Package 'mFLICA'

June 11, 2024

Title Leadership-Inference Framework for Multivariate Time Series

Version 0.1.6

Description A leadership-inference framework for multivariate time series. The framework for multiple-faction-leadership inference from coordinated activities or 'mFLICA' uses a notion of a leader as an individual who initiates collective patterns that everyone in a group follows. Given a set of time series of individual activities, our goal is to identify periods of coordinated activity, find factions of coordination if more than one exist, as well as identify leaders of each faction. For each time step, the framework infers following relations between individual time series, then identifying a leader of each faction whom many individuals follow but it follows no one. A faction is defined as a group of individuals that everyone follows the same leader. 'mFLICA' reports following relations, leaders of factions, and members of each faction for each time step. Please see Chainarong Amornbunchornvej and Tanya Berger-Wolf (2018) <doi:10.1137/1.9781611975321.62> for methodology and Chainarong Amornbunchornvej (2021) <doi:10.1016/j.softx.2021.100781> for software when referring to this package in publications.

License GPL-3

URL https://github.com/DarkEyes/mFLICA

BugReports https://github.com/DarkEyes/mFLICA/issues

Language en-US

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0)

Imports ggplot2 (>= 3.0)

Suggests knitr, rmarkdown, R.rsp

VignetteBuilder knitr, R.rsp

RoxygenNote 7.3.1

NeedsCompilation no

Author Chainarong Amornbunchornvej [aut, cre]
(<https://orcid.org/0000-0003-3131-0370>),
Namrata Banerji [ctb]

Maintainer Chainarong Amornbunchornvej <grandca@gmail.com>

Repository CRAN

Date/Publication 2024-06-11 08:10:02 UTC

Contents

`followingNetwork` *followingNetwork function*

Description

followingNetwork is a support function for calculating a following network of a set of time series

Usage

```
followingNetwork(TS, timeLagWindow, lagWindow = 0.1, sigma = 0.1)
```

Arguments

`TS`	is a set of time series where `TS[i,t,d]` is a numeric value of `i`th time series at time `t` and dimension `d`.
`timeLagWindow`	is a maximum possible time delay in the term of time steps.
`lagWindow`	is a maximum possible time delay in the term of percentage of time length of `TS`.
`sigma`	is a threshold of following relation. It is used to discretize an adjacency matrix `adjWeightedMat` to be a binary matrix `adjBinMat`.

Value

This function returns adjacency matrices of a following network of TS.

adjWeightedMat	An adjacency matrix of a following network s.t. if adjWeightedMat[i,j]>0, then TS[i,,] follows TS[j,,] with a degree adjWeightedMat[i,j].
adjBinMat	A binary version of adjWeightedMat s.t. adjBinMat[i,j] <- (adjWeightedMat[i,j] >=sigma) for any i,j.

Examples

```
# Run the function ?

out<-followingNetwork(TS=mFLICA::TS[,60:90,],sigma=0.5)
```

---

followingRelation *followingRelation*

Description

03/24/2021: Chai's code rewritten by Namrata to replicate Matlab version The changes are in the DTW function, which is implemented here as DTW2 instead of the one in the R package DTW

followingRelation is a function that infers whether Y follows X.

Usage

```
followingRelation(Y, X, timeLagWindow, lagWindow = 0.1)
```

Arguments

Y	is a T-by-D matrix of numerical time series of a follower
X	is a T-by-D matrix numerical time series of a leader
timeLagWindow	is a maximum possible time delay in the term of time steps.
lagWindow	is a maximum possible time delay in the term of percentage of length(X). If timeLagWindow is missing, then timeLagWindow=ceiling(lagWindow*length(X)). The default is 0.2.

Value

This function returns a list of following relation variables below.

follVal	is a following-relation value s.t. if follVal is positive, then Y follows X. If follVal is negative, then X follows Y. Otherwise, if follVal is zero, there is no following relation between X,Y.
dtwIndexVec	is a numeric vector of index-warping difference: dtwIndexVec[k] = dtwOut$index1[k] - dtwOut$index2[k] where dtwOut is the output from dtw::dtw(x=Y,y=X) function.

Examples

```
# Load example data ???

leader<- mFLICA::TS[1,1:200,]
follower<- mFLICA::TS[2,1:200,]

# Run the function

out<-followingRelation(Y=follower,X=leader)
```

---

getADJNetDen *getADJNetDen function*

Description

getADJNetDen is a support function for calculating a network density of a network.

Usage

```
getADJNetDen(adjMat)
```

Arguments

adjMat	is an adjacency matrix of a dominant-distribution network.

Value

This function returns a value of network density of of a network for a given adjMat.

