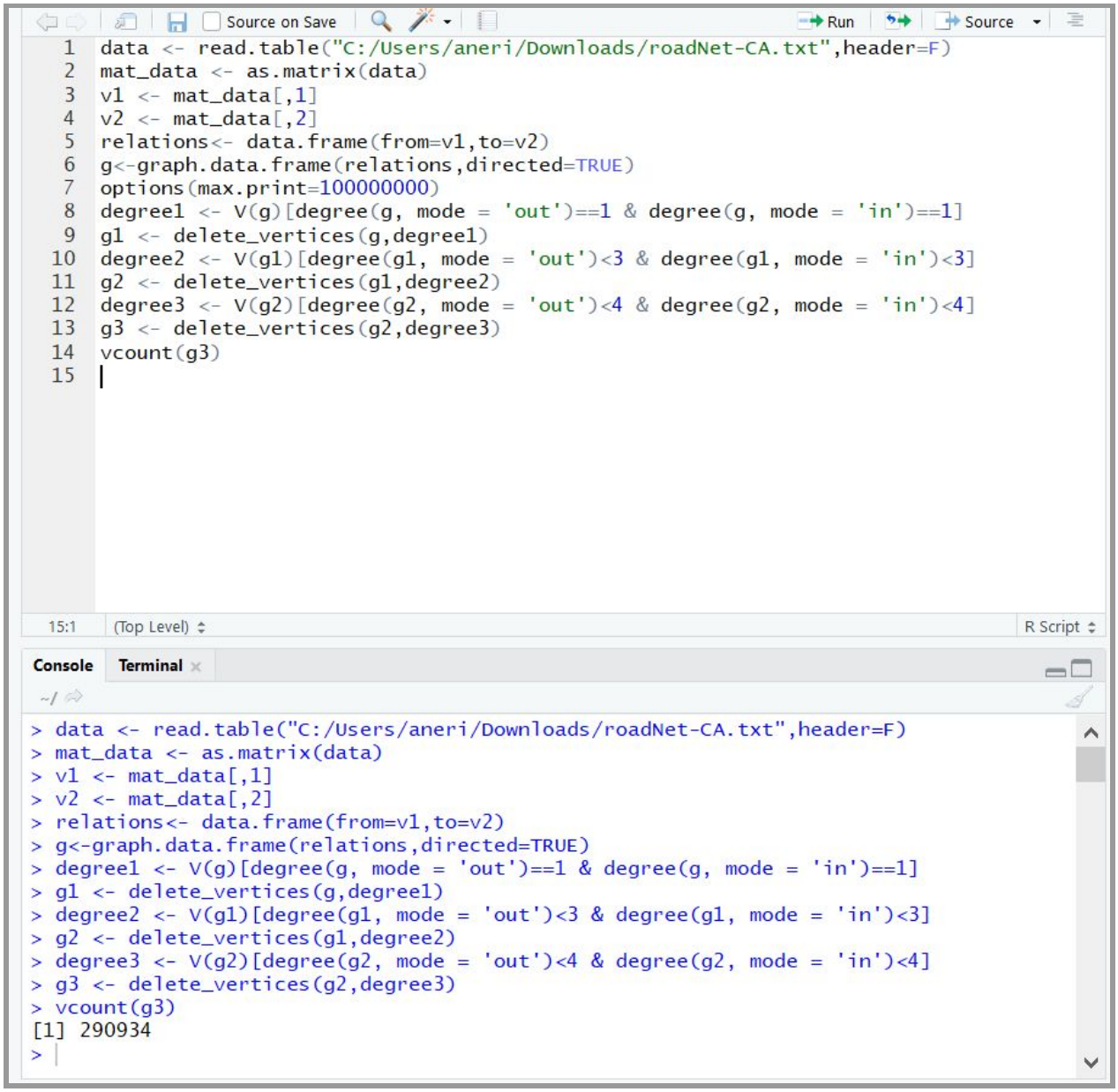
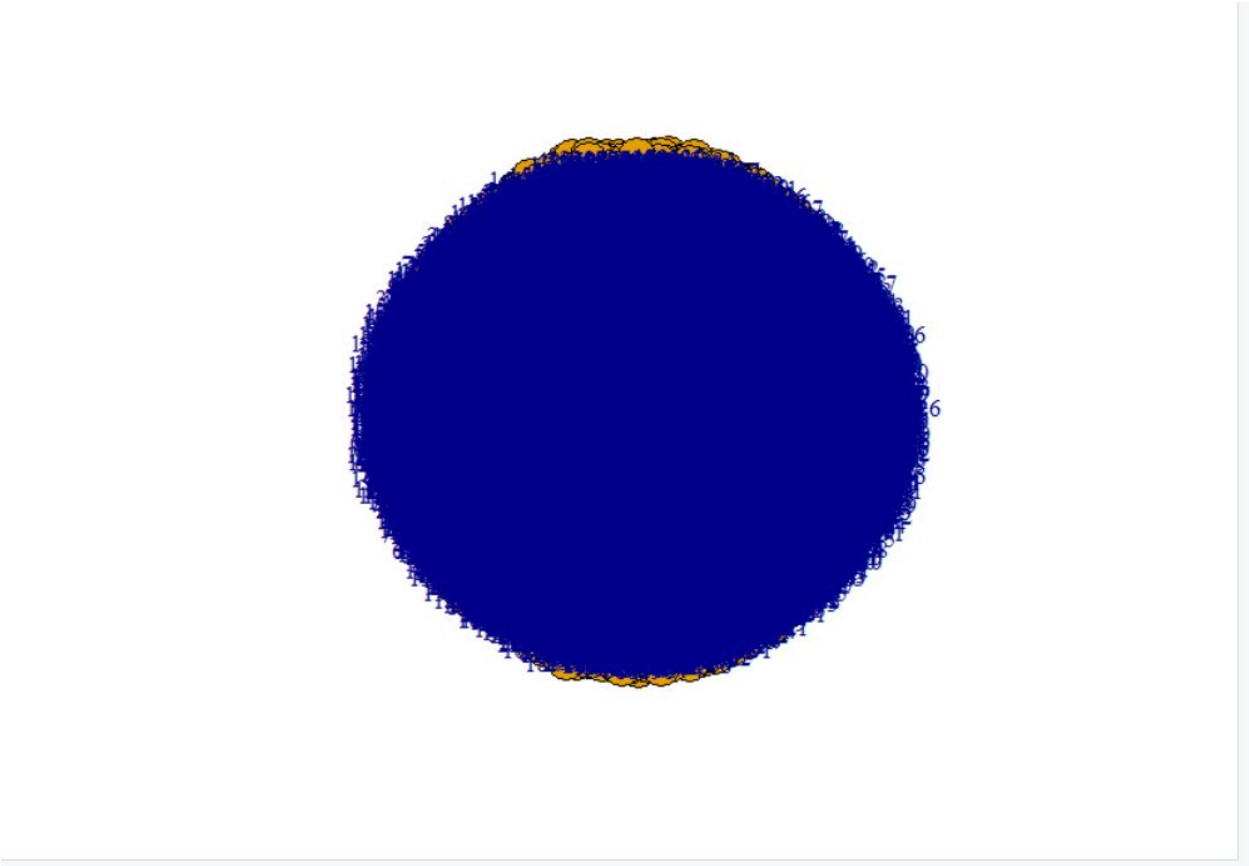
Class project #1
R and Graph Analytics

```
data <- read.table("C:/Users/aneri/Downloads/roadNet-CA.txt",header=F) #reads the files from the destination
mat_data <- as.matrix(data) #mat_data stores the data as matrix
v1 <- mat_data[,1]
v2 <- mat_data[,2]
relations<- data.frame(from=v1,to=v2) #relations has the dataframe from vertices 1 to 2
g<-graph.data.frame(relations,directed=TRUE) #converts the data frame into the graph
options(max.print=100000000) #for printing maximum
degree1 <- V(g)[degree(g, mode = 'out')==1 & degree(g, mode = 'in')==1] #stores the in and out degrees which are 1
g1 <- delete_vertices(g,degree1) #deletes the degree 1 vertices
degree2 <- V(g1)[degree(g1, mode = 'out')<3 & degree(g1, mode = 'in')<3]  #stores the in and out degrees which are less than 3
g2 <- delete_vertices(g1,degree2)  #deletes the  vertices less than 3
degree3 <- V(g2)[degree(g2, mode = 'out')<4 & degree(g2, mode = 'in')<4] #stores the in and out degrees which are less than 4
g3 <- delete_vertices(g2,degree3)  #deletes the  vertices less than 4
vcount(g3)
```

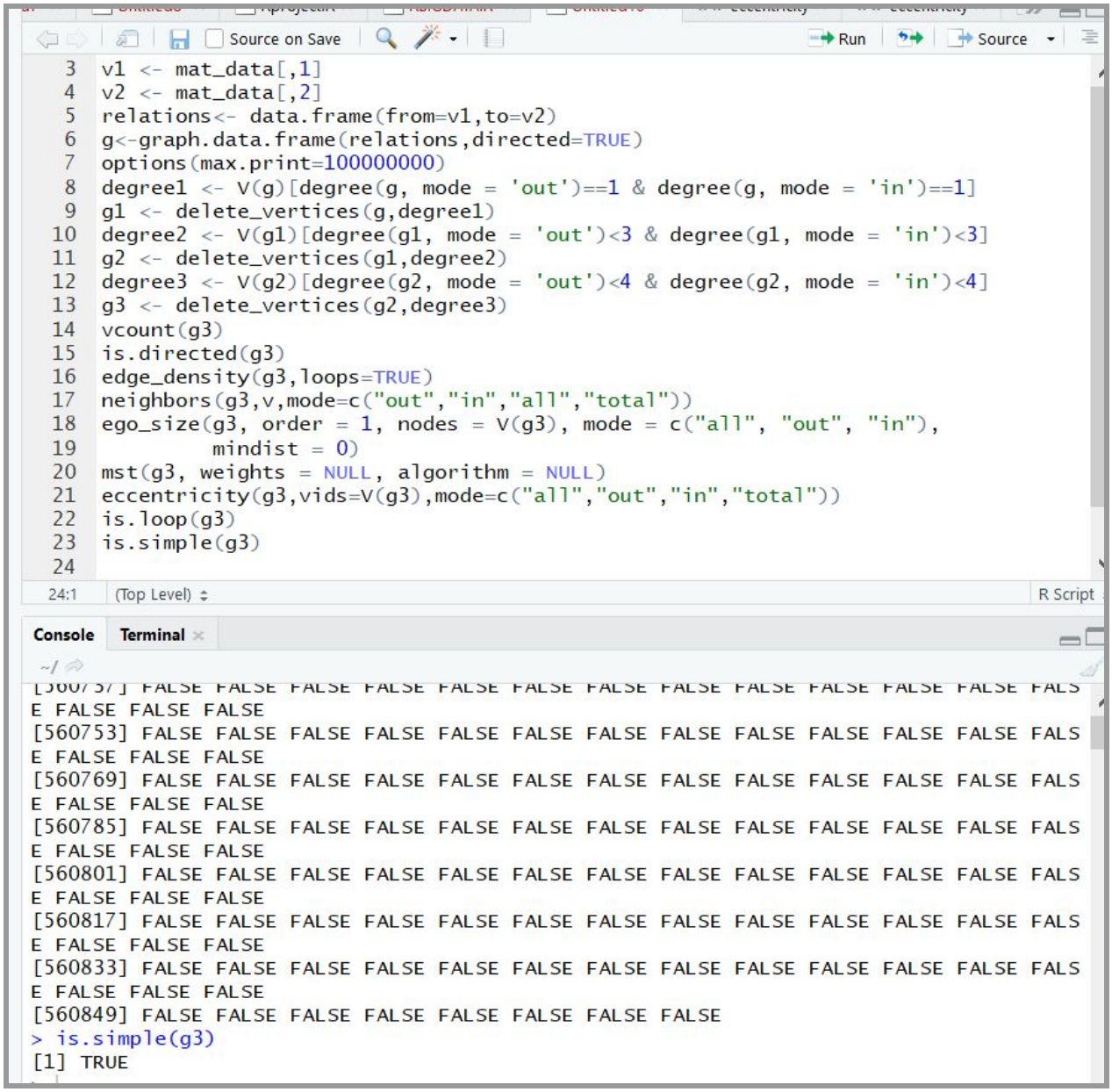

Source on Save
Run
Source
1 data <- read.table("C:/Users/aneri/Downloads/roadNet-CA.txt",header=F)
2 mat_data <- as.matrix(data)
3 v1 <- mat_data[,1]
4 v2 <- mat_data[,2]
5 relations<- data.frame(from=v1,to=v2)
6 g<-graph.data.frame(relations,directed=TRUE)
7 options(max.print=100000000)
8 degree1 <- V(g)[degree(g, mode = 'out')==1 & degree(g, mode = 'in')==1]
9 g1 <- delete_vertices(g,degree1)
10 degree2 <- V(g1)[degree(g1, mode = 'out')<3 & degree(g1, mode = 'in')<3]
11 g2 <- delete_vertices(g1,degree2)
12 degree3 <- V(g2)[degree(g2, mode = 'out')<4 & degree(g2, mode = 'in')<4]
13 g3 <- delete_vertices(g2,degree3)
14 vcount(g3)
15
15:1
(Top Level)
R Script
Console
Terminal
~/
> data <- read.table("C:/Users/aneri/Downloads/roadNet-CA.txt",header=F)
> mat_data <- as.matrix(data)
> v1 <- mat_data[,1]
> v2 <- mat_data[,2]
> relations<- data.frame(from=v1,to=v2)
> g<-graph.data.frame(relations,directed=TRUE)
> degree1 <- V(g)[degree(g, mode = 'out')==1 & degree(g, mode = 'in')==1]
> g1 <- delete_vertices(g,degree1)
> degree2 <- V(g1)[degree(g1, mode = 'out')<3 & degree(g1, mode = 'in')<3]
> g2 <- delete_vertices(g1,degree2)
> degree3 <- V(g2)[degree(g2, mode = 'out')<4 & degree(g2, mode = 'in')<4]
> g3 <- delete_vertices(g2,degree3)
> vcount(g3)
[1] 290934
>

plot(g3)



4. Experiment with at least 10 of the functions that I have shown in the lecture notes and associated PPT file on Blackboard.

