Social Network Analysis Home Assignment 2

{Mirzaaghayev Ramil}

due date - 22.05.2017 23:59

Table of Contents

Graph models. Centrality metrics

Please send your reports to hse.ntwks@gmail.com with the subject of of the following structure: *[MAGOLEGO SNA 2017] {Mirzaaghayev} {Ramil} HA{2}*

Late submission policy: -1 point per day

Use this file as a template for your report.
Support your computations with figures and comments. Send ONLY .Rmd versions of your report with datasets for your social network. Anonymize your data if necessary.

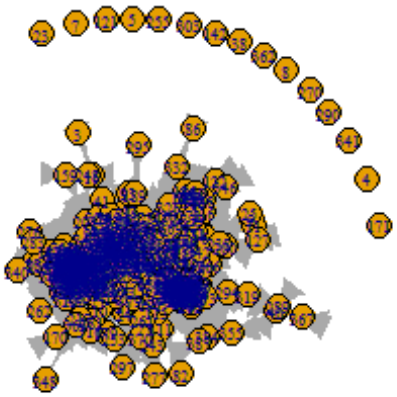
```
##
## Attaching package: 'igraph'

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

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

## Loading required package: XML

## Loading required package: Rook

## Registered S3 method overwritten by 'R.oo':
##   method        from
##   throw.default R.methodsS3

## R.matlab v3.6.2 (2018-09-26) successfully loaded. See ?R.matlab for help.
```

```
##
## Attaching package: 'R.matlab'

## The following objects are masked from 'package:base':
##
##     getOption, isOpen
```

Task 1. Your social network

For the first task, you have to load your vk.com network. Please follow the instructions posted on the course wiki or user VK Application. For FB users try to use NetVizz. If you did it correctly, you should have a GraphML file with your own network. Read it to R:

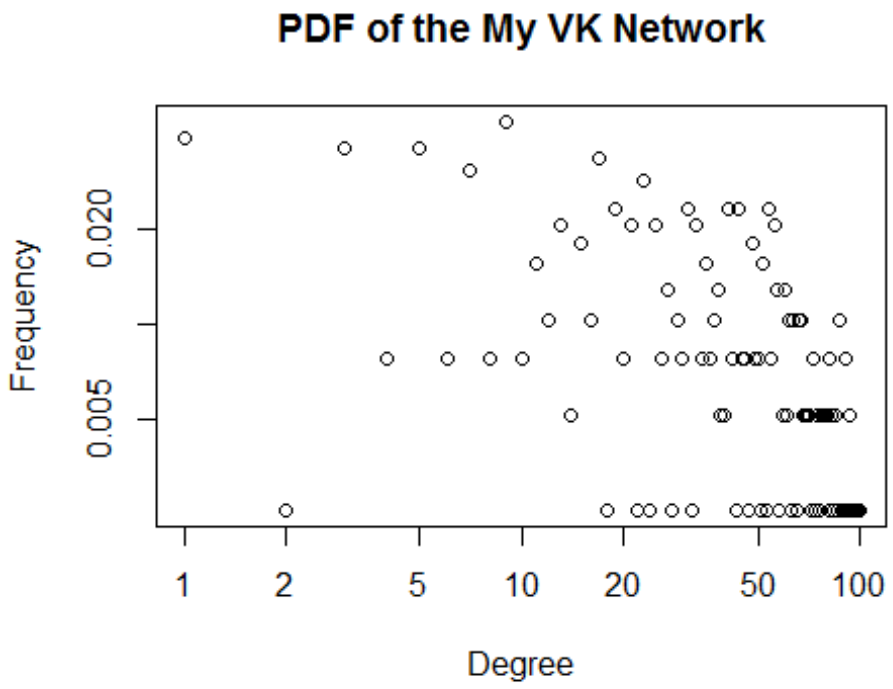
```
## Put your code here
g = gexf.to.igraph(read.gexf("vk-friends-266010761.gexf"))

z = get.adjacency(g)
z.names = 1:vcount(g)
colnames(z) = z.names
rownames(z) = colnames(z)
g = graph.adjacency(z, add.colnames = TRUE)

V(g)$label.cex = 0.5

plot(g)
```



