Class project document

Team member Mengxi Nie, Yi Tu, Cong Tang, Yibu Lin

Professor Kaisler

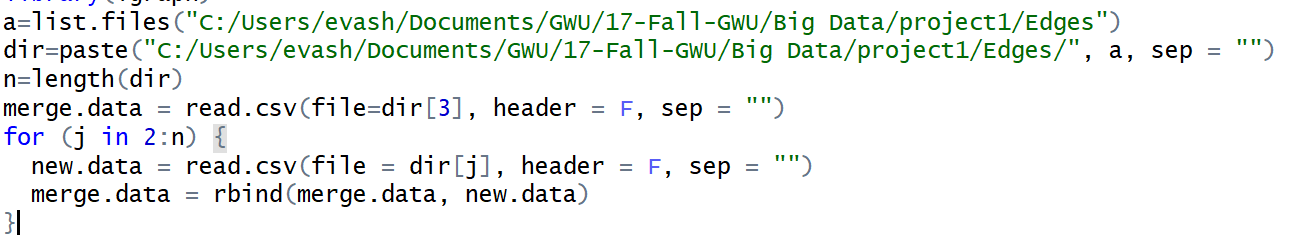
Intro. to Big Data & Analytics CSCI 6444 8

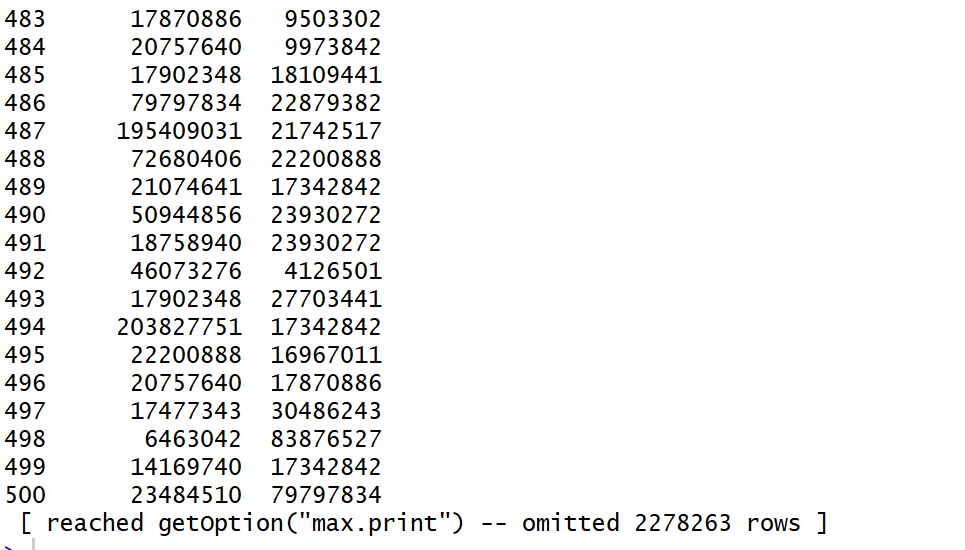
October 1, 2017

The George Washington University

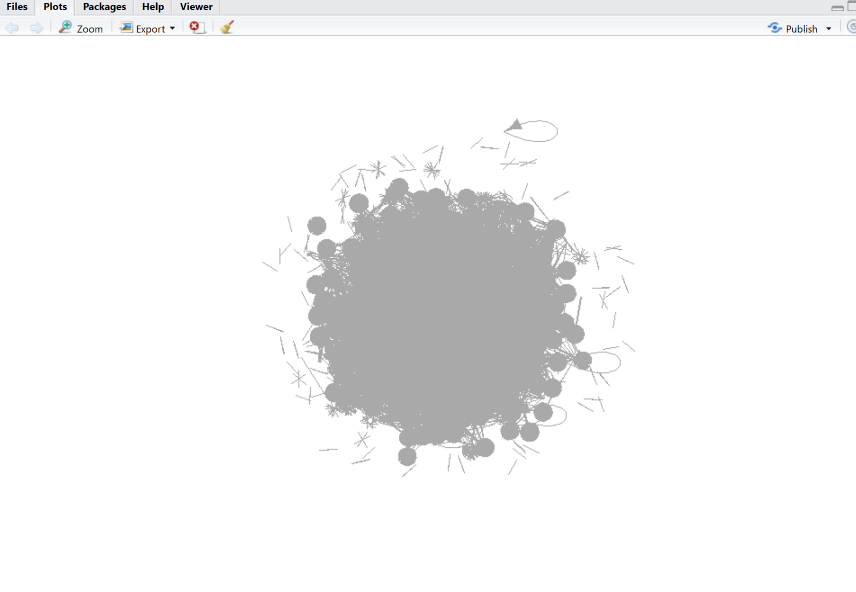
a.Data processing

First part is data processing. Since we have a huge number of the edges files, we try to combine all these data together, and build a data frame and use this data frame to build a whole graph of this twitter user dataset. The picture shows the functions of data processing and result:



