Package 'PublicationBiasBenchmark'

November 7, 2026

Type Package

Title Benchmark for Publication Bias Correction Methods

Version 0.4.0

Maintainer František Bartoš <f.bartos96@gmail.com>

Description Implements a unified interface for benchmarking meta-analytic publication bias correction methods through simulation studies (see Bartoš et al., 2025, <doi:10.48550/arXiv.2510.19489>). It provides 1) predefined data-generating mechanisms from the literature, 2) functions for running meta-analytic methods on simulated data, 3) pre-simulated datasets and pre-computed results for reproducible benchmarks, 4) tools for visualizing and comparing method performance.

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 3.5.0)

Imports stats, callr, metafor, curl, digest, httr, jsonlite, MASS, numDeriv, pwr, sandwich, clubSandwich, lmtest, puniform, MAIVE, phacking, robumeta, publipha, rstan, Rdpack

Suggests zip, RoBMA (<= 3.6.1), pkgload, BayesTools (<= 0.2.23), testthat (>= 3.0.0), withr, rmarkdown, knitr, kableExtra, DT, ggplot2, scales, ggdist, rprojroot, desc

RdMacros Rdpack

URL https://github.com/FBartos/PublicationBiasBenchmark, https://fbartos.github.io/PublicationBiasBenchmark/

BugReports https://github.com/FBartos/PublicationBiasBenchmark/issues

VignetteBuilder knitr

NeedsCompilation no

Author František Bartoš [aut, cre] (ORCID: <https://orcid.org/0000-0002-0018-5573>), Samuel Pawel [aut] (ORCID: <https://orcid.org/0000-0003-2779-320X>),

Björn S. Siepe [aut] (ORCID: <https://orcid.org/0000-0002-9558-4648>),
Petr Čala [aut]

Repository CRAN

Date/Publication 2026-11-07 19:00:02 UTC

Contents

benchmark_catalog . . . 3
compare_measures . . . 4
compare_single_measure . . . 6
compute_measures . . . 8
compute_single_measure . . . 11
create_empty_result . . . 13
dgm . . . 14
dgm.Alinaghi2018 . . . 15
dgm.Bom2019 . . . 17
dgm.Carter2019 . . . 18
dgm.default . . . 20
dgm.no_bias . . . 20
dgm.Stanley2017 . . . 21
dgm_conditions . . . 23
download_dgm . . . 24
measure . . . 25
measures . . . 26
measure_mcse . . . 29
method . . . 29
method.AK . . . 30
method.default . . . 32
method.EK . . . 32
method.FMA . . . 33
method.MAIVE . . . 34
method.MAN . . . 37
method.mean . . . 38
method.MMPH . . . 39
method.pcurve . . . 40
method.PEESE . . . 41
method.PET . . . 42
method.PETPEESE . . . 43
method.puniform . . . 44
method.RMA . . . 45
method.RoBMA . . . 47
method.RTMA . . . 48
method.SM . . . 50
method.trimfill . . . 51
method.WAAPWLS . . . 52
method.WILS . . . 53
method.WLS . . . 54

method_extra_columns 55
method_settings 56
prune_benchmark_archives 57
PublicationBiasBenchmark_options 57
retrieve_dgm_dataset 58
retrieve_dgm_measures 59
retrieve_dgm_results 60
run_method 60
simulate_dgm 62
validate_dgm_setting 63
verify_benchmark_resources 64

Index **65**

`benchmark_catalog` *Inspect Benchmark Releases and Resources*

Description

Benchmark releases are complete catalogs of immutable files. Public downloads do not require a Zenodo token. A package release pins its default catalog; another release or a local catalog can be selected explicitly. Parsed catalogs are reused by content hash; cached bytes are verified before use.

Usage

```
list_benchmark_releases()

benchmark_catalog(release = NULL)

benchmark_conditions(dgm_name, release = NULL)

list_benchmark_resources(
  release = NULL,
  dgm_name = NULL,
  kind = NULL,
  method = NULL,
  method_setting = NULL
)
```

Arguments

`release`	Benchmark release identifier, path to a catalog JSON file, or a catalog list. NULL uses the benchmark_release package option.
`dgm_name`	DGM name (optional when listing resources).
`kind`	Optional resource kind: data, results, measures, pairwise, metadata, or archive.
`method`	Optional method name(s).
`method_setting`	Optional method setting(s).

Value

list_benchmark_releases returns a data frame. benchmark_catalog returns a validated list; list_benchmark_resources returns a data frame of file references. benchmark_conditions returns the release's frozen condition data frame.

`compare_measures` *Compare method with Multiple Measures for a DGM*

Description

This is a high-level wrapper function that computes multiple pairwise comparison measures for a Data-Generating Mechanism (DGM) and saves the results to CSV files. It provides a clean and extensible interface for comparing method performance.

Usage

```
compare_measures(
  dgm_name,
  method,
  method_setting,
  measures = NULL,
  verbose = TRUE,
  estimate_col = "estimate",
  true_effect_col = "mean_effect",
  convergence_col = "convergence",
  method_replacements = NULL,
  n_repetitions = 1000,
  overwrite = FALSE,
  conditions = NULL,
  results_source = c("local", "release"),
  release = NULL,
  replacement_source = NULL
)
```

Arguments

`dgm_name`	Character string specifying the DGM name
`method`	Character vector of method names
`method_setting`	Character vector of method settings, must be same length as method
`measures`	Character vector of measures to compute. If NULL, computes all standard measures.
`verbose`	Print detailed progress of the calculation.
`estimate_col`	Character string specifying the column name containing parameter estimates. Default is "estimate"

`true_effect_col`
: Character string specifying the column name in conditions data frame containing true effect sizes. Default is "mean_effect"

`convergence_col`
: Character string specifying the column name containing convergence indicators. Default is "convergence"

`method_replacements`
: Named list of replacement method specifications. Each element should be named with the "method-method_setting" combination (e.g., "RMA-default") and contain a named list with:

  - `method`: Character vector of replacement method names
  - `method_setting`: Character vector of replacement method settings (same length as methods)
  - `power_test_type`: Optional character vector of power test types for each replacement method (same length as methods). If not specified, uses the main power_test_type parameter

  If multiple elements are specified within the vectors, these replacements are applied consecutively in case the previous replacements also failed to converge. Defaults to `NULL`, i.e., omitting repetitions without converged results on method-by-method basis.

`n_repetitions`
: Number of repetitions in each condition. Used to verify that the results contain all repetitions (an error is thrown if any repetition is missing) and for method replacement. Defaults to `1000`.

`overwrite`
: Logical indicating whether to overwrite existing results. If FALSE (default), will skip computation for method-measure combinations that already exist

`conditions`
: Data frame of conditions from dgm_conditions(). If NULL, the selected release's frozen conditions are used when `results_source = "release"` and `dgm_conditions(dgm_name)` otherwise. With local results and release replacements the package's conditions are used, and the release replacement results must cover each of them (missing repetitions are an error).

`results_source`
: Where the results of the computed methods come from: "local" (default) reads unpublished outputs under resources_directory/DGM/results; "release" reads verified results of a published benchmark release (see download_dgm_results()).

`release`
: Benchmark release identifier or catalog, passed to the result reader whenever either source is "release". It is ignored, with a warning, when both `results_source` and `replacement_source` are "local".

`replacement_source`
: Where the results of the replacement methods in `method_replacements` come from: "local" or "release". NULL (default) uses `results_source`. Use `replacement_source = "release"` to replace a new method's failures with published results of established methods.

Value

Invisible list of computed comparison data frames

compare_single_measure *Compare method with a Single Measure for a DGM*

Description

This function provides pairwise comparison of method for Data-Generating Mechanisms (DGMs). It compares method performance on a condition-by-condition basis using estimates. For each pair of method, if method A has an estimate closer to the true value than method B, it gets a score of 1, if further it gets 0, and if equal it gets 0.5.

Usage

```
compare_single_measure(
  dgm_name,
  measure_name,
  method,
  method_setting,
  conditions,
  estimate_col = "estimate",
  true_effect_col = "mean_effect",
  convergence_col = "convergence",
  method_replacements = NULL,
  n_repetitions = 1000,
  overwrite = FALSE,
  results_source = c("local", "release"),
  release = NULL,
  replacement_source = NULL,
  ...
)
```

Arguments

dgm_name	Character string specifying the DGM name
measure_name	Name of the measure to compute (e.g., "bias", "mse")
method	Character vector of method names
method_setting	Character vector of method settings, must be same length as method
conditions	Data frame of conditions from dgm_conditions(). If NULL, the selected release's frozen conditions are used when results_source = "release" and dgm_conditions(dgm_name) otherwise. With local results and release replacements the package's conditions are used, and the release replacement results must cover each of them (missing repetitions are an error).
estimate_col	Character string specifying the column name containing parameter estimates. Default is "estimate"

`true_effect_col`
: Character string specifying the column name in conditions data frame containing true effect sizes. Default is "mean_effect"

`convergence_col`
: Character string specifying the column name containing convergence indicators. Default is "convergence"

`method_replacements`
: Named list of replacement method specifications. Each element should be named with the "method-method_setting" combination (e.g., "RMA-default") and contain a named list with:

  - `method`: Character vector of replacement method names
  - `method_setting`: Character vector of replacement method settings (same length as methods)
  - `power_test_type`: Optional character vector of power test types for each replacement method (same length as methods). If not specified, uses the main power_test_type parameter

  If multiple elements are specified within the vectors, these replacements are applied consecutively in case the previous replacements also failed to converge. Defaults to `NULL`, i.e., omitting repetitions without converged results on method-by-method basis.

`n_repetitions`
: Number of repetitions in each condition. Used to verify that the results contain all repetitions (an error is thrown if any repetition is missing) and for method replacement. Defaults to `1000`.

`overwrite`
: Logical indicating whether to overwrite existing results. If FALSE (default), will skip computation for method-measure combinations that already exist

`results_source`
: Where the results of the computed methods come from: "local" (default) reads unpublished outputs under resources_directory/DGM/results; "release" reads verified results of a published benchmark release (see download_dgm_results()).

`release`
: Benchmark release identifier or catalog, passed to the result reader whenever either source is "release". It is ignored, with a warning, when both `results_source` and `replacement_source` are "local".

`replacement_source`
: Where the results of the replacement methods in `method_replacements` come from: "local" or "release". NULL (default) uses `results_source`. Use `replacement_source = "release"` to replace a new method's failures with published results of established methods.

`...`
: Additional arguments passed to measure functions

Value

Data frame with pairwise comparison scores in long format (method_a, method_b, score)

compute_measures *Compute Multiple Performance measures for a DGM*

Description

This is a high-level wrapper function that computes multiple performance measures for a Data-Generating Mechanism (DGM) and saves the results to CSV files. It provides a clean and extensible interface for computing standard simulation performance measures.