1. Degree distribution

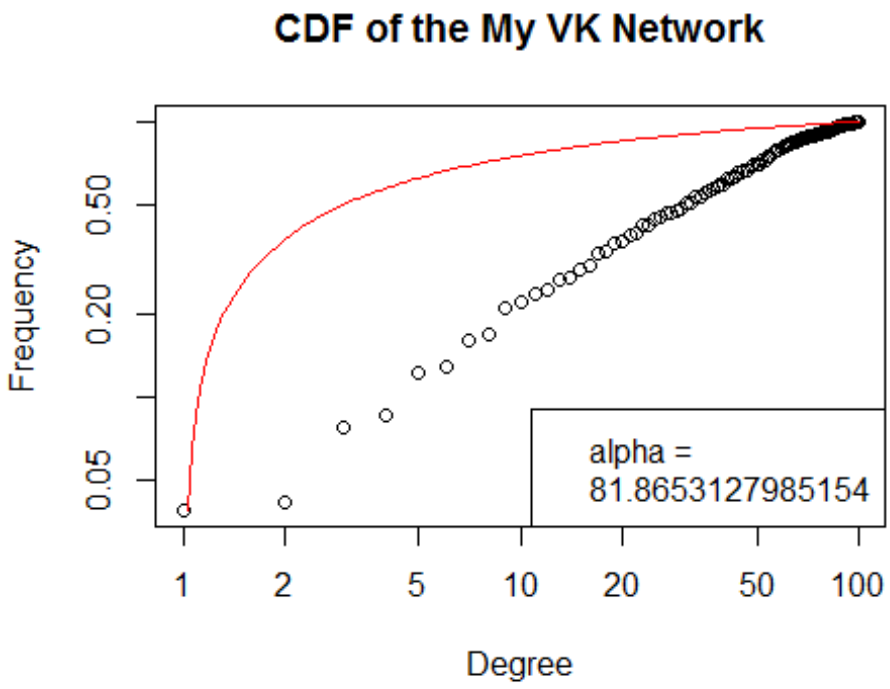
First, plot degree distribution of your network in log-log scales:

```
## Put your code here
p = degree.distribution(g, cumulative = FALSE)
pdf = p[which(p>0)]

plot(pdf, log="xy", main="PDF of the My VK Network", xlab="Degree",
ylab="Frequency")
```




```
#KS test
cdf = cumsum(pdf)
x = as.numeric(1:max(degree(g)))+1
g.fit = power.law.fit(x,implementation = "plfit")
x_min = g.fit$xmin
alpha = g.fit$alpha

func_CDF = function(x) return(1-(x_min^(alpha-1))*x^(1-alpha))
plot(cdf, log="xy", main="CDF of the My VK Network", xlab="Degree",
ylab="Frequency")
par(new=TRUE)
curve(func_CDF, from=x_min, to = max(x), n=length(x),log="xy", col="red",
add=FALSE, xlab = "", ylab = "", axes=FALSE)

## Warning in xy.coords(x, y, xlabel, ylabel, log): 1 y value <= 0 omitted
## from logarithmic plot
```

```
legend("bottomright", c("alpha = ", alpha))
```



Is there any correspondence between actual degree distribution of your network and the Power Law distribution? If not, explain why.

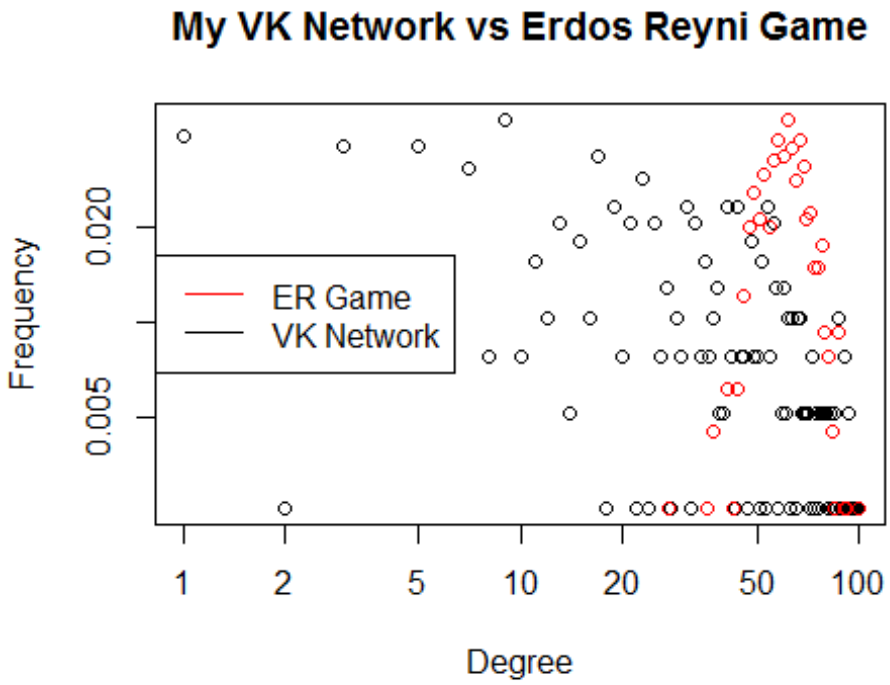
There is not a good agreement between the theoretical degree distribution and empirical data. This is due to the nature of empirical data. I should diversify my friend list.

Now, let's see how it would look if it was random. Produce Erdos-Renyi graph matching your real network (same number of nodes and same average degree). Compare it with your degree distribution.

```
## Put your code here
er = erdos.renyi.game(n=vcount(g), p.or.m = ecount(g), type=c("gnm"))
pdf.er = degree.distribution(er,cumulative = FALSE)
cdf.er = cumsum(pdf.er)

plot(pdf, log="xy", main="My VK Network vs Erdos Reyni Game", xlab="Degree",
ylab="Frequency")
par(new=TRUE)
plot(pdf.er, log="xy",col="red",  xlab="", ylab="", add=FALSE, axes=FALSE)

## Warning in xy.coords(x, y, xlabel, ylabel, log): 27 y values <= 0 omitted
## from logarithmic plot

## Warning in plot.window(...): "add" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "add" is not a graphical parameter
```

```
## Warning in title(...): "add" is not a graphical parameter
```

```
legend("left", c("ER Game","VK Network"), col = c("red", "black"), lty=1)
```



