Package 'JointNets'

July 30, 2019

Version 2.0.1

Date 2019-7-20

Encoding UTF-8

Title End-to-End Sparse Gaussian Graphical Model Simulation, Estimation, Visualization, Evaluation and Application

Author Zhaoyang Wang [aut],
Beilun Wang [aut],
Arshdeep Sekhon [aut, cre],
Yanjun Qi [aut]

Maintainer Arshdeep Sekhon <as5cu@virginia.edu>

Depends R (>= 3.4.4), lpSolve, pcaPP, igraph, parallel, JGL

Imports MASS, brainR, misc3d, oro.nifti, shiny, rgl, methods

Description An end-to-end package for learning multiple sparse Gaussian graphical models and non-paranormal models from Heterogeneous Data with Additional Knowledge. It is able to simulate multiple related graphs as well as produce samples drawn from them. Multiple state-of-the-art sparse Gaussian graphical model estimators are included to both multiple and difference estimation. Graph visualization is available in 2D as well as 3D, designed specifically for brain. Moreover, a set of evaluation metrics are integrated for easy exploration with model validity. Finally, classification using graphical model is achieved with Quadratic Discriminant Analysis. The package comes with multiple demos with datasets from various fields. Methods references: SIMULE (Wang B et al. (2017) <doi:10.1007/s10994-017-5635-7>), WSIMULE (Singh C et al. (2017) <arXiv:1709.04090v2>), DIFFEE (Wang B et al. (2018) <arXiv:1710.11223>), JEEK (Wang B et al. (2018) <arXiv:1806.00548>), JGL(Danaher P et al. (
iffnet (Sekhon A et al, preprint for publication).

License GPL-2

URL https://github.com/QData/JointNets

BugReports https://github.com/QData/JointNets

RoxygenNote 6.1.0

NeedsCompilation no

Repository CRAN

Date/Publication 2019-07-29 22:40:07 UTC

R topics documented:

`aal116coordinates` *AAL116 brain atlas coordinates in MNI space*

Description

Automated Anatomical Labeling (AAL): The AAL atlas distributed with the AAL Toolbox was fractionated to functional resolution (3x3x3 mm3) using nearest-neighbor interpolation. This data is available at `http://preprocessed-connectomes-project.org/abide/Pipelines.html` as part of ABIDE-preprocessed dataset. It can be directly downloaded at `https://fcp-indi.s3.amazonaws.com/data/Projects/ABIDE_Initiative/Resources/aal_roi_atlas.nii.gz`

Usage

```
data(aal116coordinates)
```

Format

116 observations (Brain Region Names) of 7 variables (name, x.mni, y.mni, z.mni, lobe, hemi, index)

References

Cameron Craddock, Yassine Benhajali, Carlton Chu, Francois Chouinard, Alan Evans, András Jakab, Budhachandra Singh Khundrakpam, John David Lewis, Qingyang Li, Michael Milham, Chaogan Yan, Pierre Bellec (2013). The Neuro Bureau Preprocessing Initiative: open sharing of preprocessed neuroimaging data and derivatives. In Neuroinformatics 2013, Stockholm, Sweden.

ABIDE_aal116_timeseries

ABIDE I preprocessed time series grouped by control and autism and partitioned by AAL116 atlas

Description

This time series data is available as part of Autism Brain Imaging Data Exchange (ABIDE). ABIDE is a collaboration of 16 international imaging sites that have aggregated and are openly sharing neuroimaging data from 539 individuals suffering from ASD and 573 typical controls. For data access, please refer to `http://preprocessed-connectomes-project.org/abide/download.html`. The data is preprocessed, cancatnated and organized into two data matrices for easy input.

Usage

```
data(ABIDE_aal116_timeseries)
```

Format

a list of two data matrices of time series(1:2250, 1:116) and (1:2060,1:116)

References

Cameron Craddock, Yassine Benhajali, Carlton Chu, Francois Chouinard, Alan Evans, András Jakab, Budhachandra Singh Khundrakpam, John David Lewis, Qingyang Li, Michael Milham, Chaogan Yan, Pierre Bellec (2013). The Neuro Bureau Preprocessing Initiative: open sharing of preprocessed neuroimaging data and derivatives. In Neuroinformatics 2013, Stockholm, Sweden.

add_name_to_out

helper function to add row/col names to JointNets precision matrix output To help label igraph object in returngraph and plot

Description

helper function to add row/col names to JointNets precision matrix output To help label igraph object in returngraph and plot

Usage

```
add_name_to_out(output, datalist, ...)
```

Arguments

`output`	output of jointnets
`datalist`	orginial data list
`...`	unused

Value

output with names from datalist

`AUC` *return AUC score for JointNets method*

Description

return AUC score for JointNets method

Usage

```
AUC(simulationresult, gm_method = "simule", lambdas, ...)
```

Arguments

`simulationresult`	output from the function simulation()
`gm_method`	method name from any one of the JointNets methods
`lambdas`	a vector of lambda values for the JointNets method to run with
`...`	extra parameters passed to the JointNets method such as lambda, epislon and etc, refer to each method for details (eg, ?simule)

Value

AUC score, a list of precisions and recalls

Author(s)

Zhaoyang Wang (Author), Zhaoyang Wang (maintainer) <zw4dn@virginia.edu>

Examples

```
simulationresult = simulation(n=c(100,100,100))
AUC_result = AUC(simulationresult,lambdas = seq(0.1,2,0.5),epsilon = 2)
AUC_result
graphics.off()
par(ask = FALSE)
par(mfrow = c(1, 1))
plot(AUC_result$fPM,AUC_result$tPM)
```

BIC *calculate BIC score for JointNets method*

Description

calculate BIC score for JointNets method

Usage

```
BIC(datalist, result)
```

Arguments

datalist	datalist used as an input to any of the JointNets method
result	result generated from datalist using the same JointNets method

Details

not working with DIFFEE and kdiffnet (difference estimation)

Value

BIC score

Author(s)

Zhaoyang Wang (Author), Zhaoyang Wang (maintainer) <zw4dn@virginia.edu>

Examples

```
library(JointNets)
simulateresult = simulation(p = 20,  n = c(100,100))
result = simule(simulateresult$simulatedsamples, 0.2, 0.5, covType = "cov", FALSE)
BIC(simulateresult$simulatedsamples,result)
```

cancer *Microarray data set for breast cancer*

Description

et al's paper. It concerns one hundred thirty-three patients with stage I–III breast cancer. Patients were treated with chemotherapy prior to surgery. Patient response to the treatment can be classified as either a pathologic complete response (pCR) or residual disease (not-pCR). Hess *et al* developed and tested a reliable multigene predictor for treatment response on this data set, composed by a set of 26 genes having a high predictive value.

Usage

```
data(cancer)
```

Format

a list of two objects: dataframe with 133 observations of 26 features and factors indicating whether each sample (out of 133) is of type "not" or type "pcr"

Details

The dataset splits into 2 parts (pCR and not pCR), on which network inference algorithms should be applied independently or in the multitask framework: only individuals from the same classes should be consider as independent and identically distributed.

References

J.A. Mejia, D. Booser, R.L. Theriault, U. Buzdar, P.J. Dempsey, R. Rouzier, N. Sneige, J.S. Ross, T. Vidaurre, H.L. Gomez, G.N. Hortobagyi, and L. Pustzai (2006). Pharmacogenomic predictor of sensitivity to preoperative chemotherapy with Paclitaxel and Fluorouracil, Doxorubicin, and Cyclophosphamide in breast cancer, *Journal of Clinical Oncology*, vol. 24(26), pp. 4236–4244.

---

compute_cov — *helper function to add compute covariance matrix / kendall tau correlation matrix*

Description

helper function to add compute covariance matrix / kendall tau correlation matrix

Usage

```
compute_cov(X, covType = "cov")
```

Arguments

X	data matrix
covType	"cov" or "kendall"

Value

covriance matrix / kendall tau correlation matrix

diffee *Fast and Scalable Learning of Sparse Changes in High-Dimensional Gaussian Graphical Model*

Description

Estimate DIFFerential networks via an Elementary Estimator under a high-dimensional situation. Please run demo(diffee) to learn the basics. For further details, please read the original paper: Beilun Wang, Arshdeep Sekhon, Yanjun Qi (2018) https://arxiv.org/abs/1710.11223.

Usage

```
diffee(C, D, lambda = 0.05, covType = "cov", intertwined = FALSE,
  thre = "soft")
```

Arguments

C	A input matrix for the 'control' group. It can be data matrix or covariance matrix. If C is a symmetric matrix, the matrices are assumed to be covariance matrix.
D	A input matrix for the 'disease' group. It can be data matrix or covariance matrix. If D is a symmetric matrix, the matrices are assumed to be covariance matrix.
lambda	A positive number. The hyperparameter controls the sparsity level of the matrices. The λ_n in the following section: Details.
covType	A parameter to decide which Graphical model we choose to estimate from the input data. If covType = "cov", it means that we estimate multiple sparse Gaussian Graphical models. This option assumes that we calculate (when input X represents data directly) or use (when X elements are symmetric representing covariance matrices) the sample covariance matrices as input to the simule algorithm. If covType = "kendall", it means that we estimate multiple nonparanormal Graphical models. This option assumes that we calculate (when input X represents data directly) or use (when X elements are symmetric representing correlation matrices) the kendall's tau correlation matrices as input to the simule algorithm.
intertwined	indicate whether to use intertwined covariance matrix
thre	A parameter to decide which threshold function to use for T_v. If thre = "soft", it means that we choose soft-threshold function as T_v. If thre = "hard", it means that we choose hard-threshold function as T_v.

