Package 'DPI'

February 26, 2026

Title The Directed Prediction Index for Causal Direction Inference from Observational Data

Version 2026.2

Date 2026-02-25

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

Description The Directed Prediction Index ('DPI') is a causal discovery method for observational data designed to quantify the relative endogeneity of outcome (Y) versus predictor (X) variables in regression models. By comparing the coefficients of determination (R-squared) between the Y-as-outcome and X-as-outcome models while controlling for sufficient confounders and simulating k random covariates, it can quantify relative endogeneity, providing a necessary but insufficient condition for causal direction from a less endogenous variable (X) to a more endogenous variable (Y). Methodological details are provided at <https://psychbruce.github.io/DPI/>. This package also includes functions for data simulation and network analysis (correlation, partial correlation, and Bayesian Networks).

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, bayestestR

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 2026-02-26 07:50:03 UTC

Contents

`BNs_dag` *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

```
BNs_dag(
  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 NULL (for undirected correlation networks) and 1.5 (for directed acyclic networks to better display arrows).
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 bns.dag) of Bayesian network results and qgraph 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

DPI_dag()

cor_net()

Examples

```
bn = BNs_dag(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_BNs.DAG.01_pc.stable.png"
- "airquality_BNs.DAG.02_hc.png"
- "airquality_BNs.DAG.03_rsmax2.png"

# arrange multiple plots using aplot::plot_list()
# install.packages("aplot")
c1 = cor_net(airquality, "cor")
c2 = cor_net(airquality, "pcor")
bn = BNs_dag(airquality, seed=1)
mytheme = theme(plot.title=element_text(hjust=0.5))
p = aplot::plot_list(
  plot(c1),
  plot(c2),
  plot(bn$pc.stable$DAG) + mytheme,
  plot(bn$hc$DAG) + mytheme,
  plot(bn$rsmax2$DAG) + mytheme,
  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)
```

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

Description

Produce a symmetric correlation matrix from values.

Usage

```
cor_matrix(...)
```

Arguments

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

Value

Return a symmetric correlation matrix.

Examples

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

cor_matrix(
  1.0, NA, NA,
  0.7, 1.0, NA,
  0.3, 0.5, 1.0
)
```

---

cor_net *Correlation and partial correlation networks.*

Description

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

Usage

```
cor_net(
  data,
  index = c("cor", "pcor"),
  show.label = 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.width.min = "sig",
  edge.width.max = NULL,
  edge.label.mrg = 0.01,
  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.label	Show labels of 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" (faded light grey).
edge.width.min	Minimum value of edge strength to scale all edge widths. Defaults to sig (the threshold of significant values).
edge.width.max	Maximum value of edge strength to scale all edge widths. Defaults to NULL (for undirected correlation networks) and 1.5 (for directed acyclic networks to better display arrows).
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.
...	Arguments passed on to qgraph().

Value

Return a list (class cor.net) of (partial) correlation results and qgraph object.

See Also

S3method.network

DPI_dag()

BNs_dag()

Examples

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

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

# modify ggplot attributes
p = cor_net(airquality, "pcor")
gg = plot(p)  # return a ggplot object
gg + labs(title="Partial Correlation Network")
```