Put your comments here

2. Compute centrality metrics

Compute for your network:

- degree centrality
- closeness centrality
- betweenness centrality
- eigenvector centrality
- Bonacich power centrality
- Alpha centrality

```
## Put your code here
# 1. degree centrality
deg = degree(g)

# 2. closeness centrality
close = closeness(g)
```

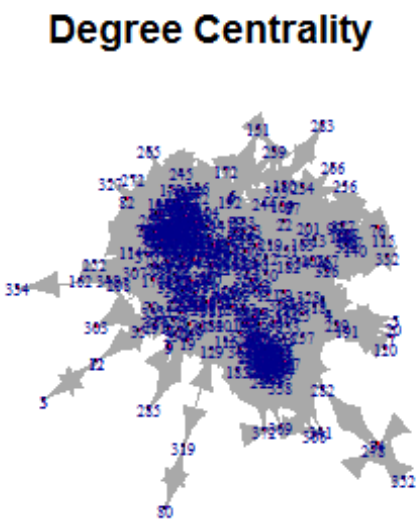
```
## Warning in closeness(g): At centrality.c:2784 :closeness centrality is not
## well-defined for disconnected graphs

# 3. betweenness centrality
btw = betweenness(g)

# 4.eigenvector centrality
eig = eigen_centrality(g)

# 5.Bonacich power centrality
bonc = bonpow(g)

# 6. Alpha centrality
alf = alpha.centrality(g)
```

Output six plots corresponding to six centrality metrics you've computed:

- Use first names of your friends as node labels on the graph (you may hide this information if you wish – change it by integer ID)
- Keep the same layout of the network
- Make node sizes and colours proportional to the respective centrality metrics

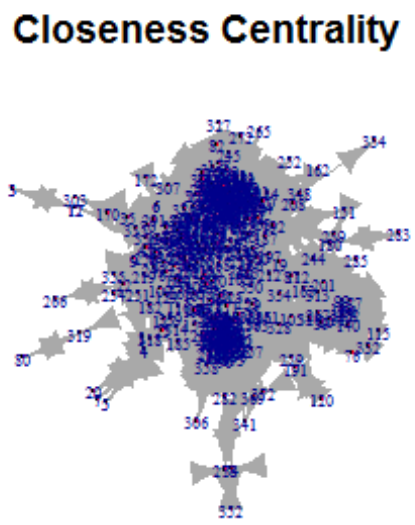
Hint: If you get stuck, lab 4 script might be helpful.

```
## Put your code here

z = delete.vertices(g, degree(g)==0 )

scl = function(c) return(scale(c, center = min(c), scale = max(c) - min(c)))

color.palette = colorRampPalette(c('red','blue'))
colors = color.palette(50)[as.numeric(cut(deg, breaks = 50))]

plot(z, layout=layout.fruchterman.reingold(z), vertex.color=colors,
vertex.size=scl(deg), main='Degree Centrality')

## Warning in layout[, 1] + label.dist * cos(-label.degree) * (vertex.size +
:
## longer object length is not a multiple of shorter object length

## Warning in layout[, 2] + label.dist * sin(-label.degree) * (vertex.size +
:
## longer object length is not a multiple of shorter object length
```



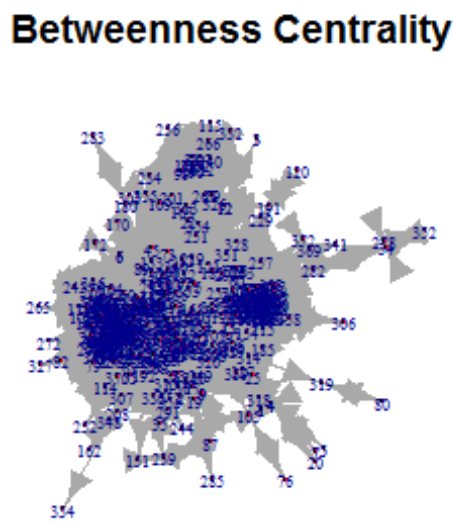

```
plot(z, layout=layout.fruchterman.reingold(z), vertex.color=colors,
vertex.size=scl(close), main='Closeness Centrality')
```

```
## Warning in layout[, 1] + label.dist * cos(-label.degree) * (vertex.size +
:
## longer object length is not a multiple of shorter object length

## Warning in layout[, 1] + label.dist * cos(-label.degree) * (vertex.size +
:
## longer object length is not a multiple of shorter object length
```