Details

The DIFFEE algorithm is a fast and scalable Learning algorithm of Sparse Changes in High-Dimensional Gaussian Graphical Model Structure. It solves the following equation:

$$\min_{\Delta} ||\Delta||_1$$

Subject to :

$$([T_v(\hat{\Sigma}_d)]^{-1} - [T_v(\hat{\Sigma}_c)]^{-1})||_\infty \leq \lambda_n$$

Please also see the equation (2.11) in our paper. The λ_n is the hyperparameter controlling the sparsity level of the matrix and it is the `lambda` in our function. For further details, please see our paper: Beilun Wang, Arshdeep Sekhon, Yanjun Qi (2018) `https://arxiv.org/abs/1710.11223`.

if labels are provided in the datalist as column names, result will contain labels (to be plotted)

Value

`$graphs`	A matrix of the estimated sparse changes between two Gaussian Graphical Models
`$share`	null

Author(s)

Beilun Wang

References

Beilun Wang, Arshdeep Sekhon, Yanjun Qi (2018). Fast and Scalable Learning of Sparse Changes in High-Dimensional Gaussian Graphical Model Structure. `https://arxiv.org/abs/1710.11223`

Examples

```
library(JointNets)
data(exampleData)
result = diffee(exampleData[[1]], exampleData[[2]], 0.45)
plot(result)
```

---

dimension_reduce *reduce the dimensionality of the datalist if needed*

Description

reduce the dimensionality of the datalist if needed

Usage

```
dimension_reduce(datalist)
```

Arguments

`datalist`	a datalist of high dimensionality

Value

a datalist of reduced dimensionality

Examples

```
library(JointNets)
data(exampleData)
reduction = dimension_reduce(exampleData)
```

`exampleData` *A simulated toy dataset that includes 2 data matrices (from 2 related tasks).*

Description

A simulated toy dataset that includes 2 data matrices (from 2 related tasks). Each data matrix is about 100 features observed in 200 samples. The two data matrices are about exactly the same set of 100 features. This multi-task dataset is generated from two related random graphs. Please run demo(diffee) to learn the basic functions provided by this package. For further details, please read the original paper: `http://link.springer.com/article/10.1007/s10994-017-5635-7`.

Usage

```
data(exampleData)
```

Format

The format is: List of 2 matrices $: num (1:200, 1:100) -0.0982 -0.2417 -1.704 0.4 ... ..- attr(*, "dimnames")=List of 2 .. ..$: NULL .. ..$: NULL $: num (1:200, 1:100) -0.161 0.41 0.17 0. ... ..- attr(*, "dimnames")=List of 2 .. ..$: NULL .. ..$: NULL

`exampleDataGraph` *A simulated toy dataset that includes 3 igraph objects*

Description

(first one being the shared graph and second and third being task specific 1 and 2 graphs) The graphs are generated from two related random graphs and the underlaying high dimensional gaussian distribution generates the exampleData dataset. exampleDataGraph serves as a groundtruth to compare in demo(synthetic).

Usage

```
data(exampleDataGraph)
```

Format

A list of 3 igraph objects

`F1` *Compute F1 score for JointNets result*

Description

Compute F1 score for JointNets result

Usage

```
F1(result, simulatedgraphs, ...)
```

Arguments

`result`	output generated from any one of the jointnet algorithms
`simulatedgraphs`	$simulatedgraphs from function simulation()
`...`	unused

Value

F1 scores (F1 score for each context and the shared part (for simule and wsimule))

Author(s)

Zhaoyang Wang (Author), Zhaoyang Wang (maintainer) <zw4dn@virginia.edu>

Examples

```
library(JointNets)
simulationresult = simulation(p = 20,  n = c(100,100))
truth = simulationresult$simulatedgraphs
result = simule(simulationresult$simulatedsamples, 0.2, 0.5, covType = "cov", FALSE)
F1(result,truth)
```

F1.diffee *computes F1 score for jointnet result*

Description

computes F1 score for jointnet result

Usage

```
## S3 method for class 'diffee'
F1(result, simulatedgraphs, ...)
```

Arguments

result	output generated from any one of the jointnet algorithms
simulatedgraphs	$simulatedgraphs from function simulation()
...	unused

Examples

```
library(JointNets)
simulationresult = simulation(p = 20,  n = c(100,100))
truth = simulationresult$simulatedgraphs
result = diffee(simulationresult$simulatedsamples[[1]],
simulationresult$simulatedsamples[[2]], 0.01)
F1(result,truth)
```

F1.jeek *computes F1 score for jointnet result*

Description

computes F1 score for jointnet result

Usage

```
## S3 method for class 'jeek'
F1(result, simulatedgraphs, ...)
```

Arguments

result	output generated from any one of the jointnet algorithms
simulatedgraphs	$simulatedgraphs from function simulation()
...	unused

Examples

```
library(JointNets)
simulationresult = simulation(p = 20,  n = c(100,100))
truth = simulationresult$simulatedgraphs
result = jeek(simulationresult$simulatedsamples,0.25,covType = "kendall",parallel = FALSE)
F1(result,truth)
```

`F1.kdiffnet` *computes F1 score for jointnet result*

Description

computes F1 score for jointnet result

Usage

```
## S3 method for class 'kdiffnet'
F1(result, simulatedgraphs, ...)
```

Arguments

`result`	output generated from any one of the jointnet algorithms
`simulatedgraphs`	$simulatedgraphs from function simulation()
`...`	unused

Examples

```
library(JointNets)
simulationresult = simulation(p = 20,  n = c(100,100))
truth = simulationresult$simulatedgraphs
result = kdiffnet(simulationresult$simulatedsamples[[1]],
simulationresult$simulatedsamples[[2]],
W = matrix(1,20,20), g = rep(0,20),epsilon = 0.2,
lambda = 0.4,covType = "cov")
F1(result,truth)
```

F1.simule *computes F1 score for jointnet result*

Description

computes F1 score for jointnet result

Usage

```
## S3 method for class 'simule'
F1(result, simulatedgraphs, ...)
```

Arguments

result	output generated from any one of the jointnet algorithms
simulatedgraphs	$simulatedgraphs from function simulation()
...	unused

Examples

```
library(JointNets)
simulationresult = simulation(p = 20,  n = c(100,100))
truth = simulationresult$simulatedgraphs
result = simule(simulationresult$simulatedsamples, 0.2, 0.5, covType = "cov", FALSE)
F1(result,truth)
```

F1.wsimule *computes F1 score for jointnet result*

Description

computes F1 score for jointnet result

Usage

```
## S3 method for class 'wsimule'
F1(result, simulatedgraphs, ...)
```

Arguments

result	output generated from any one of the jointnet algorithms
simulatedgraphs	$simulatedgraphs from function simulation()
...	unused

Examples

```
library(JointNets)
simulationresult = simulation(p = 20,  n = c(100,100))
truth = simulationresult$simulatedgraphs
result = wsimule(simulationresult$simulatedsamples,
 0.2, 1, W = matrix(1,20,20), covType = "cov", FALSE)
F1(result,truth)
```

---

generateSampleList *function to generate a list of samples from simulatedGraph result*

Description

function to generate a list of samples from simulatedGraph result

Usage

```
generateSampleList(simulate, n)
```

Arguments

simulate	result from simulateGraph
n	a vector of corresponding size to indicate number of samples for each task

Details

if n is c(100,200,300) and p is 20, the function will return a list of 3 data matrices of size (100x20,200x20,300x20)

Value

a list of length(n) data matrices

---

generateSamples *function to generate samples from a single precision matrix*

Description

function to generate samples from a single precision matrix

Usage

```
generateSamples(precision, n = 100)
```

Arguments

precision	pxp precision matrix (generated from simulateGraph)
n	number of samples

Value

a list of nXp randomly generated gaussian samples from pxp precision matrix

---

`jeek` *A Fast and Scalable Joint Estimator for Integrating Additional Knowledge in Learning Multiple Related Sparse Gaussian Graphical Models*

Description

A Fast and Scalable Joint Estimator for Integrating Additional Knowledge in Learning Multiple Related Sparse Gaussian Graphical Models. Please run demo(jeek) to learn the basic functions provided by this package. For further details, please read the original paper: Beilun Wang, Arshdeep Sekhon, Yanjun Qi (2018).

Usage

```
jeek(X, lambda, W = NA, covType = "cov", intertwined = FALSE,
  parallel = FALSE)
```

Arguments

`X`	A List of input matrices. They can be data matrices or covariance/correlation matrices. If every matrix in the X is a symmetric matrix, the matrices are assumed to be covariance/correlation matrices.
`lambda`	A positive number. The hyperparameter controls the sparsity level of the matrices. The λ_n in the following section: Details.
`W`	A list of weight matrices. The hyperparameter intergrating the additional knowledge into the model. The W_{ij} is large means that node i and node j have less probability to connect with each other. The default value of each entry is 1, which means there is no additional knowledge in the formulation.
`covType`	A parameter to decide which Graphical model we choose to estimate from the input data. If covType = "cov", it means that we estimate multiple sparse Gaussian Graphical models. This option assumes that we calculate (when input X represents data directly) or use (when X elements are symmetric representing covariance matrices) the sample covariance matrices as input to the JEEK algorithm. If covType = "kendall", it means that we estimate multiple nonparanormal Graphical models. This option assumes that we calculate (when input X represents data directly) or use (when X elements are symmetric representing correlation matrices) the kendall's tau correlation matrices as input to the JEEK algorithm.
`intertwined`	indicate whether to use intertwined covariance matrix
`parallel`	A boolean. This parameter decides if the package will use the multithreading architecture or not.

Details

The JEEK algorithm is a novel Joint Elementary Estimator incorporating additional Knowledge (JEEK) to infer multiple related sparse Gaussian Graphical models from large-scale heterogeneous data. It solves the following equation:

$$\min_{\Omega_I^{tot},\Omega_S^{tot}} ||W_I^{tot} \circ \Omega_I^{tot}||_1 + ||W_S^{tot} \circ \Omega_S^{tot}||$$

Subject to :

$$||W_I^{tot} \circ (\Omega^{tot} - inv(T_v(\hat{\Sigma}^{tot})))||_\infty \leq \lambda_n$$

$$||W_S^{tot} \circ (\Omega^{tot} - inv(T_v(\hat{\Sigma}^{tot})))||_\infty \leq \lambda_n$$

$$\Omega^{tot} = \Omega_S^{tot} + \Omega_I^{tot}$$

Please also see the equation (3.7) in our paper. The λ_n is the hyperparameter controlling the sparsity level of the matrices and it is the `lambda` in our function. For further details, please see our paper: Beilun Wang, Arshdeep Sekhon, Yanjun Qi. A Fast and Scalable Joint Estimator for Integrating Additional Knowledge in Learning Multiple Related Sparse Gaussian Graphical Models. ICML 2018

if labels are provided in the datalist as column names, result will contain labels (to be plotted)

Value

`$graphs` A list of the estimated inverse covariance/correlation matrices.

