CS532 Web Science: Assignment 4

Finished on February 25, 2016

Dr. Michael L. Nelson

Naina Sai Tipparti
ntippart@cs.odu.edu

Contents

Listings

List of Figures

List of Tables

Problem 1

Question

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):

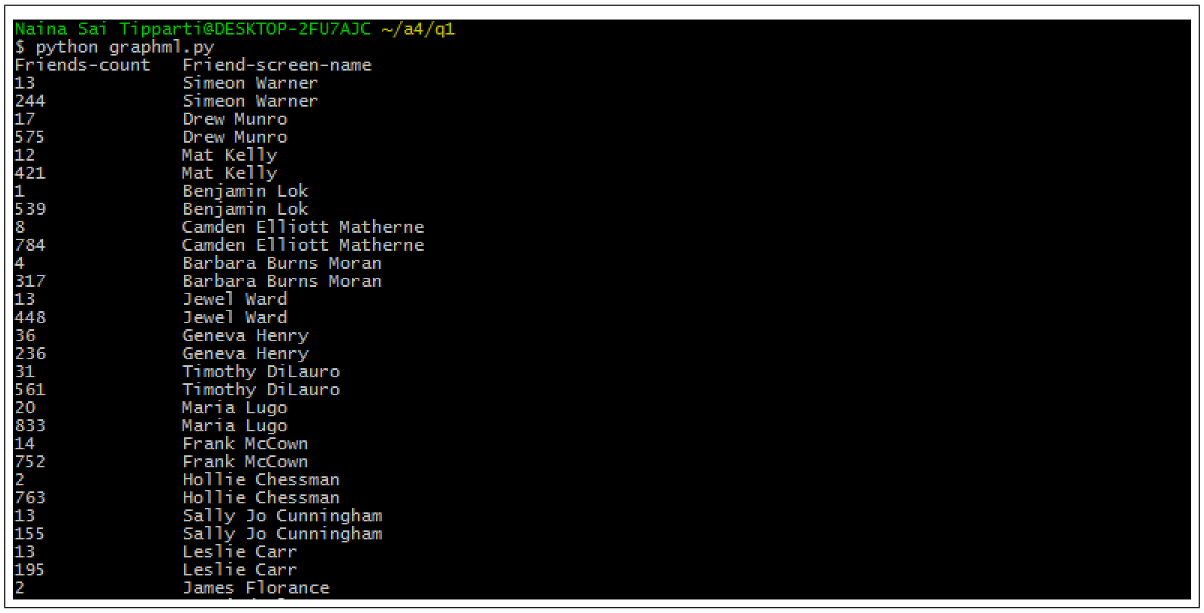
```
<node id="Johan_Bollen_1448621116">
    <data key="Label">Johan Bollen</data>
    <data key="uid"><![CDATA[1448621116]]></data>
    <data key="name"><![CDATA[Johan Bollen]]></data>
    <data key="mutual_friend_count"><![CDATA[37]]></data>
    <data key="friend_count"><![CDATA[420]]></data>
</node>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

Listing 1: mln.graphml

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

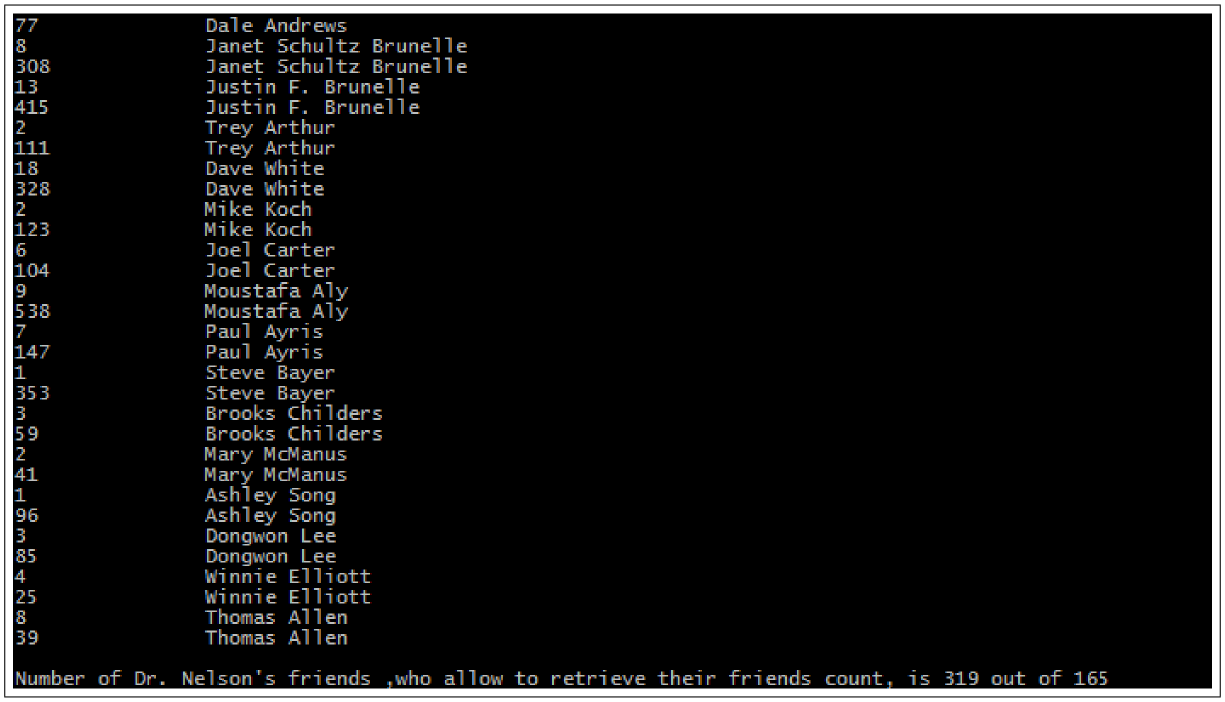
Answer

A Python program, *get_graphml.py*, has been written to extract number of friends that Dr. Nelson friends have. The program will search for this information in a file called *mln.graphml*. The output of this program would be like the following:



