Project 4: A Music Recommendation System with Social Context

James Hamski (james.hamski@spsmail.cuny.edu), Vuthy Nguy (vuthy.nguy@spsmail.cuny.edu)

July 10, 2016

```
library(readr)
library(tidyr)
library(dplyr)
library(ggplot2)
library(knitr)
library(igraph)
```

1 User-Artist Data

Reading in needed datasets. Artist information will be joined to user-artist dataset for readability

```
user.artists.pairwise <- read_delim('hetrec2011-lastfm-2k/user_artists.dat', delim = "\t")

artists <- read_delim('hetrec2011-lastfm-2k/artists.dat', delim = "\t") %>% select(id, name)
```

```
## Warning: 1 parsing failure.
##  row  col           expected actual
## 1678 name delimiter or quote
```

1.1 Data Preparation

Converting from Artist IDs to Artist Names for easier interpretation.

```
colnames(artists) <- c("artistID", "name")

#note: one duplicate artist
#artists$name[duplicated(artists$name)==TRUE]
user.artists.pairwise <- inner_join(user.artists.pairwise, artists, by="artistID")

user.artists.pairwise <- select(user.artists.pairwise, -artistID)

user.artists.pairwise$name <- strtrim(user.artists.pairwise$name, 100)

user.artists.pairwise[16060,]$name <- NA

#ggplot(user.artists.pairwise, aes(x = weight)) + geom_density() + theme_bw() + scale_x_log10()
```

The user-artist pairwise record contains multiple rows for a single artist. In order to understand the distribution of listens for artists, we group and sum for each artist.

```
artist.listens <- user.artists.pairwise %>%
  group_by(name) %>%
  summarise(total.listens = sum(weight))
```

