Homework for Network Analysis Workshop

Min Hee Seo

August 21st, 2018

Exercise 1. Nigeria Data

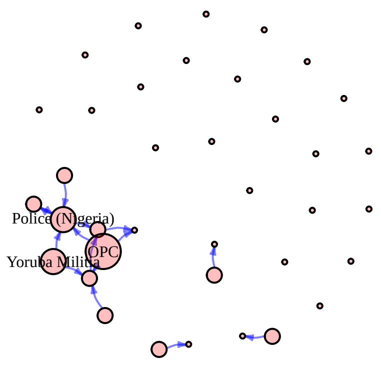
```r
# create network matrix by year and all years

# setup the working directory
setwd("/Users/minheeseo/Dropbox/Classes/2018_Classes/Network/network2018_hw1/")
# clean the workspace
rm(list = ls())
# loading data and R packages
library(igraph)
library(network)
load("nigeria.rda")
# clean the labels
nigeria$sender <- gsub("\n", " ", nigeria$sender)
nigeria$receiver <- gsub("\n", " ", nigeria$receiver)

# create an empty list where the length of list is time span
network.mat <- vector("list", length(unique(nigeria$year)))
names(network.mat) <- unique(nigeria$year)
# For each year, this loop stores one if there was a conflict
# between sender and receiver it stores 0 otherwise outcome,
# network.mat, is a list which contains 17 matrix for each
# year
time <- unique(nigeria$year)
country.sender <- unique(nigeria$sender)
country.receiver <- unique(nigeria$receiver)
for (t in 1:length(time)) {
    slice <- NULL
    empty.mat <- NULL
    empty.mat <- matrix(0, length(country.sender), length(country.receiver))
    empty.mat <- as.data.frame(empty.mat)
    rownames(empty.mat) <- country.sender
    colnames(empty.mat) <- country.receiver
    slice <- subset(nigeria, nigeria$conflict == 1 & nigeria$year ==
        time[t])
    for (i in 1:nrow(slice)) {
        empty.mat[rownames(empty.mat) == slice$sender[i], colnames(empty.mat) ==
            slice$receiver[i]] <- 1
    }
    network.mat[[t]] <- empty.mat
}

# collapse each matrix into one
tmp <- Reduce("+", network.mat)
# add to an existing list
```

```r
network.mat[[18]] <- tmp
names(network.mat)[18] <- "All Conflict"
```

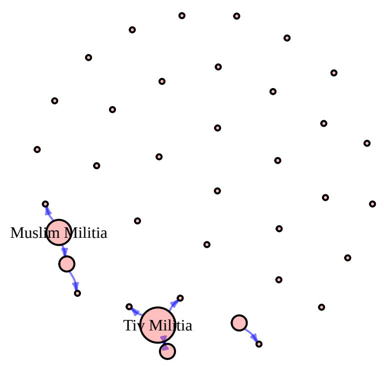
After I created 17 matrices for each year and another one for all years, I plotted them.

```r
# plot network by each year and all years

myblue <- rgb(red = 0, green = 0, blue = 1, alpha = .5)
mypink <- rgb(red = 1, green = 0, blue = 0, alpha = .25)

par(mfrow=c(2, 2), mar=c(0,0.2,1,0.2))
for(i in 1:4){
  g <- NULL
  g = graph_from_adjacency_matrix(as.matrix(network.mat[[i]]),
                                  mode='directed', weighted=TRUE, diag=F)
  tiesSum = apply(g[], 1, sum)
# condition size based on # of ties
  V(g)$size <- (tiesSum+0.5)*6
# only label if # ties greater than 10
  V(g)$label <- ifelse( tiesSum>1, V(g)$name, NA )
  V(g)$label.cex <- 0.6
  plot(g,main=paste("Year:", names(network.mat)[i]),
     vertex.label=V(g)$label,
     vertex.size=V(g)$size,
     edge.width=E(g)$weight,
     vertex.color =mypink, # change color of nodes
     vertex.label.color = "black", # change color of labels
     edge.curved=.25, # add a 25% curve to the edges
     edge.color=myblue, # change edge color to grey
     layout=layout_with_fr,
     edge.arrow.size=0.2)
}
```

