

Package: cdzoo (via r-universe)

October 10, 2026

Title A Multi-Backend Benchmark Harness and Recommender for Causal Discovery

Version 0.2.0

Description Provides a single typed interface over a curated portfolio of causal-structure-discovery algorithms drawn from several R and Python back-ends, together with a structural-causal-model scenario grid, fair graph-recovery metrics, and an evidence-routed recommender that inspects a data set and suggests a discovery method suited to its characteristics. Every back-end is optional and loaded lazily only when used, so the harness, the scenario grid, the metrics, and the recommender's routing logic all run with no suggested package installed.

License MIT + file LICENSE

Encoding UTF-8

Language en-AU

Depends R (>= 4.3)

Imports digest, methods, S7, stats, utils

Suggests bnlearn, ggplot2, knitr, pcalg, proxymix (>= 0.11.3), psych, reticulate, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

URL <https://max578.github.io/cdzoo>, <https://github.com/max578/cdzoo>

BugReports <https://github.com/max578/cdzoo/issues>

Config/testthat/edition 3

LazyData true

Roxygen list(markdown = TRUE)

Remotes max578/proxymix

Config/roxygen2/version 8.1.0

Repository <https://max578.r-universe.dev>

Date/Publication 2026-10-09 22:27:32 UTC

RemoteUrl <https://github.com/max578/cdzoo>

RemoteRef main

RemoteSha daf4bc9795c2fb8ff721066130266a35b7765edd

Contents

as_orchestra_manifest	2
cd_auto	3
cd_benchmark	5
cd_correlation_from_counts	6
cd_diagnose	7
cd_discover	8
cd_discover_suffstat	9
cd_edges	10
cd_explain	11
cd_field_benchmark	12
cd_method	13
cd_result	14
cd_scenario	16
cd_scenarios	16
cd_score	17
cd_suffstat	18
list_methods	19
orchestra_manifest	20
plot.cd_result	20
register_method	21
verify_manifest	22

Index	23
--------------	-----------

as_orchestra_manifest *Emit a discovery result as a federation manifest*

Description

The federation's emit / adapt generic. The `cd_result` method turns a recovered graph into the shared orchestra_manifest contract: `inferential_target = "structure"`, the recovered edges carried as a tidy params edge list, and the provenance – which method ran, the routing rationale, the routing confidence, and whether the result is a single method or a consensus – in the typed summary and metadata. The full adjacency matrix (and, for a consensus, the per-edge agreement) rides in metadata for a consumer that wants the matrix rather than the edge list. When no graph was recovered (an absent back-end), the manifest abstains and says so. The payload is integrity-hashed so a consumer can verify it with `verify_manifest()`.

Usage

```
as_orchestra_manifest(x, ...)
```

Arguments

`x` A `cd_result`.

`...` Further arguments passed to methods. The `cd_result` method accepts `run_id` (an optional character run identifier; a content hash is derived when omitted, so two identical results get the same identifier) and `seed` (an optional integer recorded for reproducibility).

Value

An `orchestra_manifest` S7 object.

See Also

`verify_manifest()`, `cd_auto()`, `cd_discover()`

Examples

```
g <- matrix(c(0, 0, 1, 0), 2, 2,
             dimnames = list(c("x1", "x2"), c("x1", "x2")))
as_orchestra_manifest(cd_result(graph = g, method = "demo",
                               method_name = "Demo", family = "constraint"))
```

cd_auto

Recommend and run a causal-discovery method for a data set

Description

Inspects a data set with `cd_diagnose()`'s transparent characteristics, routes it to a discovery method suited to that regime, runs the best installed method on the route, and returns the recovered graph together with the routing rationale and a confidence label. Because causal discovery has no run-time ground truth, this is a recommender, not a selector: it reports why it chose and how sure it is, and – in "consensus" mode – returns a stability-weighted union of several methods with per-edge agreement.

