Homework 1 (CSC 495)

Kai Chung, Ying

Winter 2017

Some visualizations of the Dining data

```
library(knitr)
setwd('/Users/KevQuant/Desktop/Depaul/csc495/wk1/hwk1')

read_chunk("hwk1.R")
knitr::opts_chunk$set(echo = TRUE)
```

Load packages

```
# Load packages

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

Load the necessary libraries and data

Note that I include the chunk options results="hide", etc. because loading libraries produces a lot of output that we don't want on our page. You will need to install all of these packages for this part to work.

```
# Load packages

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

Loading the data

Next we need to load the data file. This file is included in the homework zip file. You will need to set the path appropriately.

```
# Load data

path <- ('/Users/KevQuant/Desktop/Depaul/csc495/wk1/hwk1')
setwd(path)
dining <- read.graph("dining.net", format="pajek")
summary(dining)
```

```
## IGRAPH D-W- 26 52 --
## + attr: id (v/c), x (v/n), y (v/n), z (v/n), weight (e/n)
```

Pointers to help

You will find it useful to refer to the following help articles in doing this assignment.

- igraph-vs-attributes
- igraph-es-attributes
- igraph.plotting
- degree
- graph.strength

Graded portion *(14 points + 2 extra credit)*

Step 1: Introduction *(1 pt)*

- Go to Slack. Upload an actual picture of yourself as a profile picture. (Not your dog or your favorite Pokemon.) Thanks.
- Post to the "#introductions" topic:
  - Your name
  - Your degree program
  - What types of networks you are interested in
- Post to the "#smallworldstory" topic:
  - A "small world" story: a situation in which an unexpected social connection manifested itself.

Step 2: Names *(1 pt)*

Get the names of the students from the nodes in the dining network.

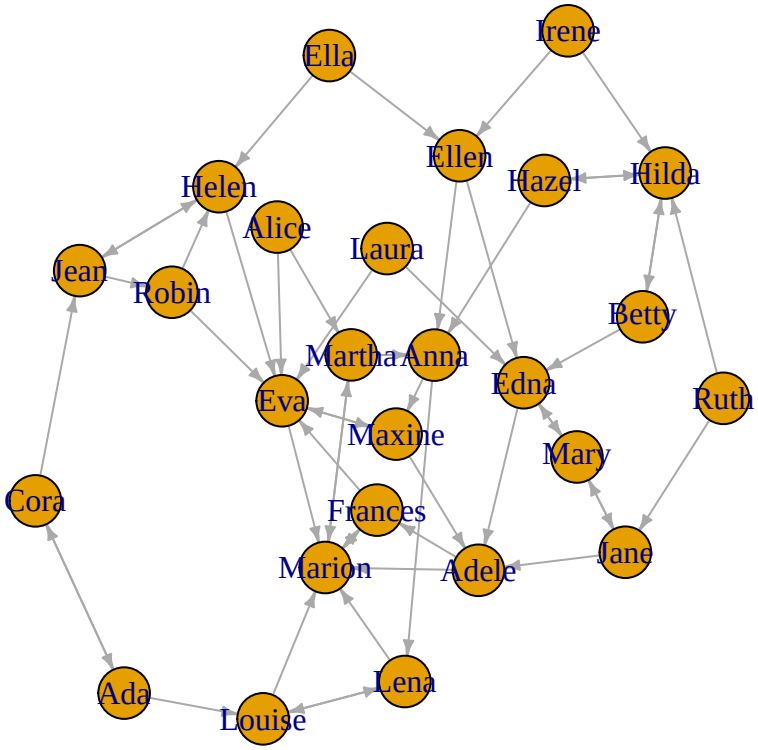
```
# Student names
#student_name<-V(dining)$id
V(dining)$id
```

```
##  [1] "Ada"     "Cora"    "Louise"  "Jean"    "Helen"   "Martha"  "Alice"
##  [8] "Robin"   "Marion"  "Maxine"  "Lena"    "Hazel"   "Hilda"   "Frances"
## [15] "Eva"     "Ruth"    "Edna"    "Adele"   "Jane"    "Anna"    "Mary"
## [22] "Betty"   "Ella"    "Ellen"   "Laura"   "Irene"
```

