Mean-Sigma", "Stocking-Lord". If NULL, defaults to all three.

Value

A list containing three data frames:

`transformed_item_params`
: New items transformed to Base scale (with SEs).

`transformed_person_params`
: New persons transformed to Base scale (if provided).

`linking_constants`
: The A (slope) and B (intercept) constants for each method.

Examples

```
# ============================================================================
# Example: Equating Form Y (New) to Form X (Base)
# ============================================================================
set.seed(123)

# 1. Generate "True" Base Parameters (Form X)
# ----------------------------------------------------------------------------
# 10 Common Items (Anchors) + 10 Unique Items
# 2PL and GRM mixed

gen_item_params <- function(n, type="2PL") {
  if(type=="2PL") {
    data.frame(
      item = paste0("Item_", 1:n),
      model = "2PL",
      a = round(runif(n, 0.8, 1.5), 2),
      b = round(rnorm(n, 0, 1), 2),
      stringsAsFactors = FALSE
    )
  } else {
    # GRM with 3 thresholds
    d <- t(apply(matrix(rnorm(n*3, 0, 0.5), n, 3), 1, sort))
    df <- data.frame(
      item = paste0("Poly_", 1:n),
      model = "GRM",
      a = round(runif(n, 0.8, 1.5), 2),
      stringsAsFactors = FALSE
    )
    df <- cbind(df, setNames(as.data.frame(d), paste0("step_", 1:3)))
    df
  }
}

# Anchors
anchor_2pl <- gen_item_params(5, "2PL")
anchor_grm <- gen_item_params(3, "GRM")
# Unique Form X
unique_x <- gen_item_params(5, "2PL")
unique_x$item <- paste0("X_", unique_x$item)
```

```
base_params <- dplyr::bind_rows(anchor_2pl, anchor_grm, unique_x)

# 2. Generate "New" Form Y Parameters (with Scale Shift)
# ---------------------------------------------------------------------------
# Scale Transformation: Theta_base = 1.2 * Theta_new + 0.5
# True Constants: A = 1.2, B = 0.5
TRUE_A <- 1.2
TRUE_B <- 0.5

# Transform Anchor Parameters to "New" scale (Inverse Logic)
# a_new = a_base * A
# b_new = (b_base - B) / A

anchor_2pl_new <- anchor_2pl
anchor_2pl_new$a <- anchor_2pl$a * TRUE_A
anchor_2pl_new$b <- (anchor_2pl$b - TRUE_B) / TRUE_A

anchor_grm_new <- anchor_grm
anchor_grm_new$a <- anchor_grm$a * TRUE_A
step_cols <- grep("step_", names(anchor_grm_new))
anchor_grm_new[, step_cols] <- (anchor_grm[, step_cols] - TRUE_B) / TRUE_A

# Unique Form Y
unique_y <- gen_item_params(5, "2PL")
unique_y$item <- paste0("Y_", unique_y$item)

new_params <- dplyr::bind_rows(anchor_2pl_new, anchor_grm_new, unique_y)

# 3. Create Dummy Person Parameters for Form Y
# ---------------------------------------------------------------------------
person_params <- data.frame(
  id = paste0("P", 1:50),
  theta = rnorm(50, 0, 1),
  theta_se = runif(50, 0.2, 0.5)
)

# 4. Perform Equating
# ---------------------------------------------------------------------------
# We expect to recover A approx 1.2 and B approx 0.5
results <- equate_irt(
  base_params = base_params,
  new_params = new_params,
  person_params = person_params,
  methods = c("Mean-Mean", "Stocking-Lord")
)

# 5. Inspect Results
# ---------------------------------------------------------------------------
# Linking Constants
print(results$linking_constants)

# Transformed Items (Form Y items on Form X scale)
head(results$transformed_item_params)
```

```
# Transformed Persons
head(results$transformed_person_params)
```

---

fixed_item *Fixed Item Calibration*

Description

Estimates unknown item parameters using Marginal Maximum Likelihood via Expectation-Maximization Algorithm. Uses a custom Bounded Newton-Raphson solver. Supports mixed-format data containing dichotomous and polytomous responses

Usage

```
fixed_item(response_df, item_params_df, control = list())
```

Arguments

`response_df` A data.frame of responses. Rows=Students, Cols=Items. Data MUST be from 0-indexed (0, 1, 2...).

`item_params_df` A data.frame of known parameters. Required: "item", "model".

`control` A `list` of control parameters for the estimation algorithm:

- `max_iter`: Maximum number of EM iterations (default = 50).
- `conv_crit`: Convergence criterion for parameter change (default = 0.005).
- `verbose`: Logical; if `TRUE`, prints progress to console.

Value

A list containing:

- `item_params`: Estimated parameters for unknown items.
- `person_params`: Estimated person parameters.
- `model_fit`: A data frame containing number off estimated parameters and fit statistics such as Akaike's Information Criterion (AIC), the Bayesian Information Criterion (BIC), and Log-Likelihood.

Examples

```
# 1. TOY EXAMPLE
# ============================================================================
set.seed(123)
# Create a very small dataset (N=50, J=4)
N_toy <- 50
df_toy <- data.frame(
  I1 = rbinom(N_toy, 1, 0.5), I2 = rbinom(N_toy, 1, 0.6), # Known items
  U1 = rbinom(N_toy, 1, 0.5), U2 = rbinom(N_toy, 1, 0.4)  # Unknown items
```

```
)

# Define the "Known" parameters for I1 and I2
known_params <- data.frame(
  item = c("I1", "I2"),
  model = c("2PL", "2PL"),
  a = c(1.0, 1.2),
  b = c(-0.5, 0.5)
)

# Run Fixed Item Calibration with very low iterations
fit_toy <- fixed_item(df_toy, known_params, control=list(max_iter=2, verbose=FALSE))
print(head(fit_toy$item_params))

  # --- Example 2: Simulation ---
  set.seed(123)
  N <- 500
  true_theta <- rnorm(N, 0, 1)

  # 1. Simulation Helpers
  sim_2pl <- function(theta, a, b) {
    p <- 1 / (1 + exp(-1.7 * a * (theta - b)))
    rbinom(N, 1, p)
  }
  sim_poly <- function(theta, a, steps) {
    n_cat <- length(steps) + 1
    probs <- matrix(0, length(theta), n_cat)
    for(k in 1:n_cat) {
      score <- k - 1
      if(score == 0) num <- 0
      else num <- a * (score * theta - sum(steps[1:score]))
      probs[, k] <- exp(num)
    }
    probs <- probs / rowSums(probs)
    apply(probs, 1, function(x) sample(0:(n_cat-1), 1, prob=x))
  }

  # 2. Generate Data (Mixed Known/Unknown Items)
  # Items 1-5: Known Binary (2PL)
  # Items 6-10: Unknown Binary (2PL)
  # Items 11-12: Known Poly (GPCM)
  # Items 13-15: Unknown Poly (GPCM)

  resp_mat <- matrix(NA, N, 15)
  colnames(resp_mat) <- paste0("Item_", 1:15)

  # Known Binary Parameters
  a_bin <- c(1.0, 1.2, 0.9, 1.1, 0.8)
  b_bin <- c(-1, -0.5, 0, 0.5, 1)

  for(i in 1:5) resp_mat[,i] <- sim_2pl(true_theta, a_bin[i], b_bin[i])
  for(i in 6:10) resp_mat[,i] <- sim_2pl(true_theta, runif(1,0.8,1.2), rnorm(1))
```

```
# Known Poly Parameters
a_poly <- c(1.0, 0.8)
d_poly <- list(c(-1, 1), c(-0.5, 0.5))

resp_mat[,11] <- sim_poly(true_theta, a_poly[1], d_poly[[1]])
resp_mat[,12] <- sim_poly(true_theta, a_poly[2], d_poly[[2]])
for(i in 13:15) resp_mat[,i] <- sim_poly(true_theta, 1.0, c(-0.5, 0.5))

df_resp <- as.data.frame(resp_mat)

# 3. Create 'Known Parameters' Dataframe
# This tells the function: "Fix these, Estimate the rest"
known_df <- data.frame(
  item = c(paste0("Item_", 1:5), "Item_11", "Item_12"),
  model = c(rep("2PL", 5), rep("GPCM", 2)),
  a = c(a_bin, a_poly),
  b = c(b_bin, NA, NA),     # Binary difficulty
  step_1 = c(rep(NA, 5), -1, -0.5), # Poly steps
  step_2 = c(rep(NA, 5),  1,  0.5),
  stringsAsFactors = FALSE
)

# 4. Run Estimation
res <- fixed_item(df_resp, known_df, control=list(max_iter=20))

# View Results
# Notice Items 1-5 and 11-12 have Status "Fixed"
head(res$item_params, 12)

# --- Example 2: With Package Data ---
data("ela1", package = "tirt")

# Let's treat the first 5 items as "Known" with arbitrary parameters
# just to demonstrate syntax.
df_real <- ela1[, 1:20]

known_real <- data.frame(
  item = paste0("Q", 1:5),
  model = "2PL",
  a = 1.0,
  b = seq(-1, 1, length.out=5)
)

# Ideally, column names in df_real should match 'item' column in known_real
colnames(df_real)[1:5] <- paste0("Q", 1:5)

real_res <- fixed_item(df_real, known_real, control=list(max_iter=10))
head(real_res$item_params)
```

fix_person *Fixed Person Calibration with or without Covariate*

Description

Estimates item parameters (difficulty, discrimination) given fixed person parameters (theta), with an optional person-level covariate. Supports Rasch and 2-Parameter Logistic models.