Author(s)

Beilun Wang

References

Beilun Wang, Arshdeep Sekhon, Yanjun Qi. A Fast and Scalable Joint Estimator for Integrating Additional Knowledge in Learning Multiple Related Sparse Gaussian Graphical Models. `https://arxiv.org/abs/1806.00548`

Examples

```
library(JointNets)
data(exampleData)
result = jeek(X = exampleData, 0.3, covType = "cov", parallel = FALSE)
plot(result)
```

jgl *wrapper for function JGL fromo package "JGL"*

Description

wrapper for function JGL fromo package "JGL"

Usage

```
jgl(X, lambda1, lambda2, ...)
```

Arguments

X	data list
lambda1	The tuning parameter for the graphical lasso penalty.
lambda2	The tuning parameter for the fused or group lasso penalty.
...	optional parameters passed to JGL() from "JGL" package

Value

a list of estimated precision matrix

Examples

```
library(JointNets)
data(exampleData)
result = jgl(exampleData,0.1,0.01)
plot(result)
```

jointplot *core function to plot*

Description

core function to plot

Usage

```
jointplot(x, type = "task", neighbouroption = "task", subID = NULL,
  index = NULL, hastitle = TRUE, haslegend = TRUE, ...)
```

Arguments

`x`	output generated from JointNets
`type`	type of graph. There are four options: • "task" (graph for each task (including shared part) specified further by subID (task number)) • "share" (shared graph for all tasks) • "taskspecific" (graph for each task specific graph (excluding shared part) specified further by subID (task number)) • "neighbour" (zoom into nodes in the graph specified further by neighbouroptoin, subID (task number) and index (node id))
`neighbouroption`	determines what type of graph to zoom into when parameter **"type"** is **"neighbour"**. There are two options: • "task" (zoom into graph for each task (including shared part)) • "taskspecific" (zoom into graph for each task specific (excluding shared part))
`subID`	selects which task to display. There are four options: • 0 (only allowed when **"type"** is **"task"** or **"type"** is **"neighbour"** and **"neighbouroption"** is **"task"**) (selects share graph) • positive task number (selects that particular task) • a vector of task number (selects multiple tasks) • NULL (selects all tasks (all graphs))
`index`	determines which node(s) to zoom into when parameter **"type"** is **"neighbour"**. This parameter could either be an integer or vector of integers representing node ids (zoom into one node or multiple nodes)
`hastitle`	determines whether the graph title is displayed or not (TRUE to display / FALSE to hide)
`haslegend`	determines whether the graph legend is displayed or not (TRUE to display / FALSE to hide)
`...`	extra parameters passed to plot.igraph() and legend() (only the argument "legend" for legend() is available). Please see `plot.igraph` and `legend`

Value

a plot of graph

kdiffnet *Fast and Scalable Estimator for Using Additional Knowledge in Learning Sparse Structure Change of High Dimensional of Sparse Changes in High-Dimensional Gaussian Graphical Models*

Description

The kdiffnet algorithm

Usage

```
kdiffnet(C, D, W, g = rep(1, 100), epsilon = 1, lambda = 0.05,
  knowledgeType = "EV", gamma = 4, covType = "cov",
  intertwined = FALSE, thre = "soft", rho = 0.05, iterMax = 20)
```

Arguments

C	A input matrix for the 'control' group. It can be data matrix or covariance matrix. If C is a symmetric matrix, the matrices are assumed to be covariance matrix.
D	A input matrix for the 'disease' group. It can be data matrix or covariance matrix. If D is a symmetric matrix, the matrices are assumed to be covariance matrix.
W	known edge level additional knowledge. It is a square matrix of dimension p X p where p is the input dimension.
g	known node level additional knowledge. It is a vector of dimension 1 X p where p is the input dimension, each entry indicating membership of node to a group, 0 for a node belonging to no group. For example, in a dataset with dimension=3,g=c(0,1,1) indicates node 1 belongs to no group, and node 2 and node 3 belong to group index 1.
epsilon	A positive number. The hyperparameter controls the sparsity level of the groups in g of the difference matrix
lambda	A positive number. The hyperparameter controls the sparsity level of the difference matrix
knowledgeType	"EV": if use overlapping node and edge level additional knowledge,"E": if only edge level additional knowledge or "V": only group level knowledge
gamma	: A positive number. This hyperparameter is used in calculating each proximity during optimization
covType	A parameter to decide which Graphical model we choose to estimate from the input data. If covType = "cov", it means that we estimate multiple sparse Gaussian Graphical models. This option assumes that we calculate (when input X represents data directly) or use (when X elements are symmetric representing covariance matrices) the sample covariance matrices as input to the simule algorithm.

	If covType = "kendall", it means that we estimate multiple nonparanormal Graphical models. This option assumes that we calculate (when input X represents data directly) or use (when X elements are symmetric representing correlation matrices) the kendall's tau correlation matrices as input to the simule algorithm.
`intertwined`	indicate whether to use intertwined covariance matrix
`thre`	A parameter to decide which threshold function to use for T_v. If thre = "soft", it means that we choose soft-threshold function as T_v. If thre = "hard", it means that we choose hard-threshold function as T_v.
`rho`	A positive number. This hyperparameter controls the learning rate of the proximal gradient method.
`iterMax`	An integer. The max number of iterations in the optimization of the proximal algorithm

Value

`$graphs`	A matrix of the estimated sparse changes between two Gaussian Graphical Models
`$share`	null

Author(s)

Arshdeep Sekhon

Examples

```
library(JointNets)
data(exampleData)
result = kdiffnet(exampleData[[1]], exampleData[[2]],
W = matrix(1,20,20), g = rep(0,20),epsilon = 0.2,
lambda = 0.4,covType = "cov")
plot(result)
```

`nip_37_data` *NIPS word count dataset*

Description

This NIPS Conference Papers 1987-2015 Data set is avaiable at UCI Machine Learning Repository. The original dataset is in the form of a 11463 x 5812 matrix of word counts (11463 words and 5812 conference papers) Due to the size of the original dataset, it is preprocessed and reduced to a list of two matrices (2900 x 37 and 2911 x 37) The dataset consists of two tasks (early (up to 2006) and recent (after 2006) NIPS conference papers) with 37 words

Usage

```
data(nip_37_data)
```

Format

a list of two nonnegative integer matrices (1:2900, 1:37) and (1:2911,1:37) Columns are named with year_paperid and rows are names with word name

References

'Poisson Random Fields for Dynamic Feature Models'. Perrone V., Jenkins P. A., Spano D., Teh Y. W. (2016)

`plot.diffee` *plot diffee result specified by user input*

Description

This function can plot diffee result

Usage

```
## S3 method for class 'diffee'
plot(x, type = "task", index = NULL,
  hastitle = TRUE, ...)
```

Arguments

`x`	output generated from diffee function (diffee class)
`type`	type of graph. There are two options: • "task" (difference graph) • "neighbour" (zoom into nodes in the difference graph specified further by parameter **"index"** (node id)
`index`	determines which node(s) to zoom into when parameter **"type"** is **"neighbour"**. This parameter could either be an integer or vector of integers representing node ids
`hastitle`	determines whether the graph title is displayed or not (TRUE to display / FALSE to hide)
`...`	extra parameters passed to `plot.igraph` (zoom into one node or multiple nodes)

Details

when only the diffee result is provided, the function will plot all graphs with default numeric labels. Users can specify multiple subID to zoom in multiple nodes. Each graph will include a descriptive title.

Value

a plot of the difference graph from diffee result specified by user input

Author(s)

Beilun Wang, Zhaoyang Wang (Author), Zhaoyang Wang (maintainer) <zw4dn@virginia.edu>

Examples

```
library(JointNets)
data(exampleData)
result = diffee(exampleData[[1]], exampleData[[2]], 0.45)
plot.diffee(result)
```

---

plot.jeek *Plot jeek result specified by user input*

Description

This function can plot and return multiple sparse graphs distinguished by edge colors from the result generated by jeek

Usage

```
## S3 method for class 'jeek'
plot(x, type = "task", neighbouroption = "task",
  subID = NULL, index = NULL, hastitle = TRUE, haslegend = TRUE,
  ...)
```

Arguments

x	output generated from jeek function (jeek class)
type	type of graph. There are four options: • "task" (graph for each task (including shared part) specified further by subID (task number)) • "share" (shared graph for all tasks) • "taskspecific" (graph for each task specific graph (excluding shared part) specified further by subID (task number)) • "neighbour" (zoom into nodes in the graph specified further by neighbouroptoin, subID (task number) and index (node id))
neighbouroption	determines what type of graph to zoom into when parameter **"type"** is **"neighbour"**. There are two options: • "task" (zoom into graph for each task (including shared part)) • "taskspecific" (zoom into graph for each task specific (excluding shared part))
subID	selects which task to display. There are four options: • 0 (only allowed when **"type"** is **"task"** or **"type"** is **"neighbour"** and **"neighbouroption"** is **"task"**) (selects share graph)

- positive task number (selects that particular task)
- a vector of task number (selects multiple tasks)
- NULL (selects all tasks (all graphs))

`index` determines which node(s) to zoom into when parameter **"type"** is **"neighbour"**. This parameter could either be an integer or vector of integers representing node ids (zoom into one node or multiple nodes)

`hastitle` determines whether the graph title is displayed or not (TRUE to display / FALSE to hide)

`haslegend` determines whether the graph legend is displayed or not (TRUE to display / FALSE to hide)

`...` extra parameters passed to plot.igraph() and legend() (only the argument "legend" for legend() is available). Please see `plot.igraph` and `legend`

Details

when only the jeek result is provided, the function will plot all graphs with default numeric labels User can specify multiple subID and multiple index to zoom in multiple nodes on multiple graphs Each graph will include a descriptive title and legend to indicate correspondence between edge color and task.

Value

a plot of graph / subgraph from jeek result specified by user input

Author(s)

Beilun Wang, Zhaoyang Wang (Author), Zhaoyang Wang (maintainer) <zw4dn@virginia.edu>

Examples

```
library(JointNets)
data(exampleData)
result = jeek(X = exampleData, 0.3, covType = "cov", parallel = FALSE)
plot(result)
```

---

plot.jgl *Plot jgl result specified by user input*

Description

This function can plot and return multiple sparse graphs distinguished by edge colors from the result generated by jgl

Usage

```
## S3 method for class 'jgl'
plot(x, type = "task", neighbouroption = "task",
  subID = NULL, index = NULL, hastitle = TRUE, haslegend = TRUE,
  ...)
```

Arguments

x	output generated from jgl function (jgl class)
type	type of graph. There are four options: • "task" (graph for each task (including shared part) specified further by subID (task number)) • "share" (shared graph for all tasks) • "taskspecific" (graph for each task specific graph (excluding shared part) specified further by subID (task number)) • "neighbour" (zoom into nodes in the graph specified further by neighbouroptoin, subID (task number) and index (node id))
neighbouroption	determines what type of graph to zoom into when parameter **"type"** is **"neighbour"**. There are two options: • "task" (zoom into graph for each task (including shared part)) • "taskspecific" (zoom into graph for each task specific (excluding shared part))
subID	selects which task to display. There are four options: • 0 (only allowed when **"type"** is **"task"** or **"type"** is **"neighbour"** and **"neighbouroption"** is **"task"**) (selects share graph) • positive task number (selects that particular task) • a vector of task number (selects multiple tasks) • NULL (selects all tasks (all graphs))
index	determines which node(s) to zoom into when parameter **"type"** is **"neighbour"**. This parameter could either be an integer or vector of integers representing node ids (zoom into one node or multiple nodes)
hastitle	determines whether the graph title is displayed or not (TRUE to display / FALSE to hide)
haslegend	determines whether the graph legend is displayed or not (TRUE to display / FALSE to hide)
...	extra parameters passed to plot.igraph() and legend() (only the argument "legend" for legend() is available). Please see `plot.igraph` and `legend`

Details

when only the jgl result is provided, the function will plot all graphs with default numeric labels User can specify multiple subID and multiple index to zoom in multiple nodes on multiple graphs Each graph will include a descriptive title and legend to indicate correspondence between edge color and task.

