

# mFLICA: an R package for inferring leadership of coordination from time Series

Chainarong Amornbunchornvej

National Electronics and Computer Technology Center

---

## Abstract

Leadership is a process that leaders influence followers to achieve collective goals. One of special cases of leadership is the coordinated pattern initiation. In this context, leaders are initiators who initiate coordinated patterns that everyone follows. Given a set of individual-multivariate time series of real numbers, the **mFLICA** package provides a framework for R users to infer coordination events within time series, initiators and followers of these coordination events, as well as dynamics of group merging and splitting. The **mFLICA** package also has a visualization function to make results of leadership inference more understandable. The package is available on Comprehensive R Archive Network (CRAN) at <https://CRAN.R-project.org/package=mFLICA>.

*Keywords:* leadership, time series, coordination, pattern initiation.

---

## 1. Introduction

Leadership is defined as a process that leaders influence a group to achieve collective goals Hogg (2001); Glowacki and von Rueden (2015). In time series context, one of leadership definitions is pattern initiation. Leaders are initiators who initiate collective patterns (e.g. movement initiation, trends of stock closing prices) that everyone follows Amornbunchornvej, Brugere, Strandburg-Peshkin, Farine, Crofoot, and Berger-Wolf (2018). Collective patterns or *coordination events* are emerging events of collective actions that aim to reach collective goals Malone and Crowston (1994). In time series context, coordination events occur when there exists some intervals such that some similar pattern occurs in all time series with possible different time delay for each time series Amornbunchornvej *et al.* (2018). A *leader* of coordination event is a time series that initiates the pattern before others having this similar pattern with arbitrary time delays.

Leadership literature in time series started in movement context. Leaders are defined as individuals who move in a front of a group in the flock model Andersson, Gudmundsson, Laube, and Wolle (2008). Afterwards, the work by Kjargaard, Blunck, Wustenberg, Gronbask, Wirz, Roggen, and Troster (2013) defined leaders as an individual who has the highest number of followers. The work defined that time series  $A$  follows  $B$  if a distance between a pattern of  $A$  in current time and a pattern of  $B$  in the past is small w.r.t. some time series distance measures, such as Dynamic Time Warping (DTW) Sakoe and Chiba (1978), cross-correlation Bracewell (1965), etc. However, the work was able to infer leaders only on a single coordination event and was unable to automatically detect coordination events. Then, the leadership inference framework, FLICA, was proposed by Amornbunchornvej *et al.* (2018)

to infer leaders and their followers within time series that have multiple coordination events. The mFLICA framework [Amornbunchornvej and Berger-Wolf \(2018\)](#) was proposed to eliminate the limitation of FLICA. mFLICA can infer leaders and their followers within multiple coordination events that occur simultaneously, while FLICA cannot. This paper provides the details of **mFLICA** as an R package.

To infer a following relation between time series, DTW can be deployed for inferring a similar pattern between two time series. The **dtw** package [Giorgino \(2009\)](#) is deployed in **mFLICA** for DTW computation.

The related concepts of leadership inference on time series are Granger causality [Granger \(1969\)](#) and Transfer Entropy [Schreiber \(2000\)](#); [Behrendt, Dimpfl, Peter, and Zimmermann \(2019\)](#). Both techniques can be used to infer whether time series  $A$  is a predictor of time series  $B$ , which is similar to the following relation concept in leadership inference. Nevertheless, leadership inference aims to identify patterns that distributes among time series and their initiators (leaders) rather than finding predictors. In Comprehensive R Archive Network (CRAN), Granger causality framework can be found in **Imtest** package [Zeileis and Hothorn \(2002\)](#), and transfer entropy is in **RTransferEntropy** package [Behrendt et al. \(2019\)](#).

In this paper, the details of **mFLICA** package in R statistical software [R Core Team \(2020\)](#) are provided. The current version of **mFLICA** is 0.1.1 on CRAN [Amornbunchornvej \(2020\)](#). **mFLICA** is a framework that is capable of:

- **Inferring coordination events:** the framework can infer and visualize coordination intervals that have high degrees of coordination; and
- **Inferring dynamics of leaders and followers:** the framework can infer leaders of coordination and their initiators that can be changed over time.

This paper is organized as follows. In Section 2, details of important definitions in leadership of coordination are provided. Section 3 provides the overview of **mFLICA** package architecture. Section 4 gives algorithms that are implemented in **mFLICA** as well as examples of how to use **mFLICA** for inferring leadership of coordination (e.g. factions, leadership dynamics) from a set of time series. Lastly, Section 5 provides the conclusion.

## 2. Definitions

In this paper, a time series is referred as  $T_i = (T_i(1), \dots, T_i(n))$  where  $T_i(t) \in \mathbb{R}^d$  is an element of time series  $T_i$  at time  $t$ , which is a  $d$ -dimensional vector of real numbers. A  $\Delta$ -time-shift  $T_i^\Delta$  is a time series  $T_i$  s.t.  $T_i^\Delta(t + \Delta) = T_i(t)$ . The time series  $T_i^\Delta$  can be considered as time series  $T_i$  that values are shifted forward into future.

Given two time series, the first building block of leadership in time series is the following relation that defines time series to be either a leader or follower.

**Definition 1 (Following relation)** *Given two  $d$ -dimensional time series  $T_1$  and  $T_2$ , a similarity measure of time series  $\text{sim}(\cdot, \cdot)$ , and a threshold  $\sigma$ . A time series  $T_1$  follows  $T_2$  if there exist  $\Delta > 0$  s.t.  $\text{sim}(T_1^\Delta, T_2) \geq \sigma$ .*

In other words, a following relation represents a relation between time series s.t. one time series initiates some pattern, then another time series follows the similar pattern with the time delay  $\Delta$ .