Usage

```
fix_person(df, theta, model = c("Rasch", "2PL"), covariate = NULL)
```

Arguments

`df`	A data frame of item responses (0/1). Columns represent items, rows represent persons.
`theta`	A numeric vector of person abilities (fixed parameters). Must match the number of rows in `df`.
`model`	A character string specifying the model type. Options are "Rasch" or "2PL".
`covariate`	An optional numeric vector representing a person-level covariate (e.g., time, group). Defaults to `NULL`.

Value

A data frame containing:

- Item statistics (difficulty, standard errors, z-values, p-values).
- Discrimination parameters (for 2PL model).
- Global covariate effect (if `covariate` is provided).
- Classical item statistics (p-value, count, point-biserial correlation).
- Mean theta per item (average ability of persons answering the item).
- Infit and Outfit statistics (for Rasch model only).

Examples

```
# --- Example: With Selected Package Data ---
data("ela1", package = "tirt")

# Subset data for a manageable example
# Select the first 500 examinees and 30 item responses
df_real <- ela1[1:500, 1:30]

# Extract pre-estimated latent traits and covariates
fixed_theta <- ela1$THETA[1:500]
fixed_cov <- ela1$COVARIATE[1:500]

# Estimate item parameters given fixed ability levels
# fitting a 2-parameter logistic (2PL) model
real_res <- fix_person(df = df_real,
                                 theta = fixed_theta,
                                 model = "2PL",
                                 covariate = fixed_cov)
```

```
head(real_res)

# --- Example: With Package Data ---
data("ela1", package = "tirt")

# Select Item Responses (Cols 1-30)
df_real <- ela1[, 1:30]

fixed_theta <- ela1$THETA
fixed_cov <- ela1$COVARIATE

real_res <- fix_person(df = df_real,
                                  theta = fixed_theta,
                                  model = "2PL",
                                  covariate = fixed_cov)
head(real_res)
```

`irt_trt` *Joint Item Response Theory and Testlet Response Theory Estimation (Dichotomous & Polytomous)*

Description

Provides a unified marginal maximum likelihood estimation framework for a broad class of item response theory and testlet response theory models. The function automatically detects data structures to apply appropriate models, along with their testlet-effect extensions (Bradlow et al., 1999).

Usage

```
irt_trt(data, item_spec, method = "EM", control = list())
```

Arguments

`data` A `data.frame` or `matrix` containing item responses. Responses should be 0-indexed integers. Missing values should be coded as `NA`.

`item_spec` A `data.frame` providing item metadata. Must include columns `"item"` (matching `colnames(data)`) and `"model"`. Optionally includes a `"testlet"` column for TRT specifications.

`method` A character string specifying the estimation method. Currently supports `"EM"` (Expectation-Maximization). Defaults to `"EM"`.

`control` A `list` of control parameters for the estimation algorithm:

- `max_iter`: Maximum number of EM iterations (default = 100).
- `converge_tol`: Convergence criterion for parameter change (default = 1e-4).
- `theta_range`: Numeric vector of length 2 specifying the integration grid bounds (default = c(-4, 4)).

- `quad_points`: Number of quadrature points (default = 21).
- `verbose`: Logical; if `TRUE`, prints progress to console.
- `fix_discrimination`: Logical; default=FALSE

Details

The estimation utilizes a robust Newton-Raphson update within the M-step. For testlet models, dimension reduction is achieved through the integration of the nuisance testlet effect (Li et al., 2006). The function automatically corrects model specifications if the data levels (binary vs. polytomous) do not align with the requested model string.

Value

A `list` containing three components:

`item_params`	A data frame of estimated item slopes (discrimination), difficulties/thresholds, and guessing parameters with associated standard errors.
`person_params`	A data frame of EAP-based ability estimates (θ) and testlet effect estimates (γ).
`model_fit`	A data frame containing Log-Likelihood, AIC, and BIC indices.

References

Bradlow, E. T., Wainer, H., & Wang, X. (1999). A testlet response model for multidimensionality in item response theory. *Psychometrika, 64*(2), 147-168.

Li, Y., Bolt, D. M., & Fu, J. (2006). A comparison of methods for estimating secondary dimensions in testlet-based data. *Applied Psychological Measurement, 30*(3), 203-223.

Examples

```
# --- Quick Example (small data) ---
set.seed(1)
n <- 50
resp <- data.frame(
  I1 = rbinom(n, 1, 0.6),
  I2 = rbinom(n, 1, 0.5),
  I3 = rbinom(n, 1, 0.4),
  I4 = sample(0:2, n, replace = TRUE),
  I5 = rbinom(n, 1, 0.5),
  I6 = rbinom(n, 1, 0.6)
)
spec <- data.frame(
  item = colnames(resp),
  model = c("2PL", "2PL", "2PL", "GRM", "2PLT", "2PLT"),
  testlet = c(NA, NA, NA, NA, "T1", "T1"),
  stringsAsFactors = FALSE
)
res <- irt_trt(resp, spec, method = "EM",
               control = list(max_iter = 5, verbose = FALSE))
head(res$item_params)
```

```
# --- Full Example: Simulation (Binary + Poly + Testlets) ---
set.seed(2025)
N <- 100; J <- 20

# 1. Generate Parameters
theta <- rnorm(N, 0, 1)
gamma_1 <- rnorm(N, 0, 0.5) # Testlet 1 effect
gamma_2 <- rnorm(N, 0, 0.6) # Testlet 2 effect

a_true <- runif(J, 0.8, 1.5)
b_true <- seq(-1.5, 1.5, length.out = J)

resp_matrix <- matrix(NA, N, J)
colnames(resp_matrix) <- paste0("Item_", 1:J)

# 2. Simulate Responses
# Items 1-10: Binary Independent (Model: 2PL)
for(j in 1:10) {
  p <- 1 / (1 + exp(-a_true[j] * (theta - b_true[j])))
  resp_matrix[,j] <- rbinom(N, 1, p)
}

# Items 11-15: Poly Independent (Model: GRM)
for(j in 11:15) {
  thresh <- sort(c(b_true[j] - 0.7, b_true[j] + 0.7))
  p1 <- 1 / (1 + exp(-a_true[j] * (theta - thresh[1])))
  p2 <- 1 / (1 + exp(-a_true[j] * (theta - thresh[2])))
  probs <- cbind(1-p1, p1-p2, p2)
  resp_matrix[,j] <- apply(probs, 1, function(p) sample(0:2, 1, prob=p))
}

# Items 16-17: Binary Testlet 1 (Model: 2PLT)
for(j in 16:17) {
  eff_theta <- theta + gamma_1
  p <- 1 / (1 + exp(-a_true[j] * (eff_theta - b_true[j])))
  resp_matrix[,j] <- rbinom(N, 1, p)
}

# Items 18-20: Poly Testlet 2 (Model: GRT)
for(j in 18:20) {
  eff_theta <- theta + gamma_2
  thresh <- sort(c(b_true[j] - 0.5, b_true[j] + 0.5))
  p1 <- 1 / (1 + exp(-a_true[j] * (eff_theta - thresh[1])))
  p2 <- 1 / (1 + exp(-a_true[j] * (eff_theta - thresh[2])))
  probs <- cbind(1-p1, p1-p2, p2)
  resp_matrix[,j] <- apply(probs, 1, function(p) sample(0:2, 1, prob=p))
}

df_sim <- as.data.frame(resp_matrix)

# 3. Create Item Specification
# STRICT naming: Independent=2PL/GRM, Testlet=2PLT/GRT
```

```
spec <- data.frame(
  item = colnames(df_sim),
  model = c(rep("2PL", 10), rep("GRM", 5), rep("2PLT", 2), rep("GRT", 3)),
  testlet = c(rep(NA, 15), rep("T1", 2), rep("T2", 3)),
  stringsAsFactors = FALSE
)

# 4. Run Estimation
res <- irt_trt(df_sim, spec, method = "EM",
               control = list(max_iter = 20, verbose = FALSE))

head(res$item_params)
head(res$person_params)
```

---

item_fit *Item Fit Statistics (Infit and Outfit)*

Description

Computes item-level fit statistics based on standardized response residuals: the information-weighted mean square (infit) and the unweighted mean square (outfit), together with their standardized (ZSTD) transformations. These statistics flag items whose observed responses depart from the fitted model, either because of noise in the tails of the ability distribution (outfit) or near the item location (infit). The statistics are computed for dichotomous and polytomous items alike.

Usage

```
item_fit(
  data,
  item_params,
  theta = NULL,
  theta_range = c(-4, 4),
  n_points = 41,
  model = NULL,
  D = 1
)
```

Arguments

data	A data frame of item responses (rows = persons, columns = items). Dichotomous items are 0/1; polytomous items are consecutive integers starting at 0. Item order must match `item_params`.
item_params	A data frame of item parameters (the `item_params` element from `binary_irt`, `polytomous_irt`, or `mixed_irt`).
theta	Optional numeric vector of estimated abilities, or a person- parameter data frame with an `ability/theta` column. If `NULL` (default), abilities are estimated internally by EAP scoring.

theta_range — Numeric vector of length 2 giving the ability grid bounds used for internal EAP scoring (default c(-4, 4)).

n_points — Integer. Number of quadrature points for internal EAP scoring (default 41).

model — Optional model override (see item_info).