```
Naina Sai Tipparti@DESKTOP-2FU7AJC ~/a4/q1
$ python graphml.py
Friends-count   Friend-screen-name
13              Simeon Warner
244             Simeon Warner
17              Drew Munro
575             Drew Munro
12              Mat Kelly
421             Mat Kelly
1               Benjamin Lok
539             Benjamin Lok
8               Camden Elliott Matherne
784             Camden Elliott Matherne
4               Barbara Burns Moran
317             Barbara Burns Moran
13              Jewel Ward
448             Jewel Ward
36              Geneva Henry
236             Geneva Henry
31              Timothy DiLauro
561             Timothy DiLauro
20              Maria Lugo
833             Maria Lugo
14              Frank McCown
752             Frank McCown
2               Hollie Chessman
763             Hollie Chessman
13              Sally Jo Cunningham
155             Sally Jo Cunningham
13              Leslie Carr
195             Leslie Carr
2               James Florance
```

Figure 1: Sample output of number of friends



```
77              Dale Andrews
8               Janet Schultz Brunelle
308             Janet Schultz Brunelle
13              Justin F. Brunelle
415             Justin F. Brunelle
2               Trey Arthur
111             Trey Arthur
18              Dave White
328             Dave White
2               Mike Koch
123             Mike Koch
6               Joel Carter
104             Joel Carter
9               Moustafa Aly
538             Moustafa Aly
7               Paul Ayris
147             Paul Ayris
1               Steve Bayer
353             Steve Bayer
3               Brooks Childers
59              Brooks Childers
2               Mary McManus
41              Mary McManus
1               Ashley Song
96              Ashley Song
3               Dongwon Lee
85              Dongwon Lee
4               Winnie Elliott
25              Winnie Elliott
8               Thomas Allen
39              Thomas Allen