Usage

```
cd_auto(
  data,
  mode = c("route", "consensus"),
  report = TRUE,
  top_k = 3L,
  threshold = 0.05,
  density_components = 4L
)
```

Arguments

data	A numeric matrix or data frame of observations by variables, an evaluable-density object (a proxymix gmm / gmm_fit / gmm_target), or a sufficient statistic (cd_suffstat()).
mode	"route" (run the single recommended method) or "consensus" (run the top top_k installed methods on the route and return their stability-weighted union).
report	A single logical. When TRUE (the default) the recommended method is run and its graph returned; when FALSE only the recommendation is returned, with an empty graph and the status "recommendation only (report = FALSE)".
top_k	The number of methods to combine in "consensus" mode.
threshold	The partial-correlation edge threshold for the density route.
density_components	The number of mixture components to fit when routing an evaluable target density.

Details

The route is decided by four data-characteristic thresholds, checked in this order and reported in the result's `provenance$thresholds`: a curvature heuristic above 0.05 routes to a kernel conditional-independence test; a non-Gaussian margin fraction of 0.5 or more routes to the LiNGAM family; a correlation condition number above 1000 (a latent-structure hint) routes to a confounder-robust method; and a residual-variance ratio below 2 routes to the equal-variance continuous-optimisation methods. Failing all four, the benchmark's strongest overall method is used.

When data is an evaluable density rather than a sample – a proxymix gmm, gmm_fit, or gmm_target (requires the suggested **proxymix** package) – the conditional-independence (Markov) structure is read off the mixture in closed form, the route for the case where a density can be evaluated but not sampled.

When data is a sufficient statistic ([cd_suffstat\(\)](#), a correlation matrix and its sample size), the recommender routes among the sufficient-statistic methods. A correlation matrix carries only the latent-structure hint, so the rationale states plainly that the non-Gaussianity, nonlinearity, and equal-variance tests could not be run.

Value

A [cd_result](#) whose rationale, confidence, and alternatives (always a list of method labels, in the route's preference order) record the recommendation. In "consensus" mode the per-edge agreement and the individual member results are stored in the result's provenance (as agreement and runs).

See Also

[cd_diagnose\(\)](#), [cd_discover\(\)](#), [list_methods\(\)](#)

Examples

```
if (requireNamespace("pcalg", quietly = TRUE)) {
  sim <- cd_scenario("linear_laplace", seed = 1)
```

```

    rec <- cd_auto(sim$data, report = FALSE)
    rec@rationale
  }

```

cd_benchmark

Benchmark discovery methods across the scenario grid

Description

Runs each installed method across the chosen scenarios and seeds, scores every run against the scenario's known truth with `cd_score()`, and returns the per-run results as a tidy data frame. Each run is isolated, so a back-end that errors on a cell records a failed status rather than stopping the benchmark.

Usage

```

cd_benchmark(
  methods = NULL,
  scenarios = NULL,
  seeds = 1:4,
  cost_limit = c("medium", "fast", "slow", "very_slow"),
  verbose = TRUE,
  mc.cores = 1L
)

```

Arguments

<code>methods</code>	A character vector of method ids to run, or NULL (the default) for every installed method.
<code>scenarios</code>	A character vector of scenario names, or NULL (the default) for all of them.
<code>seeds</code>	An integer vector of seeds; each scenario is realised once per seed.
<code>cost_limit</code>	A single string: the costliest method tier to include, one of "fast", "medium", "slow", or "very_slow".
<code>verbose</code>	A single logical: report progress per scenario.
<code>mc.cores</code>	A single positive integer: the number of cores to run the grid on. The default 1L runs serially. Each cell is deterministic given its scenario seed and independent of the others, so the result is identical for any core count; parallel execution uses <code>parallel::mclapply()</code> (which falls back to serial on Windows).

Details

The default skips methods costlier than `cost_limit` so a first pass is quick; raise the limit to include the slower kernel and search methods. With no back-end installed the result has zero rows. A requested method that cannot run is dropped with a warning naming it and the reason (an unknown id, an absent back-end, or a cost above the limit) – never silently.

Value

A data frame with one row per (scenario, method, seed) run, with columns scenario, family, method, method_family, language, seed, status, skeleton_f1, shd, orient_recall, and runtime_s.

