Package 'COSINE'

February 19, 2015

Type Package

Title COndition SpecIfic sub-NEtwork

Version 2.1

Date 2014-07-09

Author Haisu Ma

Maintainer Haisu Ma <haisu.ma.pku.2008@gmail.com>

Depends R (>= 3.1.0), MASS,genalg

Description To identify the globally most discriminative subnetwork from gene expression profiles using an optimization model and genetic algorithm

License GPL (>= 2)

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2014-07-10 07:34:03

R topics documented:

`COSINE-package` *COndition SpecIfic subNEtwork identification*

Description

This is a package to identify the single globally optimal subnetwork which differs the most between two or more datasets

Details

Package:	COSINE
Type:	Package
Version:	2.1
Date:	2014-07-09
License:	GPL (version 2 or later)
LazyLoad:	yes

install.packages("COSINE")

Author(s)

Haisu Ma Maintainer: Haisu Ma <haisu.ma.pku.2008@gmail.com>

`choose_lambda` *Choose the most appropriate weight parameter lambda*

Description

Randomly sample a large number of subnetworks with the same size as the ones chosen by the five different lambda values to get the null distribution of the scores of subnetworks corresponding to different size and lambda, in order to get the adjusted scores for the chosen subnetworks, and choose the lambda giving rise to the highest scored sub-network

Usage

```
choose_lambda(diff_expr, diff_coex, lambda, subnet_size,
num_random_sampling, best_score)
```

Arguments

diff_expr	A vector storing the F-statistics measuring the differential expression of each gene, which length equals the number of genes N
diff_coex	An N by N matrix with entry (i,j) corresponding to the ECF-statistics of gene pair (i,j), which measures the differential correlation between genes i and j
lambda	A numeric vector of length 5 which stores the five quantiles of weight parameter lambda
subnet_size	A numeric vector of length 5 which stores the size of subnetworks selected using different weight parameter lambda
num_random_sampling	the number of random subnetworks to be sampled for each lambda in order to get the null distribution
best_score	the best scores of the five sub-networks selected using genetic algorithm

Value

A list containing:

Adj_score	The adjusted scores of the five selected sub-networks according to the null distribution generated by random sampling
best_lambda	The lambda giving rise to the sub-network with the highest adjusted score
Random_Score	The matrix storing the null score distribution of random subnetworks

Author(s)

Haisu Ma

Examples

```
data(set1_scaled_diff)
data(set1_GA)

set1_quantile<-get_quantiles(diff_expr=set1_scaled_diff[[1]],
diff_coex=set1_scaled_diff[[2]],klist=c(20,25),pop_size=5)

lambda<-set1_quantile[[2]]

set1_choose_lambda <- choose_lambda(diff_expr=set1_scaled_diff[[1]],
diff_coex=set1_scaled_diff[[2]],lambda,subnet_size=set1_GA$Subnet_size,
num_random_sampling=2,best_score=set1_GA$Best_Scores)
```

cond.fyx *Compute the ECF-statistics measuring the differential correlation of gene pairs*

Description

A function to calculate the expected conditional F-statistics as a measure of differential gene co-expression patterns.

Usage

```
cond.fyx(data.y, data.x, type)
```

Arguments

data.y	A vector containing the expression values of one gene across two datasets
data.x	A vector containing the expression values of another gene across two datasets
type	A vector indicating the type of each sample, whose length is the sum of the sample sizes of data.y and data.x

Value

The ECF-statistics of a specific gene

Author(s)

Yinglei Lai

References

http://bioinformatics.oxfordjournals.org/content/20/17/3146.long

Examples

```
#load two of the simulated datasets

data(simulated_data)
set1_data<-simulated_data[[1]]
control_data<-simulated_data[[7]]

num_sample <- dim(set1_data)[1]
num_gene <- dim(set1_data)[2]

type <- c(rep(0,num_sample),rep(1,num_sample))

#Compute the ECF-statistic for the gene pair of gene 1 and gene 2
```

```
i=1
j=2
data.x <- c(set1_data[,i],control_data[,i])
data.y <- c(set1_data[,j],control_data[,j])

ecf <- ( cond.fyx(data.y,data.x,type) +
        cond.fyx(data.x,data.y,type) )/2
```

DataSimu *Simulation of the six datasets and the case dataset*

Description

This function simulates six datasets with various differential expression and differential correlation patterns.