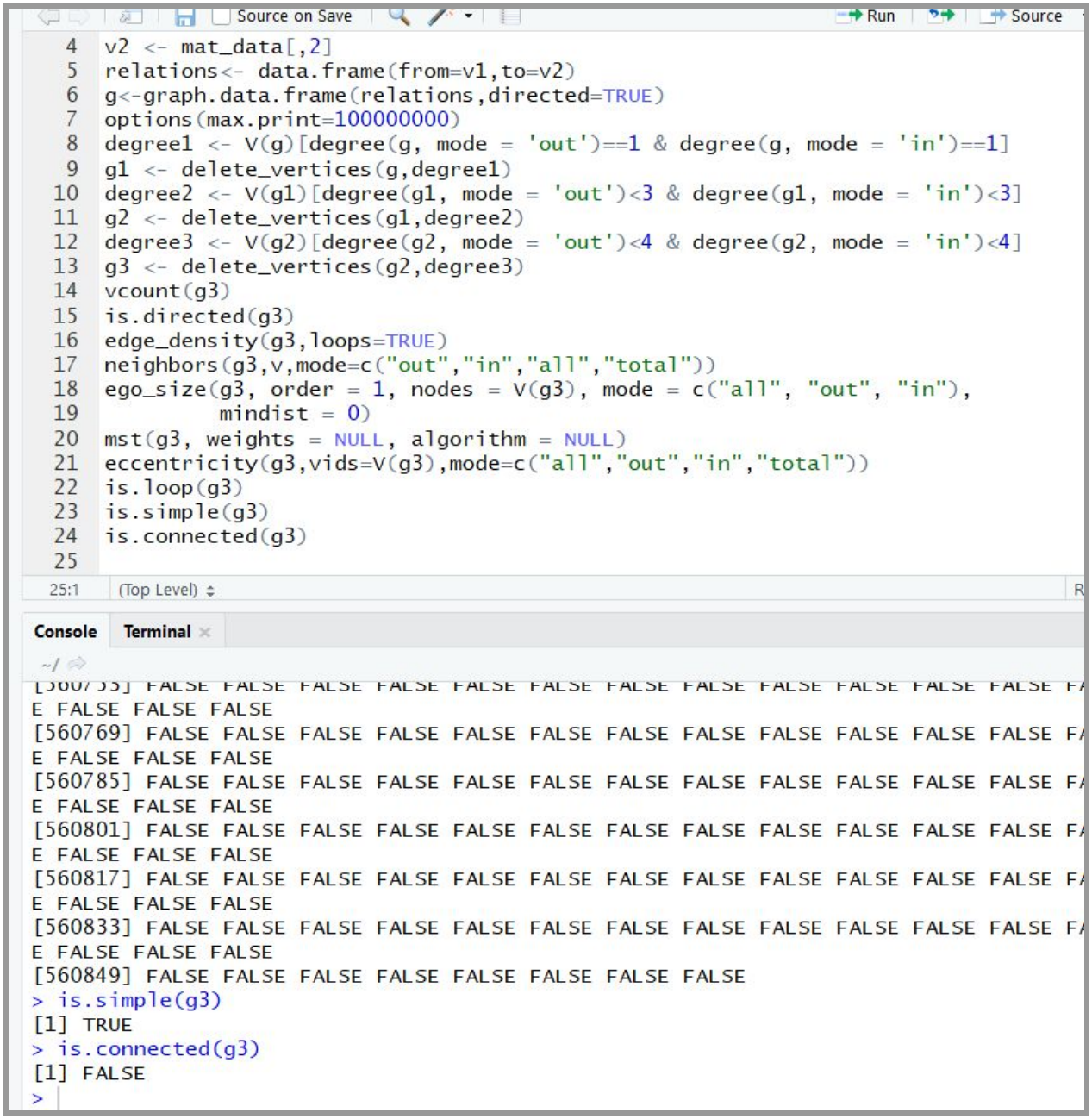
1) is.simple(g3)



```
v1 <- mat_data[,1]
v2 <- mat_data[,2]
relations<- data.frame(from=v1,to=v2)
g<-graph.data.frame(relations,directed=TRUE)
options(max.print=100000000)
degree1 <- V(g)[degree(g, mode = 'out')==1 & degree(g, mode = 'in')==1]
g1 <- delete_vertices(g,degree1)
degree2 <- V(g1)[degree(g1, mode = 'out')<3 & degree(g1, mode = 'in')<3]
g2 <- delete_vertices(g1,degree2)
degree3 <- V(g2)[degree(g2, mode = 'out')<4 & degree(g2, mode = 'in')<4]
g3 <- delete_vertices(g2,degree3)
vcount(g3)
is.directed(g3)
edge_density(g3,loops=TRUE)
neighbors(g3,v,mode=c("out","in","all","total"))
ego_size(g3, order = 1, nodes = V(g3), mode = c("all", "out", "in"),
         mindist = 0)
mst(g3, weights = NULL, algorithm = NULL)
eccentricity(g3,vids=V(g3),mode=c("all","out","in","total"))
is.loop(g3)
is.simple(g3)
```

```
E FALSE FALSE FALSE
[560753] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALS
E FALSE FALSE FALSE
[560769] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALS
E FALSE FALSE FALSE
[560785] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALS
E FALSE FALSE FALSE
[560801] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALS
E FALSE FALSE FALSE
[560817] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALS
E FALSE FALSE FALSE
[560833] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALS
E FALSE FALSE FALSE
[560849] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
> is.simple(g3)
[1] TRUE
```

2) is.connected(g3)

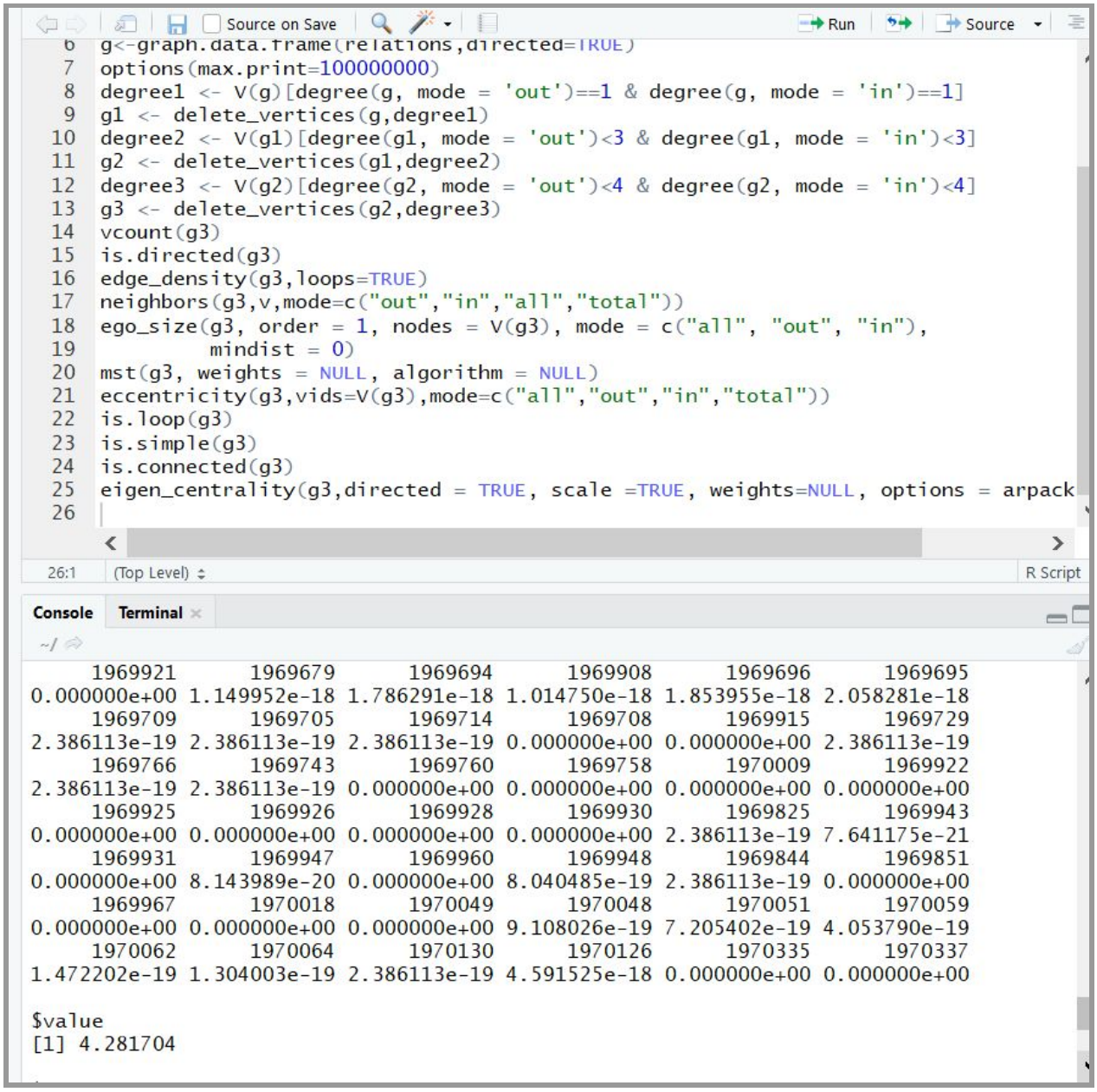


```
v2 <- mat_data[,2]
relations<- data.frame(from=v1,to=v2)
g<-graph.data.frame(relations,directed=TRUE)
options(max.print=100000000)
degree1 <- V(g)[degree(g, mode = 'out')==1 & degree(g, mode = 'in')==1]
g1 <- delete_vertices(g,degree1)
degree2 <- V(g1)[degree(g1, mode = 'out')<3 & degree(g1, mode = 'in')<3]
g2 <- delete_vertices(g1,degree2)
degree3 <- V(g2)[degree(g2, mode = 'out')<4 & degree(g2, mode = 'in')<4]
g3 <- delete_vertices(g2,degree3)
vcount(g3)
is.directed(g3)
edge_density(g3,loops=TRUE)
neighbors(g3,v,mode=c("out","in","all","total"))
ego_size(g3, order = 1, nodes = V(g3), mode = c("all", "out", "in"),
         mindist = 0)
mst(g3, weights = NULL, algorithm = NULL)
eccentricity(g3,vids=V(g3),mode=c("all","out","in","total"))
is.loop(g3)
is.simple(g3)
is.connected(g3)
```

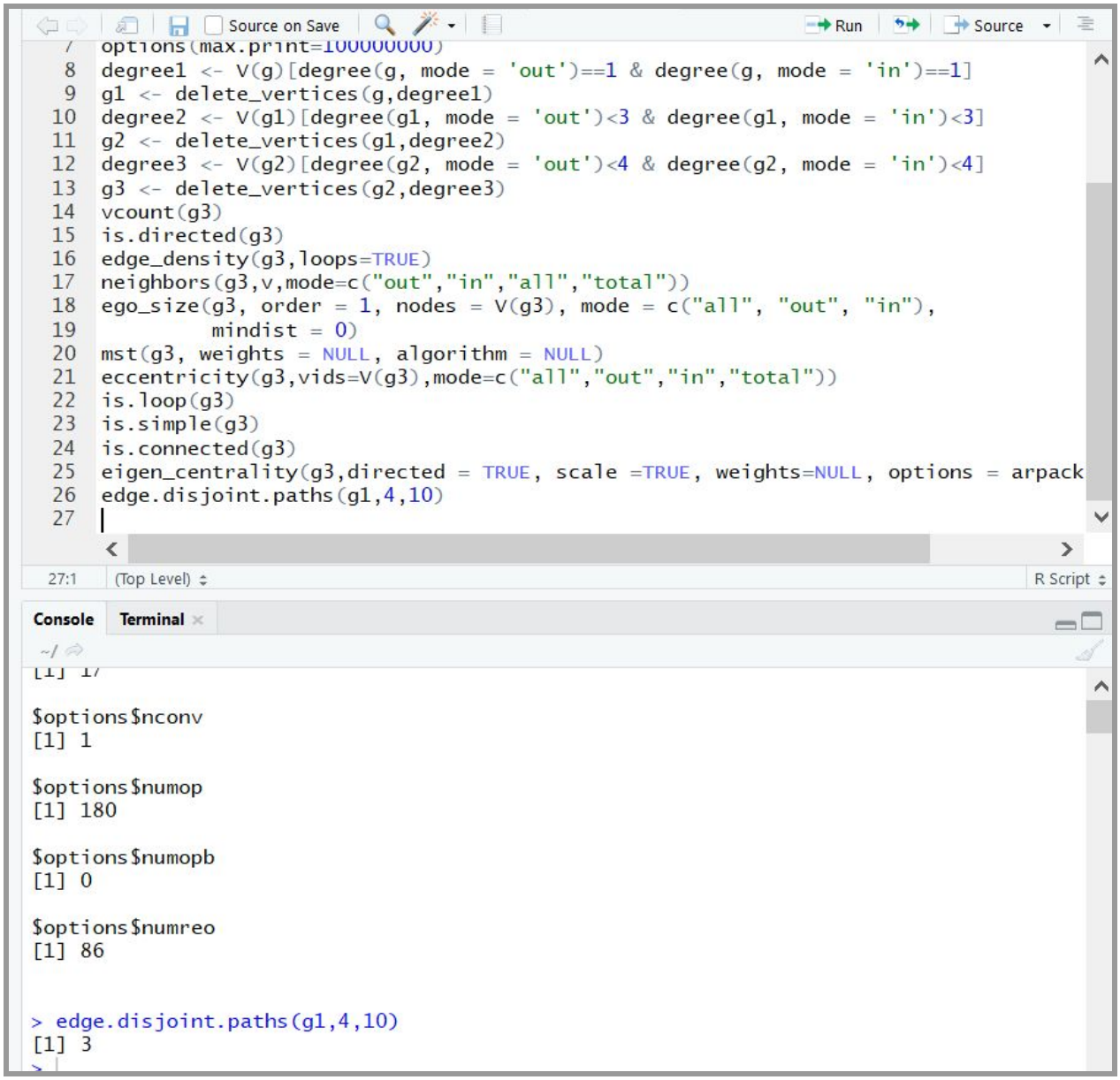
```
E FALSE FALSE FALSE
[560769] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
E FALSE FALSE FALSE
[560785] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
E FALSE FALSE FALSE
[560801] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
E FALSE FALSE FALSE
[560817] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
E FALSE FALSE FALSE
[560833] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
E FALSE FALSE FALSE
[560849] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
> is.simple(g3)
[1] TRUE
> is.connected(g3)
[1] FALSE
>
```

3) Eigen Centrality

eigen_centrality(g3,directed = TRUE, scale =TRUE, weights=NULL, options = arpack_defaults)



4) edge.disjoint.paths(g1,4,10)



5) Pagerank:

page_rank(g3)

```
23  is.simple(g3)
24  is.connected(g3)
25  eigen_centrality(g3,directed = TRUE, scale =TRUE, weights=NULL, options = a
26  edge.disjoint.paths(g1,4,10)
27  matrix.df <- as.data.frame(mat_data)
28
29  degree4 <- V(g3)[ degree(g3, mode = 'in')==8]
30  max(degree4)
31  count(degree4)
32  head(degree4)
33  degree(g3, v = V(g3), (mode = "out" )==8)
34
35  V(g3)$name[degree(g3)==max(degree(g3))]
36  page_rank(g3)
37  pagerank <- page_rank(g3)
38  head(pagerank)
39
```

