Network Practice Examples 9/23

#Data Overview

Dataset 1: Twitter data

This data consisits of "circles" from Twitter. A total number of 973 ego nodes and their social network information were crawled from the public sources of Twitter. The information of all the neighbours (the ones the ego node follow) were collected, including whether the neighbours follow each other on Twitter. Edges are directed in this dataset.

A total number of 81,306 nodes, and 1,768,148 edges were included in the dataset. The size of each specific ego node centering network varies in this dataset with the number of nodes and edges in each network shown below.

```
lines2<-read.table(file="./twitter/twitter/lines2.txt")
lines2<-lines2[-1,]
summary(lines2$V1)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##     10.0     84.0    140.0    274.9    195.0 133857.0
```

```
lines<-read.table(file="./twitter/twitter/lines.txt")
lines<-lines[-1,]
summary(lines$V1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       5     661    1752    4692    3507 2286909
```

The network with the smallest number of neighbour has 10, while the largest has 133,857. The smallest network only have 5 edges, while the largest network has 2 million edges. They did not mention how the ego were selected, but it is assumed they follow every nodes in the corresponding file.

Dataset 2: Enron email

This dataset covers all the email communication within a dataset of around half million emails. This data was originally made public, and posted to the web, by the Federal Energy Regulatory Commission during its investigation.

In this analysis, we only used the data subsetted from the original dataset with only enron's internal email address. In total, there are 11,393 senders, 11,150 receivers, and 323,450 directed edges shared between them.

```
# This part of the code is to clean the Enron email data, such that we can have a list of the senders a
#Two datasets were generated based on this part of the code. The first dataset is all the senders and r
################################################################################
#library("stringr")
#file1<-read.csv(file="./Enron/emails.csv")
#Only 14 people are multiple receivers
#file1$message<-str_extract(file1$message,"From: [a-zA-Z.@]*\nTo: [a-zA-Z.@]*")
#tmp<-strsplit(file1$message,"\n")
#file1<-NA
#tmp<-lapply(tmp, na.exclude)
#tmp<-as.data.frame(matrix(unlist(tmp),ncol=2,byrow=T))
#colnames(tmp)<-c("From","To")
#tmp$From<-str_replace(tmp$From,"From: ","")
#tmp$To<-str_replace(tmp$To,"To: ","")
#write.csv(file="Enron_cleaned.csv",tmp,quote=F,row.names=F)
#index<-grep("enron.com",tmp$To)
```

```
#tmp<-tmp[index,]
#index<-grep("enron.com",tmp$From)
#tmp<-tmp[index,]
#write.csv(file="./Enron/Enron_cleaned_enroncom.csv",tmp,quote=F,row.names=F)
```

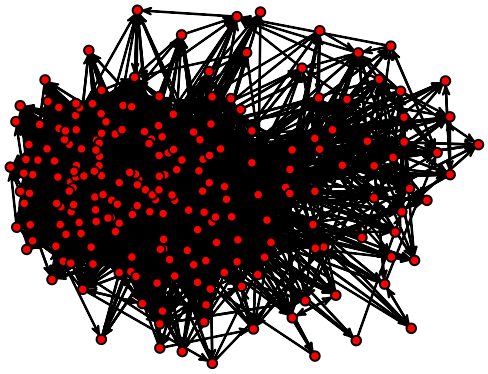
The following is a general overview of the connectivity of the network.

```
library("ggplot2")
library("gridExtra")
library("network")
```

