CS-432/532 Introduction to Web Science:
Assignment #4:
The "Friendship Paradox"

Due on Thursday, February 25, 2016

Dr. Michael L. Nelson

Plinio Vargas
pvargas@cs.odu.edu

Contents

List of Figures

Listings

List of Tables

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/`

1.1 Approach

EXTRACTING DATA

Dr. Nelson's facebook account data was utilized to solve this problem. The data was in a XML structure in the file $< mln.graphml >$; a Python XML parser was employed to parse its content. The object was stored in variable *root* (line 21) and all **number-of-friends** of Dr. Nelson's friends were stored in a dictionary (lines 25-30). Finally, this object was sorted and written into a file: $< ffriendship_paradox.dat >$ (lines 35-42).

Dr. Nelson's total **number-of-friends** in the structure was 165. However, there were 10 nodes within the structure that did not have any **friends_count** (10). They were not considered in the data, so the total number of friends in this sample is 155.

Listing 1: GetFriends.py

```
20 tree = Tree.parse('mln.graphml')
21 root = tree.getroot()
22 ns = {'structure': 'http://graphml.graphdrawing.org/xmlns'}
23 my_friends = root[len(root) - 1].findall('structure:node', ns)
24 friend = 0
25 friend_dict = {0: len(my_friends)}
26 for child in my_friends:
27     friend += 1
28     for element in child.findall('structure:data', ns):
```

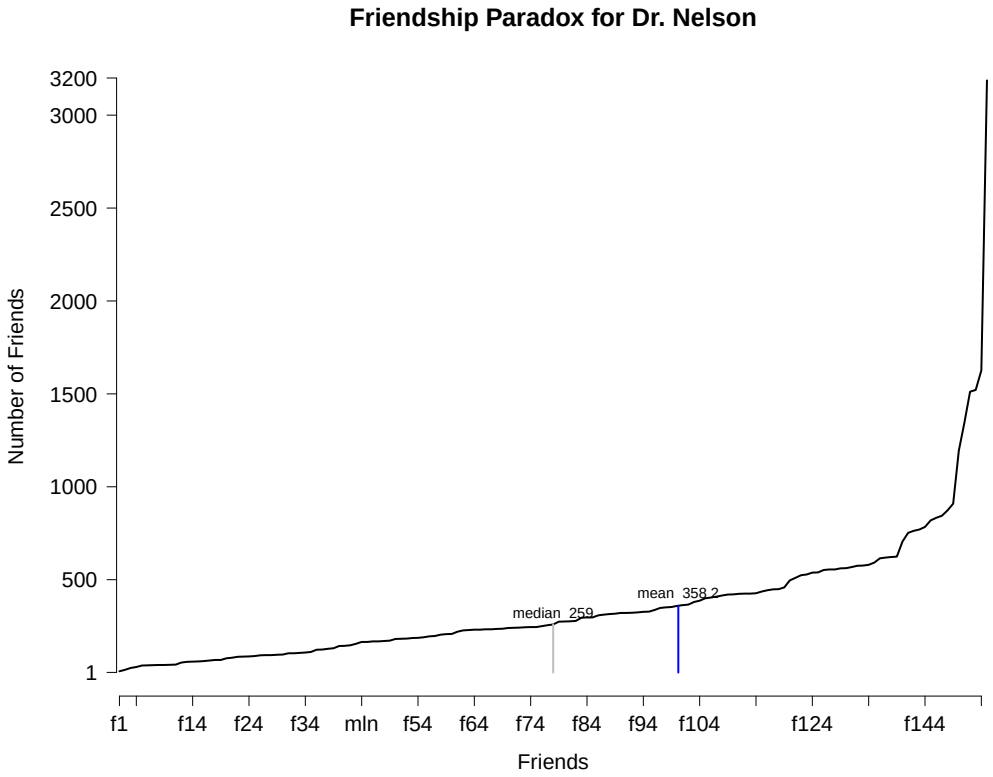
Problem 1 continued on next page...

```
        if element.attrib['key'] == 'friend_count':
            friend_dict[friend] = int(element.text)

print('Total Friends: %d, Missing Data: %d' % (len(my_friends), len(my_friends) -
    len(friend_dict)))
print(sorted(friend_dict.items(), key=operator.itemgetter(1)))
friend = 0
with open('ffriendship_paradox.dat', 'w') as file:
    file.write('Friend\tNo.Friends\n')
    for friend_tuple in sorted(friend_dict.items(), key=operator.itemgetter(1)):
        friend += 1
        if friend_tuple[0] == 0:
            file.write('mln\t%d\n' % friend_tuple[1])
        else:
            file.write('f%d\t%d\n' % (friend, friend_tuple[1]))
```

1.2 Solution

Figure 1: Dr. Nelson's Facebook Friends Graph



The data was sorted by **Number-of-Friends** field (see Table 1), the graph shows characteristics of a power-law distribution. The **Number-of-Friends** corresponding to *mln* is 165 and *mln* is positioned at the left side of the median 259. Thus, the friendship paradox holds true for *mln* Facebook account.

