Introduction to Big Data & Analytics

Instructor: Professor Abzetdin Adamov

Group 7

Students: Rustam Talibzade, Emin Alizada

1. This project is about learning and practicing Graph Analytics. We were given a huge dataset of files with messages of people in Enron Corporation. As the data is too large, we were allowed to subset this data. Our group was assigned to subset the data by using all top-folders but limiting sub-folders with first 8 and then retrieve all mails within them. To accomplish parsing of given e-mails, we have used two R packages – **readr, stringr.** Readr is used for reading the context of the given files, and from readr we have used read_file function which reads all the contents of file. Stringr is used for string manipulations, to be more precise we have used following functions from stirngr:

   tlalocite – to find position of given substring in string
   str_sub – to get substring of string
   str_replace_all – to replace given substring with other one.

   After reading all top folders and first 8 subfolders of each top folder we have received approximately 141000 row of records.

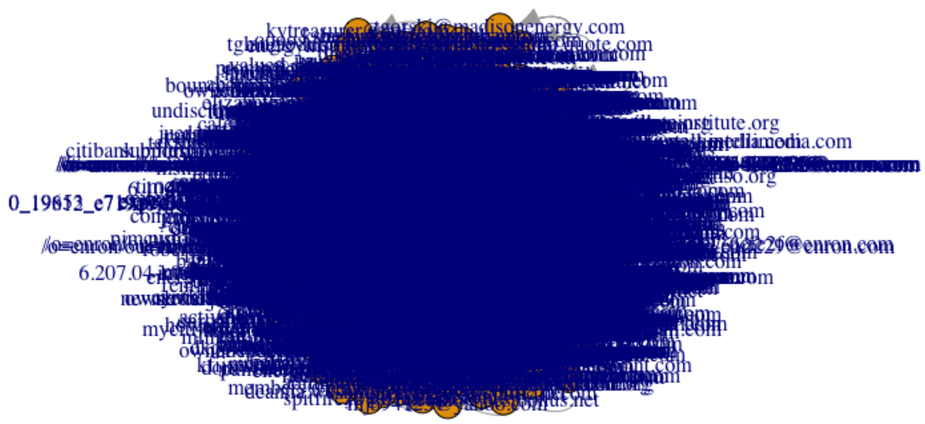
   After doing some cleanup such as removing rows having NA values and duplicated rows, we have 80860 records.

   df1 80860 obs. of 3 variables

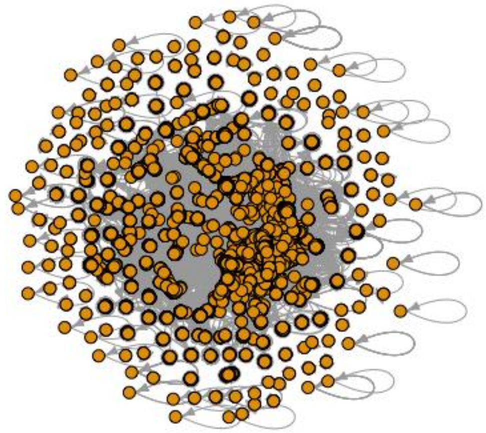
2. To reach the goals of our project we installed igraph package:

```
> install.packages("igraph")
WARNING: Rtools is required to build R packages but is not currently installed. Please d
ownload and install the appropriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Installing package into ‘C:/Users/user/Documents/R/win-library/4.1’
(as ‘lib’ is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/igraph_1.2.11.zip'
Content type 'application/zip' length 9014721 bytes (8.6 MB)
downloaded 8.6 MB

package ‘igraph’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\Public\Documents\Wondershare\CreatorTemp\RtmpGk2IB9\downloaded_packages
```

3. After installing igraph package we turned our data frame in igraph object and tried to plot it. As expected, we got a very messy blue bulb.

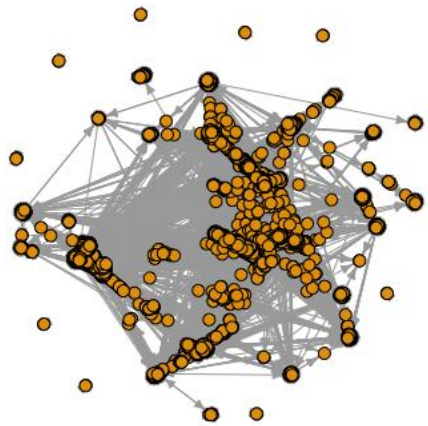


If we remove labels for simpler visuality:



Because of the fact that dataset is huge even after doing some cleanup, in order to work on this dataset on the local machine we needed to simplify it further. Firstly, we decided to remove all the mails outside of corporation, which means mail addresses that don't end with enron.com. Later we removed mails where from and to is the same (which means mails sent to yourself). The most important simplification was grouping the data by fields from, to and assigning weight according how frequent these email addresses communicate. As the final step we also used igraph's built in simplify function. As a result, following plots were produced.