Usage

```
compute_measures(
  dgm_name,
  method,
  method_setting,
  measures = NULL,
  verbose = TRUE,
  power_test_type = "p_value",
  power_threshold_p_value = 0.05,
  power_threshold_bayes_factor = 10,
  estimate_col = "estimate",
  true_effect_col = "mean_effect",
  ci_lower_col = "ci_lower",
  ci_upper_col = "ci_upper",
  p_value_col = "p_value",
  bf_col = "BF",
  convergence_col = "convergence",
  method_replacements = NULL,
  n_repetitions = 1000,
  overwrite = FALSE,
  conditions = NULL,
  results_source = c("local", "release"),
  release = NULL,
  replacement_source = NULL
)
```

Arguments

dgm_name	Character string specifying the DGM name
method	Character vector of method names
method_setting	Character vector of method settings, must be same length as method
measures	Character vector of measures to compute. If NULL, computes all standard measures.
verbose	Print detailed progress of the calculation.

power_test_type
: Character vector specifying the test type for power computation: "p_value" (default), "bayes_factor", or "ci" for each method. The "ci" option is intended for methods reporting only confidence intervals and rejects H0 if the confidence interval excludes zero. If a single value is provided, it is repeated for all methods.

power_threshold_p_value
: Numeric threshold for power computation with p-values. Default is 0.05 (reject H0 if $p < 0.05$).

power_threshold_bayes_factor
: Numeric threshold for power computation with Bayes factors. Default is 10 (reject H0 if BF > 10)

estimate_col
: Character string specifying the column name containing parameter estimates. Default is "estimate"

true_effect_col
: Character string specifying the column name in conditions data frame containing true effect sizes. Default is "mean_effect"

ci_lower_col
: Character string specifying the column name containing lower confidence interval bounds. Default is "ci_lower"

ci_upper_col
: Character string specifying the column name containing upper confidence interval bounds. Default is "ci_upper"

p_value_col
: Character string specifying the column name containing p-values. Default is "p_value"

bf_col
: Character string specifying the column name containing Bayes factors. Default is "BF"

convergence_col
: Character string specifying the column name containing convergence indicators. Default is "convergence"

method_replacements
: Named list of replacement method specifications. Each element should be named with the "method-method_setting" combination (e.g., "RMA-default") and contain a named list with:

  - `method`: Character vector of replacement method names
  - `method_setting`: Character vector of replacement method settings (same length as methods)
  - `power_test_type`: Optional character vector of power test types for each replacement method (same length as methods). If not specified, uses the main power_test_type parameter

  If multiple elements are specified within the vectors, these replacements are applied consecutively in case the previous replacements also failed to converge. Defaults to `NULL`, i.e., omitting repetitions without converged results on method-by-method basis.

n_repetitions
: Number of repetitions in each condition. Used to verify that the results contain all repetitions (an error is thrown if any repetition is missing) and for method replacement. Defaults to `1000`.

overwrite
: Logical indicating whether to overwrite existing results. If FALSE (default), will skip computation for method-measure combinations that already exist

`conditions` Data frame of conditions from dgm_conditions(). If NULL, the selected release's frozen conditions are used when `results_source = "release"` and `dgm_conditions(dgm_name)` otherwise. With local results and release replacements the package's conditions are used, and the release replacement results must cover each of them (missing repetitions are an error).

`results_source` Where the results of the computed methods come from: "local" (default) reads unpublished outputs under resources_directory/DGM/results; "release" reads verified results of a published benchmark release (see download_dgm_results()).

`release` Benchmark release identifier or catalog, passed to the result reader whenever either source is "release". It is ignored, with a warning, when both `results_source` and `replacement_source` are "local".

`replacement_source` Where the results of the replacement methods in `method_replacements` come from: "local" or "release". NULL (default) uses `results_source`. Use `replacement_source = "release"` to replace a new method's failures with published results of established methods.

Value

TRUE upon successfully computation of the results file

Examples

```
## Not run:
# Download DGM results
# Public downloads require no token.
dgm_name <- "no_bias"
download_dgm_results(dgm_name)

# Basic usage; results_source = "release" reads the downloaded published
# results, the default "local" reads unpublished outputs from the resources directory
compute_measures(
  dgm_name        = dgm_name,
  method          = c("mean", "RMA", "PET"),
  method_setting  = c("default", "default", "default"),
  measures        = c("bias", "mse", "coverage"),
  results_source  = "release"
)

# With method replacements for non-converged results
method_replacements <- list(
  "RMA-default" = list(method = "FMA", method_setting = "default"),
  "PET-default" = list(method = c("WLS", "FMA"),
                       method_setting = c("default", "default"))
)

compute_measures(
  dgm_name            = dgm_name,
  method              = c("RMA", "PET"),
  method_setting      = c("default", "default"),
  method_replacements = method_replacements,
```

```
  measures            = c("bias", "mse"),
  results_source      = "release"
)

## End(Not run)
```

`compute_single_measure` *Compute Performance Measures*

Description

This function provides a modular and extensible way to compute performance measures (PM) for Data-Generating Mechanisms (DGMs). It handles different types of measures and automatically determines the required arguments for each measure function.

Usage

```
compute_single_measure(
  dgm_name,
  measure_name,
  method,
  method_setting,
  conditions,
  measure_fun,
  measure_mcse_fun,
  power_test_type = "p_value",
  estimate_col = "estimate",
  true_effect_col = "mean_effect",
  ci_lower_col = "ci_lower",
  ci_upper_col = "ci_upper",
  p_value_col = "p_value",
  bf_col = "BF",
  convergence_col = "convergence",
  power_threshold_p_value = 0.05,
  power_threshold_bayes_factor = 10,
  method_replacements = NULL,
  n_repetitions = 1000,
  overwrite = FALSE,
  results_source = c("local", "release"),
  release = NULL,
  replacement_source = NULL,
  ...
)
```

Arguments

dgm_name	Character string specifying the DGM name
measure_name	Name of the measure to compute (e.g., "bias", "mse")
method	Character vector of method names
method_setting	Character vector of method settings, must be same length as method
conditions	Data frame of conditions from dgm_conditions(). If NULL, the selected release's frozen conditions are used when `results_source = "release"` and `dgm_conditions(dgm_name)` otherwise. With local results and release replacements the package's conditions are used, and the release replacement results must cover each of them (missing repetitions are an error).
measure_fun	Function to compute the measure
measure_mcse_fun	Function to compute the MCSE for the measure
power_test_type	Character vector specifying the test type for power computation: "p_value" (default), "bayes_factor", or "ci" for each method. The "ci" option is intended for methods reporting only confidence intervals and rejects H0 if the confidence interval excludes zero. If a single value is provided, it is repeated for all methods.
estimate_col	Character string specifying the column name containing parameter estimates. Default is "estimate"
true_effect_col	Character string specifying the column name in conditions data frame containing true effect sizes. Default is "mean_effect"
ci_lower_col	Character string specifying the column name containing lower confidence interval bounds. Default is "ci_lower"
ci_upper_col	Character string specifying the column name containing upper confidence interval bounds. Default is "ci_upper"
p_value_col	Character string specifying the column name containing p-values. Default is "p_value"
bf_col	Character string specifying the column name containing Bayes factors. Default is "BF"
convergence_col	Character string specifying the column name containing convergence indicators. Default is "convergence"
power_threshold_p_value	Numeric threshold for power computation with p-values. Default is 0.05 (reject H0 if $p < 0.05$).
power_threshold_bayes_factor	Numeric threshold for power computation with Bayes factors. Default is 10 (reject H0 if $BF > 10$)
method_replacements	Named list of replacement method specifications. Each element should be named with the "method-method_setting" combination (e.g., "RMA-default") and contain a named list with:

- method: Character vector of replacement method names
- method_setting: Character vector of replacement method settings (same length as methods)
- power_test_type: Optional character vector of power test types for each replacement method (same length as methods). If not specified, uses the main power_test_type parameter

If multiple elements are specified within the vectors, these replacements are applied consecutively in case the previous replacements also failed to converge. Defaults to NULL, i.e., omitting repetitions without converged results on method-by-method basis.

n_repetitions — Number of repetitions in each condition. Used to verify that the results contain all repetitions (an error is thrown if any repetition is missing) and for method replacement. Defaults to 1000.

overwrite — Logical indicating whether to overwrite existing results. If FALSE (default), will skip computation for method-measure combinations that already exist

results_source — Where the results of the computed methods come from: "local" (default) reads unpublished outputs under resources_directory/DGM/results; "release" reads verified results of a published benchmark release (see download_dgm_results()).

release — Benchmark release identifier or catalog, passed to the result reader whenever either source is "release". It is ignored, with a warning, when both results_source and replacement_source are "local".

replacement_source — Where the results of the replacement methods in method_replacements come from: "local" or "release". NULL (default) uses results_source. Use replacement_source = "release" to replace a new method's failures with published results of established methods.

... — Additional arguments passed to measure functions

Value

TRUE upon successfully computation of the results file

---

create_empty_result *Create standardized empty method result for convergence failures*

Description

Create standardized empty method result for convergence failures

Usage

```
create_empty_result(method_name, note, extra_columns = NULL)
```

Arguments

method_name	Character string of the method name
note	Character string describing the failure reason
extra_columns	Character vector of additional empty columns to add to the table

Value

Data frame with standardized empty result structure

---

dgm *DGM Method*

Description

S3 Method for defining data-generating mechanisms. See simulate_dgm() for usage and further details.

Usage

```
dgm(dgm_name, settings)
```

Arguments

dgm_name	Character string specifying the DGM type
settings	List containing the required parameters for the DGM or numeric condition_id

Value

A data frame with simulated data following the structure described in the Output Structure section. This is an S3 generic method that dispatches to specific DGM implementations based on dgm_name.

Output Structure

The returned data frame follows a standardized schema that downstream functions rely on. Across the currently implemented DGMs, the following columns are used:

- yi (numeric): The effect size estimate.
- sei (numeric): Standard error of yi.
- ni (integer): Total sample size for the estimate (e.g., sum over groups where applicable).
- es_type (character): Effect size type, used to disambiguate the scale of yi. Currently used values are "SMD" (standardized mean difference / Cohen's d), "logOR" (log odds ratio), and "none" (unspecified generic continuous coefficient).
- study_id (integer/character, optional): Identifier of the primary study/cluster when a DGM yields multiple estimates per study (e.g., Alinaghi2018, PRE). If absent, each row is treated as an independent study.

See Also

simulate_dgm()

Examples

```
simulate_dgm("Carter2019", 1)
```

---

dgm.Alinaghi2018 *Alinaghi and Reed (2018) Data-Generating Mechanism*

Description

This data-generating mechanism simulates univariate regression studies where a variable X affects a continuous outcome Y. Each study estimates the coefficient of X, which consists of a fixed component (alpha1) representing the overall mean effect, and a random component that varies across studies but is constant within each study. In the "Random Effects" environment ("RE"), each study produces one estimate, and the population effect differs across studies. In the "Panel Random Effects" environment ("PRE"), each study has 10 estimates, modeling the common scenario where multiple estimates per study are available, with publication selection targeting the study rather than individual estimates.

The description and code is based on Hong and Reed (2021). The data-generating mechanism was introduced in Alinaghi and Reed (2018).

Usage

```
## S3 method for class 'Alinaghi2018'
dgm(dgm_name, settings)
```

Arguments

dgm_name	DGM name (automatically passed)
settings	List containing
	environment Type of the simulation environment. One of "FE", "RE", or "PRE".
	mean_effect Mean effect
	bias Type of publication bias. One of "none", "positive", or "significant".

Details

This data-generating mechanism is based on Alinaghi & Reed (2018), who study univariate regression models where a variable X affects a continuous variable Y. The parameter of interest is the coefficient on X. In the "Random Effects" environment ("RE"), each study produces one estimate, and the population effect differs across studies. The coefficient on X equals a fixed component (alpha1) plus a random component that is fixed within a study but varies across studies. The overall mean effect of X on Y is given by alpha1. In the "Panel Random Effects" environment ("PRE"), each study has 10 estimates, modeling the common scenario where multiple estimates per study are available. In this environment, effect estimates and standard errors are simulated to be more similar

within studies than across studies, and publication selection targets the study rather than individual estimates (a study must have at least 7 out of 10 estimates that are significant or correctly signed.).

