Package ‘SEMgraph’

February 11, 2024

Title Network Analysis and Causal Inference Through Structural Equation Modeling

Version 1.2.1

Date 2024-02-01

Description Estimate networks and causal relationships in complex systems through Structural Equation Modeling. This package also includes functions to import, weight, manipulate, and fit biological network models within the Structural Equation Modeling framework proposed in Grassi M, Palluzzi F, Tarantino B (2022) <doi:10.1093/bioinformatics/btac567>.

URL `https://github.com/fernandoPalluzzi/SEMgraph`

Depends igraph (>= 1.6.0), lavaan (>= 0.6-1), R (>= 4.0)

Imports aspect, boot, corpcor, dagitty, flip, gdata, ggm, glasso, glmnet, graph, mgcv, mvtnorm, pbapply, protoclust, RBGL, Rgraphviz

License GPL-3

LazyData true

Encoding UTF-8

NeedsCompilation no

RoxygenNote 7.3.1

Maintainer Barbara Tarantino <`barbara.tarantino01@universitadipavia.it`>

Repository CRAN

Author Mario Grassi [aut],
Fernando Palluzzi [aut],
Barbara Tarantino [aut, cre]

R topics documented:

`alsData` *Amyotrophic Lateral Sclerosis (ALS) dataset*

Description

Expression profiling through high-throughput sequencing (RNA-seq) of 139 ALS patients and 21 healthy controls (HCs), from Tam et al. (2019).

Usage

```
alsData
```

Format

alsData is a list of 4 objects:

1. "graph", ALS graph as the largest connected component of the "Amyotrophic lateral sclerosis (ALS)" pathway from KEGG database;
2. "exprs", a matrix of 160 rows (subjects) and 318 columns (genes) extracted from the original 17695. This subset includes genes from KEGG pathways, needed to run SEMgraph examples. Raw data from the GEO dataset GSE124439 (Tam et al., 2019) were pre-processed applying batch effect correction, using the sva R package (Leek et al., 2012), to remove data production center and brain area biases. Using multidimensional scaling-based clustering, ALS-specific and an HC-specific clusters were generated. Misclassified samples were blacklisted and removed from the current dataset;
3. "group", a binary group vector of 139 ALS subjects (1) and 21 healthy controls (0);
4. "details", a data.frame reporting information about included and blacklisted samples.

Source

`https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE124439/`

References

Tam OH, Rozhkov NV, Shaw R, Kim D et al. (2019). Postmortem Cortex Samples Identify Distinct Molecular Subtypes of ALS: Retrotransposon Activation, Oxidative Stress, and Activated Glia. Cell Repprts, 29(5):1164-1177.e5. <https://doi.org/10.1016/j.celrep.2019.09.066>

Jeffrey T. Leek, W. Evan Johnson, Hilary S. Parker, Andrew E. Jaffe, and John D. Storey (2012). The sva package for removing batch effects and other unwanted variation in high-throughput experiments. Bioinformatics. Mar 15; 28(6): 882-883. <https://doi.org/10.1093/bioinformatics/bts034>

Examples

```
alsData$graph
dim(alsData$exprs)
table(alsData$group)
```

`ancestry` *Node ancestry utilities*

Description

Get ancestry for a collection of nodes in a graph. These functions are wrappers for the original `SEMID` R package.

Usage

```
ancestors(g, nodes)

descendants(g, nodes)

parents(g, nodes)

siblings(g, nodes)
```

Arguments

g	An igraph object.
nodes	the nodes in the graph of which to get the ancestry.

Value

a sorted vector of nodes.

References

Rina Foygel Barber, Mathias Drton and Luca Weihs (2019). SEMID: Identifiability of Linear Structural Equation Models. R package version 0.3.2. <https://CRAN.R-project.org/package=SEMID/>

Examples

```
# Get all ancestors
an <- V(sachs$graph)[ancestors(sachs$graph, "Erk")]; an

# Get parents
pa <- V(sachs$graph)[parents(sachs$graph, "PKC")]; pa

# Get descendants
de <- V(sachs$graph)[descendants(sachs$graph, "PKA")]; de

# Get siblings
sib <- V(sachs$graph)[siblings(sachs$graph, "PIP3")]; sib
```

clusterGraph *Topological graph clustering*

Description

Topological graph clustering methods.

Usage

```
clusterGraph(graph, type = "wtc", HM = "none", size = 5, verbose = FALSE, ...)
```

Arguments

`graph` An igraph object.

`type` Topological clustering methods. If `type = "tahc"`, network modules are generated using the tree agglomerative hierarchical clustering method (Yu et al., 2015). Other non-tree clustering methods from `igraph` package include: "wtc" (default value; walktrap community structure with short random walks), "ebc" (edge betweeness clustering), "fgc" (fast greedy method), "lbc" (label propagation method), "lec" (leading eigenvector method), "loc" (multi-level optimization), "opc" (optimal community structure), "sgc" (spinglass statistical mechanics).

`HM` Hidden model type. Enables the visualization of the hidden model, gHM. If set to "none" (default), no gHM igraph object is saved. For each defined hidden module: (i) if `HM = "LV"`, a latent variable (LV) will be defined as common unknown cause acting on cluster nodes; (ii) if `HM = "CV"`, cluster nodes will be considered as regressors of a latent composite variable (CV); (iii) if `HM = "UV"`, an unmeasured variable (UV) is defined, where source nodes of the module (i.e., in-degree = 0) act as common regressors influencing the other nodes via an unmeasured variable (see also `clusterScore`).

`size` Minimum number of nodes per module. By default, a minimum number of 5 nodes is required.

`verbose` A logical value. If FALSE (default), the gHM igraph will not be plotted to screen, saving execution time (they will be returned in output anyway).

`...` Currently ignored.

Value

If HM is not "none" a list of 2 objects is returned:

1. "gHM", subgraph containing hidden modules as an igraph object;
2. "membership", cluster membership vector for each node.

If HM is "none", only the cluster membership vector is returned.

Author(s)

Mario Grassi `<mario.grassi@unipv.it>`

References

Fortunato S, Hric D. Community detection in networks: A user guide (2016). Phys Rep; 659: 1-44. <https://dx.doi.org/10.1016/j.physrep.2016.09.002>

Yu M, Hillebrand A, Tewarie P, Meier J, van Dijk B, Van Mieghem P, Stam CJ (2015). Hierarchical clustering in minimum spanning trees. Chaos 25(2): 023107. <https://doi.org/10.1063/1.4908014>

See Also

`clusterScore`, `cplot`

Examples

```
# Clustering ALS graph with WTC method and LV model
G <- properties(alsData$graph)[[1]]
clv <- clusterGraph(graph = G, type = "wtc", HM = "LV")
gplot(clv$gHM, l = "fdp")
table(clv$membership)
```

`clusterScore` *Module scoring*

Description

Generate factor scores, principal component scores, or projection scores of latent, composite, and unmeasured variable modules, respectively, and fit them with an exogenous group effect.

Usage

```
clusterScore(
  graph,
  data,
  group,
  HM = "LV",
  type = "wtc",
  size = 5,
  verbose = FALSE,
  ...
)
```

Arguments

`graph` An igraph object.

`data` A matrix or data.frame. Rows correspond to subjects, and columns to graph nodes.

`group` A binary vector. This vector must be as long as the number of subjects. Each vector element must be 1 for cases and 0 for control subjects.

`HM` Hidden model type. For each defined hidden module: (i) if `HM = "LV"`, a latent variable (LV) will be defined as common unknown cause acting on cluster nodes; (ii) if `HM = "CV"`, cluster nodes will be considered as regressors of a latent composite variable (CV); (iii) if `HM = "UV"`, an unmeasured variable (UV) model will be generated for each module, where source nodes (i.e., in-degree = 0) act as common regressors influencing the other nodes via an unmeasured variable. By default, HM is set to "LV" (i.e., the latent variable model).

`type` Graph clustering method. If `type = "tahc"`, network modules are generated using the tree agglomerative hierarchical clustering method (Yu et al., 2015). Other non-tree clustering methods from igraph package include: "wtc" (default value; walktrap community structure with short random walks), "ebc" (edge betweenness clustering), "fgc" (fast greedy method), "lbc" (label propagation method), "lec" (leading eigenvector method), "loc" (multi-level optimization), "opc" (optimal communiy structure), "sgc" (spinglass statistical mechanics). By default, the "wtc" method is used.

`size` Minimum number of nodes per hidden module. By default, a minimum number of 5 nodes is required.

`verbose` A logical value. If TRUE, intermediate graphs will be displayed during the execution. In addition, a reduced graph with clusters as nodes will be fitted and showed to screen (see also `mergeNodes`). By default, `verbode = FALSE`.

`...` Currently ignored.

Value

A list of 3 objects:

1. "fit", hidden module fitting as a lavaan object;
2. "membership", hidden module nodes membership; `clusterGraph` function;
3. "dataHM", data matrix with cluster scores in first columns.

Author(s)

Mario Grassi <`mario.grassi@unipv.it`>

References

Grassi M, Palluzzi F, Tarantino B (2022). SEMgraph: An R Package for Causal Network Analysis of High-Throughput Data with Structural Equation Models. Bioinformatics, 38 (20), 4829–4830 <https://doi.org/10.1093/bioinformatics/btac567>

See Also

See `clusterGraph` and `cplot` for graph clustering.

Examples

```
# Nonparanormal(npn) transformation
als.npn <- transformData(alsData$exprs)$data

C <- clusterScore(graph = alsData$graph, data = als.npn,
                  group = alsData$group,
                  HM = "LV",
                  type = "wtc",
                  verbose = FALSE)
summary(C$fit)
head(C$dataHM)
```

```
table(C$membership)
```

colorGraph *Vertex and edge graph coloring on the base of fitting*

Description

Add vertex and edge color attributes to an igraph object, based on a fitting results data.frame generated by SEMrun.

Usage

```
colorGraph(
  est,
  graph,
  group,
  method = "none",
  alpha = 0.05,
  vcolor = c("lightblue", "white", "pink"),
  ecolor = c("royalblue3", "gray50", "red2"),
  ewidth = c(1, 2),
  ...
)
```

Arguments

est	A data.frame of estimated parameters and p-values, derived from the `fit` object returned by SEMrun. As an alternative, the user may provide a "gest" or "dest" data.frame generated by SEMrun.
graph	An igraph object.
group	group A binary vector. This vector must be as long as the number of subjects. Each vector element must be 1 for cases and 0 for control subjects.
method	Multiple testing correction method. One of the values available in `p.adjust`. By default, method is set to "none" (i.e., no multiple test correction).
alpha	Significance level for node and edge coloring (by default, `alpha = 0.05`).
vcolor	A vector of three color names. The first color is given to nodes with P-value < alpha and beta < 0, the third color is given to nodes with P-value < alpha and beta > 0, and the second is given to nodes with P-value > alpha. By default, `vcolor = c("lightblue", "white", "pink")`.
ecolor	A vector of three color names. The first color is given to edges with P-value < alpha and regression coefficient < 0, the third color is given to edges with P-value < alpha and regression coefficient > 0, and the second is given to edges with P-value > alpha. By default, `vcolor = c("blue", "gray50", "red2")`.

`ewidth`	A vector of two values. The first value refers to the basic edge width (i.e., edges with P-value > alpha), while the second is given to edges with P-value < alpha. By default ewidth = c(1, 2).
`...`	Currently ignored.

Value

An igraph object with vertex and edge color and width attributes.

Author(s)

Mario Grassi <`mario.grassi@unipv.it`>

Examples

```
# Model fitting: node perturbation
sem1 <- SEMrun(graph = alsData$graph, data = alsData$exprs,
               group = alsData$group,
               fit = 1)
est1 <- parameterEstimates(sem1$fit)

# Model fitting: edge perturbation
sem2 <- SEMrun(graph = alsData$graph, data = alsData$exprs,
               group = alsData$group,
               fit = 2)
est20 <- subset(parameterEstimates(sem2$fit), group == 1)[, -c(4, 5)]
est21 <- subset(parameterEstimates(sem2$fit), group == 2)[, -c(4, 5)]

# Graphs
g <- alsData$graph
x <- alsData$group

old.par <- par(no.readonly = TRUE)
par(mfrow=c(2,2), mar=rep(1,4))
gplot(colorGraph(est = est1, g, group = x, method = "BH"),
      main = "vertex differences")
gplot(colorGraph(est = sem2$dest, g, group = NULL),
      main = "edge differences")
gplot(colorGraph(est = est20, g, group = NULL),
      main = "edges for group = 0")
gplot(colorGraph(est = est21, g, group = NULL),
      main = "edges for group = 1")
par(old.par)
```

cplot *Subgraph mapping*

Description

Map groups of nodes onto an input graph, based on a membership vector.

Usage

```
cplot(graph, membership, l = layout.auto, map = FALSE, verbose = FALSE, ...)
```

Arguments

graph	An igraph object.
membership	Cluster membership vector for each node.
l	graph layout. One of the igraph layouts. If this argument is ignored, an automatic layout will be applied.
map	A logical value. Visualize cluster mapping over the input graph. If FALSE (default), visualization will be disabled. For large graphs, visualization may take long.
verbose	A logical value. If FALSE (default), the processed graphs will not be plotted to screen, saving execution time (they will be returned in output anyway).
...	Currently ignored.

Value

The list of clusters and cluster mapping as igraph objects.

