Package 'DPI'

September 20, 2025

Title The Directed Prediction Index for Quasi-Causal Inference with Cross-Sectional Data

Version 2025.9

Date 2025-09-20

Maintainer Han Wu Shuang Bao <baohws@foxmail.com>

Description The Directed Prediction Index ('DPI') is a quasi-causal inference method for cross-sectional data designed to quantify the relative endogeneity (relative dependence) of outcome (Y) versus predictor (X) variables in regression models. By comparing the proportion of variance explained (R-squared) between the Y-as-outcome model and the X-as-outcome model while controlling for a sufficient number of possible confounders, it suggests a plausible (admissible) direction of influence from a more exogenous variable (X) to a more endogenous variable (Y). Methodological details are provided at <https://psychbruce.github.io/DPI/>.

License GPL-3

Encoding UTF-8

URL https://psychbruce.github.io/DPI/

BugReports https://github.com/psychbruce/DPI/issues

Depends R (>= 4.0.0)

Imports glue, crayon, cli, ggplot2, cowplot, qgraph, bnlearn, MASS

Suggests bruceR, aplot

RoxygenNote 7.3.3

NeedsCompilation no

Author Han Wu Shuang Bao [aut, cre] (ORCID: <https://orcid.org/0000-0003-3043-710X>)

Repository CRAN

Date/Publication 2025-09-20 15:20:02 UTC

Contents

`cor_network` *Correlation and partial correlation networks.*

Description

Correlation and partial correlation networks (also called Gaussian graphical models, GGMs).

Usage

```
cor_network(
  data,
  index = c("cor", "pcor"),
  show.value = TRUE,
  show.insig = FALSE,
  show.cutoff = FALSE,
  faded = FALSE,
  node.text.size = 1.2,
  node.group = NULL,
  node.color = NULL,
  edge.color.pos = "#0571B0",
  edge.color.neg = "#CA0020",
  edge.color.non = "#EEEEEEEE",
  edge.label.mrg = 0.01,
  title = NULL,
  file = NULL,
  width = 6,
  height = 4,
  dpi = 500,
  ...
)
```

Arguments

`data`	Data.
`index`	Type of graph: `"cor"` (raw correlation network) or `"pcor"` (partial correlation network). Defaults to `"cor"`.

show.value	Show correlation coefficients and their significance on edges. Defaults to TRUE.
show.insig	Show edges with insignificant correlations ($p > 0.05$). Defaults to FALSE. To change significance level, please set alpha (defaults to alpha=0.05).
show.cutoff	Show cut-off values of correlations. Defaults to FALSE.
faded	Transparency of edges according to the effect size of correlation. Defaults to FALSE.
node.text.size	Scalar on the font size of node (variable) labels. Defaults to 1.2.
node.group	A list that indicates which nodes belong together, with each element of list as a vector of integers identifying the column numbers of variables that belong together.
node.color	A vector with a color for each element in node.group, or a color for each node.
edge.color.pos	Color for (significant) positive values. Defaults to "#0571B0" (blue in ColorBrewer's RdBu palette).
edge.color.neg	Color for (significant) negative values. Defaults to "#CA0020" (red in ColorBrewer's RdBu palette).
edge.color.non	Color for insignificant values. Defaults to "#EEEEEEEE" (transparent grey).
edge.label.mrg	Margin of the background box around the edge label. Defaults to 0.01.
title	Plot title.
file	File name of saved plot (".png" or ".pdf").
width, height	Width and height (in inches) of saved plot. Defaults to 6 and 4.
dpi	Dots per inch (figure resolution). Defaults to 500.
...	Arguments passed on to qgraph().

Value

Return a list (class cor.net) of (partial) correlation results and qgraph object with its grob (Grid Graphical Object).