A distinctive feature of Alinaghi & Reed's experiments is that the number of effect size estimates is fixed before publication selection, making the meta-analyst's sample size endogenous and affected by the effect size. Large population effects are subject to less publication selection, as most estimates satisfy the selection criteria (statistical significance or correct sign). The sample size of all primary studies is fixed at 100 observations. (Neither the number of estimates nor the sample size of primary studies can be changed in the current implementation of the function.)

Another feature is the separation of statistical significance and sign of the estimated effect as criteria for selection. Significant/correctly-signed estimates are always "published," while insignificant/wrong-signed estimates have only a 10% chance of being published. This allows for different and sometimes conflicting consequences for estimator performance.

Value

Data frame with

yi effect size

sei standard error

ni sample size

study_id study identifier

es_type effect size type

Author(s)

František Bartoš <f.bartos96@gmail.com> (adapted from Hong and Reed 2021)

References

Alinaghi N, Reed WR (2018). "Meta-analysis and publication bias: How well does the FAT-PET-PEESE procedure work?" *Research Synthesis Methods*, **9**(2), 285-311. doi:10.1002/jrsm.1298.

Hong S, Reed WR (2021). "Using Monte Carlo experiments to select meta-analytic estimators." *Research Synthesis Methods*, **12**(2), 192-215. doi:10.1002/jrsm.1467.

See Also

`dgm()`, `validate_dgm_setting()`

dgm.Bom2019 *Bom and Rachinger (2019) Data-Generating Mechanism*

Description

Simulates univariate regression environments to estimate the effect of X1 on Y (parameter alpha1). Effect heterogeneity is introduced via an omitted variable (X2) correlated with X1, whose coefficient (alpha2) is randomly distributed with mean zero and variance sigma2_h.

The description and code is based on Hong and Reed (2021). The data-generating mechanism was introduced in Bom and Rachinger (2019).

Usage

```
## S3 method for class 'Bom2019'
dgm(dgm_name, settings)
```

Arguments

dgm_name	DGM name (automatically passed)
settings	List containing

mean_effect Mean effect

effect_heterogeneity Mean effect heterogeneity

bias Proportion of studies affected by publication bias

n_studies Number of effect size estimates

sample_sizes Sample sizes of the effect size estimates. A vector of sample sizes needs to be supplied. The sample sizes in the vector are sequentially reused until all effect size estimates are generated.

Details

This function simulates univariate regression environments, focusing on estimating the effect of a variable X1 on a dependent variable Y, represented by the parameter alpha1. The simulation introduces variation in the standard errors of estimated effects by allowing sample sizes to differ across primary studies. Effect heterogeneity is modeled through an omitted variable (X2) that is correlated with X1, where the coefficient on the omitted variable, alpha2, is randomly distributed across studies with mean zero and variance sigma2_h.

Publication selection is modeled in two regimes: (1) no selection, and (2) 50% selection. Under 50% selection, each estimate has a 50% chance of being evaluated for inclusion. If selected, only positive and statistically significant estimates are published; otherwise, new estimates are generated until this criterion is met. This process continues until the meta-analyst's sample reaches its predetermined size.

Value

Data frame with

yi effect size

sei standard error

ni sample size

es_type effect size type

Author(s)

František Bartoš <f.bartos96@gmail.com> (adapted from Hong and Reed 2021)

References

Bom PR, Rachinger H (2019). "A kinked meta-regression model for publication bias correction." *Research Synthesis Methods*, **10**(4), 497-514. doi:10.1002/jrsm.1352.

Hong S, Reed WR (2021). "Using Monte Carlo experiments to select meta-analytic estimators." *Research Synthesis Methods*, **12**(2), 192-215. doi:10.1002/jrsm.1467.

See Also

dgm(), validate_dgm_setting()

---

dgm.Carter2019 *Carter et al. (2019) Data-Generating Mechanism*

Description

This data-generating mechanism simulates primary studies estimating treatment effects using Cohen's d. The observed effect size is modeled as a fixed mean plus random heterogeneity across studies, with sample sizes varying to generate differences in standard errors. The simulation introduces publication bias via a selection algorithm where the probability of publication depends nonlinearly on the sign and p-value of the effect, with regimes for no, medium, and strong publication bias. It also incorporates questionable research practices (QRPs) such as optional outlier removal, selection between dependent variables, use of moderators, and optional stopping.

The description and code is based on Hong and Reed (2021). The data-generating mechanism was introduced in Carter et al. (2019).

Usage

```
## S3 method for class 'Carter2019'
dgm(dgm_name, settings)
```

Arguments

`dgm_name`	DGM name (automatically passed)
`settings`	List containing

mean_effect Mean effect

effect_heterogeneity Mean effect heterogeneity

bias Degree of publication bias with one of following levels: `"none"`, `"medium"`, `"high"`.

QRP Degree of questionable research practices with one of following levels: `"none"`, `"medium"`, `"high"`.

n_studies Number of effect size estimates

Details

This simulation environment is based on the framework described by Carter, Schönbrodt, Gervais, and Hilgard (2019). In this setup, primary studies estimate the effect of a treatment using Cohen's d as the effect size metric. The observed difference between treatment and control groups is modeled as the sum of a fixed effect (alpha1) and a random component, which introduces effect heterogeneity across studies. The degree of heterogeneity is controlled by the parameter sigma2_h. Variability in the standard errors of d is generated by simulating primary studies with different sample sizes.

The simulation incorporates two main types of distortions in the research environment. First, a publication selection algorithm is used, where the probability of a study being "published" depends nonlinearly on both the sign of the estimated effect and its p-value. Three publication selection regimes are modeled: "No Publication Bias," "Medium Publication Bias," and "Strong Publication Bias," each defined by different parameters in the selection algorithm. Second, the simulation includes four types of questionable research practices (QRPs): (a) optional removal of outliers, (b) optional selection between two dependent variables, (c) optional use of moderators, and (d) optional stopping.

Value

Data frame with

yi effect size

sei standard error

ni sample size

es_type effect size type

Author(s)

František Bartoš <f.bartos96@gmail.com> (adapted from Hong and Reed 2021)

References

Carter EC, Schönbrodt FD, Gervais WM, Hilgard J (2019). "Correcting for bias in psychology: A comparison of meta-analytic methods." *Advances in Methods and Practices in Psychological Science*, **2**(2), 115-144. doi:10.1177/2515245919847196.

Hong S, Reed WR (2021). "Using Monte Carlo experiments to select meta-analytic estimators." *Research Synthesis Methods*, **12**(2), 192-215. doi:10.1002/jrsm.1467.

See Also

`dgm()`, `validate_dgm_setting()`

`dgm.default` *Default DGM handler*

Description

Default DGM handler

Usage

```
## Default S3 method:
dgm(dgm_name, settings)
```

Arguments

`dgm_name`	Character string specifying the DGM type
`settings`	List containing the required parameters for the DGM or numeric condition_id

Value

Throws an error indicating the DGM type is unknown. This default method is only called when no specific DGM implementation is found for the given `dgm_name`.

`dgm.no_bias` *Normal Unbiased Data-Generating Mechanism*

Description

An example data-generating mechanism to simulate effect sizes without publication bias.

Usage

```
## S3 method for class 'no_bias'
dgm(dgm_name, settings)
```

Arguments

`dgm_name`	DGM name (automatically passed)
`settings`	List containing
	mean_effect Mean effect
	heterogeneity Effect heterogeneity
	n_studies Number of effect size estimates

Details

Sample sizes of individual effect size estimates are generated from a negative binomial distribution based on empirical sample size distribution presented in Appendix B of Maier et al. (2023)

Value

Data frame with

yi effect size

sei standard error

ni sample size

es_type effect size type

Author(s)

František Bartoš <f.bartos96@gmail.com>

References

Maier M, Bartoš F, Wagenmakers E (2023). "Robust Bayesian meta-analysis: Addressing publication bias with model-averaging." *Psychological Methods*, **28**(1), 107-122. doi:10.1037/met0000405.

See Also

`dgm()`, `validate_dgm_setting()`

`dgm.Stanley2017` *Stanley, Doucouliagos, and Ioannidis (2017) Data-Generating Mechanism*

Description

Simulates two scenarios for meta-analysis studies investigating the effect of a treatment in: (1) Log Odds Ratio scenario, where the outcome is binary and effect heterogeneity is controlled by a random component, and (2) Cohen's d scenario, where the outcome is continuous and effect heterogeneity is introduced through a random component. Both scenarios allow for varying sample sizes and publication selection regimes, affecting the inclusion of study estimates based on their statistical significance and sign.

The description and code is based on Hong and Reed (2021). The data-generating mechanism was introduced in Stanley et al. (2017).

Usage

```
## S3 method for class 'Stanley2017'
dgm(dgm_name, settings)
```

Arguments

dgm_name	DGM name (automatically passed)
settings	List containing

environment Type of the simulation environment. One of "logOR" or "SMD".

mean_effect Mean effect

effect_heterogeneity Mean effect heterogeneity

bias Proportion of studies affected by publication bias

n_studies Number of effect size estimates

sample_sizes Sample sizes of the effect size estimates. A vector of sample sizes needs to be supplied. The sample sizes in the vector are sequentially reused until all effect size estimates are generated.

Details

This function simulates two meta-analysis scenarios to evaluate the effect of a binary treatment variable (treat = {0, 1}) on study outcomes, incorporating both effect heterogeneity and publication selection mechanisms.

In the Log Odds Ratio ("logOR") scenario, primary studies assess the impact of treatment on a binary success indicator (Y = 1). The control group has a fixed 10% probability of success, while the treatment group's probability is increased by a fixed effect and a mean-zero random component, whose variance (sigma2_h) controls effect heterogeneity. Each study estimates a logistic regression, with the coefficient on treat (alpha1) as the effect of interest. Study sample sizes vary, resulting in different standard errors for estimated effects.

In the Cohen's d ("SMD") scenario, the outcome variable is continuous. The treatment effect is modeled as a fixed effect (alpha1) plus a random component (variance sigma2_h). Each study computes Cohen's d, the standardized mean difference between treatment and control groups. Study sample sizes vary, affecting the standard errors of d.

Publication selection is modeled in two regimes: (1) no selection, and (2) 50% selection. Under 50% selection, each estimate has a 50% chance of being evaluated for inclusion. If selected, only positive and statistically significant estimates are published; otherwise, new estimates are generated until this criterion is met. This process continues until the meta-analyst's sample reaches its predetermined size.

Value

Data frame with

yi effect size

sei standard error

ni sample size

es_type effect size type

Author(s)

František Bartoš <f.bartos96@gmail.com> (adapted from Hong and Reed 2021)

References

Hong S, Reed WR (2021). "Using Monte Carlo experiments to select meta-analytic estimators." *Research Synthesis Methods*, **12**(2), 192-215. doi:10.1002/jrsm.1467.

Stanley TD, Doucouliagos H, Ioannidis JP (2017). "Finding the power to reduce publication bias." *Statistics in Medicine*, **36**(10), 1580-1598. doi:10.1002/sim.7228.

See Also

dgm(), validate_dgm_setting()

---

dgm_conditions *Return Pre-specified DGM Settings*

Description

This function returns the list of pre-specified settings for a given Data Generating Mechanism (DGM).

