

# Package: irtsim (via r-universe)

June 2, 2026

**Title** Monte Carlo Simulation-Based Sample-Size Planning for Item Response Theory

**Version** 0.1.2.9000

**Description** Provides a pipeline application programming interface (API) for Monte Carlo simulation-based sample-size planning in item response theory (IRT). Implements the 10-decision framework from Schroeders and Gnambs (2025)  [<doi:10.1177/25152459251314798 >](https://doi.org/10.1177/25152459251314798) as a three-step workflow: specify the data-generating model with `irt_design()`, add study conditions with `irt_study()`, and run simulations with `irt_simulate()`. Supports one-parameter logistic (1PL), two-parameter logistic (2PL), and graded response models with missing-completely-at-random (MCAR), missing-at-random (MAR), booklet, and linking missingness mechanisms. Results include mean squared error (MSE), bias, root mean squared error (RMSE), standard error (SE), and coverage criteria with summary and plot methods.

**License** GPL (>= 3)

**URL** <https://sward1.github.io/irsim/>, <https://github.com/sward1/irsim>

**BugReports** <https://github.com/sward1/irsim/issues>

**Depends** R (>= 4.1.0)

**Imports** cli, future.apply, ggplot2, mirt, rlang

**Suggests** future, knitr, R.rsp (>= 0.46.0), rmarkdown, scales, testthat (>= 3.0.0)

**VignetteBuilder** R.rsp, knitr

**Config/testthat/edition** 3

**Encoding** UTF-8

**Language** en-US

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.3

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

**Repository** <https://sward1.r-universe.dev>

**Date/Publication** 2026-06-02 12:46:14 UTC

**RemoteUrl** <https://github.com/sward1/irtsim>

**RemoteRef** HEAD

**RemoteSha** e5d368412341da5550fa46d838f0d3d60b3a3408

## Contents

irt_design . . . . .	2
irt_iterations . . . . .	4
irt_params_1pl . . . . .	5
irt_params_2pl . . . . .	6
irt_params_3pl . . . . .	7
irt_params_gpcm . . . . .	9
irt_params_grm . . . . .	11
irt_params_pcm . . . . .	12
irt_simulate . . . . .	13
irt_study . . . . .	16
plot.irt_results . . . . .	17
plot.summary_irt_results . . . . .	18
print.irt_design . . . . .	19
print.irt_results . . . . .	20
print.irt_study . . . . .	21
print.summary_irt_results . . . . .	21
recommended_n . . . . .	22
summary.irt_results . . . . .	24
<b>Index</b>	<b>27</b>

---

irt\_design

*Create an IRT Design Specification*

---

### Description

Define the data-generating model for an IRT simulation study. This captures decisions 1–3 from the Schroeders & Gnamb (2025) framework: dimensionality, item parameters, and item type.

### Usage

```
irt_design(model, n_items, item_params, theta_dist = "normal", n_factors = 1L)
```

**Arguments**

model	Character string specifying the IRT model. One of "1PL", "2PL", "3PL", "GRM", "PCM", or "GPCM". The canonical list is registered in <a href="#">get_model_config()</a> .
n_items	Positive integer. Number of items in the instrument.
item_params	A named list of item parameters. Contents depend on model: <ul style="list-style-type: none"> <li><b>1PL</b> b (numeric vector of length n_items). Discrimination is fixed at 1 for all items and added automatically.</li> <li><b>2PL</b> a (discrimination, positive numeric vector or matrix) and b (difficulty, numeric vector), each of length n_items.</li> <li><b>3PL</b> a, b, and c (guessing parameter, numeric vector with values in [0, 1]), each of length n_items.</li> <li><b>GRM</b> a (discrimination, positive numeric vector) of length n_items and b (threshold matrix, n_items rows by n_categories - 1 columns; thresholds ordered within row).</li> <li><b>PCM</b> a (numeric vector, all 1 — Rasch family) of length n_items and b (step matrix, n_items rows by n_categories - 1 columns; steps NOT required to be ordered within row).</li> <li><b>GPCM</b> a (positive numeric vector) of length n_items and b (step matrix, same shape as PCM; steps NOT required to be ordered within row).</li> </ul> See <a href="#">irt_params_1pl()</a> , <a href="#">irt_params_2pl()</a> , <a href="#">irt_params_3pl()</a> , <a href="#">irt_params_grm()</a> , <a href="#">irt_params_pcm()</a> , and <a href="#">irt_params_gpcm()</a> for helpers that generate item_params lists matching each schema.
theta_dist	Either a character string ("normal" or "uniform") or a function that takes a single argument n and returns a numeric vector of length n. Defaults to "normal".
n_factors	Positive integer specifying the number of latent factors. Defaults to 1L. Currently only n_factors = 1 is supported; multidimensional IRT (n_factors > 1) is planned for v0.4.0. Passing any value other than 1 raises an error rather than silently propagating an unsupported design to the estimator.

**Value**

An S3 object of class `irt_design` (a named list) with elements `model`, `n_items`, `item_params`, `theta_dist`, and `n_factors`.

**See Also**

[irt\\_study\(\)](#) to add study conditions, [irt\\_params\\_2pl\(\)](#) and [irt\\_params\\_grm\(\)](#) to generate item parameters.

