CS532 Web Science: Assignment 4

Finished on February 26, 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, *qet_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:

Figure 1: Sample output of number of friends

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)))
```

```
14  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",
19       ylab="Number of Friends", xlab="Index of Friend")

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

24  # 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)),
29           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 for Facebook
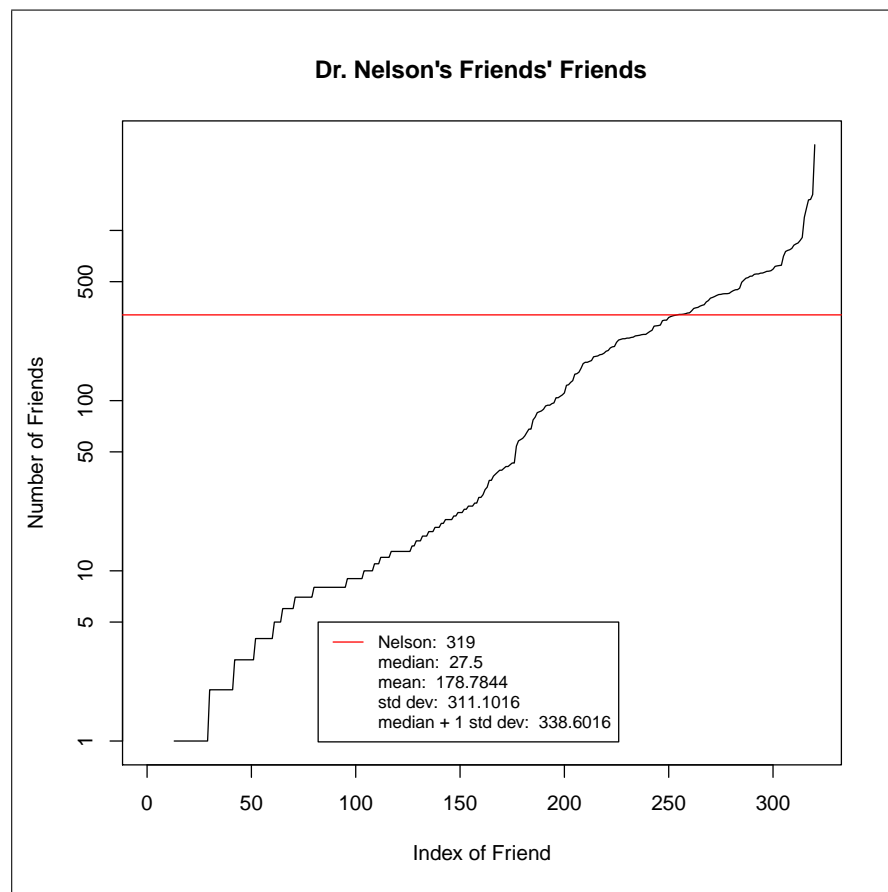
Figure 3: The Friendship Graph for Facebook

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.

```python
# -*- 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
```

```python
        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.