**Definition 2 (Following network)** *Given a set of time series  $S = \{T_1, \dots, T_m\}$ . A following network  $G = (V, E)$  is a directed acyclic graph (DAG) s.t.  $V = \{v_1, \dots, v_m\}$  represents a set of nodes where  $v_i$  is a node of time series  $T_i$ , and  $E = \{e_{i,j}\}$  is a set of edges. If  $T_i$  follows  $T_j$ , then there is a directed edge  $e_{i,j} \in E$ .*

A following network represents following relations in a form of graph. If there is a high-level of coordination that many time series follow the same pattern, then a number of edges in a following network must be high. Hence, a network density below is used as a proxy of level of coordination.

$$d(G) = \frac{|E|}{\binom{|V|}{2}}. \quad (1)$$

Where  $d(G) \in [0, 1]$ ,  $|E|, |V|$  are numbers of edges and nodes in  $G$  respectively.

In real-word situation, there are more than one pattern that several time series might follow simultaneously. We call each group of time series that follow different pattern as a faction. In a following network, a faction is a network component s.t. all nodes within a faction have directed path(s) to a faction-leader node. We provide a definition of leaders and factions below.

**Definition 3 (Faction leader)** *Given a following network  $G$ , a faction leader is a node s.t. the out-degree is zero but the in-degree is greater than zero.*

**Definition 4 (Faction members)** *Given a following network  $G$  and a faction-leader node  $L$ , members of  $L$ 's faction are nodes that have directed path(s) to  $L$  in  $G$ .*

Lastly, factions and levels of coordination can be changed overtime. To capture leadership dynamics, the dynamic following network is defined below.

**Definition 5 (Dynamic following network)** *Given a set of time series  $S = \{T_1, \dots, T_m\}$ , and a time window  $\omega$ . Let  $S_{[t_1, t_2]}$  be a revised set of  $S$  where it contains only the time interval  $[t_1, t_2]$  for each time series in  $S$ .*

*A dynamic following network  $\mathcal{G} = (G_1, \dots)$  is a sequence of following networks. At time  $t$ , a following network  $G_t$  in  $\mathcal{G}$  is a following network that is derived from  $S_{[t, t+\omega]}$ .*

By measuring network densities of following networks in  $\mathcal{G}$ , we have a time series of network density  $T_d = (d_1, \dots)$  where  $d_t$  is a network density of  $G_t$  in  $\mathcal{G}$ . The time series of network densities  $T_d$  represents a dynamic of levels of coordination.

To measure dynamics of factions or sub groups, the faction size ratio is used to measure a faction size, which is defined below:

$$f(G, v_L) = \frac{|E_L|}{\binom{|V|}{2}}. \quad (2)$$

Where  $f(G, v_L) \in [0, 1]$ ,  $G$  is a following network,  $v_L$  is a node of faction leader  $L$ , and  $E_L$  is a number of edges connected between faction members. For a dynamic following network  $\mathcal{G}$ ,

the time series of faction size ratios of leader  $L$ 's faction represents dynamics of faction sizes; it captures patterns of  $L$ 's faction growing larger or becoming smaller over time. In the case of one leader, the faction size ratio in Eq. 2 is the same as the network density in Eq. 1.

### 3. Package architecture

Given a set of time series and related parameters as inputs, **mFLICA** infers a dynamic following network, faction leader and members, as well as degrees of coordination over time.

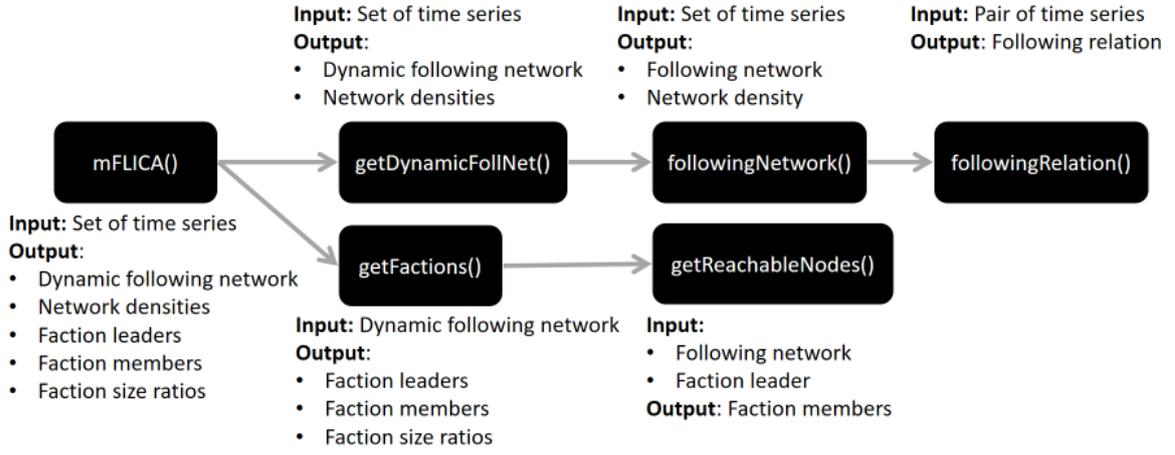


Figure 1: A high-level overview of **mFLICA** package architecture.

Figure 1 provides an overview of the package architecture. The main function is `mFLICA()` that calls two functions: `getDynamicFollNet()` and `getFactions()`. The `getDynamicFollNet()` is used to infer a dynamic following network from a set of time series, while `getFactions()` is used to infer faction leaders and faction members for each time step in a dynamic following network. In `getDynamicFollNet()`, it calls `followingNetwork()` for inferring a following network for each time intervals to create a dynamic following network. The `followingNetwork()` function uses `followingRelation()` as a main engine to infer a following relation between a pair of time series to build a following network. Lastly, `getFactions()` calls `getReachableNodes()` to find faction members, which are nodes that have directed path(s) to the faction leader.