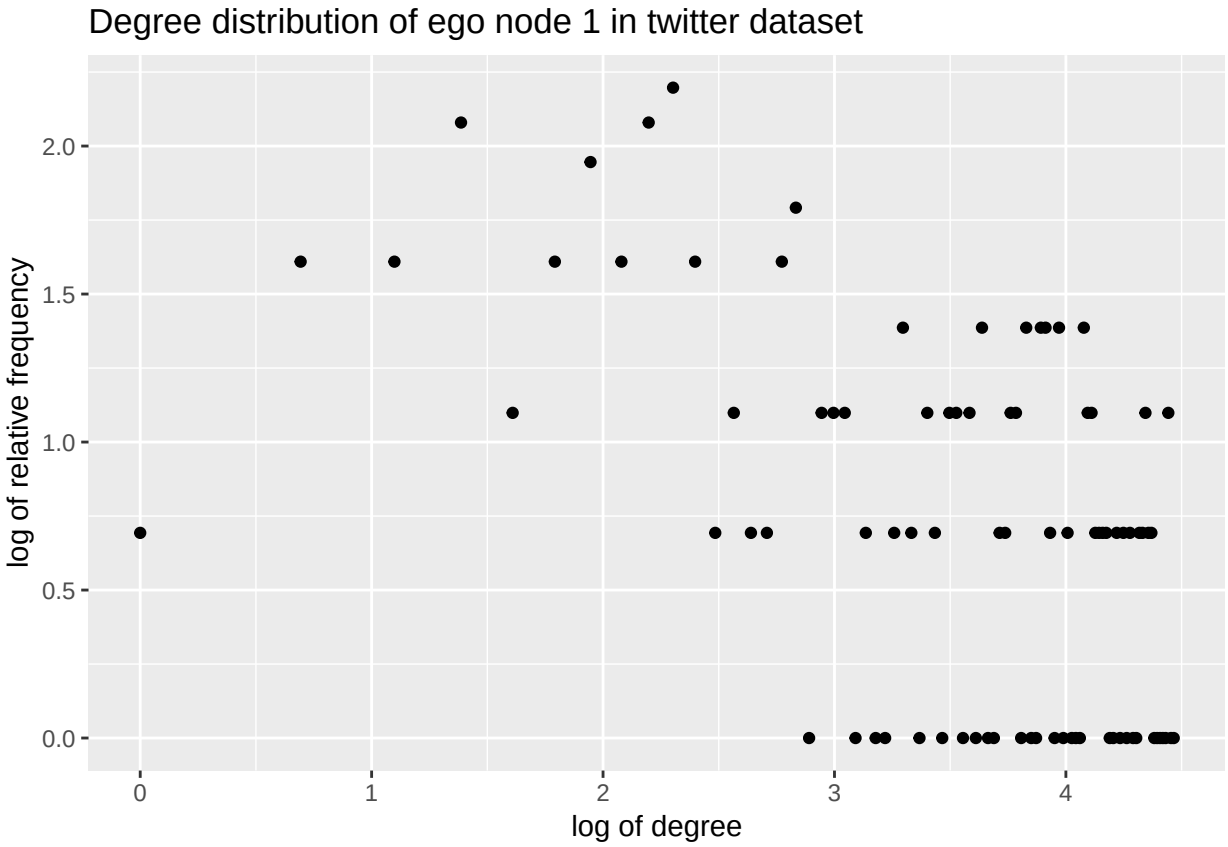
```
## network: Classes for Relational Data
## Version 1.15 created on 2019-04-01.
## copyright (c) 2005, Carter T. Butts, University of California-Irvine
##                     Mark S. Handcock, University of California -- Los Angeles
##                     David R. Hunter, Penn State University
##                     Martina Morris, University of Washington
##                     Skye Bender-deMoll, University of Washington
##  For citation information, type citation("network").
##  Type help("network-package") to get started.
```

```
file1<-read.table(file="./twitter/twitter/100318079.edges")
file1<-file1[order(file1$V2),]
file1$V3<-as.numeric(factor(file1$V2))
file1$V4<-file1$V3[match(file1$V1,file1$V2)]
file1.2<-read.table(file="./twitter/twitter/9855382.edges")
file1.2<-file1.2[order(file1.2$V2),]
file1.2$V3<-as.numeric(factor(file1.2$V2))
file1.2$V4<-file1.2$V3[match(file1.2$V1,file1.2$V2)]
file2<-read.csv(file="./Enron/Enron_cleaned_enroncom.csv")

#Construct the adjacency from the twitter data
mat1<-matrix(0L, 221, 221)
for (i in 1:dim(file1)[1]) {
  mat1[file1$V4[i],file1$V3[i]]=1
}
mat1[221,c(1:220)]=1
mat1[c(1:220),221]=1

mat1.2<-matrix(0L,15,15)
for (i in 1:dim(file1.2)[1]) {
  mat1.2[file1.2$V4[i],file1.2$V3[i]]=1
}
mat1.2[15,c(1:14)]=1
mat1.2[c(1:14),15]=1
# General view of the twitter dataset
# Edge plot of the network
g1<-network(mat1)
plot(g1,usearrows=T,main="Edge plot of an arbitrary ego node 1 in twitter dataset")
```

