Network Analysis: Homework

Jeff Ziegler

Due: August 23, 2018

1 Nigeria Data Processing

a) Process the data: turn this event dataset into a matrix.

b) Specifically, summarize the interactions across all time periods into an adjacency matrix where:

1. "1" indicates that i and j had a conflictual interaction sometime during the temporal span of the original dataset and zero otherwise.
2. Make sure all actors that existed at any point during the temporal span are included in the adjacency matrix.

```
rm(list=ls())
# set working directory to Git location
setwd('/Users/jeffziegler/Documents/Git/network2018_hw1/')
# load data
load("nigeria.rda")

###############################
# (1) Nigeria Data Processing
###############################

# create variable for row and column length of adjacency matrix to fill
rowLength <- length(unique(nigeria$sender))
colLength <- length(unique(nigeria$receiver))
# create adjacency matrix of sender and receiver, filled w/ zeroes
nigeriaAdjMat <- matrix(0, nrow=rowLength, ncol=colLength)
# adjust row and column names for sender and receiver
rownames(nigeriaAdjMat) <- unique(nigeria$sender)
colnames(nigeriaAdjMat) <- unique(nigeria$receiver)
# fill in adjacency matrix per year (i)
# start by sorting all unique years to iterate over
nigeriaAdjMatYearlyList  <- lapply(sort(unique(nigeria$year)), function(i){
  # find just those pairings for (1) a given year
  currentYear <- nigeria[nigeria$year == i,]
  # and (2) had a conflict (conflict == 1)
```

```r
  yearlyConflicts <- currentYear[currentYear$conflict == 1,]
  # now that we know which pairings had conflicts
  # fill in a "1" based on sender and receiver
  for(i in 1:nrow(yearlyConflicts)){
    nigeriaAdjMat[as.character(yearlyConflicts[i,]$sender),
                  as.character(yearlyConflicts[i,]$receiver)] <- 1
  }
  # return the adjacency matrix, which will be placed in a list
  return(nigeriaAdjMat)
})
# collapse all the matrices in list into one matrix
# since instructions are to "summarize the interactions
# across all time periods into a single matrix"
nigeriaAdjMatTotalMatrix <- Reduce('+', nigeriaAdjMatYearlyList)
```

2 Measurements & Community Detection

a) Which actor is the most "influential" in the network? Justify your response and the measure you choose to estimate "influence."

b) Employ the blockmodel function from the sna package to explore potential group level structure in the data (see slides 61-63 from day 2 for details):

  - Run blockmodel with varying levels of k.
  - Save the node classifications from each run.
  - Now how do we choose k?
    * You will do so through an out-of-sample cross-validation exercise (at least 10 folds).
    * Report the AUC (ROC) and AUC (PR) statistics from each model.