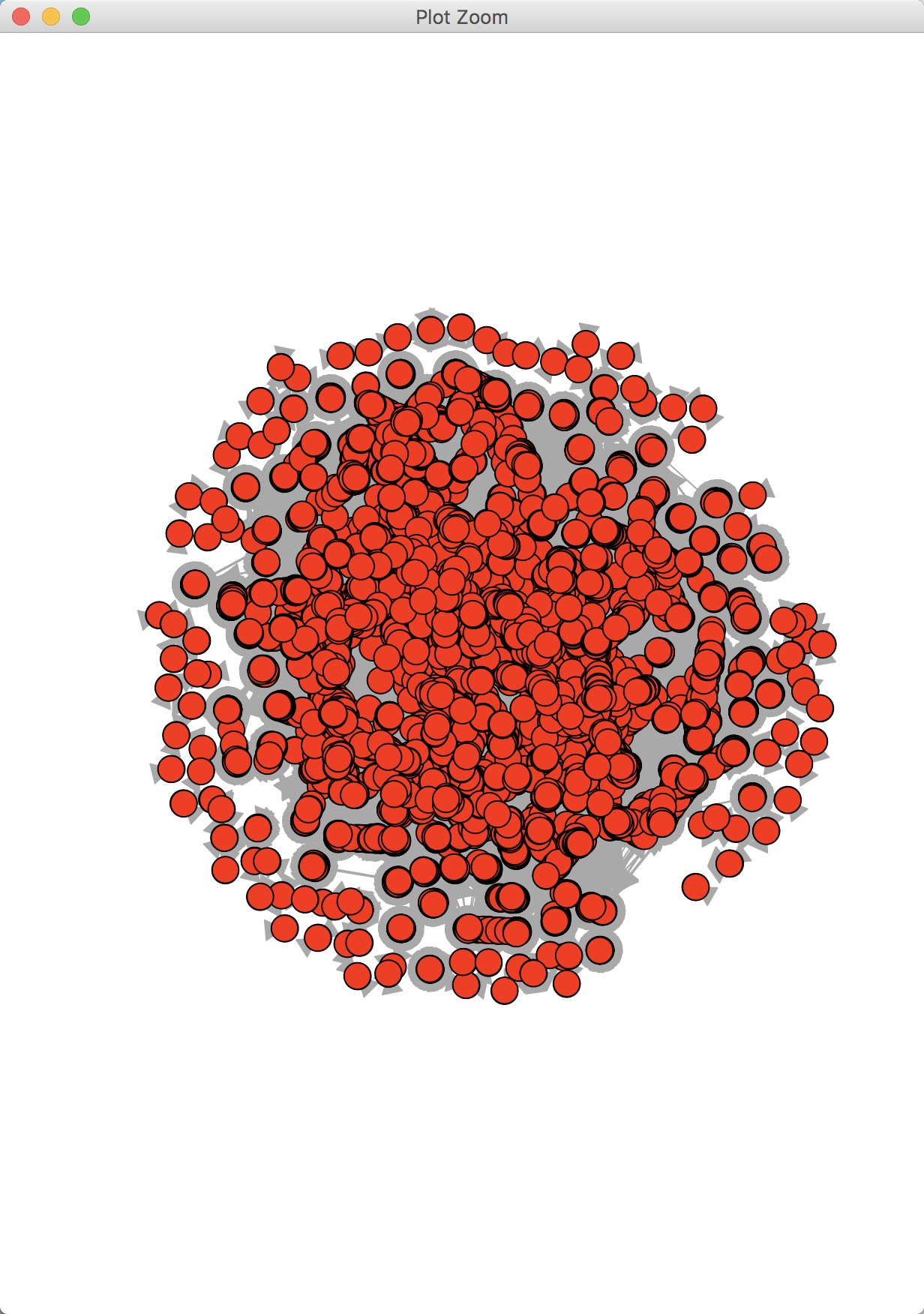


Then we use *graph<- graph\_from\_data\_frame(merge.data, directed = TRUE, vertices = NULL)* to build a directed graph. And when we try to draw the whole graph, it failed like the assignment document said. The result just like shows as followed and the program was always terminated.