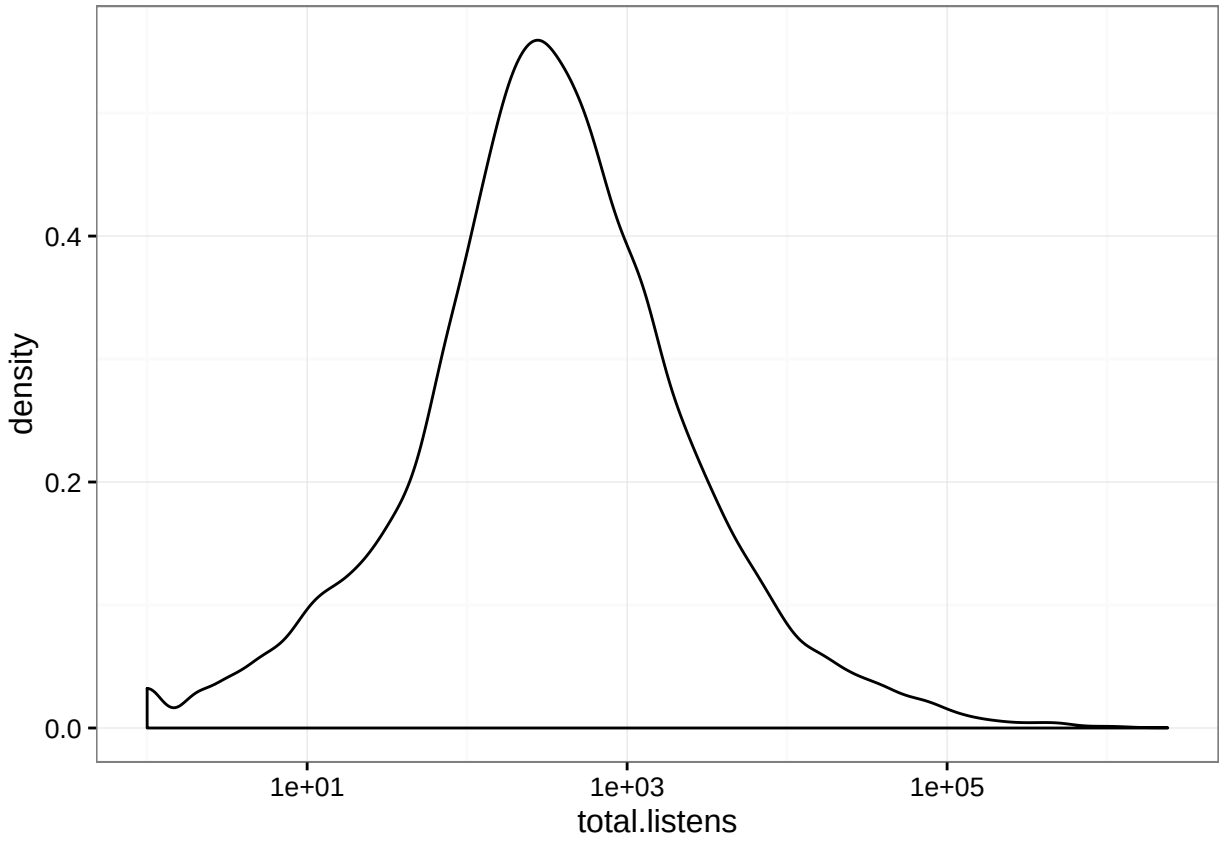
1.2 Exploratory Data Analysis

The median number for listens is 324 - quite a low number! The highest number of listens is Britney Spears with 2.3 million listens. We believe this dataset could be well modeled with a lognormal distribution.

```
summary(artist.listens$total.listens)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1     107     324    3858    1086 2393000
```

```
ggplot(artist.listens, aes(x = total.listens)) + geom_density() + theme_bw() + scale_x_log10()
```



Looking at the top 10 artists it is important to note that Britney Spears has nearly twice as many listens as the next most popular artist, Depeche Mode.

```
artist.listens %>% arrange(desc(total.listens)) %>% top_n(10) %>% kable()
```

```
## Selecting by total.listens
```

name	total.listens
Britney Spears	2393140
Depeche Mode	1301308
Lady Gaga	1291387
Christina Aguilera	1058405
Paramore	963449
Madonna	921198
Rihanna	905423
Shakira	688529
The Beatles	662116
Katy Perry	532545

We have decided to limit our recommender system to artist who have more than 900 total listens.

```
listens.cutoff <- 900
artist.cutoff <- artist.listens %>% filter(total.listens >= listens.cutoff)
```

1.3 User-Item Matrix Construction

Using the TidyR library the pairwise data was converted into a sparse matrix.

```
user.artists.matrix <- user.artists.pairwise %>%
  filter(name %in% artist.cutoff$name) %>% #filter artists with less than num_listens listens
  spread(key = name, value = weight) %>%
  distinct() %>%
  as.matrix()

rownames(user.artists.matrix) <- user.artists.matrix[,1]
user.artists <- user.artists.matrix[,-1]
dim(user.artists)
```

```
## [1] 1883 4608
```

1.3.1 Dealing with NAs

We tried two ways of dealing with NAs - imputing with column averages, and replacing with 0s.Note that due to the sparsity of the dataset, for most artists the column mean rounds to zero. In addition, the dataframe with column means was centered and scaled around 0 using the scale function. This gives us two views of the same dataset- one with NAs replaced with zeros which is relatively unmodified, and another with column means replacing NAs and normalized listen counts.

```
# http://stackoverflow.com/questions/25835643/replacing-missing-values-in-r-with-column-mean
user.artists.ave <- user.artists
for(i in 1:ncol(user.artists.ave)){
  user.artists.ave[is.na(user.artists.ave[,i]), i] <- mean(user.artists.ave[,i], na.rm = TRUE)
}

user.artists.ave <- scale(user.artists.ave)

user.artists[is.na(user.artists)] <- 0
```

2 Social Network Data

```
user.friends <- read_delim('hetrec2011-lastfm-2k/user_friends.dat', delim = "\t")
```

```
user.friends$friend <- 1
user.friends.matrix <- user.friends %>% spread(key = userID, value = friend)
user.friends.matrix[is.na(user.friends.matrix)] <- 0

rownames(user.friends.matrix) <- user.friends.matrix$friendID
user.friends.matrix <- user.friends.matrix[,-1]
dim(user.friends.matrix)
```

```
## [1] 1892 1892
```

```
user.graph <- graph.adjacency(as.matrix(user.friends.matrix), mode = "undirected")

#plot(user.graph)
```