c) After having determined the k that gives the best out of sample performance, visualize your results as shown in slide 67 from the day 2 lecture

```r
##########################################
# (2) Measurements and Community Detection
##########################################

# (a) We want to discern who the most "influential" actor in the network is
# We'll create two measures of "influence":
# (1) degree (number of connections)
# (2) eigenvector centrality (connections to high-scoring nodes
# contribute more to the score of the node)

# create graph object from igraph package
library(igraph)
igeriaGraph <- graph_from_adjacency_matrix(nigeriaAdjMatTotalMatrix,
          mode='directed', weighted=TRUE, diag=FALSE)
```

```
# (1) degree
head(sort(degree(nigeriaGraph), decreasing=T))

# interestingly, the Police (Nigeria) and the Military (Nigeria)
# are two of the top 3 most engaged actors (Fulani Militia is #2)

# (2) eigenvector centrality
head(sort(eigen_centrality(nigeriaGraph, directed = TRUE)$vector,
          decreasing=T))

# again, the police and military are not only more involved in conflicts
# but they engage w/ other highly conflicted actors

# (b) Instruction: Run blockmodel with varying levels of k
# Tasks/traits for blockmodel function (each run needs to):
# [1] Save the node classifications
# [2] Conduct out-of-sample CV (10 folds)
# [3] Report the AUC (ROC) and AUC (PR) statistics

# first, recreate matrices so that they are network objects
library(network)
nigeriaAdjMatNetworkList <- lapply(sort(unique(nigeria$year)), function(i){
  # find just those pairings for (1) a given year
  currentYear <- nigeria[nigeria$year == i,]
  # and (2) had a conflict (conflict == 1)
  yearlyConflicts <- currentYear[currentYear$conflict == 1,]
  # now that we know which pairings had conflicts
  # fill in a "1" based on sender and receiver
  for(i in 1:nrow(yearlyConflicts)){
    nigeriaAdjMat[as.character(yearlyConflicts[i,]$sender),
                  as.character(yearlyConflicts[i,]$receiver)] <- 1
  }
  # return the adjacency matrix, which will be placed in a list
  return(as.network.matrix(nigeriaAdjMat))
})

# create function that will do tasks 1-3
# then we can run CV function for varying levels of k
# Arguments:
# (remember function takes in igraph object)
# nFolds = number of folds (default = 10)
# nClusters = number of cluster (default = 2)
library(sna); library(caret); library(networkDynamic); library(btergm)
crossValidateFunc <- function(networkData, nFolds=NULL, nClusters=NULL) {
  # set seed for reproducibility
  set.seed(5)
  # createFolds function from caret package
  # argument gives a list of the indicies in each fold
  # from the groups that comprise all possible conflicts
  # return training data
```

```r
  cvFolds <- createFolds(y = unique(nigeria$sender),
                         k=nFolds, returnTrain = T)
  # create empty vectors to fill w/ goodness-of-fit stats
  # from TERGMS (AUC (ROC) and AUC (PR))
  # ROC and PR curves can be used to compare different model specifications,
  # also for within-sample goodness-of-fit
  AUC_ROC <- NULL; AUC_PR <- NULL
  # iterate over folds
  for (i in 1:nFolds) {
    # transform input list into network list
    networkList <- networkDynamic(network.list=networkData)
    # remove the necessary observations that are exempt from each fold
    delete.vertices(networkList, (1:dim(nigeriaAdjMat)[1])[-cvFolds[[i]]])
    # create clusters from structural equivalence
    equivNetClusters <- equiv.clust(networkList)
    # perform blockmodel
    blockModel <- blockmodel(networkList, equivNetClusters, k=nClusters)
    # take info that pertains to which block actors are placed in
    groupMembership <- blockModel$block.membership[blockModel$order.vec]
    # assign the block group values from the model back in the networkList
    networkList%v%"member" <- groupMembership
    # now run the out-of-sample prediction with TERGMs
    outSampleTERGM <- btergm(as.network.networkDynamic(networkList) ~ edges +
                    gwesp(.5, fixed = TRUE) + nodecov("member"))
    # now, simulate 100 networks from the model w/ rocpr
    # to condense the performance into a single measure, the area under
    # the curve (AUC) can be reported for both curves.
    goodFitStats <- gof(outSampleTERGM, statistics = rocpr, nsim = 100)
    # for each iteration/fold, remove and store statistics to existing list
    AUC_ROC <- c(AUC_ROC, goodFitStats$`Tie prediction`$auc.roc)
    AUC_PR <- c(AUC_PR, goodFitStats$`Tie prediction`$auc.pr)
  }
  # return the mean of each statistic pooled over the folds
  return(list(avgAUC_ROC=mean(AUC_ROC), avgAUC_PR=mean(AUC_PR)))
}

# create empty vectors to fill with each run of clusters
# which performs CV for each run
AUC_ROCvec <- NULL; AUC_PRvec <- NULL
# for each number of clusters
for (k in 2:10) {
  # run cross-validate function w/ 10 fold validation
  cvNetwork <- crossValidateFunc(nigeriaAdjMatNetworkList, nFolds=10, nClusters=k)
  # store AUC ROC and PR stats
  AUC_ROCvec <- c(AUC_ROCvec, cvNetwork$avgAUC_ROC)
  AUC_PRvec <- c(AUC_PRvec, cvNetwork$avgAUC_PR)
}
# show table of goodness-of-fit statistics
print(data.frame(k=2:10, AUC_ROC=AUC_ROCvec, AUC_PR=AUC_PRvec))
```

```
k   AUC_ROC     AUC_PR
1  2 0.5321138 0.07721483
2  3 0.5245793 0.07547592
3  4 0.5420858 0.07672680
4  5 0.5394230 0.08071837
5  6 0.5424247 0.07655145
6  7 0.5628951 0.08848244
7  8 0.5366850 0.08067494
8  9 0.5243585 0.07983234
9 10 0.5489216 0.08615140
```

```r
# (c) since we want these values to be higher
# we'll do 7 clusters
# re-create the network object
networkList <- networkDynamic(network.list=nigeriaAdjMatNetworkList)
# run the block model w/ 7 clusters
bestKblockModel <- blockmodel(networkList,
                         equiv.clust(networkList),
                         k=7)
# re-assign the groupings from the block model into the network object
bestGrouping <- bestKblockModel$block.membership[bestKblockModel$order.vec]
networkList%v%"member" <- bestGrouping
# now for the plotting of actors' interactions by group
# create colour paletter
library(RColorBrewer)
networkList %v% "col" <- brewer.pal(7, "Dark2")[networkList %v% "member"]
# generate plot (see figure 1 below)
pdf("figure1.pdf")
plot(networkList, label = network.vertex.names(networkList), label.cex=0.5,
     mode="circle", vertex.cex=log(degree(networkList))+1,
     label.col="black", vertex.col="col", vertex.border="col", edge.col="black")
dev.off()
```

