EX3_Sol

Part 1

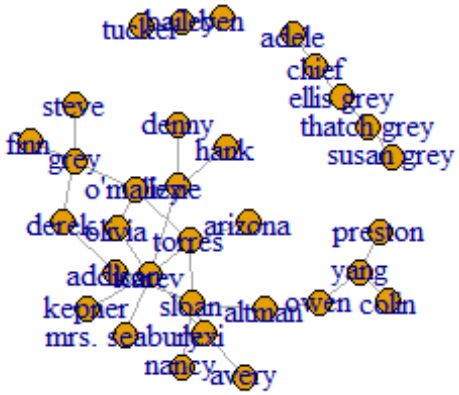
Q1 a

setup home folder

```
folder = 'C:/Users/oleg/Desktop/Study/year5/semesterA/dataAnalys/network
analys'
setwd(folder)

#Or for all chuncks in this Rmarkdown:
knitr::opts_knit$set(root.dir = folder)
set.seed(123)
```

loading libraries

```
library(igraph)

## Warning: package 'igraph' was built under R version 3.4.3

##
## Attaching package: 'igraph'

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union

ga.data <- read.csv('ga_edgelist.csv', header=TRUE, stringsAsFactors=FALSE)
ga.vrtx <- read.csv('ga_actors.csv', header=TRUE, stringsAsFactors=FALSE)
g <- graph.data.frame(ga.data, vertices=ga.vrtx, directed=FALSE)
plot(g)
```



the actor with the maximal betweenness centrality

```
g$bet=betweenness(g, v = V(g), directed = FALSE, weights = NULL,
  nobigint = TRUE, normalized = FALSE)

which.max(g$bet)

## sloan
##    26
```

the actor with the maximal closeness centrality

```
g$clos=closeness(g, vids = V(g), mode = c("all"),
  weights = NULL, normalized = FALSE)
which.max(g$clos)

## torres
##     30
```

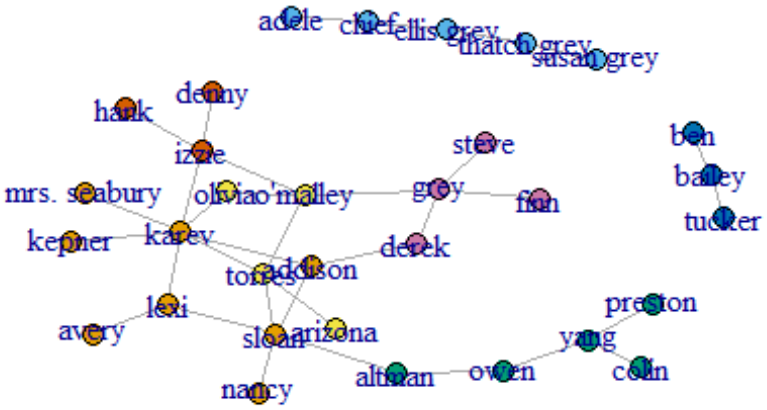
the actor with the maximal eigenvector centrality

```
g$eig=eigen_centrality(g)
max=which.max(g$eig$vector)
max

## karev
##    17
```

Q1 b

community detection by cluster edge betweenness

```
gc1 <-  edge.betweenness.community(g)
memb1 <- membership(gc1)
plot(g, vertex.size=7, vertex.color=memb1, asp=FALSE)
```



the number of communities for cluster edge betweenness

```
length(gc1)
```

```
## [1] 7
```

the size of each community by community index for cluster edge betweenness

```
print (sizes(gc1))
```

```
## Community sizes
## 1 2 3 4 5 6 7
## 8 5 5 4 3 3 4
```

the modularity value of gc1

```
gc1$modularity
```

```
##  [1] -0.04584775 -0.01773356  0.01081315  0.03849481  0.06617647
##  [6]  0.09472318  0.12326990  0.14965398  0.17560554  0.20285467
```

```
## [11]  0.23096886  0.25865052  0.28633218  0.31358131  0.34083045
## [16]  0.36894464  0.39576125  0.41479239  0.44247405  0.46712803
## [21]  0.49134948  0.50778547  0.52681661  0.54974048  0.57050173
## [26]  0.57742215  0.56098616  0.53416955  0.45804498  0.30449827
```

maximum modularity

```
max(gc1$modularity)
```

```
## [1] 0.5774221
```

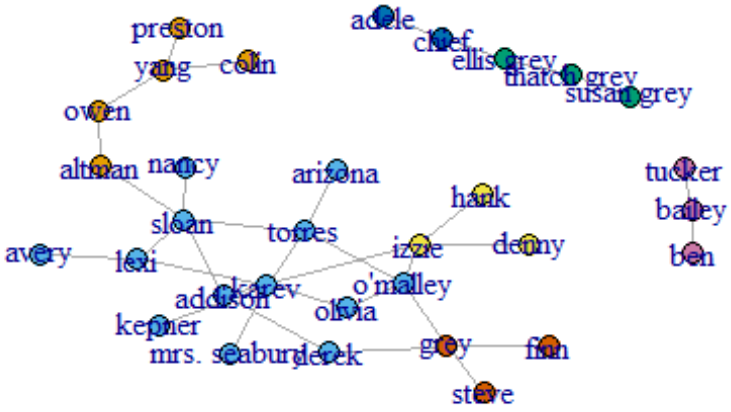
the name of the actor with maximum modularity

```
gc1$name[which.max(gc1$modularity)]
```

```
## [1] "sloan"
```

