Homework 2 (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/wk2/hwk2")
read_chunk("hwk2_2.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 *1 pt*

Load a bipartite network built from the "Beer Advocate" review data. For the purposes of this assignment, I selected only very active users (> 1000 reviews), chose the beers with the very top ratings (> 4.5 out of 5) and looked at just "American IPA" style beers. This gives a network of managable size with 278 nodes (131 beers and 147 users).

The data is in edgelist and attribute form, so we have to shape it into a network first. The two files are

- beer_edges.csv
- beer_attrs.csv

```
# Load edge and attribute data
setwd("/Users/KevQuant/Desktop/Depaul/csc495/wk2/hwk2")
beer_attrs<- read.csv("beer_attrs.csv",stringsAsFactors = TRUE)
beer_edges<-read.csv("beer_edges.csv",stringsAsFactors = FALSE)
```

Step 3: Use graph.data.frame to convert to graph *1 pt*

```
# Convert to graph
beer_network<-graph.data.frame(beer_edges,
                               directed = TRUE,
                               vertices = beer_attrs)
```

Step 4: Looking at attributes *1 pt*

List the vertex attributes and note that most attributes are for beers. We don't have attributes for users.

Use the `head` function to list the first 6 (head) beer names (not the numeric IDs)

```
# Examine attributes
list.vertex.attributes(beer_network)
```

```
## [1] "name"      "brewery"   "beer.name" "alcohol"   "type"
```

```
head(V(beer_network)$beer.name,6)
```

```
## [1] "Hoppin' To Heaven IPA"
## [2] "Point Defiance IPA"
## [3] "Founders Centennial IPA"
## [4] "Big Man Ale"
## [5] "Founders Harvest Ale"
## [6] "Sierra Nevada Anniversary Ale (2007-2009)"
```

Step 5: Projection *1 pt*

Compute the user-user and beer-beer projections and print the `summary` information for each.

```
# Projections
beer_network_b2b<-bipartite_projection(beer_network, which="TRUE")
beer_network_u2u<-bipartite_projection(beer_network, which="FALSE")
summary(beer_network_b2b)
```

```
## IGRAPH UNW- 131 2373 --
## + attr: name (v/c), brewery (v/c), beer.name (v/c), alcohol (v/n),
## | weight (e/n)
```

```
summary(beer_network_u2u)
```

```
## IGRAPH UNW- 147 5070 --
## + attr: name (v/c), brewery (v/c), beer.name (v/c), alcohol (v/n),
## | weight (e/n)
```

Step 6: Fixing the attributes *1 pt*

Remove the irrelevant attributes from the user-user network. Print the summary at the end. It should show only "name" and "weight" attributes. Use `delete_vertex_attr`.

