Package 'iconic'

November 20, 2026

Title Causal Model Selection with Genetic Instruments and Negative Controls

Version 0.99.5

biocViews StatisticalMethod, Genetics, MultipleComparison, Regression, Transcriptomics, RNASeq, Survival, GeneExpression, SNP, GeneticVariability

Description Provides a model selection workflow for causal inference with genetic instruments and negative controls in observational omics data. The package fits eight estimators of the natural direct and indirect effects (NDE/NIE), diagnoses which are valid for the user's data via instrument-strength checks and negative-control validity screens (A1, A2, A2'), stress-tests them against confounding and pleiotropy violations, and recommends the estimator most likely to be unbiased. A hybrid generative texture model (a torch GAN for sample-level structure and a Gaussian copula for the mediator panel) lets the sensitivity analysis mirror the marginal and joint structure of the user's cohort. The modular API comprises iconic_data() to standardize data, iconic_diagnose() to assess estimator eligibility, iconic_estimate() to fit all eligible estimators, iconic_sensitivity() to sweep instrument exogeneity, iconic_recommend() to rank estimators by identification strength and robustness, and iconic_prospect() to plan future studies. infer_confounding() estimates held-fixed confounding parameters from the data. Multi-omic cohorts are supplied as MultiAssayExperiment objects (omics panels as experiments, sample-level exposure, instruments and covariates as colData), and analysis data can be exported back to MultiAssayExperiment. Helper functions construct exposure instruments (polygenic scores from GWAS summary statistics or PGS Catalog scoring files), mediator instruments (GenomicRanges-based cis-eQTL scans and elastic-net predicted-expression composites from RangedSummarizedExperiment or matrix inputs), and negative-control panels (principal components of residualized omics matrices).
Time-to-event outcomes are supported on the Cox log-hazard-ratio and restricted-mean-survival-time scales. Binary outcomes are supported on the log-odds-ratio (logistic two-stage predictor-substitution outcome stage) and risk-difference (linear probability model) scales. The torch package is required for the generative texture model (sensitivity and prospective analysis); all other functionality works without it. The methods are described in Bresnahan et al. (2026) <doi:10.64898/2026.08.26.26361466>.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.1.1

Imports AER, GenomicRanges, grDevices, graphics, IRanges, MASS, methods, MultiAssayExperiment, parallel, S4Vectors, stats, SummarizedExperiment, survival, utils, withr

Suggests testthat (>= 3.0.0), ggplot2, patchwork, scales, knitr, rmarkdown, torch, bigsnpr, bigstatsr, glmnet, irlba, BiocStyle

VignetteBuilder knitr

Config/testthat/edition 3

URL https://seantbresnahan.com/iconic/, https://github.com/sbresnahan/iconic

BugReports https://github.com/sbresnahan/iconic/issues

NeedsCompilation no

git_url https://git.bioconductor.org/packages/iconic

git_branch devel

git_last_commit c7ba0c5

git_last_commit_date 2026-11-17

Repository Bioconductor 3.24

Date/Publication 2026-11-19

Author Sean T. Bresnahan [aut, cre] (ORCID: <https://orcid.org/0000-0001-6685-1930>), Charis Xiong [aut]

Maintainer Sean T. Bresnahan <seantbresnahan3@gmail.com>

Contents

iconic-package 6
.analyze_feature 7
.analyze_mediation_feature 7
.apply_composite_pvalues 8
.auto_eligible_methods 9
.auto_train_gan 9
.block_diag_cor 10
.check_instrument_strength 10
.composite_robustness 11
.estimate_mediation_feature 11
.estimate_total_feature 13
.fit_parametric_marginal 14
.generate_nc_noise 14
.generate_outcome_noise 15
.iteration_mediator_texture_features 15
.linpred_to_binary 16
.linpred_to_surv 16

.make_pd_cor . . . 17
.marginal_cdf . . . 17
.marginal_quantile . . . 18
.match_cor . . . 18
.np_cdf . . . 19
.recommend_by_scenario . . . 19
.residual_correlation . . . 20
.resolve_gan . . . 20
.subset_data_for_inference . . . 21
analyze_mediation_robust . . . 22
analyze_methods_parallel . . . 23
analyze_methods_robust . . . 24
apply_fusion_weights . . . 25
as_iconic_data . . . 26
as_MultiAssayExperiment . . . 29
beta_to_m . . . 30
bootstrap_mediation_se . . . 31
build_mediator_instruments . . . 32
build_prs_ldpred2 . . . 34
build_w_pcs . . . 36
call_cis_eqtls . . . 36
check_instrument_strength . . . 38
check_torch_setup . . . 39
composite_p_value . . . 39
compute_iter_bias . . . 40
create_discriminator . . . 41
create_generator . . . 41
delta_se_product . . . 42
fit_coca . . . 42
fit_coca_bin . . . 43
fit_coca_mediation . . . 44
fit_coca_mediation_bin . . . 45
fit_coca_mediation_surv . . . 45
fit_coca_surv . . . 46
fit_direct . . . 47
fit_direct_bin . . . 48
fit_direct_mediation . . . 49
fit_direct_mediation_bin . . . 49
fit_direct_mediation_surv . . . 50
fit_direct_surv . . . 51
fit_iv2sls . . . 52
fit_iv2sls_bin . . . 53
fit_iv2sls_mediation . . . 54
fit_iv2sls_mediation2 . . . 55
fit_iv2sls_mediation2_bin . . . 57
fit_iv2sls_mediation2_surv . . . 59
fit_iv2sls_mediation_bin . . . 61
fit_iv2sls_mediation_surv . . . 62
fit_iv2sls_surv . . . 63
fit_pgc . . . 64
fit_pgc_bin . . . 65
fit_pgc_mediation . . . 66

fit_pgc_mediation2 . . . 67
fit_pgc_mediation2_bin . . . 68
fit_pgc_mediation2_surv . . . 70
fit_pgc_mediation_bin . . . 71
fit_pgc_mediation_surv . . . 72
fit_pgc_scalar . . . 73
fit_pgc_scalar_mediation . . . 74
fit_pgc_surv . . . 75
fit_unadj_bin . . . 76
fit_unadj_mediation . . . 76
fit_unadj_mediation_bin . . . 77
fit_unadj_mediation_surv . . . 78
fit_unadj_surv . . . 79
gan_mediation_sensitivity . . . 80
gan_pleiotropy_sensitivity . . . 82
gan_sensitivity . . . 83
generate_toy_data . . . 85
iconic_data . . . 88
iconic_diagnose . . . 90
iconic_estimate . . . 92
iconic_method_colors . . . 94
iconic_method_order . . . 95
iconic_prospect . . . 95
iconic_recommend . . . 98
iconic_sensitivity . . . 100
infer_confounding . . . 104
list_nc_models . . . 105
load_real_input_data . . . 106
nc_completeness_capture . . . 107
nc_completeness_check . . . 108
nc_cpg . . . 110
nc_independence_check . . . 111
nc_independence_check_gm . . . 111
nc_proxy . . . 112
nc_support_check . . . 113
nc_validity_check . . . 114
nc_validity_screen . . . 115
plot_bias . . . 117
plot_bias_boxplot . . . 117
plot_bias_distribution . . . 118
plot_degradation_surface . . . 119
plot_estimated_vs_true . . . 120
plot_estimate_distribution . . . 120
plot_estimator_benchmark . . . 121
plot_feature_correlation_sweep . . . 122
plot_gan_diagnostics . . . 123
plot_instrument_strength_sweep . . . 124
plot_model_selection . . . 125
plot_nc_coverage_comparison . . . 126
plot_nc_validity_diagnostics . . . 127
plot_pleiotropy_sweep . . . 128
plot_power . . . 128

plot_prospective_analysis . . . 129
plot_sensitivity_heatmap . . . 130
plot_type1_boxplot . . . 130
plot_type1_error . . . 131
plot_type1_vs_conf . . . 132
print.iconic_confounding . . . 132
print.iconic_data . . . 133
print.iconic_diagnosis . . . 134
print.iconic_feature_texture . . . 134
print.iconic_gan . . . 135
print.iconic_prospect . . . 136
print.iconic_recommendation . . . 136
print.iconic_sensitivity . . . 137
qc_gwas_sumstats . . . 138
recommend_estimator . . . 139
residualize_matrix . . . 140
run_mediation_methods . . . 140
run_mediation_sim . . . 141
run_methods . . . 143
run_null_mediation_sim . . . 143
run_null_sim . . . 145
run_simulation . . . 146
run_single_iteration . . . 147
sample_feature_texture . . . 150
sample_texture . . . 150
scenario_manifest . . . 151
score_pgs_panel . . . 152
simulate_single_genetic_instrument . . . 153
summarise_mediation_results . . . 154
summarise_results . . . 155
summary.iconic_diagnosis . . . 155
summary.iconic_prospect . . . 156
summary.iconic_recommendation . . . 156
summary.iconic_sensitivity . . . 157
sweep_instrument_strength . . . 158
sweep_mediation_null_by_conf . . . 159
sweep_mediation_param . . . 160
sweep_nc_validity . . . 162
sweep_null_by_conf . . . 163
sweep_param . . . 164
train_feature_texture . . . 165
train_gan_on_real_data . . . 166

Index **168**

`iconic-package` *iconic: Causal Model Selection with Genetic Instruments and Negative Controls*

Description

Provides a model selection workflow for causal inference with genetic instruments and negative controls in observational omics data. The package fits eight estimators of the natural direct and indirect effects (NDE/NIE), diagnoses which are valid for the user's data via instrument-strength checks and negative-control validity screens (A1, A2, A2'), stress-tests them against confounding and pleiotropy violations, and recommends the estimator most likely to be unbiased. A hybrid generative texture model (a torch GAN for sample-level structure and a Gaussian copula for the mediator panel) lets the sensitivity analysis mirror the marginal and joint structure of the user's cohort. The modular API comprises iconic_data() to standardize data, iconic_diagnose() to assess estimator eligibility, iconic_estimate() to fit all eligible estimators, iconic_sensitivity() to sweep instrument exogeneity, iconic_recommend() to rank estimators by identification strength and robustness, and iconic_prospect() to plan future studies. infer_confounding() estimates held-fixed confounding parameters from the data. Multi-omic cohorts are supplied as MultiAssayExperiment objects (omics panels as experiments, sample-level exposure, instruments and covariates as colData), and analysis data can be exported back to MultiAssayExperiment. Helper functions construct exposure instruments (polygenic scores from GWAS summary statistics or PGS Catalog scoring files), mediator instruments (GenomicRanges-based cis-eQTL scans and elastic-net predicted-expression composites from RangedSummarizedExperiment or matrix inputs), and negative-control panels (principal components of residualized omics matrices). Time-to-event outcomes are supported on the Cox log-hazard-ratio and restricted-mean-survival-time scales. Binary outcomes are supported on the log-odds-ratio (logistic two-stage predictor-substitution outcome stage) and risk-difference (linear probability model) scales. The torch package is required for the generative texture model (sensitivity and prospective analysis); all other functionality works without it. The methods are described in Bresnahan et al. (2026) doi:10.64898/2026.08.26.26361466.

Author(s)

Maintainer: Sean T. Bresnahan <seantbresnahan3@gmail.com> (ORCID)

Authors:

- Sean T. Bresnahan <seantbresnahan3@gmail.com> (ORCID)
- Charis Xiong <charis.xiong@gmail.com>

References

Bresnahan ST, Xiong C, Head ST, Chang Y-H, Bhattacharya A, Huang JY. ICONIC: An R Package for Integrating Instrumental Variable- and Negative-Control-Informed Causal Discovery and Diagnostics in Multiomic Studies. medRxiv (2026). doi:10.64898/2026.08.26.26361466

See Also

Useful links:

- https://seantbresnahan.com/iconic/
- https://github.com/sbresnahan/iconic
- Report bugs at https://github.com/sbresnahan/iconic/issues

.analyze_feature *Analyse a single outcome feature with all estimators (internal)*

Description

Thin wrapper around .estimate_total_feature() that extracts vectors from the `dat` list format used by the simulation pipeline.

Usage

```
.analyze_feature(dat, f, W_avg, W_valid = NULL)
```

Arguments

`dat`	Dataset list (from generate_toy_data / run_single_iteration).
`f`	Feature (column) index into `dat$Y`, `dat$W`, `dat$G`.
`W_avg`	Row means of the full negative-control panel (for COCA).
`W_valid`	Optional: validity-screened W matrix (n x q) for the matrix-bridge PGC. If NULL, uses the full W panel.

Value

Data frame of five rows (one per method) or `NULL` if too few cases.

.analyze_mediation_feature *Apply all mediation estimators to a single feature (internal)*

Description

Thin wrapper around .estimate_mediation_feature() that extracts vectors from the `dat` list format used by the simulation pipeline.

Usage

```
.analyze_mediation_feature(
  dat,
  f,
  W_avg,
  W_valid = NULL,
  se_method = "delta",
  n_boot = 500
)
```

Arguments

dat	Dataset list (from generate_toy_data / run_single_iteration).
f	Feature (column) index.
W_avg	Row means of the full negative-control panel (for COCA).
W_valid	Optional: validity-screened W matrix for matrix-bridge PGC.
se_method	"delta" (default) or "bootstrap".
n_boot	Number of bootstrap resamples when se_method="bootstrap".

Details

When dat$Gm is present (i.e. a mediator instrument was supplied), the 2-stage MR estimator fit_iv2sls_mediation2 is also run, producing a sixth row (method = "IV2SLS2"). When dat$W1 and dat$W2 are present, the two-stage proximal estimator fit_pgc_mediation2 is run as "PGC2" (without Gm) and "PGC2Gm" (with Gm, when Gm is also present).

Value

Data frame of five to eight rows (one per method) or NULL. When dat$M is a matrix (n_mediators > 1), results are returned for each mediator separately with a mediator column.

.apply_composite_pvalues

Apply composite null p-values to a mediation results data frame (internal)

Description

Two-pass post-processing for se_method = "composite".

Usage

```
.apply_composite_pvalues(res)
```

Arguments

res	Data frame from .estimate_mediation_feature() / .estimate_mediation_driver() / run_mediation_methods(). Must contain columns: method, NIE, NIE_p, alpha_M, alpha_se, beta_M, beta_M_se.

Details

The JT-comp test (Huang 2019) requires Var(a) and Var(b), the variances of the standardized z-statistics a = alpha_M / alpha_se and b = beta_M / beta_M_se across the collection of tests sharing the same estimator. This function:

1. Groups rows by method (and mediator when present).
2. For each group, computes the z-statistics from the delta-method estimates already stored in the data frame.
3. Estimates Var(a) / Var(b) as the sample variance of the z-statistics. When fewer than 5 tests are available in a group, falls back to Var = 1 (the point-null value, conservative).

4. Replaces NIE_p with the JT-comp composite p-value.

NDE_p is left unchanged (the NDE is a single coefficient, not a product, so the composite null does not apply). NIE_se is left as the delta-method SE (used for CI construction); only the p-value is replaced.

Value

The same data frame with NIE_p replaced by composite p-values. Also adds columns var_a, var_b for transparency.

.auto_eligible_methods *Determine eligible methods from data alone (internal)*

Description

Lightweight auto-eligibility when no diagnosis is supplied.

Usage

```
.auto_eligible_methods(data)
```

Arguments

data	iconic_data object.

Value

Character vector of method names.

.auto_train_gan *Auto-train a texture model from an iconic_data object (internal)*

Description

Converts the user's iconic_data to the load_real_input_data() format and trains a GAN (or multivariate-normal fallback) on the resulting tidy frame. The trained model supplies realistic covariate and outcome texture to run_single_iteration(), and carries feature-level residual correlation matrices for the Y, M, and W panels so the simulation can inject correlated noise.

Usage

```
.auto_train_gan(data, epochs = 100)
```

Arguments

data	An iconic_data object.
epochs	GAN training epochs. Default 100.

Value

An iconic_gan object.

.block_diag_cor *Build a block-diagonal correlation matrix (internal)*

Description

Creates a p x p block-diagonal correlation matrix modelling co-expression modules: features within the same block have pairwise correlation rho, features in different blocks are uncorrelated. The number of blocks is ceiling(sqrt(p)), with block sizes as equal as possible.

Usage

```
.block_diag_cor(p, rho)
```

Arguments

p	Number of features.
rho	Within-block pairwise correlation (off-diagonal).

Value

A p x p numeric matrix with unit diagonal.

.check_instrument_strength *Check instrument strength via first-stage partial F (internal)*

Description

Computes partial F-statistics for G (in X ~ G + W + covars) and Gm (in M ~ X_hat + Gm + W + covars), matching the first-stage regressions used by fit_iv2sls() and fit_iv2sls_mediation2().

Usage

```
.check_instrument_strength(
  data,
  min_f = 10,
  g_threshold = NULL,
  gm_threshold = NULL,
  n_cores = 1
)
```

Arguments

data	iconic_data object.
min_f	Minimum acceptable partial F. Default 10.

Value

List: F_G, F_Gm, weak_G (logical), weak_Gm (logical).

.composite_robustness *Composite data-driven robustness score (internal)*

Description

Combines per-estimand robustness into a single score and discounts each estimator by the graded strength of the diagnostic assumptions it depends on. Three steps:

1. **Worst-estimand composite**: `pmin(robustness_NDE, robustness_NIE)`. An estimator is only as trustworthy as its weakest estimand, so a strong NDE cannot compensate for a weak NIE (the usual object of a mediation analysis).
2. **Confidence multiplier**: each estimator's composite is multiplied by a factor in $(0, 1]$ read from the graded completeness verdict (`diagnosis$completeness$completeness`). Only estimators whose identification routes through the negative-control bridge (DIRECT, COCA, PGC, PGC2, PGC2Gm) are discounted; instrument-only estimators (IV2SLS, IV2SLS2) and UNADJ keep their full score. When no completeness object exists (no NC panel), bridge-dependent estimators are already ineligible, so the multiplier is never misapplied.
3. **Final score**: `composite * confidence_mult`.

Usage

```
.composite_robustness(ranking, diagnosis, completeness_penalty)
```

Arguments

`ranking`	Data frame with `estimator`, `robustness_NDE`, `robustness_NIE` columns.
`diagnosis`	An `iconic_diagnosis` object (may carry a NULL `$completeness`).
`completeness_penalty`	Named numeric vector mapping completeness verdicts to multipliers in $[0, 1]$.

Value

Data frame with `composite`, `confidence_mult`, and `final_score` aligned to `ranking$estimator`.

.estimate_mediation_feature *Estimate mediation effects for a single feature with specified methods (internal)*

Description

Format-agnostic per-feature mediation estimation: takes explicit vectors and covariates (no dependency on the `dat` list format). Called by both the simulation driver (.analyze_mediation_feature) and the real-data driver (iconic_estimate).

Usage

```
.estimate_mediation_feature(
  X,
  y,
  M_vec,
  g = NULL,
  gm = NULL,
  w = NULL,
  W_mat = NULL,
  W1_mat = NULL,
  W2_mat = NULL,
  W_avg = NULL,
  covars = NULL,
  methods = NULL,
  feature_idx = 1L,
  min_f = 10,
  se_method = c("delta", "bootstrap", "composite"),
  n_boot = 500
)
```

Arguments

X	Numeric exposure vector (length n).
y	Numeric outcome vector (length n).
M_vec	Numeric mediator vector (length n).
g	Numeric instrument vector (length n), or NULL.
gm	Numeric mediator instrument vector (length n), or NULL.
w	Numeric NC vector (length n) or matrix (n x q), or NULL (for DIRECT, IV2SLS, IV2SLS2 — full panel as covariates).
W_mat	Numeric NC matrix (n x q), or NULL (for PGC matrix bridge).
W1_mat	Numeric path-specific NC matrix (n x q) for X->M, or NULL.
W2_mat	Numeric path-specific NC matrix (n x q) for M->Y, or NULL.
W_avg	Numeric vector (length n): row means of NC panel for COCA. If NULL but W_mat present, computed inline.
covars	Optional data frame of covariates (n rows).
methods	Character vector of methods to run. Default: all applicable.
feature_idx	Integer or character: feature identifier. Default 1L.
se_method	"delta" (default), "bootstrap", or "composite". When "bootstrap", NDE_se/NIE_se are replaced by the SD of n_boot nonparametric bootstrap resamples. Point estimates (NDE/NIE) and p-values are unchanged; only the SE column is swapped. When "composite", NIE_p is replaced by the Huang (2019) JT-comp composite null p-value (post-processed across features by the driver).
n_boot	Number of bootstrap resamples when se_method="bootstrap".

Value

Data frame: feature, method, NDE, NDE_se, NDE_p, NIE, NIE_se, NIE_p. Returns NULL if < 20 complete cases. When se_method="composite", also includes alpha_M, alpha_se, beta_M, beta_M_se for downstream composite p-value computation.

`.estimate_total_feature` *Estimate total effect for a single feature with specified methods (internal)*

Description

Format-agnostic per-feature estimation: takes explicit vectors and covariates (no dependency on the `dat` list format). Called by both the simulation driver (.analyze_feature) and the real-data driver (iconic_estimate).

Usage

```
.estimate_total_feature(
  X,
  y,
  g = NULL,
  w = NULL,
  W_mat = NULL,
  W_avg = NULL,
  covars = NULL,
  methods = .methods_all,
  feature_idx = 1L,
  min_f = 10
)
```

Arguments

`X`	Numeric exposure vector (length n).
`y`	Numeric outcome vector (length n).
`g`	Numeric instrument vector (length n), or NULL.
`w`	Numeric NC vector (length n) or matrix (n x q), or NULL. Used for DIRECT, IV2SLS (full panel as covariates), and COCA (via W_avg scalar).
`W_mat`	Numeric NC matrix (n x q), or NULL. Used for PGC (matrix bridge). If NULL, PGC is skipped.
`W_avg`	Numeric vector (length n): row means of the NC panel for COCA. If NULL but W_mat is present, computed inline.
`covars`	Optional data frame of covariates (n rows).
`methods`	Character vector of methods to run. Default: all five. Methods whose required inputs are missing are silently skipped.
`feature_idx`	Integer or character: feature identifier for the output `feature` column. Default 1L.

Value

Data frame: `feature`, `method`, `beta`, `se`, `pvalue`. Returns NULL if fewer than 20 complete cases.

.fit_parametric_marginal *Fit a parametric distribution to a data vector (internal)*

Description

Attempts to fit normal, log-normal, gamma, and beta distributions (where applicable) by maximum likelihood, and returns the best fit by AIC along with the KS test p-value.

Usage

```
.fit_parametric_marginal(x)
```

Arguments

x	Numeric vector (the feature's observed values).

Value

A list with `family`, `params`, `aic`, `ks_p`, or NULL if no parametric fit succeeds.

.generate_nc_noise *Generate correlated noise for negative-control models (internal)*

Description

When `noise_cor` is a valid `p x p` correlation matrix, draws `n` samples from `MVN(0, noise_cor)` and scales by `sd0`. Returns an `n x (2*p)` matrix: columns `1:p` are the "uncaptured" noise component and columns `(p+1):(2*p)` are the idiosyncratic noise component (both with the same correlation structure). When `noise_cor` is NULL, uses independent `rnorm` (backward compatible).

Usage

```
.generate_nc_noise(n, p, noise_cor, sd0)
```

Arguments

n	Number of samples.
p	Number of features.
noise_cor	A `p x p` correlation matrix, or NULL.
sd0	Idiosyncratic noise SD.

Value

An `n x (2*p)` numeric matrix.

.generate_outcome_noise

Generate outcome noise, optionally correlated across features (internal)

Description

When `noise_cor` is a valid `p x p` correlation matrix, draws `n` samples from `MVN(0, 0.2^2 * noise_cor)`. Otherwise uses independent `rnorm(n, 0, 0.2)` (backward compatible).

Usage

```
.generate_outcome_noise(n, p, noise_cor)
```

Arguments

`n`	Number of samples.
`p`	Number of features.
`noise_cor`	A `p x p` correlation matrix, or NULL.

Value

An `n x p` numeric matrix of outcome noise.

.iteration_mediator_texture_features

Exogenous, mediator-only texture component for one iteration (internal)

Description

Returns a p x n matrix of realistic mediator feature vectors drawn from the feature-level copula texture model. Each row is a feature (transcript), each column a sample. The draws preserve the marginal distributions and cross-feature correlation structure of the user's mediator panel. The texture is drawn independently of U and never fed into X, so it cannot open a confounding backdoor.

Usage

```
.iteration_mediator_texture_features(trained_gan, n, p)
```

Arguments

`trained_gan`	An iconic_gan object (may be NULL).
`n`	Number of synthetic samples.
`p`	Number of features (target n_features for this iteration).

Details

When no feature_texture is available (trained_gan is NULL or lacks $feature_texture), returns NULL and the caller falls back to scalar noise.

Value

A p x n matrix, or NULL.

.linpred_to_binary *Convert a linear predictor to a binary outcome*

Description

Given an n-vector (or n x p matrix) of linear predictors `eta` and a target prevalence, returns a list with the 0/1 outcome vector `y_bin`. When `eta` is a matrix, the first column is used (binary outcomes are scalar — `n_features = 1`).

Usage

```
.linpred_to_binary(eta, prev = 0.5)
```

Arguments

`eta`	numeric vector or matrix of linear predictors.
`prev`	target prevalence (marginal event probability); the intercept is solved so that approximately this fraction of subjects are cases. Default 0.5.

Value

list(y_bin, p, intercept, prev)

.linpred_to_surv *Convert a linear predictor to survival data*

Description

Given an n-vector (or n x p matrix) of linear predictors `eta` and a target event fraction, returns a list with `surv_time` and `surv_event` vectors. When `eta` is a matrix, the first column is used (survival outcomes are scalar — n_features = 1).

Usage

```
.linpred_to_surv(eta, h0 = 0.1, event_frac = 0.6, censor_rate = NULL)
```

Arguments

`eta`	numeric vector or matrix of linear predictors.
`h0`	baseline hazard (default 0.1).
`event_frac`	target event fraction; `censor_rate` is tuned so that approximately this fraction of subjects are observed events.
`censor_rate`	optional explicit censoring rate; if NULL it is solved from `event_frac`.

Value

list(surv_time, surv_event, true_h0, true_censor_rate)

.make_pd_cor *Near-PD correction for a correlation matrix (internal)*

Description

Clips eigenvalues to a small positive floor and renormalizes to unit diagonal. Same approach as .residual_correlation() in load_data.R.

Usage

```
.make_pd_cor(cor_mat)
```

Arguments

cor_mat	A symmetric matrix (intended to be a correlation matrix).

Value

A valid correlation matrix (symmetric, positive-definite, unit diagonal).

.marginal_cdf *CDF for a fitted marginal (internal)*

Description

Returns the cumulative distribution function for a parametric or empirical marginal.

Usage

```
.marginal_cdf(marginal, q)
```

Arguments

marginal	A marginal specification list.
q	Numeric vector of quantiles.

Value

Numeric vector of probabilities.

.marginal_quantile *Inverse CDF for a fitted marginal (internal)*

Description

Returns the quantile function for a parametric or empirical marginal.

Usage

```
.marginal_quantile(marginal, p)
```

Arguments

marginal	A marginal specification list from .fit_parametric_marginal or an empirical CDF object.
p	Vector of probabilities in (0, 1).

Value

Numeric vector of quantiles.

.match_cor *Match a correlation matrix to the target feature count (internal)*

Description

If `cor_mat` is a valid `p_target x p_target` matrix, return it as-is. If it's a different size, return NULL (the simulation falls back to independent noise). This guards against dimension mismatches when the simulation uses a different `n_features` than the training data.

Usage

```
.match_cor(cor_mat, p_target)
```

Arguments

cor_mat	A correlation matrix or NULL.
p_target	Target number of features.

Value

A `p_target x p_target` correlation matrix, or NULL.

.np_cdf *Normal product distribution CDF*

Description

Computes F(z) = 2 * $integral_{|z|}^{Inf}$ K0(x)/pi dx, the two-sided tail probability of the standard normal product distribution. Under H0(1), if Z1, Z2 ~ N(0,1) independent, then P(|Z1*Z2| >= |z|) = F(z). K0 is the modified Bessel function (besselK(x, nu=0)).

Usage

```
.np_cdf(z)
```

Arguments

z	Numeric scalar or vector.

Value

Numeric scalar or vector of CDF values in $[0, 1]$.

.recommend_by_scenario *Best estimator under each collection scenario (internal)*

Description

Maps each instrument/NC collection scenario to the estimators it makes available, then picks the most robust eligible one from the `iconic_recommend` ranking (robustness = per-estimand max|bias| over the Phase 3 rho x omega surface). An estimator appears only under scenarios that supply its required data:

- UNADJ: none (always available)
- COCA: W (negative controls only; no instrument)
- DIRECT, IV2SLS, PGC: G + W
- IV2SLS2: G + Gm (optional path-specific W1/W2 augmentation)
- PGC2: G + W1 + W2
- PGC2Gm: G + Gm + W1 + W2

Usage

```
.recommend_by_scenario(rec)
```

Arguments

rec	An `iconic_recommendation` object.

Value

A data frame with columns `scenario`, `estimator`, and `robustness_NDE`.

.residual_correlation *Compute the residual correlation matrix of a features x samples panel (internal)*

Description

Residualizes each feature (row) on the sample-level exposure and covariates, then computes the p x p correlation matrix of the residuals. Returns NULL if the matrix has fewer than 2 features or the correlation is degenerate.

Usage

```
.residual_correlation(
  mat,
  exposure,
  covars,
  residualize_on = c("XC", "XCW"),
  W_mat = NULL
)
```

Arguments

mat	A features x samples numeric matrix.
exposure	A length-n_samples numeric vector (per-sample exposure).
covars	A data frame of encoded covariates (n_samples rows).
residualize_on	Character: "XC" (default, legacy) residualizes on exposure + covariates only; "XCW" additionally residualizes on the negative-control bridge proxy $\hat{W}$, obtained by regressing W_mat on exposure + covariates. This partials out the U-signature that W captures, reducing the double-counting of U's cross-feature signature when the residual matrix is later injected as correlated noise alongside synthetic U loadings. Tradeoff: can over-partial if W is a weak proxy — recommended only when the completeness-capture test (§1) reports "strong".
W_mat	Optional features x samples NC matrix used to construct the bridge proxy $\hat{W}$ when residualize_on = "XCW". Ignored otherwise.

Value

A p x p correlation matrix, or NULL.

.resolve_gan *Resolve the trained GAN for a simulation function (internal)*

Description

Priority: explicit trained_gan argument > data$trained_gan (attached at iconic_data() construction) > auto-train from data.

Usage

```
.resolve_gan(trained_gan, data, epochs = 100)
```

Arguments

trained_gan	Explicit argument (may be NULL).
data	An iconic_data object.
epochs	Epochs for auto-training. Default 100.

Value

A list with gan (the resolved iconic_gan) and source (a string: "user-supplied GAN", "data-attached GAN", or "auto-trained from data").

.subset_data_for_inference

Subset an iconic_data object to a random panel for confounding inference

---

Description

Draws a random subset of mediators and (when more than one) outcome features, capping each at max_infer_tasks, and returns a thinned iconic_data object suitable for iconic_estimate(). Used to make the estimate = NULL auto-run in infer_confounding() fast on large panels: the confounding-strength gaps are averages across the mediator x feature grid, so a random subset is an unbiased Monte Carlo estimate of the full-panel value.

Usage

```
.subset_data_for_inference(data, max_infer_tasks = 50)
```

Arguments

data	An iconic_data object.
max_infer_tasks	Cap on mediators and on features.

Value

A thinned iconic_data object with an inference_subset attribute recording the subset sizes.

`analyze_mediation_robust` *Run all mediation estimators on one synthetic dataset*

Description

Reference-named entry point: applies UNADJ, DIRECT, COCA, IV2SLS, and PGC (matrix bridge) mediation estimators to each tested outcome feature and returns tidy per-feature results with significance flags for both NDE and NIE. When the dataset includes a mediator instrument (`Gm`), the 2-stage MR estimator `fit_iv2sls_mediation2` (method "IV2SLS2") is also run. When the dataset includes path-specific negative controls (`W1`, `W2`;), the two-stage proximal estimator `fit_pgc_mediation2` is run as "PGC2" (without Gm) and "PGC2Gm" (with Gm, when Gm is also present).

Usage

```
analyze_mediation_robust(
  iteration_data,
  test_features = NULL,
  alpha = 0.05,
  n_cores = 1,
  se_method = c("delta", "bootstrap", "composite"),
  n_boot = 500
)
```

Arguments

`iteration_data`	Dataset list from `run_single_iteration()` (or `generate_toy_data()`). When `Gm` is present, the 2-stage MR estimator is included.
`test_features`	Optional integer indices of outcome features to test. Default `NULL` (all features).
`alpha`	Significance threshold for the significance flags. Default 0.05.
`n_cores`	Number of parallel workers. Default 1 (sequential). Uses `parallel::mclapply` on Unix and a PSOCK cluster on Windows.
`se_method`	"delta" (default), "bootstrap", or "composite". See `iconic_estimate()` for details.
`n_boot`	Number of bootstrap resamples when `se_method = "bootstrap"`. Default 500.

Details

Negative controls are always summarised over the full W panel for COCA, so restricting `test_features` does not change its control summary. The matrix-bridge PGC uses the full W panel.

A scalar-bridge variant (`fit_pgc_scalar_mediation()`) is exported for standalone use but is not included in the default pipeline.

Value

Data frame: `feature`, `method`, `NDE`, `NDE_se`, `NDE_p`, `NIE`, `NIE_se`, `NIE_p`, `NDE_significant`, `NIE_significant`.

Examples

```
dat <- generate_toy_data(n = 300, mo_confounding = 0.8, seed = 1)
analyze_mediation_robust(dat)
```

`analyze_methods_parallel`
Parallel version of `analyze_methods_robust()`

Description

Distributes the per-feature analysis across workers via `parallel::mclapply` (falling back to sequential on Windows / one core).

Usage

```
analyze_methods_parallel(
  iteration_data,
  test_features = NULL,
  alpha = 0.05,
  debug = FALSE,
  n_cores = 1
)
```

Arguments

`iteration_data`	Dataset list from `run_single_iteration()` (or `generate_toy_data()`).
`test_features`	Optional integer indices of outcome features to test. Default `NULL` (all features).
`alpha`	Significance threshold for the `significant` flag. Default 0.05.
`debug`	If `TRUE`, message per-feature progress. Default `FALSE`.
`n_cores`	Number of workers. Default 1.

Value

Data frame: `feature`, `method`, `beta`, `se`, `pvalue`, `significant`.

Examples

```
dat <- run_single_iteration(NULL, n_synthetic_samples = 100,
    n_features = 5, n_confounders = 1, seed = 1)
analyze_methods_parallel(dat, n_cores = 1)
```

`analyze_methods_robust` *Run all five estimators (plus UNADJ) on one synthetic dataset*

Description

Reference-named entry point: applies UNADJ, DIRECT, COCA, IV2SLS, and PGC (matrix bridge) to each tested outcome feature and returns tidy per-feature results with a significance flag. Negative controls are summarised over the full W panel for COCA, so restricting `test_features` does not change its control summary. The matrix-bridge PGC uses the full W panel (or a validity-screened subset if provided).

Usage

```
analyze_methods_robust(
  iteration_data,
  test_features = NULL,
  alpha = 0.05,
  debug = FALSE,
  n_cores = 1
)
```

Arguments

`iteration_data`	Dataset list from `run_single_iteration()` (or `generate_toy_data()`).
`test_features`	Optional integer indices of outcome features to test. Default `NULL` (all features).
`alpha`	Significance threshold for the `significant` flag. Default 0.05.
`debug`	If `TRUE`, message per-feature progress. Default `FALSE`.
`n_cores`	Number of parallel workers. Default 1 (sequential). Uses `parallel::mclapply` on Unix and a PSOCK cluster on Windows.

Details

A scalar-bridge variant (`fit_pgc_scalar()`) is exported for standalone use but is not included in the default pipeline.

Value

Data frame: `feature`, `method`, `beta`, `se`, `pvalue`, `significant`.