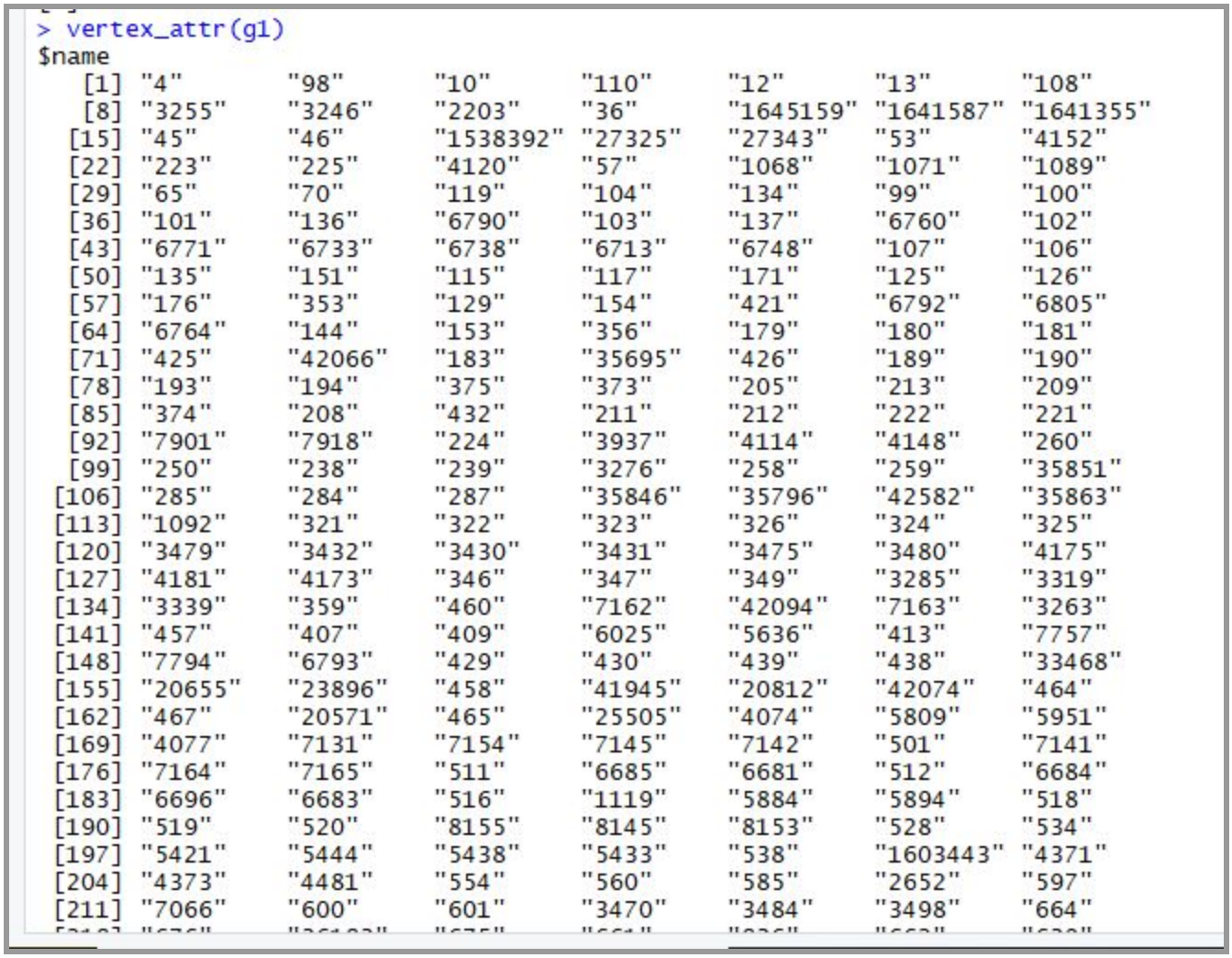
36:1 (Top Level) R Script

Console Terminal

```
3.901383e-06 3.909639e-06 5.333069e-06 2.037176e-06 3.061510e-06 5.829358e-07
     1969921      1969679      1969694      1969908      1969696      1969695
2.993454e-06 4.770203e-06 2.551472e-06 2.610272e-06 6.947775e-06 2.551472e-06
     1969709      1969705      1969714      1969708      1969915      1969729
5.829358e-07 5.829358e-07 5.829358e-07 2.993454e-06 5.671808e-06 5.829358e-07
     1969766      1969743      1969760      1969758      1970009      1969922
5.829358e-07 5.829358e-07 2.993454e-06 5.671808e-06 2.993454e-06 2.993454e-06
     1969925      1969926      1969928      1969930      1969825      1969943
5.671808e-06 2.993454e-06 3.686395e-06 5.351880e-06 5.829358e-07 3.994748e-06
     1969931      1969947      1969960      1969948      1969844      1969851
5.182253e-06 2.482604e-06 3.886239e-06 4.469807e-06 5.829358e-07 5.506860e-06
     1969967      1970018      1970049      1970048      1970051      1970059
3.727326e-06 2.099302e-06 2.421427e-06 4.325862e-06 3.964031e-06 3.629655e-06
     1970062      1970064      1970130      1970126      1970335      1970337
3.886239e-06 3.886239e-06 5.829358e-07 2.984718e-06 3.886239e-06 3.886239e-06

$value
[1] 1
```

6) Vertex Attribute:

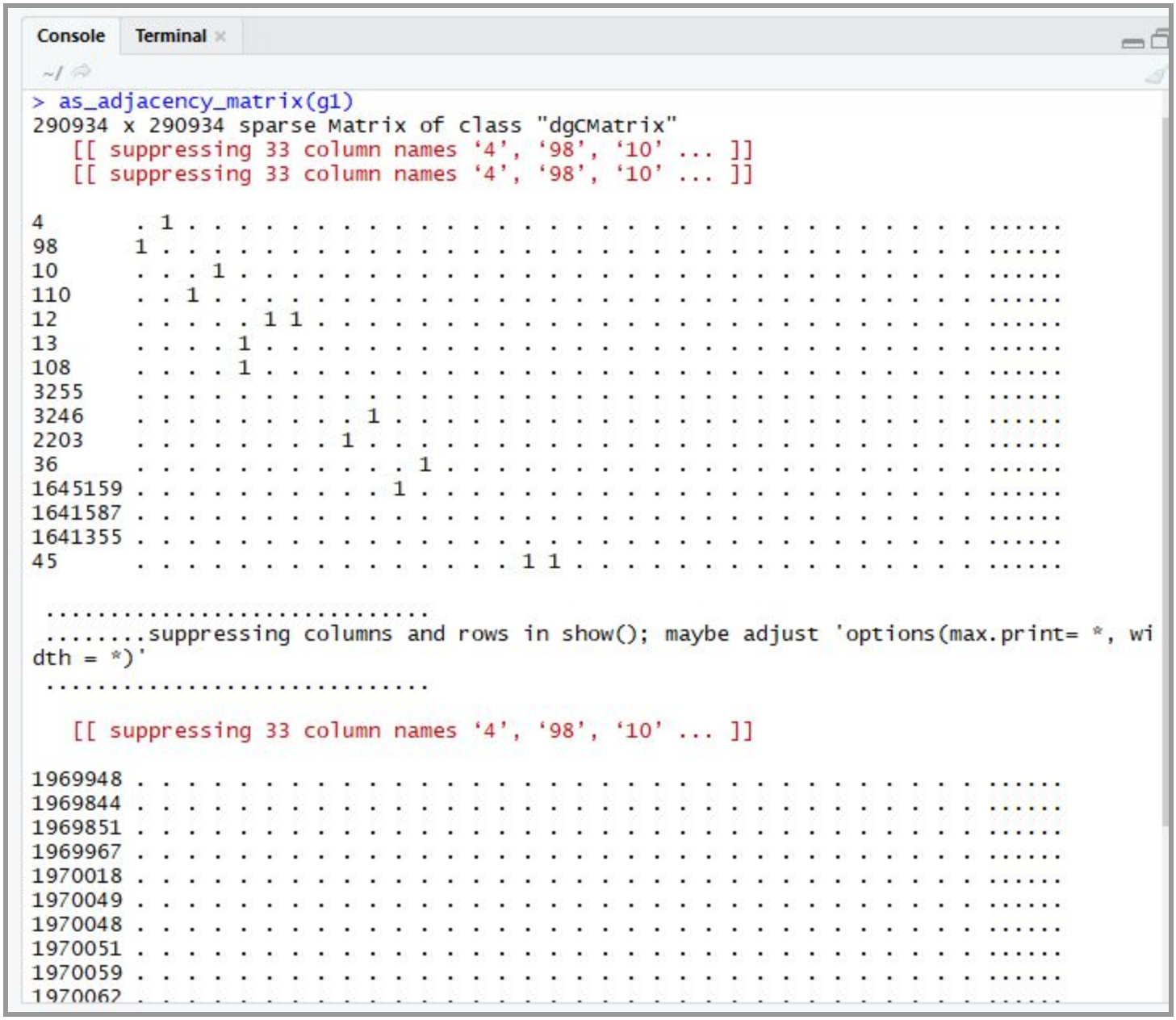
vertex_attr(g3)



```
> vertex_attr(g1)
$name
   [1] "4"        "98"       "10"       "110"      "12"       "13"       "108"
   [8] "3255"     "3246"     "2203"     "36"       "1645159"  "1641587"  "1641355"
  [15] "45"       "46"       "1538392"  "27325"    "27343"    "53"       "4152"
  [22] "223"      "225"      "4120"     "57"       "1068"     "1071"     "1089"
  [29] "65"       "70"       "119"      "104"      "134"      "99"       "100"
  [36] "101"      "136"      "6790"     "103"      "137"      "6760"     "102"
  [43] "6771"     "6733"     "6738"     "6713"     "6748"     "107"      "106"
  [50] "135"      "151"      "115"      "117"      "171"      "125"      "126"
  [57] "176"      "353"      "129"      "154"      "421"      "6792"     "6805"
  [64] "6764"     "144"      "153"      "356"      "179"      "180"      "181"
  [71] "425"      "42066"    "183"      "35695"    "426"      "189"      "190"
  [78] "193"      "194"      "375"      "373"      "205"      "213"      "209"
  [85] "374"      "208"      "432"      "211"      "212"      "222"      "221"
  [92] "7901"     "7918"     "224"      "3937"     "4114"     "4148"     "260"
  [99] "250"      "238"      "239"      "3276"     "258"      "259"      "35851"
 [106] "285"      "284"      "287"      "35846"    "35796"    "42582"    "35863"
 [113] "1092"     "321"      "322"      "323"      "326"      "324"      "325"
 [120] "3479"     "3432"     "3430"     "3431"     "3475"     "3480"     "4175"
 [127] "4181"     "4173"     "346"      "347"      "349"      "3285"     "3319"
 [134] "3339"     "359"      "460"      "7162"     "42094"    "7163"     "3263"
 [141] "457"      "407"      "409"      "6025"     "5636"     "413"      "7757"
 [148] "7794"     "6793"     "429"      "430"      "439"      "438"      "33468"
 [155] "20655"    "23896"    "458"      "41945"    "20812"    "42074"    "464"
 [162] "467"      "20571"    "465"      "25505"    "4074"     "5809"     "5951"
 [169] "4077"     "7131"     "7154"     "7145"     "7142"     "501"      "7141"
 [176] "7164"     "7165"     "511"      "6685"     "6681"     "512"      "6684"
 [183] "6696"     "6683"     "516"      "1119"     "5884"     "5894"     "518"
 [190] "519"      "520"      "8155"     "8145"     "8153"     "528"      "534"
 [197] "5421"     "5444"     "5438"     "5433"     "538"      "1603443"  "4371"
 [204] "4373"     "4481"     "554"      "560"      "585"      "2652"     "597"
 [211] "7066"     "600"      "601"      "3470"     "3484"     "3498"     "664"
```

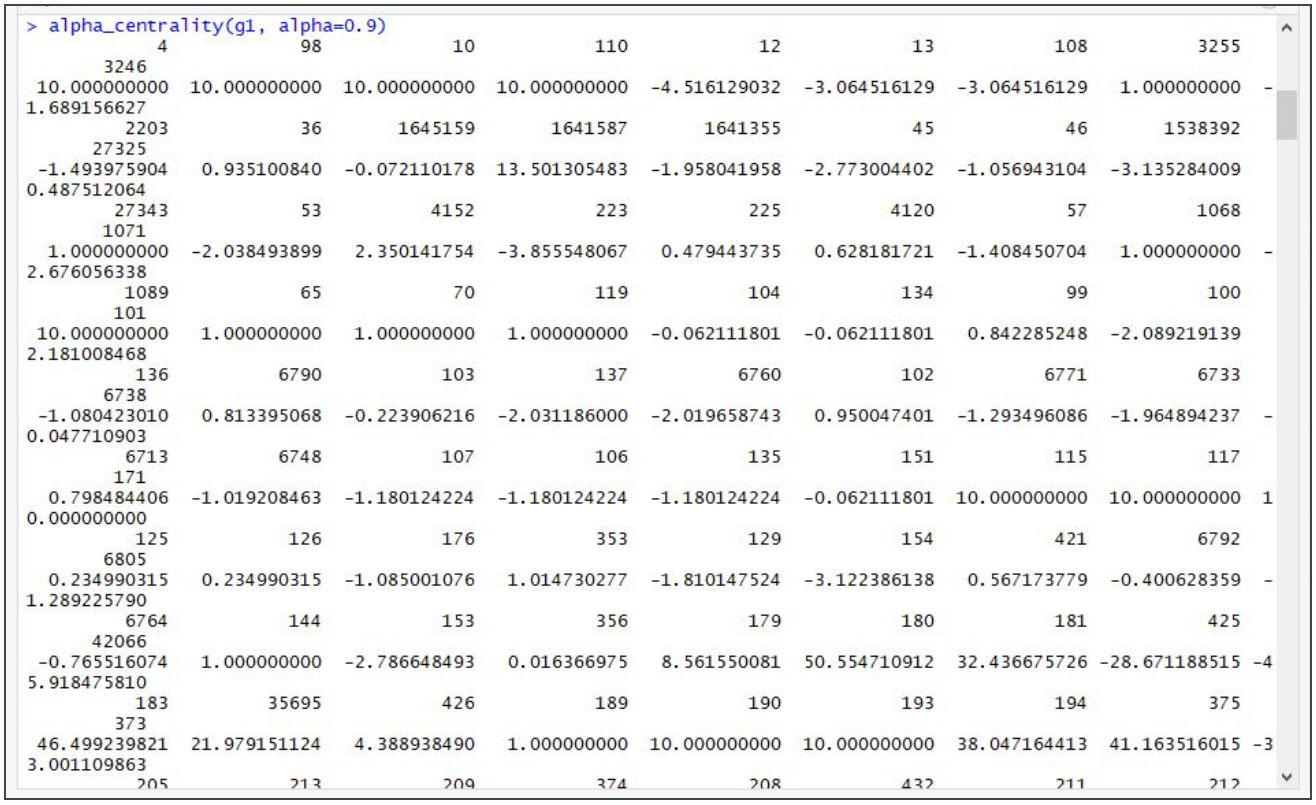
7) Adjacency Matrix:

as_adjacency_matrix(g3)



8) Alpha centrality:

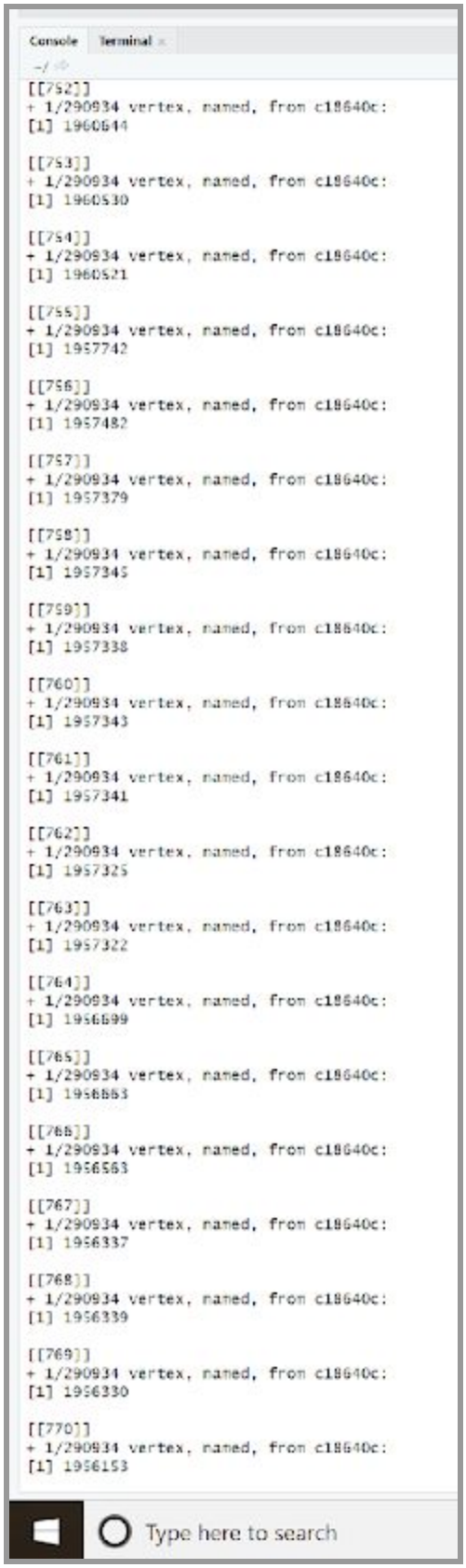
alpha_centrality(g3,alpha=0.9)



```
> alpha_centrality(g1, alpha=0.9)
          4          98          10         110          12          13         108        3255
        3246
 10.000000000 10.000000000 10.000000000 10.000000000 -4.516129032 -3.064516129 -3.064516129  1.000000000 -
1.689156627
        2203          36     1645159     1641587     1641355          45          46     1538392
       27325
 -1.493975904  0.935100840 -0.072110178 13.501305483 -1.958041958 -2.773004402 -1.056943104 -3.135284009
0.487512064
       27343          53        4152         223         225        4120          57        1068
        1071
  1.000000000 -2.038493899  2.350141754 -3.855548067  0.479443735  0.628181721 -1.408450704  1.000000000 -
2.676056338
        1089          65          70         119         104         134          99         100
         101
 10.000000000  1.000000000  1.000000000  1.000000000 -0.062111801 -0.062111801  0.842285248 -2.089219139
2.181008468
         136        6790         103         137        6760         102        6771        6733
        6738
 -1.080423010  0.813395068 -0.223906216 -2.031186000 -2.019658743  0.950047401 -1.293496086 -1.964894237 -
0.047710903
        6713        6748         107         106         135         151         115         117
         171
  0.798484406 -1.019208463 -1.180124224 -1.180124224 -1.180124224 -0.062111801 10.000000000 10.000000000  1
0.000000000
         125         126         176         353         129         154         421        6792
        6805
  0.234990315  0.234990315 -1.085001076  1.014730277 -1.810147524 -3.122386138  0.567173779 -0.400628359 -
1.289225790
        6764         144         153         356         179         180         181         425
       42066
 -0.765516074  1.000000000 -2.786648493  0.016366975  8.561550081 50.554710912 32.436675726 -28.671188515 -4
5.918475810
         183       35695         426         189         190         193         194         375
         373
 46.499239821 21.979151124  4.388938490  1.000000000 10.000000000 10.000000000 38.047164413 41.163516015 -3
3.001109863
         205         213         209         374         208         432         211         212
```

9) Cliques:

cliques(g3)


Console
Terminal
~/
[[752]]
+ 1/290934 vertex, named, from c18640c:
[1] 1960644
[[753]]
+ 1/290934 vertex, named, from c18640c:
[1] 1960530
[[754]]
+ 1/290934 vertex, named, from c18640c:
[1] 1960521
[[755]]
+ 1/290934 vertex, named, from c18640c:
[1] 1957742
[[756]]
+ 1/290934 vertex, named, from c18640c:
[1] 1957482
[[757]]
+ 1/290934 vertex, named, from c18640c:
[1] 1957379
[[758]]
+ 1/290934 vertex, named, from c18640c:
[1] 1957345
[[759]]
+ 1/290934 vertex, named, from c18640c:
[1] 1957338
[[760]]
+ 1/290934 vertex, named, from c18640c:
[1] 1957343
[[761]]
+ 1/290934 vertex, named, from c18640c:
[1] 1957341
[[762]]
+ 1/290934 vertex, named, from c18640c:
[1] 1957325
[[763]]
+ 1/290934 vertex, named, from c18640c:
[1] 1957322
[[764]]
+ 1/290934 vertex, named, from c18640c:
[1] 1956699
[[765]]
+ 1/290934 vertex, named, from c18640c:
[1] 1956863
[[766]]
+ 1/290934 vertex, named, from c18640c:
[1] 1956563
[[767]]
+ 1/290934 vertex, named, from c18640c:
[1] 1956337
[[768]]
+ 1/290934 vertex, named, from c18640c:
[1] 1956339
[[769]]
+ 1/290934 vertex, named, from c18640c:
[1] 1956330
[[770]]
+ 1/290934 vertex, named, from c18640c:
[1] 1956153
Type here to search

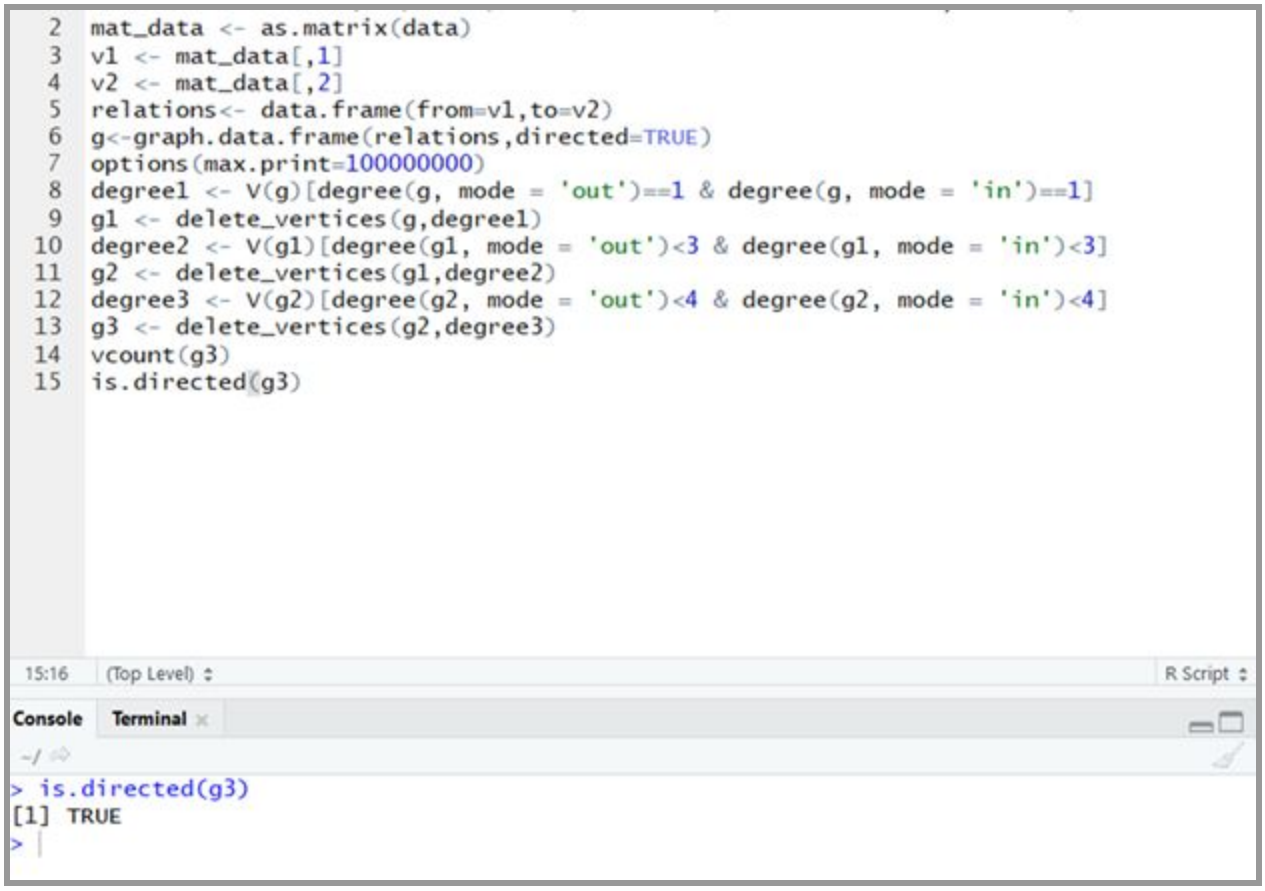
10) Weight

E(g1)$weight<-rnorm(ecount(g1))
V(g1)$weight<-rnorm(vcount(g1))
g1

```
> E(g1)$weight<-rnorm(ecount(g1))
> V(g1)$weight<-rnorm(vcount(g1))
> g1
IGRAPH c40d13a DNW- 290934 560856 --
+ attr: name (v/c), weight (v/n), weight (e/n)
+ edges from c40d13a (vertex names):
 [1] 4       ->98       98      ->4        10      ->110      110     ->10       12      ->13
 [6] 12      ->108      13      ->12       108     ->12       3246    ->2203     3246    ->3257
[11] 2203    ->2146     2203    ->3246     36      ->1645159  1645159->36       1645159->16451
56
[16] 1645159->1645157 1645159->1648644 1641587->1633418 1641587->1641606 1641355->16289
54
[21] 1641355->1639780 45      ->46       45      ->1538392  46      ->45       46      ->27325
[26] 1538392->45      1538392->1538391 27325   ->46       27325   ->27335    53      ->223
[31] 53      ->225      4152    ->4148     4152    ->4153     4152    ->4156     223     ->53
[36] 223     ->224      223     ->3937     225     ->53       225     ->224      225     ->4114
```

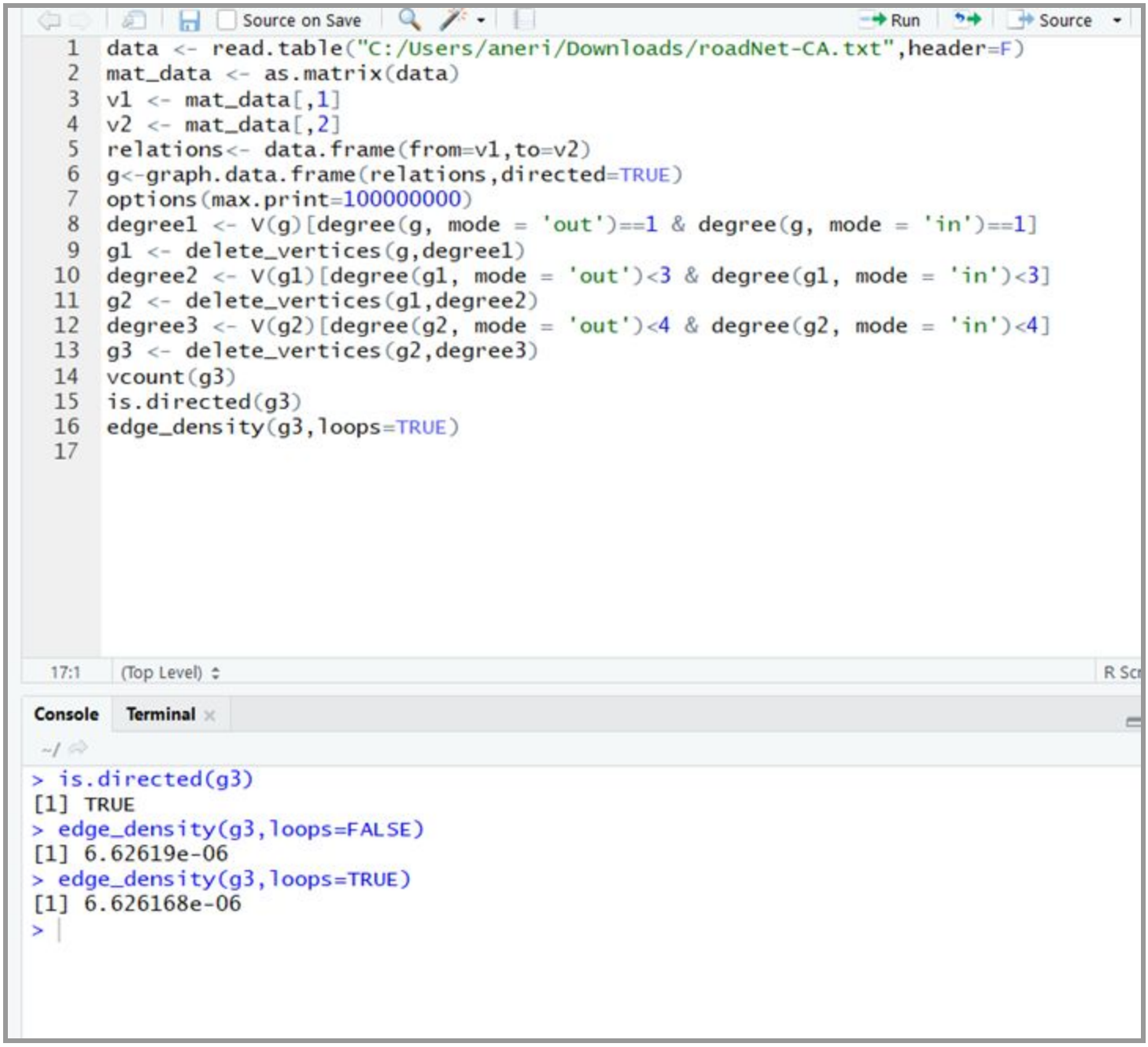
5. Explore other functions in the igraph package – at least 15 of them not shown in the lecture notes.