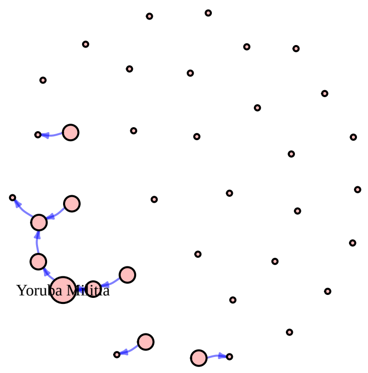
Year: 2000



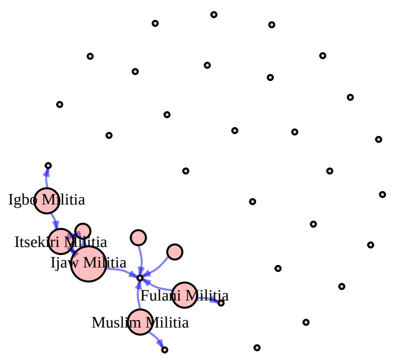
Year: 2001



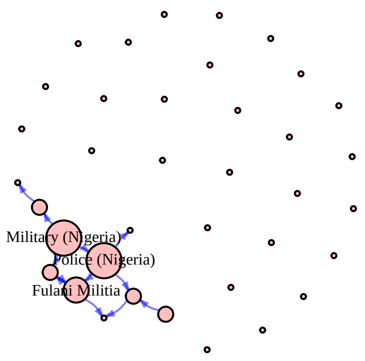
Year: 2002



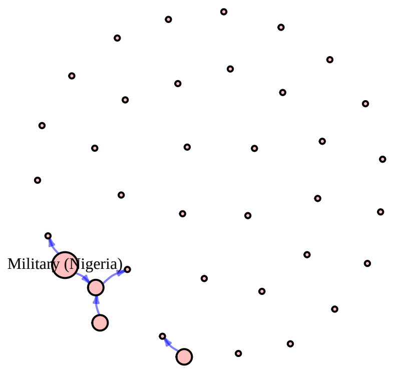
Year: 2003



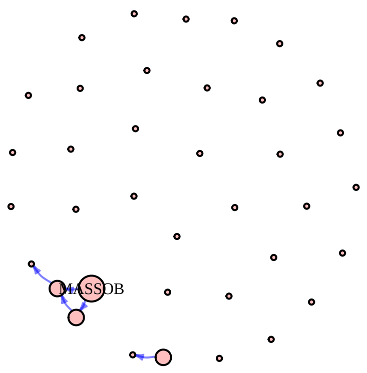
Year: 2004



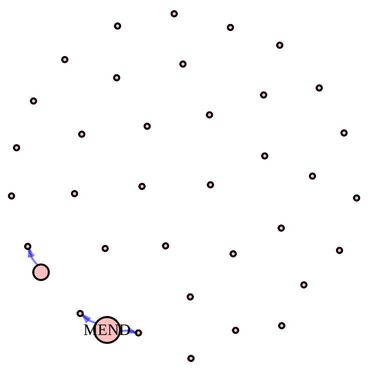
Year: 2005

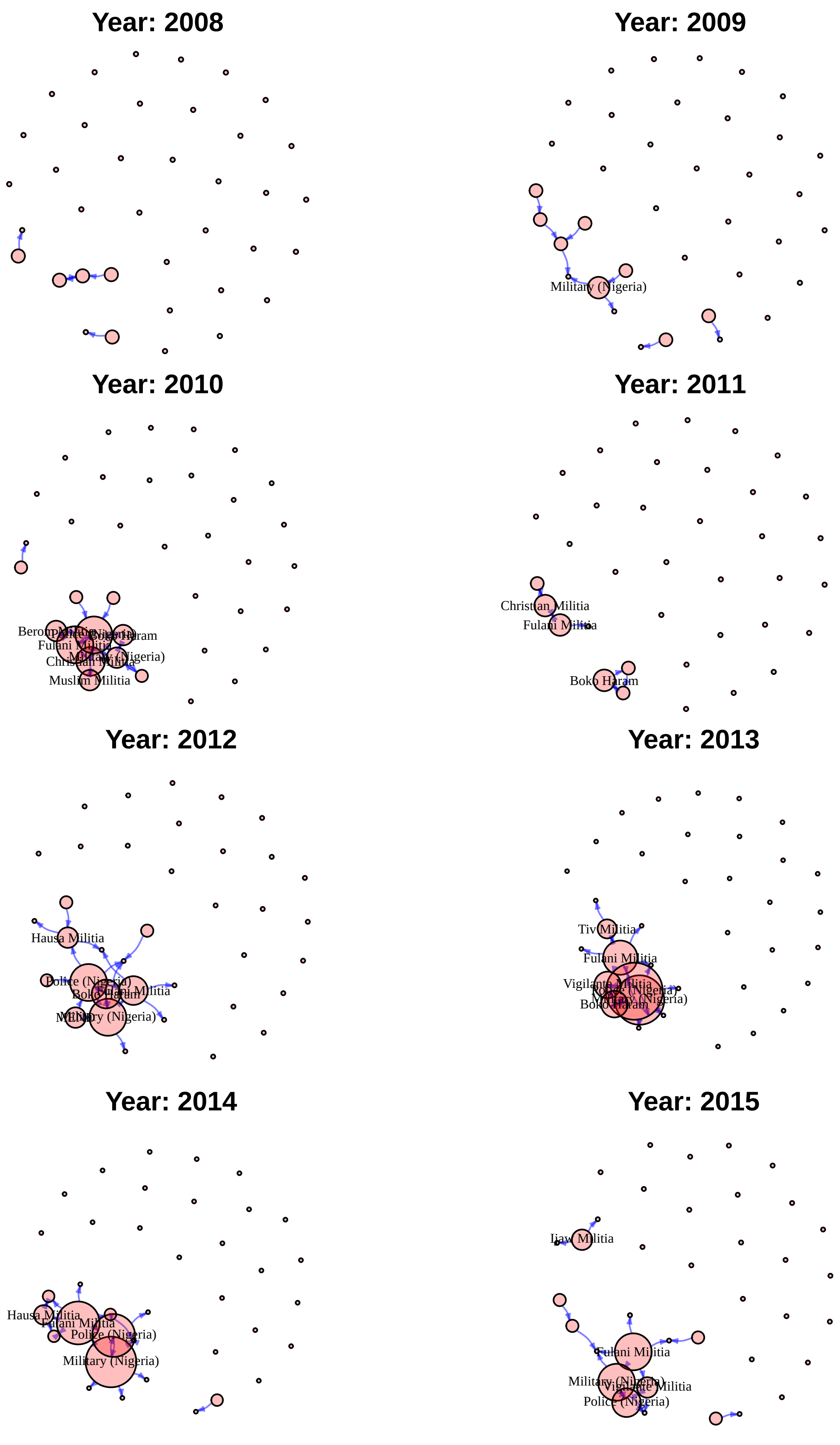


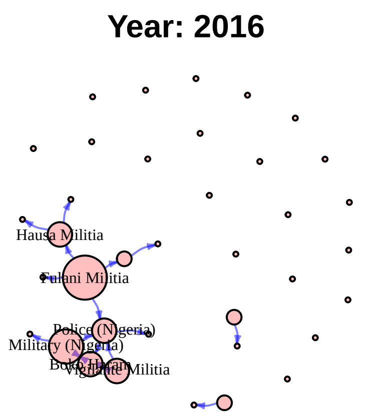
Year: 2006

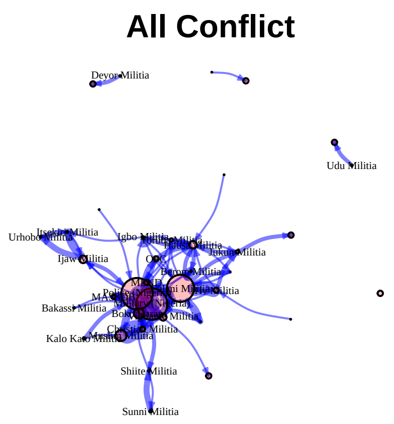


Year: 2007




Year: 2008
Year: 2009
Military (Nigeria)
Year: 2010
Berom Militia
Police (Nigeria)
Boko Haram
Fulani Militia
Christian Militia
Military (Nigeria)
Muslim Militia
Year: 2011
Christian Militia
Fulani Militia
Boko Haram
Year: 2012
Hausa Militia
Police (Nigeria)
Boko Haram
Militia
Military (Nigeria)
Year: 2013
Tiv Militia
Fulani Militia
Vigilante Militia
Police (Nigeria)
Military (Nigeria)
Boko Haram
Year: 2014
Hausa Militia
Fulani Militia
Police (Nigeria)
Military (Nigeria)
Year: 2015
Ijaw Militia
Fulani Militia
Military (Nigeria)
Vigilante Militia
Police (Nigeria)





Exercise 2. Measurements & Community detection

a

We can measure influence with the degree of nodes. For example, we can count the total number of edges connected to a node to estimate the influence of an actor. By comparing network plot (above) and the degree (below), I am quite certain that I found an influential actor using degree. Additionally, I computed eigenvector centrality to find a node that is linked to other important nodes. When I compare actors who have the most degree and eigenvector centrality, they were almost the same. I report the most influential actor for each year and all years (in terms of the number of degree).

```r
# to find an influential actor for each year

# create an empty list
influence.actor <- vector("list", 18)
names(influence.actor) <- unique(nigeria$year)
eigen.actor <- vector("list", 18)
names(eigen.actor) <- unique(nigeria$year)
# loop to find the maximum degree and eigenvector centrality
for (t in 1:17) {
    temp <- graph.adjacency(as.matrix(network.mat[[t]]), mode = "directed",
        weighted = TRUE, diag = F)
    degree <- igraph::degree(temp)