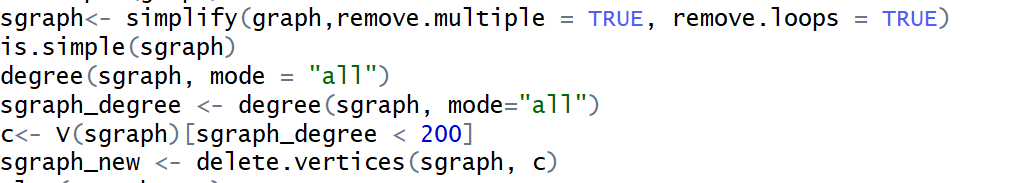


2. Simplify graph

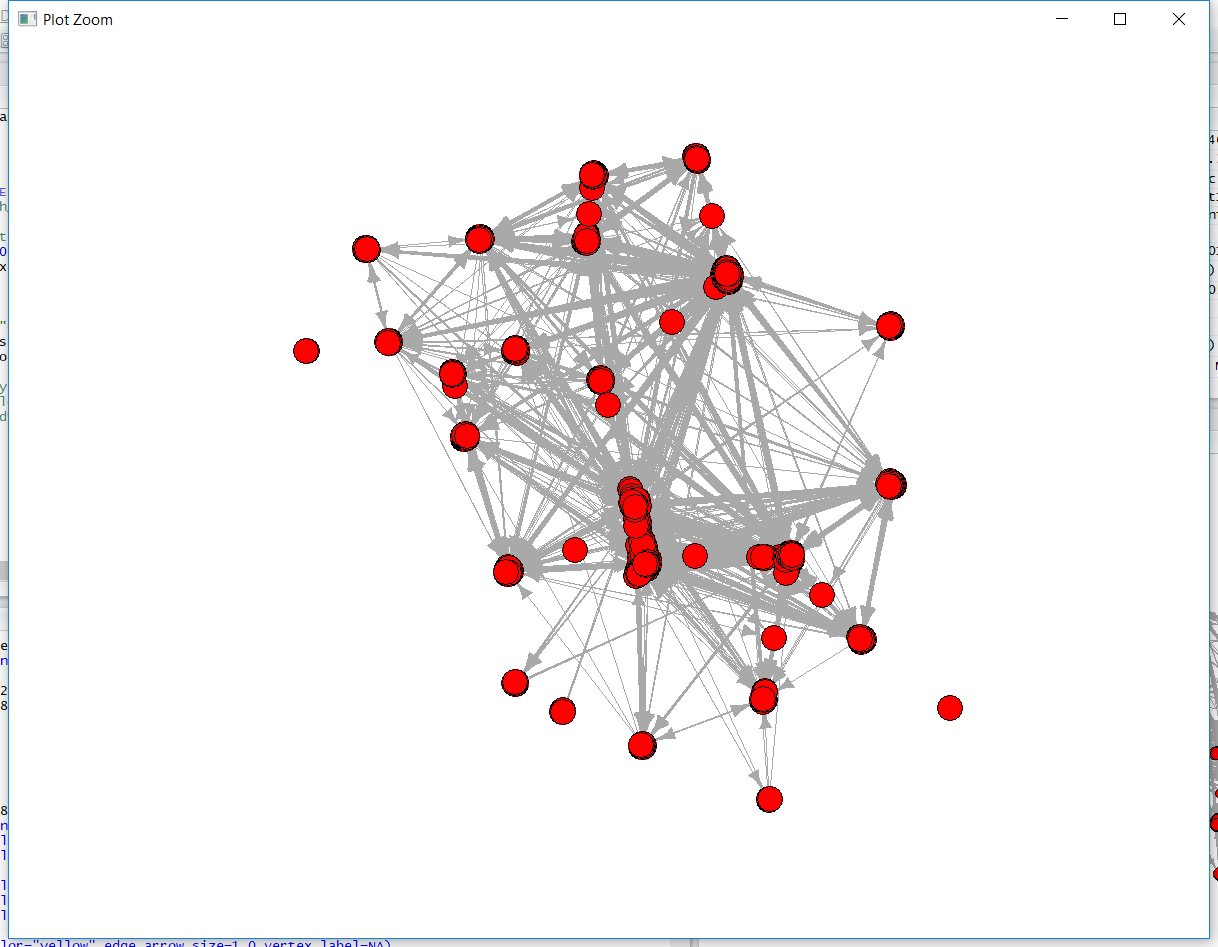
Since we cannot draw the whole graph with our huge data. So, we try to simplify the graph. First, we use the simplified function. Then we got the graph as showed:



However, in this result, the clusters cannot figure out clearly. We decide to delete more unimportant nodes whose degree is no more than 200. Here are the functions to delete those nodes:



Then, we got our simplified graph to use. We can see the clusters more clearly.



The R function list:

1. In data processing: list.files(), paste(), length(), read.csv(), rbind()

2 graphs build and simplify: graph\_from\_data\_frame(), simplify(), degree(), delete.vertices().

3 graph analyses : diameter(), get.diameter(), largest\_cliques(), power\_centrality()

What we learned about the data

1. The original data edges data is complicated and we must combine them together to build a graph.

2. We process our data with the function in R of process the data in file

3. After build graph, the simplify and delete nodes make us spend some time. We try adjust the degree value which should less than the degrees of nodes in the graph, then making the graph more clear.

4. In order to make us graph more clearly, we decide to cut the name of the nodes.

5. In the process of build ego graph, at first, we try to use the circle data. However, we are failed to find out a way to build a graph with the circle data, so we just use our graph which we built with edges data.

b. Explore other functions in the igraph package – at least 10 of them. You may have to do some programming in R. There are numerous books posted on the Blackboard.