**Examples**

```
# 1PL (Rasch) design with 20 items
design_1pl <- irt_design(
  model = "1PL",
  n_items = 20,
  item_params = list(b = seq(-2, 2, length.out = 20))
```

```

)

# 2PL design
design_2pl <- irt_design(
  model = "2PL",
  n_items = 30,
  item_params = list(
    a = rlnorm(30, 0, 0.25),
    b = seq(-2, 2, length.out = 30)
  )
)

```

---

irt\_iterations      *Compute Required Monte Carlo Replications*

---

### Description

Uses the Burton (2003) formula to determine the minimum number of simulation replications needed to achieve a desired level of Monte Carlo precision.

### Usage

```
irt_iterations(sigma, delta, alpha = 0.05)
```

### Arguments

sigma	Positive numeric. The empirical standard error of the estimand across replications (or a pilot estimate thereof).
delta	Positive numeric. The acceptable Monte Carlo error (half-width of the MC confidence interval for the estimand).
alpha	Numeric in (0, 1). Two-sided significance level. Default 0.05 (i.e., 95 percent MC confidence).

### Details

The formula is:

$$R = \lceil (z_{\alpha/2} \cdot \sigma / \delta)^2 \rceil$$

where  $\sigma$  is the empirical standard error of the estimand,  $\delta$  is the acceptable Monte Carlo error, and  $z_{\alpha/2}$  is the critical value for the desired confidence level.

### Value

An integer: the minimum number of replications.

## References

Burton, A., Altman, D. G., Royston, P., & Holder, R. L. (2006). The design of simulation studies in medical statistics. *Statistics in Medicine*, 25(24), 4279–4292. doi:10.1002/sim.2673

## See Also

`irt_simulate()` for running the simulation with the computed number of replications.

## Examples

```
# How many replications for MC SE of bias < 0.1
# when empirical SE of the estimand is 0.5?
irt_iterations(sigma = 0.5, delta = 0.1)

# Tighter tolerance with 99% MC confidence
irt_iterations(sigma = 0.5, delta = 0.05, alpha = 0.01)
```

---

irt_params_1pl	<i>Generate IPL Item Parameters</i>
----------------	-------------------------------------

---

## Description

Creates a list of difficulty (b) parameters suitable for passing to `irt_design()` with `model = "1PL"`. The 1PL model is Rasch-family: every item shares the same discrimination (fixed at 1), so only b is generated here — the `a = 1` constraint is applied downstream in the design's `validate_params` step.

## Usage

```
irt_params_1pl(
  n_items,
  b_dist = "normal",
  b_mean = 0,
  b_sd = 1,
  b_range = c(-2, 2),
  seed = NULL
)
```

## Arguments

<code>n_items</code>	Positive integer. Number of items.
<code>b_dist</code>	Character string for the difficulty distribution. One of "normal" or "even". Default: "normal".
<code>b_mean</code>	Numeric. Mean of the normal distribution for b. Only used when <code>b_dist = "normal"</code> . Default: 0.
<code>b_sd</code>	Numeric. SD of the normal distribution for b. Only used when <code>b_dist = "normal"</code> . Default: 1.

b_range	Numeric vector of length 2. Range for evenly-spaced b values. Only used when b_dist = "even". Default: c(-2, 2).
seed	Optional integer seed for reproducibility. If NULL (default), the current RNG state is used.

**Value**

A named list with a single element b (numeric vector of length n\_items). Note: no a is returned — 1PL fixes discrimination at 1 downstream rather than at generation time.

**See Also**

[irt\\_params\\_2pl\(\)](#) for the free-discrimination binary alternative, [irt\\_design\(\)](#) to use the generated parameters.

**Examples**

```
# Default 1PL parameters for 30 items
params <- irt_params_1pl(n_items = 30, seed = 42)

# Evenly-spaced difficulty across a wider range
params <- irt_params_1pl(n_items = 20, b_dist = "even", b_range = c(-3, 3))
```

---

irt_params_2pl	<i>Generate 2PL Item Parameters</i>
----------------	-------------------------------------

---

**Description**

Creates a list of discrimination (a) and difficulty (b) parameters suitable for passing to [irt\\_design\(\)](#).

**Usage**

```
irt_params_2pl(
  n_items,
  a_dist = "lnorm",
  a_mean = 0,
  a_sd = 0.25,
  b_dist = "normal",
  b_mean = 0,
  b_sd = 1,
  b_range = c(-2, 2),
  seed = NULL
)
```

**Arguments**

n_items	Positive integer. Number of items.
a_dist	Character string for the discrimination distribution. Currently only "lnorm" (log-normal) is supported. Default: "lnorm".
a_mean	Numeric. Mean of the log-normal distribution for a (i.e., meanlog). Default: 0.
a_sd	Numeric. SD of the log-normal distribution for a (i.e., sdlog). Default: 0.25.
b_dist	Character string for the difficulty distribution. One of "normal" or "even". Default: "normal".
b_mean	Numeric. Mean of the normal distribution for b. Only used when b_dist = "normal". Default: 0.
b_sd	Numeric. SD of the normal distribution for b. Only used when b_dist = "normal". Default: 1.
b_range	Numeric vector of length 2. Range for evenly-spaced b values. Only used when b_dist = "even". Default: c(-2, 2).
seed	Optional integer seed for reproducibility. If NULL (default), the current RNG state is used.

**Value**

A named list with elements a (numeric vector) and b (numeric vector), each of length n\_items.

**See Also**

[irt\\_params\\_grm\(\)](#) for GRM parameters, [irt\\_design\(\)](#) to use the generated parameters.

