Package 'mixpower'

May 8, 2026

Title Simulation-Based Power Analysis for Mixed-Effects Models

Version 0.1.0

Description A comprehensive, simulation-based toolkit for power and sample-size analysis for linear and generalized linear mixed-effects models (LMMs and GLMMs). Supports Gaussian, binomial, Poisson, and negative binomial families via 'lme4'; Wald and likelihood-ratio tests; multi-parameter sensitivity grids; power curves and minimum sample-size solvers; parallel evaluation with deterministic seeds; and full reproducibility (manifests, result bundling, and export to CSV/JSON). Delivers thorough diagnostics per run (failure rate, singular-fit rate, effective N) and publication-ready summary tables. References: Bates et al. (2015) ``Fitting Linear Mixed-Effects Models Using lme4'' <doi:10.18637/jss.v067.i01>; Green and MacLeod (2016) ``SIMR: an R package for power analysis of generalized linear mixed models by simulation'' <doi:10.1111/2041-210X.12504>.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports stats, lme4

Suggests testthat (>= 3.0.0), knitr, rmarkdown, digest, jsonlite

VignetteBuilder knitr

Config/testthat/edition 3

URL https://github.com/alitovchenko/mixpower

BugReports https://github.com/alitovchenko/mixpower/issues

NeedsCompilation no

Author Mixpower Alex Litovchenko [aut, cre]

Maintainer Mixpower Alex Litovchenko <al4877@columbia.edu>

Repository CRAN

Date/Publication 2026-02-12 08:20:18 UTC

Contents

fit_model . . . 2
mp_assumptions . . . 3
mp_backend_lme4 . . . 4
mp_backend_lme4_binomial . . . 4
mp_backend_lme4_nb . . . 5
mp_backend_lme4_poisson . . . 6
mp_bundle_results . . . 7
mp_design . . . 7
mp_manifest . . . 8
mp_power . . . 9
mp_power_curve . . . 10
mp_power_curve_parallel . . . 11
mp_report_table . . . 12
mp_scenario . . . 13
mp_scenario_lme4 . . . 14
mp_scenario_lme4_binomial . . . 15
mp_scenario_lme4_nb . . . 16
mp_scenario_lme4_poisson . . . 17
mp_sensitivity . . . 18
mp_solve_sample_size . . . 18
mp_write_results . . . 19
plot.mp_power_curve . . . 20
plot.mp_sensitivity . . . 21
plot_power . . . 21
run_parallel . . . 22
simulate_glmm_binomial_data . . . 22
simulate_glmm_nb_data . . . 23
simulate_glmm_poisson_data . . . 24
simulate_power . . . 24
summarize_simulations . . . 25
test_effect . . . 25

Index **26**

`fit_model` *Fit a model for a single simulated dataset*

Description

Fit a model for a single simulated dataset

Usage

```
fit_model(data, formula)
```

Arguments

data	A data.frame of simulated data.
formula	A model formula.

Value

A fitted model object.

---

mp_assumptions *Create modeling assumptions for simulation-based power*

Description

Assumptions encode effect sizes and nuisance parameters. Values may be scalars or vectors (for later sensitivity workflows), but `mp_power()` expects scalars unless used inside a grid wrapper.

Usage

```
mp_assumptions(fixed_effects, icc = NULL, residual_sd = NULL, notes = NULL)
```

Arguments

fixed_effects	Named list of numeric values (e.g., `list(condition = 0.4)`).
icc	Optional named list of ICC values in [0, 1).
residual_sd	Optional non-negative numeric residual SD.
notes	Optional free text.

Value

An object of class `mp_assumptions`.

