Package: PFCI (via r-universe)

June 3, 2026

Title Penalized Fast Causal Inference for High-Dimensional Structure Learning

Version 0.1.1

Date 2026-05-28

Description Implements Penalized Fast Causal Inference (PFCI), a two-stage causal structure learning procedure for high-dimensional settings with potential latent variables and selection bias. In the first stage, neighborhood selection via the Lasso constructs a sparse undirected skeleton. In the second stage, the Fast Causal Inference (FCI) algorithm orients edges on this reduced graph, producing a Partial Ancestral Graph (PAG) that accounts for latent confounders. The method is consistent under sparsity assumptions and substantially faster than standard FCI and RFCI in high dimensions. See Pal, Ghosh, and Yang (2025) <doi:10.48550/arXiv.2507.00173> for the underlying theory.

License MIT + file LICENSE

Encoding UTF-8

URL https://github.com/djghosh1123/PFCI

BugReports https://github.com/djghosh1123/PFCI/issues

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports stats, glasso, methods

Suggests pcalg, graph, RBGL, Rgraphviz, testthat (>= 3.0.0), knitr, rmarkdown, spelling

VignetteBuilder knitr

Config/testthat/edition 3

Language en-US

Repository https://djghosh1123.r-universe.dev

Date/Publication 2026-06-03 05:12:21 UTC

RemoteUrl https://github.com/djghosh1123/pfci

RemoteRef HEAD

RemoteSha 062b2b00775151f448395bb2ed0dde86a76de37d

Contents

`metrics_with_latent` *Metrics for latent simulation using oracle-FCI truth (skeleton only)*

Description

Designed for the 3-line workflow: sim <- simulate_with_latent(...) fit <- pfci_fit(sim$X, ...) met <- metrics_with_latent(sim, fit)

Usage

```
metrics_with_latent(sim, fit)
```

Arguments

`sim`	Output from simulate_with_latent().
`fit`	Output from pfci_fit() (must contain $amat and $time$total).

Details

Returns only: SHD, F1_total, MCC, Time.

Value

A named list with SHD, F1_total, MCC, Time.

See Also

`simulate_with_latent`, `pfci_fit`

Examples

```
sim <- simulate_with_latent(p_obs = 30, gamma = 0.05, n = 100, seed_graph = 1)
fit <- pfci_fit(sim$X, alpha = 0.05)
met <- metrics_with_latent(sim, fit)
print(met)
```

`pfci_fit` *Penalized FCI (PFCI): glasso screening + constrained FCI*

Description

Runs a two-stage procedure: (1) Graphical lasso screening to obtain a sparse undirected super-skeleton (2) FCI on the restricted search space using fixedGaps and a gated CI test

Usage

```
pfci_fit(
  X,
  alpha = 0.05,
  rho = NULL,
  approx = TRUE,
  skel.method = "stable",
  doPdsep = FALSE,
  labels = NULL
)
```

Arguments

`X`	Numeric matrix or data.frame of dimension n x p.
`alpha`	Significance level for conditional independence tests in FCI.
`rho`	Graphical lasso penalty. If NULL, uses a default depending on n.
`approx`	Passed to glasso::glasso.
`skel.method`	Skeleton method for pcalg::fci (default "stable").
`doPdsep`	Logical; passed to pcalg::fci. Default FALSE.
`labels`	Optional variable names (length p). If NULL uses colnames or X1..Xp.

Value

An object of class `pfci_fit`, a list containing:

amat Adjacency matrix of the estimated PAG (integer codes: 0=none, 1=circle, 2=arrowhead, 3=tail).

pag The raw `fci` output object from **pcalg**.

screen_adj Logical adjacency matrix from the glasso screening step.

fixedGaps Logical matrix of fixed gaps passed to FCI.

rho The glasso penalty used.

alpha The significance level used.

time A list with `glasso`, `fci`, and `total` runtimes in seconds.

References

Pal, S., Ghosh, D., and Yang, S. (2025). Penalized FCI for Causal Structure Learning in a Sparse DAG for Biomarker Discovery in Parkinson's Disease. *Annals of Applied Statistics*. doi:10.48550/arXiv.2507.00173

See Also

`pfci_metrics`, `plot_pag`, `simulate_pfci_toy`

Examples

```
sim <- simulate_pfci_toy(p = 30, n = 100, edge_prob = 0.05, seed = 1)
fit <- pfci_fit(sim$X, alpha = 0.05)
print(fit)
```

---

pfci_metrics *Compute PFCI metrics from a simulation object and a pfci_fit output*

Description

Designed for the 3-line workflow: sim <- simulate_pfci_toy(...) fit <- pfci_fit(sim$X, ...) met <- pfci_metrics(sim, fit)

Usage

```
pfci_metrics(sim, fit, compute_marks = FALSE)
```

Arguments

`sim`	Output from simulate_pfci_toy().
`fit`	Output from pfci_fun()/pfci_fit() with at least $amat and $time$total.
`compute_marks`	Logical. If TRUE, also computes mark-level F1 when truth amat is present.

Details

Default metrics compare estimated PAG adjacency (skeleton) to the generating DAG skeleton.

