Package ‘TNC’

January 20, 2025

Type Package

Title Temporal Network Centrality (TNC) Measures

Version 0.1.0

Description
Node centrality measures for temporal networks. Available measures are temporal degree centrality, temporal closeness centrality and temporal betweenness centrality defined by Kim and Anderson (2012) <doi:10.1103/PhysRevE.85.026107>. Applying the REN algorithm by Hanke and Foraita (2017) <doi:10.1186/s12859-017-1677-x> when calculating the centrality measures keeps the computational running time linear in the number of graph snapshots. Further, all methods can run in parallel up to the number of nodes in the network.

Depends R (>= 3.4.1)

License GPL-3

Encoding UTF-8

LazyData true

Suggests igraph (>= 1.1.2), parallel (>= 3.4.1), testthat (>= 1.0.2)

RoxygenNote 6.0.1

NeedsCompilation no

Author Moritz Hanke [aut, cre]

Maintainer Moritz Hanke <hanke@leibniz-bips.de>

Repository CRAN

Date/Publication 2017-09-06 13:06:13 UTC

Contents

tbc *Temporal betweenness centrality*

Description

tbc returns the temporal betweenness centrality for each node in a dynamic network (sequence of graph snapshots).

Usage

```
tbc(x, type = NULL, startsnapshot = 1, endsnapshot = length(x),
  vertexindices = NULL, directed = FALSE, normalize = TRUE,
  centrality_evolution = FALSE)
```

Arguments

x	A list of adjacency matrices or a list of adjacency lists.
type	Data format of x. Possible formats are "M" for a list of adjacency matrices (containing only 1s and 0s) and "L" for a list of adjacency lists (adjacency lists of the igraph package are supported). Default is NULL.
startsnapshot	Numeric. Entry of x to start the calculation of tbc. Default is 1.
endsnapshot	Numeric. Entry of x to end the calculation of tbc. Default is the last element of x.
vertexindices	Numeric. A vector of nodes. Only shortest temporal paths ending at nodes in vertexindices are considered for calculating tbc. Can be used to parallel the calculation of tbc (see section Examples). Default is NULL.
directed	Logical. Set TRUE if the dynamic network is a directed network. Default is FALSE.
normalize	Logical. Set TRUE if centrality values should be normalized with $1/((|V|-1)*(|V|-2)*m)$ where $|V|$ is the number of nodes and $m =$ endsnapshot $-$ startsnapshot. Default is TRUE.
centrality_evolution	Logical. Set TRUE if an additional matrix should be returned containing the centrality values at each snapshot. Rows correspondent to nodes, columns correspondent to snapshots. Default is FALSE.

Details

tbc calculates the temporal betweenness centrality (Kim and Anderson, 2012). To keep the computational effort linear in the number of snapshots the Reversed Evolution Network algorithm (REN; Hanke and Foraita, 2017) is used to find all shortest temporal paths.

Value

The (normalized) temporal betweenness centrality (`TBC`) values of all nodes. If `centrality_evolution` is `TRUE`, an additional matrix will be returned (`CentEvo`), containing the temporal $(|V|xT)$ matrix is returned (`CentEvo`), containing the temporal centrality value at each snapshot between `startsnapshot` and `endsnapshot`.

Warning

Using adjacency matrices as input exponentially increases the required memory. Use adjacency lists to save memory.

References

Kim, Hyoungshick and Anderson, Ross (2012). *Temporal node centrality in complex networks*. Physical Review E, 85 (2).

Hanke, Moritz and Foraita, Ronja (2017). *Clone temporal centrality measures for incomplete sequences of graph snapshots*. BMC Bioinformatics, 18 (1).

See Also

`tcc`,`tdc`

Examples

```
# Create a list of adjacency matrices, plot the corresponding graphs
# (using the igraph package) and calculate tbc

A1 <- matrix(c(0,1,0,0,0,0,
               1,0,1,0,0,0,
               0,1,0,0,0,0,
               0,0,0,0,0,0,
               0,0,0,0,0,0,
               0,0,0,0,0,0), ncol=6)

A2 <- matrix(c(0,0,0,0,0,0,
               0,0,1,0,0,0,
               0,1,0,1,1,0,
               0,0,1,0,0,0,
               0,0,1,0,0,0,
               0,0,0,0,0,0), ncol=6)

A3 <- matrix(c(0,0,0,0,0,0,
               0,0,0,0,0,0,
               0,0,0,0,0,0,
               0,0,0,0,0,0,
               0,0,0,0,0,0,
               0,0,0,0,0,0), ncol=6)

A4 <- matrix(c(0,1,0,0,0,0,
               1,0,0,1,0,0,
               0,0,0,0,0,0,
              0,1,0,0,0,0,
              0,0,0,0,0,0,
              0,0,0,0,0,0), ncol=6)

library(igraph)
par(mfrow=c(2,2))

Layout <-
 layout_in_circle(graph_from_adjacency_matrix(A1, mode = "undirected"))

plot(graph_from_adjacency_matrix(A1, "undirected"), layout=Layout)
plot(graph_from_adjacency_matrix(A2, "undirected"), layout=Layout)
plot(graph_from_adjacency_matrix(A3, "undirected"), layout=Layout)
plot(graph_from_adjacency_matrix(A4, "undirected"), layout=Layout)

As <- list(A1,A2,A3,A4)

tbc(As, "M", centrality_evolution=TRUE)

### Create list of adjacency lists
Ls <- lapply(seq_along(As), function(i){
  sapply(1:6, function(j){which(As[[i]][j,]==1)})
})

tbc(Ls, "L", centrality_evolution=TRUE)

### Run tbc in parallel ###
library(parallel)
# Calculate the number of cores
cores_avail <- detectCores()-1
# Initiate cluster
cl <- makeCluster(2)
clusterExport(cl, c("As", "tbc"))

TBC <- parLapply(cl, 1:6, function(x){
  tbc(As, "M", vertexindices = x)
 }
)

stopCluster(cl)

Reduce("+", TBC)
```