Examples

```
a <- mp_assumptions(fixed_effects = list(condition = 0.4), residual_sd = 1)
a
```

mp_backend_lme4 *Build an lme4 backend for MixPower scenarios*

Description

Build an lme4 backend for MixPower scenarios

Usage

```
mp_backend_lme4(
  predictor = "condition",
  subject = "subject",
  outcome = "y",
  item = NULL,
  test_method = c("wald", "lrt"),
  null_formula = NULL
)
```

Arguments

predictor	Predictor column name.
subject	Subject ID column name.
outcome	Outcome column name.
item	Optional item ID column name.
test_method	Inference method: "wald" (default) or "lrt".
null_formula	Optional null-model formula required when test_method = "lrt".

Value

A list containing simulate_fun, fit_fun, and test_fun.

mp_backend_lme4_binomial *Build an lme4 backend for binomial GLMM scenarios*

Description

Build an lme4 backend for binomial GLMM scenarios

Usage

```
mp_backend_lme4_binomial(
  predictor = "condition",
  subject = "subject",
  outcome = "y",
  item = NULL,
  test_method = c("wald", "lrt"),
  null_formula = NULL
)
```

Arguments

predictor	Predictor column name.
subject	Subject ID column name.
outcome	Outcome column name.
item	Optional item ID column name.
test_method	Inference method ("wald" or "lrt").
null_formula	Optional null model formula for "lrt" tests.

Value

A list containing simulate_fun, fit_fun, and test_fun.

---

mp_backend_lme4_nb *Build an lme4 backend for Negative Binomial GLMM scenarios*

Description

Build an lme4 backend for Negative Binomial GLMM scenarios

Usage

```
mp_backend_lme4_nb(
  predictor = "condition",
  subject = "subject",
  outcome = "y",
  item = NULL,
  test_method = c("wald", "lrt"),
  null_formula = NULL
)
```

Arguments

predictor	Predictor column name.
subject	Subject ID column name.
outcome	Outcome column name.
item	Optional item ID column name.
test_method	Inference method ("wald" or "lrt").
null_formula	Optional null model formula for "lrt" tests.

Value

A list containing simulate_fun, fit_fun, and test_fun.

---

mp_backend_lme4_poisson

Build an lme4 backend for Poisson GLMM scenarios

Description

Build an lme4 backend for Poisson GLMM scenarios

Usage

```
mp_backend_lme4_poisson(
  predictor = "condition",
  subject = "subject",
  outcome = "y",
  item = NULL,
  test_method = c("wald", "lrt"),
  null_formula = NULL
)
```

Arguments

predictor	Predictor column name.
subject	Subject ID column name.
outcome	Outcome column name.
item	Optional item ID column name.
test_method	Inference method ("wald" or "lrt").
null_formula	Optional null model formula for "lrt" tests.

Value

A list containing simulate_fun, fit_fun, and test_fun.

mp_bundle_results *Bundle results with manifest and optional labels*

Description

Combines a single result object (mp_power, mp_sensitivity, or mp_power_curve), a reproducibility manifest, and optional user labels into one object. Diagnostics and result structure are retained unchanged.

Usage

```
mp_bundle_results(
  result,
  manifest,
  study_id = NULL,
  analyst = NULL,
  notes = NULL
)
```

Arguments

result	An object of class mp_power, mp_sensitivity, or mp_power_curve.
manifest	An mp_manifest object (from mp_manifest()).
study_id	Optional character; study or run identifier.
analyst	Optional character; analyst name or ID.
notes	Optional character; free-form notes.

Value

An object of class mp_bundle with components result, manifest, and labels (list with study_id, analyst, notes).

mp_design *Create a study design specification*

Description

mp_design() encodes how data will be collected: cluster sizes and repeated measurements. It does not encode effect sizes or analysis decisions.

Usage

```
mp_design(clusters, trials_per_cell = 1, notes = NULL)
```

Arguments

clusters	Named list of positive integers. Example: list(subject = 50, item = 30).
trials_per_cell	Positive integer. Number of repeated observations per design cell.
notes	Optional free text.

Value

An object of class mp_design.