    eigen <- eigen_centrality(temp, directed = TRUE)$vector
    names(eigen) <- as.character(gsub("\n", " ", names(eigen)))
    names(degree) <- as.character(gsub("\n", " ", names(degree)))
    influence.actor[[t]] <- degree[which(degree == max(degree))]
    eigen.actor[[t]] <- eigen[which(eigen == max(eigen))]
}
# to find an influential actor overall
names(influence.actor)[18] <- "All Conflict"
names(eigen.actor)[18] <- "All Conflict"
temp <- graph.adjacency(as.matrix(network.mat[[18]]), mode = "directed",
    weighted = TRUE, diag = F)
degree <- igraph::degree(temp)
eigen <- eigen_centrality(temp, directed = TRUE)$vector
names(eigen) <- as.character(gsub("\n", " ", names(eigen)))
names(degree) <- as.character(gsub("\n", " ", names(degree)))
influence.actor[[18]] <- degree[which(degree == max(degree))]
```

```r
eigen.actor[[18]] <- eigen[which(eigen == max(eigen))]

influence.actor  # for each year and all years
```

```
## $`2000`
## Police (Nigeria)
##                6
##
## $`2001`
## Tiv Militia
##           4
##
## $`2002`
##    Hausa Militia Police (Nigeria)   Yoruba Militia
##                3                3                3
##
## $`2003`
## Itsekiri Militia Police (Nigeria)
##                5                5
##
## $`2004`
##   Fulani Militia Police (Nigeria)
##                4                4
##
## $`2005`
## Police (Nigeria)
##                3
##
## $`2006`
## Military (Nigeria)
##                  3
##
## $`2007`
## MEND
##    2
##
## $`2008`
## Military (Nigeria)
##                  3
##
## $`2009`
## Military (Nigeria)   Police (Nigeria)
##                  3                  3
##
## $`2010`
## Police (Nigeria)
##               11
##
## $`2011`
## Christian Militia
##                 4
##
## $`2012`
## Police (Nigeria)
##                 8
##
## $`2013`
## Police (Nigeria)
##               10
##
## $`2014`
## Fulani Militia
##             8
##
## $`2015`
##     Fulani Militia Military (Nigeria)   Police (Nigeria)
##                  5                  5                  5
##
## $`2016`
## Police (Nigeria)       Boko Haram
##                 5                5
##
## $`All Conflict`
## Police (Nigeria)
##               28
```

```
eigen.actor[[18]]
```