The mean, standard deviation, and median were computed using R. The file $< facebook.R >$ is the script that makes the computation and graph plotting. The file is attached to this document. Following is the calculation for the number of friends that *mln* friends has:

- Mean: 358.2
- σ: 259
- Median: 370.7

σ has a high value. It is skewed due to the magnitude of the last 5 friends. See Table 1. However, even if we were to disregard this data, *mln*'s Number of Friends are so far left from the mean and the median that the friendship paradox holds true for *mln* Facebook account, in which his friends have more friends than he.

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 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".

2.1 Approach

EXTRACTING DATA

My sister's Twitter account was employed to test if the paradox holds. A REST Twitter API [1] *cURL* command was used to execute the request:

```
curl --get 'https://api.twitter.com/1.1/followers/list.json' --data
    'amp%3Binclude_user_entities=false&amp%3Bscreen_name=
    roselenavargas&amp%3Bskip_status=true&cursor=-1'
    --header 'Authorization: OAuth oauth_consumer_key="L9IMyConSuMerKeyT1MC",
     oauth_nonce="33b5ab398f8145ac453017896ebe99a1",
     oauth_signature="IQgpTL2PBvHzeu%2BKa7veW7EIxrw%3D", oauth_signature_method="HMAC-SHA1",
     oauth_timestamp="1456395318", oauth_token="412664119-
     YJ5IeQTheAuThoRizaTionToken", oauth_version="1.0"' --verbose > twitter
```

The result was redirected to a file $< twitter >$. Since the result is a JSON object, the original approach was to utilize a Python library, such as *pickel* or *json* to create a structure to iterate and extract its data. Both approaches came empty, in which the number of followers extracted was less than in the data structure. This most likely was due to some type of encoding problem in the file. However, to move forward a third approach proved to be effective: the use of *regex*.

$< GetFollowers.py >$ is the Python program that helped extract the information needed to make the proof. The heart of the code is in line 25:

Listing 2: GetFollowers.py

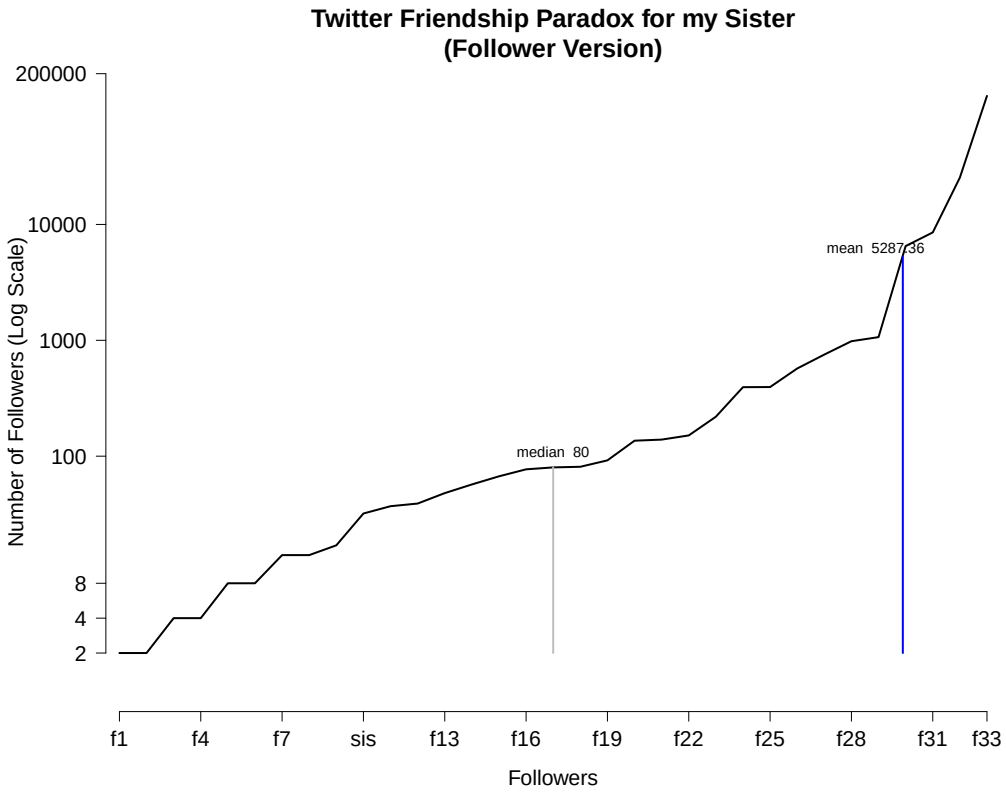
```
24 followers_dict = {}
25 for followers in re.findall('\"followers_count\":\d+', x):
26     counter += 1
27     followers_dict[counter] = int(re.findall('\d+', followers)[0])
28     print(counter, followers, re.findall('\d+', followers)[0])
29 followers_dict[0] = counter
```