Step 3: Visualization 1 *(2 pts)*

Create a simple, but not ugly visualization of the dining network using base plotting in igraph (`plot`). It should include the names for each vertex and arrows indicating the direction of the network.

```
# Plot network
# Adjust the edge arrow size for less ugliness
plot(dining,main="dining network",edge.arrow.size=0.4,vertex.label=V(dining)$id)
```



Step 4: Calculate in-degree ***(2 pts)***

Calculate in-degree for the nodes (`degree` function with `mode="in"`) and display a summary (`summary` function) of the in-degree data. (Hint: The mean should be 2 and the max should be 6.)

```
# In-degree
deg<-degree(dining,mode='in')
deg
```

```
##  [1] 1 1 2 2 3 2 0 1 6 2 2 1 4 2 6 0 4 3 2 3 2 1 0 2 0 0
```

```
summary(deg)
```

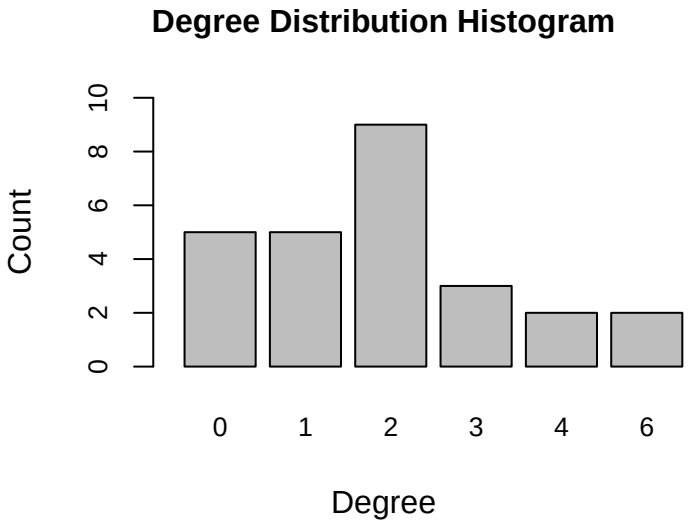
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    1.00    2.00    2.00    2.75    6.00
```

Step 5: Visualization 2 ***(2 pts)***

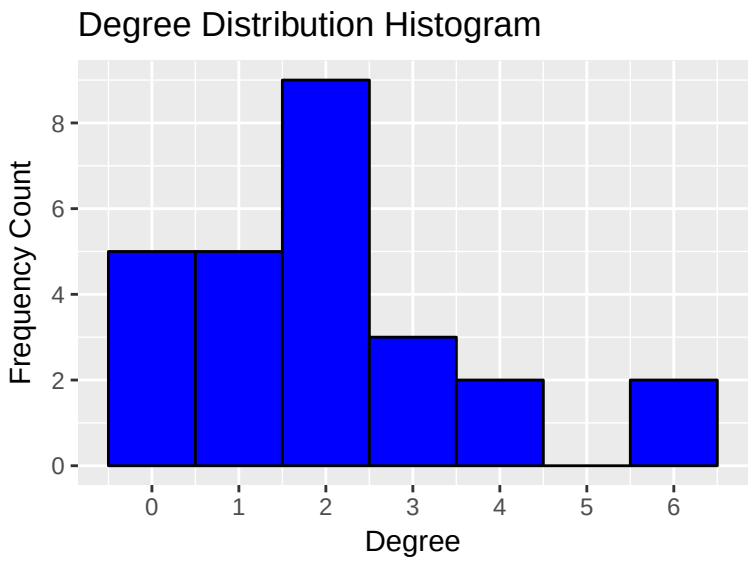
Plot the in-degree distribution in a histogram. You can use either base plotting or ggplot to do this.