community detection using cluster walktrap

```
gc2 <- walktrap.community(g)
memb2 <- membership(gc2)
plot(g, vertex.size=7, #vertex.label=NA,
     vertex.color=memb2, asp=FALSE)
```



the number of communities for cluster walktrap

```
length(gc2)
```

```
## [1] 7
```

the size of each community by community index for cluster walktrap

```
sizes(gc2)
```

```
## Community sizes
##  1  2  3  4  5  6  7
##  5 13  3  3  2  3  3
```

the modularity value of gc2

```
gc2$modularity
```

```
##  [1]  0.00000000 -0.01730104  0.01081314  0.03676470  0.06487890
##  [6]  0.09256055  0.12024222  0.14749135  0.17387544  0.19982699
## [11]  0.22837371  0.25692043  0.28460205  0.31185120  0.33910033
## [16]  0.36678201  0.39489621  0.42171276  0.44939446  0.45544982
## [21]  0.48226649  0.47923881  0.49567476  0.48875433  0.49394464
## [26]  0.51470590  0.48269898  0.50562286  0.45804498  0.30449831
## [31]  0.00000000  0.00000000
```

maximum modularity

```
max(gc2$modularity)
```

```
## [1] 0.5147059
```

the name of the actor with maximum modularity

```
gc2$name[which.max(gc2$modularity)]
```

```
## [1] "sloan"
```

Part 2

Q2 a

In part 2 of the HW we will take a list of music genres and scrap twittes related to those genres. after scarping we will make a graph of asociated genres and network analyze this graph.

Q2 b

the nodes in the graph will be the music genres and the edges will be the asociations from twitts we scraped.

Q2 c

initializing environment for twitter scraping

```
library(twitteR)
```

```
## Warning: package 'twitteR' was built under R version 3.4.3

library(httr)

## Warning: package 'httr' was built under R version 3.4.3

library(jsonlite)

## Warning: package 'jsonlite' was built under R version 3.4.3

library(wordcloud)

## Warning: package 'wordcloud' was built under R version 3.4.3

## Loading required package: RColorBrewer

library(jsonlite)
library(tm)

## Warning: package 'tm' was built under R version 3.4.3

## Loading required package: NLP

##
## Attaching package: 'NLP'

## The following object is masked from 'package:httr':
##
##     content

library(devtools)

## Warning: package 'devtools' was built under R version 3.4.3

#install.packages("base64enc")
library(base64enc)
library(XML)
library(RCurl)

## Warning: package 'RCurl' was built under R version 3.4.3

## Loading required package: bitops

library(igraph)
source("twitterOAuth.R")


myapp=oauth_app("twitter",key=consumer_key,secret=consumer_secret)
sig1=sign_oauth1.0(myapp,token=access_token,token_secret=access_secret)

sig <- setup_twitter_oauth(consumer_key, consumer_secret, access_token,
access_secret)

## [1] "Using direct authentication"
```

```
musicGen= read.csv('musicGenres.csv', header=TRUE, stringsAsFactors=FALSE)
musicGen$Genres
```

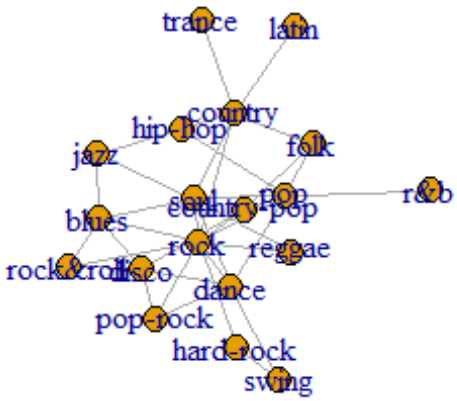
```
##  [1] "pop"         "rock"        "hard-rock"   "r&b"         "country"
##  [6] "rock&roll"   "soul"        "country-pop" "latin"       "disco"
## [11] "jazz"        "pop-rock"    "dance"       "hip-hop"     "trance"
## [16] "blues"       "folk"        "reggae"      "swing"
```

scrap twitter for related words and create the asociation graph

```
newEdges=c()
for (genre in musicGen$Genres){
  searchRes <- searchTwitter(genre, n=200)
  searchDF <- twListToDF(searchRes)
  for (twitts in searchDF$text){
    tokensForGenre <- strsplit(twitts," ")[[1]]
    tokensForGenre<-unique(tokensForGenre)
    for (token in tokensForGenre){
      if (token !=genre && token %in% musicGen$Genres){
        newEdges=c(newEdges,genre,token)
      }
    }
  }
}
```

```
## Warning in doRppAPICall("search/tweets", n, params = params,
## retryOnRateLimit = retryOnRateLimit, : 200 tweets were requested but the
## API can only return 75
```

```
g<- graph(newEdges)
A<-get.adjacency(g)
A[A>1]<-1
newg<- graph.adjacency(A)
g<- as.undirected(newg)
plot(g)
```



