# Network Practice Examples 9/16

#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 This data was originally made public, and posted to the web, by the Federal Energy Regulatory Commission during its investigation.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")
#file1$message<-str_extract(file1$message,"From: [a-zA-Z.@]*\nTo: [a-zA-Z.@]*")
#tmp<-strsplit(file1$message,"\n")
#file1<-NA
#tmp<-as.data.frame(matrix(unlist(tmp),ncol=2,byrow=T))
#colnames(tmp)<-c("From","To")
#tmp$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)
```

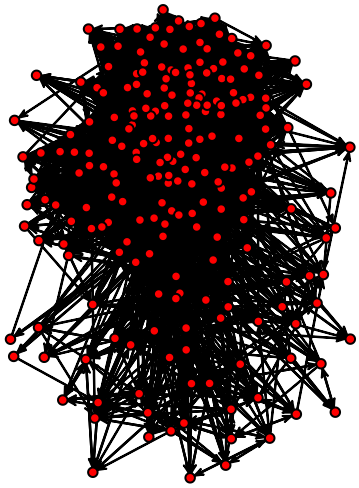
The following is a general overview of the connectivity of the network.

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

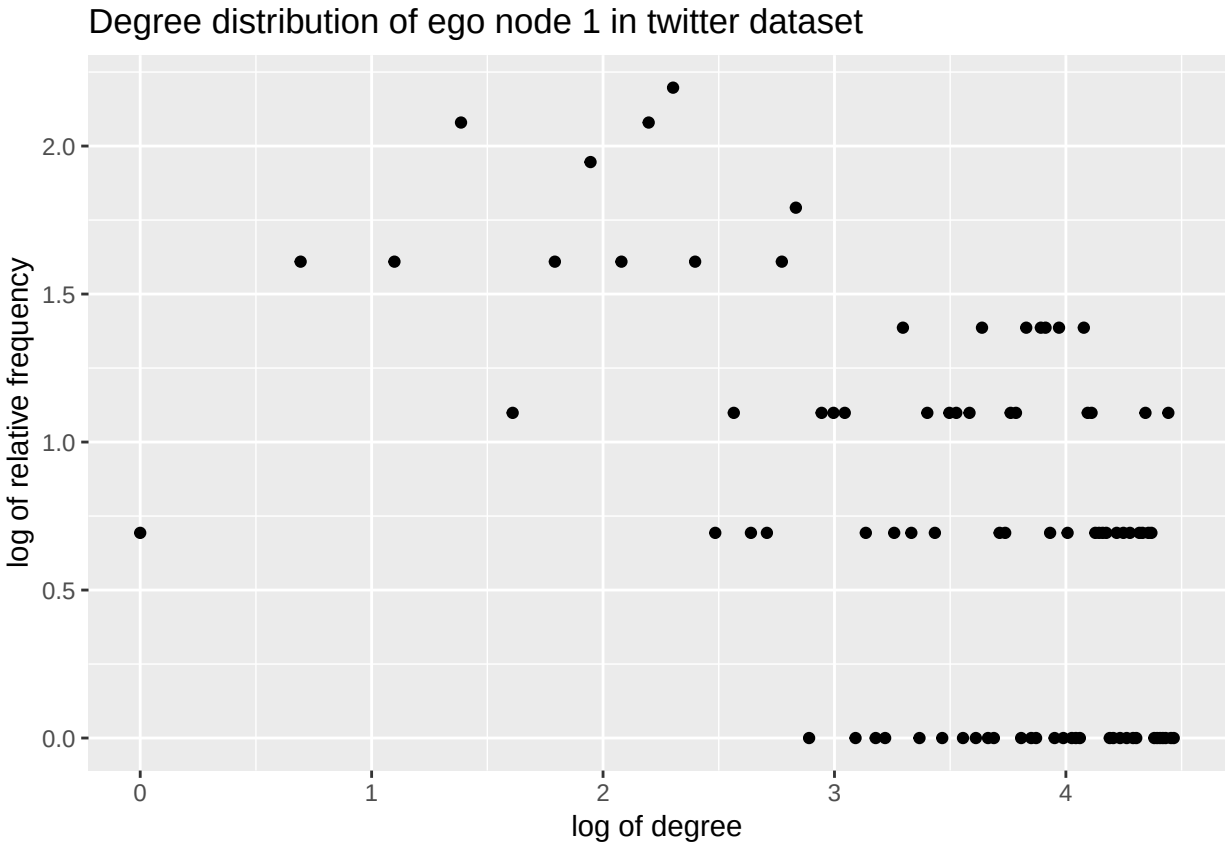
```
## network: Classes for Relational Data
## Version 1.15 created on 2019-04-01.
## copyright (c) 2005, Carter T. Butts, University of California-Irvine
##                     Mark S. Handcock, University of California -- Los Angeles
##                     David R. Hunter, Penn State University
##                     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