If we remove labels for simpler visuality:



* We have used string representation of graph with str() function.

* We have used igraph::get.adjacency function like: ***igs.adj <- igraph::get.adjacency(igs)*** and printed first 9 rows and columns. Here is small part of output.

```
adnan.patel@enron.com                     .                 .                 .                 .
                                 a..shankman@enron.com actforchange.com@mailman.enron.com adam.johnson@enron.com
40ees@enron.com                                      .                                  .                      .
a..allen@enron.com                                   .                                  .                      .
a..lindholm@enron.com                                .                                  .                      .
a..martin@enron.com                                  .                                  .                      .
a..shankman@enron.com                                .                                  .                      .
actforchange.com@mailman.enron.com                   .                                  .                      .
adam.johnson@enron.com                               1                                  .                      .
administration.enron@enron.com                       .                                  .                      .
adnan.patel@enron.com                                .                                  .                      .
```

* To get list of vertexes we have used V function like: V(igs)
* To get list of edges we have used E function like: E(igs). Small portion of output (arrows show direction)

```
E(igs)
28639/28639 edges from 170209e (vertex names):
[1] 40ees@enron.com      ->bob.deitz@enron.com       40ees@enron.com      ->steve.wurzel@enron.com
[3] a..allen@enron.com   ->katina.smith@enron.com    a..lindholm@enron.com->john.lamb@enron.com
[5] a..lindholm@enron.com->m..presto@enron.com       a..lindholm@enron.com->mike.curry@enron.com
[7] a..lindholm@enron.com->michael.payne@enron.com   a..martin@enron.com  ->david.baumbach@enron.com
[9] a..martin@enron.com  ->edward.gottlob@enron.com  a..martin@enron.com  ->elsa.villarreal@enron.co
11] a..martin@enron.com  ->eric.bass@enron.com       a..martin@enron.com  ->greg.mcclendon@enron.com
13] a..martin@enron.com  ->j..farmer@enron.com       a..martin@enron.com  ->jim.schwieger@enron.com
15] a..martin@enron.com  ->metz.carey@enron.com      a..shankman@enron.com->david.oxley@enron.com
17] a..shankman@enron.com->greg.whalley@enron.com    a..shankman@enron.com->rick.buy@enron.com
19] a..shankman@enron.com->s..bradford@enron.com     a..shankman@enron.com->ben.glisan@enron.com
... omitted several edges
```

* To get density of graph we used function from SNA library igs.density = sna::gden(igs.adj) which accepts adjacency matrix.

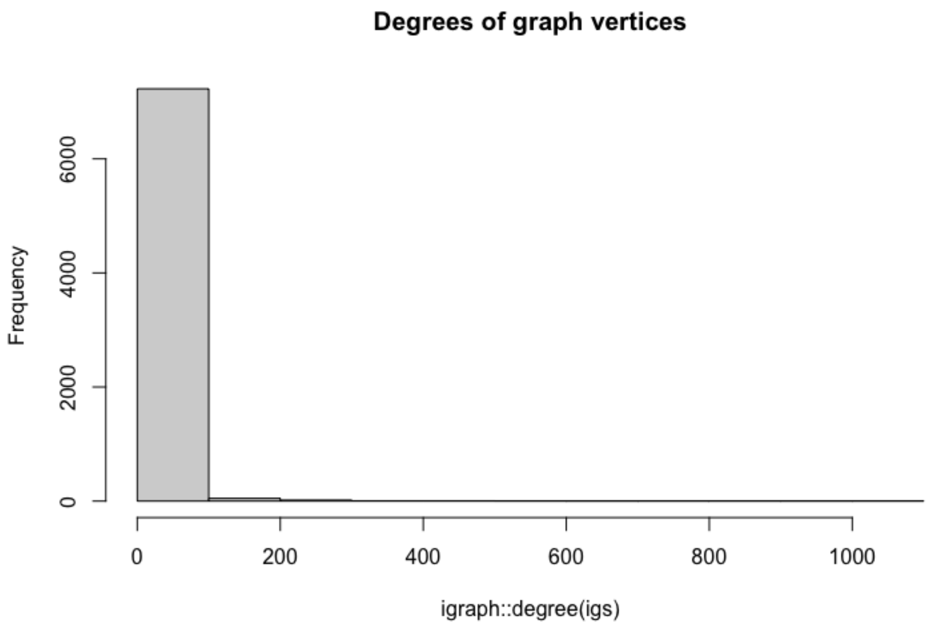
*To get edge density we have used ***igraph::edge_density(igs)*** (which was very small)

```
> igraph::edge_density(igs)
[1] 0.0005361686
```