Author(s)

Mario Grassi <mario.grassi@unipv.it>

See Also

clusterGraph, clusterScore

Examples

```
# Clustering ALS graph with WTC method
G <- alsData$graph
membership <- clusterGraph(graph = G, type = "wtc")
cplot(G, membership, map = TRUE, verbose = FALSE)
cplot(G, membership, map = FALSE, verbose = TRUE)
# The list of cluster graphs !
cg <- cplot(G, membership); cg
```

dagitty2graph *Graph conversion from dagitty to igraph*

Description

Convert a dagitty object to a igraph object.

Usage

```
dagitty2graph(dagi, verbose = FALSE, ...)
```

Arguments

dagi	A graph as a dagitty object ("dag" or "pdag").
verbose	A logical value. If TRUE, the output graph is shown. This argument is FALSE by default.
...	Currently ignored.

Value

An igraph object.

Author(s)

Mario Grassi <mario.grassi@unipv.it>

Examples

```
# Conversion from igraph to dagitty  (and viceversa)
dagi <- graph2dagitty(sachs$graph, verbose = TRUE)
graph <- dagitty2graph(dagi, verbose = TRUE)
```

extractClusters *Cluster extraction utility*

Description

Extract and fit clusters from an input graph.

Usage

```
extractClusters(
  graph,
  data,
  group = NULL,
  membership = NULL,
  map = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

`graph`	Input network as an igraph object.
`data`	A matrix or data.frame. Rows correspond to subjects, and columns to graph nodes (variables).
`group`	A binary vector. This vector must be as long as the number of subjects. Each vector element must be 1 for cases and 0 for control subjects. Group specification enables node perturbation testing. By default, `group = NULL`.
`membership`	A vector of cluster membership IDs. If NULL, clusters will be automatically generated with `clusterGraph` using the edge betweenness clustering ("ebc") algorithm.
`map`	Logical value. If TRUE, the plot of the input graph (coloured by cluster membership) will be generated along with independent module plots. If the input graph is very large, plotting could be computationally intensive (by default, `map = FALSE`).
`verbose`	Logical value. If TRUE, a plot will be showed for each cluster.
`...`	Currently ignored.

Value

A list of 3 objects:

1. "clusters", list of clusters as igraph objects;
2. "fit", list of fitting results for each cluster as a lavaan object;
3. "dfc", data.frame of summary results.

Author(s)

Fernando Palluzzi <fernando.palluzzi@gmail.com>

Examples

```
# Nonparanormal(npn) transformation
als.npn <- transformData(alsData$exprs)$data

adjdata <- SEMbap(alsData$graph, als.npn)$data

# Clusters creation
clusters <- extractClusters(alsData$graph, adjdata, alsData$group)
print(clusters$dfc)
head(parameterEstimates(clusters$fit$HM1))
head(parameterEstimates(clusters$fit$HM2))
head(parameterEstimates(clusters$fit$HM4))
gplot(clusters$clusters$HM2)

# Map cluster on the input graph
g <- alsData$graph
c <- clusters$clusters$HM2
V(g)$color <- ifelse(V(g)$name %in% V(c)$name, "gold", "white")
gplot(g)
```

factor.analysis *Factor analysis for high dimensional data*

Description

Wrapper for Factor Analysis with potentially high dimensional variables implement in the "cate" R package (Author: Jingshu Wang [aut], Qingyuan Zhao [aut, cre] Maintainer: Qingyuan Zhao <qz280@cam.ac.uk>) that is optimized for the high dimensional problem where the number of samples n is less than the number of variables p.

Usage

```
factor.analysis(Y, r = 1, method = "pc")
```

Arguments

Y	data matrix, a n*p matrix
r	number of factors (default, r =1)
method	algorithm to be used, "pc" (default) or "ml"

Details

The two methods extracted from "cate" are quasi-maximum likelihood (ml), and principal component analysis (pc). The ml is iteratively solved the EM algorithm using the PCA solution as the initial value. See Bai and Li (2012) for more details.

Value

a list of objects

Gamma estimated factor loadings

Z estimated latent factors

Sigma estimated noise variance matrix

References

Jushan Bai and Kunpeng Li (2012). Statistical Analysis of Factor Models of High Dimension. The Annals of Statistics, 40 (1), 436-465 <https://doi.org/10.1214/11-AOS966>

Jingshu Wang and Qingyuan Zhao (2020). cate: High Dimensional Factor Analysis and Confounder Adjusted Testing and Estimation. R package version 1.1.1. <https://CRAN.R-project.org/package=cate>

Examples

```
# Nonparanormal(npn) transformation
als.npn <- transformData(alsData$exprs)$data

## pc
pc<- factor.analysis(Y = als.npn, r = 2, method = "pc")
head(pc$Gamma)
head(pc$Z)
head(pc$Sigma)

## ml
ml <- factor.analysis(Y = als.npn, r = 2, method = "ml")
head(ml$Gamma)
head(ml$Z)
head(ml$Sigma)
```

---

gplot *Graph plotting with renderGraph*

Description

Wrapper for function renderGraph of the R package Rgraphwiz.

Usage

```
gplot(
  graph,
  l = "dot",
  main = "",
  cex.main = 1,
  font.main = 1,
  color.txt = "black",
  fontsize = 16,
  cex = 0.6,
  shape = "circle",
  color = "gray70",
  lty = 1,
  lwd = 1,
  w = "auto",
  h = "auto",
  psize = 80,
  ...
)
```

Arguments

`graph` An igraph or graphNEL object.

`l` Any layout supported by `Rgraphviz`. It can be one among: "dot" (default), "neato", "circo", "fdp", "osage", "twopi".

`main` Plot main title (by default, no title is added).

`cex.main` Main title size (default = 1).

`font.main` Main title font (default = 1). Available options are: 1 for plain text, 2 for bold, 3 for italics, 4 for bold italics, and 5 for symbol.

`color.txt` Node text color (default = "black").

`fontsize` Node text size (default = 16).

`cex` Another argument to control node text size (default = 0.6).

`shape` Node shape (default = "circle").

`color` Node border color (default = "gray70").

`lty` Node border outline (default = 1). Available options include: 0 for blank, 1 for solid line, 2 for dashed, 3 for dotted, 4 for dotdash, 5 for longdash, and 6 for twodash.

`lwd` Node border thickness (default = 1).

`w` Manual node width (default = "auto").

`h` Manual node height (default = "auto").

`psize` Automatic node size (default = 80).

`...` Currently ignored.

Value

gplot returns invisibly the graph object produced by Rgraphviz

Author(s)

Mario Grassi <mario.grassi@unipv.it>

Examples

```
gplot(sachs$graph, main = "input graph")

sem <- SEMrun(sachs$graph, sachs$pkc)
gplot(sem$graph, main = "output graph")
```

graph2dag *Convert directed graphs to directed acyclic graphs (DAGs)*

Description

Remove cycles and bidirected edges from a directed graph.

Usage

```
graph2dag(graph, data, bap = FALSE, time.limit = Inf, ...)
```

Arguments

graph	A directed graph as an igraph object.
data	A data matrix with subjects as rows and variables as columns.
bap	If TRUE, a bow-free acyclic path (BAP) is returned (default = FALSE).
time.limit	CPU time for the computation, in seconds (default = Inf).
...	Currently ignored.

Details

The conversion is performed firstly by removing bidirected edges and then the data matrix is used to compute edge P-values, through marginal correlation testing (see weightGraph, r-to-z method). When a cycle is detected, the edge with highest P-value is removed, breaking the cycle. If the bap argument is TRUE, a BAP is generated merging the output DAG and the bidirected edges from the input graph.

Value

A DAG as an igraph object.

Author(s)

Mario Grassi <mario.grassi@unipv.it>

Examples

```
dag <- graph2dag(graph = sachs$graph, data = log(sachs$pkc))
old.par <- par(no.readonly = TRUE)
par(mfrow=c(1,2), mar=rep(1, 4))
gplot(sachs$graph, main = "Input graph")
gplot(dag, main = "Output DAG")
par(old.par)
```

---

graph2dagitty *Graph conversion from igraph to dagitty*

Description

Convert an igraph object to a dagitty object.

Usage

```
graph2dagitty(graph, graphType = "dag", verbose = FALSE, ...)
```

Arguments

graph	A graph as an igraph or as an adjacency matrix.
graphType	character, is one of "dag" (default)' or "pdag". DAG can contain the directed (->) and bi-directed (<->) edges, while PDAG can contain the edges: ->, <->, and the undirected edges (–) that represent edges whose direction is not known.
verbose	A logical value. If TRUE, the output graph is shown. This argument is FALSE by default.
...	Currently ignored.

Value

A dagitty object.

Author(s)

Mario Grassi <mario.grassi@unipv.it>

Examples

```
# Graph as an igraph object to dagitty object
G <- graph2dagitty(sachs$graph)
plot(dagitty::graphLayout(G))
```

graph2lavaan *Graph to lavaan model*

Description

Convert an igraph object to a model (lavaan syntax).

Usage

```
graph2lavaan(graph, nodes = V(graph)$name, ...)
```

Arguments

`graph`	A graph as an igraph object.
`nodes`	Subset of nodes to be included in the model. By default, all the input graph nodes will be included in the output model.
`...`	Currently ignored.

Value

A model in lavaan syntax.

Author(s)

Mario Grassi <`mario.grassi@unipv.it`>

Examples

```
# Graph (igraph object) to structural model in lavaan syntax
model <- graph2lavaan(sachs$graph)
cat(model, "\n")
```

kegg *KEGG interactome*

Description

Interactome generated by merging KEGG pathways extracted using the `ROntoTools` R package (update: February, 2024).

Usage

```
kegg
```

Format

"kegg" is an igraph network object of 5143 nodes and 43734 edges (40424 directed and 3310/2 = 1655 bidirected) corresponding to the union of 227 KEGG pathways.

Source

https://www.genome.jp/kegg/

References

Kanehisa M, Goto S (1999). KEGG: kyoto encyclopedia of genes and genomes. Nucleic Acid Research 28(1): 27-30. <https://doi.org/10.1093/nar/27.1.29>

Calin Voichita, Sahar Ansari and Sorin Draghici (2023). ROntoTools: R Onto-Tools suite. R package version 2.30.0.

Examples

```
# KEGG graph
summary(kegg)

# KEGG degrees of freedom
vcount(kegg)*(vcount(kegg) - 1)/2 - ecount(kegg)
```

kegg.pathways *KEGG pathways*

Description

KEGG pathways extracted using the `ROntoTools` R package (update: February, 2024).

Usage

```
kegg.pathways
```

Format

"kegg.pathways" is a list of 227 igraph objects corresponding to the KEGG pathways.

Source

https://www.genome.jp/kegg/

References

Kanehisa M, Goto S (1999). KEGG: kyoto encyclopedia of genes and genomes. Nucleic Acid Research 28(1): 27-30. <https://doi.org/10.1093/nar/27.1.29>

Calin Voichita, Sahar Ansari and Sorin Draghici (2023). ROntoTools: R Onto-Tools suite. R package version 2.30.0.

Examples

```
library(igraph)

# KEGG pathways
names(kegg.pathways)

i<-which(names(kegg.pathways)=="Type II diabetes mellitus");i
ig<- kegg.pathways[[i]]
summary(ig)
V(ig)$name
E(ig)$weight

gplot(ig, l="fdp", psize=50, main=names(kegg.pathways[i]))
```

lavaan2graph *lavaan model to graph*

Description

Convert a model, specified using lavaan syntax, to an igraph object.

Usage

```
lavaan2graph(model, directed = TRUE, psi = TRUE, verbose = FALSE, ...)
```

Arguments

model	Model specified using lavaan syntax.
directed	Logical value. If TRUE (default), edge directions from the model will be preserved. If FALSE, the resulting graph will be undirected.
psi	Logical value. If TRUE (default) covariances will be converted into bidirected graph edges. If FALSE, covariances will be excluded from the output graph.
verbose	Logical value. If TRUE, a plot of the output graph will be generated. For large graphs, this could significantly increase computation time. If FALSE (default), graph plotting will be disabled.
...	Currently ignored.

Value

An igraph object.

Author(s)

Mario Grassi <mario.grassi@unipv.it>

Examples

```
# Writing path diagram in lavaan syntax

model<-"
#path model
Jnk ~ PKA + PKC
P38 ~ PKA + PKC
Akt ~ PKA + PIP3
Erk ~ PKA + Mek
Mek ~ PKA + PKC + Raf
Raf ~ PKA + PKC
PKC ~ PIP2 + Plcg
PIP2 ~ PIP3 + Plcg
Plcg ~ PIP3

#(co)variances
PKA ~~ PIP3
"

# Graph with covariances
G0 <- lavaan2graph(model, psi = TRUE)
plot(G0, layout = layout.circle)

# Graph without covariances
G1 <- lavaan2graph(model, psi = FALSE)
plot(G1, layout = layout.circle)
```

localCI.test *Conditional Independence (CI) local tests of an acyclic graph*

Description

P-values of one minimal testable implication (with the smallest possible conditioning set) is returned per missing edge given an acyclic graph (DAG or BAP) using the function impliedConditionalIndependencies plus the function localTests from package dagitty. Without assuming any particular dependence structure, the p-values of every CI test, in a DAG (BAP), is then combined using the Bonferroni's statistic in an overall test of the fitted model, B = K*min(p1,...,pK), as reviewed in Vovk & Wang (2020).

Usage