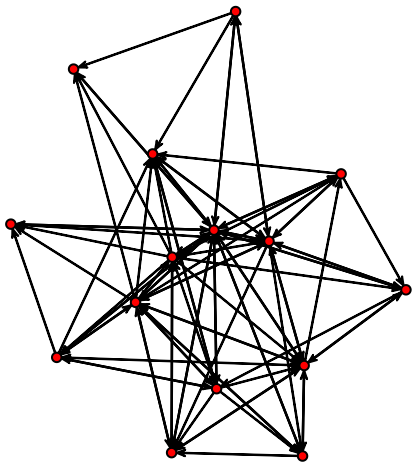


```
#Degree distribution of twitter data
tmp_plot<-table(file1$V1)
tmp_plot<-as.data.frame(tmp_plot)
tmp_plot2<-as.data.frame(table(tmp_plot$Freq))
tmp_plot2$logdf<-log(as.numeric(tmp_plot2$Var1))
tmp_plot2$logFreq<-log(tmp_plot2$Freq)
ggplot()+
  geom_point(data=tmp_plot2,aes(x=logdf,y=logFreq))+
 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

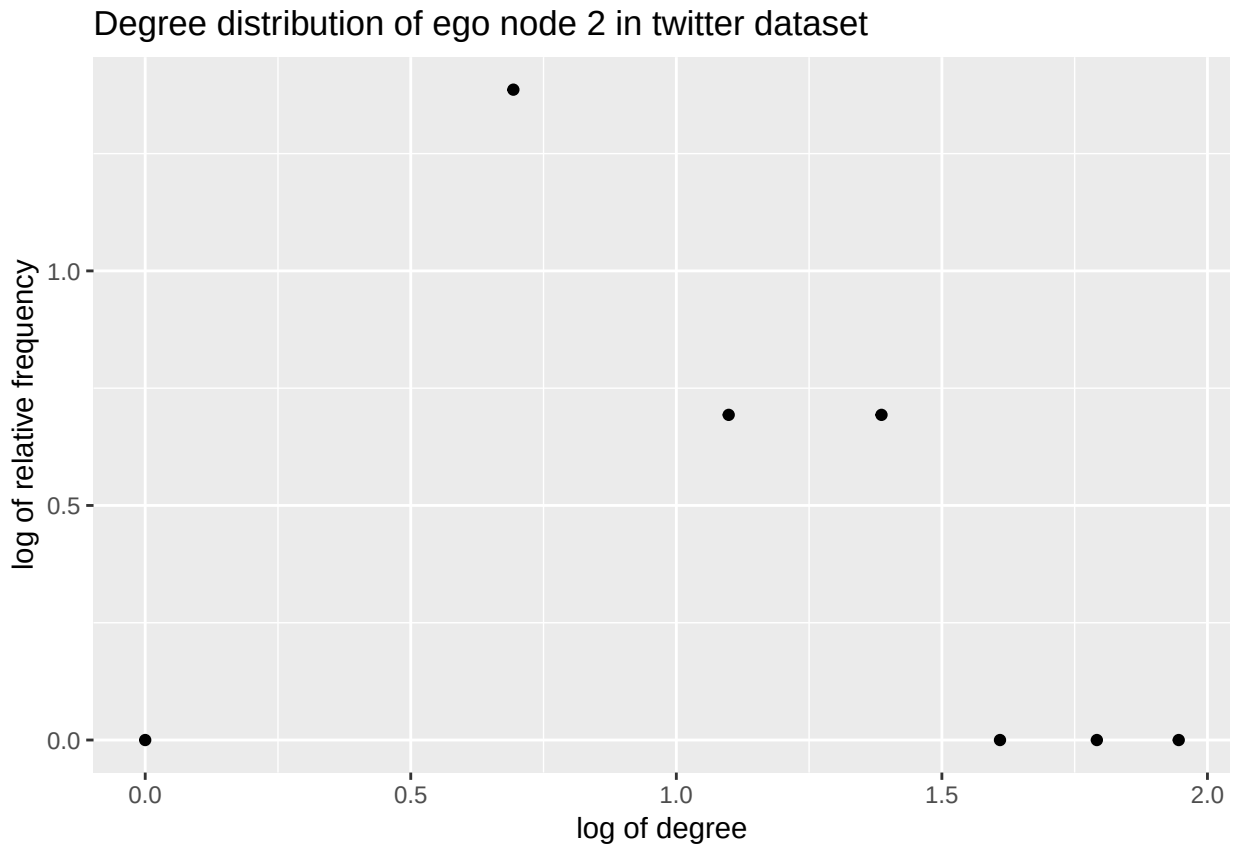