See Also

S3method.network

dag_network()

Examples

```
# correlation network
cor_network(airquality)
cor_network(airquality, show.insig=TRUE)

# partial correlation network
cor_network(airquality, "pcor")
cor_network(airquality, "pcor", show.insig=TRUE)
```

dag_network *Directed acyclic graphs (DAGs) via Bayesian networks (BNs).*

Description

Directed acyclic graphs (DAGs) via Bayesian networks (BNs). It uses `bnlearn::boot.strength()` to estimate the strength of each edge as its *empirical frequency* over a set of networks learned from bootstrap samples. It computes (1) the probability of each edge (modulo its direction) and (2) the probabilities of each edge's directions conditional on the edge being present in the graph (in either direction). Stability thresholds are usually set as `0.85` for *strength* (i.e., an edge appearing in more than 85% of BNs bootstrap samples) and `0.50` for *direction* (i.e., a direction appearing in more than 50% of BNs bootstrap samples) (Briganti et al., 2023). Finally, for each chosen algorithm, it returns the stable Bayesian network as the final DAG.

Usage

```
dag_network(
  data,
  algorithm = c("pc.stable", "hc", "rsmax2"),
  algorithm.args = list(),
  n.boot = 1000,
  seed = NULL,
  strength = 0.85,
  direction = 0.5,
  node.text.size = 1.2,
  edge.width.max = 1.5,
  edge.label.mrg = 0.01,
  file = NULL,
  width = 6,
  height = 4,
  dpi = 500,
  verbose = TRUE,
  ...
)
```

Arguments

`data`	Data.
`algorithm`	Structure learning algorithms for building Bayesian networks (BNs). Should be function name(s) from the `bnlearn` package. Better to perform BNs with all three classes of algorithms to check the robustness of results (Briganti et al., 2023). Defaults to the most common algorithms: `"pc.stable"` (PC), `"hc"` (HC), and `"rsmax2"` (RS), for the three classes, respectively. • (1) Constraint-based Algorithms

  - PC: "pc.stable" (*the first practical constraint-based causal structure learning algorithm by Peter & Clark*)
  - Others: "gs", "iamb", "fast.iamb", "inter.iamb", "iamb.fdr"
- (2) Score-based Algorithms
  - Hill-Climbing: "hc" (*the hill-climbing greedy search algorithm, exploring DAGs by single-edge additions, removals, and reversals, with random restarts to avoid local optima*)
  - Others: "tabu"
- (3) Hybrid Algorithms (combination of constraint-based and score-based algorithms)
  - Restricted Maximization: "rsmax2" (*the general 2-phase restricted maximization algorithm, first restricting the search space and then finding the optimal [maximizing the score of] network structure in the restricted space*)
  - Others: "mmhc", "h2pc"

`algorithm.args` An optional list of extra arguments passed to the algorithm.

`n.boot` Number of bootstrap samples (for learning a more "stable" network structure). Defaults to `1000`.

`seed` Random seed for replicable results. Defaults to `NULL`.

`strength` Stability threshold of edge *strength*: the minimum proportion (probability) of BNs (among the `n.boot` bootstrap samples) in which each edge appears.

- Defaults to `0.85` (85%).
- Two reverse directions share the same edge strength.
- Empirical frequency (?~100%) will be mapped onto edge *width/thickness* in the final integrated `DAG`, with wider (thicker) edges showing stronger links, though they usually look similar since the default range has been limited to 0.85~1.

`direction` Stability threshold of edge *direction*: the minimum proportion (probability) of BNs (among the `n.boot` bootstrap samples) in which a direction of each edge appears.

- Defaults to `0.50` (50%).
- The proportions of two reverse directions add up to 100%.
- Empirical frequency (?~100%) will be mapped onto edge *greyscale/transparency* in the final integrated `DAG`, with its value shown as edge text label.

`node.text.size` Scalar on the font size of node (variable) labels. Defaults to `1.2`.

`edge.width.max` Maximum value of edge strength to scale all edge widths. Defaults to `1.5` for better display of arrow.

`edge.label.mrg` Margin of the background box around the edge label. Defaults to `0.01`.

`file` File name of saved plot (`".png"` or `".pdf"`).

`width, height` Width and height (in inches) of saved plot. Defaults to `6` and `4`.