Number of Dr. Nelson's friends ,who allow to retrieve their friends count, is 319 out of 165
```

Figure 2: Sample output of number of friends

```python
# -*- encoding: utf-8 -*-
#! /usr/bin/python
from __future__ import unicode_literals
import xml.etree.cElementTree as et
from bs4 import BeautifulSoup
from urlparse import parse_qs
import unicodedata
import urllib2
import re
import os
import sys

print '%-15s %-20s' %('Friends-count','Friend-screen-name')

file = "mln.graphml"
handler = open(file).read()
soup = BeautifulSoup(handler)
i = 0
all = 0
for message in soup.find_all('node'):
    all += 1
    foo = et.XML(str(message))
    name = ''
    for e in foo:
        if ('graphml_count' in str(e.items())):
            print '%-15s %-20s' %(e.text,name)
            with open('friend_counts', 'a') as outfile:
                outfile.write('%-15s %-20s\n' %(e.text,name))
            i += 1
        if ('name' in str(e.items())):
            name = e.text
print "\nNumber of Dr. Nelson's friends ,who allow to retrieve their friends count, is "+str
    (i)+" out of "+str(all)
```

Listing 2: get_graphml.py

I would like to let you know that even though Dr. Nelson have 319 friends, only 165 allow me to see their number of friends. This will affect the statistical result. For example, instead of dividing by 319 to get the mean, we divide by 165.

The `graphml_counts` file was ordered in place with the Unix command in Listing 3.

```
Naina Sai Tipparti@DESKTOP-2FU7AJC ~/a4/q1 cat graphml_counts | sort -g -o graphml_counts
```

Listing 3: Sort command

This file was then processed by the R script shown in Listing 4 to produce the graph in Figure 3

```r
#! /usr/bin/Rscript

# read data
data <- read.table('D:/cs532/a4/q1/graphml_counts',sep=",")
x <- seq(1, length(data$V1))
y <- data$V1

# get notable values
mln_idx <- grep("phonedude_mln", data$V2)
med_val <- median(data$V1)
med_idx <- which(abs(y - med_val) == min(abs(y - med_val)))
mean_val <- mean(data$V1)
mean_idx <- which(abs(y - mean_val) == min(abs(y - mean_val)))
```

```
std_dev <- sd(data$V1)

# draw the graph
pdf("D:/cs532/a4/q1/facebook_graphml.pdf")
plot(x, y, type="l", log="y", pch=19, main="Dr. Nelson's Friends' Friends",
    ylab="Number of Friends", xlab="Index of Friend")