**Examples**

```
# Default 2PL parameters for 30 items
params <- irt_params_2pl(n_items = 30, seed = 42)

# Evenly-spaced difficulty
params <- irt_params_2pl(n_items = 20, b_dist = "even", b_range = c(-3, 3))
```

---

 irt\_params\_3pl

*Generate 3PL Item Parameters*


---

**Description**

Creates a list of discrimination (a), difficulty (b), and guessing (c) parameters suitable for passing to [irt\\_design\(\)](#) with model = "3PL".

**Usage**

```
irt_params_3pl(
  n_items,
  a_dist = "lnorm",
  a_mean = 0,
  a_sd = 0.25,
  b_dist = "normal",
  b_mean = 0,
  b_sd = 1,
  b_range = c(-2, 2),
  c_shape1 = 5,
  c_shape2 = 17,
  seed = NULL
)
```

**Arguments**

<code>n_items</code>	Positive integer. Number of items.
<code>a_dist</code>	Character string for the discrimination distribution. Currently only "lnorm" (log-normal) is supported. Default: "lnorm".
<code>a_mean</code>	Numeric. meanlog for the log-normal distribution. Default: 0.
<code>a_sd</code>	Numeric. sdlog for the log-normal distribution. Default: 0.25.
<code>b_dist</code>	Character string for the difficulty distribution. One of "normal" or "even". Default: "normal".
<code>b_mean</code>	Numeric. Mean of the normal distribution for b. Only used when <code>b_dist = "normal"</code> . Default: 0.
<code>b_sd</code>	Numeric. SD of the normal distribution for b. Only used when <code>b_dist = "normal"</code> . Default: 1.
<code>b_range</code>	Numeric vector of length 2. Range for evenly-spaced b values. Only used when <code>b_dist = "even"</code> . Default: <code>c(-2, 2)</code> .
<code>c_shape1</code>	Positive numeric. First shape parameter of the Beta distribution used to generate c. Default: 5.
<code>c_shape2</code>	Positive numeric. Second shape parameter. Default: 17. The default Beta(5, 17) has $E[c] \approx 0.227$ , $SD \approx 0.087$ , consistent with typical four-option multiple-choice items.
<code>seed</code>	Optional integer seed for reproducibility. If NULL (default), the current RNG state is used.

**Value**

A named list with elements a, b, c, each a numeric vector of length `n_items`.

**See Also**

[irt\\_params\\_2pl\(\)](#), [irt\\_params\\_grm\(\)](#), [irt\\_design\(\)](#).

**Examples**

```
# Default 3PL parameters for 30 items
params <- irt_params_3pl(n_items = 30, seed = 42)

# Custom guessing distribution (e.g., 5-option items, lower chance level)
params <- irt_params_3pl(
  n_items = 30, c_shape1 = 4, c_shape2 = 16, seed = 42
)
```

---

irt\_params\_gpcm      *Generate GPCM Item Parameters*

---

**Description**

Creates a list of discrimination (a) and step (b) parameters suitable for passing to `irt_design()` with `model = "GPCM"`.

**Usage**

```
irt_params_gpcm(
  n_items,
  n_categories,
  a_dist = "lnorm",
  a_mean = 0,
  a_sd = 0.25,
  b_dist = "normal",
  b_mean = 0,
  b_sd = 1,
  b_range = c(-2, 2),
  step_dispersion = 1,
  seed = NULL
)
```

**Arguments**

<code>n_items</code>	Positive integer. Number of items.
<code>n_categories</code>	Positive integer $\geq 2$ . Number of response categories per item. Produces <code>n_categories - 1</code> step columns in <code>b</code> .
<code>a_dist</code>	Character string for the discrimination distribution. Currently only "lnorm" (log-normal) is supported. Default: "lnorm".
<code>a_mean</code>	Numeric. meanlog for the log-normal distribution. Default: 0.
<code>a_sd</code>	Numeric. sdlog for the log-normal distribution. Default: 0.25.
<code>b_dist</code>	Character string for the item-center distribution: either "normal" (default) or "even".

b_mean	Numeric. Mean of item centers when b_dist = "normal". Default: 0.
b_sd	Numeric. SD of item centers when b_dist = "normal". Default: 1.
b_range	Length-2 numeric vector giving the minimum and maximum item-center values. Only used when b_dist = "even". Default: c(-2, 2).
step_dispersion	Non-negative numeric. SD of the within-item step offsets drawn from $rnorm(0, step\_dispersion)$ and added to each item's center. Default: 1.0. 0 is allowed (all steps within an item equal the item center — degenerate but useful for design exploration).
seed	Optional integer seed for reproducibility.

### Details

The Generalized Partial Credit Model (Muraki, 1992) is partial-credit family — like the Partial Credit Model, step parameters within each item are NOT required to be ordered (the defining contrast with the Graded Response Model). Unlike PCM, GPCM allows per-item discrimination: *a* is a free positive vector rather than fixed at 1. See [irt\\_params\\_pcm\(\)](#) for the Rasch-family alternative.

### Value

A named list with elements:

- a** Positive numeric vector of length `n_items`.
- b** Numeric matrix with `n_items` rows and `n_categories - 1` columns. Steps are NOT sorted within row.

### See Also

[irt\\_params\\_pcm\(\)](#) for the Rasch-family (a fixed at 1) alternative, [irt\\_params\\_grm\(\)](#) for the ordered-threshold polytomous model, [irt\\_design\(\)](#) to use the generated parameters.