Usage

```
DataSimu()
```

Value

A list containing:

`set1.data,set2.data,...,set6.data`
: case datasets with different expression patterns to be compared with the control datasets

`control.data`
: the control dataset

Author(s)

Haisu Ma

Examples

```
set.seed(666)
simulated.data<-DataSimu()
```

diff_gen *Calculate the F-statistics and ECF-statistics*

Description

The "diff_gen" function calculates the F-statistics which measures the differential expression of each gene and the ECF-statistcs which measures the differential correlation of each gene pair between two datasets

Usage

```
diff_gen(data1, data2)
```

Arguments

data1	one of the two gene expression datasets
data2	a second gene expression dataset

Value

A list containing:

diff_expr	a vector of the F-statistics for each gene
diff_coex	a square matrix storing the ECF-statistics for each gene pair

Author(s)

Haisu Ma

Examples

```
#Load two of the simualted datasets

data(simulated_data)

set1_data <- simulated_data[[1]]
control_data <- simulated_data[[7]]

#Compute the F-statistics and ECF-statistics for the first 10 genes

diff_gen_test <- diff_gen(set1_data[,1:10],control_data[,1:10])
```

`diff_gen_for3` *Generate the F-statistics and ECF-statistics for the comparison of three datasets*

Description

It works very similarly to "diff_gen" except that it performs the calculation across three rather than two datasets

Usage

```
diff_gen_for3(data1, data2, data3)
```

Arguments

`data1`	The first dataset of gene expression profiles
`data2`	The second dataset of gene expression profiles
`data3`	The third dataset of gene expression profiles

Value

A list containing:

`diff_expr`	a vector of the F-statistics for each gene
`diff_coex`	a square matrix storing the ECF-statistics for each gene pair

Author(s)

Haisu Ma

Examples

```
#Load the simulated datasets

data(simulated_data)

set1_data<-simulated_data[[1]]

set2_data<-simulated_data[[2]]

control_data<-simulated_data[[7]]

#Calculate the F-statistics and ECF-statistics
#for the first five genes

diff_gen_for3_test <- diff_gen_for3(set1_data[,1:5],
set2_data[,1:5],control_data[,1:5])
```

`diff_gen_PPI` *Generate the scaled node score and scaled edge score for nodes and edges in the background network*

Description

Compute the F-statistic and ECF-statistic and then standardize them

Usage

```
diff_gen_PPI(data1, data2, PPI)
```

Arguments

`data1`	The first gene expression dataset(with rows corresponding to samples)
`data2`	The second gene expression dataset
`PPI`	A matrix with two columns containing the protein interaction pairs

Value

A list containing:

`scaled_node_score`
: The standardized F-statistic measuring the differential expression of each gene

`scaled_edge_score`
: The standardized ECF-statistic measuring the differential correlation of each gene pair

Author(s)

Haisu Ma

Examples

```
data(simulated_data)
data(PPI)
data1 <- simulated_data[[1]]
data2 <- simulated_data[[7]]
colnames(data1)<-colnames(data2)<-as.character(1:500)
test <- diff_gen_PPI(data1[,1:20],data2[,1:20],PPI)
```

`f.test` *To get the F-statistics for each gene*

Description

Calculate the F-statistics measuring the differential expression of each gene

Usage

```
f.test(data, type)
```

Arguments

`data`	A vector containing the expression values of a gene across two datasets
`type`	A vector indicating the type of each sample (coming from dataset 1 or dataset 2)

Value

The F-statistics of a specific gene

Author(s)

Haisu Ma

Examples

```
#Load the simulated datasets

data(simulated_data)
data1 <- simulated_data[[1]]
data2 <- simulated_data[[7]]

#Calculate the F-statistics for genes 1~10

num_sample <- dim(data1)[1]
diff_expr <- rep(0,10)
type <- c(rep(0,num_sample),rep(1,num_sample))

for(i in 1:10){
       data <- c(data1[,i],data2[,i])
       diff_expr[i] <- f.test(data,type)
    }
```

GA_search *Use genetic algorithm to search for the globally optimal subnetwork*

Description

This function performs the stochastic search using genetic algorithm to find the globally optimal subnetwork which gives rise to the highest score defined by a scoring function, which measures the extent of the differential expression of the subnetwork across several datasets.