In this paper, we use a simulated dataset of 30 time series of movement from [Amornbun-chornevej and Berger-Wolf \(2018\)](#) to demonstrate in examples of using **mFLICA** in leadership inference tasks. The dataset consists of two-dimensional time series of 30 individuals moving along the x-axis. The time series length is 800 time steps. There are three coordination events during the time interval [1,200] leading by individual ID1, the time interval [201,400] leading by ID2, and the time interval [400,600] leading by ID3.

In the next section, algorithms and examples regarding how to use **mFLICA** are provided for inferring leadership of coordination.

### 4. Inferring leadership of coordination

#### 4.1. Inferring following relations

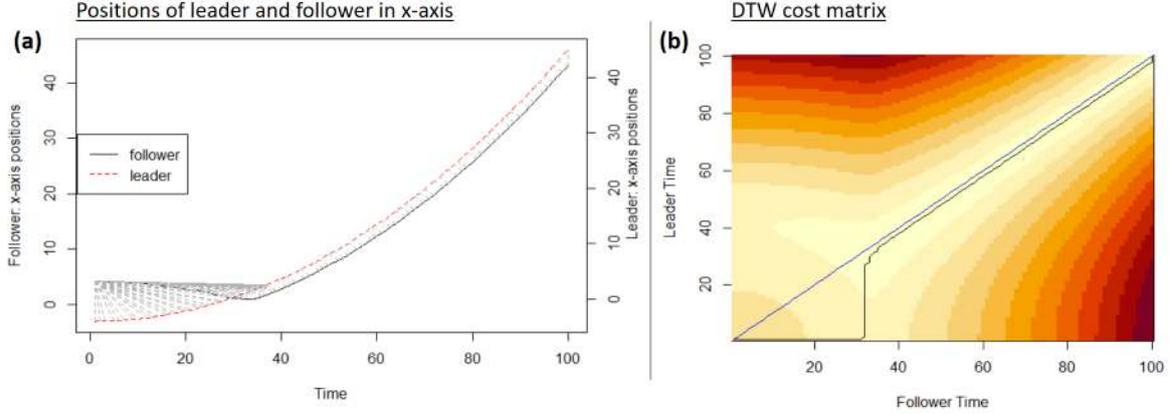


Figure 2: (a) Leader and follower time series in x-axis. These time series of positions are generated based on the movement of individuals in a two-dimensional plane where a leader moved along x-axis. A follower moved toward its leader in this plane. Both leader and follower have almost the same values in y-axis. (b) DTW cost matrix where darker-color shades represent higher distance. The black line is the optimal warping path between leader and follower, while the blue line is the diagonal line.

To infer a following relation between two time series, we deploy DTW to analyze an optimal warping path between two time series. Figure 2 shows simulated time series of movement from Amornbunchornvej and Berger-Wolf (2018). In this event, a leader was moving toward x-axis while the follower followed its leader after some time delay.

Figure 2 (a) illustrates the matching between two time series in x-axis. The cost matrix of DTW is shown in Figure 2 (b). Since the follower moved toward leader's previous positions, DTW optimal warping path matched leader's value at time  $t$  to follower's value at some time  $t + c$  where  $c > 0$ . Suppose  $P = ((r_1, c_1), \dots, (r_{|P|}, c_{|P|}))$  is the optimal warping path (the black line in Figure 2 (b)) where  $(r_i, c_i)$  represents that  $i$ th position  $P$  is at the  $r_i$ th row and  $c_i$ th column in the DTW cost matrix. The  $c_i - r_i$  is a time delay that the follower moved toward the same position as leader moved at time  $r_i$ th. By checking DTW cost matrix, a degree of following relation can be defined below.

$$s(P_{L,F}) = \frac{\sum_{i \in P_{L,F}} (\text{sign}(c_i - r_i))}{|P_{L,F}|} \quad (3)$$

Where  $P_{L,F}$  is the optimal warping path of leader  $L$ , and follower  $F$  inferred by DTW. A value of  $s(P_{L,F}) \in [-1, 1]$ . Given a threshold  $\sigma \in [0, 1]$ . If  $s(P_{L,F}) \in [-1, -\sigma]$ , then  $L$  follows  $F$ . If  $s(P_{L,F}) \in [\sigma, 1]$ , then  $F$  follows  $L$ . Otherwise, there is no following relation for  $s(P_{L,F}) \in (-\sigma, \sigma)$ . The Example 4.1 shows how we can use **mFLICA** to infer  $s(P_{L,F})$ .

**Example 4.1** *Infer a following relation from two-dimensional time series of movement. We deploy simulated time series  $TS$  from Amornbunchornvej and Berger-Wolf (2018) that included in this package. In this dataset,  $TS[1, 1 : 100, ]$  is a two-dimensional time series of leader while  $TS[2, 1 : 100, ]$  is a time series of follower. We use only the first 100 steps of*

time interval in this example. We run the code below for computing the optimal warping path between leader and follower.

```
1 R>library(mFLICA)
2 R>leader<-mFLICA::TS[1,1:100,] # Optimal warping path: obj$index2
3 R>follower<-mFLICA::TS[2,1:100,] # Optimal warping path: obj$index1
4 R>obj<-dtw(x=follower,y=leader,k=TRUE) # run dtw from 'dtw' package
```