### Examples

```
# GPCM parameters: 15 items, 4 response categories
params <- irt_params_gpcm(n_items = 15, n_categories = 4, seed = 42)

# Tighter within-item step spread and a wider discrimination distribution
params <- irt_params_gpcm(
  n_items = 15, n_categories = 4,
  a_sd = 0.50, step_dispersion = 0.5, seed = 42
)
```

---

 irt\_params\_grm      *Generate GRM Item Parameters*


---

### Description

Creates a list of discrimination (a) and threshold (b) parameters suitable for passing to `irt_design()` with `model = "GRM"`.

### Usage

```
irt_params_grm(
  n_items,
  n_categories,
  a_dist = "lnorm",
  a_mean = 0,
  a_sd = 0.25,
  b_mean = 0,
  b_sd = 1,
  seed = NULL
)
```

### Arguments

<code>n_items</code>	Positive integer. Number of items.
<code>n_categories</code>	Positive integer $\geq 2$ . Number of response categories per item. Produces <code>n_categories - 1</code> threshold columns in <code>b</code> .
<code>a_dist</code>	Character string for the discrimination distribution. Currently only "lnorm" is supported. Default: "lnorm".
<code>a_mean</code>	Numeric. meanlog for the log-normal distribution. Default: 0.
<code>a_sd</code>	Numeric. sdlog for the log-normal distribution. Default: 0.25.
<code>b_mean</code>	Numeric. Mean around which thresholds are centered. Default: 0.
<code>b_sd</code>	Numeric. SD of the base threshold distribution. Default: 1.
<code>seed</code>	Optional integer seed for reproducibility.

### Value

A named list with elements:

- a** Numeric vector of length `n_items`.
- b** Numeric matrix with `n_items` rows and `n_categories - 1` columns. Thresholds are ordered within each row.

### See Also

`irt_params_2pl()` for 2PL parameters, `irt_design()` to use the generated parameters.

**Examples**

```
# GRM parameters: 15 items, 5 response categories
params <- irt_params_grm(n_items = 15, n_categories = 5, seed = 42)
```

---

```
irt_params_pcm          Generate PCM Item Parameters
```

---

**Description**

Creates a list of discrimination (a, fixed at 1) and step (b) parameters suitable for passing to `irt_design()` with `model = "PCM"`.

**Usage**

```
irt_params_pcm(
  n_items,
  n_categories,
  b_dist = "normal",
  b_mean = 0,
  b_sd = 1,
  b_range = c(-2, 2),
  step_dispersion = 1,
  seed = NULL
)
```

**Arguments**

<code>n_items</code>	Positive integer. Number of items.
<code>n_categories</code>	Positive integer $\geq 2$ . Number of response categories per item. Produces <code>n_categories - 1</code> step columns in <code>b</code> .
<code>b_dist</code>	Character string for the item-center distribution: either "normal" (default) or "even".
<code>b_mean</code>	Numeric. Mean of item centers when <code>b_dist = "normal"</code> . Default: 0.
<code>b_sd</code>	Numeric. SD of item centers when <code>b_dist = "normal"</code> . Default: 1.
<code>b_range</code>	Length-2 numeric vector giving the minimum and maximum item-center values. Only used when <code>b_dist = "even"</code> . Default: <code>c(-2, 2)</code> .
<code>step_dispersion</code>	Non-negative numeric. SD of the within-item step offsets drawn from <code>rnorm(0, step_dispersion)</code> and added to each item's center. Default: 1.0, consistent with <code>mirt::simdata</code> 's polytomous conventions and the PCM examples in Embretson & Reise (2000). 0 is allowed (all steps within an item equal the item center — degenerate but useful for design exploration).
<code>seed</code>	Optional integer seed for reproducibility.

## Details

The Partial Credit Model (Masters, 1982) is a Rasch-family polytomous model: every item shares the same discrimination (fixed at 1), and the step parameters within each item are NOT required to be ordered. This is the defining contrast with the Graded Response Model — see [irt\\_params\\_grm\(\)](#) for the ordered-threshold alternative.

## Value

A named list with elements:

- a** Numeric vector of length `n_items`, all 1 (Rasch family).
- b** Numeric matrix with `n_items` rows and `n_categories - 1` columns. Steps are NOT sorted within row.

## See Also

[irt\\_params\\_grm\(\)](#) for the ordered-threshold polytomous model, [irt\\_design\(\)](#) to use the generated parameters.

## Examples

```
# PCM parameters: 15 items, 4 response categories
params <- irt_params_pcm(n_items = 15, n_categories = 4, seed = 42)

# Tighter within-item step spread (steps closer to the item center)
params <- irt_params_pcm(
  n_items = 15, n_categories = 4, step_dispersion = 0.5, seed = 42
)
```

---

irt\_simulate

*Run an IRT Monte Carlo Simulation*

---

## Description

Execute a Monte Carlo simulation study based on an [irt\\_study](#) specification. For each iteration and sample size, data are generated, missing values applied, the IRT model is fitted, and parameter estimates are extracted and stored.

