

Homework

```
load('nigeria.rda')

years = sort(unique(nigeria$year))
groups = with(nigeria, intersect(sender, receiver))
n = length(groups)
adjmat = matrix(0,nrow = n, ncol = n, dimnames = list(groups, groups))

nigerialist = lapply(years, function(t) {
  slice = nigeria[nigeria$year==t,]
  positivecases = slice[slice$conflict==1,]
  for(i in 1:nrow(positivecases)){
    sender= as.character(positivecases$sender[i])
    receiver= as.character(positivecases$receiver[i])
    adjmat[sender, receiver]=1
  }
  return(as.network.matrix(adjmat))
})

names(nigerialist) <- years
```

```

par(mfrow = c(1,2))
for (i in 1:length(years)) {
  g <- nigerialist[[i]]
  g1 <- asIgraph(g)
  deg <- degree(g1, mode="all")
  V(g1)$size <- deg*4
  dist <- ((-1.4)*(V(g1)$size-min(V(g1)$size)))/(max(V(g1)$size)-min(V(g1)$size))+1.4
  plot(g1, edge.arrow.size=.08, edge.arrow.width=0.8, edge.curved=.05, edge.color="black",
       vertex.label=labels, vertex.label.color="black",
       vertex.label.dist=dist, vertex.label.cex = .3, vertex.color="grey",
       main=years[i], layout=layout_with_kk)
  if ((i %% 2) == 0) par(mfrow = c(1,2))

  #degree
  deg_total[i] <- paste(groups[which( deg == max(deg) )], collapse = ' ')
  deg_total[i] <- gsub("\nMilitia", "", deg_total[i])
  deg_total[i] <- gsub("\n\\(Nigeria\\)", "", deg_total[i])

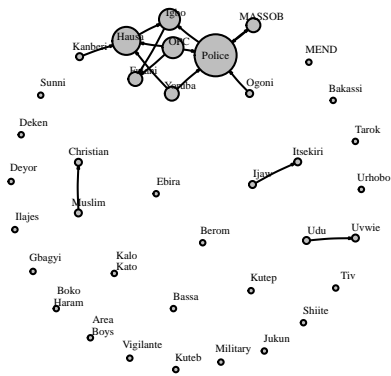
  #degree in
  deg_in <- degree(g1, mode="in")
  deg_in_total[i] <- paste(groups[which( deg_in == max(deg_in) )], collapse = ' ')
  deg_in_total[i] <- gsub("\nMilitia", "", deg_in_total[i])
  deg_in_total[i] <- gsub("\n\\(Nigeria\\)", "", deg_in_total[i])

  #degree out
  deg_out <- degree(g1, mode="out")
  deg_out_total[i] <- paste(groups[which( deg_out == max(deg_out) )], collapse = ' ')
  deg_out_total[i] <- gsub("\nMilitia", "", deg_out_total[i])
  deg_out_total[i] <- gsub("\n\\(Nigeria\\)", "", deg_out_total[i])

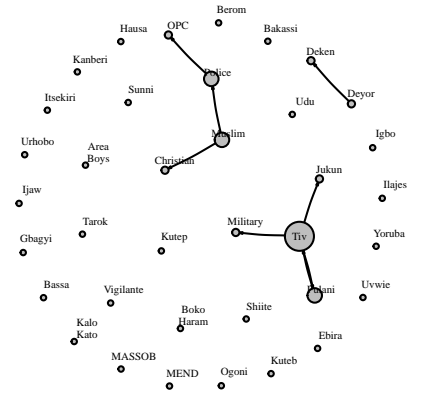
```