* To get the degree of vertices we have used ***igraph::degree(igs).*** Small potion of output

```
                          7                           5                           65
   richard.sanders@enron.com   richard.shapiro@enron.com   richard.tomaski@enron.com
                         407                          75                           5
   rick.bergsieker@enron.com          rick.buy@enron.com        rick.cates@enron.com
                           4                          68                           2
         rika.imai@enron.com        rishi.mehta@enron.com       rita.bahner@enron.com
                          52                           1                           6
        rita.wynne@enron.com        rob.bradley@enron.com         rob.brown@enron.com
                          16                          10                           5
          rob.cole@enron.com          rob.walls@enron.com      robert.badeer@enron.com
                          10                          24                          64
 robert.eickenroht@enron.com       robert.frank@enron.com      robert.gerry@enron.com
                         240                          14                           4
   robert.hemstock@enron.com    robert.johnston@enron.com robert.jones@mailman.enron.com
                          22                          70                           1
     robert.knight@enron.com
                           6
```

* We can also visualize this by hist(igraph::degree(igs), main = "Degrees of graph vertices"):



* ***To get betweenness centrality we have used igraph::centr_betw(igs)***
* We find out degree of a graph with igraph::diameter(igs)

```
> igraph::diameter(igs)
[1] 69
```

4. Functions from igraph.pdf:

a.) **neighbors** function allows us to find neighbours of our vertices – vertices from which edges are directed to the given node (in) or vertices to which edges that go out from the given vertice point to. This is quite useful for our dataset as we can find to whom the given email address mailed (out) or who mailed to the selected email address (in):

```
> neighbors(igs, 2, "in")
+ 22/7309 vertices, named, from 8064962:
 [1] andrea.dahlke@enron.com    carol.lapsley@enron.com    david.forster@enron.com    david.oxley@enron.com
 [5] debra.bailey@enron.com     duong.luu@enron.com        heather.choate@enron.com   julie.clyatt@enron.com
 [9] l..denton@enron.com        laurel.bolt@enron.com      m.hall@enron.com           madhup.kumar@enron.com
[13] mike.croucher@enron.com    r..harrington@enron.com    s..theriot@enron.com       sonia.hennessy@enron.com
[17] steve.nat@enron.com        tammie.schoppe@enron.com   tara.piazze@enron.com      veronica.espinoza@enron.com
[21] w..white@enron.com         william.crooks@enron.com
```

```
> neighbors(igs, 2, "out")
+ 1/7309 vertex, named, from 8064962:
[1] katina.smith@enron.com
```

b.) **Is_directed** function returns true or false depending on whether the graph is directed or not. Here we used the function for our directed graph, as we can see the returned values is TRUE:

```
> is_directed(igs)
[1] TRUE
```

c.) **Is_igraph** function returns true or false depending on whether the given variable is igraph. While it seems simple, this function is helpful when testing new functions and it also helped us during this project.
From the screenshot below, we got true for our igraph:

```
> is_igraph(igs)
[1] TRUE
```

d.) **Gsize()** – returns number of edges in our graph:

```
> gsize(ig)
[1] 80860
> gsize(igs)
[1] 28639
```

e.) **Gorder()** – returns number of vertices in our graph:

```
> gorder(ig)
[1] 14322
> gorder(igs)
[1] 7309
>
```

f.) **random_walk** – this function starts from the mentioned vertices which is a start point and randomly moves to different vertices X times (where X in out case is 6) if possible (f.e. if let's say the 4th node does not point to any other node then the function stops):

```
> random_walk(igs, 21, 6)
+ 6/7309 vertices, named, from 693ee03:
[1] alex.perwich@enron.com     greg.whalley@enron.com     john.lavorato@enron.com     greg.whalley@enron.com
[5] christie.patrick@enron.com jeff.skilling@enron.com
```