Examples

```
dat <- run_single_iteration(NULL, n_synthetic_samples = 100,
    n_features = 5, n_confounders = 1, seed = 1)
analyze_methods_robust(dat)
```

`apply_fusion_weights` *Apply FUSION/TWAS eQTL weights to a dosage matrix*

Description

Predicts genetically regulated expression by applying published FUSION per-gene weight files (`.wgt.RDat`) to a genotype dosage matrix. For each gene, the model with the best cross-validation performance (`cv.performance["rsq", ]`) is used. The resulting predicted-expression matrix is germline-determined and — after PCA via `build_w_pcs()` — serves as a negative-control panel that captures confounding structure without direct exposure mediation.

Usage

```
apply_fusion_weights(
  genotypes,
  pos,
  weights_dir = NULL,
  weights = NULL,
  min_snps = 2,
  n_cores = 1
)
```

Arguments

`genotypes`	Numeric dosage matrix, variants x samples, with rownames matching the SNP IDs used in the weight files (usually rsIDs).
`pos`	A data.frame with (at least) columns `ID` (gene ID) and `WGT` (weight-file name), as read from a FUSION `.pos` file; or a path to such a file.
`weights_dir`	Directory containing the `.wgt.RDat` files named in `pos$WGT`. Ignored when `weights` is supplied.
`weights`	Optional named list of pre-loaded weight objects, each a list with elements `wgt.matrix`, `snps`, and `cv.performance` (as stored in FUSION `.wgt.RDat` files). Names must match `pos$ID`. Supplying this avoids repeated file I/O and makes the function usable without the original files.
`min_snps`	Integer: minimum number of matched SNPs required to predict a gene. Default 2.
`n_cores`	Integer: parallel workers via `parallel::mclapply()`. Default 1.

Value

A list with:

predicted numeric matrix, genes x samples: predicted expression.

info data.frame per predicted gene: `gene`, `n_snps`, `best_method`, `rsq` (cross-validation R2 of the selected model).

Examples

```
# Simulate two FUSION-style weight objects in memory
dos <- matrix(rbinom(10 * 50, 2, 0.3), nrow = 10,
    dimnames = list(paste0("rs", 1:10), paste0("S", 1:50)))
mk_wgt <- function(snps, w) {
    list(wgt.matrix = matrix(w, ncol = 1, dimnames = list(NULL, "enet")),
        snps = data.frame(V2 = snps),
        cv.performance = matrix(0.2, nrow = 1,
            dimnames = list("rsq", "enet")))
}
wlist <- list(GeneA = mk_wgt(paste0("rs", 1:5), rep(0.1, 5)),
    GeneB = mk_wgt(paste0("rs", 6:10), rep(-0.2, 5)))
pos <- data.frame(ID = c("GeneA", "GeneB"), WGT = c("a.wgt.RDat", "b.wgt.RDat"))
out <- apply_fusion_weights(dos, pos = pos, weights = wlist)
dim(out$predicted)
```

---

as_iconic_data *Convert external data containers to iconic_data*

Description

S3 generic bridging other data interfaces to the estimation interface. Methods:

Usage

```
as_iconic_data(input, ...)

## Default S3 method:
as_iconic_data(input, ...)

## S3 method for class 'SummarizedExperiment'
as_iconic_data(
  input,
  assay = 1,
  mediator_assay = NULL,
  exposure,
  instrument = NULL,
  mediator_instrument = NULL,
  negative_controls = NULL,
  covariates = NULL,
  surv_time = NULL,
  surv_event = NULL,
  bin_outcome = NULL,
  outcome_type = c("continuous", "survival", "binary"),
  ...
)

## S3 method for class 'MultiAssayExperiment'
as_iconic_data(
  input,
  exposure,
  outcome = NULL,
```

```
  mediator = NULL,
  instrument = NULL,
  mediator_instrument = NULL,
  negative_controls = NULL,
  W1 = NULL,
  W2 = NULL,
  covariates = NULL,
  surv_time = NULL,
  surv_event = NULL,
  bin_outcome = NULL,
  outcome_type = c("continuous", "survival", "binary"),
  assay = 1L,
  verbose = TRUE,
  ...
)
```

Arguments

`input`	An object to convert: a MultiAssayExperiment, a SummarizedExperiment, a list returned by `load_real_input_data()`, an `iconic_data` object, or an exposure vector (named-argument form).
`...`	Named arguments passed to `iconic_data()` when using the named-argument form, or to the method.
`assay`	SummarizedExperiment method: name or index of the assay of `input` holding the outcome panel (features x samples). Default `1`. Set to `NULL` for survival outcomes (when `surv_time`/`surv_event` are colData columns and no continuous outcome panel is used). MultiAssayExperiment method: name or index of the assay used within each SummarizedExperiment experiment. Default `1`.
`mediator_assay`	Optional name or index of an assay holding the mediator panel (mediators x samples).
`exposure`	Character: name of the `colData` column holding the exposure X (MultiAssayExperiment method: or of a single experiment). For the MultiAssayExperiment method, when `exposure` is missing and `input` was produced by `as_MultiAssayExperiment()`, the roles recorded in `metadata(input)$iconic` are used (round trip).
`instrument`	Optional character: name of the `colData` column holding the exposure instrument G (e.g. a polygenic score).
`mediator_instrument`	Optional character vector of `colData` column names holding the mediator instrument(s) Gm (one column per mediator); for the MultiAssayExperiment method, alternatively the name of an experiment (instruments x samples).
`negative_controls`	Optional character vector of `colData` column names forming the negative-control panel W; for the MultiAssayExperiment method, alternatively the name of an experiment (e.g. principal components from `build_w_pcs()`, features x samples).
`covariates`	Optional character vector of `colData` column names to carry through as covariates.
`surv_time, surv_event`	Optional character: names of `colData` columns holding follow-up time and the 0/1 event indicator; set together with `outcome_type = "survival"`.

bin_outcome	Optional character: name of a colData column holding the 0/1 outcome; set together with outcome_type = "binary".
outcome_type	\"continuous\" (default), \"survival\", or \"binary\".
outcome	For the MultiAssayExperiment method: name of the experiment holding the outcome panel Y (features x samples). Required when outcome_type = "continuous".
mediator	Optional: name of the experiment (or character vector of colData columns) holding the mediator panel M.
W1, W2	Optional: names of experiments (or character vectors of colData columns) holding the path-specific negative-control panels for the X->M (W1) and M->Y (W2) paths.
verbose	Logical: report samples dropped during alignment. Default TRUE.

Details

- default: a list returned by load_real_input_data(), an existing iconic_data object (returned as-is), or the exposure vector X with named arguments matching iconic_data() (Y, M, G, Gm, W, W1, W2, covariates, feature_names, mediator_names), which delegates to iconic_data().
- MultiAssayExperiment (recommended for multi-omic data): omics panels (outcome, mediator, negative controls, mediator instruments) are experiments, sample-level fields (exposure, instrument, covariates, survival / binary outcome) are colData columns, and samples are matched through the sampleMap. See Details and as_MultiAssayExperiment() for the reverse conversion.
- SummarizedExperiment: extracts the outcome panel from an assay and sample-level fields (exposure, instruments, negative controls, covariates) from colData.

MultiAssayExperiment method. Each role (outcome, mediator, negative_controls, W1, W2, mediator_instrument, and optionally exposure / instrument) names either one experiment of input or one or more colData columns; experiment names take precedence. For SummarizedExperiment experiments the assay selected by assay is used; matrix experiments are used directly. Samples are matched through sampleMap(input) to primary IDs (the rownames of colData). Only primaries present in every experiment used, and with non-missing values in the sample-level colData columns used, are kept. Experiments with more than one column per primary (e.g. technical replicates) must be collapsed first, for example with MultiAssayExperiment::mergeReplicates(). The aligned components are passed to iconic_data(), so validation and scaling are identical to calling it directly.

Value

An iconic_data S3 object.

Examples

```
# From a load_real_input_data() result
input <- load_real_input_data(example = TRUE)
data <- as_iconic_data(input)
print(data)

# From named components (delegates to iconic_data())
data <- as_iconic_data(rnorm(100), Y = matrix(rnorm(100 * 10), 10, 100),
    G = rnorm(100), Gm = rnorm(100),
    W = matrix(rnorm(100 * 10), 10, 100))
se <- SummarizedExperiment::SummarizedExperiment(
```

```
    assays = list(expr = matrix(rnorm(20 * 60), 20, 60,
        dimnames = list(paste0("gene", 1:20),
            paste0("S", 1:60)))),
    colData = S4Vectors::DataFrame(
        bmi = rnorm(60), prs = rnorm(60),
        nc1 = rnorm(60), nc2 = rnorm(60), age = rnorm(60))
)
data <- as_iconic_data(se, assay = "expr", exposure = "bmi",
    instrument = "prs",
    negative_controls = c("nc1", "nc2"),
    covariates = "age")
print(data)
# From a MultiAssayExperiment: omics panels are experiments,
# sample-level variables are colData columns
set.seed(1)
ids <- paste0("S", 1:80)
mk <- function(p, pre) matrix(rnorm(p * 80), p, 80,
    dimnames = list(paste0(pre, seq_len(p)), ids))
mae <- MultiAssayExperiment::MultiAssayExperiment(
    experiments = list(
        rna = SummarizedExperiment::SummarizedExperiment(
            assays = list(expr = mk(10, "gene"))),
        methyl = SummarizedExperiment::SummarizedExperiment(
            assays = list(m = mk(1, "cg"))),
        nc = SummarizedExperiment::SummarizedExperiment(
            assays = list(pcs = mk(10, "PC")))),
    colData = S4Vectors::DataFrame(
        bmi = rnorm(80), prs = rnorm(80), cis_score = rnorm(80),
        age = rnorm(80), row.names = ids))
data <- as_iconic_data(mae, exposure = "bmi", outcome = "rna",
    mediator = "methyl", instrument = "prs",
    mediator_instrument = "cis_score", negative_controls = "nc",
    covariates = "age")
print(data)
```

`as_MultiAssayExperiment` *Convert an iconic_data object to a MultiAssayExperiment*

Description

Exports the standardised analysis data held in an `iconic_data` object as a MultiAssayExperiment::MultiAssayExperiment, the Bioconductor container for multi-omic data on shared samples. Use it to store, share, subset, or inspect the exact data an ICONIC analysis was run on with standard Bioconductor tools. The result can be passed straight back to `as_iconic_data()` (the role mapping is recorded in the metadata).

Usage

```
as_MultiAssayExperiment(data)
```

Arguments

`data`	An `iconic_data` object (from `iconic_data()` or `as_iconic_data()`).

Value

A `MultiAssayExperiment` with

experiments `SummarizedExperiments` (assay `value`, features x samples) named `Y` (outcome panel; continuous outcomes only), `M` (mediators), `Gm` (mediator instruments), `W`, `W1`, `W2` (negative-control panels), each present only when supplied.

colData `X` (exposure), `G` (exposure instrument), `surv_time` / `surv_event` (survival), `Y_bin` (binary outcome), and the encoded covariates (prefixed `cov_` if a name collides).

metadata `iconic`: the outcome type, scaling parameters, feature / mediator names, availability flags, the role mapping used by `as_iconic_data()`, and the attached GAN (if any).

Values are those stored in `data`, i.e. centred and scaled when `iconic_data(scale = TRUE)` (the default); original-scale parameters are kept in `metadata(mae)$iconic$scaling`.

Examples

```
d <- iconic_data(X = rnorm(60), Y = matrix(rnorm(60 * 5), 5, 60),
    M = rnorm(60), G = rnorm(60), Gm = rnorm(60),
    W = matrix(rnorm(60 * 5), 5, 60))
mae <- as_MultiAssayExperiment(d)
mae
d2 <- as_iconic_data(mae) # round trip
all.equal(d$Y, d2$Y, check.attributes = FALSE)
```

---

beta_to_m *Convert methylation beta values to M-values*

Description

Applies the logit transform $\log_2(\beta/(1-\beta))$ after clipping betas away from 0 and 1. M-values are approximately homoscedastic and are the recommended scale for linear modeling and PCA of methylation data.

Usage

```
beta_to_m(beta, clip = 1e-04, drop_nonfinite = TRUE)
```

Arguments

`beta`	Numeric matrix of beta values (probes x samples), in [0, 1].
`clip`	Numeric: betas are clipped to [`clip`, `1 - clip`] before the transform. Default 1e-4.
`drop_nonfinite`	Logical: drop probes (rows) with any missing/non-finite values after the transform. Default `TRUE`.

Value

A numeric matrix of M-values, probes x samples.

Examples

```
b <- matrix(runif(200, 0.01, 0.99), nrow = 20,
    dimnames = list(paste0("cg", 1:20), paste0("S", 1:10)))
m <- beta_to_m(b)
range(m)
```

`bootstrap_mediation_se` *Bootstrap SE for a mediation estimator*

Description

Resamples the data `n_boot` times and returns the SD of the bootstrap NDE/NIE distribution. This is an opt-in alternative to the delta-method SE for users who want a non-parametric SE (bootstrapped reasonably"). Slower than the delta method.

Usage

```
bootstrap_mediation_se(estimator_fn, n, n_boot = 500)
```

Arguments

`estimator_fn`	A closure `function(idx)` returning a fit list with `NDE` and `NIE` (and optionally `NDE_se`, `NIE_se`).
`n`	Sample size (length of the resampling index).
`n_boot`	Number of bootstrap resamples. Default 500.

Details

The estimator is supplied as a closure `estimator_fn(idx)` that captures all model-specific data (outcome, exposure, mediator, instruments, negative controls, covariates) and subsets each by `idx` internally. This guarantees every resampled draw uses a synchronised bootstrap sample across all variables — critical for the instrumented estimators (IV2SLS, PGC, PGC2Gm) whose instruments and NC panels must be resampled in lockstep with y, X, and M.

Value

A list with NDE_boot_se, NIE_boot_se, NDE_boot_dist, NIE_boot_dist.

`build_mediator_instruments` *Build mediator instruments via per-gene elastic net*

Description

Trains one cross-validated elastic-net model per gene predicting expression from cis SNPs, after residualizing both expression and dosages on `covariates` (Frisch-Waugh-Lovell). Genes are kept when the model is non-degenerate (at least one non-zero SNP weight), the signed out-of-fold R2 exceeds `cv_r2_min`, and the one-sided out-of-fold correlation p-value is below `cv_p_max`. The in-sample predicted expression of passing genes is the composite mediator instrument `Gm` consumed by `iconic_data()`.

Usage

```
build_mediator_instruments(
  expression,
  genotypes,
  gene_pos = NULL,
  snp_pos = NULL,
  covariates = NULL,
  cis_snps = NULL,
  alpha = 0.3,
  nfolds = 5,
  cis_dist = 1e+06,
  cv_r2_min = 0.005,
  cv_p_max = 0.1,
  n_cores = 1,
  seed = NULL,
  verbose = TRUE,
  expr_assay = 1L,
  geno_assay = 1L
)
```

Arguments

`expression`	Numeric matrix, genes x samples (e.g. inverse-normal-transformed TPM), or a SummarizedExperiment::RangedSummarizedExperiment whose `rowRanges()` supply gene coordinates (used when `gene_pos` is `NULL`).
`genotypes`	Numeric dosage matrix, variants x samples (0-2), or a `RangedSummarizedExperiment` whose `rowRanges()` supply variant positions (used when `snp_pos` is `NULL`).
`gene_pos`	Gene coordinates: a GenomicRanges::GRanges with gene IDs as `names()` or a gene metadata column (the TSS is the strand-aware 5' end of each range), or a data.frame with columns `gene`, `chr`, and `tss` (or `start`). May be `NULL` when `expression` is a `RangedSummarizedExperiment`.
`snp_pos`	Variant positions: a `GRanges` with variant IDs as `names()` or an `snp` metadata column, or a data.frame with columns `snp`, `chr`, `pos`. May be `NULL` when `genotypes` is a `RangedSummarizedExperiment`.
`covariates`	Optional data.frame or matrix (n rows) of covariates to partial out (ancestry PCs, technical factors, ...).

cis_snps	Optional named list: gene name -> character vector of candidate SNP IDs (e.g. significant GTEx cis-eQTL pairs, as from call_cis_eqtls() or an external resource). When supplied, the elastic net for each gene sees only these SNPs (intersected with the cis window).
alpha	Numeric: elastic-net mixing (0 = ridge, 1 = lasso). Default 0.3.
nfolds	Integer: cross-validation folds. Default 5.
cis_dist	Numeric: cis window half-width around the gene start, in bp. Default 1e6.
cv_r2_min	Numeric: minimum signed out-of-fold R2. Default 0.005.
cv_p_max	Numeric: maximum one-sided out-of-fold correlation p-value (nominal screen only; the load-bearing gates are cv_r2_min and the non-degeneracy requirement). Default 0.10.
n_cores	Integer: parallel workers via parallel::mclapply(). Default 1.
seed	Optional integer for reproducible CV fold assignments.
verbose	Logical: print progress. Default TRUE.
expr_assay, geno_assay	Name or index of the assay to use when expression / genotypes is a SummarizedExperiment. Default 1.

Details

Requires the glmnet package (listed under Suggests).

Coordinates are handled with GenomicRanges exactly as in call_cis_eqtls(); genes listed more than once in gene_pos are skipped.

Value

A list with:

Gm numeric matrix, genes x samples: in-sample predicted expression for genes passing the quality filter. Pass to iconic_data() as Gm (it is transposed internally).

qc data.frame of per-gene quality metrics: gene, n_cis_snps, n_nonzero, cv_r2, cor_oof, cv_p, lambda_min, pass.

weights named list (all trained genes) of data.frames with snp and weight (non-zero coefficients at lambda.min).

Examples

```
if (requireNamespace("glmnet", quietly = TRUE)) {
    set.seed(42)
    n <- 150; p <- 80
    dos <- matrix(rbinom(p * n, 2, 0.3), nrow = p,
        dimnames = list(paste0("1:", 1:p, ":A:G"),
            paste0("S", 1:n)))
    expr <- matrix(rnorm(4 * n), nrow = 4,
        dimnames = list(paste0("Gene", 1:4), paste0("S", 1:n)))
    expr[1, ] <- expr[1, ] + 0.9 * scale(dos[5, ]) + 0.9 * scale(dos[15, ])
    gp <- data.frame(gene = paste0("Gene", 1:4), chr = "1",
        tss = c(5, 25, 45, 65))
    sp <- data.frame(snp = rownames(dos), chr = "1", pos = 1:p)
    fit <- build_mediator_instruments(expr, dos, gp, sp, seed = 1)
    fit$qc
}
```

`build_prs_ldpred2` *Build an exposure polygenic score with LDpred2-auto*

Description

Runs the LDpred2-auto Bayesian shrinkage pipeline (Privé et al.) via the bigsnpr package: match summary statistics to an LD reference, estimate SNP heritability by LD score regression, run a grid of auto chains, filter chains by convergence, and average the per-chain effects. Optionally scores a target genotype set.

Usage

```
build_prs_ldpred2(
  sumstats,
  ld_ref,
  genotypes = NULL,
  hapmap3 = NULL,
  max_variants = NULL,
  n_chains = 30,
  p_range = c(1e-04, 0.2),
  burn_in = 500,
  num_iter = 500,
  shrink_corr = 0.95,
  allow_jump_sign = FALSE,
  use_mle = FALSE,
  chain_filter_quantile = 0.95,
  impute_mean = TRUE,
  ncores = 1,
  verbose = TRUE
)
```

Arguments

`sumstats`	A data.frame of GWAS summary statistics, ideally pre-processed by `qc_gwas_sumstats()`. Must contain (after standardization) `chr`, `pos`, `a0`, `a1`, `beta`, `beta_se`, `n_eff`; `freq` is used for the SD-ratio QC when present.
`ld_ref`	Either a path to a bigsnpr `.rds` backing file (from `bigsnpr::snp_readBed()`) or an attached `bigSNP` object. Should match the ancestry of the GWAS.
`genotypes`	Optional: path to a bigsnpr `.rds` or an attached `bigSNP` object for the target cohort to be scored. When `NULL` (default), only the LDpred2 effect sizes are returned.
`hapmap3`	Optional character vector of variant IDs (rsIDs or `chr:pos:a0:a1` keys) to restrict the analysis to (e.g. HapMap3). Restriction improves chain stability; `NULL` (default) uses all matched variants.
`max_variants`	Optional integer: if the matched variant count exceeds this, systematically thin to approximately `max_variants` variants (evenly spaced along the genome) before computing LD. Default `NULL` (no thinning). Use e.g. `250000` to bound memory.
`n_chains`	Integer: number of LDpred2-auto chains. Default 30.

p_range	Numeric length-2: range of the initial per-chain proportion-of-causal-variants grid, log-spaced. Default c(1e-4, 0.2).
burn_in	Integer: burn-in iterations per chain. Default 500.
num_iter	Integer: sampling iterations per chain (after burn-in). Default 500.
shrink_corr	Numeric: LD shrinkage parameter. Default 0.95; reduce to ~0.4 when the SD-ratio QC flags >10% of variants.
allow_jump_sign	Logical: allow sign jumps across chains. Default FALSE (recommended for auto mode).
use_mle	Logical: use MLE for the hyperparameter updates. Default FALSE (appropriate for moderate GWAS sample sizes).
chain_filter_quantile	Numeric: chains are kept when the range of their posterior corr_est exceeds chain_filter_quantile-quantile of all chain ranges times 0.95. Default 0.95.
impute_mean	Logical: mean-impute missing target genotypes before scoring. Default TRUE.
ncores	Integer: threads for bigsnpr. Default 1.
verbose	Logical: print progress. Default TRUE.

Details

This function requires the bigsnpr package (listed under Suggests) and a PLINK/bed-format LD reference panel converted with bigsnpr::snp_readBed().

Value

A list with:

beta data.frame of LDpred2 effect sizes (chr, pos, a0, a1, beta) in LD-reference orientation.

score numeric vector of per-sample polygenic scores for genotypes (scaled to unit variance), or NULL when genotypes was not supplied.

h2 LD-score-regression SNP heritability estimate used to initialize the chains.

chains_kept integer indices of chains passing the convergence filter.

qc list with variant counts at each stage.

Examples

```
if (requireNamespace("bigsnpr", quietly = TRUE)) {
    # Tiny fake genotype panel standing in for an LD reference
    # (snp_fake genotypes are all-missing mocks; fill with random dosages)
    fake <- bigsnpr::snp_fake(100, 500)
    fake$genotypes[] <- rbinom(100 * 500, 2, 0.3)
    fake$map$chromosome <- 1L
    fake$map$physical.pos <- sort(sample(1:1e6, 500))
    ss <- data.frame(
        chr = fake$map$chromosome, pos = fake$map$physical.pos,
        a0 = fake$map$allele2, a1 = fake$map$allele1,
        beta = rnorm(500, 0, 0.05), beta_se = 0.02, n_eff = 20000
    )
    prs <- build_prs_ldpred2(ss, ld_ref = fake, n_chains = 3,
        burn_in = 50, num_iter = 50, verbose = FALSE)
    str(prs$beta)
}
```

build_w_pcs *Build a negative-control panel from principal components*

Description

Computes the top `n_pcs` principal components of a features x samples matrix (typically residualized with `residualize_matrix()`) and returns the sample scores as a W panel (PCs x samples) in the orientation expected by `iconic_data()`. Uses irlba for fast truncated PCA when available, falling back to `stats::prcomp()`.

Usage

```
build_w_pcs(x, n_pcs = 20, scale_features = FALSE, prefix = "PC")
```

Arguments

`x`	Numeric matrix, features x samples (e.g. residualized M-values or predicted expression).
`n_pcs`	Integer: number of PCs to return. Default 20.
`scale_features`	Logical: scale features to unit variance before PCA. Default `FALSE` (appropriate for M-values); use `TRUE` for predicted-expression panels where feature scales differ.
`prefix`	Character: row-name prefix for the PCs. Default `"PC"`.

Value

A list with:

W numeric matrix, PCs x samples, suitable as the `W` argument of `iconic_data()`.

variance_explained numeric vector: percent of total variance explained by each returned PC.

Examples

```
x <- matrix(rnorm(500 * 60), nrow = 500,
    dimnames = list(paste0("f", 1:500), paste0("S", 1:60)))
w <- build_w_pcs(x, n_pcs = 5)
dim(w$W)
w$variance_explained
```

call_cis_eqtls *Scan for cis-eQTLs of each gene*

Description

For each gene, tests every cis SNP (within `cis_dist` of the gene start, optionally restricted to a supplied candidate set) for association with expression, after residualizing both expression and dosages on `covariates` (Frisch-Waugh-Lovell). A lightweight, pure-R alternative to MatrixEQTL for preparing mediator-instrument candidate sets.

Usage

```
call_cis_eqtls(
  expression,
  genotypes,
  gene_pos = NULL,
  snp_pos = NULL,
  covariates = NULL,
  cis_snps = NULL,
  cis_dist = 1e+06,
  min_maf = 0.01,
  fdr = 0.05,
  n_cores = 1,
  expr_assay = 1L,
  geno_assay = 1L
)
```

Arguments

expression	Numeric matrix, genes x samples (e.g. inverse-normal-transformed TPM), or a SummarizedExperiment::RangedSummarizedExperiment whose rowRanges() supply gene coordinates (used when gene_pos is NULL).
genotypes	Numeric dosage matrix, variants x samples (0-2), or a RangedSummarizedExperiment whose rowRanges() supply variant positions (used when snp_pos is NULL).
gene_pos	Gene coordinates: a GenomicRanges::GRanges with gene IDs as names() or a gene metadata column (the TSS is the strand-aware 5' end of each range), or a data.frame with columns gene, chr, and tss (or start). May be NULL when expression is a RangedSummarizedExperiment.
snp_pos	Variant positions: a GRanges with variant IDs as names() or an snp metadata column, or a data.frame with columns snp, chr, pos. May be NULL when genotypes is a RangedSummarizedExperiment.
covariates	Optional data.frame or matrix (n rows) of covariates to partial out (e.g. ancestry PCs, technical factors).
cis_snps	Optional named list: gene name -> character vector of candidate SNP IDs (e.g. significant GTEx pairs). When supplied, only these SNPs are tested for each gene (still intersected with the cis window).
cis_dist	Numeric: cis window half-width around the gene start, in bp. Default 1e6.
min_maf	Numeric: minimum minor allele frequency (computed from mean dosage). Default 0.01.
fdr	Numeric: BH FDR level for the per-gene pass flag in the best table. Default 0.05.
n_cores	Integer: parallel workers via parallel::mclapply(). Default 1.
expr_assay, geno_assay	Name or index of the assay to use when expression / genotypes is a SummarizedExperiment. Default 1.

Details

Gene and variant coordinates are handled as GenomicRanges::GRanges and cis variants are resolved with GenomicRanges::findOverlaps() against symmetric windows of cis_dist bp around each gene's TSS. Chromosome naming styles are harmonised first, so "chr1" and "1" match.

Value

A list with:

all data.frame of all tested gene-SNP pairs: `gene`, `snp`, `beta`, `se`, `t`, `p`.

best data.frame with the top SNP per gene (`gene`, `snp`, `beta`, `p`, `q` (BH within gene), `pass`).

Examples

```
n <- 80
dos <- matrix(rbinom(200 * n, 2, 0.3), nrow = 200,
    dimnames = list(paste0("1:", 1:200, ":A:G"), NULL))
expr <- matrix(rnorm(20 * n), nrow = 20,
    dimnames = list(paste0("Gene", 1:20), NULL))
expr[1, ] <- expr[1, ] + 0.5 * dos[5, ] # one true cis-eQTL
gp <- data.frame(gene = paste0("Gene", 1:20), chr = "1",
    tss = seq(1, 191, by = 10))
sp <- data.frame(snp = rownames(dos), chr = "1", pos = 1:200)
hits <- call_cis_eqtls(expr, dos, gp, sp)
head(hits$best)

# Equivalent call with GRanges coordinates (UCSC-style "chr1" names
# are harmonised with the Ensembl-style "1" used above)
gp_gr <- GenomicRanges::GRanges("chr1",
    IRanges::IRanges(seq(1, 191, by = 10), width = 1),
    gene = paste0("Gene", 1:20))
sp_gr <- GenomicRanges::GRanges("1", IRanges::IRanges(1:200, width = 1),
    snp = rownames(dos))
hits_gr <- call_cis_eqtls(expr, dos, gp_gr, sp_gr)
identical(hits$best, hits_gr$best)
```

---

check_instrument_strength

Check first-stage instrument strength (partial F)

Description

Regresses the exposure `X` on the instrument `G` (plus optional covariates) and reports the partial F statistic and partial R2 for the instrument — the weak-instrument diagnostic. The partial F (not the overall model F) is the relevant quantity when covariates such as ancestry PCs explain much of the exposure.

Usage

```
check_instrument_strength(G, X, covariates = NULL, min_f = 10)
```

Arguments

`G`	Numeric vector (length n): the exposure instrument (e.g. a polygenic score from `build_prs_ldpred2()` or `score_pgs_panel()`).
`X`	Numeric vector (length n): the exposure.
`covariates`	Optional data.frame or matrix (n rows) of covariates to partial out (e.g. ancestry PCs).
`min_f`	Numeric: weak-instrument threshold on the partial F. Default 10 (Staiger-Stock rule of thumb).

Value

A list with F (partial F statistic), df1, df2, pvalue, partial_r2, n, and weak (TRUE when F < min_f).

Examples

```
n <- 300
G <- rnorm(n)
X <- 0.3 * G + rnorm(n)
pcs <- matrix(rnorm(n * 3), n, 3, dimnames = list(NULL, paste0("PC", 1:3)))
check_instrument_strength(G, X, covariates = pcs)
```

---

check_torch_setup *Check whether a working torch installation is available*

Description

Verifies the package is installed, libtorch is present, and a basic tensor op succeeds. Used to gate the GAN path; a FALSE result triggers the multivariate-normal fallback.

Usage

```
check_torch_setup()
```

Value

TRUE if torch is usable, otherwise FALSE.

Examples

```
check_torch_setup()
```

---

composite_p_value *Composite null hypothesis test p-value*

Description

Computes the Huang (2019) JT-comp composite p-value for testing H0: alpha * beta = 0 against H1: alpha * beta != 0.

Usage

```
composite_p_value(a, b, var_a = 1, var_b = 1)
```

Arguments

a	Standardized z-statistic for alpha (alpha_hat / SE).
b	Standardized z-statistic for beta (beta_hat / SE).
var_a	Variance of the z-statistic for alpha across the collection of tests. Under the point null this is ~1; under H0(2)/H0(3) it is >1. Default 1 (conservative).
var_b	Variance of the z-statistic for beta across the collection of tests. Default 1.

Details

The composite null decomposes into three cases (alpha=beta=0, alpha!=0/beta=0, alpha=0/beta!=0). The Wald/Sobel test is conservative under the first case because the product of two independent normals follows a normal product distribution, not a normal. This function computes a closed-form p-value that accounts for all three cases without estimating their proportions.

Value

Numeric scalar in $[0, 1]$.

References

Huang, Y.-T. (2019). Genome-wide analyses of sparse mediation effects under composite null hypotheses. *Annals of Applied Statistics*, 13(1), 60-84.

Examples

```
composite_p_value(0, 0) # null: p = 1
composite_p_value(2, 2) # strong signal: small p
composite_p_value(2, 0) # H0(2): p = 1 (no mediation)
```

---

compute_iter_bias *Extract per-seed bias vector (internal)*

Description

Extract per-seed bias vector (internal)

Usage

```
compute_iter_bias(combined, true_total)
```

Arguments

combined	Data frame from run_methods().
true_total	Scalar true total causal effect.

Value

Data frame with columns: iter, method, bias.

create_discriminator *Build the GAN discriminator network (internal to the torch path)*

Description

MLP `input_dim -> 1024 -> 512 -> 256 -> 1` with leaky-ReLU and dropout, returning a real-vs-fake logit (no sigmoid; paired with a BCE-with-logits loss).

Usage

```
create_discriminator(input_dim)
```

Arguments

`input_dim`	Number of input variables.

Value

A torch `nn_module` instance. Requires torch.

create_generator *Build the GAN generator network (internal to the torch path)*

Description

MLP mapping a noise vector to a synthetic row: `noise_dim -> 256 -> 512 -> 1024 -> output_dim`, ReLU + batch-norm + dropout, linear output.

Usage

```
create_generator(output_dim, noise_dim = 100)
```

Arguments

`output_dim`	Number of output variables (columns of the training frame).
`noise_dim`	Latent noise dimension. Default 100.

Value

A torch `nn_module` instance. Requires torch.

delta_se_product *Delta-method SE for a product alpha * beta (internal)*

Description

Delta-method SE for a product alpha * beta (internal)

Usage

```
delta_se_product(alpha, alpha_se, beta, beta_se, cov_ab = 0)
```

Arguments

alpha	Point estimate of alpha.
alpha_se	SE of alpha.
beta	Point estimate of beta.
beta_se	SE of beta.
cov_ab	Estimated covariance of alpha and beta. Default 0.

Value

Scalar SE of the product alpha * beta.

fit_coca *COCA estimator: Negative-Control Outcome Correction via ratio*

Description

Implements the Correlated Outcome Control Approach (COCA). Fits `w ~ y + X + covars` and recovers the causal effect as $\hat{\beta} = -\hat{\beta}_X / \hat{\beta}_Y$. Standard errors are obtained via the delta method.

Usage

```
fit_coca(y, X, w, covars = NULL, ratio_cap = 10, se_cap = 5)
```

Arguments

y	Numeric primary outcome vector (length n).
X	Numeric exposure vector (length n).
w	Numeric negative-control outcome vector (length n). Recommended: pass `rowMeans(W_matrix)` for stability.
covars	Optional data frame of additional covariates (n rows).
ratio_cap	Maximum absolute value of the ratio estimate before flagging as unstable and returning `NA`. Default 10.
se_cap	Maximum SE before flagging as unstable. Default 5.

Details

The negative-control W should be an outcome that shares the same unmeasured confounders as Y but has no direct causal path from X.

Value

Named list: `beta`, `se`, `pvalue`. Returns `list(beta=NA, se=NA, pvalue=NA)` if estimation is unstable (near-zero $\hat{\beta}_Y$ or extreme ratio).

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 200, seed = 1)
fit_coca(dat$Y[, 1], dat$X, rowMeans(dat$W))
```

`fit_coca_bin` *COCA binary estimator: NOT supported (returns NA)*

Description

COCA (Correlated Outcome Control Approach) regresses the negative control W on the outcome Y (`W ~ y + X`) and recovers the causal effect as a ratio $-\hat{\beta}_X/\hat{\beta}_Y$. That identification argument assumes a linear structural outcome model: the W-Y regression coefficient must be proportional to the confounder-Outcome association on the same scale as the causal effect. With a binary outcome the structural model is nonlinear (logistic), so the linear COCA ratio recovers neither the causal log-odds ratio nor the risk difference. COCA is therefore unsupported for binary outcomes and always returns `list(beta=NA, se=NA, pvalue=NA)`.

Usage

```
fit_coca_bin(y, X, w, covars = NULL, ...)
```

Arguments

`y`	Numeric 0/1 outcome vector (length n).
`X`	Numeric exposure vector (length n).
`w`	Numeric NC vector (length n).
`covars`	Optional data frame of covariates (n rows).
`...`	Ignored (accepted for signature compatibility).

Value

`list(beta = NA, se = NA, pvalue = NA)` with an informative `"reason"` attribute.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 200, outcome_type = "binary", seed = 1)
fit_coca_bin(dat$y_bin, dat$X, dat$W[, 1])
# $beta [1] NA (COCA unsupported for binary outcomes)
```

`fit_coca_mediation` *COCA mediation estimator: negative-control calibration of both stages*

Description

Uses the negative-control outcome W to calibrate both the mediator and outcome regressions, extending `fit_coca` to the mediation setting.

Usage

```
fit_coca_mediation(y, X, M, w, covars = NULL, ratio_cap = 10)
```

Arguments

`y`	Numeric primary outcome vector (length n).
`X`	Numeric exposure vector (length n).
`M`	Numeric mediator vector (length n).
`w`	Numeric negative-control outcome vector (length n). Recommended: pass `rowMeans(W_matrix)` for stability.
`covars`	Optional data frame of additional covariates (n rows).
`ratio_cap`	Maximum absolute value of any ratio estimate before flagging as unstable. Default 10.