```
localCI.test(graph, data, bap = FALSE, limit = 100, verbose = TRUE, ...)
```

Arguments

graph	A directed graph as an igraph object.
data	A data matrix with subjects as rows and variables as columns.

`bap` If TRUE, the input graph is trasformend in a BAP, if FALSE (defult) the input graph is reduced in a DAG.

`limit` An integer value corresponding to the size of the extracted acyclic graph. Beyond this limit, switch to Shipley's C-test (Shipley 2000) is enabled to reduce the computational burden. By default, `limit = 100`.

`verbose` If TRUE, LocalCI results will be showed to screen (default = TRUE).

`...` Currently ignored.

Value

A list of three objects: (i) "dag": the DAG used to perform the localCI test (ii) "msep": the list of all m-separation tests over missing edges in the input graph and (iii) "mtest":the overall Bonferroni's P-value.

Author(s)

Mario Grassi <`mario.grassi@unipv.it`>

References

Vovk V, Wang R (2020). Combining p-values via averaging. Biometrika 107(4): 791-808. <https://doi.org/10.1093/biomet/as

Shipley B (2000). A new inferential test for path models based on DAGs. Structural Equation Modeling, 7(2): 206-218. <https://doi.org/10.1207/S15328007SEM0702_4>

Examples

```
# Nonparanormal(npn) transformation
als.npn <- transformData(alsData$exprs)$data

sem <- SEMrun(alsData$graph, als.npn)
B_test <- localCI.test(sem$graph, als.npn, verbose = TRUE)
```

mergeNodes *Graph nodes merging by a membership attribute*

Description

Merge groups of graph nodes using hierarchical clustering with prototypes derived from `protoclust` or custom membership attribute (e.g., cluster membership derived from `clusterGraph`).

Usage

```
mergeNodes(
  graph,
  data,
  h = 0.5,
  membership = NULL,
  HM = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

`graph`	network as an igraph object.
`data`	A matrix or data.frame. Rows correspond to subjects, and columns to graph nodes. If `membership` is not NULL, is currently ignored, `data = NULL`.
`h`	Cutting the minimax clustering at height, h = 1 - abs(cor(j,k)), yielding a merged node (and a reduced data set) in which every node in the cluster has correlation of at least cor(j,k) with the prototype node. By default, `h = 0.5`, i.e. cor(j,k) = 0.5.
`membership`	Cluster membership. A vector of cluster membership identifiers as numeric values, where vector names correspond to graph node names. By default, `membership = NULL`.
`HM`	Hidden cluster label. If membership is derived from clusterGraph: HM = "LV", a latent variable (LV) will be defined as common unknown cause acting on cluster nodes. If HM = "CV", cluster nodes will be considered as regressors of a latent composite variable (CV). Finally, if HM = "UV", an unmeasured variable (UV) is defined, where source nodes of the module (i.e., in-degree = 0) act as common regressors influencing the other nodes via an unmeasured variable. By default, `HM = NULL`
`verbose`	A logical value. If FALSE (default), the merged graphs will not be plotted to screen.
`...`	Currently ignored.

Details

Hierarchical clustering with prototypes (or Minmax linkage) is unique in naturally associating a node (the prototypes) with every interior node of the dendogram. Thus, for each merge we have a single representative data point for the resulting cluster (Bien, Tibshirani, 2011). These prototypes can be used to greatly enhance the interpretability of merging nodes and data reduction for SEM fitting.

Value

A list of 2 objects is returned:

1. "gLM", A graph with merged nodes as an igraph object;
2. "membership", cluster membership vector for each node.

Author(s)

Mario Grassi <mario.grassi@unipv.it>

References

Bien J, Tibshirani R (2011). Hierarchical Clustering With Prototypes via Minimax Linkage. Journal of the American Statistical Association 106(495): 1075-1084. <doi:10.1198/jasa.2011.tm10183>

See Also

clusterGraph

Examples

```
# Gene memberships with prototypes with h=0.5
G <- properties(alsData$graph)[[1]]
M <- mergeNodes(G, data = alsData$exprs, h = 0.5, verbose=TRUE)

# Gene memberships with EBC method and size=10
m <- clusterGraph(G, type = "ebc", size = 10)
M <- mergeNodes(G, membership = m, HM = "LV", verbose=TRUE)

# Gene memberships defined by user
c1 <- c("5894", "5576", "5567", "572", "598")
c2 <- c("6788", "84152", "2915", "836", "5530")
c3 <- c("5603", "6300", "1432", "5600")
m <- c(rep(1,5), rep(2,5), rep(3,4))
names(m) <- c(c1, c2, c3)
M <- mergeNodes(G, membership = m, HM = "CV", verbose=TRUE)
```

modelSearch *Optimal model search strategies*

Description

Four model search strategies are implemented combining `SEMdag()`, `SEMbap()`, and `resizeGraph()` functions. All strategies estimate a new graph by 1) adjusting (BAP deconfounding) the the data matrix and 2) re-sizing the output DAG.

Usage

```
modelSearch(
  graph,
  data,
  gnet = NULL,
  d = 2,
  search = "basic",
  beta = 0,
  method = "BH",
  alpha = 0.05,
  verbose = FALSE,
  ...
)
```

Arguments

`graph` Input graph as an igraph object.

`data` A matrix or data.frame. Rows correspond to subjects, and columns to graph nodes (variables).

`gnet` Reference directed network used to validate and import nodes and interactions.

`d` Maximum allowed geodesic distance for directed or undirected shortest path search. A distance `d = 0` disables shortest path search (fixed in `search = "basic"`), while `d = 1` (fixed in `search = "direct"`) only search for directed links (i.e., no mediators are allowed). A distance `d` > 1 (defaults to `d = 2` for "outer" and "inner" strategies), will search for shortest paths with at most `d - 1` mediators between nodes sharing a significant estimated interaction. Connectors are imported from the reference interactome, as specified by the argument `gnet`. If the edges of the reference interactome are weighted by P-value, as defined by the `E(gnet)$pv` attribute, the shortest path with the smallest sum of weights will be chosen (e.g., see `weightGraph` for graph weighting options).

`search` Search strategy. Four model search strategies are available:

- "outer". The estimated DAG is re-sized using `resizeGraph` to find new indirect paths (i.e., inferred directed connections that may hide new mediators). New interactions and connectors will be searched and imported from the reference network (argument `gnet`, see above). Both DAG and extended graph complexity can be controlled with `beta` > 0 and `d` > 1 arguments, respectively. The term "outer" means that new model mediator variables are imported from an external resource (i.e., the reference network).
- "inner". This strategy is analogous to the "outer" one, but disables external mediator search. In other words, new indirect paths are generated by adding new interactions of the input model, so that mediators will be nodes already present in the input graph. The reference network is still used to validate new model paths. Also in this case, `beta` > 0 and `d` > 1 are used.
- "direct". The input graph structure is improved through direct (i.e., adjacent) link search, followed by interaction validation and import from the reference network, with no mediators (i.e., `d = 1`).
- "basic" (default). While the previous strategies rely on the input graph and the reference network to integrate knowledge to the final model, the "basic" strategy is data-driven. The input graph is needed to define the topological order. The argument `gnet` is set to NULL (i.e., no reference network is needed) and argument `d = 0`. Model complexity can be still controlled by setting `beta` > 0.

beta Numeric value. Minimum absolute LASSO beta coefficient for a new interaction to be retained in the estimated DAG backbone. Lower beta values correspond to more complex DAGs. By default, beta is set to 0 (i.e., maximum complexity).

method Multiple testing correction method. One of the values available in p.adjust. By default, method is set to "BH" (i.e., Benjamini-Hochberg multiple test correction).

alpha Significance level for false discovery rate (FDR) used for local d-separation tests. This argument is used to control data de-correlation. A higher alpha level includes more hidden covariances, thus considering more sources of confounding. If alpha = 0, data de-correlation is disabled. By default, alpha = 0.05.

verbose If TRUE, it shows intermediate graphs during the execution (not recommended for large graphs).

... Currently ignored.

Details

Search strategies can be ordered by decreasing conservativeness respect to the input graph, as: "direct", "inner", "outer", and "basic". The first three strategies are knowledge-based, since they require an input graph and a reference network, together with data, for knowledge-assisted model improvement. The last one does not require any reference and the output model structure will be data-driven. Output model complexity can be limited using arguments d and beta. While d is fixed to 0 or 1 in "basic" or "direct", respectively; we suggest starting with d = 2 (only one mediator) for the other two strategies. For knowledge-based strategies, we suggest to to start with beta = 0. Then, beta can be relaxed (0 to < 0.1) to improve model fitting, if needed. Since data-driven models can be complex, we suggest to start from beta = 0 when using the "basic" strategy. The beta value can be relaxed until a good model fit is obtained. Argument alpha determines the extent of data adjustment: lower alpha values for FDR correction correspond to a smaller number of significant confounding factors, hence a weaker correction (default alpha = 0.05).

Value

The output model as well as the adjusted dataset are returned as a list of 2 objects:

- "graph", the output model as an igraph object;
- "data", the adjusted dataset.

Author(s)

Mario Grassi <mario.grassi@unipv.it>

Examples

```
# Comparison among different model estimation strategies

# Nonparanormal(npn) transformation
als.npn <- transformData(alsData$exprs)$data
```

```
# Models estimation
m1 <- modelSearch(graph = alsData$graph, data = als.npn, gnet = kegg,
      search = "direct", beta = 0, alpha = 0.05)
m2 <- modelSearch(graph = alsData$graph, data = als.npn, gnet = kegg,
      d = 2, search = "inner", beta = 0, alpha = 0.05)
m3 <- modelSearch(graph = alsData$graph, data = als.npn, gnet = kegg,
      d = 2, search = "outer", beta = 0, alpha = 0.05)
m4 <- modelSearch(graph = alsData$graph, data = als.npn, gnet = NULL,
      search = "basic", beta = 0.1, alpha = 0.05)

# Graphs
#old.par <- par(no.readonly = TRUE)
#par(mfrow=c(2,2), mar= rep(1,4))
gplot(m1$graph, main = "direct graph")
gplot(m2$graph, main = "inner graph")
gplot(m3$graph, main = "outer graph")
gplot(m4$graph, main = "basic graph")
#par(old.par)
```

orientEdges *Assign edge orientation of an undirected graph*

Description

Assign edge orientation of an undirected graph through a given reference directed graph. The vertex (color) and edge (color, width and weight) attributes of the input undirected graph are preserved in the output directed graph.

Usage

```
orientEdges(ug, dg, ...)
```

Arguments

ug	An undirected graph as an igraph object.
dg	A directed reference graph.
...	Currently ignored.

Value

A directed graph as an igraph object.

Examples

```
# Graphs definition
G0 <- as.undirected(sachs$graph)

# Reference graph-based orientation
G1 <- orientEdges(ug = G0, dg = sachs$graph)

# Graphs plotting
old.par <- par(no.readonly = TRUE)
par(mfrow=c(1,2), mar=rep(2,4))
plot(G0, layout=layout.circle, main = "Input undirected graph")
plot(G1, layout=layout.circle, main = "Output directed graph")
par(old.par)
```

`pairwiseMatrix` *Pairwise plotting of multivariate data*

Description

Display a pairwise scatter plot of two datasets for a random selection of variables. If the second dataset is not given, the function displays a histogram with normal curve superposition.

Usage

```
pairwiseMatrix(x, y = NULL, size = nrow(x), r = 4, c = 4, ...)
```

Arguments

`x`	A matrix or data.frame (n x p) of continuous data.
`y`	A matrix or data.frame (n x q) of continuous data.
`size`	number of rows to be sampled (default `size = nrow(x)`).
`r`	number of rows of the plot layout (default `r = 4`).
`c`	number of columns of the plot layout (default `c = 4`).
`...`	Currently ignored.

Value

No return value

Author(s)

Mario Grassi <`mario.grassi@unipv.it`>

Examples

```
adjdata <- SEMbap(sachs$graph, log(sachs$pkc))$data
rawdata <- log(sachs$pkc)
pairwiseMatrix(adjdata, rawdata, size = 1000)
```

parameterEstimates *Parameter Estimates of a fitted SEM*

Description

Wrapper of the lavaan parameterEstimates() function for RICF and CGGM algorithms

Usage

```
parameterEstimates(fit, ...)
```

Arguments

fit	A RICF or constrained GGM fitted model object.
...	Currently ignored.

Value

A data.frame containing the estimated parameters

Author(s)

Mario Grassi <mario.grassi@unipv.it>

Examples

```
ricf1 <- SEMrun(sachs$graph, log(sachs$pkc), sachs$group, algo = "ricf")
parameterEstimates(ricf1$fit)

cggm1 <- SEMrun(sachs$graph, log(sachs$pkc), sachs$group, algo = "cggm")
parameterEstimates(cggm1$fit)
```

pathFinder *Perturbed path search utility*

Description

This function uses SEMace to find significant causal effects between source-sink pairs and SEMpath to fit them and test their edge perturbation.

Usage

```
pathFinder(
  graph,
  data,
  group = NULL,
  ace = NULL,
  path = "directed",
  method = "BH",
  alpha = 0.05,
  ...
)
```

Arguments

graph	Input network as an igraph object.
data	A matrix or data.frame. Rows correspond to subjects, and columns to graph nodes (variables).
group	group A binary vector. This vector must be as long as the number of subjects. Each vector element must be 1 for cases and 0 for control subjects. Group specification enables edge perturbation testing. By default, group = NULL.
ace	A data.frame generated by SEMace. If NULL, SEMace will be automatically run.
path	If path = "directed", all directed paths between the two nodes will be included in the fitted model. If path = "shortest", only shortest paths will be considered.
method	Multiple testing correction method. One of the values available in p.adjust. By default, method = "BH" (i.e., FDR multiple test correction).
alpha	Significance level for ACE selection (by default, alpha = 0.05).
...	Currently ignored.

