Assignment 4
CS532, Web Science, Spring 2017
Old Dominion University, Computer Science Dept

Hussam Hallak
CS Master's Student
Prof: Dr. Nelson

Question 1:

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). (The friends don't need to be labeled on the x-axis: just f1, f2, f3, ... fn.) Do include me in the graph and label me accordingly.

* = This used to be more interesting when you could more easily download your friend's friends data from Facebook. Facebook now requires each friend to approve this operation, effectively making it impossible.

I will email to the list the XML file that contains my Facebook friendship graph ca. Oct, 2013. The interesting part of the file looks like this (for 1 friend):

```
<node id="Johan_Bollen_1448621116">
        <data key="Label">Johan Bollen</data>
        <data key="uid"><![CDATA[1448621116]]></data>
        <data key="name"><![CDATA[Johan Bollen]]></data>
        <data key="mutual_friend_count"><![CDATA[37]]></data>
        <data key="friend_count"><![CDATA[420]]></data>
</node>
```

It is in GraphML format: http://graphml.graphdrawing.org/

Answer:

I created a simple python program to parse the file "mln.graphml" and capture the number of friends for each friend of Dr. Nelson. These numbers are sorted and saved in a file named "friendscount.txt", which will be used later to generate the friendship paradox graph in R.

Listing 1: The content of parse.py

```
import sys
import numpy as np
import urllib2
import re

if len(sys.argv) != 2:
   print "Usage: Python parse.py <file_name>"
   print "e.g: Python parse.py mln.graphml"
   exit()

fh_input = open(sys.argv[1], 'r')
fh_output = open("friendscount.txt", 'w')
friends = []
total = 0

for tag in fh_input:
   if 'key="friend_count' in tag:
      num = tag[35:]
```

```
        num = num[:-11]
        num = int(num)
        total = total + num
        friends.append(num)

npar = np.array(friends)
median = np.median(npar)
count = float(len(friends))
mean = total/count
print 'Total friends of friends of mln including his:'
print total
print 'mln friends count:'
print len(friends)
print 'Mean:'
print round(mean, 1)
print 'Median:'
print round(median, 1)
print "STD:"
print round(np.std(friends), 1)

friends.append(len(friends))
friends.sort()
for number in friends:
        number = str(number)
        fh_output.write(number)
        fh_output.write('\n')

fh_input.close()
fh_output.close()
```

Listing 2: Running parse.py to capture the number of friends for each friend of Dr. Nelson

```
root@ima-app:/var/www/Hussam/A4# python parse.py mln.graphml
Total friends of friends of mln including his:
55284
mln friends count:
154
Mean:
359.0
Median:
266.5
STD:
370.4
```