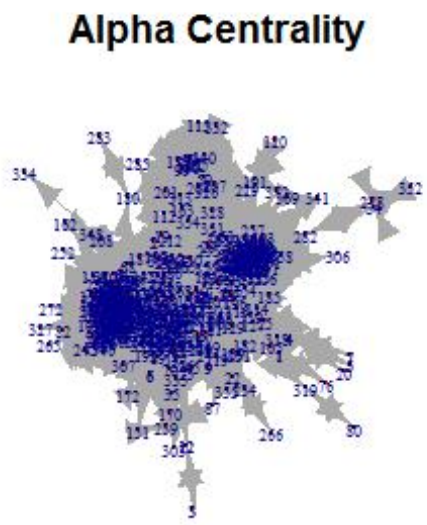

```
plot(z, layout=layout.fruchterman.reingold(z), vertex.color=colors,
vertex.size=scl(btw), main='Betweenness Centrality')
```

```
## Warning in layout[, 1] + label.dist * cos(-label.degree) * (vertex.size +
:
## longer object length is not a multiple of shorter object length

## Warning in layout[, 1] + label.dist * cos(-label.degree) * (vertex.size +
:
## longer object length is not a multiple of shorter object length
```



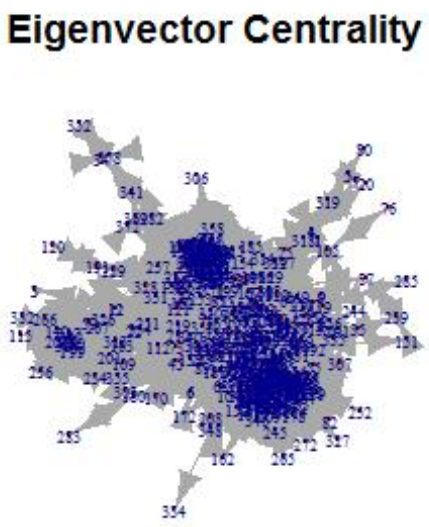

```
plot(z, layout=layout.fruchterman.reingold(z), vertex.color=colors,
vertex.size=scl(alf), main='Alpha Centrality')
```

```
## Warning in layout[, 1] + label.dist * cos(-label.degree) * (vertex.size +
:
## longer object length is not a multiple of shorter object length

## Warning in layout[, 1] + label.dist * cos(-label.degree) * (vertex.size +
:
## longer object length is not a multiple of shorter object length
```



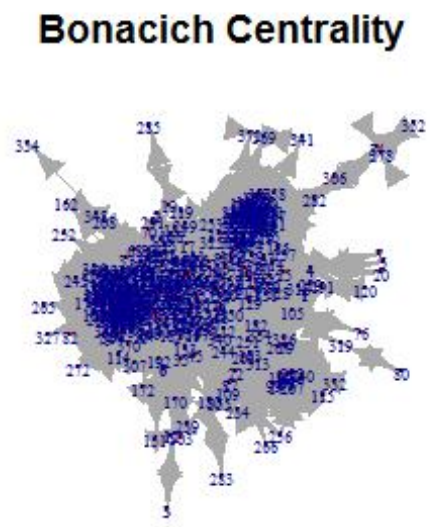

```
plot(z, layout=layout.fruchterman.reingold(z), vertex.color=colors,
vertex.size=scl(eig$vector), main='Eigenvector Centrality')
```

```
## Warning in layout[, 1] + label.dist * cos(-label.degree) * (vertex.size +
:
## longer object length is not a multiple of shorter object length

## Warning in layout[, 1] + label.dist * cos(-label.degree) * (vertex.size +
:
## longer object length is not a multiple of shorter object length
```




```
plot(z, layout=layout.fruchterman.reingold(z), vertex.color=colors,
vertex.size=scl(bonc), main='Bonacich Centrality')
```

```
## Warning in layout[, 1] + label.dist * cos(-label.degree) * (vertex.size +
:
## longer object length is not a multiple of shorter object length

## Warning in layout[, 1] + label.dist * cos(-label.degree) * (vertex.size +
:
## longer object length is not a multiple of shorter object length
```



Now, output top ten nodes in each ranking. Again, print only first names in your table to keep privacy:

```
## Put your code here
top10 = function(c) return(which(c > sort(c)[ length(c)-10 ]))

data.frame(Rank=1:10, Degree_centrality=top10(deg))
```

```
##    Rank Degree_centrality
## 1     1                31
## 2     2                46
## 3     3                80
## 4     4               155
## 5     5               183
## 6     6               275
## 7     7               292
## 8     8               312
## 9     9               323
## 10   10               344
```

```
data.frame(Rank=1:10, Closeness_centrality=top10(close))
```

```
##    Rank Closeness_centrality
## 1     1                  46
## 2     2                  89
## 3     3                 106
## 4     4                 114
```

```
## 5     5                   116
## 6     6                   183
## 7     7                   188
## 8     8                   284
## 9     9                   323
## 10   10                   344
```

```
data.frame(Rank=1:10, Bonacich_centrality=top10(bonc))
```

```
##    Rank Bonacich_centrality
## 1     1                  24
## 2     2                  28
## 3     3                  32
## 4     4                  93
## 5     5                 118
## 6     6                 167
## 7     7                 234
## 8     8                 248
## 9     9                 297
## 10   10                 375
```

```
data.frame(Rank=1:10, Alpha_centrality=top10(alf))
```

```
##    Rank Alpha_centrality
## 1     1                75
## 2     2               148
## 3     3               167
## 4     4               186
## 5     5               221
## 6     6               260
## 7     7               293
## 8     8               346
## 9     9               380
## 10   10               382
```

```
data.frame(Rank=1:10, Betweenness_centrality=top10(btw))
```

```
##    Rank Betweenness_centrality
## 1     1                    33
## 2     2                    46
## 3     3                    89
## 4     4                   114
## 5     5                   203
## 6     6                   292
## 7     7                   301
## 8     8                   321
## 9     9                   323
## 10   10                   344
```

```
data.frame(Rank=1:10, Eigenvector_centrality=top10(eig$vector))
```

```
##    Rank Eigenvector_centrality
## 1     1                     26
## 2     2                     31
## 3     3                     68
## 4     4                     80
## 5     5                     97
## 6     6                    183
## 7     7                    244
## 8     8                    275
## 9     9                    292
## 10   10                    337
```