Problem 2 continued on next page...

An iterable object was formed with the regular expression `'\"followers_count\":\d+'`, extracting followers_count amounts. Finally, my sister's information along her followers was placed in file $< twitter_paradox.dat >$. Table 2 shows the data set result.

2.2 Solution

Figure 2: Sister's Twitter Followers Graph



The data was sorted by **Number-of-Friends**, the graph shows characteristics of a power-law distribution. The **Number-of-Friends** corresponding to *sis* is 32 (see Table 2). Since *my sister* on the x-axis is positioned at the left side of the median 259 and the mean 5287, then the friendship paradox holds for my sister Twitter account, in which her friends have more followers than she.

The mean, standard deviation, and median were computed using R. The file $< twitter.R >$ is the script that makes the computation and graph plotting. The file is attached to this document. Following is the calculation for the number of friends that my sister's friends have:

- Mean: 5,287.35
- σ: 22,623
- Median: 80

Problem 3 Extra Credit

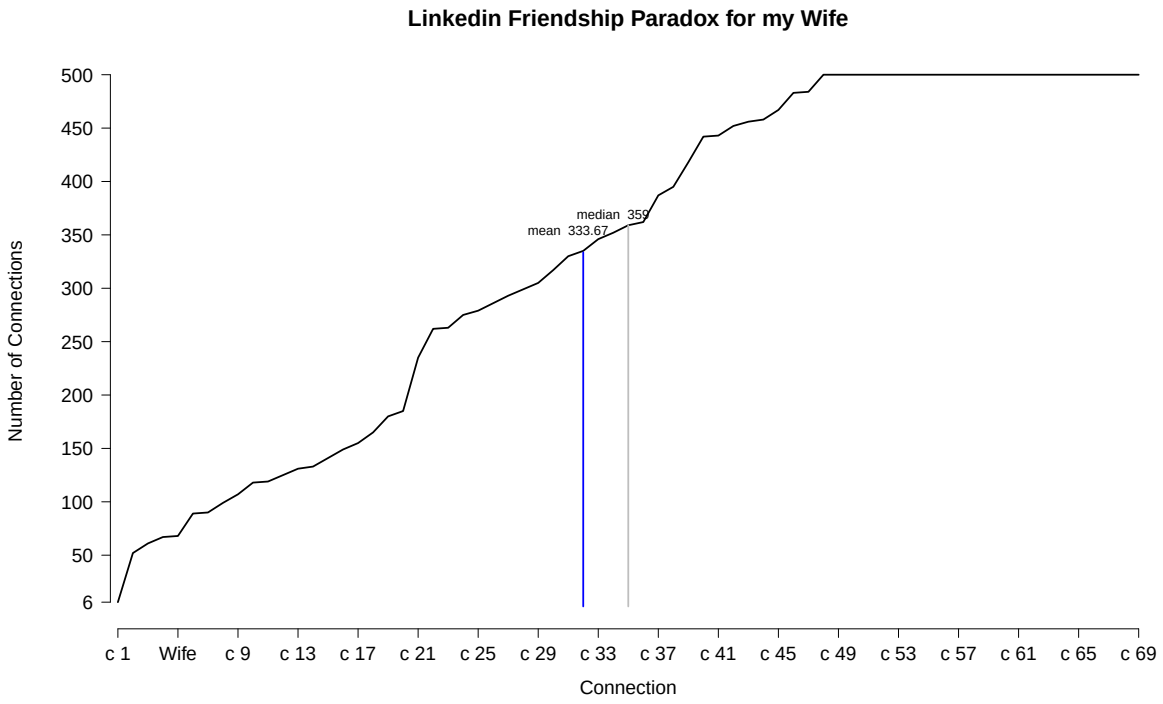
Repeat question #1, but with your LinkedIn profile.