# illustrate points of interest
abline(h=data$V1[mln_idx], col="red")

# The Legend of the Data
legend(x=82, y=5, cex=0.8, lty=c(1, 1),
    col=c("red", "white", "white", "white", "white"),
    c(paste("Nelson: ", data$V1[mln_idx]), paste("median: ", med_val),
        paste("mean: ", format(round(mean_val, 4), nsmall = 4)),
        paste("std dev: ", format(round(std_dev, 4), nsmall = 4)),
        paste("median + 1 std dev: ", format(round(med_val + std_dev, 4), nsmall = 4))))
dev.off()
```

Listing 4: Graph Creation Script

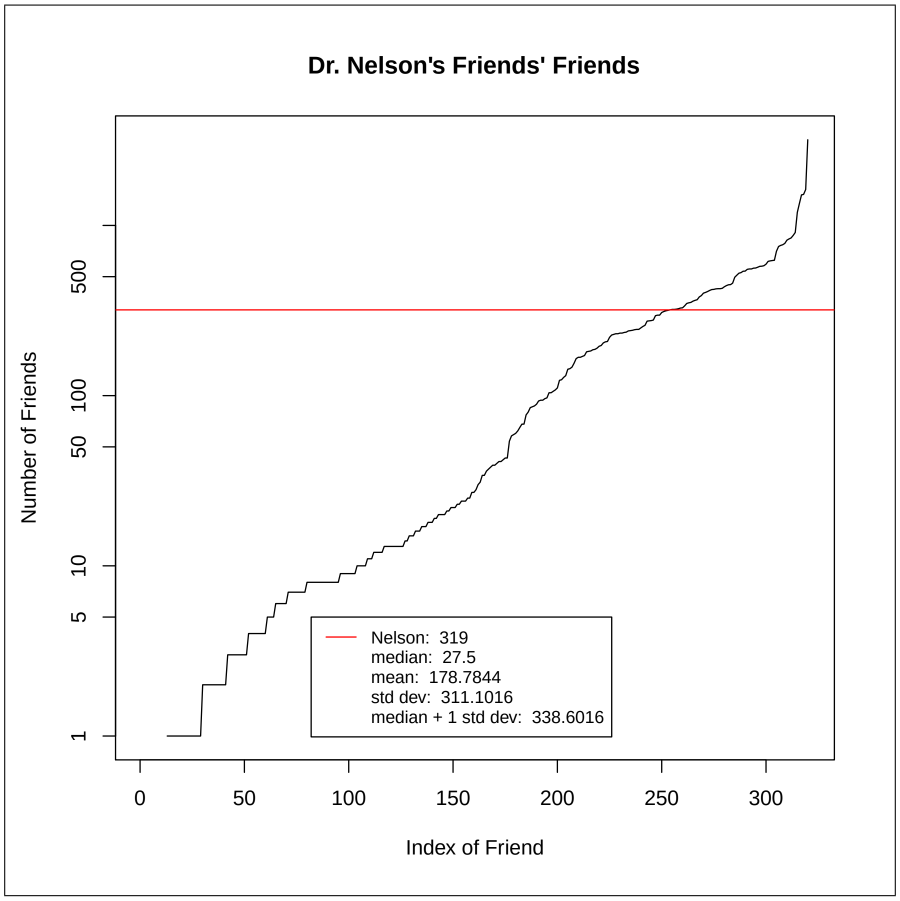


Figure 3: The Friendship Graph for Facebook

Mean	178.7844
Median	27.5
Std Dev	311.1016

Table 1: Statistics on the count of Dr. Nelson Facebook Friends' Friends, values straight from R

The median, mean and standard deviation were all calculated, with the median, mean and median plus one standard deviation.

Problem 2

Question

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 calcuate 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

Using Dr. Michael Nelson’s Twitter account and the Twitter API [?], specifically the GET friends/list [?] request, all of Dr. Nelson’s Twitter friends were obtained and saved to the file called `followers`. This method also uses the API’s paginating scheme: when there are a large number of results for a query, the API will send a cursor index to show that there are more results to process and that more requests are needed. The code to do this is in Listing 5.

```
# -*- encoding: utf-8 -*-
from __future__ import unicode_literals
import re
import os
import sys
import json
import requests
import subprocess
from urlparse import parse_qs
from requests_oauthlib import OAuth1

REQUEST_TOKEN_URL = "https://api.twitter.com/oauth/request_token"
AUTHORIZE_URL = "https://api.twitter.com/oauth/authorize?oauth_token="
ACCESS_TOKEN_URL = "https://api.twitter.com/oauth/access_token"

CONSUMER_KEY = "lA3ACTYCPDE8G5rNYFBMI1hMm"
CONSUMER_SECRET = "4zGZNDRA2m32dsq7nCMfwJojGSanz6ohgf4ZaNWKDxCaabPUai"
OAUTH_TOKEN = "798668178-bH8DbMpNuWkfhAHxuODgWSHwQE65B1WZnc4Ahtej"
OAUTH_TOKEN_SECRET = "FhykPKnQcgKQBE43os2bDZ31ugH9RVSG3HYoOL7QG7RNC"

def setup_oauth():
    """Authorize your app via identifier."""
    # Request token
    oauth = OAuth1(CONSUMER_KEY, client_secret=CONSUMER_SECRET)
    r = requests.post(url=REQUEST_TOKEN_URL, auth=oauth)
    credentials = parse_qs(r.content)

    resource_owner_key = credentials.get('oauth_token')[0]
    resource_owner_secret = credentials.get('oauth_token_secret')[0]

