Web Science cs532-s16

Assignment 4 Report

Dr. Michael L. Nelson

By: Huan Huang
02/25/2016

Problem 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

For this problem, I had to parse the graphml file provided by Dr. Nelson. To do this, I received help from a fellow classmate Zhetan Li, who taught me how to use *pygraphml*. We had to tell the system to operate with UTF-8 coding since the file is coded with UTF-8 instead of ascii. Then we just use *pygraphml* as a parser and call for the needed attributes in each node.

```
1  from pygraphml import Graph
2  from pygraphml import GraphMLParser
3  import codecs
4  import sys
5
6  reload(sys)
7  sys.setdefaultencoding('utf-8')
8
9  savefile = open('faceFriendCount.txt', 'a')
10
11 parser = GraphMLParser()
```

```
g = parser.parse("mln.graphml")

nodes = g.nodes()
count = 0

for node in nodes:
  count = count + 1
  try:
    friendName = node['name']
    friendCount = node['friend_count']
    print friendCount, friendName
    savefile.write('{},{}'.format(friendCount, friendName))
    savefile.write('\n')
  except KeyError:
    pass

mainUserName = "Michael L. Nelson"
print count, mainUserName
savefile.write('{},{}'.format(count, mainUserName))
```

To make the graph more comprehensive later, I sorted the list by numeric value. It is done in the terminal with the sort -n command, the output is saved in a file called sortedfacefriend.txt.

```
143,Fran Austin
144,Cynthia Suarez Eudy
147,Paul Ayris
155,Sally Jo Cunningham
165,Cathy Marshall
165,Michael L. Nelson
168,Julianne Allen
168,Lisa Owsley Rippy
170,Laura Garlow Yarborough
172,Brenda F. Nelson
181,Miriam Blake
```

Figure 1: Sample of part of sortedfacefriend.txt

Next, I load this sorted file into R and plotted the graph. The mean, median, and standard deviation are also calculated in R.

```
pdf("FaceFriends.pdf")

data2 = read.table(file.choose(), head=F, sep = ",")


plot(type="p", data2$V1, xlab = "Friends", ylab="Friend Counts",
     main = "Facebook Friendship Paradox", xlim=c(0, 160),
     ylim=c(0, 3500), las=0, col="red", pch=16, cex=0.3)
muser=which(data2$V2 == "Michael L. Nelson")
abline(v=muser, col=84, lty = 1)


legend(x=55,y=3400,c("Michael L. Nelson"),
       col = 84, lty = 1)
```

```
15
16 mean(data2$V1)
17 #357.74
18
19 median(data2$V1)
20 #259
21
22 sd(data2$V1)
23 #370.7
24
25 dev.off()
```

