Big Data & Analytjcs

I found 3 very useful functions in igraph package which simplifies work with any given type of graph, it is easy to change the format of representation of graph by these three:

```
> matr ← get.adjacency(graph)
```

```
> edgelis ← get.adjedgelist(graph)
```

```
edgelis                          List of 308
    $ christi.nicolay@enron.com
     ..- attr(*, "vnames")= chr [1:3] "christi.nicolay@enron.com|james.steffes@enron.com" "christi.nicolay@enron.com|jeff…
     ..- attr(*, "env")=<weakref>
     ..- attr(*, "graph")= chr "ec80bc58-97d9-11ec-ba97-217ec8e064c9"
    $ oscar.dalton@enron.com
     ..- attr(*, "vnames")= chr [1:7] "oscar.dalton@enron.com|ozzie.pagan@enron.com" "oscar.dalton@enron.com|benjamin.rog…
     ..- attr(*, "env")=<weakref>
     ..- attr(*, "graph")= chr "ec80bc58-97d9-11ec-ba97-217ec8e064c9"
    $ twanda.sweet@enron.com
     ..- attr(*, "vnames")= chr [1:22] "twanda.sweet@enron.com|gfergus@brobeck.com" "twanda.sweet@enron.com| " "twanda.sw…
     ..- attr(*, "env")=<weakref>
     ..- attr(*, "graph")= chr "ec80bc58-97d9-11ec-ba97-217ec8e064c9"
    $ chairman.ken@enron.com
     ..- attr(*, "vnames")= chr [1:2] "chairman.ken@enron.com|dl-ga-all_enron_worldwide1@enron.com " "chairman.ken@enron.…
     ..- attr(*, "env")=<weakref>
     ..- attr(*, "graph")= chr "ec80bc58-97d9-11ec-ba97-217ec8e064c9"
    $ rosalee.fleming@enron.com
     ..- attr(*, "vnames")= chr [1:4] "rosalee.fleming@enron.com|dstraszheim@milken-inst.org " "rosalee.fleming@enron.com…
     ..- attr(*, "env")=<weakref>
     ..- attr(*, "graph")= chr "ec80bc58-97d9-11ec-ba97-217ec8e064c9"
```

```
> liss ← get.adjlist(graph)
```

This function finds how many max cliques there are in the given graph (makes graph undirected as the clique can be found only in undirected graphs)

```
> max_cliques_amount ← count_max_cliques(graph)
```

```
> max_cliques_amount <- count_max_cliques(graph)
Warning message:
In count_max_cliques(graph) :
  At maximal_cliques_template.h:261 :Edge directions are ignored for maximal clique calculation
> max_cliques_amount
[1] 280
```

Random walk function finds a random connected way from “start” with constant steps.

```
> random_walk(graph, start = 10, steps =  12)
```

```
> random_walk(graph, start = 10, steps =  12)
+ 2/308 vertices, named, from ec80bc5:
[1] robert.walker@enron.com shawna.flynn@enron.com
```

Triangles function finds all cycles with the size of 3 in graph.

```
>  triangles(graph)
```

```
> triangles(graph)
+ 15/308 vertices, named, from ec80bc5:
 [1] twanda.sweet@enron.com    gfergus@brobeck.com       pmeringolo@brobeck.com
 [4] twanda.sweet@enron.com    gfergus@brobeck.com        djn@pkns.com
 [7] twanda.sweet@enron.com    gfergus@brobeck.com       jfrizzell@gibbs-bruns.com
[10] twanda.sweet@enron.com                              gfergus@brobeck.com
[13]                           susan.mara@enron.com      bruno.gaillard@enron.com
```

Topological sort have three different modes to work with directed graphs (in/out/all).

```
> topological.sort(graph, mode = "out")
```

```
> topological.sort(graph, mode = "out")
+ 308/308 vertices, named, from ec80bc5:
  [1] christi.nicolay@enron.com

  [2] oscar.dalton@enron.com

  [3] twanda.sweet@enron.com

  [4] chairman.ken@enron.com
```