Examples

```
d <- mp_design(clusters = list(subject = 40), trials_per_cell = 10)
d
```

---

mp_manifest *Reproducibility manifest for power analyses*

Description

Captures scenario fingerprint, seed strategy, session info, timestamp, and optional git SHA so results can be reproduced or audited. Output is a plain list (and one-row data frame via as.data.frame()) suitable for saving alongside results.

Usage

```
mp_manifest(scenario, seed = NULL, session = TRUE)
```

Arguments

scenario	An mp_scenario object (used for digest).
seed	The seed value used (or NULL). Stored as-is; strategy is inferred as "fixed" if non-null else "none".
session	Include full sessionInfo() (default TRUE). If FALSE, only R version and mixpower version are stored.

Value

A list with components: scenario_digest, seed, seed_strategy, timestamp, r_version, mixpower_version, session_info (if requested), git_sha (if in a git repo). Use as.data.frame() on the list for a single-row table (list components become columns where possible).

mp_power *Simulation-based power estimation (engine-agnostic core)*

Description

`mp_power()` runs repeated simulations under a scenario and estimates power for the scenario's test decision rule (typically p < alpha).

Usage

```
mp_power(
  scenario,
  nsim,
  alpha = 0.05,
  seed = NULL,
  failure_policy = c("count_as_nondetect", "exclude"),
  keep = c("minimal", "fits", "data"),
  conf_level = 0.95
)
```

Arguments

`scenario`	An `mp_scenario`.
`nsim`	Positive integer number of simulations.
`alpha`	Significance threshold for a detection (default 0.05).
`seed`	Optional seed for reproducibility.
`failure_policy`	How to treat failed fits / missing p-values: • `"count_as_nondetect"` (default): failures count as non-detections. • `"exclude"`: drop failures from the denominator (always reported).
`keep`	What to store: • `"minimal"`: only per-sim summary rows. • `"fits"`: also store fit objects (may be large). • `"data"`: also store simulated data (can be very large).
`conf_level`	Confidence level for the Wald interval (default 0.95).

Details

In Phase 4 core, the scenario must provide engine functions: `simulate_fun`, `fit_fun`, and `test_fun`. Later phases will supply defaults based on specific backends (e.g., lme4).

Value

An object of class `mp_power`.

Examples

```
# A tiny toy engine (not mixed models) just to demonstrate the workflow:
d <- mp_design(list(subject = 30), trials_per_cell = 1)
a <- mp_assumptions(list(condition = 0.3), residual_sd = 1)

sim_fun <- function(scn, seed) {
  n <- scn$design$clusters$subject
  x <- stats::rbinom(n, 1, 0.5)
  y <- scn$assumptions$fixed_effects$condition * x +
    stats::rnorm(n, sd = scn$assumptions$residual_sd)
  data.frame(y = y, condition = x)
}
fit_fun <- function(dat, scn) stats::lm(scn$formula, data = dat)
test_fun <- function(fit, scn) {
  sm <- summary(fit)
  p <- sm$coefficients["condition", "Pr(>|t|)"]
  list(p_value = as.numeric(p))
}

s <- mp_scenario(
  y ~ condition, d, a,
  simulate_fun = sim_fun,
  fit_fun = fit_fun,
  test_fun = test_fun
)
res <- mp_power(s, nsim = 50, seed = 1)
summary(res)
```

`mp_power_curve` *Power curve for a single design/assumption parameter*

Description

Runs `mp_power()` across a one-dimensional grid of values for one parameter (e.g. cluster size) via `mp_sensitivity()`. Results include power estimates and per-grid-point diagnostics: failure rate, singular rate, and effective N.

Usage

```
mp_power_curve(
  scenario,
  vary,
  nsim = 100,
  alpha = 0.05,
  seed = NULL,
  failure_policy = c("count_as_nondetect", "exclude"),
  conf_level = 0.95
)
```

Arguments

scenario	An mp_scenario.
vary	Named list with a single key (e.g. clusters.subject).
nsim	Number of simulations per grid point (default 100).
alpha	Significance level (default 0.05).
seed	Optional seed for reproducibility.
failure_policy	How to treat failed fits: "count_as_nondetect" or "exclude".
conf_level	Confidence level for power intervals (default 0.95).