g.) **articulation_points** – this function returns those vertices that if removed will make the graph more connected:

```
> articulation_points(igs)
+ 344/7309 vertices, named, from 693ee03:
  [1] clickathome@enron.com                andrew.lewis@enron.com              drew.fossum@enron.com
  [4] jeff.skilling@enron.com              center.dl-portland@enron.com        chris.wiebe@enron.com
  [7] burton.mcintyre@enron.com            colleen.koenig@enron.com            larry.lawyer@enron.com
 [10] dl-ga-all_enron_houston@enron.com    chairman.ken@enron.com              administration.enron@enron.com
 [13] dl-ga-all_enron_worldwide1@enron.com announcements.enron@enron.com       energy.dl-ga-all_ubsw@enron.com
 [16] houston.dl-ubsw@enron.com            infrastructure.ubsw@enron.com       georgeanne.hodges@enron.com
 [19] lindon.chiu@enron.com                preston.ochsner@enron.com           james.noles@enron.com
 [22] bob.lee@enron.com                    shirley.crenshaw@enron.com          william.kendrick@enron.com
 [25] marie.hejka@enron.com                ozzie.pagan@enron.com               sam.romero@enron.com
 [28] carol.lapsley@enron.com              domingo.drakes@enron.com            vince.kaminski@enron.com
```

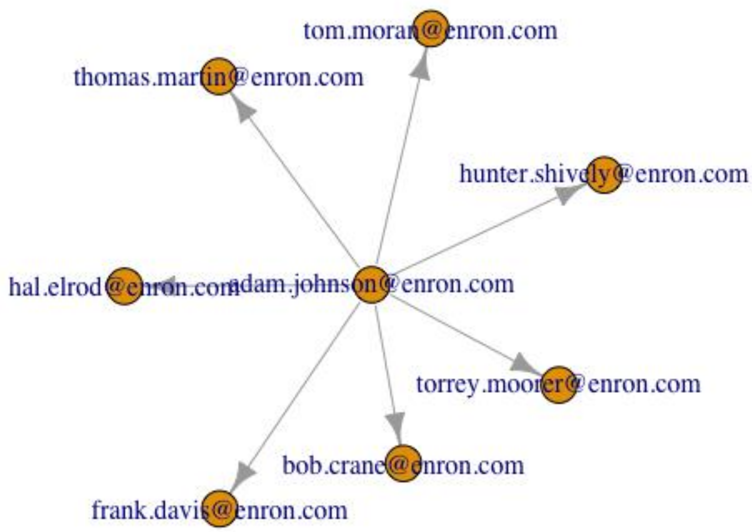
h.) **mean_distance** – this function returns average value of all shortest paths between vertices. This is what we got for our graph:

```
> mean_distance(igs)
[1] 4.330467
```

i.) **igraph::transitivity** - Transitivity measures the probability that the adjacent vertices of a vertex are connected. This is sometimes also called the clustering coefficient.

```
> igraph::transitivity(igs)
[1] 0.03923655
```

j.) Using **subgraph.edges** function we can get subgraph of our graph with specified edges. In our dataset it can be useful to find the connections for the given email addresses. We plotted our new subgraph using the piece of code mentioned below:
plot.igraph(subgraph.edges(igs, 37:43))



5. Determine the (a) central nodes(s) in the graph, (b) longest path(s), (c) largest clique(s), (d) ego(s), (e) power centrality, (f) find communities.