Edge plot of an arbitrary ego node 1 in twitter dataset

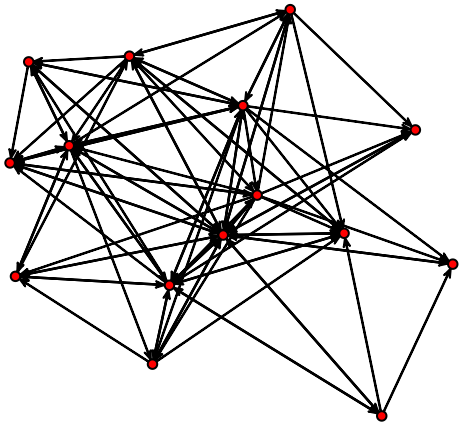


```
#Degree distribution of twitter data
tmp_plot<-table(file1$V1)
tmp_plot<-as.data.frame(tmp_plot)
tmp_plot2<-as.data.frame(table(tmp_plot$Freq))
tmp_plot2$logdf<-log(as.numeric(tmp_plot2$Var1))
tmp_plot2$logFreq<-log(tmp_plot2$Freq)
ggplot()+
  geom_point(data=tmp_plot2,aes(x=logdf,y=logFreq))+
  xlab("log of degree")+
  ylab("log of relative frequency")+
  ggtitle("Degree distribution of ego node 1 in twitter dataset")
```

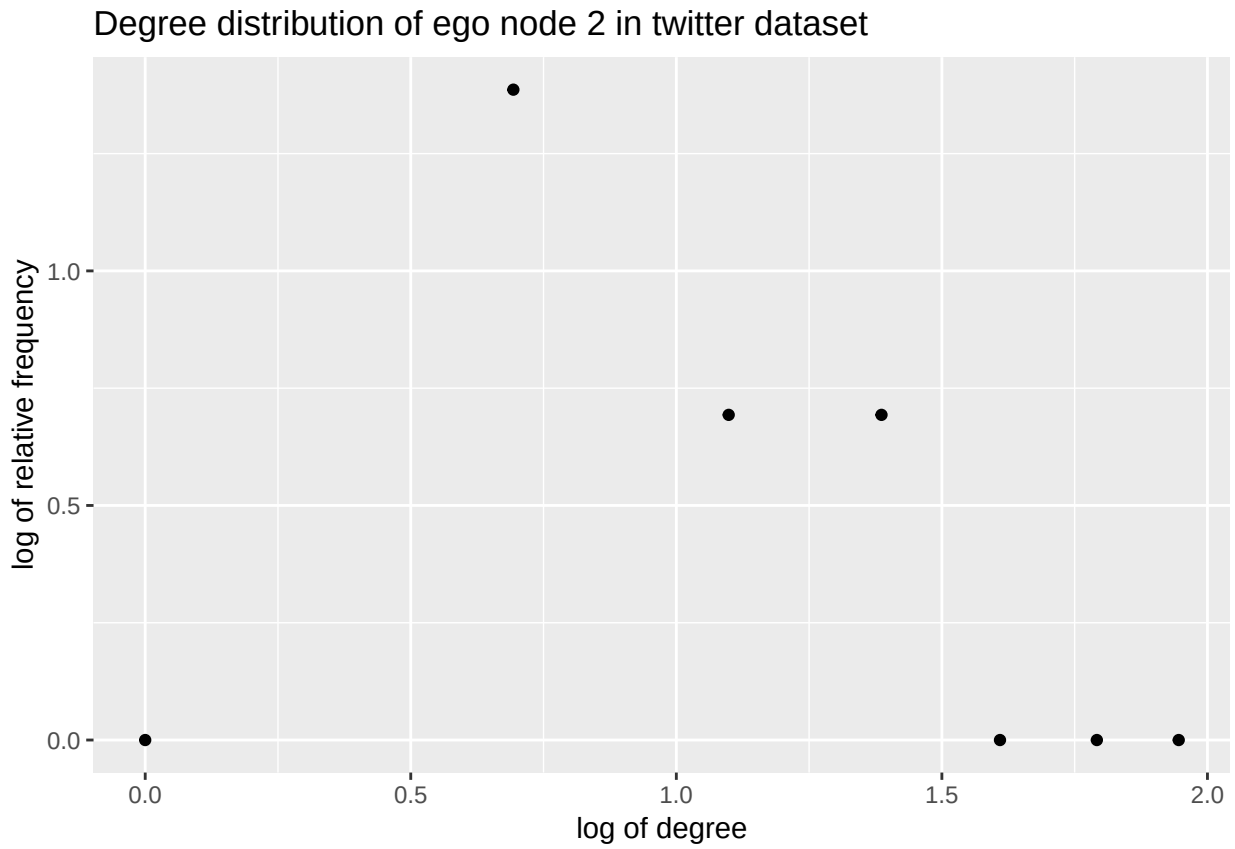



```
#Edge plot of another node
g2<-network(mat1.2)
plot(g2,usearrows=T,main="Edge plot of an arbitrary ego node 2 in twitter dataset")
```