1) is.directed(g3)



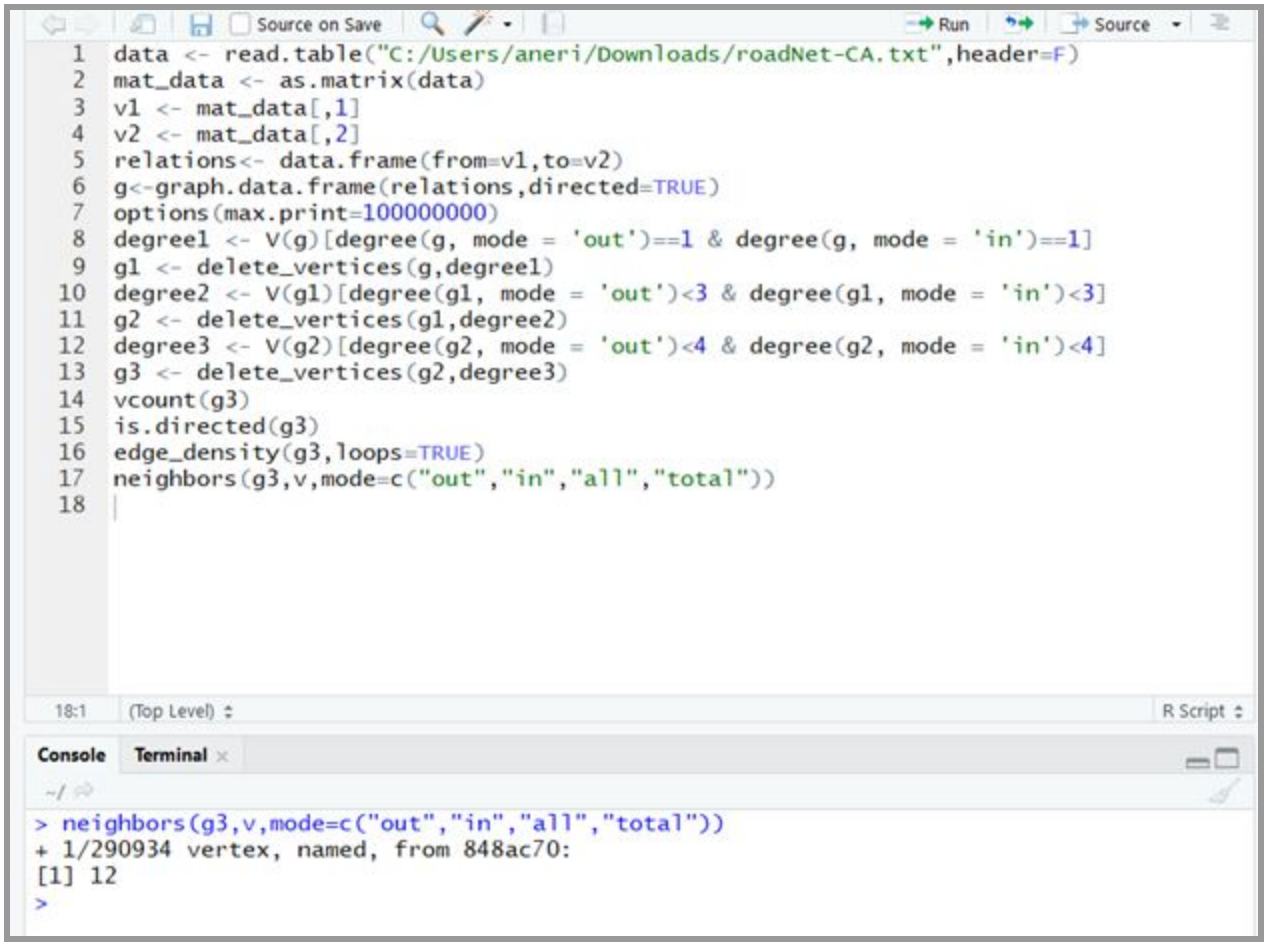
2) Edge_density

edge_density(g3,loops=FALSE)
edge_density(g3,loops=TRUE))



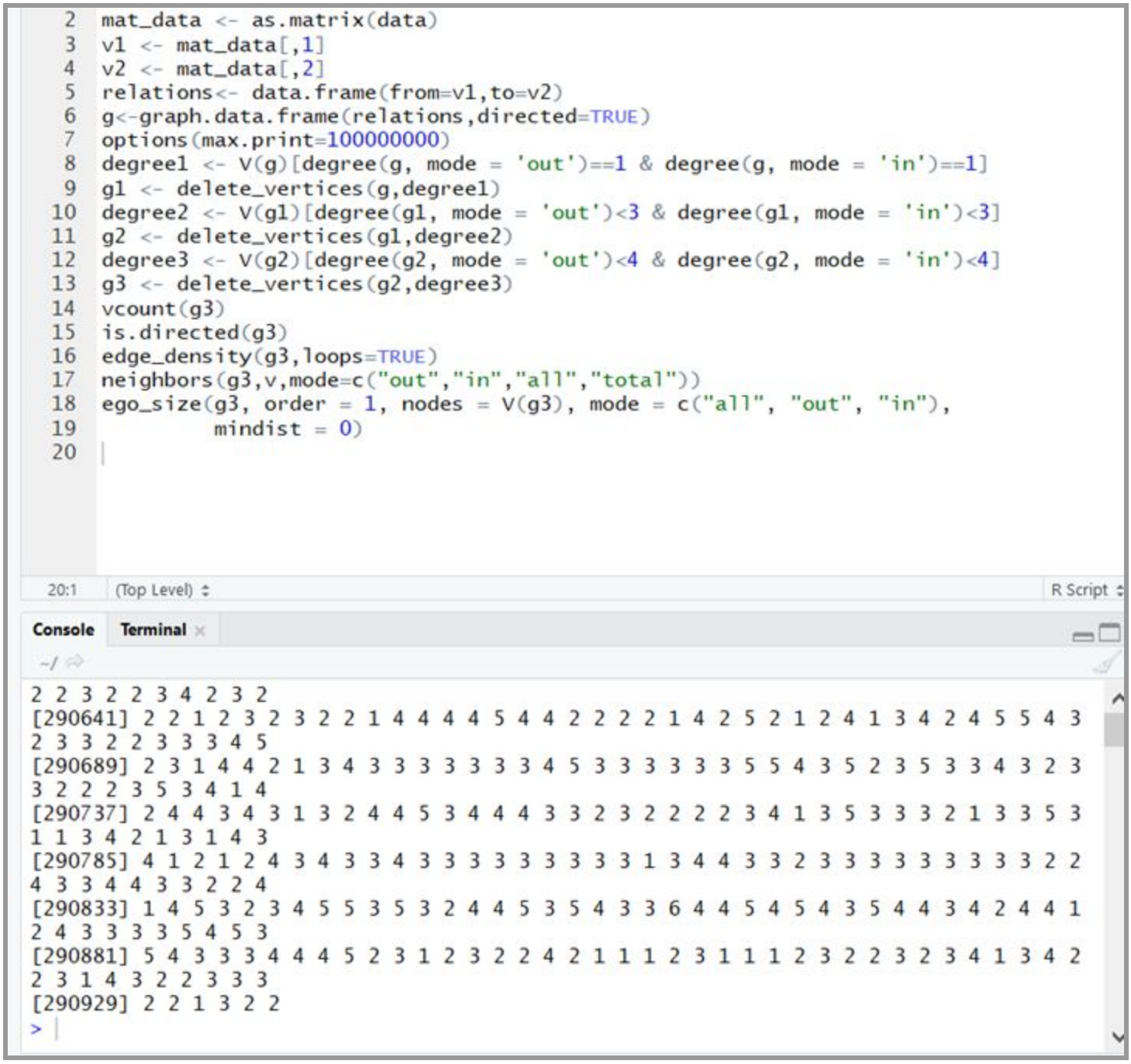
3) Neighbors

neighbors(g3,v,mode=c(“in”,”out”,”total”,”all”)



4) Ego Size

ego_size(g3, order = 1, nodes = V(g3), mode = c("all", "out", "in"), mindist = 0)



```
mat_data <- as.matrix(data)
v1 <- mat_data[,1]
v2 <- mat_data[,2]
relations<- data.frame(from=v1,to=v2)
g<-graph.data.frame(relations,directed=TRUE)
options(max.print=100000000)
degree1 <- V(g)[degree(g, mode = 'out')==1 & degree(g, mode = 'in')==1]
g1 <- delete_vertices(g,degree1)
degree2 <- V(g1)[degree(g1, mode = 'out')<3 & degree(g1, mode = 'in')<3]
g2 <- delete_vertices(g1,degree2)
degree3 <- V(g2)[degree(g2, mode = 'out')<4 & degree(g2, mode = 'in')<4]
g3 <- delete_vertices(g2,degree3)
vcount(g3)
is.directed(g3)
edge_density(g3,loops=TRUE)
neighbors(g3,v,mode=c("out","in","all","total"))
ego_size(g3, order = 1, nodes = V(g3), mode = c("all", "out", "in"),
         mindist = 0)
```

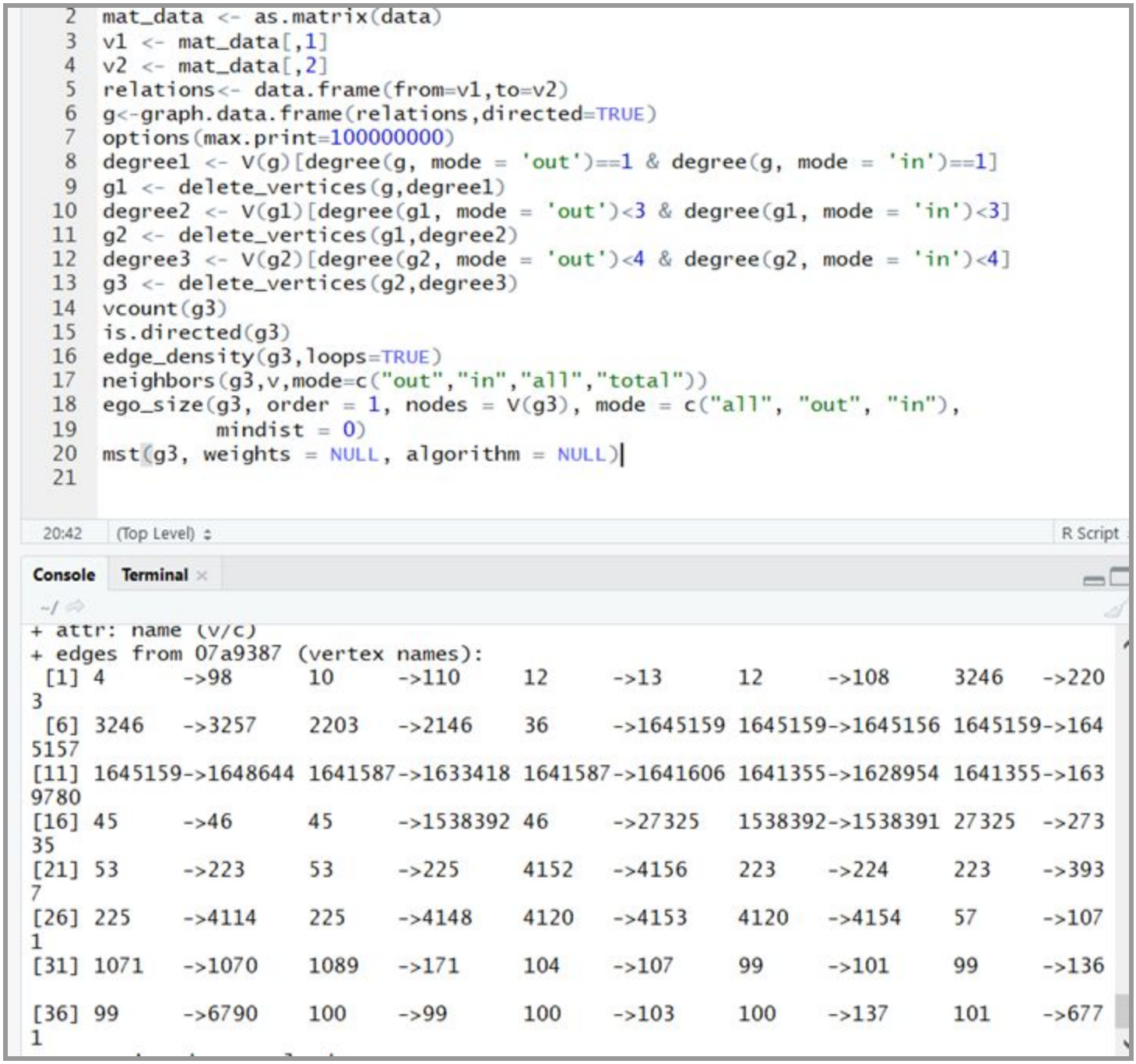
20:1 (Top Level) R Script

Console Terminal

```
2 2 3 2 2 3 4 2 3 2
[290641] 2 2 1 2 3 2 3 2 2 1 4 4 4 4 5 4 4 2 2 2 2 1 4 2 5 2 1 2 4 1 3 4 2 4 5 5 4 3
2 3 3 2 2 3 3 3 4 5
[290689] 2 3 1 4 4 2 1 3 4 3 3 3 3 3 3 3 4 5 3 3 3 3 3 3 5 5 4 3 5 2 3 5 3 3 4 3 2 3
3 2 2 2 3 5 3 4 1 4
[290737] 2 4 4 3 4 3 1 3 2 4 4 5 3 4 4 4 3 3 2 3 2 2 2 2 3 4 1 3 5 3 3 3 2 1 3 3 5 3
1 1 3 4 2 1 3 1 4 3
[290785] 4 1 2 1 2 4 3 4 3 3 4 3 3 3 3 3 3 3 3 3 1 3 4 4 3 3 2 3 3 3 3 3 3 3 3 2 2
4 3 3 4 4 3 3 2 2 4
[290833] 1 4 5 3 2 3 4 5 5 3 5 3 2 4 4 5 3 5 4 3 3 6 4 4 5 4 5 4 3 5 4 4 3 4 2 4 4 1
2 4 3 3 3 3 5 4 5 3
[290881] 5 4 3 3 3 4 4 4 5 2 3 1 2 3 2 2 4 2 1 1 1 2 3 1 1 1 2 3 2 2 3 2 3 4 1 3 4 2
2 3 1 4 3 2 2 3 3 3
[290929] 2 2 1 3 2 2
>
```

5) Minimum Spanning Tree

mst(g3, weights = NULL, algorithm = NULL)



```
mat_data <- as.matrix(data)
v1 <- mat_data[,1]
v2 <- mat_data[,2]
relations<- data.frame(from=v1,to=v2)
g<-graph.data.frame(relations,directed=TRUE)
options(max.print=100000000)
degree1 <- V(g)[degree(g, mode = 'out')==1 & degree(g, mode = 'in')==1]
g1 <- delete_vertices(g,degree1)
degree2 <- V(g1)[degree(g1, mode = 'out')<3 & degree(g1, mode = 'in')<3]
g2 <- delete_vertices(g1,degree2)
degree3 <- V(g2)[degree(g2, mode = 'out')<4 & degree(g2, mode = 'in')<4]
g3 <- delete_vertices(g2,degree3)
vcount(g3)
is.directed(g3)
edge_density(g3,loops=TRUE)
neighbors(g3,v,mode=c("out","in","all","total"))
ego_size(g3, order = 1, nodes = V(g3), mode = c("all", "out", "in"),
         mindist = 0)
mst(g3, weights = NULL, algorithm = NULL)
```