D — Scaling constant for the dichotomous logistic metric (default 1).

Value

A data frame with one row per item and the columns:

- item: item name.
- n: number of responses used.
- outfit: outfit mean square (unweighted).
- outfit_z: standardized outfit (ZSTD).
- infit: infit mean square (information-weighted).
- infit_z: standardized infit (ZSTD).

Mean-square values near 1 indicate good fit; values above about 1.3-1.5 suggest underfit (noise), and values below about 0.7 suggest overfit (dependency or redundancy).

References

Wright, B. D., & Masters, G. N. (1982). *Rating scale analysis*. MESA Press.

Examples

```
set.seed(123)
sim <- sim_irt(n_people = 400,
               item_structure = list(list(model = "2PL", n_items = 8)))
fit <- binary_irt(sim$resp, model = "2PL", method = "EM",
                  control = list(max_iter = 15, verbose = FALSE))

item_fit(sim$resp, fit$item_params)
```

---

item_info *Item Information Function*

Description

Computes the Fisher item information function for every item at a set of ability (theta) points. The item information function shows how precisely an item measures ability across the latent trait continuum: high information at a given theta means the item discriminates well among examinees located there. It is the building block of the test information function (test_info) and of the conditional standard error of measurement.

Usage

```
item_info(item_params, theta = seq(-4, 4, by = 0.1), model = NULL, D = 1)
```

Arguments

`item_params`	A data frame of item parameters, typically the `item_params` element returned by `binary_irt`, `polytomous_irt`, or `mixed_irt`. The function auto-detects the model of each item from the available columns (discrimination, difficulty, guessing, thresholds/steps) or from a `model` column if present. All models in the package are supported: Rasch, 2PL, 3PL, GRM, GPCM, and PCM.
`theta`	A numeric vector of ability values at which to evaluate the information (default = `seq(-4, 4, by = 0.1)`).
`model`	Optional character string (applied to all items) or character vector of length equal to the number of items, forcing the model family used for each item. Advanced use only; when `NULL` (default) the model is detected automatically.
`D`	Scaling constant for the dichotomous logistic metric (default `1`, matching the estimation functions in this package). Advanced users who calibrated on the normal-metric scale may set `D = 1.702`.

Details

For dichotomous models the information is $I(\theta) = (Da)^2 P(\theta)(1 - P(\theta))$ for the Rasch and 2PL models, and the standard Birnbaum form $I(\theta) = (Da)^2(1-c)(1-P^*)P^{*2}/P$ for the 3PL model, where P^* is the 2PL probability and P the full 3PL probability (this reduces to the 2PL expression when the guessing parameter $c = 0$). For polytomous models the general expected-information formula $I(\theta) = \sum_k (\partial P_k/\partial\theta)^2/P_k$ is used, which for the GPCM and PCM equals $(Da)^2 \mathrm{Var}(X \mid \theta)$.

Value

A numeric matrix with one row per item and one column per theta point. Row names are the item names and column names are the theta values.

See Also

`test_info`, `binary_irt`, `polytomous_irt`

Examples

```
# --- Example 1: dichotomous items ---
set.seed(123)
sim <- sim_irt(n_people = 400,
               item_structure = list(list(model = "2PL", n_items = 6)))
fit <- binary_irt(sim$resp, model = "2PL", method = "EM",
                  control = list(max_iter = 15, verbose = FALSE))

# Information for each item at a grid of abilities
info <- item_info(fit$item_params, theta = seq(-3, 3, by = 0.5))
round(info, 3)
```

```
# --- Example 2: polytomous (GRM) items ---
simp <- sim_irt(n_people = 400,
                item_structure = list(list(model = "GRM", n_items = 5,
                                           categories = 4)))
fitp <- polytomous_irt(simp$resp, model = "GRM", method = "EM",
                       control = list(max_iter = 15, verbose = FALSE))
item_info(fitp$item_params, theta = c(-2, -1, 0, 1, 2))
```

`ld_stats` *Local Dependence Statistics (Yen's Q3)*

Description

Computes Yen's (1984) Q_3 statistic for every item pair. Q_3 is the correlation between the model residuals of two items after the latent trait has been partialled out. Under the local-independence assumption of unidimensional item response theory these residual correlations should be small and negative (around $-1/(J-1)$); large positive values signal that two items share something beyond the common trait, such as a passage or a scenario. Q_3 is therefore used both to decide whether testlet modeling is needed and to check whether a testlet model has adequately absorbed the residual dependence.

Usage

```
ld_stats(
  data,
  item_params,
  theta = NULL,
  theta_range = c(-4, 4),
  n_points = 41,
  model = NULL,
  D = 1
)
```

Arguments

`data`	A data frame of item responses (rows = persons, columns = items). Dichotomous items are 0/1; polytomous items are consecutive integers starting at 0.
`item_params`	A data frame of item parameters (the `item_params` element from `binary_irt`, `polytomous_irt`, or `mixed_irt`). Item order must match the columns of `data`.
`theta`	Optional numeric vector of estimated abilities (length equal to the number of rows in `data`), or a person-parameter data frame with an `ability/theta` column. If `NULL` (default), abilities are estimated internally by expected a posteriori (EAP) scoring from `item_params`.

theta_range	Numeric vector of length 2 giving the ability grid bounds used for internal EAP scoring (default c(-4, 4)).
n_points	Integer. Number of quadrature points for internal EAP scoring (default 41).
model	Optional model override (see item_info).
D	Scaling constant for the dichotomous logistic metric (default 1).

Value

A symmetric numeric matrix of Q_3 statistics with one row and one column per item (diagonal set to 1). The off-diagonal entries are the residual correlations for each item pair. The average off-diagonal Q_3 is attached as the attribute "mean_q3" and the largest absolute value as "max_abs_q3".

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.

Examples

```
set.seed(123)
sim <- sim_irt(n_people = 400,
               item_structure = list(list(model = "2PL", n_items = 8)))
fit <- binary_irt(sim$resp, model = "2PL", method = "EM",
                  control = list(max_iter = 15, verbose = FALSE))

q3 <- ld_stats(sim$resp, fit$item_params)
round(q3, 3)
attr(q3, "max_abs_q3")
```

---

mirt_binary *Multidimensional Binary Item Response Theory Estimation*

Description

Estimates item and person parameters for multidimensional binary item response models (M-Rasch, M-2PL, M-3PL). The function is fully self-contained, using custom Newton-Raphson optimization without relying on external optimizers.

Usage

```
mirt_binary(
  data,
  model = "2PL",
  dimension = 2,
  method = "MML",
  control = list()
)
```

Arguments

`data`	A N x J data.frame or matrix of dichotomous responses (0/1).
`model`	String. "Rasch", "2PL", or "3PL".
`dimension`	Integer. Number of latent dimensions to estimate (D).
`method`	String. Estimation method: "MML" (Marginal Maximum Likelihood with EM, best for D <= 3), "MHRM" (Metropolis-Hastings Robbins-Monro, for high D), or "RVEM" (Dimension-Reduction EM).
`control`	A `list` of control parameters for the estimation algorithm: • `max_iter`: Maximum number of iterations (default = 100). • `converge_tol`: Convergence criterion (default = 1e-4). • `theta_range`: Bounds for quadrature integration (default = c(-4, 4)). • `quad_points`: Quadrature points PER DIMENSION for MML (default = 15). • `ability`: Logical; estimate person abilities (EAP) after item calibration? (default = TRUE). • `verbose`: Logical; prints progress and psychometric messages. • `Q_matrix`: Optional J x D matrix of 0s and 1s indicating item-to-dimension loading. If NULL, an exploratory lower-triangular constraint is applied for identifiability.

Value

A list containing:

- `item_params`: A data frame of estimated item parameters (a_1...a_D, d, g) and standard errors.
- `person_params`: A data frame of estimated multidimensional person abilities (if ability=TRUE).
- `model_fit`: Fit statistics (Log-Likelihood, AIC, BIC).
- `settings`: Control parameters used in the run.

Examples

```
# --- Simulation for Multidimensional Data ---
set.seed(202)
N <- 800
J <- 20
D <- 2

# Simulate Abilities (2 Dimensions, correlated)
Sigma <- matrix(c(1, 0.4, 0.4, 1), 2, 2)
Z <- matrix(rnorm(N * D), N, D)
theta <- Z %*% chol(Sigma)

# Define Q-matrix (Items 1-10 on Dim1, Items 11-20 on Dim 2)
Q <- matrix(0, J, D)
Q[1:10, 1] <- 1
Q[11:20, 2] <- 1
```

```
# Simulate Item Parameters (Multidimensional 2PL)
a_true <- matrix(runif(J * D, 0.8, 1.8), J, D) * Q
d_true <- seq(-2, 2, length.out = J)

# Generate Responses
data_mat <- matrix(NA, N, J)
for(i in 1:N) {
  # Matrix multiply JxD slopes by Dx1 person abilities, add Jx1 intercepts
  z <- as.vector(a_true %*% theta[i, ]) + d_true
  p <- 1 / (1 + exp(-z))
  data_mat[i, ] <- rbinom(J, 1, p)
}
df <- as.data.frame(data_mat)
names(df) <- paste0("Item", 1:J)

# --- Run Custom Multidimensional Function ---
res <- mirt_binary(df, model="2PL", dimension=2, method="MML",
                   control=list(Q_matrix=Q, ability=TRUE, max_iter=10))

print(head(res$item_params))
print(head(res$person_params))
print(res$model_fit)

# --- Validation with 'mirt' (For comparison, if installed) ---
# library(mirt)
# mirt_model <- paste0("F1 = 1-10\nF2 = 11-20\nCOV = F1*F2")
# mirt_res <- mirt(df, mirt.model(mirt_model), itemtype='2PL', method='EM')
# coef(mirt_res, IRTpars=FALSE, simplify=TRUE)$items
```

`mixed_irt` *Mixed Item Response Model Estimation (Dichotomous & Polytomous) with Prior Support*

Description

Provides a estimation framework for a broad class of different item response theory models. This function can model different combinations of item categories. Now supports flexible prior distributions for Bayesian estimation (MAP estimation).