Details

Stage 1: `W ~ M + X` -> calibrated alpha_M = -beta_X / beta_M. Stage 2: `W ~ Y + X + M` -> calibrated NDE = -beta_X / beta_Y, calibrated beta_M = -beta_M / beta_Y. NIE = alpha_M * beta_M (both calibrated).

Value

Named list: `NDE`, `NDE_se`, `NDE_p`, `NIE`, `NIE_se`, `NIE_p`.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 200, mo_confounding = 0.8, seed = 1)
fit_coca_mediation(dat$Y[, 1], dat$X, dat$M, rowMeans(dat$W))
```

fit_coca_mediation_bin

COCA binary mediation estimator: NOT supported (returns NA)

Description

COCA mediation regresses the negative control W on the outcome Y in both stages (`W ~ M + X`, `W ~ y + X + M`) and recovers the effect as a ratio of regression coefficients. That identification argument assumes a linear structural outcome model; with a binary (nonlinear, logistic) outcome the ratio recovers neither the causal log-odds ratio nor the risk difference. COCA mediation is therefore unsupported for binary outcomes and always returns NA.

Usage

```
fit_coca_mediation_bin(y, X, M, w, covars = NULL, ...)
```

Arguments

`y`	Numeric 0/1 outcome vector (length n).
`X`	Numeric exposure vector (length n).
`M`	Numeric mediator vector (length n).
`w`	Numeric NC vector (length n).
`covars`	Optional data frame of covariates (n rows).
`...`	Ignored (signature compatibility).

Value

All-NA list with an informative `"reason"` attribute.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 200, outcome_type = "binary", seed = 1)
fit_coca_mediation_bin(dat$y_bin, dat$X, dat$M, dat$W[, 1])
# all NA (COCA mediation unsupported for binary outcomes)
```

fit_coca_mediation_surv

COCA survival mediation estimator: NOT supported (returns NA)

Description

COCA mediation regresses the negative control W on the outcome Y in both stages (`W ~ M + X`, `W ~ y + X + M`), placing the outcome on the right-hand side. This is structurally impossible when the outcome is a `survival::Surv` object. COCA mediation is therefore unsupported for survival outcomes and always returns NA.

Usage

```
fit_coca_mediation_surv(time, event, X, M, w, covars = NULL, ...)
```

Arguments

time	Numeric follow-up time vector (length n).
event	Numeric 0/1 event indicator (length n).
X	Numeric exposure vector (length n).
M	Numeric mediator vector (length n).
w	Numeric NC vector (length n).
covars	Optional data frame of covariates (n rows).
...	Ignored (signature compatibility).

Value

All-NA list with an informative "reason" attribute.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 200, outcome_type = "survival", seed = 1)
fit_coca_mediation_surv(dat$surv_time, dat$surv_event, dat$X, dat$M, dat$W[, 1])
# all NA (COCA mediation unsupported for survival outcomes)
```

---

fit_coca_surv *COCA survival estimator: NOT supported (returns NA)*

Description

COCA (Correlated Outcome Control Approach) regresses the negative control W on the outcome Y (W ~ y + X) and recovers the causal effect as a ratio $-\hat{\beta}_X/\hat{\beta}_Y$. This places the outcome on the right-hand side of the regression, which is structurally impossible when the outcome is a survival::Surv object (time-to-event). COCA is therefore unsupported for survival outcomes and always returns list(beta=NA, se=NA, pvalue=NA).

Usage

```
fit_coca_surv(time, event, X, w, covars = NULL, ...)
```

Arguments

time	Numeric follow-up time vector (length n).
event	Numeric 0/1 event indicator (length n).
X	Numeric exposure vector (length n).
w	Numeric NC vector (length n).
covars	Optional data frame of covariates (n rows).
...	Ignored (accepted for signature compatibility).

Value

`list(beta = NA, se = NA, pvalue = NA)` with an informative `"reason"` attribute.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 200, outcome_type = "survival", seed = 1)
fit_coca_surv(dat$surv_time, dat$surv_event, dat$X, dat$W[, 1])
# $beta [1] NA (COCA unsupported for survival outcomes)
```

---

fit_direct *DIRECT estimator: OLS with instrument and negative-control as covariates*

Description

Regresses Y on X plus the genetic instrument G, the negative-control W, and any additional covariates. This is a "naive" adjustment that uses whatever observables are available but does NOT correct for unmeasured confounding via a ratio or IV approach.

Usage

```
fit_direct(y, X, g, w, covars = NULL)
```

Arguments

`y`	Numeric outcome vector (length n).
`X`	Numeric exposure vector (length n), assumed pre-scaled.
`g`	Numeric instrument vector (length n).
`w`	Numeric negative-control vector (length n) or matrix (n x q). When a matrix, all q columns are included as separate covariates.
`covars`	Optional data frame of additional covariates (n rows).

Value

Named list: `beta`, `se`, `pvalue`.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 200, seed = 1)
fit_direct(dat$Y[, 1], dat$X, dat$G[, 1], dat$W[, 1])
```

`fit_direct_bin` *DIRECT binary estimator: logistic / LPM with instrument and NC covariates*

Description

Regresses the binary outcome on X plus the genetic instrument G, the negative-control panel W, and covariates. Naive adjustment; does not correct for unmeasured confounding via a ratio or IV approach.

Usage

```
fit_direct_bin(
  y,
  X,
  g,
  w,
  covars = NULL,
  effect_scale = c("logor", "riskdiff")
)
```

Arguments

`y`	Numeric 0/1 outcome vector (length n).
`X`	Numeric exposure vector (length n).
`g`	Numeric instrument vector (length n).
`w`	Numeric NC vector (length n) or matrix (n x q).
`covars`	Optional data frame of covariates (n rows).
`effect_scale`	Character: `"logor"` or `"riskdiff"`.

Value

Named list: `beta`, `se`, `pvalue`.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 200, outcome_type = "binary", seed = 1)
fit_direct_bin(dat$y_bin, dat$X, dat$G[, 1], dat$W[, 1])
```

fit_direct_mediation *DIRECT mediation estimator: OLS with instrument and NC as covariates*

Description

Adjusts for the genetic instrument G and negative-control W in both the mediator and outcome regressions. Like fit_direct, this is a naive adjustment that does not correct for unmeasured confounding via a ratio or IV approach.

Usage

```
fit_direct_mediation(y, X, M, g, w, covars = NULL)
```

Arguments

y	Numeric outcome vector (length n).
X	Numeric exposure vector (length n).
M	Numeric mediator vector (length n).
g	Numeric instrument vector (length n).
w	Numeric negative-control vector (length n).
covars	Optional data frame of additional covariates (n rows).

Value

Named list: NDE, NDE_se, NDE_p, NIE, NIE_se, NIE_p.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 200, mo_confounding = 0.8, seed = 1)
fit_direct_mediation(dat$Y[, 1], dat$X, dat$M, dat$G[, 1], dat$W[, 1])
```

fit_direct_mediation_bin *DIRECT binary mediation estimator: logistic / LPM with G and W covariates*

Description

Adjusts for the instrument G and negative-control W in both the mediator (OLS) and outcome (logistic / LPM) stages. Naive adjustment.

Usage

```
fit_direct_mediation_bin(
  y,
  X,
  M,
  g,
  w,
  covars = NULL,
  effect_scale = c("logor", "riskdiff")
)
```

Arguments

y	Numeric 0/1 outcome vector (length n).
X	Numeric exposure vector (length n).
M	Numeric mediator vector (length n).
g	Numeric instrument vector (length n).
w	Numeric NC vector (length n) or matrix (n x q).
covars	Optional data frame of covariates (n rows).
effect_scale	Character: "logor" or "riskdiff".

Value

Named list (same fields as fit_unadj_mediation_bin).

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 200, outcome_type = "binary",
    mo_confounding = 0.8, seed = 1)
fit_direct_mediation_bin(dat$y_bin, dat$X, dat$M,
    dat$G[, 1], dat$W[, 1])
```

---

fit_direct_mediation_surv

DIRECT survival mediation estimator: Cox / RMST with G and W covariates

---

Description

Adjusts for the instrument G and negative-control W in both the mediator (OLS) and outcome (Cox / RMST) stages. Naive adjustment.

Usage

```
fit_direct_mediation_surv(
  time,
  event,
  X,
  M,
  g,
  w,
  covars = NULL,
  effect_scale = c("loghr", "rmst"),
  tau = NULL
)
```

Arguments

time	Numeric follow-up time vector (length n).
event	Numeric 0/1 event indicator (length n).
X	Numeric exposure vector (length n).
M	Numeric mediator vector (length n).
g	Numeric instrument vector (length n).
w	Numeric NC vector (length n) or matrix (n x q).
covars	Optional data frame of covariates (n rows).
effect_scale	Character: "loghr" or "rmst".
tau	RMST horizon (rmst only).

Value

Named list (same fields as fit_unadj_mediation_surv).

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 200, outcome_type = "survival",
    mo_confounding = 0.8, seed = 1)
fit_direct_mediation_surv(dat$surv_time, dat$surv_event, dat$X, dat$M,
    dat$G[, 1], dat$W[, 1])
```

---

fit_direct_surv *DIRECT survival estimator: Cox / RMST with instrument and NC covariates*

Description

Regresses the survival outcome on X plus the genetic instrument G, the negative-control panel W, and covariates. Naive adjustment; does not correct for unmeasured confounding via a ratio or IV approach.

Usage

```
fit_direct_surv(
  time,
  event,
  X,
  g,
  w,
  covars = NULL,
  effect_scale = c("loghr", "rmst"),
  tau = NULL
)
```

Arguments

time	Numeric follow-up time vector (length n).
event	Numeric 0/1 event indicator (length n).
X	Numeric exposure vector (length n).
g	Numeric instrument vector (length n).
w	Numeric NC vector (length n) or matrix (n x q).
covars	Optional data frame of covariates (n rows).
effect_scale	Character: "loghr" or "rmst".
tau	RMST horizon (rmst only).

Value

Named list: beta, se, pvalue.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 200, outcome_type = "survival", seed = 1)
fit_direct_surv(dat$surv_time, dat$surv_event, dat$X, dat$G[, 1], dat$W[, 1])
```

---

fit_iv2sls *IV2SLS estimator: Two-Stage Least Squares with genetic instrument*

Description

Uses the genetic instrument G to instrument for the exposure X, controlling for the negative-control W and any additional covariates. Requires **AER**.

Usage

```
fit_iv2sls(y, X, g, w, covars = NULL, min_f = 10)
```

Arguments

y	Numeric outcome vector (length n).
X	Numeric exposure vector (length n).
g	Numeric instrument vector (length n).
w	Numeric negative-control vector (length n).
covars	Optional data frame of additional covariates (n rows).
min_f	Minimum acceptable partial F-statistic for the excluded instrument. Default 10.

Details

A weak-instrument check is applied using the partial F-statistic for the excluded instrument G (testing G conditional on W and covariates in the first stage), following Stock & Yogo (2005). If the partial F is below `min_f`, the function returns `NA`.

Value

Named list: `beta`, `se`, `pvalue`.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 200, seed = 1)
fit_iv2sls(dat$Y[, 1], dat$X, dat$G[, 1], dat$W[, 1])
```

---

fit_iv2sls_bin *IV2SLS binary estimator: two-stage predictor substitution with logistic / LPM*

Description

Two-stage predictor substitution (2SPS): the first stage regresses X on the instrument G (plus W and covariates) via OLS, producing fitted $\hat{X}$; the second stage regresses the binary outcome on $\hat{X}$ (plus W and covariates) via logistic regression (log-OR) or a linear probability model (risk difference).

Usage

```
fit_iv2sls_bin(
  y,
  X,
  g,
  w,
  covars = NULL,
  min_f = 10,
  effect_scale = c("logor", "riskdiff")
)
```

Arguments

y	Numeric 0/1 outcome vector (length n).
X	Numeric exposure vector (length n).
g	Numeric instrument vector (length n).
w	Numeric NC vector (length n) or matrix (n x q).
covars	Optional data frame of covariates (n rows).
min_f	Minimum partial F for the excluded instrument. Default 10.
effect_scale	Character: "logor" or "riskdiff".

Details

A weak-instrument check (partial F for the excluded instrument G, Stock & Yogo 2005) is applied to the OLS first stage. If the partial F is below min_f, the function returns NA.

Value

Named list: beta, se, pvalue.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 200, outcome_type = "binary", seed = 1)
fit_iv2sls_bin(dat$y_bin, dat$X, dat$G[, 1], dat$W[, 1])
```

---

fit_iv2sls_mediation *IV2SLS mediation estimator: instrumented exposure in both stages*

Description

Uses the genetic instrument G to purge U1 from X, then estimates the mediator and outcome regressions with the cleaned exposure.

Usage

```
fit_iv2sls_mediation(y, X, M, g, w, covars = NULL, min_f = 10)
```

Arguments

y	Numeric outcome vector (length n).
X	Numeric exposure vector (length n).
M	Numeric mediator vector (length n).
g	Numeric instrument vector (length n).
w	Numeric negative-control vector (length n).
covars	Optional data frame of additional covariates (n rows).
min_f	Minimum acceptable partial F-statistic for the excluded instrument. Default 10.

Details

Strategy:

1. `X ~ G + W` -> X_hat (purge U1 from X).
2. `M ~ X_hat` -> alpha_M (clean effect of X on M).
3. `Y ~ X_hat + M + W` -> NDE = beta_X, beta_M (OLS).

The IV cleans X of U1 confounding, but M remains endogenous via U1 -> M. With a single instrument, natural effects are not fully identified (Rudolph et al., 2024); NDE and NIE are approximations whose bias from M-O confounding is the key finding the simulation demonstrates.

Value

Named list: `NDE`, `NDE_se`, `NDE_p`, `NIE`, `NIE_se`, `NIE_p`.

References

Rudolph, K. E., et al. (2024). Natural direct and indirect effects with an instrumental variable. *Biometrics*.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 300, mo_confounding = 0.8, seed = 1)
fit_iv2sls_mediation(dat$Y[, 1], dat$X, dat$M, dat$G[, 1], dat$W[, 1])
```

`fit_iv2sls_mediation2` *IV2SLS2 mediation estimator: 2-stage MR with instruments for both X and M*

Description

Uses TWO genetic instruments: G for the exposure X and Gm for the mediator M. This is the key extension: by instrumenting both endogenous variables, NDE and NIE become **point-identified** even under mediator-outcome (M-O) confounding, resolving the identification failure that limits the single-instrument estimators.

Usage

```
fit_iv2sls_mediation2(
  y,
  X,
  M,
  g,
  gm,
  covars = NULL,
  min_f = 10,
  W1 = NULL,
  W2 = NULL,
  w = NULL
)
```

Arguments

y	Numeric outcome vector (length n).
X	Numeric exposure vector (length n).
M	Numeric mediator vector (length n).
g	Numeric instrument for X (length n).
gm	Numeric instrument for M (length n).
covars	Optional data frame of additional covariates (n rows).
min_f	Minimum acceptable partial F-statistic for each excluded instrument. Default 10.
W1	Optional negative-control panel (vector length n or matrix n x q) proxying the exposure-mediator confounder (X->M path); added to stage 1 only. Default NULL (stage 1 unaugmented).
W2	Optional negative-control panel (vector length n or matrix n x q) proxying the mediator-outcome confounder (M->Y path); added to stages 2 and 3. Default NULL (stages 2–3 unaugmented).
w	Defunct. The pooled single-panel argument was removed because conditioning on a pooled panel in all three stages opens a collider path under multi-confounder designs. Use W1 and/or W2 instead.

Details

The motivating design uses two distinct genetic instruments: a mediator instrument (e.g. a cis-eQTL) instruments the mediator (M), while a separate exposure instrument (e.g. a polygenic score) instruments the exposure (X). The mediator set must be restricted to features for which a mediator instrument is available.

Strategy (sequential 2SLS, three OLS stages), with optional **path-specific** negative-control (NC) augmentation:

1. X ~ G (+ W1) + covars -> X_hat (purge U1 from X). Weak-IV check: partial F for G >= min_f.
2. M ~ X_hat + Gm (+ W2) + covars -> M_hat, alpha_M = coef on X_hat. Weak-IV check: partial F for Gm >= min_f.
3. Y ~ X_hat + M_hat (+ W2) + covars -> NDE = beta_X_hat, beta_M = coef on M_hat.

NIE = alpha_M * beta_M (delta-method SE).

Path-specific NC augmentation (optional). W1 proxies the exposure-mediator confounder (U1, the X->M path) and is added to stage 1 only; W2 proxies the mediator-outcome confounder (U2, the M->Y path) and is added to stages 2 and 3. Either panel may be omitted: with W1 = NULL stage 1 is unaugmented, with W2 = NULL stages 2–3 are unaugmented, and with both NULL the estimator reduces to plain two-instrument 2-stage MR. Conditioning on a *pooled* panel (W1 + W2) / 2 in all three stages is a collider under multi-confounder designs (the pooled panel is a common child of the independent confounders U1 and U2) and is therefore not supported: if W1 and W2 are identical they are treated as absent (pure MR). Pass the two path-specific panels separately to obtain the coverage-improves-accuracy behaviour.

Path-specific augmentation is only defined when W1 and W2 proxy *distinct* confounders. When the two panels are distinct-noise proxies of the *same* latent composite (the single-confounder / shared-loading design), their column spaces are near-collinear and stage-1 W1 would inject the shared M->Y confounder into X_hat. The estimator detects this (leading-PC correlation between the panels) and falls back to plain two-instrument 2-stage MR, exactly as for an identical pooled panel. Genuinely distinct panels are unaffected.

Including X_hat in the M first-stage (stage 2) is essential: M structurally depends on X, so the X -> M path must be captured for alpha_M and the NIE to be correctly estimated. Gm provides the exogenous variation that identifies the M -> Y effect net of U1 confounding.

The sequential 2SLS implementation matches the pattern of `fit_iv2sls_mediation` (three separate OLS stages). Standard errors in the outcome stage do not account for the two-stage estimation of M_hat, consistent with the existing estimators; the simulation benchmarks bias, not SE accuracy. A unit test cross-validates this estimator's NDE and beta_M against `AER::ivreg` on the just-identified system (`Y ~ X + M | G + Gm`, pure-MR form).

Value

Named list: `NDE`, `NDE_se`, `NDE_p`, `NIE`, `NIE_se`, `NIE_p`. Returns all-`NA` if either first-stage partial F is below `min_f`.

References

Rudolph, K. E., et al. (2024). Natural direct and indirect effects with an instrumental variable. *Biometrics*.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 500, mo_confounding = 0.8,
    phi = 0.8, lambda_XM = c(1, 0), lambda_MY = c(0, 1),
    omega_1 = 0.7, omega_2 = 0.7, seed = 1)
fit_iv2sls_mediation2(dat$Y[, 1], dat$X, dat$M, dat$G[, 1], dat$Gm,
    W1 = dat$W1, W2 = dat$W2)
```

`fit_iv2sls_mediation2_bin`

IV2SLS2 binary mediation estimator: 2-stage MR with logistic / LPM outcome

---

Description

Two-stage Mendelian-randomization mediation with a binary outcome stage (2SPS), with optional **path-specific** negative-control (NC) augmentation:

1. OLS: `X ~ g (+ W1) + covars` -> X_hat (purge U1 from X).
2. OLS: `M ~ X_hat + gm (+ W2) + covars` -> M_hat, alpha_M.
3. Logistic / LPM: `y ~ X_hat + M_hat (+ W2) + covars` -> NDE (coef on X_hat), beta_M (coef on M_hat).

`NIE = alpha_M * beta_M`. Weak-instrument gates (partial F for G and Gm) apply to the OLS first stages.

Usage

```
fit_iv2sls_mediation2_bin(
  y,
  X,
  M,
  g,
  gm,
  covars = NULL,
  min_f = 10,
  W1 = NULL,
  W2 = NULL,
  w = NULL,
  effect_scale = c("logor", "riskdiff")
)
```

Arguments

y	Numeric 0/1 outcome vector (length n).
X	Numeric exposure vector (length n).
M	Numeric mediator vector (length n).
g	Numeric instrument for X (length n).
gm	Numeric instrument for M (length n).
covars	Optional data frame of covariates (n rows).
min_f	Minimum partial F for each excluded instrument. Default 10.
W1	Optional NC panel (vector length n or matrix n x q) proxying the exposure-mediator confounder (X->M path); added to stage 1 only. Default NULL.
W2	Optional NC panel (vector length n or matrix n x q) proxying the mediator-outcome confounder (M->Y path); added to stages 2 and 3. Default NULL.
w	Defunct. The pooled single-panel argument was removed (collider under multi-confounder designs). Use W1 and/or W2 instead.
effect_scale	Character: "logor" or "riskdiff".

Details

Path-specific NC augmentation (optional). W1 proxies the exposure-mediator confounder (U1, X->M path) and is added to stage 1 only; W2 proxies the mediator-outcome confounder (U2, M->Y path) and is added to stages 2 and 3. Either panel may be omitted; with both NULL the estimator reduces to plain two-instrument 2-stage MR. Conditioning on a *pooled* panel in all three stages is a collider under multi-confounder designs and is not supported: identical W1/W2 are treated as absent (pure MR). When W1 and W2 are distinct-noise proxies of the *same* latent composite (single-confounder design), their column spaces are near-collinear and the estimator likewise falls back to plain two-instrument 2-stage MR.

Value

Named list (same fields as fit_unadj_mediation_bin). Returns all-NA if either first-stage partial F is below min_f.

References

Rudolph, K. E., et al. (2024). Natural direct and indirect effects with an instrumental variable. *Biometrics*.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 500, outcome_type = "binary",
    mo_confounding = 0.8, phi = 0.8, lambda_XM = c(1, 0),
    lambda_MY = c(0, 1), omega_1 = 0.7, omega_2 = 0.7, seed = 1)
fit_iv2sls_mediation2_bin(dat$y_bin, dat$X, dat$M,
    dat$G[, 1], dat$Gm, W1 = dat$W1, W2 = dat$W2)
```

---

fit_iv2sls_mediation2_surv

IV2SLS2 survival mediation estimator: 2-stage MR with Cox / RMST outcome

---

Description

Two-stage Mendelian-randomization mediation with a survival outcome stage (2SPS), with optional **path-specific** negative-control (NC) augmentation:

1. OLS: `X ~ g (+ W1) + covars` -> X_hat (purge U1 from X).
2. OLS: `M ~ X_hat + gm (+ W2) + covars` -> M_hat, alpha_M.
3. Cox / RMST: `Surv(t,e) ~ X_hat + M_hat (+ W2) + covars` -> NDE (coef on X_hat), beta_M (coef on M_hat).

`NIE = alpha_M * beta_M`. Weak-instrument gates (partial F for G and Gm) apply to the OLS first stages.

Usage

```
fit_iv2sls_mediation2_surv(
  time,
  event,
  X,
  M,
  g,
  gm,
  covars = NULL,
  min_f = 10,
  W1 = NULL,
  W2 = NULL,
  w = NULL,
  effect_scale = c("loghr", "rmst"),
  tau = NULL
)
```

Arguments

time	Numeric follow-up time vector (length n).
event	Numeric 0/1 event indicator (length n).
X	Numeric exposure vector (length n).
M	Numeric mediator vector (length n).
g	Numeric instrument for X (length n).
gm	Numeric instrument for M (length n).
covars	Optional data frame of covariates (n rows).
min_f	Minimum partial F for each excluded instrument. Default 10.
W1	Optional NC panel (vector length n or matrix n x q) proxying the exposure-mediator confounder (X->M path); added to stage 1 only. Default NULL.
W2	Optional NC panel (vector length n or matrix n x q) proxying the mediator-outcome confounder (M->Y path); added to stages 2 and 3. Default NULL.
w	Defunct. The pooled single-panel argument was removed (collider under multi-confounder designs). Use W1 and/or W2 instead.
effect_scale	Character: "loghr" or "rmst".
tau	RMST horizon (rmst only).

Details

Path-specific NC augmentation (optional). W1 proxies the exposure-mediator confounder (U1, X->M path) and is added to stage 1 only; W2 proxies the mediator-outcome confounder (U2, M->Y path) and is added to stages 2 and 3. Either panel may be omitted; with both NULL the estimator reduces to plain two-instrument 2-stage MR. Conditioning on a *pooled* panel in all three stages is a collider under multi-confounder designs and is not supported: identical W1/W2 are treated as absent (pure MR). When W1 and W2 are distinct-noise proxies of the *same* latent composite (single-confounder design), their column spaces are near-collinear and the estimator likewise falls back to plain two-instrument 2-stage MR.

Value

Named list (same fields as fit_unadj_mediation_surv). Returns all-NA if either first-stage partial F is below min_f.

References

Rudolph, K. E., et al. (2024). Natural direct and indirect effects with an instrumental variable. *Biometrics*.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 500, outcome_type = "survival",
    mo_confounding = 0.8, phi = 0.8, lambda_XM = c(1, 0),
    lambda_MY = c(0, 1), omega_1 = 0.7, omega_2 = 0.7, seed = 1)
fit_iv2sls_mediation2_surv(dat$surv_time, dat$surv_event, dat$X, dat$M,
    dat$G[, 1], dat$Gm, W1 = dat$W1, W2 = dat$W2)
```

`fit_iv2sls_mediation_bin` *IV2SLS binary mediation estimator: single-instrument 2SPS with logistic / LPM*

Description

Two-stage predictor substitution with a single instrument (G for X, no mediator instrument):

1. OLS: `X ~ g + W + covars` -> X_hat (purge U1 from X).
2. OLS: `M ~ X_hat + covars` -> alpha_M.
3. Logistic / LPM: `y ~ X_hat + M + W + covars` -> NDE (coef on X_hat), beta_M (coef on M).

`NIE = alpha_M * beta_M`. Weak-instrument gate (partial F for G) applies to the OLS first stage. Unlike `fit_iv2sls_mediation2_bin`, this estimator does not instrument the mediator and is NOT point-identified under M-O confounding — it is included for parity with the continuous `fit_iv2sls_mediation`.

Usage

```
fit_iv2sls_mediation_bin(
  y,
  X,
  M,
  g,
  w,
  covars = NULL,
  min_f = 10,
  effect_scale = c("logor", "riskdiff")
)
```

Arguments

`y`	Numeric 0/1 outcome vector (length n).
`X`	Numeric exposure vector (length n).
`M`	Numeric mediator vector (length n).
`g`	Numeric instrument for X (length n).
`w`	Numeric NC vector (length n) or matrix (n x q).
`covars`	Optional data frame of covariates (n rows).
`min_f`	Minimum partial F for the excluded instrument. Default 10.
`effect_scale`	Character: `"logor"` or `"riskdiff"`.

Value

Named list (same fields as `fit_unadj_mediation_bin`). Returns all-NA if the first-stage partial F for G is below `min_f`.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 500, outcome_type = "binary",
    mo_confounding = 0.8, seed = 1)
fit_iv2sls_mediation_bin(dat$y_bin, dat$X, dat$M, dat$G[, 1], dat$W[, 1])
```

`fit_iv2sls_mediation_surv` *IV2SLS survival mediation estimator: single-instrument 2SPS with Cox / RMST*

Description

Two-stage predictor substitution with a single instrument (G for X, no mediator instrument):

1. OLS: `X ~ g + W + covars` -> X_hat (purge U1 from X).
2. OLS: `M ~ X_hat + covars` -> alpha_M.
3. Cox / RMST: `Surv(t,e) ~ X_hat + M + W + covars` -> NDE (coef on X_hat), beta_M (coef on M).

`NIE = alpha_M * beta_M`. Weak-instrument gate (partial F for G) applies to the OLS first stage. Unlike `fit_iv2sls_mediation2_surv`, this estimator does not instrument the mediator and is NOT point-identified under M-O confounding — it is included for parity with the continuous `fit_iv2sls_mediation`.

Usage

```
fit_iv2sls_mediation_surv(
  time,
  event,
  X,
  M,
  g,
  w,
  covars = NULL,
  min_f = 10,
  effect_scale = c("loghr", "rmst"),
  tau = NULL
)
```

Arguments

`time`	Numeric follow-up time vector (length n).
`event`	Numeric 0/1 event indicator (length n).
`X`	Numeric exposure vector (length n).
`M`	Numeric mediator vector (length n).
`g`	Numeric instrument for X (length n).
`w`	Numeric NC vector (length n) or matrix (n x q).
`covars`	Optional data frame of covariates (n rows).

min_f	Minimum partial F for the excluded instrument. Default 10.
effect_scale	Character: "loghr" or "rmst".
tau	RMST horizon (rmst only).

Value

Named list (same fields as fit_unadj_mediation_surv).

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 200, outcome_type = "survival", seed = 1)
fit_iv2sls_mediation_surv(dat$surv_time, dat$surv_event, dat$X, dat$M,
    dat$G[, 1], dat$W[, 1])
```

---

fit_iv2sls_surv *IV2SLS survival estimator: two-stage predictor substitution with Cox / RMST*

Description

Two-stage predictor substitution (2SPS): the first stage regresses X on the instrument G (plus W and covariates) via OLS, producing fitted $\hat{X}$; the second stage regresses the survival outcome on $\hat{X}$ (plus W and covariates) via Cox (log-HR) or RMST pseudo-observation OLS.

Usage

```
fit_iv2sls_surv(
  time,
  event,
  X,
  g,
  w,
  covars = NULL,
  min_f = 10,
  effect_scale = c("loghr", "rmst"),
  tau = NULL
)
```

Arguments

time	Numeric follow-up time vector (length n).
event	Numeric 0/1 event indicator (length n).
X	Numeric exposure vector (length n).
g	Numeric instrument vector (length n).
w	Numeric NC vector (length n) or matrix (n x q).
covars	Optional data frame of covariates (n rows).
min_f	Minimum partial F for the excluded instrument. Default 10.
effect_scale	Character: "loghr" or "rmst".
tau	RMST horizon (rmst only).

Details

A weak-instrument check (partial F for the excluded instrument G, Stock & Yogo 2005) is applied to the OLS first stage. If the partial F is below `min_f`, the function returns NA.

Value

Named list: `beta`, `se`, `pvalue`.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 200, outcome_type = "survival", seed = 1)
fit_iv2sls_surv(dat$surv_time, dat$surv_event, dat$X, dat$G[, 1], dat$W[, 1])
```

`fit_pgc` *PGC estimator: Proxy G-Component Correction (matrix bridge)*

Description

A three-step bridge-function estimator:

1. Residualises X on G to isolate the U-driven component X_resid.
2. Regresses X_resid on the FULL W matrix to construct W_hat, a proxy for unmeasured confounding. This step requires `ncol(W) >= k` (the proximal completeness condition): if W has fewer valid columns than confounders, the bridge cannot span the confounder subspace and the estimator is under-identified.
3. Fits Y ~ X + W_hat to absorb confounding bias.

Usage

```
fit_pgc(y, X, g, W, covars = NULL)
```

Arguments

`y`	Numeric outcome vector (length n).
`X`	Numeric exposure vector (length n).
`g`	Numeric instrument vector (length n).
`W`	Numeric negative-control matrix (n x q) or vector (length n). If a matrix, the bridge uses all q columns. Pass only validity-screened columns for the completeness condition to be meaningful.
`covars`	Optional data frame of additional covariates (n rows).

Details

Unlike the scalar version (`fit_pgc_scalar`), which collapses W to `rowMeans(W)` and is algebraically equivalent to IV/2SLS, the matrix bridge preserves the dimensional structure of W and is the estimator for which the completeness condition is binding.

Value

Named list: `beta`, `se`, `pvalue`.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 200, seed = 1)
fit_pgc(dat$Y[, 1], dat$X, dat$G[, 1], dat$W)
```

`fit_pgc_bin` *PGC binary estimator: proxy G-component correction with logistic / LPM*

Description

Three-step bridge-function estimator with a binary outcome stage:

1. Residualise X on G -> X_resid (OLS).
2. Bridge X_resid on the FULL W matrix -> W_hat (OLS).
3. Regress the binary outcome on X + W_hat via logistic regression (log-OR) or a linear probability model (risk difference).

Usage

```
fit_pgc_bin(y, X, g, W, covars = NULL, effect_scale = c("logor", "riskdiff"))
```

Arguments

`y`	Numeric 0/1 outcome vector (length n).
`X`	Numeric exposure vector (length n).
`g`	Numeric instrument vector (length n).
`W`	Numeric NC matrix (n x q) or vector (length n).
`covars`	Optional data frame of covariates (n rows).
`effect_scale`	Character: `"logor"` or `"riskdiff"`.

Value

Named list: `beta`, `se`, `pvalue`.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 200, outcome_type = "binary", seed = 1)
fit_pgc_bin(dat$y_bin, dat$X, dat$G[, 1], dat$W)
```

`fit_pgc_mediation` *PGC mediation estimator: bridge-function-adjusted natural effects (matrix bridge)*

Description

Extends `fit_pgc` (matrix bridge) to the mediation setting by constructing a confounding proxy $\hat{W}$ from the full W matrix and including it in both the mediator and outcome regressions.

Usage

```
fit_pgc_mediation(y, X, M, g, W, covars = NULL)
```

Arguments

`y`	Numeric outcome vector (length n).
`X`	Numeric exposure vector (length n).
`M`	Numeric mediator vector (length n).
`g`	Numeric instrument vector (length n).
`W`	Numeric negative-control matrix (n x q) or vector (length n). If a matrix, the bridge uses all q columns.
`covars`	Optional data frame of additional covariates (n rows).

Details

Steps:

1. Residualise X on G -> X_resid (U-driven residual).
2. Bridge X_resid on the FULL W matrix -> W_hat (proxy for U).
3. `M ~ X + W_hat` -> alpha_M (adjusted for confounding proxy).
4. `Y ~ X + M + W_hat` -> NDE = beta_X, beta_M (adjusted).

The matrix bridge requires `ncol(W) >= k` (proximal completeness) for the bridge to span the confounder subspace.

Value

Named list: `NDE`, `NDE_se`, `NDE_p`, `NIE`, `NIE_se`, `NIE_p`.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 200, mo_confounding = 0.8, seed = 1)
fit_pgc_mediation(dat$Y[, 1], dat$X, dat$M, dat$G[, 1], dat$W)
```

`fit_pgc_mediation2` *PGC-2 mediation estimator: two-stage proximal mediation with path-specific bridges*

Description

Extends the proximal-inference approach to the two-linked-DAG mediation setting. Uses **path-specific** negative controls — W1 capturing the X->M confounder conf_XM and W2 capturing the M->Y confounder conf_MY — to purge confounding at both stages via bridge functions.

Usage

```
fit_pgc_mediation2(y, X, M, g, W1, W2, gm = NULL, covars = NULL, min_f = 10)
```

Arguments

`y`	Numeric outcome vector (length n).
`X`	Numeric exposure vector (length n).
`M`	Numeric mediator vector (length n).
`g`	Numeric instrument for X (length n).
`W1`	Numeric negative-control matrix (n x q) or vector for the X->M path (captures conf_XM). If a matrix, the bridge uses all q columns.
`W2`	Numeric negative-control matrix (n x q) or vector for the M->Y path (captures conf_MY). If a matrix, the bridge uses all q columns.
`gm`	Optional numeric mediator instrument vector (length n). When `NULL` (default), Stage 2 uses pure NC identification. When supplied, Gm helps isolate conf_MY before bridging — robust to Gm-U correlation.
`covars`	Optional data frame of additional covariates (n rows).
`min_f`	Minimum acceptable partial F-statistic for the excluded instrument G1. Default 10.

Details

When `gm` is `NULL` (no mediator instrument), Stage 2 residualises M on the cleaned exposure X_hat to isolate the conf_MY-driven component, then bridges on W2. This is pure negative-control identification at both stages — no mediator instrument is required.

When `gm` is supplied (mediator instrument present but its exogeneity may be in doubt), Stage 2 uses Gm to help isolate conf_MY's effect on M before bridging on W2. The bridge W_hat_M does the confounding removal, so the estimator is robust to Gm being correlated with conf_MY — the residual bias from Gm-U correlation is smaller than IV2SLS2's.