Value

An object of class mp_power_curve with components vary, grid, results (estimate, mcse, conf_low, conf_high, failure_rate, singular_rate, n_effective, nsim, plus the varying parameter column), alpha, failure_policy, and conf_level.

---

mp_power_curve_parallel

Parallel power curve evaluation

Description

Evaluates power over a one-parameter grid by running mp_power() for each grid cell in parallel. Uses explicit per-cell seeds (seed + cell_index - 1L) so results are deterministic and match serial mp_power_curve() for the same seed. Does not modify mp_power(); parallelization is at the scenario-grid level only.

Usage

```
mp_power_curve_parallel(
  scenario,
  vary,
  workers = 2L,
  nsim = 100,
  alpha = 0.05,
  seed = NULL,
  failure_policy = c("count_as_nondetect", "exclude"),
  conf_level = 0.95,
  progress = FALSE,
  ...
)
```

Arguments

scenario	An mp_scenario.
vary	Named list with exactly one parameter (e.g. clusters.subject).
workers	Number of parallel workers (default 2).
nsim	Number of simulations per grid point (default 100).
alpha	Significance level (default 0.05).
seed	Optional base seed; each cell gets seed + cell_index - 1L.
failure_policy	How to treat failed fits: "count_as_nondetect" or "exclude".
conf_level	Confidence level for power intervals (default 0.95).
progress	If TRUE, run serially with a progress bar; if FALSE, run in parallel.
...	Unused; reserved for future arguments.

Value

An object of class mp_power_curve (same structure as mp_power_curve()).

Note

Parallel execution requires the **parallel** package (base R) and that **mixpower** is installed (e.g. install.packages() or devtools::install()) so that workers can load it.

---

mp_report_table *Publication-ready summary table for power results*

Description

Returns a flat data frame with power estimate, CI, failure/singularity rates, and effective simulation counts. Works with mp_power, mp_sensitivity, mp_power_curve, or the result of mp_bundle_results() (uses the bundled result).

Usage

```
mp_report_table(x, ...)
```

Arguments

x	An object of class mp_power, mp_sensitivity, mp_power_curve, or from mp_bundle_results().
...	Unused; reserved for future arguments.

Value

A data frame: for mp_power one row; for sensitivity/curve one row per grid cell with parameter column(s), power_estimate, ci_low, ci_high, failure_rate, singular_rate, n_effective, nsim.

mp_scenario *Create a power-analysis scenario*

Description

A scenario combines: (1) a design, (2) assumptions, (3) a model specification, and (4) an analysis engine.

Usage

```
mp_scenario(
  formula,
  design,
  assumptions,
  test = c("wald", "lrt", "custom"),
  simulate_fun = NULL,
  fit_fun = NULL,
  test_fun = NULL,
  notes = NULL
)
```

Arguments

formula	A model formula (stored for later backends).
design	An mp_design.
assumptions	An mp_assumptions.
test	Character string or list identifying the test type (metadata).
simulate_fun	Function or NULL.
fit_fun	Function or NULL.
test_fun	Function or NULL.
notes	Optional free text.

Details

In Phase 1, the engine is *pluggable* via three functions:

- simulate_fun(scenario, seed) returns a data.frame
- fit_fun(data, scenario) returns a fit object
- test_fun(fit, scenario) returns a list with at least p_value (numeric scalar)

This allows mp_power() to run before selecting a specific backend (e.g., lme4).

Value

An object of class mp_scenario.