See Also

[cd_scenario\(\)](#), [cd_score\(\)](#), [list_methods\(\)](#)

Examples

```
if (requireNamespace("pcalg", quietly = TRUE)) {
  res <- cd_benchmark(
    methods = "pc_pcalg",
    scenarios = "linear_gaussian",
    seeds = 1:2, verbose = FALSE
  )
  res[, c("scenario", "method", "skeleton_f1", "shd")]
}
```

cd_correlation_from_counts

Assemble a correlation matrix from pairwise binary counts

Description

Builds a positive-definite correlation matrix over a set of binary variables from their pairwise two-by-two co-occurrence counts – the aggregated form in which co-occurrence data (comorbidity, market-basket, species co-occurrence) is usually published. The result feeds straight into [cd_discover_suffstat\(\)](#).

Usage

```
cd_correlation_from_counts(pairs, type = c("phi", "tetra"), repair = TRUE)
```

Arguments

pairs	A data frame with one row per variable pair and columns var_i, var_j (the two variable identifiers) and n11, n10, n01, n00 (the counts of: both present, the first only, the second only, both absent).
type	"phi" (default; the Pearson correlation of the two indicators, base R) or "tetra" (the latent-Gaussian tetrachoric correlation, which needs the psych package).
repair	Whether to project an indefinite matrix to the nearest positive-definite correlation matrix. Defaults to TRUE.

Details

A matrix assembled pair by pair is not guaranteed positive semi-definite, so it is repaired to the nearest positive-definite correlation matrix by eigenvalue clipping when `repair = TRUE`; the eigenvalue floor reached and the repair distortion are recorded in the result's "repair" attribute.

Every unordered variable pair must appear exactly once; a repeated pair is an error rather than a silent overwrite. A pair that is absent from `pairs` is taken to have zero correlation (the off-diagonal entry stays zero before repair).

Value

A correlation matrix with a "repair" attribute recording the minimum eigenvalue before and after repair, the Frobenius distortion, and the pair and variable counts.

See Also

[cd_discover_suffstat\(\)](#)

Examples

```
pairs <- data.frame(
  var_i = "a", var_j = "b",
  n11 = 900, n10 = 100, n01 = 100, n00 = 8900)
cd_correlation_from_counts(pairs)
```

cd_diagnose

Describe the characteristics of a discovery data set

Description

Reports the data characteristics the recommender uses to route a problem, as a one-row data frame. Use it to see what [cd_auto\(\)](#) sees before it chooses a method.

Usage

```
cd_diagnose(data)
```

Arguments

`data` A numeric matrix or data frame of observations (rows) by variables (columns).

Details

On large data sets the statistics are computed on a deterministic, evenly spaced subsample of rows (at most 5000 rows for the normality test, 2000 for the regression heuristics), so the diagnostics are pure: repeated calls on the same data return the same characteristics, and the caller's random-number state is never touched.

Value

A one-row data frame of the characteristics described in `cd_auto()`'s routing.

See Also

`cd_auto()`

Examples

```
sim <- cd_scenario("linear_laplace", seed = 1)
cd_diagnose(sim$data)
```

cd_discover

Discover a causal graph with one method

Description

Runs a single registered discovery method on a data set and returns the recovered graph as a `cd_result`. Use `list_methods()` to see the registered methods and which back-ends are installed, and `cd_auto()` to let the recommender choose a method for you.

Usage

```
cd_discover(data, method)
```

Arguments

data	A numeric matrix or data frame of observations (rows) by variables (columns).
method	A single string: the id of a registered method (see <code>list_methods()</code>).

Details

A method that consumes a sufficient statistic (its accepts is "suffstat", such as "pc_suffstat") can still be run here on a data matrix: the data is reduced to its correlation matrix and sample size first. To run such a method on an already-aggregated statistic instead, use `cd_discover_suffstat()`.

Value

A `cd_result`. When the back-end raises an error, the result carries an empty graph and the error in its status; an absent back-end is a hard error pointing at the package or module to install.