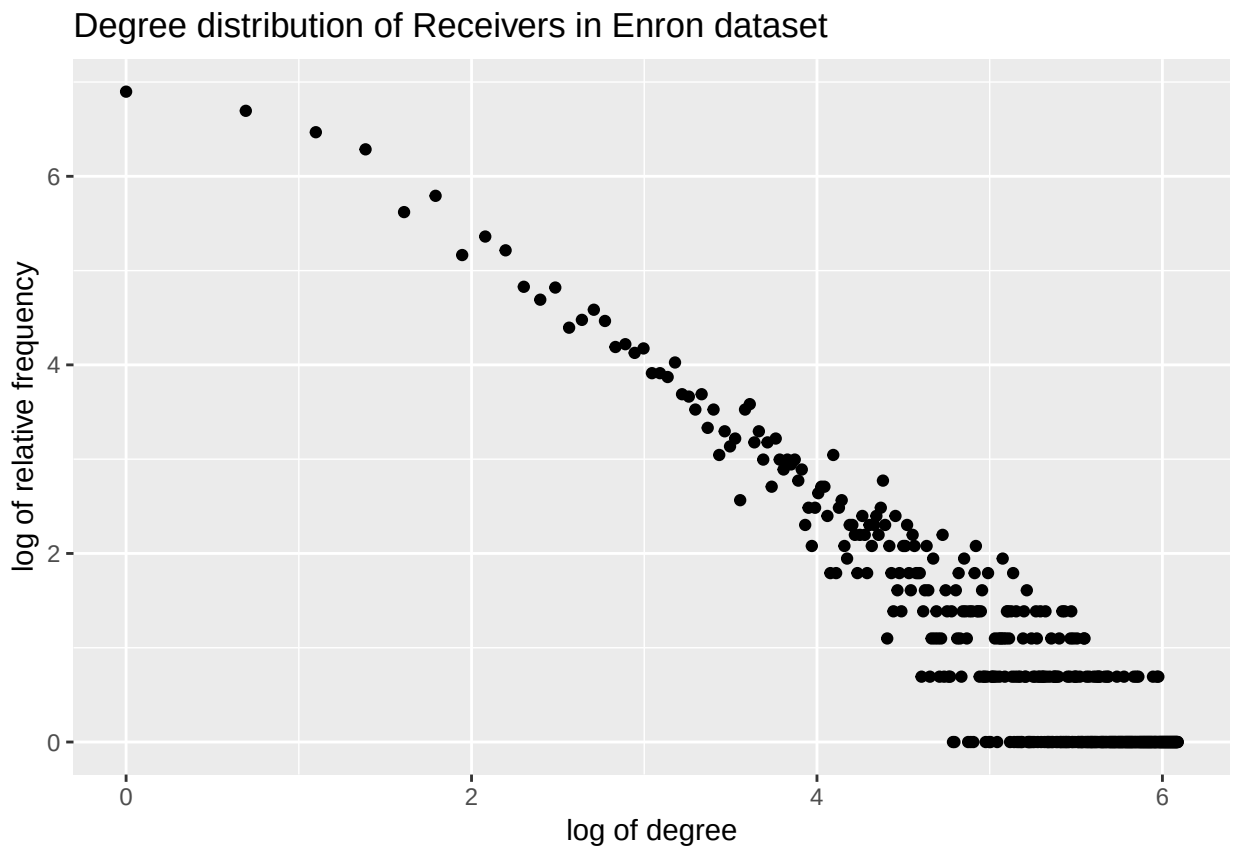
Edge plot of an arbitrary ego node 2 in twitter dataset



```
#Degree distribution of twitter data
tmp_plot<-table(file1.2$V1)
tmp_plot<-as.data.frame(tmp_plot)
tmp_plot2<-as.data.frame(table(tmp_plot$Freq))
tmp_plot2$logdf<-log(as.numeric(tmp_plot2$Var1))
tmp_plot2$logFreq<-log(tmp_plot2$Freq)
ggplot()+
  geom_point(data=tmp_plot2,aes(x=logdf,y=logFreq))+
  xlab("log of degree")+
  ylab("log of relative frequency")+
  ggtitle("Degree distribution of ego node 2 in twitter dataset")
```