When I checked for 31st node it is my friend Nicko, he is my close friend and we communicate a lot and also he is a public figure. It is interesting seeing him both in degree centrality and eigenvector centrality in the top rank

Task 2. Flickr network

In the second task, you will work with a large directed graph.

Please download flickr.mat

Data contains sparse matrix A and list of user names. This is a denser part of the Flickr photo sharing site friendship graph from 2006. Edge direction corresponds to friendship requests (following). Some of the links are reciprocal, others not

It's a Matlab file. How to deal with it in R? There is a package `R.matlab`. Please install it and call `library(R.matlab)`

Now use `readMat` function to read the file and extract adjacency matrix and a list of user names:

```
flickr = readMat("flickr.mat")
fmatrix=as.matrix(flickr[1]$A)
fnames=flickr[2]$names

f = graph.adjacency(fmatrix)
```

If you have trouble loading large mat file on your laptop — try to use HSE computer classes with installed R+RStudio.

Look at user names. You might want to remove spaces from the names. Use a function `gsub` to remove them:

```
fnames=gsub(" ", "", fnames)
```

Now create a graph, output the number of vertices and edges:

```
## Put your code here
```

```
paste("Number of vertices", vcount(f))
```

```
## [1] "Number of vertices 15724"
```

```
paste("Number of edges", ecount(f))
```

```
## [1] "Number of edges 510983"
```

Compute in- and out- degree centralities, PageRank, Hubs and Authorities for this network:

```
## Put your code here
f.deg.in = degree(f, mode="in")
f.deg.out = degree(f, mode="out")
f.pg.rnk = page.rank(f)
f.hubs = hub.score(f)
f.auth = authority.score(f)
```

Print top ten names in each ranking:

```
## Put your code here
data.frame(Degree_in=top10(f.deg.in), Names = fnames[top10(f.deg.in)])
```

```
##   Degree_in                                                Names
## 1      1465 awfulsara
## 2      2129 DrJoanne
## 3      3747 drp
## 4      4013 BombDog
## 5      5629 *Ivan*
## 6      7534 deborah lattimore
## 7      8235 Simon Pais
## 8     10846 antimethod
## 9     12813      MaD GiÂ®Lâ¢\231
```

```
data.frame(DEgree_out=top10(f.deg.out), Names = fnames[top10(f.deg.out)])
```

```
##    DEgree_out                                                 Names
## 1          60 anildash
## 2          92 maximolly
## 3         190 jakedobkin
## 4        2854 brainware3000
## 5        2903 Jakes_World
## 6        2959 Buntekuh
## 7        3809 pixietart
## 8        5039 tozzer
## 9        6483 AtiRanA
## 10       7569 Andreia Lopes
```

```
data.frame(Page_rank=top10(f.pg.rnk$vector), Names =
fnames[top10(f.pg.rnk$vector)])
```

```
##    Page_rank                                                  Names
## 1       1465 awfulsara
## 2       2129 DrJoanne
## 3       3747 drp
## 4       4013 BombDog
## 5       4965 cymagen
## 6       5629 *Ivan*
## 7       7534 deborah lattimore
## 8       8235 Simon Pais
## 9      10846 antimethod
## 10     12813      MaD GiÂ®Lâ¢\231
```

```
data.frame(Hub_score=top10(f.hubs$vector),
Names=fnames[top10(f.hubs$vector)])
```

```
##    Hub_score                                                  Names
## 1       3821 mrpiink
## 2       4996 noahstone
## 3       6518 isherwood
## 4       6572 sgoralnick
## 5       7380 *starlet*
## 6       7954 automat
## 7      10487 brynfoto
## 8      11304 liquid pixel
## 9      12675 lorrainemd
## 10     14004 schizoo23
```

```
data.frame(Authority_score=top10(f.auth$vector),
Names=fnames[top10(f.auth$vector)])
```