`dpi` Dots per inch (figure resolution). Defaults to `500`.

`verbose` Print information about BN algorithm and number of bootstrap samples when running the analysis. Defaults to `TRUE`.

`...` Arguments passed on to `qgraph()`.

Value

Return a list (class `dag.net`) of Bayesian network results and `qgraph` object with its `grob` (Grid Graphical Object).

References

Briganti, G., Scutari, M., & McNally, R. J. (2023). A tutorial on Bayesian networks for psychopathology researchers. *Psychological Methods, 28*(4), 947–961. doi:10.1037/met0000479

Burger, J., Isvoranu, A.-M., Lunansky, G., Haslbeck, J. M. B., Epskamp, S., Hoekstra, R. H. A., Fried, E. I., Borsboom, D., & Blanken, T. F. (2023). Reporting standards for psychological network analyses in cross-sectional data. *Psychological Methods, 28*(4), 806–824. doi:10.1037/met0000471

Scutari, M., & Denis, J.-B. (2021). *Bayesian networks: With examples in R* (2nd ed.). Chapman and Hall/CRC. doi:10.1201/9780429347436

https://www.bnlearn.com/

See Also

S3method.network

`cor_network()`

Examples

```
bn = dag_network(airquality, seed=1)
bn
# bn$pc.stable
# bn$hc
# bn$rsmax2

## All DAG objects can be directly plotted
## or saved with print(..., file="xxx.png")
# bn$pc.stable$DAG.edge
# bn$pc.stable$DAG.strength
# bn$pc.stable$DAG.direction
# bn$pc.stable$DAG
# ...

## Not run:

print(bn, file="airquality.png")
# will save three plots with auto-modified file names:
- "airquality_DAG.NET_BNs.01_pc.stable.png"
- "airquality_DAG.NET_BNs.02_hc.png"
- "airquality_DAG.NET_BNs.03_rsmax2.png"

# arrange multiple plots using aplot::plot_list()
# install.packages("aplot")
c1 = cor_network(airquality, "cor")
c2 = cor_network(airquality, "pcor")
bn = dag_network(airquality, seed=1)
p = aplot::plot_list(
```

```
  c1$plot,
  c2$plot,
  bn$pc.stable$DAG$plot,
  bn$hc$DAG$plot,
  bn$rsmax2$DAG$plot,
  design="111222
          334455",
  tag_levels="A"
)  # return a patchwork object
ggsave(p, filename="p.png", width=12, height=8, dpi=500)
ggsave(p, filename="p.pdf", width=12, height=8)

## End(Not run)
```

DPI *The Directed Prediction Index (DPI).*

Description

The Directed Prediction Index (DPI) is a quasi-causal inference method for cross-sectional data designed to quantify the *relative endogeneity* (relative dependence) of outcome (Y) vs. predictor (X) variables in regression models. By comparing the proportion of variance explained (R-squared) between the Y-as-outcome model and the X-as-outcome model while controlling for a sufficient number of possible confounders, it suggests a plausible (admissible) direction of influence from a more exogenous variable (X) to a more endogenous variable (Y). Methodological details are provided at https://psychbruce.github.io/DPI/.

Usage

```
DPI(
  model,
  y,
  x,
  data = NULL,
  k.cov = 1,
  n.sim = 1000,
  alpha = 0.05,
  seed = NULL,
  progress,
  file = NULL,
  width = 6,
  height = 4,
  dpi = 500
)
```