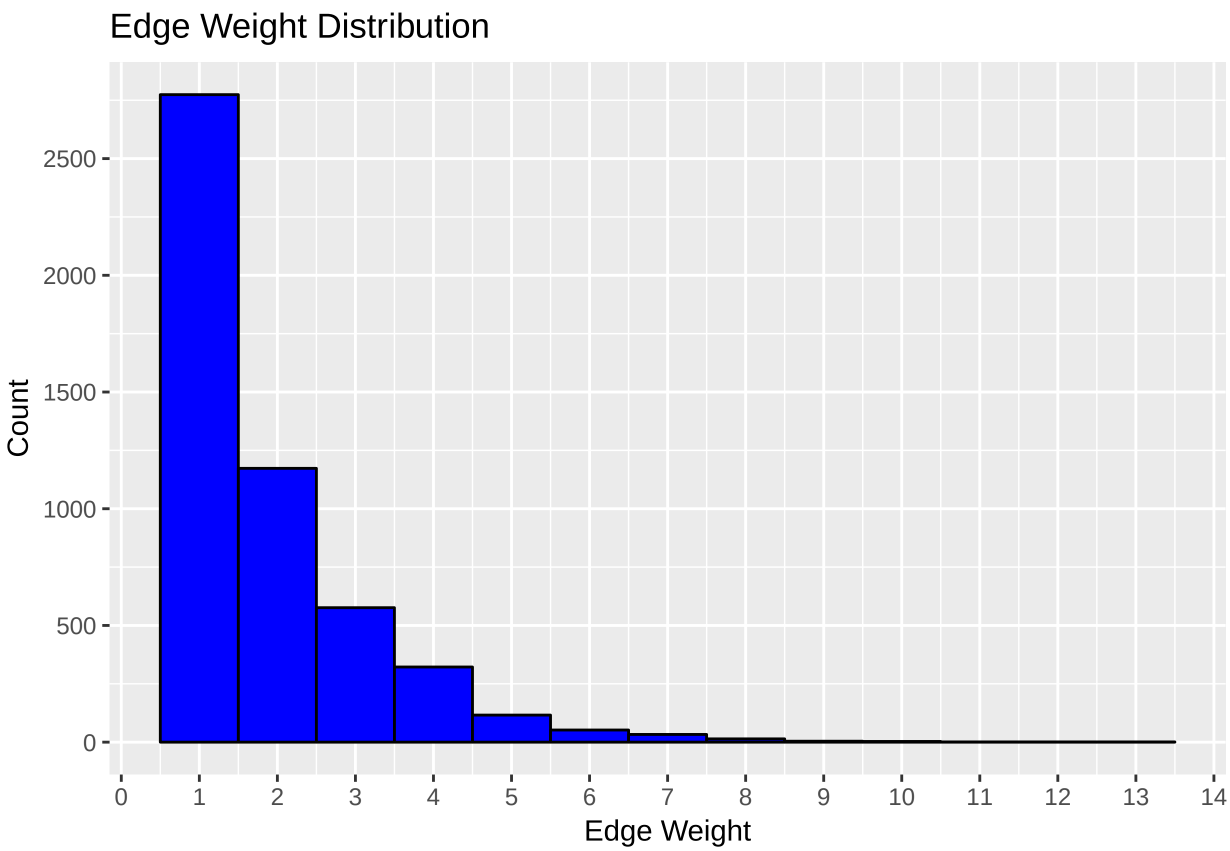
```
# Remove non-user attributes
beer_network_u2u<-delete_vertex_attr(beer_network_u2u,"brewery")
beer_network_u2u<-delete_vertex_attr(beer_network_u2u,"beer.name")
beer_network_u2u<-delete_vertex_attr(beer_network_u2u,"alcohol")
summary(beer_network_u2u)
```

```
## IGRAPH UNW- 147 5070 --
## + attr: name (v/c), weight (e/n)
```

Step 7: Plot the edge weight distribution *2 pt*

You can use either base plotting or ggplot. Label your plots appropriately (x and y axes, main title). Make sure that the whole distribution is included – set x axis limits correctly.

```
# Edge weight histogrqm
library(ggplot2)
par(mar=c(1,1,1,1))
g<-ggplot(data.frame(weight=E(beer_network_u2u)$weight),aes(x=weight))
g<-g+geom_histogram(binwidth = 1,col="black",fill="blue")
g<-g+xlab("Edge Weight")
g<-g+ylab("Count")
g<-g+ggtitle("Edge Weight Distribution")
g<-g+scale_x_continuous(breaks=seq(0,14,1))
g<-g+scale_y_continuous(breaks=seq(0,3000,500))
print(g)
```




```
#Alternative plotting option
#barplot(table(E(beer_network_u2u)$weight))
```

Step 8: Filter out edges of weight < 6 *2 pts*

As is typical with projections of bipartite networks, we'll filter out the low weight edges of which there are very many.

```
# Filter edges of weight less than 6
beer_network_u2u_filt<-delete.edges(beer_network_u2u,E(beer_network_u2u)[E(beer_network_u2u)$weight<6])
```

```r
#Showing the weight of the edges after filtering
table(E(beer_network_u2u_filt)$weight)
```

```
##
##  6  7  8  9 10 11 12 13
## 52 33 14  4  3  1  1  1
```

Step 9: Remove singletons (nodes of degree == 0) *1 pt*

Removing edges leaves some nodes disconnected from the network so we remove them also. Another way to do this would be calculate the largest (giant) component.