DPI *The Directed Prediction Index (DPI).*

Description

The Directed Prediction Index (DPI) is a causal discovery method for observational data designed to quantify the *relative endogeneity* of outcome (Y) vs. predictor (X) variables in regression models. By comparing the coefficients of determination (R-squared) between the Y-as-outcome and X-as-outcome models while controlling for sufficient confounders and simulating k random covariates, it can quantify relative endogeneity, providing a necessary but insufficient condition for causal direction 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,
  x,
  y,
  data = NULL,
  k.cov = 1,
  n.sim = 1000,
  alpha = 0.05,
  bonf = FALSE,
  pseudoBF = FALSE,
  seed = NULL,
  progress,
  file = NULL,
  width = 6,
  height = 4,
  dpi = 500
)
```

Arguments

`model`	Model object (`lm`).
`x`	Independent (predictor) variable.
`y`	Dependent (outcome) 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 *Normalized Penalty* score (0~1) based on *p* value of partial correlation between `X` and `Y`. Defaults to `0.05`.
`bonf`	Bonferroni correction to control for false positive rates: `alpha` is divided by, and *p* values are multiplied by, the number of comparisons. • Defaults to `FALSE`: No correction, suitable if you plan to test only one pair of variables. • `TRUE`: Using `k * (k - 1) / 2` (all pairs of variables) where `k = length(data)`. • A user-specified number of comparisons.
`pseudoBF`	Use normalized pseudo Bayes Factors `sigmoid(log(PseudoBF10))` alternatively as the *Normalized Penalty* score (0~1). Pseudo Bayes Factors are computed from *p* value of X-Y partial relationship and total sample size, using the transformation rules proposed by Wagenmakers (2022) doi:10.31234/osf.io/egydq. Defaults to `FALSE` because it makes less penalties for insignificant partial relationships between `X` and `Y`, see Examples in `DPI()` and online documentation.
`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 = Relative Endogeneity * Normalized Penalty`

  `– = (R2.Y - R2.X) * (1 - tanh(p.beta.xy/alpha/2))`

    - if `pseudoBF=FALSE` (default, suggested)
    - more conservative estimates
  - `= (R2.Y - R2.X) * plogis(log(pseudo.BF.xy))`
    - if `pseudoBF=TRUE`
    - less conservative for insignificant X-Y relationship

- `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
- `sigmoid.p.xy`
  - sigmoid *p* value as `1 - tanh(p.beta.xy/alpha/2)`
- `pseudo.BF.xy`
  - pseudo Bayes Factors (BF_{10}) computed from *p* value `p.beta.xy` and sample size `nobs(model)`, see `p_to_bf()`

See Also

S3method.dpi

`DPI_curve()`

`DPI_dag()`

`BNs_dag()`

`cor_net()`

`p_to_bf()`

Examples

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

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

# or use pseudo Bayes Factors for normalized penalty
# (less conservative for insignificant X-Y relationship)
DPI(data=airquality, x="Solar.R", y="Ozone", k.cov=10,
    pseudoBF=TRUE, seed=1)  # DPI > 0 (true positive)
DPI(data=airquality, x="Wind", y="Ozone", k.cov=10,
    pseudoBF=TRUE, seed=1)  # DPI > 0 (true positive)
DPI(data=airquality, x="Solar.R", y="Wind", k.cov=10,
    pseudoBF=TRUE, seed=1)  # DPI > 0 (false positive!)
```

`DPI_curve` *DPI curve analysis across multiple random covariates.*

Description

DPI curve analysis across multiple random covariates.

Usage

```
DPI_curve(
  model,
  x,
  y,
  data = NULL,
  k.covs = 1:10,
  n.sim = 1000,
  alpha = 0.05,
  bonf = FALSE,
  pseudoBF = FALSE,
  seed = NULL,
  progress,
  file = NULL,
  width = 6,
```

```
  height = 4,
  dpi = 500
)
```

Arguments

`model`	Model object (`lm`).
`x`	Independent (predictor) variable.
`y`	Dependent (outcome) 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 *Normalized Penalty* score (0~1) based on *p* value of partial correlation between `X` and `Y`. Defaults to `0.05`.
`bonf`	Bonferroni correction to control for false positive rates: `alpha` is divided by, and *p* values are multiplied by, the number of comparisons. • Defaults to `FALSE`: No correction, suitable if you plan to test only one pair of variables. • `TRUE`: Using `k * (k - 1) / 2` (all pairs of variables) where `k = length(data)`. • A user-specified number of comparisons.
`pseudoBF`	Use normalized pseudo Bayes Factors `sigmoid(log(PseudoBF10))` alternatively as the *Normalized Penalty* score (0~1). Pseudo Bayes Factors are computed from *p* value of X-Y partial relationship and total sample size, using the transformation rules proposed by Wagenmakers (2022) doi:10.31234/osf.io/egydq. Defaults to `FALSE` because it makes less penalties for insignificant partial relationships between `X` and `Y`, see Examples in `DPI()` and online documentation.
`seed`	Random seed for replicable results. Defaults to `NULL`.
`progress`	Show progress bar. Defaults to `TRUE` (if `length(k.covs)` >= 5).
`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()

DPI_dag()

BNs_dag()

cor_net()

p_to_bf()

Examples

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

DPI_dag *Directed acyclic graphs (DAGs) via DPI exploratory analysis (causal discovery) for all significant partial* rs.

Description

Directed acyclic graphs (DAGs) via DPI exploratory analysis (causal discovery) for all significant partial *r*s.

