Homework 4 (CSC 495)

Kai Chung Ying

Spring 2017

Working with the Beer Advocate data

Bipartite networks and ego networks

```
library(knitr)
setwd("/Users/KevQuant/Desktop/Depaul/csc495/wk4/hwk4/hwk4_submit")
read_chunk("hwk4.R")
knitr::opts_chunk$set(echo = TRUE)
```

Step 1: Load libraries

```
# Load libraries

library("ggplot2")
# Must load other packages first before SAND
library("sand")
library("intergraph")
```

Step 2: Load the data

Load the "Cities" network provided in Pajek format. This is a bipartite network from 1999 of multinational companies and the cities where they had offices.

```
# Load edge and attribute data
setwd("/Users/KevQuant/Desktop/Depaul/csc495/wk4/hwk4")
g<-read_graph("cities.net",format="pajek")

#write_graph(g_cities,"cities.net",format="pajek")
```

Step 3: Determine which nodes are cities and which are companies.

```
# Determine cities or companies
#Check id names
head(V(g)$id,10)
```

```
##  [1] "Amsterdam"    "Atlanta"      "Bangkok"      "Barcelona"
##  [5] "Beijing"      "Berlin"       "Boston"       "Brussels"
##  [9] "Budapest"     "Buenos Aires"
```

```
tail(V(g)$id,10)
```

```
##  [1] "DW Dorsey & Whitney"
##  [2] "GJ Graham & James"
##  [3] "HH Hogan & Hartson"
```

```
##  [4] "JD Jones, Day, Reavis & Pogue"
##  [5] "MC Miller, Canfield, Paddock & Stone"
##  [6] "SQ Squire, Sanders & Dempsey"
##  [7] "WE Wilson, Elser, Moskowitz, Edelman & Dicker"
##  [8] "AO Allen & Overy"
##  [9] "CC Clifford Chance"
## [10] "FF Freshfields"
```

```
#Check Vertices type
V(g)[V(g)$type==FALSE]
```

```
## + 55/101 vertices:
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
## [24] 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
## [47] 47 48 49 50 51 52 53 54 55
```

```
V(g)[V(g)$type==TRUE]
```

```
## + 46/101 vertices:
##  [1]  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
## [18]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89
## [35]  90  91  92  93  94  95  96  97  98  99 100 101
```

```
#So that Cities Vertex with type "FALSE"
```

Step 4: Create a city-city projection

```
# Create a city-city projection
#Assign to Cities Variables with type is FALSE
g_cities<-bipartite_projection(g,which = "FALSE")
#Assign to Cities Variables with type is TRUE
g_companies<-bipartite_projection(g,which = "TRUE")
#Check cities id
head(V(g_cities)$id,10)
```