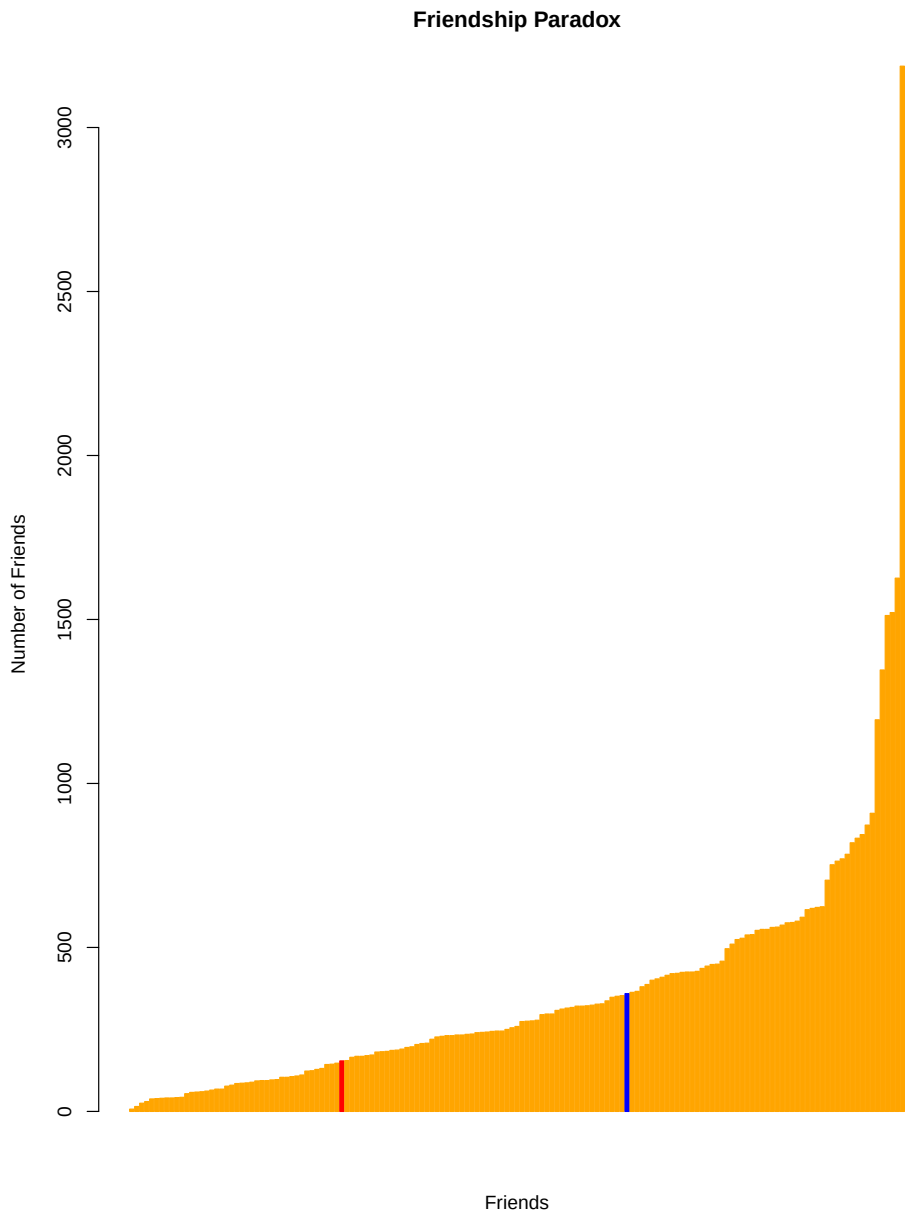
The program created the output file and printed out the mean, standard deviation, and median of the number of friends that mln friends have. The number of Dr. Nelson's friends is not included in the calculations, but included in the output file to use it in the graph.

I wrote R code to create the graph of the number of friends (y-axis) and the friends themselves, sorted by number of friends (x-axis). The graph shows the position of Dr. Nelson among his friends colored in red on the graph and the mean value colored in blue.

Listing 3: Friendship Paradox graph code in R

```
> friendcounts <- read.table("C:/R/friendscount.txt")
```

```
> mn <- 359
> mln <- 154
> cols = ifelse((friendcounts$V1 == mln), "red", (ifelse(abs(friendcounts$V1 ==
    mn), "blue" , "orange")))
> barplot(friendcounts$V1, col= cols, border=cols, main="Friendship Paradox",
    xlab="Friends", ylab = "Number of Friends")
```



Looking at the graph, it is obvious that the friendship paradox holds for Dr. Nelsons account. His friends count is 154 and the mean is higher. The mean is 359.0 and the median is 266.5.

Included Files:

parse.py, friendscount.txt, A4-1.Rproj, facebook.pdf, mln.graphml

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?").

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

For the Twitter 1.1 API to help gather this data, see:

https://dev.twitter.com/docs/api/1.1/get/followers/list

If you do not have followers on Twitter (or don't have more than 50), then use my twitter account "phonedude_mln".

Answer:

I do not have a minimum of 50 followers on Twitter, so I used "phonedude_mln" instead. I wrote a python program to capture the number of followers for users following "phonedude_mln". The extracted data is saved to a file named "followerscount.txt".

Note: Followers of "phonedude_mln" that do not have any followers have been taken in consideration when calculating the mean, median, and standard deviation; however, they are not saved in the output file because 0 generates an error when creating the graph and setting the "log" argument to "y" because $log(0)$ is undefined and causes R to produce an error.