Usage

```
dgm_conditions(dgm_name)

get_dgm_condition(dgm_name, condition_id)
```

Arguments

dgm_name	Character string specifying the DGM type
condition_id	which conditions should settings be returned for.

Value

A data frame containing the pre-specified settings including a condition_id column which maps settings id to the corresponding settings.

Examples

```
head(dgm_conditions("Carter2019"))
get_dgm_condition("Carter2019", condition_id = 1)

head(dgm_conditions("Alinaghi2018"))

head(dgm_conditions("Stanley2017"))
```

download_dgm *Download Benchmark Datasets, Results, and Measures*

Description

Download immutable catalog-selected direct files or ZIP units. Archives and extracted members are checked against size, SHA-256 and MD5. A condition/metric filter may fetch other members of its selected unit. Public downloads require no token; verified member caches avoid new downloads.

Usage

```
download_dgm_datasets(
  dgm_name,
  overwrite = FALSE,
  progress = TRUE,
  max_try = 10,
  release = NULL,
  condition_id = NULL
)

download_dgm_results(
  dgm_name,
  overwrite = FALSE,
  progress = TRUE,
  max_try = 10,
  method = NULL,
  method_setting = NULL,
  release = NULL
)

download_dgm_measures(
  dgm_name,
  overwrite = FALSE,
  progress = TRUE,
  max_try = 10,
  method = NULL,
  method_setting = NULL,
  release = NULL,
  measure = NULL,
  replacement = NULL
)

download_dgm_metadata(
  dgm_name,
  overwrite = FALSE,
  progress = TRUE,
  max_try = 10,
```

```
  release = NULL
)
```

Arguments

dgm_name	DGM name.
overwrite	Re-download selected files even if their cached copies verify.
progress	Display download progress.
max_try	Maximum attempts per file.
release	Release identifier or catalog; NULL uses the package default.
condition_id	Optional condition ID(s) for datasets.
method	Optional method name(s); NULL selects all available methods.
method_setting	Optional method setting(s).
measure	Optional measure name(s); pairwise comparisons require "pairwise".
replacement	Whether replacement measures are selected; NULL downloads both variants.

Value

Invisible TRUE on success or FALSE when an interactive download is declined.

---

measure *Get performance measure function*

Description

Returns the function for computing a specific performance measure. Can also be used to list available measures.

Usage

```
measure(measure, ...)
```

Arguments

measure	Character string specifying the measure name. If missing, returns a vector of all available measures.
...	Additional arguments passed to methods.

Value

A function that computes the requested measure, or a character vector of measure names.

measures *Performance Measures and Monte Carlo Standard Errors*

Description

A comprehensive set of functions for computing performance measures and their Monte Carlo Standard Errors (MCSE) for simulation studies. All functions are based on definitions from Table 3 in Siepe et al. (2024). Winkler interval score is defined in Winkler (1972). Positive and negative likelihood ratios are defined in Huang and Trinquart (2023) and Deeks and Altman (2004). Also see Morris et al. (2019) for additional details. Bias and relative bias were modified to account for possibly different true values across repetitions.

Usage

```
bias(theta_hat, theta)

bias_mcse(theta_hat)

relative_bias(theta_hat, theta)

relative_bias_mcse(theta_hat, theta)

mse(theta_hat, theta)

mse_mcse(theta_hat, theta)

rmse(theta_hat, theta)

rmse_mcse(theta_hat, theta)

empirical_variance(theta_hat)

empirical_variance_mcse(theta_hat)

empirical_se(theta_hat)

empirical_se_mcse(theta_hat)

coverage(ci_lower, ci_upper, theta)

coverage_mcse(ci_lower, ci_upper, theta)

power(test_rejects_h0)

power_mcse(test_rejects_h0)

mean_ci_width(ci_upper, ci_lower)
```

```
mean_ci_width_mcse(ci_upper, ci_lower)

mean_generic_statistic(G)

mean_generic_statistic_mcse(G)

positive_likelihood_ratio(tp, fp, fn, tn)

positive_likelihood_ratio_mcse(tp, fp, fn, tn)

negative_likelihood_ratio(tp, fp, fn, tn)

negative_likelihood_ratio_mcse(tp, fp, fn, tn)

interval_score(ci_lower, ci_upper, theta, alpha = 0.05)

interval_score_mcse(ci_lower, ci_upper, theta, alpha = 0.05)
```

Arguments

`theta_hat`	Vector of parameter estimates from simulations
`theta`	True parameter value
`ci_lower`	Vector of lower confidence interval bounds
`ci_upper`	Vector of upper confidence interval bounds
`test_rejects_h0`	Logical vector indicating whether statistical tests reject the null hypothesis
`G`	Vector of generic statistics from simulations
`tp`	Numeric with the count of true positive hypothesis tests
`fp`	Numeric with the count of false positive hypothesis tests
`fn`	Numeric with the count of false negative hypothesis tests
`tn`	Numeric with the count of true negative hypothesis tests
`alpha`	Numeric indicating the 1 - coverage level for interval_score calculation

Details

The package provides the following performance measures and their corresponding MCSE functions:

- `bias(theta_hat, theta)`: Bias estimate
- `relative_bias(theta_hat, theta)`: Relative bias estimate
- `mse(theta_hat, theta)`: Mean Square Error
- `rmse(theta_hat, theta)`: Root Mean Square Error
- `empirical_variance(theta_hat)`: Empirical variance
- `empirical_se(theta_hat)`: Empirical standard error

- `coverage(ci_lower, ci_upper, theta)`: Coverage probability
- `mean_ci_width(ci_upper, ci_lower)`: Mean confidence interval width
- `interval_score(ci_lower, ci_upper, theta, alpha)`: interval_score
- `power(test_rejects_h0)`: Statistical power
- `positive_likelihood_ratio(tp, fp, fn, tn)`: Log positive likelihood ratio
- `negative_likelihood_ratio(tp, fp, fn, tn)`: Log negative likelihood ratio
- `mean_generic_statistic(G)`: Mean of any generic statistic

Value

Each metric function returns a numeric value representing the performance measure. Each MCSE function returns a numeric value representing the Monte Carlo standard error.

References

Deeks JJ, Altman DG (2004). "Diagnostic tests 4: likelihood ratios." *BMJ*, **329**(7458), 168–169. doi:10.1136/bmj.329.7458.168.

Huang Q, Trinquart L (2023). "Relative likelihood ratios for neutral comparisons of statistical tests in simulation studies." *Biometrical Journal*, **66**(1), 2200102. doi:10.1002/bimj.202200102.

Morris TP, White IR, Crowther MJ (2019). "Using simulation studies to evaluate statistical methods." *Statistics in Medicine*, **38**(11), 2074–2102. doi:10.1002/sim.8086.

Siepe BS, Bartoš F, Morris TP, Boulesteix A, Heck DW, Pawel S (2024). "Simulation studies for methodological research in psychology: A standardized template for planning, preregistration, and reporting." *Psychological Methods*. doi:10.1037/met0000695.

Winkler RL (1972). "A decision-theoretic approach to interval estimation." *Journal of the American Statistical Association*, **67**(337), 187–191. doi:10.1080/01621459.1972.10481224.

Examples

```
# Generate some example data
set.seed(123)
theta_true <- 0.5
theta_estimates <- rnorm(1000, mean = theta_true, sd = 0.1)

# Compute bias and its MCSE
bias_est <- bias(theta_estimates, theta_true)
bias_se <- bias_mcse(theta_estimates)

# Compute MSE and its MCSE
mse_est <- mse(theta_estimates, theta_true)
mse_se <- mse_mcse(theta_estimates, theta_true)

# Example with coverage
ci_lower <- theta_estimates - 1.96 * 0.1
ci_upper <- theta_estimates + 1.96 * 0.1
```

```
coverage_est <- coverage(ci_lower, ci_upper, theta_true)
coverage_se <- coverage_mcse(ci_lower, ci_upper, theta_true)
```

`measure_mcse` *Get performance measure MCSE function*

Description

Returns the function for computing the Monte Carlo Standard Error (MCSE) of a specific performance measure. Can also be used to list available measures with MCSE functions.

Usage

```
measure_mcse(measure, ...)
```

Arguments

`measure`	Character string specifying the measure name. If missing, returns a vector of all available measures.
`...`	Additional arguments passed to methods.

Value

A function that computes the MCSE of the requested measure, or a character vector of measure names.

`method` *Method Method*

Description

S3 Method for defining methods. See `run_method()` for usage and further details.

Usage

```
method(method_name, data, settings)
```

Arguments

`method_name`	Character string specifying the method type
`data`	Data frame containing yi (effect sizes) and sei (standard errors)
`settings`	Either a character identifying a method version or list containing method-specific settings. An emty input will result in running the default (first implemented) version of the method.

Value

A data frame with method results following the structure described in the Output Structure section. This is an S3 generic method that dispatches to specific method implementations based on `method_name`.

Output Structure

The returned data frame follows a standardized schema that downstream functions rely on. All methods return the following columns:

- `method` (character): The name of the method used.
- `estimate` (numeric): The meta-analytic effect size estimate.
- `standard_error` (numeric): Standard error of the estimate.
- `ci_lower` (numeric): Lower bound of the 95% confidence interval.
- `ci_upper` (numeric): Upper bound of the 95% confidence interval.
- `p_value` (numeric): P-value for the estimate.
- `BF` (numeric): Bayes Factor for the estimate.
- `convergence` (logical): Whether the method converged successfully.
- `note` (character): Additional notes describing convergence issues.

Some methods may include additional method-specific columns beyond these standard columns. Use `get_method_extra_columns()` to query which additional columns a particular method returns.

See Also

`run_method()`

Examples

```
data <- data.frame(
  yi = c(0.2, 0.3, 0.1, 0.4),
  sei = c(0.1, 0.15, 0.08, 0.12)
)
result <- run_method("RMA", data, "default")
```

`method.AK` *AK Method*

Description

Implements the Andrews & Kasy (AK) method for publication bias correction in meta-analysis. The AK method categorizes estimated effects into groups with different probabilities of being published. AK1 uses symmetric selection grouping estimates into significant (|t| >= 1.96) and insignificant (|t| < 1.96) estimates. AK2 uses asymmetric selection with four groups based on both significance and sign: highly significant positive/negative effects and marginally significant positive/negative effects, each with different publication probabilities. See Andrews and Kasy (2019) for details.

Usage

```
## S3 method for class 'AK'
method(method_name, data, settings)
```

Arguments

method_name	Method name (automatically passed)
data	Data frame with yi (effect sizes), sei (standard errors), and study_id (for clustering wherever available)
settings	List of method settings (see Details)

Details

The following settings are implemented

"default" Uses AK1 estimator (symmetric selection)

"AK1" Symmetric selection model grouping estimates into significant (|t| >= 1.96) and insignificant (|t| < 1.96) categories with relative publication probabilities of 1 and p1 respectively.

"AK2" Asymmetric selection model with four groups based on t-statistics: (a) t >= 1.96, (b) t < -1.96, (c) -1.96 <= t < 0, and (d) 0 <= t < 1.96, with relative publication probabilities of 1, p1, p2, and p3 respectively.

Value

Data frame with AK results

Author(s)

František Bartoš <f.bartos96@gmail.com>

References

Andrews I, Kasy M (2019). "Identification of and correction for publication bias." *American Economic Review*, **109**(8), 2766–2794. doi:10.1257/aer.20180310.

