Package 'tirt'

May 8, 2026

Title Testlet Item Response Theory

Version 0.3.1

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.

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 2026-03-24 08:40:08 UTC

Contents

`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)
```

`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

```
# --- 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)
```

`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)
```

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

```
# --- 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)
```

`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_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"])
```

`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

```
# --- 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