Usage

```
GA_search(lambda, diff_expr, diff_coex, num_iter = 1000,
muCh = 0.05, zToR = 10)
```

Arguments

lambda	A vector containing the five quantiles of the weight parameter lambda
diff_expr	A vector storing the F-statistics measuring the differential expression of each gene, which length equals the number of genes N
diff_coex	An N by N matrix with entry (i,j) corresponding to the ECF-statistics of gene pair (i,j), which measures the differential correlation between genes i and j
num_iter	The number of iterations to be performed by the genetic algorithm
muCh	the mutation chance used by genetic algorithm
zToR	zero to one ratio

Value

A list containing the following components:

Subnet_size	A vector containing the size of the subnetwork identified using each lambda
Best_Scores	A vector containing the best scores of the subnetworks
Subnet	A list containing the extracted subnetworks (a list of genes) for each of the five lambda values
GA_obj	A list of the returned objects of the genetic algorithm function

Author(s)

Haisu Ma

References

http://cran.r-project.org/web/packages/genalg/index.html

Examples

```
# Load the scaled F-statistics and ECF-statistics
# for the simulated datasets

data(set1_scaled_diff)

# Get the quantiles of lambda

klist<-c(25,30)
set1_quantile<-get_quantiles(diff_expr=set1_scaled_diff[[1]],
diff_coex=set1_scaled_diff[[2]],klist,pop_size=10)
lambda<-set1_quantile[[2]]

#Perform genetic algorithm to search-just show the first iteration here

set1_GA<-GA_search(lambda[1:2],diff_expr=set1_scaled_diff[[1]],
diff_coex=set1_scaled_diff[[2]], num_iter=1, muCh=0.05, zToR=50)
```

GA_search_PPI *Run genetic algorithm to search for the PPI sub-network*

Description

This function performs the stochastic search using genetic algorithm to find the globally optimal subnetwork which gives rise to the highest score defined by a scoring function, which measures the extent of the differential expression of the PPI subnetwork across several datasets.

Usage

```
GA_search_PPI(lambda, scaled_node_score, scaled_edge_score, PPI,
num_iter = 1000, muCh = 0.05, zToR = 10, minsize = 10)
```

Arguments

lambda	One of the five quantiles of the weight parameter lambda
scaled_node_score	A vector storing the F-statistics measuring the differential expression of each gene, which length equals the number of genes N
scaled_edge_score	A vector storing the ECF-statistics measuring the differential correlation of each gene pair
PPI	A two-column matrix containing the protein interaction pairs
num_iter	The number of iterations to be performed by the genetic algorithm
muCh	the mutation chance used by genetic algorithm
zToR	zero to one ratio
minsize	The minimal size of selected sub-network

Value

A list containing the following components:

Subnet_size	A vector containing the size of the subnetwork identified using each lambda
Best_Scores	A vector containing the best scores of the subnetworks
Subnet	A list containing the extracted subnetworks (a list of genes) for each of the five lambda values
GA_obj	A list of the returned objects of the genetic algorithm function

Author(s)

Haisu Ma

Examples

```
data(scaled_node_score)
data(scaled_edge_score)
data(PPI)

GA_result<-GA_search_PPI(lambda=0.5,scaled_node_score,scaled_edge_score,PPI,
num_iter=1, muCh=0.05, zToR=10, minsize=50)
```

---

get_components_PPI *Get all the components (connected clusters) of the sub-network*

Description

Map the edges in the selected sub-network to the background PPI network and get all the clusters with size larger than the minimumset by the user

Usage

```
get_components_PPI(gene_names, vector, PPI, minsize)
```

Arguments

gene_names	The gene names of all the nodes
vector	A binary vector indicating whether each node is selected or not
PPI	A two column matrix including the protein interaction data
minsize	The minimal size of clusters

Value

A list with each element corresponding to one cluster in the selected sub-network

Author(s)

Haisu Ma

Examples

```
data(scaled_node_score)
data(scaled_edge_score)
data(PPI)
gene_names<-names(scaled_node_score)
vector<-rep(0,length(scaled_node_score))
vector[sample(1:length(scaled_node_score),length(scaled_node_score)/3)]<-1
components<-get_components_PPI(gene_names,vector,PPI,minsize=3)
```

---

get_quantiles *Get the five quantiles of the weight parameter lambda*

Description

Use random sampling to get a large number of subnetworks and then calculate the distribution of the ratio between the edge-score term and node-score term