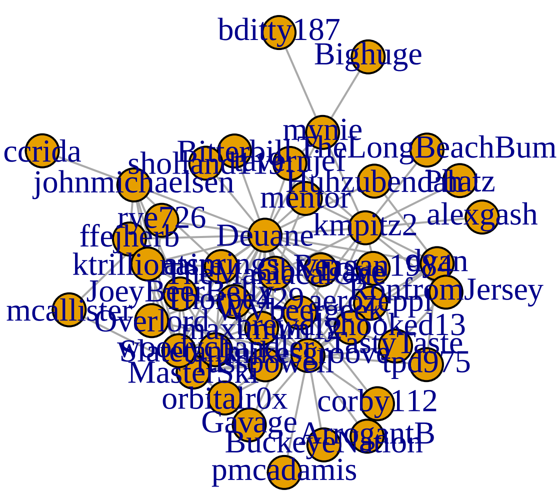
```r
# Remove singleton nodes
beer_network_u2u_filt2<-delete_vertices(beer_network_u2u_filt,
                                         V(beer_network_u2u_filt)[degree(beer_network_u2u_filt)==0])
```

Note: ***Use the filtered version of the network for the rest of the assignment.***

Step 10: Plot the network *1 pt*

The graph is now simplified enough that it can be visualized. Use the Kamada-Kawai layout: `layout=layout_with_kk`.

```r
# Plot
plot(beer_network_u2u_filt2,layout=layout_with_kk)
```