Value

a plot of graph / subgraph from jgl result specified by user input

Author(s)

Beilun Wang, Zhaoyang Wang (Author), Zhaoyang Wang (maintainer) <zw4dn@virginia.edu>

Examples

```
library(JointNets)
data(exampleData)
result = jgl(exampleData,0.1,0.5)
plot(result)
```

---

plot.kdiffnet *plot kdiffnet result specified by user input*

Description

This function can plot kdiffnet result

Usage

```
## S3 method for class 'kdiffnet'
plot(x, type = "task", index = NULL,
  hastitle = TRUE, ...)
```

Arguments

x	output generated from diffee function (diffee class)
type	type of graph. There are two options: • "task" (difference graph) • "neighbour" (zoom into nodes in the difference graph specified further by parameter **"index"** (node id)
index	determines which node(s) to zoom into when parameter **"type"** is **"neighbour"**. This parameter could either be an integer or vector of integers representing node ids
hastitle	determines whether the graph title is displayed or not (TRUE to display / FALSE to hide)
...	extra parameters passed to plot.igraph (zoom into one node or multiple nodes)

Details

when only the kdiffnet result is provided, the function will plot all graphs with default numeric labels. Users can specify multiple subID to zoom in multiple nodes. Each graph will include a descriptive title.

Value

a plot of the difference graph from kdiffnet result specified by user input

Author(s)

Beilun Wang, Zhaoyang Wang (Author), Zhaoyang Wang (maintainer) <zw4dn@virginia.edu>

Examples

```
library(JointNets)
data(exampleData)
result = kdiffnet(exampleData[[1]], exampleData[[2]],
W = matrix(1,20,20), g = rep(0,20),epsilon = 0.2,
lambda = 0.4,covType = "cov")
plot(result)
```

---

plot.simulation *Plot simulatedgraph result (generated from function simulation()) (class simulation)*

Description

This function can plot and return multiple sparse graphs distinguished by edge colors from the result generated by simulation()

Usage

```
## S3 method for class 'simulation'
plot(x, type = "task", neighbouroption = "task",
  subID = NULL, index = NULL, hastitle = TRUE, haslegend = TRUE,
  ...)
```

Arguments

x	output generated from simule function (simule class)
type	type of graph. There are four options: • "task" (graph for each task (including shared part) specified further by subID (task number)) • "share" (shared graph for all tasks) • "taskspecific" (graph for each task specific graph (excluding shared part) specified further by subID (task number)) • "neighbour" (zoom into nodes in the graph specified further by neighbouroptoin, subID (task number) and index (node id))
neighbouroption	determines what type of graph to zoom into when parameter **"type"** is **"neighbour"**. There are two options:

- "task" (zoom into graph for each task (including shared part))
- "taskspecific" (zoom into graph for each task specific (excluding shared part))

`subID` selects which task to display. There are four options:

- 0 (only allowed when **"type"** is **"task"** or **"type"** is **"neighbour"** and **"neighbouroption"** is **"task"**) (selects share graph)
- positive task number (selects that particular task)
- a vector of task number (selects multiple tasks)
- NULL (selects all tasks (all graphs))

`index` determines which node(s) to zoom into when parameter **"type"** is **"neighbour"**. This parameter could either be an integer or vector of integers representing node ids (zoom into one node or multiple nodes)

`hastitle` determines whether the graph title is displayed or not (TRUE to display / FALSE to hide)

`haslegend` determines whether the graph legend is displayed or not (TRUE to display / FALSE to hide)

`...` extra parameters passed to plot.igraph() and legend() (only the argument "legend" for legend() is available). Please see `plot.igraph` and `legend`

Details

when only the simulatedgraph is provided, the function will plot all graphs with default numeric labels. User can specify multiple subID and multiple index to zoom in multiple nodes on multiple graphs. Each graph will include a descriptive title and legend to indicate correspondence between edge color and task.

Value

a plot of graph / subgraph from simulatedgraph result specified by user input

Author(s)

Beilun Wang, Zhaoyang Wang (Author), Zhaoyang Wang (maintainer) <zw4dn@virginia.edu>

Examples

```
library(JointNets)
data(exampleData)
result = simulation(n = c(100,100,100))$simulatedgraphs
plot(result)
```

`plot.simule` *Plot simule result specified by user input*

Description

This function can plot and return multiple sparse graphs distinguished by edge colors from the result generated by simule

Usage

```
## S3 method for class 'simule'
plot(x, type = "task", neighbouroption = "task",
  subID = NULL, index = NULL, hastitle = TRUE, haslegend = TRUE,
  ...)
```

Arguments

`x`	output generated from simule function (simule class)
`type`	type of graph. There are four options: • "task" (graph for each task (including shared part) specified further by subID (task number)) • "share" (shared graph for all tasks) • "taskspecific" (graph for each task specific graph (excluding shared part) specified further by subID (task number)) • "neighbour" (zoom into nodes in the graph specified further by neighbouroptoin, subID (task number) and index (node id))
`neighbouroption`	determines what type of graph to zoom into when parameter **"type"** is **"neighbour"**. There are two options: • "task" (zoom into graph for each task (including shared part)) • "taskspecific" (zoom into graph for each task specific (excluding shared part))
`subID`	selects which task to display. There are four options: • 0 (only allowed when **"type"** is **"task"** or **"type"** is **"neighbour"** and **"neighbouroption"** is **"task"**) (selects share graph) • positive task number (selects that particular task) • a vector of task number (selects multiple tasks) • NULL (selects all tasks (all graphs))
`index`	determines which node(s) to zoom into when parameter **"type"** is **"neighbour"**. This parameter could either be an integer or vector of integers representing node ids (zoom into one node or multiple nodes)
`hastitle`	determines whether the graph title is displayed or not (TRUE to display / FALSE to hide)

haslegend	determines whether the graph legend is displayed or not (TRUE to display / FALSE to hide)
...	extra parameters passed to plot.igraph() and legend() (only the argument "legend" for legend() is available). Please see plot.igraph and legend

Details

when only the simule result is provided, the function will plot all graphs with default numeric labels. User can specify multiple subID and multiple index to zoom in multiple nodes on multiple graphs. Each graph will include a descriptive title and legend to indicate correspondence between edge color and task.

Value

a plot of graph / subgraph from simule result specified by user input

Author(s)

Beilun Wang, Zhaoyang Wang (Author), Zhaoyang Wang (maintainer) <zw4dn@virginia.edu>

Examples

```
library(JointNets)
data(exampleData)
result = simule(X = exampleData , lambda = 0.1, epsilon = 0.45, covType = "cov", FALSE)
plot(result)
```

---

plot.wsimule *Plot wsimule result specified by user input*

Description

This function can plot and return multiple sparse graphs distinguished by edge colors from the result generated by wsimule

Usage

```
## S3 method for class 'wsimule'
plot(x, type = "task", neighbouroption = "task",
  subID = NULL, index = NULL, hastitle = TRUE, haslegend = TRUE,
  ...)
```

Arguments

x	output generated from wsimule function (wsimule class)
type	type of graph. There are four options: • "task" (graph for each task (including shared part) specified further by subID (task number)) • "share" (shared graph for all tasks) • "taskspecific" (graph for each task specific graph (excluding shared part) specified further by subID (task number)) • "neighbour" (zoom into nodes in the graph specified further by neighbouroptoin, subID (task number) and index (node id))
neighbouroption	determines what type of graph to zoom into when parameter **"type"** is **"neighbour"**. There are two options: • "task" (zoom into graph for each task (including shared part)) • "taskspecific" (zoom into graph for each task specific (excluding shared part))
subID	selects which task to display. There are four options: • 0 (only allowed when **"type"** is **"task"** or **"type"** is **"neighbour"** and **"neighbouroption"** is **"task"**) (selects share graph) • positive task number (selects that particular task) • a vector of task number (selects multiple tasks) • NULL (selects all tasks (all graphs))
index	determines which node(s) to zoom into when parameter **"type"** is **"neighbour"**. This parameter could either be an integer or vector of integers representing node ids (zoom into one node or multiple nodes)
hastitle	determines whether the graph title is displayed or not (TRUE to display / FALSE to hide)
haslegend	determines whether the graph legend is displayed or not (TRUE to display / FALSE to hide)
...	extra parameters passed to plot.igraph() and legend() (only the argument "legend" for legend() is available). Please see `plot.igraph` and `legend`

Details

when only the wsimule result is provided, the function will plot all graphs with default numeric labels. User can specify multiple subID and multiple index to zoom in multiple nodes on multiple graphs. Each graph will include a descriptive title and legend to indicate correspondence between edge color and task.