Arguments

`model`	Model object (`lm`).
`y`	Dependent (outcome) variable.
`x`	Independent (predictor) variable.
`data`	[Optional] Defaults to `NULL`. If `data` is specified, then `model` will be ignored and a linear model `lm({y} ~ {x} + .)` will be fitted inside. This is helpful for exploring all variables in a dataset.
`k.cov`	Number of random covariates (simulating potential omitted variables) added to each simulation sample. • Defaults to `1`. Please also test different `k.cov` values as robustness checks (see `DPI_curve()`). • If `k.cov > 0`, the raw data (without bootstrapping) are used, with `k.cov` random variables appended, for simulation. • If `k.cov = 0` (not suggested), bootstrap samples (resampling with replacement) are used for simulation.
`n.sim`	Number of simulation samples. Defaults to `1000`.
`alpha`	Significance level for computing the `Strength` score (0~1) based on *p* value of partial correlation between `X` and `Y`. Defaults to `0.05`. • `Direction = R2.Y - R2.X` • `Strength = 1 - tanh(p.beta.xy/alpha/2)`
`seed`	Random seed for replicable results. Defaults to `NULL`.
`progress`	Show progress bar. Defaults to `FALSE` (if `n.sim < 5000`).
`file`	File name of saved plot (`".png"` or `".pdf"`).
`width, height`	Width and height (in inches) of saved plot. Defaults to `6` and `4`.
`dpi`	Dots per inch (figure resolution). Defaults to `500`.

Value

Return a data.frame of simulation results:

- `DPI`
  - `= Direction * Strength`
  - `= (R2.Y - R2.X) * (1 - tanh(p.beta.xy/alpha/2))`
- `delta.R2`
  - `R2.Y - R2.X`
- `R2.Y`
  - R^2 of regression model predicting Y using X and all other covariates
- `R2.X`
  - R^2 of regression model predicting X using Y and all other covariates
- `t.beta.xy`

  - *t* value for coefficient of X predicting Y (always equal to *t* value for coefficient of Y predicting X) when controlling for all other covariates

- `p.beta.xy`
  - *p* value for coefficient of X predicting Y (always equal to *p* value for coefficient of Y predicting X) when controlling for all other covariates
- `df.beta.xy`
  - residual degree of freedom (df) of `t.beta.xy`
- `r.partial.xy`
  - partial correlation (always with the same *t* value as `t.beta.xy`) between X and Y when controlling for all other covariates

See Also

S3method.dpi

`DPI_curve()`

`cor_network()`

`dag_network()`

Examples

```
# input a fitted model
model = lm(Ozone ~ ., data=airquality)
DPI(model, y="Ozone", x="Solar.R", seed=1)  # DPI > 0
DPI(model, y="Ozone", x="Wind", seed=1)     # DPI > 0
DPI(model, y="Wind", x="Solar.R", seed=1)   # unrelated

# input raw data, test with more random covs
DPI(data=airquality, y="Ozone", x="Solar.R", k.cov=10, seed=1)
DPI(data=airquality, y="Ozone", x="Wind", k.cov=10, seed=1)
DPI(data=airquality, y="Wind", x="Solar.R", k.cov=10, seed=1)
```

`DPI_curve` *The DPI curve analysis.*

Description

The DPI curve analysis.

Usage

```
DPI_curve(
  model,
  y,
  x,
  data = NULL,
  k.covs = 1:10,
  n.sim = 1000,
  alpha = 0.05,
  seed = NULL,
  file = NULL,
  width = 6,
  height = 4,
  dpi = 500
)
```

Arguments

`model`	Model object (`lm`).
`y`	Dependent (outcome) variable.
`x`	Independent (predictor) variable.
`data`	[Optional] Defaults to `NULL`. If `data` is specified, then `model` will be ignored and a linear model `lm({y} ~ {x} + .)` will be fitted inside. This is helpful for exploring all variables in a dataset.
`k.covs`	An integer vector of number of random covariates (simulating potential omitted variables) added to each simulation sample. Defaults to `1:10` (producing DPI results for `k.cov`=1~10). For details, see `DPI()`.
`n.sim`	Number of simulation samples. Defaults to `1000`.
`alpha`	Significance level for computing the `Strength` score (0~1) based on *p* value of partial correlation between `X` and `Y`. Defaults to `0.05`. • `Direction = R2.Y - R2.X` • `Strength = 1 - tanh(p.beta.xy/alpha/2)`
`seed`	Random seed for replicable results. Defaults to `NULL`.
`file`	File name of saved plot (`".png"` or `".pdf"`).
`width, height`	Width and height (in inches) of saved plot. Defaults to `6` and `4`.
`dpi`	Dots per inch (figure resolution). Defaults to `500`.