tcc *Temporal closeness centrality*

Description

tcc returns the temporal closeness centrality for each node in a dynamic network (sequence of graph snapshots).

Usage

```
tcc(x, type = NULL, startsnapshot = 1, endsnapshot = length(x),
  vertexindices = NULL, directed = FALSE, normalize = TRUE,
  centrality_evolution = FALSE)
```

Arguments

`x`	A list of adjacency matrices or a list of adjacency lists.
`type`	Data format of `x`. Possible formats are `"M"` for a list of adjacency matrices (containing only 1s and 0s) and `"L"` for a list of adjacency lists (adjacency lists of the igraph package are supported). Default is `NULL`.
`startsnapshot`	Numeric. Entry of `x` to start the calculation of `tcc`. Default is 1.
`endsnapshot`	Numeric. Entry of `x` to end the calculation of `tcc`. Default is the last element of `x`.
`vertexindices`	Numeric. A vector of nodes. Only shortest temporal paths ending at nodes in `vertexindices` are considered for calculating `tcc`. Can be used to parallel the calculation of `tcc` (see section Examples). Default is `NULL`.
`directed`	Logical. Set `TRUE` if the dynamic network is a directed network. Default is `FALSE`.
`normalize`	Logical. Set `TRUE` if centrality values should be normalized with $1/((\|V\| - 1) * m)$ where $\|V\|$ is the number of nodes and $m =$ `endsnapshot` $-$ `startsnapshot`. Default is `TRUE`.
`centrality_evolution`	Logical. Set `TRUE` if an additional matrix should be returned containing the centrality values at each snapshot. Rows correspondent to nodes, columns correspondent to snapshots. Default is `FALSE`.

Details

`tcc` calculates the temporal closeness centrality (Kim and Anderson, 2012). To keep the computational effort linear in the number of snapshots the Reversed Evolution Network algorithm (REN; Hanke and Foraita, 2017) is used to find all shortest temporal paths.

Value

The (normalized) temporal betweenness centrality values of all nodes (`TCC`). If `centrality_evolution` is `TRUE`, an additional $(|V|xT)$ matrix is returned (`CentEvo`), containing the temporal centrality value at each snapshot between `startsnapshot` and `endsnapshot`.

Warning

Using adjacency matrices as input exponentially increases the required memory. Use adjacency lists to save memory.

References

Kim, Hyoungshick and Anderson, Ross (2012). *Temporal node centrality in complex networks*. Physical Review E, 85 (2).

Hanke, Moritz and Foraita, Ronja (2017). *Clone temporal centrality measures for incomplete sequences of graph snapshots*. BMC Bioinformatics, 18 (1).

See Also

tbc,tdc

Examples

```
# Create a list of adjacency matrices, plot the corresponding graphs
# (using the igraph package) and calculate tcc

A1 <- matrix(c(0,1,0,0,0,0,
               1,0,1,0,0,0,
               0,1,0,0,0,0,
               0,0,0,0,0,0,
               0,0,0,0,0,0,
               0,0,0,0,0,0), ncol=6)

A2 <- matrix(c(0,0,0,0,0,0,
               0,0,1,0,0,0,
               0,1,0,1,1,0,
               0,0,1,0,0,0,
               0,0,1,0,0,0,
               0,0,0,0,0,0), ncol=6)

A3 <- matrix(c(0,0,0,0,0,0,
               0,0,0,0,0,0,
               0,0,0,0,0,0,
               0,0,0,0,0,0,
               0,0,0,0,0,0,
               0,0,0,0,0,0), ncol=6)

A4 <- matrix(c(0,1,0,0,0,0,
               1,0,0,1,0,0,
               0,0,0,0,0,0,
               0,1,0,0,0,0,
               0,0,0,0,0,0,
               0,0,0,0,0,0), ncol=6)

library(igraph)
par(mfrow=c(2,2))

Layout <-
 layout_in_circle(graph_from_adjacency_matrix(A1, mode = "undirected"))

plot(graph_from_adjacency_matrix(A1, "undirected"), layout=Layout)
plot(graph_from_adjacency_matrix(A2, "undirected"), layout=Layout)
plot(graph_from_adjacency_matrix(A3, "undirected"), layout=Layout)
plot(graph_from_adjacency_matrix(A4, "undirected"), layout=Layout)

As <- list(A1,A2,A3,A4)

tcc(As, "M", centrality_evolution=TRUE)

### Create list of adjacency lists
Ls <- lapply(seq_along(As), function(i){
  sapply(1:6, function(j){which(As[[i]][j,]==1)})
})

tcc(Ls, "L", centrality_evolution=TRUE)

### Run tbc in parallel ###
library(parallel)
# Calculate the number of cores
cores_avail <- detectCores()-1
# Initiate cluster
cl <- makeCluster(2)
clusterExport(cl, c("As", "tcc"))

TCC <- parLapply(cl, 1:6, function(x){
  tcc(As, "M", vertexindices = x)
 }
)

stopCluster(cl)

Reduce("+", TCC)
```