Usage

```
mixed_irt(data, model = "2PL", method = "EM", control = list())
```

Arguments

`data` A N x J data.frame. Binary items must be 0/1. Polytomous items should be continuous integers (0, 1, 2...).

`model` A character vector of length J (one model per item). Supported: "Rasch", "2PL" (2-Parameter Logistic), "3PL" (3-Parameter Logistic), "GRM" (Graded Response Model), "GPCM" (Generalized Partial Credit Model), "PCM" (Partial Credit Model). If a single string is provided, it is applied to all the same type of items.

`method` String. "EM" (Marginal Maximum Likelihood via Expectation-Maximization) or "MLE" (Joint Maximum Likelihood).

`control` A `list` of control parameters for the estimation algorithm:

- `max_iter`: Maximum number of EM iterations (default = 100).
- `converge_tol`: Convergence criterion for parameter change (default = 1e-4).
- `theta_range`: Numeric vector of length 2 specifying the integration grid bounds (default = c(-4, 4)).
- `quad_points`: Number of quadrature points (default = 21).
- `verbose`: Logical; if `TRUE`, prints progress to console.
- `prior`: A list of model-specific prior distributions. Default is NULL (no priors). Format: `list("2PL" = list(a = function(x) ..., b = function(x) ...), "GRM" = list(...))` Each model gets its own prior specification. All items of the same model share the same prior. Example: `prior = list( "2PL" = list(a = function(x) dlnorm(x, 0, 0.5, log=TRUE), b = function(x) dnorm(x, 0, 2, log=TRUE)), "GRM" = list(a = function(x) dlnorm(x, 0, 0.5, log=TRUE), d = function(x) dnorm(x, 0, 2, log=TRUE)) )`

Value

A list containing:

- `item_params`: Data frame of item parameters (discrimination, difficulty/thresholds, guessing).
- `person_params`: A data frame of estimated person abilities (theta) and standard errors.
- `model_fit`: A data frame containing fit statistics such as Akaike's Information Criterion (AIC) and the Bayesian Information Criterion (BIC).
- `settings`: A list of control parameters used in the estimation.

Examples

```
# --- Example 1: Simulation (Mixed 2PL + GPCM) ---
set.seed(2025)
N <- 500
n_bin <- 5
n_poly <- 2
J <- n_bin + n_poly

# 1. Generate Theta (Wide range to match user request)
true_theta <- rnorm(N, mean = 0, sd = 3)

# 2. Simulation Helper: GPCM
sim_gpcm <- function(theta, a, steps) {
```

```
  n_cat <- length(steps) + 1
  probs <- matrix(0, length(theta), n_cat)
  for(k in 1:n_cat) {
    score <- k - 1
    if(score == 0) numer <- rep(0, length(theta))
    else numer <- a * (score * theta - sum(steps[1:score]))
    probs[, k] <- exp(numer)
  }
  probs <- probs / rowSums(probs)
  apply(probs, 1, function(p) sample(0:(n_cat-1), 1, prob=p))
}

# 3. Create Data
data_sim <- data.frame(matrix(NA, nrow = N, ncol = J))
colnames(data_sim) <- paste0("Item_", 1:J)

# Binary Items (2PL)
a_bin <- runif(n_bin, 0.8, 1.5)
b_bin <- seq(-3, 3, length.out = n_bin)
for(j in 1:n_bin) {
  prob <- 1 / (1 + exp(-(a_bin[j] * (true_theta - b_bin[j]))))
  data_sim[, j] <- rbinom(N, 1, prob)
}

# Polytomous Items (GPCM)
# Item 6: 2 steps (-2, 2)
data_sim[, 6] <- sim_gpcm(true_theta, a=1.0, steps=c(-2, 2))
# Item 7: 5 steps
data_sim[, 7] <- sim_gpcm(true_theta, a=1.2, steps=c(-5, -2.5, 0, 2.5, 5))

# 4. Run Estimation without prior
# Note: Wide theta_range needed due to SD=3 in simulation
my_models <- c(rep("2PL", n_bin), rep("GPCM", n_poly))

res <- mixed_irt(data = data_sim, model = my_models, method = "EM",
                 control = list(max_iter = 20, theta_range = c(-6, 6)))

head(res$item_params)
print(res$model_fit)


# 5. Run Estimation with prior (MAP)
res_prior <- mixed_irt(data = data_sim, model = my_models, method = "EM",
                       control = list(max_iter = 20, theta_range = c(-6, 6),
                                      prior = list(
                                        "2PL" = list(
                                          a = function(x) dlnorm(x, 0, 0.5, log=TRUE),
                                          b = function(x) dnorm(x, 0, 2, log=TRUE)
                                        ),
                                        "GPCM" = list(
                                          a = function(x) dlnorm(x, 0, 0.5, log=TRUE),
                                          d = function(x) dnorm(x, 0, 2, log=TRUE)
                                        )
```

```
                                        )))
head(res_prior$item_params)
print(res_prior$model_fit)
# --- Example 2: With Package Data ---
data("ela2", package = "tirt")

# Define Models (7 Binary, 3 Poly)
real_models <- c(rep("2PL", 7), rep("GRM", 3))

# Run Estimation
real_res <- mixed_irt(ela2, model = real_models, method = "EM",
                      control = list(max_iter = 10))

head(real_res$item_params)
print(real_res$model_fit)
```

`mixture_irt` *Mixture Item Response Theory Model (Latent-Class Rasch / 2PL)*

Description

Fits a mixture item response theory model (Rost, 1990) in which the examinee population is assumed to consist of a small number of unobserved latent classes, each with its own set of item parameters. Mixture IRT models are used to detect qualitatively different response strategies, unmodeled subpopulations, or classes for which item difficulty ordering differs. The model is estimated by the Expectation-Maximization algorithm with a fixed standard-normal ability distribution within each class.

Usage

```
mixture_irt(data, n_class = 2, model = "Rasch", control = list())
```

Arguments

`data`	A data frame of dichotomous (0/1) item responses (rows = persons, columns = items).
`n_class`	Integer. Number of latent classes to estimate (default `2`).
`model`	String. `"Rasch"` (default) or `"2PL"`; controls whether class-specific discriminations are estimated.
`control`	A `list` of control parameters for the algorithm: • `max_iter`: Maximum number of EM iterations (default = 100). • `converge_tol`: Convergence criterion on the log-likelihood change (default = 1e-4). • `quad_points`: Number of quadrature points for the within-class ability distribution (default = 21). • `theta_range`: Ability grid bounds (default = c(-4, 4)). • `verbose`: Logical; print progress (default = TRUE).

Value

A list containing:

- `item_params`: A data frame of class-specific item difficulties (and discriminations for the 2PL model), with one column per class.
- `class_params`: A data frame of estimated mixing proportions for each latent class.
- `person_params`: A data frame of posterior class-membership probabilities and the modal (most likely) class for each person.
- `model_fit`: A data frame with the log-likelihood, AIC, BIC, number of classes, and classification entropy.

References

Rost, J. (1990). Rasch models in latent classes: An integration of two approaches to item analysis. *Applied Psychological Measurement, 14*(3), 271-282.

Examples

```
# Two classes with reversed difficulty ordering
set.seed(2025)
N <- 300; J <- 8
b1 <- seq(-1.5, 1.5, length.out = J)
b2 <- rev(b1)
theta <- rnorm(N)
cls <- rep(1:2, each = N / 2)
resp <- matrix(0, N, J)
for (i in 1:N) {
  b <- if (cls[i] == 1) b1 else b2
  resp[i, ] <- rbinom(J, 1, 1 / (1 + exp(-(theta[i] - b))))
}
df <- as.data.frame(resp); names(df) <- paste0("I", 1:J)

fit <- mixture_irt(df, n_class = 2, model = "Rasch",
                   control = list(max_iter = 30, verbose = FALSE))
fit$class_params
head(fit$item_params)
```

`person_fit` *Person Fit Statistics (Standardized Log-Likelihood, lz)*

Description

Computes person-level fit statistics to detect examinees whose response patterns are unlikely under the fitted item response model (for example, careless responding, cheating, or unusual guessing). The function returns the l_z standardized log-likelihood index of Drasgow, Levine, and Williams (1985), which under model fit is approximately standard normal, and flags persons whose l_z falls below a critical value.