Value

A list of 3 objects:

- "paths", list of paths as igraph objects;
- "fit", fitting results for each path as a lavaan object;
- "dfp", a data.frame containing SEM global fitting statistics.

Author(s)

Fernando Palluzzi <fernando.palluzzi@gmail.com>

Examples

```
# Find and evaluate significantly perturbed paths

# Nonparanormal(npn) transformation
als.npn <- transformData(alsData$exprs)$data

adjData <- SEMbap(alsData$graph, als.npn)$data

paths <- pathFinder(alsData$graph, adjData,
                    group = alsData$group,
                    ace = NULL)

print(paths$dfp)
head(parameterEstimates(paths$fit[[1]]))
gplot(paths$paths[[1]])
```

predictSink *SEM-based out-of-sample predictions*

Description

Given the values of (observed) x-variables in a structural equation model, this function may be used to predict the values of (observed) y-variables. Response variables (y) represent sink nodes, and predictor variables (x) might consist of either (i) just source nodes or (ii) source and mediators from the fitted graph structure.

Usage

```
predictSink(
  object,
  newdata = NULL,
  K_fold = 5,
  source = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

`object`	An object, as that created by the function `SEMrun()` with the argument `fit` set to `fit = 0` or `fit = 1`.
`newdata`	An optional matrix with rows corresponding to subjects, and columns to graph nodes (variables). If `object$fit` is a model with the group variable (`fit = 1`), the first column of newdata must be the new group binary vector (0=control, 1=case). As a default `newdata = NULL`, meaning that the K-fold cross validation is applied on the `object$data`. Conversely, if the argument `newdata` is specified, this matrix will be used for testing (out-of-sample predictions) and `object$data` will be used for training.
`K_fold`	The number of subsets (folds) into which the data will be partitioned for performing K-fold cross-validation. The model is refit K times, each time leaving out one of the K folds (default, K_fold=5). If the argument `newdata` is specified, the K-fold cross validation will not be done.
`source`	A logical value. If FALSE (default), the predictor variables (x) include source and mediators. If TRUE, x includes only the source nodes.
`verbose`	A logical value. If FALSE (default), the processed graph will not be plotted to screen.
`...`	Currently ignored.

Details

The function uses a SEM-based predictive approach (Rooij et al., 2022) to produce predictions while accounting for the given graph structure. Predictions (for y given x) are based on the (joint y and x) model-implied variance-covariance (Sigma) matrix and mean vector (Mu) of the fitted SEM, and the standard expression for the conditional mean of a multivariate normal distribution. Thus, the structure described in the SEM is taken into consideration, which differs from ordinary least squares (OLS) regression. Note that if the model is saturated (and hence df = 0), or when `source = TRUE`, i.e., the set of predictors will include only the source nodes, the SEM-based predictions are identical or similar to OLS predictions.

Value

A list of 3 objects:

1. "yobs", the matrix of observed continuous values of sink nodes based on out-of-bag samples.
2. "yhat", the matrix of continuous predicted values of sink nodes ased on out-of-bag samples.
3. "PE", vector of the prediction error equal to the Root Mean Squared Error (RMSE) for each out-of-bag sink prediction. The first value of PE is the total RMSE, where we sum over all sink nodes.

Author(s)

Mario Grassi `<mario.grassi@unipv.it>`

References

de Rooij M, Karch JD, Fokkema M, Bakk Z, Pratiwi BC, and Kelderman H (2023). SEM-Based Out-of-Sample Predictions, Structural Equation Modeling: A Multidisciplinary Journal, 30:1, 132-148 <https://doi.org/10.1080/10705511.2022.2061494>

Examples

```
# load ALS data
ig<- alsData$graph
X<- alsData$exprs
X<- transformData(X)$data
group<- alsData$group

#...with train-test (0.8-0.2) samples
set.seed(1)
train<- sample(1:nrow(X), 0.8*nrow(X))

# SEM fitting
#sem0<- SEMrun(ig, X[train,], algo="lavaan", SE="none")
#sem0<- SEMrun(ig, X[train,], algo="ricf", n_rep=0)
sem0<- SEMrun(ig, X[train,], algo="cggm")

# predictors, source+mediator variables
res1<- predictSink(sem0, newdata=X[-train,])
print(res1$PE)

# predictors, source variables
res2<- predictSink(sem0, newdata=X[-train,], source=TRUE)
print(res2$PE)

#...with 5-fold cross-validation samples
set.seed(2)

# SEM fitting
#sem0<- SEMrun(ig, X, algo="lavaan", SE="none")
#sem0<- SEMrun(ig, X, algo="ricf", n_rep=0)
sem0<- SEMrun(ig, X, algo="cggm")

# predictors, source+mediator variables
res3<- predictSink(sem0, K_fold = 5, verbose=TRUE)
print(res3$PE)

# predictors, source variables
res4<- predictSink(sem0, K_fold = 5, source=TRUE, verbose=TRUE)
print(res4$PE)

## Not run:

#...with 10-fold cross-validation samples and 10-iterations

# SEM fitting
#sem1<- SEMrun(ig, X, group, algo="lavaan", SE="none")
```

```
#sem1<- SEMrun(ig, X, group, algo="ricf", n_rep=0)
sem1<- SEMrun(ig, X, group, algo="cggm")

# predictors, source+mediator+group variables
res<- NULL
for (r in 1:10) {
set.seed(r)
cat("rep = ", r, "\n")
resr<- predictSink(sem1, K_fold = 10)
res<- rbind(res, resr$PE)
}
res
apply(res, 2, mean)


## End(Not run)
```

properties *Graph properties summary and graph decomposition*

Description

Produces a summary of network properties and returns graph components (ordered by decreasing size), without self-loops.

Usage

```
properties(graph, data = NULL, ...)
```

Arguments

graph	Input network as an igraph object.
data	An optional data matrix (default data = NULL) whith rows corresponding to subjects, and columns to graph nodes (variables). Nodes will be mapped onto variable names.
...	Currently ignored.

Value

List of graph components, ordered by decreasing size (the first component is the giant one), without self-loops.

Author(s)

Mario Grassi <mario.grassi@unipv.it>

Examples

```
# Extract the "Type II diabetes mellitus" pathway:
g <- kegg.pathways[["Type II diabetes mellitus"]]
summary(g)
properties(g)
```

`resizeGraph` *Interactome-assisted graph re-seizing*

Description

An input directed graph is re-sized, removing edges or adding edges/nodes. This function takes three input graphs: the first is the input causal model (i.e., a directed graph), and the second can be either a directed or undirected graph, providing a set of connections to be checked against a directed reference network (i.e., the third input) and imported to the first graph.

Usage

```
resizeGraph(g = list(), gnet, d = 2, v = TRUE, verbose = FALSE, ...)
```

Arguments

`g` A list of two graphs as igraph objects, g=list(graph1, graph2).

`gnet` External directed network as an igraph object. The reference network should have weighted edges, corresponding to their interaction p-values, as an edge attribute `E(gnet)$pv`. Then, connections in `graph2` will be checked by known connections from the reference network, intercepted by the minimum-weighted shortest path found among the equivalent ones by the Dijkstra algorithm, as implemented in the **igraph** function `all_shortest_paths()`.

`d` An integer value indicating the maximum geodesic distance between two nodes in the interactome to consider the inferred interaction between the same two nodes in `graph2` as validated, otherwise the edges are removed. For instance, if `d = 2`, two interacting nodes must either share a direct interaction or being connected through at most one mediator in the reference interactome (in general, at most `d - 1` mediators are allowed). Typical `d` values include `2` (at most one mediator), or `mean_distance(gnet)` (i.e., the average shortest path length for the reference network). Setting d = 0, is equivalent to `gnet = NULL`.

`v` A logical value. If TRUE (default) new nodes and edges on the validated shortest path in the reference interactome will be added in the re-sized graph.

`verbose` A logical value. If FALSE (default), the processed graphs will not be plotted to screen, saving execution time (for large graphs)

`...` Currently ignored.

Details

Typically, the first graph is an estimated causal graph (DAG), and the second graph is the output of either SEMdag or SEMbap. Alternatively, the first graph is an empthy graph, and the second graph is a external covariance graph. In the former we use the new inferred causal structure stored in the dag.new object. In the latter, we use the new inferred covariance structure stored in the guu object. Both directed (causal) edges inferred by SEMdag() and covariances (i.e., bidirected edges) added by SEMbap(), highlight emergent hidden topological proprieties, absent in the input graph. Estimated directed edges between nodes X and Y are interpreted as either direct links or direct paths mediated by hidden connector nodes. Covariances between any two bow-free nodes X and Y may hide causal relationships, not explicitly represented in the current model. Conversely, directed edges could be redundant or artifact, specific to the observed data and could be deleted. Function resizeGraph() leverage on these concepts to extend/reduce a causal model, importing new connectors or deleting estimated edges, if they are present or absent in a given reference network. The whole process may lead to the discovery of new paths of information flow, and cut edges not corroborate by a validated network. Since added nodes can already be present in the causal graph, network resize may create cross-connections between old and new paths and their possible closure into circuits.

Value

"Ug", the re-sized graph, the graph union of the causal graph graph1 and the re-sized graph graph2

Author(s)

Mario Grassi <mario.grassi@unipv.it>

References

Grassi M, Palluzzi F, Tarantino B (2022). SEMgraph: An R Package for Causal Network Analysis of High-Throughput Data with Structural Equation Models. Bioinformatics, 38 (20), 4829–4830 <https://doi.org/10.1093/bioinformatics/btac567>

Examples

```
# Extract the "Protein processing in endoplasmic reticulum" pathway:

g <- kegg.pathways[["Protein processing in endoplasmic reticulum"]]
G <- properties(g)[[1]]; summary(G)

# Extend a graph using new inferred DAG edges (dag+dag.new):

# Nonparanormal(npn) transformation
als.npn <- transformData(alsData$exprs)$data

dag <- SEMdag(graph = G, data = als.npn, beta = 0.1)
gplot(dag$dag)
ext <- resizeGraph(g=list(dag$dag, dag$dag.new), gnet = kegg, d = 2)
gplot(ext)
```

```
# Create a directed graph from correlation matrix, using
# i) an empty graph as causal graph,
# ii) a covariance graph,
# iii) KEGG as reference:

corr2graph<- function(R, n, alpha=5e-6, ...)
{
Z <- qnorm(alpha/2, lower.tail=FALSE)
thr <- (exp(2*Z/sqrt(n-3))-1)/(exp(2*Z/sqrt(n-3))+1)
A <- ifelse(abs(R) > thr, 1, 0)
diag(A) <- 0
return(graph_from_adjacency_matrix(A, mode=”undirected”))
}

v <- which(colnames(als.npn) %in% V(G)$name)
selectedData <- als.npn[, v]
G0 <- make_empty_graph(n = ncol(selectedData))
V(G0)$name <- colnames(selectedData)
G1 <- corr2graph(R = cor(selectedData), n= nrow(selectedData))
ext <- resizeGraph(g=list(G0, G1), gnet = kegg, d = 2, v = TRUE)

#Graphs
old.par <- par(no.readonly = TRUE)
par(mfrow=c(1,2), mar=rep(1,4))
plot(G1, layout = layout.circle)
plot(ext, layout = layout.circle)
par(old.par)
```

`sachs` *Sachs multiparameter flow cytometry data and consensus model*

Description

Flow cytometry data and causal model from Sachs et al. (2005).

Usage

```
sachs
```

Format

"sachs" is a list of 5 objects:

1. "rawdata", a list of 14 data.frames containing raw flow cytometry data (Sachs et al., 2005);
2. "graph", consensus signaling network;
3. "model", consensus model (lavaan syntax);

4. "pkc", data.frame of 1766 samples and 11 variables, containing cd3cd28 (baseline) and pma (PKC activation) data;
5. "group", a binary group vector, where 0 is for cd3cd28 samples (n = 853) and 1 is for pma samples (n = 913).
6. "details", a data.frame containing dataset information.

Source

doi:10.1126/science.1105809

References

Sachs K, Perez O, Pe'er D, Lauffenburger DA, Nolan GP (2019). Causal Protein-Signaling Networks Derived from Multiparameter Single-Cell Data. Science, 308(5721): 523-529.

Examples

```
# Dataset content
names(sachs$rawdata)
dim(sachs$pkc)
table(sachs$group)
cat(sachs$model)
gplot(sachs$graph)
```

SEMace *Compute the Average Causal Effect (ACE) for a given source-sink pair*

Description

Compute total effects as ACEs of variables X on variables Y in a directed acyclic graph (DAG). The ACE will be estimated as the path coefficient of X (i.e., theta) in the linear equation Y ~ X + Z. The set Z is defined as the adjustment (or conditioning) set of Y over X, applying various adjustement sets. Standard errors (SE), for each ACE, are computed following the `lm` standard procedure or a bootstrap-based procedure (see `boot` for details).

Usage