    # Authorize
    authorize_url = AUTHORIZE_URL + resource_owner_key
    print 'Please go here and authorize: ' + authorize_url

    verifier = raw_input('Please input the verifier: ')
    oauth = OAuth1(CONSUMER_KEY,
                   client_secret=CONSUMER_SECRET,
                   resource_owner_key=resource_owner_key,
                   resource_owner_secret=resource_owner_secret,
                   verifier=verifier)

    # Finally, Obtain the Access Token
    r = requests.post(url=ACCESS_TOKEN_URL, auth=oauth)
    credentials = parse_qs(r.content)
    token = credentials.get('oauth_token')[0]
    secret = credentials.get('oauth_token_secret')[0]

    return token, secret


def get_oauth():
    oauth = OAuth1(CONSUMER_KEY,
                client_secret=CONSUMER_SECRET,
                resource_owner_key=OAUTH_TOKEN,
                resource_owner_secret=OAUTH_TOKEN_SECRET)
    return oauth

if __name__ == "__main__":
    if not OAUTH_TOKEN:
        token, secret = setup_oauth()
        print "OAUTH_TOKEN: " + token
        print "OAUTH_TOKEN_SECRET: " + secret
        print
    else:
        twitterUser = "phonedude_mln"

        print 'Searching Twitter for followers counts of '+twitterUser+"'s followers: "
        oauth = get_oauth()

        print '%-15s %-20s' %('Followers_count','Follower-screen-name')

        # initial reading from the twitter account where cursor = -1 (e.g. first page)
        r = requests.get(url="https://api.twitter.com/1.1/followers/list.json?cursor=-1&
            count=2000&screen_name="+twitterUser+"&skip_status=true&include_user_entities=
            false", auth=oauth)
        counter = 0
        res = r.json()
        while True:
            raw_res = res['users']
            for init_url in raw_res:
                counter = counter + 1
                print '%-15d %-20s' %(init_url['followers_count'],init_url['screen_name'].
                    encode('ascii', 'replace'))
                with open('friend_counts', 'a') as outfile:
                    outfile.write('%-15d %-20s\n' %(init_url['followers_count'],init_url['
                        screen_name'].encode('ascii', 'replace')))
            if str(res['next_cursor']) == '0':
                break
            else:
                r = requests.get(url="https://api.twitter.com/1.1/followers/list.json?cursor
                    ="+str(res['next_cursor'])+"&count=100&screen_name="+twitterUser+"&
                    skip_status=true&include_user_entities=false", auth=oauth)
                res = r.json()

print '\nNumber of '+twitterUser+"'s followers is: "+str(counter)
```

Listing 5: get_followers.py

To reduce the impact of high HTTP traffic, the Twitter API rate-limits most requests – the one needed to obtain a user's friends list has a limit of fifteen message per fifteen minutes. Any requests received from a user or service that has reached the limit will be denied.
The friends of Dr. Nelson's friends were then obtained with the same `get_followers` method from Listing 5 and stored in a file called `followers_counts`, each on a single line preceded by their friend count. All of these operations were controlled by a main method, which is shown in Listing 5.
The `followers_counts` file was ordered in place with the Unix command in Listing 6.

```
Naina Sai Tipparti@DESKTOP-2FU7AJC ~/a4/q3 cat followers_counts | sort -g -o
    followers_counts
```

Listing 6: Sort command

This file was then processed by the R script shown in Listing 7 to produce the graph in Figure 4

```r
#! /usr/bin/Rscript

# read data
data <- read.table('D:/cs532/a4/q2/followers_counts')
x <- seq(1, length(data$V1))
y <- data$V1

# get notable values
mln_idx <- grep("phonedude_mln", data$V2)
med_val <- median(data$V1)
med_idx <- which(abs(y - med_val) == min(abs(y - med_val)))
mean_val <- mean(data$V1)
mean_idx <- which(abs(y - mean_val) == min(abs(y - mean_val)))
std_dev <- sd(data$V1)