Usage

```
person_fit(data, item_params, theta, critical = -1.96, model = NULL, D = 1)
```

Arguments

data	A data frame of item responses (rows = persons, columns = items). Dichotomous items are 0/1; polytomous items are consecutive integers starting at 0. Missing values (NA) are allowed and are skipped person-by-person.
item_params	A data frame of item parameters (the item_params element from binary_irt, polytomous_irt, or mixed_irt). Item order must match the columns of data.
theta	A numeric vector of estimated person abilities (length equal to the number of rows in data), or a person-parameter data frame containing an ability or theta column (such as the person_params element returned by the estimation functions).
critical	Numeric. Persons with l_z below this value are flagged as potentially misfitting (default = -1.96, the lower 2.5\ the standard normal distribution; -2 is another common choice).
model	Optional model override (see item_info).
D	Scaling constant for the dichotomous logistic metric (default 1).

Details

For a person with ability θ, let l_0 be the log-likelihood of the observed responses. The index is $l_z = (l_0 - E[l_0])/\sqrt{\mathrm{Var}(l_0)}$, where the expectation and variance are taken over the model-implied category probabilities at θ. Large negative values indicate response patterns that are less likely than the model predicts.

Value

A data frame with one row per person and the columns:

- person: person index (row number).
- n_items: number of items answered.
- theta: the ability estimate used.
- loglik: the observed log-likelihood of the response pattern.
- lz: the standardized log-likelihood person-fit index.
- flag: logical; TRUE if lz < critical.

References

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

Examples

```
set.seed(123)
sim <- sim_irt(n_people = 300,
               item_structure = list(list(model = "2PL", n_items = 15)))
fit <- binary_irt(sim$resp, model = "2PL", method = "EM",
                  control = list(max_iter = 15, verbose = FALSE))

pf <- person_fit(sim$resp, fit$item_params, fit$person_params)
head(pf)

# How many examinees are flagged as misfitting?
sum(pf$flag, na.rm = TRUE)
```

---

polytomous_irt *Polytomous Item Response Theory Estimation Using Likelihood or Bayesian*

Description

Estimates item and person parameters for polytomous item response theory models using either Marginal Maximum Likelihood or Joint Maximum Likelihood. Now supports flexible prior distributions for Bayesian estimation (MAP estimation).

Usage

```
polytomous_irt(data, model = "GPCM", method = "EM", control = list())
```

Arguments

data	A N x J data.frame of polytomous responses (0, 1, 2...). Missing values should be NA. Categories must be continuous integers.
model	String. "GPCM" (Generalized Partial Credit Model), "PCM" (Partial Credit Model), or "GRM" (Graded Response Model).
method	String. "EM" (Marginal Maximum Likelihood via Expectation-Maximization) or "MLE" (Joint Maximum Likelihood). However, using Bayesian will override the likelihood estimation.
control	A `list` of control parameters for the estimation algorithm: • `max_iter`: Maximum number of EM iterations (default = 100). • `converge_tol`: Convergence criterion for parameter change (default = 1e-4). • `theta_range`: Numeric vector of length 2 specifying the integration grid bounds (default = c(-4, 4)). • `quad_points`: Number of quadrature points (default = 21). • `verbose`: Logical; if `TRUE`, prints progress to console.

- prior: A list specifying prior distributions for item parameters. Default is NULL (no priors). For GRM: `list(a = function(x) dlnorm(x, 0, 0.5, log=TRUE), d = function(x) dnorm(x, 0, 2, log=TRUE))`. For GPCM: `list(a = function(x) dlnorm(x, 0, 0.5, log=TRUE), d = function(x) dnorm(x, 0, 2, log=TRUE))`. For PCM: `list(d = function(x) dnorm(x, 0, 2, log=TRUE))`. Each prior is a function returning log-density and applies to ALL items. Fixed value priors are NOT supported. Item-specific priors are NOT supported.

Value

A list containing:

- item_params: Data frame of estimated parameters (a, thresholds).
- person_params: A data frame of estimated person abilities (theta) and standard errors.
- model_fit: A data frame containing fit statistics such as Akaike's Information Criterion (AIC) and the Bayesian Information Criterion (BIC).
- settings: A list of control parameters used in the estimation.

Examples

```
# --- Quick Example (small data) ---
set.seed(1)
n <- 50
resp <- data.frame(
  I1 = sample(0:2, n, replace = TRUE),
  I2 = sample(0:2, n, replace = TRUE),
  I3 = sample(0:3, n, replace = TRUE)
)
res <- polytomous_irt(resp, model = "GPCM", method = "EM",
                      control = list(max_iter = 5, verbose = FALSE))
head(res$item_params)

# --- Full Example 1: Simulation (GPCM) ---
set.seed(2026)
N <- 500; J <- 5
n_cats <- c(3, 4, 3, 5, 4)

true_theta <- rnorm(N)
true_a <- runif(J, 0.8, 1.2)
true_d <- list()

# Generate Thresholds
for(j in 1:J) {
  steps <- sort(rnorm(n_cats[j]-1, mean = 0, sd = 1.0))
  true_d[[j]] <- c(0, cumsum(steps))
}

# Simulation Helper (GPCM Logic)
generate_resp <- function(theta, a, d_vec, n_cat) {
  probs <- matrix(0, length(theta), n_cat)
```

```
  for(k in 1:n_cat) {
    z <- a * (k-1) * theta - d_vec[k]
    probs[,k] <- exp(z)
  }
  probs <- probs / rowSums(probs)
  apply(probs, 1, function(p) sample(0:(n_cat-1), 1, prob=p))
}

# Create Data
sim_data <- matrix(NA, nrow = N, ncol = J)
for(j in 1:J) {
  sim_data[,j] <- generate_resp(true_theta, true_a[j], true_d[[j]], n_cats[j])
}
df_sim <- as.data.frame(sim_data)

# Run Estimation (GPCM to match simulation logic without prior)
res <- polytomous_irt(df_sim, model="GPCM", method="EM",
                      control=list(max_iter=20, verbose=TRUE))

head(res$item_params)
print(res$model_fit)

# Run Estimation with prior (MAP)
res_prior <- polytomous_irt(df_sim, model="PCM", method="EM",
                            control=list(max_iter=20, verbose=FALSE,
                                         prior=list(
                                           d = function(x) dnorm(x, 0, 2, log=TRUE)
                                         )))
head(res$item_params)
print(res$model_fit)

# --- Example 2: With Package Data (GRM) ---
data("ela1", package = "tirt")

# Subset polytomous items (columns 31 to 45)
df_poly <- ela1[, 31:45]

# Run Estimation using GRM
real_res <- polytomous_irt(df_poly, model="GRM", method="EM",
                           control = list(max_iter = 1000))

head(real_res$item_params)
head(real_res$person_params)
print(real_res$model_fit)
# Run Estimation using GRM with prior
real_res2 <- polytomous_irt(df_poly, model="GRM", method="EM",
                           control = list(max_iter = 1000,
                                          prior = list(
                                           a = function(x) dlnorm(x, 0, 0.5, log=TRUE),
                                            d = function(x) dnorm(x, 0, 2, log=TRUE)
                                          )))

head(real_res2$item_params)
```

```
head(real_res2$person_params)
print(real_res2$model_fit)
```

reliability *Test Reliability Indices*

Description

Computes commonly reported reliability coefficients for a test. Depending on the information supplied it returns the empirical (marginal) reliability from latent-trait estimates and their standard errors, the model-based marginal reliability from the test information function, and the classical Cronbach's alpha from the raw responses. Reliability summarizes how consistently the test orders examinees and is a standard entry in every technical report.

Usage

```
reliability(
  person_params = NULL,
  data = NULL,
  item_params = NULL,
  prior_sd = 1,
  theta_range = c(-4, 4),
  n_points = 61,
  model = NULL,
  D = 1
)
```

Arguments

person_params	Optional. A person-parameter data frame containing an ability/theta column and a standard-error column (such as ability_se or se); typically the person_params element returned by the estimation functions. Used for the empirical reliability.
data	Optional. A data frame of item responses used to compute Cronbach's alpha (rows = persons, columns = items).
item_params	Optional. A data frame of item parameters used to compute the model-based marginal reliability from the test information function.
prior_sd	Numeric. Standard deviation of the population ability distribution used for the model-based marginal reliability (default 1).
theta_range	Numeric vector of length 2 giving the ability grid bounds for the marginal-reliability integration (default c(-4, 4)).
n_points	Integer. Number of grid points for the marginal-reliability integration (default 61).
model	Optional model override (see item_info).
D	Scaling constant for the dichotomous logistic metric (default 1).

Details

The empirical reliability is $\mathrm{Var}(\hat{\theta})/(\mathrm{Var}(\hat{\theta}) + \overline{SE^2})$. The model-based marginal reliability is $\sigma^2/(\sigma^2 + \overline{1/I(\theta)})$, where the average error variance is taken over the population ability distribution. Cronbach's alpha is $\frac{J}{J-1}\left(1 - \frac{\sum_j s_j^2}{s_T^2}\right)$.

Value

A data frame with the columns `Index` and `Value`, reporting whichever reliability coefficients could be computed from the supplied arguments:

- `empirical_reliability`: from person estimates and their SEs.
- `marginal_reliability`: from the test information function.
- `cronbach_alpha`: classical internal-consistency reliability.
- `n_items`, `n_persons`: sample descriptors when available.