2.1 Graph Analysis

The proportion of present edges from all possible edges in the network.

```
edge_density(user.graph)
```

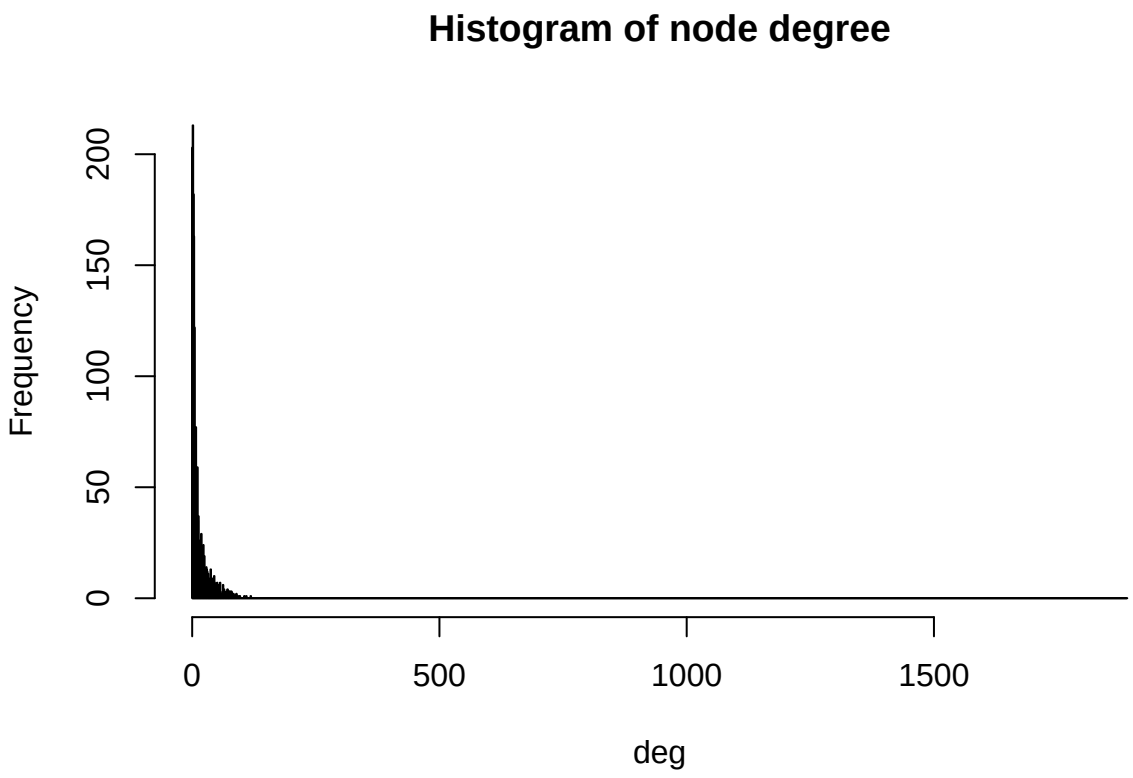
```
## [1] 0.007108893
```

A network diameter is the longest geodesic distance (length of the shortest path between two nodes) in the network.