   a. Central Nodes - ***igraph::centr_degree(igs)*** (Part of output, because it's too large)

```
[573]    4   94   32   12   17   22   14   15   39    1   28   11    4    4   10   13   33   16    8   35   31  100
[595]  109    3   29   43    2   12    1   43   22   12   47  373    3   21    9    7   63   14    2    3   73  162
[617]   14    3   12   29   25  127   63   29   89   11   51   39    1    6   14    9    5  408   37   18    5   17
[639]   60   19   19    4   21    9    2  142 1013   54   76   20   25   20   28   29   30   32   22    8    7    4
[661]   59   16    2    1   10   11    2   59   12    2    2   17   10  105    6    2   14    1   10   30   10    6
[683]   28  186    4  116   15   14    9   21    3   15  243   13    3    5   18   10   45    2    1   11   17   17
[705]   19    6   24  234   15   24    8   12    3  133   45   41   11   55    1    7    2   39    3   11   19    7
[727]    7   26    4   46   27   35    7   23    1    1   23   55   16    8  247   81    2   99   17   27   10    2
[749]    3  153   26  193   11   48   60   16   22  129   11   36   20    4    4   10   56    4    2    2   10    7
[771]   35   35   49    8    3    5    1   11   11    8   20    3   20   73    8   25   11   76   36   50   24    8
[793]   36    1    8   30   10   62    5   18   65   17    2  353    2    3   50   19  125    4   12   23   24   75
[815]    5  104    1    9    9    9   12    4   21   17    7    4    5    1    6    6   12   27    4   67   17   10
[837]   66    6   15   15   37   35   50   12   13   10   16   59   55    1    3    1   74   15   11   31    1    1
[859]    1    8   25    5   32   56    7   24   23    4   15   12   88   15   19   25    3   29   34   12    2  171
[881]    3    9    3  323    3    7    2   10    1  113    3   12    8    1   12   28    1   10   50   22   27   25
[903]    3    2   59    3    1    2    4  268    5   44   13   98   16    9   12    6    9   10   28   20    2   13
[925]    2    2   28   40   40    1    1   19    5    4    1   31    5  241  100    3   14    9    7   35   32   15
[947]    2   20   15   33    3    6   10    7    9    3    3    2   60   10   31    6    6   10    2    2    2   84
[969]    5    4   12  224   20    8    2    7    5   65  407   75    5    4   68    2   52    1    6   16   10    5
[991]   10   24   64  240   14    4   22   70    1    6
 [ reached getOption("max.print") -- omitted 6309 entries ]

$centralization
[1] 0.06878085

$theoretical_max
[1] 106813728
```

b. Longest path
   igraph::diameter(igs)
   igraph::farthest_vertices(igs)

```
> igraph::diameter(igs)
[1] 69
> igraph::farthest_vertices(igs)
$vertices
+ 2/7309 vertices, named, from 693ee03:
[1] diane.cutsforth@enron.com dan.bump@enron.com

$distance
[1] 69
```

c. Largest clique - igraph::largest_cliques(igs)

```
> igraph::largest_cliques(igs)
[[1]]
+ 13/7309 vertices, named, from 693ee03:
 [1] kenneth.lay@enron.com      vanessa.groscrand@enron.com rosalee.fleming@enron.com   katherine.brown@enron.com
 [5] sherri.sera@enron.com      tori.wells@enron.com        bobbie.power@enron.com      nicki.daw@enron.com
 [9] rex.rogers@enron.com       john.sherriff@enron.com     greg.whalley@enron.com      mark.frevert@enron.com
[13] steven.kean@enron.com

[[2]]
+ 13/7309 vertices, named, from 693ee03:
 [1] kenneth.lay@enron.com      vanessa.groscrand@enron.com rosalee.fleming@enron.com   katherine.brown@enron.com
 [5] sherri.sera@enron.com      tori.wells@enron.com        bobbie.power@enron.com      nicki.daw@enron.com
 [9] rex.rogers@enron.com       david.delainey@enron.com    greg.whalley@enron.com      mark.frevert@enron.com
[13] steven.kean@enron.com
```

d. Ego - igraph::ego(igs)

```
> igraph::ego(igs)[1:10]
[[1]]
+ 3/7309 vertices, named, from 693ee03:
[1] 40ees@enron.com        bob.deitz@enron.com    steve.wurzel@enron.com

[[2]]
+ 24/7309 vertices, named, from 693ee03:
 [1] a..allen@enron.com         andrea.dahlke@enron.com     carol.lapsley@enron.com     david.forster@enron.com
 [5] david.oxley@enron.com      debra.bailey@enron.com      duong.luu@enron.com         heather.choate@enron.com
 [9] julie.clyatt@enron.com     katina.smith@enron.com      l..denton@enron.com         laurel.bolt@enron.com
[13] m.hall@enron.com           madhup.kumar@enron.com      mike.croucher@enron.com     r..harrington@enron.com
[17] s..theriot@enron.com       sonia.hennessy@enron.com    steve.nat@enron.com         tammie.schoppe@enron.com
[21] tara.piazze@enron.com      veronica.espinoza@enron.com w..white@enron.com          william.crooks@enron.com

[[3]]
+ 5/7309 vertices, named, from 693ee03:
```