Value

a plot of graph / subgraph from wsimule result specified by user input

Author(s)

Beilun Wang, Zhaoyang Wang (Author), Zhaoyang Wang (maintainer) <zw4dn@virginia.edu>

Examples

```
library(JointNets)
data(exampleData)
result = wsimule(X = exampleData , lambda = 0.1, epsilon = 0.45,
W = matrix(1,20,20), covType = "cov", FALSE)
plot(result)
```

`plot_brain` *plot 3d brain network from JointNets result*

Description

This function plots 3d brain network from JointNets result

Usage

```
plot_brain(x, ...)
```

Arguments

`x`	output generated from any one of the JointNets functions
`...`	additional arguments, please see `plot_brain.simule`, `plot_brain.wsimule` and etc for details

Details

The function plots brain network using `rglplot.igraph`

Value

3d (rgl) brain network

Author(s)

Zhaoyang Wang (Author), Zhaoyang Wang (maintainer) <zw4dn@virginia.edu>

Examples

```
library(JointNets)
graphics.off()
par(ask=FALSE)
par(mfrow=c(1,1))
data(ABIDE_aal116_timeseries)
data(aal116coordinates)
layout = cbind(aal116coordinates$x.mni + 90,
aal116coordinates$y.mni+126, aal116coordinates$z.mni+72)
result = simulation(p=116, s = 0.001, ss = 0.001, n = c(1,1))$simulatedgraphs
class(result) = "simule"
```

```
plot_brain(result, type = "task", neighbouroption = "task",
subID = NULL, index = NULL, layout = layout, hasbackground = FALSE)

result = simule(ABIDE_aal116_timeseries, 0.2, 1, covType = "cov", FALSE)
plot_brain(result, type = "task", neighbouroption = "task",
subID = NULL, index = NULL, layout = layout)
```

`plot_brain.diffee` *plot 3d brain network from diffee result*

Description

This function plots 3d brain network from diffee result

Usage

```
## S3 method for class 'diffee'
plot_brain(x, type = "task", neighbouroption = "task",
  subID = NULL, index = NULL, hastitle = TRUE, haslegend = TRUE,
  hasbackground = TRUE, ...)
```

Arguments

`x`	output generated from diffee function (diffee class)
`type`	type of graph. There are two options: • "task" (difference graph) • "neighbour" (zoom into nodes in the difference graph specified further by parameter **"index"** (node id)
`neighbouroption`	not used
`subID`	not used
`index`	determines which node(s) to zoom into when parameter **"type"** is **"neighbour"**. This parameter could either be an integer or vector of integers representing node ids
`hastitle`	determines whether the graph title is displayed or not (TRUE to display / FALSE to hide)
`haslegend`	not used
`hasbackground`	determines whether the reference brain is plotted or not (TRUE to display / FALSE to hide)
`...`	extra parameters passed to igraph::rglplot()

Details

The function plots brain network using `rglplot.igraph`

Value

3d (rgl) brain network

Author(s)

Zhaoyang Wang (Author), Zhaoyang Wang (maintainer) <zw4dn@virginia.edu>

Examples

```
library(JointNets)
graphics.off()
par(ask=FALSE)
par(mfrow=c(1,1))
data(ABIDE_aal116_timeseries)
data(aal116coordinates)
layout = cbind(aal116coordinates$x.mni + 90,
aal116coordinates$y.mni+126, aal116coordinates$z.mni+72)
result = simulation(p=116, s = 0.001, ss = 0.001, n = c(1,1))$simulatedgraphs
class(result) = "simule"
plot_brain(result, type = "task", neighbouroption = "task",
subID = NULL, index = NULL, layout = layout, hasbackground = FALSE)

result = diffee(ABIDE_aal116_timeseries[[1]],
ABIDE_aal116_timeseries[[2]], 0.001)
plot_brain(result, type = "task", neighbouroption = "task",
subID = NULL, index = NULL, layout = layout)
```

---

plot_brain.jeek *plot 3d brain network from jeek result*

Description

This function plots 3d brain network from jeek result

Usage

```
## S3 method for class 'jeek'
plot_brain(x, type = "task", neighbouroption = "task",
  subID = NULL, index = NULL, hastitle = TRUE, haslegend = TRUE,
  hasbackground = TRUE, ...)
```

Arguments

x	output generated from jeek function (jeek class)
type	type of graph. There are four options: • "task" (graph for each task (including shared part) specified further by subID (task number))

- "share" (shared graph for all tasks)
- "taskspecific" (graph for each task specific graph (excluding shared part) specified further by subID (task number))
- "neighbour" (zoom into nodes in the graph specified further by neighbouroptoin, subID (task number) and index (node id))

`neighbouroption`
: determines what type of graph to zoom into when parameter **"type"** is **"neighbour"**. There are two options:
  - "task" (zoom into graph for each task (including shared part))
  - "taskspecific" (zoom into graph for each task specific (excluding shared part))

`subID`
: selects which task to display. There are four options:
  - 0 (only allowed when **"type"** is **"task"** or **"type"** is **"neighbour"** and **"neighbouroption"** is **"task"**) (selects share graph)
  - positive task number (selects that particular task)
  - a vector of task number (selects multiple tasks)
  - NULL (selects all tasks (all graphs))

`index`
: determines which node(s) to zoom into when parameter **"type"** is **"neighbour"**. This parameter could either be an integer or vector of integers representing node ids (zoom into one node or multiple nodes)

`hastitle`
: determines whether the graph title is displayed or not (TRUE to display / FALSE to hide)

`haslegend`
: determines whether the graph legend is displayed or not (TRUE to display / FALSE to hide)

`hasbackground`
: determines whether the reference brain is plotted or not (TRUE to display / FALSE to hide)

`...`
: extra parameters passed to igraph::rglplot()

Details

The function plots brain network using `rglplot.igraph`

Value

3d (rgl) brain network

Author(s)

Zhaoyang Wang (Author), Zhaoyang Wang (maintainer) `<zw4dn@virginia.edu>`

Examples

```
library(JointNets)
graphics.off()
par(ask=FALSE)
```

```
par(mfrow=c(1,1))
data(ABIDE_aal116_timeseries)
data(aal116coordinates)
layout = cbind(aal116coordinates$x.mni + 90,
aal116coordinates$y.mni+126, aal116coordinates$z.mni+72)
result = simulation(p=116, s = 0.001, ss = 0.001, n = c(1,1))$simulatedgraphs
class(result) = "simule"
plot_brain(result, type = "task", neighbouroption = "task",
subID = NULL, index = NULL, layout = layout, hasbackground = FALSE)

result = jeek(X = ABIDE_aal116_timeseries,0.25,
covType = "kendall",parallel = FALSE)
plot_brain(result, type = "task", neighbouroption = "task",
subID = NULL, index = NULL, layout = layout)
```

`plot_brain.jgl` *plot 3d brain network from jgl result*

Description

This function plots 3d brain network from jgl result

Usage

```
## S3 method for class 'jgl'
plot_brain(x, type = "task", neighbouroption = "task",
  subID = NULL, index = NULL, hastitle = TRUE, haslegend = TRUE,
  hasbackground = TRUE, ...)
```

Arguments

`x`	output generated from jgl function (jgl class)
`type`	type of graph. There are four options: • "task" (graph for each task (including shared part) specified further by subID (task number)) • "share" (shared graph for all tasks) • "taskspecific" (graph for each task specific graph (excluding shared part) specified further by subID (task number)) • "neighbour" (zoom into nodes in the graph specified further by neighbouroptoin, subID (task number) and index (node id))
`neighbouroption`	determines what type of graph to zoom into when parameter **"type"** is **"neighbour"**. There are two options: • "task" (zoom into graph for each task (including shared part)) • "taskspecific" (zoom into graph for each task specific (excluding shared part))

`subID` selects which task to display. There are four options:

- 0 (only allowed when **"type"** is **"task"** or **"type"** is **"neighbour"** and **"neighbouroption"** is **"task"**) (selects share graph)
- positive task number (selects that particular task)
- a vector of task number (selects multiple tasks)
- NULL (selects all tasks (all graphs))

`index` determines which node(s) to zoom into when parameter **"type"** is **"neighbour"**. This parameter could either be an integer or vector of integers representing node ids (zoom into one node or multiple nodes)

`hastitle` determines whether the graph title is displayed or not (TRUE to display / FALSE to hide)

`haslegend` determines whether the graph legend is displayed or not (TRUE to display / FALSE to hide)

`hasbackground` determines whether the reference brain is plotted or not (TRUE to display / FALSE to hide)

`...` extra parameters passed to igraph::rglplot()

Details

The function plots brain network using `rglplot.igraph`

Value

3d (rgl) brain network

Author(s)

Zhaoyang Wang (Author), Zhaoyang Wang (maintainer) <zw4dn@virginia.edu>

Examples

```
library(JointNets)
graphics.off()
par(ask=FALSE)
par(mfrow=c(1,1))
data(ABIDE_aal116_timeseries)
data(aal116coordinates)
layout = cbind(aal116coordinates$x.mni + 90,
aal116coordinates$y.mni+126, aal116coordinates$z.mni+72)
result = simulation(p=116, s = 0.001, ss = 0.001, n = c(1,1))$simulatedgraphs
class(result) = "jgl"
plot_brain(result, type = "task", neighbouroption = "task",
subID = NULL, index = NULL, layout = layout, hasbackground = FALSE)

result = jgl(ABIDE_aal116_timeseries, 0.2, 1)
plot_brain(result, type = "task", neighbouroption = "task",
subID = NULL, index = NULL, layout = layout)
```

`plot_brain.kdiffnet` *plot 3d brain network from kdiffnet result*

Description

This function plots 3d brain network from kdiffnet result

Usage

```
## S3 method for class 'kdiffnet'
plot_brain(x, type = "task",
  neighbouroption = "task", subID = NULL, index = NULL,
  hastitle = TRUE, haslegend = TRUE, hasbackground = TRUE, ...)
```

Arguments

`x`	output generated from kdiffnet function (kdiffnet class)
`type`	type of graph. There are two options: • "task" (difference graph) • "neighbour" (zoom into nodes in the difference graph specified further by parameter **"index"** (node id)
`neighbouroption`	not used
`subID`	not used
`index`	determines which node(s) to zoom into when parameter **"type"** is **"neighbour"**. This parameter could either be an integer or vector of integers representing node ids
`hastitle`	determines whether the graph title is displayed or not (TRUE to display / FALSE to hide)
`haslegend`	not used
`hasbackground`	determines whether the reference brain is plotted or not (TRUE to display / FALSE to hide)
`...`	extra parameters passed to igraph::rglplot()

Details

The function plots brain network using `rglplot.igraph`

Value

3d (rgl) brain network

Author(s)

Zhaoyang Wang (Author), Zhaoyang Wang (maintainer) <zw4dn@virginia.edu>

Examples

```
library(JointNets)
graphics.off()
par(ask=FALSE)
par(mfrow=c(1,1))
data(ABIDE_aal116_timeseries)
data(aal116coordinates)
layout = cbind(aal116coordinates$x.mni + 90,
aal116coordinates$y.mni+126, aal116coordinates$z.mni+72)
result = simulation(p=116, s = 0.001, ss = 0.001, n = c(1,1))$simulatedgraphs
class(result) = "simule"
plot_brain(result, type = "task", neighbouroption = "task",
subID = NULL, index = NULL, layout = layout, hasbackground = FALSE)

result = kdiffnet(ABIDE_aal116_timeseries[[1]], ABIDE_aal116_timeseries[[2]],
W = matrix(1,116,116), g = rep(0,116), epsilon = 0.1, lambda = 0.001)
plot_brain(result, type = "task", neighbouroption = "task",
subID = NULL, index = NULL, layout = layout)
```

