Bo Zhang
01063214

1 Friendship Paradox with Facebook Account

Determine if the friendship paradox holds for my 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). Do include me in the graph and label me accordingly.

Algorithm:

1. Use the package "pygraphml" (http://hadim.fr/pygraphml/index.html) to parse the GraphML file.
2. Read the nodes 1 by 1.
3. For every node, if it has friend_count, save the friend index and its friend_count into the csv file.
4. Save my friend_count into "facebookFriends.csv".
5. Open the csv file in R and read all the data to a table.
6. Delete the row of me and save it to a friends table.
7. Compute the mean, standard deviation, and median of the number of friends with the friends table.
8. Plot the scatter with the full data table.

Source code:

Listing 1: The content of graphRead.py

```
import pygraphml

parser = pygraphml.GraphMLParser()
g = parser.parse("mln.graphml")

f = open("facebookFriends.csv", "w", encoding='utf-8')
f.write('label\tfriends\n')

index = 1
for node in g.nodes():
    try:
        f.write('f{}'.format(index)+'\t'+node['friend_count']+'\n')
        index = index + 1
    except:
        continue

f.write('me\t{}'.format(index-1)+'\n')
f.close()
```

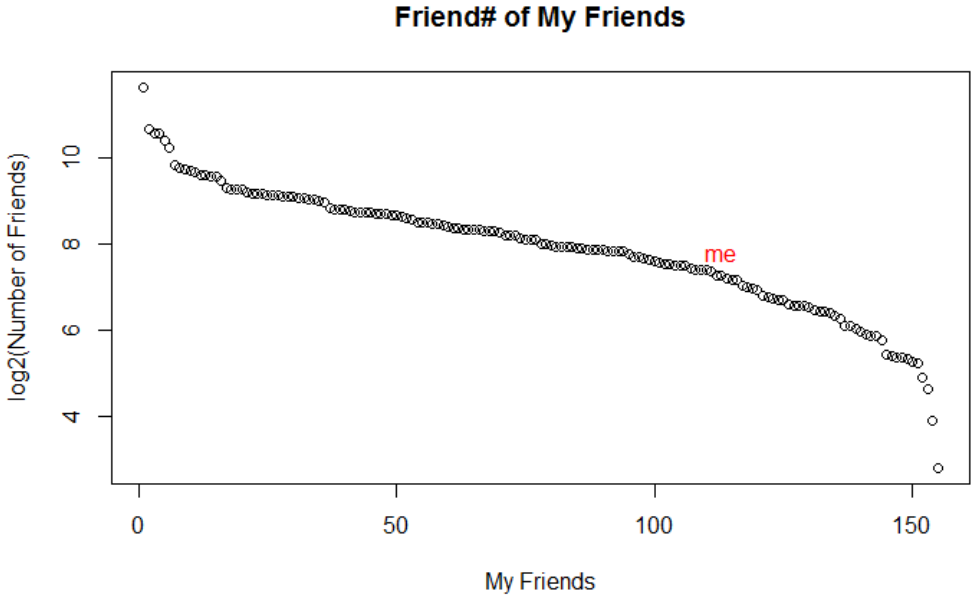
Listing 2: The content of Q1.R

```
data=read.csv("facebookFriends.csv", header=T, sep='\t')
friends=data[-nrow(data),]
print(mean(friends[,"friends"]))
print(sd(friends[,"friends"]))
print(median(friends[,"friends"]))
data=data[order(data[,"friends"],decreasing=TRUE),]
me=which(data[,"label"]=='me')
plot(log(data[,"friends"],base=2),main="Friend# of My Friends", xlab="My Friends",ylab=
text(me, log(data[me,"friends"],base=2)+0.5,"me", col="Red")
```

Results:

Table 1: My Friend# v.s. Friends Friend#

Mean	Standard Deviation	Median	Me
358.987	371.5853	266.5	154



According to these figures, more than 2/3 of "my" friends have more friends than "me" and the mean is more than double of "me". Therefore, the friendship paradox seems hold for "my" Facebook account.

