Group 9

Yucheng Lu (G20257365)

Zefeng Song (G22237721)

Linge Yan(G45235107)

Bo Yuan（G30021277）

Abdulmohsen Almalki (G21772155)

Functions:

1. Setwd
2. Library
3. graph_from_data_frame
4. simplify
5. plot
6. is.simple
7. degree
8. hist
9. igraph.version
10. igraph.options
11. has.multiple
12. is.weighted
13. get.adjacency
14. gsize
15. is.directed
16. is.named
17. is.bipartite
18. degree(g)
19. diameter
20. max_cliques
21. ego(g)

22. betweenness

23. power_centrality

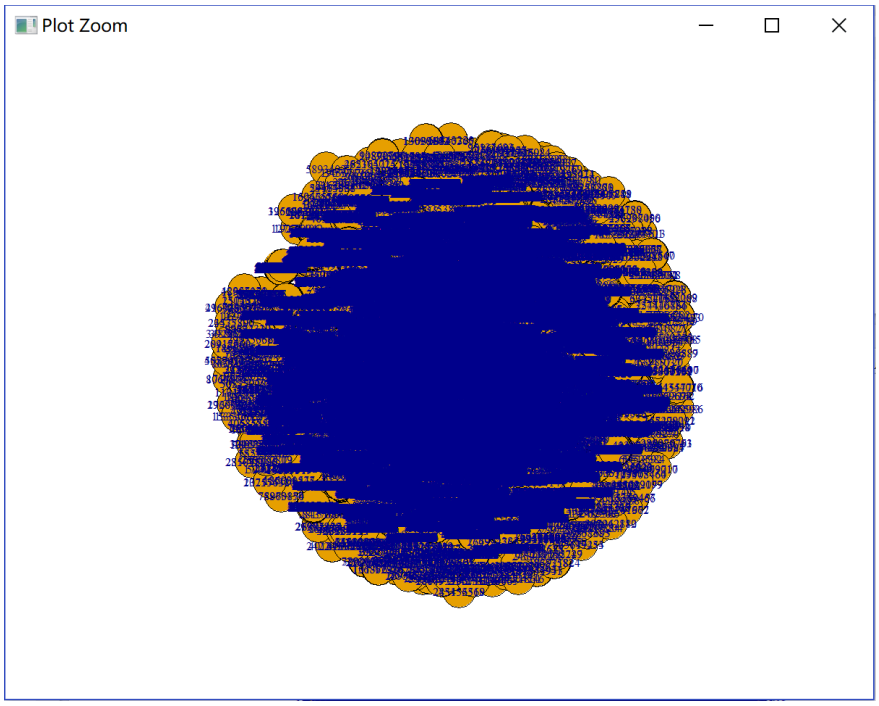
1-3 Loading the data and building the graph

```
require(igraph)
#loading the file
FileName <- dir("Edges")
NodeId <- unlist(strsplit(FileName, ".edges"))
EdgesFileName <- paste("Edges/",FileName,sep="")
dat <- data.frame(stringsAsFactors=FALSE)
#reading the relationship
edges_list <- list()
for( i in EdgesFileName)
{
    dati <- read.table(i,stringsAsFactors=FALSE)
    dat <- rbind(dati,dat)
    edges_list <- c(edges_list, dati)
}
#building the graph
g= graph.data.frame(dat,directed=TRUE)
g <- simplify(g, remove.loops = TRUE)
g <- simplify(g, remove.multiple = TRUE)
```

4. Experiment with some of the functions that I have shown in the lecture notes and associated PPT file on Blackboard. Present the results in your writeup.

```
# Change file path
setwd("C:/Course/Big data/Edges")
# Select all files end with "edges"
filelist = list.files(pattern = ".*.edges")
```

```
# Read data
datalist = lapply(filelist, function(x)read.table(x, header=F))
# Make dataframe
dataframe = do.call("rbind", datalist)
# make a graph
library(igraph)
g = graph_from_data_frame(dataframe, directed = FALSE, vertices = NULL)
# Simplify function
g = simplify(g, remove.loops=TRUE)
g = simplify(g, remove.multiple=TRUE)
plot(g)
# Original size of the graph is 2286909
# After simplify the edges is 1242390
# Use is.simple()to check whether it is simple graph
```