It was done after removal of all loops to be certain about it.

```
>  is.loop(graph)
```

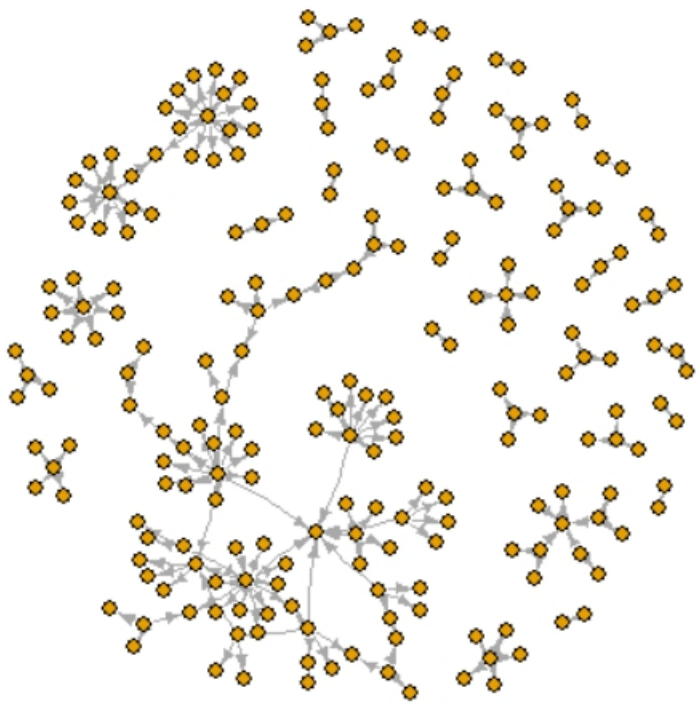
```
> is.loop(graph)
  [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [17] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [33] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [65] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [81] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [97] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[113] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[129] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[145] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[161] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[177] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[193] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[209] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[225] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[241] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[257] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
[273] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
> 
```

Deleting of particular vertices using query-alike mechanism.

```
> delete.vertices(graph = graph, which(igraph::degree(graph) > 65))
```

Was used in order to create a Fruchterman-Reingold layout algorithm generated graph. Also there are a lot of other layout algorithms within igraph package.

```
> layout_with_fr(graph)
```



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) centralization.degree(graph)

```
> igraph::centralization.degree(graph)
$res
  [1]  3  7 22  2  4 28  1  5 12  3  3  3  2  1  5 44  5  4  2  2  1  5  3  6  4  3  5  4  1  3  1  5  8  1  1  2  3  1
 [39]  2  1  2  3  3  2  3  5  1  5  5  5  3  1  3  2  3  1  1  1  3  2  1  3  5  4  1  3  1  1  3  1  1  1  1  1  1  1
 [77]  1  1  1 12  1  1  2  1  2  2  1  1  1  2  2  2  1  1  1  1  2  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
[115]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  2  1  1  1  1  2  1  1  1  1  1  1  1  1  2  2  2  1
[153]  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  1  1  1  1  1  1  1  1  1  1  1
[191]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  2  2  1  1  1  1  1  1  1  1  1  2  1  1  1  2  2
[229]  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  2  1  1  1  1  1  1  1  1  1  2  1  1  1  1  1  1
[267]  2  1  1  1  1  1  1  1  3  1  1  1  1  1  1  3  2  2  1  1  1  1  1  2  2  2  1  2  1  1  1  1  1  1  1  1  1  1
[305]  1  1  1  1

$centralization
[1] 0.06884954

$theoretical_max
[1] 188498
```

```
drew.fossum@enron.com
```