Examples

```
d <- mp_design(list(subject = 20), trials_per_cell = 5)
a <- mp_assumptions(list(condition = 0.3), residual_sd = 1)
s <- mp_scenario(y ~ condition, d, a, test = "wald")
s
```

`mp_scenario_lme4` *Create a fully specified MixPower scenario with the lme4 backend*

Description

Create a fully specified MixPower scenario with the lme4 backend

Usage

```
mp_scenario_lme4(
  formula,
  design,
  assumptions,
  predictor = "condition",
  subject = "subject",
  outcome = "y",
  item = NULL,
  test_term = predictor,
  test_method = c("wald", "lrt"),
  null_formula = NULL
)
```

Arguments

`formula`	Model formula.
`design`	A `mp_design` object.
`assumptions`	A `mp_assumptions` object.
`predictor`	Predictor column name.
`subject`	Subject ID column name.
`outcome`	Outcome column name.
`item`	Optional item ID column name.
`test_term`	Optional explicit term to test. Defaults to `predictor`.
`test_method`	Inference method: `"wald"` (default) or `"lrt"`.
`null_formula`	Optional null-model formula required for `test_method = "lrt"`.

Value

An object of class `mp_scenario`.

`mp_scenario_lme4_binomial` *Create a fully specified MixPower scenario with the binomial lme4 backend*

Description

Create a fully specified MixPower scenario with the binomial lme4 backend

Usage

```
mp_scenario_lme4_binomial(
  formula,
  design,
  assumptions,
  predictor = "condition",
  subject = "subject",
  outcome = "y",
  item = NULL,
  test_term = predictor,
  test_method = c("wald", "lrt"),
  null_formula = NULL
)
```

Arguments

`formula`	Model formula.
`design`	A `mp_design` object.
`assumptions`	A `mp_assumptions` object.
`predictor`	Predictor column name.
`subject`	Subject ID column name.
`outcome`	Outcome column name.
`item`	Optional item ID column name.
`test_term`	Optional explicit term to test. Defaults to `predictor`.
`test_method`	Inference method (`"wald"` or `"lrt"`).
`null_formula`	Optional null model formula for `"lrt"` tests.

Value

An object of class `mp_scenario`.

`mp_scenario_lme4_nb` *Create a fully specified MixPower scenario with the NB lme4 backend*

Description

Create a fully specified MixPower scenario with the NB lme4 backend

Usage

```
mp_scenario_lme4_nb(
  formula,
  design,
  assumptions,
  predictor = "condition",
  subject = "subject",
  outcome = "y",
  item = NULL,
  test_term = predictor,
  test_method = c("wald", "lrt"),
  null_formula = NULL
)
```

Arguments

`formula`	Model formula.
`design`	A `mp_design` object.
`assumptions`	A `mp_assumptions` object.
`predictor`	Predictor column name.
`subject`	Subject ID column name.
`outcome`	Outcome column name.
`item`	Optional item ID column name.
`test_term`	Optional explicit term to test. Defaults to `predictor`.
`test_method`	Inference method (`"wald"` or `"lrt"`).
`null_formula`	Optional null model formula for `"lrt"` tests.

Value

An object of class `mp_scenario`.

`mp_scenario_lme4_poisson` *Create a fully specified MixPower scenario with the Poisson lme4 backend*

Description

Create a fully specified MixPower scenario with the Poisson lme4 backend

Usage

```
mp_scenario_lme4_poisson(
  formula,
  design,
  assumptions,
  predictor = "condition",
  subject = "subject",
  outcome = "y",
  item = NULL,
  test_term = predictor,
  test_method = c("wald", "lrt"),
  null_formula = NULL
)
```

Arguments

`formula`	Model formula.
`design`	A `mp_design` object.
`assumptions`	A `mp_assumptions` object.
`predictor`	Predictor column name.
`subject`	Subject ID column name.
`outcome`	Outcome column name.
`item`	Optional item ID column name.
`test_term`	Optional explicit term to test. Defaults to `predictor`.
`test_method`	Inference method (`"wald"` or `"lrt"`).
`null_formula`	Optional null model formula for `"lrt"` tests.