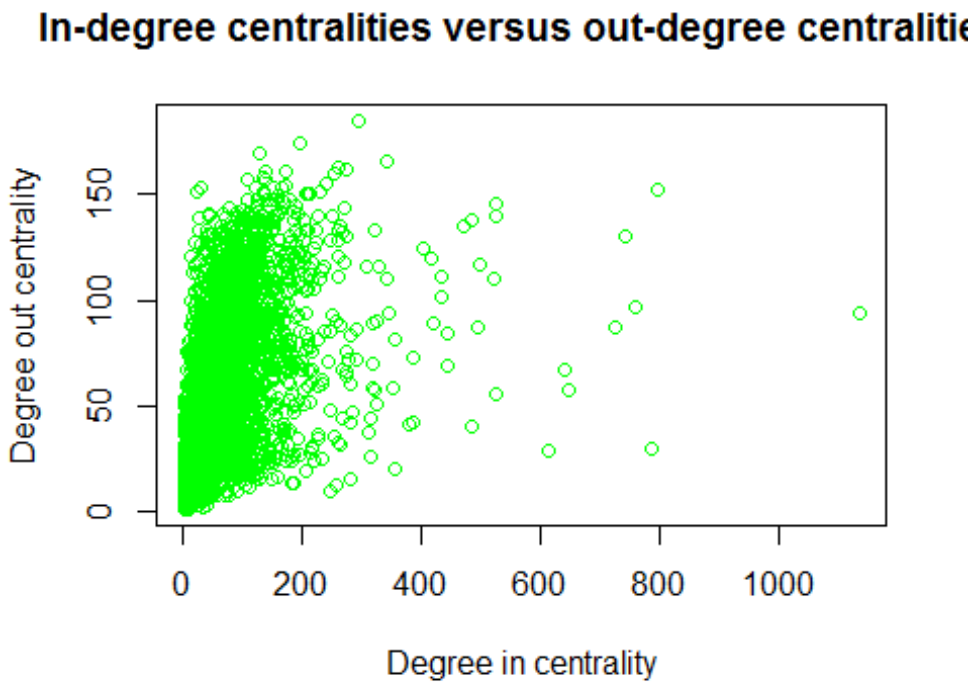
```
##    Authority_score                                            Names
## 1             1465 awfulsara
## 2             2129 DrJoanne
## 3             3747 drp
## 4             4013 BombDog
## 5             4301 :Nikola
## 6             4965 cymagen
## 7             5629 *Ivan*
## 8             7534 deborah lattimore
## 9             8235 Simon Pais
## 10           10846 antimethod
```

Produce the following plots:

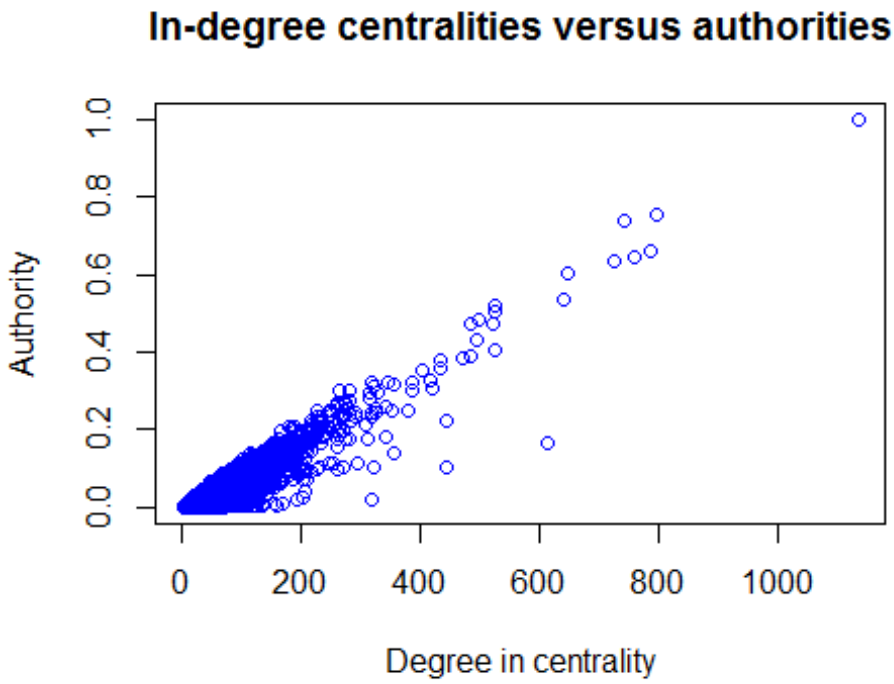
- In-degree centralities versus out-degree centralities
- In-degree centralities versus authorities
- Out-degree centralities versus hubs
- Hubs versus authorities

