

# Package: DCG (via r-universe)

July 11, 2024

**Type** Package

**Title** Data Cloud Geometry (DCG): Using Random Walks to Find Community Structure in Social Network Analysis

**Version** 0.9.3

**Date** 2019-04-03

**Depends** R (>= 2.14.0)

**Description** Data cloud geometry (DCG) applies random walks in finding community structures for social networks. Fushing, VanderWaal, McCowan, & Koehl (2013) (<[doi:10.1371/journal.pone.0056259](https://doi.org/10.1371/journal.pone.0056259)>).

**License** GPL (>= 2)

**Copyright** Fushing Lab & McCowan Lab, University of California, Davis

**Imports** stats

**LazyData** TRUE

**Suggests** testthat, knitr, rmarkdown, devtools, lattice, png

**RoxygenNote** 6.1.1

**VignetteBuilder** knitr

**Repository** <https://hanettools.r-universe.dev>

**RemoteUrl** <https://github.com/hanettools/dcg>

**RemoteRef** HEAD

**RemoteSha** 93d15e2cc50e799c816c6c2dea5ed084ae8f3b0d

## Contents

as.SimilarityMatrix . . . . .	2
as.symmetricAdjacencyMatrix . . . . .	2
getEigenvalueList . . . . .	3
getEns . . . . .	4
getEnsList . . . . .	5
GetSim . . . . .	6
monkeyGrooming . . . . .	7

plotCLUSTERS . . . . .	7
plotMultiEigenvalues . . . . .	8
plotTheCluster . . . . .	10
temperatureSample . . . . .	10

<b>Index</b>	<b>12</b>
--------------	-----------

---

as.SimilarityMatrix	<i>Convert a matrix to a similarity matrix. as.SimilarityMatrix convert an adjacency matrix to a similarity matrix.</i>
---------------------	---

---

### Description

Convert a matrix to a similarity matrix. as.SimilarityMatrix convert an adjacency matrix to a similarity matrix.

### Usage

```
as.SimilarityMatrix(mat)
```

### Arguments

mat                    a symmetric adjacency matrix

### Value

a similarity matrix.

### Examples

```
symmetricMatrix <- as.symmetricAdjacencyMatrix(monkeyGrooming, weighted = TRUE, rule = "weak")
similarityMatrix <- as.SimilarityMatrix(symmetricMatrix)
```

---

as.symmetricAdjacencyMatrix	<i>convert to a symmetric adjacency matrix</i>
-----------------------------	--

---

### Description

as.symmetricAdjacencyMatrix convert an edgelist or a raw matrix to a symmetric adjacency matrix.

### Usage

```
as.symmetricAdjacencyMatrix(Data, weighted = FALSE, rule = "weak")
```

**Arguments**

Data	either a dataframe or a matrix, representing raw interactions using either an edgelist or a matrix. Frequency of interactions for each dyad can be represented either by multiple occurrences of the dyad for a 2-column edgelist, or by a third column specifying the frequency of the interaction for a 3-column edgelist.
weighted	If the edgelist is a 3-column edgelist in which weight was specified by frequency, use weighted = TRUE.
rule	a character vector of length 1, being one of "weak", "strong", "upper", or "lower". Ways of symmetrizing the matrix. See details for more information.

**Details**

There are ways of symmetrizing a matrix. The "weak" rule symmetrize the matrix by building an edge between nodes [i, j] and [j, i] if there is an edge either from i to j OR from j to i. The "strong" rule symmetrize the matrix by building an edge between nodes [i, j] and [j, i] if there is an edge BOTH from i to j AND from j to i. The "upper" and the "lower" rule symmetrize the matrix by using the "upper" or the "lower" triangle respectively.

Note, when using a 3-column edgelist (e.g. a weighted edgelist) to represent raw interactions, each dyad must be unique. If more than one rows are found with the same Initiator and recipient, sum of the frequencies will be taken to represent the frequency of interactions between this unique dyad. A warning message will prompt your attention to the accuracy of your raw data when duplicated dyads were found in a three-column edgelist.

**Value**

a named matrix with the [i, j]th entry equal to the number of times i grooms j.