# draw the graph
pdf("D:/cs532/a4/q2/followers_plot.pdf")
plot(x, y, type="l", log="y", pch=19, main="Dr. Nelson's Friends' Friends",
    ylab="Number of Friends", xlab="Index of Friend")

# illustrate points of interest
abline(h=data$V1[mln_idx], col="red")

# The Legend of the Data
legend(x=82, y=5, cex=0.8, lty=c(1, 1),
    col=c("red", "white", "white", "white", "white"),
    c(paste("Nelson: ", data$V1[mln_idx]), paste("median: ", med_val),
        paste("mean: ", format(round(mean_val, 4), nsmall = 4)),
        paste("std dev: ", format(round(std_dev, 4), nsmall = 4)),
        paste("median + 1 std dev: ", format(round(med_val + std_dev, 4), nsmall = 4))))
dev.off()
```

Listing 7: Graph Creation Script for Twitter

Mean	1024.7424
Median	258
Std Dev	4135.7974

Table 2: Statistics on the count of Dr.Nelson Followers, values straight from R

The median, mean and standard deviation were all calculated, with the median, mean and median plus one standard deviation.

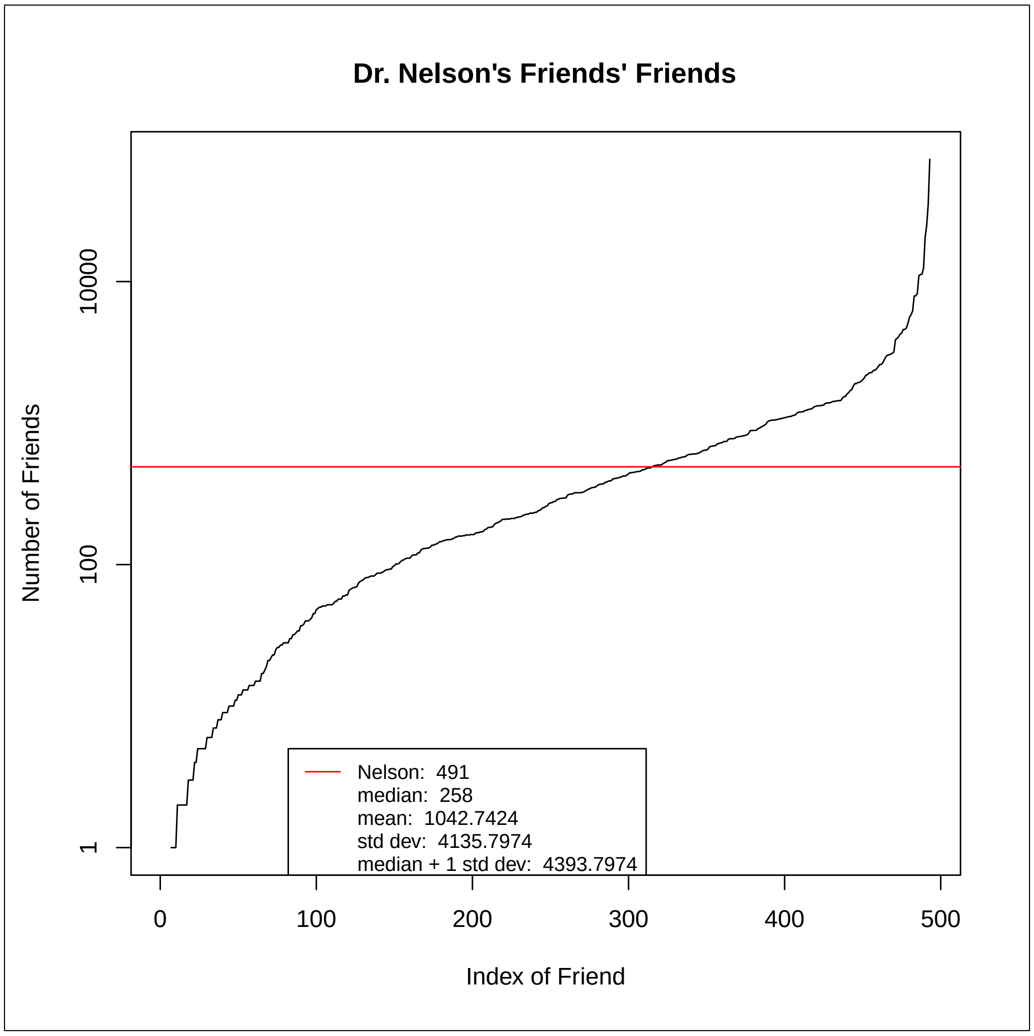


Figure 4: The Friendship Graph for Twitter Followers

Problem 4

Question

Repeat question #2, but change “follower” to “following”? In other words, are the people I am following following more people?

Answer

Same instructions explained in question 2 are used to answer this question. As I remember, only one change has been made to *get_followers.py*.

- The old request:

```
r = requests.get(url="https://api.twitter.com/1.1/followers/list.json?cursor
    =-1&count=2000&screen_name="+twitterUser+"&skip_status=true&
    include_user_entities=false", auth=oauth)
```

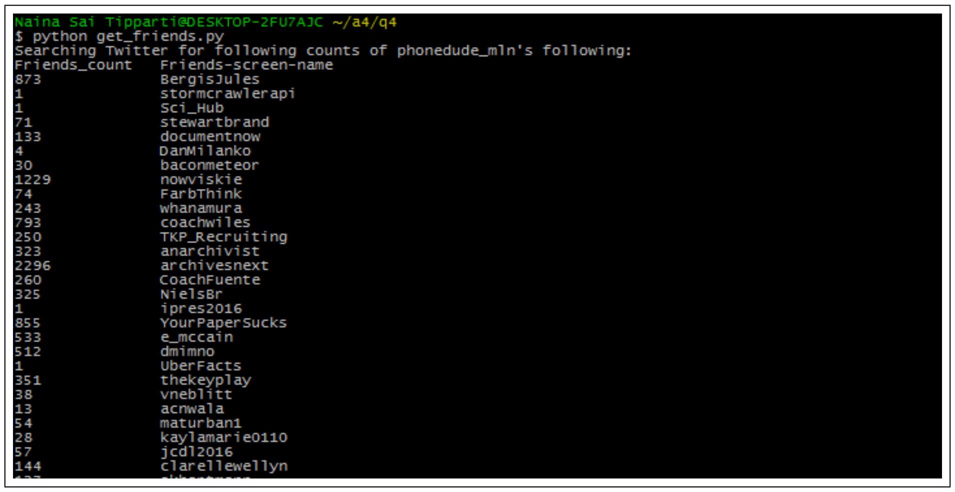
Listing 8: get_followers.py

- After modifying:

```
r = requests.get(url="https://api.twitter.com/1.1/friends/list.json?cursor=-1&
    count=2000&screen_name="+twitterUser+"&skip_status=true&
    include_user_entities=false", auth=oauth)
```

Listing 9: get_friends.py

All new changes are stored in a new file called *get_friends.py*. The following is the output after running the Python program:



```
Naina Sai Tipparti@DESKTOP-2FU7AJC ~/a4/q4
$ python get_friends.py
Searching Twitter for following counts of phonedude_mln's following:
Friends_count   Friends-screen-name
873             BergisJules
1               stormcrawlerapi
1               Sci_Hub
71              stewartbrand
133             documentnow
4               DanMilanko
30              baconmeteor
1229            nowviskie
74              FarbThink
243             whanamura
793             coachwiles
250             TKP_Recruiting
323             anarchivist
2296            archivesnext
260             CoachFuente
325             NielsBr
1               ipres2016
855             YourPaperSucks
533             e_mccain
512             dmimno
1               UberFacts
351             thekeyplay
38              vneblitt
13              acnwala
54              maturban1
28              kaylamarie0110
57              jcdl2016
144             clarellewellyn
```

Figure 5: Sample output of number of following

Mean	484.9214
Median	225
Std Dev	729.5275

Table 3: Statistics on the count of Dr.Nelson Following, values straight from R

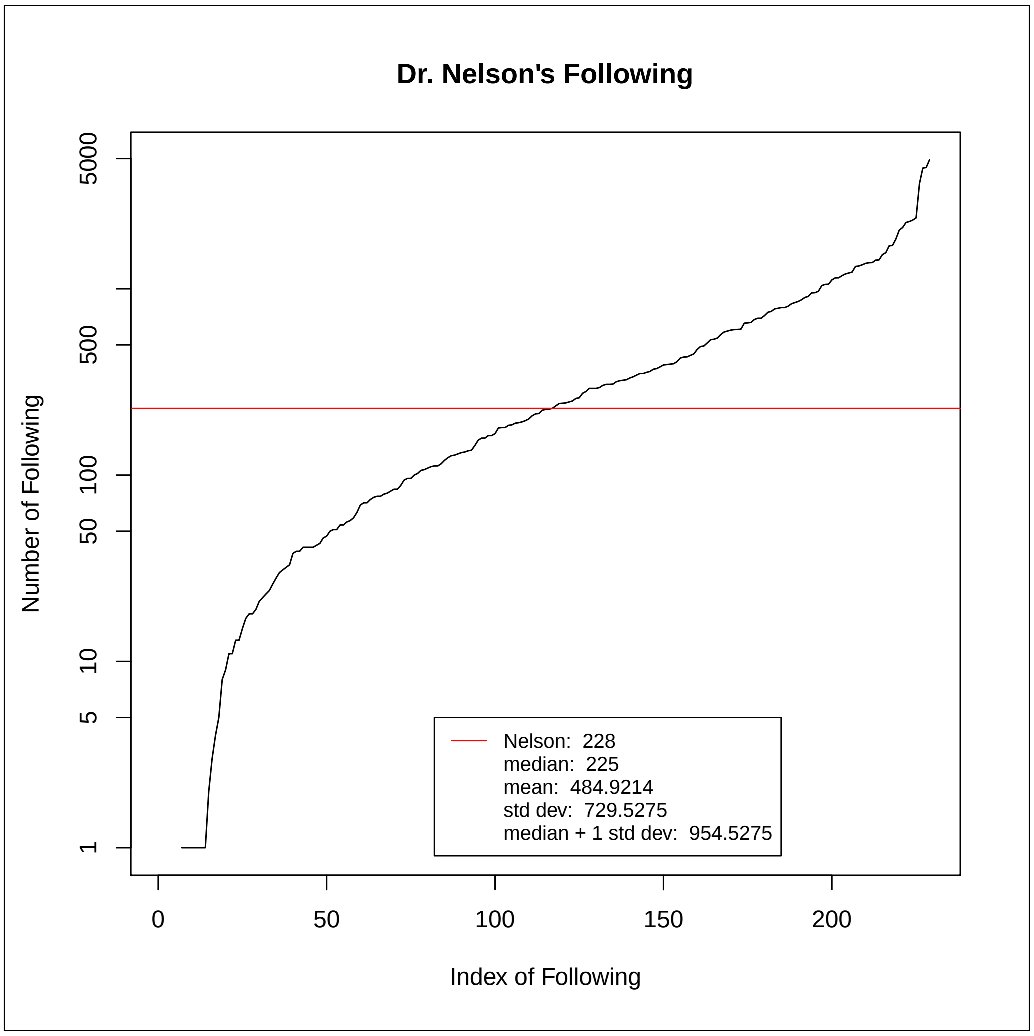


Figure 6: The Friendship Graph for Twitter Following

I think here also the friendship paradox holds for Dr. Nelson Twitter account (for following) since the mean is equal to 484.9214 which greater than most of the number of following of Dr. Nelson following including himself.