Usage

```
get_quantiles(diff_expr, diff_coex, klist, pop_size)
```

Arguments

diff_expr	The vector storing the F-statistics measuring the differential expression of each gene
diff_coex	The matrix storing the ECF-statistics measuring the differential correlation of each gene pair
klist	A vector of the sizes (number of genes) of random subnetworks to be sampled
pop_size	The number of random subnetworks to be sampled

Value

A list containing two components:

ratio	The five quantiles of the log-ratios between the edge-score term and node-score term
lambda	The five quantiles of lambdas calculated based on the ratios

Author(s)

Haisu Ma

Examples

```
data(set1_scaled_diff)

klist<-c(20,25)

set1_quantile<-get_quantiles(diff_expr=set1_scaled_diff[[1]],
diff_coex=set1_scaled_diff[[2]],klist,pop_size=20)
```

`get_quantiles_PPI` *Get the five quantile values of lambda for analysis of gene expression and PPI network data*

Description

Perform random sampling a large number of times to get the distribution of node score term and edge score term and calculate the quantiles of lambda

Usage

```
get_quantiles_PPI(scaled_node_score, scaled_edge_score, PPI, klist, pop_size)
```

Arguments

`scaled_node_score`	The scaled F-statistics measuring the differential expression of each gene
`scaled_edge_score`	The scaled ECF-statistics measuring the differential correlation of each gene pair
`PPI`	The two column matrix containing the protein interaction pairs
`klist`	A list of size of random networks to be sampled
`pop_size`	The number of random networks to be sampled

Value

A list containing two components:

`ratio`	The quantiles of the ratio of the edge score term and node score term
`lambda`	The quantiles of lambda

Author(s)

Haisu Ma

Examples

```
data(scaled_node_score)
data(scaled_edge_score)
data(PPI)
quantiles<-get_quantiles_PPI(scaled_node_score,scaled_edge_score,
PPI,klist=seq(50,60,by=5),pop_size=10)
```

PPI *The protein protein interaction network data*

Description

A two column matrix containing with each row containing the names of the interacting gene pairs

Usage

```
data(PPI)
```

Format

The format is: int [1:1000, 1:2] 184 270 85 386 11 302 4 42 173 233 ...

Examples

```
data(PPI)
```

---

random_network_sampling_PPI *To sample random sub-network from the PPI data*

Description

Randomly sample a set of nodes from the gene pool, and check the number of edges contained, if there are edges among the nodes, return the random sub-network

Usage

```
random_network_sampling_PPI(size, PPI, all_genes)
```

Arguments

size	number of nodes to be sampled
PPI	The PPI network data
all_genes	The names of all genes

Value

current The names of selected genes

Author(s)

Haisu Ma

Examples

```
data(PPI)
all_genes<-union(PPI[,1],PPI[,2])
ran_net<-random_network_sampling_PPI(size=30,PPI,all_genes)
```

---

scaled_edge_score *The scaled ECF statistics of all the edges*

Description

The scaled ECF statistics measuring the differential correlation of all the edges

Usage

```
data(scaled_edge_score)
```

Format

The format is: num [1:1000] 3.1315 -0.6512 -0.8184 -0.596 -0.0935 ...

Examples

```
data(scaled_edge_score)
```

---

scaled_node_score *The scaled ECF-statistics of all the edges*

Description

The scaled ECF-statistics measuring the differential correlation of all the edges

Usage

```
data(scaled_node_score)
```

Format

The format is: Named num [1:500] -0.416 -0.392 -0.326 -0.335 0.237 ... - attr(*, "names")= chr [1:500] "1" "2" "3" "4" ...

Examples

```
data(scaled_node_score)
```

`Score_adjust_PPI` *To adjust the score of the selected PPI sub-network using random sampling*

Description

Randomly sample a large number of node-sets and edge-sets from the background PPI network to derive the null distribution of the scores for subnetworks with certain size, and to compute the adjusted scores for the selected sub-network

Usage

```
Score_adjust_PPI(scaled_node_score, scaled_edge_score,
PPI, lam, subnet, num_random_sampling, best_score)
```

Arguments

`scaled_node_score`	The scaled F-statistics of each node(gene) in the network
`scaled_edge_score`	The scaled ECF-statistics of each edge in the network
`PPI`	A matrix with two columns containing the interacting gene pairs
`lam`	The weight parameter lambda used for the selection of this sub-network
`subnet`	A vector of the index of selected genes
`num_random_sampling`	Number of random subnetworks to be sampled
`best_score`	The original score of selected sub-network

Value

The adjusted score of the selected sub-network

Author(s)