Center of the largest component.

```
11.2852798
```

(b)

```
> farthest_vertices(graph)
$vertices
+ 2/308 vertices, named, from ec80bc5:
[1] daphne@katzlaw.com      jeffrey.hodge@enron.com

$distance
[1] 3

> farthest_vertices(as.undirected(graph))
$vertices
+ 2/308 vertices, named, from d2f561b:
[1] bcash@interwest.com   roger.balog@enron.com

$distance
[1] 14
```

(c)
```
#Finding of max_clique size
max_clique = cliques(graph)
max_clique_val = 0
for (i in   max_clique)      {
        if (length(unlist(i)) > max_clique_val) {
        max_clique_val   =   length(unlist(i)) }
}
```

```
max_clique                  Large list (600 elements,  657.5 kB)
    $ : 'igraph.vs' Named int 16
     ..- attr(*, "names")= chr "drew.fossum@enron.com"
     ..- attr(*, "env")=<weakref>
     ..- attr(*, "graph")= chr "ec80bc58-97d9-11ec-ba97-217ec8e064c9"
    $ : 'igraph.vs' Named int 6
     ..- attr(*, "names")= chr "mary.hain@enron.com"
     ..- attr(*, "env")=<weakref>
     ..- attr(*, "graph")= chr "ec80bc58-97d9-11ec-ba97-217ec8e064c9"
    $ : 'igraph.vs' Named int 3
     ..- attr(*, "names")= chr "twanda.sweet@enron.com"
     ..- attr(*, "env")=<weakref>
     ..- attr(*, "graph")= chr "ec80bc58-97d9-11ec-ba97-217ec8e064c9"
    $ : 'igraph.vs' Named int 9
  max_clique_val            3L
```

```
[[572]]
+ 3/308 vertices, named, from ec80bc5:
[1] twanda.sweet@enron.com gfergus@brobeck.com    pmeringolo@brobeck.com
```

(d) graph_ego <- ego(graph)

```
graph_ego                 List of  308
   $ : 'igraph.vs' Named int [1:4] 1 41 72 73
    ..- attr(*, "names")= chr [1:4] "christi.nicolay@enron.com" "james.steffes@enron.com" "jeff.browr
    ..- attr(*, "env")=<weakref>
    ..- attr(*, "graph")= chr "ec80bc58-97d9-11ec-ba97-217ec8e064c9"
   $ : 'igraph.vs' Named int [1:8] 2 74 75 76 77 78 79 80
    ..- attr(*, "names")= chr [1:8] "oscar.dalton@enron.com" "ozzie.pagan@enron.com" "benjamin.rogers
    ..- attr(*, "env")=<weakref>
    ..- attr(*, "graph")= chr "ec80bc58-97d9-11ec-ba97-217ec8e064c9"
   $ : 'igraph.vs' Named int [1:23] 3 24 80 81 82 83 84 85 86 87 ...
    ..- attr(*, "names")= chr [1:23] "twanda.sweet@enron.com" "gfergus@brobeck.com" " " "rob.milnthor
    ..- attr(*, "env")=<weakref>
    ..- attr(*, "graph")= chr "ec80bc58-97d9-11ec-ba97-217ec8e064c9"
   $ : 'igraph.vs' Named int [1:3] 4 101 102
```

(e) pow ← power_centrality(graph = graph)

. . .