- PageRank versus hubs
- PageRank versus authorities

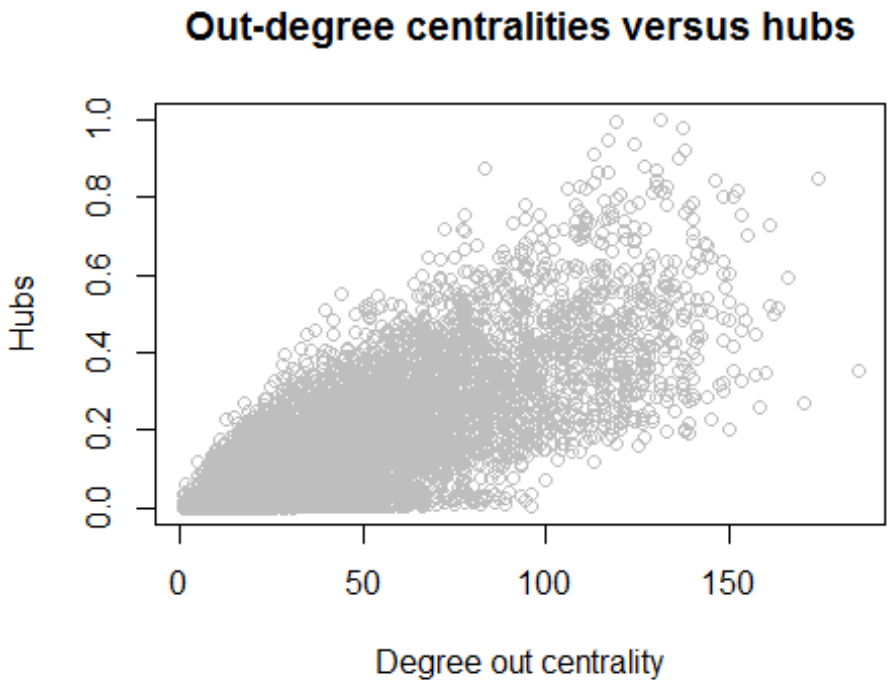
```
## Put your code here
plot(f.deg.in, f.deg.out, xlab = "Degree in centrality", ylab = "Degree out
centrality", col="Green", main="In-degree centralities versus out-degree
centralities")
```



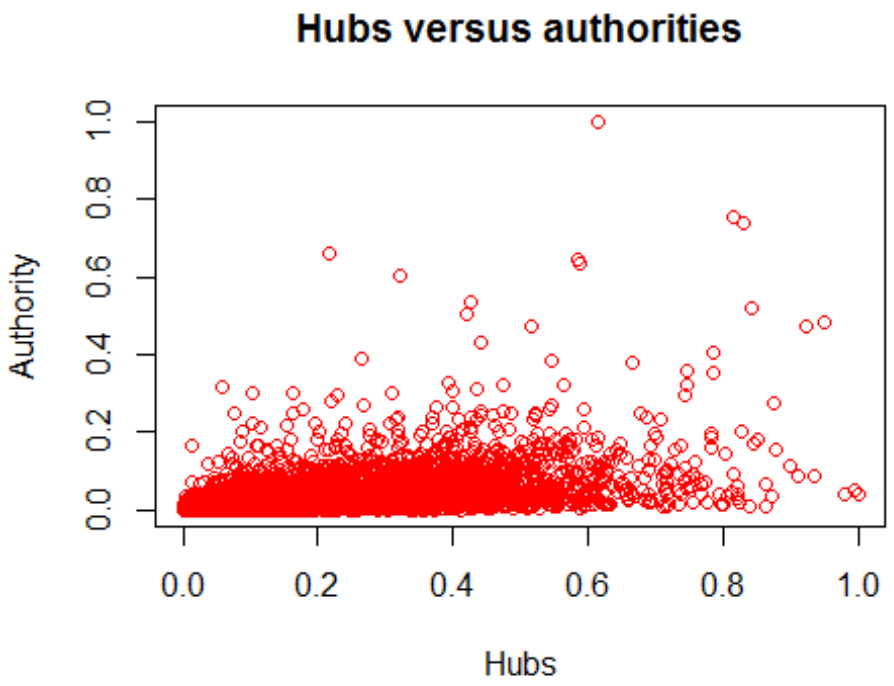

```
plot(f.deg.in, f.auth$vector, xlab = "Degree in centrality", ylab =
"Authority", col="Blue", main="In-degree centralities versus authorities")
```



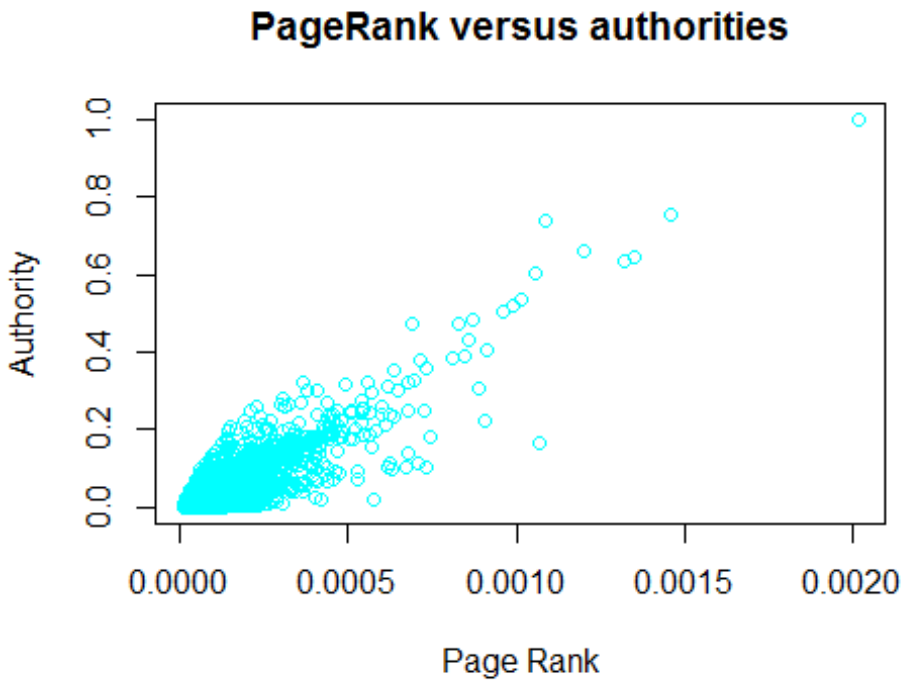

```
plot(f.deg.out, f.hubs$vector, xlab = "Degree out centrality", ylab = "Hubs",
col="Grey", main="Out-degree centralities versus hubs")
```