## Usage

```
irt_simulate(
  study,
  iterations,
  seed,
  progress = TRUE,
  parallel = FALSE,
  se = TRUE,
  compute_theta = TRUE
)
```

## Arguments

study	An <a href="#">irt_study</a> object specifying the design and study conditions.
iterations	Positive integer. Number of Monte Carlo replications.
seed	Integer. Base random seed for reproducibility. Each iteration uses <code>seed + iteration - 1</code> .
progress	Logical. Print progress messages? Default TRUE.
parallel	Logical. Run iterations in parallel using <code>future.apply::future_lapply()</code> ? Default FALSE. Requires users to set up a future plan (e.g., <code>future::plan(multisession)</code> ) before calling. See Details.
se	Logical. Compute standard errors and confidence intervals for item parameter estimates? Default TRUE. Set to FALSE for significant speed improvement when only point estimates are needed (e.g., MSE, bias, RMSE criteria). When FALSE, <code>se/ci_lower/ci_upper</code> columns in <code>item_results</code> are NA.
compute_theta	Logical. Compute EAP theta estimates and recovery metrics (correlation, RMSE)? Default TRUE. Set to FALSE to skip the <code>mirt::fscores()</code> call when theta recovery is not needed. When FALSE, <code>theta_cor</code> and <code>theta_rmse</code> in <code>theta_results</code> are NA (but <code>converged</code> is still tracked).

## Details

The returned `irt_results` object stores raw per-iteration estimates. Use `summary.irt_results()` to compute performance criteria (bias, MSE, RMSE, coverage, etc.) and `plot.irt_results()` to visualize results.

### Parallelization:

When `parallel = TRUE`, the Monte Carlo loop over iterations is parallelized via `future.apply::future_lapply()`. Each parallel task processes one iteration across all sample sizes sequentially.

**Important:** This function does NOT configure a future plan. Users must set their own plan before calling with `parallel = TRUE`:

```
library(future)
plan(multisession, workers = 4) # or your preferred backend
results <- irt_simulate(study, iterations = 100, seed = 42, parallel = TRUE)
```

Without an explicit plan, future defaults to sequential execution (no parallelism).

### Reproducibility contract:

Reproducibility is guaranteed **within a given dispatch mode**, not across modes:

- **Serial mode** (`parallel = FALSE`) uses deterministic per-cell seeds under the session's default RNG kind (Mersenne-Twister). Re-running with the same base seed reproduces identical results bit-for-bit.
- **Parallel mode** (`parallel = TRUE`) delegates RNG management to `future.apply::future_lapply(..., future.seed = TRUE)`, which assigns each iteration a formally independent L'Ecuyer-CMRG substream. Re-running with the same base seed reproduces identical results bit-for-bit across parallel runs, including across different worker counts.

- **Across modes**, numerical results will differ because the two paths use different RNG algorithms and different seeding strategies. Both are statistically valid; the parallel path has the stronger formal guarantee of independent substreams, which is the standard for Monte Carlo work.

Progress messages are suppressed in parallel mode (workers cannot stream to stdout safely). Set `progress = FALSE` in serial mode to suppress messages (they appear every 10% of iterations).

## Value

An S3 object of class `irt_results` containing:

**item\_results** Data frame with per-iteration item parameter estimates (columns: `iteration`, `sample_size`, `item`, `param`, `true_value`, `estimate`, `se`, `ci_lower`, `ci_upper`, `converged`).

**theta\_results** Data frame with per-iteration theta recovery summaries (columns: `iteration`, `sample_size`, `theta_cor`, `theta_rmse`, `converged`).

**study** The original `irt_study` object.

**iterations** Number of replications run.

**seed** Base seed used.

**elapsed** Elapsed wall-clock time in seconds.

**se** Logical flag indicating whether SEs and CIs were computed.

**compute\_theta** Logical flag indicating whether theta recovery metrics were computed.

## See Also

`irt_study()` for specifying study conditions, `summary.irt_results()` and `plot.irt_results()` for analyzing output, `irt_iterations()` for determining the number of replications.

## Examples

```
# Minimal example (iterations and sample sizes reduced for speed;
# use iterations >= 100 and 3+ sample sizes in practice)
design <- irt_design(
  model = "1PL", n_items = 5,
  item_params = list(b = seq(-2, 2, length.out = 5))
)
study <- irt_study(design, sample_sizes = c(200, 500))
results <- irt_simulate(study, iterations = 10, seed = 42)
summary(results)
plot(results)
```

irt\_study

*Define Study Conditions for an IRT Simulation***Description**

Add study-level conditions to an IRT design specification. This captures decisions 4–5 from the Schroeders & Gnamb (2025) framework: sample sizes and missing data mechanism.

**Usage**

```
irt_study(
  design,
  sample_sizes,
  missing = "none",
  missing_rate = NULL,
  test_design = NULL,
  estimation_model = NULL
)
```

**Arguments**

design	An <a href="#">irt_design</a> object specifying the data-generating model.
sample_sizes	Integer vector of sample sizes to evaluate. Values are coerced to integer, sorted in ascending order, and deduplicated.
missing	Character string specifying the missing data mechanism. One of "none" (default), "mcar", "mar", "booklet", or "linking".
missing_rate	Numeric value in $[0, 1)$ specifying the proportion of missing data. Required when missing is "mcar" or "mar"; ignored when missing is "none".
test_design	A list specifying the test design for structured missingness. Required when missing is "booklet" or "linking". <b>booklet</b> Must contain <code>booklet_matrix</code> : a binary matrix ( $n\_booklets \times n\_items$ ) where 1 indicates the item is administered. <b>linking</b> Must contain <code>linking_matrix</code> : a binary matrix ( $n\_forms \times n\_items$ ) where 1 indicates the item appears on the form.
estimation_model	Character string specifying the IRT model to fit. One of "1PL", "2PL", "3PL", "GRM", "PCM", or "GPCM" (canonical list registered in <a href="#">get_model_config</a> ). If NULL (default), defaults to <code>design\$model</code> (i.e., the generation model is also the estimation model). Set to a different model to perform misspecification studies (e.g., generate 2PL, estimate 1PL). Cross-fits are only allowed within the same response format (binary: 1PL, 2PL, 3PL; polytomous: GRM, PCM, GPCM).