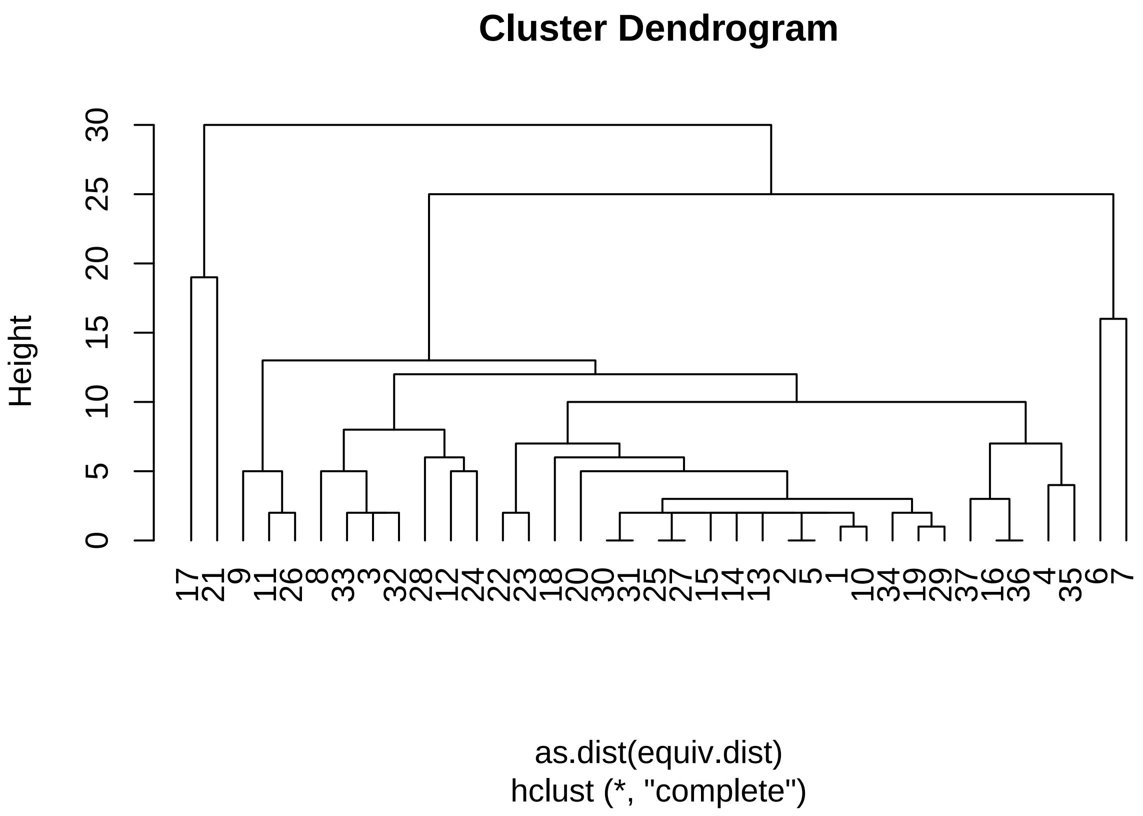
```
## Police (Nigeria)
##                 1
```

Police has the maximum degree for all years. Police also has the largest eigenvector centrality, which indicates that police is connected to other important actors.

b

Before I delve into the next question, I gauge data. Seeing a hierarchical clustering plot, it seems like there can be 3 (as little as) to 10 (as large as) groups.

```
library(sna)
g <- network(network.mat[[18]], directed = T)
eclusts <- equiv.clust(g)
plot(eclusts, hang = -1)
```



```r
# running the block model using cross-validation using
# different K (3 to 10)
library(caret)
library(intergraph)
library(btergm)
# check whether name order is the same
nigeria.dat <- network(as.matrix(network.mat[[18]]), directed = T,
    loops = F)
identical(rownames(network.mat[[18]]), nigeria.dat %v% "vertex.names")
roc <- pr <- c()

nigeriaCV <- function(dat, folds = 10, k = 7) {
    set.seed(51253)
    # create folds
    index <- createFolds(y = unique(rownames(network.mat[[18]])),
        k = folds, returnTrain = T)
    for (i in 1:folds) {
        # create train/test dataset
        removeVertex <- which(rownames(network.mat[[18]]) %in%
            rownames(network.mat[[18]])[-index[[i]]])
        outofsamplevertex <- which(rownames(network.mat[[18]]) %in%
            rownames(network.mat[[18]])[index[[i]]])
        train <- delete.vertices(nigeria.dat, removeVertex)
        test <- delete.vertices(nigeria.dat, outofsamplevertex)
        train.cluster <- equiv.clust(train)
        # run block model
        train.blockmodel <- blockmodel(train, train.cluster,
            k = k)
        bmembership <- train.blockmodel$block.membership[order(train.blockmodel$order.vec)]
        train %v% "group" <- bmembership
        # fit a model
        modelout <- ergm(train ~ edges + nodecov("group"))
        # predict to out of sample
        gof.mod <- gof(modelout, rocprgof = T, target = test,
            outofsample = T, statistics = c(rocpr), nsim = 100)
        roc[i] <- gof.mod$"Tie prediction"$auc.roc
        pr[i] <- gof.mod$"Tie prediction"$auc.pr
    }
}
```