3 ERGMs

a) Run a cross-sectional ERGM on the Nigerian conflict network, develop at least one or two network level hypotheses.

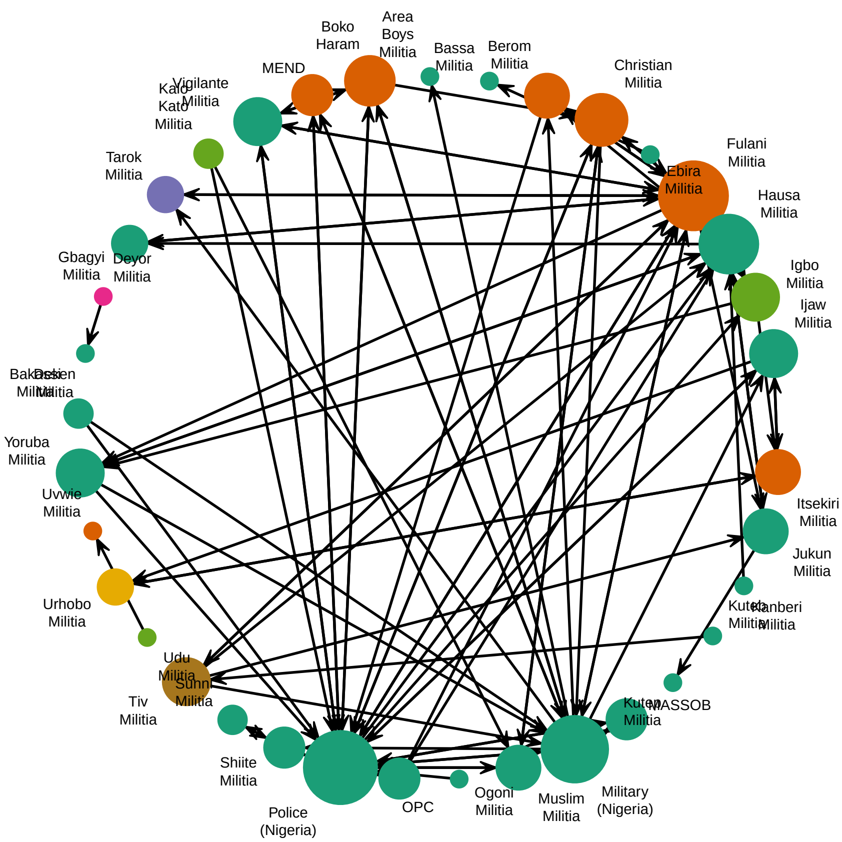
b) Briefly discuss the results.

