Package ‘nlnet’

October 20, 2020

Type Package

Title Nonlinear Network, Clustering, and Variable Selection Based on DCOL

Version 1.4

Date 2020-10-16

Author Tianwei Yu, Haodong Liu

Maintainer Tianwei Yu<yutianwei@cuhk.edu.cn>


License GPL (>= 2)

Imports ROCR, TSP, igraph, fdrtool, coin, methods, graphics, stats, earth, randomForest, e1071

NeedsCompilation no

Repository CRAN

Date/Publication 2020-10-19 23:10:05 UTC

R topics documented:

data.gen ......................................................... 2
KPC .............................................................. 3
nlhc ............................................................. 4
nlnet ............................................................ 6
nvsd ............................................................ 8
stage.forward ................................................. 9

Index 11
data.gen

Simulated Data Generation

Description

Generating gene matrix as a example of input.

Usage

data.gen(n.genes=100, n.samples=100, n.grps=10, aver.grp.size=10, n.fun.types=6, epsilon=0.1, n.depend=0)

Arguments

- **n.genes**: the number of rows of the matrix.
- **n.samples**: the number of columns of the matrix.
- **n.grps**: the number of hidden clusters.
- **aver.grp.size**: average number of genes in a cluster.
- **n.fun.types**: number of function types to use.
- **epsilon**: noise level.
- **n.depend**: data generation dependence structure. can be 0, 1, 2.

Details


Value

- **data**: the gene matrix
- **grps**: the predicted clustering

Author(s)

Tianwei Yu<tyu8@emory.edu>

Examples

```r
## generating a gene matrix with 100 genes, some in 5 clusters, and 100 samples per gene.
output<-data.gen(n.genes=100, n.samples=10, n.grps=5)
## get the gene matrix from the source of data.
matrix<-output$data
## get the hidden clusters from the source of data.
grps<-output$grp
```
**KPC**

**implementation of K- Profiles Clustering**

---

**Description**

Implementation of K-Profiles Clustering

**Usage**

KPC(dataset, nCluster, maxIter = 100, p.max = 0.2, p.min = 0.05)

**Arguments**

- **dataset**: the data matrix with genes in the row and samples in the column
- **nCluster**: the number of clusters K
- **maxIter**: the maximum number of iterations
- **p.max**: the starting p-value cutoff to exclude noise genes
- **p.min**: the final p-value cutoff to exclude noise genes

**Value**

Return a list about gene cluster and the list of value p

- **cluster**: gene cluster
- **p.list**: a list of value p

**Author(s)**

Tianwei Yu <tianwei.yu@emory.edu>

**References**

http://www.hindawi.com/journals/bmri/aa/918954/

**See Also**

data.gen

**Examples**

```r
## generating the data matrix & hiden clusters as a sample
input<-data.gen(n.genes=40, n.grps=4)
## now input includes data matrix and hiden clusters, so get the matrix as input.
input<-input$data

## set nCluster value to 4
```
nlhc <- KPC(input, nCluster = 4)

## get the hidden cluster result from "KPC"

cluster <- kpc$cluster

## get the list of p

p <- kpc$p.list

---

### nlhc  

**Non-Linear Hierarchical Clustering**

**Description**

The non-linear hierarchical clustering based on DCOL

**Usage**

```r
nlhc(array, hamil.method = "nn", concorde.path = NA,
     use.normal.approx = FALSE, normalization = "standardize",
     combine.linear = TRUE, use.traditional.hclust = FALSE,
     method.traditional.hclust = "average")
```

**Arguments**

- `array`: the data matrix with no missing values
- `hamil.method`: the method to find the hamiltonian path.
- `concorde.path`: If using the Concorde TSP Solver, the local directory of the solver
- `use.normal.approx`: whether to use the normal approximation for the null hypothesis.
- `normalization`: the normalization method for the array.
- `combine.linear`: whether linear association should be found by correlation to combine with non-linear association found by DCOL.
- `use.traditional.hclust`: whether traditional agglomerative clustering should be used.
- `method.traditional.hclust`: the method to pass on to hclust() if traditional method is chosen.