}

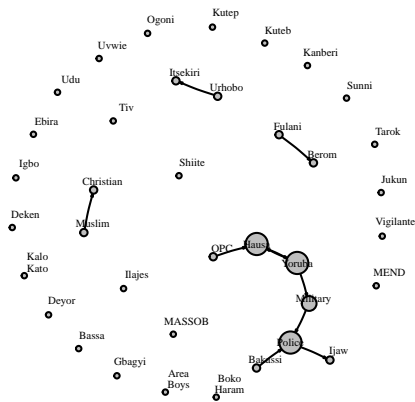
2000



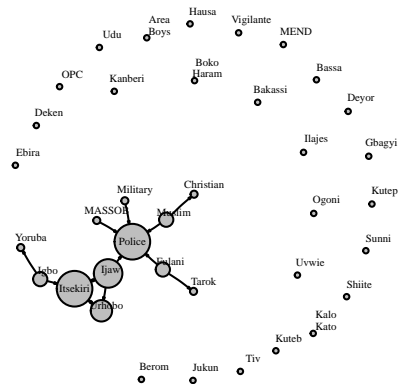
2001



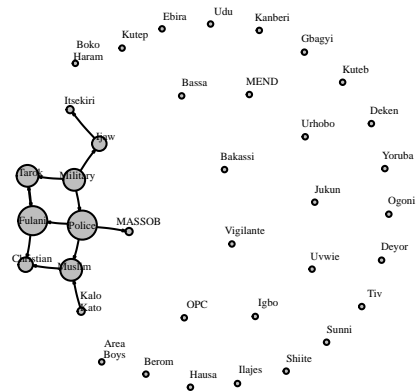
2002



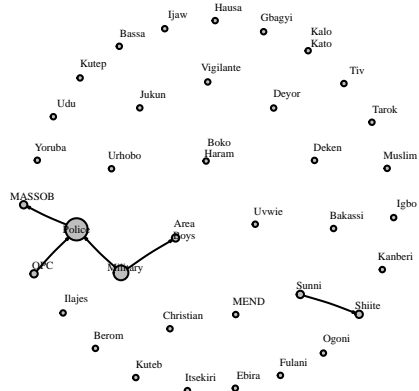
2003



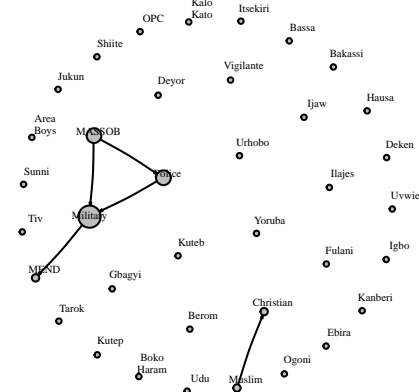
2004



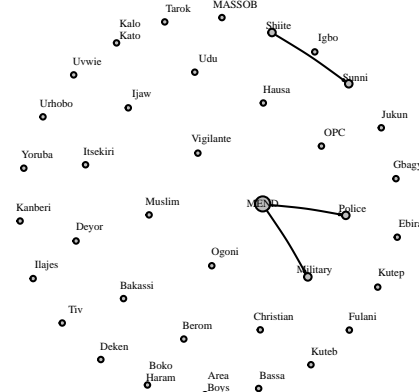
2005



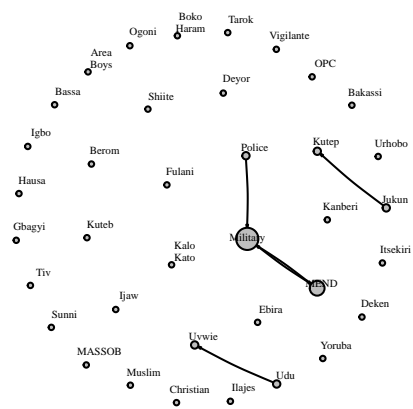
2006



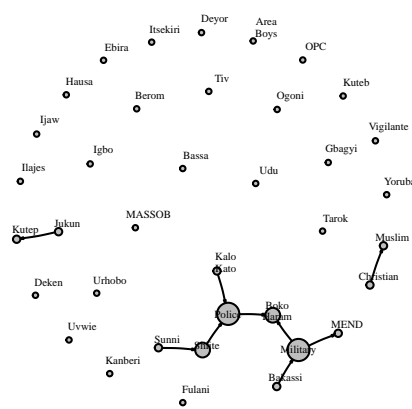
2007



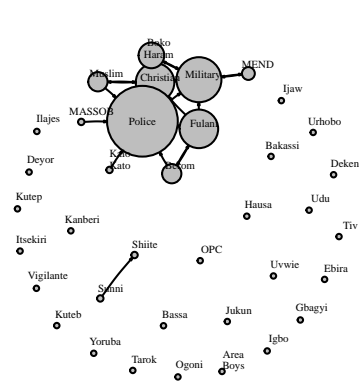
2008



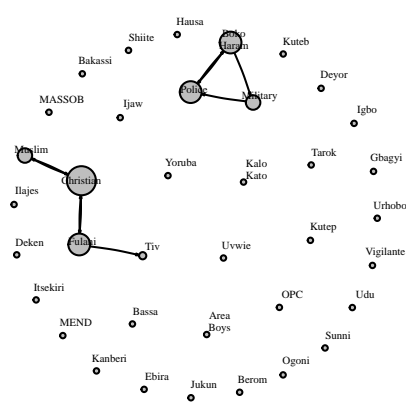
2009



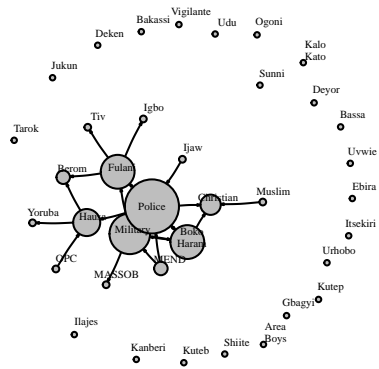
2010



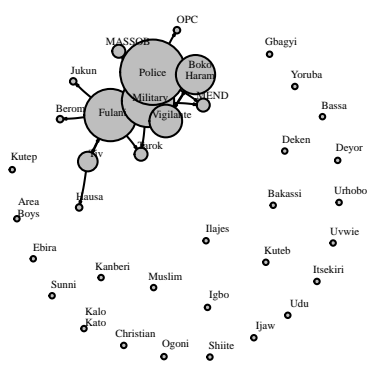
2011



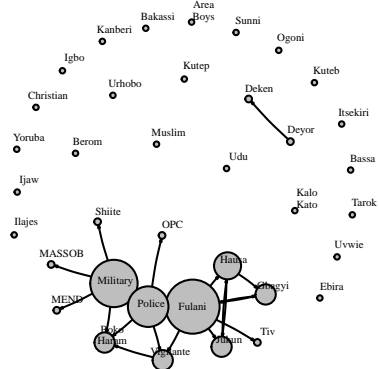
2012



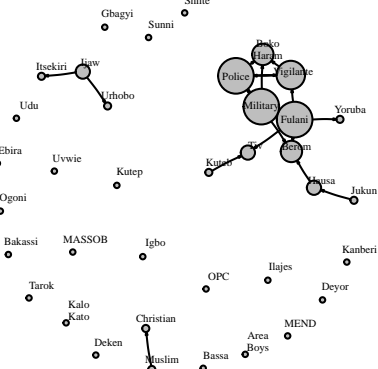
2013



2014



2015




```

#Degree
deg_all <- sna::degree(nigeriaDyn)
paste(groups[which( deg_all == max(deg_all) )], collapse = ' ')

## [1] "Police\n(Nigeria)"

deg_in_all <- sna::degree(nigeriaDyn, cmode="indegree")
paste(groups[which( deg_in_all == max(deg_in_all) )], collapse = ' ')

## [1] "Police\n(Nigeria)"

deg_out_all <- sna::degree(nigeriaDyn, cmode="outdegree")
paste(groups[which( deg_out_all == max(deg_out_all) )], collapse = ' ')

## [1] "Fulani\nMilitia Police\n(Nigeria)"

#Eigenvector centrality
eigen <- sna::evcent(nigeriaDyn)
paste(groups[which( eigen == max(eigen) )], collapse = ' ')

## [1] "Police\n(Nigeria)"

#Degree year by year
as.data.frame(cbind(years, "Higher Degree (all)"=deg_total))

##      years      Higher Degree (all)
## 1  2000          Police
## 2  2001             Tiv
## 3  2002   Hausa Police Yoruba
## 4  2003      Itsekiri Police
## 5  2004      Fulani Police