Listing 4: The content of config.py

```
#Get your Twitter API credentials and enter them here
consumer_key = "put_your_consumer_key_here"
consumer_secret = "put_your_consumer_secret_here"
access_key = "put_your_access_key_here"
access_secret = "put_your_access_secret_here"
```

Listing 5: The content of getfollowersdata.py

```
import sys
import tweepy
from tweepy import *
import time
import numpy as np
if len(sys.argv) < 2:
   print "Usage: python getfollowersdata.py <twitter_username>"
   print "e.g: python getfollowersdata.py phonedude_mln"
   exit()

username = sys.argv[1]
#config.py contains your twitter's API consumer_key, consumer_secret, access_key,
     and access_secret
config = {}
execfile("config.py", config)
fh_output = open("followerscount.txt", 'w')

if __name__ == '__main__':
   auth = tweepy.OAuthHandler(config["consumer_key"], config["consumer_secret"])
   auth.set_access_token(config["access_key"], config["access_secret"])
```

```
    api = tweepy.API(auth)
    data = api.get_user(username)
    followers = []
    total = 0
    for user in tweepy.Cursor(api.followers, screen_name=username, count = 200).
        items():
       num = int(user.followers_count)
       total = total + num
       followers.append(num)

    npar = np.array(followers)
    median = np.median(npar)
    count = float(len(followers))
    mean = total/count
    print 'Total followers of followers of ', username, ' including his:'
    print total
    print username, '\'s followerss count:'
    print len(followers)
    print 'Mean:'
    print round(mean, 1)
    print 'Median:'
    print round(median, 1)
    print "STD:"
    print round(np.std(followers), 1)
    followers.append(len(followers))
    followers.sort()
    for number in followers:
       if number > 0:
            number = str(number)
            fh_output.write(number)
            fh_output.write('\n')

fh_output.close()
```

Listing 6: Running getfollowersdata.py

```
root@ima-app:/var/www/Hussam/A4# python getfollowersdata.py phonedude_mln
/usr/local/lib/python2.7/dist-packages/requests/packages/urllib3/util/ssl_.py
    :334: SNIMissingWarning: An HTTPS request has been made, but the SNI (Subject
     Name Indication) extension to TLS is not available on this platform. This
    may cause the server to present an incorrect TLS certificate, which can cause
     validation failures. You can upgrade to a newer version of Python to solve
    this. For more information, see https://urllib3.readthedocs.io/en/latest/
    advanced-usage.html#ssl-warnings
  SNIMissingWarning
/usr/local/lib/python2.7/dist-packages/requests/packages/urllib3/util/ssl_.py
    :132: InsecurePlatformWarning: A true SSLContext object is not available.
    This prevents urllib3 from configuring SSL appropriately and may cause
    certain SSL connections to fail. You can upgrade to a newer version of Python
     to solve this. For more information, see https://urllib3.readthedocs.io/en/
    latest/advanced-usage.html#ssl-warnings
  InsecurePlatformWarning
/usr/local/lib/python2.7/dist-packages/requests/packages/urllib3/util/ssl_.py
    :132: InsecurePlatformWarning: A true SSLContext object is not available.
    This prevents urllib3 from configuring SSL appropriately and may cause
```