```
SEMace(
  graph,
  data,
  group = NULL,
  type = "parents",
  effect = "all",
  method = "BH",
  alpha = 0.05,
  boot = NULL,
  ...
)
```

Arguments

`graph` An igraph object.

`data` A matrix or data.frame. Rows correspond to subjects, and columns to graph nodes (variables).

`group` A binary vector. This vector must be as long as the number of subjects. Each vector element must be 1 for cases and 0 for control subjects. If `group = NULL` (default), group influence will not be considered.

`type` character Conditioning set Z. If "parents" (default) the Pearl's back-door set (Pearl, 1998), "minimal" the dagitty minimal set (Perkovic et al, 2018), or "optimal" the O-set with the smallest asymptotic variance (Witte et al, 2020) are computed.

`effect` character X to Y effect. If "all" (default) all effects from X to Y, "source2sink" only effects from source X to sink Y, or "direct" only direct effects from X to Y are computed.

`method` Multiple testing correction method. One of the values available in `p.adjust`. By default, `method = "BH"` (i.e., FDR multiple test correction).

`alpha` Significance level for ACE selection (by default, `alpha = 0.05`).

`boot` The number of bootstrap samplings enabling bootstrap computation of ACE standard errors. If `NULL` (default), bootstrap is disabled.

`...` Currently ignored.

Value

A data.frame of ACE estimates between network sources and sinks.

Author(s)

Mario Grassi <`mario.grassi@unipv.it`>

References

Pearl J (1998). Graphs, Causality, and Structural Equation Models. Sociological Methods & Research, 27(2):226-284. <https://doi.org/10.1177/0049124198027002004>

Perkovic E, Textor J, Kalisch M, Maathuis MH (2018). Complete graphical characterization and construction of adjustment sets in Markov equivalence classes of ancestral graphs. Journal of Machine Learning Research, 18:1-62. <http://jmlr.org/papers/v18/16-319.html>

Witte J, Henckel L, Maathuis MH, Didelez V (2020). On efficient adjustment in causal graphs. Journal of Machine Learning Research, 21:1-45. <http://jmlr.org/papers/v21/20-175.htm>

Examples

```
# ACE without group, O-set, all effects:
ace1 <- SEMace(graph = sachs$graph, data = log(sachs$pkc),
               group = NULL, type = "optimal", effect = "all",
               method = "BH", alpha = 0.05, boot = NULL)
print(ace1)
```

```
# ACE with group perturbation, Pa-set, direct effects:
ace2 <- SEMace(graph = sachs$graph, data = log(sachs$pkc),
              group = sachs$group, type = "parents", effect = "direct",
              method = "none", alpha = 0.05, boot = NULL)
print(ace2)
```

SEMbap *Bow-free covariance search and data de-correlation*

Description

SEMbap() function implements different deconfounding methods to adjust the data matrix by removing latent sources of confounding encoded in them. The selected methods are either based on: (i) Bow-free Acyclic Paths (BAP) search, (ii) LVs proxies as additional source nodes of the data matrix, Y or (iii) spectral transformation of Y.

Usage

```
SEMbap(
  graph,
  data,
  group = NULL,
  dalgo = "cggm",
  method = "BH",
  alpha = 0.05,
  hcount = "auto",
  cmax = Inf,
  limit = 200,
  verbose = FALSE,
  ...
)
```

Arguments

graph An igraph object.

data A matrix whith rows corresponding to subjects, and columns to graph nodes (variables).

group A binary vector. This vector must be as long as the number of subjects. Each vector element must be 1 for cases and 0 for control subjects. If NULL (default), confouding within group will not be considered.

dalgo Deconfounding method. Four algorithms are available:

- "cggm" (default). The algorithm make: (i) exhaustive search of bow-free significant covariances (see details) through Shipley.test function; (ii) estimation of the inverse of the selected covariance matrix (i.e. the precision matrix, W) through `fitConGraph` function; (iii) obtain the de-correlated data matrix, Z by multiplying the data matrix, Y rightward by the square root of the estimated precision matrix, Z=YW^(1/2) as suggested by Grassi, Palluzzi and Tarantino (2022).

- "glpc". The algorithm first makes an exhaustive search of bow-free significant covariances through `Shipley.test` function. Once obtained the adjacency matrix, Graph-Laplacian PCA (gLPCA) algorithm (Jiang et al., 2013) learns a low dimensional representation of the observed data matrix that incorporates bow-free structure. Then, the DAG is extended by including the confounding proxies, i.e. LVs, as additional source nodes defined by last q principal component scores of gLPCA and these LV scores are added to the data matrix, Z=cbind(LV,Y).
- "pc". The procedure add additional source nodes to DAG as in "glpc" algorithm, but confounding proxies are the q principal component scores extracted by Spectral decomposition (SVD) selecting only graph nodes and without graph edge information and bow-free covariance search.
- "trim". Ćevid et al. (2020) suggest multiplying the data matrix, Y leftward by a well selected spectrum transformation matrix, T which modifies the singular values of Y, while keeping its singular vectors intact, Z=TY. Trim transform limits all singular values to be at most some costant (t), where t = median of the singuar values.

`method` Multiple testing correction method. One of the values available in `p.adjust`. By default, `method` is set to "BH" (i.e., Benjamini-Hochberg multiple test correction).

`alpha` Significance level for false discovery rate (FDR) used for d-separation test. This argument is used to control data de-correlation. A higher `alpha` level includes more hidden covariances, thus considering more sources of confounding. If `alpha = 0`, data de-correlation is disabled. By default, `alpha = 0.05`.

`hcount` The number of latent (or hidden) variables. By default `hcount="auto"`, the hidden count is determined with a permutation method (see details). Currently ignored if (dalgo ="cggm" or "trim").

`cmax` Maximum number of parents set, C. This parameter can be used to perform only those tests where the number of conditioning variables does not exceed the given value. High-dimensional conditional independence tests can be very unreliable. By default, cmax = Inf.

`limit` An integer value corresponding to the graph size (vcount) tolerance. Beyond this limit, the precision matrix is estimated by "glasso" algorithm (FHT, 2008) to reduce the computational burden of the exaustive BAP search of the `Shipley.test` procedure. By default, `limit = 200`.

`verbose` A logical value. If FALSE (default), the processed graphs will not be plotted to screen.

`...` Currently ignored.

Details

Missing edges in causal network inference using a directed acyclic graph (DAG) are frequently hidden by unmeasured confounding variables. A Bow-free Acyclic Paths (BAP) search is performed with d-separation tests between all pairs of variables with missing connection in the input DAG, adding a bidirected edge (i.e., bow-free covariance) to the DAG when there is an association between them. The d-separation test evaluates if two variables (Y1, Y2) in a DAG are conditionally independent for a given conditioning set, C represented in a DAG by the union of the parent sets of Y1 and Y2 (Shipley, 2000). A new bow-free covariance is added if there is a significant (Y1, Y2) association at a significance level `alpha`, after multiple testing correction. The selected covariance between pairs of nodes is interpreted as the effect of a latent variable (LV) acting on both nodes; i.e., the LV is an unobserved confounder. BAP-based algorithms adjust (or de-correlate) the observed data matrix by conditioning out the latent triggers responsible for the nuisance edges. For "pc" algorithm the number of hidden proxies, q is determined by a permutation method. It compares the singular values to what they would be if the variables were independent, which is estimated by permuting the columns of the data matrix, Y and selects components if their singular values are larger than those of the permuted data (for a review see Dobriban, 2020). While for "glpc" algorithm, q is determined by the number of clusters by spectral clustering through `cluster_leading_eigen` function. If the input graph is not acyclic, a warning message will be raised, and a cycle-breaking algorithm will be applied (see `graph2dag` for details).

Value

A list of four objects:

- "dag", the directed acyclic graph (DAG) extracted from input graph. If (dalgo = "glpc" or "pc"), the DAG also includes LVs as source nodes.
- "guu", the bow-free covariance graph, BAP = dag + guu. If (dalgo = "pc" or "trim"), guu is equal to NULL
- "adj", the adjacency matrix of selected bow-free covariances; i.e, the missing edges selected after multiple testing correction. If (dalgo = "pc" or "trim"), adj matrix is equal to NULL.
- "data", the adjusted (de-correlated) data matrix or if (dalgo = "glpc", or "pc"), the combined data matrix, where the first columns represent LVs scores and the other columns are the raw data.

Author(s)

Mario Grassi `<mario.grassi@unipv.it>`

References

Grassi M, Palluzzi F, Tarantino B (2022). SEMgraph: An R Package for Causal Network Analysis of High-Throughput Data with Structural Equation Models. Bioinformatics, 38(20), 4829–4830. <https://doi.org/10.1093/bioinformatics/btac567>

Shipley B (2000). A new inferential test for path models based on DAGs. Structural Equation Modeling, 7(2), 206-218. <https://doi.org/10.1207/S15328007SEM0702_4>

Jiang B, Ding C, Bin L, Tang J (2013). Graph-Laplacian PCA: Closed-Form Solution and Robustness. IEEE Conference on Computer Vision and Pattern Recognition, 3492-3498. <https://doi.org/10.1109/CVPR.2013.448>

Ćevid D, Bühlmann P, Meinshausen N (2020). Spectral deconfounding via perturbed sparse linear models. J. Mach. Learn. Res, 21(232), 1-41. <http://jmlr.org/papers/v21/19-545.html>

Dobriban E (2020). Permuatation methods for Factor Analysis and PCA. Ann. Statist. 48(5): 2824-2847 <https://doi.org/10.1214/19-AOS1907>

Friedman J, Hastie T, Tibshirani R (2008). Sparse inverse covariance estimation with the graphical lasso. Biostatistics, 9(3), 432-441. <https://doi.org/10.1093/biostatistics/kxm045>

Examples

```
#Set function param
graph <- sachs$graph
data <- log(sachs$pkc)
group <-sachs$group

# BAP decounfounding with CGGM (default)
bap <- SEMbap(graph, data, verbose = TRUE)

# SVD decounfounding with trim method
svd <- SEMbap(graph, data, dalgo = "trim")

# Model fitting (with node-perturbation)
sem1 <- SEMrun(graph, data, group)
bap1 <- SEMrun(bap$dag, bap$data, group)
svd1 <- SEMrun(svd$dag, svd$data, group)
```

SEMdag *Estimate a DAG from an input (or empty) graph*

Description

Two-step extraction of the optimal DAG from an input (or empty) graph, using in step 1) graph topological order or bottom-up search order, and in step 2) parent recovery with the LASSO-based algorithm (FHT, 2010), implemented in `glmnet`.

Usage

```
SEMdag(
  graph,
  data,
  LO = "TO",
  beta = 0,
  eta = NULL,
  lambdas = NA,
  penalty = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

`graph` An igraph object or a graph with no edges (make_empty_graph(n=0)).

`data` A matrix whith n rows corresponding to subjects, and p columns to graph nodes (variables).

`LO` character for linear order method. If LO="TO" or LO="TL" the topological order (resp. level) of the input graph is enabled, while LO="BU" the data-driven bottom-up search of vertex (resp. layer) order is performed using the vertices of the empty graph. By default `LO = "TO"`.

`beta` Numeric value. Minimum absolute LASSO beta coefficient for a new direct link to be retained in the final model. By default, `beta = 0`.

`eta` Numeric value. Minimum fixed eta threshold for bottom-up search of vertex (eta = 0) or layer (eta > 0) ordering. Use eta = NULL, for estimation of eta adaptively with half of the sample data. By default, `eta = 0`.

`lambdas` A vector of regularization LASSO lambda values. If lambdas is NULL, the `glmnet` default using cross-validation lambdas is enabled. If lambdas is NA (default), the tuning-free scheme is enabled by fixing lambdas = sqrt(log(p)/n), as suggested by Janková and van de Geer (2015) and many others. This will both reduce computational time and provide the same result at each run.

`penalty` A logical value. Separate penalty factors can be applied to each coefficient. This is a number that multiplies lambda to allow differential shrinkage. Can be 0 for some variables, which implies no shrinkage, and that variable is always included in the model. If TRUE (default) weights are based on the graph edges: 0 (i.e., edge present) and 1 (i.e., missing edge) ensures that the input edges will be retained in the final model. If FALSE the `glmnet` default is enabled (all weights equal to 1). Note: the penalty factors are internally rescaled to sum p (the number of variables).

`verbose` A logical value. If FALSE (default), the processed graphs will not be plotted to screen.

`...` Currently ignored.

Details

The extracted DAG is estimated using the two-step order search approach. First a vertex (node) or level (layer) order of p nodes is determined, and from this sort, the DAG can be learned using in step 2) penalized (L1) regressions (Shojaie and Michailidis, 2010). The estimate linear order are obtained from *a priori* graph topological vertex (TO) or level (TL) ordering, or with a data-driven Bottom-up (BU) approach, assuming a SEM whose error terms have equal variances (Peters and Bühlmann, 2014). The BU algorithm first estimates the last element (the terminal vertex) using the diagonal entries of the inverse covariance matrix with: t = argmin(diag(Omega)), or the terminal layer (> 1 vertices) with d = diag(Omega)- t < eta. And then, it determines its parents with L1 regression. After eliminating the last element (or layer) of the ordering, the algorithm applies the same procedure until a DAG is completely estimated. In high-dimensional data (n < p), the inverse covariance matrix is computed by glasso-based algorithm (FHT, 2008), implemented in `glasso`. If the input graph is not acyclic, in TO or TL, a warning message will be raised, and a cycle-breaking algorithm will be applied (see `graph2dag` for details). Output DAG will be colored: vertices in cyan, if they are source nodes, and in orange, if they are sink nodes, and edges in gray, if they were present in the input graph, and in green, if they are new edges generated by LASSO screening.