Examples

```
# Given an example of adjacency matrix
A<-matrix(FALSE,5,5)
A[2,1]<-TRUE
A[c(3,4),2]<-TRUE

# Get a network density of an adjacency matrix

getADJNetDen(adjMat=A)
```

getDynamicFollNet *getDynamicFollNet function*

Description

getDynamicFollNet is a support function for calculating a dynamic following network of a set of time series

Usage

```
getDynamicFollNet(
  TS,
  timeWindow,
  timeShift,
  sigma = 0.5,
  lagWindow = 0.1,
  silentFlag = FALSE
)
```

Arguments

TS	is a set of time series where TS[i,t,d] is a numeric value of ith time series at time t and dimension d.
timeWindow	is a time window parameter that limits a length of each sliding window. The default is 10 percent of time series length.
timeShift	is a number of time steps a sliding window shifts from a previous window to the next one. The default is 10 percent of timeWindow.
sigma	is a threshold of following relation. The default is 0.5.
lagWindow	is a maximum possible time delay in the term of percentage of time length of timeWindow supplying to the followingNetwork function.
silentFlag	is a flag that prohibit the function to print the current status of process.

Value

This function returns adjacency matrices of a dynamic following network of TS as well as the corresponding time series of network densities.

dyNetWeightedMat
: An adjacency matrix of a dynamic following network s.t. if dyNetWeightedMat[i,j,t]>0, then TS[i,,] follows TS[j,,] at time t with a degree dyNetWeightedMat[i,j,t].

dyNetBinMat
: A binary version of dyNetWeightedMat s.t. dyNetWeightedMat[i,j,t] <- (dyNetWeightedMat[i,j,t] >=sigma) for any i,j,t.

dyNetWeightedDensityVec
: A time series of dynamic network densities of dyNetWeightedMat

dyNetBinDensityVec
: A time series of dynamic network densities of dyNetBinDensityVec

Examples

```
# Run the function
out<-getDynamicFollNet(TS=mFLICA::TS[,1:10,],timeWindow=5,timeShift = 5,sigma=0.5)
```

`getFactions` *getFactions function*

Description

getFactions is a support function for inferring faction leaders and their members as well as a faction size ratio of each faction. Leaders are nodes that have zero outgoing degree. Members of leader A's faction are nodes that have some directed path to A in a following network.

Usage

```
getFactions(adjMat)
```

Arguments

`adjMat`	is an adjacency matrix of a following network.

Value

This function returns a list of leader IDs, a list of faction members, and network densities of factions.

`leaders`	is a list of faction leader IDs
`factionMembers`	is a list of members of factions where `factionMembers[[i]]` is a list of faction members of a leader `leaders[i]`'s faction.
`factionSizeRatio`	is a vector of faction size ratio of each faction. `factionSizeRatio[i]` is a number of edges within a leader `leaders[i]`'s faction divided by N choose 2 where N is a number of all nodes.

Examples

```
# Given an example of adjacency matrix
A<-matrix(FALSE,5,5)
A[2,1]<-TRUE
A[c(3,4),2]<-TRUE
A[5,3]<-TRUE
# Get faction leaders and their members as well as a network density of each faction.

out<-getFactions(adjMat=A)
```

getFactionSizeRatio *getFactionSizeRatio function*

Description

getFactionSizeRatio is a support function for calculating a faction size ratio of a given faction. A faction size ratio is a number of edges that connect between faction-member nodes divided by a number of total nodes within a following network.

Usage

```
getFactionSizeRatio(adjMat, members)
```

Arguments

adjMat	is an adjacency matrix of a dominant-distribution network.
members	is a list of member IDs of a given faction.

Value

This function returns a faction size ratio of a given faction.

Examples

```
# Given an example of adjacency matrix
A<-matrix(FALSE,5,5)
A[2,1]<-TRUE
A[c(3,4),2]<-TRUE

# Get a faction size ratio of a given faction

getFactionSizeRatio(adjMat=A,members=c(1,2,3,4))
```

getReachableNodes *getReachableNodes function*

Description

getReachableNodes is a support function for inferring reachable nodes that have some directed path to a node targetNode. This function uses Breadth-first search (BFS) algorithm.

Usage

```
getReachableNodes(adjMat, targetNode)
```

Arguments

adjMat	is an adjacency matrix of a following network of which its elements are binary: zero for no edge, and one for having an edge.
targetNode	is a node in a graph that we want to find a set of nodes that can reach this target node via some paths.

Value

This function returns a set of node IDs `followers` that have some directed path to a node `targetNode`.

Examples

```
# Given an example of adjacency matrix
A<-matrix(FALSE,5,5)
A[2,1]<-TRUE
A[c(3,4),2]<-TRUE
A[5,3]<-TRUE
# Get a set of reachable nodes of targetNode.

followers<-getReachableNodes(adjMat=A,targetNode=1)$followers
```

---

mFLICA *mFLICA: leadership-inference framework for multivariate time series*

Description

A leadership-inference framework for multivariate time series. The framework uses a notion of a leader as an individual who initiates collective patterns that everyone in a group follows. Given a set of time series of individual activities, our goal is to identify periods of coordinated activity, find factions of coordination if more than one exist, as well as identify leaders of each faction. For each time step, the framework infers following relations between individual time series, then identifying a leader of each faction whom many individuals follow but it follows no one. A faction is defined as a group of individuals that everyone follows the same leader. mFLICA reports following relations, leaders of factions, and members of each faction for each time step. Please see Chainarong Amornbunchornvej and Tanya Berger-Wolf (2018) <doi:10.1137/1.9781611975321.62> when referring to this package in publications.