Examples

```
# Generate some example data
data <- data.frame(
  yi = c(0.2, 0.3, 0.1, 0.4, 0.25),
  sei = c(0.1, 0.15, 0.08, 0.12, 0.09)
)

# Apply AK method
result <- run_method("AK", data, "default")
print(result)
```

method.default *Default method handler*

Description

Default method handler

Usage

```
## Default S3 method:
method(method_name, data, settings = list())
```

Arguments

method_name	Character string specifying the method type
data	Data frame containing yi (effect sizes) and sei (standard errors)
settings	Either a character identifying a method version or list containing method-specific settings. An emty input will result in running the default (first implemented) version of the method.

Value

Throws an error indicating the method type is unknown. This default method is only called when no specific method implementation is found for the given method_name.

method.EK *Endogenous Kink Method*

Description

Implements the endogenous kink (EK) method proposed by Bom and Rachinger for publication bias correction in meta-analysis. This method modifies the PET-PEESE approach by incorporating a non-linear relationship between publication bias and standard errors through a kinked regression specification. The method recognizes that when the true effect is non-zero, there is minimal publication selection when standard errors are very small (since most estimates are significant), but selection increases as standard errors grow. The kink point is endogenously determined using a two-step procedure based on the confidence interval of the initial effect estimate. See Bom and Rachinger (2019) for details.

Usage

```
## S3 method for class 'EK'
method(method_name, data, settings = NULL)
```

Arguments

method_name	Method name (automatically passed)
data	Data frame with yi (effect sizes) and sei (standard errors)
settings	List of method settings (no settings version are implemented)

Value

Data frame with EK results

Author(s)

František Bartoš <f.bartos96@gmail.com>

References

Bom PR, Rachinger H (2019). "A kinked meta-regression model for publication bias correction." *Research Synthesis Methods*, **10**(4), 497-514. doi:10.1002/jrsm.1352.

Examples

```
# Generate some example data
data <- data.frame(
  yi = c(0.2, 0.3, 0.1, 0.4, 0.25),
  sei = c(0.1, 0.15, 0.08, 0.12, 0.09)
)

# Apply EK method
result <- run_method("EK", data)
print(result)
```

---

method.FMA *Fixed Effects Meta-Analysis Method*

Description

Implements the publication bias-unadjusted fixed effects meta-analysis.

Usage

```
## S3 method for class 'FMA'
method(method_name, data, settings)
```

Arguments

method_name	Method name (automatically passed)
data	Data frame with yi (effect sizes) and sei (standard errors)
settings	List of method settings (see Details)

Details

The following settings are implemented

`"default"` T-distribution adjustment (`test = "t"`) and cluster robust standard errors with small-sample adjustment (if converged, otherwise no small-sample adjustment or no cluster robust standard errors) for fixed effects meta-analysis if `study_ids` is specified in the data

Value

Data frame with FMA results

Author(s)

František Bartoš <f.bartos96@gmail.com>

References

There are no references for Rd macro `\insertAllCites` on this help page.

Examples

```
# Generate some example data
data <- data.frame(
  yi = c(0.2, 0.3, 0.1, 0.4, 0.25),
  sei = c(0.1, 0.15, 0.08, 0.12, 0.09)
)

# Apply FMA method
result <- run_method("FMA", data)
print(result)
```

`method.MAIVE` *MAIVE: Meta-Analysis Instrumental Variable Estimator*

Description

Implements the MAIVE method for publication bias correction using instrumental variable estimation with variance instrumentation. MAIVE addresses spurious precision in meta-analysis by instrumenting standard errors with inverse sample sizes, providing consistent estimates even when precision is manipulated through p-hacking (Irsova et al. 2025).

The method implements several estimators:

- PET (Precision-Effect Test): Linear precision-effect model
- PEESE (Precision-Effect Estimate with Standard Error): Quadratic model
- PET-PEESE: Conditional selection based on PET significance
- EK (Endogenous Kink): Flexible bias function with kink point
- WAIVE: Robust variant with outlier downweighting

Usage

```
## S3 method for class 'MAIVE'
method(method_name, data, settings)
```

Arguments

method_name	Method identifier (automatically passed by framework)
data	Data frame with yi (effect sizes), sei (standard errors), ni (sample sizes), and optionally study_id for clustering
settings	List of method settings from method_settings.MAIVE()

Details

MAIVE uses inverse sample sizes (1/N) as instruments for variances (SE^2) in the first stage, then uses the instrumented variances in second-stage PET/PEESE models. This approach provides consistent estimation when precision is endogenous due to p-hacking or selective reporting.

The Anderson-Rubin confidence interval is robust to weak instruments and is automatically computed for unweighted IV estimators when feasible (n < 5000). For weighted estimators or large samples, standard CIs are used. PublicationBiasBenchmark targets MAIVE 0.2.4 and persists upstream MAIVE warnings in the standardized `note` column.

WAIVE extends MAIVE by downweighting: (1) negative residuals (spurious precision) using exponential decay, and (2) extreme residuals (|z| > 2) as potential outliers. This provides additional robustness against publication bias and outliers. When IV is used and the first-stage F-statistic is numeric and below 10, PublicationBiasBenchmark blanks the standardized inferential outputs (`estimate`, `standard_error`, `ci_lower`, `ci_upper`, `p_value`) to `NA` while keeping `convergence = TRUE`. These rows remain available as diagnostics and are treated as missing estimates in downstream performance summaries rather than as convergence failures.

Available settings (see method_settings.MAIVE()):

default PET-PEESE with IV, unweighted, levels first-stage

PET PET with IV, unweighted

PEESE PEESE with IV, unweighted

EK Endogenous Kink model with IV

weighted PET-PEESE with MAIVE-adjusted weighting

WAIVE Robust WAIVE variant with outlier downweighting

log_first_stage PET-PEESE with log-linear first stage

no_IV Standard PET-PEESE without instrumentation (baseline)

Value

Single-row data frame with standardized output columns:

method Method identifier

estimate Meta-analytic effect size estimate

standard_error Standard error of estimate

ci_lower, ci_upper 95% confidence interval bounds (Anderson-Rubin if available)

p_value Two-tailed p-value

BF Bayes factor (NA for MAIVE)

convergence Logical convergence indicator

note Error messages if any

first_stage_f First-stage F-statistic for instrument strength

hausman_stat Hausman test statistic comparing IV vs OLS

bias_p_value P-value for publication bias test

used_ar_ci Whether Anderson-Rubin CI was used

ar_ci_available Whether AR CI was computed successfully

instrument_strength Instrument strength classification returned by MAIVE or derived from the first-stage F-statistic

Author(s)

Petr Cala <cala.p@seznam.cz>

References

Irsova Z, Bom PR, Havranek T, Rachinger H (2025). "Spurious precision in meta-analysis of observational research." *Nature Communications*, **16**, 8454. doi:10.1038/s41467025632610.

See Also

run_method(), method_settings(), method_extra_columns()

Examples

```
## Not run:
# Generate test data
data <- simulate_dgm("Stanley2017", condition_id = 1)

# Apply default MAIVE (PET-PEESE with IV)
result <- run_method("MAIVE", data, "default")

# Apply WAIVE variant
result_waive <- run_method("MAIVE", data, "WAIVE")

# Apply weighted MAIVE
result_weighted <- run_method("MAIVE", data, "weighted")

# View available configurations
method_settings("MAIVE")

## End(Not run)
```

`method.MAN` *Meta-Analysis of Nonaffirmative Studies (MAN) Method*

Description

Implements standard meta-analysis of only the nonaffirmative studies (MAN) which can serve as a sensitivity analysis for worst-case meta-analytic point estimate for maximal publication bias under the selection model. effects of p-hacking and publication bias in meta-analysis. See Mathur and VanderWeele (2020) for details.

Usage

```
## S3 method for class 'MAN'
method(method_name, data, settings)
```

Arguments

`method_name`	Method name (automatically passed)
`data`	Data frame with yi (effect sizes) and sei (standard errors)
`settings`	List of method settings (see Details.)

Details

The following settings are implemented

`"default"` MAN with affirmative results defined by positive direction `favor_positive = TRUE` and statistical significance `alpha_select = 0.05`

Value

Data frame with MAN results

Author(s)

Frantisek Bartos <f.bartos96@gmail.com>

References

Mathur MB, VanderWeele TJ (2020). "Sensitivity analysis for publication bias in meta-analyses." *Journal of the Royal Statistical Society Series C: Applied Statistics*, **69**(5), 1091–1119. doi:10.1111/rssc.12440.

Examples

```
# Generate some example data with at least one nonaffirmative study
data <- data.frame(
  yi = c(0.20, 0.30, 0.10, -0.05, 0.12),
  sei = c(0.10, 0.15, 0.08, 0.12, 0.09)
)

# Apply MAN method
result <- run_method("MAN", data)
print(result)
```

`method.mean` *Mean Method*

Description

Implements the unweighted mean method. I.e., the mean of observed effect sizes.

Usage

```
## S3 method for class 'mean'
method(method_name, data, settings)
```

Arguments

`method_name`	Method name (automatically passed)
`data`	Data frame with yi (effect sizes)
`settings`	List of method settings (see Details)

Details

The following settings are implemented

`"default"` No settings

Value

Data frame with mean results

Author(s)

František Bartoš <f.bartos96@gmail.com>

References

There are no references for Rd macro `\insertAllCites` on this help page.

Examples

```
# Generate some example data
data <- data.frame(
  yi = c(0.2, 0.3, 0.1, 0.4, 0.25),
  sei = c(0.1, 0.15, 0.08, 0.12, 0.09)
)

# Apply mean method
result <- run_method("mean", data)
print(result)
```

`method.MMPH` *Mixture Model of P-Hacking (MMPH)*

Description

Implements mixture model of p-hacking as described in Moss and De Bin (2023).

The model is estimated via `publipha::phma()`.

Usage

```
## S3 method for class 'MMPH'
method(method_name, data, settings)
```

Arguments

`method_name`	Method name (automatically passed)
`data`	Data frame with yi (effect sizes) and sei (standard errors)
`settings`	List of method settings (see Details.)

Details

The following settings are implemented

`"default"` MMPH with Stan control settings `control = list(adapt_delta = 0.95, max_treedepth = 15)`, `warmup = 2000`, and `iter = 4000`, and `chains = 3`, and convergence thresholds `max_r_hat = 1.05` and `min_ess = 300`

A fit is considered converged if the R-hat of the effect size is below `max_r_hat` and its effective sample size is above `min_ess`. Both thresholds must be specified with custom settings.

Value

Data frame with MMPH results

Author(s)

Frantisek Bartos <f.bartos96@gmail.com>

References

Moss J, De Bin R (2023). "Modelling publication bias and p-hacking." *Biometrics*, **79**(1), 319–331. doi:10.1111/biom.13560.

Examples

```
# Generate some example data
data <- data.frame(
  yi      = c(0.2, 0.3, 0.1, 0.4, 0.25),
  sei     = c(0.1, 0.15, 0.08, 0.12, 0.09),
  es_type = "SMD"
)

# Apply MAN method
result <- run_method("MMPH", data)
print(result)
```

`method.pcurve` *pcurve (P-Curve) Method*

Description

Implements the p-Curve method which analyzes the distribution of p-values from significant studies to assess whether the significant findings reflect true effects or QRP/publication bias. The method also provides tests for the evidential value, lack of evidential value, and p-hacking. See Simonsohn et al. (2014) for details.

The current implementation does not provide a test against the null hypothsis of no effect and does not produce confidence intervals of the estimate.