Exercise 3. ERGMs

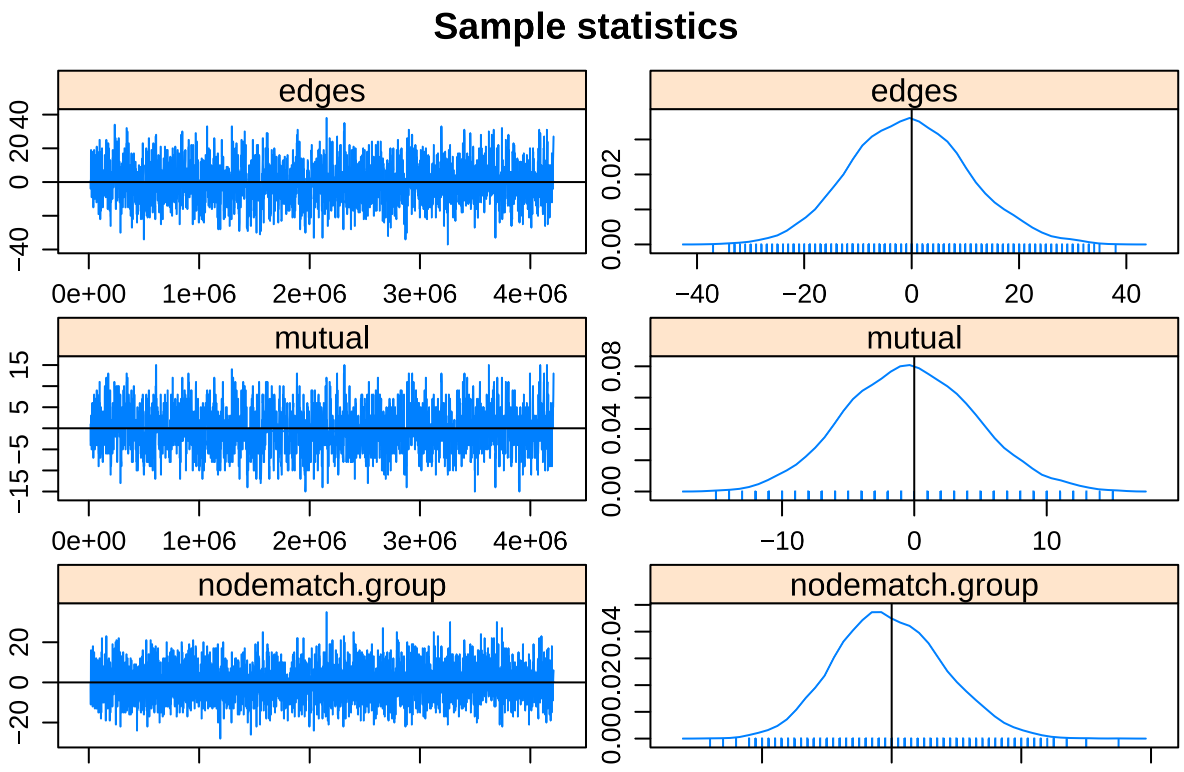
```
library(statnet)
set.seed(688346)
nigeria <- as.matrix(network.mat[[18]])
diag(nigeria) <- NA
nigeria <- as.network.matrix(nigeria)
temp <- graph.adjacency(as.matrix(network.mat[[18]]), mode = "directed",
    weighted = TRUE, diag = F)
nigeria %v% "group" <- ifelse(names(V(temp)) == "Police (Nigeria)" |
    names(V(temp)) == "Military (Nigeria)", 1, 0)
ergm.network <- ergm(nigeria ~ edges + mutual + nodematch("group"))
summary(ergm.network)
```

```
##
## ==========================
## Summary of model fit
## ==========================
##
## Formula:   nigeria ~ edges + mutual + nodematch("group")
##
## Iterations:  2 out of 20
##
## Monte Carlo MLE Results:
##                 Estimate Std. Error MCMC % p-value
## edges            -2.1988     0.2350      0  <1e-04 ***
## mutual            3.4449     0.4064      0  <1e-04 ***
## nodematch.group  -1.4725     0.1972      0  <1e-04 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      Null Deviance: 1846.5  on 1332  degrees of freedom
##  Residual Deviance:  519.7  on 1329  degrees of freedom
##
## AIC: 525.7    BIC: 541.3    (Smaller is better.)
```

Here, my hypotheses is that a conflict in Nigeria would be reciprocal, and the government actors wouldn't involve in a conflict together. Since my hypotheses test reciprocity and group-homophily, I include "mutual" and "nodematch" variables in an ERGM model. The result shows that the number of ties, reciprocity, and the group-homophily variables are all statistically significant. The coefficient of mutual variable is positive and its effect size is large. This implies that reciprocity exists in Nigeria's conflict. The coefficient for nodematch is negative and significant as I expected. Government actors are less like to involve in a conflict with each other.