```
1  Naina Sai Tipparti@DESKTOP−2FU7AJC ~/a4/q2 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

```
#! /usr/bin/Rscript

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

# get notable values
9  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$V1)
mean_idx <− which(abs(y − mean_val) == min(abs(y − mean_val)))
14 std_dev <− sd(data$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",
19     ylab="Number of Friends", xlab="Index of Friend")

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

24 # 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)),
29      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.
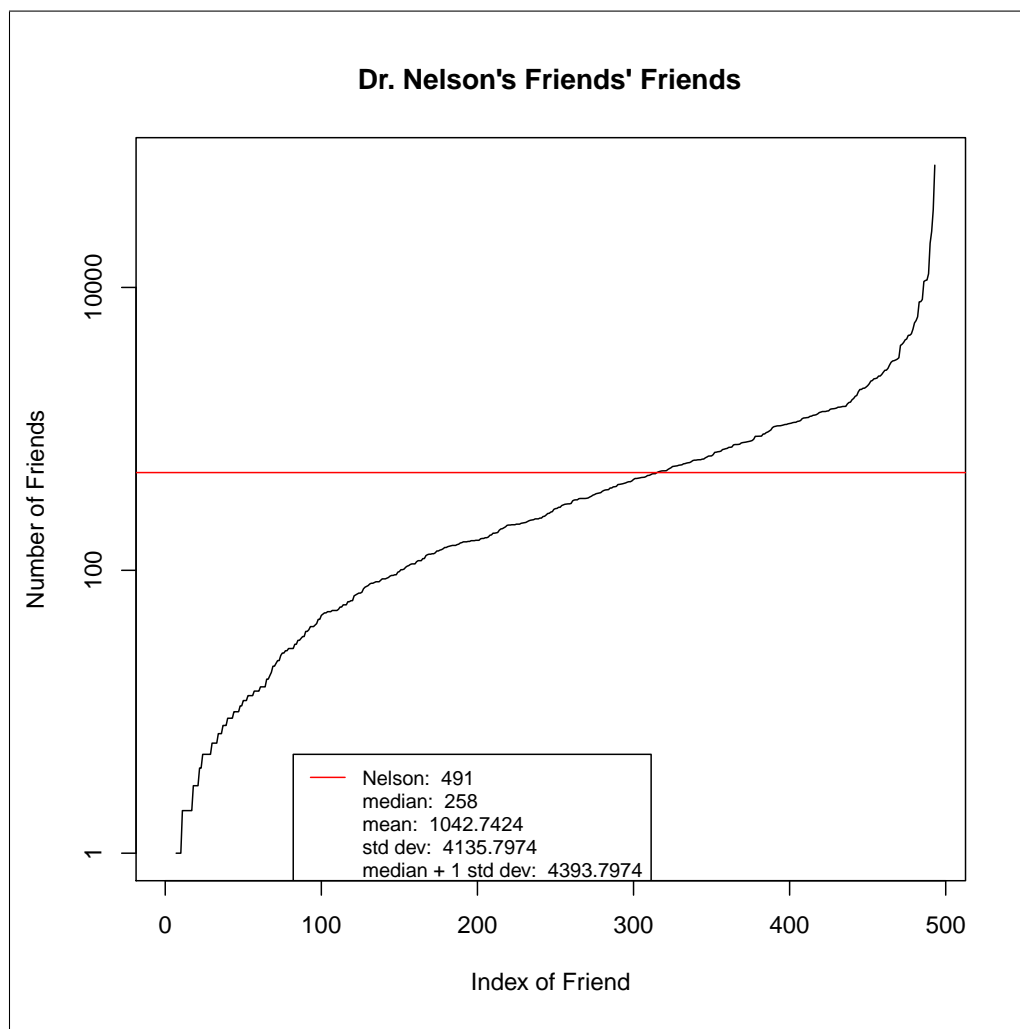

Figure 4: The Friendship Graph for Twitter Followers

Problem 3

Question

Repeat question #1, but with your LinkedIn profile.

Answer

Fortunately, *linkedIn* provides information about connections (people) through an API (`https://www.linkedin.com/secure/developer`. I have written a Python program, *qet_linkedin*, that requests data from *linkedIn* and saves the output into linkedin_count.

```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
import oauth2 as oauth
import time

consumer = oauth.Consumer(
    key="",
    secret="")
token = oauth.Token(
    "",
    secret=""
    )
client = oauth.Client(consumer, token)

url = "http://api.linkedin.com/v1/people/~/connections:(num-connections,first-name,last-name)"

resp, content = client.request(url)
soup = BeautifulSoup(content)
i = 0
all = 0
print '%-15s %-20s' %('Connections','LinkedIn User Name')
for message in soup.find_all('person'):
    all += 1
    if len(message.find_all('num-connections')) > 0:
        i += 1
        fullName = message.find('first-name').text+' '+ message.find('last-name').text
        print '%-15s %-20s' %(message.find('num-connections').text, fullName)
        with open('linkedin', 'a') as outfile:
            outfile.write('%-15d %-20s\n' %(init_url['friends_count'],init_url['screen_name'].encode('ascii', 'replace')))

print "\nNumber of Naina Sai's LinkedIn connections is "+str(all)+'(can not retrieve data of '+str(all - i)+' connection(s)) '
```

Listing 8: get_linkedin.py

I would like indicate here that the API that LinkedIn provides returns no more than 500 connection. In other words, if I have someone in my connection has more than 500 connections. LinkedIn will give me this person has 500+.

The `linkedin_count` file was ordered in place with the Unix command in Listing 9.

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

Listing 9: Sort command

This file was then processed by the R script shown in Listing 10 to produce the graph in Figure 5

```
#! /usr/bin/Rscript

# read data
data <- read.table('D:/cs532/a4/q3/linkedin_count', sep=",")
x <- seq(1, length(data$V1))
y <- data$V1

# get notable values
idx <- grep("Naina Sai Tipparti", 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/q3/linkedin_plot.pdf")
plot(x, y, type="l", log="y", pch=10, main="Naina Sai Tipparti Linkedin Friends",
    ylab="Number of Friends", xlab="Index of Friends")

# illustrate points of interest
abline(h=data$V1[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("Naina Sai Tipparti: ", data$V1[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 10: Graph Creation Script for LinkedIn