christi.nicolay@enron.com

1.0259345

oscar.dalton@enron.com

1.7953854

twanda.sweet@enron.com

6.9250580

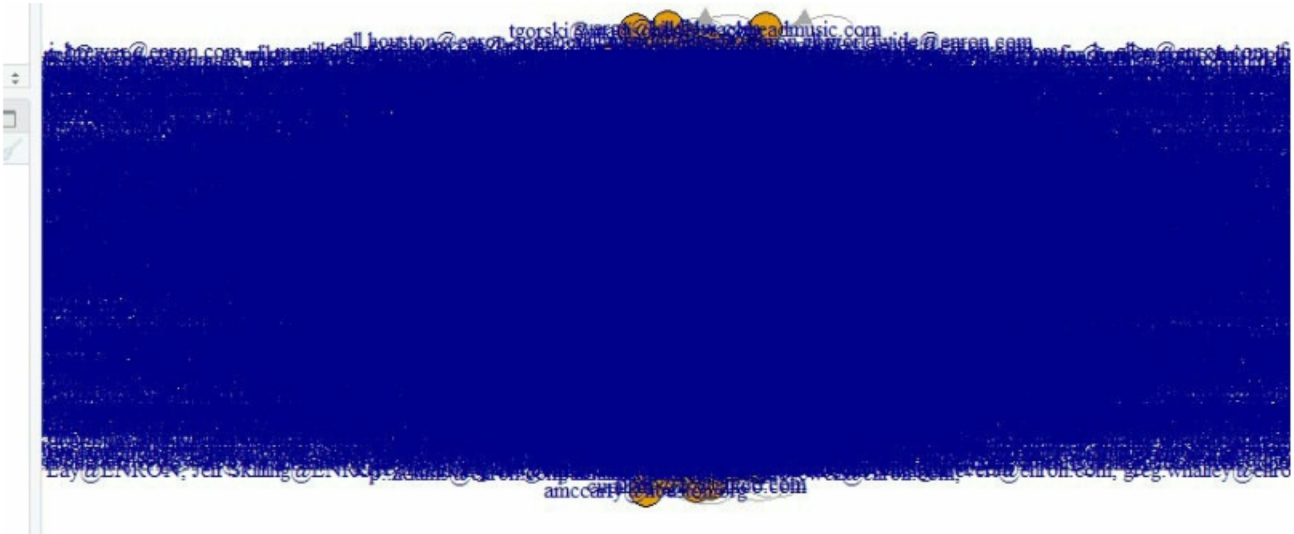
. . .