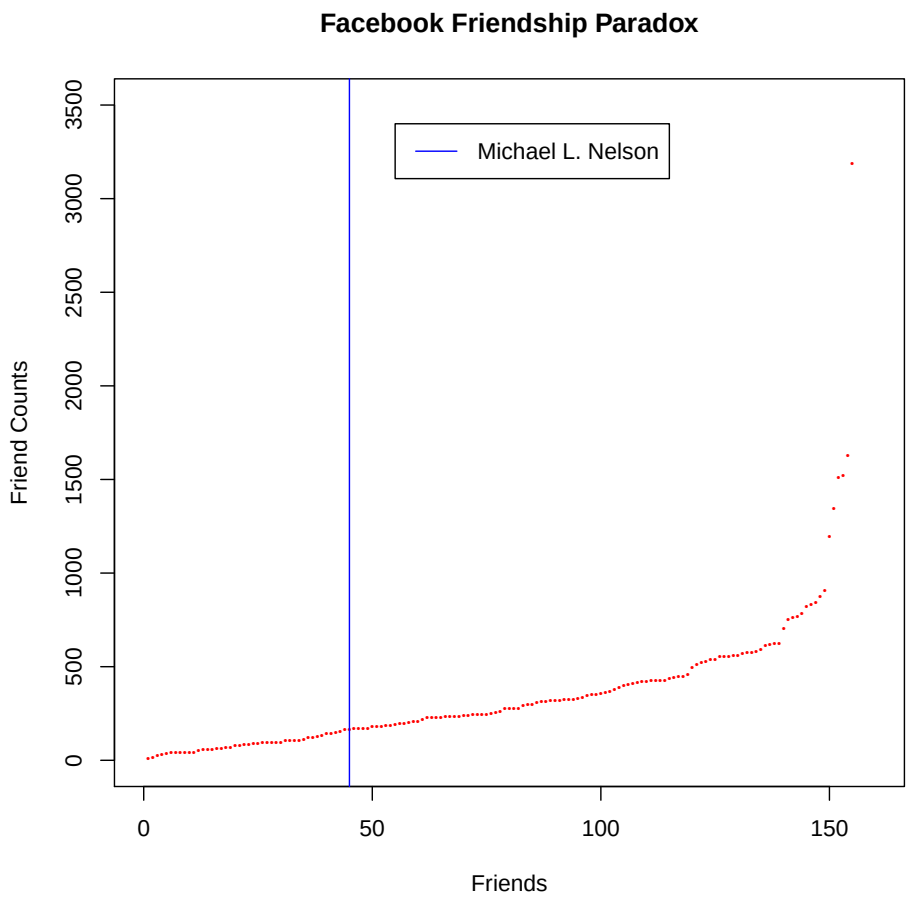

Figure 2: A dot graph of friend counts per friend

Dr. Nelson's Friends: 165

Mean: 357.74

Median: 259

Standard Deviation: 370.7

There are 11 individuals who do not have the "friend_count" attribute in their nodes. Therefore, they are not included in the graph and calculations.

The Friendship Paradox holds true for Dr. Nelson's Facebook account, which is to be expected. The only way for the paradox to be false is if every single one of your friends only have one friend, which is you, then the count of your friends' friends is equal to the count of your friends.

Problem 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 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

To solve this problem, I revisited a program from assignment 2 and made some changes. This time, instead of stream listen for links in people's tweets, we are looking for the followers count of "phonedude_mln" and its followers. I decide to use tweepy this time, because it seems to be simpler and more suited to handle this problem. First, gain access with my consumer and access keys and tokens. Then, I get the follower count of "phonedude_mln" and write it to a file along with the screen name, which is "phonedude_mln". Next, I get phonedude_mln's followers and the follower count for each of them. I did encounter an issue at this part. To protect its server from overwhelming traffics, Twitter has set limitations on access rate. Since I used tweepy's cursor function, I am limited to 15 calls in a period of 15 minutes. However, with 15 calls, I was only able to get 300 follower counts before being kicked out by Twitter(phonedude_mln had 489 followers at that time, the number has increased to 491 the next day). I did try to solve this issue by setting a sleeping period of 60s*15, but it gives me issues as well. At the end, I solved this problem by increasing the size of each call page. Tweepy cursor has a default setting of 20 items per call page, therefore, all I had to do is to increase the items per page. Just because I can, I set the page size to

its maximum, which is 200, and I was able to get all of the follower screen names and their follower counts in 3 calls.

```
#!/usr/bin/python
import time
import tweepy
from tweepy import OAuthHandler


savefile = open('followersCount.txt','a')

# load my API credentials

accesstoken = "110800099-ZiXf6imxd24OVqw5nnm4nkHXmVwhX46QQI0Wb3Zt"
accesstokensecret = "FlDFz2Ol035FONaiL9uTzfqRauVArVwwVMuxiHDWQsmwo"
consumerkey = "1N9TkDHB9g4UIbjVj5j6idqnn"
consumersecret = "5Yjw1hRViWd5TloKLmEibPoregSltxO1bY7YJckjnecuSw4n9B"

# create twitter API object

auth = tweepy.OAuthHandler(consumerkey, consumersecret)
auth.set_access_token(accesstoken, accesstokensecret)


api = tweepy.API(auth)
mainuser = api.get_user('phonedude_mln')
userName = mainuser.screen_name
followersCounts = mainuser.followers_count
print followersCounts, userName
savefile.write('{:<10} {}'.format(followersCounts, userName))
savefile.write('\n')

users = tweepy.Cursor(api.followers, userName, count=200).items()

for follower in users:
  try:
    fcount = follower.followers_count
    fname = follower.screen_name
    print str(fcount) + " " + fname
    savefile.write('{:<10} {}'.format(fcount, fname))
    savefile.write('\n')
  except tweepy.error.RateLimitError:
    monitor_rate_limit=True, wait_on_rate_limit=True
    continue
  except StopIteration:
    break
```