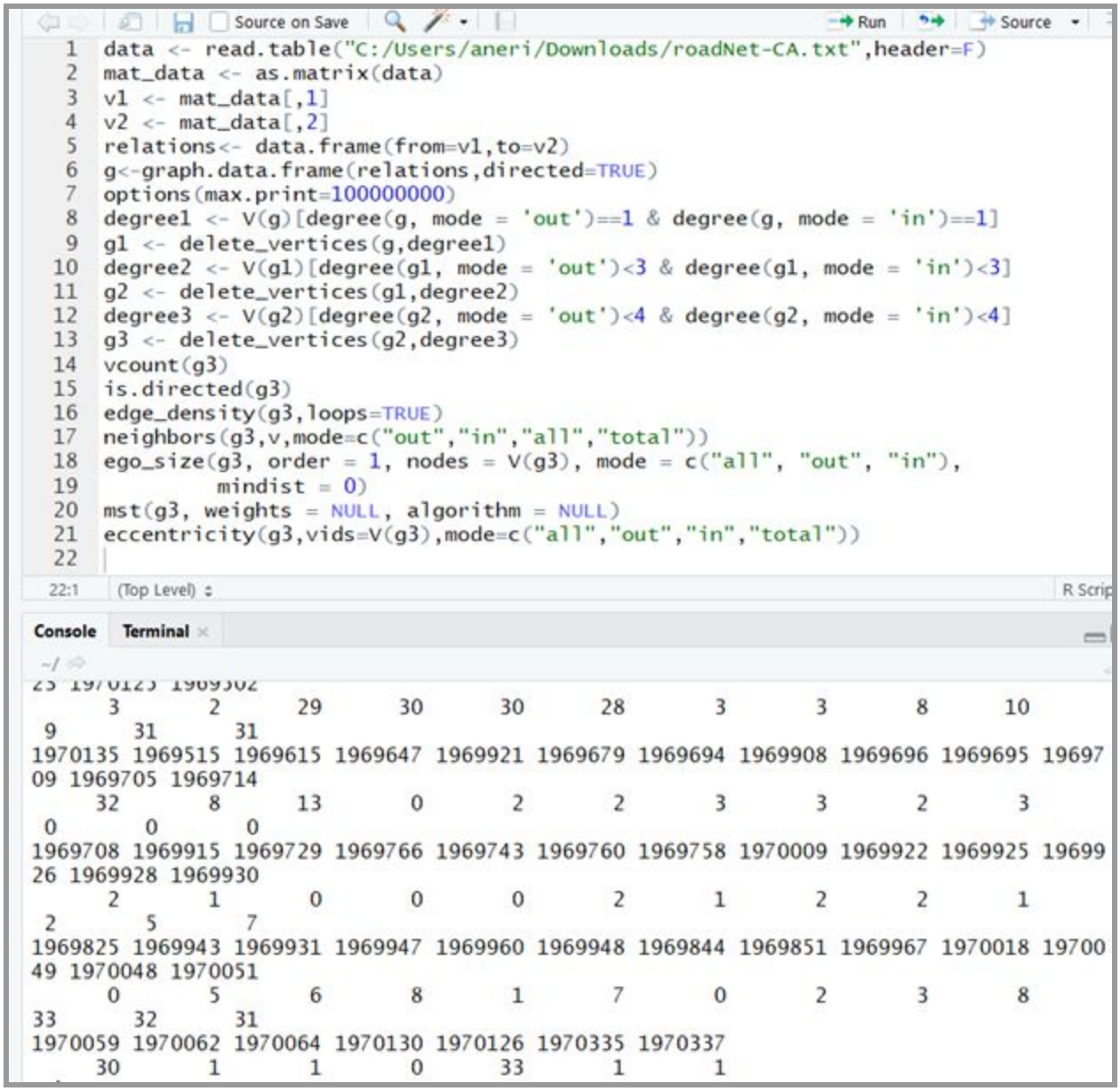
20:42 (Top Level) R Script

Console Terminal

```
+ attr: name (v/c)
+ edges from 07a9387 (vertex names):
 [1] 4       ->98       10      ->110      12      ->13       12      ->108      3246    ->220
3
 [6] 3246    ->3257     2203    ->2146     36      ->1645159  1645159->1645156  1645159->164
5157
[11] 1645159->1648644  1641587->1633418  1641587->1641606  1641355->1628954  1641355->163
9780
[16] 45      ->46       45      ->1538392  46      ->27325    1538392->1538391  27325   ->273
35
[21] 53      ->223      53      ->225      4152    ->4156     223     ->224      223     ->393
7
[26] 225     ->4114     225     ->4148     4120    ->4153     4120    ->4154     57      ->107
1
[31] 1071    ->1070     1089    ->171      104     ->107      99      ->101      99      ->136

[36] 99      ->6790     100     ->99       100     ->103      100     ->137      101     ->677
1
```

6) Eccentricity

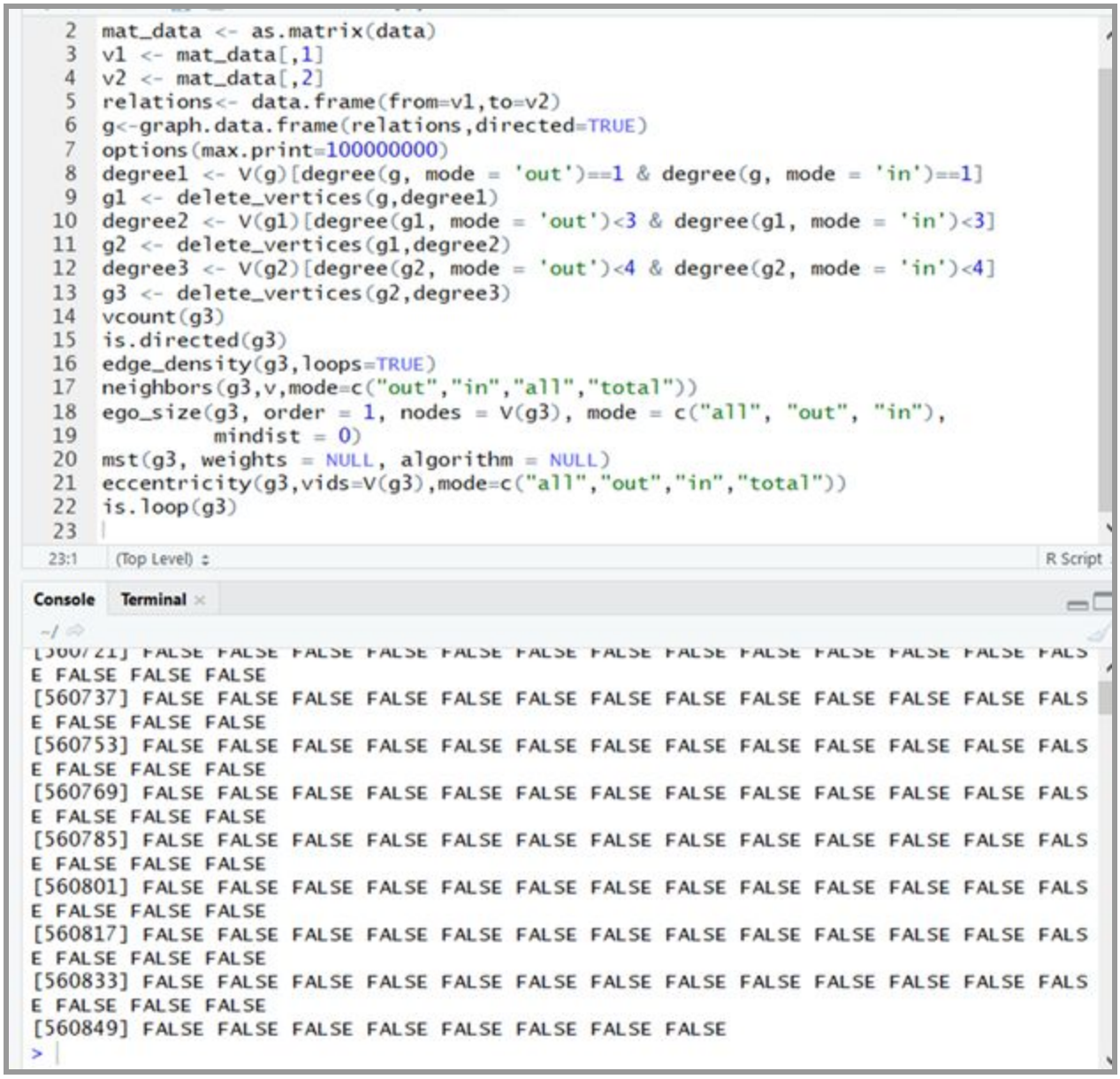
eccentricity(g3,vids=V(g3),mode=c("all","out","in","total"))



```
data <- read.table("C:/Users/aneri/Downloads/roadNet-CA.txt",header=F)
mat_data <- as.matrix(data)
v1 <- mat_data[,1]
v2 <- mat_data[,2]
relations<- data.frame(from=v1,to=v2)
g<-graph.data.frame(relations,directed=TRUE)
options(max.print=100000000)
degree1 <- V(g)[degree(g, mode = 'out')==1 & degree(g, mode = 'in')==1]
g1 <- delete_vertices(g,degree1)
degree2 <- V(g1)[degree(g1, mode = 'out')<3 & degree(g1, mode = 'in')<3]
g2 <- delete_vertices(g1,degree2)
degree3 <- V(g2)[degree(g2, mode = 'out')<4 & degree(g2, mode = 'in')<4]
g3 <- delete_vertices(g2,degree3)
vcount(g3)
is.directed(g3)
edge_density(g3,loops=TRUE)
neighbors(g3,v,mode=c("out","in","all","total"))
ego_size(g3, order = 1, nodes = V(g3), mode = c("all", "out", "in"),
         mindist = 0)
mst(g3, weights = NULL, algorithm = NULL)
eccentricity(g3,vids=V(g3),mode=c("all","out","in","total"))
```

```
      3       2      29      30      30      28       3       3       8      10
 9      31      31
1970135 1969515 1969615 1969647 1969921 1969679 1969694 1969908 1969696 1969695 19697
09 1969705 1969714
     32       8      13       0       2       2       3       3       2       3
 0       0       0
1969708 1969915 1969729 1969766 1969743 1969760 1969758 1970009 1969922 1969925 19699
26 1969928 1969930
      2       1       0       0       0       2       1       2       2       1
 2       5       7
1969825 1969943 1969931 1969947 1969960 1969948 1969844 1969851 1969967 1970018 19700
49 1970048 1970051
      0       5       6       8       1       7       0       2       3       8
33      32      31
1970059 1970062 1970064 1970130 1970126 1970335 1970337
     30       1       1       0      33       1       1
```

7) is.loop(g3)



```
mat_data <- as.matrix(data)
v1 <- mat_data[,1]
v2 <- mat_data[,2]
relations<- data.frame(from=v1,to=v2)
g<-graph.data.frame(relations,directed=TRUE)
options(max.print=100000000)
degree1 <- V(g)[degree(g, mode = 'out')==1 & degree(g, mode = 'in')==1]
g1 <- delete_vertices(g,degree1)
degree2 <- V(g1)[degree(g1, mode = 'out')<3 & degree(g1, mode = 'in')<3]
g2 <- delete_vertices(g1,degree2)
degree3 <- V(g2)[degree(g2, mode = 'out')<4 & degree(g2, mode = 'in')<4]
g3 <- delete_vertices(g2,degree3)
vcount(g3)
is.directed(g3)
edge_density(g3,loops=TRUE)
neighbors(g3,v,mode=c("out","in","all","total"))
ego_size(g3, order = 1, nodes = V(g3), mode = c("all", "out", "in"),
         mindist = 0)
mst(g3, weights = NULL, algorithm = NULL)
eccentricity(g3,vids=V(g3),mode=c("all","out","in","total"))
is.loop(g3)
```

23:1 (Top Level) R Script

Console Terminal

```
[560721] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALS
E FALSE FALSE FALSE
[560737] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALS
E FALSE FALSE FALSE
[560753] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALS
E FALSE FALSE FALSE
[560769] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALS
E FALSE FALSE FALSE
[560785] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALS
E FALSE FALSE FALSE
[560801] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALS
E FALSE FALSE FALSE
[560817] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALS
E FALSE FALSE FALSE
[560833] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALS
E FALSE FALSE FALSE
[560849] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
>
```

8) diameter

A network diameter is the longest geodesic distance (length of the shortest path between two nodes) in the network. The result indicates the number of nodes along the path.

```
> diameter(g3, directed = TRUE)
[1] 416
```

9) get_diameter

Returns the nodes along the first path found in the network with the longest geodesic distance. The result is a vertex sequence.

```
> diam <- get_diameter(g3)
> diam
+ 417/290934 vertices, named, from 92007aa:
  [1] 515136 515137 515138 515124 515119 514997 534895
  [8] 534503 534502 514966 514965 534892 514991 534890
 [15] 534889 534855 532317 514940 534853 514915 506303
 [22] 505950 534849 505959 534874 534873 534840 534876
 [29] 505933 505932 534870 505902 505901 534857 505997
 [36] 505170 505008 505009 505011 505167 505179 505178
 [43] 505184 505182 505186 505185 505730 505743 505744
 [50] 505751 505752 505749 505754 505717 505701 505719
 [57] 505820 505714 505711 505707 505692 504695 504696
 [64] 505685 505682 505680 505649 505644 505642 505639
+ ... omitted several vertices
```

10) hub_score

Hubs are generally used to examine web pages containing a large number of outgoing links. The result displays the hub score of each vertex.

```
> hub_score(g3, scale = TRUE)
$vector
           4           98           10          110           12           13
1.779357e-16 1.779357e-16 1.779357e-16 1.779357e-16 8.625324e-17 4.312662e-17
         108         3255         3246         2203           36      1645159
4.312662e-17 0.000000e+00 2.095981e-16 4.078202e-16 0.000000e+00 0.000000e+00
     1641587      1641355           45           46      1538392        27325
0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
       27343           53         4152          223          225         4120
0.000000e+00 5.655526e-16 9.424399e-16 6.432512e-16 1.118045e-15 4.137488e-16
          57         1068         1071         1089           65           70
0.000000e+00 0.000000e+00 0.000000e+00 1.779357e-16 0.000000e+00 0.000000e+00
         119          104          134           99          100          101
0.000000e+00 0.000000e+00 0.000000e+00 5.498508e-15 4.610829e-15 5.425069e-15
         136         6790          103          137         6760          102
5.812587e-15 5.605652e-15 3.237296e-15 4.927670e-15 2.730909e-15 4.325552e-15
        6771         6733         6738         6713         6748          107
6.113392e-15 1.375330e-15 6.805932e-15 7.134937e-16 1.635621e-15 1.445901e-15
         106          135          151          115          117          171
1.446062e-15 1.445900e-15 0.000000e+00 1.779357e-16 1.779357e-16 1.779357e-16
         125          126          176          353          129          154
1.194685e-16 1.194685e-16 0.000000e+00 0.000000e+00 0.000000e+00 8.694271e-17
         421         6792         6805         6764          144          153
4.026211e-15 4.143359e-15 3.887047e-15 3.342207e-15 0.000000e+00 0.000000e+00
```

(remaining vertices not displayed)

11) betweenness

This function returns the number of geodesics passing through a node, hence indicating its centrality in the network.

```
> betweenness(g3, v = V(g3), directed = TRUE)
           4           98           10          110           12           13
0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 2.000000e+00 0.000000e+00
         108         3255         3246         2203           36      1645159
0.000000e+00 0.000000e+00 2.000000e+00 4.000000e+00 0.000000e+00 1.833333e+01
     1641587      1641355           45           46      1538392        27325
1.000000e+00 6.000000e+00 2.800000e+01 3.600000e+01 1.600000e+01 4.000000e+01
       27343           53         4152          223          225         4120
0.000000e+00 6.948056e+02 5.415216e+04 2.471412e+03 3.740853e+04 7.781860e+03
          57         1068         1071         1089           65           70
0.000000e+00 0.000000e+00 4.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
         119          104          134           99          100          101
0.000000e+00 0.000000e+00 0.000000e+00 1.738478e+05 3.115911e+05 1.605799e+04
         136         6790          103          137         6760          102
4.123721e+04 1.528893e+05 2.392165e+05 5.437765e+04 4.409269e+05 2.405678e+05
        6771         6733         6738         6713         6748          107
1.074170e+04 9.731864e+04 1.870769e+05 0.000000e+00 1.154737e+05 8.000000e+00
         106          135          151          115          117          171
8.000000e+00 8.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
```

(remaining vertices not displayed)

12) mean_distance

This function displays the mean of the shortest distances between every pair of nodes in the network.

```
> mean_distance(g3, directed = TRUE)
[1] 98.23975
```

13) which_multiple

This function returns if any of the vertices have an loop or multiple edge. Since we are using a simplified graph, the function should return FALSE for all the vertices.

```
> which_multiple(g1, eids = E(g1))
   [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
LSE FALSE
  [16] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
LSE FALSE
  [31] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
LSE FALSE
  [46] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
LSE FALSE
  [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
LSE FALSE
  [76] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
LSE FALSE
  [91] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
LSE FALSE
 [106] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
LSE FALSE
 [121] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
LSE FALSE
 [136] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
LSE FALSE
 [151] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FA
LSE FALSE
```

(remaining vertices not displayed)

14) are_adjacent

This function returns TRUE if two vertices are adjacent to one another and FALSE otherwise.

