Network_10_2

Fit the Hollywood model

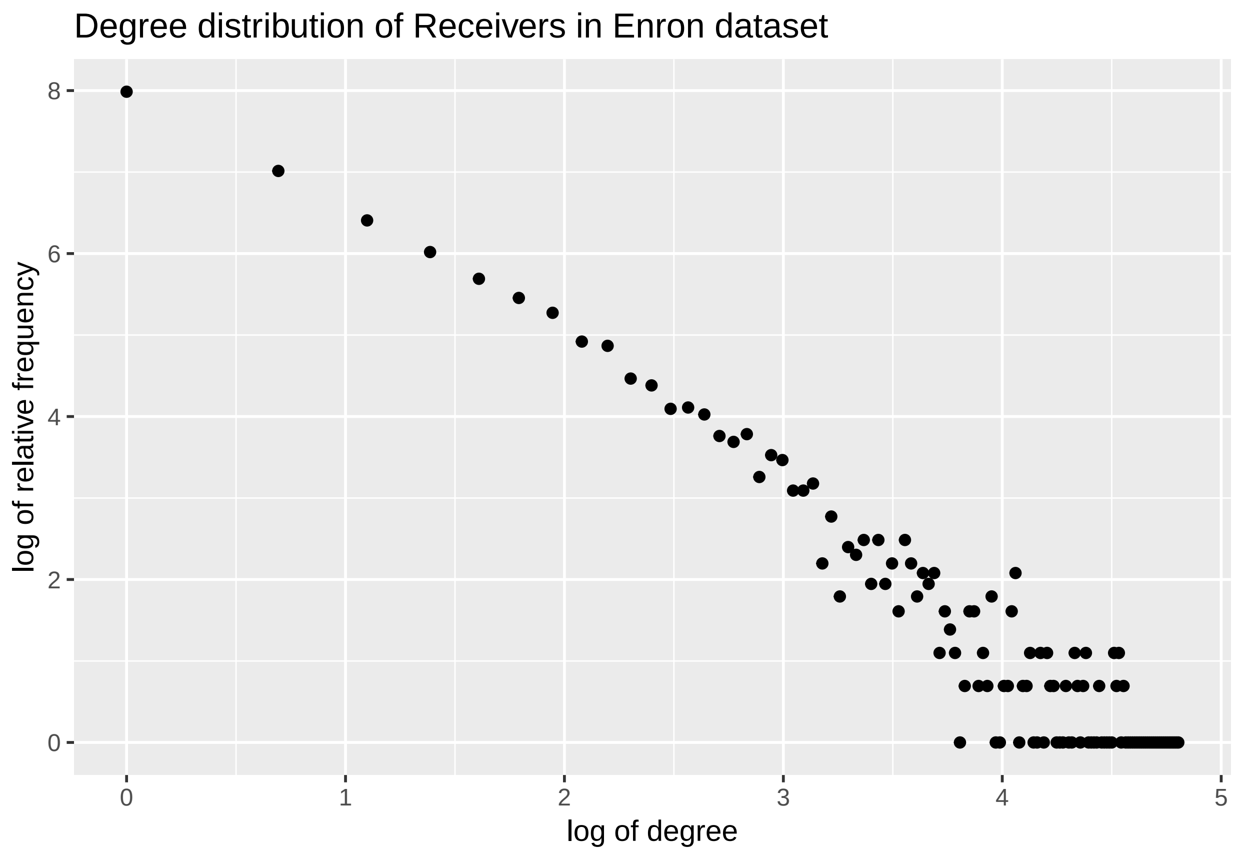
```r
file2<-read.csv(file="./Enron/Enron_cleaned_enroncom.csv")
#remove multiple edges
file2_nodup<-file2[!(duplicated(file2[c("From","To")])), ]
#6062 unique senders and receivers
#8573 unique nodes
length(unique(file2_nodup$From))
```

```
## [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),]

# Power law plot of dfs
tmp_plot<-table(file2_nodup$To_ind)
tmp_plot<-as.data.frame(tmp_plot)
tmp_plot2<-as.data.frame(table(tmp_plot$Freq))
tmp_plot2$logdf<-log(as.numeric(tmp_plot2$Var1))
tmp_plot2$logFreq<-log(tmp_plot2$Freq)

# Receiver's degree distribution
library("ggplot2")
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")
```



```r
#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))
}

#Test the result
num_vertex=length(unique(c(file2_nodup$From_ind,file2_nodup$To_ind)))
total_deg=2*nrow(file2_nodup)
k_deg=data.frame(V1=unique(file2_nodup$From_ind),V2=c(table(file2_nodup$From_ind)))
index=which(!(file2_nodup$To_ind%in%unique(file2_nodup$From_ind)))
k_deg2=data.frame(V1=unique(file2_nodup[index,]$To_ind),V2=c(table(file2_nodup[index,]$To_ind)))
k_deg=rbind(k_deg,k_deg2)
llkoptim(num_vertex,total_deg,k_deg,llk,100)
```

```
## $x
## [1]   0.5183879 196.4304672
##
## $fmin
## [1] -746449.1
##
## $iter
## [1] 18
##
## $convergence
## [1] 0
```

```r
#Using function.R as the benchmark
source("./function.R")
edgeset=file2_nodup[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,]       1e-03   0.015
## [2,]       1e+04 202.345
```