2017

Assignment 4

CS 432

MIRANDA SMITH

Question 1

Determine if the friendship paradox holds for Dr. Nelson's Facebook account. Compute the mean, standard deviation, and median of the number of friends that my friends have. Create a graph of the number of friends (y-axis) and the friends themselves, sorted by number of friends (x-axis).

I created a python program to collect all of the friend counts for each friend from the 'mln.graphml' file. I ignore the friends that didn't have a friend count, that way they wouldn't interfere with the calculations of mean, standard deviation, and median.

```
from pygraphml import Graph
from pygraphml import GraphMLParser
import statistics as stat

outfile = open("facebookFriendCount.txt", 'w')
parser = GraphMLParser()
g = parser.parse("mln.graphml")
nodes = g.nodes()

friendcounts = []
for friend in range(0,len(nodes)):
    try:
        outfile.write(str(nodes[friend]['friend_count']) + "\n")
        friendcounts.append( int (nodes[friend]['friend_count']))
        #not all friends will have friend count available
    except KeyError:
        continue

count = len(friendcounts)
mean = stat.mean(friendcounts)
std = stat.pstdev(friendcounts)
median = stat.median(friendcounts)

print("mean: %.1f" % mean)
print("std: %.1f" % std)
print("median: %.1f" % median)

outfile.write(str(count) + "\n")
outfile.close()
```

The output of this program was the mean, standard deviation, and median along with a file containing all of the friend counts including Dr. Nelson's as the last entry.

```
>>>
 RESTART: C:\U
bookParser.py
mean: 359.0
std: 370.4
median: 266.5
```

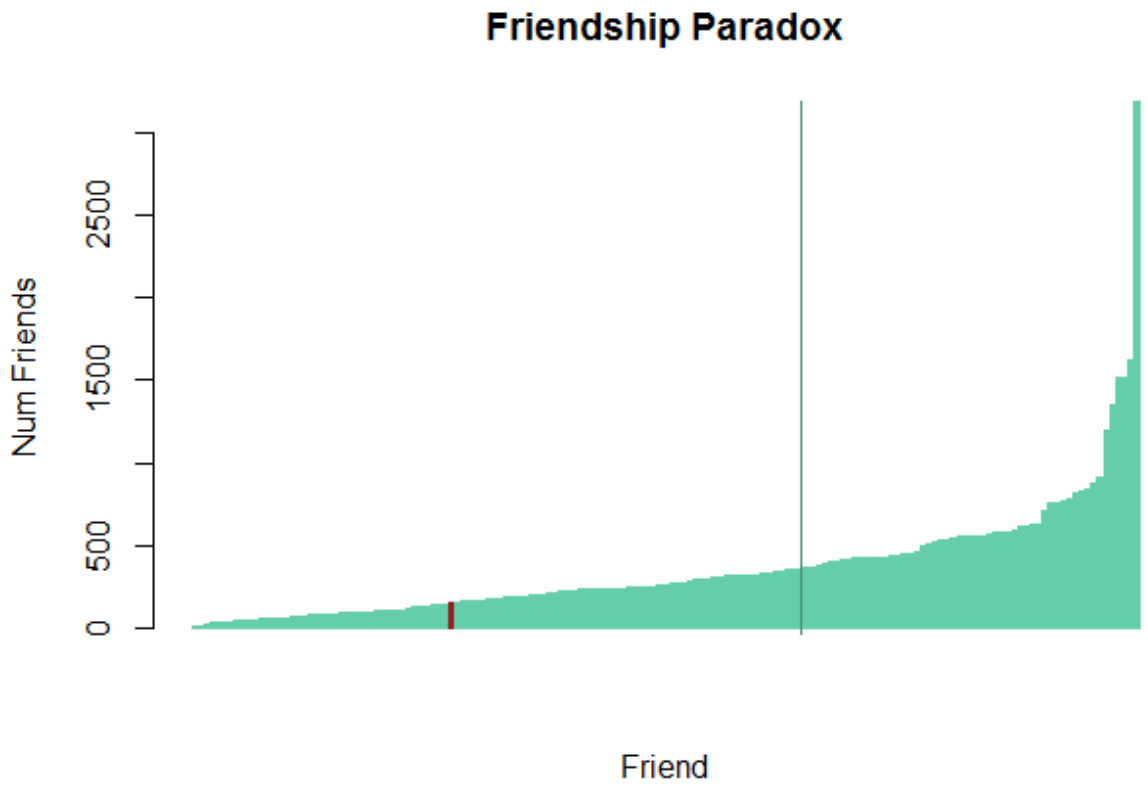
I created a program in R to make a histogram with Dr. Nelson's entry highlighted in a dark red color with a line to mark the mean.

```
friendcounts <- read.table("C:/Users/bitto/Documents/GitHub/cs532-s17/assignments/a4-solution/facebookFriendCount.txt")

orderedfriendcounts <- sort(friendcounts$V1)
mn <- mean(orderedfriendcounts)
nelson <- 154
cols <- c("aquamarine3", "brown4" )[(orderedfriendcounts == nelson) +1]

mp <- barplot(orderedfriendcounts, col= cols, border = cols, main="Friendship Paradox", xlab="Friend", ylab = "Num Friends")
par(new =TRUE)
valueclosestindex <- which.min(abs(orderedfriendcounts - mn))
abline(v = mp[valueclosestindex], col = "aquamarine4")
```

To get the mean for a histogram, I had to find the bar plot index that was the closest number and plot it there.