Examples

```
set.seed(123)
sim <- sim_irt(n_people = 500,
               item_structure = list(list(model = "2PL", n_items = 12)))
fit <- binary_irt(sim$resp, model = "2PL", method = "EM",
                  control = list(max_iter = 15, verbose = FALSE))

reliability(person_params = fit$person_params,
            data = sim$resp,
            item_params = fit$item_params)
```

`score_table` *Summed-Score to Theta Conversion Table*

Description

Generates a raw-score (summed-score) to scale-score conversion table, mapping every possible integer summed score to its corresponding ability estimate and standard error. Such tables are standard in operational testing programs because they let each examinee's number-correct (or number-of-points) score be translated directly into a theta estimate without needing the full response pattern. Three scoring methods are supported: expected a posteriori (`"EAP"`), weighted likelihood (`"WLE"`), and maximum likelihood (`"MLE"`).

Usage

```
score_table(
  item_params,
  method = c("EAP", "WLE", "MLE"),
  prior_mean = 0,
  prior_sd = 1,
```

```
  theta_range = c(-4, 4),
  n_points = 81,
  model = NULL,
  D = 1
)
```

Arguments

`item_params`	A data frame of item parameters (the `item_params` element from `binary_irt`, `polytomous_irt`, or `mixed_irt`). All models in the package are supported.
`method`	Character. Scoring method: `"EAP"` (default), `"WLE"`, or `"MLE"`.
`prior_mean, prior_sd`	Mean and standard deviation of the normal population prior used for the EAP method and for the quadrature weighting (defaults `0` and `1`).
`theta_range`	Numeric vector of length 2 giving the ability grid bounds (default `c(-4, 4)`).
`n_points`	Integer. Number of ability grid points used to evaluate the summed-score distributions (default `81`).
`model`	Optional model override (see `item_info`).
`D`	Scaling constant for the dichotomous logistic metric (default `1`).

Details

The distribution of the summed score at each ability grid point is obtained with the Lord-Wingersky (1984) recursion, which works for any mixture of dichotomous and polytomous items. The `"EAP"` estimate is the mean of the posterior of theta given the summed score, with `se` equal to the posterior standard deviation. The `"MLE"` estimate maximizes the summed-score likelihood; the `"WLE"` estimate (Warm, 1989) maximizes the summed-score likelihood weighted by the square root of the test information, which reduces the bias of the MLE. At the two extreme summed scores the MLE is not finite and is reported at the boundary of `theta_range` with a missing standard error; the WLE remains finite there and is estimated normally.

Value

A data frame with one row per possible summed score and the columns:

- `summed_score`: the integer summed score (0 to the maximum possible score).
- `theta`: the ability estimate for that summed score.
- `se`: the standard error of the ability estimate.

References

Lord, F. M., & Wingersky, M. S. (1984). Comparison of IRT true-score and equipercentile observed-score equatings. *Applied Psychological Measurement, 8*(4), 453-461.

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

Examples

```
set.seed(123)
sim <- sim_irt(n_people = 400,
               item_structure = list(list(model = "2PL", n_items = 10)))
fit <- binary_irt(sim$resp, model = "2PL", method = "EM",
                  control = list(max_iter = 15, verbose = FALSE))

# EAP conversion table (0 to 10 correct)
score_table(fit$item_params, method = "EAP")


# Maximum-likelihood conversion table
score_table(fit$item_params, method = "MLE")
```

`sim_irt` *Simulate Item Response Theory Data*

Description

Simulate item responses data. Support both dichotomous and polytomous responses. Provide an easy implementation with a few default settings.

Usage

```
sim_irt(
  n_people = 1000,
  item_structure = list(),
  theta = NULL,
  theta_mean = 0,
  theta_sd = 1
)
```

Arguments

`n_people`	Integer. Number of students.
`item_structure`	List of lists defining item blocks.
`theta`	Numeric vector (Optional). If provided, these exact ability values are used.
`theta_mean`	Numeric. Mean of latent trait (used if theta is NULL).
`theta_sd`	Numeric. SD of latent trait (used if theta is NULL).

Value

A list containing:

`resp`	data.frame of responses (rows=people, cols=items)
`true_params`	data.frame of true item parameters
`theta`	vector of true latent traits

Examples

```
# 1. Define the Test Blueprint
# We want:
# - 10 items using 2PL (medium difficulty)
# - 5 items using 3PL (difficult, with guessing)
# - 5 items using GPCM (4-point Likert scale)
# - 5 items using GRM (5-point Likert scale)

my_test_structure <- list(
  # Block 1: 2PL
  list(model = "2PL", n_items = 10, a = c(0.8, 1.2), b = c(-1, 1)),

  # Block 2: 3PL (Harder items, b from 1 to 2.5, fixing guessing at 0.2)
  list(model = "3PL", n_items = 5, a = c(1.0, 1.5), b = c(1.0, 2.5), c = 0.2),

  # Block 3: GPCM (Polytomous, 4 categories 0-3)
  list(model = "GPCM", n_items = 5, categories = 4, a = c(0.7, 1.3), b = c(-1, 1)),

  # Block 4: GRM (Polytomous, 5 categories 0-4)
  list(model = "GRM", n_items = 5, categories = 5, a = c(1.0, 2.0))
)

# 2. Run the Simulation
# Define N and a specific Theta vector
N <- 2000
theta_vec <- rnorm(N, 0, 2)

sim_data <- sim_irt(
  n_people = N,
  theta = theta_vec,
  item_structure = my_test_structure
)

# 3. Inspect the Output
# The Response Matrix
head(sim_data$resp)

# The True Parameters (Useful for recovery studies)
# Note how it aligns a, b, and threshold parameters (step_1, step_2...)
head(sim_data$true_params)
```

---

sim_mirt *Simulate Multidimensional Item Response Theory Data*

Description

Simulates responses from compensatory multidimensional item response models, the multidimensional counterpart of `sim_irt`. It supports the multidimensional binary models estimated by `mirt_binary` (M-Rasch, M-2PL, M-3PL) as well as multidimensional polytomous models (multidimensional GPCM and GRM). Each block of items may load on any subset of the latent dimensions, so both simple-structure and complex-structure designs are easy to generate.

Usage

```
sim_mirt(
  n_people = 1000,
  item_structure = list(),
  dimension = 2,
  theta = NULL,
  theta_mean = 0,
  Sigma = NULL
)
```

Arguments

`n_people`	Integer. Number of examinees.
`item_structure`	List of lists defining item blocks (see Details).
`dimension`	Integer. Number of latent dimensions (D).
`theta`	Numeric matrix (Optional). An `n_people` x `dimension` matrix of abilities. If supplied it is used directly.
`theta_mean`	Numeric. Mean of the latent traits, recycled across dimensions (used when `theta` is `NULL`).
`Sigma`	Numeric matrix (Optional). A `dimension` x `dimension` covariance matrix for the latent traits (used when `theta` is `NULL`). Defaults to the identity matrix (uncorrelated dimensions).

Details

Each element of `item_structure` is a list describing one block:

- `model`: one of `"M2PL"`, `"M3PL"`, `"MRasch"` (dichotomous) or `"MGPCM"`, `"MGRM"` (polytomous). The non-prefixed names (`"2PL"`, `"GRM"`, ...) are also accepted.
- `n_items`: number of items in the block.
- `dims`: integer vector of the dimensions the block loads on (default: all dimensions). Slopes on the remaining dimensions are 0.
- `a`: discrimination/slope, given as a range `c(lo, hi)` to sample from, or a single fixed value (default `c(0.8, 1.8)`).
- `d`: intercept, given as a range or a fixed value (default `c(-1.5, 1.5)`). Note the compensatory model uses the intercept metric $z = a'\theta + d$, matching `mirt_binary()`.
- `c`: lower asymptote for `"M3PL"` (default `0.2`).
- `categories`: number of categories for polytomous blocks (default `3`).

Value

A list containing:

`resp`	data.frame of responses (rows = people, cols = items).
`true_params`	data.frame of true item parameters, including one `a_Dim` column per dimension, the intercept `d`, guessing, and step/threshold columns for polytomous items.
`theta`	the `n_people` x `dimension` matrix of true abilities.

See Also

mirt_binary, sim_irt

Examples

```
# Two correlated dimensions, simple structure
set.seed(2025)
Sigma <- matrix(c(1, 0.4, 0.4, 1), 2, 2)

design <- list(
  list(model = "M2PL", n_items = 8, dims = 1),          # loads on Dim 1
  list(model = "M2PL", n_items = 8, dims = 2),          # loads on Dim 2
  list(model = "M3PL", n_items = 4, dims = c(1, 2), c = 0.2)  # both dims
)

sim <- sim_mirt(n_people = 500, item_structure = design,
                dimension = 2, Sigma = Sigma)

head(sim$resp)
head(sim$true_params)


# Recover with the multidimensional estimator
Q <- as.matrix(sim$true_params[, c("a_Dim1", "a_Dim2")] != 0) * 1
fit <- mirt_binary(sim$resp, model = "2PL", dimension = 2,
                   control = list(Q_matrix = Q, max_iter = 10, verbose = FALSE))
head(fit$item_params)
```

---

sim_tirt *Simulate Mixed IRT and Testlet (TRT) Data*

Description

Simulates a data set that mixes ordinary independent item response theory items with testlet (locally dependent) items in a single test form, the simulation counterpart of irt_trt. Independent blocks depend only on the primary trait (theta); testlet blocks additionally depend on a testlet-specific nuisance effect (gamma). Both dichotomous and polytomous formats are supported for the independent and the testlet parts, so the whole family of models handled by irt_trt can be generated at once.