Usage

```
## S3 method for class 'pcurve'
method(method_name, data, settings)
```

Arguments

`method_name`	Method name (automatically passed)
`data`	Data frame with yi (effect sizes), sei (standard errors), and ni (sample sizes wherever available, otherwise set to Inf)
`settings`	List of method settings (see Details)

Details

The following settings are implemented

`"default"` no options

Value

Data frame with P-Curve results

Author(s)

František Bartoš <f.bartos96@gmail.com>

References

Simonsohn U, Nelson LD, Simmons JP (2014). "p-curve and effect size: Correcting for publication bias using only significant results." *Perspectives on Psychological Science*, **9**(6), 666–681. doi:10.1177/1745691614553988.

Examples

```
# Generate some example data
data <- data.frame(
  yi = c(0.2, 0.3, 0.1, 0.4, 0.25),
  sei = c(0.1, 0.15, 0.08, 0.12, 0.09)
)

# Apply pcurve method
result <- run_method("pcurve", data)
print(result)
```

---

method.PEESE *PEESE (Precision-Effect Estimate with Standard Errors) Method*

Description

Implements the Precision-Effect Estimate with Standard Errors method for publication bias correction. PEESE regresses effect sizes against standard errors^2 to correct for publication bias. The intercept represents the bias-corrected effect size estimate. See Stanley and Doucouliagos (2014) for details.

Usage

```
## S3 method for class 'PEESE'
method(method_name, data, settings = NULL)
```

Arguments

method_name	Method name (automatically passed)
data	Data frame with yi (effect sizes) and sei (standard errors)
settings	List of method settings (no settings version are implemented)

Value

Data frame with PEESE results

Author(s)

František Bartoš <f.bartos96@gmail.com>

References

Stanley TD, Doucouliagos H (2014). "Meta-regression approximations to reduce publication selection bias." *Research Synthesis Methods*, **5**(1), 60–78. doi:10.1002/jrsm.1095.

Examples

```
# Generate some example data
data <- data.frame(
  yi = c(0.2, 0.3, 0.1, 0.4, 0.25),
  sei = c(0.1, 0.15, 0.08, 0.12, 0.09)
)

# Apply PEESE method
result <- run_method("PEESE", data)
print(result)
```

---

method.PET *PET (Precision-Effect Test) Method*

Description

Implements the Precision-Effect Test for publication bias correction. PET regresses effect sizes against standard errors to test for and correct publication bias. The intercept represents the bias-corrected effect size estimate. See Stanley and Doucouliagos (2014) for details.

Usage

```
## S3 method for class 'PET'
method(method_name, data, settings = NULL)
```

Arguments

method_name	Method name (automatically passed)
data	Data frame with yi (effect sizes) and sei (standard errors)
settings	List of method settings (no settings version are implemented)

Value

Data frame with PET results

Author(s)

František Bartoš <f.bartos96@gmail.com>

References

Stanley TD, Doucouliagos H (2014). "Meta-regression approximations to reduce publication selection bias." *Research Synthesis Methods*, **5**(1), 60–78. doi:10.1002/jrsm.1095.

Examples

```
# Generate some example data
data <- data.frame(
  yi = c(0.2, 0.3, 0.1, 0.4, 0.25),
  sei = c(0.1, 0.15, 0.08, 0.12, 0.09)
)

# Apply PET method
result <- run_method("PET", data)
print(result)
```

`method.PETPEESE` *PET-PEESE (Precision-Effect Test and Precision-Effect Estimate with Standard Errors) Method*

Description

Implements the Precision-Effect Test and Precision-Effect Estimate with Standard Errors (PET-PEESE) regresses effect sizes against standard errors^2 to correct for publication bias. The intercept represents the bias-corrected effect size estimate. See Stanley and Doucouliagos (2014) for details.

Usage

```
## S3 method for class 'PETPEESE'
method(method_name, data, settings)
```

Arguments

`method_name`	Method name (automatically passed)
`data`	Data frame with yi (effect sizes) and sei (standard errors)
`settings`	List of method settings (see Details)

Details

The following settings are implemented

`"default"` (`conditional_alpha = 0.10`) determines whether to use PET (PET's effect is not significant at alpha = 0.10 or PEESE estimate (PET's effect is significant at alpha = 0.10)

Value

Data frame with PET-PEESE results

Author(s)

František Bartoš <f.bartos96@gmail.com>

References

Stanley TD, Doucouliagos H (2014). "Meta-regression approximations to reduce publication selection bias." *Research Synthesis Methods*, **5**(1), 60–78. doi:10.1002/jrsm.1095.

Examples

```
# Generate some example data
data <- data.frame(
  yi = c(0.2, 0.3, 0.1, 0.4, 0.25),
  sei = c(0.1, 0.15, 0.08, 0.12, 0.09)
)

# Apply PETPEESE method
result <- run_method("PETPEESE", data)
print(result)
```

---

method.puniform *puniform (P-Uniform) Method*

Description

Implements the p-uniform method for publication bias detection and correction. P-uniform uses the distribution of p-values from significant studies to test for publication bias and estimate the effect size corrected for publication bias. The method assumes that p-values follow a uniform distribution under the null hypothesis of no effect, and uses this to detect and correct for bias. See van Assen et al. (2015) and van Aert and van Assen (2025) for details.

Usage

```
## S3 method for class 'puniform'
method(method_name, data, settings)
```

Arguments

method_name	Method name (automatically passed)
data	Data frame with yi (effect sizes) and sei (standard errors)
settings	List of method settings (see Details)

Details

The following settings are implemented

`"default"` Default p-uniform analysis settings.

`"star"` P-uniform star version of the method.

Value

Data frame with P-Uniform results

Author(s)

František Bartoš <f.bartos96@gmail.com>

References

van Aert RCM, van Assen MALM (2025). "Correcting for publication bias in a meta-analysis with the p-uniform* method." *Psychonomic Bulletin & Review*. https://osf.io/preprints/metaarxiv/zqjr9/.

van Assen MALM, van Aert RCM, Wicherts JM (2015). "Meta-analysis using effect size distributions of only statistically significant studies." *Psychological Methods*, **20**(3), 293–309. doi:10.1037/met0000025.

Examples

```
# Generate some example data
data <- data.frame(
  yi = c(0.2, 0.3, 0.1, 0.4, 0.25),
  sei = c(0.1, 0.15, 0.08, 0.12, 0.09)
)

# Apply puniform method
result <- run_method("puniform", data)
print(result)
```

`method.RMA` *Random Effects Meta-Analysis Method*

Description

Implements the publication bias-unadjusted random-effects meta-analysis.

Usage

```
## S3 method for class 'RMA'
method(method_name, data, settings)
```

Arguments

method_name	Method name (automatically passed)
data	Data frame with yi (effect sizes) and sei (standard errors)
settings	List of method settings (see Details)

Details

The following settings are implemented

"default" Restricted Maximum Likelihood estimator (method = "REML") with Knapp-Hartung adjustment (test = "knha") for a simple random effects meta-analysis and Restricted Maximum Likelihood estimator (method = "REML") with t-distribution adjustment (test = "t") and cluster robust standard errors with small-sample adjustment (if converged, otherwise no small-sample adjustment or no cluster robust standard errors) for a multilevel random effects meta-analysis if study_ids is specified in the data

Value

Data frame with RMA results

Author(s)

František Bartoš <f.bartos96@gmail.com>

References

There are no references for Rd macro \insertAllCites on this help page.

Examples

```
# Generate some example data
data <- data.frame(
  yi = c(0.2, 0.3, 0.1, 0.4, 0.25),
  sei = c(0.1, 0.15, 0.08, 0.12, 0.09)
)

# Apply RMA method
result <- run_method("RMA", data)
print(result)
```

`method.RoBMA` *Robust Bayesian Meta-Analysis (RoBMA) Method*

Description

Implements the robust Bayesian meta-analysis (RoBMA) method that uses Bayesian model-averaging to combine results across several complementary publication bias adjustment methods. See Maier et al. (2023) and Bartoš et al. (2023) for details. If `"study_id"` column is included in the data input, the method uses multilevel parameterization as described in Bartoš et al. (2025).

Note that the prior settings is dispatched based on `"es_type"` column attached to the dataset. The resulting estimates are then summarized on the same scale as was the dataset input (for `"r"`, heterogeneity is summarized on Fisher's z).

Important: This method requires JAGS (Just Another Gibbs Sampler) to be installed on your system. Please download and install JAGS from https://mcmc-jags.sourceforge.io/ before using this method.

Usage

```
## S3 method for class 'RoBMA'
method(method_name, data, settings)
```

Arguments

`method_name`	Method name (automatically passed)
`data`	Data frame with yi (effect sizes), sei (standard errors), es_type (either `"SMD"` for Cohen's d / Hedge's g, `"logOR"` for log odds ratio, `"z"` for Fisher's z, or `"r"` for correlations. Defaults to `"none"` which re-scales the default priors to unit-information width based on total sample size supplied `"ni"`.)
`settings`	List of method settings (see Details.)

Details

The following settings are implemented

`"default"` RoBMA-PSMA with publication bias adjustment as described in Bartoš et al. (2023). (the MCMC settings was reduced to speed-up the simulations) with the three-level specification whenever `"study_ids"` are supplied with the data

`"PSMA"` RoBMA-PSMA with publication bias adjustment as described in Bartoš et al. (2023). (the MCMC settings was reduced to speed-up the simulations) with the three-level specification whenever `"study_ids"` are supplied with the data

Value

Data frame with RoBMA results

Author(s)

František Bartoš <f.bartos96@gmail.com>

References

Bartoš F, Maier M, Wagenmakers E (2025). "Robust Bayesian multilevel meta-analysis: Adjusting for publication bias in the presence of dependent effect sizes." *ArXiV Preprint*. doi:10.31234/osf.io/9tgp2_v1.

Bartoš F, Maier M, Wagenmakers E, Doucouliagos H, Stanley TD (2023). "Robust Bayesian meta-analysis: Model-averaging across complementary publication bias adjustment methods." *Research Synthesis Methods*, **14**(1), 99–116. doi:10.1002/jrsm.1594.

Maier M, Bartoš F, Wagenmakers E (2023). "Robust Bayesian meta-analysis: Addressing publication bias with model-averaging." *Psychological Methods*, **28**(1), 107-122. doi:10.1037/met0000405.

Examples

```
# Generate some example data
data <- data.frame(
  yi      = c(0.2, 0.3, 0.1, 0.4, 0.25),
  sei     = c(0.1, 0.15, 0.08, 0.12, 0.09),
  es_type = "SMD"
)

# Apply RoBMA method (requires RoBMA 3.6.1 version of the package)
#result <- run_method("RoBMA", data)
#print(result)
```

`method.RTMA` *Right-Truncated Meta-Analysis (RTMA) Method*

Description

Implements right-truncated meta-analysis (RTMA) for correcting the joint effects of p-hacking and publication bias in meta-analysis. See Mathur (2024) for details.

RTMA is estimated via `phacking::phacking_meta()`.

Usage

```
## S3 method for class 'RTMA'
method(method_name, data, settings)
```

Arguments

`method_name`	Method name (automatically passed)
`data`	Data frame with yi (effect sizes) and sei (standard errors)
`settings`	List of method settings (see Details.)