Strategy (three stages):

1. **Bridge for X** (purge conf_XM from X): `X_resid = residuals(X ~ G1 + C)`; `W_hat_X = bridge(X_resid ~ W1)` (fitted values, proxy for conf_XM); `X_hat = fitted(X ~ G1 + W_hat_X + C)`. Weak-IV check: partial F for G1 >= `min_f`.

2. **Bridge for M** (purge conf_MY from M): `M_resid = residuals(M ~ X_hat + C)` (`gm = NULL`), or `M_resid = residuals(M ~ Gm + C)` (`gm` supplied); `W_hat_M = bridge(M_resid ~ W2)` (fitted values, proxy for conf_MY); `M_hat = fitted(M ~ X_hat + W_hat_M + C)` (`gm = NULL`), or `M_hat = fitted(M ~ X_hat + Gm + W_hat_M + C)` (`gm` supplied); `alpha_M = coefficient on X_hat`.
3. **Outcome**: `Y ~ X_hat + M_hat + W_hat_X + W_hat_M + C`; `NDE = coefficient on X_hat`, `beta_M = coefficient on M_hat`.

NIE = alpha_M * beta_M (delta-method SE).

The key advantage over `fit_iv2sls_mediation2`: PGC-2 does not require instrument exogeneity. When the mediator instrument Gm is correlated with the confounder conf_MY (rho_G2 > 0), IV2SLS2 is biased but PGC-2's bridge absorbs conf_MY regardless of the instrument violation. The tipping-point simulation maps where PGC-2 bias crosses below IV2SLS2 bias.

Value

Named list: `NDE`, `NDE_se`, `NDE_p`, `NIE`, `NIE_se`, `NIE_p`. Returns all-`NA` if the first-stage partial F for G1 is below `min_f`.

References

Miao, W., Geng, Z., & Tchetgen Tchetgen, E. (2018). Identifying causal effects with proxy variables of an unmeasured confounder. *Biometrika*, 105(4), 987-993.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 500, mo_confounding = 0.8,
    rho_G2 = 0.3, lambda_XM = c(1, 0), lambda_MY = c(0, 1),
    omega_1 = 0.7, omega_2 = 0.7, seed = 1)
# Without mediator instrument (pure NC identification)
fit_pgc_mediation2(dat$Y[, 1], dat$X, dat$M, dat$G1, dat$W1, dat$W2)
# With (possibly imperfect) mediator instrument
fit_pgc_mediation2(dat$Y[, 1], dat$X, dat$M, dat$G1, dat$W1, dat$W2,
    gm = dat$Gm)
```

---

fit_pgc_mediation2_bin

PGC2 / PGC2Gm binary mediation estimator: path-specific bridges with logistic / LPM

Description

Two-stage proximal mediation with path-specific negative controls and a binary outcome stage. Stages 1-2 (X bridge on W1, M bridge on W2, X_hat / M_hat construction) are identical to `fit_pgc_mediation2` and remain OLS. Only stage 3 switches to logistic regression (log-OR) or a linear probability model (risk difference): `y ~ X_hat + M_hat + W_hat_X + W_hat_M + covars`.

Usage

```
fit_pgc_mediation2_bin(
  y,
  X,
  M,
  g,
  W1,
  W2,
  gm = NULL,
  covars = NULL,
  min_f = 10,
  effect_scale = c("logor", "riskdiff")
)
```

Arguments

y	Numeric 0/1 outcome vector (length n).
X	Numeric exposure vector (length n).
M	Numeric mediator vector (length n).
g	Numeric instrument for X (length n).
W1	Numeric NC matrix (n x q) or vector for the X->M path.
W2	Numeric NC matrix (n x q) or vector for the M->Y path.
gm	Optional numeric mediator instrument (length n).
covars	Optional data frame of covariates (n rows).
min_f	Minimum partial F for G1. Default 10.
effect_scale	Character: "logor" or "riskdiff".

Details

When gm = NULL (PGC2), stage 2 uses pure NC identification. When gm is supplied (PGC2Gm), the mediator instrument helps isolate conf_MY before bridging W2.

Value

Named list (same fields as fit_unadj_mediation_bin). Returns all-NA if the first-stage partial F for G1 is below min_f.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 500, outcome_type = "binary",
    mo_confounding = 0.8, rho_G2 = 0.3,
    lambda_XM = c(1, 0), lambda_MY = c(0, 1), seed = 1)
fit_pgc_mediation2_bin(dat$y_bin, dat$X, dat$M,
    dat$G[, 1], dat$W1, dat$W2, gm = dat$Gm)
```

`fit_pgc_mediation2_surv` *PGC2 / PGC2Gm survival mediation estimator: path-specific bridges with Cox / RMST*

Description

Two-stage proximal mediation with path-specific negative controls and a survival outcome stage. Stages 1-2 (X bridge on W1, M bridge on W2, X_hat / M_hat construction) are identical to `fit_pgc_mediation2` and remain OLS. Only stage 3 switches to Cox (log-HR) or RMST pseudo-observation OLS: `Surv(t,e) ~ X_hat + M_hat + W_hat_X + W_hat_M + covars`.

Usage

```
fit_pgc_mediation2_surv(
  time,
  event,
  X,
  M,
  g,
  W1,
  W2,
  gm = NULL,
  covars = NULL,
  min_f = 10,
  effect_scale = c("loghr", "rmst"),
  tau = NULL
)
```

Arguments

`time`	Numeric follow-up time vector (length n).
`event`	Numeric 0/1 event indicator (length n).
`X`	Numeric exposure vector (length n).
`M`	Numeric mediator vector (length n).
`g`	Numeric instrument for X (length n).
`W1`	Numeric NC matrix (n x q) or vector for the X->M path.
`W2`	Numeric NC matrix (n x q) or vector for the M->Y path.
`gm`	Optional numeric mediator instrument (length n).
`covars`	Optional data frame of covariates (n rows).
`min_f`	Minimum partial F for G1. Default 10.
`effect_scale`	Character: `"loghr"` or `"rmst"`.
`tau`	RMST horizon (rmst only).

Details

When `gm = NULL` (PGC2), stage 2 uses pure NC identification. When `gm` is supplied (PGC2Gm), the mediator instrument helps isolate conf_MY before bridging W2.

Value

Named list (same fields as fit_unadj_mediation_surv). Returns all-NA if the first-stage partial F for G1 is below min_f.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 500, outcome_type = "survival",
    mo_confounding = 0.8, rho_G2 = 0.3,
    lambda_XM = c(1, 0), lambda_MY = c(0, 1), seed = 1)
fit_pgc_mediation2_surv(dat$surv_time, dat$surv_event, dat$X, dat$M,
    dat$G[, 1], dat$W1, dat$W2, gm = dat$Gm)
```

---

fit_pgc_mediation_bin *PGC binary mediation estimator: single-panel bridge with logistic / LPM*

Description

Bridge-function-adjusted natural direct and indirect effects with a binary outcome stage, using a single negative-control panel W for both the X->M and M->Y confounding paths. This is the binary analogue of fit_pgc_mediation (continuous outcome):

1. OLS: residualise X on G -> X_resid.
2. OLS: bridge X_resid on the FULL W matrix -> W_hat (proxy for U).
3. OLS: M ~ X + W_hat + covars -> alpha_M.
4. Logistic / LPM: y ~ X + M + W_hat + covars -> NDE (coef on X), beta_M (coef on M).

NIE = alpha_M * beta_M. Unlike fit_pgc_mediation2_bin, which uses path-specific W1/W2 bridges, this estimator uses a single combined W panel and is appropriate when separate conf_XM / conf_MY confounders are not assumed.

Usage

```
fit_pgc_mediation_bin(
  y,
  X,
  M,
  g,
  W,
  covars = NULL,
  min_f = 10,
  effect_scale = c("logor", "riskdiff")
)
```

Arguments

y	Numeric 0/1 outcome vector (length n).
X	Numeric exposure vector (length n).
M	Numeric mediator vector (length n).

g	Numeric instrument for X (length n).
W	Numeric NC matrix (n x q) or vector (length n).
covars	Optional data frame of covariates (n rows).
min_f	Minimum partial F for G. Default 10.
effect_scale	Character: "logor" or "riskdiff".

Value

Named list (same fields as fit_unadj_mediation_bin).

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 500, outcome_type = "binary",
    mo_confounding = 0.8, seed = 1)
fit_pgc_mediation_bin(dat$y_bin, dat$X, dat$M, dat$G[, 1], dat$W)
```

---

fit_pgc_mediation_surv

PGC survival mediation estimator: single-panel bridge with Cox / RMST

Description

Bridge-function-adjusted natural direct and indirect effects with a survival outcome stage, using a single negative-control panel W for both the X->M and M->Y confounding paths. This is the survival analogue of fit_pgc_mediation (continuous outcome):

1. OLS: residualise X on G -> X_resid.
2. OLS: bridge X_resid on the FULL W matrix -> W_hat (proxy for U).
3. OLS: M ~ X + W_hat + covars -> alpha_M.
4. Cox / RMST: Surv(t,e) ~ X + M + W_hat + covars -> NDE (coef on X), beta_M (coef on M).

NIE = alpha_M * beta_M. Unlike fit_pgc_mediation2_surv, which uses path-specific W1/W2 bridges, this estimator uses a single combined W panel and is appropriate when separate conf_XM / conf_MY confounders are not assumed.

Usage

```
fit_pgc_mediation_surv(
  time,
  event,
  X,
  M,
  g,
  W,
  covars = NULL,
  min_f = 10,
  effect_scale = c("loghr", "rmst"),
  tau = NULL
)
```

Arguments

time	Numeric follow-up time vector (length n).
event	Numeric 0/1 event indicator (length n).
X	Numeric exposure vector (length n).
M	Numeric mediator vector (length n).
g	Numeric instrument for X (length n).
W	Numeric NC matrix (n x q) or vector (length n).
covars	Optional data frame of covariates (n rows).
min_f	Minimum partial F for G. Default 10.
effect_scale	Character: "loghr" or "rmst".
tau	RMST horizon (rmst only).

Value

Named list (same fields as fit_unadj_mediation_surv). Returns all-NA if the first-stage partial F for G is below min_f.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 200, outcome_type = "survival", seed = 1)
fit_pgc_mediation_surv(dat$surv_time, dat$surv_event, dat$X, dat$M,
    dat$G[, 1], dat$W)
```

---

fit_pgc_scalar *PGC estimator: Proxy G-Component Correction (scalar bridge)*

Description

The original ICONIC PGC implementation, which summarises the negative-control panel as a scalar (rowMeans(W)) before bridging. This version is numerically stable and works in small samples, but the scalar bridge produces $\hat{W}$ proportional to the G-residualised exposure by construction, making the estimator algebraically equivalent to IV/2SLS when the instrument is valid. The proximal completeness condition (dim(W_valid) >= k) is NOT binding for this estimator.

Usage

```
fit_pgc_scalar(y, X, g, w, covars = NULL)
```

Arguments

y	Numeric outcome vector (length n).
X	Numeric exposure vector (length n).
g	Numeric instrument vector (length n).
w	Numeric negative-control vector (length n). Pass rowMeans(W_matrix) for stability.
covars	Optional data frame of additional covariates (n rows).

Details

Use `fit_pgc` (matrix bridge) when the completeness condition is of interest. Use this function as a stable fallback when `n` is small relative to the number of valid controls, or as an IV-equivalent benchmark.

Value

Named list: `beta`, `se`, `pvalue`.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 200, seed = 1)
fit_pgc_scalar(dat$Y[, 1], dat$X, dat$G[, 1], rowMeans(dat$W))
```

`fit_pgc_scalar_mediation` *PGC mediation estimator: bridge-function-adjusted natural effects (scalar bridge)*

Description

The original ICONIC PGC mediation implementation, which summarises the negative-control panel as a scalar (`rowMeans(W)`) before bridging. Like `fit_pgc_scalar`, this version is algebraically equivalent to the IV2SLS mediation estimator when the instrument is valid. Use `fit_pgc_mediation` (matrix bridge) when the completeness condition is of interest.

Usage

```
fit_pgc_scalar_mediation(y, X, M, g, w, covars = NULL)
```

Arguments

`y`	Numeric outcome vector (length n).
`X`	Numeric exposure vector (length n).
`M`	Numeric mediator vector (length n).
`g`	Numeric instrument vector (length n).
`w`	Numeric negative-control vector (length n). Pass `rowMeans(W_matrix)` for stability.
`covars`	Optional data frame of additional covariates (n rows).

Value

Named list: `NDE`, `NDE_se`, `NDE_p`, `NIE`, `NIE_se`, `NIE_p`.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 200, mo_confounding = 0.8, seed = 1)
fit_pgc_scalar_mediation(dat$Y[, 1], dat$X, dat$M, dat$G[, 1], rowMeans(dat$W))
```

`fit_pgc_surv` *PGC survival estimator: proxy G-component correction with Cox / RMST*

Description

Three-step bridge-function estimator with a survival outcome stage:

1. Residualise X on G -> X_resid (OLS).
2. Bridge X_resid on the FULL W matrix -> W_hat (OLS).
3. Regress the survival outcome on X + W_hat via Cox (log-HR) or RMST pseudo-observation OLS.

Usage

```
fit_pgc_surv(
  time,
  event,
  X,
  g,
  W,
  covars = NULL,
  effect_scale = c("loghr", "rmst"),
  tau = NULL
)
```

Arguments

`time`	Numeric follow-up time vector (length n).
`event`	Numeric 0/1 event indicator (length n).
`X`	Numeric exposure vector (length n).
`g`	Numeric instrument vector (length n).
`W`	Numeric NC matrix (n x q) or vector (length n).
`covars`	Optional data frame of covariates (n rows).
`effect_scale`	Character: `"loghr"` or `"rmst"`.
`tau`	RMST horizon (rmst only).

Value

Named list: `beta`, `se`, `pvalue`.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 200, outcome_type = "survival", seed = 1)
fit_pgc_surv(dat$surv_time, dat$surv_event, dat$X, dat$G[, 1], dat$W)
```

fit_unadj_bin *UNADJ binary estimator: unadjusted logistic / linear-probability regression*

Description

Regresses the binary outcome on the exposure X (plus covariates) with no instrument or negative-control adjustment. Bias reference.

Usage

```
fit_unadj_bin(y, X, covars = NULL, effect_scale = c("logor", "riskdiff"))
```

Arguments

y	Numeric 0/1 outcome vector (length n).
X	Numeric exposure vector (length n).
covars	Optional data frame of covariates (n rows).
effect_scale	Character: "logor" (logistic) or "riskdiff" (linear probability model). Default "logor".

Details

When effect_scale = "logor" (default), fits a logistic regression and returns the conditional log-odds ratio for X. When effect_scale = "riskdiff", fits a linear probability model (OLS on the 0/1 outcome), returning a risk difference — a collapsible, linear alternative to the non-collapsible odds ratio.

Value

Named list: beta, se, pvalue.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 200, outcome_type = "binary", seed = 1)
fit_unadj_bin(dat$y_bin, dat$X)
fit_unadj_bin(dat$y_bin, dat$X, effect_scale = "riskdiff")
```

fit_unadj_mediation *UNADJ mediation estimator: naive Baron-Kenny style*

Description

Stage 1: M ~ X (estimate alpha_M). Stage 2: Y ~ X + M (estimate NDE = beta_X, beta_M). NIE = alpha_M * beta_M.

Usage

```
fit_unadj_mediation(y, X, M, covars = NULL)
```

Arguments

y	Numeric outcome vector (length n).
X	Numeric exposure vector (length n).
M	Numeric mediator vector (length n).
covars	Optional data frame of additional covariates (n rows).

Details

Does not adjust for unmeasured confounding. Provided as a bias reference floor, analogous to UNADJ in the total-effect setting.

Value

Named list: NDE, NDE_se, NDE_p, NIE, NIE_se, NIE_p.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 200, mo_confounding = 0.8, seed = 1)
fit_unadj_mediation(dat$Y[, 1], dat$X, dat$M)
```

---

fit_unadj_mediation_bin

UNADJ binary mediation estimator: naive Baron-Kenny with logistic / LPM

Description

Stage 1 (OLS): M ~ X -> alpha_M. Stage 2 (logistic / LPM): binary outcome ~ X + M -> NDE (coef on X), beta_M (coef on M). NIE = alpha_M * beta_M. No confounding adjustment; bias reference.

Usage

```
fit_unadj_mediation_bin(
  y,
  X,
  M,
  covars = NULL,
  effect_scale = c("logor", "riskdiff")
)
```

Arguments

y	Numeric 0/1 outcome vector (length n).
X	Numeric exposure vector (length n).
M	Numeric mediator vector (length n).
covars	Optional data frame of covariates (n rows).
effect_scale	Character: "logor" or "riskdiff".

Value

Named list: NDE, NDE_se, NDE_p, NIE, NIE_se, NIE_p, alpha_M, alpha_se, beta_M, beta_M_se.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 200, outcome_type = "binary",
    mo_confounding = 0.8, seed = 1)
fit_unadj_mediation_bin(dat$y_bin, dat$X, dat$M)
```

---

fit_unadj_mediation_surv

UNADJ survival mediation estimator: naive Baron-Kenny with Cox / RMST

Description

Stage 1 (OLS): M ~ X -> alpha_M. Stage 2 (Cox / RMST): survival outcome ~ X + M -> NDE (coef on X), beta_M (coef on M). NIE = alpha_M * beta_M. No confounding adjustment; bias reference.

Usage

```
fit_unadj_mediation_surv(
  time,
  event,
  X,
  M,
  covars = NULL,
  effect_scale = c("loghr", "rmst"),
  tau = NULL
)
```

Arguments

time	Numeric follow-up time vector (length n).
event	Numeric 0/1 event indicator (length n).
X	Numeric exposure vector (length n).
M	Numeric mediator vector (length n).
covars	Optional data frame of covariates (n rows).
effect_scale	Character: "loghr" or "rmst".
tau	RMST horizon (rmst only).

Value

Named list: NDE, NDE_se, NDE_p, NIE, NIE_se, NIE_p, alpha_M, alpha_se, beta_M, beta_M_se.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 200, outcome_type = "survival",
    mo_confounding = 0.8, seed = 1)
fit_unadj_mediation_surv(dat$surv_time, dat$surv_event, dat$X, dat$M)
```

`fit_unadj_surv` *UNADJ survival estimator: unadjusted Cox / RMST regression*

Description

Regresses the survival outcome on the exposure X (plus covariates) with no instrument or negative-control adjustment. Bias reference.

Usage

```
fit_unadj_surv(
  time,
  event,
  X,
  covars = NULL,
  effect_scale = c("loghr", "rmst"),
  tau = NULL
)
```

Arguments

`time`	Numeric follow-up time vector (length n).
`event`	Numeric 0/1 event indicator (length n).
`X`	Numeric exposure vector (length n).
`covars`	Optional data frame of covariates (n rows).
`effect_scale`	Character: `"loghr"` (Cox) or `"rmst"` (pseudo-observation OLS). Default `"loghr"`.
`tau`	RMST restriction time horizon. Default `NULL` (90th percentile of follow-up). Used only when `effect_scale = "rmst"`.

Details

When `effect_scale = "loghr"` (default), fits a Cox proportional-hazards model and returns the log-hazard ratio for X. When `effect_scale = "rmst"`, regresses leave-one-out RMST pseudo-observations (Graw et al. 2009) on X via OLS, returning an effect on the restricted-mean-survival-time (time) scale — a collapsible, linear alternative to the non-collapsible hazard ratio.

Value

Named list: `beta`, `se`, `pvalue`.

Examples

```
set.seed(1)
dat <- generate_toy_data(n = 200, outcome_type = "survival", seed = 1)
fit_unadj_surv(dat$surv_time, dat$surv_event, dat$X)
fit_unadj_surv(dat$surv_time, dat$surv_event, dat$X, effect_scale = "rmst")
```

`gan_mediation_sensitivity` *Benchmark mediation estimators across confounding scenarios*

Description

For each scenario in the grid, generates `n_iter` synthetic datasets with mediator-outcome confounding (via `mo_confounding`), runs every mediation estimator, and summarises NDE/NIE bias / RMSE / Type I error. This is the mediation analogue of `gan_sensitivity()`.

Usage

```
gan_mediation_sensitivity(
  trained_gan = NULL,
  conf_grid = c(0.2, 0.5, 0.8),
  coverage_grid = c(0.3, 0.7, 1),
  k_grid = 1,
  mo_confounding = 0.8,
  phi = 0,
  rho_G1 = 0,
  rho_G2 = 0,
  rho_pop = 0,
  lambda_XM = NULL,
  lambda_MY = NULL,
  omega_1 = NULL,
  omega_2 = NULL,
  nc_model = "proxy",
  n_iter = 50,
  n_samples = 500,
  n_features = 20,
  beta_X = 0.1,
  alpha_M = 0.5,
  beta_M = 0.3,
  base_seed = 750,
  n_cores = 1,
  outcome_type = c("continuous", "survival", "binary"),
  effect_scale = c("loghr", "rmst", "logor", "riskdiff"),
  surv_h0 = 0.1,
  surv_event_frac = 0.6,
  surv_censor_rate = NULL,
  bin_prev = 0.5
)
```

Arguments

`trained_gan`	An `iconic_gan` (or `NULL` to use default texture).
`conf_grid`	Confounding-strength values to sweep. Default `c(0.2, 0.5, 0.8)`.
`coverage_grid`	Negative-control coverage values in `[0,1]`. Default `c(0.3, 0.7, 1)`.
`k_grid`	Numbers of latent confounders to sweep. Default `1`.
`mo_confounding`	Strength of U1 -> M (mediator-outcome confounding). Default 0.80.

phi	Strength of the mediator instrument Gm -> M. 0 = no mediator instrument (five estimators). > 0 = generates Gm and includes the 2-stage MR estimator (IV2SLS2). Default 0.
rho_G1	Correlation of G1 with conf_XM. Default 0.
rho_G2	Correlation of G2 with conf_MY. Default 0.
rho_pop	Shared population structure. Default 0.
lambda_XM	Optional per-path confounder loading vector (X->M path).
lambda_MY	Optional per-path confounder loading vector (M->Y path).
omega_1	Coverage of conf_XM by W1. NULL = use coverage.
omega_2	Coverage of conf_MY by W2. NULL = use coverage.
nc_model	Negative-control model (function or name). Default "proxy".
n_iter	Replicates per scenario. Default 50.
n_samples	Samples per replicate. Default 500.
n_features	Features per replicate. Default 20.
beta_X, alpha_M, beta_M	Causal paths (ground truth). Defaults 0.10 / 0.50 / 0.30.
base_seed	Base RNG seed. Default 750.
n_cores	Parallel workers across replicates. Default 1.
outcome_type	"continuous" (default), "survival", or "binary". When survival, the DGP generates time-to-event outcomes and estimation uses the Cox / RMST survival mediation drivers via iconic_estimate(). When binary, the DGP generates a 0/1 outcome and estimation uses the logistic / linear-probability-model binary mediation drivers via iconic_estimate().
effect_scale	"loghr" (default), "rmst", "logor", or "riskdiff". "loghr"/"rmst" apply to survival outcomes; "logor"/"riskdiff" apply to binary outcomes (an inert default is remapped with a message).
surv_h0	Baseline hazard for survival DGP. See run_single_iteration().
surv_event_frac	Target event fraction for survival DGP.
surv_censor_rate	Censoring rate for survival DGP.
bin_prev	Target prevalence for the binary DGP. Default 0.5.

Details

When phi > 0, a mediator-specific genetic instrument (Gm) is generated and the 2-stage MR estimator (IV2SLS2) is included in the results, enabling point identification of NDE/NIE under M-O confounding.

Value

A list with summary (one row per scenario x method, with conf_strength, coverage, k, mo_confounding, phi, true_NDE, true_NIE and NDE/NIE bias/RMSE/Type I columns) and grid.

Examples

```
sens <- gan_mediation_sensitivity(NULL, conf_grid = 0.8,
    coverage_grid = 0.7, mo_confounding = 0.8,
    n_iter = 2, n_samples = 100, n_features = 5)
head(sens$summary)
```

`gan_pleiotropy_sensitivity` *Benchmark estimators across pleiotropy and confounding scenarios*

Description

For each cell in the grid (pleiotropy strength x confounding strength), generates `n_iter` synthetic datasets with `run_single_iteration()`, runs every estimator, and summarises bias / RMSE / power. Two arms are run per cell:

Usage

```
gan_pleiotropy_sensitivity(
  trained_gan = NULL,
  pleio_grid = c(0, 0.05, 0.1),
  conf_grid = c(0.2, 0.5, 0.8),
  tau = 0.25,
  nc_model = "proxy",
  n_iter = 50,
  n_samples = 500,
  n_features = 10,
  coverage = 0.7,
  k = 1,
  base_seed = 900,
  n_cores = 1
)
```

Arguments

`trained_gan`	An `iconic_gan` (or `NULL` to use default texture).
`pleio_grid`	Horizontal-pleiotropy strengths (direct G -> Y coefficients) to sweep. Default `c(0, 0.05, 0.10)`.
`conf_grid`	Confounding-strength values. Default `c(0.2, 0.5, 0.8)`.
`tau`	True total effect for the alternative arm. Default 0.25.
`nc_model`	Negative-control model (function or name). Default `"proxy"`.
`n_iter`	Replicates per cell per arm. Default 50.
`n_samples`	Samples per replicate. Default 500.
`n_features`	Features per replicate. Default 10.
`coverage`	Negative-control coverage. Default 0.7.
`k`	Number of latent confounders. Default 1.
`base_seed`	Base RNG seed. Default 900.
`n_cores`	Parallel workers across replicates. Default 1.

Details

- **alternative** (`effect_size = tau`): the true total effect is `tau`, so `power` is the empirical power to detect it.
- **null** (`effect_size = 0`): the true total effect is 0, so `power` is the empirical Type I error rate.

The `pleio` parameter adds a direct `G -> Y` path of the requested strength, violating the exclusion restriction. IV/2SLS is consistent only when `pleio = 0`; any `pleio > 0` introduces bias that does not shrink with sample size.

Value

A list with `summary` (one row per cell x arm x method, with `pleio`, `conf_strength`, `arm`, `true_total`, and the columns from `summarise_results()`) and `grid`.

Examples

```
sens <- gan_pleiotropy_sensitivity(NULL,
    pleio_grid = c(0, 0.10), conf_grid = 0.8,
    n_iter = 2, n_samples = 100, n_features = 5)
head(sens$summary)
```

---

gan_sensitivity *Benchmark estimators across confounding scenarios on synthetic data*

Description

For each scenario in the grid, generates `n_iter` synthetic datasets with `run_single_iteration()` (using the trained generator for texture), runs every estimator, and summarises bias / RMSE / power. This is the multi-confounder, negative-control-aware generalisation of the package's parameter sweeps.

Usage

```
gan_sensitivity(
  trained_gan = NULL,
  conf_grid = c(0.2, 0.5, 0.8),
  coverage_grid = c(0.3, 0.7, 1),
  k_grid = 1,
  nc_model = "proxy",
  n_iter = 50,
  n_samples = 500,
  n_features = 20,
  beta_X = 0.1,
  alpha_M = 0.5,
  beta_M = 0.3,
  effect_size = NULL,
  base_seed = 700,
  n_cores = 1,
  outcome_type = c("continuous", "survival", "binary"),
  effect_scale = c("loghr", "rmst", "logor", "riskdiff"),
  surv_h0 = 0.1,
```

```
  surv_event_frac = 0.6,
  surv_censor_rate = NULL,
  bin_prev = 0.5
)
```

Arguments

trained_gan	An iconic_gan (or NULL to use default texture).
conf_grid	Confounding-strength values to sweep. Default c(0.2, 0.5, 0.8).
coverage_grid	Negative-control coverage values in [0,1]. Default c(0.3, 0.7, 1).
k_grid	Numbers of latent confounders to sweep. Default 1.
nc_model	Negative-control model (function or name). Default "proxy".
n_iter	Replicates per scenario. Default 50.
n_samples	Samples per replicate. Default 500.
n_features	Features per replicate. Default 20.
beta_X, alpha_M, beta_M	Causal paths (ground truth). Defaults 0.10 / 0.50 / 0.30.
effect_size	Optional pure-direct total effect override (see run_single_iteration()).
base_seed	Base RNG seed. Default 700.
n_cores	Parallel workers across replicates. Default 1.
outcome_type	"continuous" (default), "survival", or "binary". When survival, the DGP generates time-to-event outcomes and estimation uses the Cox / RMST survival drivers via iconic_estimate(). When binary, the DGP generates a 0/1 outcome and estimation uses the logistic / linear-probability-model binary drivers via iconic_estimate().
effect_scale	"loghr" (default), "rmst", "logor", or "riskdiff". "loghr"/"rmst" apply to survival outcomes; "logor"/"riskdiff" apply to binary outcomes (an inert default is remapped with a message).
surv_h0	Baseline hazard for the survival DGP. Default 0.1.
surv_event_frac	Target fraction of observed events. Default 0.6.
surv_censor_rate	Explicit censoring rate. Default NULL.
bin_prev	Target prevalence for the binary DGP. Default 0.5.

Value

A list with summary (one row per scenario x method, with conf_strength, coverage, k, true_total and the columns from summarise_results) and grid (the scenario grid).

Examples

```
sens <- gan_sensitivity(NULL, conf_grid = 0.8, coverage_grid = 0.7,
    n_iter = 2, n_samples = 100, n_features = 5)
head(sens$summary)
```

`generate_toy_data` *Generate one synthetic dataset (internal)*

Description

When `mo_confounding > 0`, the unmeasured confounder U1 also affects the mediator M, creating mediator-outcome confounding — the key extension for the mediation simulation. When `mo_confounding = 0` (default), the original DGP is preserved exactly.

Usage

```
generate_toy_data(
  n = 500,
  n_features = 20,
  n_mediators = 1,
  beta_X = 0.1,
  alpha_M = 0.5,
  beta_M = 0.3,
  conf_str = 0.8,
  w_signal = 0.7,
  mo_confounding = 0,
  pleio = 0,
  phi = 0,
  gamma_G = 0.6,
  rho_G1 = 0,
  rho_G2 = 0,
  rho_pop = 0,
  lambda_XM = NULL,
  lambda_MY = NULL,
  omega_1 = NULL,
  omega_2 = NULL,
  feat_cor = 0,
  separate_U = NULL,
  u_strength = NULL,
  w_coverage_profile = NULL,
  outcome_type = c("continuous", "survival", "binary"),
  surv_h0 = 0.1,
  surv_event_frac = 0.6,
  surv_censor_rate = NULL,
  bin_prev = 0.5,
  seed = NULL,
  beta_Z = NULL
)
```

Arguments

`n`	Sample size. Default 500.
`n_features`	Number of outcome and negative-control features. Default 20.

`n_mediators` Number of independent mediators. When > 1, each mediator M_m has its own genetic instrument Gm_m and contributes additively to Y. M and Gm are returned as `n_mediators` x `n` matrices. Default 1 (single mediator, backward compatible).

`beta_X` Direct effect of X on Y (true NDE). Default 0.10.

`alpha_M` Effect of X on mediator M. Default 0.50.

`beta_M` Effect of M on Y (per-mediator NIE = alpha_M * beta_M; total NIE = n_mediators * alpha_M * beta_M). Default 0.30.

`conf_str` Confounding strength delta. Default 0.80.

`w_signal` Proxy quality omega (0 = noise, 1 = perfect U proxy). Default 0.70.

`mo_confounding` Strength of U1 -> M (mediator-outcome confounding). 0 = no M-O confounding (original DGP). Default 0.

`pleio` Strength of a direct G -> Y path (horizontal pleiotropy), violating the exclusion restriction. 0 = no pleiotropy (valid instrument, original DGP). Default 0.

`phi` Strength of the mediator instrument Gm -> M. 0 = no mediator instrument (original DGP, no Gm generated). Default 0. When > 0, a valid instrument for M is generated, enabling point identification of NDE/NIE via `fit_iv2sls_mediation2`.

`gamma_G` Strength of the exposure instrument G -> X. Default 0.6.

`rho_G1` Correlation of G1 with conf_XM. Default 0.

`rho_G2` Correlation of G2 with conf_MY. Default 0.

`rho_pop` Shared population structure inducing G1-G2 correlation . Default 0.

`lambda_XM` Optional per-path loading vector for the X->M confounder composite. NULL (default) = shared loadings.

`lambda_MY` Optional per-path loading vector for the M->Y confounder composite. NULL (default) = shared loadings.

`omega_1` Coverage of conf_XM by W1. NULL = use w_signal.

`omega_2` Coverage of conf_MY by W2. NULL = use w_signal.

`feat_cor` Within-module feature correlation. When > 0, the idiosyncratic noise in the Y and W panels is drawn from a multivariate normal with a block-diagonal correlation matrix modelling co-expression modules. 0 = independent noise (backward compatible). Default 0. Sensitivity sweeps in the ICONIC preprint show estimator performance is largely insensitive to `feat_cor`, so it is retained for completeness rather than as a primary stress axis.

`separate_U` Defunct. Passing a value errors with a message pointing to the replacement per-path loading vectors `lambda_XM` / `lambda_MY`. Retained in the signature only to catch and redirect old calls.

`u_strength` Numeric scalar: scales the single confounder's effect on X, M, and Y. Default NULL → 1 (backward compatible). See `run_single_iteration` for the length-k vector version (multiple confounders).

`w_coverage_profile` A list with optional `w1` and `w2` numeric vectors: per-control coverage of conf_XM / conf_MY. Default NULL → scalar omega applied uniformly (backward compatible).

`outcome_type` `"continuous"` (default), `"survival"`, or `"binary"`. When `"survival"`, the continuous linear predictor Y is converted to `surv_time` and `surv_event` via an exponential PH model. When `"binary"`, the linear predictor is converted to a 0/1 outcome `y_bin` via a logistic (Bernoulli) model; the true effects are then on the conditional log-odds-ratio scale.

surv_h0 Baseline hazard for the survival DGP. Default 0.1.

surv_event_frac
Target fraction of observed events. Default 0.6. The censoring rate is solved internally from this.

surv_censor_rate
Explicit censoring rate. Default NULL → solved from surv_event_frac.

bin_prev Target prevalence (marginal event probability) for the binary DGP. Default 0.5. The logistic intercept is solved internally from this.

seed Optional integer RNG seed for reproducibility.

beta_Z Defunct. Renamed to beta_X; passing a value errors with a message pointing to beta_X. Retained in the signature only to catch and redirect old calls.

Details

When phi > 0, a mediator-specific genetic instrument Gm is generated: $Gm \sim \mathcal{N}(0,1)$, independent of G, U, and all other variables, and the mediator equation becomes $M = \alpha_M X + \delta_{mo} 0.5 U_1 + \phi Gm + \varepsilon_M$. Gm is a valid instrument for M: it moves M but is independent of the confounders and has no direct path to Y. This enables the 2-stage MR mediation estimator (fit_iv2sls_mediation2) to point-identify NDE and NIE even under M-O confounding. When phi = 0 (default), no Gm is generated and the original DGP is preserved exactly.

Value

A named list with elements X, G (n x n_features matrix), Y, W, U1, M, synthetic_data, true_total, true_NDE, true_NIE. When phi > 0, also includes Gm (numeric vector, length n, or n_mediators x n matrix when n_mediators > 1). When any parameter is non-default, also includes G1, G2, W1, W2, conf_XM, conf_MY, and (when rho_pop > 0) P. When n_mediators > 1, M is an n_mediators x n matrix.

imperfect instrument independence

The generates instruments as pure noise, independent of the confounders — the best possible case. adds parameters that violate this independence, modelling realistic genomic structure:

rho_G1 Correlation of the exposure instrument G1 with the X->M confounder conf_XM (instrument exogeneity violation). Default 0.

rho_G2 Correlation of the mediator instrument G2 with the M->Y confounder conf_MY (instrument exogeneity violation). Default 0.

rho_pop Shared population-structure factor P that induces correlation between G1 and G2 (linkage / stratification). Default 0.

lambda_XM, lambda_MY Per-path confounder loading vectors; distinct values draw conf_XM and conf_MY as independent confounders for the X->M and M->Y paths respectively. If FALSE (default), conf_XM = conf_MY = U1 (backward compatible).

omega_1 Coverage of conf_XM by the path-specific negative controls W1. NULL (default) uses w_signal.

omega_2 Coverage of conf_MY by the path-specific negative controls W2. NULL (default) uses w_signal.