Value

An object of class `mp_scenario`.

mp_sensitivity *Run power sensitivity analysis over a parameter grid*

Description

Run power sensitivity analysis over a parameter grid

Usage

```
mp_sensitivity(
  scenario,
  vary,
  nsim = 100,
  alpha = 0.05,
  seed = NULL,
  failure_policy = c("count_as_nondetect", "exclude"),
  conf_level = 0.95
)
```

Arguments

scenario	A base mp_scenario object.
vary	Named list of vectors. Names are dotted paths such as "fixed_effects.condition" or "clusters.subject".
nsim	Number of simulations for each grid cell.
alpha	Significance threshold.
seed	Optional seed for reproducible cell-wise execution.
failure_policy	Failure policy passed to mp_power().
conf_level	Confidence level passed to mp_power().

Value

An object of class mp_sensitivity.

mp_solve_sample_size *Solve for minimum sample size achieving target power*

Description

Evaluates power on a user-supplied grid of values for one parameter (e.g. cluster size) via mp_power_curve(), then returns the smallest grid value whose power estimate meets or exceeds the target. Diagnostics (failure rate, singular rate, n_effective) are exposed in the returned results table.

Usage

```
mp_solve_sample_size(
  scenario,
  parameter,
  grid,
  target_power = 0.8,
  nsim = 100,
  alpha = 0.05,
  seed = NULL,
  failure_policy = c("count_as_nondetect", "exclude"),
  conf_level = 0.95
)
```

Arguments

scenario	An mp_scenario.
parameter	Dotted path of the single parameter to vary (e.g. "clusters.subject").
grid	Numeric vector of candidate values.
target_power	Target power threshold (default 0.8).
nsim	Number of simulations per grid point (default 100).
alpha	Significance level (default 0.05).
seed	Optional seed for reproducibility.
failure_policy	How to treat failed fits: "count_as_nondetect" or "exclude".
conf_level	Confidence level for power intervals (default 0.95).

Value

A list with target_power, parameter, solution (numeric: minimum grid value achieving target power, or NA if none), and results (data frame with estimate, failure_rate, singular_rate, n_effective, etc., per grid point).

---

mp_write_results *Write results or bundle to CSV or JSON*

Description

Writes the report table (and for bundles, manifest/labels) to file. CSV writes the publication-ready table only; JSON writes report table plus manifest and labels when x is an mp_bundle.

Usage

```
mp_write_results(x, file, format = c("csv", "json"), ...)
```

Arguments

x	An object from mp_bundle_results(), or mp_power, mp_sensitivity, or mp_power_curve.
file	Path to output file (extension need not match format).
format	"csv" or "json".
...	For CSV, arguments passed to utils::write.csv() (e.g. row.names = FALSE).

Value

Invisibly the path file.

---

plot.mp_power_curve *Plot a power curve*

Description

Plot a power curve

Usage

```
## S3 method for class 'mp_power_curve'
plot(x, y = c("estimate", "failure_rate", "singular_rate", "n_effective"), ...)
```

Arguments

x	An mp_power_curve object.
y	What to plot on the y-axis: "estimate" (power), "failure_rate", "singular_rate", or "n_effective".
...	Arguments passed to graphics::plot().

Value

Invisibly returns the plotted data.

plot.mp_sensitivity *Plot a one-dimensional sensitivity curve*

Description

Plot a one-dimensional sensitivity curve

Usage

```
## S3 method for class 'mp_sensitivity'
plot(x, y = c("estimate", "failure_rate"), ...)
```

Arguments

x	An mp_sensitivity object.
y	What to plot on the y-axis ("estimate" or "failure_rate").
...	Additional graphical arguments passed to graphics::plot().

Value

Invisibly returns the plotted data.

plot_power *Plot power results*

Description

Plot power results

Usage

```
plot_power(results, ...)
```

Arguments

results	A data.frame with effect and power columns.
...	Additional arguments passed to plot.

Value

Invisibly returns the plot data.

run_parallel *Placeholder for parallel execution*

Description

Placeholder for parallel execution

Usage

```
run_parallel(fun, ...)
```

Arguments

fun	Function to run.
...	Additional arguments to pass to fun.