e. Power centrality - igraph::power_centrality(igs, loops = F, exponent = 0.9)

```
> igraph::power_centrality(igs, loops = F, exponent = 0.9)
                40ees@enron.com              a..allen@enron.com           a..lindholm@enron.com
                    0.015803099                    -0.004667143                    -0.395589265
             a..martin@enron.com            a..shankman@enron.com  actforchange.com@mailman.enron.com
                    0.063212394                     1.062004881                     0.007901549
           adam.johnson@enron.com     administration.enron@enron.com           adnan.patel@enron.com
                   -0.494336709                     0.023704648                    -0.072680054
        adriane.schultea@enron.com         adrianne.engler@enron.com           aile@mailman.enron.com
                   -0.013965213                     3.476447640                     0.007901549
          airam.arteaga@enron.com          alan.aronowitz@enron.com             alan.chen@enron.com
                    1.135799240                    -0.562905593                     0.595471496
            alan.comnes@enron.com            alan.engberg@enron.com          albert.meyers@enron.com
                   -1.452069738                    -0.704964881                     0.022914493
           aleck.dadson@enron.com            alex.hidalgo@enron.com           alex.perwich@enron.com
                   -2.180893073                    -3.294402859                     0.105274190
           alex.saldana@enron.com         alex.villarreal@enron.com        alhamd.alkhayat@enron.com
```

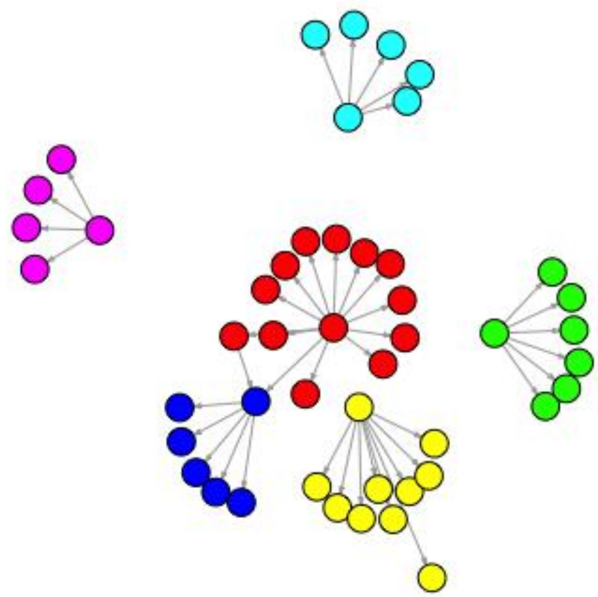
f. Communities - igraph::walktrap.community(igs)

```
> igraph::walktrap.community(igs)
IGRAPH clustering walktrap, groups: 108, mod: 0.61
+ groups:
  $`1`
    [1] "a..lindholm@enron.com"                    "alan.comnes@enron.com"
    [3] "albert.meyers@enron.com"                  "amy.copeland@enron.com"
    [5] "amy.jon@enron.com"                        "andrea.bertone@enron.com"
    [7] "anna.mehrer@enron.com"                    "bachelor.conf.@enron.com"
    [9] "bernadette.hawkins@enron.com"             "bert.meyers@enron.com"
   [11] "beth.perlman@enron.com"                   "beverly.stephens@enron.com"
   [13] "bill.williams@enron.com"                  "brad.alford@enron.com"
   [15] "brett.wiggs@enron.com"                    "brian.redmond@enron.com"
   [17] "bruno.gaillard@enron.com"                 "bryan.garrett@enron.com"
```