Details

The following settings are implemented

`"default"` RTMA with affirmative results defined by positive direction `favor_positive = TRUE` and statistical significance `alpha_select = 0.05`, posterior interval level `ci_level = 0.95`, Stan control settings `adapt_delta = 0.98`, `max_treedepth = 20`, `parallelize = FALSE`, and convergence thresholds `max_r_hat = 1.01` and `min_ess = 500`,

`"relaxed"` RTMA with affirmative results defined by positive direction `favor_positive = TRUE` and statistical significance `alpha_select = 0.05`, posterior interval level `ci_level = 0.95`, relaxed Stan control settings `adapt_delta = 0.95`, `max_treedepth = 15`, `parallelize = FALSE`, and relaxed convergence thresholds `max_r_hat = 1.05` and `min_ess = 300`

A fit is considered converged if the R-hat of the effect size is below `max_r_hat` and its effective sample size is above `min_ess`. Both thresholds must be specified with custom settings.

Value

Data frame with RTMA results

Author(s)

Frantisek Bartos <f.bartos96@gmail.com>

References

Mathur MB (2024). "P-hacking in meta-analyses: A formalization and new meta-analytic methods." *Research Synthesis Methods*, **15**(3), 483–499. doi:10.1002/jrsm.1701.

Examples

```
# Generate some example data with at least one nonaffirmative study
data <- data.frame(
  yi = c(0.20, 0.30, 0.10, -0.05, 0.12),
  sei = c(0.10, 0.15, 0.08, 0.12, 0.09)
)

# Apply RTMA method
result <- run_method("RTMA", data)
print(result)
```

`method.SM` *SM (Selection Models) Method*

Description

Implements selection models for publication bias correction in meta-analysis. The method first fits a random effects meta-analysis model, then applies selection modeling to adjust for publication bias using the metafor package. Selection models account for the probability that studies are published based on their p-values or effect sizes. See Vevea and Hedges (1995) for details.

Usage

```
## S3 method for class 'SM'
method(method_name, data, settings)
```

Arguments

`method_name`	Method name (automatically passed)
`data`	Data frame with yi (effect sizes) and sei (standard errors)
`settings`	List of method settings (see Details)

Details

The following settings are implemented

`"default"` **or** `"3PSM"` 3-parameter step function selection model with Maximum Likelihood estimator (`method = "ML"`) and one step at one-sided p = 0.025 (i.e., selection for significance))

`"4PSM"` 4-parameter step function selection model with Maximum Likelihood estimator (`method = "ML"`) and two steps at one-sided p = 0.025 and p = 0.50 (i.e., selection for significance and direction of the effect)

Value

Data frame with SM results

Author(s)

František Bartoš <f.bartos96@gmail.com>

References

Vevea JL, Hedges LV (1995). "A general linear model for estimating effect size in the presence of publication bias." *Psychometrika*, **60**(3), 419–435. doi:10.1007/BF02294384.

Examples

```
# Generate some example data
data <- data.frame(
  yi = c(0.2, 0.3, 0.1, 0.4, 0.25),
  sei = c(0.1, 0.15, 0.08, 0.12, 0.09)
)

# Apply SM method
result <- run_method("SM", data, "3PSM")
print(result)
```

---

method.trimfill *Trim-and-Fill Meta-Analysis Method*

Description

Implements the trim-and-fill method for adjusting publication bias in meta-analysis using the metafor package.

Usage

```
## S3 method for class 'trimfill'
method(method_name, data, settings)
```

Arguments

method_name	Method name (automatically passed)
data	Data frame with yi (effect sizes) and sei (standard errors)
settings	List of method settings (see Details.)

Details

The following settings are implemented

"default" Random effects model fitted with Restricted Maximum Likelihood estimator (method = "REML") with Knapp-Hartung adjustment (test = "knha"), followed by trim-and-fill using left-side trimming (side = "left") and L0 estimator (estimator = "L0").

Value

Data frame with trim-and-fill results

Author(s)

František Bartoš <f.bartos96@gmail.com>

References

There are no references for Rd macro \insertAllCites on this help page.

Examples

```
# Generate some example data
data <- data.frame(
  yi = c(0.2, 0.3, 0.1, 0.4, 0.25),
  sei = c(0.1, 0.15, 0.08, 0.12, 0.09)
)

# Apply trimfill method
result <- run_method("trimfill", data)
print(result)
```

---

method.WAAPWLS *WAAPWLS (Weighted Average of Adequately Powered Studies) Method*

Description

Implements the WAAP-WLS method for meta-analysis, which combines WLS and WAAP approaches. First fits a WLS model to all studies, then identifies high-powered studies based on the criterion that the WLS estimate divided by 2.8 is greater than or equal to the standard error. If at least 2 high-powered studies are found, uses WAAP (weighted average of adequate power studies only), otherwise uses the original WLS estimate. See Stanley et al. (2017) for details.

Usage

```
## S3 method for class 'WAAPWLS'
method(method_name, data, settings = NULL)
```

Arguments

method_name	Method name (automatically passed)
data	Data frame with yi (effect sizes) and sei (standard errors)
settings	List of method settings (no settings version are implemented)

Value

Data frame with WAAPWLS results

Author(s)

František Bartoš <f.bartos96@gmail.com>

References

Stanley TD, Doucouliagos H, Ioannidis JP (2017). "Finding the power to reduce publication bias." *Statistics in Medicine*, **36**(10), 1580-1598. doi:10.1002/sim.7228.

Examples

```
# Generate some example data
data <- data.frame(
  yi = c(0.2, 0.3, 0.1, 0.4, 0.25),
  sei = c(0.1, 0.15, 0.08, 0.12, 0.09)
)

# Apply WAAPWLS method
result <- run_method("WAAPWLS", data)
print(result)
```

`method.WILS` *Weighted and Iterated Least Squares (WILS) Method*

Description

Implements the weighted and iterated least squares (WILS) method for publication bias correction in meta-analysis. The method is based on the idea of using excess statistical significance (ESS) to identify how many underpowered studies should be removed to reduce publication selection bias. See Stanley and Doucouliagos (2024) for details.

Usage

```
## S3 method for class 'WILS'
method(method_name, data, settings = NULL)
```

Arguments

`method_name`	Method name (automatically passed)
`data`	Data frame with yi (effect sizes) and sei (standard errors)
`settings`	List of method settings (see Details)

Details

The WILS method has two implementation versions based on Stanley & Doucouliagos (2024). The following settings are implemented

`"default"` The simulation version (default) uses residuals from the t ~ Precision regression for the first iteration, then switches to individual excess statistical significance (ESS) for subsequent iterations.

`"example"` The example version consistently uses residuals from the t ~ Precision regression to identify studies to remove across all iterations.

Value

Data frame with WILS results

Author(s)

František Bartoš <f.bartos96@gmail.com>

References

Stanley TD, Doucouliagos H (2024). "Harnessing the power of excess statistical significance: Weighted and iterative least squares." *Psychological Methods*, **29**(2), 407–420. doi:10.1037/met0000502.

Examples

```
# Generate some example data
data <- data.frame(
  yi = c(0.2, 0.3, 0.1, 0.4, 0.25),
  sei = c(0.1, 0.15, 0.08, 0.12, 0.09)
)

# Apply WILS method
result <- run_method("WILS", data)
print(result)
```

`method.WLS` *WLS (Weighted Least Squares) Method*

Description

Implements the Weighted Least Squares method for meta-analysis. WLS fits a weighted regression model with effect sizes as the outcome and weights based on the inverse of the squared standard errors. The intercept represents the weighted average effect size estimate.

Usage

```
## S3 method for class 'WLS'
method(method_name, data, settings = NULL)
```

Arguments

`method_name`	Method name (automatically passed)
`data`	Data frame with yi (effect sizes) and sei (standard errors)
`settings`	List of method settings (no settings version are implemented)

Value

Data frame with WLS results

Author(s)

František Bartoš <f.bartos96@gmail.com>

References

There are no references for Rd macro \insertAllCites on this help page.

Examples

```
# Generate some example data
data <- data.frame(
  yi = c(0.2, 0.3, 0.1, 0.4, 0.25),
  sei = c(0.1, 0.15, 0.08, 0.12, 0.09)
)

# Apply WLS method
result <- run_method("WLS", data)
print(result)
```

---

method_extra_columns *Method Extra Columns*

Description

Retrieves the character vector of custom columns for a given method. These are method-specific columns beyond the standard columns (method, estimate, standard_error, ci_lower, ci_upper, p_value, BF, convergence, note) that each method returns.

When implementing new methods, consider using standardized column names for consistency:

Heterogeneity tau_estimate, tau_ci_lower, tau_ci_upper, tau_p_value, tau_BF

Publication Bias bias_coefficient, bias_coefficient_se, bias_p_value, bias_BF

Usage

```
get_method_extra_columns(method_name)

method_extra_columns(method_name)

## Default S3 method:
method_extra_columns(method_name)
```

Arguments

method_name	Character string of the method name

Value

Character vector of extra column names, or empty character vector if no extra columns are defined for the method

Examples

```
# Get extra columns for PET method
get_method_extra_columns("PET")

# Get extra columns for RMA method
get_method_extra_columns("RMA")
```

`method_settings` *Return Pre-specified Method Settings*

Description

This function returns the list of pre-specified settings for a given Method

Usage

```
method_settings(method_name)

get_method_setting(method_name, version_id)
```

Arguments

`method_name`	Character string specifying the method type
`version_id`	which method version should be used.

Value

A list containing the pre-specified settings. For most methods, the list contains extension of the function call, however, a more elaborate list of settings that is dispatched within the method call is possible.

Examples

```
method_settings("RMA")
get_method_setting("RMA", version_id = "default")
```

prune_benchmark_archives

Remove Cached Benchmark Archives

Description

Remove ZIP copies whose selected catalog members all have verified extracted copies. Logical member files and historical catalogs are retained.

Usage

```
prune_benchmark_archives(release = NULL, dgm_name = NULL, dry_run = TRUE)
```

Arguments

release	Benchmark release identifier, path to a catalog JSON file, or a catalog list. NULL uses the benchmark_release package option.
dgm_name	DGM name (optional when listing resources).
dry_run	Report removable archives without deleting them.

Value

A data frame with archive IDs, paths, byte sizes and removal status.

PublicationBiasBenchmark_options

Options for the PublicationBiasBenchmark package

Description

A placeholder object and functions for the PublicationBiasBenchmark package.

Usage

```
PublicationBiasBenchmark.options(...)

PublicationBiasBenchmark.get_option(name)
```

Arguments

...	named option(s) to change - for a list of available options, see details below.
name	the name of the option to get the current value of - for a list of available options, see details below.

Details

`"resources_directory"` Location where the benchmark data/results/measures are stored

`"prompt_for_download"` Whether each file download should ask for explicit approval

`"benchmark_release"` Release identifier or catalog. NULL selects the release pinned by this package version.

Value

The current value of all available PublicationBiasBenchmark options (after applying any changes specified) is returned invisibly as a named list.

`retrieve_dgm_dataset` *Retrieve Archived DGM Datasets*

Description

Read locally cached dataset shards using frozen condition definitions.

Usage

```
retrieve_dgm_dataset(
  dgm_name,
  condition_id,
  repetition_id = NULL,
  release = NULL,
  source = c("release", "local")
)
```

Arguments

`dgm_name`	DGM name.
`condition_id`	Optional condition ID(s) for datasets.
`repetition_id`	Repetition ID(s); NULL selects all repetitions.
`release`	Release identifier or catalog; NULL uses the package default.
`source`	"release" reads verified catalog-selected files; "local" reads unpublished computation files in resources_directory/DGM.

Value

A data frame.

`retrieve_dgm_measures` *Retrieve Archived Performance Measures*

Description

Read measures stored separately for each DGM, method and setting. Pairwise comparisons are only read when measure = "pairwise" is requested.