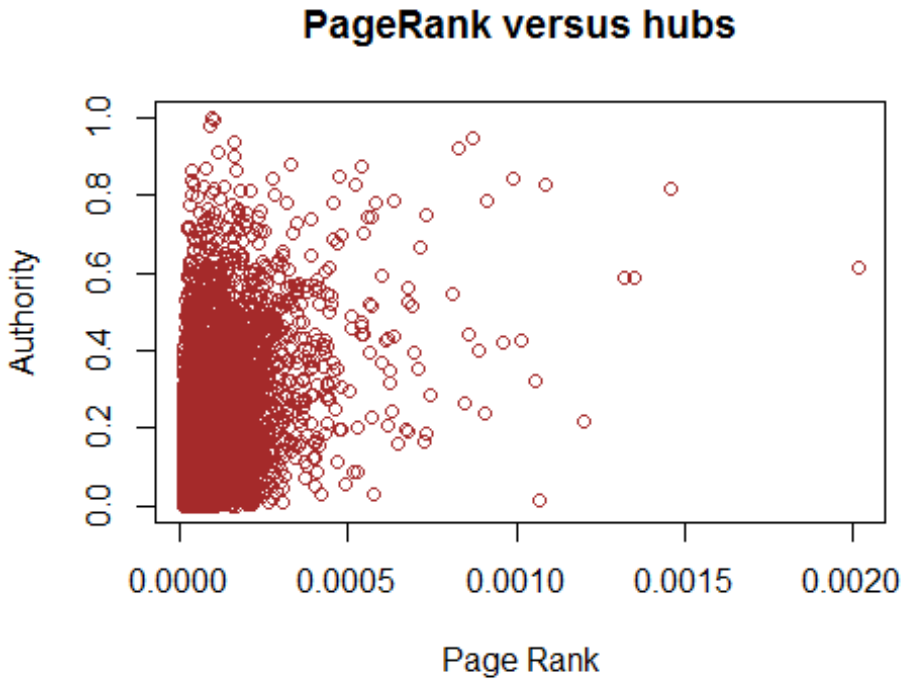
```
plot(f.hubs$vector, f.auth$vector, xlab = "Hubs", ylab = "Authority",
col="Red", main="Hubs versus authorities")
```



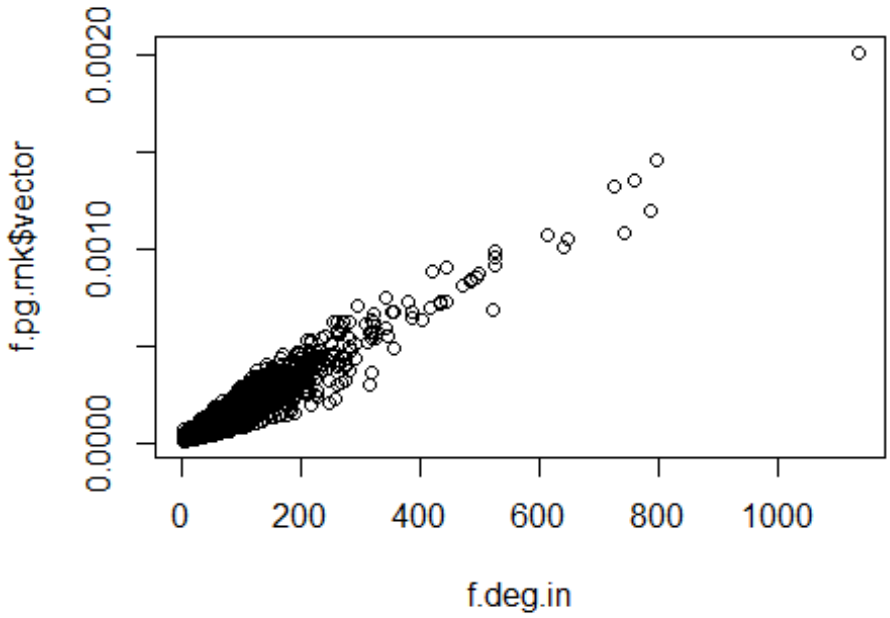

```
plot(f.pg.rnk$vector, f.auth$vector, xlab = "Page Rank", ylab = "Authority",
col="Cyan", main="PageRank versus authorities")
```




```r
plot(f.pg.rnk$vector, f.hubs$vector, xlab = "Page Rank", ylab = "Authority",
col="Brown", main="PageRank versus hubs")
```



```
plot(f.deg.in, f.pg.rnk$vector)
```



One can observe a strong correlation between, 1. Degree in centrality and Authority 2. Hubs and Degree out 3. Authority and Page rank This also implies that Degree-in and Page rank also should be strongly correlated which was also plotted