Usage

```
mFLICA(
  TS,
  timeWindow,
  timeShift,
  lagWindow = 0.1,
  sigma = 0.5,
  silentFlag = FALSE
)
```

Arguments

TS	is a set of time series where TS[i,t,d] is a numeric value of ith time series at time t and dimension d.
timeWindow	is a time window parameter that limits a length of each sliding window. The default is 10 percent of time series length.
timeShift	is a number of time steps a sliding window shifts from a previous window to the next one. The default is 10 percent of timeWindow.
lagWindow	is a maximum possible time delay in the term of percentage of time length of timeWindow supplying to the getDynamicFollNet function.
sigma	is a threshold of following relation. The default is 0.5. Note that if sigma is not one, an individual might be a member of multiple factions.
silentFlag	is a flag that prohibit the function to print the current status of process.

Value

This function returns dynamic following networks, as well as leaders of factions, and members of each faction for each time step.

dyNetOut$dyNetWeightedMat
: An adjacency matrix of a dynamic following network s.t. if dyNetWeightedMat[i,j,t]>0, then TS[i,,] follows TS[j,,] at time t with a degree dyNetWeightedMat[i,j,t].

dyNetOut$dyNetBinMat
: A binary version of dyNetWeightedMat s.t. dyNetWeightedMat[i,j,t] <- (dyNetWeightedMat[i,j,t] >=sigma) for any i,j,t.

dyNetOut$dyNetWeightedDensityVec
: A time series of dynamic network densities of dyNetWeightedMat

dyNetOut$dyNetBinDensityVec
: A time series of dynamic network densities of dyNetBinDensityVec

leadersTimeSeries
: A time series of leaders of each faction where leadersTimeSeries[[t]] is a set of leaders at time t. A number of factions is the same as a number of leaders.

factionMembersTimeSeries
: A time series of sets of faction members where factionMembersTimeSeries[[t]][[k]] is a set of faction-members at time t leading by a leader leadersTimeSeries[[t]][k].

factionSizeRatioTimeSeries
: A time series of faction-size ratios of all individuals. A faction size ratio is a number of edges that connect between faction-member nodes divided by a number of total nodes within a following network. If a leader has a higher faction-size ratio, then it has more followers than a leader with a lower faction-size ratio. A faction-size ratio has a value between 0 and 1.

Author(s)

Chainarong Amornbunchornvej, <chai@ieee.org>

Examples

```
# Run the function

obj1<-mFLICA(TS=mFLICA::TS[,60:90,],timeWindow=10,timeShift=10,sigma=0.5)

# Plot time series of faction size ratios of all leaders

plotMultipleTimeSeries(TS=obj1$factionSizeRatioTimeSeries, strTitle="Faction Size Ratios")
```

`plotMultipleTimeSeries`

plotMultipleTimeSeries

Description

plotMultipleTimeSeries is a function for visualizing time series

Usage

```
plotMultipleTimeSeries(TS, strTitle = "Time Series Plot", TSnames)
```

Arguments

`TS`	is a set of time series where `TS[i,t,d]` is a numeric value of `i`th time series at time `t` and dimension `d`.
`strTitle`	is a string of the plot title
`TSnames`	is a list of legend of `X,Y` where TSnames[1] is a legend of `X` and TSnames[2] is a legend of `Y`.

Value

This function returns an object of ggplot class.

Examples

```
# Run the function
plotMultipleTimeSeries(TS=mFLICA::TS[1:5,1:60,1])
```

TS *A simulation time series of movement coordination of 30 individuals*

Description

A dataset containing simulated trajectories of 30 individuals moving to form coordination in x-y coordinates. In the interval [1,200], ID1 leads the group and everyone follows. ID2 leads the group during the interval [201,400]. Lastly, ID3 leads the group during the interval [401,600]. The interval [601,800] is the time when everyone trying to stop moving.

Usage

```
TS
```

Format

An array with 30 rows of individuals, 800 columns of time steps, and 2 dimensions of coordinate (x,y):

TS It is a set of time series where `TS[i,t,d]` is a numeric value of `i`th time series at time `t` and dimension `d`. ...

TSNANNearestNeighborPropagation *TSNANNearestNeighborPropagation*

Description

TSNANNearestNeighborPropagation is a function that fills NA values with nearest real values in the past (or future if the first position of time series is NA), for time series `X`.

Usage

```
TSNANNearestNeighborPropagation(X)
```

Arguments

`X`	is a T-by-D matrix numerical time series

Value

This function returns a list of following relation variables below.

`Xout`	is a T-by-D matrix numerical time series that all NAN have been filled with nearest real values.

Examples

```
# Load example data

z<-1:20
z[2:5]<-NA
z<-TSNANNearestNeighborPropagation(z)
```

Index