

Big Data and Analytics (CS6444-10)

Class Project #1 - Group #1

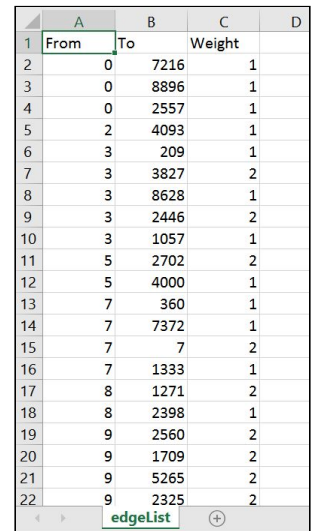
19 June 2017

Jing Si, Deme Sambo, James Frick, Ian Ferri

1. Downloaded data set and used Python to pre-process data from individual subdirectories into an EdgeList suitable for reading into R. Python code used to pre-process data is saved as untitled0.py within our zip file.

Assumptions during pre-processing:

- If email does not have data within To, From, and Date lines, it is not included in R analysis.
 - Only Enron employees were included in our data set. If an email address did not end with “@enron.com”, the email was not included in our data set.
 - Kept a running set of datetimes at which the Sender send an email to each Receiver. If our dataset already contains Sender X sent email to Receiver Y at datetime Z, it is ignored, if it is a new email, the event is added to our dataset. (This minimizes duplicates from having a record for the sent email and a record of the received email.)
 - Emails sent to several people using CC or BCC are treated as if the Sender sent an individual, separate email to each Receiver. In other words, there is no special weighting applied to whether a Receiver was in the To, CC, or BCC line.
2. Installed R, RStudio, and igraph package.
 3. By pre-processing the data, we created edgeList.csv (screenshot at right) to read the data into R. The csv contains only 3 variables, From = who sent the email, To = who received the email, and Weight = how many emails to Sender sent the Receiver. An index was created for the email addresses (e.g., 0 is dan.connally@enron.com, 1 is scott.dicke@enron.com, etc.)



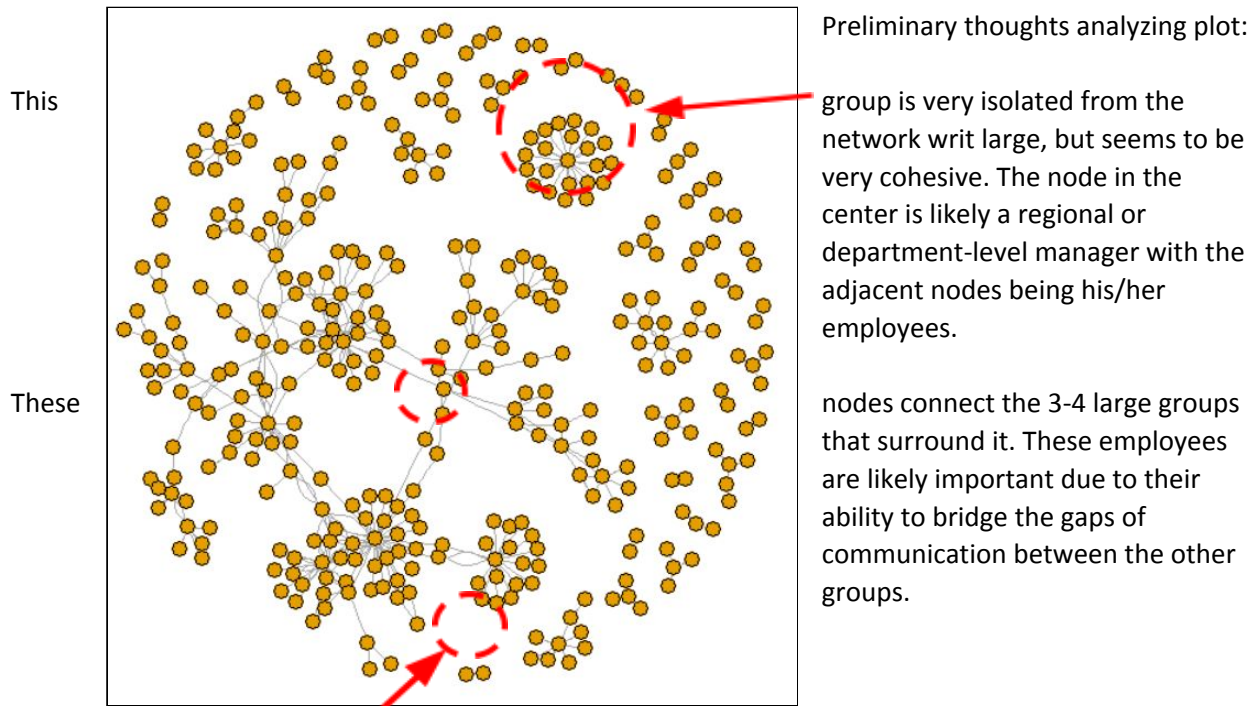
	A	B	C	D
1	From	To	Weight	
2	0	7216	1	
3	0	8896	1	
4	0	2557	1	
5	2	4093	1	
6	3	209	1	
7	3	3827	2	
8	3	8628	1	
9	3	2446	2	
10	3	1057	1	
11	5	2702	2	
12	5	4000	1	
13	7	360	1	
14	7	7372	1	
15	7	7	2	
16	7	1333	1	
17	8	1271	2	
18	8	2398	1	
19	9	2560	2	
20	9	1709	2	
21	9	5265	2	
22	9	2325	2	