---

plot_brain.simule *plot 3d brain network from simule result*

Description

This function plots 3d brain network from simule result

Usage

```
## S3 method for class 'simule'
plot_brain(x, type = "task", neighbouroption = "task",
  subID = NULL, index = NULL, hastitle = TRUE, haslegend = TRUE,
  hasbackground = TRUE, ...)
```

Arguments

x	output generated from simule function (simule class)
type	type of graph. There are four options: • "task" (graph for each task (including shared part) specified further by subID (task number)) • "share" (shared graph for all tasks) • "taskspecific" (graph for each task specific graph (excluding shared part) specified further by subID (task number))

- "neighbour" (zoom into nodes in the graph specified further by neighbouroptoin, subID (task number) and index (node id))

`neighbouroption`
: determines what type of graph to zoom into when parameter **"type"** is **"neighbour"**. There are two options:
  - "task" (zoom into graph for each task (including shared part))
  - "taskspecific" (zoom into graph for each task specific (excluding shared part))

`subID`
: selects which task to display. There are four options:
  - 0 (only allowed when **"type"** is **"task"** or **"type"** is **"neighbour"** and **"neighbouroption"** is **"task"**) (selects share graph)
  - positive task number (selects that particular task)
  - a vector of task number (selects multiple tasks)
  - NULL (selects all tasks (all graphs))

`index`
: determines which node(s) to zoom into when parameter **"type"** is **"neighbour"**. This parameter could either be an integer or vector of integers representing node ids (zoom into one node or multiple nodes)

`hastitle`
: determines whether the graph title is displayed or not (TRUE to display / FALSE to hide)

`haslegend`
: determines whether the graph legend is displayed or not (TRUE to display / FALSE to hide)

`hasbackground`
: determines whether the reference brain is plotted or not (TRUE to display / FALSE to hide)

`...`
: extra parameters passed to igraph::rglplot()

Details

The function plots brain network using `rglplot.igraph`

Value

3d (rgl) brain network

Author(s)

Zhaoyang Wang (Author), Zhaoyang Wang (maintainer) <zw4dn@virginia.edu>

Examples

```
library(JointNets)
graphics.off()
par(ask=FALSE)
par(mfrow=c(1,1))
data(ABIDE_aal116_timeseries)
data(aal116coordinates)
layout = cbind(aal116coordinates$x.mni + 90,
```

```
aal116coordinates$y.mni+126, aal116coordinates$z.mni+72)
result = simulation(p=116, s = 0.001, ss = 0.001, n = c(1,1))$simulatedgraphs
class(result) = "simule"
plot_brain(result, type = "task", neighbouroption = "task",
subID = NULL, index = NULL, layout = layout, hasbackground = FALSE)

result = simule(ABIDE_aal116_timeseries, 0.2, 1, covType = "cov", FALSE)
plot_brain(result, type = "task", neighbouroption = "task",
subID = NULL, index = NULL, layout = layout)
```

`plot_brain.wsimule` *plot 3d brain network from wsimule result*

Description

This function plots 3d brain network from wsimule result

Usage

```
## S3 method for class 'wsimule'
plot_brain(x, type = "task",
  neighbouroption = "task", subID = NULL, index = NULL,
  hastitle = TRUE, haslegend = TRUE, hasbackground = TRUE, ...)
```

Arguments

`x`	output generated from wsimule function (wsimule class)
`type`	type of graph. There are four options: • "task" (graph for each task (including shared part) specified further by subID (task number)) • "share" (shared graph for all tasks) • "taskspecific" (graph for each task specific graph (excluding shared part) specified further by subID (task number)) • "neighbour" (zoom into nodes in the graph specified further by neighbouroptoin, subID (task number) and index (node id))
`neighbouroption`	determines what type of graph to zoom into when parameter **"type"** is **"neighbour"**. There are two options: • "task" (zoom into graph for each task (including shared part)) • "taskspecific" (zoom into graph for each task specific (excluding shared part))
`subID`	selects which task to display. There are four options: • 0 (only allowed when **"type"** is **"task"** or **"type"** is **"neighbour"** and **"neighbouroption"** is **"task"**) (selects share graph)

- positive task number (selects that particular task)
- a vector of task number (selects multiple tasks)
- NULL (selects all tasks (all graphs))

`index` determines which node(s) to zoom into when parameter **"type"** is **"neighbour"**. This parameter could either be an integer or vector of integers representing node ids (zoom into one node or multiple nodes)

`hastitle` determines whether the graph title is displayed or not (TRUE to display / FALSE to hide)

`haslegend` determines whether the graph legend is displayed or not (TRUE to display / FALSE to hide)

`hasbackground` determines whether the reference brain is plotted or not (TRUE to display / FALSE to hide)

`...` extra parameters passed to igraph::rglplot()

Details

The function plots brain network using `rglplot.igraph`

Value

3d (rgl) brain network

Author(s)

Zhaoyang Wang (Author), Zhaoyang Wang (maintainer) <zw4dn@virginia.edu>

Examples

```
library(JointNets)
graphics.off()
par(ask=FALSE)
par(mfrow=c(1,1))
data(ABIDE_aal116_timeseries)
data(aal116coordinates)
layout = cbind(aal116coordinates$x.mni + 90,
aal116coordinates$y.mni+126, aal116coordinates$z.mni+72)
result = simulation(p=116, s = 0.001, ss = 0.001, n = c(1,1))$simulatedgraphs
class(result) = "simule"
plot_brain(result, type = "task", neighbouroption = "task",
subID = NULL, index = NULL, layout = layout, hasbackground = FALSE)

result = wsimule(ABIDE_aal116_timeseries, 0.2, 1,
W = matrix(1,116,116), covType = "cov", FALSE)
plot_brain(result, type = "task", neighbouroption = "task",
subID = NULL, index = NULL, layout = layout)
```

`plot_brain_joint` *plot 3d brain network*

Description

plot 3d brain network

Usage

```
plot_brain_joint(x, type = "task", neighbouroption = "task",
  subID = NULL, index = NULL, hastitle = TRUE, haslegend = TRUE,
  hasbackground = TRUE, ...)
```

Arguments

`x`	output generated from JointNets Methods
`type`	type of graph. There are four options: • "task" (graph for each task (including shared part) specified further by subID (task number)) • "share" (shared graph for all tasks) • "taskspecific" (graph for each task specific graph (excluding shared part) specified further by subID (task number)) • "neighbour" (zoom into nodes in the graph specified further by neighbouroptoin, subID (task number) and index (node id))
`neighbouroption`	determines what type of graph to zoom into when parameter **"type"** is **"neighbour"**. There are two options: • "task" (zoom into graph for each task (including shared part)) • "taskspecific" (zoom into graph for each task specific (excluding shared part))
`subID`	selects which task to display. There are four options: • 0 (only allowed when **"type"** is **"task"** or **"type"** is **"neighbour"** and **"neighbouroption"** is **"task"**) (selects share graph) • positive task number (selects that particular task) • a vector of task number (selects multiple tasks) • NULL (selects all tasks (all graphs))
`index`	determines which node(s) to zoom into when parameter **"type"** is **"neighbour"**. This parameter could either be an integer or vector of integers representing node ids (zoom into one node or multiple nodes)
`hastitle`	determines whether the graph title is displayed or not (TRUE to display / FALSE to hide)
`haslegend`	determines whether the graph legend is displayed or not (TRUE to display / FALSE to hide)

`hasbackground`
: determines whether the reference brain is plotted or not (TRUE to display / FALSE to hide)

`...`
: extra parameters passed to igraph::rglplot() and level in misc::contour3d()

Value

3d (rgl) brain network

`plot_gui` *GUI of JointNets plot*

Description

GUI version of JointNets plot (input from the global environment)

Usage

```
plot_gui()
```

Details

please refer to plot.simule, plot.wsimule and etc for details in plotting. value -1 for subID and index corresponds to NUL value

Author(s)

Zhaoyang Wang (Author), Zhaoyang Wang (maintainer) <zw4dn@virginia.edu>

Examples

```
library(JointNets)
if(interactive()){
plot_gui()
}
```

QDA_eval *graphical model model evaluation using QDA as a classifier*

Description

graphical model model evaluation using QDA as a classifier

Usage

```
QDA_eval(train, valid, test, lambda_range, v_seeking_length = 10,
  method = "diffee", ...)
```

Arguments

train	a list of training data
valid	a list of validation data
test	a list of test data
lambda_range	a vector of lambda values to train to given method, eg c(0.1,0.2,0.3)
v_seeking_length	second hyperparameter length, default to 10
method	name of the method to be evaluated
...	optional parameters passed to your method from JointNets package

Value

covriance matrix / kendall tau correlation matrix

Examples

```
library(JointNets)
data("nip_37_data")
split = train_valid_test_split(nip_37_data,c(0.8,0.1,0.1),10000)
train = split[["train"]]
valid = split[["valid"]]
test = split[["test"]]
v_seeking_length = 2
lambda_range = seq(0.5,1, length.out = 2)
result = QDA_eval(train,valid,test,lambda_range, v_seeking_length, method = "diffee")
result[["best test accuracy"]]
```

`returngraph` *return igraph object from jointnet result specified by user input*

Description

This function returns an igraph object from jointnet result for user to work with directly

Usage

```
returngraph(x, ...)
```

Arguments

`x`	output generated from any one of the jointnet functions
`...`	additional arguments, see `returngraph.simule`, `returngraph.wsimule`, `returngraph.diffee`, `returngraph.jeek` for details.