When all are at their defaults, the DGP is identical to the default. When any is non-default, the output additionally includes G1, G2, W1, W2, conf_XM, conf_MY, and (when rho_pop > 0) P.

Examples

```
# Basic total-effect data
dat <- generate_toy_data(n = 200, n_features = 10, seed = 42)

# Mediation with mediator instrument and path-specific NCs
dat <- generate_toy_data(n = 200, n_features = 10, phi = 0.8,
    mo_confounding = 0.8, lambda_XM = c(1, 0), lambda_MY = c(0, 1),
    omega_1 = 0.7, omega_2 = 0.7, seed = 42)
idata <- iconic_data(X = dat$X, Y = dat$Y, M = dat$M,
    G = dat$G[, 1], Gm = dat$Gm,
    W1 = dat$W1, W2 = dat$W2)
```

---

iconic_data *Construct a standardized data object for ICONIC model selection*

Description

Creates an `iconic_data` S3 object from the user's real data, standardizing vectors and matrices into a consistent format that all downstream model selection functions consume.

Usage

```
iconic_data(
  X,
  Y = NULL,
  M = NULL,
  G = NULL,
  Gm = NULL,
  W = NULL,
  W1 = NULL,
  W2 = NULL,
  covariates = NULL,
  feature_names = NULL,
  mediator_names = NULL,
  trained_gan = NULL,
  outcome_type = c("continuous", "survival", "binary"),
  surv_time = NULL,
  surv_event = NULL,
  scale = TRUE,
  recycle_lone_panel = FALSE,
  Z = NULL
)
```

Arguments

`X`	Exposure: numeric vector (length n) or features x samples matrix. If a matrix, column means are taken and scaled (one exposure per sample).
`Y`	Outcome: numeric vector (length n) or features x samples matrix. When a matrix, estimation runs per-feature. When `outcome_type = "binary"`, `Y` is the 0/1 outcome vector (length n).

`M` Optional mediator: numeric vector (length n) or features x samples matrix. When a matrix, estimation runs per-mediator x per-outcome.

`G` Optional exposure instrument: numeric vector (length n) or n x n_features matrix (as returned by `generate_toy_data()`). If a matrix, the first column is extracted. E.g., a polygenic risk score.

`Gm` Optional mediator instrument: numeric vector (length n) or matrix (n x n_mediators). When a matrix, each column is the instrument for the corresponding mediator (e.g., per-isoform cis-eQTLs).

`W` Optional negative-control panel: features x samples matrix. Single-panel NCs used for COCA, PGC.

`W1` Optional path-specific NCs for the X->M path: features x samples matrix. Captures conf_XM. When W1/W2 are absent but W is present, W1 = W2 = W.

`W2` Optional path-specific NCs for the M->Y path: features x samples matrix. Captures conf_MY.

`covariates` Optional data frame of sample-level covariates (n rows). Recognized columns `sex`, `GA`, `mother_ethnicity` are encoded; other numeric columns are z-scored. Names colliding with estimator-reserved tokens are renamed.

`feature_names` Optional character vector of outcome feature names.

`mediator_names` Optional character vector of mediator names.

`trained_gan` Optional `iconic_gan` from `train_gan_on_real_data()`. When supplied, the GAN is attached to the data object and reused by `iconic_sensitivity()` and `iconic_prospect()` instead of auto-training a new one. This avoids retraining when the same data is used across multiple workflow steps.

`outcome_type` Character: `"continuous"` (default, backward-compatible), `"survival"`, or `"binary"`. When `"survival"`, `Y` is not required; instead supply `surv_time` and `surv_event`. Estimation uses Cox proportional-hazards (`coxph`) or RMST pseudo-observation OLS (see `effect_scale` in `iconic_estimate()`). When `"binary"`, supply `Y` as the 0/1 outcome vector (length n); estimation uses logistic regression (log-OR scale) or a linear probability model (risk-difference scale).

`surv_time` Numeric follow-up time vector (length n). Required when `outcome_type = "survival"`; ignored otherwise.

`surv_event` Numeric 0/1 event indicator (length n; 1 = event observed, 0 = censored). Required when `outcome_type = "survival"`; ignored otherwise.

`scale` Logical: center and scale all continuous inputs (X, Y, M, G, Gm, W, W1, W2, and numeric covariates) to mean 0 / sd 1. Default `TRUE`. Scaling parameters are recorded in `$scaling` for back-transformation. Set `FALSE` to preserve the original scale.

`recycle_lone_panel` Logical: when exactly one of `W1` / `W2` is supplied (no pooled `W`), use that lone panel as BOTH path-specific bridges (`W1 = W2`), making the two-bridge estimators PGC2 / PGC2Gm eligible. Default `FALSE`: a lone panel is retained for IV2SLS2's path-specific augmentation and used to derive the pooled `W` (so DIRECT / COCA / PGC and the NC validity screens run), but `has_path_nc` stays `FALSE` and PGC2 / PGC2Gm remain ineligible. Set to `TRUE` only when the single panel is assumed complete for BOTH path confounder composites (the shared-panel special case); IV2SLS2 is the more defensible primary estimator when coverage of the other path's composite is in doubt. A warning is emitted when the recycle is activated.

`Z` Defunct. Renamed to `X`; passing a value errors with a message pointing to `X`. Retained in the signature only to catch and redirect old calls.

Value

An iconic_data S3 object: a named list with $X, $Y, $M, $G, $Gm, $W, $W1, $W2, $covariates, $n, $n_features, $n_mediators, $has_instrument, $has_mediator_instrument, $has_nc, $has_path_nc, $is_mediation, $feature_names, $mediator_names, $trained_gan, $outcome_type, and (when survival) $surv_time, $surv_event, or (when binary) $Y_bin, plus $scaling and $sample_names (from names(X), else the column names of Y/M/W, else S1..Sn).

Bioconductor containers

For Bioconductor workflows, the recommended entry point is a MultiAssayExperiment::MultiAssayExperiment passed to as_iconic_data(): omics panels (outcome, mediator, negative controls, mediator instruments) are experiments and sample-level variables (exposure, instrument, covariates, survival / binary outcome) are colData columns. as_MultiAssayExperiment() converts an iconic_data object back. iconic_data itself is the internal, standardised analysis view that the estimators consume.

Examples

```
# Total-effect only (no mediation)
data <- iconic_data(X = rnorm(100), Y = matrix(rnorm(100 * 10), 10, 100))

# Full mediation with instruments and NCs
data <- iconic_data(
    X = rnorm(100), Y = matrix(rnorm(100 * 10), 10, 100),
    M = rnorm(100), G = rnorm(100), Gm = rnorm(100),
    W = matrix(rnorm(100 * 10), 10, 100)
)
print(data)

# Survival outcome
data <- iconic_data(
    X = rnorm(100), outcome_type = "survival",
    surv_time = rexp(100), surv_event = rbinom(100, 1, 0.6),
    M = rnorm(100), G = rnorm(100), Gm = rnorm(100),
    W = matrix(rnorm(100 * 10), 10, 100)
)
print(data)

# Binary outcome (Y is the 0/1 outcome vector)
data <- iconic_data(
    X = rnorm(100), Y = rbinom(100, 1, 0.4), outcome_type = "binary",
    M = rnorm(100), G = rnorm(100), Gm = rnorm(100),
    W = matrix(rnorm(100 * 10), 10, 100)
)
print(data)
```

---

iconic_diagnose *Diagnose data and determine estimator eligibility*

Description

Runs a battery of diagnostic checks on the user's data and returns an eligibility report for all eight ICONIC estimators. The checks include:

Usage

```
iconic_diagnose(
  data,
  fdr_level = 0.1,
  min_f = 10,
  k = NULL,
  g_threshold = NULL,
  gm_threshold = NULL,
  n_cores = 1,
  allow_no_proxy = TRUE
)
```

Arguments

data	An iconic_data object from iconic_data().
fdr_level	FDR level for NC screens. Default 0.10.
min_f	Minimum partial F for instrument strength. Default 10. Used as the scalar threshold when g_threshold/gm_threshold are NULL (legacy behavior). When thresholds are supplied, min_f is unused for the panel decision (the threshold's R governs).
k	Number of latent confounders assumed for the completeness check. Default NULL: infer from the data via Horn parallel analysis on the residualized-outcome correlation matrix (infer_confounding()); falls back to 1 when inference is not possible (fewer than 5 outcome features). Supply an integer to override inference.
g_threshold	Optional list list(E = 0.5, R = 10) controlling G-dependent method eligibility via the panel distribution. A method is eligible if at least fraction E of instruments have F_G >= R. Default NULL: legacy scalar behavior (median F_G vs min_f).
gm_threshold	Optional list list(E = 0.5, R = 10) controlling Gm-dependent method eligibility (IV2SLS2, PGC2Gm) via the panel distribution. A method is eligible if at least fraction E of mediators have F_Gm >= R. Default NULL: legacy scalar behavior (median F_Gm vs min_f).
n_cores	Number of parallel workers for NC validity screens and panel instrument-strength computation. Default 1 (sequential). Uses parallel::mclapply on Unix and a PSOCK cluster on Windows.
allow_no_proxy	Logical: when TRUE (default), proceed with a message if no instruments and no NCs are supplied (only UNADJ is eligible). When FALSE, error instead.

Details

- **Instrument strength**: first-stage partial F-statistics for G (exposure instrument) and Gm (mediator instrument), with the Stock-Yogo weak-instrument threshold (F >= 10).
- **NC validity (A1)**: screens each negative-control feature for association with the exposure X (the empirically testable projection of "W independent of X given C, U"; nc_validity_screen()).
- **NC independence (A2)**: screens each NC for association with the instrument G (the empirically testable projection of "W independent of G given C, U"; nc_independence_check()).
- **NC independence (A2')**: screens each NC for association with the mediator instrument Gm (the empirically testable projection of "W independent of G_m given C, U"; nc_independence_check_gm()).
- **Path completeness**: checks whether the valid NC panel has enough features to span the confounder subspace and captures the confounder covariance (nc_completeness_check()).

Value

An `iconic_diagnosis` S3 object: a named list with `$instrument_strength`, `$nc_validity`, `$nc_independence`, `$nc_independence_gm`, `$completeness`, `$eligibility`, and `$summary`. `$instrument_strength` and `$instrument_strength$F_Gm` are numeric vectors containing the full panel distributions (one entry per instrument / mediator).

Defaults

Parameter	Default	Source
`fdr_level`	0.10	BH-FDR level for NC screens
`min_f`	10	Stock-Yogo weak-instrument threshold
`k`	1	Single-confounder assumption (typical mediation)
`g_threshold`	NULL	Legacy scalar (median F_G vs min_f)
`gm_threshold`	NULL	Legacy scalar (median F_Gm vs min_f)
`n_cores`	1	Sequential
`allow_no_proxy`	TRUE	Proceed with UNADJ-only when no IV/NC

Examples

```
set.seed(1)
data <- iconic_data(X = rnorm(100), Y = matrix(rnorm(100 * 10), 10, 100),
    G = rnorm(100), W = matrix(rnorm(100 * 10), 10, 100))
diag <- iconic_diagnose(data)
print(diag)
```

---

iconic_estimate *Fit all eligible estimators on real data*

Description

Applies all eligible causal estimators to the user's real data (supplied as an `iconic_data` object) and returns tidy per-feature (and per-mediator) point estimates, standard errors, and p-values.

Usage

```
iconic_estimate(
  data,
  methods = NULL,
  diagnosis = NULL,
  alpha = 0.05,
  n_cores = 1,
  min_f = NULL,
  run_all = FALSE,
  se_method = c("delta", "bootstrap", "composite"),
  n_boot = 500,
  effect_scale = c("loghr", "rmst", "logor", "riskdiff"),
  tau = NULL
)
```

Arguments

data	An iconic_data object from iconic_data().
methods	Optional character vector of estimator names to run (e.g. c("IV2SLS", "PGC")). Default NULL: run all eligible.
diagnosis	Optional iconic_diagnosis object from iconic_diagnose(). When supplied, only eligible estimators are run (unless methods or run_all overrides).
alpha	Significance threshold for significance flags. Default 0.05.
n_cores	Number of parallel workers for per-feature (and per-mediator) estimation. Default 1 (sequential). Uses parallel::mclapply on Unix and a PSOCK cluster on Windows.
min_f	Minimum partial F for the per-transcript weak-instrument gate inside IV2SLS, IV2SLS2, and PGC2Gm. Default NULL: inherits from diagnosis$min_f when a diagnosis is supplied, otherwise 10. Pass an explicit value to override.
run_all	Logical. Default FALSE. When TRUE, overrides diagnosis-based eligibility and runs every method whose required data exists (via .auto_eligible_methods()). The per-transcript min_f gate still applies. This is the "force run" escape hatch for exploratory analysis.
se_method	Character: "delta" (default), "bootstrap", or "composite". When "bootstrap", mediation NDE_se/NIE_se are replaced by the SD of n_boot nonparametric bootstrap resamples. When "composite", mediation NIE_p is replaced by the Huang (2019) JT-comp composite null p-value, which accounts for the three-case structure of H0: alpha*beta=0 and provides higher power than the Sobel/Wald test when signals are sparse. Only applies in mediation mode. NDE_p and NIE_se are unchanged.
n_boot	Integer: number of bootstrap resamples when se_method = "bootstrap". Default 500.
effect_scale	Character: "loghr", "rmst", "logor", or "riskdiff". Only used for non-continuous outcomes. For data$outcome_type = "survival": "loghr" (default) fits Cox proportional-hazards models and reports log-hazard ratios; "rmst" regresses leave-one-out RMST pseudo-observations (Graw et al. 2009) via OLS, reporting effects on the restricted-mean-survival-time (time) scale — a collapsible alternative where the NDE/NIE product decomposition is exact. For data$outcome_type = "binary": "logor" (default) fits logistic outcome stages and reports conditional log-odds ratios; "riskdiff" fits linear probability models (OLS on the 0/1 outcome), reporting risk differences — again a collapsible scale where the product decomposition is exact. Ignored (with a message) when outcome_type = "continuous"; an inert default is remapped (with a message) when it does not match the outcome type.
tau	Numeric: RMST restriction time horizon. Default NULL (90th percentile of follow-up). Used only when effect_scale = "rmst".

Details

If a diagnosis from iconic_diagnose() is supplied, only eligible estimators are run by default. If methods is supplied, only those methods are run (user override). If neither is supplied, all estimators whose required inputs are present are run.

Value

A data frame. For total-effect mode: `feature`, `method`, `beta`, `se`, `pvalue`, `significant`. For mediation mode: `feature`, `mediator`, `method`, `NDE`, `NDE_se`, `NDE_p`, `NIE`, `NIE_se`, `NIE_p`, `NDE_significant`, `NIE_significant`. When `outcome_type = "survival"`, estimates are on the log-HR scale (`effect_scale = "loghr"`) or the RMST/time scale (`effect_scale = "rmst"`). When `outcome_type = "binary"`, estimates are on the log-OR scale (`effect_scale = "logor"`) or the risk-difference scale (`effect_scale = "riskdiff"`).

Examples

```
set.seed(1)
data <- iconic_data(X = rnorm(100), Y = matrix(rnorm(100 * 10), 10, 100),
    G = rnorm(100), W = matrix(rnorm(100 * 10), 10, 100))
diag <- iconic_diagnose(data)
est <- iconic_estimate(data, diagnosis = diag)
head(est)

# Survival outcome
sdat <- iconic_data(X = rnorm(100), outcome_type = "survival",
    surv_time = rexp(100), surv_event = rbinom(100, 1, 0.6),
    G = rnorm(100), W = matrix(rnorm(100 * 10), 10, 100))
est <- iconic_estimate(sdat, effect_scale = "loghr")
est_rmst <- iconic_estimate(sdat, effect_scale = "rmst")

# Binary outcome
bdat <- iconic_data(X = rnorm(100), Y = rbinom(100, 1, 0.4),
    outcome_type = "binary",
    G = rnorm(100), W = matrix(rnorm(100 * 10), 10, 100))
est <- iconic_estimate(bdat, effect_scale = "logor")
est_rd <- iconic_estimate(bdat, effect_scale = "riskdiff")
```

---

iconic_method_colors *Colour palette for iconic methods*

Description

Colour palette for iconic methods

Usage

```
iconic_method_colors
```

Value

A named character vector of colors for the eight ICONIC estimators.

Examples

```
iconic_method_colors
```

iconic_method_order *Default method display order*

Description

Default method display order

Usage

```
iconic_method_order
```

Value

A character vector giving the canonical display order of the eight ICONIC estimators.

Examples

```
iconic_method_order
```

iconic_prospect *Prospective bias-reduction analysis for data without instruments or negative controls*

Description

For users who have exposure (X), mediator (M), and outcome (Y) but no genetic instruments (G, Gm) or negative controls (W), this function simulates what estimates they could expect if they were to collect such data.

Usage

```
iconic_prospect(
  data,
  trained_gan = NULL,
  confounding = c("default", "inferred", "manual"),
  gan_epochs = 100,
  gamma_G_grid = c(0.2, 0.4, 0.6, 0.8, 1),
  target_gamma_G = 0.6,
  n_iter = 30,
  n_features = 10,
  mo_confounding = 0.8,
  phi = 0.8,
  lambda_XM = NULL,
  lambda_MY = NULL,
  omega_1 = 0.7,
  omega_2 = 0.7,
  rho_G1_grid = c(0, 0.1, 0.2, 0.3, 0.5),
  rho_G2_grid = c(0, 0.1, 0.2, 0.3, 0.5),
  omega_grid_rho = c(0.3, 0.7, 1),
  run_rho_sweep = TRUE,
```

```
  bias_threshold = 0.1,
  base_seed = 500,
  verbose = FALSE,
  allow_no_proxy = TRUE,
  outcome_type = c("continuous", "survival", "binary"),
  effect_scale = c("loghr", "rmst", "logor", "riskdiff"),
  surv_h0 = 0.1,
  surv_event_frac = 0.6,
  surv_censor_rate = NULL,
  bin_prev = 0.5
)
```

Arguments

data	An iconic_data object (must have X and Y; M is required for mediation prospect).
trained_gan	Optional iconic_gan from train_gan_on_real_data(). If NULL, a texture model is auto-trained from data.
confounding	Confounding parameter source: "default", "inferred", or "manual". Default "default".
gan_epochs	Epochs for auto-trained GAN. Default 100.
gamma_G_grid	Instrument strength values to sweep (Phase 1). Default c(0.2, 0.4, 0.6, 0.8, 1.0).
target_gamma_G	Target instrument strength for Phase 2. Default 0.6 (matching the DGP default).
n_iter	Replicates per grid cell. Default 30.
n_features	Features per replicate. Default 10.
mo_confounding	Assumed M-O confounding strength. Default 0.8.
phi	Assumed mediator instrument strength. Default 0.8.
lambda_XM	Optional per-path confounder loading vector (X->M path).
lambda_MY	Optional per-path confounder loading vector (M->Y path). Default TRUE.
omega_1, omega_2	Assumed NC coverage. Default 0.7.
rho_G1_grid, rho_G2_grid	Instrument-exogeneity violation values (correlation of each instrument with its path's confounder composite) to sweep in Phase 3 at the target instrument strength. Default c(0, 0.1, 0.2, 0.3, 0.5), matching iconic_sensitivity.
omega_grid_rho	Negative-control coverage values swept jointly with the rho grid in Phase 3, on the diagonal (omega_1 == omega_2). Default c(0.3, 0.7, 1.0), matching iconic_sensitivity. This is independent of the Phase 1 omega_1/omega_2 sweep.
run_rho_sweep	Logical: run the Phase 3 robustness sweep (default TRUE). The sweep crosses instrument-exogeneity violations (rho) with negative-control coverage (omega) and feeds the resulting degradation surface into iconic_recommend() so the recommended estimator is chosen by robustness to both imperfect instruments and weakening controls, rather than by eligibility alone. Set FALSE to skip (faster, but the recommendation then falls back to a single-point / eligibility ranking).
bias_threshold	Tipping-point threshold. Default 0.10.
base_seed	Base RNG seed. Default 500.

verbose	Logical: print progress messages during the sweep. Default FALSE (quiet).
allow_no_proxy	Logical: when TRUE (default), proceed with the prospective sweep even if the data already has instruments/NCs (with a message). When FALSE, error if the data already has IV+NC (use iconic_estimate instead).
outcome_type	"continuous" (default), "survival", or "binary". Threads through to the simulation DGP.
effect_scale	"loghr" (default), "rmst", "logor", or "riskdiff". "loghr"/"rmst" apply to survival outcomes; "logor"/"riskdiff" apply to binary outcomes (an inert default is remapped with a message).
surv_h0, surv_event_frac, surv_censor_rate	Survival DGP parameters See generate_toy_data.
bin_prev	Target prevalence for the binary DGP. Default 0.5.

Details

Phase 1 sweeps instrument strength (gamma_G) to show how estimates converge as the instrument strengthens. Phase 2 runs a full prospective simulation at a target strength, generating synthetic instruments and NCs calibrated to the user's sample size and confounding level.

Value

An iconic_prospect S3 object: a named list with $strength_surface (Phase 1: gamma_G x method estimates), $prospective (Phase 2: full simulation at target strength), $rho_surface (Phase 3: rho_G1 x rho_G2 exogeneity-robustness surface at the target strength; NULL when run_rho_sweep = FALSE), $summary, $recommendation, $texture_source, and $inferred_confounding (when confounding = "inferred").

When to use

Use this as a **bias-reduction prospective** when you have an observational exposure-mediator-outcome triplet but lack the genetic instruments and negative controls that the core estimators require. It quantifies the **relative bias improvement** you could expect by collecting such data: the sweep shows how much of the naive confounding bias is removed as instrument strength and NC coverage increase, letting you decide whether the marginal gain justifies the cost of genotyping / profiling the additional assays. It is a planning tool, not an estimator – it does not produce a causal estimate from your current data, but tells you what a future instrumented study would yield.

Texture model

When trained_gan is NULL and no GAN is attached to data, a texture model is auto-trained from the user's data.

Confounding calibration

The confounding argument controls how the held-fixed confounding parameters are set. In the prospective setting (no instruments or NCs), most parameters cannot be inferred and will fall back to defaults with warnings – this is an honest limitation, not a silent failure.

Defaults

Parameter	Default	Source
confounding	"default"	Use DGP defaults below
gan_epochs	100	Texture-model training budget
gamma_G_grid	c(0.2,0.4,0.6,0.8,1.0)	Instrument-strength sweep
target_gamma_G	0.6	DGP default (gamma_G)
n_iter	30	Replicates per grid cell
n_features	10	Features per replicate
mo_confounding	0.8	Simulation calibration (delta_mo)
phi	0.8	Strong mediator instrument assumption
lambda_XM, lambda_MY	shared	Per-path confounder loadings
omega_1, omega_2	0.7	NC coverage (simulation calibration)
rho_G1_grid, rho_G2_grid	c(0,0.1,0.2,0.3,0.5)	Phase 3 exogeneity sweep
omega_grid_rho	c(0.3,0.7,1.0)	Phase 3 NC-coverage sweep (diagonal)
run_rho_sweep	TRUE	Run Phase 3 robustness sweep
bias_threshold	0.10	Tipping-point threshold
allow_no_proxy	TRUE	Proceed in prospective setting

Examples

```
if (check_torch_setup()) {
    data <- iconic_data(X = rnorm(100), Y = matrix(rnorm(100 * 10), 10, 100),
        M = rnorm(100))
    result <- iconic_prospect(data, n_iter = 2, gan_epochs = 5,
        gamma_G_grid = c(0.4, 0.8), run_rho_sweep = FALSE)
    print(result)
}
```

---

iconic_recommend *Recommend the best causal estimator for the user's data*

Description

Ranks all eligible estimators directly on per-estimand robustness (NDE and NIE separately) from a sensitivity analysis. Returns the top-ranked estimator with a transparent rationale. Every estimator is ranked on its measured robustness to assumption violations, not on an a priori identification-strength class.

Usage

```
iconic_recommend(
  data,
  diagnosis = NULL,
  estimate = NULL,
  sensitivity = NULL,
  criterion = c("combined", "minimax_bias", "ci_coverage"),
  completeness_penalty = c(satisfied = 1, borderline = 0.7, `weak-capture` = 0.5,
```

```
    `under-identified` = 0),
  min_f = 10,
  g_threshold = NULL,
  gm_threshold = NULL,
  auto_sensitivity = TRUE,
  rho_G1_grid = c(0, 0.1, 0.2, 0.3, 0.5),
  rho_G2_grid = c(0, 0.1, 0.2, 0.3, 0.5),
  omega_1 = c(0.3, 0.7, 1),
  omega_2 = c(0.3, 0.7, 1),
  n_iter_sens = 30,
  gan_epochs = 100,
  n_cores = 1,
  verbose = FALSE
)
```

Arguments

data	An iconic_data object.
diagnosis	Optional iconic_diagnosis from iconic_diagnose(). If NULL, auto-eligibility is computed from the data.
estimate	Optional estimate data frame from iconic_estimate(). Used to report point estimates alongside the recommendation.
sensitivity	Optional iconic_sensitivity from iconic_sensitivity(). Used for robustness ranking. When NULL and auto_sensitivity = TRUE (the default), the sensitivity suite is run automatically so the recommendation is robustness-based out of the box.
criterion	Character: "combined" (default), "minimax_bias", or "ci_coverage". Controls how robustness is ranked. "combined" blends bias and CI coverage; "minimax_bias" ranks by worst-case bias across the violation grid; "ci_coverage" ranks by CI coverage of the true effect. When sensitivity is supplied, also populates $per_scenario (best estimator at the origin vs at violation cells).
completeness_penalty	Named numeric vector with values in $[0, 1]$ mapping the graded completeness verdict to a confidence multiplier applied to bridge-dependent estimators (DIRECT, COCA, PGC, PGC2, PGC2Gm). Default c(satisfied = 1.0, borderline = 0.7, "weak-capture" = 0.5, "under-identified" = 0). Instrument-only estimators (IV2SLS, IV2SLS2) are not discounted. Override to sensitivity-check the discount.
min_f	Instrument-strength gate (first-stage F threshold) used when diagnosis is NULL and iconic_diagnose() is run internally. Default 10. Also used for the requirement labels when the supplied diagnosis predates the stored min_f field.
g_threshold, gm_threshold	Optional instrument-selection thresholds passed to iconic_diagnose() when diagnosis is NULL. Default NULL (use the iconic_diagnose defaults).
auto_sensitivity	Logical: when TRUE (default) and sensitivity = NULL, run iconic_sensitivity() internally to obtain the robustness surface. Requires the torch backend; when torch is unavailable the function falls back to eligibility-only ranking with a message. Set FALSE to skip the auto-run.

`rho_G1_grid`, `rho_G2_grid`
: Instrument-exogeneity violation grid used for the auto-run sensitivity suite. Default `c(0, 0.1, 0.2, 0.3, 0.5)`.

`omega_1`, `omega_2`
: Negative-control coverage grid for the auto-run. Default `c(0.3, 0.7, 1.0)`, swept on the diagonal (`omega_1 == omega_2`).

`n_iter_sens`
: Replicates per grid cell for the auto-run. Default 30.

`gan_epochs`
: GAN training epochs for the auto-run. Default 100.

`n_cores`
: Cores for the auto-run sensitivity sweep. Default 1.

`verbose`
: Logical: print progress messages. Default `FALSE` (quiet). Also silences the auto-run sensitivity suite.

Details

When `sensitivity` is supplied (from `iconic_sensitivity()`), eligible estimators are ranked by robustness: the estimator whose estimates degrade least across the assumption-violation surface ranks higher. NDE and NIE robustness are computed separately, so a mediation estimator is ranked on the estimand of interest rather than a pooled maximum.

Value

An `iconic_recommendation` S3 object: a named list with `$ranking` (data frame: estimator, eligible, rank, per-estimand robustness scores, `composite`, `confidence_mult`, `final_score`, rationale), `$recommended` (top eligible, non-naive estimator by the data-driven composite score), `$recommended_NDE` and `$recommended_NIE` (top estimator per estimand), `$per_scenario` (when `sensitivity` is supplied), `$completeness_penalty`, and `$summary`.

Examples

```
data <- iconic_data(X = rnorm(100), Y = matrix(rnorm(100 * 10), 10, 100),
    G = rnorm(100), W = matrix(rnorm(100 * 10), 10, 100))
diag <- iconic_diagnose(data)
est <- iconic_estimate(data, diagnosis = diag)
rec <- iconic_recommend(data, diagnosis = diag, estimate = est,
    auto_sensitivity = FALSE)
print(rec)
```

---

iconic_sensitivity *Sensitivity (degradation) surface and effect-decomposition bias sweep*

Description

Sweeps a 2D grid of instrument-independence violations (`rho_G1` x `rho_G2`) and reports how each estimator's NDE/NIE bias degrades as the genetic instruments become correlated with the unmeasured confounders.

Usage

```
iconic_sensitivity(
  data,
  diagnosis = NULL,
  trained_gan = NULL,
  confounding = c("default", "inferred", "manual"),
  gan_epochs = 100,
  rho_G1_grid = c(0, 0.1, 0.2, 0.3, 0.5),
  rho_G2_grid = c(0, 0.1, 0.2, 0.3, 0.5),
  n_iter = 30,
  n_samples = NULL,
  n_features = 10,
  phi = NULL,
  mo_confounding = 0.8,
  lambda_XM = NULL,
  lambda_MY = NULL,
  omega_1 = c(0.3, 0.7, 1),
  omega_2 = c(0.3, 0.7, 1),
  bias_threshold = 0.1,
  base_seed = 700,
  n_cores = 1,
  verbose = FALSE,
  outcome_type = NULL,
  effect_scale = c("loghr", "rmst", "logor", "riskdiff"),
  surv_h0 = 0.1,
  surv_event_frac = 0.6,
  surv_censor_rate = NULL,
  bin_prev = 0.5
)
```

Arguments

data	An iconic_data object.
diagnosis	Optional iconic_diagnosis from iconic_diagnose(). Used to infer phi and mo_confounding when not explicitly supplied.
trained_gan	Optional iconic_gan from train_gan_on_real_data(). If NULL, a texture model is auto-trained from data.
confounding	Confounding parameter source: "default" (fixed defaults), "inferred" (data-calibrated via infer_confounding()), or "manual" (use explicitly supplied arguments). Default "default". Alternatively, supply a precomputed iconic_confounding object from a prior infer_confounding() call to reuse it as-is (equivalent to "inferred" but skips recomputation). When "inferred" runs internally, the gap-based calibration uses infer_confounding()'s documented random subset of mediators/features (max_infer_tasks), not the full panel. Inferred scalars only fill in mo_confounding / omega_1 / omega_2 when those arguments are left at their defaults; explicitly supplied values (e.g. an omega sweep) always take precedence.
gan_epochs	Epochs for auto-trained GAN. Default 100.
rho_G1_grid	Values of rho_G1 (G correlation with conf_XM). Default c(0, 0.1, 0.2, 0.3, 0.5).

rho_G2_grid	Values of rho_G2 (Gm correlation with conf_MY). Default c(0, 0.1, 0.2, 0.3, 0.5).
n_iter	Replicates per grid cell. Default 30.
n_samples	Samples per replicate. If NULL, uses data$n.
n_features	Features per replicate. Default 10.
phi	Mediator instrument strength. If NULL, inferred from diagnosis (F_Gm) or defaults to 0.8.
mo_confounding	M-O confounding strength. Default 0.8. Used when confounding = "default" or "manual".
lambda_XM	Optional per-path confounder loading vector (X->M path). NULL (default) = shared loadings.
lambda_MY	Optional per-path confounder loading vector (M->Y path). NULL (default) = shared loadings.
omega_1, omega_2	NC coverage of each path's confounder composite (conf_XM / conf_MY). Default c(0.3, 0.7, 1.0), swept jointly with the rho grid. When the two vectors are identical (the default), the sweep is taken on the diagonal (omega_1 == omega_2); supply distinct vectors to cross the full omega_1 x omega_2 grid. Used when confounding = "default" or "manual".
bias_threshold	Absolute bias threshold for tipping-point annotation. Default 0.10.
base_seed	Base RNG seed. Default 700.
n_cores	Number of parallel workers for simulation replicates. Default 1 (sequential). Uses parallel::mclapply on Unix and a PSOCK cluster on Windows.
verbose	Logical: print progress messages during the sweep. Default FALSE (quiet).
outcome_type	NULL (inherit from data, default) or "continuous" / "survival" / "binary". When survival, the sensitivity sweep uses the Cox / RMST survival mediation drivers; when binary, the logistic / linear-probability-model binary mediation drivers.
effect_scale	"loghr" (default), "rmst", "logor", or "riskdiff". "loghr"/"rmst" apply to survival outcomes; "logor"/"riskdiff" apply to binary outcomes (an inert default is remapped with a message).
surv_h0	Baseline hazard for survival DGP. Default 0.1.
surv_event_frac	Target fraction of observed events. Default 0.6.
surv_censor_rate	Explicit censoring rate. Default NULL.
bin_prev	Target prevalence for the binary DGP. Default 0.5.

Details

The grid is calibrated to the user's data: sample size, mediator instrument strength (phi), confounding level, and covariate / outcome texture are inferred from the iconic_data object and diagnosis when available.

Value

An iconic_sensitivity S3 object: a named list with $surface (data frame: rho_G1, rho_G2, method, NDE_bias, NIE_bias, NDE_rmse, NIE_rmse, NDE_type1, NIE_type1, tipped), $grid, $tipping_points, $summary, $texture_source (how the GAN was obtained), and $inferred_confounding (when confounding = "inferred").

When to use

Use this as an **effect-decomposition bias sweep** when you need to know how robust a mediation estimate (NDE/NIE) is to violations of the genetic instruments' independence from the confounders. The degradation surface shows, for each estimator, the point at which bias becomes material — the "tipping point" — so you can report not just a point estimate but the range of violations it tolerates. This is the mediation analogue of a total-effect sensitivity sweep and is most informative when comparing estimators that make different independence assumptions (e.g. IV2SLS2 vs PGC2Gm).

Texture model

When `trained_gan` is `NULL` and no GAN is attached to `data`, a texture model is auto-trained from the user's data (GAN via `torch` if available, otherwise a multivariate-normal fallback). This ensures the synthetic covariate and outcome texture matches the user's cohort. Supply `trained_gan` explicitly or attach one via `iconic_data(trained_gan = ...)` to reuse a pre-trained model and avoid retraining.

Confounding calibration

The `confounding` argument controls how the held-fixed confounding parameters (`mo_confounding`, `omega_1`, `omega_2`) are set:

- `"default"`: fixed defaults (0.8, 0.7, 0.7).
- `"inferred"`: calls `infer_confounding()` to estimate parameters from the data. Parameters that cannot be inferred fall back to defaults with warnings.
- `"manual"`: use the explicitly supplied arguments.

Defaults

Parameter	Default	Source
`confounding`	"default"	Use DGP defaults below
`gan_epochs`	100	Texture-model training budget
`rho_G1_grid`	c(0,0.1,0.2,0.3,0.5)	Independence-violation sweep
`rho_G2_grid`	c(0,0.1,0.2,0.3,0.5)	Independence-violation sweep
`n_iter`	30	Replicates per grid cell
`n_features`	10	Features per replicate
`mo_confounding`	0.8	Simulation calibration (delta_mo)
`lambda_XM`, `lambda_MY`	shared	Per-path confounder loadings
`omega_1`, `omega_2`	c(0.3,0.7,1.0)	NC coverage (swept on the diagonal)
`bias_threshold`	0.10	Tipping-point threshold

Examples

```
if (check_torch_setup()) {
    data <- iconic_data(X = rnorm(100), Y = matrix(rnorm(100 * 10), 10, 100),
        M = rnorm(100), G = rnorm(100), Gm = rnorm(100),
        W = matrix(rnorm(100 * 10), 10, 100))
    sens <- iconic_sensitivity(data, n_iter = 2, gan_epochs = 5,
        rho_G1_grid = c(0, 0.2), rho_G2_grid = c(0, 0.2))
    print(sens)
}
```

`infer_confounding` *Infer confounding parameters from the user's data*

Description

Estimates the held-fixed confounding parameters that `iconic_sensitivity()` and `iconic_prospect()` use when `confounding = "inferred"`.