Haisu Ma

Examples

```
data(scaled_node_score)
data(scaled_edge_score)
data(PPI)
data(set1_GA)
adj_Score<-Score_adjust_PPI(scaled_node_score,scaled_edge_score,
PPI,lam=0.1,subnet=set1_GA$Subnet[[1]],num_random_sampling=2,
best_score=set1_GA$Best_Scores[1])
```

`score_scaling` *To get the normalzied F-statistics and ECF-statistics*

Description

Perform standardization of the node score and edge score

Usage

```
score_scaling(diff_expr, diff_coex)
```

Arguments

`diff_expr`	The vector storing the F-statistics measuring the differential expression of each gene
`diff_coex`	The matrix storing the ECF-statistics measuring the differential correlation of each gene pair

Value

A list containing two components:

`diff_expr`	A vector of the standardized F-statistics
`diff_coex`	A matrix of the standardized ECF-statistics

Author(s)

Haisu Ma

Examples

```
data(set1_unscaled_diff)

scaled_diff_set1 <- score_scaling(diff_expr=set1_unscaled_diff[[1]],
diff_coex=set1_unscaled_diff[[2]])
```

set1_GA *Result of genetic algorithm search for simulated data set1*

Description

This data set contains the result of the subnetwork extraction using genetic algorithm applied to the analysis of the differential expression pattern between simulated dataset1 and the control dataset

Usage

```
data(set1_GA)
```

Format

The format is:

List of 4

$ Subnet_size: num [1:5] 23 58 61 75 99

$ Best_Scores: num [1:5] 16.7 24.7 26.6 29.3 77.7

$ Subnet :List of 5

..$: int [1:23] 36 40 41 112 121 148 163 184 185 206 ... ..$: int [1:58] 25 36 38 39 40 41 61 71 78 79 ... ..$: int [1:61] 25 36 38 39 40 41 48 61 78 79 ... ..$: int [1:75] 8 25 36 38 39 40 41 61 71 78 ... ..$: int [1:99] 3 5 8 11 25 29 36 38 39 40 ...

$ GA_obj :List of 5

..$:List of 11 .. ..$ type : chr "binary chromosome" .. ..$ size : int 500 .. ..$ popSize : num 200 .. ..$ iters : num 1000 .. ..$ suggestions : NULL .. ..$ population : num [1:200, 1:500] 0 0 0 0 0 0 0 0 0 0 ... .$ elitism : num 40 .. ..$ mutationChance: num 0.05 .. ..$ evaluations : num [1:200] -16.7 -16.7 -16.7 -16.7 -16.7 ... .$ best : num [1:1000] -2.85 -2.85 -3.68 -4.49 -4.59 ... .$ mean : num [1:1000] 0.0192 -0.743 -1.4033 -2.1104 -2.7789 ... .- attr(*, "class")= chr "rbga"

..$:List of 11 .. ..$ type : chr "binary chromosome" .. ..$ size : int 500 .. ..$ popSize : num 200 .. ..$ iters : num 1000 .. ..$ suggestions : NULL .. ..$ population : num [1:200, 1:500] 0 0 0 0 0 0 0 0 0 0 ... .$ elitism : num 40 .. ..$ mutationChance: num 0.05 .. ..$ evaluations : num [1:200] -24.7 -24.7 -24.7 -24.7 -24.7 ... .$ best : num [1:1000] -4.27 -5.88 -6.64 -6.73 -7.95 ... .$ mean : num [1:1000] 0.00387 -1.27523 -2.59219 -3.73991 -4.7593 ... .- attr(*, "class")= chr "rbga"

..$:List of 11 .. ..$ type : chr "binary chromosome" .. ..$ size : int 500 .. ..$ popSize : num 200 .. ..$ iters : num 1000 .. ..$ suggestions : NULL .. ..$ population : num [1:200, 1:500] 0 0 0 0 0 0 0 0 0 0 ... .$ elitism : num 40 .. ..$ mutationChance: num 0.05 .. ..$ evaluations : num [1:200] -26.6 -26.6 -26.6 -26.6 -26.6 ... .$ best : num [1:1000] -5.13 -6.63 -6.84 -7.4 -8.47 ... .$ mean : num [1:1000] 0.0412 -1.3408 -2.6099 -3.9691 -5.1945 ... .- attr(*, "class")= chr "rbga"