**Value**

An S3 object of class `irt_study` (a named list) with elements `design`, `missing`, `missing_rate`, `sample_sizes`, `test_design`, and `estimation_model`.

**See Also**

[irt\\_design\(\)](#) for the design specification, [irt\\_simulate\(\)](#) to run the simulation.

**Examples**

```
# Simple study with no missing data
d <- irt_design(
  model = "1PL", n_items = 20,
  item_params = list(b = seq(-2, 2, length.out = 20))
)
study <- irt_study(d, sample_sizes = c(100, 250, 500))

# Study with MCAR missingness
study_mcar <- irt_study(d, sample_sizes = c(200, 400),
  missing = "mcar", missing_rate = 0.2)

# Model misspecification: generate 2PL, fit 1PL
d_2pl <- irt_design(
  model = "2PL", n_items = 15,
  item_params = list(a = rlnorm(15, 0, 0.25), b = rnorm(15))
)
study_misspec <- irt_study(d_2pl, sample_sizes = c(100, 300),
  estimation_model = "1PL")
```

---

plot.irt\_results      *Plot IRT Simulation Results*

---

**Description**

Visualize performance criteria across sample sizes from an [irt\\_simulate\(\)](#) result. Calls [summary.irt\\_results\(\)](#) internally, then plots the requested criterion by sample size.

**Usage**

```
## S3 method for class 'irt_results'
plot(x, criterion = "rmse", param = NULL, item = NULL, threshold = NULL, ...)
```

**Arguments**

x	An <code>irt_results</code> object from <a href="#">irt_simulate()</a> .
criterion	Character string. Which criterion to plot. Default "rmse". Valid values: "bias", "empirical_se", "mse", "rmse", "coverage", "mcse_bias", "mcse_mse".
param	Optional character vector. Filter to specific parameter types (e.g., "a", "b", "b1").
item	Optional integer vector. Filter to specific item numbers.
threshold	Optional numeric. If provided, draws a horizontal reference line at this value.
...	Additional arguments passed to <a href="#">summary.irt_results()</a> .

**Value**

A `ggplot2::ggplot` object, returned invisibly.

**See Also**

`summary.irt_results()` for the underlying criteria, `recommended_n()` for sample-size recommendations.

**Examples**

```
design <- irt_design(
  model = "1PL", n_items = 5,
  item_params = list(b = seq(-2, 2, length.out = 5))
)
study <- irt_study(design, sample_sizes = c(200, 500))
results <- irt_simulate(study, iterations = 10, seed = 42)
plot(results)
plot(results, criterion = "bias", threshold = 0.05, param = "b")
```

---

plot.summary\_irt\_results

*Plot Summary of IRT Simulation Results*

---

**Description**

Visualize performance criteria from a `summary.irt_results()` object. This is a convenience method for users who already have a summary; `plot.irt_results()` is the primary interface.

**Usage**

```
## S3 method for class 'summary_irt_results'
plot(x, criterion = "rmse", param = NULL, item = NULL, threshold = NULL, ...)
```

**Arguments**

<code>x</code>	A <code>summary_irt_results</code> object from <code>summary.irt_results()</code> .
<code>criterion</code>	Character string. Which criterion to plot. Default "rmse".
<code>param</code>	Optional character vector. Filter to specific parameter types.
<code>item</code>	Optional integer vector. Filter to specific item numbers.
<code>threshold</code>	Optional numeric. If provided, draws a horizontal reference line at this value.
<code>...</code>	Additional arguments (ignored).

**Value**

A `ggplot2::ggplot` object, returned invisibly.

**See Also**

[plot.irt\\_results\(\)](#), [summary.irt\\_results\(\)](#)

**Examples**

```
design <- irt_design(  
  model = "1PL", n_items = 5,  
  item_params = list(b = seq(-2, 2, length.out = 5))  
)  
study <- irt_study(design, sample_sizes = c(200, 500))  
results <- irt_simulate(study, iterations = 10, seed = 42)  
s <- summary(results)  
plot(s, criterion = "rmse", threshold = 0.15)
```

---

print.irt\_design      *Print an IRT Design*

---

**Description**

Display a compact summary of an [irt\\_design](#) object, including model type, number of items, theta distribution, and parameter ranges.

**Usage**

```
## S3 method for class 'irt_design'  
print(x, ...)
```

**Arguments**

x                    An irt\_design object.  
...                   Additional arguments (ignored).

**Value**

x, invisibly.

**See Also**

[irt\\_design\(\)](#)

**Examples**

```
d <- irt_design("1PL", 10, list(b = seq(-2, 2, length.out = 10)))  
print(d)
```

---

print.irt\_results      *Print an IRT Simulation Result*

---

### Description

Display a compact summary of an `irt_simulate()` result, including model, items, sample sizes, iterations, convergence rate, and elapsed time.