Details

the function aims to provide users the flexibility to explore and visualize the graph on their own generated from jointnet

Value

an igraph object of graph / subgraph from jointnet result specified by user input

Author(s)

Beilun Wang, Zhaoyang Wang (Author), Zhaoyang Wang (maintainer) <zw4dn@virginia.edu>

Examples

```
library(JointNets)
data(exampleData)
result = jeek(X = exampleData, 0.3, covType = "cov", parallel = FALSE)
graph = returngraph(result)
```

`returngraph.diffee` *return igraph object from diffee result specified by user input*

Description

This function can return an igraph object from diffee result for user to work with directly

Usage

```
## S3 method for class 'diffee'
returngraph(x, type = "task",
  neighbouroption = "task", subID = NULL, index = NULL, ...)
```

Arguments

`x`	output generated from diffee function (diffee class)
`type`	type of graph. There are two options: • "task" (difference graph) • "neighbour" (zoom into nodes in the difference graph specified further by parameter **"index"** (node id)
`neighbouroption`	unused
`subID`	unused
`index`	determines which node(s) to zoom into when parameter **"type"** is **"neighbour"**. This parameter could either be an integer or vector of integers representing node ids (zoom into one node or multiple nodes)
`...`	unused

Details

the function aims to provide users the flexibility to explore and visualize the graph own their own generated from diffee

Value

an igraph object of graph / subgraph from diffee result specified by user input

Author(s)

Beilun Wang, Zhaoyang Wang (Author), Zhaoyang Wang (maintainer) <zw4dn@virginia.edu>

Examples

```
library(JointNets)
data(exampleData)
result = diffee(exampleData[[1]], exampleData[[2]], 0.45)
graph = returngraph(result)
```

`returngraph.jeek` *return igraph object from jeek result specified by user input*

Description

This function can return an igraph object from jeek result for user to work with directly

Usage

```
## S3 method for class 'jeek'
returngraph(x, type = "task", neighbouroption = "task",
  subID = NULL, index = NULL, ...)
```

Arguments

`x`	output generated from jeek function (jeek class)
`type`	type of graph. There are four options: • "task" (graph for each task (including shared part) specified further by subID (task number)) • "share" (shared graph for all tasks) • "taskspecific" (graph for each task specific graph (excluding shared part) specified further by subID (task number)) • "neighbour" (zoom into nodes in the graph specified further by neighbouroptoin, subID (task number) and index (node id))
`neighbouroption`	determines what type of graph to zoom into when parameter **"type"** is **"neighbour"**. There are two options: • "task" (zoom into graph for each task (including shared part)) • "taskspecific" (zoom into graph for each task specific (excluding shared part))
`subID`	selects which task to display. There are four options: • 0 (only allowed when **"type"** is **"task"** or **"type"** is **"neighbour"** and **"neighbouroption"** is **"task"**) (selects share graph) • positive task number (selects that particular task) • a vector of task number (selects multiple tasks) • NULL (selects all tasks (all graphs))
`index`	determines which node(s) to zoom into when parameter **"type"** is **"neighbour"**. This parameter could either be an integer or vector of integers representing node ids (zoom into one node or multiple nodes)
`...`	not used

Details

the function aims to provide users the flexibility to explore and visualize the graph on their own generated from jeek

Value

an igraph object of graph / subgraph from jeek result specified by user input

Author(s)

Beilun Wang, Zhaoyang Wang (Author), Zhaoyang Wang (maintainer) <zw4dn@virginia.edu>

Examples

```
library(JointNets)
data(exampleData)
result = jeek(X = exampleData, 0.3, covType = "cov", parallel = FALSE)
graph = returngraph(result)
```

---

returngraph.jgl *return igraph object from jgl result specified by user input*

Description

This function can return an igraph object from jgl result for user to work with directly

Usage

```
## S3 method for class 'jgl'
returngraph(x, type = "task", neighbouroption = "task",
  subID = NULL, index = NULL, ...)
```

Arguments

x	output generated from jgl function (jgl class)
type	type of graph. There are four options: • "task" (graph for each task (including shared part) specified further by subID (task number)) • "share" (shared graph for all tasks) • "taskspecific" (graph for each task specific graph (excluding shared part) specified further by subID (task number)) • "neighbour" (zoom into nodes in the graph specified further by neighbouroptoin, subID (task number) and index (node id))
neighbouroption	determines what type of graph to zoom into when parameter **"type"** is **"neighbour"**. There are two options: • "task" (zoom into graph for each task (including shared part)) • "taskspecific" (zoom into graph for each task specific (excluding shared part))
subID	selects which task to display. There are four options:

- 0 (only allowed when **"type"** is **"task"** or **"type"** is **"neighbour"** and **"neighbouroption"** is **"task"**) (selects share graph)
- positive task number (selects that particular task)
- a vector of task number (selects multiple tasks)
- NULL (selects all tasks (all graphs))

`index` determines which node(s) to zoom into when parameter **"type"** is **"neighbour"**. This parameter could either be an integer or vector of integers representing node ids (zoom into one node or multiple nodes)

`...` not used

Details

the function aims to provide users the flexibility to explore and visualize the graph on their own generated from jgl

Value

an igraph object of graph / subgraph from jgl result specified by user input

Author(s)

Beilun Wang, Zhaoyang Wang (Author), Zhaoyang Wang (maintainer) <zw4dn@virginia.edu>

Examples

```
library(JointNets)
data(exampleData)
result = jgl(X = exampleData , lambda1 = 1, lambda2 = 1)
graph = returngraph(result)
```

---

returngraph.kdiffnet

return igraph object from kdiffnet result specified by user input

---

Description

This function can return an igraph object from kdiffnet result for user to work with directly

Usage

```
## S3 method for class 'kdiffnet'
returngraph(x, type = "task",
  neighbouroption = "task", subID = NULL, index = NULL, ...)
```

Arguments

x	output generated from kdiffnet function (kdiffnet class)
type	type of graph. There are two options: • "task" (difference graph) • "neighbour" (zoom into nodes in the difference graph specified further by parameter **"index"** (node id)
neighbouroption	unused
subID	unused
index	determines which node(s) to zoom into when parameter **"type"** is **"neighbour"**. This parameter could either be an integer or vector of integers representing node ids (zoom into one node or multiple nodes)
...	unused

Details

the function aims to provide users the flexibility to explore and visualize the graph own their own generated from kdiffnet

Value

an igraph object of graph / subgraph from kdiffnet result specified by user input

Author(s)

Beilun Wang, Zhaoyang Wang (Author), Zhaoyang Wang (maintainer) <zw4dn@virginia.edu>

Examples

```
library(JointNets)
data(exampleData)
result = kdiffnet(exampleData[[1]], exampleData[[2]],
W = matrix(1,20,20), g = rep(0,20),epsilon = 0.2,
lambda = 0.4,covType = "cov")
graph = returngraph(result)
```

---

returngraph.simulation

return igraph object from simulation result specified by user input

---

Description

This function can return an igraph object from simulation result for user to work with directly

Usage

```
## S3 method for class 'simulation'
returngraph(x, type = "task",
  neighbouroption = "task", subID = NULL, index = NULL, ...)
```

Arguments

x	output generated from simulatino functino
type	type of graph. There are four options: • "task" (graph for each task (including shared part) specified further by subID (task number)) • "share" (shared graph for all tasks) • "taskspecific" (graph for each task specific graph (excluding shared part) specified further by subID (task number)) • "neighbour" (zoom into nodes in the graph specified further by neighbouroptoin, subID (task number) and index (node id))
neighbouroption	determines what type of graph to zoom into when parameter **"type"** is **"neighbour"**. There are two options: • "task" (zoom into graph for each task (including shared part)) • "taskspecific" (zoom into graph for each task specific (excluding shared part))
subID	selects which task to display. There are four options: • 0 (only allowed when **"type"** is **"task"** or **"type"** is **"neighbour"** and **"neighbouroption"** is **"task"**) (selects share graph) • positive task number (selects that particular task) • a vector of task number (selects multiple tasks) • NULL (selects all tasks (all graphs))
index	determines which node(s) to zoom into when parameter **"type"** is **"neighbour"**. This parameter could either be an integer or vector of integers representing node ids (zoom into one node or multiple nodes)
...	not used

Details

the function aims to provide users the flexibility to explore and visualize the graph on their own generated from simulation

Value

an igraph object of graph / subgraph from simulation result specified by user input

Author(s)

Beilun Wang, Zhaoyang Wang (Author), Zhaoyang Wang (maintainer) <zw4dn@virginia.edu>

Examples

```
library(JointNets)
data(exampleData)
result = simulation(n=c(100,100,100))$simulatedgraphs
graph = returngraph(result)
```

`returngraph.simule` *return igraph object from simule result specified by user input*

Description

This function can return an igraph object from simule result for user to work with directly

Usage

```
## S3 method for class 'simule'
returngraph(x, type = "task",
  neighbouroption = "task", subID = NULL, index = NULL, ...)
```

Arguments

`x`	output generated from simule function (simule class)
`type`	type of graph. There are four options: • "task" (graph for each task (including shared part) specified further by subID (task number)) • "share" (shared graph for all tasks) • "taskspecific" (graph for each task specific graph (excluding shared part) specified further by subID (task number)) • "neighbour" (zoom into nodes in the graph specified further by neighbouroptoin, subID (task number) and index (node id))
`neighbouroption`	determines what type of graph to zoom into when parameter **"type"** is **"neighbour"**. There are two options: • "task" (zoom into graph for each task (including shared part)) • "taskspecific" (zoom into graph for each task specific (excluding shared part))
`subID`	selects which task to display. There are four options: • 0 (only allowed when **"type"** is **"task"** or **"type"** is **"neighbour"** and **"neighbouroption"** is **"task"**) (selects share graph) • positive task number (selects that particular task) • a vector of task number (selects multiple tasks) • NULL (selects all tasks (all graphs))
`index`	determines which node(s) to zoom into when parameter **"type"** is **"neighbour"**. This parameter could either be an integer or vector of integers representing node ids (zoom into one node or multiple nodes)
`...`	not used

Details

the function aims to provide users the flexibility to explore and visualize the graph on their own generated from simule

Value

an igraph object of graph / subgraph from simule result specified by user input

Author(s)

Beilun Wang, Zhaoyang Wang (Author), Zhaoyang Wang (maintainer) <zw4dn@virginia.edu>

Examples

```
library(JointNets)
data(exampleData)
result = simule(X = exampleData , lambda = 0.1, epsilon = 0.45, covType = "cov", FALSE)
graph = returngraph(result)
```

---

returngraph.wsimule

return igraph object from wsimule result specified by user input

---

Description

This function can return an igraph object from wsimule result for user to work with directly

Usage

```
## S3 method for class 'wsimule'
returngraph(x, type = "task",
  neighbouroption = "task", subID = NULL, index = NULL, ...)
```

Arguments

x	output generated from wsimule function (wsimule class)
type	type of graph. There are four options: • "task" (graph for each task (including shared part) specified further by subID (task number)) • "share" (shared graph for all tasks) • "taskspecific" (graph for each task specific graph (excluding shared part) specified further by subID (task number)) • "neighbour" (zoom into nodes in the graph specified further by neighbouroptoin, subID (task number) and index (node id))
neighbouroption	determines what type of graph to zoom into when parameter **"type"** is **"neighbour"**. There are two options:

- "task" (zoom into graph for each task (including shared part))
- "taskspecific" (zoom into graph for each task specific (excluding shared part))