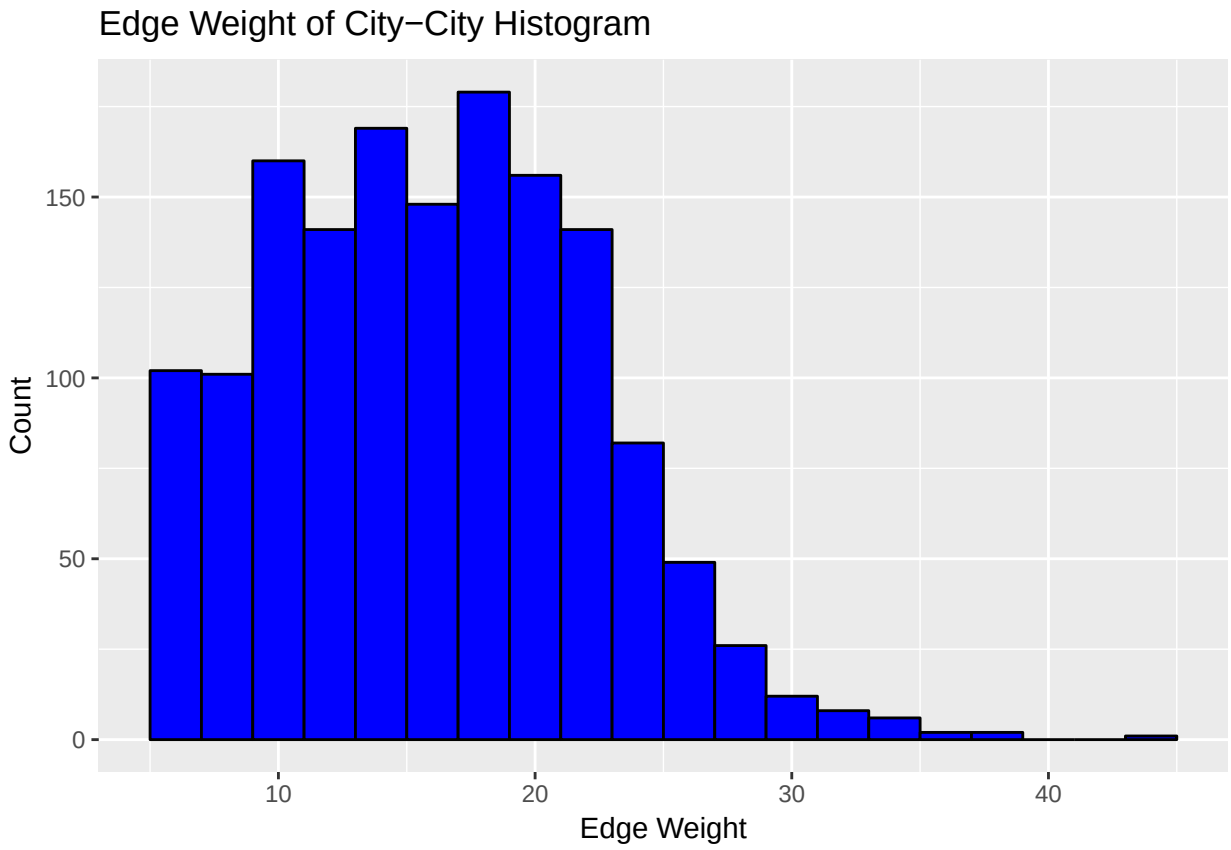
```
##  [1] "Amsterdam"    "Atlanta"      "Bangkok"      "Barcelona"
##  [5] "Beijing"      "Berlin"       "Boston"       "Brussels"
##  [9] "Budapest"     "Buenos Aires"
```

Step 5: Plot the distribution of edge weights

```
# edge weight
summary(g_cities)
```

```
## IGRAPH U-W- 55 1485 --
## + attr: id (v/c), weight (e/n)
```

```
g_cities_ewt_df<-data.frame(edge_weight=E(g_cities)$weight)
g1<-ggplot(g_cities_ewt_df,aes(x=edge_weight))
g1<-g1+geom_histogram(binwidth = 2,col="black",fill='blue')
g1<-g1+ggtitle("Edge Weight of City-City Histogram ")
g1<-g1+xlab('Edge Weight')
g1<-g1+ylab('Count')
print(g1)
```



Step 6: Set a weight threshold so that approximately 80% of the edges are removed (actually it will be 83%), then remove singletons (nodes of degree 0).