Usage

```
DPI_dag(
  data,
  k.covs = 1,
  n.sim = 1000,
  alpha = 0.05,
  bonf = FALSE,
  pseudoBF = FALSE,
  seed = NULL,
  node.text.size = 1.2,
  progress,
  file = NULL,
  width = 6,
  height = 4,
  dpi = 500
)
```

Arguments

`data`	A dataset with at least 3 variables.
`k.covs`	An integer vector (e.g., `1:10`) of number of random covariates (simulating potential omitted variables) added to each simulation sample. Defaults to `1`. For details, see `DPI()`.
`n.sim`	Number of simulation samples. Defaults to `1000`.
`alpha`	Significance level for computing the *Normalized Penalty* score (0~1) based on *p* value of partial correlation between `X` and `Y`. Defaults to `0.05`.
`bonf`	Bonferroni correction to control for false positive rates: `alpha` is divided by, and *p* values are multiplied by, the number of comparisons. • Defaults to `FALSE`: No correction, suitable if you plan to test only one pair of variables. • `TRUE`: Using `k * (k - 1) / 2` (all pairs of variables) where `k = length(data)`. • A user-specified number of comparisons.
`pseudoBF`	Use normalized pseudo Bayes Factors `sigmoid(log(PseudoBF10))` alternatively as the *Normalized Penalty* score (0~1). Pseudo Bayes Factors are computed from *p* value of X-Y partial relationship and total sample size, using the transformation rules proposed by Wagenmakers (2022) doi:10.31234/osf.io/egydq. Defaults to `FALSE` because it makes less penalties for insignificant partial relationships between `X` and `Y`, see Examples in `DPI()` and online documentation.
`seed`	Random seed for replicable results. Defaults to `NULL`.
`node.text.size`	Scalar on the font size of node (variable) labels. Defaults to `1.2`.
`progress`	Show progress bar. Defaults to `TRUE` (if `length(k.covs)` >= 5).
`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 (class `dpi.dag`) of DPI exploration results.

See Also

S3method.network

`DPI()`

`DPI_curve()`

`BNs_dag()`

`cor_net()`

`p_to_bf()`

Examples

```
# partial correlation networks (undirected)
cor_net(airquality, "pcor")

# directed acyclic graphs (grey edge = insignificant DPI)
dpi.dag = DPI_dag(airquality, k.covs=c(1,3,5), seed=1)
print(dpi.dag, k=1)  # DAG with DPI(k=1)
print(dpi.dag, k=3)  # DAG with DPI(k=3)
print(dpi.dag, k=5)  # DAG with DPI(k=5)

# set edge labels and edge transparency
# (grey edge = insignificant DPI)
print(dpi.dag, k=5, show.label=FALSE, faded.dpi=TRUE)

# modify ggplot attributes
gg = plot(dpi.dag, k=5, show.label=FALSE, faded.dpi=TRUE)
gg + labs(title="DAG with DPI (k=5)")

# visualize DPIs of multiple paths
ggplot(dpi.dag$DPI, aes(x=k.cov, y=DPI)) +
  geom_ribbon(
    aes(ymin=Sim.LLCI, ymax=Sim.ULCI, fill=path),
        alpha=0.1) +
  geom_line(aes(color=path), linewidth=0.7) +
  geom_point(aes(color=path)) +
  geom_hline(yintercept=0, color="red",
             linetype="dashed") +
  scale_y_continuous(limits=c(NA, 0.5)) +
  labs(color="Directed Prediction",
       fill="Directed Prediction") +
  theme_classic()
```

---

p_to_bf *Convert* p *values to approximate (pseudo) Bayes Factors (PseudoBF10).*

Description

Convert *p* values to approximate (pseudo) Bayes Factors (PseudoBF10). This transformation has been suggested by Wagenmakers (2022).

Usage

```
p_to_bf(p, n, log = FALSE, label = FALSE)
```

Arguments

p	*p* value(s).
n	Number of observations.
log	Return log(BF10) or raw BF10. Defaults to FALSE.
label	Add labels (i.e., names) to returned values. Defaults to FALSE.

Value

A (named) numeric vector of pseudo Bayes Factors (PseudoBF_{10}).

References

Wagenmakers, E.-J. (2022). *Approximate objective Bayes factors from p-values and sample size: The* $3p\sqrt{n}$ *rule.* PsyArXiv. doi:10.31234/osf.io/egydq

See Also

bayestestR::p_to_bf()

Examples

```
p_to_bf(0.05, 100)
p_to_bf(c(0.01, 0.05), 100)
p_to_bf(c(0.001, 0.01, 0.05, 0.1), 100, label=TRUE)
p_to_bf(c(0.001, 0.01, 0.05, 0.1), 1000, label=TRUE)
```

---

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

cor_matrix()

sim_data_exp()

Examples

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

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

cor.mat = cor_matrix(
  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_net(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