Again, I sorted the list by by using sort -n and saved to a file called sortedCount.txt.

```
308 468        tombortreasure
309 473        HistWebArchives
310 479        webrecorder_io
311 481        Philipp_Mayr
312 482        Jakdemir
313 489        phonedude_mln
314 498        bethcron
315 501        tsuomela
316 504        aag1091
317 505        mousta
318 505        NielsBr
```

Figure 3: Sample of part of sortedCount.txt

With this sorted file, I loaded the data into and plotted the graph. Below are the R code and graphs.

```
pdf("TwitterFollowerslg.pdf")

data = read.table(file.choose(), head=F, sep = "")


plot(type="p", data$V1, xlab = "Followers", ylab="Follower Counts",
     main = "Twitter Friendship Paradox", xlim=c(0, 500),
     ylim=c(1, 75000), las=0, col="red", pch=16, cex=0.3, log='y')
muser=which(data$V2 == "phonedude_mln")
abline(v=muser, col=84, lty = 1)
y = c(0, 10, 100, 1000, 10000, 100000)

legend(x=.1,y=70000,c("Michael L. Nelson"),
       col = 84, lty = 1)

mean(data$V1)
#1045.867

median(data$V1)
#259

sd(data$V1)
#4146.195

dev.off()
```

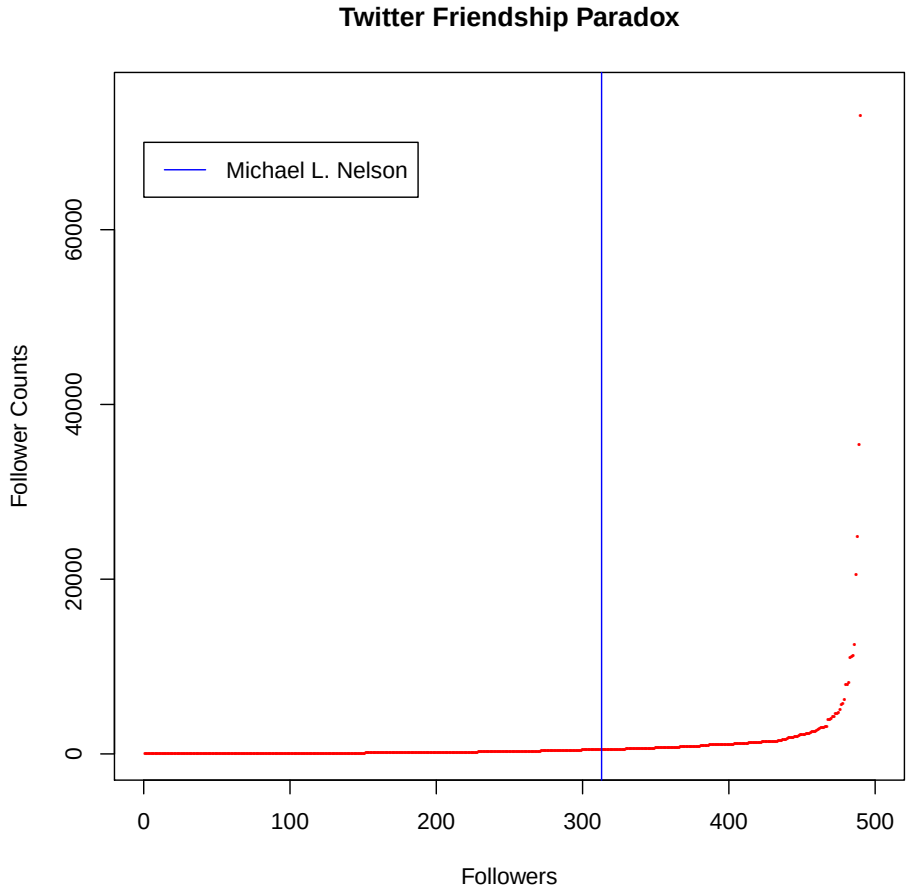

Figure 4: A dot graph of follower counts per follower

