HW: Temporal and Dynamic Networks

Marwin Carmo

Introduction

Welcome to the Homework Assignment for Temporal and Dynamic Networks! For this week's assignment, you'll find the dataset's `.RDS` file on Canvas.

Instructions

- Download the dataset from Canvas as a `.RDS` file
- Load that `.RDS` file into your `R` environment and begin the assignment
  - This can be done with the command, 'readRDS()"

Below, you will find a template of the questions and fields to provide answers in either `R` or text format. Please use a mix of code and text to answer each question.

```
library(sna)
library(tsna)
library(ndtv)
library(OpenMx)
```

Questions

***Question 1*. Read in your temporal network via the `.RDS` file. Pick a random node and visualize its forward path.**

Answer

```r
df <- readRDS("hwdf.RDS")

n_nodes = max(c(df$tail, df$head))
  base_net = network.initialize(n_nodes, directed = FALSE)

net_dyn = networkDynamic(
  base.net = base_net,
  edge.spells = df[, c("onset", "terminus", "tail", "head")]
  )
```

```
Created net.obs.period to describe network
 Network observation period info:
  Number of observation spells: 1
  Maximal time range observed: 2 until 300
  Temporal mode: continuous
  Time unit: unknown
  Suggested time increment: NA
```

```r
vpath = tPath(net_dyn,v = 13, direction = "fwd")
print(vpath)
```

```
$tdist
 [1]  2  6  6  3  4  8  4  8  4  4  4  9  0  2  2  9  0  2  4  4  5  7  1  9  9
[26]  3  2  4  4  7 11  7 10  5  2  2  2  3  3  6  4  3  7 12  7  9  7  6  8  5

$previous
 [1] 27 23 48 27 10 18 27 30 20  7 29 25  0 13 14 24 13 15 28 10 26  2 17 37 43
[26] 39 15  5 28  3 39 30 42 13 18 37 27 26  4 38  7 26 32 18 36 42 40 40 35 39

$gsteps
 [1]  4  3 10  4  6  4  4 12  7  5  9 15  0  1  2  6  1  3  8  6  7  4  2  5 14
[26]  6  3  7  8 11  6 12  8  1  4  5  4  7  5  8  5  7 13  4  6  8  9  9  5  6

$start
[1] 2

$end
[1] Inf

$direction
[1] "fwd"
```

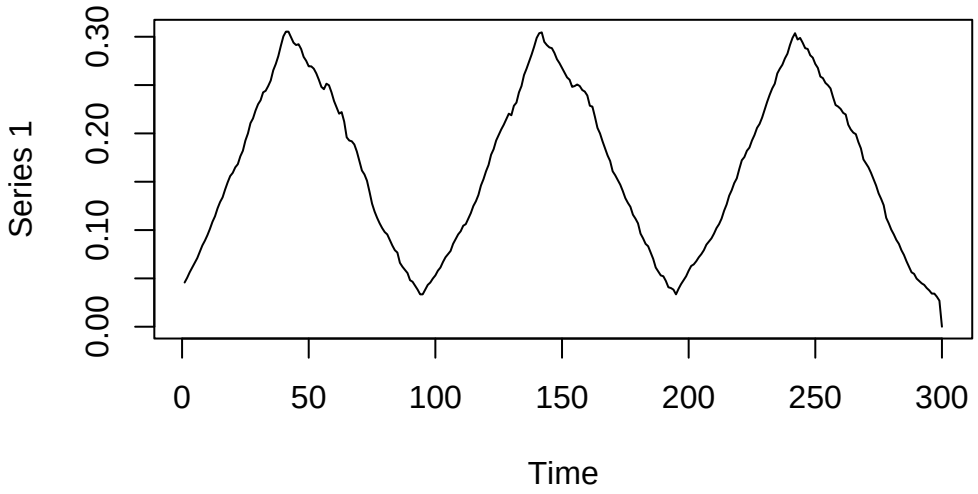
```
$type
[1] "earliest.arrive"

attr(,"class")
[1] "tPath" "list"
```

***Question 2*. Plot your temporal network's density over time.**

Answer

```r
dynamicdensity = tSnaStats(
  net_dyn,
  snafun = "gden",
  start = 1,
  end = 300,
  time.interval = 1,
  aggregate.dur = 10
)
plot(dynamicdensity)
```



