Package 'DCD'

September 12, 2017

Type Package

Title Differential Community Detection in Paired Biological Networks

Version 0.1.0

Author Raghvendra Mall [aut, cre], Khalid Kunji [ctb], Michele Ceccarelli [ctb]

Maintainer Raghvendra Mall <rmall@hbku.edu.qa>

Description

A differential community detection (DCD) based approach to effectively locate differential sub-networks in paired scale-free biological networks, e.g. case vs control - Raghvendra Mall et al (2017) <doi:10.1145/3107411.3107418>.

License GPL

URL https://www.r-project.org,

http://dl.acm.org/citation.cfm?id=3107418

Imports Rdpack, data.table, igraph, ggplot2, ROCR, lsa, Matrix, lattice, WGCNA, qlcMatrix, foreach, plyr, doParallel

Depends R (>= 3.4.0)

Encoding UTF-8

LazyData true

RdMacros Rdpack

NeedsCompilation no

Repository CRAN

Date/Publication 2017-09-12 17:43:07 UTC

R topics documented:

`calculate_jaccard` *Calculate jaccard coefficient w.r.t. to a hub differential node*

Description

Calculate jaccard coefficients of all nodes w.r.t. a hub differential node which help in formation of approximate block diagonals

Usage

```
calculate_jaccard(nodeid, matrix_A, mink)
```

Arguments

`nodeid`	Id of the hub differential node
`matrix_A`	Noisy differential topological adjacency matrix
`mink`	Minimum number of neighbours that each node should have in common with hub node to have a non-zero Jaccard co-efficient.

Value

Returns a data-frame with fields: jaccard_coefficient, intersection_length and degree. Here jaccard_coefficient between hub node and every node, intersection_length is number of common nodes and degree represents degree of each node in differential topological matrix.

Author(s)

Raghvendra Mall <rmall@hbku.edu.qa>

References

Francois R (2014). *bibtex: bibtex parser*. R package version 0.4.0, https://CRAN.R-project.org/package=bibtex.

Examples

```
library(igraph)
mink<-7;
g_A <- sample_grg(200, 0.15, torus=FALSE, coords = FALSE)
A <- get.adjacency(g_A)
nodeid <- 1;

#Calculate Jaccard coefficient of all nodes w.r.t. nodeid 1 given the adjacency matrix A
output <- calculate_jaccard(nodeid,A,mink)
```

DCD *Differential Community Detection for Paired Biological Networks, for e.g. case vs control*

Description

The task of identifying differential sub-networks in paired biological networks (A:control,B:case) can be re-phrased as one of finding dense communities in a single noisy differential topological (DT) graph constructed by taking absolute difference between the topological graphs of A and B. In this software, we propose a fast three-stage approach, namely Differential Community Detection (DCD), to identify differential sub-networks as differential communities in a de-noised version of the DT graph. In the first stage, we iteratively re-order the nodes of the DT graph to determine approximate block diagonals present in the DT adjacency matrix using neighbourhood information of the nodes and Jaccard similarity. In the second stage, the ordered DT adjacency matrix is traversed along the diagonal to remove all the edges associated with a node, if that node has no immediate edges within a window. Finally, we apply community detection methods on this de-noised DT graph to discover differential sub-networks as communities.

Usage

```
DCD(g_A = sample_grg(200, 0.15, torus = FALSE, coords = FALSE),
    g_B = permute(g_A, c(sample(20), 21:200)), method = "Louvain",
    mink = 7, ground_truth = c(rep(1, 20), rep(0, 180)),
    plot_flag = 1, color = "blue", iter = 1, cores = 1)
```

Arguments

g_A	Igraph object representing graph A
g_B	Igraph object representing graph B
method	Community detection method can be either: "Louvain", "Infomap" or "Spectral", default "Louvain".
mink	Minimum number of nodes for a community to be considered differential community, default 7.
ground_truth	In case ground truth community information is available i.e. which nodes are part of differential communities and which are not. For example in 2 networks comprising 1000 nodes, if first 100 nodes are differential then ground_truth = c(rep(1,100),rep(0,900)).
plot_flag	Makes a precision recall plot if plot_flag=1 when ground_truth is available, else set plot_flag to 0.
color	Colour for the area under precision recall curve. Can be "blue", "black", "red" etc.
iter	No of iterations of experiment to run, default iter = 1.
cores	No of cores to use to calculate the Jaccard co-efficient in parallel for the hub node during each iteration of the greedy block diagonal identification approach.

Value

Returns a data frame with NodeIds and Predicted_Label columns. The Predicted_Label corresponds to the Differential ClusterId a node belongs to . All nodes with Predicted_Label = 0 are non-differential nodes or are nodes which are not part of any differential community.