See Also

`cd_auto()`, `list_methods()`, `cd_score()`

Examples

```
if (requireNamespace("pcalg", quietly = TRUE)) {
  sim <- cd_scenario("linear_gaussian", seed = 1)
  res <- cd_discover(sim$data, "pc_pcalg")
  res
}
```

cd_discover_suffstat *Discover a causal graph from a sufficient statistic*

Description

Runs constraint-based discovery directly on a correlation (or covariance) matrix and its sample size, returning a [cd_result](#) in the same `amat.cpdag` convention as [cd_discover\(\)](#). This is the entry point for data that arrives already aggregated – a published correlation matrix, a privacy-preserving summary, or the output of [cd_correlation_from_counts\(\)](#) – with no row-level observation matrix to pass to [cd_discover\(\)](#).

Usage

```
cd_discover_suffstat(
  C,
  n,
  method = c("pc", "fci", "ggm"),
  test = c("gauss", "magnitude"),
  alpha = 0.05,
  tau = 0.05
)
```

Arguments

C	A square correlation or covariance matrix; covariance input is standardised. Variable names are taken from its <code>dimnames</code> .
n	A single positive number: the sample size the statistic summarises.
method	One of "pc" (default), "fci", or "ggm".
test	One of "gauss" (default) or "magnitude".
alpha	The Fisher-Z significance level for <code>test = "gauss"</code> .
tau	The partial-correlation magnitude threshold for <code>test = "magnitude"</code> and for <code>method = "ggm"</code> .

Details

method = "ggm" returns the Gaussian-graphical-model skeleton (an undirected edge wherever the partial correlation given all other variables clears tau) and needs no back-end. method = "pc" and method = "fci" run the PC and FCI algorithms from pcalg and add such orientation as the data identify; FCI additionally allows latent confounders.

test = "gauss" uses the Fisher-Z significance test at level alpha, as `cd_discover()` does. test = "magnitude" instead prunes an edge when the partial correlation magnitude falls below tau; use it when n is so large that a significance test would call every pair dependent.

Value

A `cd_result` whose `provenance$source` is "suffstat".

See Also

`cd_discover()`, `cd_correlation_from_counts()`, `cd_score()`

Examples

```
if (requireNamespace("pcalg", quietly = TRUE)) {
  sim <- cd_scenario("linear_gaussian", seed = 1)
  # discover from the sufficient statistic alone -- no row-level data
  res <- cd_discover_suffstat(stats::cor(sim$data), nrow(sim$data), "pc")
  cd_score(res, sim$truth)
}
```

cd_edges

Edge list of a discovery result

Description

Returns the edges of a `cd_result` (or a bare adjacency matrix) as a data frame, one row per edge, with the orientation made explicit.

Usage

```
cd_edges(x, agreement = FALSE)
```

Arguments

x	A <code>cd_result</code> or a square adjacency matrix in <code>amat.cpdag</code> convention.
agreement	A single logical. When TRUE and x is a consensus <code>cd_result</code> carrying per-edge agreement (from <code>cd_auto()</code> with mode = "consensus"), an agreement column reports the share of the combined methods that found each edge. NA where no agreement is recorded.

Value

A data frame with columns from, to, and type ("-->" for a directed edge, "---" for an undirected one), and – when agreement = TRUE – an agreement column. Zero rows when the graph is empty.

See Also

Other results: [cd_result\(\)](#)

Examples

```
g <- matrix(c(0, 0, 1, 0), 2, 2,
            dimnames = list(c("x1", "x2"), c("x1", "x2")))
cd_edges(cd_result(graph = g, method = "demo"))
```

cd_explain

*Explain a discovery recommendation in words***Description**

Prints a readable account of a [cd_result](#): the method that ran and its status, the routing rationale and confidence, the alternatives the recommender considered, the data-characteristic thresholds that decided the route (when [cd_auto\(\)](#) produced the result), and – for a "consensus" result – the per-edge agreement among the combined methods. It is the auditable companion to `print()`, which stays compact.