tdc *Temporal degree centrality*

Description

`tdc` returns the temporal degree centrality for each node in a dynamic network (sequence of graph snapshots).

Usage

```
tdc(x, type = NULL, startsnapshot = 1, endsnapshot = length(x),
  directed = FALSE, normalize = TRUE, centrality_evolution = FALSE)
```

Arguments

`x`	A list of adjacency matrices or a list of adjacency lists.
`type`	Data format of `x`. Possible formats are `"M"` for a list of adjacency matrices (containing only 1s and 0s) and `"L"` for a list of adjacency lists (adjacency lists of the igraph package are supported). Default is `NULL`.

`startsnapshot`	Numeric. Entry of `x` to start the calculation of `tdc`. Default is 1.
`endsnapshot`	Numeric. Entry of `x` to end the calculation of `tdc`. Default is the last element of `x`.
`directed`	Logical. Set `TRUE` if the temporal network is a directed network. Default is `FALSE`.
`normalize`	Logical. Set `TRUE` if centrality values should be normalized with $1/((\|V\|-1)*m)$ where $\|V\|$ is the number of nodes and $m = \texttt{endsnapshot} - \texttt{startsnapshot}$. Default is `TRUE`.
`centrality_evolution`	Logical. Set `TRUE` if an additional matrix should be returned containing the centrality values at each snapshot. Rows correspondent to nodes, columns correspondent to snapshots. Default is `FALSE`.

Details

`tdc` calculates the temporal degree centrality (see Kim and Anderson, 2012), which is defined as the average degree centrality over all snapshots.

Value

The (normalized) temporal degree centrality values of all nodes (`TDC`). If `centrality_evolution` is `TRUE` an additional matrix is returned (`CentEvo`), containing the temporal centrality value at each snapshot between `startsnapshot` and `endsnapshot`.

Warning

Using adjacency matrices as input exponentially increases the required memory. Use adjacency lists to save memory.

References

Kim, Hyoungshick and Anderson, Ross, 2012. *Temporal node centrality in complex networks*. Physical Review E, 85 (2).

See Also

`tbc`,`tcc`

Examples

```
# Create a list of adjacency matrices, plot the corresponding graphs
# (using the igraph package) and calculating tdc

A1 <- matrix(c(0,1,0,0,0,0,
               1,0,1,0,0,0,
               0,1,0,0,0,0,
               0,0,0,0,0,0,
               0,0,0,0,0,0,
               0,0,0,0,0,0), ncol=6)
```

```
A2 <- matrix(c(0,0,0,0,0,0,
               0,0,1,0,0,0,
               0,1,0,1,1,0,
               0,0,1,0,0,0,
               0,0,1,0,0,0,
               0,0,0,0,0,0), ncol=6)

A3 <- matrix(c(0,0,0,0,0,0,
               0,0,0,0,0,0,
               0,0,0,0,0,0,
               0,0,0,0,0,0,
               0,0,0,0,0,0,
               0,0,0,0,0,0), ncol=6)

A4 <- matrix(c(0,1,0,0,0,0,
               1,0,0,1,0,0,
               0,0,0,0,0,0,
               0,1,0,0,0,0,
               0,0,0,0,0,0,
               0,0,0,0,0,0), ncol=6)

library(igraph)
par(mfrow=c(2,2))

Layout <-
 layout_in_circle(graph_from_adjacency_matrix(A1, mode = "undirected"))

plot(graph_from_adjacency_matrix(A1, "undirected"), layout=Layout)
plot(graph_from_adjacency_matrix(A2, "undirected"), layout=Layout)
plot(graph_from_adjacency_matrix(A3, "undirected"), layout=Layout)
plot(graph_from_adjacency_matrix(A4, "undirected"), layout=Layout)

As <- list(A1,A2,A3,A4)

tdc(As, "M", centrality_evolution=TRUE)

#' ### Create list of adjacency lists
Ls <- lapply(seq_along(As), function(i){
  sapply(1:6, function(j){which(As[[i]][j,]==1)})
})

tdc(Ls, "L", centrality_evolution=TRUE)
```

`TNC` *TNC: A package for computing temporal network centrality values for nodes of a dynamic network.*

Description

The TNC package provides three functions: tbc, tcc and tdc for calculating temporal betweenness, temporal closeness and temporal degree centrality.

Index