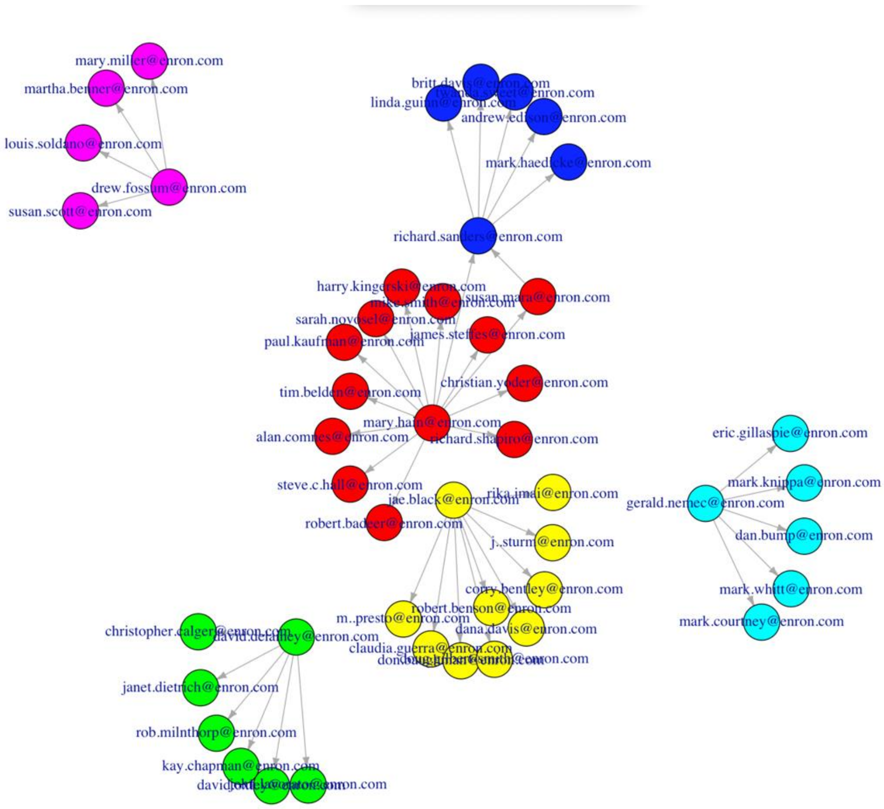
6. Resulting graph with too many vertices and edges will look very messy in the plot. Try to filter vertices and their edges in some way having in resulting plot (visualization) 30 – 100 vertexes. Differentiate vertices (by color, size, shape) and edges (color, type) of graph. Think about opportunity to assign weights to edges differentiating them accordingly.

   In order to have around 30-100 vertexes we have used our weights that were assigned earlier, so we decided to filter out all records where weight is less than 33.

   df2 <- df1[df1$weight > 34,]



The same plot with mail addresses.



Here is the list of final vertexes that we have. Which means they have written to each other the biggest number of emails

```
> V(ig2)
+ 47/47 vertices, named, from 58e4533:
 [1] david.delainey@enron.com     drew.fossum@enron.com         gerald.nemec@enron.com
 [4] jae.black@enron.com          mary.hain@enron.com           richard.sanders@enron.com
 [7] susan.mara@enron.com         christopher.calger@enron.com  david.oxley@enron.com
[10] janet.dietrich@enron.com     john.lavorato@enron.com       kay.chapman@enron.com
[13] rob.milnthorp@enron.com      louis.soldano@enron.com       martha.benner@enron.com
[16] mary.miller@enron.com        susan.scott@enron.com         dan.bump@enron.com
[19] eric.gillaspie@enron.com     mark.courtney@enron.com       mark.knippa@enron.com
[22] mark.whitt@enron.com         claudia.guerra@enron.com      corry.bentley@enron.com
[25] dana.davis@enron.com         don.baughman@enron.com        doug.gilbert-smith@enron.com
[28] j..sturm@enron.com           m..presto@enron.com           rika.imai@enron.com
+ ... omitted several vertices
```

Conclusion:

a. R programming language was relatively new area for both of us, so we started with Tutorial that is uploaded to Blackboard. After getting some essentials of working with the graphs in R with igraph package. We started implementation of the project. Reading the contents of the datasets was the starting point of this project, so we decided to do it together. We extracted *from, to* and *subject* with the help of functions in *stringr* package. Afterwards, we made small research about the ways to simplify data, and answer questions from assignment independently, later combined our ideas, and implemented it altogether. The hardest part of the assignment for us was simplification of data, which took some time for us to do it. We get rid of duplicate data and then used in-built simplify function as we mentioned above. For the first time in our lives, we understood how slow our computers can be when they are working with big data. RAM was full, CPU was loaded to 100% and even though it took around 15-20 minutes to construct CSV from the given dataset with the method that was assigned to us. Next, we get acquainted with new functions in *igraph* package and calculated the values we were asked in Question 5. Lastly, we plotted the graph by coloring the vertices to differentiate them.

b. Adding to what we mentioned above, eventually, we managed to find the email addresses who contacted the most number of people and sent or received the most amount mails among each other.