```
#Edge plot of another node
g2<-network(mat1.2)
plot(g2,usearrows=T,main="Edge plot of an arbitrary ego node 2 in twitter dataset")
```

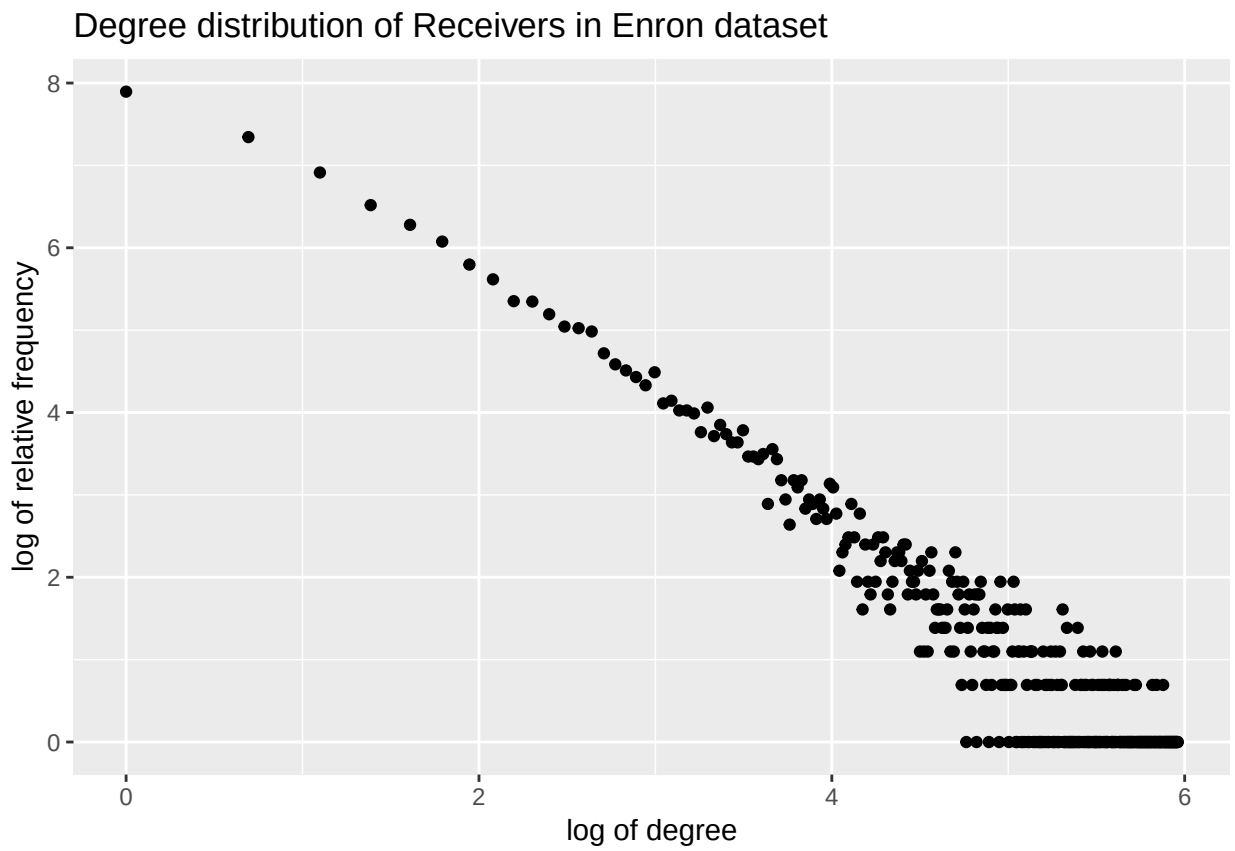
## Edge plot of an arbitrary ego node 2 in twitter dataset