**Examples**

```
symmetricMatrix <- as.symmetricAdjacencyMatrix(monkeyGrooming, weighted = TRUE, rule = "weak")
```

---

getEigenvalueList	<i>generate eigenvalues for all ensemble matrices</i>	getEigenvalueList
	<i>get eigenvalues from ensemble matrices</i>	

---

**Description**

generate eigenvalues for all ensemble matrices  
getEigenvalueList  
get eigenvalues from ensemble matrices

**Usage**

```
getEigenvalueList(EnsList)
```

**Arguments**

EnsList            a list of ensemble matrices

**Value**

a list of eigenvalues for each of the ensemble matrix in the ensemble matrices list.

---

getEns	<i>generate ensemble matrix getEns get ensemble matrix from given similarity matrix and temperature</i>
--------	---

---

**Description**

generate ensemble matrix getEns get ensemble matrix from given similarity matrix and temperature

**Usage**

```
getEns(simMat, temperature, MaxIt = 1000, m = 5)
```

**Arguments**

simMat            a similarity matrix

temperature      a numeric vector of length 1, indicating the temperature used to transform the similarity matrix to ensemble matrix

MaxIt             number of iterations for regulated random walks

m                 maximum number of time a node can be visited during random walks

**Details**

This function involves two steps. It first generate similarity matrices of different variances by taking the raw similarity matrix to the power of each temperature. Then it called the function EstClust to perform random walks in the network to identify clusters.

**Value**

a matrix.

---

getEnsList	<i>generating a list of ensemble matrices based on the similarity matrix and temperatures</i>
------------	---

---

### Description

getEnsList get ensemble matrices from given similarity matrix at all temperatures

### Usage

```
getEnsList(simMat, temperatures, MaxIt = 1000, m = 5)
```

### Arguments

simMat	a similarity matrix
temperatures	temperatures selected
MaxIt	number of iterations for regulated random walks
m	maximum number of time a node can be visited during random walks

### Details

This step is crucial in finding community structure based on the similarity matrix of the social network. For each temperatures, the similarity matrix was taken to the power of temperature as saved as a new similarity matrix. This allows the random walk to explore the similarity matrix at various variations. Random walks are then performed in similarity matrices of various temperatures. In order to prevent random walks being stucked in a locale, the parameter m was set (to 5 by default) to remove a node after m times of visits of the node. An ensemble matrix is generated at each temperature in which values represent likelihood of two nodes being in the same community.

### Value

a list of ensemble matrices

### References

- Fushing, H., & McAssey, M. P. (2010). Time, temperature, and data cloud geometry. *Physical Review E*, 82(6), 061110.
- Chen, C., & Fushing, H. (2012). Multiscale community geometry in a network and its application. *Physical Review E*, 86(4), 041120.
- Fushing, H., Wang, H., VanderWaal, K., McCowan, B., & Koehl, P. (2013). Multi-scale clustering by building a robust and self correcting ultrametric topology on data points. *PloS one*, 8(2), e56259.

## Examples

```
symmetricMatrix <- as.symmetricAdjacencyMatrix(monkeyGrooming, weighted = TRUE, rule = "weak")
Sim <- as.SimilarityMatrix(symmetricMatrix)
temperatures <- temperatureSample(start = 0.01, end = 20, n = 20, method = 'random')
## Not run:
# Note: It takes a while to run the getEnsList example.
Ens_list <- getEnsList(Sim, temperatures, MaxIt = 1000, m = 5)

## End(Not run)
```

---

GetSim

GetSim *get similarity matrix from a distance matrix*

---

## Description

GetSim get similarity matrix from a distance matrix

## Usage

```
GetSim(D, T)
```

## Arguments

D	A distance matrix
T	Temperature. <a href="#">temperatureSample</a>

## Details

the similarity matrix is calculated at each temperature T.

## References

Fushing, H., & McAssey, M. P. (2010). Time, temperature, and data cloud geometry. *Physical Review E*, 82(6), 061110.

Chen, C., & Fushing, H. (2012). Multiscale community geometry in a network and its application. *Physical Review E*, 86(4), 041120.

Fushing, H., Wang, H., VanderWaal, K., McCowan, B., & Koehl, P. (2013). Multi-scale clustering by building a robust and self correcting ultrametric topology on data points. *PloS one*, 8(2), e56259.

---

monkeyGrooming	<i>Grooming network data</i>
----------------	------------------------------

---

**Description**

A dataset containing grooming edgelist among monkeys.