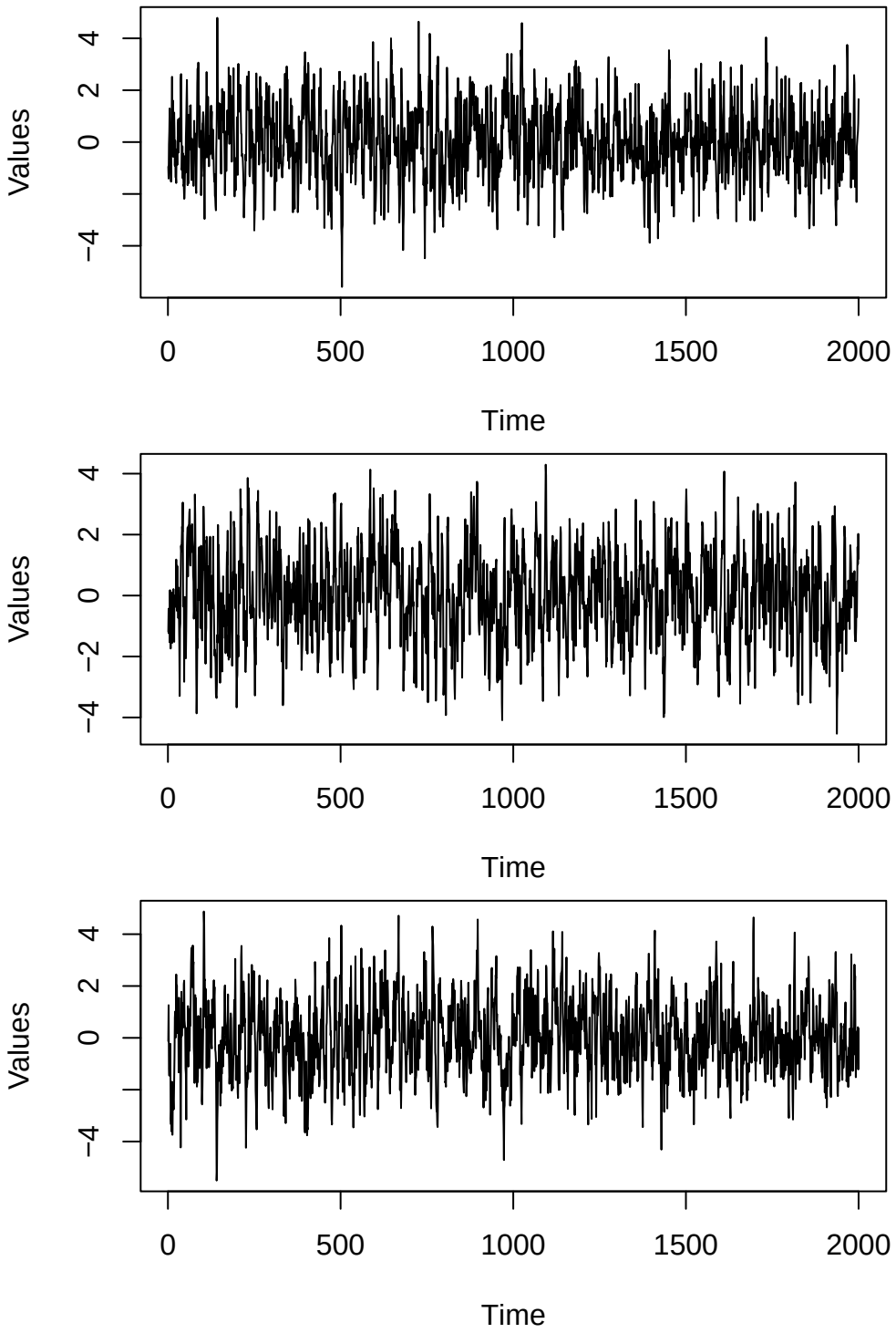
***Question 3*. Using OpenMx, simulate a stationary three variable system that evolves over 2000 time points. Define the 3×3 transition matrix using the following values (but make sure the time series are stationary by plotting them):**

$$\mathbf{A}_{mat} = \begin{bmatrix} 0.58 & 0.00 & -0.31 \\ 0.27 & 0.65 & 0.00 \\ 0.00 & 0.20 & 0.62 \end{bmatrix}$$

Answer

```
ne = 3
  VAR.params = matrix(c(0.58, 0.27, 0.00,
                        0.00, 0.65, 0.25,
                        -0.31, 0.00, 0.62), ne, ne)
  amat = mxMatrix('Full', ne, ne, TRUE, VAR.params, name = 'A')
  bmat = mxMatrix('Zero', ne, ne, name='B')
  cdim = list(paste0("V", 1:ne), paste0('F', 1:ne))
  cmat = mxMatrix('Diag', ne, ne, FALSE, 1, name = 'C', dimnames = cdim)
  dmat = mxMatrix('Zero', ne, ne, name='D')
  qmat = mxMatrix('Symm', ne, ne, FALSE, diag(ne), name='Q', lbound=0)
  rmat = mxMatrix('Symm', ne, ne, FALSE, diag(1e-5, ne), name='R')
  xmat = mxMatrix('Full', ne, 1, FALSE, 0, name='x0', lbound=-10, ubound=10)
  pmat = mxMatrix('Diag', ne, ne, FALSE, 1, name='P0')
  umat = mxMatrix('Zero', ne, 1, name='u')
  osc = mxModel("OUMod", amat, bmat, cmat, dmat, qmat, rmat, xmat, pmat, umat,
                mxExpectationStateSpace('A', 'B', 'C', 'D', 'Q',
                                        'R', 'x0', 'P0', 'u'))
  sim.data = mxGenerateData(osc, nrows = 2000)
```

```
apply(sim.data, 2, function(var) {
  plot(x = (1:nrow(sim.data)), y = var, type = "l",
       ylab = "Values", xlab = "Time")
})
```


Values
4
2
0
−4
0
500
1000
1500
2000
Time
Values
4
2
0
−2
−4
0
500
1000
1500
2000
Time
Values
4
2
0
−4
0
500
1000
1500
2000
Time

```
NULL
```

***Question 4*. Fit a discrete time VAR to the simulated data and print the parameter values. Comment on the recovery, i.e., did the VAR do a good job at estimating the parameters?**

Answer

```
VAR1 = mx.var(dataframe = sim.data, varnames = paste0("V", 1:ne))
```

```
params = matrix(0, ne, ne)
  for(i in 1:nrow(summary(VAR1)$parameters)){
    params[summary(VAR1)$parameters[i,"row"], summary(VAR1)$parameters[i, "col"]] =
      ifelse(abs(summary(VAR1)$parameters[i,"Estimate"]) >
               qnorm(0.975) * summary(VAR1)$parameters[i,"Std.Error"],
             summary(VAR1)$parameters[i,"Estimate"],
             0.00)
  }

# True
VAR.params
```

```
     [,1] [,2]  [,3]
[1,] 0.58 0.00 -0.31
[2,] 0.27 0.65  0.00
[3,] 0.00 0.25  0.62
```

```
# Estimated
round(params, 2)
```

```
     [,1] [,2]  [,3]
[1,] 0.57 0.00 -0.29
[2,] 0.25 0.67  0.00
[3,] 0.00 0.24  0.63
```

Yes! The estimated parameter values are very close to the true values. There are only small discrepancies, with maximum differences of 0.02.

***Question 5*. Now fit a continuous time VAR to the same data. Convert the drift matrix to its discrete time counterpart. Comment on the recovery.**

Answer

```
sim.data$Time = 0:(nrow(sim.data)-1)
ct.var = mx.ctvar(dataframe = sim.data, varnames = paste0("V", 1:ne))
```

```
# discrete values
expm(matrix(summary(ct.var)$parameters$Estimate, 3,3) * 1.00)
```

```
           [,1]       [,2]        [,3]
[1,] 0.6999624 -0.0764931 -0.21165831
[2,] 0.1895519  0.7464790 -0.05713195
[3,] 0.1212383  0.1646350  0.74763353
```

```
#true
VAR.params
```

```
     [,1] [,2]  [,3]
[1,] 0.58 0.00 -0.31
[2,] 0.27 0.65  0.00
[3,] 0.00 0.25  0.62
```

The recovery of the parameters after fitting a continuous-time VAR is noticeably worse than the direct discrete-time VAR. The model is clearly misspecified, since I generated data from a discrete-time process.