```



```
## 6 2005 Police
## 7 2006 Military
## 8 2007 MEND
## 9 2008 Military
## 10 2009 Military Police
## 11 2010 Police
## 12 2011 Christian
## 13 2012 Police
## 14 2013 Police
## 15 2014 Fulani
## 16 2015 Fulani Military Police
## 17 2016 Police Boko\nHaram
```

```
as.data.frame(cbind(years, "Higher Degree (in)"=deg_in_total))
```

```
##      years      Higher Degree (in)
## 1  2000      Police
## 2  2001 Christian Fulani Jukun Military OPC Police Tiv Deken
## 3  2002      Hausa Police
## 4  2003      Police
## 5  2004      Christian Fulani Muslim Tarok
## 6  2005      Police
## 7  2006      Military
## 8  2007      Military Police Sunni
## 9  2008      Military
## 10 2009      Police Boko\nHaram
## 11 2010      Police
## 12 2011      Christian Police
```

```
## 13 2012 Police
## 14 2013 Fulani
## 15 2014 Fulani Boko\nHaram
## 16 2015 Berom Boko\nHaram
## 17 2016 Police Boko\nHaram
```

```
as.data.frame(cbind(years, "HigherDegree (out)"=deg_out_total))
```

```
##      years      HigherDegree (out)
## 1  2000      OPC
## 2  2001      Tiv
## 3  2002      Yoruba
## 4  2003      Ijaw
## 5  2004      Military Police
## 6  2005      Military
## 7  2006      MASSOB
## 8  2007      MEND
## 9  2008 Jukun Military Police Udu MEND
## 10 2009      Military
## 11 2010      Fulani Police
## 12 2011      Christian Fulani Boko\nHaram
## 13 2012      Military Police
## 14 2013      Police
## 15 2014      Military
## 16 2015      Fulani Military
## 17 2016      Fulani
```

As seen bellow, k=3 seems to be the best number of groups according to both the AUC (PR) and AUC(ROC).

```

set.seed(1234)

cross <- function(data, f=10, k=2) {
  eclusts <- equiv.clust(data)
  BlockM <- blockmodel(data, eclusts, k=k)
  member_or <- BlockM$block.membership[BlockM$order.vec]
  set.seed(1234)
  folds <- createFolds(groups, k=f, returnTrain = T)
  tot_pr <- c()
  tot_roc <- c()
  for (i in 1:f) {
    nigeriaDyn2 <- data
    network::delete.vertices(nigeriaDyn2, (1:n)[-folds[[i]]])
    eclusts <- equiv.clust(nigeriaDyn2)
    BlockM <- blockmodel(nigeriaDyn2, eclusts, k=k)
    member <- BlockM$block.membership[BlockM$order.vec]

    tot_pr <- c(tot_pr, pr.curve( member_or[folds[[i]]], member)$auc.integral)
    tot_roc <- c(tot_pr, roc.curve( member_or[folds[[i]]], member)$auc.integral)

  }
  return(list(PR=mean(tot_pr), ROC=mean(tot_roc)))
}

pr_results <- c()

```

```

roc_results <- c()
for (k in 2:10) {
  cross_results <- cross(nigeriaDyn, k=k)
  pr_results <- c(pr_results, cross_results$PR)
  roc_results <- c(roc_results, cross_results$ROC)
}
data.frame(k=2:10, PR=pr_results, ROC=roc_results)

```

```

##      k      PR      ROC
## 1  2 0.4940981 0.4940981
## 2  3 0.5046866 0.5046866
## 3  4 0.4930574 0.4930574
## 4  5 0.4757369 0.4757369
## 5  6 0.4750518 0.4750518
## 6  7 0.4904888 0.4904888
## 7  8 0.4954614 0.4954614
## 8  9 0.4953397 0.4953397
## 9 10 0.5025631 0.5025631

```

```

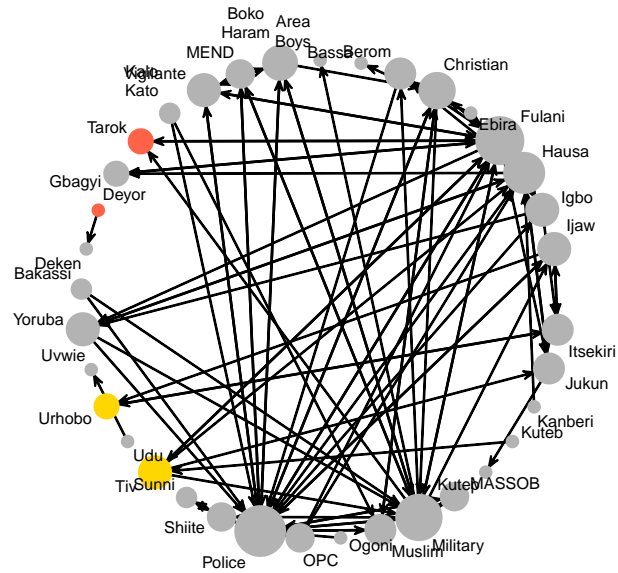
eclusts <- equiv.clust(nigeriaDyn)
BlockM <- blockmodel(nigeriaDyn, eclusts, k=3)
member_or <- BlockM$block.membership[BlockM$order.vec]

nigeriaDyn%v%"member" <- member_or
#nigeriaDyn %v% "member"

nigeriaDyn %v% "col" <- c("gray70", "tomato", "gold")[nigeriaDyn %v% "member"]