Author(s)

Raghvendra Mall <rmall@hbku.edu.qa>

References

Francois R (2014). *bibtex: bibtex parser*. R package version 0.4.0, https://CRAN.R-project.org/package=bibtex.

See Also

get_ordered_community_output, calculate_jaccard, order_topological_matrix, prune_edges

Examples

```
DCD_output <- DCD()

## Not run:
library(igraph)
g_A <- sample_grg(200, 0.15, torus=FALSE, coords = FALSE)
g_B <- permute(g_A,c(sample(20),21:200))
ground_truth <- c(rep(1,20),rep(0,180))
DCD_output2 <- DCD(g_A = g_A, g_B = g_B, method = "Spectral", mink = 7,
                   ground_truth = ground_truth, plot_flag = 1, color = "red",
                   iter = 1, cores = 1 )

## End(Not run)
```

---

get_ordered_community_output

Get differential communities and nodes belonging to each such community

---

Description

Third step of DCD: Performs community detection on de-noised ordered differential topological graph.

Usage

```
get_ordered_community_output(g_output, method, output_adjacency, plot_flags = 0,
                             ground_truth = NULL, color = NULL, iter = 0)
```

Arguments

g_output	De-noised ordered differential topological graph.
method	Method can be either "Louvain", "Infomap" or "Spectral", by default "Louvain".
output_adjacency	De-noised ordered differential topological adjacency matrix.
plot_flags	Makes a precision recall plot if plot_flag=1 when ground_truth is available, else set plot_flag to 0.
ground_truth	In case ground truth community information is available i.e. which nodes are part of differential communities and which are not. For example in 2 networks comprising 1000 nodes, if first 100 nodes are differential then ground_truth = c(rep(1,100),rep(0,900)).
color	Colour for the area under precision recall curve. Can be "blue", "black", "red" etc.
iter	No of iterations of experiment to run, default iter = 1.

Value

Returns a data frame with NodeIds and Predicted_Label columns. The Predicted_Label corresponds to the Differential ClusterId a node belongs to . All nodes with Predicted_Label = 0 are non-differential nodes or are nodes which are not part of any differential community.

Author(s)

Raghvendra Mall <rmall@hbku.edu.qa>

References

Francois R (2014). *bibtex: bibtex parser*. R package version 0.4.0, https://CRAN.R-project.org/package=bibtex.

Examples

```
library(igraph)
library(WGCNA)
g_A <- sample_grg(200, 0.15, torus=FALSE, coords = FALSE)
A <- get.adjacency(g_A)
g_B = permute(g_A,c(sample(20),21:200))
B <- get.adjacency(g_B)

cosine_sim_A <- TOMsimilarity(as.matrix(A),TOMType = "unsigned",TOMDenom = "min");
cosine_sim_B <- TOMsimilarity(as.matrix(B),TOMType = "unsigned",TOMDenom = "min")
edgelist_A <- get.edgelist(g_A);
edgelist_B <- get.edgelist(g_B);
if (is.null(E(g_A)$weight))
{
  edgelist_A <- cbind(edgelist_A,rep(1,nrow(edgelist_A)));
}else
```

```
{
  edgelist_A <- cbind(edgelist_A,E(g_A)$weight);
}
if (is.null(E(g_B)$weight))
{
  edgelist_B <- cbind(edgelist_B,rep(1,nrow(edgelist_B)));
}else
{
  edgelist_B <- cbind(edgelist_B,E(g_B)$weight);
}
edgelist_A <- as.data.frame(edgelist_A);
edgelist_B <- as.data.frame(edgelist_B);

mink <- 7
#Noisy Difference in topological matrices
diff_topological_matrix <- abs(cosine_sim_A-cosine_sim_B);

#Order the nodes in topological graph using greedy approach
ordered_list <- order_topological_matrix(diff_topological_matrix,mink);
temp_output_adjacency <- diff_topological_matrix[ordered_list,ordered_list];

output_adjacency <- prune_edges(temp_output_adjacency,mink);
g_output <- graph_from_adjacency_matrix(output_adjacency,mode=c("undirected"),weighted=TRUE);

#Community detection using Louvain method
louvain_cluster_node_rank <- get_ordered_community_output(g_output,"Louvain",
                          output_adjacency);
output <- cbind(as.numeric(louvain_cluster_node_rank$NodeIds),
                louvain_cluster_node_rank$Predicted_Label)
```

`order_topological_matrix` *Order the noisy differential topological adjacency matrix*

Description

First step of DCD approach. Orders the noisy differential topological adjacency matrix to generate approximate block diagonals.