We list more that 15 functions:

1. Read the file

edge1<-read.csv("./data/Edges/edges.csv",header=F)

vertex3<-read.csv("./data/Vertex/vertex.csv",header=F)

1. Set the format of the variable

edge1[,1]=as.character(edge1[,1])  
vertex3[,1]=as.character(vertex3[,1])

1. Graph build

people = data.frame(id = vertex3[,1], name = vertex3[,2])

g=graph.data.frame(d=edge,direct=T,vertices=people)

1. Draw

png('./data/picture.png',width=900,height=900)

set.seed(20)

plot(g, layout = layout.fruchterman.reingold, vertex.size = V(g)$size+2,vertex.color=V(g)$color,vertex.label=V(g)$label,vertex.label.cex=1,edge.color = grey(0.5), edge.arrow.mode = "-",edge.arrow.size=5)

dev.off()

Explain the variables in plot function:

* 1. layout: Set the layout of the graph

layout、layout.auto、layout.bipartite、layout.circle、layout.drl、layout.fruchterman.reingold、layout.fruchterman.reingold.grid、layout.graphopt、layout.grid、layout.grid.3d、layout.kamada.kawai、layout.lgl、layout.mds、layout.merge、layout.norm、layout.random、layout.reingold.tilford、layout.sphere、layout.spring、layout.star、layout.sugiyama、layout.svd

* 1. vertex.size: set the size of different node

de<-read.csv("c:/degree-info.csv",header=F)

V(g)$deg<-de[,2]

V(g)$size=2

V(g)[deg>=1]$size=4

V(g)[deg>=2]$size=6

* 1. vertex.color: set the color

color<-read.csv("c:/color.csv",header=F)

col<-c("red","skyblue")

V(g)$color=col[color[,1]]

* 1. vertex.label:set the laber of node

V(g)$label=V(g)$name

vertex.label=V(g)$label

* 1. vertex.label.cex: set the font size of the node label
  2. edge.color:

E(g)$color="grey"

for(i in 1:length(pa3[,1])){

    E(g,path=pa3[i,])$color="red"

}

edge.color=E(g)$color

* 1. Set the width of edge

E(g)$width=1

1. Cluster analysis
   1. cluster of edge betweenness

system.time(ec <- edge.betweenness.community(g))

print(modularity(ec))

plot(ec, g,vertex.size=5,vertex.label=NA)

* 1. random walk

system.time(wc <- walktrap.community(g))

print(modularity(wc))

#membership(wc)

plot(wc , g,vertex.size=5,vertex.label=NA)

* 1. Eigenvalue

system.time(lec <-leading.eigenvector.community(g))

print(modularity(lec))

plot(lec,g,vertex.size=5,vertex.label=NA)

* 1. Greedy strategy

system.time(fc <- fastgreedy.community(g))

print(modularity(fc))

plot(fc, g,vertex.size=5,vertex.label=NA)

* 1. Muti-level clustering

system.time(mc <- multilevel.community(g, weights=NA))

print(modularity(mc))

plot(mc, g,vertex.size=5,vertex.label=NA)

* 1. Label propagation

system.time(lc <- label.propagation.community(g))

print(modularity(lc))

plot(lc , g,vertex.size=5,vertex.label=NA)

1. File output

zz<-file("d:/test.txt","w")

cat(x,file=zz,sep="\n")

close(zz)

1. View the variable data type and length

mode(x)

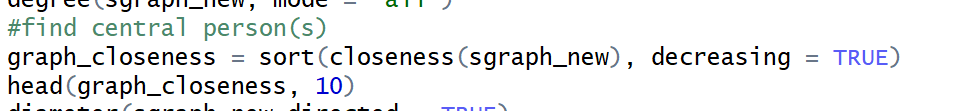
length(x)

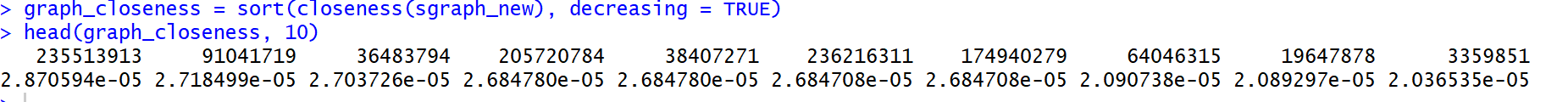
1. Diameter of a graph: The diameter of a graph is the length of the longest geodesic

diameter(graph,directed=True,unconnected=TRUE,weights=null)

c. Graph analysis(Q5)

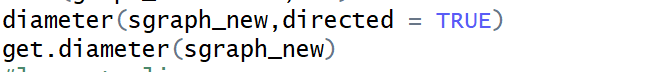
1 central person  
In order to determine the central person, we try to find the nodes with the highest closeness centrality. This is the most straightforward method. It calculates the shortest path between all people, then assign a value to each person based on its sum of shortest path. We list the first 10 highest closeness centrality nodes, find out it should be some clusters existed.

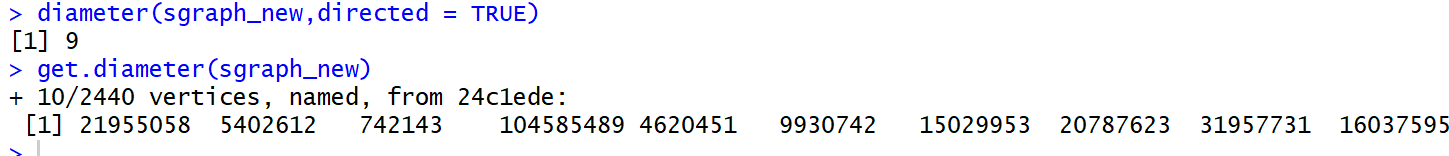




2 longest path

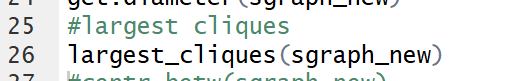
In order to determine the longest path, we used the diameter() function in the igraph package.