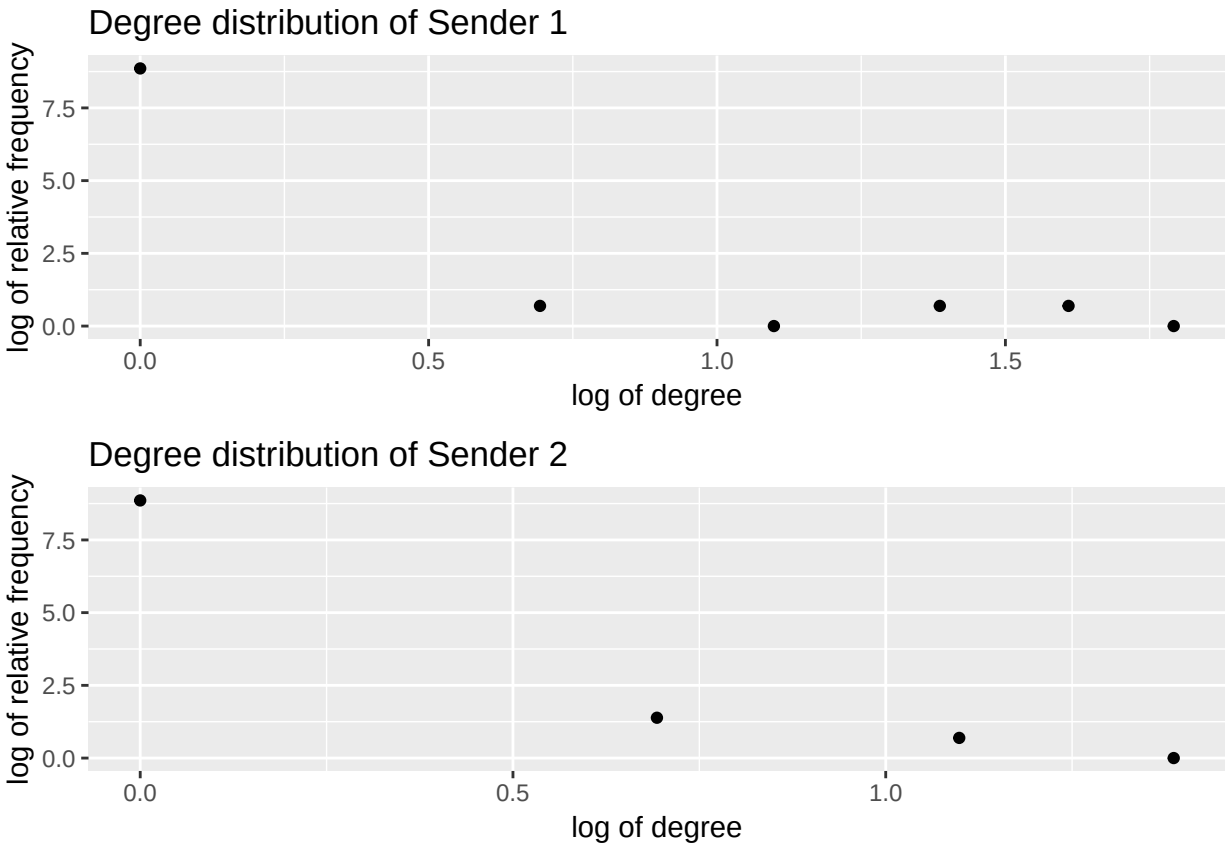



```
# General view of df of Enron's dataset
tmp_plot<-table(file2$To)
tmp_plot<-as.data.frame(tmp_plot)
tmp_plot2<-as.data.frame(table(tmp_plot$Freq))
tmp_plot2$logdf<-log(as.numeric(tmp_plot2$Var1))
tmp_plot2$logFreq<-log(tmp_plot2$Freq)

# Receiver's degree distribution
ggplot()+
  geom_point(data=tmp_plot2,aes(x=logdf,y=logFreq))+
  xlab("log of degree")+
  ylab("log of relative frequency")+
  ggtitle("Degree distribution of Receivers in Enron dataset")
```




```
#Sender's degree distributions
tmp_<-as.data.frame(table(file2$From))
index<-sample(seq_len(nrow(tmp_)), 2, prob=tmp_$Freq)
#index<-sample(nrow(data.frame(table(tmp$From))),1)
tmp<-file2[which(file2$From==tmp_[index[1],1]),]
tmp_plot<-table(tmp$To)
tmp_plot<-as.data.frame(tmp_plot)
tmp_plot2<-as.data.frame(table(tmp_plot$Freq))
tmp_plot2$logdf<-log(as.numeric(tmp_plot2$Var1))
tmp_plot2$logFreq<-log(tmp_plot2$Freq)
p1<-ggplot()+
  geom_point(data=tmp_plot2,aes(x=logdf,y=logFreq))+
  xlab("log of degree")+
  ylab("log of relative frequency")+
  ggtitle("Degree distribution of Sender 1")

tmp<-file2[which(file2$From==tmp_[index[2],1]),]
tmp_plot<-table(tmp$To)
tmp_plot<-as.data.frame(tmp_plot)
tmp_plot2<-as.data.frame(table(tmp_plot$Freq))
tmp_plot2$logdf<-log(as.numeric(tmp_plot2$Var1))
tmp_plot2$logFreq<-log(tmp_plot2$Freq)
p2<-ggplot()+
  geom_point(data=tmp_plot2,aes(x=logdf,y=logFreq))+
  xlab("log of degree")+
  ylab("log of relative frequency")+
```

```
  ggtitle("Degree distribution of Sender 2")
grid.arrange(p1,p2)
```