Usage

```
cd_explain(x)
```

Arguments

x A [cd_result](#), typically from [cd_auto\(\)](#).

Value

x, invisibly. Called for the account it prints.

See Also

[cd_auto\(\)](#), [cd_result](#), [cd_edges\(\)](#)

Examples

```
if (requireNamespace("pcalg", quietly = TRUE)) {
  sim <- cd_scenario("linear_laplace", seed = 1)
  cd_explain(cd_auto(sim$data, report = FALSE))
}
```

cd_field_benchmark	<i>Field benchmark of causal-discovery methods across scenarios</i>
--------------------	---

Description

A summary of sixteen established causal-discovery methods run across the nine structural-causal-model scenarios of `cd_scenarios()`, each method scored against the scenario's known DAG and averaged over random-graph seeds. It is the evidence that grounds the routing preferences in `cd_auto()`, and the worked example in the package vignette.

Usage

```
cd_field_benchmark
```

Format

A data frame with 143 rows and 11 columns:

scenario Scenario name (see `cd_scenarios()`).

scenario_family The scenario's structural-causal-model family.

method Method id (see `list_methods()`).

method_name Human-readable method name.

family Algorithm family.

shd Mean structural Hamming distance to the true DAG (lower is better).

shd_sd Standard deviation of the structural Hamming distance across seeds.

skeleton_f1 Mean skeleton (adjacency) F1 (higher is better).

orient_recall Mean orientation recall.

runtime_s Mean wall-clock runtime in seconds.

n_seed Number of random-graph seeds averaged.

Details

One row per (scenario, method). The kernel conditional-independence method (`pc_kci_cl`) was not run on the ten-variable scenario, so that one (scenario, method) cell is absent, giving 143 rows.

See Also

`cd_benchmark()`, `cd_auto()`, `cd_scenarios()`

Examples

```
# overall ranking by mean structural Hamming distance across scenarios
agg <- aggregate(shd ~ method_name, data = cd_field_benchmark, FUN = mean)
agg[order(agg$shd), ]
```

cd_method

*A causal-discovery method specification***Description**

Bundles the metadata that drives selection – family, language, native output type, cost, and assumptions – with the adapter that runs a back-end algorithm on a data matrix and returns its estimate in the shared `amat.cpdag` convention. The available predicate reports whether the back-end can run in the current session, so a method whose package or Python module is absent is skipped rather than failing.

Usage

```
cd_method(
  id = character(0),
  name = "",
  family = "constraint",
  language = "R",
  output = "cpdag",
  citation = "",
  cost = "fast",
  assumptions = "",
  accepts = "data",
  available = function() NULL,
  run = function() NULL
)
```

Arguments

<code>id</code>	A single string identifying the method; this is the value passed to <code>method = in cd_discover()</code> .
<code>name</code>	A single string giving the human-readable method name.
<code>family</code>	A single string naming the algorithm family, one of "constraint", "score", "functional", "continuous_optim", "hybrid", or "density".
<code>language</code>	A single string, "R" or "Python".
<code>output</code>	A single string giving the native output type, one of "dag", "cpdag", or "pag".
<code>citation</code>	A single string citing the method's source.
<code>cost</code>	A single string giving the rough computational cost, one of "fast", "medium", "slow", or "very_slow".
<code>assumptions</code>	A single string describing the method's modelling assumptions.
<code>accepts</code>	A single string naming the input the method's run consumes, "data" (the default; a numeric observation matrix) or "suffstat" (a sufficient statistic, the list <code>list(C, n)</code> of a correlation matrix and its sample size). A "suffstat" method can still be called on a data matrix – cd_discover() reduces the data to its sufficient statistic first.

available	A function of no arguments returning a single logical: can the method run in the current session?
run	A function taking the method's input (a numeric data matrix, or a <code>list(C, n)</code> sufficient statistic when <code>accepts = "suffstat"</code>) and returning a list with elements <code>amat</code> (the estimate in <code>amat.cpdag</code> convention, or <code>NULL</code>), <code>runtime_s</code> , and <code>status</code> .