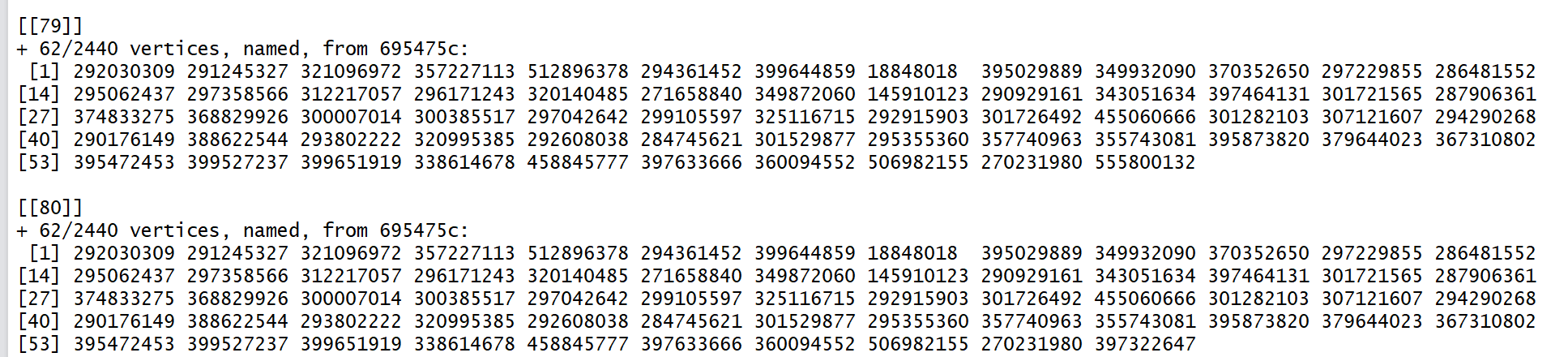




3 Largest clique

In order to find the largest clique, we use the largest\_cliques() function in the igraph package. We found out that the largest clique in this network have 62 nodes and there 80 cliques have same numbers of nodes.