Mean	156.875
Median	95
Std Dev	154.6315

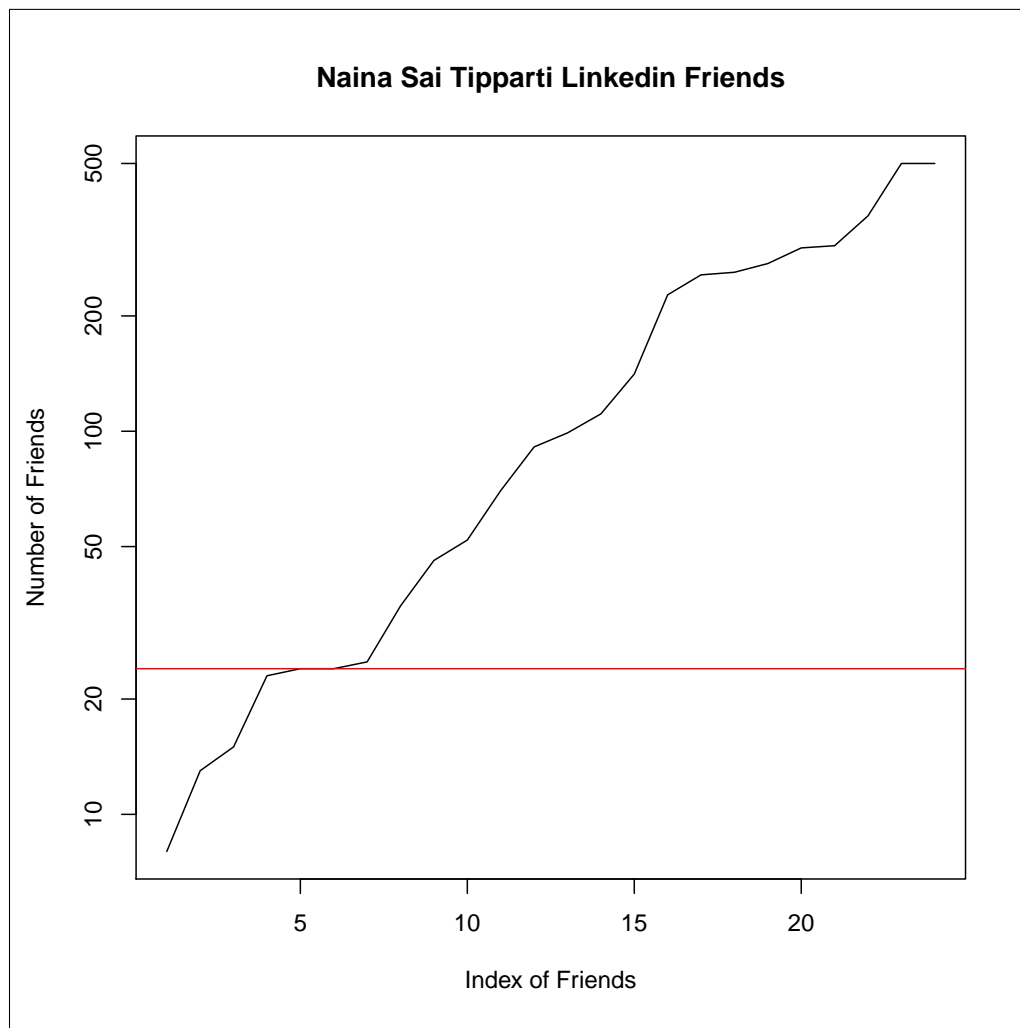Table 3: Statistics on the count of Naina Sai Tipparti's LinkedIn, values straight from R

Figure 5: The Friendship Graph for LinkedIn

I have number of connections which is a way less than the mean, but we can not conclude from the above statistics that the friendship paradox holds for My LinkedIn account since must of my connections have connections greater the mean.