Usage

```
retrieve_dgm_measures(
  dgm_name,
  measure = NULL,
  method = NULL,
  method_setting = NULL,
  condition_id = NULL,
  replacement = FALSE,
  release = NULL,
  source = c("release", "local")
)
```

Arguments

`dgm_name`	DGM name.
`measure`	Optional measure name(s); pairwise comparisons require "pairwise".
`method`	Optional method name(s); NULL selects all available methods.
`method_setting`	Optional method setting(s).
`condition_id`	Optional condition ID(s) for datasets.
`replacement`	Whether replacement measures are selected; NULL downloads both variants.
`release`	Release identifier or catalog; NULL uses the package default.
`source`	"release" reads verified catalog-selected files; "local" reads unpublished computation files in resources_directory/DGM.

Value

A data frame of selected measures and Monte Carlo standard errors.

retrieve_dgm_results *Retrieve Archived Method Results*

Description

Read and combine verified shards. Missing shards and overlapping method/setting/condition/repetition keys are errors. Use source = "local" for unpublished distributed computation outputs.

Usage

```
retrieve_dgm_results(
  dgm_name,
  method = NULL,
  method_setting = NULL,
  condition_id = NULL,
  repetition_id = NULL,
  release = NULL,
  source = c("release", "local")
)
```

Arguments

dgm_name	DGM name.
method	Optional method name(s); NULL selects all available methods.
method_setting	Optional method setting(s).
condition_id	Optional condition ID(s) for datasets.
repetition_id	Repetition ID(s); NULL selects all repetitions.
release	Release identifier or catalog; NULL uses the package default.
source	"release" reads verified catalog-selected files; "local" reads unpublished computation files in resources_directory/DGM.

Value

A data frame with the existing method-specific result columns.

run_method *Generic method function for publication bias correction*

Description

This function provides a unified interface to various publication bias correction methods. The specific method is determined by the first argument. See `vignette("Adding_New_Methods", package = "PublicationBiasBenchmark")` for details of extending the package with new methods

Usage

```
run_method(
  method_name,
  data,
  settings = NULL,
  silent = FALSE,
  fit_limit = NULL
)
```

Arguments

method_name	Character string specifying the method type
data	Data frame containing yi (effect sizes) and sei (standard errors)
settings	Either a character identifying a method version or list containing method-specific settings. An emty input will result in running the default (first implemented) version of the method.
silent	Logical indicating whether error messages from the method should be suppressed.
fit_limit	Optional numeric giving the maximum time, in minutes, that the method is allowed to run. NULL (the default) or a non-finite value imposes no limit. When the limit is exceeded, the fit is aborted and a failure result with convergence = FALSE and note = "time limit exceeded with <fit_limit> minutes" is returned. See the Time Limits section for the mechanism and its platform differences.

Value

A data frame with standardized method results

Time Limits

Methods that spend their time inside compiled sampling code (RoBMA via JAGS, RTMA and MMPH via Stan) do not return to R's evaluator and therefore cannot be stopped by R's own elapsed-time limit. fit_limit is consequently enforced by evaluating the method in a separate R process (callr::r_session) that is killed once the limit passes, which stops a fit regardless of what it is executing, on every platform.

That process is started on the first limited fit and reused by the following ones, adding roughly 0.05 seconds per fit; it is discarded and replaced whenever a fit is killed or the process dies. A fit that exceeds the limit therefore leaves no work behind, but also keeps nothing from the fits before it.

The worker has its own random number stream, which is seeded from the calling session for every fit. Results of methods that use randomness stay reproducible from the calling session's seed, but differ from those obtained without a limit.

Output Structure

The returned data frame follows a standardized schema that downstream functions rely on. All methods return the following columns:

- method (character): The name of the method used.
- estimate (numeric): The meta-analytic effect size estimate.
- standard_error (numeric): Standard error of the estimate.
- ci_lower (numeric): Lower bound of the 95% confidence interval.
- ci_upper (numeric): Upper bound of the 95% confidence interval.
- p_value (numeric): P-value for the estimate.
- BF (numeric): Bayes Factor for the estimate.
- convergence (logical): Whether the method converged successfully.
- note (character): Additional notes describing convergence issues.

Some methods may include additional method-specific columns beyond these standard columns. Use get_method_extra_columns() to query which additional columns a particular method returns.

Examples

```
# Example usage with RMA method
data <- data.frame(
  yi = c(0.2, 0.3, 0.1, 0.4),
  sei = c(0.1, 0.15, 0.08, 0.12)
)
result <- run_method("RMA", data, "default")
```

---

simulate_dgm *Simulate From Data-Generating Mechanism*

Description

This function provides a unified interface to various data-generating mechanisms for simulation studies. The specific DGM is determined by the first argument. See vignette("Adding_New_DGMs", package = "PublicationBiasBenchmark") for details of extending the package with new DGMs.

Usage

```
simulate_dgm(dgm_name, settings)
```

Arguments

dgm_name	Character string specifying the DGM type
settings	List containing the required parameters for the DGM or numeric condition_id

Value

A data frame containing the generated data with standardized structure

Output Structure

The returned data frame follows a standardized schema that downstream functions rely on. Across the currently implemented DGMs, the following columns are used:

- `yi` (numeric): The effect size estimate.
- `sei` (numeric): Standard error of `yi`.
- `ni` (integer): Total sample size for the estimate (e.g., sum over groups where applicable).
- `es_type` (character): Effect size type, used to disambiguate the scale of `yi`. Currently used values are `"SMD"` (standardized mean difference / Cohen's d), `"logOR"` (log odds ratio), and `"none"` (unspecified generic continuous coefficient).
- `study_id` (integer/character, optional): Identifier of the primary study/cluster when a DGM yields multiple estimates per study (e.g., Alinaghi2018, PRE). If absent, each row is treated as an independent study.

See Also

`validate_dgm_setting()`, `dgm.Stanley2017()`, `dgm.Alinaghi2018()`, `dgm.Bom2019()`, `dgm.Carter2019()`

Examples

```
simulate_dgm("Carter2019", 1)

simulate_dgm("Carter2019", list(mean_effect = 0, effect_heterogeneity = 0,
                      bias = "high", QRP = "high", n_studies = 10))

simulate_dgm("Stanley2017", list(environment = "SMD", mean_effect = 0,
                       effect_heterogeneity = 0, bias = 0, n_studies = 5,
                       sample_sizes = c(32,64,125,250,500)))
```

---

validate_dgm_setting *Validate DGM Settings*

Description

This function validates the settings provided for a given Data Generating Mechanism (DGM).

Usage

```
validate_dgm_setting(dgm_name, settings)
```

Arguments

`dgm_name`	Character string specifying the DGM type
`settings`	List containing the required parameters for the DGM or numeric condition_id

Value

Error or TRUE depending whether the settings are valid for the specified DGM.

Examples

```
validate_dgm_setting("Carter2019", list(mean_effect = 0,
                        effect_heterogeneity = 0, bias = "high",
                        QRP = "high", n_studies = 10))

validate_dgm_setting("Alinaghi2018", list(environment = "FE",
                        mean_effect = 0, bias = "positive"))

validate_dgm_setting("Stanley2017", list(environment = "SMD",
                        mean_effect = 0,
                        effect_heterogeneity = 0, bias = 0, n_studies = 5,
                        sample_sizes = c(32,64,125,250,500)))
```

---

verify_benchmark_resources

Verify Cached Benchmark Resources

Description

Verify Cached Benchmark Resources

Usage

```
verify_benchmark_resources(
  release = NULL,
  dgm_name = NULL,
  kind = NULL,
  method = NULL,
  method_setting = NULL
)
```

Arguments

release	Benchmark release identifier, path to a catalog JSON file, or a catalog list. NULL uses the benchmark_release package option.
dgm_name	DGM name (optional when listing resources).
kind	Optional resource kind: data, results, measures, pairwise, metadata, or archive.
method	Optional method name(s).
method_setting	Optional method setting(s).

Value

A data frame giving the verification status of selected resources.

Index

benchmark_catalog, 3
benchmark_conditions (benchmark_catalog), 3
bias (measures), 26
bias_mcse (measures), 26

callr::r_session, *61*
compare_measures, 4
compare_single_measure, 6
compute_measures, 8
compute_single_measure, 11
coverage (measures), 26
coverage_mcse (measures), 26
create_empty_result, 13

dgm, 14
dgm(), *16*, *18*, *20*, *21*, *23*
dgm.Alinaghi2018, 15
dgm.Alinaghi2018(), *63*
dgm.Bom2019, 17
dgm.Bom2019(), *63*
dgm.Carter2019, 18
dgm.Carter2019(), *63*
dgm.default, 20
dgm.no_bias, 20
dgm.Stanley2017, 21
dgm.Stanley2017(), *63*
dgm_conditions, 23
download_dgm, 24
download_dgm_datasets (download_dgm), 24
download_dgm_measures (download_dgm), 24
download_dgm_metadata (download_dgm), 24
download_dgm_results (download_dgm), 24

empirical_se (measures), 26
empirical_se_mcse (measures), 26
empirical_variance (measures), 26
empirical_variance_mcse (measures), 26

get_dgm_condition (dgm_conditions), 23
get_method_extra_columns (method_extra_columns), 55
get_method_extra_columns(), *30*, *62*
get_method_setting (method_settings), 56

interval_score (measures), 26
interval_score_mcse (measures), 26

list_benchmark_releases (benchmark_catalog), 3
list_benchmark_resources (benchmark_catalog), 3

mean_ci_width (measures), 26
mean_ci_width_mcse (measures), 26
mean_generic_statistic (measures), 26
mean_generic_statistic_mcse (measures), 26
measure, 25
measure_mcse, 29
measures, 26
method, 29
method.AK, 30
method.default, 32
method.EK, 32
method.FMA, 33
method.MAIVE, 34
method.MAN, 37
method.mean, 38
method.MMPH, 39
method.pcurve, 40
method.PEESE, 41
method.PET, 42
method.PETPEESE, 43
method.puniform, 44
method.RMA, 45
method.RoBMA, 47
method.RTMA, 48
method.SM, 50
method.trimfill, 51

method.WAAPWLS, 52
method.WILS, 53
method.WLS, 54
method_extra_columns, 55
method_extra_columns(), *36*
method_settings, 56
method_settings(), *36*
mse *(*measures*)*, 26
mse_mcse *(*measures*)*, 26

negative_likelihood_ratio *(*measures*)*, 26
negative_likelihood_ratio_mcse
*(*measures*)*, 26

positive_likelihood_ratio *(*measures*)*, 26
positive_likelihood_ratio_mcse
*(*measures*)*, 26
power *(*measures*)*, 26
power_mcse *(*measures*)*, 26
prune_benchmark_archives, 57
PublicationBiasBenchmark.get_option
*(*PublicationBiasBenchmark_options*)*,
57
PublicationBiasBenchmark.options
*(*PublicationBiasBenchmark_options*)*,
57
PublicationBiasBenchmark_options, 57

relative_bias *(*measures*)*, 26
relative_bias_mcse *(*measures*)*, 26
retrieve_dgm_dataset, 58
retrieve_dgm_measures, 59
retrieve_dgm_results, 60
rmse *(*measures*)*, 26
rmse_mcse *(*measures*)*, 26
run_method, 60
run_method(), *29, 30*, *36*

simulate_dgm, 62
simulate_dgm(), *14, 15*

validate_dgm_setting, 63
validate_dgm_setting(), *16*, *18*, *20, 21*, *23*,
63
verify_benchmark_resources, 64