We called `dtw` function and recorded the result in `obj` where `obj$index1` contains optimal warping path of a follower, and `obj$index2` contains optimal warping path of a leader. This means the time series of follower at time `obj$index1[i]` is matched (most similar w.r.t. DTW matching) with the leader time series at time `obj$index2[i]`. Then, we compute the average number of time steps until the follower reached the leader's previous positions.

```
1 R> mean(obj$index1 - obj$index2)
2 [1] 8.238462
```

On average, the follower required eight time steps to reach the its leader. Next, we calculate  $s(P_{L,F})$  in Eq. 3.

```
1 R>mean(sign(obj$index1 - obj$index2) )
2 [1] 0.9846154
```

This implies that there is a high degree of following relation between leader and follower ( $s(P_{L,F}) \approx 0.98$ ). In **mFLICA**, we implemented `followingRelation()` function for computing  $s(P_{L,F})$ . We deploy Sakoe-Chiba Banding [Sakoe and Chiba \(1978\)](#) for speeding up DTW computation. We can set the limitation of band via `lagWindow` parameter. In this example, we set the band parameter at 10% of the time series length (`lagWindow = 0.1`).

```
1 R> mFLICA::followingRelation(Y=follower,X=leader,lagWindow=0.1)$followVal
2 [1] 0.99
```

We have  $s(P_{L,F}) = 0.99$  in this example.

## 4.2. Inferring following networks

After we know how to infer a following relation between two time series, the next step is to infer a following network from a set of multiple time series using `followingNetwork()` function in **mFLICA**. The pseudo code for the `followingNetwork()` is in Algorithm 1. Given a set of time series and a threshold  $\sigma$ , the `followingNetwork()` function returns two adjacency matrices of the following network: `adjWeightedMat` and `adjBinMat`. The matrix `adjWeightedMat` is the adjacency matrix with weighted-directed edges, while `adjBinMat` is a binary version of `adjWeightedMat` where an edge weight can be either zero (if the weight is below  $\sigma$ ) or one (if the weight is greater than or equal  $\sigma$ ).

The Algorithm 1 computes  $s(P_{i,j})$  in Eq. 3 for all pairs of time series but it computes either  $s(P_{i,j})$  or  $s(P_{j,i})$  only once since the results of these two cases are the same with the opposite sign.

Next, we show how to use `followingNetwork()` in **mFLICA** to infer an adjacency matrix of a following network.

**Example 4.2** Infer adjacency matrices of following networks by using a set of simulated time series *TS*, which contains 30 trajectories of movement in two-dimensional plane along *x*-axis.

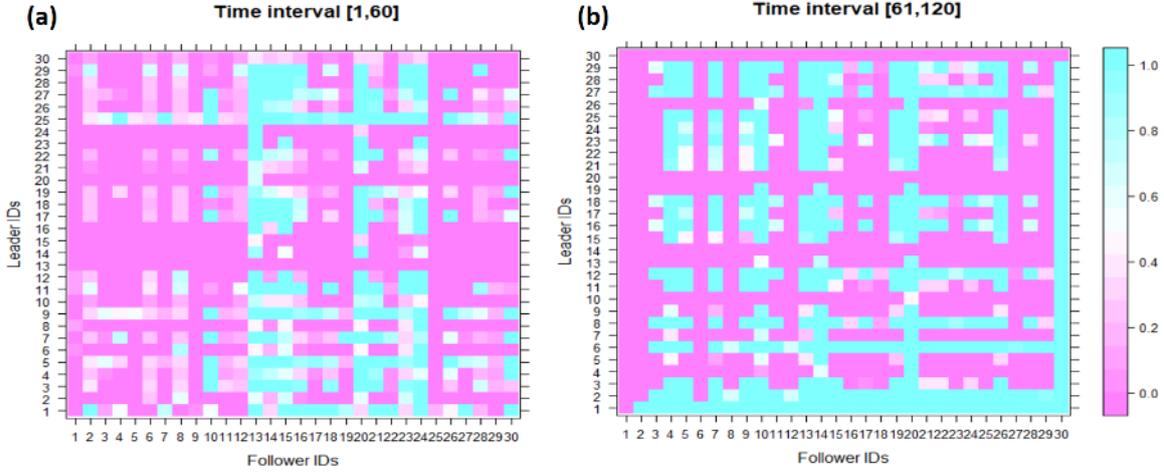


Figure 3: Weighted adjacency matrices of the following networks from difference time intervals. Elements in matrices represent degrees of following derived from  $s(P_{L,F})$  in Eq. 3 where leaders are rows and followers are columns (e.g.  $(i,j) = 0.5$  implies  $j$  follows  $i$  with degree 0.5). A higher value (light blue) implies a higher degree of following relation. (a) The matrix from the interval  $[1,60]$ , which has low degrees of coordination. (b) The matrix from the interval  $[61,120]$ , which has high degrees of coordination leading by ID1.

The low-coordination interval  $[1, 60]$  and high-coordination interval  $[61, 120]$  are chosen in the example. We set  $\sigma = 0,5$  for this example.

```
1 R>library(mFLICA)
2 R>mat1<-followingNetwork(TS=TS[,1:60,],sigma=0.5)$adjWeightedMat
3 R>mat2<-followingNetwork(TS=TS[,61:120,],sigma=0.5)$adjWeightedMat
```

Figure 3 illustrates the adjacency matrices from both intervals. The weighted adjacency matrix `mat1` at Figure 3 (a) is computed from the time interval  $[1,60]$  when the group initiated movement. In Figure 3 (b), the weighted adjacency matrix `mat2` is computed from the time interval  $[61, 120]$  when everyone followed its leader ID1, which implies it is a high-coordination event. `mFLICA` provides `getADJNetDen()` for computing a network density from an adjacency matrix. Based on the result, `mat1` has a lower network density than `mat2`'s network density. The network densities can be computed below.

```
1 R>getADJNetDen(mat1)
2 [1] 0.5559004
3 R>getADJNetDen(mat2)
4 [1] 0.7961686
```

In Figure 3 (b), in the row of ID1, all individuals have high degrees of following ID1, which implies that ID1 is a leader in this interval. In contrast, there are no individuals followed by the majority in Figure 3 (a), which implies that this interval has low degrees of coordination.

