Package 'Corbi'

May 3, 2017

Version 0.4-2

Title Collection of Rudimentary Bioinformatics Tools

Description Provides a bundle of basic and fundamental bioinformatics tools, such as network querying and alignment, subnetwork extraction and search, network biomarker identification.

ByteCompile TRUE

Depends R (>= 3.0.2)

Imports CRF, Matrix, mpmi

Suggests MASS, matrixcalc, knitr, rmarkdown

VignetteBuilder knitr

License GPL (>= 2)

BugReports https://github.com/wulingyun/Corbi/issues

URL https://github.com/wulingyun/Corbi

RoxygenNote 6.0.1

Author Ling-Yun Wu [aut, cre],
Qiang Huang [aut],
Duanchen Sun [aut]

Maintainer Ling-Yun Wu <wulingyun@gmail.com>

Repository CRAN

Repository/R-Forge/Project corbi

Repository/R-Forge/Revision 32

Repository/R-Forge/DateTimeStamp 2017-05-01 15:08:41

Date/Publication 2017-05-03 15:41:46 UTC

NeedsCompilation yes

R topics documented:

`Corbi-package` *Corbi - Collection of Rudimentary Bioinformatics Tools*

Description

This pakcage provides a bundle of basic and fundamental bioinformatics tools.

Details

These bioinformatics tools are developed by WuLab at Academy of Mathematics and Systems Science, Chinese Academy of Sciences.

Network querying and alignment:

- `net_query` Network querying method based on conditional random fields
- `net_query_batch` Batch processing version of `net_query`
- `net_align` Network alignment method based on conditional random fields

Subnetwork extraction and search:

- `get_subnets` Enumerate all subnetworks of limited size
- `extend_subnets` Extend subnetworks from smaller subnetworks
- `best_subnets` Search best subnetworks that maximize given objective function

Biomarker identification:

- `markrank` Biomarker identification and prioritization by integrating gene expression with biomolecular network

References

Qiang Huang, Ling-Yun Wu, and Xiang-Sun Zhang. An Efficient Network Querying Method Based on Conditional Random Fields. Bioinformatics, 27(22):3173-3178, 2011.

Qiang Huang, Ling-Yun Wu, and Xiang-Sun Zhang. Corbi: A new R package for biological network alignment and querying. BMC Systems Biology, 7(Suppl 2):S6, 2013.

`best_subnets` *The best subnetworks*

Description

Search best subnetworks that maximize given objective functions.

Usage

```
best_subnets(func, net.matrix, max.size = 10, exhaust.size = 5,
  max.top = 10000)
```

Arguments

`func`	The objective function to maximize
`net.matrix`	The adjacent matrix of network
`max.size`	The maximal size of subnetworks
`exhaust.size`	The maximal size of subnetworks that use exhaustive searching strategy
`max.top`	The maiximal number of top candidates kept for evaluation of next size, used in heuristic searching strategy

Details

Enumerate and search the best subnetworks that maximize given objective function. If the size of subnetworks <= `exhaust.size`, exact exhaustive searching is applied, otherwise, heuristic searching algorithm is used.

Value

A list with the following two components:

`subnets`	The list of top subnetworks in different sizes
`obj.values`	The list of objective values of corresponding subnetworks

See Also

get_subnets, extend_subnets

Examples

```
library(Corbi)
net <- matrix(FALSE, nrow=10, ncol=10)
net[sample.int(100, 20)] <- TRUE
net <- net | t(net)
func <- function(subnet) max(subnet) - min(subnet)
result <- best_subnets(func, net, 5)
```

`column` *Extract a column from a matrix*

Description

Extract a specified column from a sparse matrix rapidly

Usage

```
column(m, i)
```

Arguments

`m`	The matrix
`i`	The column index

Details

This function implements faster column extraction algorithm for the `CsparseMatrix` class in the package **Matrix**.

Value

This function will return the specified column as a vector of corresponding type.

`extend_subnets` *Extend subnetworks from smaller subnetworks*

Description

Extend subnetworks by pairwise overlapping two sets of smaller subnetworks.

Usage

```
extend_subnets(subnet1, subnet2, size = 0)
```

Arguments

subnet1	The matrix representing the first set of subnetworks
subnet2	The matrix representing the second set of subnetworks
size	The desired size of extended subnetworks

Details

Enumerate all possible subnetworks of desired size by pairwise overlapping two sets of subnetworks of size s1 and s2. The desired size should be between max(s1,s2)+1 and s1+s2-1. Invalid desired size will be replaced by the minimum allowed value max(s1,s2)+1.

Value

A matrix represents the extended subnetworks, in which each row represents a subnetwork.