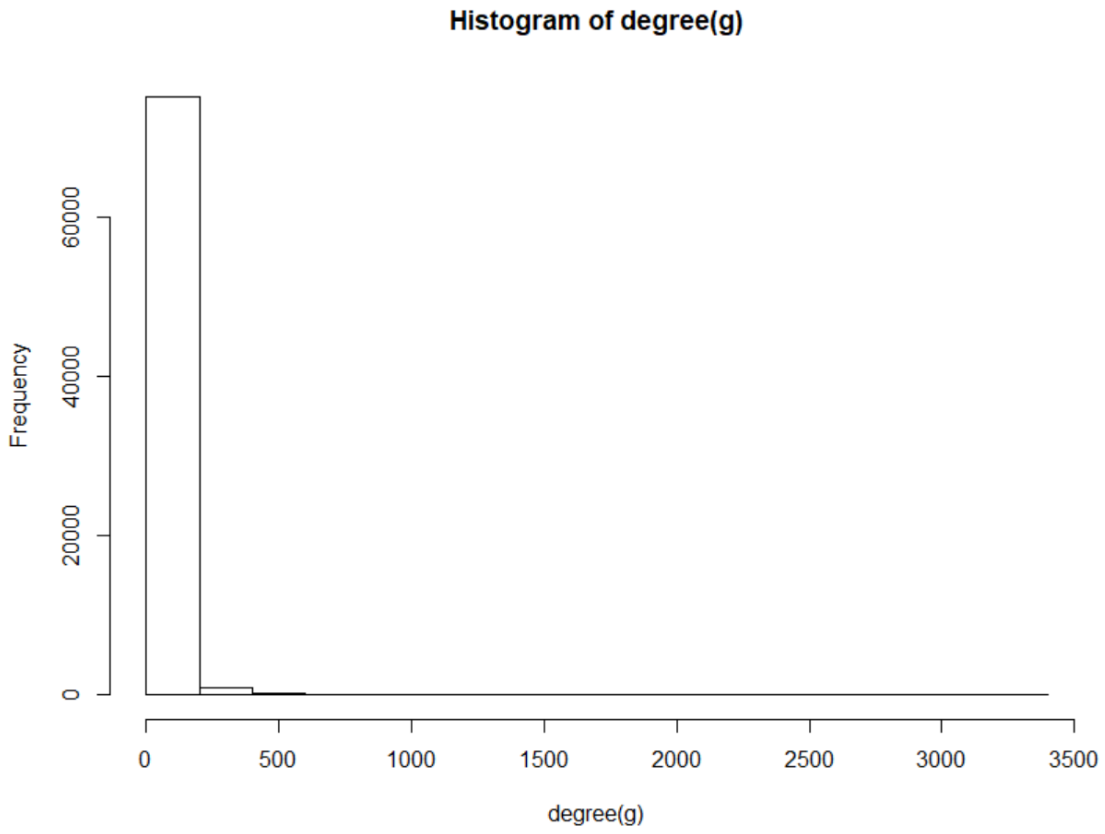


```
> degree(g)
214328887  17116707 380580781 221036078 107830991 151338729  19705747 222261763  19933035 158419434 149538028
       284        257        306        116        242        600        311        277         56        520        436
364971269 100581193 113058991 406628822 460282402 280935165 285312927 279787626 394263193 254839786 204317520
        59        374        267        199         59        526        115        382        303        404        586
 67864340 270449528 153226312 124528830 220368467 206923844 207594668 451250774   6581292  34428380 196327549
       171        510        531        288        198        336         25        195        214       2476        194
274153775 324201646  88097807 276308596  16870853 314316607 299715516 276577539 107511013 258140947 123371682
       430          8        398        208         85        380         78        209         17        319        347
461410856 117901353  56860418  43003845 413275344 358775055 307458983  69592091  35148062  57490887  37699718
       327        219        356       2749         20        346        392        475         23        195        275
148519842 225444667 283306479 220068522 155661154 400689940  72818790 195475105 157829215 134940306  31414569
        93        173        225         43        230        103        267        680        335        330        130
131613362 115221382 200559228 439788025 262340283 257236842 463952369 254610699 226629405 529007327  70492333
       288         98        196        256        351        159        318         41        278        111        375
166214735 236184723  86221475 116498875 222411742  17675120 250178329 103598216 172883064 292598082 145845459
       169        198        203        332         47        227         48        270        418         98        189
294752666 163238842  29911100 197504076 248883350 260769396 288485704 229425177 252770012  26929220 259842341
        72        162         93        547        198        112        228        151         80        568        356
262802533 213777144  46209291 239144776   8163442  16503181 109740608 179138862 276706356 236143101  40981798
       145        204        355        100        297         42         52         61        529        198       3220
 19358562 226165839  37270037  49104918 437804658  83032174 170460311 125120339 265077741 186212304  59588845
       470         37         38         32        152         25         68        210         69        140         29
```