```
#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")
```

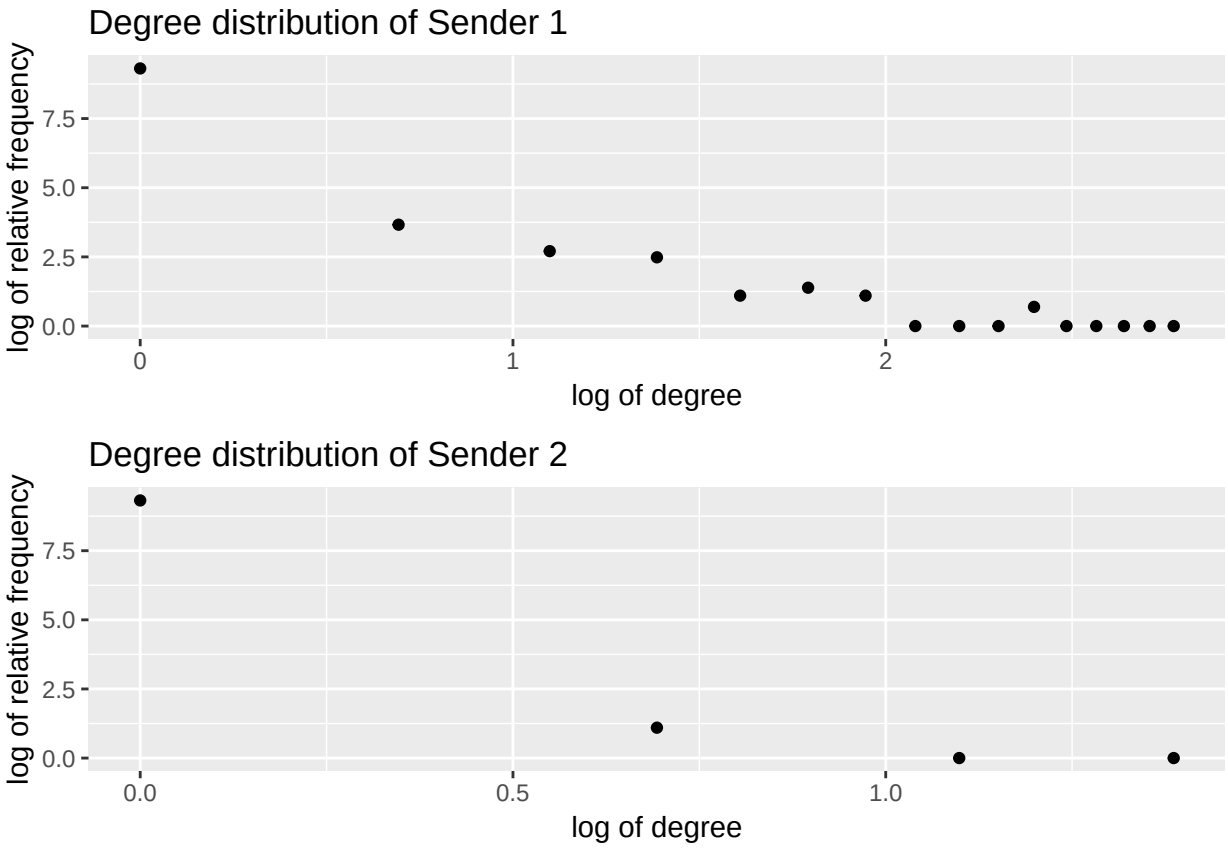



```
# General view of df of Enron's dataset
tmp_plot<-table(file2$To)
tmp_plot<-as.data.frame(tmp_plot)
tmp_plot2<-as.data.frame(table(tmp_plot$Freq))
tmp_plot2$logdf<-log(as.numeric(tmp_plot2$Var1))
tmp_plot2$logFreq<-log(tmp_plot2$Freq)

# 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 $\mu_{i,j} = \alpha_i + \beta_j + \mu$; $\mu_{j,i} = \alpha_j + \beta_i + \mu$; $\gamma$ 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 0.2026.