Usage

```
sim_tirt(
  n_people = 1000,
  item_structure = list(),
  theta = NULL,
  theta_mean = 0,
  theta_sd = 1
)
```

Arguments

`n_people`	Integer. Number of examinees.
`item_structure`	List of lists defining item blocks (see Details).
`theta`	Numeric vector (Optional). If provided, these exact ability values are used.
`theta_mean`	Numeric. Mean of the latent trait (used if `theta` is `NULL`).
`theta_sd`	Numeric. SD of the latent trait (used if `theta` is `NULL`).

Details

Each element of `item_structure` is a list describing one block. The `model` name determines whether the block is independent or a testlet:

- Independent (theta only): `"Rasch"`, `"2PL"`, `"3PL"`, `"PCM"`, `"GPCM"`, `"GRM"`.
- Testlet (theta + gamma): `"RaschT"`, `"2PLT"`, `"3PLT"`, `"BiFT"`, `"PCT"`, `"GPCT"`, `"GRT"`.

Other recognized keys are `n_items`, `categories` (polytomous), `a`, `b`, `c` (given as a fixed value or a `c(lo, hi)` range to sample from), `s` (testlet loading for `"BiFT"`), `testlet_id` (the testlet label, required for testlet blocks), `testlet_var` (variance of the gamma effect), and `gamma_vector` (a user-supplied gamma effect of length `n_people`).

Value

A list containing:

`resp` data.frame of responses (rows = people, cols = items).

`true_item_params`
data.frame of true item parameters, including a `model` column and a `testlet` column (`NA` for independent items). Its `item_id`, `model`, and `testlet` columns provide everything needed to build the `item_spec` argument of `irt_trt` (rename `item_id` to `item`).

`true_person_params`
data.frame of true person parameters: the primary ability plus one gamma column per testlet.

See Also

`irt_trt`, `sim_trt`, `sim_irt`

Examples

```
# A form with independent items and two testlets
set.seed(2025)
design <- list(
  list(model = "2PL",  n_items = 8),                        # independent
  list(model = "GRM",  n_items = 4, categories = 3),        # independent poly
  list(model = "2PLT", n_items = 4, testlet_id = "P1",
       testlet_var = 0.6),                                  # testlet
  list(model = "GPCT", n_items = 3, categories = 3,
       testlet_id = "P2", testlet_var = 0.5)                # testlet poly
```

```
)

sim <- sim_tirt(n_people = 600, item_structure = design)

head(sim$resp)
sim$true_item_params[, c("item_id", "model", "testlet")]
head(sim$true_person_params)


# Feed the true structure straight into the joint estimator
spec <- data.frame(
  item = sim$true_item_params$item_id,
  model = sim$true_item_params$model,
  testlet = sim$true_item_params$testlet,
  stringsAsFactors = FALSE
)
fit <- irt_trt(sim$resp, spec, method = "EM",
               control = list(max_iter = 15, verbose = FALSE))
head(fit$item_params)
```

---

sim_trt *Simulate Testlet Response Theory Data (Vector Supported Version)*

Description

Simulate testlet responses data. Support both dichotomous and polytomous responses. Provide an easy implementation with a few default settings.

Usage

```
sim_trt(
  n_people = 1000,
  item_structure = list(),
  theta = NULL,
  theta_mean = 0,
  theta_sd = 1
)
```

Arguments

n_people	Integer. Number of examinees.
item_structure	List of lists defining item blocks.
theta	Numeric vector (Optional). If provided, these exact ability values are used.
theta_mean	Numeric. Mean of latent trait (used if theta is NULL).
theta_sd	Numeric. SD of latent trait (used if theta is NULL).

Value

A list containing:

`resp` data.frame of responses (rows=people, cols=items)

`true_item_params` data.frame of true item parameters

`true_person_params` vector of true latent traits

Examples

```
# ==========================================================================
# Example 1: Complex Testlet Design
# ==========================================================================
# Define the Testlet Blueprint
trt_design <- list(
  # Testlet 1: Rasch Testlet Model (High dependence: var=0.8)
  list(model = "RaschT", n_items = 5, testlet_id = "Read_A",
       testlet_var = 0.8, b = c(-1, 1)),

  # Testlet 2: 2PL Testlet Model (Default dependence: var=0.5)
  list(model = "2PLT", n_items = 5, testlet_id = "Read_B",
       a = c(0.7, 1.3)),

  # Testlet 3: Graded Response Testlet (Polytomous, 4 categories)
  list(model = "GRT", n_items = 4, testlet_id = "Survey",
       categories = 4, testlet_var = 0.2)
)

# Run Simulation
trt_data <- sim_trt(n_people = 500, item_structure = trt_design)

# Inspect Results
# 1. Responses
head(trt_data$resp)

# 2. Item Parameters
# (Notice 'testlet_loading' equals 'discrimination' for standard models)
head(trt_data$true_item_params)

# 3. Person Parameters (Ability + Gamma for each testlet)
head(trt_data$true_person_params)

# ==========================================================================
# Example 2: Manual Control (Theta, Gamma, and Parameters)
# ==========================================================================

# 1. Manual Theta (e.g., everyone has high ability)
manual_theta <- rep(2.0, 100)

# 2. Manual Gamma (e.g., zero effect for T1)
manual_gamma <- rep(0, 100)
```

```
# 3. Item Parameters: Exact Match vs Range Sampling
custom_structure <- list(
  # Case A: Manual Gamma Vector
  list(model = "2PLT", n_items = 5, testlet_id = "T1",
       gamma_vector = manual_gamma),

  # Case B: Exact Parameter Match (Length of 'a' equals n_items)
  list(model = "2PLT", n_items = 2, testlet_id = "T2",
       a = c(0.5, 2.5)),

  # Case C: Range Sampling (Length of 'a' is 2, but n_items != 2)
  list(model = "2PLT", n_items = 5, testlet_id = "T3",
       a = c(0.5, 2.5))
)

res_custom <- sim_trt(n_people = 100, theta = manual_theta,
                      item_structure = custom_structure)

# Verify Manual Theta
print(mean(res_custom$true_person_params$ability)) # Should be 2.0

# Verify Manual Gamma (T1 should be 0)
print(head(res_custom$true_person_params$testlet_T1))

# Verify Exact Match (T2 discrimination should be 0.5 and 2.5)
print(res_custom$true_item_params[res_custom$true_item_params$testlet_id == "T2",
                                  "discrimination"])
```

---

tcc *Test and Item Characteristic Curves (Expected Scores)*

Description

Computes item characteristic curves (the expected score of each item as a function of ability) and the test characteristic curve (the expected total score as a function of ability). The test characteristic curve maps the latent ability scale onto the number-correct (true-score) scale and underlies true-score equating and score reporting; the item curves show how each item's expected score rises across the trait.

Usage

```
tcc(item_params, theta = seq(-4, 4, by = 0.1), model = NULL, D = 1)
```

Arguments

`item_params`	A data frame of item parameters (the `item_params` element from `binary_irt`, `polytomous_irt`, or `mixed_irt`). All models in the package are supported.
`theta`	A numeric vector of ability values at which to evaluate the curves (default = `seq(-4, 4, by = 0.1)`).

model	Optional model override (see item_info).
D	Scaling constant for the dichotomous logistic metric (default 1).

Value

A list with two elements:

- test_curve: a data frame with columns theta and expected_score (the test characteristic curve).
- item_curves: a numeric matrix with one row per item and one column per theta value, giving the expected score of each item.

See Also

item_info, score_table

Examples

```
set.seed(123)
sim <- sim_irt(n_people = 400,
               item_structure = list(list(model = "2PL", n_items = 10)))
fit <- binary_irt(sim$resp, model = "2PL", method = "EM",
                  control = list(max_iter = 15, verbose = FALSE))

curves <- tcc(fit$item_params, theta = seq(-3, 3, by = 0.5))
curves$test_curve

# Expected score on the first item across ability
curves$item_curves[1, ]
```

---

test_info *Test Information Function and Conditional Standard Error*

Description

Computes the test information function (TIF) as the sum of all item information values at each ability point, together with the conditional standard error of measurement (SEM). The test information function summarizes where along the ability scale the test measures most precisely, and the conditional SEM ($1/\sqrt{TIF}$) translates that precision back onto the theta metric. This is a routine part of test evaluation and form assembly in operational testing programs.

Usage

```
test_info(item_params, theta = seq(-4, 4, by = 0.1), model = NULL, D = 1)
```

Arguments

item_params	A data frame of item parameters (see item_info).
theta	A numeric vector of ability values at which to evaluate the test information (default = seq(-4, 4, by = 0.1)).
model	Optional model override (see item_info).
D	Scaling constant for the dichotomous logistic metric (default 1).

Value

A data frame with one row per theta point and the columns:

- theta: the ability value.
- test_info: the test information (sum of item information).
- sem: the conditional standard error of measurement, $1/\sqrt{test\_info}$.
- reliability: the marginal-style conditional reliability $TIF/(TIF + 1)$ for a standard-normal ability scale.

See Also

item_info

Examples

```
set.seed(123)
sim <- sim_irt(n_people = 400,
               item_structure = list(list(model = "2PL", n_items = 10)))
fit <- binary_irt(sim$resp, model = "2PL", method = "EM",
                  control = list(max_iter = 15, verbose = FALSE))

tif <- test_info(fit$item_params, theta = seq(-3, 3, by = 0.5))
print(tif)

# The ability where the test is most informative
tif$theta[which.max(tif$test_info)]
```

---

trt_binary *Unidimensional Binary (Dichotomous) Testlet Response Theory Estimation*

Description

Estimates item and person parameters for Unidimensional Binary (Dichotomous) Testlet response models using Penalized Expectation-Maximization or Joint Maximum Likelihood Estimation with stabilization.