Problem 4

Question

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

Answer

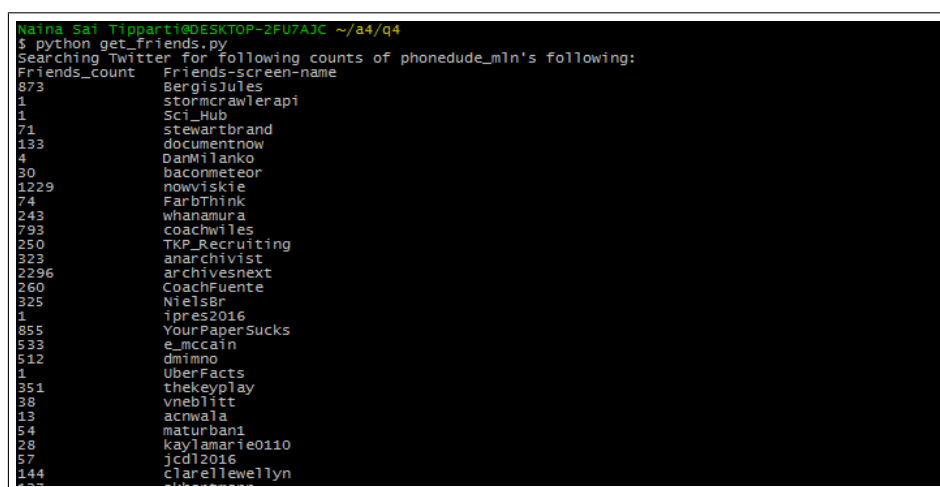Same instructions explained in question 2 are used to answer this question. As I remember, only one change has been made to *get_followers.py*.

- 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 11: 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 12: 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:

Figure 6: Sample output of number of following

Mean	484.9214
Median	225
Std Dev	729.5275

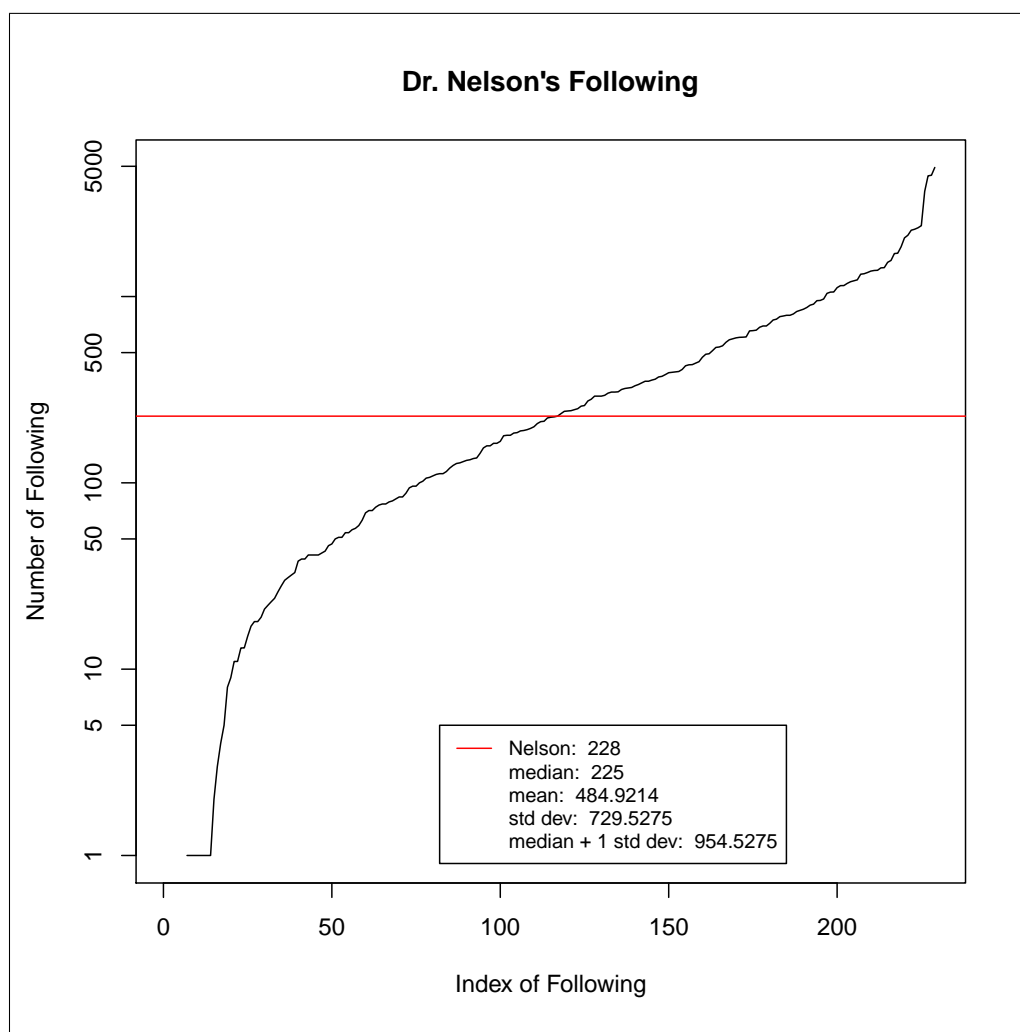Table 4: Statistics on the count of Dr.Nelson Following, values straight from R

Figure 7: 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.