### Usage

```
## S3 method for class 'irt_results'  
print(x, ...)
```

### Arguments

`x`                    An irt\_results object.  
`...`                 Additional arguments (ignored).

### Value

`x`, invisibly.

### See Also

[irt\\_simulate\(\)](#)

### Examples

```
design <- irt_design(  
  model = "1PL", n_items = 5,  
  item_params = list(b = seq(-2, 2, length.out = 5))  
)  
study <- irt_study(design, sample_sizes = c(200, 500))  
results <- irt_simulate(study, iterations = 10, seed = 42)  
print(results)
```

---

print.irt_study	<i>Print an IRT Study</i>
-----------------	---------------------------

---

**Description**

Display a compact summary of an [irt\\_study](#) object, including model, items, sample sizes, and missing data mechanism.

**Usage**

```
## S3 method for class 'irt_study'  
print(x, ...)
```

**Arguments**

x	An irt_study object.
...	Additional arguments (ignored).

**Value**

x, invisibly.

**See Also**

[irt\\_study\(\)](#)

**Examples**

```
d <- irt_design("1PL", 10, list(b = seq(-2, 2, length.out = 10)))  
s <- irt_study(d, sample_sizes = c(100, 500))  
print(s)
```

---

print.summary_irt_results	<i>Print Summary of IRT Simulation Results</i>
---------------------------	--

---

**Description**

Display item parameter criteria and theta recovery statistics from a [summary.irt\\_results\(\)](#) object.

**Usage**

```
## S3 method for class 'summary_irt_results'  
print(x, ...)
```

**Arguments**

x                    A `summary_irt_results` object.  
 ...                  Additional arguments (ignored).

**Value**

x, invisibly.

**See Also**

[summary.irt\\_results\(\)](#)

**Examples**

```
design <- irt_design(
  model = "1PL", n_items = 5,
  item_params = list(b = seq(-2, 2, length.out = 5))
)
study <- irt_study(design, sample_sizes = c(200, 500))
results <- irt_simulate(study, iterations = 10, seed = 42)
s <- summary(results)
print(s)
```

---

recommended\_n

*Find the Minimum Sample Size Meeting a Criterion Threshold*

---

**Description**

Given a [summary.irt\\_results\(\)](#) object, find the smallest sample size at which a performance criterion meets the specified threshold for each item and parameter combination.

**Usage**

```
recommended_n(object, ...)

## S3 method for class 'summary_irt_results'
recommended_n(
  object,
  criterion,
  threshold,
  param = NULL,
  item = NULL,
  aggregate = c("max", "mean", "median", "none"),
  ...
)
```

**Arguments**

<code>object</code>	A <code>summary_irt_results</code> object from <code>summary.irt_results()</code> .
<code>...</code>	Additional arguments (ignored).
<code>criterion</code>	Character string. Which criterion to evaluate. One of: "bias", "empirical_se", "mse", "rmse", "coverage", "mcse_bias", "mcse_mse".
<code>threshold</code>	Positive numeric. The threshold value the criterion must meet.
<code>param</code>	Optional character vector. Filter to specific parameter types (e.g., "a", "b", "b1").
<code>item</code>	Optional integer vector. Filter to specific item numbers.
<code>aggregate</code>	Character. How to roll the per-item recommended sample sizes up into a single recommendation. One of "max" (default — the smallest N that powers every item/param), "mean", "median", or "none" (return the per-item data frame unchanged). "mean" and "median" round up via <code>ceiling()</code> so the recommendation is never under the computed central tendency.

**Details**

For criteria where smaller is better (bias, empirical\_se, mse, rmse, mcse\_bias, mcse\_mse), the threshold is met when the criterion value is at or below the threshold. For bias, the absolute value is used. For coverage (where higher is better), the threshold is met when coverage is at or above the threshold.

**Value**

When `aggregate = "none"`, a data frame with columns:

**item** Item number.

**param** Parameter name.

**recommended\_n** Minimum sample size meeting the threshold, or NA if no tested sample size meets it.

**criterion** The criterion used (echoed back for reference).

**threshold** The threshold used (echoed back for reference).

When `aggregate` is "max", "mean", or "median" (the typical case), an integer scalar carrying the recommended sample size with attributes `details` (the per-item data frame above), `aggregate`, `criterion`, and `threshold`. If any item/param combination fails to meet the threshold at every tested sample size, the `aggregate` is `NA_integer_` and a warning lists the affected combinations.

**See Also**

`summary.irt_results()` for computing criteria, `plot.irt_results()` for visualization.

**Examples**

```

design <- irt_design(
  model = "1PL", n_items = 5,
  item_params = list(b = seq(-2, 2, length.out = 5))
)
study <- irt_study(design, sample_sizes = c(200, 500))
results <- irt_simulate(study, iterations = 10, seed = 42)
s <- summary(results)

# Default - single recommended N (max across items) for RMSE <= 0.20
n_rec <- recommended_n(s, criterion = "rmse", threshold = 0.20)
n_rec
attr(n_rec, "details") # per-item breakdown

# Mean / median aggregates (rounded up via ceiling)
recommended_n(s, criterion = "rmse", threshold = 0.20, aggregate = "mean")

# Legacy behavior - full per-item data frame
recommended_n(s, criterion = "rmse", threshold = 0.20, aggregate = "none")

# Minimum N for 95% coverage on difficulty parameters only
recommended_n(s, criterion = "coverage", threshold = 0.95, param = "b")

```

---

summary.irt\_results     *Summarize IRT Simulation Results*

---

**Description**

Compute performance criteria for each sample size, item, and parameter combination from an [irt\\_simulate\(\)](#) result. Criteria follow Morris et al. (2019) definitions. Optionally, users can provide a custom callback function to compute additional item-level performance criteria (e.g., conditional reliability, external criterion SE).