As you can see from the graph the friendship paradox holds. Empirically, Dr. Nelson's count was 154 and the mean and median were above it at 359 and 266.

Question 2

Determine if the friendship paradox holds for your Twitter account. Since Twitter is a directed graph, use "followers" as value you measure (i.e., "do your followers have more followers than you?").

I did not have more than 50 followers so I used "phonedude_mln". I created a python program using python-twitter to download all of the user ids of users that were following “phonedude_mln”. The results were outputted to a file with the first one being Dr. Nelson.

```
import twitter

outfile = open("twitterFollowers.txt", 'w')
# Variables that contains the user credentials to access Twitter API
ACCESS_TOKEN = '824794440409706496-CGUoqhcZpfk4lOcblNVG1OuubwaCqZp'
ACCESS_SECRET = 'Mgd7wctkwI3nNUL6RHD4Cqjg9HVngccRrynoWfEsTUzlJ'
CONSUMER_KEY = '72haUMMgwOXNNGXXz4G0N7B9m'
CONSUMER_SECRET = 'zSx800UtH7OUTkVUMw2uORB39o1aP59uZgmXMWo8AZpoFkNgJQ'

api = twitter.Api(consumer_key=CONSUMER_KEY,
                  consumer_secret=CONSUMER_SECRET,
                  access_token_key=ACCESS_TOKEN,
                  access_token_secret=ACCESS_SECRET,
                  sleep_on_rate_limit=True)

user = api.GetUser(screen_name="phonedude_mln")
userdict = user.AsDict()
userid = userdict["id"]

followerslist=[]
followersdata = api.GetFollowerIDsPaged(user_id=userid)
next_cursor = followersdata[0]
followerslist.extend(followersdata[2])

while next_cursor != 0 :
    followersdata = api.GetFollowerIDsPaged(user_id=userid, cursor=next_cursor)
    next_cursor = followersdata[0]
    followerslist.extend(followersdata[2])

outfile.write(str(userid)+ "\n")
for item in followerslist:
    outfile.write(str(item) + "\n")


outfile.close()
```

I then created another python program to go through that list and collect the followers for each. I switched to using tweepy because python-twitter did not work well to collect multiple users at the same time and I got account suspended error. Once I switched I didn't have any problems with my account and I could look up users much faster. I outputted the user ids and followers count to another file to be used in R.

```
import tweepy
#from tweepy import Stream
#from tweepy.streaming import StreamListener
from tweepy import OAuthHandler

infile = open("twitterFollowers.txt", 'r')
outfile = open("twitterCounts.txt", 'w')
# Variables that contains the user credentials to access Twitter API
ACCESS_KEY = '824794440409706496-CGUoqhcZpfk4lOcblNVG1OuubwaCqZp'
ACCESS_SECRET = 'Mgd7wctkwI3nNUL6RHD4Cqjg9HVngccRrynoWfEsTUz1J'
CONSUMER_KEY = '72haUMMgwOXNNGXXz4G0N7B9m'
CONSUMER_SECRET = 'zSx800UtH7OUTkVUMw2uORB39o1aP59uZgmXMWo8AZpoFkNgJQ'

auth = OAuthHandler(CONSUMER_KEY,CONSUMER_SECRET)
auth.set_access_token(ACCESS_KEY, ACCESS_SECRET)
api = tweepy.API(auth_handler=auth, wait_on_rate_limit=True)


flist = []
for line in infile:
    line = line.strip()
    flist.append(line)

infile.close()

lower_bound = 0
upper_bound = 99
while lower_bound < len(flist)-1 :
    shortlist = flist[lower_bound: upper_bound]
    followers = api.lookup_users(user_ids = shortlist)
    lower_bound = upper_bound + 1
    upper_bound = lower_bound + 99
    print(len(followers))
    for user in followers:

        if user.followers_count:
            outfile.write(str(user.id) + "\t" + str(user.followers_count) + "\n")

outfile.close()
```

I created another R program to plot the followers count on a logarithmic scale on the y axis. I used the same strategy as the previous program to draw the mean on the x axis.

```
friendcounts <- read.table("C:/Users/bitto/Documents/GitHub/cs532-s17/assignments/a4-solution/twitterCounts.txt")

orderedfriendcounts <- sort(friendcounts$V2)
mn <- mean(orderedfriendcounts)
nelson <- 622
cols <- c("aquamarine3", "brown4" )[(orderedfriendcounts == nelson) +1]

mp <- barplot(orderedfriendcounts, col= cols, border = cols, main="Friendship Paradox", xlab="Friend", ylab = "Num Friends", log = "y")
par(new =TRUE)
valueclosestindex <- which.min(abs(orderedfriendcounts - mn))
abline(v = mp[valueclosestindex], col = "aquamarine4")
```

I forgot to get the mean, median and standard deviation in the python program so I did it in R and named them appropriately.

mean	1538.433
median	325
orderedfriendc…	int [1:6
std	10256.35

As you can see in the graph below, the Friendship Paradox still holds. Dr. Nelson has less followers than the average of his followers have. Stated empirically, Dr. Nelson has 622 followers and the mean is 1538. However, the median is lower than Dr. Nelson meaning he has more than the middle value of the set, but the upper values are just so high it drags the mean very far above Dr. Nelson's number of followers.