Usage

```
infer_confounding(
  data,
  diagnosis = NULL,
  estimate = NULL,
  n_cores = 1,
  max_infer_tasks = 50
)
```

Arguments

`data`	An `iconic_data` object.
`diagnosis`	Optional `iconic_diagnosis` from `iconic_diagnose()`. Used to check instrument strength before using estimator gaps.
`estimate`	Optional estimate data frame from `iconic_estimate()`. If `NULL`, estimates are computed internally.
`n_cores`	Number of parallel workers for omega inference and k permutation. Default 1 (sequential). Uses `parallel::mclapply` on Unix and a PSOCK cluster on Windows.
`max_infer_tasks`	Cap on the number of mediators and of outcome features used when `estimate = NULL` and estimates must be computed internally. The confounding-strength gaps are averages across the mediator x feature grid, so a random subset of at most `max_infer_tasks` mediators and `max_infer_tasks` features gives an unbiased Monte Carlo estimate at much lower cost. Default 50. Panels smaller than the cap are used in full.

Value

An `iconic_confounding` S3 object: a named list with `$conf_strength`, `$mo_confounding`, `$omega_1`, `$omega_2`, `$k`, `$unavailable` (character vector of parameters that could not be inferred), and `$warnings` (character vector of accumulated warnings). Each parameter slot is a list with `estimate`, `method`, `assumption`, `available` (logical), and `warning` (character or NULL).

Inference methods

- **conf_strength** (delta): the gap between the unadjusted OLS estimate and the IV2SLS estimate, averaged across features. Requires a valid exposure instrument G (F >= 10). Assumes the exclusion restriction holds – using IV2SLS validity to calibrate a benchmark that tests IV2SLS validity is circular, so the estimate should be interpreted as a best-case calibration.

- **mo_confounding** (delta_mo): the gap between the IV2SLS NIE and the IV2SLS2 NIE, averaged across features. Requires valid G + Gm (both F >= 10). Same circularity caveat.
- **omega** (omega_1, omega_2): the square root of the R-squared from regressing each negative-control feature on the outcome residualized on X + C, averaged across features. Conflates NC coverage with confounder strength – reported as a composite, not pure coverage.
- **k**: the number of latent confounders, estimated via parallel analysis (Horn, 1965) on the correlation matrix of outcomes residualized on X + C. Requires at least 5 outcome features. Returns a point estimate and a bootstrap confidence interval.

Defaults (when inference is unavailable)

When a parameter cannot be inferred from the data, the following defaults are used (from the simulation calibration):

Parameter	Default	Source
`conf_strength` (delta)	0.8	Simulation calibration
`mo_confounding` (delta_mo)	0.8	Simulation calibration
`omega_1`, `omega_2`	0.7	NC coverage (simulation calibration)
`k`	1	Single-confounder assumption (typical mediation)
`phi`	0.8	Strong mediator instrument assumption
`lambda_XM`, `lambda_MY`	shared	Per-path confounder loadings

These defaults are reported with a warning so the user knows the value was not inferred from their data.

Examples

```
data <- iconic_data(X = rnorm(100), Y = matrix(rnorm(100 * 10), 10, 100),
    M = rnorm(100), G = rnorm(100), Gm = rnorm(100),
    W = matrix(rnorm(100 * 10), 10, 100))
diag <- iconic_diagnose(data)
est <- iconic_estimate(data, diagnosis = diag)
conf <- infer_confounding(data, diagnosis = diag, estimate = est,
    max_infer_tasks = 5)
print(conf)
```

`list_nc_models` *List the built-in negative-control models*

Description

List the built-in negative-control models

Usage

```
list_nc_models()
```

Value

Character vector of registered NC model names.

Examples

```
list_nc_models()
```

`load_real_input_data` *Load and standardise a real multi-omic dataset for GAN training*

Description

Accepts the exposure, outcome, mediator (optional), and negative-control panels as `features x samples` matrices together with a sample-level covariate frame, and returns the tidy structures used by `train_gan_on_real_data()` and the calibration step of `run_single_iteration()`.

Usage

```
load_real_input_data(
  X_matrix = NULL,
  Y_matrix = NULL,
  M_matrix = NULL,
  W_matrix = NULL,
  covariates_df = NULL,
  example = FALSE,
  marginal_method = "auto",
  residualize_on = c("XC", "XCW"),
  Z_matrix = NULL
)
```

Arguments

`X_matrix`	Exposure panel, `features x samples` (e.g. an exposure or molecular score across probes). Column means become the per-sample exposure.
`Y_matrix`	Outcome, either a scalar vector of length `n_samples` (e.g. a clinical or molecular trait) or a `features x samples` matrix. When a vector, it defines `n_samples`; when a matrix, `n_samples` and `n_features`.
`M_matrix`	Mediator panel, `features x samples` (e.g. gene or transcript expression). When supplied, its per-sample summary is added to the GAN training frame (so the generator learns the mediator's marginal distribution) and its residual correlation matrix is computed and stored.
`W_matrix`	Optional negative-control panel, `features x samples`. When supplied the controls are treated as user-provided real structure; when `NULL` they are generated by a negative-control model at simulation time.
`covariates_df`	Optional data frame of sample-level covariates. A `sample_id` column is used to align samples if present; recognised columns `sex`, `GA`, `mother_ethnicity` are encoded, and any other numeric columns are z-scored and kept. Names colliding with estimator-reserved tokens are renamed with a `cov_` prefix.
`example`	Force return of the built-in example dataset. Default `FALSE` (auto-enabled when no matrices are given).

`marginal_method`
: Marginal fitting method for the feature-level copula texture model (mediator panel): `"auto"` (default, uses parametric if KS test passes at $p > 0.05$, otherwise empirical), `"empirical"` (always empirical CDF), or `"parametric"` (always best parametric fit by AIC). See `train_feature_texture()`.

`residualize_on`
: Character: `"XC"` (default) residualizes the Y/W correlation on exposure + covariates only; `"XCW"` additionally residualizes on the W bridge proxy (PC1 of W), reducing U-signature double-counting. Recommended only when the completeness-capture test reports "strong".

`Z_matrix`
: Defunct. Renamed to `X_matrix`; passing a value errors with a message pointing to `X_matrix`. Retained in the signature only to catch and redirect old calls.

Details

In addition to the sample-level training frame (one row per sample with exposure level, outcome level, and encoded covariates), the function computes feature-level residual correlation matrices for each available panel (Y, M, W). These capture the cross-feature correlation structure that remains after removing the sample-level signal, so the simulation can inject realistic correlated noise into the mediator and negative- control panels.

If no matrices are supplied a small built-in example dataset is returned, so the pipeline can be exercised end-to-end without real data.

Value

A list with `gan_training_data` (sample x variable data frame with exposure level, outcome level, mediator level, and encoded covariates), `original_matrices` (`X`, `Y`, `M`, `W`), `covariates` (encoded frame), `feature_correlations` (list with `Y`, `W` residual correlation matrices, or `NULL` for absent panels), `feature_texture` (an `iconic_feature_texture` object for the mediator panel, or `NULL` when no M matrix is supplied), `sample_names`, `feature_names`, `n_samples`, `n_features`.

Examples

```
dat <- load_real_input_data(example = TRUE)
names(dat)
```

---

nc_completeness_capture

Covariance-capture completeness test

Description

Operationalizes the proximal completeness condition as whether the negative-control panel W *captures the confounder covariance* — i.e., whether adding W reduces U's contribution to the outcome toward zero.

Usage

```
nc_completeness_capture(
  dat,
  outcome = "Y",
  n_perm = 1000,
```

```
  n_cores = 1,
  thresholds = list(strong = 0.3, weak = 0.1)
)
```

Arguments

dat	Dataset list from run_single_iteration() or generate_toy_data(), containing W, an outcome block (Y or M), and synthetic_data (covariates).
outcome	Outcome block to test against: "Y" (default) or "M".
n_perm	Number of permutations for the null. Default 1000.
n_cores	Number of parallel workers. Default 1.
thresholds	Named list with strong and weak R^2 cutoffs for the verdict. Default list(strong = 0.3, weak = 0.1).

Details

The test computes the incremental R^2 of W for the outcome above covariates C alone: R^2(W | C) = R^2(Y ~ C + W) - R^2(Y ~ C), averaged across outcome features. A permutation null is generated by permuting the W-outcome association (holding the C-outcome association fixed) over n_perm permutations, yielding a permutation p-value for "W captures U-signal beyond chance."

This addresses the concern that the count-based completeness check (dim(W_valid) >= k) is necessary but not sufficient: completeness is about the covariance the controls capture, not the number of proxies. A panel can have many controls that each weakly capture U, or few controls that together capture it well.

Value

A list with: capture_R2 (point estimate, averaged across outcome features), capture_pvalue (permutation p-value), capture_verdict ("strong", "weak", or "negligible"), null_distribution (numeric vector of permuted R^2 values), n_features (number of outcome features tested).

Examples

```
dat <- run_single_iteration(n_features = 10, n_confounders = 1, seed = 1)
nc_completeness_capture(dat, n_perm = 99)
```

---

nc_completeness_check *Check negative-control completeness (dimensional + covariance-capture)*

Description

Reports two components of the proximal-inference completeness condition:

Usage

```
nc_completeness_check(
  dat,
  n_valid_controls = NULL,
  fdr_level = 0.1,
  n_cores = 1,
  outcome = "Y",
  n_perm = 1000,
  capture_thresholds = list(strong = 0.3, weak = 0.1)
)
```

Arguments

dat	Dataset list from run_single_iteration() or generate_toy_data().
n_valid_controls	Optional override: the number of valid controls known from the study design. If NULL (default), the function runs both empirical screens and counts controls that pass both.
fdr_level	FDR level for the empirical screens (used only when n_valid_controls is NULL).
n_cores	Number of parallel workers for the empirical screens. Default 1 (sequential).
outcome	Outcome block for the capture test: "Y" (default) or "M" (mediator). When NULL, the capture component is skipped and only the dimensional verdict is returned.
n_perm	Number of permutations for the capture-test null. Default 1000.
capture_thresholds	Named list with strong and weak R^2 cutoffs for the capture verdict. Default list(strong = 0.3, weak = 0.1).

Details

(1) **Dimensional** (the legacy count-based check): the number of valid negative-control features vs the number of latent confounders k. Bridge-function estimators (COCA, PGC) require at least as many valid controls as confounders (Miao, Geng & Tchetgen Tchetgen, 2018). When dim(W_valid) < k, no estimator built on those controls can recover the causal effect, regardless of sample size.

(2) **Covariance-capture**: whether the controls actually *capture the confounder covariance* — i.e., whether adding W reduces U's contribution to the outcome toward zero, operationalized as the incremental R^2 of W for the outcome above covariates alone, with a permutation null. This addresses the concern that completeness is about covariance captured, not proxy number.

The composite completeness verdict requires the dimensional component to pass (satisfied/borderline) AND the capture component to be non- negligible (strong/weak). When the dimensional component passes but capture is negligible, the verdict is "weak-capture".

Value

A list with: n_valid_controls (count), k (number of confounders), dim_W, dimensional ("satisfied", "borderline", "under-identified"), capture (output of nc_completeness_capture(), or NULL), completeness (composite: "satisfied", "borderline", "under-identified", or "weak-capture"), screen_X (A1 screen results, if run), screen_G (A2 screen results, if run).

Examples

```
dat <- run_single_iteration(n_features = 10, n_confounders = 1, seed = 1)
nc_completeness_check(dat, n_perm = 50)
```

---

nc_cpg *CpG-predicted-expression negative-control model (SCENIC case)*

Description

Simulates spatially correlated CpG methylation whose signal is partly driven by the captured confounders, then forms each negative control as a linear prediction from the methylation sites ("CpG-predicted expression"). The controls therefore carry confounder information only to the extent the methylation does, mediated through a realistic spatial methylation layer.

Usage

```
nc_cpg(U, covariates, params)
```

Arguments

U	n x k confounder matrix.
covariates	Covariate data frame (unused; kept for the NC contract).
params	List with n_features, and optionally coverage (confounder->methylation strength, default 0.7), captured (confounder indices, default all), n_cpg (methylation sites, default 60), rho (AR(1) spatial correlation across sites, default 0.6), MMCpG (methylation-confounding multiplier, default 1), MMCon (default 1), and noise_cor (a p x p correlation matrix for correlated idiosyncratic noise, or NULL for independent noise).

Details

When noise_cor is supplied (a p x p correlation matrix), the idiosyncratic noise added to each control is drawn from a multivariate normal with that correlation structure, so the controls retain realistic cross-feature correlations conditional on the confounder.

Value

n x n_features matrix of CpG-predicted negative controls.

Examples

```
U <- matrix(rnorm(100), 100, 1)
W <- nc_cpg(U, covariates = NULL,
    params = list(n_features = 10, coverage = 0.7, n_cpg = 20))
dim(W)
```

nc_independence_check *Test instrument-independence of negative controls (W | G | C)*

Description

For each control feature, computes the partial correlation with the genetic instrument G after residualising both on observed covariates C, and reports the p-value. Controls significantly associated with G after FDR correction may carry meQTL / allele-specific effects that violate the instrument-independence assumption (A2).

Usage

```
nc_independence_check(dat, fdr_level = 0.1, n_cores = 1)
```

Arguments

`dat`	Dataset list from `run_single_iteration()` or `generate_toy_data()`.
`fdr_level`	Target FDR for BH correction. Default 0.10.
`n_cores`	Number of parallel workers. Default 1 (sequential). Uses `parallel::mclapply` on Unix and a PSOCK cluster on Windows.

Value

A data frame with one row per control feature: `feature`, `partial_r`, `p_value`, `fdr`, `significant`, `verdict`.

Examples

```
dat <- run_single_iteration(n_features = 10, seed = 1)
nc_independence_check(dat)
```

nc_independence_check_gm *Test mediator-instrument independence of negative controls (W | Gm | C)*

Description

For each control feature, computes the partial correlation with the mediator-specific genetic instrument Gm after residualising both on observed covariates C, and reports the p-value. Controls significantly associated with Gm after FDR correction may carry eQTL / allele-specific effects that violate the mediator-instrument-independence assumption (A2'). This is the mediator-instrument analogue of `nc_independence_check()`: just as the exposure instrument G must be independent of the negative controls, so must the mediator instrument Gm.

Usage

```
nc_independence_check_gm(dat, fdr_level = 0.1, n_cores = 1)
```

Arguments

dat	Dataset list from run_single_iteration() or generate_toy_data(), containing Gm, W, and synthetic_data. If Gm is absent (i.e. no mediator instrument was generated), the function returns NULL with a message.
fdr_level	Target FDR for BH correction. Default 0.10.
n_cores	Number of parallel workers. Default 1 (sequential). Uses parallel::mclapply on Unix and a PSOCK cluster on Windows.

Details

For a mediator cis-eQTL instrument, this screen tests whether the eQTL instrument is associated with the negative-control panel – a violation would indicate shared genomic structure between the instrument SNPs and the control features.

Value

A data frame with one row per control feature: feature, partial_r, p_value, fdr, significant, verdict. Returns NULL if dat$Gm is not present.

Examples

```
dat <- run_single_iteration(n_features = 10, phi = 0.8, seed = 1)
nc_independence_check_gm(dat)
```

---

nc_proxy *Direct-proxy negative-control model*

Description

Each control is a coverage-weighted mixture of the captured confounders plus noise. With one confounder and coverage = omega this reduces to the classic w_signal proxy in generate_toy_data(). Setting captured to a strict subset of the confounders models controls that miss part of the confounder subspace (invalidating the negative-control assumption).

Usage

```
nc_proxy(U, covariates, params)
```

Arguments

U	n x k confounder matrix.
covariates	Covariate data frame (unused; kept for the NC contract).
params	List with n_features, and optionally coverage (scalar in [0, 1], default 0.7), captured (integer confounder indices the controls see, default all), noise_sd (default 0.3), MMCon (loading multiplier, default 1), mode (default "shared"), and noise_cor (a p x p correlation matrix for correlated noise, or NULL for independent noise).

Details

When noise_cor is supplied (a p x p correlation matrix), the noise component is drawn from a multivariate normal with that correlation structure, so the negative controls retain realistic cross-feature correlations conditional on the confounder. When noise_cor is NULL (default), the noise is independent across features.

Value

n x n_features matrix of negative controls.

Modes

"shared" All columns carry the same rowMeans(U[, captured]) signal. This is the original behaviour (backward-compatible) and is numerically stable, but the W matrix has only one effective dimension regardless of n_features, so the proximal completeness condition is never binding for the matrix-bridge PGC.

"distinct" Column f captures confounder captured[((f - 1) \%\% length(captured)) + 1]. Different columns therefore carry signals from different confounders, giving the W matrix genuine dimensional structure. This is the mode to use when benchmarking the completeness cliff: the matrix-bridge PGC is identified only when n_features >= k.

Examples

```
U <- matrix(rnorm(100), 100, 1)
W <- nc_proxy(U, covariates = NULL,
    params = list(n_features = 10, coverage = 0.7))
dim(W)
```

---

nc_support_check *Negative-control support/range check*

Description

Diagnostic for whether the negative-control panel captures the full support of the confounder, or only part of it. Uses a confounder proxy U_tilde = resid(X ~ G + C) (the instrument-purged exposure residual, which carries the confounding signal) and asks how much of U_tilde the NC panel explains, and whether each individual NC covers a distinct share of that signal.

Usage

```
nc_support_check(dat, fdr_level = 0.1)
```

Arguments

dat	Dataset list from run_single_iteration(), generate_toy_data(), or the .to_nc_dat() bridge in iconic_diagnose(), containing W, X, and synthetic_data (covariates). G is used when present.
fdr_level	FDR level for flagging controls whose unique contribution is indistinguishable from zero. Default 0.10.

Details

A panel can pass the count-based completeness check (`dim(W_valid) >= k`) and the covariance-capture test while still covering only part of the confounder support — for example when every control loads on the same single confounder direction. This diagnostic reports the multivariate `R^2(U_tilde | W)` (how much of the confounder proxy the panel explains) and a per-NC `support_ratio` (the partial correlation of each control with `U_tilde` given the other controls), which flags controls that add no unique coverage.

This is a diagnostic, not a gate: `U_tilde` is itself an imperfect proxy (it mixes the confounder with exposure noise), so the values are interpretable only comparatively across controls and panels.

Value

A list with: `R2_utilde_given_W` (multivariate R^2 of the confounder proxy on the full NC panel), `support` (data frame with per-NC `support_ratio` = partial correlation of the control with `U_tilde` given the other controls, `p_value`, and `adds_coverage` flag), `n_controls`, and `verdict` ("broad" when `R2_utilde_given_W >= 0.5`, "partial" when >= 0.2, else "narrow").

Examples

```
dat <- run_single_iteration(n_features = 10, n_confounders = 1, seed = 1)
nc_support_check(dat)
```

`nc_validity_check` *Check negative-control validity across confounding scenarios*

Description

Focuses on whether the negative-control estimators (COCA, PGC) hold as the controls' coverage of the confounder subspace drops and as the number of latent confounders grows. Reports their bias/RMSE alongside IV2SLS, and flags scenarios where identification is not credible — in particular when the number of latent confounders exceeds the effective number of valid controls (the proximal-inference completeness condition).

Usage

```
nc_validity_check(
  trained_gan = NULL,
  coverage_grid = c(0.2, 0.5, 0.8, 1),
  k_grid = c(1, 2, 3),
  conf_strength = 0.8,
  n_valid_controls = 1,
  n_iter = 50,
  n_samples = 500,
  n_features = 20,
  nc_model = "proxy",
  base_seed = 800,
  n_cores = 1
)
```

Arguments

`trained_gan`	An `iconic_gan` (or `NULL`).
`coverage_grid`	Coverage values to sweep. Default `c(0.2, 0.5, 0.8, 1)`.
`k_grid`	Numbers of confounders. Default `c(1, 2, 3)`.
`conf_strength`	Fixed confounding strength. Default 0.8.
`n_valid_controls`	Number of *distinct valid* controls the design provides (for the identifiability flag). Default 1.
`n_iter`, `n_samples`, `n_features`, `nc_model`, `base_seed`, `n_cores`	As in `gan_sensitivity()`.

Details

The matrix-bridge PGC is the estimator for which the completeness condition is binding: when k > `n_valid_controls`, PGC is under-identified while IV2SLS remains unbiased (it does not depend on NC completeness).

Value

A list with `summary` (COCA/PGC/IV2SLS bias & RMSE per scenario, with an `identified` flag) and `verdict` (short per-scenario diagnosis).

Examples

```
chk <- nc_validity_check(NULL, coverage_grid = 0.7, k_grid = 1,
    n_iter = 2, n_samples = 100, n_features = 5)
chk$verdict
```

`nc_validity_screen` *Screen negative controls for exposure dependence (W | X | C)*

Description

Regresses each negative-control feature on the exposure X (plus observed covariates C) and flags controls whose association with X survives Benjamini-Hochberg FDR control at the requested level.

Usage

```
nc_validity_screen(
  dat,
  fdr_level = 0.1,
  alpha = 0.05,
  n_cores = 1,
  criterion = c("both", "fdr", "magnitude"),
  magnitude_threshold = 0.1,
  u_proxy = NULL
)
```

Arguments

dat	Dataset list from run_single_iteration() or generate_toy_data(), containing W, X, and synthetic_data (covariates).
fdr_level	Target false-discovery rate for BH correction. Default 0.10.
alpha	Per-test significance level used before FDR adjustment (informational only). Default 0.05.
n_cores	Number of parallel workers. Default 1 (sequential). Uses parallel::mclapply on Unix and a PSOCK cluster on Windows.
criterion	Which criterion to use: "fdr" (legacy), "magnitude", or "both" (default).
magnitude_threshold	|partial_r(W,X|C)| cutoff for the magnitude branch. Default 0.10.
u_proxy	Optional outcome-derived proxy for U (e.g. the leading residualized-outcome principal component) used to compute relative_effect = |partial_r(W,X|C)| / |partial_r(W,U_proxy|C)|. When NULL (default), relative_effect is NA and only the FDR/magnitude branches are used.

Details

A control that is significantly associated with X after adjusting for C violates the negative-control assumption (A1): it is either Such controls should be dropped before running COCA / PGC.

Value

A data frame with one row per control feature: feature, p_value, fdr, partial_r, relative_effect, significant (composite, per criterion), verdict (composite), verdict_fdr, verdict_magnitude.

Assumption

A1 states that negative-control outcomes are independent of the exposure **conditional on observed covariates C and the unmeasured confounder U** (U is unobserved). NC outcome proxies may cause or be caused by the outcome or instruments; the screen tests the observable marginal association with X given C.

Criterion

criterion = "fdr" (the legacy behavior) flags controls whose association with X survives BH-FDR. criterion = "magnitude" flags controls by partial-correlation *size* (|partial_r(W,X|C)| > magnitude_threshold), which is less vulnerable to the "always significant via U" problem: because W shares U with X by construction, the FDR test can flag valid controls as violations purely from the intended confounder-sharing signal. criterion = "both" (default) requires a control to pass both branches to be deemed valid.

Distinguishing "W downstream of X" (a true A1 violation) from "W shares a cause with X" (the intended NC behavior) remains an open problem without a clean test statistic. The relative_effect column (|partial_r(W,X|C)| / |partial_r(W,U_proxy|C)|, when u_proxy is supplied) is exposed as a *diagnostic*: a control downstream of X tends to have a larger W-X association relative to its W-U association, but the separation is not always clean. Analysts should inspect partial_r and relative_effect together rather than relying on a single gate.

Examples

```
dat <- run_single_iteration(n_features = 10, seed = 1)
nc_validity_screen(dat)
```

plot_bias *Plot absolute bias vs a swept parameter*

Description

Plot absolute bias vs a swept parameter

Usage

```
plot_bias(
  sweep_df,
  param_label = "Parameter Value",
  methods = iconic_method_order,
  title = "Absolute Bias vs Parameter"
)
```

Arguments

sweep_df	Data frame from sweep_param()$summary.
param_label	X-axis label. Default "Parameter Value".
methods	Methods to include. Default: all five.
title	Plot title.

Value

Invisibly returns NULL; called for the side effect of plotting absolute bias vs a parameter.

Examples

```
sweep <- sweep_param("conf_str", c(0.5, 0.8), n_iter = 2,
    n_samples = 100, n_features = 5)
plot_bias(sweep$summary)
```

plot_bias_boxplot *Grouped boxplots of per-seed bias across a parameter sweep*

Description

Grouped boxplots of per-seed bias across a parameter sweep

Usage

```
plot_bias_boxplot(
  iter_bias,
  param_grid,
  xlab,
  ylab = "Bias (mean estimate - true)",
  main = "",
  xfmt = "%.2f",
  legend_pos = "topleft",
  methods = iconic_method_order
)
```

Arguments

iter_bias	Data frame with columns iter, method, bias, pval.
param_grid	Sorted numeric vector of parameter values.
xlab	X-axis label.
ylab	Y-axis label. Default "Bias (mean estimate - true)".
main	Plot title.
xfmt	Format string for x-axis labels. Default "%.2f".
legend_pos	Legend position. Default "topleft".
methods	Methods to include. Default: all five.

Value

Invisibly returns NULL; called for the side effect of drawing a boxplot of bias across parameter values.

Examples

```
sweep <- sweep_param("conf_str", c(0.5, 0.8), n_iter = 2,
    n_samples = 100, n_features = 5)
plot_bias_boxplot(sweep$iter_bias, c(0.5, 0.8),
    xlab = "Confounding strength")
```

---

plot_bias_distribution

Baseline bias distribution (single-setting hero plot)

Description

Baseline bias distribution (single-setting hero plot)

Usage

```
plot_bias_distribution(sim_result, methods = iconic_method_order, title = NULL)
```

Arguments

sim_result	Object returned by run_simulation().
methods	Methods to include. Default: all five.
title	Plot title. If NULL a default is constructed.

Value

A ggplot object showing the distribution of bias estimates by method.

Examples

```
sim <- run_simulation(n_iter = 2, n_samples = 100, n_features = 5)
plot_bias_distribution(sim)
```

plot_degradation_surface *Degradation surface figure*

Description

Produces a 3-panel figure showing estimator bias under simultaneous instrument exogeneity violations. Panels: (A) IV2SLS2 NDE bias surface, (B) PGC2Gm NDE bias surface, (C) crossover map showing which estimator has lower |NDE bias| at each grid cell.

Usage

```
plot_degradation_surface(
  results,
  rho_G1_grid = c(0, 0.1, 0.2, 0.3, 0.5),
  rho_G2_grid = c(0, 0.1, 0.2, 0.3, 0.5),
  omega_facet = TRUE,
  file = NULL,
  width = 15,
  height = 5.5
)
```

Arguments

results	Data frame with columns rho_G1, rho_G2, method, NDE_bias, NIE_bias, and derived NDE_abs, NIE_abs. Typically built by sweeping sweep_mediation_param() across a rho_G1 x rho_G2 grid.
rho_G1_grid	Numeric vector of exposure-instrument violation values.
rho_G2_grid	Numeric vector of mediator-instrument violation values.
omega_facet	Logical: when the sensitivity surface swept omega_1 / omega_2 (more than one distinct value), facet the rho_G1 x rho_G2 heatmaps by the omega cell. Default TRUE. Ignored when omega was not swept.
file	Optional file path to save the figure.
width	Figure width in inches.
height	Figure height in inches.

Value

A patchwork ggplot object.

Examples

```
results <- expand.grid(rho_G1 = c(0, 0.2), rho_G2 = c(0, 0.2),
    method = c("IV2SLS2", "PGC2Gm"))
results$NDE_bias <- rnorm(nrow(results), 0, 0.05)
results$NIE_bias <- rnorm(nrow(results), 0, 0.05)
plot_degradation_surface(results, rho_G1_grid = c(0, 0.2),
    rho_G2_grid = c(0, 0.2))
```

plot_estimated_vs_true

Plot estimated effect vs true effect

Description

Plot estimated effect vs true effect

Usage

```
plot_estimated_vs_true(
  sweep_df,
  methods = iconic_method_order,
  title = "Estimated vs True Effect"
)
```

Arguments

sweep_df	Data frame from sweep_param()$summary.
methods	Methods to plot. Default: all five.
title	Plot title.

Value

Invisibly returns NULL; called for the side effect of plotting estimated vs true effects.

Examples

```
sweep <- sweep_param("beta_X", c(0.1, 0.3), n_iter = 2,
    n_samples = 100, n_features = 5)
plot_estimated_vs_true(sweep$summary)
```

plot_estimate_distribution

Boxplot of estimate distributions from run_simulation()

Description

Boxplot of estimate distributions from run_simulation()

Usage

```
plot_estimate_distribution(
  sim_result,
  methods = iconic_method_order,
  title = NULL
)
```

Arguments

sim_result	Object returned by run_simulation().
methods	Methods to include. Default: all five.
title	Plot title.

Value

A ggplot object showing the distribution of estimates by method.

Examples

```
sim <- run_simulation(n_iter = 2, n_samples = 100, n_features = 5)
plot_estimate_distribution(sim)
```

---

plot_estimator_benchmark

Estimator benchmark figure

Description

Produces a 4-panel publication figure benchmarking the eight ICONIC estimators under unmeasured confounding and mediator-outcome confounding. Panels: (A) total-effect bias vs confounding strength, (B) NDE/NIE bias vs confounding strength, (C) NDE/NIE bias vs sample size, (D) NIE Type I error vs confounding strength.

Usage

```
plot_estimator_benchmark(
  panel_a,
  panel_b,
  panel_c,
  panel_d,
  conf_grid = c(0.2, 0.4, 0.6, 0.8, 1),
  n_grid = c(100, 200, 500, 1000),
  file = NULL,
  width = 8,
  height = 6
)
```

Arguments

panel_a	Result of sweep_param("conf_str", ...) for the total-effect panel.
panel_b	Result of sweep_mediation_param("conf_str", ...) for the mediation confounding-strength panel.
panel_c	Result of sweep_mediation_param("n_samples", ...) for the mediation sample-size panel.
panel_d	Result of sweep_mediation_null_by_conf(...) for the Type I error panel.
conf_grid	Numeric vector of confounding-strength values.
n_grid	Numeric vector of sample sizes.

file	Optional file path to save the figure (PDF or PNG).
width	Figure width in inches.
height	Figure height in inches.

Value

A patchwork ggplot object.

Examples

```
# Toy inputs standing in for sweep_param()/sweep_mediation_param() output
methods <- c("UNADJ", "DIRECT", "COCA", "IV2SLS", "PGC",
    "IV2SLS2", "PGC2", "PGC2Gm")
it <- expand.grid(method = methods, pval = c(0.5, 0.8), rep = 1:3)
it$bias <- rnorm(nrow(it), 0, 0.05)
it$NDE_bias <- rnorm(nrow(it), 0, 0.05)
it$NIE_bias <- rnorm(nrow(it), 0, 0.05)
panel <- list(iter_bias = it)
panel_d <- expand.grid(method = methods, conf_str = c(0.5, 0.8))
panel_d$NIE_type1 <- runif(nrow(panel_d), 0, 0.1)
plot_estimator_benchmark(panel, panel, panel, panel_d,
    conf_grid = c(0.5, 0.8), n_grid = c(0.5, 0.8))
```

---

plot_feature_correlation_sweep

Feature correlation sweep figure

Description

Produces a 3-panel figure showing how estimator performance changes as within-module feature correlation increases, modelling co-expression modules in a correlated omics panel.

Usage

```
plot_feature_correlation_sweep(
  sweep_results,
  null_results = NULL,
  file = NULL,
  width = 10,
  height = 12
)
```

Arguments

sweep_results	Results from sweep_mediation_param("feat_cor", ...): a list with $summary (data frame with columns param_value, method, NDE_bias, NIE_bias, NDE_rmse, NIE_rmse).
null_results	Optional results from sweep_mediation_null_by_conf(..., feat_cor = ...) run with feat_cor swept: a data frame with columns feat_cor, method, NIE_type1. If NULL, Panel C is omitted and the figure has 2 panels.
file	Optional file path to save the figure.
width	Figure width in inches. Default 10.
height	Figure height in inches. Default 12.

Details

Panel A: NDE and NIE bias vs feature correlation strength, faceted by estimand. Panel B: NDE and NIE RMSE vs feature correlation strength, faceted by estimand. Panel C: NIE Type I error vs feature correlation strength.

Value

A patchwork ggplot object.

Examples

```
# Toy inputs standing in for sweep_mediation_param("feat_cor", ...) output
sweep <- list(summary = expand.grid(param_value = c(0, 0.5),
    method = c("UNADJ", "IV2SLS2", "PGC2Gm")))
sweep$summary$NDE_bias <- rnorm(nrow(sweep$summary), 0, 0.05)
sweep$summary$NIE_bias <- rnorm(nrow(sweep$summary), 0, 0.05)
sweep$summary$NDE_rmse <- runif(nrow(sweep$summary), 0.05, 0.15)
sweep$summary$NIE_rmse <- runif(nrow(sweep$summary), 0.05, 0.15)
null <- expand.grid(feat_cor = c(0, 0.5),
    method = c("UNADJ", "IV2SLS2", "PGC2Gm"))
null$NIE_type1 <- runif(nrow(null), 0, 0.1)
plot_feature_correlation_sweep(sweep, null)
```

---

plot_gan_diagnostics *Compare real vs synthetic marginals from a trained generator*

Description

For continuous variables, overlays kernel densities of the real training variables against draws from the trained generator. For binary (0/1) variables – detected at training time and stored in the iconic_gan object – uses side-by-side bar charts of the proportion of 1s, which is the appropriate visualisation for discrete data. One panel per variable.

Usage

```
plot_gan_diagnostics(trained_gan, real_data, n_draw = NULL, M_matrix = NULL)
```

Arguments

trained_gan	An iconic_gan from train_gan_on_real_data().
real_data	The data frame the generator was trained on.
n_draw	Synthetic rows to draw for the overlay. Default: number of real rows.
M_matrix	Optional mediator panel (features x samples) for feature-level marginal comparison. When supplied and the GAN carries a feature_texture, an additional panel of density overlays compares real vs copula-sampled mediator marginals.

Value

Called for its side effects; invisibly returns NULL.

Examples

```
if (check_torch_setup()) {
    dat <- load_real_input_data(example = TRUE)
    gan <- train_gan_on_real_data(dat$gan_training_data, epochs = 5,
        verbose = FALSE)
    plot_gan_diagnostics(gan, dat$gan_training_data)
}
```

---

plot_instrument_strength_sweep

Instrument strength sweep figure

Description

Produces a 2-panel figure showing IV2SLS bias and Type I error as a function of first-stage partial F-statistic, with the conventional weak-instrument threshold (F = 10) marked.

Usage

```
plot_instrument_strength_sweep(
  results,
  tau = 0.25,
  file = NULL,
  width = 10,
  height = 4.5
)
```

Arguments

results	Data frame from sweep_instrument_strength().
tau	True total effect used in the sweep.
file	Optional file path to save the figure.
width	Figure width in inches.
height	Figure height in inches.

Value

A patchwork ggplot object.

Examples

```
res <- sweep_instrument_strength(pi_GX_grid = c(0.05, 0.2), n_iter = 2)
plot_instrument_strength_sweep(res)
```

plot_model_selection *Model selection workflow figure*

Description

Produces a 3-panel figure showing the ICONIC model selection workflow on example data. Panels: (A) eligibility table from `iconic_diagnose()`, (B) NDE/NIE forest plot from `iconic_estimate()`, (C) degradation surface heatmap from `iconic_sensitivity()`.