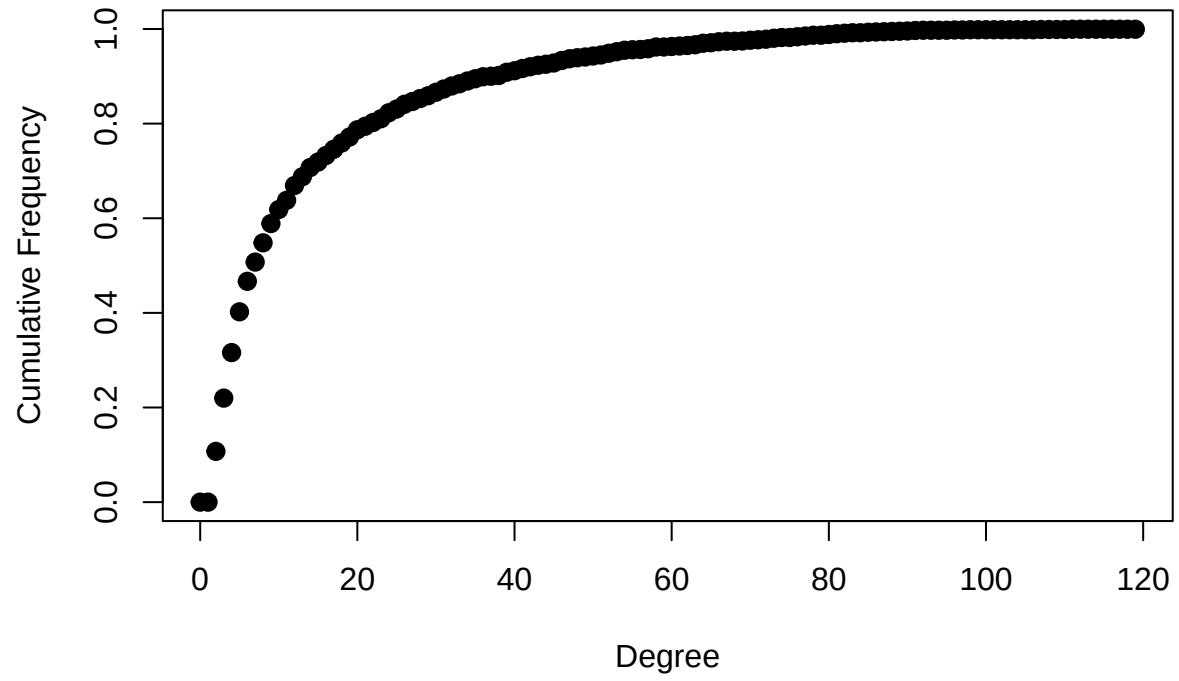
```
diameter(user.graph)
```

```
## [1] 9
```

```
# http://kateto.net/networks-r-igraph
deg <- degree(user.graph, mode="all")
#plot(user.graph, vertex.size=deg*3)
hist(deg, breaks=1:vcount(user.graph)-1, main="Histogram of node degree")
```




```
deg.dist <- degree_distribution(user.graph, cumulative=T, mode="all")
plot( x=0:max(deg), y=1-deg.dist, pch=19, cex=1.2, xlab="Degree", ylab="Cumulative Frequency")
```




```
mean_distance(user.graph, directed=F)
```

```
## [1] 3.518552
```

2.1.1 Communities

```
#cliques(user.graph) # list of cliques
#sapply(cliques(user.graph), length) # clique sizes
largest_cliques(user.graph) # cliques with max number of nodes
```

```
## [[1]]
## + 10/1892 vertices, named:
##  [1] 1343 499  534  31   575  859  211  1173 1213 1164
##
## [[2]]
## + 10/1892 vertices, named:
##  [1] 1343 499  534  31   575  859  211  1173 1213 306
##
## [[3]]
## + 10/1892 vertices, named:
##  [1] 1343 499  534  31   575  859  210  1173 1213 1164
##
## [[4]]
## + 10/1892 vertices, named:
##  [1] 1343 499  534  31   575  859  210  1173 1213 306
##
## [[5]]
## + 10/1892 vertices, named:
##  [1] 1343 499  99   859  31   575  211  1173 1213 1164
##
## [[6]]
## + 10/1892 vertices, named:
##  [1] 1343 499  99   859  31   575  211  1173 1213 306
##
## [[7]]
## + 10/1892 vertices, named:
##  [1] 1343 499  99   859  31   575  210  1173 1213 1164
##
## [[8]]
## + 10/1892 vertices, named:
##  [1] 1343 499  99   859  31   575  210  1173 1213 306
```

```
ceb <- cluster_edge_betweenness(user.graph)
plot(ceb, user.graph)
```