Q2 d

the genre with the maximal betweenness centrality

```
g$bet=betweenness(g, v = V(g), directed = FALSE, weights = NULL,
  nobigint = TRUE, normalized = FALSE)

which.max(g$bet)

## rock
##    2
```

the genre with the maximal closeness centrality

```
g$clos=closeness(g, vids = V(g), mode = c("all"),
  weights = NULL, normalized = FALSE)
which.max(g$clos)

## rock
##    2
```

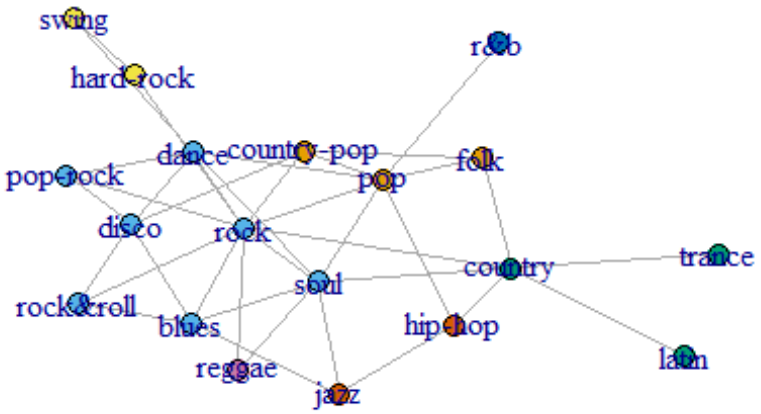
the genre with the maximal eigenvector centrality

```
g$eig=eigen_centrality(g)
max=which.max(g$eig$vector)
max
```

```
## rock
##    2
```

community detection by cluster edge betweenness

```
gc3 <-  edge.betweenness.community(g)
memb3 <- membership(gc3)
plot(g, vertex.size=7,vertex.color=memb3, asp=FALSE)
```



the number of communities for cluster edge betweenness

```
length(gc3)
```

```
## [1] 7
```

the size of each community by community index for cluster edge betweenness

```
print (sizes(gc3))
```

```
## Community sizes
## 1 2 3 4 5 6 7
## 3 7 3 2 1 2 1
```

the modularity value of gc3

```
gc3$modularity
```

```
##  [1] -0.072315559 -0.047479912 -0.027027027 -0.003287071  0.019357195
##  [6]  0.055514974  0.077063550  0.116508400  0.142074507  0.163257852
```

```
## [11]  0.168736304  0.193206720  0.204894083  0.140613587  0.155953251
## [16]  0.116508400  0.081446311  0.084733382  0.000000000
```

maximum modularity

```
max(gc3$modularity)
```

```
## [1] 0.2048941
```

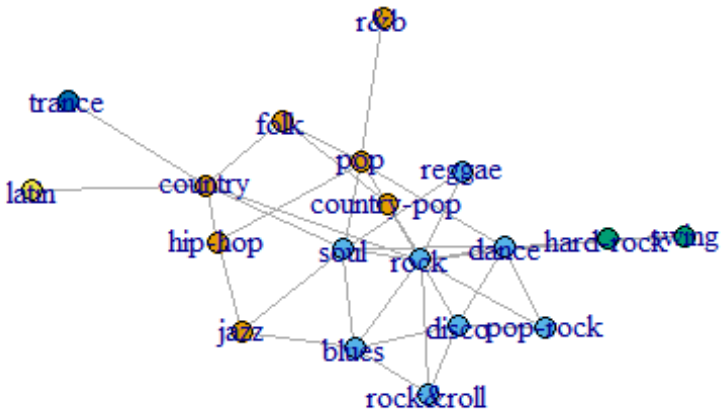
the name of the genre with maximum modularity

```
gc3$name[which.max(gc3$modularity)]
```

```
## [1] "folk"
```

community detection using cluster walktrap

```
gc4 <- walktrap.community(g)
memb4 <- membership(gc4)
plot(g, vertex.size=7,vertex.color=memb4, asp=FALSE)
```



the number of communities for cluster walktrap

```
length(gc4)
```

```
## [1] 5
```

the size of each community by community index for cluster walktrap

```
sizes(gc4)
```

```
## Community sizes
## 1 2 3 4 5
## 7 8 2 1 1
```

the modularity value of gc4

```
gc4$modularity
```

```
##  [1]  0.000000000 -0.046749465 -0.025200881 -0.004747988  0.017165817
##  [6]  0.033966400  0.055149741  0.094594598  0.118334554  0.160336018
## [11]  0.171658128  0.181154117  0.195032850  0.187728256  0.205259293
## [16]  0.044192817  0.046384230  0.048210360  0.000000000
```

maximum modularity

```
max(gc4$modularity)
```

```
## [1] 0.2052593
```

the name of the genre with maximum modularity

```
gc4$name[which.max(gc4$modularity)]
```

```
## [1] "jazz"
```