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

## Iteration 2 of at most 20:

## Optimizing with step length 1.

## The log-likelihood improved by 0.01912.

## 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.
```

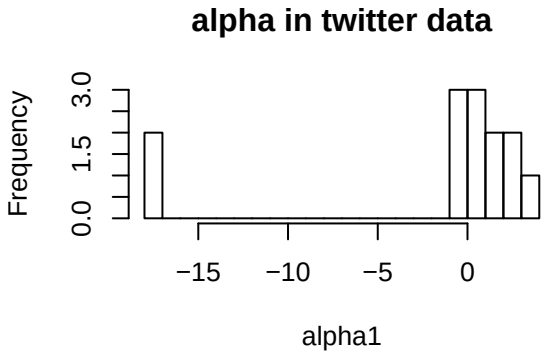
```
#Enron data
#Out of the computational concern, I didn't include the mutual effects into the model
model2<-ergm(mat2~edges+sender+receiver,iterations=5)
```

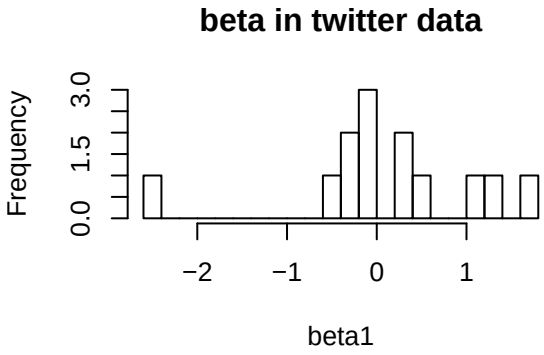
```
## Observed statistic(s) sender56 are at their greatest attainable values. Their coefficients will be fi
## Starting maximum pseudolikelihood estimation (MPLE):
## Evaluating the predictor and response matrix.
## Maximizing the pseudolikelihood.
## Finished MPLE.
## Stopping at the initial estimate.
## Evaluating log-likelihood at the estimate.
```

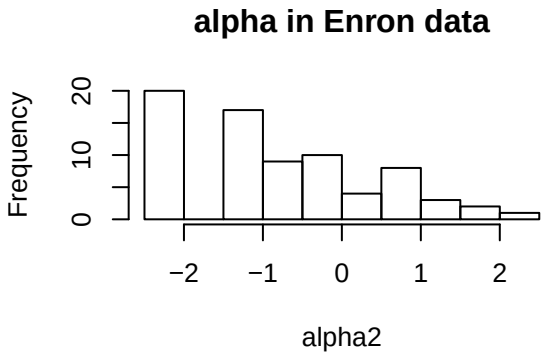
In the twitter dataset, there are 15 nodes; in the Enron dataset, there are 76 nodes. A directed network was construct based on these nodes, and the correponding network model was fitted. (I didn't include the reciprocal effect while fitting the Euron model)

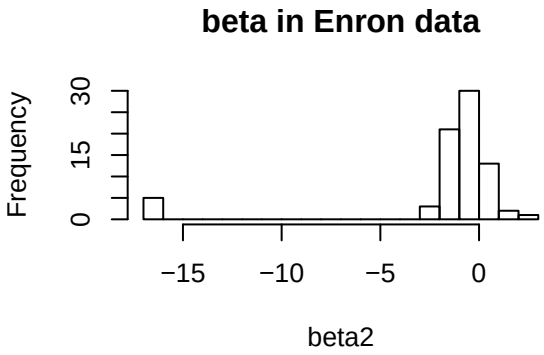
The result is shown below:

```
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 )
```

```
##
## Goodness-of-fit for in-degree
##
##    obs min mean max MC p-value
## 1    0   0 0.42   2       1.00
## 2    1   0 0.70   2       1.00
## 3    0   0 1.18   4       0.72
## 4    1   0 1.98   7       0.76
## 5    7   0 2.48   6       0.00
## 6    0   0 2.77   7       0.10
## 7    3   0 1.88   4       0.64
## 8    2   0 1.28   4       0.72
## 9    0   0 0.78   3       0.82
## 10   0   0 0.48   4       1.00
## 11   0   0 0.05   1       1.00
## 14   1   1 1.00   1       1.00
##
## Goodness-of-fit for out-degree
##
##    obs min mean max MC p-value
## 1    2   2 2.16   4       1.00
## 2    0   0 0.83   3       0.84
```

```
## 3    1   0 1.48   4       1.00
## 4    5   0 2.01   6       0.06
## 5    1   0 1.60   4       0.98
## 6    0   0 1.00   3       0.60
## 7    0   0 0.77   3       0.88
## 8    2   0 0.85   3       0.40
## 9    1   0 0.84   3       1.00
## 10   0   0 0.72   3       0.90
## 11   1   0 0.67   3       1.00
## 12   1   0 0.57   2       0.98
## 13   0   0 0.39   2       1.00
## 14   1   1 1.11   2       1.00
##
## Goodness-of-fit for triad census
##
##      obs min  mean max MC p-value
## 003   54  21 59.31 110       0.78
## 012   83  49 80.74 113       0.82
## 102   36   9 31.15  73       0.62
## 021D  27   9 29.36  60       0.86
## 021U  11   2 12.08  27       1.00
## 021C  14   0 14.09  32       1.00
## 111D   9   2 14.39  27       0.44
## 111U  52  19 41.13  59       0.20
## 030T  15   1 12.32  32       0.60
## 030C   1   0  0.47   4       0.66
## 201   59  42 58.72  86       1.00
## 120D   5   0  4.43  14       1.00
## 120U   6   2 16.03  30       0.02
## 120C  11   0  6.76  23       0.28
## 210   50  28 50.00  76       1.00
## 300   22   8 24.02  45       0.94
##
## Goodness-of-fit for model statistics
##
##            obs min  mean max MC p-value
## edges       92  78 92.07 115          1
## sender2     12   8 11.94  14          1
## sender3      3   1  2.90   6          1
## sender4      9   4  8.76  13          1
## sender5      8   4  7.87  13          1
## sender6      4   1  4.27   8          1
## sender7      8   4  7.92  12          1
## sender8      5   2  5.03   9          1
## sender9      4   1  4.16   8          1
## sender10     1   1  1.00   1          1
## sender11    11   8 11.05  14          1
## sender12     4   1  3.98   8          1
## sender13     1   1  1.00   1          1
## sender14     4   1  3.99   8          1
## sender15    14  14 14.00  14          1
## receiver2    7   3  7.18  10          1
## receiver3    8   5  7.83  11          1
## receiver4    5   2  5.01   8          1
```