Value

A list of 3 igraph objects plus the vertex ordering:

1. "dag", the estimated DAG;
2. "dag.new", new estimated connections;
3. "dag.old", connections preserved from the input graph;
4. "LO", the estimated vertex ordering.

Author(s)

Mario Grassi <mario.grassi@unipv.it>

References

Friedman J, Hastie T, Tibshirani R (2008). Sparse inverse covariance estimation with the graphical lasso. Biostatistics, 9(3), 432-441. <https://doi.org/10.1093/biostatistics/kxm045>

Friedman J, Hastie T, Tibshirani R (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. Journal of Statistical Software, Vol. 33(1), 1-22. <https://doi.org/10.18637/jss.v033.i01>

Shojaie A, Michailidis G (2010). Penalized likelihood methods for estimation of sparse high-dimensional directed acyclic graphs. Biometrika, 97(3): 519-538. <https://doi.org/10.1093/biomet/asq038>

Jankova J, van de Geer S (2015). Confidence intervals for high-dimensional inverse covariance estimation. Electronic Journal of Statistics, 9(1): 1205-1229. <https://doi.org/10.1214/15-EJS1031>

Peters J, Bühlmann P (2014). Identifiability of Gaussian structural equation models with equal error variances. Biometrika, 101(1):219–228. <https://doi.org/10.1093/biomet/ast043>

See Also

modelSearch

Examples

```
#Set function param
ig <- sachs$graph
X <- log(sachs$pkc)
group <- sachs$group

# DAG estimation (default values)
dag0 <- SEMdag(ig, X)
sem0 <- SEMrun(ig, X, group)

# Graphs
old.par <- par(no.readonly = TRUE)
par(mfrow=c(2,2), mar=rep(1,4))
plot(sachs$graph, layout=layout.circle, main="input graph")
plot(dag0$dag, layout=layout.circle, main = "Output DAG")
plot(dag0$dag.old, layout=layout.circle, main = "Inferred old edges")
plot(dag0$dag.new, layout=layout.circle, main = "Inferred new edges")
par(old.par)
```

```
# Four DAG estimation
dag1 <- SEMdag(ig, X, LO="TO")
dag2 <- SEMdag(ig, X, LO="TL")
dag3 <- SEMdag(ig, X, LO="BU", eta=0)
dag4 <- SEMdag(ig, X, LO="BU", eta=NULL)

unlist(dag1$LO)
dag2$LO
unlist(dag3$LO)
dag4$LO

# Graphs
old.par <- par(no.readonly = TRUE)
par(mfrow=c(2,2), mar=rep(2,4))
gplot(dag1$dag, main="TO")
gplot(dag2$dag, main="TL")
gplot(dag3$dag, main="BU")
gplot(dag4$dag, main="TLBU")
par(old.par)
```

SEMdci *SEM-based differential network analysis*

Description

Creates a sub-network with perturbed edges obtained from the output of SEMace, comparable to the procedure in Jablonski et al (2022), or of SEMrun with two-group and CGGM solver, comparable to the algorithm 2 in Belyaeva et al (2021). To increase the efficiency of computations for large graphs, users can select to break the network structure into clusters, and select the topological clustering method (see clusterGraph). The function SEMrun is applied iteratively on each cluster (with size min > 10 and max < 500) to obtain the graph with the full list of perturbed edges.

Usage

```
SEMdci(graph, data, group, type = "ace", method = "BH", alpha = 0.05, ...)
```

Arguments

graph	Input network as an igraph object.
data	A matrix or data.frame. Rows correspond to subjects, and columns to graph nodes (variables).
group	A binary vector. This vector must be as long as the number of subjects. Each vector element must be 1 for cases and 0 for control subjects.
type	Average Causal Effect (ACE) with two-group, "parents" (back-door) adjustement set, and "direct" effects (type = "ace", default), or CGGM solver with two-group using a clustering method. If type = "tahc", network modules are

generated using the tree agglomerative hierarchical clustering method, or non-tree clustering methods from igraph package, i.e., `type = "wtc"` (walktrap community structure with short random walks), `type ="ebc"` (edge betweeness clustering), `type = "fgc"` (fast greedy method), `type = "lbc"` (label propagation method), `type = "lec"` (leading eigenvector method), `type = "loc"` (multi-level optimization), `type = "opc"` (optimal community structure), `type = "sgc"` (spinglass statistical mechanics), `type = "none"` (no breaking network structure into clusters).

`method` Multiple testing correction method. One of the values available in `p.adjust`. By default, method is set to "BH" (i.e., FDR multiple test correction).

`alpha` Significance level (default = 0.05) for edge set selection.

`...` Currently ignored.

Value

An igraph object.

Author(s)

Mario Grassi <`mario.grassi@unipv.it`>

References

Belyaeva A, Squires C, Uhler C (2021). DCI: learning causal differences between gene regulatory networks. Bioinformatics, 37(18): 3067–3069. <https://doi: 10.1093/bioinformatics/btab167>

Jablonski K, Pirkl M, Ćevid D, Bühlmann P, Beerenwinkel N (2022). Identifying cancer pathway dysregulations using differential causal effects. Bioinformatics, 38(6):1550–1559. <https://doi.org/10.1093/bioinformatics/bt

Examples

```
## Not run:

#load SEMdata package for ALS data with 17K genes:
#devtools::install_github("fernandoPalluzzi/SEMdata")
#library(SEMdata)

# Nonparanormal(npn) transformation
library(huge)
data.npn<- huge.npn(alsData$exprs)
dim(data.npn) #160 17695

# Extract KEGG interactome (max component)
KEGG<- properties(kegg)[[1]]
summary(KEGG)

# KEGG modules with ALS perturbed edges using fast gready clustering
gD<- SEMdci(KEGG, data.npn, alsData$group, type="fgc")
summary(gD)
gcD<- properties(gD)
```

```
old.par <- par(no.readonly = TRUE)
par(mfrow=c(2,2), mar=rep(2,4))
gplot(gcD[[1]], l="fdp", main="max component")
gplot(gcD[[2]], l="fdp", main="2nd component")
gplot(gcD[[3]], l="fdp", main="3rd component")
gplot(gcD[[4]], l="fdp", main="4th component")
par(old.par)


## End(Not run)
```

SEMgsa *SEM-based gene set analysis*

Description

Gene Set Analysis (GSA) via self-contained test for group effect on signaling (directed) pathways based on SEM. The core of the methodology is implemented in the RICF algorithm of `SEMrun()`, recovering from RICF output node-specific group effect p-values, and Brown's combined permutation p-values of node activation and inhibition.

Usage

```
SEMgsa(g = list(), data, group, method = "BH", alpha = 0.05, n_rep = 1000, ...)
```

Arguments

`g`	A list of pathways to be tested.
`data`	A matrix or data.frame. Rows correspond to subjects, and columns to graph nodes (variables).
`group`	A binary vector. This vector must be as long as the number of subjects. Each vector element must be 1 for cases and 0 for control subjects.
`method`	Multiple testing correction method. One of the values available in `p.adjust`. By default, method is set to "BH" (i.e., Benjamini-Hochberg correction).
`alpha`	Gene set test significance level (default = 0.05).
`n_rep`	Number of randomization replicates (default = 1000).
`...`	Currently ignored.

Details

For gaining more biological insights into the functional roles of pre-defined subsets of genes, node perturbation obtained from RICF fitting has been combined with up- or down-regulation of genes from KEGG to obtain overall pathway perturbation as follows:

- The node perturbation is defined as activated when the minimum among the p-values is positive; if negative, the status is inhibited.

- Up- or down- regulation of genes (derived from KEGG database) has been obtained from the weighted adjacency matrix of each pathway as column sum of weights over each source node. If the overall sum of node weights is below 1, the pathway is flagged as down-regulated otherwise as up-regulated.
- The combination between these two quantities allows to define the direction (up or down) of gene perturbation. Up- or down regulated gene status, associated with node inhibition, indicates a decrease in activation (or increase in inhibition) in cases with respect to control group. Conversely, up- or down regulated gene status, associated with node activation, indicates an increase in activation (or decrease in inhibition) in cases with respect to control group.

Value

A list of 2 objects:

1. "gsa", A data.frame reporting the following information for each pathway in the input list:
   - "No.nodes", pathway size (number of nodes);
   - "No.DEGs", number of differential espression genes (DEGs) within the pathway, after multiple test correction with one of the methods available in `p.adjust`;
   - "pert", pathway perturbation status (see details);
   - "pNA", Brown's combined P-value of pathway node activation;
   - "pNI", Brown's combined P-value of pathway node inhibition;
   - "PVAL", Bonferroni combined P-value of pNA, and pNI; i.e., 2* min(pNA, PNI);
   - "ADJP", Adjusted Bonferroni P-value of pathway perturbation; i.e., min(No.pathways * PVAL; 1).
2. "DEG", a list with DEGs names per pathways.

Author(s)

Mario Grassi `<mario.grassi@unipv.it>`

References

Grassi, M., Tarantino, B. SEMgsa: topology-based pathway enrichment analysis with structural equation models. BMC Bioinformatics 23, 344 (2022). https://doi.org/10.1186/s12859-022-04884-8

Examples

```
## Not run:

# Nonparanormal(npn) transformation
als.npn <- transformData(alsData$exprs)$data

# Selection of FTD-ALS pathways from kegg.pathways.Rdata

paths.name <- c("MAPK signaling pathway",
                "Protein processing in endoplasmic reticulum",
                "Endocytosis",
                "Wnt signaling pathway",
                "Neurotrophin signaling pathway",
                "Amyotrophic lateral sclerosis")

j <- which(names(kegg.pathways) %in% paths.name)

GSA <- SEMgsa(kegg.pathways[j], als.npn, alsData$group,
              method = "bonferroni", alpha = 0.05,
              n_rep = 1000)
GSA$gsa
GSA$DEG


## End(Not run)
```

SEMpath *Search for directed or shortest paths between pairs of source-sink nodes*

Description

Find and fit all directed or shortest paths between two source-sink nodes of a graph.

Usage

```
SEMpath(graph, data, group, from, to, path, verbose = FALSE, ...)
```

Arguments

`graph`	An igraph object.
`data`	A matrix or data.frame. Rows correspond to subjects, and columns to graph nodes (variables).
`group`	A binary vector. This vector must be as long as the number of subjects. Each vector element must be 1 for cases and 0 for control subjects. If `NULL` (default), group influence will not be considered.
`from`	Starting node name (i.e., source node).
`to`	Ending node name (i.e., sink node).
`path`	If `path = "directed"`, all directed paths between the two nodes will be included in the fitted model. If `path = "shortest"`, only shortest paths will be returned.
`verbose`	Show the directed (or shortest) path between the given source-sink pair inside the input graph.
`...`	Currently ignored.

Value

A list of four objects: a fitted model object of class `lavaan` ("fit"), aggregated and node-specific group effect estimates and P-values ("gest"), the extracted subnetwork as an igraph object ("graph"), and the input graph with a color attribute mapping the chosen path ("map").

Author(s)

Mario Grassi <mario.grassi@unipv.it>

Examples

```
# Directed path fitting
path <- SEMpath(graph = sachs$graph, data = log(sachs$pkc),
                group = sachs$group,
                from = "PIP3",
                to = "Erk",
                path = "directed")

# Summaries
summary(path$fit)
print(path$gest)

# Graphs
gplot(path$map, main="path from PiP2 to Erk")
plot(path$map, layout=layout.circle, main="path from PiP2 to Erk")
```

SEMrun *Fit a graph as a Structural Equation Model (SEM)*

Description

SEMrun() converts a (directed, undirected, or mixed) graph to a SEM and fits it. If a binary group variable (i.e., case/control) is present, node-level or edge-level perturbation is evaluated. This function can handle loop-containing models, although multiple links between the same two nodes (including self-loops and mutual interactions) and bows (i.e., a directed and a bidirected link between two nodes) are not allowed.

Usage

```
SEMrun(
  graph,
  data,
  group = NULL,
  fit = 0,
  algo = "lavaan",
  start = NULL,
  SE = "standard",
  n_rep = 1000,
  limit = 100,
  ...
)
```

Arguments

`graph` An igraph object.

`data` A matrix whith rows corresponding to subjects, and columns to graph nodes (variables).

`group` A binary vector. This vector must be as long as the number of subjects. Each vector element must be 1 for cases and 0 for control subjects. If `NULL` (default), group influence will not be considered.

`fit` A numeric value indicating the SEM fitting mode. If `fit = 0` (default), no group effect is considered. If `fit = 1`, a "common" model is used to evaluate group effects on graph nodes. If `fit = 2`, a two-group model is used to evaluate group effects on graph edges.

`algo` MLE method used for SEM fitting. If `algo = "lavaan"` (default), the SEM will be fitted using the NLMINB solver from `lavaan` R package, with standard errors derived from the expected Fisher information matrix. If `algo = "ricf"`, the model is fitted via residual iterative conditional fitting (RICF; Drton et al. 2009), with standard error derived from randomization or bootstrap procedures. If `algo = "cggm"`, model fitting is based on constrained Gaussian Graphical Modeling (CGGM), with DAG nodewise Lasso procedure and de-biasing asymptotic inference (Jankova & Van De Geer, 2019).

`start` Starting value of SEM parameters for `algo = "lavaan"`. If start is `NULL` (default), the algorithm will determine the starting values. If start is a numeric value, it will be used as a scaling factor for the edge weights in the graph object (graph attribute `E(graph)$weight`). For instance, a scaling factor is useful when weights have fixed values (e.g., 1 for activated, -1 for repressed, and 0 for unchanged interaction). Fixed values may compromise model fitting, and scaling them is a safe option to avoid this problem. As a rule of thumb, to our experience, `start = 0.1` generally performs well with (-1, 0, 1) weights.