Value

Return a data.frame of DPI curve results.

See Also

S3method.dpi

`DPI()`

cor_network()

dag_network()

Examples

```
model = lm(Ozone ~ ., data=airquality)
DPIs = DPI_curve(model, y="Ozone", x="Solar.R", seed=1)
plot(DPIs)  # ggplot object
```

`matrix_cor` *Produce a symmetric correlation matrix from values.*

Description

Produce a symmetric correlation matrix from values.

Usage

```
matrix_cor(...)
```

Arguments

`...`	Correlation values to transform into the symmetric correlation matrix (by row).

Value

Return a symmetric correlation matrix.

Examples

```
matrix_cor(
  1.0, 0.7, 0.3,
  0.7, 1.0, 0.5,
  0.3, 0.5, 1.0
)
```

sim_data *Simulate data from a multivariate normal distribution.*

Description

Simulate data from a multivariate normal distribution.

Usage

```
sim_data(n, k, cor = NULL, exact = TRUE, seed = NULL)
```

Arguments

n	Number of observations (cases).
k	Number of variables. Will be ignored if cor specifies a correlation matrix.
cor	A correlation value or correlation matrix of the variables. Defaults to NULL that generates completely random data regardless of their empirical correlations.
exact	Ensure the sample correlation matrix to be exact as specified in cor. This argument is passed on to empirical in mvrnorm(). Defaults to TRUE.
seed	Random seed for replicable results. Defaults to NULL.

Value

Return a data.frame of simulated data.

See Also

matrix_cor()

sim_data_exp()

Examples

```
d1 = sim_data(n=100, k=5, seed=1)
cor_network(d1)

d2 = sim_data(n=100, k=5, cor=0.2, seed=1)
cor_network(d2)

cor.mat = matrix_cor(
  1.0, 0.7, 0.3,
  0.7, 1.0, 0.5,
  0.3, 0.5, 1.0
)
d3 = sim_data(n=100, cor=cor.mat, seed=1)
cor_network(d3)
```

`sim_data_exp` *Simulate experiment-like data with* independent *binary Xs.*

Description

Simulate experiment-like data with *independent* binary Xs.

Usage

```
sim_data_exp(
  n,
  r.xy,
  approx = TRUE,
  tol = 0.01,
  max.iter = 30,
  verbose = FALSE,
  seed = NULL
)
```

Arguments

`n`	Number of observations (cases).
`r.xy`	A vector of expected correlations of each X (binary independent variable: 0 or 1) with Y.
`approx`	Make the sample correlation matrix approximate more to values as specified in `r.xy`, using the method of orthogonal decomposition of residuals (i.e., making residuals more independent of Xs). Defaults to `TRUE`.
`tol`	Tolerance of absolute difference between specified and empirical correlations. Defaults to `0.01`.
`max.iter`	Maximum iterations for approximation. More iterations produce more approximate correlations, but the absolute differences will be convergent after about 30 iterations. Defaults to `30`.
`verbose`	Print information about iterations that satisfy tolerance. Defaults to `FALSE`.
`seed`	Random seed for replicable results. Defaults to `NULL`.

Value

Return a data.frame of simulated data.

See Also

`sim_data()`

Examples

```
data = sim_data_exp(n=1000, r.xy=c(0.5, 0.3), seed=1)
cor(data)  # tol = 0.01

data = sim_data_exp(n=1000, r.xy=c(0.5, 0.3), seed=1,
                    verbose=TRUE)
cor(data)  # print iteration information

data = sim_data_exp(n=1000, r.xy=c(0.5, 0.3), seed=1,
                    verbose=TRUE, tol=0.001)
cor(data)  # more approximate, though not exact

data = sim_data_exp(n=1000, r.xy=c(0.5, 0.3), seed=1,
                    approx=FALSE)
cor(data)  # far less exact
```

Index