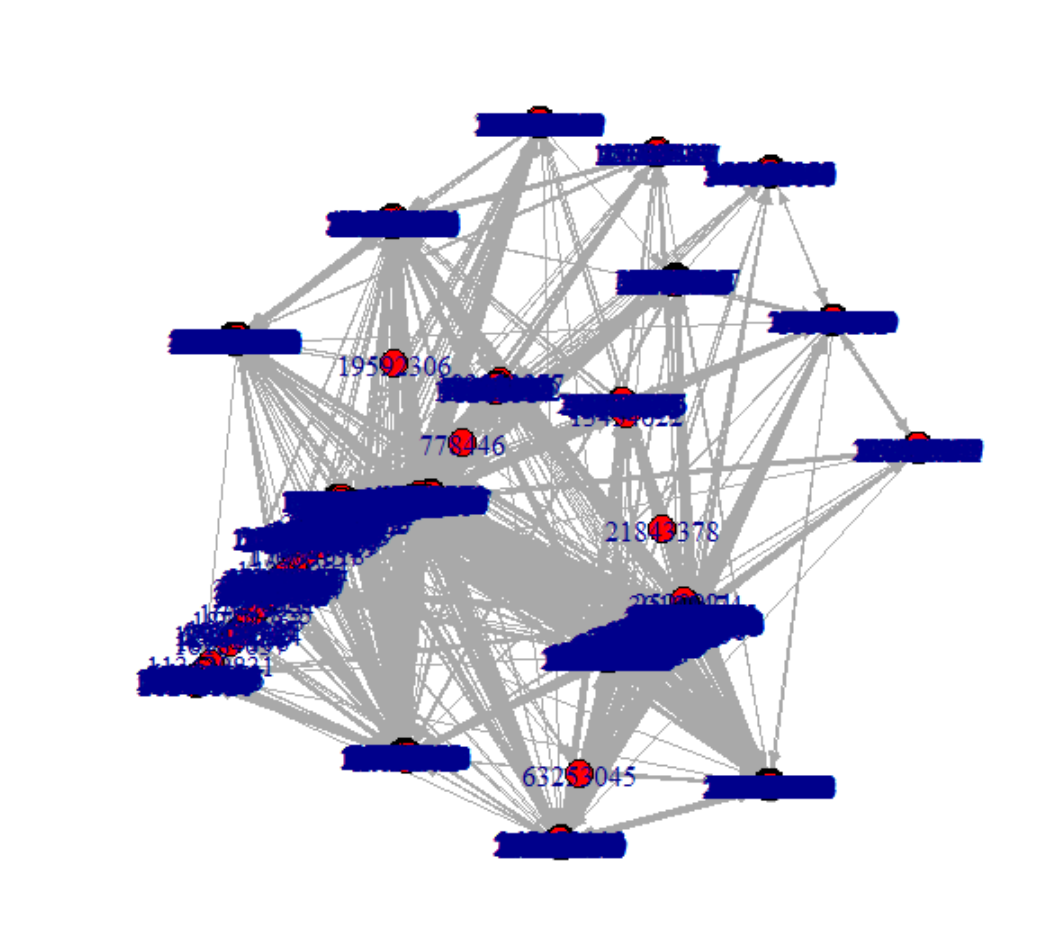


4 Ego

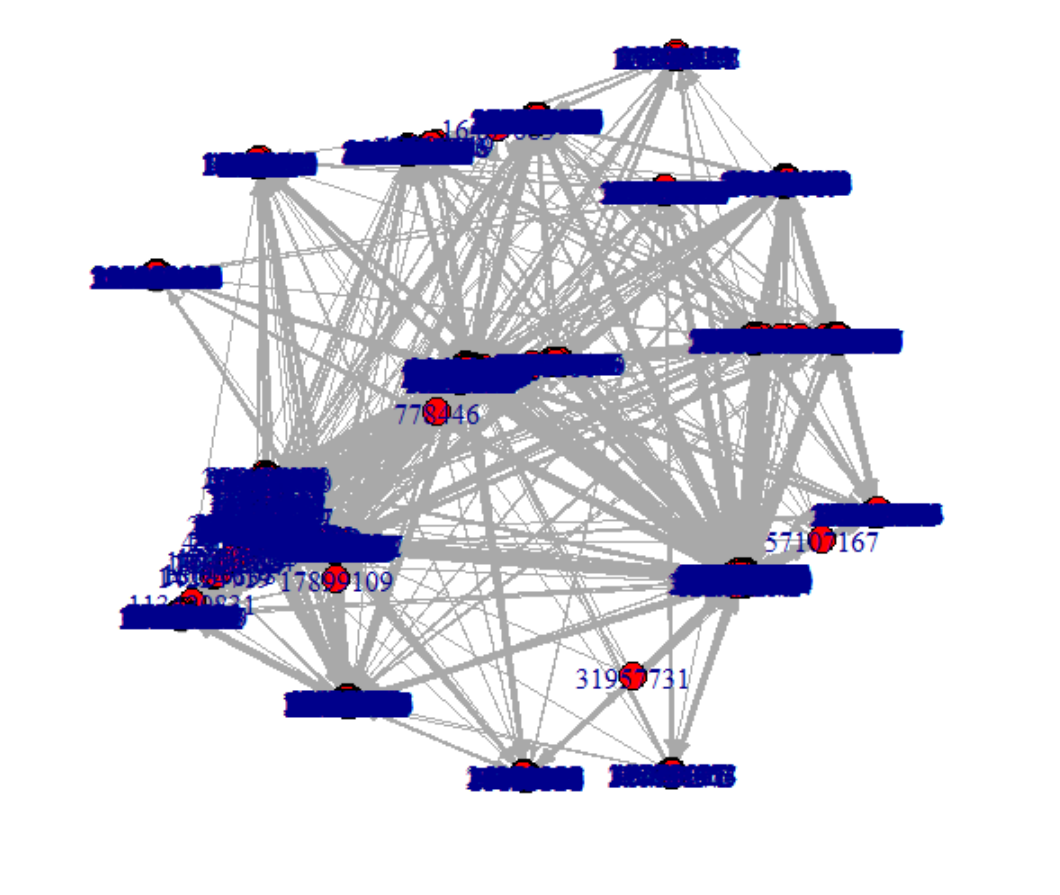
We've tried to make and plot an ego network of a given vertex. At first, we used the function make\_ego\_graph() from igraph package to build graphs for each vertex. We set the order 3 which means the graph shows all vertices which are within 3 node lengths from our chosen vertex.

Because the return value of make\_ego\_graph() is a list format, we can plot the graph with different item in the list and get different graph result.