Because of few individuals who have many more followers than other users, it made this graph hard to read. To make this graph more comprehensive, I changed the y axis to log scale. However, logarithm does not work with 0, therefore, the six individuals who have 0 follower are not included in this graph.

```
1 0          ahmedkalka97
2 0          CrowNaito
3 0          hdo003
4 0          katjakra
5 0          unknown_6
6 0          witch_doctor81
7 1          amaranaas
```

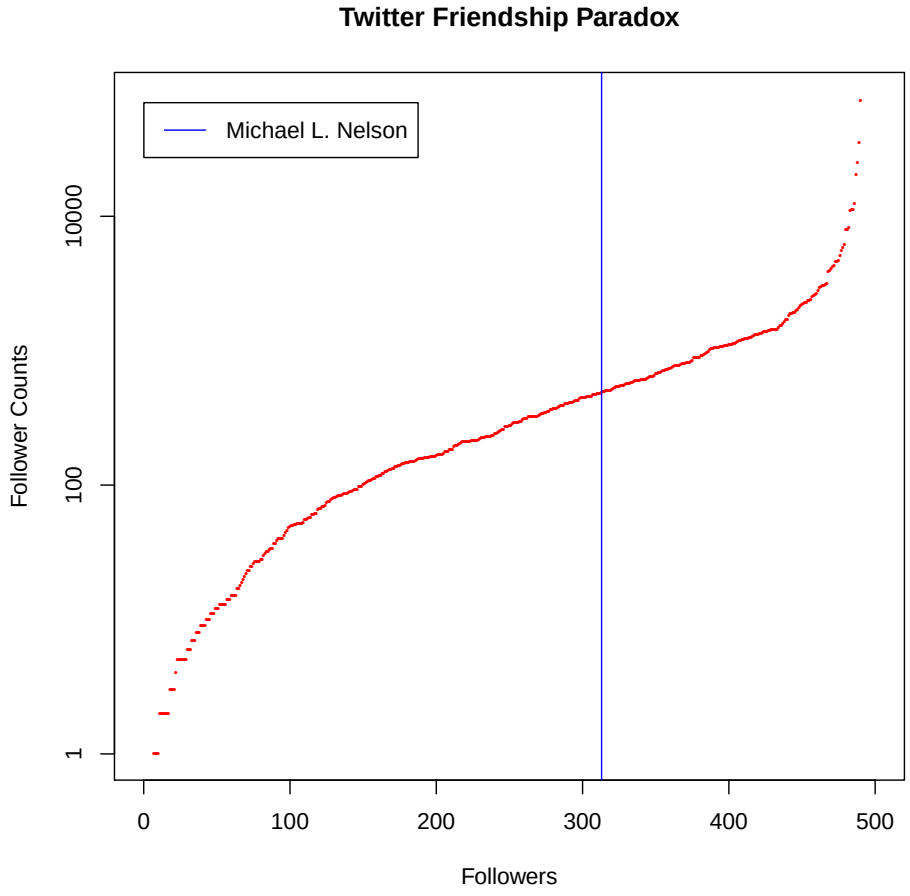

Figure 5: Second dot graph of follower counts per follower

Dr. Nelson's Followers: 489

Mean: 1045.867

Median: 259

Standard Deviation: 4146.195

Interesting enough, the median for Dr. Nelson's followers' follower counts is also 259. I thought that I made a mistake at first, but after checking several times, I am certain it is 259. However, the number has change now since Dr. Nelson has gained few more followers.

The Friendship Paradox still holds true for Dr. Nelson's Twitter account, which is not a surprise. However, we are using the following system on Twitter, which does not go both ways, this paradox has more ways to break now. For example, if every follower of Dr. Nelson has 0 follower, then Dr. Nelson would have more "friends" than his followers. But we all know this is very unlikely to happen in actual world, so it is save to say that Friendship Paradox still works for twitter.