```
## receiver5    8   4  7.84  11          1
## receiver6    5   2  5.07   8          1
## receiver7    5   2  4.98   8          1
## receiver8    5   2  5.26   8          1
## receiver9    7   4  7.29  10          1
## receiver10   5   2  4.82   8          1
## receiver11   7   4  7.17  11          1
## receiver12   5   2  4.98   8          1
## receiver13   4   1  3.94   8          1
## receiver14   2   1  1.83   4          1
## receiver15  14  14 14.00  14          1
## mutual      31  22 31.03  42          1
```

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

```
##
## Goodness-of-fit for in-degree
##
##    obs min mean max MC p-value
## 1    5   5 7.82  11       0.08
## 2    3   1 5.89  12       0.30
## 3    6   2 7.56  15       0.66
## 4   15   2 8.48  17       0.08
## 5    7   2 8.43  14       0.80
## 6    9   3 7.32  13       0.62
## 7   12   1 6.27  12       0.04
## 8    1   1 5.10  10       0.08
## 9    5   0 4.47   8       1.00
## 10   5   0 3.46   7       0.48
## 11   3   0 2.52   6       0.96
## 12   0   0 1.78   5       0.34
## 13   0   0 1.24   4       0.58
## 14   1   0 0.97   3       1.00
## 15   0   0 0.59   3       1.00
## 16   1   0 0.41   2       0.66
## 17   0   0 0.40   2       1.00
## 18   1   0 0.53   2       0.88
## 19   0   0 0.29   2       1.00
## 20   0   0 0.33   3       1.00
## 21   0   0 0.11   2       1.00
## 22   0   0 0.18   1       1.00
## 23   0   0 0.13   1       1.00
## 24   1   0 0.18   1       0.36
## 25   0   0 0.20   1       1.00
## 26   0   0 0.12   1       1.00
## 27   0   0 0.11   1       1.00
## 28   0   0 0.03   1       1.00
## 29   0   0 0.01   1       1.00
## 30   0   0 0.09   1       1.00
## 31   0   0 0.08   1       1.00
## 32   0   0 0.14   1       1.00
## 33   0   0 0.17   1       1.00
## 34   1   0 0.12   1       0.24
## 35   0   0 0.09   1       1.00
```

```
## 36   0   0 0.12   1       1.00
## 37   0   0 0.08   1       1.00
## 38   0   0 0.08   1       1.00
## 39   0   0 0.04   1       1.00
## 40   0   0 0.01   1       1.00
## 41   0   0 0.05   1       1.00
##
## Goodness-of-fit for out-degree
##
##    obs min  mean max MC p-value
## 0    0   2  8.96  16       0.00
## 1   20   3 12.27  20       0.02
## 2   11   4  9.72  20       0.64
## 3    6   2  7.61  14       0.72
## 4    9   0  6.19  12       0.28
## 5    5   1  4.34   9       0.82
## 6    3   0  3.44   8       1.00
## 7    3   0  2.82   8       1.00
## 8    1   0  2.06   5       0.72
## 9    1   0  1.96   5       0.70
## 10   2   0  1.99   6       1.00
## 11   1   0  1.59   5       1.00
## 12   2   0  1.55   4       0.90
## 13   3   0  1.43   5       0.34
## 14   2   0  1.32   5       0.72
## 15   1   0  1.25   4       1.00
## 16   0   0  1.04   4       0.64
## 17   2   0  0.67   3       0.26
## 18   0   0  0.63   2       0.98
## 19   0   0  0.52   2       1.00
## 20   0   0  0.44   2       1.00
## 21   0   0  0.36   2       1.00
## 22   0   0  0.28   2       1.00
## 23   0   0  0.22   2       1.00
## 24   1   0  0.34   2       0.60
## 25   1   0  0.26   2       0.48
## 26   0   0  0.21   2       1.00
## 27   0   0  0.22   2       1.00
## 28   0   0  0.22   2       1.00
## 29   0   0  0.18   2       1.00
## 30   0   0  0.13   2       1.00
## 31   0   0  0.15   2       1.00
## 32   1   0  0.15   1       0.30
## 33   0   0  0.07   1       1.00
## 34   0   0  0.13   1       1.00
## 35   0   0  0.05   1       1.00
## 36   0   0  0.09   1       1.00
## 37   0   0  0.02   1       1.00
## 38   0   0  0.06   1       1.00
## 39   0   0  0.02   1       1.00
## 40   0   0  0.03   1       1.00
## 41   0   0  0.01   1       1.00
## 75   1   1  1.00   1       1.00
##
```