**Usage**

```

## S3 method for class 'irt_results'
summary(object, criterion = NULL, param = NULL, criterion_fn = NULL, ...)

```

**Arguments**

object	An irt_results object from <a href="#">irt_simulate()</a> .
criterion	Optional character vector. Which criteria to include in the output. Valid values: "bias", "empirical_se", "mse", "rmse", "coverage", "mcse_bias", "mcse_mse". If NULL (default), all criteria are returned.
param	Optional character vector. Which parameter types to include (e.g., "a", "b", "b1"). If NULL (default), all parameters are summarized.

**criterion\_fn** Optional function. A user-defined callback to compute custom performance criteria. Must accept named arguments `estimates` (numeric vector), `true_value` (scalar), `ci_lower` (numeric), `ci_upper` (numeric), `converged` (logical), and ... (for future use). Must return a named numeric vector of length  $\geq 1$ . The names become new columns in `item_summary`, appended after `n_converged`. If NULL (default), no custom criteria are computed.

... Additional arguments (ignored).

## Value

An S3 object of class `summary_irt_results` containing:

**item\_summary** Data frame with one row per `sample_size`  $\times$  `item`  $\times$  `param` combination, containing the requested criteria plus `n_converged` and any custom columns from `criterion_fn`.

**theta\_summary** Data frame with one row per `sample_size`, containing `mean_cor`, `sd_cor`, `mean_rmse`, `sd_rmse`, and `n_converged`.

**iterations** Number of replications.

**seed** Base seed used.

**model** IRT model type.

## References

Morris, T. P., White, I. R., & Crowther, M. J. (2019). Using simulation studies to evaluate statistical methods. *Statistics in Medicine*, 38(11), 2074–2102. doi:10.1002/sim.8086

## See Also

[irt\\_simulate\(\)](#) for running simulations, [plot.irt\\_results\(\)](#) for visualization, [recommended\\_n\(\)](#) for sample-size recommendations.

## Examples

```
# Minimal example (iterations reduced for speed; use 100+ in practice)
design <- irt_design(
  model = "1PL", n_items = 5,
  item_params = list(b = seq(-2, 2, length.out = 5))
)
study <- irt_study(design, sample_sizes = c(200, 500))
results <- irt_simulate(study, iterations = 10, seed = 42)

s <- summary(results)
s$item_summary
s$theta_summary

# Only bias and RMSE for difficulty parameters
summary(results, criterion = c("bias", "rmse"), param = "b")

# Compute custom criterion: relative bias
custom_fn <- function(estimates, true_value, ci_lower, ci_upper, converged, ...) {
  valid_est <- estimates[!is.na(estimates)]
```

```
    rel_bias <- (mean(valid_est) - true_value) / true_value
    c(relative_bias = rel_bias)
  }
summary(results, criterion_fn = custom_fn)

# Multiple custom criteria
multi_fn <- function(estimates, true_value, ci_lower, ci_upper, converged, ...) {
  valid_est <- estimates[!is.na(estimates)]
  c(mean_est = mean(valid_est), sd_est = sd(valid_est))
}
summary(results, criterion_fn = multi_fn)
```

# Index

`future.apply::future_lapply()`, [14](#)

`get_model_config`, [16](#)  
`get_model_config()`, [3](#)  
`ggplot2::ggplot`, [18](#)

`irt_design`, [2](#), [16](#), [19](#)  
`irt_design()`, [5–13](#), [17](#), [19](#)  
`irt_iterations`, [4](#)  
`irt_iterations()`, [15](#)  
`irt_params_1pl`, [5](#)  
`irt_params_1pl()`, [3](#)  
`irt_params_2pl`, [6](#)  
`irt_params_2pl()`, [3](#), [6](#), [8](#), [11](#)  
`irt_params_3pl`, [7](#)  
`irt_params_3pl()`, [3](#)  
`irt_params_gpcm`, [9](#)  
`irt_params_gpcm()`, [3](#)  
`irt_params_grm`, [11](#)  
`irt_params_grm()`, [3](#), [7](#), [8](#), [10](#), [13](#)  
`irt_params_pcm`, [12](#)  
`irt_params_pcm()`, [3](#), [10](#)  
`irt_simulate`, [13](#)  
`irt_simulate()`, [5](#), [17](#), [20](#), [24](#), [25](#)  
`irt_study`, [13–15](#), [16](#), [21](#)  
`irt_study()`, [3](#), [15](#), [21](#)

`mirt::fscores()`, [14](#)

`plot.irt_results`, [17](#)  
`plot.irt_results()`, [14](#), [15](#), [18](#), [19](#), [23](#), [25](#)  
`plot.summary_irt_results`, [18](#)  
`print.irt_design`, [19](#)  
`print.irt_results`, [20](#)  
`print.irt_study`, [21](#)  
`print.summary_irt_results`, [21](#)

`recommended_n`, [22](#)  
`recommended_n()`, [18](#), [25](#)

`summary.irt_results`, [24](#)