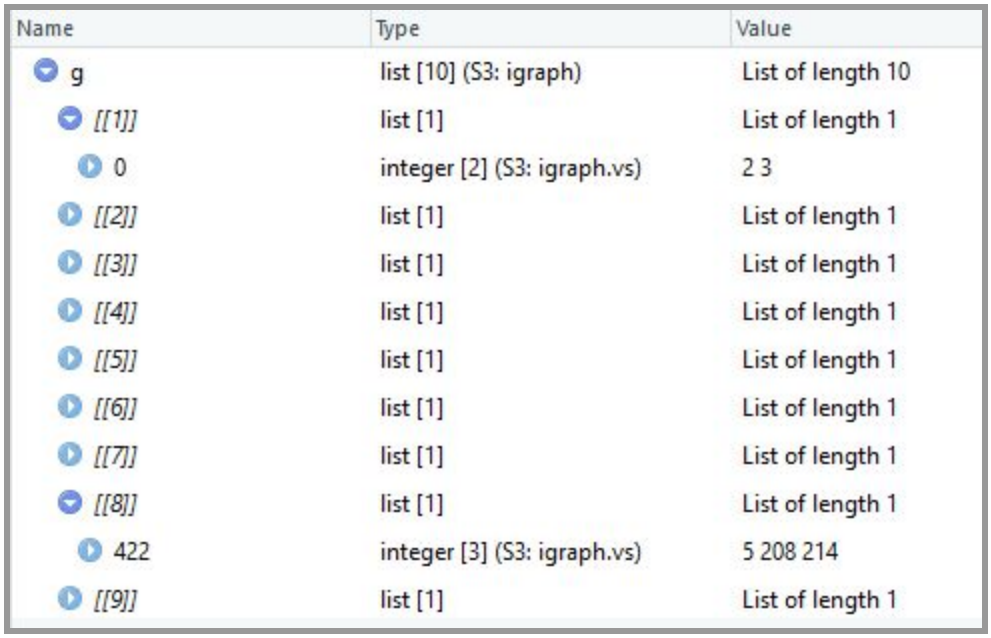
```
> are_adjacent(g1, 1, 2)
[1] TRUE
>
```

15) shortest_paths

This function returns the shortest path between two vertices.

```
> shortest_paths(g, 1, 8)
$vpath
$vpath[[1]]
+ 5/1231291 vertices, named, from c390685:
[1] 0   469 380 183 422


$epath
NULL

$predecessors
NULL

$inbound_edges
NULL

>
```

Here, vertex 1 is 0 and vertex 8 is 422



Name	Type	Value
g	list [10] (S3: igraph)	List of length 10
[[1]]	list [1]	List of length 1
0	integer [2] (S3: igraph.vs)	2 3
[[2]]	list [1]	List of length 1
[[3]]	list [1]	List of length 1
[[4]]	list [1]	List of length 1
[[5]]	list [1]	List of length 1
[[6]]	list [1]	List of length 1
[[7]]	list [1]	List of length 1
[[8]]	list [1]	List of length 1
422	integer [3] (S3: igraph.vs)	5 208 214
[[9]]	list [1]	List of length 1

6. Determine the

(a) central person(s) in the graph,
(b) longest path,
(c) largest clique,
(d) ego, and
(e) betweenness centrality and power centrality.

a. Is there more than one person with the most degrees?

Yes, there are more than one person with the most degrees. It is "291797" and "534751". We used the function V(g3)$name[degree(g3)==max(degree(g3))] to find the people with the most degrees.

The central person in the graph is the one which has the most degree.The central person will be the person with most number of edges. In degree and out degree.

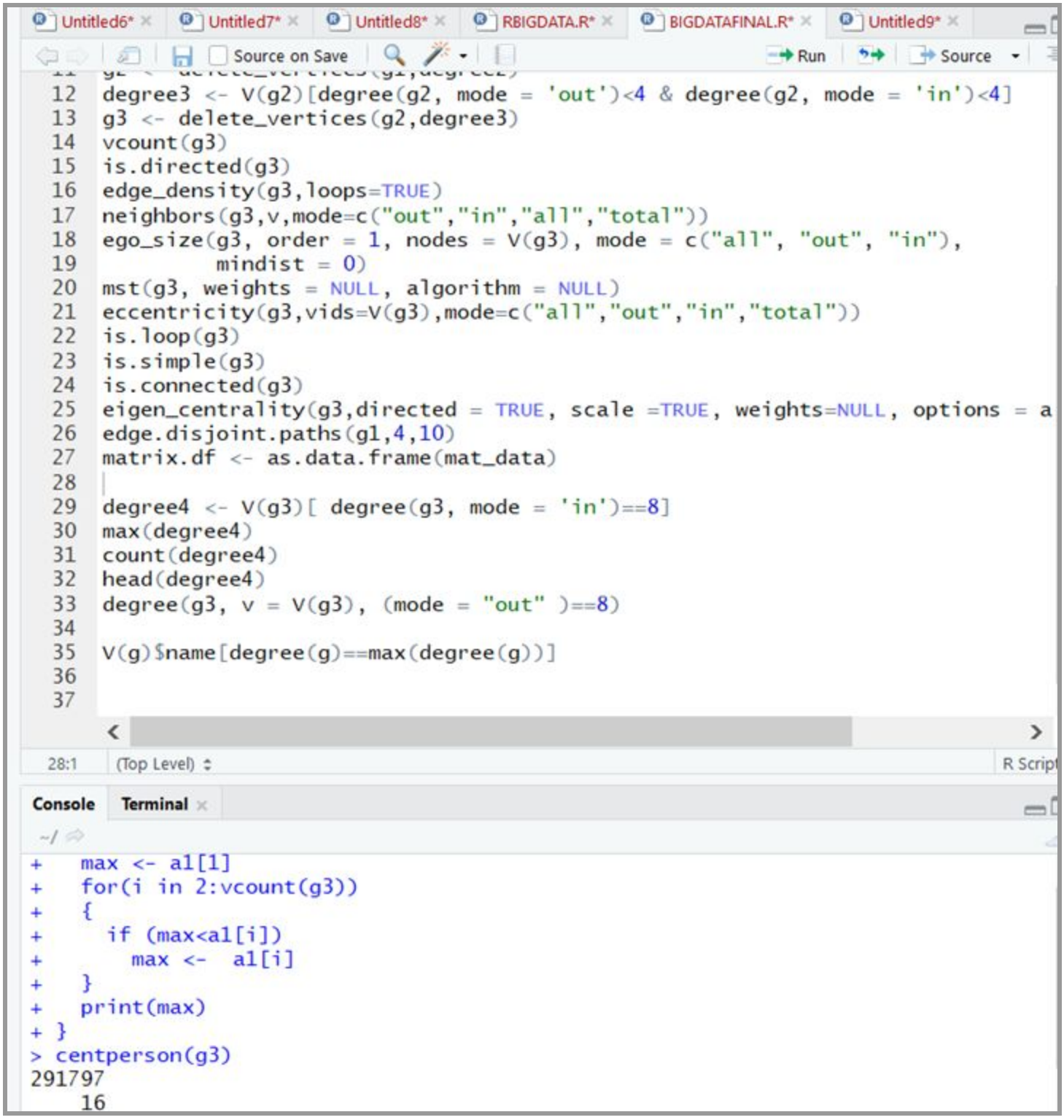
```
centperson <- function(g3) {
  a1 <- degree(g3) #it stores the degree of all vertices in g3
  max <- a1[1]   #a1[1] has the degree of node 1
```

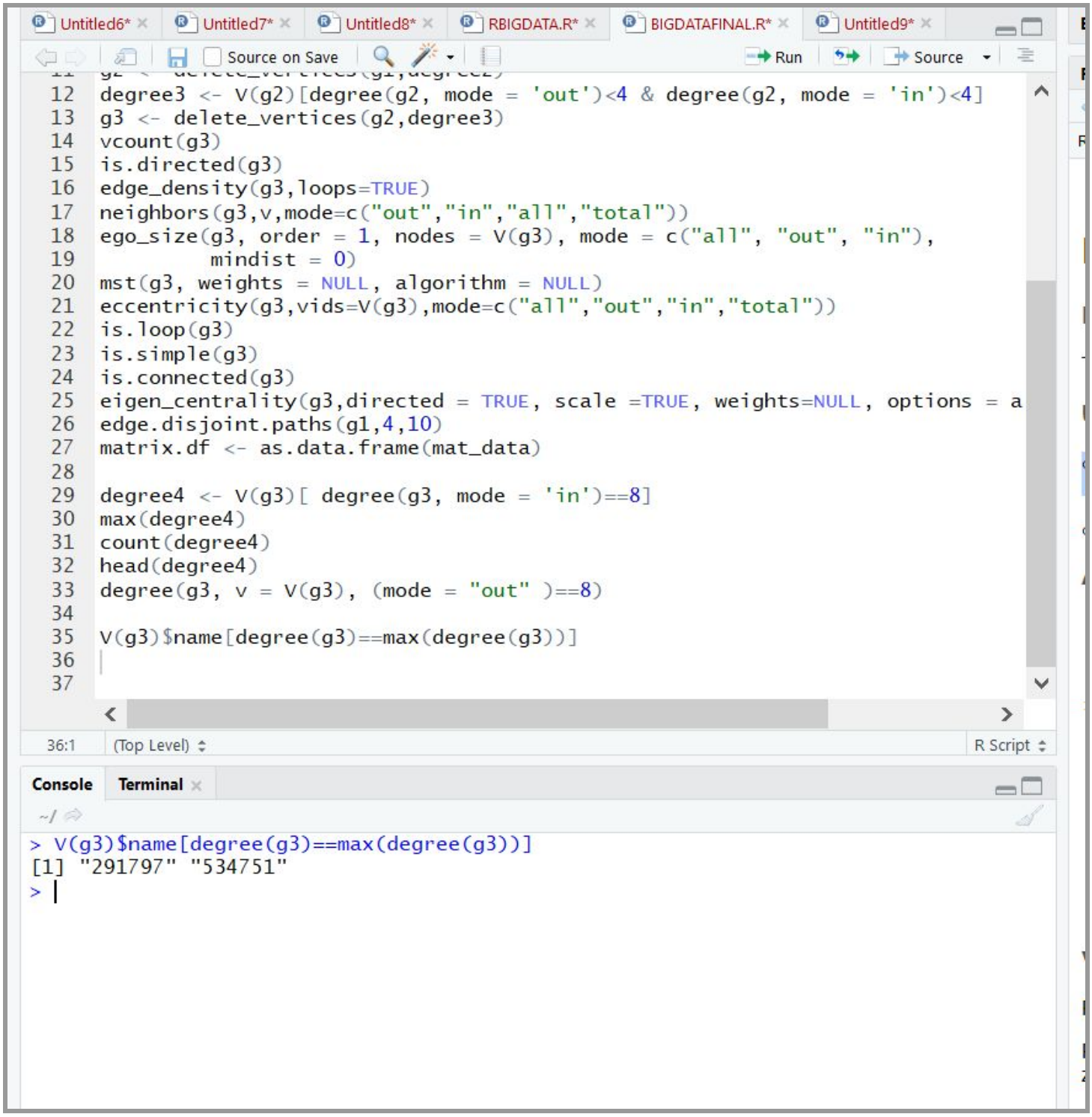
```
  for(i in 2:vcount(g3)) #we will iterate from 2 to the vcount(g3) which will the last count
  {
    if (max<a1[i]) #this will help us in finding the maximum of the degree node
      max <-  a1[i] #this will substitute the value of it
}
  print(max) #prints the max value
}
```

The above is the code which is used for finding the central person in the graph

Alternatively, we can also use to find the number of multiple nodes with largest degree.

V(g3)$name[degree(g3)==max(degree(g3))]


Untitled6*
Untitled7*
Untitled8*
RBIGDATA.R*
BIGDATAFINAL.R*
Untitled9*
Source on Save
Run
Source
degree3 <- V(g2)[degree(g2, mode = 'out')<4 & degree(g2, mode = 'in')<4]
g3 <- delete_vertices(g2,degree3)
vcount(g3)
is.directed(g3)
edge_density(g3,loops=TRUE)
neighbors(g3,v,mode=c("out","in","all","total"))
ego_size(g3, order = 1, nodes = V(g3), mode = c("all", "out", "in"),
mindist = 0)
mst(g3, weights = NULL, algorithm = NULL)
eccentricity(g3,vids=V(g3),mode=c("all","out","in","total"))
is.loop(g3)
is.simple(g3)
is.connected(g3)
eigen_centrality(g3,directed = TRUE, scale =TRUE, weights=NULL, options = a
edge.disjoint.paths(g1,4,10)
matrix.df <- as.data.frame(mat_data)
degree4 <- V(g3)[degree(g3, mode = 'in')==8]
max(degree4)
count(degree4)
head(degree4)
degree(g3, v = V(g3), (mode = "out")==8)
V(g)$name[degree(g)==max(degree(g))]
28:1
(Top Level)
R Script
Console
Terminal
~/
+ max <- a1[1]
+ for(i in 2:vcount(g3))
+ {
+ if (max<a1[i])
+ max <- a1[i]
+ }
+ print(max)
+ }
> centperson(g3)
291797
16


Untitled6*
Untitled7*
Untitled8*
RBIGDATA.R*
BIGDATAFINAL.R*
Untitled9*
Source on Save
Run
Source
degree3 <- V(g2)[degree(g2, mode = 'out')<4 & degree(g2, mode = 'in')<4]
g3 <- delete_vertices(g2,degree3)
vcount(g3)
is.directed(g3)
edge_density(g3,loops=TRUE)
neighbors(g3,v,mode=c("out","in","all","total"))
ego_size(g3, order = 1, nodes = V(g3), mode = c("all", "out", "in"),
mindist = 0)
mst(g3, weights = NULL, algorithm = NULL)
eccentricity(g3,vids=V(g3),mode=c("all","out","in","total"))
is.loop(g3)
is.simple(g3)
is.connected(g3)
eigen_centrality(g3,directed = TRUE, scale =TRUE, weights=NULL, options = a
edge.disjoint.paths(g1,4,10)
matrix.df <- as.data.frame(mat_data)
degree4 <- V(g3)[degree(g3, mode = 'in')==8]
max(degree4)
count(degree4)
head(degree4)
degree(g3, v = V(g3), (mode = "out")==8)
V(g3)$name[degree(g3)==max(degree(g3))]
36:1
(Top Level)
R Script
Console
Terminal
> V(g3)$name[degree(g3)==max(degree(g3))]
[1] "291797" "534751"
>

b. Are there multiple longest paths?