Rest omitted.

```
> degree(g, "214328887")
214328887
      284
```

```
> hist(degree(g))
```



4. 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.

1. **igraph.version**: Query igraph's version string

```
> igraph.version()
[1] "1.1.1"
```

2. **igraph.options**: Parameters for the igraph package

```
> igraph.options()
$print.vertex.attributes
[1] FALSE

$print.edge.attributes
[1] FALSE

$print.graph.attributes
[1] FALSE

$verbose
[1] FALSE
```

3. **has.multiple:** Find the multiple or loop edges in a graph

```
> g = graph_from_data_frame(dataframe, directed = FALSE, vertices = NULL)
> g = simplify(g, remove.loops=TRUE)
> g = simplify(g, remove.multiple=TRUE)
> has.multiple(g)
[1] FALSE
```

4. **is.weighted:** Weighted graphs

```
> is.weighted(g)
[1] FALSE
```

5. **get.adjacency:** Convert a graph to an adjacency matrix

```
> get.adjacency(g)
76245 x 76245 sparse Matrix of class "dgCMatrix"
   [[ suppressing 35 column names '214328887', '17116707', '380580781' ... ]]
   [[ suppressing 35 column names '214328887', '17116707', '380580781' ... ]]

214328887 . 1 1 1 1 1 . 1 . 1 1 . 1 . 1 . 1 1 1 1 1 1 1 1 1 1 1 1 . 1 1 1 1 1 . ......
17116707  1 . . 1 1 . . 1 1 1 . . 1 . 1 . . . 1 1 1 . 1 1 1 1 . 1 . 1 . 1 1 1 . ......
380580781 1 . . . . 1 . 1 . . . . 1 . 1 . 1 . . 1 1 1 1 1 1 1 . . . 1 . 1 . 1 . ......
221036078 1 1 . . 1 1 . . . . . . 1 . 1 . . 1 1 1 1 1 1 1 1 1 . 1 . 1 . 1 1 1 . ......
107830991 1 1 . 1 . 1 . 1 . 1 1 . 1 . 1 . 1 1 1 1 1 1 1 1 1 1 1 1 . 1 . 1 1 1 . ......
151338729 1 . 1 1 1 . 1 1 1 1 . . 1 1 1 . 1 1 1 . 1 1 1 1 1 1 1 1 . 1 1 1 1 1 . ......
19705747  . . . . . 1 . 1 . 1 . . . . 1 . . 1 1 . . 1 1 1 1 . . . . 1 . 1 . 1 . ......
222261763 1 1 1 . 1 1 1 . . . 1 . 1 . 1 . . . . 1 1 . . 1 1 1 . 1 . 1 1 1 . 1 . ......
19933035  . 1 . . . 1 . . . . 1 . 1 . . . 1 1 1 1 . 1 . 1 . . . . . 1 . 1 1 . . ......
158419434 1 1 . . 1 1 1 . . . 1 1 1 1 . 1 1 . 1 1 1 . . 1 1 1 1 1 . 1 1 1 1 . . ......
```

Rest omitted.