In the next step, we use `followingNetwork()` to infer a dynamic following network. Given a set of time series `TS` that has its time length at  $n$ , and the time window  $\omega$ , by Definition 5, we can use the sliding window technique to compute a dynamic following network from

**Algorithm 1:** followingNetwork()

---

```

input : a set of time series TS and a threshold  $\sigma$ 
output: adjBinMat, adjWeightedMat
1 Let  $m$  be a number of time series in TS and adjBinMat, adjWeightedMat are  $m \times m$  matrices
  of zeroes;
2 for  $i$  in  $[1, m - 1]$  do
3   for  $j$  in  $[i + 1, m]$  do
4     Compute  $s(P_{i,j})$  for time series TS[ $i$ ] as a leader and TS[ $j$ ] as a follower.;
     if  $|s(P_{i,j})| \geq \sigma$  then
       | flag=1
     else
       | flag=0
     end
     if  $s(P_{i,j}) > 0$  then
5       | set adjWeightedMat[ $j, i$ ] =  $s(P_{i,j})$  and adjBinMat[ $j, i$ ] = flag
     else
6       | set adjWeightedMat[ $i, j$ ] =  $|s(P_{i,j})|$  and adjBinMat[ $j, i$ ] = flag;
     end
   end
end
Return adjBinMat, adjWeightedMat;

```

---

followingNetwork() by dividing a set of time series TS into sub-intervals s.t. each sub interval has the time length  $\omega$ . We can start computing a following network at time interval  $[t, t + \omega]$ , then  $[t + 1, t + 1 + \omega]$ , and so on. However, sliding from one interval to another only one time step might require expensive computational cost.

Hence, **mFLICA** provides the time shift parameter  $\delta \in [1, \omega]$  to allow users to adjust the number of time steps between a previous interval and the next one. The getDynamicFollNet() in Algorithm 2 is a function for inferring a dynamic following network from a set of time series in **mFLICA**. The algorithm separates a set of time series  $TS$  into  $k$  sub-intervals where each interval has a length  $\omega$ . For each interval, the algorithm computes a following network using Algorithm 1, then it records the results of the following network into time series of adjacency matrices *dyNetWeightedMat* and its binary version *dyNetBinMat* as well as computing their network densities using Eq. 1. After finish computing following networks for all sub-intervals, the algorithm returns the results.

The next example provides the details how to use **mFLICA** to compute a dynamic following network.

**Example 4.3** *In this example, we use the set of simulated time series TS, which has the time length at 800 time steps. In this dataset, there are three coordination events:  $[1, 200]$ ,  $[201, 400]$ , and  $[401, 600]$ . We set the time window  $\omega = 60$ , the time shift  $\delta = 6$ , and the threshold  $\sigma = 0.5$ . The next commands are used to infer our dynamic following network of TS.*

```

1 R>library(mFLICA)
2 R>obj1<-getDynamicFollNet(TS=mFLICA::TS[,1:800,],timeWindow=60,timeShift = 6,
  sigma=0.5)

```

**Algorithm 2:** `getDynamicFollNet()`


---

**input :** a set of time series TS, a time window  $\omega$ , a time shift  $\delta$ , and a threshold  $\sigma$   
**output:** `dyNetWeightedMat`, `dyNetWeightedDensityVec`,  
`dyNetBinMat`, `dyNetBinDensityVec`

- 1 Let  $n$  be a length of time series;
- 2 Let  $s_1, \dots, s_k$  be starting times of  $k$  intervals where  $s_{i+1} = s_i + \delta$  and  
 $s_k = \max_{c \in \mathbb{N}}(c \times \delta + \omega \leq n)$ ;
- for**  $i$  *in*  $\{1, \dots, k\}$  **do**
- 3   Let TS' be a revised set of TS where it contains only the time interval  $[s_i, s_i + \omega]$  for each time series in TS. ;
- 4   Compute `followingNetwork(TS',  $\sigma$ )` to get adjacency matrices `adjBinMat`, `adjWeightedMat`;
- 5   Compute network densities for both `adjBinMat`, `adjWeightedMat` by Eq. 1;
- 6   Record `adjBinMat`, `adjWeightedMat` to `dyNetBinMat`, `dyNetWeightedMat` as adjacency matrices of the dynamic network for the time interval  $[s_i, s_i + \delta]$ ;
- 7   Record the network densities of `adjBinMat`, `adjWeightedMat` to `dyNetBinDensityVec`, `dyNetWeightedDensityVec` for the time interval  $[s_i, s_i + \delta]$ ;
- end**
- 8 Return `dyNetWeightedMat`, `dyNetWeightedDensityVec`, `dyNetBinMat`, `dyNetBinDensityVec`;

---

Suppose we want to know the following degree for ID19 follows ID1 at time step 150, we can use the command below.

```
1 R>obj1$dyNetWeightedMat[19,1,150]
2 [1] 0.9833333 # the following degree for ID19 follows ID1 at time step 150
```

We can query the network density at time step 150 with the command below.

```
1 R>obj1$dyNetWeightedDensityVec[150]
2 [1] 0.7755939
```

We can plot the time series of network density using the `plotMultipleTimeSeries` function implemented in **mFLICA** below.

```
1 R>plotMultipleTimeSeries(TS=obj1$dyNetWeightedDensityVec, strTitle="Network
   Dnesity")
```

Figure 4 shows the result of the plot. The plot shows that there are three coordination events that have high network densities (high degrees of coordination):  $[1,200]$ ,  $[201,400]$ , and  $[401,600]$ , which are consistent with our ground truth.