When plot(graph3[[1]])：



When plot(graph3[[11]])：



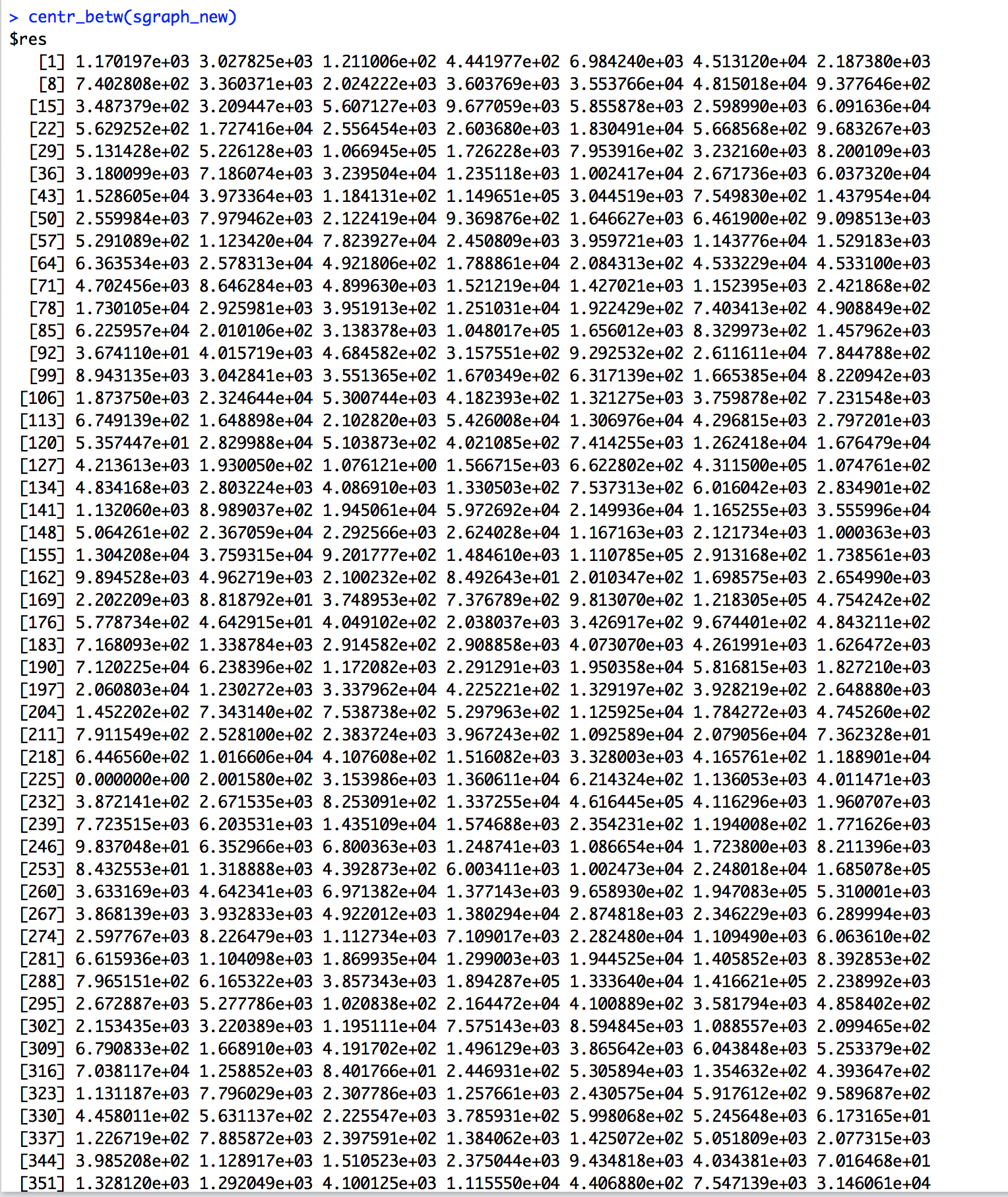
We can see the different nodes shows in the graph.