..$:List of 11 .. ..$ type : chr "binary chromosome" .. ..$ size : int 500 .. ..$ popSize : num 200 .. ..$ iters : num 1000 .. ..$ suggestions : NULL .. ..$ population : num [1:200, 1:500] 0 0 0 0 0 0 0 0 0 0 ... .$ elitism : num 40 .. ..$ mutationChance: num 0.05 .. ..$ evaluations : num [1:200] -29.3 -29.2 -29.2 -29.2 -29.2 ... .$ best : num [1:1000] -4.43 -4.71 -6.21 -6.92 -8.31 ... .$ mean : num [1:1000] -0.126 -1.546 -2.71 -3.8 -4.788 ... .- attr(*, "class")= chr "rbga"

..$:List of 11 .. ..$ type : chr "binary chromosome" .. ..$ size : int 500 .. ..$ popSize : num 200 .. ..$ iters : num 1000 .. ..$ suggestions : NULL .. ..$ population : num [1:200, 1:500] 0 0 0 0 0 0 0 0 0 0 ... .$ elitism : num 40 .. ..$ mutationChance: num 0.05 .. ..$ evaluations : num [1:200] -77.7 -77.6 -77.3 -77.3 -77.3 ... .$ best : num [1:1000] -11.7 -15.2 -15.2 -20.6 -21.4 ... .$ mean : num [1:1000] -0.0381 -2.8258 -5.5397 -8.3003 -10.7343 ... .- attr(*, "class")= chr "rbga"

Details

This dataset is a list containing the following components: Subnet_size: A vector of length 5 showing the size of the selected subnetwork using five different lambdas. Best_Scores: The scores of the selected subnetworks corresponding to five lambdas. Subnet: The selected subnetworks (gene indices) for five lambdas. GA_obj: The objected returned by the function "rbga.bin", which stores the results of the genetic algorithm.

Examples

```
data(set1_GA)
```

---

set1_scaled_diff *The standardized F-statistics and ECF-statistics for the comparison between simulated data1 and the control data*

Description

This data is a list containing two components, the first component is a vector of length 500 which contains the F-statistics measuring differential expression of single genes, and the second component is a 500 by 500 matrix containing the ECF-statistics measuring the differential correlation of gene pairs

Usage

```
data(set1_scaled_diff)
```

Format

The format is:

List of 2

$: num [1:500] -0.416 -0.392 -0.326 -0.335 0.237 ...

$: num [1:500, 1:500] -0.986 -0.352 -0.828 -0.79 -0.208 ...

Examples

```
data(set1_scaled_diff)
```

set1_unscaled_diff *The unstandardized F-statistics and ECF-statistics of simulated dataset 1*

Description

Generated by the function of "diff_gen" which performs comparative analysis of differential gene expression patterns between the simulated case dataset1 and control dataset

Usage

```
data(set1_unscaled_diff)
```

Format

The format is:

List of 2

$: num [1:500] 0.251 0.291 0.398 0.383 1.312 ... $: num [1:500, 1:500] 0 0.03702 0.00924 0.01146 0.04543 ...

Details

It is a list of length 2, which includes a vector of the F-statistics and a matrix of the ECF-statistics.

Examples

```
data(set1_unscaled_diff)
```

simulated_data *The simulated data sets used in the paper*

Description

It contains case data set1 to set6, as well as a common control dataset

Usage

```
data(simulated_data)
```

Format

The format is: List of 7 $: num [1:20, 1:500] 0.00324 0.48968 0.51217 -0.70392 -0.72268 ... ..- attr(*, "dimnames")=List of 2

$: num [1:20, 1:500] 0.3621 -0.00548 1.83257 0.01457 -1.0223 ... ..- attr(*, "dimnames")=List of 2

$: num [1:20, 1:500] 0.3842 1.4062 2.5474 -0.0579 0.0179 ... ..- attr(*, "dimnames")=List of 2

$: num [1:20, 1:500] 0.278 -0.36 0.606 -1.37 0.853 ... ..- attr(*, "dimnames")=List of 2

$: num [1:20, 1:500] 2.53 1.092 1.937 0.822 -1.073 ... ..- attr(*, "dimnames")=List of 2

$: num [1:20, 1:500] 0.3621 -0.00548 1.83257 0.01457 -1.0223 ... ..- attr(*, "dimnames")=List of 2

$: num [1:20, 1:500] -0.754 0.586 0.487 -0.16 -0.975 ... ..- attr(*, "dimnames")=List of 2

Examples

```
data(simulated_data)
```

Index