c) Make sure to show that you checked for convergence.

```r
###########
# (3) ERGMs
###########

# (a) Run a cross-sectional ERGM on the Nigerian conflict network
```

Figure 1: Network plot by groups generated from block model with 7 clusters.



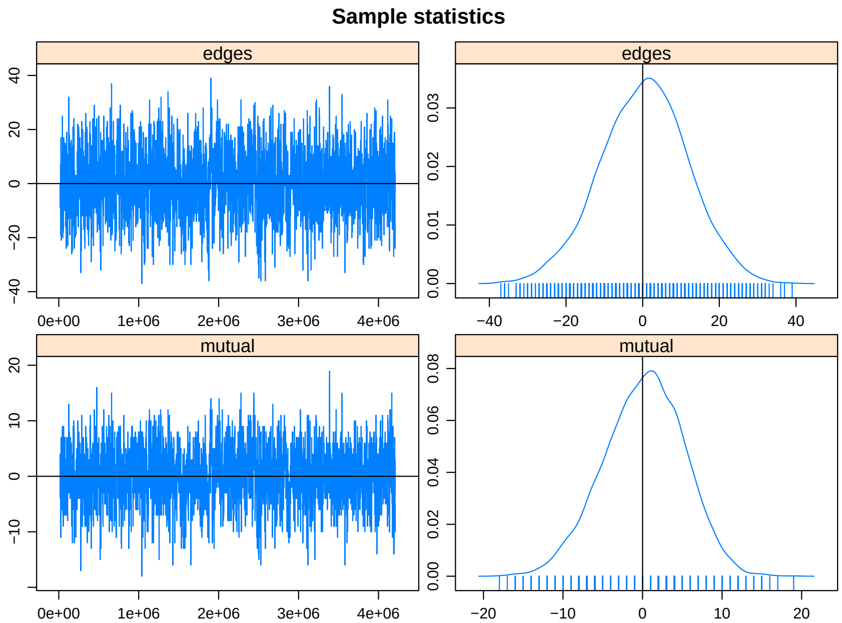

```
# H1: First order (edges: Some actors (like the gov't) are
# engaged in conflict a lot, basically intercepts per actor)
# H2: Second order (mutual: Reciprocity should be high)
# wanted to test triangle in case allies exist in the network (police,
   military), transitivity might be high?)
# but so few existed MCMC wouldn't get out of one region
library(statnet)
ERGMmodel <- ergm(as.network.matrix(nigeriaAdjMatTotalMatrix) ~ edges +
                        mutual)
```

```
# (b) results
# reciprocity is high (when one actor is attacked, they retaliate)
summary(ERGMmodel)

# (c) check for convergence
mcmc.diagnostics(ERGMmodel)
```

```
Monte Carlo MLE Results:
Estimate Std. Error MCMC % z value Pr(>|z|)
edges   -3.4543     0.1653      0   -20.9   <1e-04 ***
mutual   3.8490     0.3738      0    10.3   <1e-04 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Figure 2: MCMC chain and estimate distribution.



4 Find your own data

a) Locate data that relates to your field of interest.

b) Transform the data, or a subset of it into a matrix, and plot (similar to step 1 in Section 1).

c) Include descriptive features in your network graph (similar to step 2, but choose your own measurements).

d) Run a model, it can be any network model from the course but justify your choices!

e) Discuss the results in a brief write up. Present for 3-5 minutes to the class.