Usage

```
plot_model_selection(
  diagnosis,
  estimate,
  sensitivity,
  recommendation,
  file = NULL,
  width = 12,
  height = 14
)
```

Arguments

`diagnosis`	Result of `iconic_diagnose()`.
`estimate`	Result of `iconic_estimate()`.
`sensitivity`	Result of `iconic_sensitivity()`.
`recommendation`	Result of `iconic_recommend()`.
`file`	Optional file path to save the figure.
`width`	Figure width in inches.
`height`	Figure height in inches.

Value

A `patchwork` ggplot object.

Examples

```
if (check_torch_setup()) {
    data <- iconic_data(X = rnorm(100), Y = matrix(rnorm(100 * 10), 10, 100),
        M = rnorm(100), G = rnorm(100), Gm = rnorm(100),
        W = matrix(rnorm(100 * 10), 10, 100))
    diag <- iconic_diagnose(data)
    est <- iconic_estimate(data, diagnosis = diag)
    sens <- iconic_sensitivity(data, n_iter = 2, gan_epochs = 5,
        rho_G1_grid = c(0, 0.2), rho_G2_grid = c(0, 0.2))
    rec <- iconic_recommend(data, diagnosis = diag, estimate = est,
        sensitivity = sens, auto_sensitivity = FALSE)
    plot_model_selection(diag, est, sens, rec)
}
```

plot_nc_coverage_comparison *NC coverage comparison figure*

Description

Produces a 4-panel figure comparing PGC2, PGC2Gm, and IV2SLS2 as negative-control coverage of each path's confounder composite (omega_1, omega_2) drops. This is the coverage-axis analogue of the instrument-violation degradation surface: it shows how bias accumulates as the proxy panel captures less of the confounder.

Usage

```
plot_nc_coverage_comparison(
  omega1_sweep,
  omega2_sweep,
  t1e_omega2 = NULL,
  file = NULL,
  width = 10,
  height = 12
)
```

Arguments

omega1_sweep	Summary from sweep_mediation_param("omega_1", ...): a list with $summary (columns param_value, method, NDE_bias, NIE_bias).
omega2_sweep	Same, sweeping omega_2.
t1e_omega2	Result of sweep_mediation_null_by_conf(...) run across an omega_2 grid: a data frame with columns omega_2, method, NIE_type1. If NULL, Panel D is omitted.
file	Optional file path to save the figure.
width	Figure width in inches.
height	Figure height in inches.

Details

Panels: (A) NDE bias vs omega_1, (B) NIE bias vs omega_2, (C) NDE/NIE bias vs omega (both paths), (D) NIE Type I error vs omega_2.

Value

A patchwork ggplot object.

Examples

```
# Toy sweeps standing in for sweep_mediation_param("omega_1"/"omega_2", ...)
mk <- function() {
    s <- expand.grid(param_value = c(0.3, 0.7),
        method = c("IV2SLS2", "PGC2", "PGC2Gm"))
    s$NDE_bias <- rnorm(nrow(s), 0, 0.05)
```

```
    s$NIE_bias <- rnorm(nrow(s), 0, 0.05)
    list(summary = s)
}
plot_nc_coverage_comparison(mk(), mk())
```

`plot_nc_validity_diagnostics` *NC validity diagnostics figure (5 panels)*

Description

Produces a 5-panel figure sweeping the empirical NC validity diagnostics. Panels: (A) A1 W perp X | C, (B) A2 W perp G | C, (C) A2' W perp Gm | C, (D) A3 covariance-capture versus true coverage omega (with permutation-null mean and the fraction of replicates with permutation p < 0.05 on a secondary axis), and (E) A3 support R2(Utilde | W) versus omega.

Usage

```
plot_nc_validity_diagnostics(panels, file = NULL, width = 10, height = 6.5)
```

Arguments

`panels`	List returned by `sweep_nc_validity()`.
`file`	Optional file path to save the figure (cairo_pdf, so the omega and Utilde glyphs render).
`width`	Figure width in inches. Default 10.
`height`	Figure height in inches. Default 6.5.

Value

A `patchwork` ggplot object.

Examples

```
# Toy panels standing in for sweep_nc_validity() output
panels <- list(
    panel_a = data.frame(contamination = c(0, 0.1),
        violated_mean = c(0.1, 0.3), violated_sd = 0.05,
        confounding_mean = c(0.1, 0.12), confounding_sd = 0.05),
    panel_b = data.frame(meqtl = c(0, 0.1),
        violated_mean = c(0.1, 0.3), violated_sd = 0.05,
        clean_mean = c(0.1, 0.12), clean_sd = 0.05),
    panel_d = data.frame(eqtl = c(0, 0.1),
        violated_mean = c(0.1, 0.3), violated_sd = 0.05,
        clean_mean = c(0.1, 0.12), clean_sd = 0.05),
    panel_capture_support = data.frame(
        omega = rep(c(0, 0.5), each = 2), rep = 1:2,
        capture_R2 = c(0.02, 0.03, 0.25, 0.30),
        capture_p = c(0.40, 0.60, 0.01, 0.02),
        null_R2 = 0.02,
        support_R2 = c(0.02, 0.03, 0.35, 0.40),
        frac_adds_coverage = 1))
plot_nc_validity_diagnostics(panels)
```

plot_pleiotropy_sweep *Pleiotropy sweep figure*

Description

Produces a faceted figure showing estimator bias and Type I error under horizontal pleiotropy, across three confounding strengths.

Usage

```
plot_pleiotropy_sweep(sensitivity, file = NULL, width = 10, height = 5.5)
```

Arguments

sensitivity	Result of gan_pleiotropy_sensitivity().
file	Optional file path to save the figure.
width	Figure width in inches.
height	Figure height in inches.

Value

A ggplot object.

Examples

```
sens <- gan_pleiotropy_sensitivity(NULL, pleio_grid = c(0, 0.1),
    conf_grid = 0.8, n_iter = 2, n_samples = 100, n_features = 5)
plot_pleiotropy_sweep(sens)
```

plot_power *Plot detection rate (power) vs a swept parameter*

Description

Plot detection rate (power) vs a swept parameter

Usage

```
plot_power(
  sweep_df,
  param_label = "Parameter Value",
  methods = iconic_method_order,
  title = "Detection Rate vs Parameter",
  legend_pos = "bottomleft"
)
```

Arguments

sweep_df	Data frame from sweep_param()$summary.
param_label	X-axis label.
methods	Methods to include.
title	Plot title.
legend_pos	Legend position. Default "bottomleft".

Value

Invisibly returns NULL; called for the side effect of plotting detection rate vs a parameter.

Examples

```
sweep <- sweep_param("conf_str", c(0.5, 0.8), n_iter = 2,
    n_samples = 100, n_features = 5)
plot_power(sweep$summary)
```

---

plot_prospective_analysis

Prospective analysis figure

Description

Produces a 2-panel figure showing expected gains from adding genetic instruments. Panel A: NDE bias vs instrument strength. Panel B: prospective NDE/NIE estimates at target instrument strength.

Usage

```
plot_prospective_analysis(prospect, file = NULL, width = 12, height = 8)
```

Arguments

prospect	Result of iconic_prospect().
file	Optional file path to save the figure.
width	Figure width in inches.
height	Figure height in inches.

Value

A patchwork ggplot object.

Examples

```
if (check_torch_setup()) {
    data <- iconic_data(X = rnorm(100), Y = matrix(rnorm(100 * 10), 10, 100),
        M = rnorm(100))
    pros <- iconic_prospect(data, n_iter = 2, gan_epochs = 5,
        gamma_G_grid = c(0.4, 0.8), run_rho_sweep = FALSE)
    plot_prospective_analysis(pros)
}
```

plot_sensitivity_heatmap *Heatmap of a sensitivity metric across the scenario grid*

Description

Draws a coverage x confounding-strength heatmap of a chosen metric for one estimator, at a fixed number of confounders. Useful for reading off where a method's bias/RMSE blows up.

Usage

```
plot_sensitivity_heatmap(
  sens,
  metric = "rmse",
  method = "IV2SLS",
  k = NULL,
  title = NULL
)
```

Arguments

sens	Object returned by gan_sensitivity().
metric	Column of sens$summary to map (e.g. "rmse", "abs_bias", "power"). Default "rmse".
method	Estimator to display. Default "IV2SLS".
k	Number of confounders to slice on. Default: first in the grid.
title	Optional plot title.

Value

Invisibly returns NULL; called for the side effect of drawing a sensitivity heatmap.

Examples

```
sens <- gan_sensitivity(NULL, conf_grid = c(0.5, 0.8),
    coverage_grid = c(0.3, 0.7), n_iter = 2, n_samples = 100,
    n_features = 5)
plot_sensitivity_heatmap(sens)
```

plot_type1_boxplot *Type I error boxplot per method*

Description

Type I error boxplot per method

Usage

```
plot_type1_boxplot(
  null_result,
  methods = iconic_method_order,
  conf_str = 0.8,
  alpha = 0.05
)
```

Arguments

null_result	Object returned by run_null_sim().
methods	Methods to include. Default: all five.
conf_str	Confounding strength used (for the title). Default 0.80.
alpha	Nominal significance level. Default 0.05.

Value

Invisibly returns NULL; called for the side effect of drawing a boxplot of Type I error rates.

Examples

```
null <- run_null_sim(n_iter = 2, n_samples = 100, n_features = 5)
plot_type1_boxplot(null)
```

---

plot_type1_error *Bar chart of Type I error rates*

Description

Bar chart of Type I error rates

Usage

```
plot_type1_error(null_df, title = "Type I Error Rate by Method")
```

Arguments

null_df	Data frame: either run_null_sim()$rates or a data frame with columns method and type1_error.
title	Plot title.

Value

Invisibly returns NULL; called for the side effect of plotting Type I error rates by method.

Examples

```
t1e <- sweep_null_by_conf(0.8, n_iter = 2, n_samples = 100,
    n_features = 5)
plot_type1_error(t1e)
```

plot_type1_vs_conf *Type I error rate vs confounding strength*

Description

Type I error rate vs confounding strength

Usage

```
plot_type1_vs_conf(
  t1e_df,
  methods = iconic_method_order,
  alpha = 0.05,
  title = "Type I Error Rate vs Confounding Strength"
)
```

Arguments

t1e_df	Data frame returned by sweep_null_by_conf().
methods	Methods to plot. Default: all five.
alpha	Nominal significance level. Default 0.05.
title	Plot title.

Value

Invisibly returns NULL; called for the side effect of drawing Type I error vs confounding strength.

Examples

```
t1e <- sweep_null_by_conf(c(0.5, 0.8), n_iter = 2, n_samples = 100,
    n_features = 5)
plot_type1_vs_conf(t1e)
```

print.iconic_confounding *Print method for iconic_confounding objects*

Description

Print method for iconic_confounding objects

Usage

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

Arguments

x	An iconic_confounding object.
...	Unused.

Value

Invisibly returns x (the iconic_confounding object); called for its side effect of printing a human-readable summary.

Examples

```
set.seed(1)
data <- iconic_data(X = rnorm(100), Y = matrix(rnorm(100 * 10), 10, 100),
    G = rnorm(100), W = matrix(rnorm(100 * 10), 10, 100))
cf <- infer_confounding(data, max_infer_tasks = 5)
print(cf)
```

---

print.iconic_data *Print method for iconic_data objects*

Description

Print method for iconic_data objects

Usage

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

Arguments

x	An iconic_data object.
...	Unused.

Value

Invisibly returns x (the iconic_data object); called for its side effect of printing a human-readable summary.

Examples

```
data <- iconic_data(X = rnorm(50), Y = matrix(rnorm(50 * 5), 5, 50),
    G = rnorm(50), W = matrix(rnorm(50 * 5), 5, 50))
print(data)
```

print.iconic_diagnosis

Print method for iconic_diagnosis objects

Description

Print method for iconic_diagnosis objects

Usage

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

Arguments

x	An iconic_diagnosis object.
...	Unused.

Value

Invisibly returns x (the iconic_diagnosis object); called for its side effect of printing a human-readable summary.

Examples

```
set.seed(1)
data <- iconic_data(X = rnorm(100), Y = matrix(rnorm(100 * 10), 10, 100),
    G = rnorm(100), W = matrix(rnorm(100 * 10), 10, 100))
diag <- iconic_diagnose(data)
print(diag)
```

print.iconic_feature_texture

Print method for iconic_feature_texture objects

Description

Print method for iconic_feature_texture objects

Usage

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

Arguments

x	An iconic_feature_texture object.
...	Unused.

Value

Invisibly returns `x` (the `iconic_feature_texture` object); called for its side effect of printing a human-readable summary.

Examples

```
ft <- train_feature_texture(matrix(rnorm(20 * 100), 20, 100))
print(ft)
```

`print.iconic_gan` *Print method for iconic_gan objects*

Description

Print method for iconic_gan objects

Usage

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

Arguments

`x`	An `iconic_gan` object.
`...`	Unused.

Value

Invisibly returns `x` (the `iconic_gan` object); called for its side effect of printing a human-readable summary.

Examples

```
if (check_torch_setup()) {
    dat <- load_real_input_data(example = TRUE)
    gan <- train_gan_on_real_data(dat$gan_training_data, epochs = 5,
        verbose = FALSE)
    print(gan)
}
```

print.iconic_prospect *Print method for iconic_prospect objects*

Description

Print method for iconic_prospect objects

Usage

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

Arguments

x	An iconic_prospect object.
...	Unused.

Value

Invisibly returns x (the iconic_prospect object); called for its side effect of printing a human-readable summary.

Examples

```
if (check_torch_setup()) {
    data <- iconic_data(X = rnorm(100), Y = matrix(rnorm(100 * 10), 10, 100),
        M = rnorm(100))
    result <- iconic_prospect(data, n_iter = 2, gan_epochs = 5,
        gamma_G_grid = c(0.4, 0.8), run_rho_sweep = FALSE)
    print(result)
}
```

print.iconic_recommendation *Print method for iconic_recommendation objects*

Description

Print method for iconic_recommendation objects

Usage

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

Arguments

x	An iconic_recommendation object.
...	Unused.

Value

Invisibly returns `x` (the `iconic_recommendation` object); called for its side effect of printing a human-readable summary.

Examples

```
data <- iconic_data(X = rnorm(100), Y = matrix(rnorm(100 * 10), 10, 100),
    G = rnorm(100), W = matrix(rnorm(100 * 10), 10, 100))
diag <- iconic_diagnose(data)
est <- iconic_estimate(data, diagnosis = diag)
rec <- iconic_recommend(data, diagnosis = diag, estimate = est,
    auto_sensitivity = FALSE)
print(rec)
```

`print.iconic_sensitivity` *Print method for iconic_sensitivity objects*

Description

Print method for iconic_sensitivity objects

Usage

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

Arguments

`x`	An `iconic_sensitivity` object.
`...`	Unused.

Value

Invisibly returns `x` (the `iconic_sensitivity` object); called for its side effect of printing a human-readable summary.

Examples

```
if (check_torch_setup()) {
    data <- iconic_data(X = rnorm(100), Y = matrix(rnorm(100 * 10), 10, 100),
        M = rnorm(100), G = rnorm(100), Gm = rnorm(100),
        W = matrix(rnorm(100 * 10), 10, 100))
    sens <- iconic_sensitivity(data, n_iter = 2, gan_epochs = 5,
        rho_G1_grid = c(0, 0.2), rho_G2_grid = c(0, 0.2))
    print(sens)
}
```

qc_gwas_sumstats *Standardize and QC GWAS summary statistics*

Description

Renames common GWAS summary-statistic column conventions to the bigsnpr convention (chr, pos, a0, a1, beta, beta_se, n_eff, freq) and applies the standard LDpred2 QC filters: missing/invalid values, extreme effect sizes, ambiguous-strand (A/T and C/G) SNPs, and — when effect allele frequencies are available — the standard-deviation ratio check from the LDpred2 tutorial.

Usage

```
qc_gwas_sumstats(
  sumstats,
  n_eff = NULL,
  drop_ambiguous = TRUE,
  beta_iqr_mult = 10,
  sd_ratio_range = c(0.5, 2),
  sd_y = NULL
)
```

Arguments

sumstats	A data.frame of GWAS summary statistics. Recognized column aliases: chr/chromosome/CHR; pos/base_pair_location/BP; a1/effect_allele/ALT; a0/other_allele/REF; beta/BETA/effect/logOR; beta_se/standard_error/se/SE; freq/effect_allele_frequency/EAF n_eff/n/N; p/p_value/P; rsid/rs_id/SNP/ID.
n_eff	Optional scalar: effective sample size, used when the input has no sample-size column.
drop_ambiguous	Logical: drop A/T and C/G SNPs whose strand cannot be resolved. Default TRUE.
beta_iqr_mult	Numeric: drop SNPs with $\lvert\beta - \mathrm{median}(\beta)\rvert >$ beta_iqr_mult $\times \mathrm{IQR}(\beta)$. Default 10. Set to Inf to skip.
sd_ratio_range	Numeric length-2: acceptable range for the ratio of the summary-statistic-implied genotype SD to the frequency-implied SD, $\sqrt{2f(1-f)}$. SNPs outside the range are dropped. Default c(0.5, 2) (LDpred2 tutorial). Requires freq; skipped with a message when frequencies are absent.
sd_y	Optional scalar: phenotypic SD used in the SD-ratio check. When NULL (default), estimated as the 1st percentile of $\sqrt{0.5(n_{eff} \cdot se^2 + \beta^2)}$ across SNPs, the continuous-trait approximation used in the LDpred2 tutorial.

Value

A list with:

sumstats data.frame of QC-passing variants with standardized columns chr, pos, a0, a1, beta, beta_se, and (when available) n_eff, freq, p, rsid.

qc data.frame with counts of input variants and variants dropped by each filter.

sd_ratio numeric vector of SD ratios for retained variants, or NULL when the check was skipped.

Examples

```
ss <- data.frame(
    chromosome = 1, base_pair_location = 1000 + 0:9 * 1000L,
    effect_allele = rep(c("A", "C"), 5), other_allele = rep(c("G", "T"), 5),
    beta = rnorm(10, 0, 0.05), standard_error = 0.01,
    effect_allele_frequency = runif(10, 0.1, 0.5)
)
out <- qc_gwas_sumstats(ss, n_eff = 50000)
str(out$sumstats)
out$qc
```

---

recommend_estimator *Recommend the preferred estimator from a sensitivity sweep*

Description

Ranks methods by RMSE within each scenario, and identifies an overall pick that is robust across scenarios (smallest worst-case RMSE). UNADJ is excluded from recommendations (it is a bias reference).

Usage

```
recommend_estimator(sens, exclude = c("UNADJ"))
```

Arguments

sens	Object returned by gan_sensitivity().
exclude	Methods to exclude from the recommendation. Default c("UNADJ").

Value

A list with per_scenario (best method + RMSE per scenario), worst_case (max RMSE per method across scenarios), and overall (the method minimising worst-case RMSE).

Examples

```
sens <- gan_sensitivity(NULL, conf_grid = 0.8, coverage_grid = 0.7,
    n_iter = 2, n_samples = 100, n_features = 5)
recommend_estimator(sens)
```

residualize_matrix *Residualize a feature matrix on covariates*

Description

Regresses each feature (row) of x on the covariate design and returns the residuals, using a pre-computed projection for memory- efficient chunked processing of large (e.g. 450k-probe) matrices. This is the workhorse for removing cell composition, batch, and technical factors before extracting negative-control PCs.

Usage

```
residualize_matrix(x, covariates, chunk_size = 5000)
```

Arguments

x	Numeric matrix, features x samples.
covariates	data.frame or matrix with one row per sample (i.e. nrow(covariates) == ncol(x)). An intercept is added internally. Factors should already be expanded to dummies (see stats::model.matrix()).
chunk_size	Integer: number of features processed per block. Default 5000.

Value

A numeric matrix of residuals, features x samples, centered per feature.

Examples

```
x <- matrix(rnorm(100 * 40), nrow = 100,
    dimnames = list(paste0("f", 1:100), paste0("S", 1:40)))
batch <- factor(rep(c("A", "B"), each = 20))
cv <- model.matrix(~batch)[, -1, drop = FALSE]
xr <- residualize_matrix(x, cv)
cor(as.numeric(xr[1, ]), as.numeric(cv))
```

run_mediation_methods *Apply all mediation estimators across features (internal)*

Description

Apply all mediation estimators across features (internal)

Usage

```
run_mediation_methods(
  dat,
  n_features = ncol(dat$Y),
  W_valid = NULL,
  n_cores = 1,
  se_method = "delta",
  n_boot = 500
)
```

Arguments

dat	List returned by generate_toy_data() / run_single_iteration().
n_features	Number of outcome columns to process.
W_valid	Optional: validity-screened W matrix for matrix-bridge PGC.
n_cores	Number of parallel workers. Default 1 (sequential).
se_method	"delta" (default), "bootstrap", or "composite".
n_boot	Number of bootstrap resamples when se_method="bootstrap".

Value

Data frame with columns: feature, method, NDE, NDE_se, NDE_p, NIE, NIE_se, NIE_p. When se_method="composite", also includes alpha_M, alpha_se, beta_M, beta_M_se, var_a, var_b.

---

run_mediation_sim *Run repeated mediation simulations for a single parameter configuration*

Description

Generates n_iter synthetic datasets with mediator-outcome confounding (via mo_confounding), runs all five (or six, when phi > 0) mediation estimators, and summarises bias / RMSE / Type I error for both NDE and NIE.

Usage

```
run_mediation_sim(
  n_iter = 100,
  n_samples = 500,
  n_features = 20,
  n_mediators = 1,
  beta_X = 0.1,
  alpha_M = 0.5,
  beta_M = 0.3,
  conf_str = 0.8,
  w_signal = 0.7,
  mo_confounding = 0.8,
  phi = 0,
  rho_G1 = 0,
  rho_G2 = 0,
  rho_pop = 0,
  lambda_XM = NULL,
  lambda_MY = NULL,
  omega_1 = NULL,
  omega_2 = NULL,
  feat_cor = 0,
  base_seed = 100,
  n_cores = 1
)
```

Arguments

n_iter	Number of simulation replicates. Default 100.
n_samples	Observations per replicate. Default 500.
n_features	Number of outcome and negative-control features. Default 20.
n_mediators	Number of independent mediators. When > 1, each mediator has its own genetic instrument Gm and contributes additively to Y. Default 1 (single mediator, backward compatible).
beta_X	Direct effect of X on Y (true NDE). Default 0.10.
alpha_M	Effect of X on mediator. Default 0.50.
beta_M	Effect of mediator on Y (per-mediator true NIE = alpha_M * beta_M). Default 0.30.
conf_str	Confounding strength delta. Default 0.80.
w_signal	Proxy quality omega. Default 0.70.
mo_confounding	Strength of U1 -> M (mediator-outcome confounding). Default 0.80.
phi	Strength of the mediator instrument Gm -> M. 0 = no mediator instrument (five estimators, backward compatible). > 0 = generates Gm and includes the 2-stage MR estimator (IV2SLS2). Default 0.
rho_G1	Correlation of G1 with conf_XM. Default 0.
rho_G2	Correlation of G2 with conf_MY. Default 0.
rho_pop	Shared population structure. Default 0.
lambda_XM	Optional per-path confounder loading vector (X->M path).
lambda_MY	Optional per-path confounder loading vector (M->Y path).
omega_1	Coverage of conf_XM by W1. NULL = use w_signal.
omega_2	Coverage of conf_MY by W2. NULL = use w_signal.
feat_cor	Within-module correlation for block-diagonal co-expression modules in Y and W. 0 = independent features. Default 0.
base_seed	Starting seed; replicate i uses base_seed + i. Default 100.
n_cores	Number of parallel workers. Default 1.

Details

When are non-default (rho_G1, rho_G2, rho_pop, lambda_XM, lambda_MY, omega_1, omega_2), the two-stage proximal estimators (PGC2, PGC2Gm) are also included.

Value

A list with summary (one row per method with NDE/NIE bias, RMSE, Type I), raw (full per-feature results), true_NDE, true_NIE, and params.

Examples

```
res <- run_mediation_sim(n_iter = 3, n_samples = 100,
    mo_confounding = 0.8)
res$summary
```

`run_methods` *Apply all estimators across features (internal)*

Description

Apply all estimators across features (internal)

Usage

```
run_methods(dat, n_features = ncol(dat$Y), W_valid = NULL, n_cores = 1)
```

Arguments

`dat`	List returned by `generate_toy_data()` / `run_single_iteration()`.
`n_features`	Number of outcome columns to process.
`W_valid`	Optional: validity-screened W matrix for matrix-bridge PGC.
`n_cores`	Number of parallel workers. Default 1 (sequential).

Value

Data frame with columns: feature, method, beta, se, pvalue.

`run_null_mediation_sim`
Run null mediation simulations to estimate Type I error rates

Description

Sets `alpha_M = 0` and `beta_M = 0` (no true NIE) while keeping `beta_X` active (NDE still present). Reports Type I error for both NIE (should be 0 under the null) and NDE (should reflect power for the direct effect).

Usage

```
run_null_mediation_sim(
  n_iter = 200,
  n_samples = 500,
  n_features = 20,
  beta_X = 0.1,
  conf_str = 0.8,
  w_signal = 0.7,
  mo_confounding = 0.8,
  phi = 0,
  rho_G1 = 0,
  rho_G2 = 0,
  rho_pop = 0,
  lambda_XM = NULL,
  lambda_MY = NULL,
  omega_1 = NULL,
```

```
  omega_2 = NULL,
  feat_cor = 0,
  base_seed = 300,
  n_cores = 1,
  alpha = 0.05
)
```

Arguments

n_iter	Number of replicates. Default 200.
n_samples	Observations per replicate. Default 500.
n_features	Features per replicate. Default 20.
beta_X	Direct effect (NDE, still active under null NIE). Default 0.10.
conf_str	Confounding strength delta. Default 0.80.
w_signal	Proxy quality omega. Default 0.70.
mo_confounding	Strength of U1 -> M. Default 0.80.
phi	Strength of the mediator instrument Gm -> M. 0 = no mediator instrument. Default 0.
rho_G1	Correlation of G1 with conf_XM. Default 0.
rho_G2	Correlation of G2 with conf_MY. Default 0.
rho_pop	Shared population structure. Default 0.
lambda_XM	Optional per-path confounder loading vector (X->M path).
lambda_MY	Optional per-path confounder loading vector (M->Y path).
omega_1	Coverage of conf_XM by W1. NULL = use w_signal.
omega_2	Coverage of conf_MY by W2. NULL = use w_signal.
feat_cor	Within-module correlation for block-diagonal co-expression modules in Y and W. 0 = independent features. Default 0.
base_seed	Seed offset. Default 300.
n_cores	Parallel workers. Default 1.
alpha	Significance threshold. Default 0.05.

Value

A list with `rates` (data frame: method, NIE_type1, NDE_type1) and `raw` (full results).

Examples

```
null <- run_null_mediation_sim(n_iter = 2, n_samples = 100,
    n_features = 5, mo_confounding = 0.8, phi = 0.8)
null$rates
```

`run_null_sim` *Run null simulations to estimate Type I error rates*

Description

Run null simulations to estimate Type I error rates

Usage

```
run_null_sim(
  n_iter = 200,
  n_samples = 500,
  n_features = 20,
  conf_str = 0.8,
  w_signal = 0.7,
  feat_cor = 0,
  base_seed = 300,
  n_cores = 1,
  alpha = 0.05
)
```

Arguments

`n_iter`	Number of replicates. Default 200.
`n_samples`	Observations per replicate. Default 500.
`n_features`	Features per replicate. Default 20.
`conf_str`	Confounding strength delta. Default 0.80.
`w_signal`	Proxy quality omega. Default 0.70.
`feat_cor`	Within-module feature correlation. Default 0.
`base_seed`	Seed offset. Default 300.
`n_cores`	Parallel workers. Default 1.
`alpha`	Significance threshold. Default 0.05.

Value

A list with rates (data frame) and raw (full results).

Examples

```
null <- run_null_sim(n_iter = 2, n_samples = 100, n_features = 5)
null$rates
```

`run_simulation` *Run repeated simulations for a single parameter configuration*

Description

Run repeated simulations for a single parameter configuration

Usage

```
run_simulation(
  n_iter = 100,
  n_samples = 500,
  n_features = 20,
  beta_X = 0.1,
  alpha_M = 0.5,
  beta_M = 0.3,
  conf_str = 0.8,
  w_signal = 0.7,
  feat_cor = 0,
  base_seed = 100,
  n_cores = 1
)
```

Arguments

`n_iter`	Number of simulation replicates. Default 100.
`n_samples`	Observations per replicate. Default 500.
`n_features`	Number of outcome and negative-control features. Default 20.
`beta_X`	Direct effect of X on Y. Default 0.10.
`alpha_M`	Effect of X on mediator. Default 0.50.
`beta_M`	Effect of mediator on Y. Default 0.30.
`conf_str`	Confounding strength delta. Default 0.80.
`w_signal`	Proxy quality omega. Default 0.70.
`feat_cor`	Within-module correlation for block-diagonal co-expression modules in Y and W. 0 = independent features. Default 0.
`base_seed`	Starting seed; replicate i uses base_seed + i. Default 100.
`n_cores`	Number of parallel workers. Default 1.

Value

A list with raw, summary, iter_bias, true_total, params.

Examples

```
res <- run_simulation(n_iter = 3, n_samples = 100, beta_X = 0.1,
    conf_str = 0.8)
res$summary
```

`run_single_iteration` *Generate one synthetic dataset under the generalised SCM*

Description

Generate one synthetic dataset under the generalised SCM

Usage

```
run_single_iteration(
  trained_gan = NULL,
  n_synthetic_samples = 500,
  n_features = 20,
  n_confounders = 1,
  n_mediators = 1,
  beta_X = 0.1,
  alpha_M = 0.5,
  beta_M = 0.3,
  effect_size = NULL,
  conf_strength = 0.8,
  coverage = 1,
  captured = NULL,
  nc_model = "proxy",
  nc_params = list(),
  mo_confounding = 0,
  pleio = 0,
  phi = 0,
  gamma_G = 0.6,
  rho_G1 = 0,
  rho_G2 = 0,
  rho_pop = 0,
  lambda_XM = NULL,
  lambda_MY = NULL,
  omega_1 = NULL,
  omega_2 = NULL,
  feat_cor = 0,
  separate_U = NULL,
  u_strength = NULL,
  w_coverage_profile = NULL,
  MMExp = 1,
  MMOut = 1,
  MMCon = 1,
  MMCpG = 1,
  outcome_type = c("continuous", "survival", "binary"),
  surv_h0 = 0.1,
  surv_event_frac = 0.6,
  surv_censor_rate = NULL,
  bin_prev = 0.5,
  seed = NULL
)
```

Arguments

trained_gan	Optional iconic_gan from train_gan_on_real_data(), supplying realistic covariate/outcome texture. If NULL, default synthetic covariates are used.
n_synthetic_samples	Sample size. Default 500.
n_features	Number of outcome (and control) features. Default 20.
n_confounders	Number of latent confounders k. Default 1 (backward-compatible single-confounder model).
n_mediators	Number of independent mediators. When > 1, each mediator has its own genetic instrument Gm and contributes additively to Y. Default 1 (single mediator, backward compatible).
beta_X	Direct effect of X on Y (true NDE). Default 0.10.
alpha_M	Effect of X on the mediator M. Default 0.50.
beta_M	Effect of M on Y. Default 0.30.
effect_size	Optional shortcut: if non-NULL, sets a pure direct total effect (beta_X = effect_size, alpha_M = beta_M = 0). Use 0 for null (Type I error) simulations. Default NULL (use the mediation parameters).
conf_strength	Overall confounding strength (analogue of conf_str); scales the confounder loadings into X and Y. Default 0.80.
coverage	How well the negative controls span the confounder subspace, in [0, 1] (passed to nc_model). Default 1.
captured	Integer indices of the confounders the negative controls see (passed to nc_model). Default all k.
nc_model	Negative-control model: a function (U, covariates, params) -> W, or a registered name ("proxy", "cpg"). Default '"proxy".
nc_params	Extra named parameters forwarded to nc_model.
mo_confounding	Strength of U1 -> M (mediator-outcome confounding). 0 = no M-O confounding (original DGP). Default 0. When > 0, the first confounder U[,1] also affects M, creating the M-O confounding that the mediation estimators are benchmarked against.
pleio	Strength of a direct G -> Y path (horizontal pleiotropy), violating the exclusion restriction. 0 = no pleiotropy (valid instrument, original DGP). Default 0. When > 0, G affects Y directly in addition to through X, allowing benchmarking of IV/2SLS under instrument invalidity.
phi	Strength of the mediator instrument Gm -> M. 0 = no mediator instrument (original DGP, no Gm generated). Default 0. When > 0, a valid instrument for M is generated (independent of U and G, no direct path to Y), enabling point identification of NDE/NIE via fit_iv2sls_mediation2() even under M-O confounding.
gamma_G	Strength of the exposure instrument G -> X. Default 0.6.
rho_G1	Correlation of G1 with conf_XM (instrument exogeneity violation). Default 0.
rho_G2	Correlation of G2 with conf_MY (instrument exogeneity violation). Default 0.
rho_pop	Shared population structure inducing G1-G2 correlation Default 0.
lambda_XM	Optional length-k loading vector giving each confounder's weight on the X->M backdoor path. NULL (default) = shared loadings.

`lambda_MY` Optional length-k loading vector giving each confounder's weight on the M->Y backdoor path. NULL (default) = shared loadings.

`omega_1` Coverage of conf_XM by W1. NULL = use `coverage`.

`omega_2` Coverage of conf_MY by W2. NULL = use `coverage`.

`feat_cor` Within-module feature correlation. When > 0 and the GAN does not provide feature_correlations, a block-diagonal correlation matrix is used for the NC and outcome noise. GAN-learned correlations take precedence. Default 0.

`separate_U` Defunct. Passing a value errors with a message pointing to the replacement per-path loading vectors `lambda_XM` / `lambda_MY`. Retained in the signature only to catch and redirect old calls.

`u_strength` Numeric vector: per-confounder strength profile (length k). Default NULL → rep(1, k) (equal strength, backward compatible). Recycled to length k and normalized so the total confounding budget is unchanged.

`w_coverage_profile`
A list with optional `w1` and `w2` numeric vectors: per-control coverage of conf_XM / conf_MY (length n_features). Default NULL → scalar omega applied uniformly.

`MMExp`, `MMOut`, `MMCon`, `MMCpG`
Per-pathway confounding multipliers (exposure, outcome, controls, methylation). Default 1.

`outcome_type` `"continuous"` (default), `"survival"`, or `"binary"`. When `"survival"`, the linear predictor is converted to `surv_time` and `surv_event` via an exponential PH model. When `"binary"`, the linear predictor is converted to a 0/1 outcome `y_bin` via a logistic (Bernoulli) model; true effects are on the conditional log-odds-ratio scale.

`surv_h0` Baseline hazard for the survival DGP. Default 0.1.

`surv_event_frac`
Target fraction of observed events. Default 0.6.

`surv_censor_rate`
Explicit censoring rate. Default NULL.

`bin_prev` Target prevalence (marginal event probability) for the binary DGP. Default 0.5.

`seed` Optional RNG seed.

Value

A named list matching `generate_toy_data()` — `X`, `G` (n x n_features), `Y`, `W`, `U1`, `M`, `synthetic_data`, `true_total`, `true_NDE`, `true_NIE` — plus `U` (full n x k confounder matrix), `genetic_instrument`, `successful_features`, `failed_features`, and `params`. When `phi` > `0`, also includes `Gm` (numeric vector, length n, or `n_mediators x n` matrix when `n_mediators` > 1). When any parameter is non-default, also includes `G1`, `G2`, `W1`, `W2`, `conf_XM`, `conf_MY`, and (when `rho_pop` > `0`) `P`. When `n_mediators` > 1, `M` is an `n_mediators x n` matrix.

Examples

```
dat <- run_single_iteration(NULL, n_synthetic_samples = 100,
    n_features = 5, n_confounders = 1, seed = 1)
analyze_methods_robust(dat)
```

sample_feature_texture *Draw synthetic feature vectors from a trained feature texture model*

Description

Samples from the Gaussian copula: draws n_samples from MVN(0, copula_cor), then transforms each feature through its inverse marginal CDF. The result is centered and scaled to zero mean and unit variance per feature, so the texture acts as noise that the structural signal in `run_single_iteration()` provides the mean for.