```
## Goodness-of-fit for triad census
##
##        obs   min     mean   max MC p-value
## 003  49267 43780 46115.77 48356       0.00
## 012   9896 13288 15186.35 16514       0.00
## 102   5414   728  1459.83  1927       0.00
## 021D  1627  1751  2264.27  2736       0.00
## 021U   216   477   683.19   851       0.00
## 021C   297   954  1323.06  1634       0.00
## 111D   463   310   433.18   626       0.56
## 111U  1832  1224  1570.77  1831       0.00
## 030T   106   186   340.20   439       0.00
## 030C     5    21    49.55    79       0.00
## 201    664   133   311.16   607       0.00
## 120D    39    27    55.51    87       0.16
## 120U   114   130   174.60   216       0.00
## 120C    77   118   157.63   219       0.00
## 210    200    86   150.89   217       0.06
## 300     83     7    24.04    44       0.00
##
## Goodness-of-fit for model statistics
##
##            obs min   mean max MC p-value
## edges      504 465 507.06 563       0.90
## sender2      1   0   1.12   5       1.00
## sender3      6   2   6.42  15       1.00
## sender4      2   0   1.88   6       1.00
## sender5     13   2  13.04  23       1.00
## sender6      2   0   2.04   5       1.00
## sender7      2   0   2.34   8       1.00
## sender8      1   0   0.92   4       1.00
## sender9      1   0   0.98   4       1.00
## sender10     2   0   2.22   6       1.00
## sender11     4   1   3.62  11       0.98
## sender12     9   2   9.07  16       1.00
## sender13     5   1   5.03  10       1.00
## sender14     3   0   2.86   8       1.00
## sender15     4   1   4.12  10       1.00
## sender16     5   1   5.13  10       1.00
## sender17    13   6  12.80  22       0.94
## sender18     2   0   1.88   5       1.00
## sender19     1   0   0.95   5       1.00
## sender20     2   0   1.87   8       1.00
## sender21     5   1   4.99  10       1.00
## sender22     1   0   1.03   4       1.00
## sender23     4   1   4.07   9       1.00
## sender24     4   0   4.13   9       1.00
## sender25     2   0   2.10   7       1.00
## sender26     1   0   1.02   4       1.00
## sender27    17   9  17.34  25       1.00
## sender28     1   0   0.81   4       1.00
## sender29    12   5  11.97  20       1.00
## sender30    11   4  11.59  19       1.00
## sender31    14   6  13.62  22       0.98
```

```
## sender32     3   0   2.77   7       1.00
## sender33    17  10  17.07  27       1.00
## sender34     3   0   2.99   8       1.00
## sender35     3   0   2.99  10       1.00
## sender36     1   0   1.06   4       1.00
## sender37    25  16  25.30  36       1.00
## sender38     1   0   1.04   4       1.00
## sender39     1   0   0.83   4       1.00
## sender40     6   1   6.25  12       1.00
## sender41     1   0   0.96   3       1.00
## sender42     1   0   1.25   5       1.00
## sender43     5   1   4.98  10       1.00
## sender44     1   0   0.98   4       1.00
## sender45    10   4   9.93  18       1.00
## sender46     4   0   4.27   9       1.00
## sender47     1   0   1.11   4       1.00
## sender48    14   4  14.09  21       1.00
## sender49    24  16  23.97  34       1.00
## sender50    32  22  32.12  41       1.00
## sender51     7   2   7.12  13       1.00
## sender52     1   0   1.00   4       1.00
## sender53     1   0   0.84   3       1.00
## sender54     6   1   6.19  12       1.00
## sender55     1   0   1.10   5       1.00
## sender56    75  75  75.00  75       1.00
## sender57    12   4  12.47  21       1.00
## sender58     5   1   4.80  11       1.00
## sender59     7   1   7.06  15       1.00
## sender60     3   0   2.72   7       1.00
## sender61     2   0   1.88   7       1.00
## sender62     4   0   4.13   9       1.00
## sender63     4   0   3.78   7       1.00
## sender64     4   0   3.65   9       1.00
## sender65    13   6  13.32  21       1.00
## sender66     1   0   1.04   4       1.00
## sender67     8   3   8.41  17       1.00
## sender68     1   0   0.96   4       1.00
## sender69     2   0   1.87   5       1.00
## sender70     2   0   2.10   7       1.00
## sender71    10   5  10.49  16       1.00
## sender72     4   0   3.99   8       1.00
## sender73    15   7  15.25  23       1.00
## sender74     3   0   2.88   9       1.00
## sender75     2   0   2.08   9       1.00
## sender76     1   0   1.06   6       1.00
## receiver2    2   1   2.13   6       1.00
## receiver3    7   2   7.12  13       1.00
## receiver4    6   3   6.02  12       1.00
## receiver5   10   4   9.70  17       1.00
## receiver6    4   1   3.75   7       1.00
## receiver7    5   2   5.16  10       1.00
## receiver8    1   1   1.00   1       1.00
## receiver9    4   1   3.82   9       1.00
## receiver10   3   1   2.80   7       1.00
```