Yes, there are multiple longest paths. The longest path distance we got is 416 between two nodes. Diameter is the length of the longest path between two nodes. We use get_diameter to identify this path

We use the following functions to find our longest path:

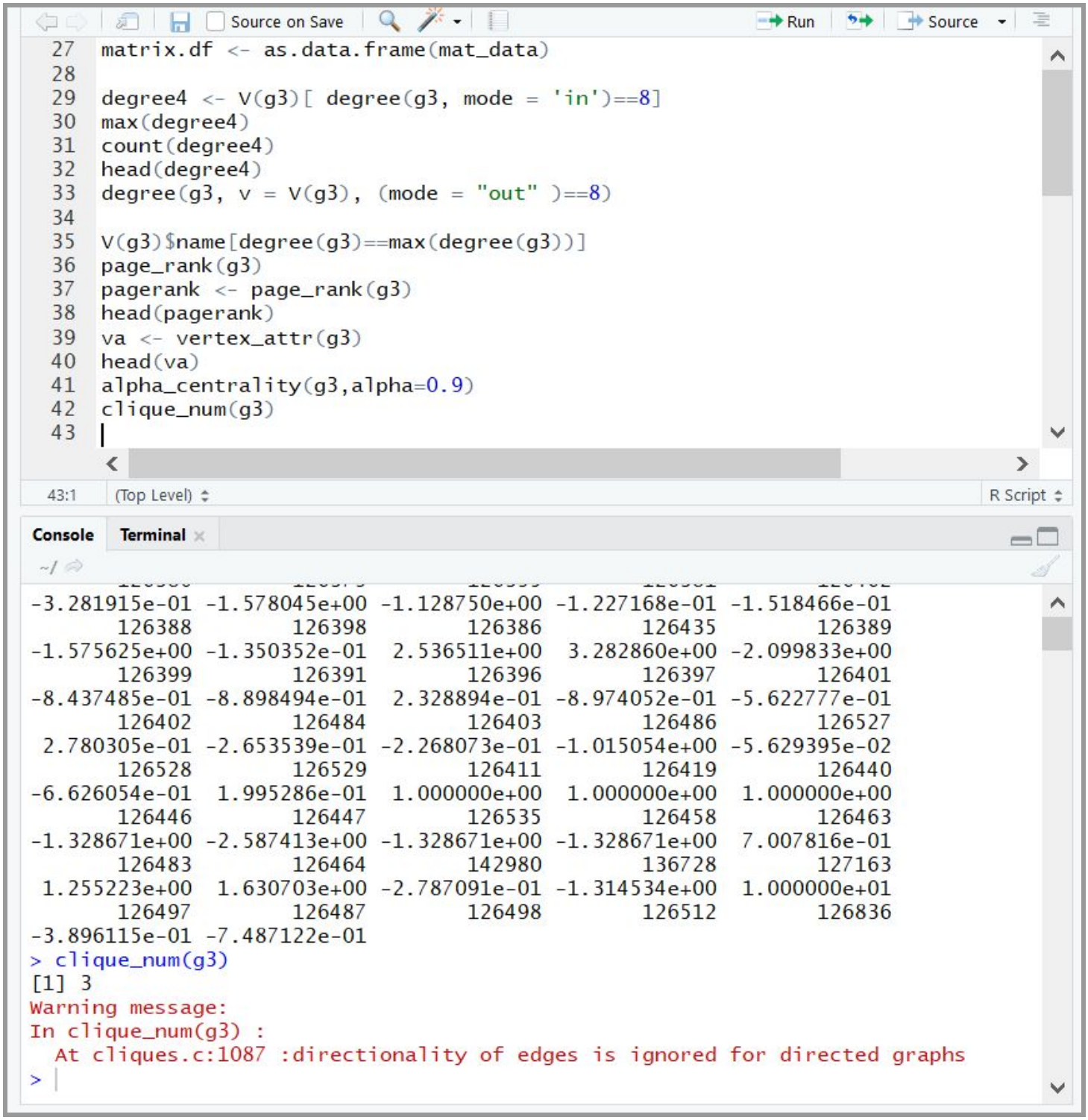
diameter(g3,directed=TRUE,weights=NA)
get_diameter(g3,directed=TRUE,weights=NA)

After using the get_diameter function, we can see that there are multiple longest paths.

```
> diameter(g3, directed=TRUE, weights=NA)
[1] 416
> get_diameter(g3, directed=TRUE, weights=NA)
+ 417/290934 vertices, named, from 4653a36:
  [1] 515136 515137 515138 515124 515119 514997 534895 534503 534502 514966
 [11] 514965 534892 514991 534890 534889 534855 532317 514940 534853 514915
 [21] 506303 505950 534849 505959 534874 534873 534840 534876 505933 505932
 [31] 534870 505902 505901 534857 505997 505170 505008 505009 505011 505167
 [41] 505179 505178 505184 505182 505186 505185 505730 505743 505744 505751
 [51] 505752 505749 505754 505717 505701 505719 505820 505714 505711 505707
 [61] 505692 504695 504696 505685 505682 505680 505649 505644 505642 505639
 [71] 505636 505634 505091 504596 504594 504132 504128 504126 504119 504114
 [81] 504113 504103 504092 504086 504071 504070 504066 504055 503991 503990
 [91] 504046 504049 504054 504063 504067 503964 503959 503881 503880 503994
+ ... omitted several vertices
```

c. Are there multiple cliques?
Yes, there are multiple cliques in the graph that are highest.

The function clique_num( ) returns the value 3 which indicates that there are 3 largest cliques in the graph



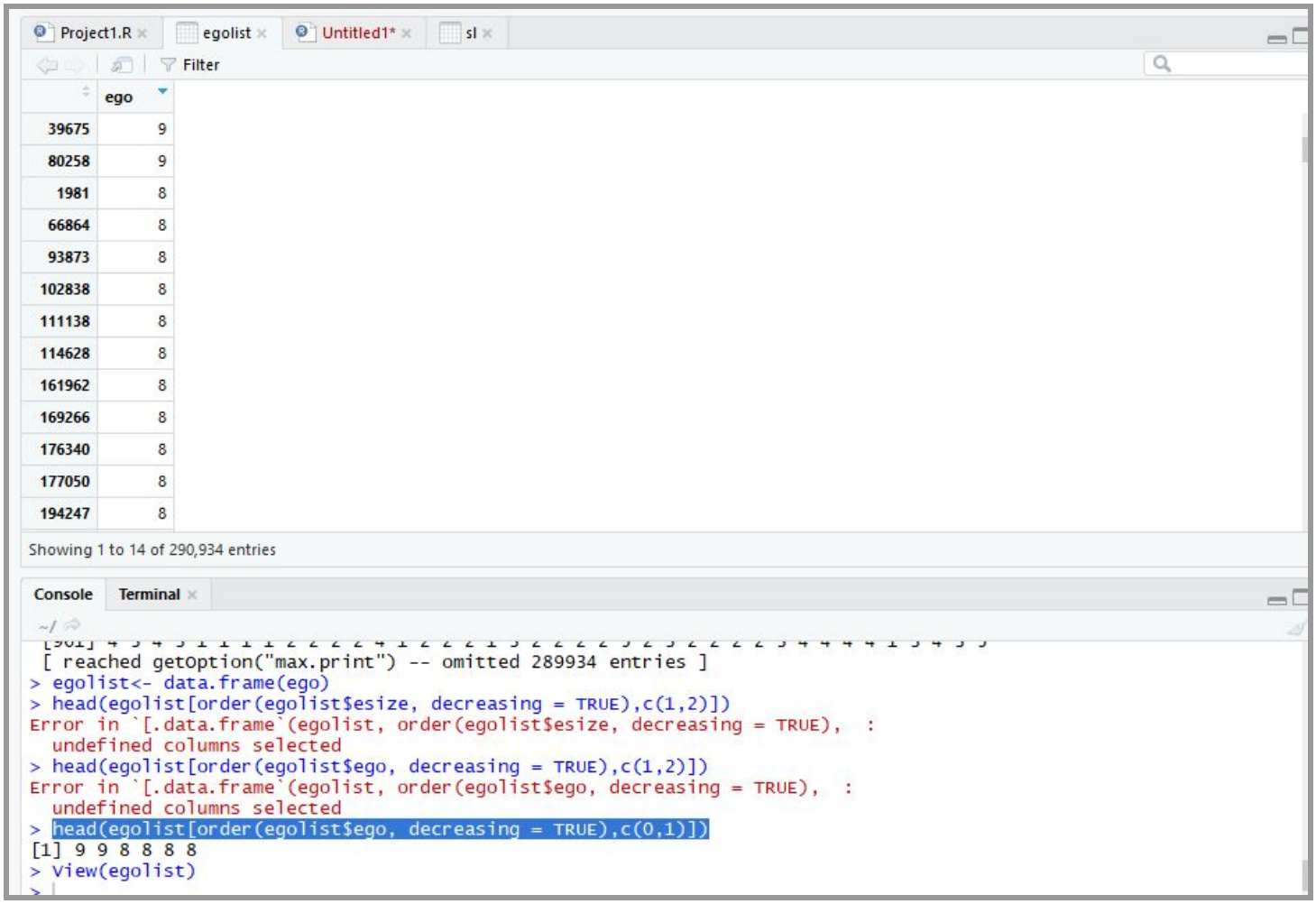
d. Are there more than one person with the highest ego?

Yes, there are more than one person with the highest ego.

To find this we used

ego<-ego_size(g3, nodes = V(g3), mode = c("all"), mindist = 0) *#find the ego of the graph*
egolist<- data.frame(ego) *#convert the ego(numeric data) to a dataframe to sort descendingly*
head(egolist[order(egolist$ego, decreasing = TRUE),c(0,1)]) *#sort descendingly and print out the top few nodes.*

To the above command, the system returns multiple entries with ego size of 9 which clearly indicates that there are more that one person with the highest ego.



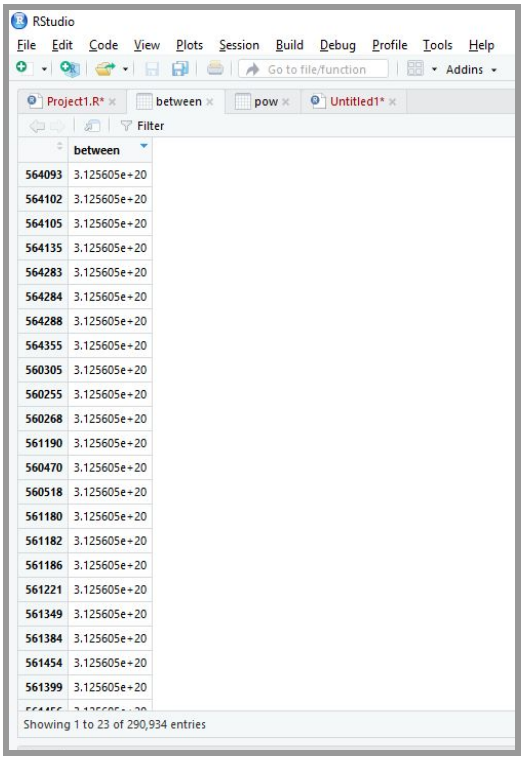
e. What is the difference in betweenness centrality vs. power centrality for the cases you find? Consider comparing the nodes that are members of each set. Are there common nodes?

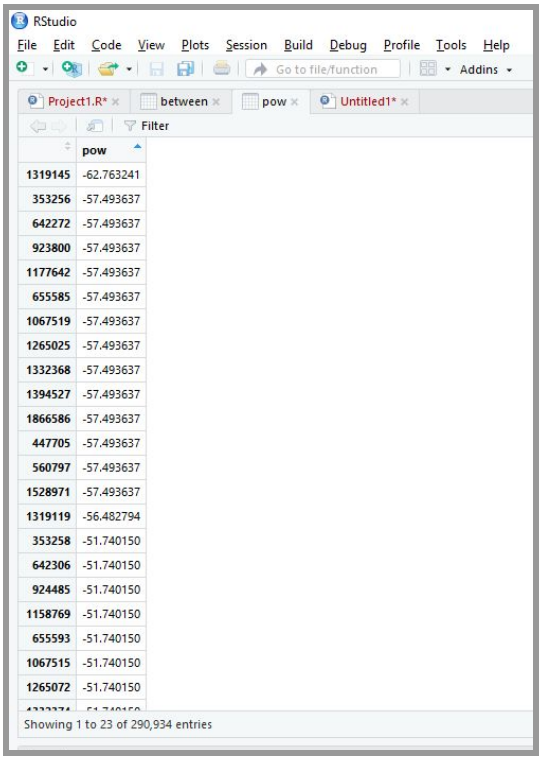
Betweenness centrality: It measures the extent to which a vertex is on paths between other vertices in the graph. Vertices with high betweenness value tend to have considerable influence within a network.

Power Centrality: Also known as '*Bonacich's approach'* to degree centrality is a function of the connections of the vertices in one's neighborhood. The more connections the vertices in the neighborhood have, the more central the vertex is.

```
between<-betweenness(g3))
between<-data.frame(between)
View(between)

pow<-power_centrality(g3,exponent = 0.9)
View(pow)
```

Through the results, we can infer that the centralities of both the different methods turn out different. The nodes that seem powerful or the most influential according to the power centrality might not be as influential according to betweenness.





7. Find the 20 nodes with the greatest neighborhood out to a distance 3 from the node. Do any of these neighborhoods overlap?

To get the neighborhood size of each node

```
> #Store the neighbourhood size of each vertex going out to a order of 3
> #g1 is the graph
> #order of the neighbourhood is 3
> #nodes takes all the vertices in graph g1
> #out mode calculate the neighbourhood using only the outgoing edges
> #mindlist = 0, considers minimum distance 0.
> esize <- ego_size(g1, order = 3, nodes = V(g1), mode = c("out"),
+           mindist = 0)
> esize
   [1]  2  2  2  2  3  3  3  1  7  7  8 10  5  5  6  7  5  7  1 13 23 11 17 16  4  1  4
  2  1  1
  [31]  1  6  6 25 25 25 23 24 21 24 23 24 25 17 27 11 17  6  6  6  6  2  2  2  5  5  7
  6  7  9
  [61] 21 21 22 22  1  9  7 11  9 11 10 11  6 10  9  1  2  2  7 10  6  9  3 10 10  9  9
  3  3  4
  [91]  4  4  4 13 12 16 22  8  1  2  2  5  6  8  8  4  4  4  4  5  1  5  1 21 23 22 22
 24 25 21
 [121] 16 22 24 22 23 23 26 16  5 10  9 15 15 13  6 10  4  4  5  6  4  4  4  4  4  1 12
 10 22  1
 [151]  1  4  4  1  4  4  5  4  7 11  2  6  1  2  8  1  4  5  4  4  1  4  6  5  6  5  6
 15 17 17
```

(remaining vertices not displayed)

To get the 20 greatest nodes

```
> #Get the 20 nodes with the highest neighborhood size to order 3 from esize
> b <- order(esize, na.last=TRUE, decreasing=TRUE)[1:20]
> b
 [1]  80258 229484  80137  89901 195437 195453 229944  80140  80256   2336   2370  4491
6  80215
[14]  83196  85707 195426 229634 229938   1686   2328
```

Display the neighborhood size of the 20 greatest nodes

```
> #Display the neighborhood size of nodes in b
> ego_size(g1, order = 3, nodes = b, mode = c("out"),
+                   mindist = 0)
 [1] 40 37 35 34 34 34 34 33 33 32 32 32 32 32 32 32 32 32 31 31
```