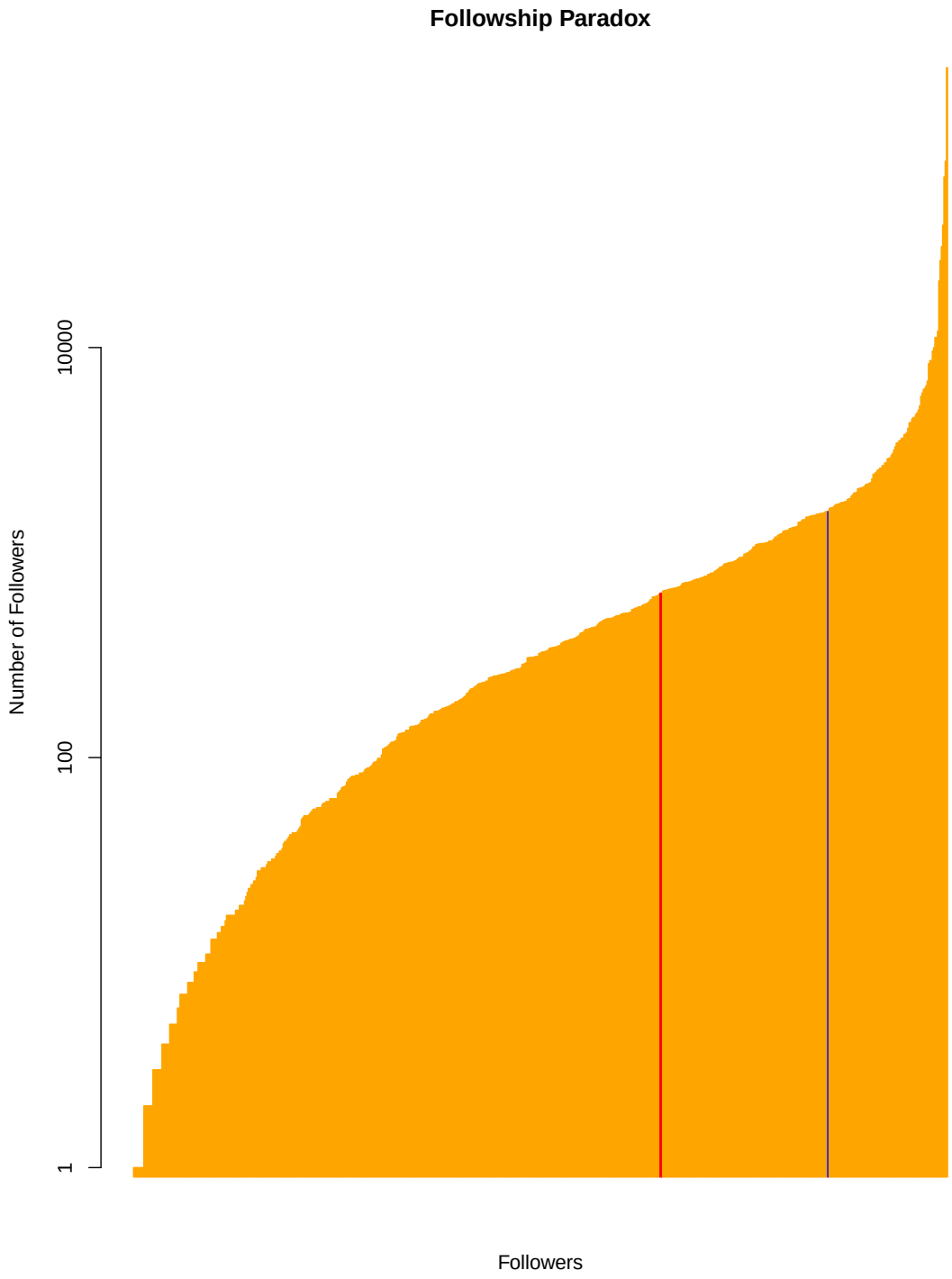
```
    certain SSL connections to fail. You can upgrade to a newer version of Python
     to solve this. For more information, see https://urllib3.readthedocs.io/en/
    latest/advanced-usage.html#ssl-warnings
  InsecurePlatformWarning
Total followers of followers of phonedude_mln including his:
1017606
phonedude_mln 's followerss count:
634
Mean:
1605.1
Median:
311.5
STD:
10418.1
```

The program created the output file and printed out the mean, standard deviation, and median of the number of followers that "phonedude_mln' followers have. The number of Dr. Nelson's followers is not included in the calculations, but included in the output file to use it in the graph.

I wrote R code to create the graph of the number of followers (y-axis) and the followers themselves, sorted by number of followers (x-axis). The graph shows the position of Dr. Nelson among his followers colored in red on the graph and the mean value colored in blue.

Listing 7: Followship Paradox graph code in R

```
> followercounts <- read.table("C:/R/followerscount.txt")
> mn <- 1605.1
> mln <- 634
> cols = ifelse((followercounts$V1 == mln), "red", (ifelse((abs(followercounts$V1
     - mn) < 21), "blue" , "orange")))
> barplot(followercounts$V1, col= cols, border=cols, main="Followship Paradox",
    xlab="Followers", ylab = "Number of Followers", log="y")
```



Looking at the graph, it is obvious that the Followship Paradox holds for Dr. Nelsons account. He has 634 followers and the mean is higher. The mean is 1605.1 and the median is 311.5.

Included Files:

getfollowersdata.py, config.py followerscount.txt, A4-2.Rproj, twitter.pdf