2 Friendship Paradox with Twitter Followers

Determine if the friendship paradox holds for your Twitter account. Since Twitter is a directed graph, use "followers" as value you measure.

Generate the same graph as in question #1, and calcuate the same mean, standard deviation, and median values.

Algorithm:

1. Use the script from http://stackoverflow.com/questions/31000178/how-to-get-large-list-of-followers-tweepy to download the followers_count of all followers, and save them to "twitterFollowers.csv".
2. Open the csv file in R and read all the data to a table.
3. Delete the row of me and save it to a friends table.
4. Compute the mean, standard deviation, and median of the number of friends with the friends table.
5. Plot the scatter with the full data table.
6. Use the package "pygraphml" (http://hadim.fr/pygraphml/index.html) to parse the GraphML file.
7. Read the lines of the csv file 1 by 1.
8. Split the line into 2 parts and add the first part to the node's id and Label. Add the second part to the node's friend_count.
9. Save the nodes into "twitterFollowers.graphml".

Source code:

Listing 3: The content of downloadTwitterFollowers.py

```
import tweepy
import time

#Variables that contains the user credentials to access Twitter API
access_token = "825062339653271552-q2y3e35bUt1pKxdbqZ9leWIcgIT1mvt"
access_token_secret = "GIyuMRZB2xoIFVVJzJBRCt3kFWZCJh36rHY1T125GcVN0"
consumer_key = "EVKHzzDy0B3mtbvN426yrEZOM"
consumer_secret = "jAydpzL5jYnQGcUkxuwG1DeYGYgF6hu3zzlH9Vx1sG0iHbPKPT"

#This handles Twitter authetification
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth)

users = tweepy.Cursor(api.followers, screen_name='phonedude_mln').items()

f = open("twitterFollowers.csv", "w", encoding='utf-8')
f.write('label\tfriends\n')

#Print the follower# of every follower into the file 1 by 1
```

```
index = 1
while True:
    try:
        user = next(users)
    except tweepy.TweepError:
        time.sleep(60*15)
        user = next(users)
    except StopIteration:
        break
    f.write('f{}'.format(index) + '\t{}'.format(user.followers_count) + '\n')
    index = index + 1

f.write('me\t{}'.format(index-1)+'\n')
f.close()
```

Listing 4: The content of graphWrite.py

```
import pygraphml

file = input('Please input file name: ')

g = pygraphml.Graph()

f = open(file+".csv", "r", encoding='utf-8')
f.readline()

nodes=f.readlines()

for i in range(len(nodes)):
    line = nodes[i].split('\t')
    n = g.add_node(line[0])
    n['Label'] = line[0]
    n['friend_count'] = line[1]

f.close()
parser = pygraphml.GraphMLParser()
parser.write(g, file+".graphml")
```

Listing 5: The content of Q2.R

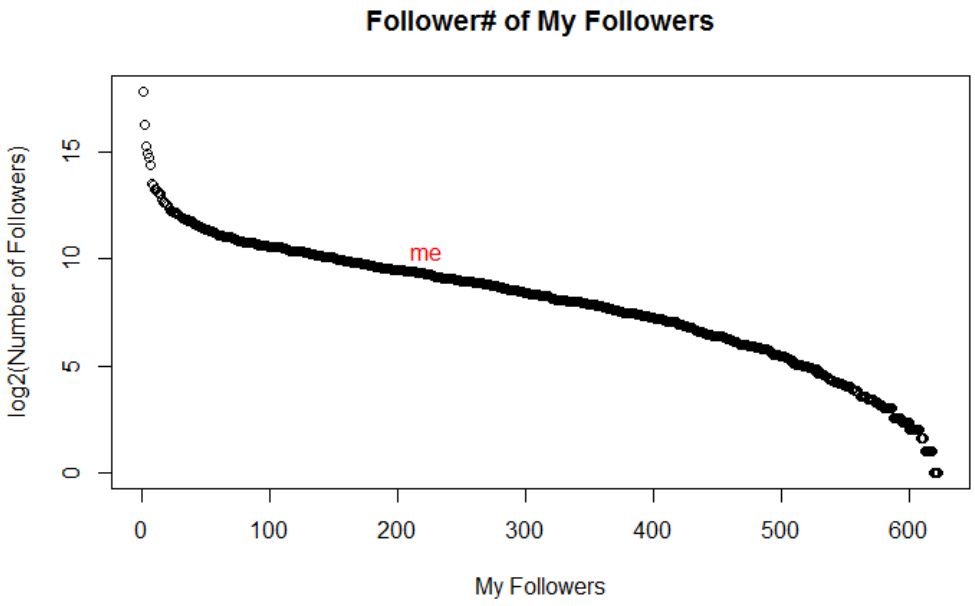
```
data=read.csv("twitterFollowers.csv", header=T, sep='\t')
friends=data[-nrow(data),]
print(mean(friends[,"friends"]))
print(sd(friends[,"friends"]))
print(median(friends[,"friends"]))
data=data[order(data[,"friends"],decreasing=TRUE),]
```

```
me=which(data[,"label"]=='me')
plot(log(data[,"friends"]+1,base=2),main="Follower# of My Followers", xlab="My Follower
text(me, log(data[me,"friends"]+1,base=2)+1,"me", col="Red")
```