`SE` If "standard" (default), with `algo = "lavaan"`, conventional standard errors are computed based on inverting the observed information matrix. If "none", no standard errors are computed.

`n_rep` Number of randomization replicates (default = 1000), for permutation flip or boostrap samples, if `algo = "ricf"`.

`limit` An integer value corresponding to the network size (i.e., number of nodes). Beyond this limit, the execution under `algo = "lavaan"` will run with `SE = "none"`, if `fit = 0`, or will be ridirected to `algo = "ricf"`, if `fit = 1`, or to `algo = "cggm"`, if `fit = 2`. This redirection is necessary to reduce the computational demand of standard error estimation by lavaan. Increasing this number will enforce lavaan execution when `algo = "lavaan"`.

`...` Currently ignored.

Details

SEMrun maps data onto the input graph and converts it into a SEM. Directed connections (X -> Y) are interpreted as direct causal effects, while undirected, mutual, and bidirected connections are converted into model covariances. SEMrun output contains different sets of parameter estimates.

Beta coefficients (i.e., direct effects) are estimated from directed interactions and residual covariances (psi coefficients) from bidirected, undirected, or mutual interactions. If a group variable is given, exogenous group effects on nodes (gamma coefficients) or edges (delta coefficients) will be estimated. By default, maximum likelihood parameter estimates and P-values for parameter sets are computed by conventional z-test (= estimate/SE), and fits it through the `lavaan` function, via Maximum Likelihood Estimation (estimator = "ML", default estimator in `lavOptions`). In case of high dimensionality (n.variables » n.subjects), the covariance matrix could not be semi-definite positive and thus parameter estimates could not be done. If this happens, covariance matrix regularization is enabled using the James-Stein-type shrinkage estimator implemented in the function `pcor.shrink` of corpcor R package. Argument `fit` determines how group influence is evaluated in the model, as absent (`fit = 0`), node perturbation (`fit = 1`), or edge perturbation (`fit = 2`). When `fit = 1`, the group is modeled as an exogenous variable, influencing all the other graph nodes. When `fit = 2`, SEMrun estimates the differences of the beta and/or psi coefficients (network edges) between groups. This is equivalent to fit a separate model for cases and controls, as opposed to one common model perturbed by the exogenous group effect. Once fitted, the two models are then compared to assess significant edge (i.e., direct effect) differences (d = beta1 - beta0). P-values for parameter sets are computed by z-test (= d/SE), through `lavaan`. As an alternative to standard P-value calculation, SEMrun may use either RICF (randomization or bootstrap P-values) or GGM (de-biased asymptotically normal P-values) methods. These algorithms are much faster than `lavaan` in case of large input graphs.

Value

A list of 5 objects:

1. "fit", SEM fitted lavaan, ricf, or cggm object, depending on the MLE method specified by the `algo` argument;
2. "gest" or "dest", a data.frame of node-specific ("gest") or edge-specific ("dest") group effect estimates and P-values;
3. "model", SEM model as a string if `algo = "lavaan"`, and `NULL` otherwise;
4. "graph", the induced subgraph of the input network mapped on data variables. Graph edges (i.e., direct effects) with P-value < 0.05 will be highlighted in red (beta > 0) or blue (beta < 0). If a group vector is given, nodes with significant group effect (P-value < 0.05) will be red-shaded (beta > 0) or lightblue-shaded (beta < 0);
5. "data", input data subset mapping graph nodes, plus group at the first column (if no group is specified, this column will take NA values).

Author(s)

Mario Grassi `<mario.grassi@unipv.it>`

References

Pearl J (1998). Graphs, Causality, and Structural Equation Models. Sociological Methods & Research., 27(2):226-284. <https://doi.org/10.1177/0049124198027002004>

Yves Rosseel (2012). lavaan: An R Package for Structural Equation Modeling. Journal of Statistical Software, 48(2): 1-36. <https://www.jstatsoft.org/v48/i02/>

Pepe D, Grassi M (2014). Investigating perturbed pathway modules from gene expression data via Structural Equation Models. BMC Bioinformatics, 15: 132. <https://doi.org/10.1186/1471-2105-15-132>

Drton M, Eichler M, Richardson TS (2009). Computing Maximum Likelihood Estimated in Recursive Linear Models with Correlated Errors. Journal of Machine Learning Research, 10(Oct): 2329-2348. <https://www.jmlr.org/papers/volume10/drton09a/drton09a.pdf>

Jankova, J., & Van De Geer, S (2019). Inference in high-dimensional graphical models. In Handbook of Graphical Models (2019). Chapter 14 (sec. 14.2): 325-349. Chapman & Hall/CRC. ISBN: 9780429463976

Hastie T, Tibshirani R, Friedman J. (2009). The Elements of Statistical Learning (2nd ed.). Springer Verlag. ISBN: 978-0-387-84858-7

Grassi M, Palluzzi F, Tarantino B (2022). SEMgraph: An R Package for Causal Network Analysis of High-Throughput Data with Structural Equation Models. Bioinformatics, 38 (20), 4829–4830 <https://doi.org/10.1093/bioinformatics/btac567>

See Also

See `fitAncestralGraph` and `fitConGraph` for RICF algorithm and constrained GGM algorithm details, respectively.

Examples

```
#### Model fitting (no group effect)

sem0 <- SEMrun(graph = sachs$graph, data = log(sachs$pkc))
summary(sem0$fit)
head(parameterEstimates(sem0$fit))

# Graphs
gplot(sem0$graph, main = "significant edge weights")
plot(sem0$graph, layout = layout.circle, main = "significant edge weights")


#### Model fitting (common model, group effect on nodes)

sem1 <- SEMrun(graph = sachs$graph, data = log(sachs$pkc),
               group = sachs$group)

# Fitting summaries
summary(sem1$fit)
print(sem1$gest)
head(parameterEstimates(sem1$fit))

# Graphs
gplot(sem1$graph, main = "Between group node differences")
plot(sem1$graph, layout = layout.circle, main = "Between group node differences")


#### Two-group model fitting (group effect on edges)
```

```
sem2 <- SEMrun(graph = sachs$graph, data = log(sachs$pkc),
               group = sachs$group,
               fit = 2)

# Summaries
summary(sem2$fit)
print(sem2$dest)
head(parameterEstimates(sem2$fit))

# Graphs
gplot(sem2$graph, main = "Between group edge differences")
plot(sem2$graph, layout = layout.circle, main = "Between group edge differences")



# Fitting and visualization of a large pathway:

g <- kegg.pathways[["Neurotrophin signaling pathway"]]
G <- properties(g)[[1]]
summary(G)

# Nonparanormal(npn) transformation
als.npn <- transformData(alsData$exprs)$data

g1 <- SEMrun(G, als.npn, alsData$group, algo = "cggm")$graph
g2 <- SEMrun(g1, als.npn, alsData$group, fit = 2, algo = "cggm")$graph

# extract the subgraph with node and edge differences
g2 <- g2 - E(g2)[-which(E(g2)$color != "gray50")]
g <- properties(g2)[[1]]

# plot graph
E(g)$color<- E(g2)$color[E(g2) %in% E(g)]
gplot(g, l="fdp", psize=40, main="node and edge group differences")
```

SEMtree *Tree-based structure learning methods*

Description

Four tree-based structure learning methods are implemented with graph and data-driven algorithms.

Usage

```
SEMtree(
  graph,
  data,
  seed,
  type = "ST",
  eweight = NULL,
  alpha = 0.05,
  verbose = FALSE,
  ...
)
```

Arguments

`graph` An igraph object.

`data` A matrix or data.frame. Rows correspond to subjects, and columns to graph nodes (variables).

`seed` A vector of seed nodes.

`type` Tree-based structure learning method. Four algorithms are available:

- "ST"(default). Steiner Tree (ST) identification via fast Kou's algorithm (Kou et al, 1981) connecting a set of seed nodes (called Terminal vertices) with connector nodes (called Steiner vertices) from input graph as defined in `graph` with minimal total distance on its edges. By default the edge weights are based on the pairwise correlation, 1-abs(cor(j,k)). If input graph has E(graph)$weight=1, and `eweight = "custom"`, ST seeks a minimum subtree (i.e., the subtree with minimal number of edges).
- "CAT". Causal additive trees (CAT) algorithm as in Jakobsen et al. (2022). The argument `graph` is set to NULL (i.e., no input graph is needed). In the first step, a (univariate) generalized additive model (GAM) is employed to estimate the residual variances, var(X(j) - [X(j)|X(k)]) for all j != k, then use these to construct edge weights as inputs to the Chu–Liu–Edmonds' algorithm (Chow and Liu, 1968) to recover the arborescence. Argument `seed` must be specified to analyse a subset of nodes (variables) of interest.
- "CPDAG". CLE algorithm for Skeleton Recovery and CPDAG estimation as in Lou et al. (2021). Together with "CAT" algorithm, "CPDAG" is data-driven and the argument `graph` is set to NULL. The key idea is to first recover the skeleton of the polytree by applying the CLE algorithm to the pairwise sample correlations of the data matrix. After the skeleton is recovered, the set of all v-structures can be correctly identified via a simple thresholding approach to pairwise sample correlations. CPDAG can be found applying iteratively only Rule 1 of Meek (1995). Argument `seed` must be specified to analyse a subset of nodes (variables) of interest.
- "MST". Minimum Spanning Tree (MST) identification via Prim's algorithm (Prim, 1957). The latter finds the subset of edges that includes every vertex of the graph (as defined in `graph`) such that the sum of the weights of the edges can be minimized. The argument `seed` is set to NULL (i.e., no seed nodes are needed).

`eweight` Edge weight type for igraph object can be externally derived using `weightGraph` or from user-defined distances. This option determines the weight-to-distance transform. If set to:

- "NULL" (default), edge weights will be internally computed equal to 1 - abs(pairwise Pearson's correlation).
- "kegg", repressing(-1), neutral(0) and activating(+1) kegg interactions will be multiplied by "zsign" attributes, and positive (i.e., concordant) values will be set to 1 (minimum distance), while negative (i.e., discordant) values will be set to 2.
- "zsign", all significant interactions (abs(zsign) > 0) will be set to 1 (minimum distance), while non-significant (zsign=0) ones will be set to 2.
- "pvalue", edge p-value atributes will be transformed to the inverse of negative base-10 logarithm, 1/(-log(E(graph)$pv)).
- "custom", the algorithm will use the distance measure specified by the user as "weight" edge attribute in the input graph.

`alpha` Threshold for rejecting a pair of node being independent in "CPDAG" algorithm. The latter implements a natural v-structure identification procedure by thresholding the pairwise sample correlations over all adjacent pairs of edges with some appropriate threshold. By default, `alpha = 0.05`.

`verbose` If TRUE, it shows the output tree (not recommended for large graphs).

`...` Currently ignored.

Details

A tree ia an acyclic graph with p vertices and p-1 edges. The graph method refers to the Steiner Tree (ST), a tree from an undirected graph that connect "seed" with additional nodes in the "most compact" way possible. The data-driven methods propose fast and scalable procedures based on Chu-Liu–Edmonds' algorithm (CLE) to recover a tree from a full graph. The first method, called Causal Additive Trees (CAT) uses pairwise mutual weights as input for CLE algorithm to recover a directed tree (an "arborescence"). The second one applies CLE algorithm for skeleton recovery and extends the skeleton to a tree (a "polytree") represented by a Completed Partially Directed Acyclic Graph (CPDAG). Finally, the Minimum Spanning Tree (MST) connecting an undirected graph with minimal edge weights can be identified. To note, if the input graph is a directed graph, ST and MST undirected trees are converted in directed trees using the `orientEdges` function.

Value

An `igraph` object. If `type = "ST"`, seed nodes are colored in "aquamarine" and connectors in "white". If `type = "ST"` and `type = "MST"`, edges are colored in "green" if not present in the input, graph. If `type = "CPDAG"`, bidirected edges are colored in "black" (if the algorithm is not able to establish the direction of the relationship between x and y).

Author(s)

Mario Grassi `<mario.grassi@unipv.it>`

References

Grassi M, Tarantino B (2023). SEMtree: tree-based structure learning methods with structural equation models. Bioinformatics, 39 (6), 4829–4830 <https://doi.org/10.1093/bioinformatics/btad377>

Kou, L., Markowsky, G., Berman, L. (1981). A fast algorithm for Steiner trees. Acta Informatica 15, 141–145. <https://doi.org/10.1007/BF00288961>

Prim, R.C. (1957). Shortest connection networks and some generalizations Bell System Technical Journal, 37 1389–1401.

Chow, C.K. and Liu, C. (1968). Approximating discrete probability distributions with dependence trees. IEEE Transactions on Information Theory, 14(3):462–467.

Meek, C. (1995). Causal inference and causal explanation with background knowledge. In Proceedings of the Eleventh conference on Uncertainty in artificial intelligence, 403–410.

Jakobsen, M, Shah, R., Bühlmann, P., Peters, J. (2022). Structure Learning for Directed Trees. arXiv: <https://doi.org/10.48550/arxiv.2108.08871>.