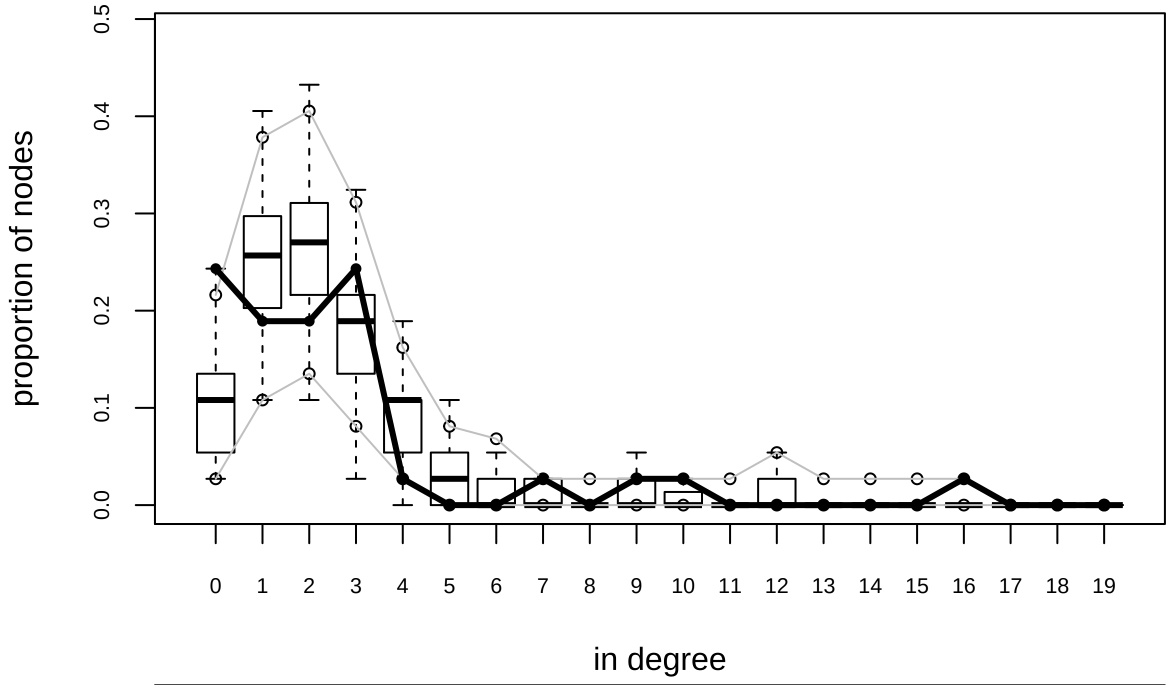
```
mcmc.diagnostics(ergm.network)
```

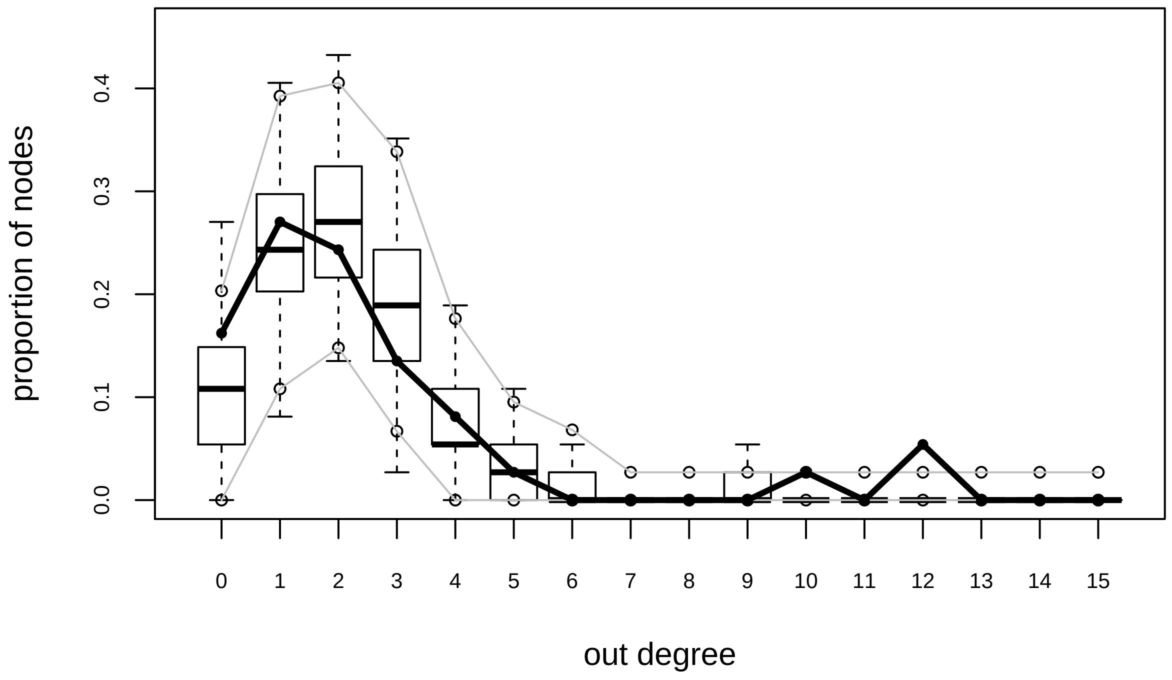
```
## Sample statistics summary:
##
## Iterations = 16384:4209664
## Thinning interval = 1024
## Number of chains = 1
## Sample size per chain = 4096
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                    Mean     SD Naive SE Time-series SE
## edges           -0.3462 10.994  0.17178         0.3447
## mutual          -0.2629  4.859  0.07592         0.1748
## nodematch.group -0.2202  8.317  0.12995         0.1864
##
## 2. Quantiles for each variable:
##
##                 2.5% 25% 50% 75% 97.5%
## edges            -21  -8  -1   7    22
## mutual           -10  -4   0   3     9
## nodematch.group  -16  -6  -1   5    17
##
##
## Sample statistics cross-correlations:
##                     edges    mutual nodematch.group
## edges           1.0000000 0.8464189       0.7672229
## mutual          0.8464189 1.0000000       0.5440742
## nodematch.group 0.7672229 0.5440742       1.0000000
##
## Sample statistics auto-correlation:
## Chain 1
##              edges    mutual nodematch.group
## Lag 0    1.0000000 1.0000000      1.00000000
## Lag 1024 0.4596140 0.5949033      0.28880039
## Lag 2048 0.2770329 0.3747001      0.10349614
## Lag 3072 0.2017980 0.2784819      0.04543666
## Lag 4096 0.1711471 0.2136351      0.04572211
## Lag 5120 0.1195025 0.1700790      0.02486071
##
## Sample statistics burn-in diagnostic (Geweke):
## Chain 1
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##           edges          mutual nodematch.group
##          1.9416          1.9910         -0.4924
##
## Individual P-values (lower = worse):
##           edges          mutual nodematch.group
##      0.05218638      0.04647966      0.62243139
## Joint P-value (lower = worse):  0.1421544 .
```