Usage

```
trt_binary(
  data,
  group,
  model = c("RaschT", "2PLT", "3PLT", "BiFT"),
  method = c("EM", "MLE"),
  control = list()
)
```

Arguments

`data`	A data.frame of binary responses (0/1). Rows=persons, Cols=items in testlets.
`group`	A list defining testlet structures. Example: `list(c(1,2,3), c(4,5,6))`.
`model`	Character. One of "RaschT" (Rasch Testlet), "2PLT" (2-Parameter Logistic Testlet), "3PLT" (3-Parameter Logistic Testlet), "BiFT" (Bifactor).
`method`	Character. "EM" (Marginal Maximum Likelihood via Expectation-Maximization) or "MLE" (Joint Maximum Likelihood).
`control`	A `list` of control parameters for the estimation algorithm: • `max_iter`: Maximum number of EM iterations (default = 100). • `converge_tol`: Convergence criterion for parameter change (default = 1e-4). • `theta_range`: Numeric vector of length 2 specifying the integration grid bounds (default = c(-4, 4)). • `quad_points`: Number of quadrature points (default = 21). • `verbose`: Logical; if `TRUE`, prints progress to console.

Value

A list containing:

- `item_params`: Estimated item parameters.
- `person_params`: Estimated person abilities and testlet effects.
- `model_fit`: A data frame containing iterations and fit statistics such as Akaike's Information Criterion (AIC), the Bayesian Information Criterion (BIC), and Log-Likelihood.

Examples

```
# --- Quick Example (small data) ---
set.seed(1)
n <- 50
J <- 4
theta <- rnorm(n)
gamma <- rnorm(n, 0, 0.5)
resp <- matrix(NA, n, J)
colnames(resp) <- paste0("Item_", 1:J)
b <- c(-0.5, 0, 0.5, 1)
for(j in 1:J) {
```

```
  p <- 1 / (1 + exp(-(theta + gamma - b[j])))
  resp[, j] <- rbinom(n, 1, p)
}
res <- trt_binary(
  data = as.data.frame(resp),
  group = list(1:2, 3:4),
  model = "2PLT",
  method = "EM",
  control = list(max_iter = 5, verbose = FALSE)
)
head(res$item_params)


# --- Full Example: Simulation (2PLT) ---
set.seed(2025)
n_persons <- 500
n_testlets <- 3
items_per_testlet <- 3
n_items <- n_testlets * items_per_testlet

# 1. Generate Parameters
# Discrimination (a): Varying -> 2PLT
a_true <- runif(n_items, 0.8, 1.5)
# Difficulty (b)
b_true <- seq(-1, 1, length.out = n_items)
# Testlet Variances (Sigma)
sigma_true <- c(1.0, 1.5, 2.0)

# 2. Generate Person Params
theta_true <- rnorm(n_persons, 0, 1)
gamma_matrix <- matrix(0, nrow = n_persons, ncol = n_testlets)
for(d in 1:n_testlets) {
  gamma_matrix[, d] <- rnorm(n_persons, 0, sigma_true[d])
}

# 3. Generate Responses
resp_matrix <- matrix(0, nrow = n_persons, ncol = n_items)
colnames(resp_matrix) <- paste0("Item_", 1:n_items)
group_list <- list()

idx_counter <- 1
for(d in 1:n_testlets) {
  indices <- idx_counter:(idx_counter + items_per_testlet - 1)
  group_list[[d]] <- indices

  for(i in indices) {
    # 2PLT Model: a * (theta + gamma - b)
    lin <- a_true[i] * (theta_true + gamma_matrix[, d] - b_true[i])
    prob <- 1 / (1 + exp(-lin))
    resp_matrix[, i] <- rbinom(n_persons, 1, prob)
  }
  idx_counter <- idx_counter + items_per_testlet
}
```

```
df_sim <- as.data.frame(resp_matrix)

# 4. Run Estimation
# We use "2PLT" because data was generated with varying 'a'
res <- trt_binary(
  data = df_sim,
  group = group_list,
  model = "2PLT",
  method = "EM",
  control = list(max_iter = 20, verbose = FALSE)
)

head(res$item_params)
head(res$person_params)
```

---

trt_poly *Unidimensional Polytomous Testlet Response Theory Estimation*

Description

Estimates item and person parameters for Polytomous Testlet models using Robust Newton-Raphson optimization.

Usage

```
trt_poly(
  data,
  group,
  model = c("GRT", "PCMT", "BiFT"),
  method = c("MLE", "EM"),
  control = list()
)
```

Arguments

`data`	A data.frame of polytomous responses. Rows=persons, Cols=items in testlets.
`group`	A list defining testlet structures. Example: `list(c(1,2,3), c(4,5,6))`.
`model`	Character. "GRT" (Graded Response Model), "PCMT" (Partial Credit Model for Testlet), or "BiFT" (Biffactor).
`method`	Character. "EM" (Marginal Maximum Likelihood via Expectation-Maximization) or "MLE" (Joint Maximum Likelihood).
`control`	A `list` of control parameters for the estimation algorithm: • `max_iter`: Maximum number of EM iterations (default = 100). • `converge_tol`: Convergence criterion for parameter change (default = 1e-4).

- `theta_range`: Numeric vector of length 2 specifying the integration grid bounds (default = c(-4, 4)).
- `quad_points`: Number of quadrature points (default = 21).
- `verbose`: Logical; if `TRUE`, prints progress to console.

Value

A list containing:

- `item_params`: A data frame of estimated item parameters.
- `person_params`: A data frame of estimated person abilities and testlet effects .
- `model_fit`: A data frame containing iterations and fit statistics such as Akaike's Information Criterion (AIC), the Bayesian Information Criterion (BIC), and Log-Likelihood.

Examples

```
# --- Example: Simulation (Mixed Categories GRT) ---
set.seed(42)
N <- 500; J <- 16

# Define Groups (4 Testlets)
groups <- list(c(1:4), c(5:8), c(9:12), c(13:16))

# Define Categories (Binary, 3-cat, 4-cat, Mixed)
# Items 1-4: 2 cats; 5-8: 3 cats; 9-12: 4 cats; 13-16: mixed
cats <- c(rep(2, 4), rep(3, 4), rep(4, 4), 3, 5, 3, 5)

# 1. Generate Parameters
theta <- rnorm(N)
# Gamma for 4 testlets (SD = 0.8)
gamma <- matrix(rnorm(N * 4, 0, 0.8), N, 4)

a <- rlnorm(J, 0, 0.2)
b_list <- vector("list", J)

# Generate Thresholds based on category count
for(j in 1:J) {
  n_thresh <- cats[j] - 1
  if(n_thresh == 1) {
    b_list[[j]] <- rnorm(1)
  } else {
    # Spread thresholds
    b_list[[j]] <- sort(rnorm(1) + seq(-1, 1, length.out=n_thresh))
  }
}

# 2. Generate Responses (GRT Logic)
resp <- matrix(NA, N, J)
colnames(resp) <- paste0("Item_", 1:J)

for(i in 1:N) {
```

```
  for(j in 1:J) {
    # Identify Testlet ID
    tid <- which(sapply(groups, function(x) j %in% x))
    eff <- theta[i] + gamma[i, tid]

    # Calculate Probabilities (Graded Response)
    K <- cats[j]
    probs <- numeric(K)
    P_prev <- 1
    for(k in 1:(K-1)) {
      term <- a[j] * (eff - b_list[[j]][k])
      P_star <- 1 / (1 + exp(-term))
      probs[k] <- P_prev - P_star
      P_prev <- P_star
    }
    probs[K] <- P_prev

    # Sample Response
    resp[i, j] <- sample(0:(K-1), 1, prob = probs)
  }
}
df_sim <- as.data.frame(resp)

# 3. Run Estimation
fit <- trt_poly(
  data = df_sim,
  group = groups,
  model = "GRT",
  method = "EM",
  control = list(max_iter = 20, verbose = FALSE)
)

head(fit$item_params)
head(fit$person_params)
```

Index

∗ **datasets**
ela1, 6
ela2, 7
ela3, 8
ela3_testmap, 8

binary_irt, 3, *19*, *21, 22*, *30*, *36*, *44*

dif, 5

ela1, 6
ela2, 7
ela3, 8
ela3_testmap, 8
equate_irt, 9

fix_person, 14
fixed_item, 12

irt_trt, 16, *40, 41*
item_fit, 19
item_info, *20*, 20, *23*, *30*, *34*, *36*, *45, 46*

ld_stats, 22

mirt_binary, 23, *38*, *40*
mixed_irt, *19*, *21, 22*, 25, *30*, *36*, *44*
mixture_irt, 28

person_fit, 29
polytomous_irt, *19*, *21, 22*, *30*, 31, *36*, *44*

reliability, 34

score_table, 35, *45*
sim_irt, 37, *38*, *40, 41*
sim_mirt, 38
sim_tirt, 40
sim_trt, *41*, 42

tcc, 44
test_info, *20, 21*, 45
trt_binary, 46
trt_poly, 49