Computational Network Analysis - Exercise 4

Tobias Kloht, ID: 4596192

February 24, 2014

1

Please load the dataframe provided in the first homework and filter the first three columns (gender, age, game). Check the dimension of your dataset.

```
data_min = data[c(1:3)]  # restrict on columns 1 to 3
dim(data_min)  # show number of rows and columns

## [1] 3246    3
```

Now select only those rows that contain a valid value (i.e., no NAN values or empty cells should be contained in your dataframe). Check again the dimension of your dataframe.

```
data_min[data_min == ""] = NA  # replace empty fields (=='') with NA
data_min = na.omit(data_min)  # omit all rows with NA values
dim(data_min)  # show number of rows and columns

## [1] 3115    3
```

Now, aggregate all players that have the same age, gender, and played the same game. For example, your dataframe contains some players who are 18 years old and male and played WoW. Aggregate those to one person but keep their number in an additional column.

```
# add column number to count occurrences. init with 1
data_min["number"] = 1
# aggregate rows with equal values
# sum aggregated rows in number column
aggregateData = aggregate(number ~ Gender + Age + Game,
                          FUN = sum,
                          data=data_min)
```

Your (new) dataframe should follow the following structure:

personId *(corresponds to the aggregated people) for example m18WoW or f20WoW*

game *(corresponds to the game) for example WoW*

number *(corresponds to the number of players to whom the attributes apply)*

```
# replace "female" and "male with "f" and "m" respectively
levels(aggregateData$Gender)

## [1] ""       "female" "male"

levels(aggregateData$Gender)[3] = "m"
levels(aggregateData$Gender)[2] = "f"

# concatenate Gender, Age and Game columns
aggregateData$personId = do.call(paste,
    c(aggregateData[c("Gender", "Age", "Game")],
    sep = ""))

# drop unnecessary columns and reorder
aggregateData = aggregateData[,c(5,3,4)]

str(aggregateData)

## 'data.frame': 516 obs. of  3 variables:
##  $ personId: chr  "m20AC1" "m21AC1" "m32AC1" "f35AC1" ...
##  $ Game    : Factor w/ 12 levels "","AC1","AO",..: 2 2 2 2 2 2 2 2 3 3 ...
##  $ number  : num  1 1 1 1 2 1 1 1 1 3 ...
```

Now, use this dataframe to create a bipartite network.

```
# create graph from edgelist
gameGraph = graph.edgelist(as.matrix(aggregateData[, c(1, 2)]))
# set type attribute on vertex
V(gameGraph)$type = V(gameGraph)$name %in% get.edgelist(gameGraph)[, 1]
```

Additional question: Do you have an idea how to keep the number of gamers in each aggregate in your network model?

```
# add number of gamers as edge weight
E(gameGraph)$weight = as.numeric(aggregateData[, 3])
```

Please check the number of vertices and edges in your network.

```
# number of vertices
vcount(gameGraph)

## [1] 527
```

```r
# number of edges
ecount(gameGraph)

## [1] 516
```

Now create an unipartite network. Please make your projection weighted, by giving each edge between two vertices in the projected network a weight equal to the number of common games the vertices share. Please reveal again the number of vertices and edges in your network.

```r
# create bipartite projection
gameGraphProjection = bipartite.projection (
                      gameGraph,multiplicity = TRUE)$proj2

#number of vertices
vcount(gameGraphProjection)

## [1] 516

#number of edges
ecount(gameGraphProjection)

## [1] 14639
```