Usage

```
order_topological_matrix(A, mink)
```

Arguments

`A`	Noisy differential topological adjacency matrix
`mink`	Minimum size of a community for it to be considered differential, default value is 7.

Value

Returns an ordered list for nodeids, by using which the noisy differential matrix is ordered such that it consists of approximate block diagonals.

Author(s)

Raghvendra Mall <rmall@hbku.edu.qa>

References

Francois R (2014). *bibtex: bibtex parser*. R package version 0.4.0, https://CRAN.R-project.org/package=bibtex.

See Also

calculate_jaccard

Examples

```
#Get ordered list for approximate block diagonals
library(igraph)
library(WGCNA)
g_A <- sample_grg(200, 0.15, torus=FALSE, coords = FALSE)
A <- get.adjacency(g_A)
g_B = permute(g_A,c(sample(20),21:200))
B <- get.adjacency(g_B)

cosine_sim_A <- TOMsimilarity(as.matrix(A),TOMType = "unsigned",TOMDenom = "min");
cosine_sim_B <- TOMsimilarity(as.matrix(B),TOMType = "unsigned",TOMDenom = "min")
edgelist_A <- get.edgelist(g_A);
edgelist_B <- get.edgelist(g_B);
if (is.null(E(g_A)$weight))
{
  edgelist_A <- cbind(edgelist_A,rep(1,nrow(edgelist_A)));
}else
{
  edgelist_A <- cbind(edgelist_A,E(g_A)$weight);
}
if (is.null(E(g_B)$weight))
{
  edgelist_B <- cbind(edgelist_B,rep(1,nrow(edgelist_B)));
}else
{
  edgelist_B <- cbind(edgelist_B,E(g_B)$weight);
}
edgelist_A <- as.data.frame(edgelist_A);
edgelist_B <- as.data.frame(edgelist_B);

mink <- 7
#Noisy Difference in topological matrices
diff_topological_matrix <- abs(cosine_sim_A-cosine_sim_B);
```

```
#Order the nodes in topological graph using greedy approach
ordered_list <- order_topological_matrix(diff_topological_matrix,mink);
```

---

prune_edges *De-noises ordered differential topological matrix*

Description

Second step of DCD: De-noises the ordered differential topological matrix to generate BDs.

Usage

```
prune_edges(A, min_size_cluster)
```

Arguments

A	Ordered noisy differential topological adjacency matrix
min_size_cluster	Minimum size of a community for it to be considered differential, default value is 7.

Value

De-noised ordered topological adjacency matrix

Author(s)

Raghvendra Mall <rmall@hbku.edu.qa>

References

Francois R (2014). *bibtex: bibtex parser*. R package version 0.4.0, https://CRAN.R-project.org/package=bibtex.

Examples

```
#Denoise the noisy ordered adjacency matrix
library(igraph)
library(WGCNA)
g_A <- sample_grg(200, 0.15, torus=FALSE, coords = FALSE)
A <- get.adjacency(g_A)
g_B = permute(g_A,c(sample(20),21:200))
B <- get.adjacency(g_B)

cosine_sim_A <- TOMsimilarity(as.matrix(A),TOMType = "unsigned",TOMDenom = "min");
cosine_sim_B <- TOMsimilarity(as.matrix(B),TOMType = "unsigned",TOMDenom = "min")
```

```
edgelist_A <- get.edgelist(g_A);
edgelist_B <- get.edgelist(g_B);
if (is.null(E(g_A)$weight))
{
  edgelist_A <- cbind(edgelist_A,rep(1,nrow(edgelist_A)));
}else
{
  edgelist_A <- cbind(edgelist_A,E(g_A)$weight);
}
if (is.null(E(g_B)$weight))
{
  edgelist_B <- cbind(edgelist_B,rep(1,nrow(edgelist_B)));
}else
{
  edgelist_B <- cbind(edgelist_B,E(g_B)$weight);
}
edgelist_A <- as.data.frame(edgelist_A);
edgelist_B <- as.data.frame(edgelist_B);

mink <- 7
#Noisy Difference in topological matrices
diff_topological_matrix <- abs(cosine_sim_A-cosine_sim_B);

#Order the nodes in topological graph to create block diagonals
ordered_list <- order_topological_matrix(diff_topological_matrix,mink);
temp_output_adjacency <- diff_topological_matrix[ordered_list,ordered_list];

#Perform the greedy deterministic approach to remove spurious edges and keep significant ones
output_adjacency <- prune_edges(temp_output_adjacency,mink);
```

Index