```

```
plot(nigeriaDyn, label = labels, label.cex=0.5, mode="circle", vertex.cex=log(deg_all)+1,
     label.col="black", vertex.col="col", vertex.border="col", edge.col="black")
```



The first logical hypothesis would be that if a reciprocal tie is present then the odds of a tie would be higher. If an actor attacks, the odd of a retaliation should be higher. For this, it is important to include the term mutual in the ERGM. Including an attribute variable of whether the actor is the Police or the Military may be another important variable to include. It should be expected no attacks between them. Another hypothesis could be that the more an actor has 2 stars the more likely that actor would attack others. This may be important given that two popular actors are in consideration, the police and the military. If two actors attack another third, it should be more likely for them not to attack themselves. For this, it is included the term triangles. And we may discount each additional tie by including the

term gwidegree.

As expected the coefficient for mutual is positive so it is very likely retaliation among actors. The strong negative coefficient for the group-homophily term shows that the police and the military do not attack themselves. The triangle couldn't be estimated because there are few triangles in this network.

```
nigeria1 <- nigerialist[[1]]
nigeria1%v%"gov" <- ifelse(groups=="Police\n(Nigeria)" | groups=="Military\n(Nigeria)", 1, 0)
m = ergm(nigeria1 ~ edges + mutual + nodematch("gov") + istar(2) + triangle + gwidegree(decay = 0.5, fixed = TRUE))
summary(m)
```

```
##
## =====
## Summary of model fit
## =====
##
## Formula:   nigeria1 ~ edges + mutual + nodematch("gov") + istar(2) + triangle +
##           gwidegree(decay = 0.5, fixed = TRUE)
##
## Iterations: 2 out of 20
##
## Monte Carlo MLE Results:
```

	Estimate	Std. Error	MCMC %	p-value
edges	-0.004633	2.665354	0	0.9986
mutual	2.299057	1.196665	0	0.0549 .
nodematch.gov	-1.127850	0.505145	0	0.0257 *
istar2	-0.703172	1.002197	0	0.4830
triangle	-Inf	0.000000	0	<1e-04 ***

```
## gwidegree      -4.718878    2.873098      0  0.1007
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##      Null Deviance: 1847  on 1332  degrees of freedom
## Residual Deviance:  NaN  on 1326  degrees of freedom
##
## AIC: NaN      BIC: NaN      (Smaller is better.)
##
## Warning: The following terms have infinite coefficient estimates:
## triangle
```

Bellow it is shown the MCM diagnostics, it seems well-mixed and with stationary chains.

```
## 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        -0.85474 5.289  0.08264      0.08884
## mutual         0.03882 1.129  0.01764      0.01868
## nodematch.gov  0.63550 4.160  0.06500      0.06828
```

```

## istar2      -2.19727 5.754 0.08991      0.09136
## gwidegree   0.03342 3.321 0.05190      0.05439
##
## 2. Quantiles for each variable:
##
##           2.5%    25%    50%    75%    97.5%
## edges      -10.000 -5.000 -1.0000 2.000 11.000
## mutual     -1.000 -1.000 0.0000 1.000 3.000
## nodematch.gov -6.000 -2.000 0.0000 3.000 10.000
## istar2     -10.000 -7.000 -3.0000 1.000 11.000
## gwidegree  -5.848 -2.396 -0.1423 2.178 6.957
##
##
## Sample statistics cross-correlations:
##           edges    mutual nodematch.gov    istar2 gwidegree
## edges      1.0000000 0.6348831      0.8448027 0.8327898 0.9445675
## mutual     0.6348831 1.0000000      0.4207942 0.5473948 0.5903634
## nodematch.gov 0.8448027 0.4207942      1.0000000 0.5959572 0.8533925
## istar2     0.8327898 0.5473948      0.5959572 1.0000000 0.6144747
## gwidegree  0.9445675 0.5903634      0.8533925 0.6144747 1.0000000
##
## Sample statistics auto-correlation:
## Chain 1
##           edges    mutual nodematch.gov    istar2    gwidegree
## Lag 0      1.00000000 1.000000000      1.00000000 1.000000e+00 1.000000000
## Lag 1024   0.07206654 0.126972968      0.02687274 8.049122e-02 0.046796963
## Lag 2048   0.01631024 0.033883995      0.02285306 -2.683274e-05 0.016064825