#Convert the dataset into a network structure

```
#Construct the adjacency matrix from the Enron data
#Suppose we only sample one sender and construct the directed network centering around this center
file2$V3<-as.numeric(factor(file2$From))
file2$V4<-file2$V3[match(file2$To,file2$From)]
sub_file2<-file2[which(file2$V3==3903),]
sub_file2<-sub_file2[complete.cases(sub_file2),]
sub_file2_add<-file2[which(file2$From%in%sub_file2$To),]
sub_file2_add<-sub_file2_add[which(sub_file2_add$To%in%sub_file2$To|sub_file2_add$To%in%sub_file2$From)
sub_file2<-rbind(sub_file2,sub_file2_add)
sub_file2<-sub_file2[!(duplicated(sub_file2[c("V3","V4")])), ]
sub_file2<-sub_file2[which(sub_file2$V3!=sub_file2$V4),]
sub_file2$V5<-as.numeric(factor(sub_file2$From))
sub_file2$V6<-sub_file2$V5[match(sub_file2$To,sub_file2$From)]
sub_file2<-sub_file2[complete.cases(sub_file2),]
sub_file2<-sub_file2[which(sub_file2$V5!=sub_file2$V6),]

mat2<-matrix(0L,ncol=length(unique(sub_file2$V5)),length(unique(sub_file2$V5)))
for (i in 1:dim(sub_file2)[1]) {
  mat2[sub_file2$V5[i],sub_file2$V6[i]]=1
}
```

#p1 Model

Denote $Pr(Y_{i,j} = y_{i,j}, Y_{j,i} = y_{j,i}) = p(y_{i,j}, y_{j,i})$. Then

$p(0,0) = c_{i,j}$

$p(1,0) = c_{i,j}exp(\mu_{i,j})$

$p(0,1) = c_{i,j}exp(\mu_{j,i})$

$p(1,1) = c_{i,j}exp(\mu_{i,j} + \mu_{j,i} + \gamma)$

where $\mu_{i,j} = \alpha_i + \beta_j + \mu$; $\mu_{j,i} = \alpha_j + \beta_i + \mu$; γ is the reciprocal effect; and $c_{i,j} = \frac{1}{1+exp(\mu_{i,j})+exp(\mu_{j,i})+exp(\mu_{i,j}+\mu_{j,i}+\gamma)}$