Value

An S7 object of class `cd_method`.

See Also

Other registry: [list_methods\(\)](#), [register_method\(\)](#)

Examples

```
cd_method(
  id = "demo", name = "Demo", family = "constraint",
  available = function() TRUE,
  run = function(data) list(amat = NULL, runtime_s = 0, status = "ok")
)
```

cd_result

A causal-discovery result

Description

The object returned by [cd_discover\(\)](#) and [cd_auto\(\)](#). It holds the recovered graph as an adjacency matrix in the `pcalg amat.cpdag` convention, where `graph[i, j] == 1` and `graph[j, i] == 0` encodes the directed edge $i \rightarrow j$, and `graph[i, j] == graph[j, i] == 1` encodes an undirected (unoriented) edge $i - j$.

Usage

```
cd_result(
  graph = numeric(0),
  method = NA_character_,
  method_name = "",
  family = NA_character_,
  language = "R",
  output = "cpdag",
  runtime_s = NA_real_,
  status = "ok",
  rationale = "",
  confidence = NA_character_,
  alternatives = list(),
  provenance = list()
)
```

Arguments

graph	A square numeric adjacency matrix in the <code>amat.cpdag</code> convention, with variable names as <code>dimnames</code> ; empty when the method failed.
method	A single string: the identifier of the method that produced the result.
method_name	A single string: the human-readable method name.
family	A single string: the algorithm family (constraint, score, functional, continuous-optimisation, hybrid, or density).
language	A single string: the back-end language, "R" or "Python".
output	A single string: the native output type, one of "dag", "cpdag", or "pag".
runtime_s	A single number: wall-clock runtime in seconds.
status	A single string: "ok", or a short error description.
rationale	A single string: why the recommender chose this method.
confidence	A single string: "high", "moderate", or "low".
alternatives	A list of method labels the recommender considered as alternatives, in its preference order.
provenance	A list of call metadata (data characteristics, options).

Details

The `rationale`, `confidence`, and `alternatives` properties are populated by `cd_auto()` and are empty for a single-method `cd_discover()` call.

Value

An S7 object of class `cd_result`.

See Also

Other results: `cd_edges()`

Examples

```
g <- matrix(c(0, 0, 1, 0), 2, 2,
             dimnames = list(c("x1", "x2"), c("x1", "x2")))
cd_result(graph = g, method = "pc_pcalg", method_name = "PC (Fisher-Z)")
```

cd_scenario	<i>Generate a benchmark scenario with known ground truth</i>
-------------	--

Description

Realises one of the built-in structural-causal-model scenarios at a given seed and returns the data together with the true graph, the independent oracle a recovered structure is scored against by `cd_score()`.

Usage

```
cd_scenario(name, seed = 1L)
```

Arguments

name	A single string: the scenario name (see <code>cd_scenarios()</code>).
seed	A single integer-valued seed for reproducibility; any finite number is accepted (a value beyond integer range is folded back into it).

Value

A list with data (the observation matrix), truth (the true graph in `amat.cpdag` convention), dag (the raw simulator DAG), scenario, family, p, and n.

See Also

`cd_scenarios()`, `cd_score()`, `cd_benchmark()`

Examples

```
sim <- cd_scenario("linear_gaussian", seed = 1)
dim(sim$data)
sim$truth
```

cd_scenarios	<i>The built-in benchmark scenarios</i>
--------------	---

Description

Lists the structural-causal-model scenarios available to `cd_scenario()` and `cd_benchmark()`.

Usage

```
cd_scenarios()
```

Value

A data frame with columns name, p, n, family, and description.

See Also

[cd_scenario\(\)](#), [cd_benchmark\(\)](#)

Examples

```
cd_scenarios()
```

cd_score	<i>Score a recovered graph against a known truth</i>
----------	--

Description

Compares a discovery result – a [cd_result](#) or a bare adjacency matrix – with the true graph and returns the skeleton, distance, and orientation metrics. The truth is typically the truth element of a [cd_scenario\(\)](#).