### 4.3. Inferring leadership dynamics

After having a dynamic following network, the next step is to infer leaders of factions and members of factions.

The Algorithm 3 provides the details of `getFactions()` implemented in **mFLICA** for inferring faction leaders, faction members, and faction size ratios. For each time step, a network structure of a following network at time  $t$  is analyzed. According to Definition 3, nodes of leaders are inferred as nodes in a following network that have zero out-degree. Then, for each leader node  $L$ , nodes that are members of  $L$ 's faction are identified by transitive closure property. If there exists a path from any node  $v$  to  $L$ , then  $v$  is a member of  $L$ 's faction.

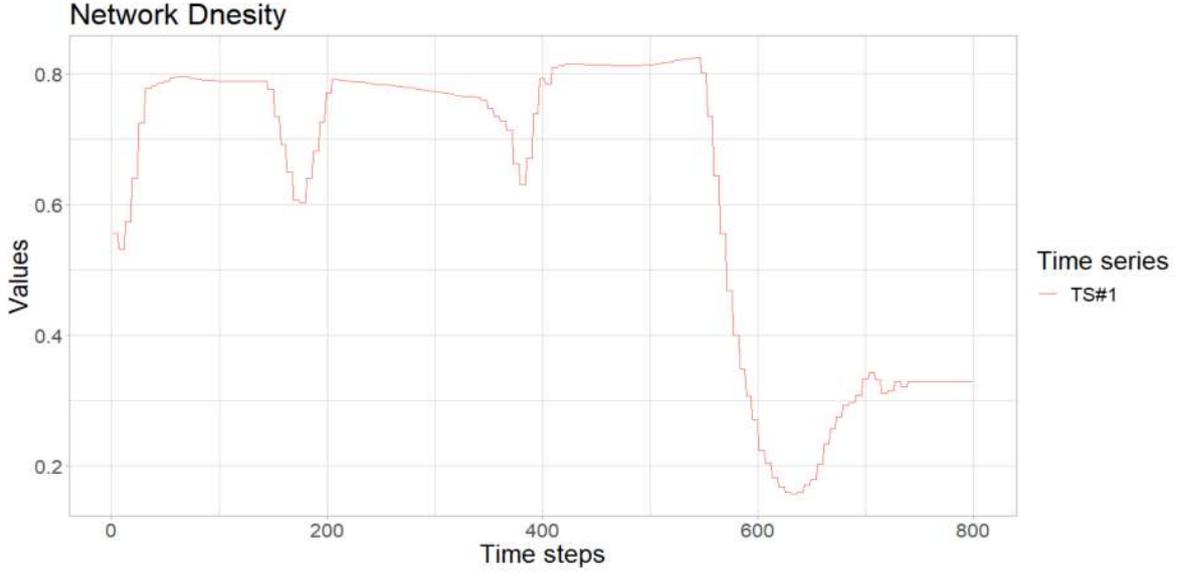


Figure 4: Time series of network densities of a dynamic following network from simulated time series **TS** in **mFLICA** package. The plot shows that there are three coordination events that have high network densities (high degrees of coordination):  $[1,200]$ ,  $[201,400]$ , and  $[401,600]$ , which are consistent with our ground truth.

Hence, a faction is a tree with reversed directed paths in a following network s.t. all node members have a path to its root (leader). The step of inferring faction members can be done by applying Breadth-First Search (BFS) algorithm on an adjacency matrix of a following network. The BFS algorithm is implemented in `getReachableNodes()` in the package. After we have all factions within a following network, the last step is to measure the size of each faction using a faction size ratio in Eq. 2. In the next example, we provide the way to use **mFLICA** to infer faction leaders and their members using `getFactions()`.

**Example 4.4** *In this example, we use the interval  $[25,45]$  to demonstrate the time when there are more than one factions occur simultaneously. After having a following network, `getFactions()` takes a binary version of adjacency matrix as its input.*

```
1 R>library(mFLICA)
```

---

### Algorithm 3: `getFactions()`

---

**input** : A binary adjacency matrix `adjBinMat`  
**output**: `leaders,factionMembers,factionSizeRatio`

- 1 Infer zero-out-degree nodes in `adjBinMat` and keep them in `leaders`;
- for** `leader` *in* `leaders` **do**
- 2     Finding all nodes that are reachable to `leader` in `adjBinMat` using Breadth First Search (BFS) algorithm;
- 3     Keep these nodes as faction members of `leader` in `factionMembers[leader]`;
- 4     Computing faction-size ration in Eq. 2 and keep it in `factionSizeRatio[leader]`;
- end**

Return `leaders,factionMembers,factionSizeRatio`;

---

```

2 R>mat1 <-followingNetwork(TS=TS[,25:45,], sigma =0.95)$adjBinMat
3 R>out<-getFactions(adjMat=mat1)
4 R>out$leaders # show leader IDs
5 [1] 1 11

```

The code above shows that there are two faction leaders in the interval [25,45]: ID1 and ID11. This implies that there are two factions. The next step is to query faction members of ID1's faction as well as its faction size ratio.

```

1 R>L1<-out$leaders[1] # leader ID1
2 R>out$factionMembers[[1]] # show faction members
3 [1] 1 2 3 4 5 6 7 8 9 10 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26
    27 28 29 30
4 R>out$factionSizeRatio[L1] # show faction size ratio
5 [1] 0.5034483

```

Note that a leader is also a faction member itself. Since there are 30 individuals, almost everyone is a member of ID1's faction. However, the faction size ratio at 0.5 indicates that faction members are not coordinated following the same pattern yet. The next one is the code for querying details about a faction leading by ID11.

```

1 R>L1<-out$leaders[2] # leader ID11
2 R>out$factionMembers[[2]] # show faction members
3 [1] 11 7 10 13 14 15 16 18 19 20 21 22 23 24 26 28 30
4 R>out$factionSizeRatio[L2] # show faction size ratio
5 [1] 0.1632184

```

We can see that there are a few number of members in this faction. Note that one individual can belong to more than one faction since the individual might follow some pattern that seems partially similar to several leaders' patterns.