(f) com ← communities(x = components(graph))

```
com                         List of  38
    $ 1 : chr [1:5] "christi.nicolay@enron.com" "james.steffes@enron.com" "jeff.brown@enron.com" "ron.mcn…
    $ 2 : chr [1:99] "oscar.dalton@enron.com" "twanda.sweet@enron.com" "bruno.gaillard@enron.com" "robert…
    $ 3 : chr [1:3] "chairman.ken@enron.com" "dl-ga-all_enron_worldwide1@enron.com " " dl-ga-all_enron_wo…
    $ 4 : chr [1:5] "rosalee.fleming@enron.com" "dstraszheim@milken-inst.org " " dyergin@cera.com " " han…
    $ 5 : chr [1:29] "mary.hain@enron.com" " also introduced five bills to provide funding to \n\nReplace…
    $ 6 : chr [1:3] "kristenmansure@economist.com" "savont@email.msn.com" "klay@enron.com"
    $ 7 : chr [1:18] "elizabeth.sager@enron.com" "losullivan@isda.org" "richard.sanders@enron.com" "janic…
    $ 8 : chr [1:4] "wanda.holloway@compaq.com" "lawrence.t.babbio.jr@verizon.com" "lynnj@iname.com" "ted…
    $ 9 : chr [1:4] "denys.watson@enron.com" "steve.pruett@enron.com" "scott.josey@enron.com" "ccarver@al…
    $ 10: chr [1:3] "jmaas@llgm.com" "lysa.akin@enron.com " " lysa.akin@enron.com"
    $ 11: chr [1:45] "drew.fossum@enron.com" "ward to a point near the city of El Paso.  It's primarily a…
    $ 12: chr [1:4] "jrnels@sunvalley.net" "eserver@enron.com" "richard.b.sanders@enron.com " " richard.b…
    $ 13: chr [1:3] "russ.porter@enron.com" "benjamin.rogers@enron.com " " benjamin.rogers@enron.com"
    $ 14: chr [1:2] "m..presto@enron.com" "liz.taylor@enron.com"
    $ 15: chr [1:6] "lysa.akin@enron.com" "lmrig@uswest.net " " charles.yeung@enron.com" "tom.briggs@enro…
    $ 16: chr [1:4] "susan.scott@enron.com" "steven.harris@enron.com" "jeffery.fawcett@enron.com" "mary.m…
    $ 17: chr [1:5] "angela.mcculloch@enron.com" "kay.chapman@enron.com" "jason.biever@enron.com" "derek.…
    $ 18: chr [1:2] "mccue@mdcsystems.com" "distributionlist@mdcsystems.com"
    $ 19: chr [1:4] "b..sanders@enron.com" "zack.starbird@mirant.com" "sheryl_gussett@reliantenergy.com "…
    $ 20: chr [1:2] "dave.darnell@systrends.com" "tdtwg@ercot.com"
    $ 21: chr [1:6] "jeffrey.shankman@enron.com" "thomas.gros@enron.com " " michael.rosen@enron.com " " j…
    $ 22: chr [1:2] "a..martin@enron.com" "jim.schwieger@enron.com"
    $ 23: chr [1:3] "cindy.derecskey@enron.com" "jessica.ramirez@enron.com " " jessica.ramirez@enron.com"
    $ 24: chr [1:4] "lorna.brennan@enron.com" "julie.mccoy@enron.com" "steve.klimesh@enron.com" "gary.sov…
    $ 25: chr [1:3] "offley@hoover.stanford.edu" "paffairs2@hoover.stanford.edu " " paffairs2@hoover.stan…
    $ 26: chr [1:3] "press.release@enron.com" "ken.skilling@enron.com" "all.worldwide@enron.com"
    $ 27: chr [1:4] "cbi_mail@igate.cbinet.com" "dfossum@enron.com " " dfossum@enron.com " " dfossum@enro…
    $ 28: chr [1:4] "jgallagher@epsa.org" "bhawkin@enron.com" "bmerola@enron.com" "christi.l.nicolay@enro…
    $ 29: chr [1:3] "vanessa.groscrand@enron.com" "james.bannantine@enron.com" "cliff.baxter@enron.com"
    $ 30: chr [1:5] "ehuff@llgm.com" "rcjosephson@stoel.com" "dminson@aepnet.org" "pcooper@aepnet.org" ...
    $ 31: chr [1:2] "businessweek@clickaction.net" "gwhalle@enron.com"
    $ 32: chr [1:3] "cgrant@caiso.com" "dfuller@caiso.com" "20participants@caiso.com"
    $ 33: chr [1:3] "stephanie.truss@enron.com" "richard.b.sanders@enron.com" "chris.behney@enron.com"
    $ 34: chr [1:5] "phillip_fantle@cargill.com" "asettanni@bracepatt.com" "ayudkowsky@strook.com" "bcurr…
    $ 35: chr [1:2] "tony_r_frontino@calpx.com" "block_forward_traders_and_contacts@calpx.com"
    $ 36: chr [1:2] "mark.haedicke@enron.com" "mark.taylor@enron.com"
    $ 37: chr [1:2] "announcements.enron@enron.com" "dl-ga-all_domestic@enron.com"
    $ 38: chr [1:2] "careed@aep.com" "set@ercot.com"
      attr(*, "dim")= int 38
```