Lou, X., Hu, Y., Li, X. (2022). Linear Polytree Structural Equation Models: Structural Learning and Inverse Correlation Estimation. arXiv: <https://doi.org/10.48550/arxiv.2107.10955>

Examples

```
# Nonparanormal(npn) transformation
als.npn <- transformData(alsData$exprs)$data

# graph-based trees
graph <- alsData$graph
seed <- V(graph)$name[sample(1:vcount(graph), 10)]
tree1 <- SEMtree(graph, als.npn, seed=seed, type="ST", verbose=TRUE)
tree2 <- SEMtree(graph, als.npn, seed=NULL, type="MST", verbose=TRUE)

# data-driven trees
V <- colnames(als.npn)[colnames(als.npn) %in% V(graph)$name]
tree3 <- SEMtree(NULL, als.npn, seed=V, type="CAT", verbose=TRUE)
tree4 <- SEMtree(NULL, als.npn, seed=V, type="CPDAG", alpha=0.05, verbose=TRUE)
```

`Shipley.test` *Missing edge testing implied by a DAG with Shipley’s basis-set*

Description

Compute all the P-values of the d-separation tests implied by the missing edges of a given acyclic graph (DAG). The conditioning set Z is represented, in a DAG, by the union of the parent sets of X and Y (Shipley, 2000). The results of every test, in a DAG, is then combined using the Fisher’s statistic in an overall test of the fitted model C = -2*sum(log(P-value(k))), where C is distributed as a chi-squared variate with df = 2k, as suggested by Shipley (2000).

Usage

```
Shipley.test(
  graph,
  data,
  MCX2 = FALSE,
  cmax = Inf,
  limit = 100,
  verbose = TRUE,
  ...
)
```

Arguments

`graph` A directed graph as an igraph object.

`data` A data matrix with subjects as rows and variables as columns.

`MCX2` If TRUE, a Monte Carlo P-value of the combined C test is enabled using the R code of Shipley extracted from <https://github.com/BillShipley/CauseAndCorrelation>.

`cmax` Maximum number of parents set, C. This parameter can be used to perform only those tests where the number of conditioning variables does not exceed the given value. High-dimensional conditional independence tests can be very unreliable. By default, cmax = Inf.

`limit` An integer value corresponding to the graph size (vcount) tolerance. Beyond this limit, multicore computation is enabled to reduce the computational burden. By default, `limit = 100`.

`verbose` If TRUE, Shipley's test results will be showed to screen (default = TRUE).

`...` Currently ignored.

Value

A list of three objects: (i) "dag": the DAG used to perform the Shipley test (ii) "dsep": the data.frame of all d-separation tests over missing edges in the DAG and (iii) "ctest": the overall Shipley's' P-value.

Author(s)

Mario Grassi <`mario.grassi@unipv.it`>

References

Shipley B (2000). A new inferential test for path models based on DAGs. Structural Equation Modeling, 7(2): 206-218. <https://doi.org/10.1207/S15328007SEM0702_4>

Examples

```
#\donttest{

# Nonparanormal(npn) transformation
als.npn <- transformData(alsData$exprs)$data

sem <- SEMrun(alsData$graph, als.npn)
C_test <- Shipley.test(sem$graph, als.npn, MCX2 = FALSE)
#MC_test <- Shipley.test(sem$graph, als.npn, MCX2 = TRUE)

#}
```

summary.GGM *GGM model summary*

Description

Generate a summary for a constrained Gaussian Graphical Model (GGM) similar to lavaan-formated summary

Usage

```
## S3 method for class 'GGM'
summary(object, ...)
```

Arguments

`object`	A constrained GGM fitted model object.
`...`	Currently ignored.

Value

Shown the lavaan-formatted summary to console

Author(s)

Mario Grassi <mario.grassi@unipv.it>

See Also

SEMrun.

Examples

```
sem0 <- SEMrun(sachs$graph, log(sachs$pkc), algo = "cggm")
summary(sem0$fit)
```

summary.RICF *RICF model summary*

Description

Generate a summary for a RICF solver similar to lavaan-formatted summary

Usage

```
## S3 method for class 'RICF'
summary(object, ...)
```

Arguments

`object`	A RICF fitted model object.
`...`	Currently ignored.

Value

Shown the lavaan-formatted summary to console

Author(s)

Mario Grassi <`mario.grassi@unipv.it`>

See Also

`SEMrun`.

Examples

```
sem1 <- SEMrun(sachs$graph, log(sachs$pkc), sachs$group, algo = "ricf")
summary(sem1$fit)
```

transformData *Transform data methods*

Description

Implements various data trasformation methods with optimal scaling for ordinal or nominal data, and to help relax the assumption of normality (gaussianity) for continuous data.

Usage

```
transformData(x, method = "npn", ...)
```

Arguments

`x` A matrix or data.frame (n x p). Rows correspond to subjects, and columns to graph nodes.

`method` Trasform data method. It can be one of the following:

1. "npn" (default), performs nonparanormal(npn) or semiparametric Gaussian copula model (Liu et al, 2009), estimating the Gaussian copula by marginally transforming the variables using smooth ECDF functions. The npn distribution corresponds to the latent underlying multivariate normal distribution, preserving the conditional independence structure of the original variables.
2. "spearman", computes a trigonometric trasformation of Spearman rho correlation for estimation of latent Gaussian correlations parameter of a nonparanormal distribution (Harris & Dorton (2013), and generates the data matrix with the exact same sample covariance matrix as the estimated one.
3. "kendall", computes a trigonometric trasformation of Kendall tau correlation for estimation of latent Gaussian correlations parameter of a nonparanormal distribution (Harris & Dorton (2013), and generates the data matrix with the exact same sample covariance matrix as the estimated one.
4. "polichoric", computes the polychoric correlation matrix and generates the data matrix with the exact same sample covariance matrix as the estimated one. The polychoric correlation (Olsson, 1974) is a measure of association between two ordinal variables. It is based on the assumption that two latent bivariate normally distributed random variables generate couples of ordinal scores. Tetrachoric (two binary variables) and biserial (an ordinal and a numeric variables) correlations are special cases.
5. "lineals", performs optimal scaling in order to achieve linearizing transformations for each bivariate regression between pairwise variables for subsequent structural equation models using the resulting correlation matrix computed on the transformed data (de Leeuw, 1988).
6. "mca", performs optimal scaling of categorical data by Multiple Correspondence Analysis (MCA, a.k.a homogeneity analysis) maximizing the first eigenvalues of the trasformed correlation matrix. The estimates of the corresponding structural parameters are consistent if the underlying latent space of the observed variables is unidimensional.

`...` Currently ignored.

Details

Nonparanormal trasformation is computationally very efficient and only requires one ECDF pass of the data matrix. Polychoric correlation matrix is computed with the `lavCor()` function of the `lavaan` package. Optimal scaling (lineals and mca) is performed with the `lineals()` and `corAspect()` functions of the `aspect` package (Mair and De Leeuw, 2008). To note, SEM fitting of the generate data (fake data) must be done with a covariance-based method and bootstrap SE, i.e., with `SEMrun(..., algo="ricf", n_rep=1000)`.

Value

A list of 2 objects is returned:

1. "data", the matrix (n x p) of n observations and p transformed variables or the matrix (n x p) of simulate observations based on the selected correlation matrix.
2. "catscores", the category weights for "lineals" or "mca" methods or NULL otherwise.

Author(s)

Mario Grassi <mario.grassi@unipv.it>

References

Liu H, Lafferty J, and Wasserman L (2009). The Nonparanormal: Semiparametric Estimation of High Dimensional Undirected Graphs. Journal of Machine Learning Research 10(80): 2295-2328

Harris N, and Drton M (2013). PC Algorithm for Nonparanormal Graphical Models. Journal of Machine Learning Research 14 (69): 3365-3383

Olsson U (1979). Maximum likelihood estimation of the polychoric correlation coefficient. Psychometrika, 44(4), 443-460.

Mair P, and De Leeuw J (2008). Scaling variables by optimizing correlational and non-correlational aspects in R. Journal of Statistical Software, 32(9), 1-23.

de Leeuw J (1988). Multivariate analysis with linearizable regressions. Psychometrika, 53, 437-454.

Examples

```
#... with continuous ALS data
graph<- alsData$graph
data<- alsData$exprs; dim(data)
X<- data[, colnames(data) %in% V(graph)$name]; dim(X)

npn.data<- transformData(X, method="npn")
sem0.npn<- SEMrun(graph, npn.data$data)

mvnS.data<- transformData(X, method="spearman")
sem0.mvnS<- SEMrun(graph, mvnS.data$data)

mvnK.data<- transformData(X, method="kendall")
sem0.mvnK<- SEMrun(graph, mvnK.data$data)

#...with ordinal (K=4 categories) ALS data
Xord <- data.frame(X)
Xord <- as.data.frame(lapply(Xord, cut, 4, labels = FALSE))
colnames(Xord) <- sub("X", "", colnames(Xord))

## Not run:

mvnP.data<- transformData(Xord, method="polychoric")
sem0.mvnP<- SEMrun(graph, mvnP.data$data, algo="ricf", n_rep=1000)


## End(Not run)
```

```
lin.data<- transformData(Xord, method="lineals")
sem0.lin<- SEMrun(graph, lin.data$data)
lin.data$catscores; head(lin.data$data)

#...with nominal (K=4 categories) ALS data
mca.data<- transformData(Xord, method="mca")
sem0.mca<- SEMrun(graph, mca.data$data)
mca.data$catscores

# plot colored graphs
#par(mfrow=c(3,2), mar=rep(1,4))
#gplot(sem0.npn$graph, l="fdp", main="ALS npm")
#gplot(sem0.mvnS$graph, l="fdp", main="ALS mvnS")
#gplot(sem0.mvnK$graph, l="fdp", main="ALS mvnK")
#gplot(sem0.mvnP$graph, l="fdp", main="ALS mvnP")
#gplot(sem0.lin$graph, l="fdp", main="ALS lin")
#gplot(sem0.mca$graph, l="fdp", main="ALS mca")
```

`weightGraph` *Graph weighting methods*

Description

Add data-driven edge and node weights to the input graph.

Usage

```
weightGraph(graph, data, group = NULL, method = "r2z", limit = 10000, ...)
```

Arguments

`graph` An igraph object.

`data` A matrix or data.frame. Rows correspond to subjects, and columns to graph nodes.

`group` Binary vector. This vector must be as long as the number of subjects. Each vector element must be 1 for cases and 0 for control subjects. By default, `group = NULL`. If group is not NULL, also node weighting is actived, and node weights correspond to the attribute: V(graph)$pv (P-value of the z-test = b/SE(b) from simple linear regression y ~ x, i.e., lm(node ~ group)) and V(graph)$sign (-1 if z<-2, +1 if z>2, 0 otherwise).

`method` Edge weighting method. It can be one of the following:

1. "r2z", weight edges are defined using Fisher's r-to-z transform (Fisher, 1915) to test the correlation coefficient of pairs of interacting nodes, if `group=NULL`. Otherwise, the difference between group of the r-to-z trasform will be tested. Edge weights correspond to the attribute: E(graph)$pv (P-value of the z-test) and E(graph)$sign (-1 if z<-2, +1 if z>2, 0 otherwise).

2. "sem", edge weights are defined by a SEM model that implies testing the group effect simultaneously on source and sink nodes. A new parameter w is defined as the weighted sum of the total effect of the group on source and sink nodes, adjusted by node degree centrality. Edge weights correspond to the attribute: E(graph)$pv (P-value of the z-test = w/SE(w)) and E(graph)$sign (-1 if z<-2, +1 if z>2, 0 otherwise). Not available if `group=NULL`.
3. "cov", edge weights are defined by a new parameter w combining the group effect on the source node (mean group difference, adjusted by source degree centrality), the sink node (mean group difference, adjusted by sink degree centrality), and the source–sink interaction (correlation difference). Edge weights correspond to the attribute: E(graph)$pv (P-value of the z-test = w/SE(w) of the combined difference of the group over source node, sink node, and their connection) and E(graph)$sign (-1 if z<-2, +1 if z>2, 0 otherwise). Not available if `group=NULL`.
4. "cfa", edge weights are defined by a CFA1 model that implies testing the group effect, w on a latent variable (LV) with observed indicators two interacting nodes, fixing loading coefficients and residual variances for model identification. Edge weights correspond to the attribute: E(graph)$pv (P-value of the z-test = w/SE(w) of the group effect on the LV) and E(graph)$sign (-1 if z<-2, +1 if z>2, 0 otherwise). Not available if `group=NULL`.

`limit` An integer value corresponding to the number of graph edges. Beyond this limit, multicore computation is enabled to reduce the computational burden. By default, `limit = 10000`.

`...` Currently ignored.

Value

A weighted graph, as an igraph object.

Author(s)

Mario Grassi `<mario.grassi@unipv.it>`

References

Grassi M, Tarantino B (2023). [Supplementary material of] SEMtree: tree-based structure learning methods with structural equation models. Bioinformatics, 39 (6), 4829–4830 <https://doi.org/10.1093/bioinformatics/btad377

Fisher RA (1915). Frequency Distribution of the Values of the Correlation Coefficient in Samples from an Indefinitely Large Population. Biometrika, 10(4), 507–521. <doi:10.2307/2331838>

Examples

```
# Graph weighting
G <- weightGraph(graph = sachs$graph,
                 data = log(sachs$pkc),
                 group = sachs$group,
                 method = "r2z")
```

```
# New edge attributes
head(E(G)$pv); summary(E(G)$pv)
head(E(G)$zsign); table(E(G)$zsign)

# New node attributes
head(V(G)$pv); summary(V(G)$pv)
head(V(G)$zsign); table(V(G)$zsign)
```

Index