#Fit a p1 model using ergm package

```
library("ergm")
```

```
##
## ergm: version 3.10.4, created on 2019-06-10
## Copyright (c) 2019, Mark S. Handcock, University of California -- Los Angeles
##                     David R. Hunter, Penn State University
##                     Carter T. Butts, University of California -- Irvine
##                     Steven M. Goodreau, University of Washington
##                     Pavel N. Krivitsky, University of Wollongong
##                     Martina Morris, University of Washington
##                     with contributions from
##                     Li Wang
##                     Kirk Li, University of Washington
##                     Skye Bender-deMoll, University of Washington
##                     Chad Klumb
## Based on "statnet" project software (statnet.org).
## For license and citation information see statnet.org/attribution
## or type citation("ergm").
```

```
## NOTE: Versions before 3.6.1 had a bug in the implementation of the
## bd() constriant which distorted the sampled distribution somewhat.
## In addition, Sampson's Monks datasets had mislabeled vertices. See
## the NEWS and the documentation for more details.
```

```
## NOTE: Some common term arguments pertaining to vertex attribute
## and level selection have changed in 3.10.0. See terms help for
## more details. Use 'options(ergm.term=list(version="3.9.4"))' to
## use old behavior.
```

```
#Twitter data
#I used a fairly small network dataset here out of computational concern
model1<-ergm(mat1.2~edges+sender + receiver + mutual)
```

```
## Observed statistic(s) sender15 and receiver15 are at their greatest attainable values. Their coeffici
```

```
## Starting maximum pseudolikelihood estimation (MPLE):
```

```
## Evaluating the predictor and response matrix.
```

```
## Maximizing the pseudolikelihood.
```

```
## Finished MPLE.
```

```
## Starting Monte Carlo maximum likelihood estimation (MCMLE):
```

```
## Iteration 1 of at most 20:
```

```
## Optimizing with step length 1.
```

```
## The log-likelihood improved by 0.2319.

## Step length converged once. Increasing MCMC sample size.

## Iteration 2 of at most 20:

## Optimizing with step length 1.

## The log-likelihood improved by 0.02583.

## Step length converged twice. Stopping.

## Finished MCMLE.

## Evaluating log-likelihood at the estimate. Using 20 bridges: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 
## This model was fit using MCMC.  To examine model diagnostics and
## check for degeneracy, use the mcmc.diagnostics() function.
```

```
#Enron data
#Out of the computational concern, I didn't include the mutual effects into the model
model2<-ergm(mat2~edges+sender+receiver,iterations=5)
```

```
## Observed statistic(s) receiver2 are at their smallest attainable values. Their coefficients will be 
## Observed statistic(s) sender2 and receiver3 are at their greatest attainable values. Their coefficie
## Starting maximum pseudolikelihood estimation (MPLE):
## Evaluating the predictor and response matrix.
## Maximizing the pseudolikelihood.
## Finished MPLE.
## Stopping at the initial estimate.
## Evaluating log-likelihood at the estimate.
```

In the twitter dataset, there are 15 nodes; in the Enron dataset, there are 76 nodes. A directed network was construct based on these nodes, and the correponding network model was fitted. (I didn't include the reciprocal effect while fitting the Euron model)

The result is shown below:

```
#alpha1<-model1$coef[2:15]
#beta1<-model1$coef[16:29]
#alpha2<-model2$coef[2:76]
#beta2<-model2$coef[77:151]
#par(mfrow=c(2,2))
#hist(alpha1,breaks = 15,main="alpha in twitter data")
#hist(beta1,breaks = 15, main="beta in twitter data")
#hist(alpha2,breaks = 15,main="alpha in Enron data")
#hist(beta2,breaks = 15, main="beta in Enron data")

#Check the goodness of fit
#twitter data
#gof(model1, GOF= ~ idegree + odegree + triadcensus )

#Enron network dataset
#gof(model2, GOF= ~ idegree + odegree + triadcensus )
```

Based on the goodness of fit test, the model seems to work well for both dataset.