To load the data, we read the edgeList.csv file into the `inp` variable. In the Environment pane, RStudio told us we had 67,012 observations with our 3 variables (To, From, Weight). We also read in the NamesToIndices.csv file into `emailNames` using the `read.csv` function. In the Environment pane, we now know we are dealing with 10,904 unique email addresses. Considering up to 10,904 unique vertices (nodes) and up to 67,012 edges (relationships), we simplified our data set for use with igraph using the `subset` function. To simplify our data, we left out any observation with a Weight less than 50 and left out any observation where From was equal to To. This brought our dataset down to 443 observations and more suitable for analysis with igraph.

Experimenting with igraph:

```
> # load from data frame, set directed=FALSE to de-clutter visual
> myGraph = graph_from_data_frame(simpler, directed = FALSE)
```

```
> # create a graph visual
> plot.igraph(myGraph, vertex.label = NA, vertex.size = 5)
```



The nodes around the perimeter of the graph with one edge between them are likely co-workers that communicate with one another frequently, but do not communicate often with others.

4. Exploring 10 functions within igraph package:
 - a. Basic descriptive analytics:

```
>
> gorder(myGraph)
[1] 362
> gsize(myGraph)
[1] 443
> edge_density(myGraph)
[1] 0.006779817
>
```

gorder counts the number of vertices (i.e., Enron employees in graph)

gsize counts the number of edges (i.e., relationships/lines of communication between employees)

edge_density provides a measure of density within the graph. A higher density means employees are more interconnected, communications are no “siloes” and the network can resist more link failures.

$edge_density = gsize / (gorder(gorder-1)/2)$ where $gsize = 443$ and $gorder = 362$

- b. Experimenting with single employee (#991)

E(myGraph)[from(“991”)] prints a list of employees that 991 is connected to (i.e., has sent at least 50 emails with). Based on the results, we know 991 was only connected to two

other nodes, 4093 and 5661. Therefore removing 991 will result in 1 less vertex and 2 less edges.

myGraphwithout991 <- delete_vertices(myGraph, "991") removes employee 991 from myGraph and saves to new graph called myGraphwithout991

Re-run ***gorder*** and ***gsize*** to confirm vertex (node) and edges have been removed from graph. Now we can confirm 1 vertex (node) and 2 edges (connections) were removed.

```
> E(myGraph)[from("991")]
+ 2/443 edges (vertex names):
[1] 991--4093 991--5661
>
> myGraphwithout991 <- delete_vertices(myGraph, "991")
> gorder(myGraphwithout991)
[1] 361
> gsize(myGraphwithout991)
[1] 441
> |
```

c. Measuring cliques, betweenness centrality, and degree

clique.number()

betweenness.estimate()

degree()

degree.distribution()

d. Testing a few other functions

walktrap.community()