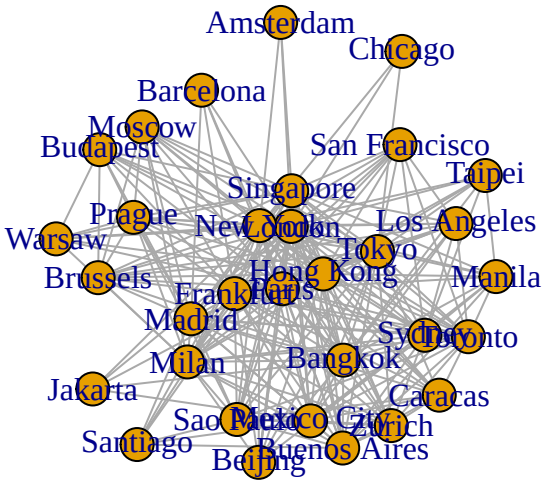
```
# Remove 83% of edges and singletons
#Check again the numbers edges exist in the graph
summary(g_cities)
```

```
## IGRAPH U-W- 55 1485 --
## + attr: id (v/c), weight (e/n)
```

```
#Check the threshold and remaining number of edges
E(g_cities)[E(g_cities)$weight>22]
```

```
## + 252/1485 edges:
##   [1]  1--25  1--37  1--47  3-- 5  3-- 8  3--10  3--16  3--19  3--22  3--25
##  [11]  3--26  3--27  3--28  3--30  3--32  3--37  3--39  3--47  3--49  3--50
##  [21]  3--51  3--52  4--16  4--25  4--27  4--37  4--39  5--16  5--19  5--25
##  [31]  5--32  5--37  5--39  5--47  5--49  5--51  8-- 9  8--16  8--19  8--25
##  [41]  8--26  8--27  8--32  8--35  8--37  8--39  8--40  8--47  8--51  9--16
##  [51]  9--19  9--25  9--27  9--37  9--39  9--40  9--47  9--53 10--11 10--16
##  [61] 10--19 10--25 10--27 10--28 10--30 10--32 10--37 10--39 10--43 10--44
##  [71] 10--47 10--49 10--51 10--52 10--55 11--16 11--19 11--25 11--27 11--30
##  [81] 11--37 11--39 11--43 11--51 11--52 11--55 12--25 12--37 12--51 16--19
##  [91] 16--25 16--26 16--27 16--30 16--32 16--35 16--37 16--39 16--40 16--42
## + ... omitted several edges
```

```
#Set the threshold @ Edge Weight <=22 (Removed 83% of Edges)
g_cities_2<-delete_edges(g_cities,E(g_cities)[E(g_cities)$weight<=22])
#Remove all the singletons(Degree=1)
g_cities_2<-delete_vertices(g_cities_2,degree(g_cities_2)==0)
#Review the graph after removing the edges and singletons
plot(g_cities_2,layout=layout_with_kk,vertex.label=V(g_cities_2)$id)
```



Step 7: Important: rename the "id" attribute, which is the name of the city, to the "label" attribute. Gephi will overwrite an "id" attribute on import.

```
# Rename attribute "id" to "label"
V(g_cities_2)$label<-V(g_cities_2)$id
g_cities_2<-delete_vertex_attr(g_cities_2,"id")
summary(g_cities_2)
```

```
## IGRAPH U-W- 32 252 --
## + attr: label (v/c), weight (e/n)
```

Step 8: Save the network in GraphML format

```
# Save the network in GraphML format
write_graph(g_cities_2,"cities.graphml",format="graphml")
```

Step 9: Import the network into Gephi

Step 10: In Gephi, make the following visualization:

layout: force directed (your choice) color: betweenness centrality node size: weighted degree edge size: weight (this is automatic)