**Details**

- **Hamil.method**: It is passed onto the function tsp of library TSP. To use linkern method, the user needs to install concord as instructed in TSP.
- **use.normal.approx**: If TRUE, normal approximation is used for every feature, AND all covariances are assumed to be zero. If FALSE, generates permutation based null distribution - mean vector and a variance-covariance matrix.
- **normalization**: There are three choices - "standardize" means removing the mean of each row and make the standard deviation one; "normal_score" means normal score transformation; "none" means do nothing. In that case we still assume some normalization has been done by the user such that each row has approximately mean 0 and sd 1.
- **combine.linear**: The two pieces of information is combined at the start to initiate the distance matrix.
Value

Returns a hclust object same as the output of hclust(). Reference: help(hclust)

merge an n-1 by 2 matrix. Row i of merge describes the merging of clusters at step i of the clustering. If an element j in the row is negative, then observation -j was merged at this stage. If j is positive then the merge was with the cluster formed at the (earlier) stage j of the algorithm.

height a set of n-1 real values, the value of the criterion associated with the clustering method for the particular agglomeration

order a vector giving the permutation of the original observations suitable for plotting, in the sense that a cluster plot using this ordering and matrix merge will not have crossings of the branches.

labels labels for each of the objects being clustered

call the call which produced the result

dist.method the distance that has been used to create d

height.0 original calculation of merging height

Author(s)

Tianwei Yu <tianwei.yu@emory.edu>

References


See Also

data.gen

Examples

## generating the data matrix & hiden clusters as a sample
input<-data.gen(n.genes=40, n.grps=4)
## now input includes data matrix and hiden clusters, so get the matrix as input.
input<-input$data

nlhc.data<-nlhc(input)
plot(nlhc.data)
##get the merge from the input.
merge<-nlhc.data$merge
\textbf{nlnet} \hspace{2cm} \textit{Non-Linear Network reconstruction from expression matrix}

\textbf{Description}

Non-Linear Network reconstruction method

\textbf{Usage}

\begin{verbatim}
nlnet(input, min.fdr.cutoff=0.05, max.fdr.cutoff=0.2, conn.proportion=0.007, gene.fdr.plot=FALSE, min.module.size=0, gene.community.method="multilevel", use.normal.approx=FALSE, normalization="standardize", plot.method="communitygraph")
\end{verbatim}

\textbf{Arguments}

- \texttt{input} \hspace{0.5cm} the data matrix with no missing values.
- \texttt{min.fdr.cutoff} \hspace{0.5cm} the minimum allowable value of the local false discovery cutoff in establishing links between genes.
- \texttt{max.fdr.cutoff} \hspace{0.5cm} the maximum allowable value of the local false discovery cutoff in establishing links between genes.
- \texttt{conn.proportion} \hspace{0.5cm} the target proportion of connections between all pairs of genes, if allowed by the fdr cutoff limits.
- \texttt{gene.fdr.plot} \hspace{0.5cm} whether plot a figure with estimated densities, distribution functions, and (local) false discovery rates.
- \texttt{min.module.size} \hspace{0.5cm} the min number of genes together as a module.
- \texttt{gene.community.method} \hspace{0.5cm} the method for community detection.
- \texttt{use.normal.approx} \hspace{0.5cm} whether to use the normal approximation for the null hypothesis.
- \texttt{normalization} \hspace{0.5cm} the normalization method for the array.
- \texttt{plot.method} \hspace{0.5cm} the method for graph and community plotting.

\textbf{Details}

gene.community.method: It provides three kinds of community detection method: "mutilevel", "label.propagation" and "leading.eigenvector".

use.normal.approx: If TRUE, normal approximation is used for every feature, AND all covariances are assumed to be zero. If FALSE, generates permutation based null distribution - mean vector and a variance-covariance matrix.

normalization: There are three choices: "standardize" means removing the mean of each row and make the standard deviation one; "normal_score" means normal score transformation; "none"
means do nothing. In that case we still assume some normalization has been done by the user such that each row has approximately mean 0 and sd 1.

plot.method: It provides three kinds of plotting method: "none" means plotting no graph, "communitygraph" means plotting community with graph, "graph" means plotting graph, "membership" means plotting membership of the community.

Value

it returns a graph and the community membership of the graph.

algorithm The algorithm name for community detection

graph An igraph object including edges: Numeric vector defining the edges, the first edge points from the first element to the second, the second edge from the third to the fourth, etc.

community Numeric vector, one value for each vertex, the membership vector of the community structure.

Author(s)

Haodong Liu <liuhaodong0828@gmail.com>

References


See Also
data.gen

Examples

## generating the data matrix & hidden clusters as a sample
input<-data.gen(n.genes=40, n.grps=4)
## now input includes data matrix and hidden clusters, so get the matrix as input.
input<-input$data
## change the plotting method
result<-nlnet(input,plot.method="graph")
## get the result and see its values
graph<-result$graph ## a igraph object.
comm<-result$community ## community of the graph

## use different community detection method
nlnet(input,gene.community.method="label.propagation")

## change the fdr pro to control connections of genes
## adjust the modularity size
nlnet(input,conn.proportion=0.005,min.module.size=10)
Nonlinear Variable Selection based on DCOL

Description

This is a nonlinear variable selection procedure for generalized additive models. It's based on DCOL, using forward stagewise selection. In addition, a cross-validation is conducted to tune the stopping alpha level and finalize the variable selection.

Usage

nvsd(X, y, fold = 10, step.size = 0.01, stop.alpha = 0.05, stop.var.count = 20, max.model.var.count = 10, roughening.method = "DCOL", do.plot = F, pred.method = "MARS")

Arguments

X  The predictor matrix. Each row is a gene (predictor), each column is a sample. Notice the dimensionality is different than most other packages, where each column is a predictor. This is to conform to other functions in this package that handles gene expression type of data.

y  The numerical outcome vector.

fold  The fold of cross-validation.

step.size  The step size of the roughening process.

stop.alpha  The alpha level (significance of the current selected predictor) to stop the iterations.

stop.var.count  The maximum number of predictors to select in the forward stagewise selection. Once this number is reached, the iteration stops.

max.model.var.count  The maximum number of predictors to select. Notice this can be smaller than the stop.var.count. Stop.var.count can be set more liniently, and this parameter controls the final maximum model size.

roughening.method  The method for roughening. The choices are "DCOL" or "spline".

do.plot  Whether to plot the points change in each step.

pred.method  The prediction method for the cross validation variable selection. As forward stagewise procedure doesn't do prediction, a method has to be borrowed from existing packages. The choices include "MARS", "RF", and "SVM".

Details

Please refer to the reference for details.
stage.forward

Value

A list object is returned. The components include the following.

- selected.pred: The selected predictors (row number).
- all.pred: The selected predictors by the forward stagewise selection. The $selected.pred is a subset of this.

Author(s)

Tianwei Yu<tianwei.yu@emory.edu>

References

https://arxiv.org/abs/1601.05285

See Also

stage.forward

Examples

```r
X<-matrix(rnorm(2000),ncol=20)
y<-sin(X[,1])+X[,2]^2+X[,3]
nvsd(t(X),y,stop.alpha=0.001,step.size=0.05)
```

---

stage.forward  Nonlinear Forward stagewise regression using DCOL

Description

The subroutine conducts forward stagewise regression using DCOL. Either DCOL roughening or spline roughening is conducted.

Usage

```r
stage.forward(X, y, step.size = 0.01, stop.alpha = 0.01, stop.var.count = 20, roughening.method = "DCOL", tol = 1e-08, spline.df = 5, dcol.sel.only = FALSE, do.plot = F)
```

Arguments

- `X`: The predictor matrix. Each row is a gene (predictor), each column is a sample. Notice the dimensionality is different than most other packages, where each column is a predictor. This is to conform to other functions in this package that handles gene expression type of data.
- `y`: The numerical outcome vector.
- `step.size`: The step size of the roughening process.
stage.forward

stop.alpha  The alpha level (significance of the current selected predictor) to stop the iterations.

stop.var.count  The maximum number of predictors to select. Once this number is reached, the iteration stops.

roughening.method  The method for roughening. The choices are "DCOL" or "spline".

tol  The tolerance level of sum of squared changes in the residuals.

spline.df  The degree of freedom for the spline.

dcol.sel.only  TRUE or FALSE. If FALSE, the selection of predictors will consider both linear and nonlinear association significance.

do.plot  Whether to plot the points change in each step.

Details

Please refer to the reference manuscript for details.

Value

A list object is returned. The components include the following.

found.pred  The selected predictors (row number).

ssx.rec  The magnitude of variance explained using the current predictor at each step.

$sel.rec  The selected predictor at each step.

$p.rec  The p-value of the association between the current residual and the selected predictor at each step.

Author(s)

Tianwei Yu<tianwei.yu@emory.edu>

References

https://arxiv.org/abs/1601.05285

See Also

nvsd

Examples

X<-matrix(rnorm(2000),ncol=20)
y<-sin(X[,1])+X[,2]^2+X[,3]
stage.forward(t(X),y,stop.alpha=0.001,step.size=0.05)
Index

* nonparametric
  nvsd, 8
  stage.forward, 9
* variable selection
  nvsd, 8
  stage.forward, 9

data.gen, 2, 3, 5, 7

KPC, 3

nlhc, 4
nlnet, 6
nvsd, 8

stage.forward, 9