```
# Degree distribution histogram
par(mar=c(4,4,4,4),cex.axis=0.8,cex.main=1,cex.lab=1)
barplot(table(deg),
        xlab="Degree",
        ylab = "Count",
        main="Degree Distribution Histogram",
```

```
        ylim = c(0,10),
        xlim = c(0,6))
```




```
# GGPlot version (optional)
g<-ggplot(data.frame(deg),aes(x=deg))
g<-g+geom_histogram(binwidth = 1,col="black",fill="blue")
g<-g+scale_y_continuous(breaks = seq(0,8,2))
g<-g+scale_x_continuous(breaks = seq(0,8))
g<-g+xlab("Degree")
g<-g+ylab("Frequency Count")
g<-g+ggtitle("Degree Distribution Histogram")
print(g)
```



Step 7: Computing edge weights ***(1 pt)***

Next we will add appropriate edge weights to our calculations. There are weights on the edges, but they are the values 1 and 2, representing first and second choice. The value 1 should be a **stronger** tie and the value 2 a **weaker** one.

Create a new edge attribute `wt` that is 1 when the `weight` attribute is 1 and 0.5 when the `weight` attribute is 2. You can use `ifelse` or arithmetic calculation to do this.

```
# Calculate wt attribute
w<-E(dining)$weight
w[w==2]<-0.5
w[w==1]<-1
E(dining)$wt<-rep(0,vcount(dining))
E(dining)$wt<-w
E(dining)$wt
```

```
##  [1] 0.5 1.0 1.0 0.5 1.0 0.5 1.0 0.5 1.0 0.5 0.5 1.0 1.0 0.5 1.0 0.5 1.0
## [18] 0.5 1.0 0.5 1.0 0.5 1.0 0.5 1.0 0.5 1.0 0.5 1.0 0.5 1.0 0.5 1.0 0.5
## [35] 1.0 0.5 1.0 0.5 1.0 0.5 1.0 0.5 1.0 0.5 1.0 0.5 1.0 0.5 1.0 0.5 1.0
## [52] 0.5
```

Step 8: Compute weighted degree *(1 pt)*

The `graph.strength` function computes weighted degree values. Use this function to compute weighted in-degree using the `wt` attribute computed in Step 7.

(Hint: the summary should show a mean of 1.5 and a max of 5.)

```
# Compute weighted degree
weighted.degree<-graph.strength(dining,mode = "in",weight=E(dining)$wt)
weighted.degree
```

```
##  [1] 1.0 1.0 1.0 1.5 2.0 1.5 0.0 0.5 4.0 2.0 1.0 0.5 3.0 1.5 5.0 0.0 3.0
## [18] 2.5 1.5 2.5 1.5 1.0 0.0 1.5 0.0 0.0
```

```
summary(weighted.degree)
```

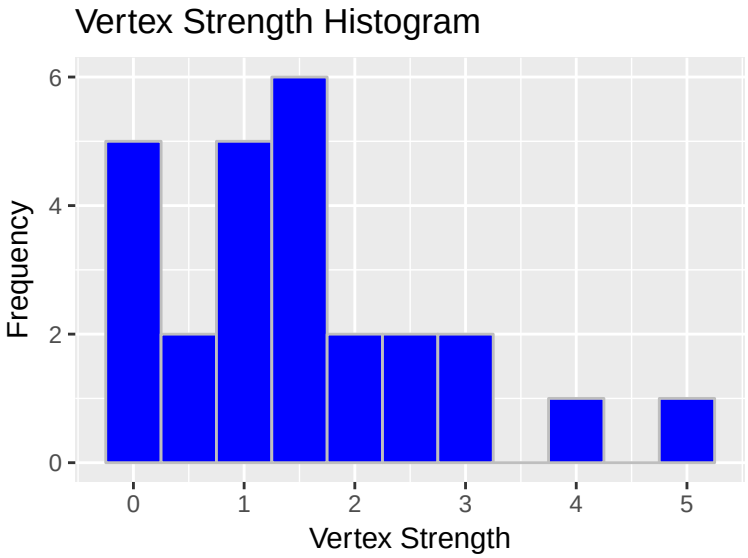
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.000   0.625   1.500   1.500   2.000   5.000
```

```
weighted.degree2<-weighted.degree
#Reset the Zero weighted.degree to 0.5 for better visualization plot
weighted.degree2[weighted.degree2==0.5]<-0.8
weighted.degree2[weighted.degree2==0.0]<-0.5
```