Value

The result of fun.

simulate_glmm_binomial_data *Simulate binary outcome data for a GLMM with random intercepts*

Description

Simulate binary outcome data for a GLMM with random intercepts

Usage

```
simulate_glmm_binomial_data(
  scenario,
  predictor = "condition",
  subject = "subject",
  outcome = "y",
  item = NULL
)
```

Arguments

scenario	An mp_scenario object.
predictor	Predictor column name.
subject	Subject ID column name.
outcome	Outcome column name.
item	Optional item ID column name.

Value

A data.frame with outcome and predictors.

`simulate_glmm_nb_data` *Simulate count outcome data for a Negative Binomial GLMM*

Description

Simulate count outcome data for a Negative Binomial GLMM

Usage

```
simulate_glmm_nb_data(
  scenario,
  predictor = "condition",
  subject = "subject",
  outcome = "y",
  item = NULL,
  theta = NULL
)
```

Arguments

`scenario`	An `mp_scenario` object.
`predictor`	Predictor column name.
`subject`	Subject ID column name.
`outcome`	Outcome column name.
`item`	Optional item ID column name.
`theta`	NB dispersion parameter (size). Larger = less over-dispersion.

Value

A data.frame with outcome and predictors.

simulate_glmm_poisson_data

Simulate count outcome data for a Poisson GLMM with random intercepts

Description

Simulate count outcome data for a Poisson GLMM with random intercepts

Usage

```
simulate_glmm_poisson_data(
  scenario,
  predictor = "condition",
  subject = "subject",
  outcome = "y",
  item = NULL
)
```

Arguments

scenario	An mp_scenario object.
predictor	Predictor column name.
subject	Subject ID column name.
outcome	Outcome column name.
item	Optional item ID column name.

Value

A data.frame with outcome and predictors.

simulate_power

Run a simple simulation-based power study

Description

Run a simple simulation-based power study

Usage

```
simulate_power(scenario, nsim = 100, seed = NULL)
```

Arguments

scenario	A scenario object.
nsim	Number of simulations.
seed	Optional random seed.

Value

A data.frame of simulated p-values.

---

summarize_simulations *Summarize simulation outputs*

Description

Summarize simulation outputs

Usage

```
summarize_simulations(simulations)
```

Arguments

simulations	A data.frame of simulations.

Value

A summary data.frame.

---

test_effect *Extract a test statistic for a model term*

Description

Extract a test statistic for a model term

Usage

```
test_effect(fit, term)
```

Arguments

fit	A fitted model object.
term	Term name to test.

Value

A data.frame with coefficient information.

Index

as.data.frame(), *8*

fit_model, 2

graphics::plot(), *20, 21*

mp_assumptions, 3
mp_backend_lme4, 4
mp_backend_lme4_binomial, 4
mp_backend_lme4_nb, 5
mp_backend_lme4_poisson, 6
mp_bundle_results, 7
mp_bundle_results(), *12*, *20*
mp_design, 7
mp_manifest, 8
mp_manifest(), *7*
mp_power, *7*, 9, *12*
mp_power(), *10, 11*, *18*
mp_power_curve, *7*, 10, *12*
mp_power_curve(), *11, 12*, *18*
mp_power_curve_parallel, 11
mp_report_table, 12
mp_scenario, 13
mp_scenario_lme4, 14
mp_scenario_lme4_binomial, 15
mp_scenario_lme4_nb, 16
mp_scenario_lme4_poisson, 17
mp_sensitivity, *7*, *12*, 18
mp_sensitivity(), *10*
mp_solve_sample_size, 18
mp_write_results, 19

plot.mp_power_curve, 20
plot.mp_sensitivity, 21
plot_power, 21

run_parallel, 22

simulate_glmm_binomial_data, 22
simulate_glmm_nb_data, 23
simulate_glmm_poisson_data, 24
simulate_power, 24
summarize_simulations, 25

test_effect, 25

utils::write.csv(), *20*