Lastly, we can use `mFLICA()` to infer a dynamic following network and dynamics of factions from a set of time series. Algorithm 4 provides the details of `mFLICA()` implementation. First, `mFLICA()` computes a dynamic following network and its time series of network densities from a set of time series w.r.t. given parameters. Then, for each time step, `mFLICA()` infers faction leaders, faction members, and faction size ratios from the dynamic following network.

---

**Algorithm 4:** `mFLICA()`


---

**input :** a set of time series  $TS$ , a time window  $\omega$ , a time shift  $\delta$ , and a threshold  $\sigma$   
**output:** `dyNetWeightedMat`, `dyNetWeightedDensityVec`,  
`dyNetBinMat`, `dyNetBinDensityVec`,  
`TSleaders`, `TSfactionMembers`, `TSfactionSizeRatio`

- 1 Computing a dynamic following network using `getDynamicFollNet()`;
- for**  $t$  *in*  $[1, m]$  **do**
- 2   Let `dyNetBinMat[t]` be an adjacency matrix at time  $t$ ;
- 3   Computing faction members, leaders, and faction size ratios using  
    `getFactions(dyNetBinMat[t])`;
- 4   Keep faction members in `TSfactionMembers[t]`, leaders in `TSleaders[t]`, and faction  
    size ratio in `TSfactionSizeRatio[t]`;
- end**
- Return `dyNetWeightedMat`, `dyNetWeightedDensityVec`, `dyNetBinMat`, `dyNetBinDensityVec`,  
    `TSleaders`, `TSfactionMembers`, `TSfactionSizeRatio` ;

---

The next example shows the details of using `mFLICA()` to infer dynamics of factions.

**Example 4.5** In this example, we show how to use **mFLICA** to infer dynamics of factions. In other words, we would like to find changes of faction members and faction leaders over time. Given a set of time series **TS** as an input along with related parameters: time window  $\omega = 60$ , time shift  $\delta = 6$ , and the threshold  $\sigma = 0.5$ , we run **mFLICA()** below.

```
1 R>library(mFLICA)
2 R>obj1<-mFLICA(TS=mFLICA::TS[,1:800,],timeWindow=60,timeShift = 6,sigma=0.5)
```

All results of faction inference are in **obj1**. Here, we focus on a set of time series of faction size ratios **obj1\$factionSizeRatioTimeSeries** where **obj1\$factionSizeRatioTimeSeries[i,t]** is a faction size ratio of a faction leading by **ID<sub>i</sub>** at time **t**. We can plot the time series of faction size ratios using **plotMultipleTimeSeries** below.

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

The result of the plot is in Figure 5. According to the ground truth for this simulated dataset mentioned in Section 3, there are three coordination events. First, during the time interval  $[1,200]$ , **ID1** is a sole leader who leads its faction of 30 individuals. Then, **ID2** leads the faction for the time interval  $[201,400]$ . Afterward, **ID3** leads the faction during the interval  $[401,600]$ . Finally, the group slows down and stop moving. The result in Figure 5 reflexes this ground truth. **ID1** has its high faction size ratios during  $[1,200]$ , **ID2**'s faction continues to have high faction size ratios during  $[201,400]$ . Lastly, **ID3**'s faction has high faction size ratios during  $[401,600]$ . No factions have high faction size ratios during  $[601,800]$ .