3.1 Approach

All the connections from my wife's LinkedIn account were downloaded in file $< source.txt >$. I couldn't cross reference the internal connection ID with their public ID, so a manual computation on the number of connections were employed to get the data.

3.1 Solution

Figure 3: Wife Linkedin Connection Graph



The data was sorted by **Connection Count**. The **Connection Count** corresponding to $Wife$ is 68 (see Table 3) and $my\ wife$ is positioned at the left side of the median 359. Then, the friendship paradox holds for my wife's Linkedin account in which her connections have more connections than her.

The mean, standard deviation, and median was computed using R. The file $< linkedin.R >$ is the script that makes the computation and graph plotting. The file is attached to this document. Below the calculation results:

- Mean: 333.67
- σ: 163.2
- Median: 359

Problem 3 Extra Credit

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

4.1 Approach

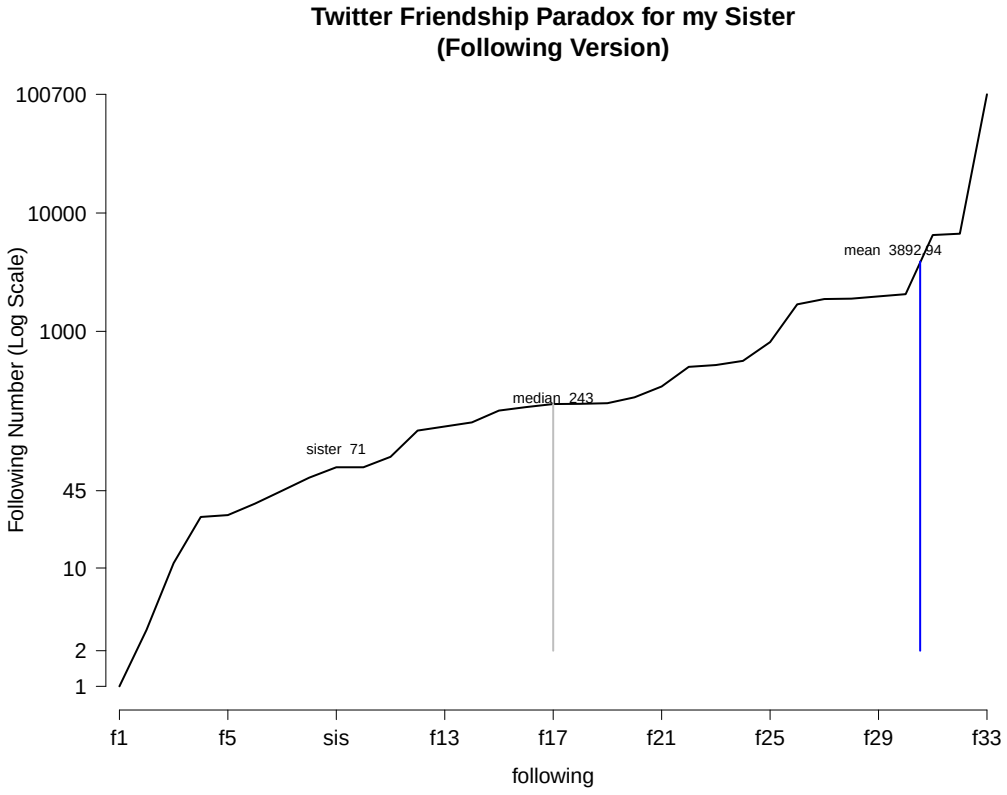
The same approach as in 2.1 was followed. The only difference was that instead of using **follower_count** as in the *regex* filter, we used **friends_count** which points to the number of particular person may be following. See listing below:

Listing 3: GetFollowing.py

```
following_dict = {counter: 71}
for following in re.findall('\"friends_count\":\d+', x):
    counter += 1
    following_dict[counter] = int(re.findall('\d+', following)[0])
    print(counter, following, re.findall('\d+', following)[0])
```

4.1 Solution

Figure 4: Sister's Twitter Following Friends Graph



Plotting the data sorted by **Number-Following**, the graph shows characteristics of a power-law distribution. The **Number-Following** corresponding to *sis* is 71 and *my sister* is position at the left side of the median 243. Then, the friendship paradox holds for my sister Twitter account in which her friends are following more people than she.