Results: twitterFollowers.graphml

Table 2: My Follower# v.s. Followers Follower#

Mean	Standard Deviation	Median	Me
1516.9	10183.27	311	621



According to these figures, about 2/3 of "my" followers have less followers than "me" although the mean is more than double of "me". Therefore, the friendship paradox seems not hold for "my" twitter account measured by "followers".

3 Friendship Paradox with LinkedIn Connections

Repeat question #1, but with your LinkedIn profile.

Method & Algorithm:

1. Use Octoparse(a Web Scraping Tool) to get all my connections' profile page links. Open the

page of "My Connections"(When using this tool, must click the link from another page. If opening this page directly from the address bar, the connections will be incomplete.), then extract the texts with my connections' profile page links and save it to "linkedinConnections.txt".

2. Open the TXT file with python, extract the link part, complete it with domain and save it to "linkedinContactLinks.txt".

3. Use Octoparse to open all these links, extract the connections number and save it to "linkedinConnectionsNumber.txt".

4. Open the TXT file with python, manipulate the data and save it to "linkedinConnections.csv"(Because the connection number actually count self in and when the connection number is larger than 500, it shows 500+).

5. Open the csv file in R and read all the data to a table.

6. Delete the row of me and save it to a friends table.

7. Compute the mean, standard deviation, and median of the number of friends with the friends table.

8. Plot the scatter with the full data table.

Source code:

Listing 6: The content of extractLinkedinContactLinks.py

```
import re

f = open('linkedinConnections.txt', 'r')
lines=f.readlines()
f.close()

f = open('linkedinContactLinks.txt', 'w', encoding='utf-8')
for line in lines:
    m = re.search(r'(?<=href=")\S*(?=")', line)
    if m:
        f.write('https://www.linkedin.com'+m.group()+'\n')

f.close()
```

Listing 7: The content of linkedinConnectionsEdit.py

```
f = open("linkedinConnectionsNumber.txt", "r", encoding='utf-8')
lines = f.readlines()
f.close()

f = open("linkedinConnections.csv", "w", encoding='utf-8')
f.write('label\tfriends\n')
for i in range(len(lines)):
    if lines[i]=='500+\n':
        f.write('f{}'.format(i+1)+'\t500\n')
    else:
        f.write('f{}'.format(i+1)+'\t{}'.format(int(lines[i])-1)+'\n')
```

```
f.write('me\t{}'.format(len(lines))+'\n')
f.close()
```