**Usage**

```
monkeyGrooming
```

**Format**

A data frame with 1595 rows and 2 variables:

**Initiator** Grooming Initiator ID

**Recipient** Grooming Recipient ID

**Groom** Grooming Frequency ...

---

plotCLUSTERS	<i>generate tree plots for each ensemble matrix plotCLUSTERS plot all cluster trees</i>
--------------	---

---

**Description**

generate tree plots for each ensemble matrix plotCLUSTERS plot all cluster trees

**Usage**

```
plotCLUSTERS(EnsList, mfrow, mar = c(1, 1, 1, 1), line = -1.5,
  cex = 0.5, ...)
```

**Arguments**

EnsList	a list in which elements are ensemble matrices.
mfrow	A vector of the form <code>c(nr, nc)</code> passed to <code>par</code> .
mar	plotting parameters with useful defaults ( <code>par</code> )
line	plotting parameters with useful defaults ( <code>par</code> )
cex	plotting parameters with useful defaults ( <code>par</code> )
...	further plotting parameters

**Details**

plotCLUSTERS plots all cluster trees with each tree corresponding to each ensemble matrix in the list of ens\_list. EnsList is the output from getEnsList.

mfrow determines the arrangement of multiple plots. It takes the form of `c(nr, nc)` with the first parameter being the number of rows and the second parameter being the number of columns. When deciding parameters for mfrow, one should take into considerations size of the plotting device and number of cluster plots. For example, there are 20 cluster plots, mfrow can be set to `c(4, 5)` or `c(2, 10)` depending on the size and shape of the plotting area.

**Value**

a graph containing all tree plots with each tree plot corresponding to the community structure from each of the ensemble matrix.

**References**

Fushing, H., & McAssey, M. P. (2010). Time, temperature, and data cloud geometry. *Physical Review E*, 82(6), 061110.

Chen, C., & Fushing, H. (2012). Multiscale community geometry in a network and its application. *Physical Review E*, 86(4), 041120.

Fushing, H., Wang, H., VanderWaal, K., McCowan, B., & Koehl, P. (2013). Multi-scale clustering by building a robust and self correcting ultrametric topology on data points. *PloS one*, 8(2), e56259.

**See Also**

[getEnsList](#)

**Examples**

```
symmetricMatrix <- as.symmetricAdjacencyMatrix(monkeyGrooming, weighted = TRUE, rule = "weak")
Sim <- as.SimilarityMatrix(symmetricMatrix)
temperatures <- temperatureSample(start = 0.01, end = 20, n = 20, method = 'random')
## Not run:
# for illustration only. skip CRAN check because it ran forever.
Ens_list <- getEnsList(Sim, temperatures, MaxIt = 1000, m = 5)

## End(Not run)

plotCLUSTERS(EnsList = Ens_list, mfrow = c(2, 10), mar = c(1, 1, 1, 1))
```

---

plotMultiEigenvalues *plot eigenvalues plotMultiEigenvalues plot eigenvalues to determine number of communities by finding the elbow point*

---

**Description**

plot eigenvalues plotMultiEigenvalues plot eigenvalues to determine number of communities by finding the elbow point



**Usage**

```
plotMultiEigenvalues(Ens_list, mfrow, mar = c(2, 2, 2, 2), line = -1.5,  
  cex = 0.5, ...)
```

**Arguments**

Ens_list	a list in which elements are numeric vectors representing eigenvalues.
mfrow	A vector of the form <code>c(nr, nc)</code> passed to <code>par</code> .
mar	plotting parameters with useful defaults ( <code>par</code> )
line	plotting parameters with useful defaults ( <code>par</code> )
cex	plotting parameters with useful defaults ( <code>par</code> )
...	further plotting parameters

**Details**

`plotMultiEigenvalues` plot multiple eigenvalue plots. The dark blue colored dots indicate eigenvalue greater than 0. Each of the ensemble matrices is decomposed into eigenvalues which is used to determine appropriate number of communities. Plotting out eigenvalues allow us to see where the elbow point is. The curve starting from the elbow point flatten out. The number of points above (excluding) the elbow point indicates number of communities.

`mfrow` determines the arrangement of multiple plots. It takes the form of `c(nr, nc)` with the first parameter being the number of rows and the second parameter being the number of columns. When deciding parameters for `mfrow`, one should take into considerations size of the plotting device and number of plots. For example, there are 20 plots, `mfrow` can be set to `c(4, 5)` or `c(2, 10)` depending on the size and shape of the plotting area.