#Fit the Hollywood model

```
file2<-read.csv(file="./Enron/Enron_cleaned_enroncom.csv")
#remove multiple edges
file2_nodup<-file2[!(duplicated(file2[c("From","To")])), ]
```

```r
#6062 unique senders and receivers
#8573 unique nodes
length(unique(file2_nodup$From))
```

```
## [1] 6062
```

```r
#factorize the senders and receivers
fac_index<-union(unique(file2_nodup$From),unique(file2_nodup$To))
fac_index<-data.frame(index1=fac_index,index2=c(1:length(fac_index)))
#matching the enron senders and receivers with the index
file2_nodup<-merge(x=file2_nodup,y=fac_index,by.x="From",by.y="index1")
file2_nodup<-merge(x=file2_nodup,y=fac_index,by.x="To",by.y="index1")
colnames(file2_nodup)<-c("To","From","From_ind","To_ind")
file2_nodup<-file2_nodup[which(file2_nodup$From_ind!=file2_nodup$To_ind),]
file2_nodup<-file2_nodup[order(file2_nodup$From_ind),]
file2_sub<-file2_nodup[which(file2_nodup$From_ind<=40),]
file2_sub<-file2_sub[order(file2_sub$From_ind),]

#The llk function
llk<-function(num_vertex,total_deg,k_deg,x){
  #num_vertex = #of vertex
  #total_deg = total degree, 2*nrow
  #k_deg = a table with column(node, degree)
  alpha=x[1]
  theta=x[2]
  k_deg_new=k_deg[which(k_deg[,2]>1),]
  llk_=num_vertex*log(alpha)+
    lgamma(theta/alpha+num_vertex)-lgamma(theta/alpha)-
    lgamma(theta+total_deg)+lgamma(theta)+
    sum(lgamma(1-alpha+k_deg_new[,2]-1)-lgamma(1-alpha))
  #print(llk_)
  return(llk_)
}

#Optimize the llk function
llkoptim<-function(num_vertex,total_deg,k_deg,fn,max.iter){
  #num_vertex = #of vertex
  #total_deg = total degree, 2*nrow
  #k_deg = a table with column(node, degree)
  #fn: A function that evaluates the llk
  #fn: Input llk
  #max.iter: maximum number of iter
  alpha_init=0.5
  theta_init=1
  init<-c(alpha_init,theta_init)
  result<-optim(init,function(x){0-fn(num_vertex,total_deg,k_deg,x)},method = 'L-BFGS-B',lower=c(1e-5,0

  return(list(
    x=result$par,
    fmin=-result$value,
    iter=result$counts[[1]],
    convergence=result$convergence))
}
```

```r
#Test the result
num_vertex=length(unique(c(file2_sub$From_ind,file2_sub$To_ind)))
total_deg=2*nrow(file2_sub)
k_deg=data.frame(V1=unique(file2_sub$From_ind),V2=c(table(file2_sub$From_ind)))
index=which(!(file2_sub$To_ind%in%unique(file2_sub$From_ind)))
k_deg2=data.frame(V1=unique(file2_sub[index,]$To_ind),V2=c(table(file2_sub[index,]$To_ind)))
k_deg=rbind(k_deg,k_deg2)
llkoptim(num_vertex,total_deg,k_deg,llk,100)
```

```
## $x
## [1]  0.6383366 25.6450676
##
## $fmin
## [1] -10735.74
##
## $iter
## [1] 19
##
## $convergence
## [1] 0
```

```r
#Using function.R as the benchmark
source("./function.R")
edgeset=file2_sub[index,]
deg=k_deg[,2]
wiki.params = mg.hat(c(0.5,1), edge.set=edgeset,deg=deg)
```

```
## Warning in optim(init, mg.log.lik(edge.set, deg), lower = c(0.001, 0),
## upper = c(0.999, : bounds can only be used with method L-BFGS-B (or Brent)
```

```r
library(numDeriv)
wiki.info = hessian(func=mg.log.lik(edge.set=edgeset, deg = deg), x = wiki.params)
std.err = sqrt(diag(solve(wiki.info)))
round(cbind(wiki.params, std.err),3)
```

```
##      wiki.params std.err
## [1,]       0.626   0.023
## [2,]      34.726  10.888
```