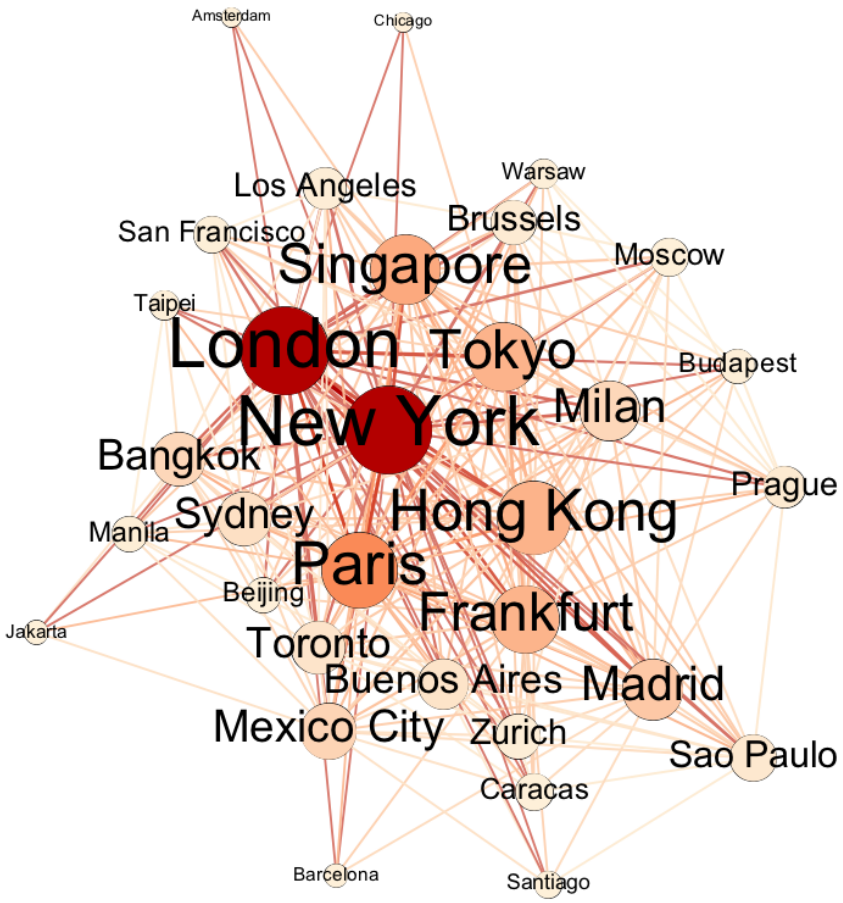

Amsterdam
Chicago
Warsaw
Los Angeles
Brussels
San Francisco
Singapore
Moscow
Taipei
London
Tokyo
Budapest
New York
Milan
Bangkok
Prague
Sydney
Hong Kong
Manila
Paris
Beijing
Frankfurt
Jakarta
Toronto
Buenos Aires
Madrid
Mexico City
Zurich
Sao Paulo
Caracas
Barcelona
Santiago

Figure 1:

Step 11: Import the png image generated from Gephi into your markdown file using the following syntax:

Step 12: Run the modularity operation. Adjust the "Resolution" parameter until you get exactly 4 clusters. There is a certain amount of randomness in this operation, so you won't necessarily get exactly the same clusters that are shown on the sample image.

Step 13: Make a second visualization as follows:

layout: circle layout (by modularity class), adjusted so the labels don't overlap color: modularity class size: weighted degree

Step 14: Include your second visualization in the markdown file.

Step 15: The clusters found by the modularity computation are mostly geographical, but some nodes appear to be in the "wrong" geographical group. Why do you think this might be the case? Recall that the edges exist because a large number of companies have offices in both cities. Answer this question in your HTML output.

Answer: Recall that the edges exist because a large number of companies have offices in both cities. In other words, the bigger size of circle implies the more large corporations have offices in those cities at the same time. Based on the plot, we could observse that every cluster includes various Cities where are supposed to located in different Regions.

The reason is the algroithm might probably NOT calculate the modularity by geographical. The algorithem might actually check on the similarity of the nodes' connections in order to classify the clusters. In our case, the algoritem might group the nodes based on the edge connections of the nodes(Cities) connecting to the Same set of Companies to form the cluster Community). For example, based on the above Circular plot, New York, London and Tokyo are grouped in the same community because they possess certain numbers of similarity on Connections of same set of Companies.

Also, When we generate the plot with modularity as color classifier, we could adjust the resolution in order to choose how many cluster we are lookng for (Need 4 in our case!!!). By adjusting the resolution, the algorithm would tune the clusters (Lower resolution comes up a finer result which means more clusters). So that, this process might force the algorithm to group some nodes with differnt geographical into the same cluster.