**Value**

a pdf file in the working directory containing all eigenvalue plots

**References**

- Fushing, H., & McAssey, M. P. (2010). Time, temperature, and data cloud geometry. *Physical Review E*, 82(6), 061110.
- Chen, C., & Fushing, H. (2012). Multiscale community geometry in a network and its application. *Physical Review E*, 86(4), 041120.
- Fushing, H., Wang, H., VanderWaal, K., McCowan, B., & Koehl, P. (2013). Multi-scale clustering by building a robust and self correcting ultrametric topology on data points. *PloS one*, 8(2), e56259.

**See Also**

[plotCLUSTERS](#), [getEnsList](#)

**Examples**

```

symmetricMatrix <- as.symmetricAdjacencyMatrix(monkeyGrooming, weighted = TRUE, rule = "weak")
Sim <- as.SimilarityMatrix(symmetricMatrix)
temperatures <- temperatureSample(start = 0.01, end = 20, n = 20, method = 'random')
## Not run:
# for illustration only. skip CRAN check because it ran forever.
Ens_list <- getEnsList(Sim, temperatures, MaxIt = 1000, m = 5)

## End(Not run)

plotMultiEigenvalues(Ens_list = Ens_list, mfrow = c(10, 2), mar = c(1, 1, 1, 1))

```

---

plotTheCluster	<i>generate tree plots for selected ensemble matrix plotTrees plot one cluster tree</i>
----------------	---

---

**Description**

generate tree plots for selected ensemble matrix plotTrees plot one cluster tree

**Usage**

```
plotTheCluster(EnsList, index, ...)
```

**Arguments**

EnsList	a list in which elements are ensemble matrices.
index	an integer. index of which ensemble matrix you want to plot.
...	plotting parameters passed to <a href="#">par</a> .

**Value**

a tree plot

---

temperatureSample	<i>generate temperatures temperatureSample generate temperatures based on either random or fixed intervals</i>
-------------------	--

---

**Description**

generate temperatures temperatureSample generate temperatures based on either random or fixed intervals

**Usage**

```
temperatureSample(start = 0.01, end = 20, n = 20,  
  method = "random")
```

**Arguments**

start	a numeric vector of length 1, indicating the lowest temperature
end	a numeric vector of length 1, indicating the highest temperature
n	an integer between 10 to 30, indicating the number of temperatures (more explanations on what temperatures are).
method	a character vector indicating the method used in selecting temperatures. It should take either 'random' or 'fixedInterval', case-sensitive.

**Details**

In using random walks to find community structure, each normalized similarity matrix is evaluated at different temperatures. This allows greater variations in the normalized similarity matrices. It is recommended to try out 20 - 30 temperatures to allow for a thorough exploration of the matrices. A range of temperatures which lead to stable community structures should be considered as reliable. The temperature in the middle of the range should be selected.

**Value**

a numeric vector of length n representing temperatures sampled.

**References**

Fushing, H., & McAssey, M. P. (2010). Time, temperature, and data cloud geometry. *Physical Review E*, 82(6), 061110.

Chen, C., & Fushing, H. (2012). Multiscale community geometry in a network and its application. *Physical Review E*, 86(4), 041120.

Fushing, H., Wang, H., VanderWaal, K., McCowan, B., & Koehl, P. (2013). Multi-scale clustering by building a robust and self correcting ultrametric topology on data points. *PloS one*, 8(2), e56259.

**See Also**

[getEnList](#)

**Examples**

```
symmetricMatrix <- as.symmetricAdjacencyMatrix(monkeyGrooming, weighted = TRUE, rule = "weak")  
Sim <- as.SimilarityMatrix(symmetricMatrix)  
temperatures <- temperatureSample(start = 0.01, end = 20, n = 20, method = 'random')
```

# Index

## \* datasets

- monkeyGrooming, [7](#)
  
- as.SimilarityMatrix, [2](#)
- as.symmetricAdjacencyMatrix, [2](#)
  
- getEigenvalueList, [3](#)
- getEns, [4](#)
- getEnsList, [5](#), [8](#), [9](#), [11](#)
- GetSim, [6](#)
  
- monkeyGrooming, [7](#)
  
- par, [7](#), [9](#), [10](#)
- plotCLUSTERS, [7](#), [9](#)
- plotMultiEigenvalues, [8](#)
- plotTheCluster, [10](#)
  
- temperatureSample, [6](#), [10](#)