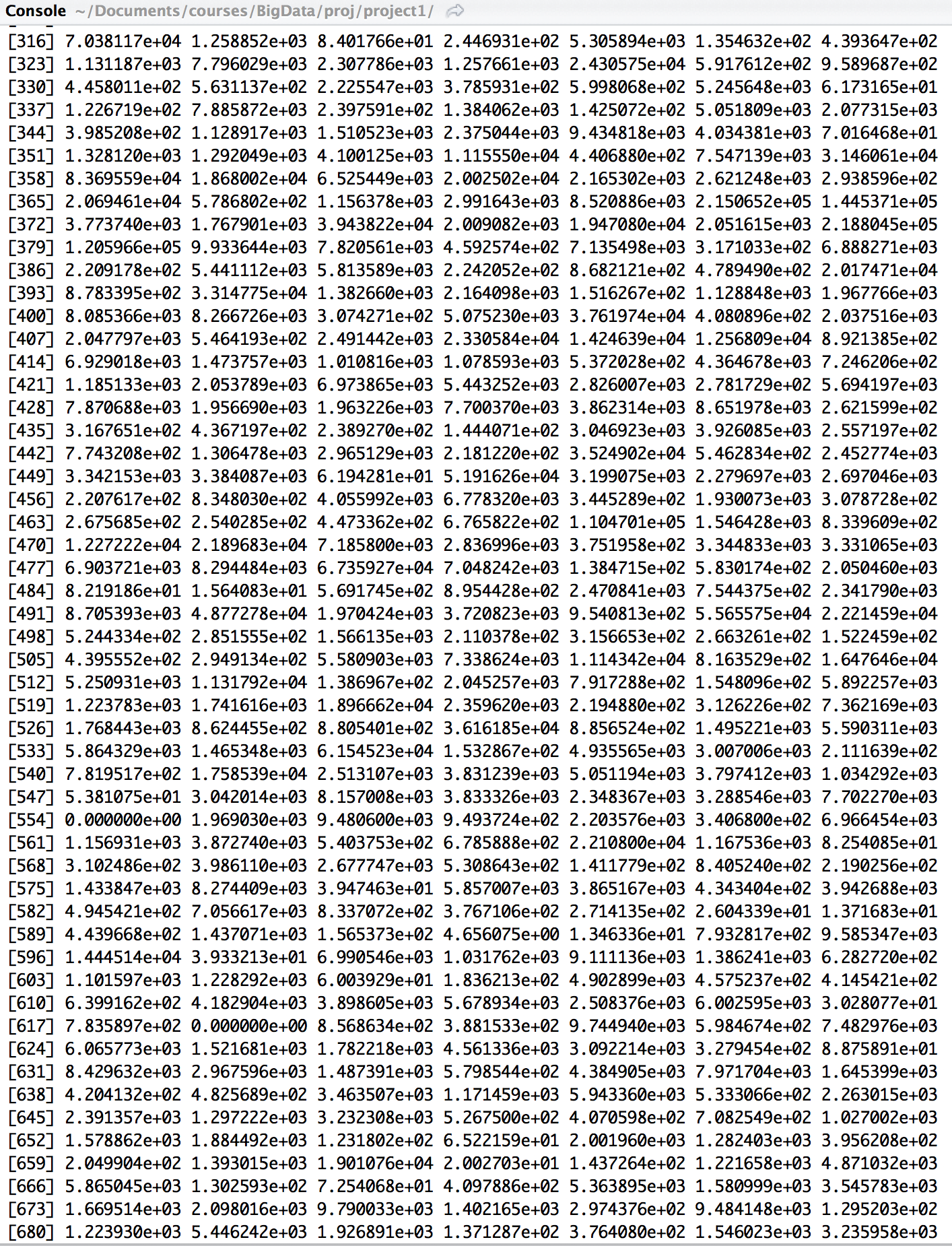
5

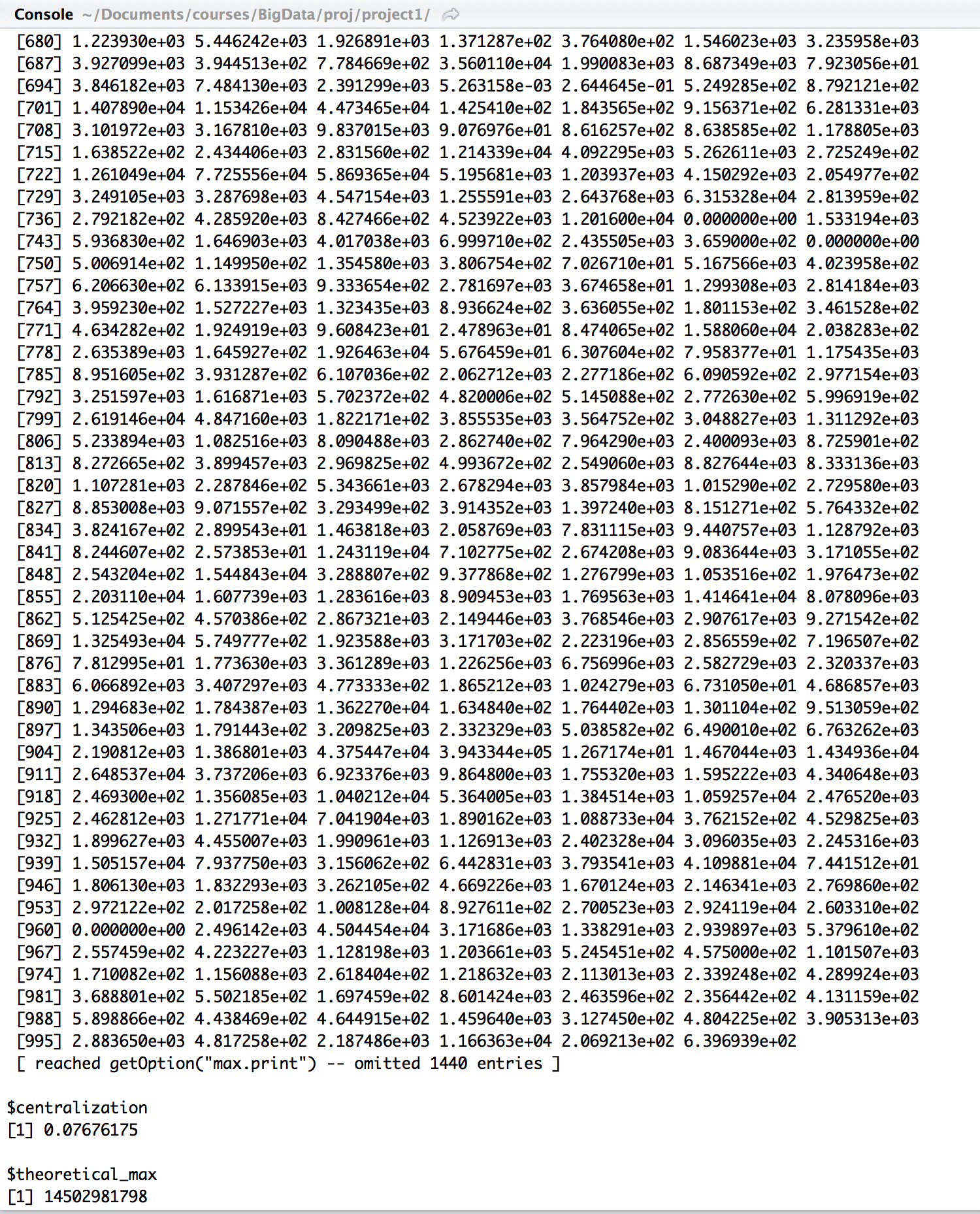
1. Betweenness centrality

We used centr\_betw() function from the igraph package to deter-

mine betweenness centrality of this dataset.







2）Power centrality

We simply used power\_centrality() function from the igraph package to deter-

mine power centrality of this dataset.