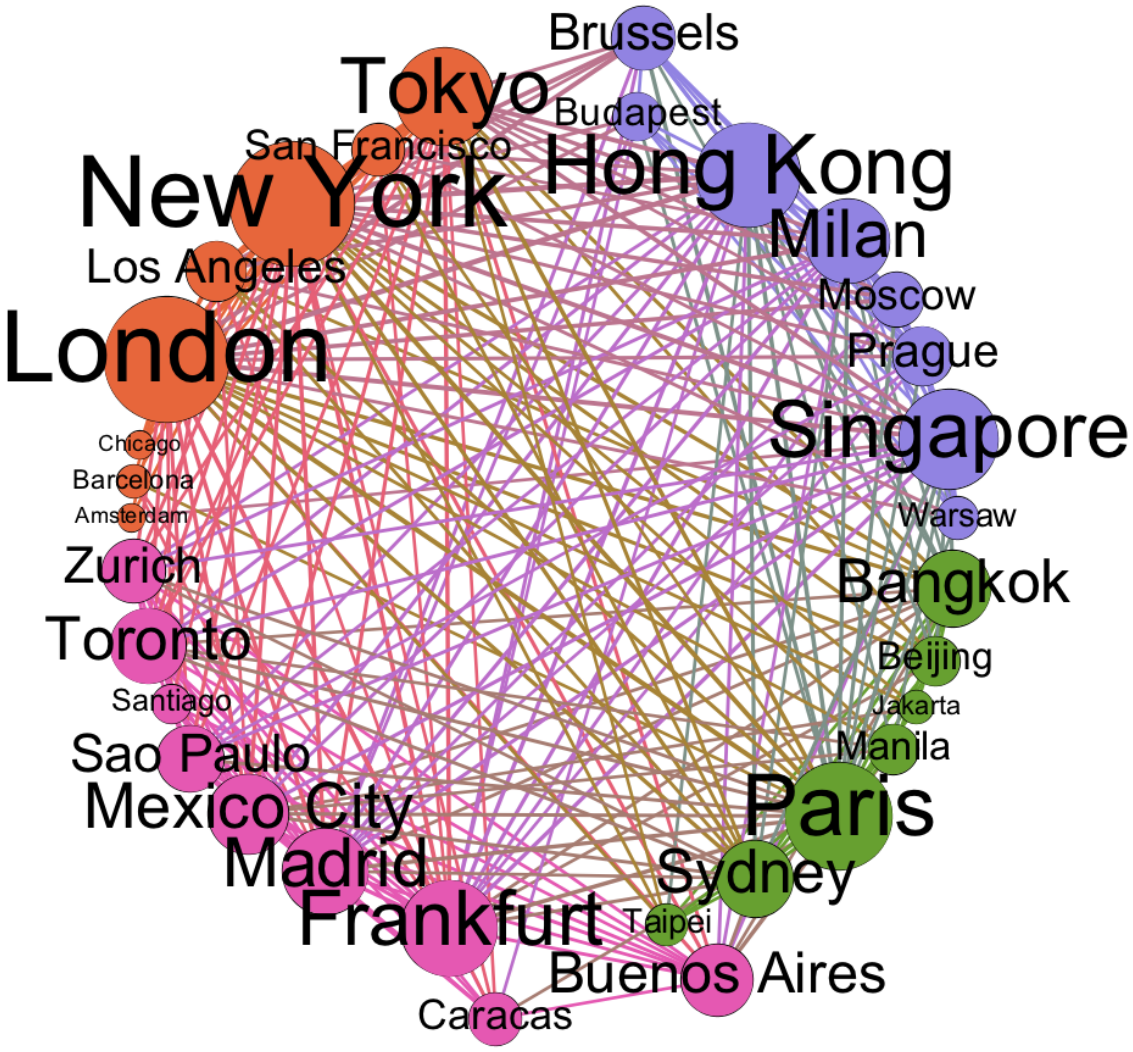

Brussels
Tokyo
Budapest
San Francisco
Hong Kong
New York
Milan
Los Angeles
Moscow
London
Prague
Singapore
Chicago
Barcelona
Amsterdam
Warsaw
Zurich
Bangkok
Toronto
Beijing
Santiago
Jakarta
Manila
Sao Paulo
Paris
Mexico City
Madrid
Sydney
Frankfurt
Taipei
Buenos Aires
Caracas

Figure 2: