Simulation and estimation of longitudinal partition models - Example script with random data

Marion Hoffman

28/09/2020

In this example script, we use the Exponential Random Partition Model to simulate longitudinal partitions and estimate the model for a series of longitudinal observations (here, synthetic data for three time points). More information can be found in the manual of this package present on the github repository stocnet/erpm or by looking up the documentation of specific functions.

This script can be used as a starting point for anyone wishing to use the model on longitudinal observations of partitions, like teams, interaction groupos, political parties, or animal herds observed at different time points.

0. Load dependencies

The following packages and scripts are required to fully run this example script.

```r
library(ERPM)
library(ggplot2)
theme_set(theme_minimal())
```

1. Simulate partitions

1.1 Define nodesets and attributes

We first define the same arbitrary set of n = 6 nodes with attributes, and an arbitrary covariate matrix as in the first example script.

```r
n <- 6
nodes <- data.frame(label = c("A","B","C","D","E","F"),
                    gender = c(1,1,2,1,2,2),
                    age = c(20,22,25,30,30,31))
friendship <- matrix(c(0, 1, 1, 1, 0, 0,
                       1, 0, 0, 0, 1, 0,
                       1, 0, 0, 0, 1, 0,
                       1, 0, 0, 0, 0, 0,
                       0, 1, 1, 0, 0, 1,
                       0, 0, 0, 0, 1, 0), 6, 6, TRUE)
```

Here we need to also specify when the nodes are present. Here we assume all nodes are present in the first time point, but node A is absent in the second and node B is absent in the third. We do this by creating a presence matrix:

```r
presence.tables <- matrix(c(1, 1, 1, 1, 1, 1,
                            0, 1, 1, 1, 1, 1,
                            1, 0, 1, 1, 1, 1), 6, 3)
```

2.2 Define a model and simulate

First, we need to choose the effects we want to include (see manual for all effect names). For example we set five (which is of course not reasonable for 6 nodes). The first is for the number of groups, the second is related to gender similarity, the third to age similarity, the fourth to the presence of friendships, and the fifth is related to inertia (do individuals tend to stay in the same group from one partition to the next):

```
effects_multiple <- list(names = c("num_groups","same","diff","tie","inertia_1"),
                   objects = c("partitions","gender","age","friendship","partitions"),
                   objects2 = c("","","","",""))
objects_multiple <- list()
objects_multiple[[1]] <- list(name = "friendship", object = friendship)
```

and we can set parameter values for each of these effects.

```
parameters <- c(-0.2,0.2,-0.1,0.5,1)
```

We can set a starting point for the simulation

```
first.partitions <- matrix(c(1, 1, 2, 2, 2, 3,
                             NA, 1, 1, 2, 2, 2,
                             1, NA, 2, 3, 3, 1), 6, 3)
```

Now we can generate our simulated sample, by setting the desired additional parameters for the Metropolis sampler:

```
nsteps <- 50
sample <- draw_Metropolis_multiple(theta = parameters,
                           first.partitions = first.partitions,
                           nodes = nodes,
                           presence.tables = presence.tables,
                           effects = effects_multiple,
                           objects = objects_multiple,
                           burnin = 100,
                           thining = 100,
                           num.steps = nsteps,
                           neighborhood = c(0,1,0),
                           numgroups.allowed = 1:n,
                           numgroups.simulated = 1:n,
                           sizes.allowed = 1:n,
                           sizes.simulated = 1:n,
                           return.all.partitions = TRUE)
```

2.3 Trace plots

We can check the mixing of the chain with autocorrelations for the five statistic (they should remain below 0.4 in absolute value).

```
cor(sample$draws[1:(nsteps-1),1], sample$draws[2:nsteps,1])
```

```
## [1] 0.1725703
```

```
cor(sample$draws[1:(nsteps-1),2], sample$draws[2:nsteps,2])
```

```
## [1] 0.2614216
```

```
cor(sample$draws[1:(nsteps-1),3], sample$draws[2:nsteps,3])
```

```
## [1] 0.01037624
```

```r
cor(sample$draws[1:(nsteps-1),4], sample$draws[2:nsteps,4])
```

```
## [1] 0.1147235
```

```r
cor(sample$draws[1:(nsteps-1),5], sample$draws[2:nsteps,5])
```

```
## [1] 0.3416614
```

3. Estimate for an observed partition

3.1 Define the observation

```r
partitions <- matrix(c(1, 1, 2, 2, 2, 3,
                       NA, 1, 1, 2, 2, 2,
                       1, NA, 2, 3, 3, 1), 6, 3)
```

3.2 Estimate

We can use the estimate function for longitudinal partitions in a very similar way as in the cross-sectional case.

```r
startingestimates <- c(-2,0,0,0,0)
estimation_multiple <- estimate_multipleERPM(partitions,
                          presence.tables,
                          nodes,
                          objects_multiple,
                          effects_multiple,
                          startingestimates = startingestimates,
                          burnin = 100,
                          thining = 50,
                          gainfactor = 0.6,
                          length.p1 = 200,
                          multiplication.iter.p2 = 20,
                          num.steps.p2 = 4,
                          length.p3 = 1000)

# get results table
estimation_multiple
```

```
##       effect     object        est   std.err sig          t        conv
## 1 num_groups partitions -3.8874010 1.8429835   * -2.1092978 -0.01859912
## 2       same     gender -0.1407062 1.0923752     -0.1288076  0.07549121
## 3       diff        age -0.2454244 0.1285190     -1.9096348  0.24542274
## 4        tie friendship -0.9782836 1.1216975     -0.8721457  0.03238567
## 5  inertia_1 partitions -0.8645400 0.8436466     -1.0247655  0.34641560
```

Again, the convergence should be as small as possible, below 0.1 for example, so it might be good to rerun the algorithm using the new estimated parameters. Similar procedures as in the estimation of ERGM or SAOM models can be used.