Step 9: Visualization 4 *(2 pts)*

Weighted degree will be a real number, not an integer. Use a binwidth of 0.5 and make sure that the limits on X axis are correct.

```
# Visualization
g<-ggplot(data.frame(weighted.degree),aes(x=weighted.degree))
g<-g+geom_histogram(binwidth = 0.5,color="grey",fill="blue")
g<-g+xlab("Vertex Strength")
g<-g+ylab("Frequency")
g<-g+ggtitle("Vertex Strength Histogram")
g<-g+scale_x_continuous(breaks=seq(0,5,1))
print(g)
```



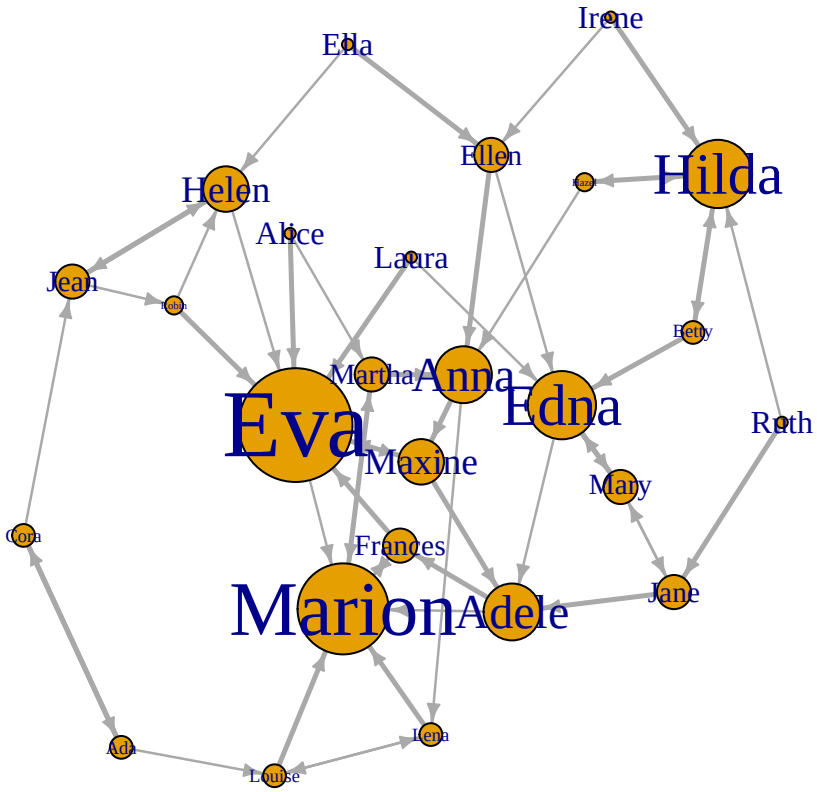
Extra credit: Weighted plot ***(2 pts)***

Produce a graph plot in which the nodes are sized as a function of weighted in-degree. (Use the `size` parameter to the `plot` function.) The label should also be sized by weighted degree. (The `label.cex` parameter does this.) Edge width should be function of the edge weight, using the `edge.width` parameter.

Note that you are striving for readability in the visualization. Some nodes have in-degree of zero – it is not a good idea for these to have zero size vertices and labels. You will need to compute a function of the in-degree value to get appropriate sizes and this function will be different for node size and label size.

A sample visualization is included in the zip file for inspiration.

```
# Visualization with weighted degree and weighted edges
plot(dining,
     rescale=TRUE,
     ylim = c(-0.6,0.6),
     xlim = c(-1.1,1.3),
     edge.arrow.size=weighted.degree2*0.5,
     #vertex size with zero values were adjusted at the section #8.
     vertex.size=weighted.degree2*6,
     edge.width=E(dining)$wt*2.5,
     vertex.label.cex=weighted.degree*0.6,
     vertex.label=V(dining)$id)
```


Irene
Ella
Ellen
Hilda
Helen
Alice
Laura
Jean
Betty
Martha
Anna
Edna
Eva
Ruth
Maxine
Mary
Cora
Frances
Marion
Adele
Jane
Ada
Louise
Lena