Usage

```
sample_feature_texture(feature_texture, n_samples, n_features = NULL)
```

Arguments

feature_texture	An `iconic_feature_texture` object from `train_feature_texture()`.
n_samples	Number of synthetic samples to draw.
n_features	Target number of features. If NULL, uses the number of features in the training data. If larger, additional features are drawn by sampling existing columns with replacement and adding independent noise. If smaller, the first n_features are used.

Value

A `n_features` x `n_samples` matrix of synthetic texture values, centered and scaled per feature.

Examples

```
M <- matrix(rnorm(30 * 200), 30, 200) # 30 transcripts, 200 samples
ft <- train_feature_texture(M)
draws <- sample_feature_texture(ft, 500, n_features = 20)
```

sample_texture *Draw synthetic base rows from a trained texture model*

Description

Binary columns (detected at training time) are rounded to $0, 1$ after de-normalisation, and one-hot dummy groups are made mutually exclusive (the column with the highest pre-rounding value wins per row).

Usage

```
sample_texture(trained_gan, n)
```

Arguments

trained_gan	An iconic_gan object from train_gan_on_real_data().
n	Number of rows to draw.

Value

A data frame of n rows with the trained columns, on the original (de-normalised) scale. Binary columns contain only 0/1 values.

Examples

```
if (check_torch_setup()) {
    dat <- load_real_input_data(example = TRUE)
    gan <- train_gan_on_real_data(dat$gan_training_data, epochs = 5,
        verbose = FALSE)
    head(sample_texture(gan, 5))
}
```

---

scenario_manifest *Scenario manifest: truth and parameter ranges for a simulation*

Description

Returns a reader-orienting summary of a simulation scenario: the ground-truth estimands (NDE, NIE, total effect), the modifiable parameters with their swept ranges, and the fixed parameters with their values. Intended to be rendered as a table preceding simulation results in the manuscript (, #582: "state the truth and the parameter ranges up front to orient the reader").

Usage

```
scenario_manifest(
  dat_or_params,
  conf_grid = NULL,
  coverage_grid = NULL,
  mo_confounding_grid = NULL,
  phi_grid = NULL,
  rho_G1_grid = NULL,
  rho_G2_grid = NULL,
  rho_pop_grid = NULL,
  omega_1_grid = NULL,
  omega_2_grid = NULL
)
```

Arguments

dat_or_params	A dat list from generate_toy_data() or a named list of DGP parameters.
conf_grid	Swept values of conf_str (confounding strength). Default NULL.
coverage_grid	Swept values of NC coverage (w_signal / omega). Default NULL.
mo_confounding_grid	Swept values of mediator-outcome confounding strength. Default NULL.

phi_grid	Swept values of the mediator-instrument strength phi. Default NULL.
rho_G1_grid	Swept values of the G1-conf_XM correlation. Default NULL.
rho_G2_grid	Swept values of the G2-conf_MY correlation. Default NULL.
rho_pop_grid	Swept values of the population-stratification correlation. Default NULL.
omega_1_grid	Swept values of omega_1 (coverage of conf_XM by W1). Default NULL.
omega_2_grid	Swept values of omega_2 (coverage of conf_MY by W2). Default NULL.

Details

dat_or_params may be either:

- a list returned by generate_toy_data(), in which case the estimands and fixed parameters are read from the object (true_NDE, true_NIE, true_total, n, n_features, n_mediators, lambda_XM, lambda_MY, feat_cor); or
- a bare named list of DGP parameters (beta_X, alpha_M, beta_M, n_mediators, n, n_features, lambda_XM, lambda_MY, feat_cor, ...), in which case the estimands are recomputed as NDE = beta_X, NIE = n_mediators * alpha_M * beta_M, total = NDE + NIE.

The *_grid arguments record the swept ranges for the modifiable parameters. When a grid argument is omitted or NULL, the corresponding modifiable parameter is reported with its scalar value only (no range).

Value

A named list with elements:

estimands Named numeric vector: NDE, NIE, total.

modifiable_parameters Data frame with columns parameter, value, swept_range. value is the scalar used (or NA when only a grid was supplied); swept_range is a comma-separated string of grid values, or NA when the parameter was held fixed.

fixed_parameters Data frame with columns parameter, value.

Examples

```
dat <- generate_toy_data(n = 200, phi = 0.8, lambda_XM = c(1, 0), lambda_MY = c(0, 1),
    omega_1 = 0.7, omega_2 = 0.7, seed = 42)
scenario_manifest(dat, conf_grid = c(0.3, 0.8),
    coverage_grid = c(0.3, 0.7, 1))
```

---

score_pgs_panel *Score a published polygenic-score panel on a dosage matrix*

Description

Applies a published set of per-variant weights (e.g. a PGS Catalog scoring file, which is already LD-aware) directly to a genotype dosage matrix. Matching is by chr:pos:ref:alt key (with automatic allele-flip handling: when the genotype stores the opposite orientation, the weight is negated) or by rsID when present.

Usage

```
score_pgs_panel(weights, genotypes, impute_mean = TRUE, scale_score = TRUE)
```

Arguments

weights	A data.frame of variant weights. Recognized columns: chr/chr_name, pos/chr_position, a1/effect_allele, a0/other_allele, weight/effect_weight, and optionally rsid. PGS Catalog scoring files can be read directly with utils::read.delim() (skip the header comment lines).
genotypes	A numeric dosage matrix (variants x samples, dosages in 0-2) whose rownames are either chr:pos:ref:alt keys or rsIDs matching weights. A bigsnpr::snp_attach() bigSNP object is also accepted (requires bigsnpr).
impute_mean	Logical: mean-impute missing dosages per variant. Default TRUE.
scale_score	Logical: scale the resulting score to mean 0 / sd 1. Default TRUE.

Value

A list with:

score named numeric vector of per-sample scores.

matched data.frame of matched variants with the weights used (after any flip correction).

n_input number of weight-panel variants.

Examples

```
# Toy dosage matrix: 4 variants x 50 samples
dos <- matrix(rbinom(4 * 50, 2, 0.3), nrow = 4,
    dimnames = list(c("1:100:A:G", "1:200:C:T",
        "1:300:G:A", "2:400:T:C"),
    paste0("S", 1:50)))
wts <- data.frame(chr = c(1, 1, 2), pos = c(100, 300, 400),
    a1 = c("A", "G", "T"), a0 = c("G", "A", "C"),
    weight = c(0.1, -0.2, 0.05))
out <- score_pgs_panel(wts, dos)
plot(out$score)
```

---

simulate_single_genetic_instrument

Simulate a single genetic instrument

Description

Draws one standardised genetic instrument for the exposure, as an additive score over n_snps bialleic SNPs (dosages Binomial(2, maf)). Mirrors the single-instrument (PRS) design used throughout the SCENIC framework.

Usage

```
simulate_single_genetic_instrument(n, n_snps = 20, maf = 0.3, seed = NULL)
```

Arguments

n	Number of samples.
n_snps	SNPs contributing to the score. Default 20.
maf	Minor-allele frequency. Default 0.3.
seed	Optional RNG seed.

Value

List with G (standardised instrument, length n), dosages (n x n_snps), and maf.

Examples

```
g <- simulate_single_genetic_instrument(100, n_snps = 10, seed = 1)
length(g$G)
```

---

summarise_mediation_results

Summarise mediation simulation results across features (internal)

Description

Iterates over the methods present in combined (rather than a hardcoded list), so it handles both the 5-method (no Gm) and 6-method (with Gm) cases.

Usage

```
summarise_mediation_results(combined, true_NDE, true_NIE)
```

Arguments

combined	Data frame from run_mediation_methods().
true_NDE	Scalar true natural direct effect.
true_NIE	Scalar true natural indirect effect.

Value

Data frame with one row per method: NDE/NIE mean, bias, sd, rmse, Type I error rates, and counts.

summarise_results *Summarise simulation results across features (internal)*

Description

Summarise simulation results across features (internal)

Usage

```
summarise_results(combined, true_total)
```

Arguments

combined	Data frame from run_methods().
true_total	Scalar true total causal effect.

Value

Data frame with one row per method: mean, median, sd, bias, abs_bias, rmse, power, n.

summary.iconic_diagnosis *Summary method for iconic_diagnosis objects*

Description

Prints the full diagnosis summary (same as print()).

Usage

```
## S3 method for class 'iconic_diagnosis'
summary(object, ...)
```

Arguments

object	An iconic_diagnosis object.
...	Unused.

Value

Invisibly returns object.

Examples

```
set.seed(1)
data <- iconic_data(X = rnorm(100), Y = matrix(rnorm(100 * 10), 10, 100),
    G = rnorm(100), W = matrix(rnorm(100 * 10), 10, 100))
diag <- iconic_diagnose(data)
summary(diag)
```

summary.iconic_prospect

Summary method for iconic_prospect objects

Description

Prints the full prospective summary (same as `print()`).

Usage

```
## S3 method for class 'iconic_prospect'
summary(object, ...)
```

Arguments

`object`	An `iconic_prospect` object.
`...`	Unused.

Value

Invisibly returns `object`.

Examples

```
if (check_torch_setup()) {
    data <- iconic_data(X = rnorm(100), Y = matrix(rnorm(100 * 10), 10, 100),
        M = rnorm(100))
    result <- iconic_prospect(data, n_iter = 2, gan_epochs = 5,
        gamma_G_grid = c(0.4, 0.8), run_rho_sweep = FALSE)
    summary(result)
}
```

summary.iconic_recommendation

Summary method for iconic_recommendation objects

Description

Prints the full recommendation summary (same as `print()`).

Usage

```
## S3 method for class 'iconic_recommendation'
summary(object, ...)
```

Arguments

`object`	An `iconic_recommendation` object.
`...`	Unused.

Value

Invisibly returns `object`.

Examples

```
data <- iconic_data(X = rnorm(100), Y = matrix(rnorm(100 * 10), 10, 100),
    G = rnorm(100), W = matrix(rnorm(100 * 10), 10, 100))
diag <- iconic_diagnose(data)
est <- iconic_estimate(data, diagnosis = diag)
rec <- iconic_recommend(data, diagnosis = diag, estimate = est,
    auto_sensitivity = FALSE)
summary(rec)
```

---

summary.iconic_sensitivity

Summary method for iconic_sensitivity objects

Description

Prints the full sensitivity summary (same as `print()`).

Usage

```
## S3 method for class 'iconic_sensitivity'
summary(object, ...)
```

Arguments

`object`	An `iconic_sensitivity` object.
`...`	Unused.

Value

Invisibly returns `object`.

Examples

```
if (check_torch_setup()) {
    data <- iconic_data(X = rnorm(100), Y = matrix(rnorm(100 * 10), 10, 100),
        M = rnorm(100), G = rnorm(100), Gm = rnorm(100),
        W = matrix(rnorm(100 * 10), 10, 100))
    sens <- iconic_sensitivity(data, n_iter = 2, gan_epochs = 5,
        rho_G1_grid = c(0, 0.2), rho_G2_grid = c(0, 0.2))
    summary(sens)
}
```

sweep_instrument_strength *Sweep instrument strength*

Description

Runs a custom DGP that varies the G->X coefficient (pi_GX) and records IV2SLS performance binned by first-stage partial F. Uses `fit_iv2sls()` with `min_f = 0` (no weak-IV guard) so bias is visible across all instrument strengths.

Usage

```
sweep_instrument_strength(
  pi_GX_grid = c(0.02, 0.05, 0.1, 0.15, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8),
  n_iter = 50,
  n = 200,
  k = 1,
  conf_str = 0.8,
  tau = 0.25,
  coverage = 0.7,
  n_cores = 1
)
```

Arguments

pi_GX_grid	Numeric vector of G->X coefficients to sweep.
n_iter	Number of replications per grid point.
n	Sample size.
k	Number of confounders.
conf_str	Confounding strength.
tau	True total effect.
coverage	Negative-control coverage.
n_cores	Number of parallel workers for the replicate loops. Default 1 (sequential). Uses `parallel::mclapply` on Unix and a PSOCK cluster on Windows.

Value

A data frame with columns `pi_GX`, `arm`, `iter`, `partial_F`, `beta`, `se`, `pvalue`, `rejected`.

Examples

```
res <- sweep_instrument_strength(pi_GX_grid = c(0.05, 0.2), n_iter = 2)
head(res)
```

`sweep_mediation_null_by_conf` *Sweep mediation Type I error across confounding strength levels*

Description

Sweep mediation Type I error across confounding strength levels

Usage

```
sweep_mediation_null_by_conf(
  conf_grid = c(0.2, 0.4, 0.6, 0.8, 1),
  n_iter = 100,
  n_samples = 500,
  n_features = 20,
  w_signal = 0.7,
  mo_confounding = 0.8,
  phi = 0,
  rho_G1 = 0,
  rho_G2 = 0,
  rho_pop = 0,
  lambda_XM = NULL,
  lambda_MY = NULL,
  omega_1 = NULL,
  omega_2 = NULL,
  feat_cor = 0,
  base_seed = 900,
  n_cores = 1,
  alpha = 0.05
)
```

Arguments

`conf_grid`	Numeric vector of confounding strength values. Default c(0.2, 0.4, 0.6, 0.8, 1.0).
`n_iter`	Replicates per conf_str value. Default 100.
`n_samples`	Observations per replicate. Default 500.
`n_features`	Features per replicate. Default 20.
`w_signal`	Proxy quality omega. Default 0.70.
`mo_confounding`	Strength of U1 -> M. Default 0.80.
`phi`	Strength of the mediator instrument Gm -> M. 0 = no mediator instrument. Default 0.
`rho_G1`	Correlation of G1 with conf_XM. Default 0.
`rho_G2`	Correlation of G2 with conf_MY. Default 0.
`rho_pop`	Shared population structure. Default 0.
`lambda_XM`	Optional per-path confounder loading vector (X->M path).
`lambda_MY`	Optional per-path confounder loading vector (M->Y path).

omega_1	Coverage of conf_XM by W1. NULL = use w_signal.
omega_2	Coverage of conf_MY by W2. NULL = use w_signal.
feat_cor	Within-module feature correlation. Default 0.
base_seed	Seed offset. Default 900.
n_cores	Parallel workers. Default 1.
alpha	Significance threshold. Default 0.05.

Value

A data frame with columns: conf_str, method, NIE_type1, NDE_type1.

Examples

```
t1e <- sweep_mediation_null_by_conf(c(0.2, 0.8), n_iter = 3,
    n_samples = 100)
head(t1e)
```

---

sweep_mediation_param *Sweep a single mediation simulation parameter across a grid*

Description

Sweep a single mediation simulation parameter across a grid

Usage

```
sweep_mediation_param(
  param,
  param_grid,
  n_iter = 100,
  n_samples = 500,
  n_features = 20,
  n_mediators = 1,
  beta_X = 0.1,
  alpha_M = 0.5,
  beta_M = 0.3,
  conf_str = 0.8,
  w_signal = 0.7,
  mo_confounding = 0.8,
  phi = 0,
  rho_G1 = 0,
  rho_G2 = 0,
  rho_pop = 0,
  lambda_XM = NULL,
  lambda_MY = NULL,
  omega_1 = NULL,
  omega_2 = NULL,
  feat_cor = 0,
  u_strength = NULL,
  w_coverage_profile = NULL,
```

```
  base_seed = 0,
  n_cores = 1
)
```

Arguments

param	Parameter to vary: one of "beta_X", "conf_str", "w_signal", "alpha_M", "beta_M", "n_samples", "mo_confounding", "phi", "rho_G1", "rho_G2", "rho_pop", "omega_1", "omega_2", "feat_cor".
param_grid	Numeric vector of values to sweep.
n_iter	Replicates per grid point. Default 100.
n_samples	Observations per replicate. Default 500.
n_features	Features per replicate. Default 20.
n_mediators	Number of independent mediators. Default 1.
beta_X	Baseline direct effect. Default 0.10.
alpha_M	Baseline mediator path. Default 0.50.
beta_M	Baseline mediator effect. Default 0.30.
conf_str	Baseline confounding strength. Default 0.80.
w_signal	Baseline proxy quality. Default 0.70.
mo_confounding	Baseline M-O confounding. Default 0.80.
phi	Baseline mediator-instrument strength. 0 = no mediator instrument. Default 0.
rho_G1	Baseline G1-conf_XM correlation. Default 0.
rho_G2	Baseline G2-conf_MY correlation. Default 0.
rho_pop	Baseline population structure. Default 0.
lambda_XM	Optional per-path confounder loading vector (X->M path).
lambda_MY	Optional per-path confounder loading vector (M->Y path).
omega_1	Baseline W1 coverage. NULL = use w_signal.
omega_2	Baseline W2 coverage. NULL = use w_signal.
feat_cor	Baseline within-module feature correlation. Default 0.
u_strength	Numeric vector: per-confounder strength scaling. NULL = uniform (legacy behavior). See generate_toy_data().
w_coverage_profile	A list with w1/w2 per-control coverage vectors. NULL = uniform coverage. See generate_toy_data().
base_seed	Seed offset. Default 0.
n_cores	Parallel workers. Default 1.

Value

A list with summary (data frame) and iter_bias (per-iteration NDE/NIE bias).

Examples

```
res <- sweep_mediation_param("conf_str", c(0.2, 0.8), n_iter = 3,
    n_samples = 100)
res$summary
```

`sweep_nc_validity` *Sweep negative-control validity diagnostics*

Description

Runs simulation sweeps for the empirical NC validity diagnostics and returns data frames ready for plotting with `plot_nc_validity_diagnostics()`. Panels: (A) A1 W perp X | C, (B) A2 W perp G | C, (C) A2' W perp Gm | C, (D) A3 covariance-capture versus true coverage omega, (E) A3 support R2(Utilde | W) versus omega.

Usage

```
sweep_nc_validity(
  n_samples = 500,
  n_iter = 50,
  phi_val = 0.8,
  contam_grid = c(0, 0.05, 0.1, 0.15, 0.2, 0.3, 0.5),
  meqtl_grid = c(0, 0.05, 0.1, 0.15, 0.2, 0.3, 0.5),
  eqtl_grid = c(0, 0.05, 0.1, 0.15, 0.2, 0.3, 0.5),
  omega_grid = c(0, 0.1, 0.2, 0.3, 0.5, 0.7, 0.9, 1),
  n_perm = 200,
  cs_confounders = 2,
  include_a3_grid = FALSE,
  k_grid = c(1, 2, 3),
  n_valid_grid = c(1, 2, 3),
  n_cores = 1
)
```

Arguments

`n_samples`	Number of synthetic samples per iteration.
`n_iter`	Number of replications per sweep point.
`phi_val`	Mediator-instrument strength (for the A2' panel and the capture/support sweeps).
`contam_grid`	X->W contamination strength grid (A1).
`meqtl_grid`	G->W (meQTL) strength grid (A2).
`eqtl_grid`	Gm->W (eQTL) strength grid (A2').
`omega_grid`	True negative-control coverage grid for the capture (D) and support (E) sweeps. Includes 0 by default so the permutation-null calibration is visible.
`n_perm`	Permutations per replicate for the capture test. Default 200, matching `iconic_diagnose()`.
`cs_confounders`	Number of latent confounders for the capture/support sweeps (distinct loadings). Default 2.
`include_a3_grid`	Logical; also run the legacy A3 dimensional grid (PGC bias over n_valid x k). Default FALSE (no longer plotted).
`k_grid, n_valid_grid`	Grids for the legacy A3 dimensional sweep.
`n_cores`	Number of parallel workers for the replicate loops. Default 1 (sequential). Uses `parallel::mclapply` on Unix and a PSOCK cluster on Windows.

Value

A list with elements panel_a, panel_b, panel_d, panel_capture_support (per-replicate raw), and optionally panel_c when include_a3_grid = TRUE.

Examples

```
## A deliberately tiny sweep (two grid points per panel, two
## replicates) so the example runs in a few seconds. Real sweeps use
## the defaults (n_samples = 500, n_iter = 50, n_perm = 200).
panels <- sweep_nc_validity(n_samples = 60, n_iter = 2,
    contam_grid = c(0, 0.3), meqtl_grid = c(0, 0.3),
    eqtl_grid = c(0, 0.3), omega_grid = c(0, 0.8), n_perm = 10)
panels$panel_a
head(panels$panel_capture_support)
```

---

sweep_null_by_conf *Sweep Type I error rate across confounding strength levels*

Description

Sweep Type I error rate across confounding strength levels

Usage

```
sweep_null_by_conf(
  conf_grid = c(0.2, 0.4, 0.6, 0.8, 1),
  n_iter = 100,
  n_samples = 500,
  n_features = 20,
  w_signal = 0.7,
  feat_cor = 0,
  base_seed = 900,
  n_cores = 1,
  alpha = 0.05
)
```

Arguments

conf_grid	Numeric vector of confounding strength values. Default c(0.2, 0.4, 0.6, 0.8, 1.0).
n_iter	Replicates per conf_str value. Default 100.
n_samples	Observations per replicate. Default 500.
n_features	Features per replicate. Default 20.
w_signal	Proxy quality omega. Default 0.70.
feat_cor	Within-module feature correlation. Default 0.
base_seed	Seed offset. Default 900.
n_cores	Parallel workers. Default 1.
alpha	Significance threshold. Default 0.05.

Value

A data frame with columns: conf_str, method, type1_error.

Examples

```
t1e <- sweep_null_by_conf(c(0.2, 0.8), n_iter = 3, n_samples = 100)
plot_type1_vs_conf(t1e)
```

`sweep_param` *Sweep a single simulation parameter across a grid*

Description

Sweep a single simulation parameter across a grid

Usage

```
sweep_param(
  param,
  param_grid,
  n_iter = 100,
  n_samples = 500,
  n_features = 20,
  beta_X = 0.1,
  alpha_M = 0.5,
  beta_M = 0.3,
  conf_str = 0.8,
  w_signal = 0.7,
  feat_cor = 0,
  base_seed = 0,
  n_cores = 1
)
```

Arguments

`param`	Parameter to vary: one of "beta_X", "conf_str", "w_signal", "alpha_M", "beta_M", "n_samples", "feat_cor".
`param_grid`	Numeric vector of values to sweep.
`n_iter`	Replicates per grid point. Default 100.
`n_samples`	Observations per replicate. Default 500.
`n_features`	Features per replicate. Default 20.
`beta_X`	Baseline direct effect. Default 0.10.
`alpha_M`	Baseline mediator path. Default 0.50.
`beta_M`	Baseline mediator effect. Default 0.30.
`conf_str`	Baseline confounding strength. Default 0.80.
`w_signal`	Baseline proxy quality. Default 0.70.
`feat_cor`	Baseline within-module feature correlation. Default 0.
`base_seed`	Seed offset. Default 0.
`n_cores`	Parallel workers. Default 1.

Value

A list with summary (data frame) and iter_bias (data frame).

Examples

```
res <- sweep_param("conf_str", c(0.2, 0.8), n_iter = 3, n_samples = 100)
res$summary
```

`train_feature_texture` *Train a feature-level texture model for the mediator panel*

Description

Learns the joint distribution of a features x samples mediator matrix using a Gaussian copula with flexible marginal distributions. For each feature, the marginal is fitted as either an empirical CDF (default) or the best parametric family (normal, log-normal, gamma, beta) selected by AIC with a Kolmogorov-Smirnov goodness-of-fit check. Cross-feature dependence is captured by the Gaussian copula correlation matrix.

Usage

```
train_feature_texture(M_matrix, marginal_method = "auto")
```

Arguments

`M_matrix`	Mediator panel, `features x samples` (e.g. gene or transcript expression, one row per feature, one column per sample).
`marginal_method`	Marginal fitting method: `"auto"` (default, uses parametric if KS test passes at $p > 0.05$, otherwise empirical), `"empirical"` (always use empirical CDF), or `"parametric"` (always use best parametric fit by AIC).

Details

The resulting model can be sampled via `sample_feature_texture()` to produce realistic synthetic feature vectors that preserve the marginal shapes and correlation structure of the user's mediator panel.

Value

An `iconic_feature_texture` S3 object: a named list with `marginals` (list of per-feature marginal specs), `copula_cor` (p x p correlation matrix), `n_features`, `n_samples`, `marginal_method`, and `marginal_types` (summary of how many features used each method).

Examples

```
M <- matrix(rnorm(30 * 200), 30, 200) # 30 transcripts, 200 samples
ft <- train_feature_texture(M)
draws <- sample_feature_texture(ft, 100) # 30 x 100 synthetic draws
```

`train_gan_on_real_data` *Train a generative texture model on real data*

Description

Fits an adversarial network (torch GAN) to the tidy training frame. Requires the **torch** package; if torch is not available, an error is raised with installation instructions.

Usage

```
train_gan_on_real_data(
  real_data,
  feature_correlations = NULL,
  feature_texture = NULL,
  epochs = 300,
  batch_size = 32,
  lr = 2e-04,
  seed = NULL,
  verbose = TRUE
)
```

Arguments

`real_data`	Data frame of numeric variables (e.g. the `gan_training_data` element from `load_real_input_data()`). Complete cases are used.
`feature_correlations`	Optional list with elements `Y`, `W`, each a p x p residual correlation matrix from `load_real_input_data()`. When supplied, the matrices are stored in the returned `iconic_gan` object and used by `run_single_iteration()` to generate correlated noise. Default `NULL`.
`feature_texture`	Optional `iconic_feature_texture` object from `train_feature_texture()`. When supplied, it is stored in the returned `iconic_gan` object and used by `run_single_iteration()` to inject realistic mediator texture. Default `NULL`.
`epochs`	GAN training epochs. Default 300.
`batch_size`	Mini-batch size. Default 32.
`lr`	Adam learning rate. Default 2e-4.
`seed`	Optional RNG seed (sets both R and torch seeds).
`verbose`	Print progress. Default TRUE.

Details

Binary columns (values in $0, 1$, e.g. encoded sex or one-hot ethnicity dummies) are detected automatically and stored in the returned object. `sample_texture()` rounds them back to 0/1 and enforces one-hot mutual exclusivity, so synthetic draws respect the discrete structure.

Feature-level residual correlation matrices (for the Y and W panels) can be attached via the `feature_correlations` argument. When present, `run_single_iteration()` uses them to inject correlated noise into the simulated outcome and negative-control panels.

A feature-level copula texture model for the mediator (M) panel can be attached via the `feature_texture` argument (an `iconic_feature_texture` object from `train_feature_texture()`). When present, `run_single_iteration()` uses it to draw realistic mediator feature vectors that preserve the marginal distributions and cross-feature correlation structure of the user's mediator panel.

Value

An `iconic_gan` object: a list with `model_type` (`"gan"`), `columns`, `norm` (per-column centre/scale), `binary_cols` (names of 0/1 columns), `onehot_groups` (list of one-hot dummy groups), `feature_correlations` (list or NULL), `feature_texture` (object or NULL), training statistics, and the trained networks + loss history.

Examples

```
if (check_torch_setup()) {
    dat <- load_real_input_data(example = TRUE)
    gan <- train_gan_on_real_data(dat$gan_training_data,
        feature_correlations = dat$feature_correlations,
        feature_texture = dat$feature_texture,
        epochs = 5, verbose = FALSE)
    head(sample_texture(gan, 5))
}
```

Index

∗ **internal**
.analyze_feature, 7
.analyze_mediation_feature, 7
.apply_composite_pvalues, 8
.auto_eligible_methods, 9
.auto_train_gan, 9
.block_diag_cor, 10
.check_instrument_strength, 10
.composite_robustness, 11
.estimate_mediation_feature, 11
.estimate_total_feature, 13
.fit_parametric_marginal, 14
.generate_nc_noise, 14
.generate_outcome_noise, 15
.iteration_mediator_texture_features, 15
.linpred_to_binary, 16
.linpred_to_surv, 16
.make_pd_cor, 17
.marginal_cdf, 17
.marginal_quantile, 18
.match_cor, 18
.np_cdf, 19
.recommend_by_scenario, 19
.residual_correlation, 20
.resolve_gan, 20
.subset_data_for_inference, 21
bootstrap_mediation_se, 31
compute_iter_bias, 40
create_discriminator, 41
create_generator, 41
delta_se_product, 42
iconic-package, 6
run_mediation_methods, 140
run_methods, 143
summarise_mediation_results, 154
summarise_results, 155
.analyze_feature, 7
.analyze_mediation_feature, 7
.apply_composite_pvalues, 8
.auto_eligible_methods, 9
.auto_train_gan, 9
.block_diag_cor, 10
.check_instrument_strength, 10
.composite_robustness, 11
.estimate_mediation_feature, 11
.estimate_total_feature, 13
.fit_parametric_marginal, 14
.generate_nc_noise, 14
.generate_outcome_noise, 15
.iteration_mediator_texture_features, 15
.linpred_to_binary, 16
.linpred_to_surv, 16
.make_pd_cor, 17
.marginal_cdf, 17
.marginal_quantile, 18
.match_cor, 18
.np_cdf, 19
.recommend_by_scenario, 19
.residual_correlation, 20
.resolve_gan, 20
.subset_data_for_inference, 21

analyze_mediation_robust, 22
analyze_methods_parallel, 23
analyze_methods_robust, 24
analyze_methods_robust(), *23*
apply_fusion_weights, 25
as_iconic_data, 26
as_iconic_data(), *29, 30*, *90*
as_MultiAssayExperiment, 29
as_MultiAssayExperiment(), *27, 28*, *90*

beta_to_m, 30
bias_reduction_prospective (iconic_prospect), 95
bigsnpr::snp_attach(), *153*
bigsnpr::snp_readBed(), *34, 35*
bootstrap_mediation_se, 31
build_mediator_instruments, 32
build_prs_ldpred2, 34
build_prs_ldpred2(), *38*
build_w_pcs, 36
build_w_pcs(), *25*, *27*

call_cis_eqtls, 36

call_cis_eqtls(), *33*
check_instrument_strength, 38
check_torch_setup, 39
composite_p_value, 39
compute_iter_bias, 40
coxph, *89*
create_discriminator, 41
create_generator, 41

delta_se_product, 42

effect_decomposition_bias_sweep *(*iconic_sensitivity*)*, 100

fit_coca, 42, *44*
fit_coca_bin, 43
fit_coca_mediation, 44
fit_coca_mediation_bin, 45
fit_coca_mediation_surv, 45
fit_coca_surv, 46
fit_direct, 47, *49*
fit_direct_bin, 48
fit_direct_mediation, 49
fit_direct_mediation_bin, 49
fit_direct_mediation_surv, 50
fit_direct_surv, 51
fit_iv2sls, 52
fit_iv2sls_bin, 53
fit_iv2sls_mediation, 54, *57*, *61, 62*
fit_iv2sls_mediation2, *8*, *22*, 55, *68*, *86, 87*
fit_iv2sls_mediation2(), *148*
fit_iv2sls_mediation2_bin, 57, *61*
fit_iv2sls_mediation2_surv, 59, *62*
fit_iv2sls_mediation_bin, 61
fit_iv2sls_mediation_surv, 62
fit_iv2sls_surv, 63
fit_pgc, 64, *66*, *74*
fit_pgc_bin, 65
fit_pgc_mediation, 66, *71, 72*, *74*
fit_pgc_mediation2, *8*, *22*, 67, *68*, *70*
fit_pgc_mediation2_bin, 68, *71*
fit_pgc_mediation2_surv, 70, *72*
fit_pgc_mediation_bin, 71
fit_pgc_mediation_surv, 72
fit_pgc_scalar, *64*, 73, *74*
fit_pgc_scalar(), *24*
fit_pgc_scalar_mediation, 74
fit_pgc_scalar_mediation(), *22*
fit_pgc_surv, 75
fit_unadj_bin, 76
fit_unadj_mediation, 76
fit_unadj_mediation_bin, *50*, *58*, *61*, *69*, *72*, 77
fit_unadj_mediation_surv, *51*, *60*, *63*, *71*, *73*, 78
fit_unadj_surv, 79

gan_mediation_sensitivity, 80
gan_pleiotropy_sensitivity, 82
gan_sensitivity, 83
gan_sensitivity(), *80*, *115*, *130*, *139*
generate_toy_data, 85, *97*, *161*
generate_toy_data(), *23, 24*, *89*, *108, 109*, *111–113*, *116*, *149*, *151, 152*
GenomicRanges::findOverlaps(), *37*
GenomicRanges::GRanges, *32*, *37*

iconic *(*iconic-package*)*, 6
iconic-package, 6
iconic_data, 88, *91–93*, *103*
iconic_data(), *27–29*, *32, 33*, *36*
iconic_diagnose, 90, *93*, *99*, *101*, *104*
iconic_diagnose(), *113*
iconic_estimate, *22*, *89*, 92, *99*, *104*
iconic_estimate(), *81*, *84*
iconic_method_colors, 94
iconic_method_order, 95
iconic_prospect, *89*, 95, *104*
iconic_recommend, *96*, 98
iconic_sensitivity, *89*, *96*, *99, 100*, 100, *104*
infer_confounding, *91*, *101*, *103*, 104

list_nc_models, 105
load_real_input_data, 106
load_real_input_data(), *27, 28*, *166*

MultiAssayExperiment::mergeReplicates(), *28*
MultiAssayExperiment::MultiAssayExperiment, *29*, *90*

nc_completeness_capture, 107
nc_completeness_capture(), *109*
nc_completeness_check, *91*, 108
nc_cpg, 110
nc_independence_check, *91*, 111
nc_independence_check(), *111*
nc_independence_check_gm, *91*, 111
nc_proxy, 112
nc_support_check, 113
nc_validity_check, 114
nc_validity_screen, *91*, 115

parallel::mclapply, *23*
parallel::mclapply(), *25*, *33*, *37*
plot_bias, 117

plot_bias_boxplot, 117
plot_bias_distribution, 118
plot_degradation_surface, 119
plot_estimate_distribution, 120
plot_estimated_vs_true, 120
plot_estimator_benchmark, 121
plot_feature_correlation_sweep, 122
plot_gan_diagnostics, 123
plot_instrument_strength_sweep, 124
plot_model_selection, 125
plot_nc_coverage_comparison, 126
plot_nc_validity_diagnostics, 127
plot_pleiotropy_sweep, 128
plot_power, 128
plot_prospective_analysis, 129
plot_sensitivity_heatmap, 130
plot_type1_boxplot, 130
plot_type1_error, 131
plot_type1_vs_conf, 132
print.iconic_confounding, 132
print.iconic_data, 133
print.iconic_diagnosis, 134
print.iconic_feature_texture, 134
print.iconic_gan, 135
print.iconic_prospect, 136
print.iconic_recommendation, 136
print.iconic_sensitivity, 137

qc_gwas_sumstats, 138
qc_gwas_sumstats(), *34*

recommend_estimator, 139
residualize_matrix, 140
residualize_matrix(), *36*
run_mediation_methods, 140
run_mediation_sim, 141
run_methods, 143
run_null_mediation_sim, 143
run_null_sim, 145
run_simulation, 146
run_single_iteration, *22*, *86*, 147
run_single_iteration(), *23*, *24*, *81–84*, *106*, *108*, *109*, *111–113*, *116*, *150*, *166*, *167*

sample_feature_texture, 150
sample_feature_texture(), *165*
sample_texture, 150
sample_texture(), *166*
scenario_manifest, 151
score_pgs_panel, 152
score_pgs_panel(), *38*
simulate_single_genetic_instrument, 153
stats::model.matrix(), *140*
stats::prcomp(), *36*
summarise_mediation_results, 154
summarise_results, 155
SummarizedExperiment::RangedSummarizedExperiment, *32*, *37*
summary.iconic_diagnosis, 155
summary.iconic_prospect, 156
summary.iconic_recommendation, 156
summary.iconic_sensitivity, 157
sweep_instrument_strength, 158
sweep_mediation_null_by_conf, 159
sweep_mediation_param, 160
sweep_nc_validity, 162
sweep_null_by_conf, 163
sweep_param, 164

train_feature_texture, 165
train_feature_texture(), *107*, *150*, *166*, *167*
train_gan_on_real_data, *89*, *96*, *101*, 166
train_gan_on_real_data(), *106*, *123*, *148*, *151*

utils::read.delim(), *153*