```
## receiver11   7   2   6.92  14       1.00
## receiver12   7   3   7.19  14       1.00
## receiver13   6   2   6.00  12       1.00
## receiver14   6   2   5.92  11       1.00
## receiver15   4   1   4.05   8       1.00
## receiver16   6   2   6.02  10       1.00
## receiver17  10   3  10.15  17       1.00
## receiver18   3   1   2.86   7       0.94
## receiver19   4   1   3.95   8       1.00
## receiver20   3   1   2.71   7       1.00
## receiver21   5   1   4.69   9       1.00
## receiver22   4   1   3.98   7       1.00
## receiver23   6   2   6.00  13       1.00
## receiver24   6   1   6.18  10       1.00
## receiver25   4   1   4.12  11       1.00
## receiver26   3   1   2.78   6       1.00
## receiver27   9   4   9.14  18       1.00
## receiver28   1   1   1.00   1       1.00
## receiver29  11   4  11.19  19       1.00
## receiver30  11   5  11.11  18       1.00
## receiver31  11   6  10.75  18       0.94
## receiver32   7   3   6.89  13       1.00
## receiver33  14   6  14.30  20       1.00
## receiver34   7   2   6.99  12       1.00
## receiver35   9   5   9.34  15       1.00
## receiver36   1   1   1.00   1       1.00
## receiver37  34  25  34.52  41       1.00
## receiver38   8   4   8.47  16       1.00
## receiver39   4   1   4.16   9       1.00
## receiver40   5   1   4.90  10       1.00
## receiver41  10   5  10.55  18       1.00
## receiver42   3   1   3.13   8       1.00
## receiver43   4   1   3.82   8       1.00
## receiver44   4   1   3.71   8       1.00
## receiver45   7   2   7.28  13       1.00
## receiver46   4   1   3.75   8       1.00
## receiver47   1   1   1.00   1       1.00
## receiver48  10   4  10.04  16       1.00
## receiver49   9   4   9.35  17       1.00
## receiver50  18   9  18.55  29       0.98
## receiver51   6   1   5.78  11       1.00
## receiver52   5   1   4.80  12       1.00
## receiver53   7   3   7.33  13       1.00
## receiver54   7   2   7.27  14       1.00
## receiver55   6   2   6.04  15       1.00
## receiver56  24  15  24.21  34       1.00
## receiver57   4   1   3.78   8       1.00
## receiver58   6   2   6.47  11       1.00
## receiver59   7   2   7.26  13       1.00
## receiver60   5   1   4.83   9       1.00
## receiver61   3   1   2.91   8       1.00
## receiver62   4   1   3.96  10       1.00
## receiver63   5   1   5.12  10       1.00
## receiver64  10   5  10.05  18       1.00
```

```
## receiver65   9   4   9.19  16       1.00
## receiver66   4   1   3.98   8       1.00
## receiver67   7   2   6.95  13       1.00
## receiver68   4   1   4.01   8       1.00
## receiver69   2   1   1.89   5       1.00
## receiver70   1   1   1.00   1       1.00
## receiver71  16   9  16.19  26       1.00
## receiver72   7   3   6.99  16       1.00
## receiver73   7   3   7.14  16       1.00
## receiver74   4   1   3.88   7       1.00
## receiver75   5   1   5.05  12       1.00
## receiver76   2   1   1.89   6       1.00
```

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