The mean, standard deviation, and median were computed using R. The file $< twitter2.R >$ is the script that makes the computation and graph plotting. The file is attached to this document. Following is the calculation for the number of friends that my sister's friends have:

- Mean: 3,892.94
- σ: 17,449
- Median: 243

Table 1: Facebook Number of Friends that Dr. Nelson's friends have

Friend	No. Friends	Friend	No. Friends	Friend	No. Friends	Friend	No. Friends
f1	7	f48	170	f95	328	f142	763
f2	15	f49	172	f96	337	f143	770
f3	25	f50	181	f97	348	f144	784
f4	30	f51	182	f98	351	f145	819
f5	38	f52	183	f99	353	f146	833
f6	39	f53	186	f100	359	f147	844
f7	40	f54	187	f101	363	f148	873
f8	41	f55	190	f102	366	f149	909
f9	41	f56	195	f103	380	f150	1194
f10	42	f57	197	f104	387	f151	1346
f11	43	f58	204	f105	400	f152	1512
f12	54	f59	207	f106	404	f153	1521
f13	58	f60	208	f107	409	f154	1626
f14	59	f61	220	f108	415	f155	3187
f15	60	f62	227	f109	420		
f16	62	f63	229	f110	421		
f17	65	f64	231	f111	424		
f18	68	f65	231	f112	425		
f19	68	f66	233	f113	425		
f20	77	f67	233	f114	427		
f21	80	f68	235	f115	436		
f22	85	f69	236	f116	443		
f23	86	f70	240	f117	448		
f24	87	f71	241	f118	449		
f25	89	f72	242	f119	458		
f26	93	f73	244	f120	496		
f27	94	f74	245	f121	510		
f28	94	f75	245	f122	524		
f29	96	f76	250	f123	528		
f30	97	f77	255	f124	538		
f31	104	f78	259	f125	539		
f32	104	f79	274	f126	552		
f33	106	f80	275	f127	555		
f34	108	f81	276	f128	555		
f35	111	f82	278	f129	561		
f36	123	f83	295	f130	562		
f37	124	f84	297	f131	568		
f38	128	f85	297	f132	575		
f39	131	f86	308	f133	576		
f40	143	f87	312	f134	580		
f41	144	f88	315	f135	592		
f42	147	f89	317	f136	615		
f43	155	f90	321	f137	619		
mln	165	f91	321	f138	622		
f45	165	f92	322	f139	624		
f46	168	f93	324	f140	705		
f47	168	f94	327	f141	752		

Data was sorted by Number of Friends that *mln* has. This data was use to plot Figure 1

Table 2: Twitter Followers Data

Friend	Followers
f1	2
f2	2
f3	4
f4	4
f5	8
f6	8
f7	14
f8	14
f9	17
sister	32
f11	37
f12	39
f13	48
f14	57
f15	66
f16	77
f17	80
f18	81
f19	92
f20	136
f21	139
f22	151
f23	220
f24	394
f25	395
f26	571
f27	753
f28	981
f29	1066
f30	6504
f31	8527
f32	25361
f33	128602

Data for Figure 2.

Table 3: My Wife's Linkedin Connection Data

Connection	Count
C1	6
C2	52
C3	61
C4	67
Wife	68
C6	89
C7	90
C8	99
C9	107
C10	118
C11	119
C12	125
C13	131
C14	133
C15	141
C16	149
C17	155
C18	165
C19	180
C20	185
C21	235
C22	262
C23	263
C24	275
C25	279
C26	286
C27	293
C28	299
C29	305
C30	317
C31	330
C32	335
C33	346
C34	352
C35	359
C36	362
C37	387
C38	395
C39	418
C40	442
C41	443
C42	452
C43	456
C44	458
C45	467
C46	483
C47	484
C48	500
C49	500
C50	500
C51	500
C52	500
C53	500
C54	500
C55	500
C56	500
C57	500
C58	500
C59	500
C60	500
C61	500
C62	500
C63	500
C64	500
C65	500
C66	500
C67	500
C68	500
C69	500

Data for Figure 3. It was sorted by Connection Count.

Table 4: Twitter Following Friends Data

Friend	Following
f1	1
f2	3
f3	11
f4	27
f5	28
f6	35
f7	45
f8	58
sis	71
f10	71
f11	87
f12	145
f13	157
f14	170
f15	214
f16	229
f17	243
f18	244
f19	247
f20	277
f21	342
f22	501
f23	519
f24	563
f25	811
f26	1691
f27	1875
f28	1889
f29	1974
f30	2061
f31	6514
f32	6687
f33	100677

Data for Figure 4. It was sorted by number of friends a friend is following.

References

[1] Twitter API. (n.d.) Retrieved February 24, 2016, from `https://dev.twitter.com/docs/api/1.1/get/followers/list`