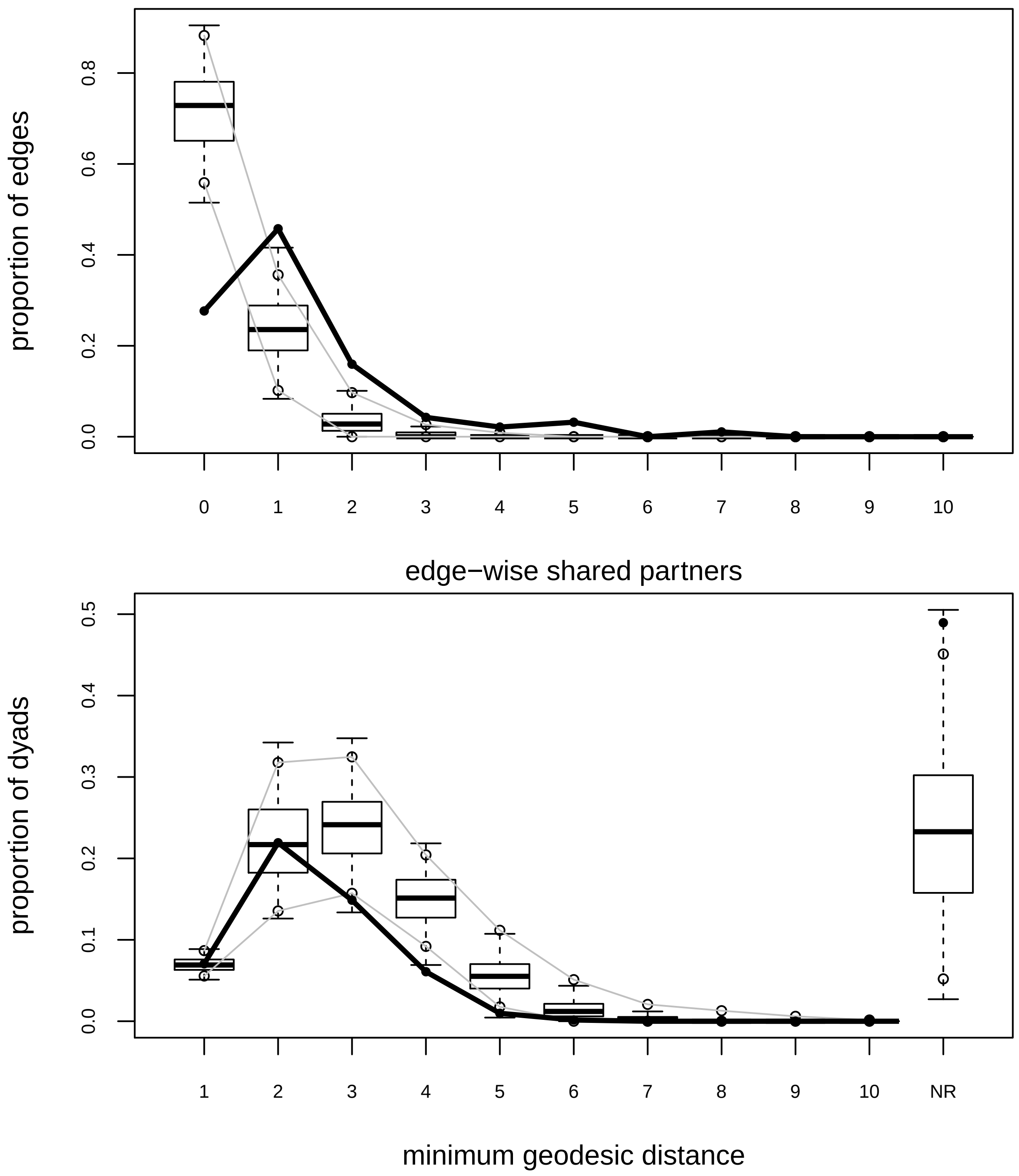


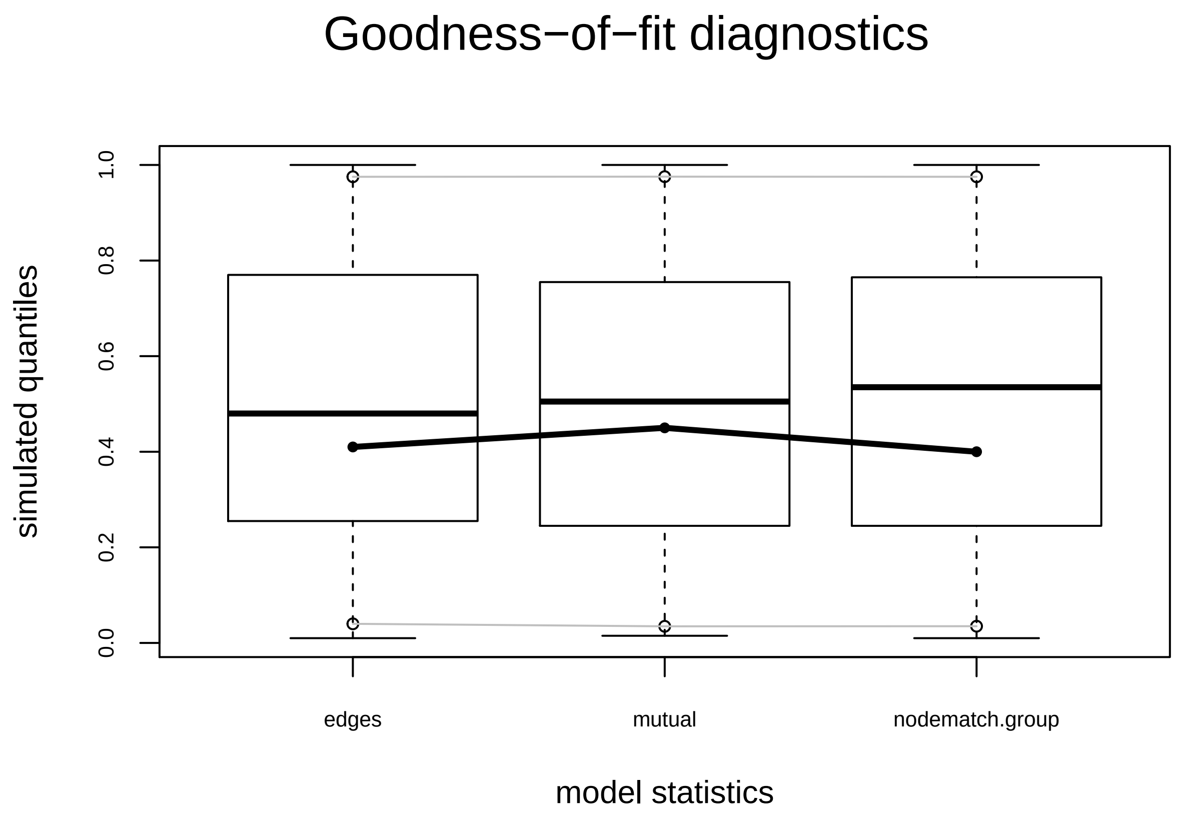
```
##
## MCMC diagnostics shown here are from the last round of simulation, prior to computation of final para
```

```
ergm.network.gof = gof(ergm.network)
plot(ergm.network.gof)
```


proportion of nodes
0.0
0.1
0.2
0.3
0.4
0.5
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
in degree


proportion of nodes
0.0
0.1
0.2
0.3
0.4
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
out degree


proportion of edges
0.0
0.2
0.4
0.6
0.8
0
1
2
3
4
5
6
7
8
9
10
edge−wise shared partners
proportion of dyads
0.0
0.1
0.2
0.3
0.4
0.5
1
2
3
4
5
6
7
8
9
10
NR
minimum geodesic distance



To check the model fit, first, I examine the MCMC chains. They are well-mixed and stationary. Next, I check a goodness-of-fit of the model. Observed distributions (black lines) are generally within simulated ones. By looking at these plots, my model seems to be a reasonable approximation of the network. Interestingly, for minimum geodescic distance plot, you can see NR is quite high. This indicates “Nonreachable” distance. From exercise 1, we can see that certain nodes never interact with other nodes, and these nodes are unreachable. This is why we see NR in this graph. Other than this, they all look normal.