6. **gsize:** The size of the graph (number of edges)

```
> gsize(g)
[1] 1242390
```

7. **is.directed:** Check whether a graph is directed

```
> is.directed(g)
[1] FALSE
```

8. **is.named:** Named graphs

```
> is.named(g)
[1] TRUE
```

9. **is.bipartite**: Create a bipartite graph

```
> is.bipartite(g)
[1] FALSE
```

10. **V**: Vertices of a graph

```
> V(g)
+ 76245/76245 vertices, named, from 41d3889:
   [1] 214328887 17116707  380580781 221036078 107830991 151338729 19705747  222261763
   [9] 19933035  158419434 149538028 364971269 100581193 113058991 406628822 460282402
  [17] 280935165 285312927 279787626 394263193 254839786 204317520 67864340  270449528
  [25] 153226312 124528830 220368467 206923844 207594668 451250774 6581292   34428380
  [33] 196327549 274153775 324201646 88097807  276308596 16870853  314316607 299715516
  [41] 276577539 107511013 258140947 123371682 461410856 117901353 56860418  43003845
  [49] 413275344 358775055 307458983 69592091  35148062  57490887  37699718  148519842
  [57] 225444667 283306479 220068522 155661154 400689940 72818790  195475105 157829215
  [65] 134940306 31414569  131613362 115221382 200559228 439788025 262340283 257236842
  [73] 463952369 254610699 226629405 529007327 70492333  166214735 236184723 86221475
+ ... omitted several vertices
```

5. Using igraph

5.1 central person

degree(graph, v = V(graph), mode = c("all", "out", "in", "total"), loops = TRUE, normalized = FALSE)

```
> degree <- degree(g)
> degree[order(degree,decreasing = TRUE)]
   813286 115485051  40981798   3359851  43003845  22462180  34428380  59804598   7861312     15913
     3687      3321      3316      3031      2831      2535      2513      2375      2167      2109
  5442012  11348282  17093617  10671602   1183041  18776017  48485771  18927441   7860742    972651
     1957      1848      1839      1818      1772      1726      1635      1627      1596      1573
```

we use the degree as the scores for valuing the central person. The one has more degree means that he/she has many connection with others.

5.2 longest path

diameter(graph, directed = TRUE, unconnected = TRUE, weights = NULL)

```
> lpath <- diameter(g)
> lpath
[1] 19
> ecount(g)
[1] 1667871
```

```
> farthest.nodes(g)
$vertices
+ 2/76245 vertices, named, from 82d4e5e:
[1] 398034217 216867579

$distance
[1] 19
```

The diameter of a graph is the length of the longest geodesic.

5.3 largest clique

max_cliques(graph, min = NULL, max = NULL, subset = NULL, file = NULL)

```
> max_cliq <- max_cliques(g, min = 3, max = 7, subset = NULL, file = NULL)
```

```
> max_cliq[545660]
[[1]]
+ 7/76245 vertices, named, from bed755f:
[1] 31353077  20999409  28192145  143950772 14448205  21888696  15342811
```

We use the data to calculate the size from 3 to 7 cliques.

5.4 ego

ego(graph, order, nodes = V(graph), mode = c("all", "out", "in"),mindist = 0)

```
> ego_of_g <- ego(g)
> ego_of_g
[[1]]
+ 9/76245 vertices, named, from 905c87b:
[1] 364350420 124563980 152011250 53811382  75829996  310687757 21068395  407259902 251925073

[[2]]
+ 284/76245 vertices, named, from 905c87b:
 [1] 194403468 195475105 196429355 360914929 100318079 187773078 210072500 394263193 263838766 148062502
 [ reached getOption("max.print") -- omitted 274 entries ]

[[3]]
+ 24/76245 vertices, named, from 905c87b:
 [1] 124563980 364350420 56222976  162157731 152011250 336218834 316736588 150319193 218640513 43003845
 [ reached getOption("max.print") -- omitted 14 entries ]

[[4]]
+ 681/76245 vertices, named, from 905c87b:
 [1] 195475105 194403468 175848601 74603580  137772184 100318079 187773078 154263215 394263193 263838766
 [ reached getOption("max.print") -- omitted 671 entries ]

[[5]]
+ 8/76245 vertices, named, from 905c87b:
[1] 165153568 336218834 196429355 43003845  40981798  34428380  22462180  21068395
```