If compute_marks=TRUE and sim$truth$amat exists, it also reports mark-level F1s:

- F1_dir (->)
- F1_oDir (o->)
- F1_bidir (<->)
- F1_circ (o-o)
- F1_arrow (arrowheads)
- F1_tail (tails)

Value

A named list of metrics.

See Also

pfci_fit, simulate_pfci_toy

Examples

```
sim <- simulate_pfci_toy(p = 30, n = 100, edge_prob = 0.05, seed = 1)
fit <- pfci_fit(sim$X, alpha = 0.05)
met <- pfci_metrics(sim, fit)
print(met)
```

---

plot_pag *Plot a PAG returned by PFCI*

Description

Plots the Partial Ancestral Graph (PAG) estimated by pfci_fit using the **pcalg** plot method. Requires **Rgraphviz** to be installed.

Usage

```
plot_pag(fit, ...)
```

Arguments

fit	A pfci_fit object returned by pfci_fit, or a raw pcalg fci object.
...	Additional arguments passed to the **pcalg** plot method.

Value

Invisibly returns NULL. Called for its side effect of producing a graph plot.

See Also

pfci_fit

Examples

```
sim <- simulate_pfci_toy(p = 20, n = 100, edge_prob = 0.05, seed = 1)
fit <- pfci_fit(sim$X, alpha = 0.05)
plot_pag(fit)
```

`simulate_pfci_toy` *Simulate toy data for PFCI using topo-ordered DAG + rmvDAG*

Description

Workflow: sim <- simulate_pfci_toy(...) fit <- pfci_fun(sim$X, ...) met <- pfci_metrics(sim, fit)

Usage

```
simulate_pfci_toy(
  p = NULL,
  sparsity = NULL,
  n = 100,
  edge_prob = 0.02,
  errDist = c("normal", "t4", "mixt3"),
  seed = 1L,
  p_obs = NULL,
  gamma = 0.1
)
```

Arguments

p	Number of observed variables (preferred).
sparsity	Number of nodes eligible for edges (<= p). Default p.
n	Sample size.
edge_prob	Edge probability among eligible nodes.
errDist	Error distribution for pcalg::rmvDAG ("normal","t4","mixt3").
seed	Random seed.
p_obs	(legacy) alias for p.
gamma	(legacy) ignored (kept only for backward compatibility).

Details

This simulator:

- generates a *topologically ordered* DAG (edges only i -> j for i < j)
- simulates data via pcalg::rmvDAG with requested errDist
- returns truth skeleton (undirected) and an "amat-style" truth from dag2cpdag

NOTE: The returned truth_amat is derived from the CPDAG of the generating DAG (so it contains directed and o-o circle edges, but not latent-induced o-> / <->).

Backward-compat: accepts old args p_obs/gamma (ignored) so old vignettes won't fail.

Value

A list: X, truth (true_dag, adj_mat, skel, amat), meta

See Also

pfci_fit, pfci_metrics

Examples

```
sim <- simulate_pfci_toy(p = 30, n = 100, edge_prob = 0.05, seed = 1)
str(sim$truth)
```

---

simulate_with_latent *Simulate data with latent variables and oracle-FCI truth skeleton*

Description

This follows the exact latent SEM + oracle truth scheme:

- Build a DAG over (observed + latent) nodes with:
  - observed->observed edges only for i<j (acyclic)
  - latent->observed edges (Poisson out-degree)
- Simulate data from linear SEM with chosen error distribution
- Construct "truth" by running FCI on the ORACLE correlation of observed nodes using a very large virtual sample size and alpha_truth (oracle-ish), with m.max controlling speed (e.g., m.max = 2)

Usage

```
simulate_with_latent(
  p_obs = 100,
  gamma = 0.05,
  n = 100,
  edge_prob_obs = 0.02,
  latent_out_deg = 3,
  w_sd = 0.8,
  errDist = c("normal", "t4", "mixt3"),
  noise_sd = 1,
  mix = 0.05,
  seed_graph = 1,
  seed_data = 2,
  truth_alpha = 0.9999,
  truth_mmax = 2,
  truth_verbose = FALSE
)
```

Arguments

p_obs	Number of observed variables.
gamma	Latent ratio; p_lat = max(1, round(gamma * p_obs)).
n	Sample size.
edge_prob_obs	Edge probability among observed nodes (i<j only).
latent_out_deg	Mean outgoing degree for each latent to observed (Poisson).
w_sd	SD of nonzero edge weights.
errDist	Error distribution for SEM noise: "normal", "t4", "mixt3".
noise_sd	Noise SD multiplier.
mix	Mixing proportion for "mixt3" heavy tail component.
seed_graph	Seed controlling graph + weights.
seed_data	Seed controlling data noise draws.
truth_alpha	Alpha for oracle-truth FCI (typical: 0.9999).
truth_mmax	Maximum conditioning set size in oracle FCI (speed knob; e.g., 2).
truth_verbose	Logical; verbose output from oracle FCI.

Details

The returned truth is the *skeleton* implied by the oracle-FCI PAG (not marks).

Value

A list with elements: X, truth (skel + amat), meta, sem (A,W,indices).

See Also

pfci_fit, metrics_with_latent

Examples

```
sim <- simulate_with_latent(p_obs = 30, gamma = 0.05, n = 100, seed_graph = 1)
str(sim$truth)
```

Index