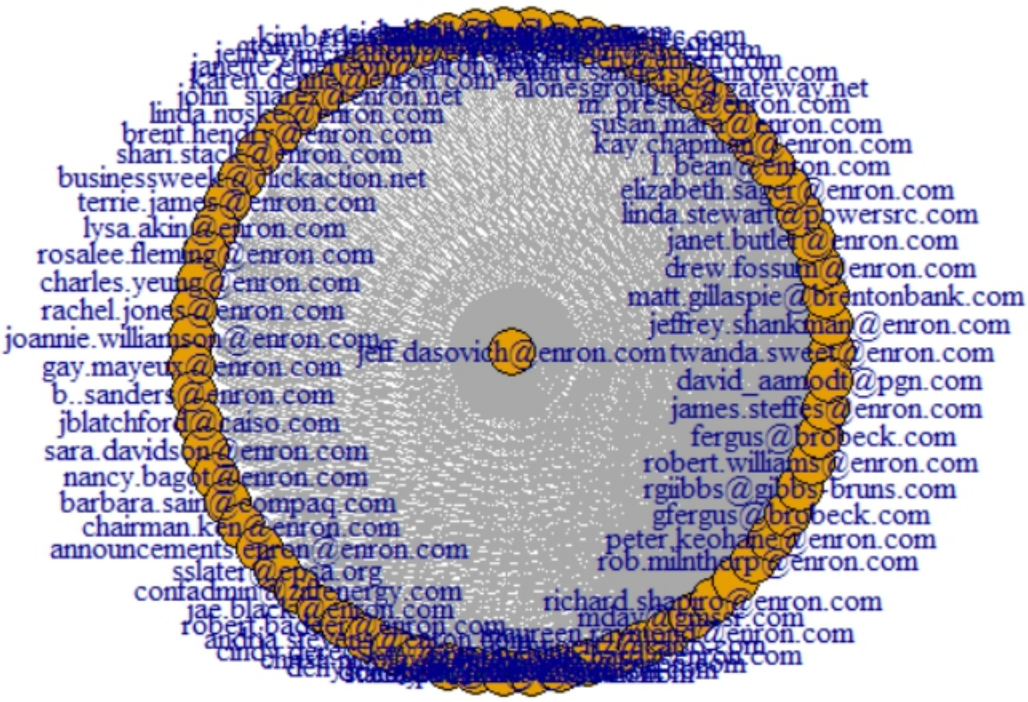
6. Resulting graph with too many vertices and edges will look very messy in the plot. Try to filter vertices and their edges in some way having in resulting plot (visualization) 30 – 100 vertexes. Differentiate vertices (by color, size, shape) and edges (color, type) of graph. Think about opportunity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

Graph before optimization in visualisation:



After first optimization / Star layout (was not very informative, but helped us move forward)
I used vertex with the highest centrality in order to create star ([25])

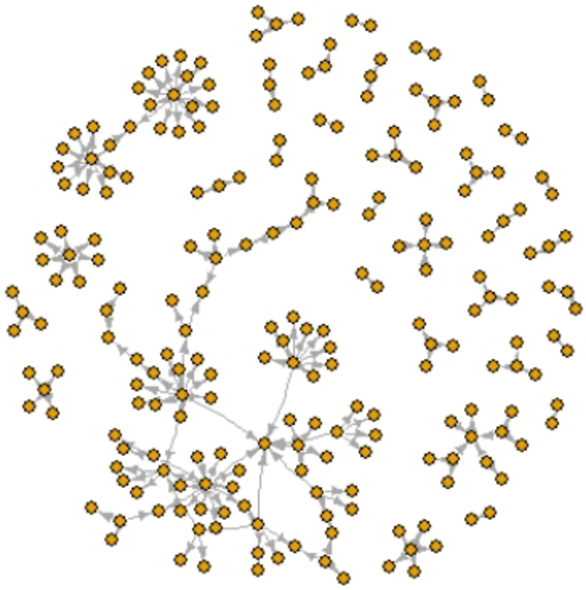
```
plot(graph, layout = layout_as_star(graph, center = V(graph)[25]))
```



The biggest optimization was using only E-Mails which contained "conference"in Subject:

```
if (!grepl(pattern = "conference", x = final_data_subject[counter],
ignore.case = T)) {
        found3 = F
      }
```

This way I was able to apply some real-world analysis to this model.I decided to use the Fruchterman-Reingold layout algorithm in order to create a more representative graph. Also I removed the naming in order to make it more readable.



Final decision in visualization was to add coloring accordingly to the component in which the vertex is (in our case this means, that people with the same color are visiting the same conferences.