```

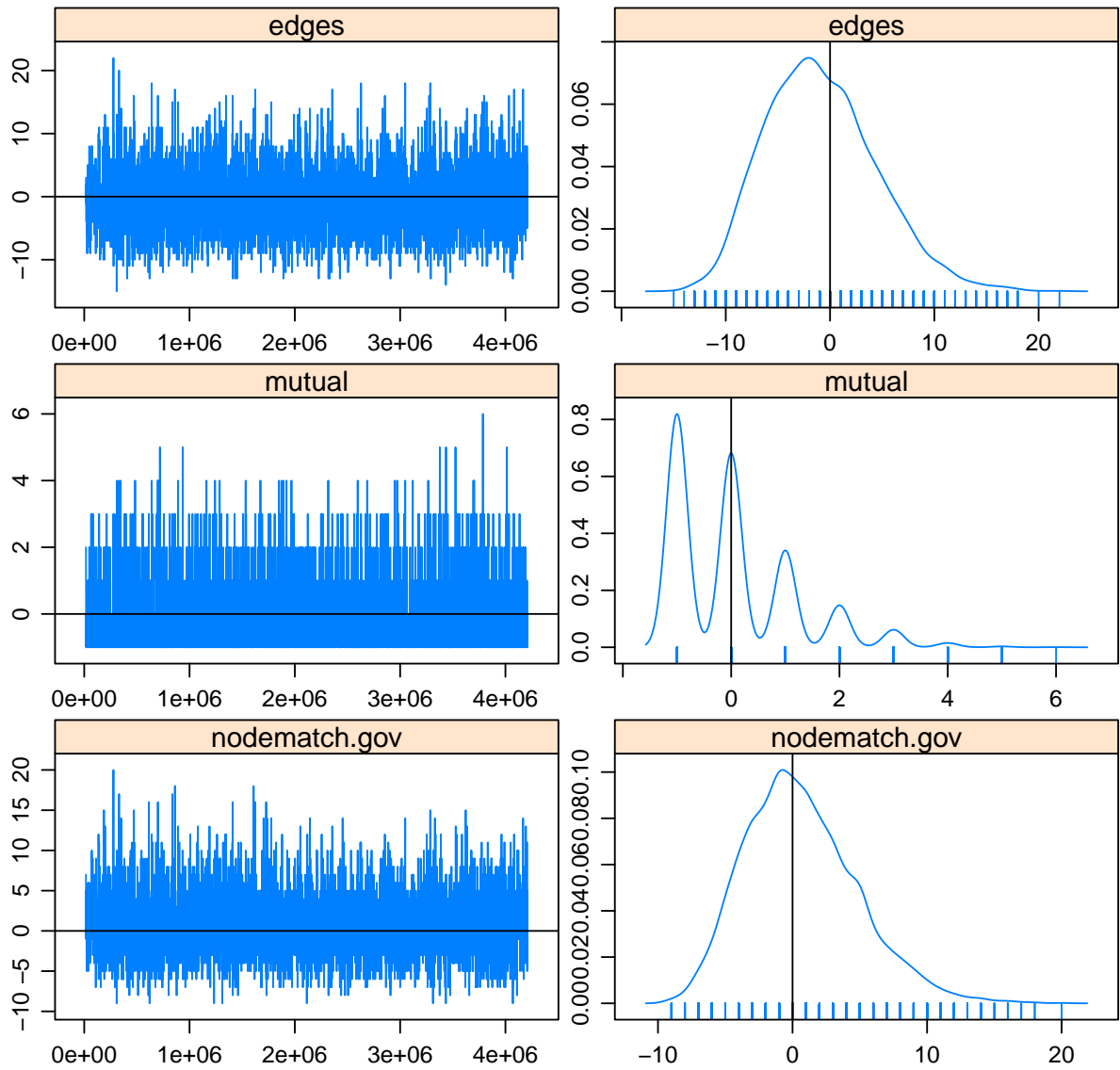


```

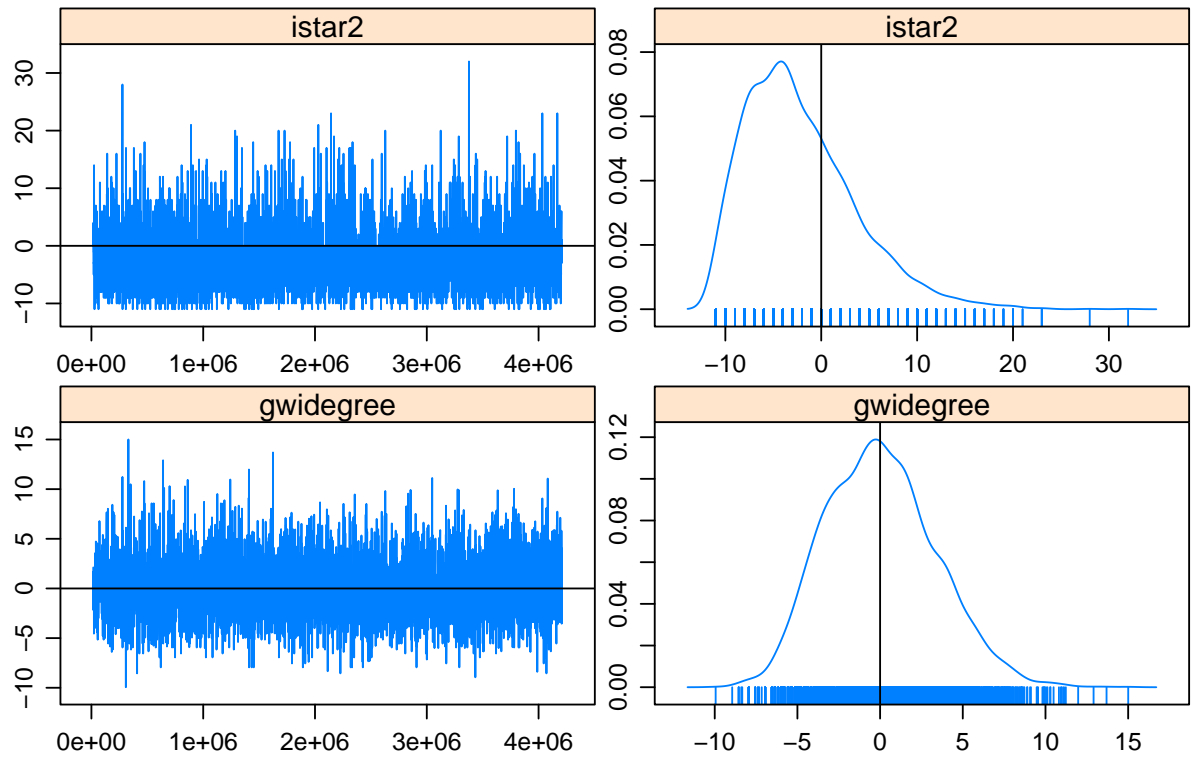
## Lag 3072 -0.01388164 -0.002715401 -0.01066121 -2.788065e-02 -0.004236398
## Lag 4096 -0.02763104 -0.030105981 -0.02546932 -3.530011e-02 -0.014735790
## Lag 5120 -0.01684109 -0.028573946 -0.01513682 -1.656373e-02 -0.013808677
##
## Sample statistics burn-in diagnostic (Geweke):
## Chain 1
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##          edges          mutual nodematch.gov          istar2          gwidegree
##      -0.04145      -0.04490          0.66493          0.18824          -0.14042
##
## Individual P-values (lower = worse):
##          edges          mutual nodematch.gov          istar2          gwidegree
##      0.9669375      0.9641838          0.5060935          0.8506890          0.8883275
## Joint P-value (lower = worse): 0.7508206 .

```

Sample statistics



Sample statistics



##

MCMC diagnostics shown here are from the last round of simulation, prior to computati