`subID` selects which task to display. There are four options:

- 0 (only allowed when **"type"** is **"task"** or **"type"** is **"neighbour"** and **"neighbouroption"** is **"task"**) (selects share graph)
- positive task number (selects that particular task)
- a vector of task number (selects multiple tasks)
- NULL (selects all tasks (all graphs))

`index` determines which node(s) to zoom into when parameter **"type"** is **"neighbour"**. This parameter could either be an integer or vector of integers representing node ids (zoom into one node or multiple nodes)

`...` not used

Details

the function aims to provide users the flexibility to explore and visualize the graph on their own generated from wsimule

Value

an igraph object of graph / subgraph from wsimule result specified by user input

Author(s)

Beilun Wang, Zhaoyang Wang (Author), Zhaoyang Wang (maintainer) <zw4dn@virginia.edu>

Examples

```
library(JointNets)
data(exampleData)
result = wsimule(X = exampleData , lambda = 0.1, epsilon = 0.45,
W = matrix(1,20,20), covType = "cov", FALSE)
graph = returngraph(result)
```

---

simulateGraph *function to simulate multiple sparse graphs*

Description

function to simulate multiple sparse graphs

Usage

```
simulateGraph(p = 20, N = 2, seedNum = 37, s = 0.1, ss = 0.1)
```

Arguments

p	number of features
N	number of tasks
seedNum	seed number for random simulation
s	controls sparsity of the generated graph
ss	controls sparsity of the generated graph

Value

a list of N related sparse pXp precision matrices (graphs)

---

simulation *simulate multiple sparse graphs and generate samples*

Description

simulate multiple sparse graphs and generate samples

Usage

```
simulation(p = 20, n, seedNum = 37, s = 0.1, ss = 0.1)
```

Arguments

p	number of features (number of nodes)
n	a vector indicating number of samples and tasks, for example c(100,200,300) for 3 tasks and 100,200 and 300 samples for task 1, 2 and 3
seedNum	seed number for random simulation
s	positive number that controls sparsity of the generated graphs
ss	positive number that controls sparsity of the shared part of generated graphs

Value

a list comprising $simulatedgraphs (multiple related simulated graphs) and $simulatedsamples (samples generated from multiple related graphs)

Examples

```
library(JointNets)
simulateresult = simulation(p = 20,  n = c(100,100))
plot(simulateresult$simulatedgraphs)
```

simule *A constrained l1 minimization approach for estimating multiple Sparse Gaussian or Nonparanormal Graphical Models Estimate multiple, related sparse Gaussian or Nonparanormal graphical*

Description

models from multiple related datasets using the SIMULE algorithm. Please run demo(simule) to learn the basic functions provided by this package. For further details, please read the original paper: Beilun Wang, Ritambhara Singh, Yanjun Qi (2017) doi: 10.1007/s1099401756357[1].

Usage

```
simule(X, lambda, epsilon = 1, covType = "cov", intertwined = FALSE,
  parallel = FALSE)
```

Arguments

X	A List of input matrices. They can be data matrices or covariance/correlation matrices. If every matrix in the X is a symmetric matrix, the matrices are assumed to be covariance/correlation matrices.
lambda	A positive number. The hyperparameter controls the sparsity level of the matrices. The λ_n in the following section: Details.
epsilon	A positive number. The hyperparameter controls the differences between the shared pattern among graphs and the individual part of each graph. The ϵ in the following section: Details. If epsilon becomes larger, the generated graphs will be more similar to each other. The default value is 1, which means that we set the same weights to the shared pattern among graphs and the individual part of each graph.
covType	A parameter to decide which Graphical model we choose to estimate from the input data. If covType = "cov", it means that we estimate multiple sparse Gaussian Graphical models. This option assumes that we calculate (when input X represents data directly) or use (when X elements are symmetric representing covariance matrices) the sample covariance matrices as input to the simule algorithm. If covType = "kendall", it means that we estimate multiple nonparanormal Graphical models. This option assumes that we calculate (when input X represents data directly) or use (when X elements are symmetric representing correlation matrices) the kendall's tau correlation matrices as input to the simule algorithm.
intertwined	indicate whether to use intertwined covariance matrix
parallel	A boolean. This parameter decides if the package will use the multithreading architecture or not.

[1] http://doi.org/10.1007/s10994-017-5635-7

Details

The SIMULE algorithm is a constrained l1 minimization method that can detect both the shared and the task-specific parts of multiple graphs explicitly from data (through jointly estimating multiple sparse Gaussian graphical models or Nonparanormal graphical models). It solves the following equation:

$$\hat{\Omega}_I^{(1)}, \hat{\Omega}_I^{(2)}, \ldots, \hat{\Omega}_I^{(K)}, \hat{\Omega}_S = \min_{\Omega_I^{(i)}, \Omega_S} \sum_i ||\Omega_I^{(i)}||_1 + \epsilon K ||\Omega_S||_1$$

Subject to :

$$||\Sigma^{(i)}(\Omega_I^{(i)} + \Omega_S) - I||_\infty \leq \lambda_n, i = 1, \ldots, K$$

Please also see the equation (7) in our paper. The λ_n is the hyperparameter controlling the sparsity level of the matrices and it is the `lambda` in our function. The ϵ is the hyperparameter controlling the differences between the shared pattern among graphs and the individual part of each graph. It is the `epsilon` parameter in our function and the default value is 1. For further details, please see our paper: `http://link.springer.com/article/10.1007/s10994-017-5635-7`.

if labels are provided in the datalist as column names, result will contain labels (to be plotted)

Value

`$graphs`	A list of the estimated inverse covariance/correlation matrices.
`$share`	The shared graph among multiple tasks.

Author(s)

Beilun Wang

References

Beilun Wang, Ritambhara Singh, Yanjun Qi (2017). A constrained L1 minimization approach for estimating multiple Sparse Gaussian or Nonparanormal Graphical Models. http://link.springer.com/article/10.1007/s10994-017-5635-7

Examples

```
library(JointNets)
data(exampleData)
result = simule(X = exampleData , lambda = 0.1, epsilon = 0.45, covType = "cov", FALSE)
plot(result)
```

train_valid_test_split

split a datalist to train,validation and test

Description

split a datalist to train,validation and test

Usage

```
train_valid_test_split(datalist, ratio, seed)
```

Arguments

datalist	a datalist
ratio	ratio of the split (train, validation and test), eg, c(0.8,0.1,0.1)
seed	seed number

Value

a list of train,validation and test datalist

Examples

```
library(JointNets)
data("nip_37_data")
```

wsimule

A constrained and weighted l1 minimization approach for estimating multiple Sparse Gaussian or Nonparanormal Graphical Models

Description

Estimate multiple, related sparse Gaussian or Nonparanormal graphical models from multiple related datasets using the SIMULE algorithm. Please run demo(wsimule) to learn the basic functions provided by this package. For further details, please read the original paper: Beilun Wang, Ritambhara Singh, Yanjun Qi (2017) doi10.1007/s10994-017-5635-7.

Usage

```
wsimule(X, lambda, epsilon = 1, W, covType = "cov",
  intertwined = FALSE, parallel = FALSE)
```

Arguments

X	A List of input matrices. They can be data matrices or covariance/correlation matrices. If every matrix in the X is a symmetric matrix, the matrices are assumed to be covariance/correlation matrices. More details at https://github.com/QData/SIMULE
lambda	A positive number. The hyperparameter controls the sparsity level of the matrices. The λ_n in the following section: Details.
epsilon	A positive number. The hyperparameter controls the differences between the shared pattern among graphs and the individual part of each graph. The ϵ in the following section: Details. If epsilon becomes larger, the generated graphs will be more similar to each other. The default value is 1, which means that we set the same weights to the shared pattern among graphs and the individual part of each graph.
W	A weight matrix. This matrix uses the prior knowledge of the graphs. For example, if we use wsimule to infer multiple human brain connectome graphs, the W can be the anatomical distance matrix of human brain. The default value is a matrix, whose entries all equals to 1. This means that we do not have any prior knowledge.
covType	A parameter to decide which Graphical model we choose to estimate from the input data. If covType = "cov", it means that we estimate multiple sparse Gaussian Graphical models. This option assumes that we calculate (when input X represents data directly) or use (when X elements are symmetric representing covariance matrices) the sample covariance matrices as input to the simule algorithm. If covType = "kendall", it means that we estimate multiple nonparanormal Graphical models. This option assumes that we calculate (when input X represents data directly) or use (when X elements are symmetric representing correlation matrices) the kendall's tau correlation matrices as input to the simule algorithm.
intertwined	indicate whether to use intertwined covariance matrix
parallel	A boolean. This parameter decides if the package will use the multithreading architecture or not.

Details

The SIMULE algorithm is a constrained l1 minimization method that can detect both the shared and the task-specific parts of multiple graphs explicitly from data (through jointly estimating multiple sparse Gaussian graphical models or Nonparanormal graphical models). It solves the following equation:

$$\hat{\Omega}_I^{(1)}, \hat{\Omega}_I^{(2)}, \ldots, \hat{\Omega}_I^{(K)}, \hat{\Omega}_S = \min_{\Omega_I^{(i)}, \Omega_S} \sum_i ||W \cdot \Omega_I^{(i)}||_1 + \epsilon K ||W \cdot \Omega_S||_1$$

Subject to :

$$||\Sigma^{(i)}(\Omega_I^{(i)} + \Omega_S) - I||_\infty \leq \lambda_n, i = 1, \ldots, K$$

Please also see the equation (7) in our paper. The λ_n is the hyperparameter controlling the sparsity level of the matrices and it is the lambda in our function. The ϵ is the hyperparameter controlling

the differences between the shared pattern among graphs and the individual part of each graph. It is the `epsilon` parameter in our function and the default value is 1. For further details, please see our paper: `http://link.springer.com/article/10.1007/s10994-017-5635-7`.

if labels are provided in the datalist as column names, result will contain labels (to be plotted)

Value

`$graphs`	A list of the estimated inverse covariance/correlation matrices.
`$share`	The share graph among multiple tasks.

Author(s)

Beilun Wang

References

Beilun Wang, Ritambhara Singh, Yanjun Qi (2017). A constrained L1 minimization approach for estimating multiple Sparse Gaussian or Nonparanormal Graphical Models. http://link.springer.com/article/10.1007/s10994-017-5635-7

Examples

```
library(JointNets)
data(exampleData)
result = wsimule(X = exampleData , lambda = 0.1, epsilon = 0.45,
W = matrix(1,20,20), covType = "cov", FALSE)
plot(result)
```