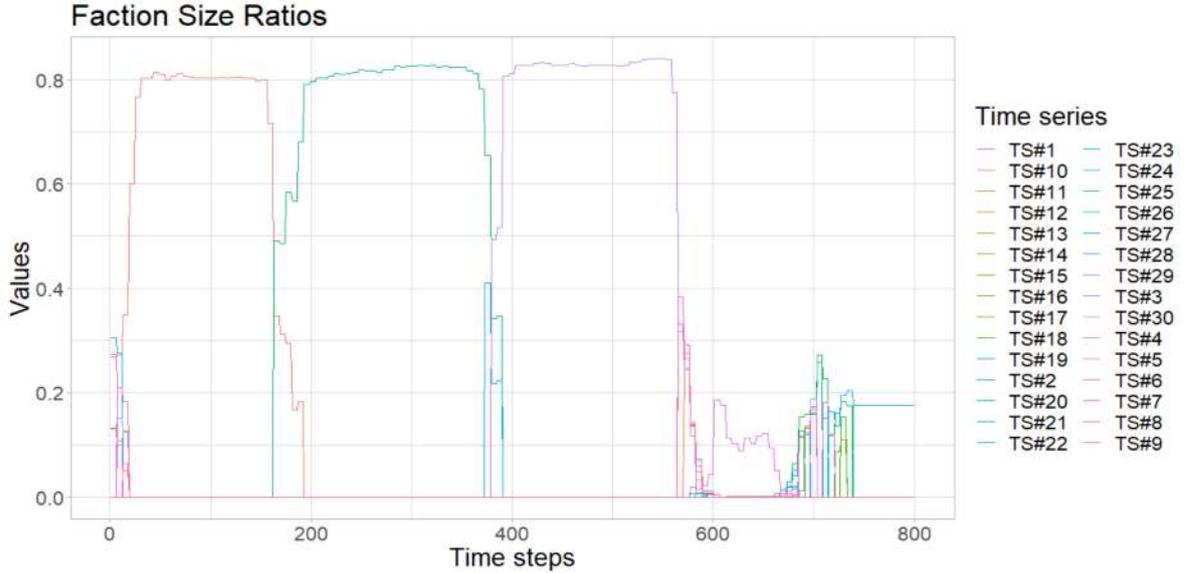


Figure 5: Time series of faction size ratios. Each time series of faction sizes ratios is represented by ID (TS#ID) of its faction leader.

## 5. Conclusion

In this paper, the details of **mFLICA** package for inferring leadership of coordination from time series are provides. Leaders are defined as individuals who initiate some patterns and

others follow the same patterns with some time delays. A following relation between time series can be detected by analyzing an optimal warping path of Dynamic Time Warping (DTW), which is the main component that **mFLICA** deploys.

Given a set of time series and related parameters, the **mFLICA** package can infer a following relation between two time series, following networks, faction leaders, faction members, degrees of coordination, and faction size ratios for each time step.

The network densities inferred by **mFLICA** tell us regarding the magnitude of coordination: how many time-series individuals follow the same pattern in a given time interval. The faction size ratios provide information regarding faction dynamics; the changes of faction leaders, and/or faction members over time. We provided the examples of how to use **mFLICA** for solving many tasks in leadership inference. Our framework can be applied to any multivariate time series. All figures and results can be reproduced using the R notebook at <https://github.com/DarkEyes/mFLICA/blob/master/exp/JSSnotebook.Rmd>

## References

- Amornbunchornvej C (2020). **mFLICA: Leadership-Inference Framework for Multivariate Time Series**. R package version 0.1.0, URL <https://cran.r-project.org/package=mFLICA>.
- Amornbunchornvej C, Berger-Wolf T (2018). “Framework for Inferring Leadership Dynamics of Complex Movement from Time Series.” In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pp. 549–557. SIAM. doi:10.1137/1.9781611975321.62. <https://epubs.siam.org/doi/pdf/10.1137/1.9781611975321.62>, URL <https://epubs.siam.org/doi/abs/10.1137/1.9781611975321.62>.
- Amornbunchornvej C, Brugere I, Strandburg-Peshkin A, Farine DR, Crofoot MC, Berger-Wolf TY (2018). “Coordination Event Detection and Initiator Identification in Time Series Data.” *ACM Trans. Knowl. Discov. Data*, **12**(5), 53:1–53:33. ISSN 1556-4681. doi:10.1145/3201406. URL <http://doi.acm.org/10.1145/3201406>.
- Andersson M, Gudmundsson J, Laube P, Wolle T (2008). “Reporting leaders and followers among trajectories of moving point objects.” *GeoInformatica*, **12**(4), 497–528.
- Behrendt S, Dimpfl T, Peter FJ, Zimmermann DJ (2019). “**RTransferEntropy** — Quantifying information flow between different time series using effective transfer entropy.” *SoftwareX*, **10**, 100265. ISSN 2352-7110. doi:<https://doi.org/10.1016/j.softx.2019.100265>. URL <http://www.sciencedirect.com/science/article/pii/S2352711019300779>.
- Bracewell R (1965). “Pentagram notation for cross correlation. The Fourier transform and its applications.” *New York: McGraw-Hill*, **46**, 243.
- Giorgino T (2009). “Computing and Visualizing Dynamic Time Warping Alignments in R: The **dtw** Package.” *Journal of Statistical Software*, **31**(7), 1–24. URL <http://www.jstatsoft.org/v31/i07/>.
- Glowacki L, von Rueden C (2015). “Leadership solves collective action problems in small-scale societies.” *Phil. Trans. R. Soc. B*, **370**(1683), 20150010.

- Granger CW (1969). “Investigating causal relations by econometric models and cross-spectral methods.” *Econometrica: Journal of the Econometric Society*, pp. 424–438.
- Hogg MA (2001). “A social identity theory of leadership.” *Personality and social psychology review*, **5**(3), 184–200.
- Kjargaard MB, Blunck H, Wustenberg M, Gronbask K, Wirz M, Roggen D, Troster G (2013). “Time-lag method for detecting following and leadership behavior of pedestrians from mobile sensing data.” In *Proceedings of the IEEE PerCom*, pp. 56–64. IEEE.
- Malone TW, Crowston K (1994). “The interdisciplinary study of coordination.” *ACM Computing Surveys (CSUR)*, **26**(1), 87–119.
- R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL <https://www.R-project.org/>.
- Sakoe H, Chiba S (1978). “Dynamic programming algorithm optimization for spoken word recognition.” *IEEE transactions on acoustics, speech, and signal processing*, **26**(1), 43–49.
- Schreiber T (2000). “Measuring information transfer.” *Physical review letters*, **85**(2), 461.
- Zeileis A, Hothorn T (2002). “Diagnostic Checking in Regression Relationships.” *R News*, **2**(3), 7–10. URL <https://CRAN.R-project.org/doc/Rnews/>.

**Affiliation:**

Chainarong Amornbunchornvej  
National Electronics and Computer Technology Center  
12120 Pathum Thani, Thailand  
E-mail: [chainarong.amo@nectec.or.th](mailto:chainarong.amo@nectec.or.th)  
URL: <https://orcid.org/0000-0003-3131-0370>