Usage

```
cd_score(x, truth)
```

Arguments

x	A cd_result or a square adjacency matrix in <code>amat.cpdag</code> convention.
truth	The true graph in <code>amat.cpdag</code> convention (for example <code>cd_scenario(...)\$truth</code>).

Details

Both matrices are normalised before scoring, so any non-zero entry counts as an edge mark and the three metrics see the same graph whether the input is 0/1 or weighted. When the estimate carries no `dimnames` but matches the truth's dimensions, it is scored positionally against the truth's variable order.

Value

A one-row data frame with `skeleton_f1`, `skeleton_precision`, `skeleton_recall`, `shd`, and `orient_recall`. `skeleton_f1` is NA when the truth has no edges (recall is undefined there); an empty estimate against a non-empty truth scores 0.

See Also

[cd_scenario\(\)](#), [cd_benchmark\(\)](#)

Examples

```
if (requireNamespace("pcalg", quietly = TRUE)) {
  sim <- cd_scenario("linear_gaussian", seed = 1)
  res <- cd_discover(sim$data, "pc_pcalg")
  cd_score(res, sim$truth)
}
```

cd_suffstat

*A sufficient statistic as a discovery input***Description**

Wraps a correlation (or covariance) matrix and its sample size as a tagged object that [cd_auto\(\)](#) accepts in place of a data matrix, so the recommender can route and run on data that arrives already aggregated. The matrix is validated and standardised exactly as [cd_discover_suffstat\(\)](#) does.

Usage

```
cd_suffstat(C, n)
```

Arguments

C	A square correlation or covariance matrix; covariance input is standardised. Variable names are taken from its dimnames.
n	A single positive number: the sample size the statistic summarises.

Value

An object of class `cd_suffstat`: a list with the validated correlation matrix `C` and the sample size `n`.

See Also

[cd_auto\(\)](#), [cd_discover_suffstat\(\)](#)

Examples

```
sim <- cd_scenario("linear_gaussian", seed = 1)
ss <- cd_suffstat(stats::cor(sim$data), nrow(sim$data))
ss$n
```

list_methods	<i>The registered discovery methods</i>
--------------	---

Description

Returns every registered method as a [cd_method](#) list, or – with `as_data_frame = TRUE` – a data frame summarising the registry and which back-ends are installed in the current session.

Usage

```
list_methods(as_data_frame = TRUE, refresh = FALSE)
```

Arguments

<code>as_data_frame</code>	A single logical. When TRUE (the default) a data frame is returned; when FALSE a named list of cd_method objects.
<code>refresh</code>	A single logical. When TRUE, the session cache of Python back-end availability checks is discarded and the probes run again – use it after installing a Python module mid-session.

Details

Python back-end availability is checked passively: `cdzoo` never initialises Python (which can trigger interpreter provisioning) just to report it. A Python back-end shows `installed = TRUE` once `reticulate`'s interpreter is initialised or configured (for example via `RETICULATE_PYTHON`); set `options(cdzoo.python_init = TRUE)` to let this check initialise Python eagerly. Probe results are cached for the session.

Value

A data frame with columns `id`, `name`, `family`, `language`, `cost`, `accepts` ("data" or "suffstat"), and `installed`, or a named list of [cd_method](#) objects.

See Also

Other registry: [cd_method\(\)](#), [register_method\(\)](#)

Examples

```
list_methods()
```

orchestra_manifest	<i>The federation manifest contract (cdzoo-side implementation)</i>
--------------------	---

Description

A versioned, hashed, provenance-complete S7 result object, property-compatible with the federation's reference orchestra_manifest. A [cd_result](#) is adapted into one through [as_orchestra_manifest\(\)](#): the recovered edges ride in params (inferential_target = "structure"), the method, routing rationale, confidence, and consensus flag ride in metadata and the typed summary, and the payload is integrity-hashed so a tampered graph is detected downstream.

Value

An S7 object of class orchestra_manifest.