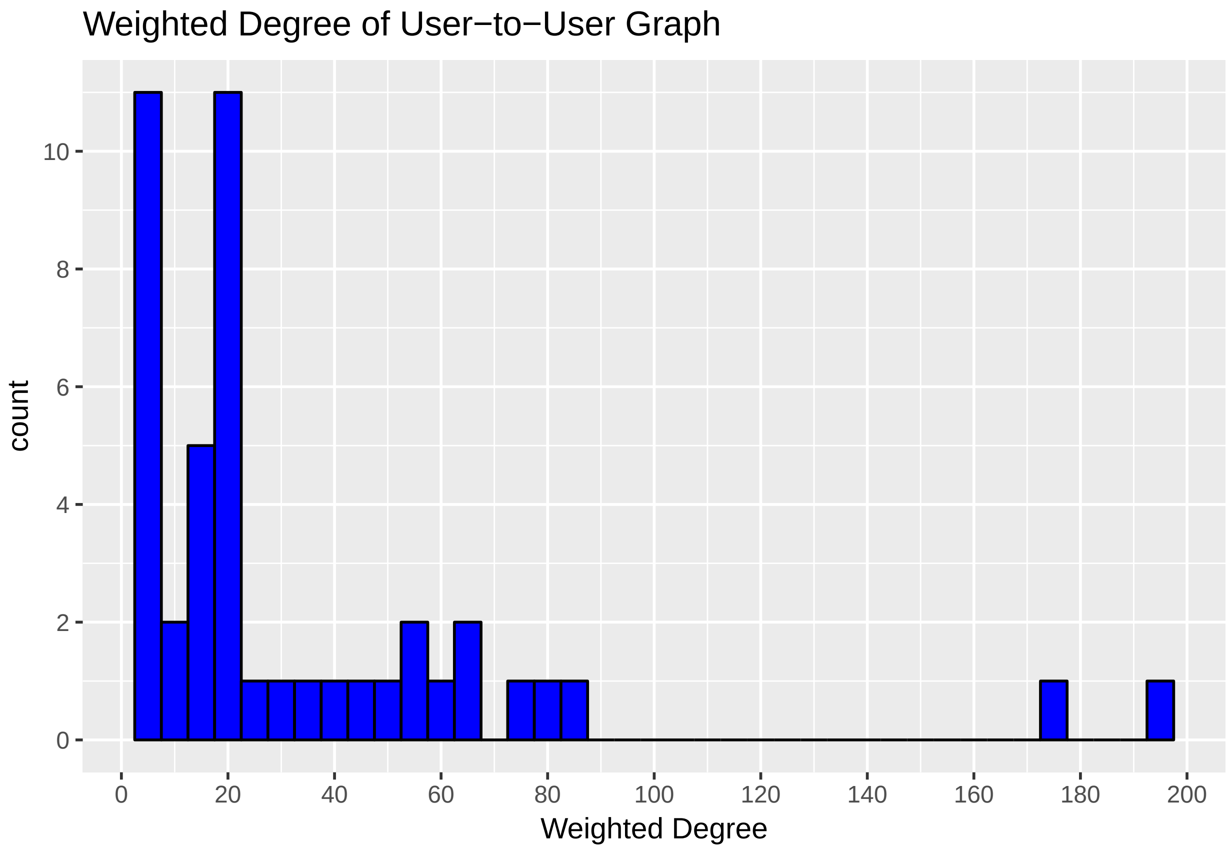


Step 11: Weighted degree *2 pt*

Compute a histogram of the weighted degree. (`graph.strength` function)

```r
# Weighted degree histogram
beer_network_u2u_filt2_weight_deg<-graph.strength(beer_network_u2u_filt2)
beer_network_u2u_filt2_weight_deg_df<-data.frame(weight_degree=beer_network_u2u_filt2_weight_deg)
g2<-ggplot(beer_network_u2u_filt2_weight_deg_df,aes(x=weight_degree))
g2<-g2+geom_histogram(binwidth = 5,col="black",fill="blue")
g2<-g2+ggtitle("Weighted Degree of User-to-User Graph")
g2<-g2+xlab("Weighted Degree")
```

```
g2<-g2+scale_x_continuous(breaks=seq(0,200,20))
g2<-g2+scale_y_continuous(breaks=seq(0,14,2))
print(g2)
```



Step 12: Find outliers ***2 pt***

Find the names of the two individuals with highest weighted degree (2 outliers)

```
# Find 2 highest weighted degree individuals
beer_network_u2u_filt2_weight_deg_sort<-data.frame(row.names=row.names(beer_network_u2u_filt2_weight_deg
                                                    [order(beer_network_u2u_filt2_weight_deg_df$weight_d
                                                    weight_degree=beer_network_u2u_filt2_weight_deg_df
                                                    [with(beer_network_u2u_filt2_weight_deg_df,order(wei

row.names(beer_network_u2u_filt2_weight_deg_sort)[c(1,2)]
```

```
## [1] "Deuane"      "mikesgroove"
```

Step 13: Compute the ego networks ***1 pt***

Compute the ego networks for these two users.

```
# Create ego networks
which(V(beer_network_u2u_filt2)$name=="Deuane")
```

```
## [1] 12
```

```r
which(V(beer_network_u2u_filt2)$name=="mikesgroove")
```

```
## [1] 26
```

```r
#Compute the ego network with order =1
egos<-make_ego_graph(beer_network_u2u_filt2,1,nodes = c(12,26))
```

Step 14: Plot the ego networks side by side *1 pt*

Use par(mfrow=c(1,2)) to get the side-by-side layout. Remember to switch back to the normal layout c(1,1) after.

```r
# Plot ego networks
#Extract the ego graph from the egos
deuane<-egos[[1]]
mikesgroove<-egos[[2]]

#Find their order on the graph
which(V(deuane)$name=="Deuane")
```

```
## [1] 5
```

```r
which(V(mikesgroove)$name=="mikesgroove")
```

```
## [1] 15
```

```r
#The order numbers were found 5 and 15 respectively,
#And then use these information to make the layout
deuane.lo<-layout_as_star(deuane,V(deuane)[5])
mikesgroove.lo<-layout_as_star(mikesgroove,V(mikesgroove)[15])

#plot row=1, col=2 format
par(mfrow=c(1,2))
plot(deuane,layout=deuane.lo)
plot(mikesgroove,layout=mikesgroove.lo)
```

Step 15: Question *3 pts*

In reducing the size of the network through edge and vertex filtering (steps 8 and 9) so that it is easier to visualize, what information about the original user-user network has been lost? What consequences does this have for our interpretation of the filtered version of the network?

```
### From step 8, we removed the edges with weight less than 6.
### From step 9, we removed the node with degree equals 0.

### For the above action, we might skip some network connection for further analysis.
### For the removal of weight less than 6 (Step 8), we might have skip some users with
### less beer preferences, but actually they might be favorite to different kind of beer.
### In other words, this group of users might be our potential users in the future.
### And discovering this area of users might lead to other area of beer industry.

### For those node with degree equals 0 (Step 9.) which takes up 102 out of 147 vertices is
### about two-third of the overall vertices. This isolated group is actually a huge resources for us.
### We could definitly perform another analysis to discover why they don't have any edge connection
### with other vertices OR they might actually have NO any preference from the beer list. This might
### be able to help the prove the beer category/ marketing stragegy / district investment ratio.
```