Examples

```
library(Corbi)
net <- matrix(FALSE, nrow=10, ncol=10)
net[sample.int(100, 20)] <- TRUE
net <- net | t(net)
subnets <- get_subnets(net, 3)
subnets[[4]] <- extend_subnets(subnets[[3]], subnets[[2]], 4)
```

---

get_shortest_distances

Calculate shortest distances of unweighted network

Description

Calculate all pairs of shortest distances of unweighted network

Usage

```
get_shortest_distances(net.matrix, source.nodes = rep_len(TRUE,
  dim(net.matrix)[1]))
```

Arguments

net.matrix	Logical adjacency matrix of given unweighted network
source.nodes	Logical vector to indicate the source nodes that need to calculate the shortest distances

Details

This function calculates all pairs of shortest distances of unweighted network by using breadth-first-search (BFS) algorithm.

Value

This function will return the shortest distance matrix, where the element `[i, j]` is the shortest distance between node i and j. Value -1 means unreachable. If `source.nodes[i]` equals FALSE, the shortest distance from i to other nodes will not be calculated and the row i will be all -1.

---

get_subnets *All subnetworks of limited size*

Description

Enumerate all subnetworks of size <= `max.size` from given network.

Usage

```
get_subnets(net.matrix, max.size = 2)
```

Arguments

`net.matrix`	The adjacent matrix of network
`max.size`	The maximal size of subnetworks

Value

A list of generated subnetworks, with element i corresponds the subnetworks of size i. Each element is a matrix, in which each row represents a subnetwork.

Examples

```
library(Corbi)
net <- matrix(FALSE, nrow=10, ncol=10)
net[sample.int(100, 20)] <- TRUE
net <- net | t(net)
subnets <- get_subnets(net, 3)
```

kappa_score *Cohen's kappa score*

Description

Calculate Cohen's kappa score for two vectors.

Usage

```
kappa_score(x1, x2)
```

Arguments

x1	The first logical vector
x2	The second logical vector

Details

This function calculate Cohen's kappa score for two logical vectors.

Value

The Cohen's kappa score

markrank *MarkRank*

Description

MarkRank is a novel proposed network-based model, which can identify the cooperative biomarkers for heterogeneous complex disease diagnoses.

Usage

```
markrank(dataset, label, adj_matrix, alpha = 0.8, lambda = 0.2,
  eps = 1e-10, E_value = NULL, trace = TRUE, d = Inf,
  Given_NET2 = NULL)
```

Arguments

dataset	The microarray expression matrix of related disease. Each row represents a sample and each column represents a gene.
label	The 0-1 binary phenotype vector of dataset samples. The size of label must accord with the sample number in dataset.
adj_matrix	The 0-1 binary adjacent matrix of a connected biological network. Here the node set should be the same order as the gene set in expression matrix.
alpha	The convex combination coefficient of network effect and prior information vector E_value. The range of alpha is in [0,1]. A larger alpha will lay more emphasis on the network information. The default value is 0.8.
lambda	In the random walk-based iteration, matrix A1 reflects the stucture information of the biological network, whereas matrix A2 reflects the cooperative effect of gene combinations. Parameter lambda is the convex combination coefficient of two network effects. The range of lambda is in [0,1]. A larger lambda will lay more emphasis on the A1. The default value is 0.2.
eps	The stop criteria for the iterative solution method. The default value is 1e-10.
E_value	A vector containing the prior information about the importance of nodes. Default is the absolute Pearson correlation coefficient (PCC).
trace	Locaical variable indicated whether tracing information on the progress of the gene cooperation network construction is produced.
d	Threshold for simplifying the G_2 computation. Only the gene pairs whose shortest distances in PPI network are less than d participate in the G_2 computation. The default value is Inf.
Given_NET2	Whether a computed cooperation network is given for tuning parameter. See Details for a more specific description.

Details

MarkRank is a network-based biomarker identification method to prioritize disease genes by integrating multi-source information including the biological network, e.g protein-protein interaction (PPI) network, the prior information about related diseases, and the discriminative power of cooperative gene combinations. MarkRank shows that explicit modeling of gene cooperative effects can greatly improve biomarker identification for complex disease, especially for diseases with high heterogeneity.

MarkRank algorithm contains mainly two steps: 1) The construction of gene cooperation network G_2 and 2) a random walk based iteration procedure. The following descriptions will help the users to using markrank more convenient:

1) As for the construction of the gene cooperation network, we suggest the user to set trace=TRUE to output the G_2 computation process. The G_2 construction step finished if the output number is identical to the gene number of the input expression matrix. The parameter d introduced the structure information of used biological network to facilitate the construction of G_2, only the gene pairs whose shortest distances in network are less than d participate the G_2 computation. We suggest d=Inf, the default value, to fully use the information of expression matrix. If the user given a preset d, the distance matrix of input network dis will be returned.

2) MarkRank uses a random-walk based iteration procedure to score each gene. The detailed formula is:

score = alpha*[lambda*A1 + (1-lambda)*A2]*score + (1-alpha)*E_value.

The users could set an appropriate parameter settings in their pracitical application. Our suggested value is alpha=0.8 and lambda=0.2. The model input parameter combinations and iteration steps will be returned in output components initial_pars and steps, respectively. Because the iteration step is separate with the cooperation network construction, the user can use the parameter Given_NET2 to tune the model parameters. In detail, the user could set

```
Given_NET2 = result$NET2
```

in markrank input to avoid the repeated computation of G_2, where the object result is the returned variable of markrank function.

3) The final MarkRank score for each gene is in output score. The users could sort this result and use the top ranked genes for further analysis.

Value

This function will return a list with the following components:

score	The vector of final MarkRank scores for each gene.
steps	The final iteration steps in random walk based scoring procedure.
NET2	The weighted adjacent matrix of gene cooperation network.
initial_pars	The initial/input parameter values used in MarkRank.
dis	The pairwise distance matrix of input network. This variable will be Null if input d=Inf.

References

Duanchen Sun, Xianwen Ren, Eszter Ari, Tamas Korcsmaros, Peter Csermely, Ling-Yun Wu. Discovering cooperative biomarkers for heterogeneous complex disease diagnoses. Manuscript, 2017.

---

net_align *Network alignment method based on conditional random fields*

Description

Find the maximal matching subnetworks from a target network for a query network based on the conditional random fields (CRF) model.

Usage

```
net_align(query.net, target.net, node.sim, query.type = 4, delta.d = 1e-10,
  delta.c = 0.5, delta.e = 1, delta.s = 1, output = "result.txt")
```

Arguments

query.net	The input file name of the query network.
target.net	The input file name of the target network.
node.sim	The input file name of the node similarity scores between the query network and the target network.
query.type	The querying network type: 1 - general, 2 - chain, 3 - tree, 4 - heuristic.
delta.d	The parameter delta.d is a parameter for deletions.
delta.c	The parameter delta.c is a parameter for consecutive deletions.
delta.e	The parameter delta.e is a parameter for single deletion.
delta.s	The parameter delta.s is a parameter for insertions.
output	The suffix of output file name. The output contains two files in the working directory. One is the matching nodes and edges between query network and target network, the other is the unique matching node pairs.

Details

This is an approach for network alignment problem based on conditional random field (CRF) model which uses the node similarity and structure information equally. This method is based on our network querying method net_query. This method uses an iterative strategy to get the one-to-one map between the query network and target netowrk.

More details can be seen in net_query.

References

Qiang Huang, Ling-Yun Wu, and Xiang-Sun Zhang. CNetA: Network alignment by combining biological and topological features. In Proceedings of 2012 IEEE International Conference on Systems Biology (ISB), 220-225, IEEE, 2012.

Qiang Huang, Ling-Yun Wu, and Xiang-Sun Zhang. Corbi: A new R package for biological network alignment and querying. BMC Systems Biology, 7(Suppl 2):S6, 2013.

Examples

```
## Not run:
library(Corbi)

## An example: "querynet.txt", "targetnet.txt", "nodesim.txt" are
## three input files in the working directory
net_align("querynet.txt", "targetnet.txt", "nodesim.txt")

## End(Not run)
```

net_query *Network querying method based on conditional random fields*

Description

Find the best matching subnetworks from a large target network for small query networks based on the conditional random fields (CRF) model.

Usage

```
net_query(query.net, target.net, node.sim, query.type = 4, delta.d = 1e-10,
  delta.c = 0.5, delta.e = 1, delta.s = 1, output = "result.txt")

net_query_batch(query.nets, target.net, node.sim, query.type = 4,
  delta.d = 1e-10, delta.c = 0.5, delta.e = 1, delta.s = 1,
  output = "result.txt")
```

Arguments

query.net	The input file name of the query network.
target.net	The input file name of the target network.
node.sim	The input file name of the node similarity scores between the query network and the target network.
query.type	The querying network type: 1 - general, 2 - chain, 3 - tree, 4 - heuristic.
delta.d	The parameter delta.d is a parameter for deletions.
delta.c	The parameter delta.c is a parameter for consecutive deletions.
delta.e	The parameter delta.e is a parameter for single deletion.
delta.s	The parameter delta.s is a parameter for insertions.
output	The suffix of output file name.
query.nets	The vector of input file names of the query networks.

Details

This is an approach for network querying problem based on conditional random field (CRF) model which can handle both undirected and directed networks, acyclic and cyclic networks, and any number of insertions/deletions.

When querying several networks in the same target network, net_query_batch will save much time.

- query.net: The query network file is written as follows:
  v1 v2 v3 v4 v5
  v3 v4
  ...
  where v1, v2, v3, v4, v5 ... are the nodes' names and each line indicates there are edges between the first node and other nodes in the line. For example, the first line denotes 4 edges: (v1, v2), (v1, v3), (v1, v4), and (v1, v5).

- target.net: The format of this file is the same as the query network file.
- node.sim: This similarity file's format is as follows:
  v1 V1 s1
  v1 V2 s2
  ...
  v1 is the node from the query network, V1 is the node from the target network, s1 is the similarity score between the node v1 and V1, and so on.
- query.type: If query.type = 1, the loopy belief propagation (LBP) algorithm will be applied, which is an approximate algorithm for a general graph with loops. If the query is a chain or tree, there are exact algorithms. Set query.type = 2 when the query is a chain, and query.type = 3 when the query is a tree. The heuristic algorithm will be used when query.type = 4, which will try the exact algorithm (junction tree algorithm) first and resort to LBP algorithm when the exact algorithm failed. The default value is 4.
- delta.d: The smaller delta.d is, the heavier penalty for deletions.
- delta.c: The smaller delta.c is, the heavier penalty for consecutive deletions.
- delta.e: The smaller delta.e is, the heavier penalty for single deletion.
- delta.s: The larger delta.s indicates heavier penalty for insertions.

References

Qiang Huang, Ling-Yun Wu, and Xiang-Sun Zhang. An Efficient Network Querying Method Based on Conditional Random Fields. Bioinformatics, 27(22):3173-3178, 2011.

Examples

```
## Not run:
library(Corbi)

## An example: "querynet.txt", "targetnet.txt", "nodesim.txt" are
## three input files in the working directory
net_query("querynet.txt", "targetnet.txt", "nodesim.txt", query.type=3)

## End(Not run)


## Not run:
## Batch example
net_query_batch(c("querynet.txt", "querynet2.txt"),
  "targetnet.txt", "nodesim.txt", query.type=3)

## End(Not run)
```

nnzero *The number of non-zero values of a submatrix*

Description

Retuen the number of non-zero values of the specified submatrix of a given sparse matrix rapidly

Usage

```
nnzero(m, rows = 1:dim(m)[1], cols = 1:dim(m)[2])
```

Arguments

m	The matrix
rows	The integer vector of row index(es) or logical vector indicated the selected rows
cols	The integer vector of column index(es) or logical vector indicated the selected cols

Details

This function implements faster calculation algorithm for the CsparseMatrix and RsparseMatrix class in the package **Matrix**.

Value

This function will return the number of non-zero values in the specified submatrix.

read_net *Read network information from text file*

Description

Read the network information from a text file with specific format.

Usage

```
read_net(file)
```

Arguments

file	The name of text file

Details

This function reads the network information from a text file with specific format: each line contains two strings separated by spaces, which correspond to the names of two end points of one edge in the network.

Value

A list with the following components:

size	The number of network nodes
node	The vector of network node names
matrix	The logical adjacency matrix

See Also

write_net

---

rmultihyper *The Multivariate Hypergeometric Distribution*

Description

Generate random variables for the multivariate hypergeometric distribution

Usage

```
rmultihyper(n, k, m)
```

Arguments

n	The number of observations.
k	The total number of balls drawn from the urn.
m	The integer vector containing the number of balls of each color in the urn. Length of vector is the number of colors.

Details

This function generates random variables for the multivariate hypergeometric distribution by iteratively calling hypergeometric random variable generator rhyper.

Value

This function will return a matrix of length(m) rows and n columns, and each column contains the number of balls of each color drawn from the urn.

See Also

rhyper

submatrix *Extract a submatrix from a matrix*

Description

Extract a specified submatrix from a sparse matrix rapidly

Usage

```
submatrix(m, rows, cols)
```

Arguments

m	The matrix
rows	The integer vectors of row index(es)
cols	The integer vectors of column index(es)

Details

This function implements faster submatrix extraction algorithm for the CsparseMatrix class in the package **Matrix**.

Value

This function will return the specified submatrix as a matrix of corresponding type.

write_net *Write network information to text file*

Description

Write the network information to a text file with specific format.

Usage

```
write_net(net, file)
```

Arguments

net	A list as returned by read_net
file	The name of text file

Details

This function writes the network information to a text file with specific format: each line contains two strings separated by spaces, which correspond to the names of two end points of one edge in the network.

See Also

read_net

Index