See Also

[as_orchestra_manifest\(\)](#), [verify_manifest\(\)](#)

Examples

```
g <- matrix(0, 2, 2, dimnames = list(c("x", "y"), c("x", "y")))
g["x", "y"] <- 1
m <- as_orchestra_manifest(cd_result(graph = g, method = "demo"))
m@inferential_target
m@summary$abstained
verify_manifest(m)$ok
```

plot.cd_result	<i>Plot a discovery result as a graph</i>
----------------	---

Description

Draws the recovered graph with the variables on a circle and an arrow for each directed edge, a plain segment for each undirected one. It uses base graphics only, so it needs no plotting back-end.

Arguments

x	A cd_result with a non-empty graph.
...	Further graphical parameters passed to the node points.

Value

x, invisibly.

Examples

```
g <- matrix(0, 3, 3, dimnames = list(paste0("x", 1:3), paste0("x", 1:3)))
g["x1", "x2"] <- 1
g["x2", "x3"] <- 1
plot(cd_result(graph = g, method = "demo"))
```

register_method	<i>Register a discovery method</i>
-----------------	------------------------------------

Description

Adds a [cd_method](#) to the registry so it can be run by [cd_discover\(\)](#) and considered by [cd_auto\(\)](#). Registering a method with the id of an existing one replaces it.

Usage

```
register_method(method)
```

Arguments

method A [cd_method](#) to register.

Value

The method id, invisibly.

See Also

Other registry: [cd_method\(\)](#), [list_methods\(\)](#)

Examples

```
m <- cd_method(
  id = "noop", name = "No-op", family = "constraint",
  available = function() TRUE,
  run = function(data) list(amat = NULL, runtime_s = 0, status = "ok")
)
register_method(m)
"noop" %in% list_methods()$id
```

verify_manifest	<i>Verify a manifest's payload integrity</i>
-----------------	--

Description

Recomputes the payload hash from the object's own load-bearing slots and compares it to the stored `data_hash`. A mismatch means the payload was modified after emission. Matches the reference contract's `verify_manifest()`.

Usage

```
verify_manifest(m)
```

Arguments

<code>m</code>	An <code>orchestra_manifest</code> object.
----------------	--

Value

A list with logical ok and a human-readable message.

See Also

[as_orchestra_manifest\(\)](#)

Examples

```
g <- matrix(c(0, 0, 1, 0), 2, 2,
             dimnames = list(c("x1", "x2"), c("x1", "x2")))
m <- as_orchestra_manifest(cd_result(graph = g, method = "demo"))
verify_manifest(m)
```

Index

- * **datasets**
 - cd_field_benchmark, 12
- * **registry**
 - cd_method, 13
 - list_methods, 19
 - register_method, 21
- * **results**
 - cd_edges, 10
 - cd_result, 14

as_orchestra_manifest, 2
as_orchestra_manifest(), 20, 22

cd_auto, 3
cd_auto(), 3, 7, 8, 10–12, 14, 15, 18, 21
cd_benchmark, 5
cd_benchmark(), 12, 16, 17
cd_correlation_from_counts, 6
cd_correlation_from_counts(), 9, 10
cd_diagnose, 7
cd_diagnose(), 3, 4
cd_discover, 8
cd_discover(), 3, 4, 9, 10, 13–15, 21
cd_discover_suffstat, 9
cd_discover_suffstat(), 6–8, 18
cd_edges, 10
cd_edges(), 11, 15
cd_explain, 11
cd_field_benchmark, 12
cd_method, 13, 19, 21
cd_method(), 19, 21
cd_result, 2–4, 8–11, 14, 17, 20
cd_result(), 11
cd_scenario, 16
cd_scenario(), 6, 16, 17
cd_scenarios, 16
cd_scenarios(), 12, 16
cd_score, 17
cd_score(), 5, 6, 8, 10, 16
cd_suffstat, 18
cd_suffstat(), 4
list_methods, 19
list_methods(), 4, 6, 8, 12, 14, 21
orchestra_manifest, 20
parallel::mclapply(), 5
plot.cd_result, 20
register_method, 21
register_method(), 14, 19
verify_manifest, 22
verify_manifest(), 2, 3, 20