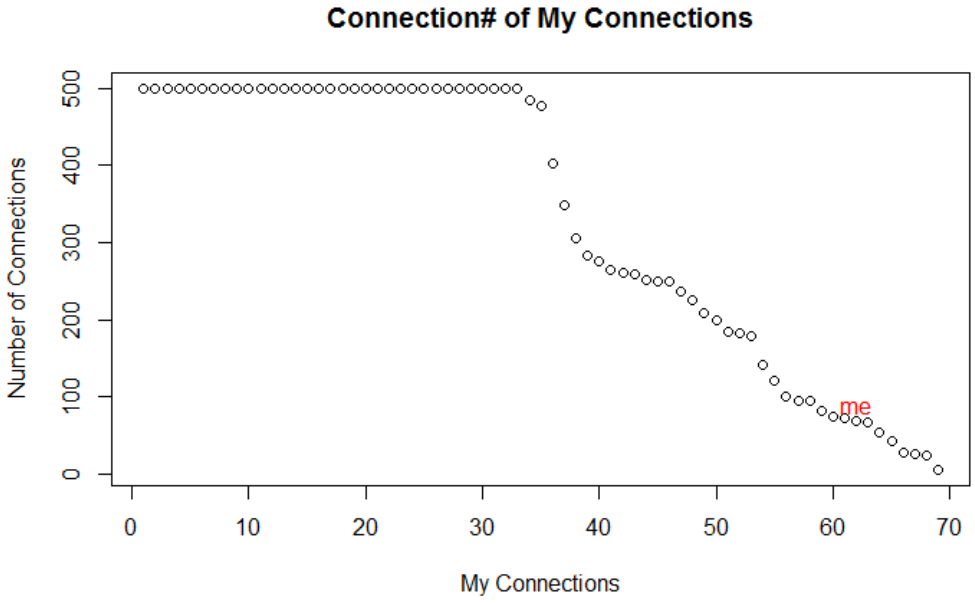
Listing 8: The content of Q3.R

```
data=read.csv("linkedinConnections.csv", header=T, sep='\t')
friends=data[-nrow(data),]
print(mean(friends[,"friends"]))
print(sd(friends[,"friends"]))
print(median(friends[,"friends"]))
data=data[order(data[,"friends"],decreasing=TRUE),]
me=which(data[,"label"]=='me')
plot(data[,"friends"],main="Connection# of My Connections", xlab="My Connections",ylab=
text(me, data[me,"friends"]+20,"me", col="Red")
```

Results:

Table 3: My Connection# v.s. Connections Connection#

Mean	Standard Deviation	Median	Me
339.0294	181.3933	481	68



According to these figures, nearly 90% of my connections have more connections than me and the mean is nearly 5 times of me. Therefore, the friendship paradox seems hold for my LinkedIn profile.

4 Friendship Paradox with Facebook Profile

Repeat question #1, but with your own facebook profile. Explain in detail how you got the information.

Answer: I don't have a facebook account.

5 Friendship Paradox with Twitter Following

Repeat question #2, but change "followers" to "following".

Algorithm:

1. Use the script from http://stackoverflow.com/questions/31000178/how-to-get-large-list-of-followers-tweepy, but replace the followers_count of all followers with friends_count of all friends, and save them to "twitterFollowing.csv".
2. Open the csv file in R and read all the data to a table.
3. Delete the row of me and save it to a friends table.
4. Compute the mean, standard deviation, and median of the number of friends with the friends table.
5. Plot the scatter with the full data table.
6. Use the package "pygraphml" (http://hadim.fr/pygraphml/index.html) to parse the GraphML file.
7. Read the lines of the csv file 1 by 1.
8. Split the line into 2 parts and add the first part to the node's id and Label. Add the second part to the node's friend_count.
9. Save the nodes into the "twitterFollowing.graphml".

Source code:

Listing 9: The content of downloadTwitterFollowering.py

```
import tweepy
import time

#Variables that contains the user credentials to access Twitter API
access_token = "825062339653271552-q2y3e35bUt1pKxdbqZ9leWIcgIT1mvt"
access_token_secret = "GIyuMRZB2xoIFVVJzJBRCt3kFWZCJh36rHY1T125GcVN0"
consumer_key = "EVKHzzDy0B3mtbvN426yrEZOM"
consumer_secret = "jAydpzL5jYnQGcUkxuwG1DeYGYgF6hu3zzlH9Vx1sG0iHbPKPT"
```

```
#This handles Twitter authetification and the connection to Twitter Streaming API
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
api = tweepy.API(auth)

users = tweepy.Cursor(api.friends, screen_name='phonedude_mln').items()

f = open("twitterFollowing.csv", "w", encoding='utf-8')
f.write('label\tfriends\n')

#Print the following# of every following into the file 1 by 1
index = 1
while True:
    try:
        user = next(users)
    except tweepy.TweepError:
        time.sleep(60*15)
        user = next(users)
    except StopIteration:
        break
    f.write('f{}'.format(index) + '\t{}'.format(user.friends_count) + '\n')
    index = index + 1

f.write('me\t{}'.format(index-1)+'\n')
f.close()
```

Listing 10: The content of graphWrite.py

```
import pygraphml

file = input('Please input file name: ')

g = pygraphml.Graph()

f = open(file+".csv", "r", encoding='utf-8')
f.readline()

nodes=f.readlines()

for i in range(len(nodes)):
    line = nodes[i].split('\t')
    n = g.add_node(line[0])
    n['Label'] = line[0]
    n['friend_count'] = line[1]

f.close()
```

```
parser = pygraphml.GraphMLParser()
parser.write(g, file+".graphml")
```

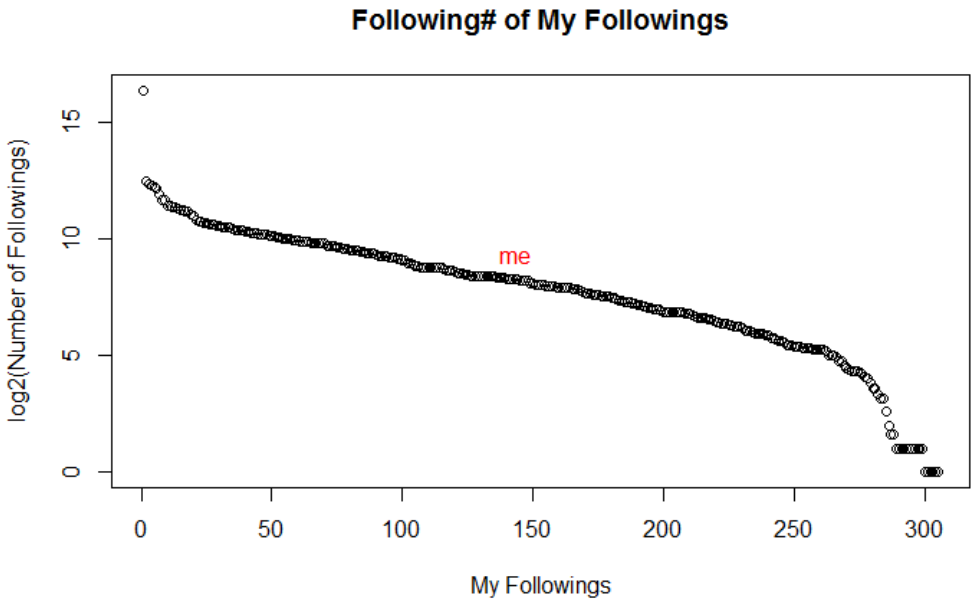
Listing 11: The content of Q5.R

```
data=read.csv("twitterFollowing.csv", header=T, sep='\t')
friends=data[-nrow(data),]
print(mean(friends[,"friends"]))
print(sd(friends[,"friends"]))
print(median(friends[,"friends"]))
data=data[order(data[,"friends"],decreasing=TRUE),]
me=which(data[,"label"]=='me')
plot(log(data[,"friends"]+1,base=2),main="Following# of My Followings", xlab="My Follow
text(me, log(data[me,"friends"]+1,base=2)+1,"me", col="Red")
```

Results: twitterFollowing.graphml

Table 4: My Following# v.s. Followings Following#

Mean	Standard Deviation	Median	Me
859.4967	4892.806	256	304



According to these figures, more than half of "my" followings have less followings than "me"

although the mean is nearly triple of “me”. Therefore, the friendship paradox seems not hold for “my” twitter account measured by “followering”.