5.5 betweenness centrality

betweenness(graph, v = V(graph), directed = TRUE, weights = NULL,nobigint = TRUE, normalized = FALSE)

edge_betweenness(graph, e = E(graph), directed = TRUE, weights = NULL)

```
> betweenness(g,V(g),TRUE,NULL,TRUE,FALSE)
   214328887     17116707    380580781    221036078    107830991    151338729     19705747    222261763
1.418609e+05 1.679505e+06 9.394674e+05 7.109781e+03 5.085454e+05 4.698463e+06 2.064799e+07 1.463076e+06
    19933035    158419434    149538028    364971269    100581193    113058991    406628822    460282402
4.799163e+02 5.867211e+06 1.538103e+06 4.246571e+04 6.884619e+05 9.173751e+05 5.538481e+04 8.273625e+03
   280935165    285312927    279787626    394263193    254839786    204317520     67864340    270449528
1.305387e+06 5.520781e+03 5.505987e+05 6.830677e+05 1.348626e+06 5.536123e+06 4.561647e+05 4.815134e+06
   153226312    124528830    220368467    206923844    207594668    451250774      6581292     34428380
9.946136e+06 1.746971e+05 2.836343e+05 1.927629e+06 2.641600e+05 5.098175e+04 1.974144e+06 1.803378e+07
   196327549    274153775    324201646     88097807    276308596     16870853    314316607    299715516
1.382159e+05 2.550898e+06 2.813125e-01 1.197973e+06 2.529651e+05 2.105520e+06 1.257589e+06 3.240398e+05
   276577539    107511013    258140947    123371682    461410856    117901353     56860418     43003845
5.541338e+04 5.487659e-01 1.294866e+06 1.324908e+06 1.753276e+06 3.791631e+05 1.812858e+06 3.963229e+07
   413275344    358775055    307458983     69592091     35148062     57490887     37699718    148519842
1.029767e+02 2.770120e+05 2.386040e+06 1.264479e+06 1.635377e+03 9.488274e+04 1.746394e+06 7.886600e+05
   225444667    283306479    220068522    155661154    400689940     72818790    195475105    157829215
6.236555e+04 2.951097e+05 7.120156e+04 3.129667e+05 3.081042e+03 1.928557e+05 1.341908e+07 1.650241e+06
   134940306     31414569    131613362    115221382    200559228    439788025    262340283    257236842
3.367300e+05 1.297172e+06 1.278952e+05 6.424102e+04 2.353232e+06 7.620058e+05 8.636531e+05 2.093134e+05
   463952369    254610699    226629405    529007327     70492333    166214735    236184723     86221475
2.299635e+05 1.787728e+05 1.034153e+06 1.120222e+05 3.441996e+06 9.476599e+04 3.044685e+04 3.190241e+05
   116498875    222411742     17675120    250178329    103598216    172883064    292598082    145845459
9.858149e+05 9.990352e+02 6.914143e+06 4.340658e+02 5.706000e+05 2.034857e+06 3.308082e+04 6.790172e+04
   294752666    163238842     29911100    197504076    248883350    260769396    288485704    229425177
1.471832e+05 1.894176e+05 2.074917e+04 4.438545e+06 2.216085e+05 1.045568e+05 7.234130e+05 2.150604e+05
   252770012     26929220    259842341    262802533    213777144     46209291    239144776      8163442
1.439231e+04 6.052954e+06 2.959056e+06 1.106728e+04 1.837342e+06 3.873659e+07 1.413231e+05 5.934417e+06
    16503181    109740608    179138862    276706356    236143101     40981798     19358562    226165839
1.553939e+05 8.664354e+02 1.287893e+06 3.874111e+06 1.825722e+05 8.144336e+07 4.284517e+06 2.098445e+05
    37270037     49104918    437804658     83032174    170460311    125120339    265077741    186212304
7.410208e+04 4.630541e+04 4.654830e+04 1.404433e+06 2.064355e+05 1.141755e+06 1.912243e+03 3.736115e+06
    59588845    203338499     74107696    238201669    477094958    247741328     83417972    233248636
3.933613e+04 2.202682e+06 7.802578e+01 5.610323e+05 3.334178e+02 2.575020e+05 1.484650e+06 6.054856e+04
    86799233     35359596     17627996     30971165     81446304    187773078      2367911    447688115
```

5.6 power centrality

```
pc <- power_centrality(g,nodes = c(17116707), exp = 0.1,loops = FALSE, rescale = FALSE)
```

```
> pc
[1] 1.1752600 0.7241297 0.0413450 0.1238120 0.8271002 1.9172612
```

power_centrality takes a graph (dat) and returns the Boncich power centralities of positions (selected by nodes). The decay rate for power contributions is specified by exponent (1 by default).