Assignment Five

Sybil Melton

October 15, 2014

Contents

List of Figures

List of Tables

Listings

1 TWITTER FRIENDS

Explore the friendship paradox for your Twitter account.

Create a graph of the number of friends (y-axis) and the friends sorted by number of friends (x-axis).

Do include yourself in the graph and label yourself accordingly. Compute the mean, standard deviation, and median of the number of friends that your friends have.

1.1 SOLUTION

I was able to modify the code from a previous Twitter assignment to extract the number of my friends' friends. I set the count to 200 because the documentation stated that is the maximum number of records can be extracted per page. Since I have less than 200 friends, I only had to extract one page. [3] Because Twitter sends the friend count for each friend, the resulting json was put into a list and parsed. The list was sorted by count and written to a text file using the codecs module, to account for friends with non-ascii characters in their names. [8] Listing 1 is the python program used to extract the friend counts, perform the calculations, and write the results to file. The output files "counts.txt" and "calculations.txt" are included with this report.
Functions were written in order to calculate the required mean, median, and standard deviation. [9] Listing 2 is the python program written for these calculations. Because I am new to Twitter, I mostly follow celebrities and companies and this is the reason for the high mean and high standard deviation. The regular people I follow on Twitter are in the twenty-fifth percentile and I am in the sixteenth percentile. Table 1.1 shows the calculation results for Twitter.

Mean	1989.93
Median	499
Std. Dev	4088.53

Table 1.1: Twitter Statistics

R was used to create the graph from the output file and is Figure 1.1. A large dot was placed on the graph with the text "Me" to denote my position on it. [7] I am at position 12 of 73, so most of my friends have more friends than I do. The majority of friends with fewer friends are celebrities, but my sister and a college friend still have fewer than I do. The file "a5.r" lists the code used to create the graph and is included in this report.

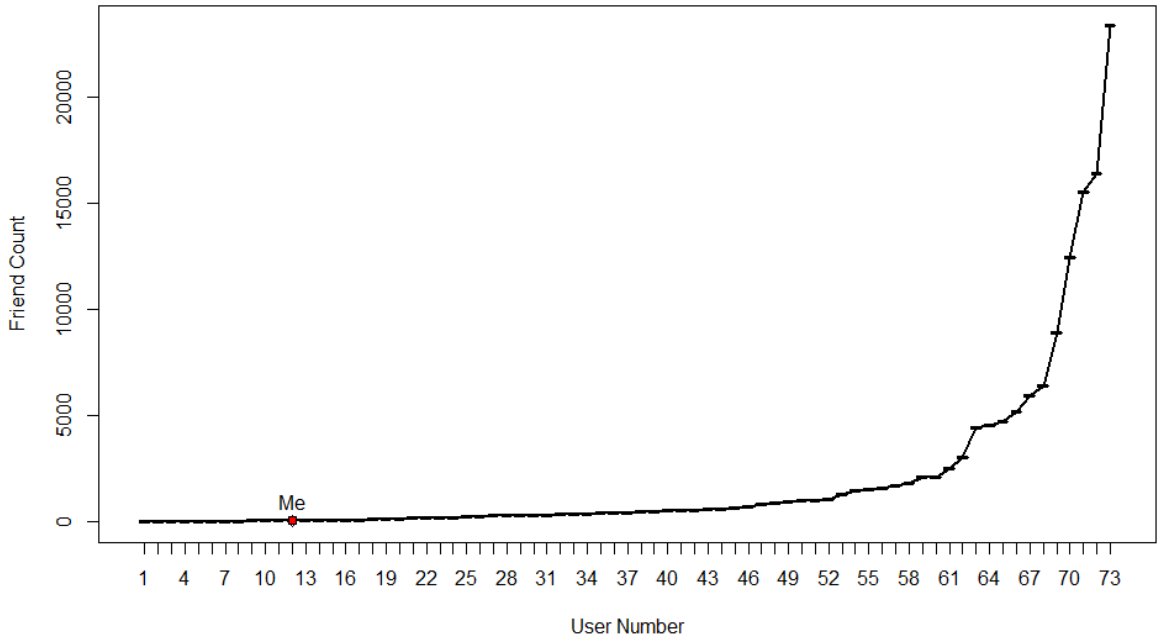

Figure 1.1: Twitter Number of Friends per Friend

1.2 Python Code

```
from __future__ import unicode_literals
import requests
from requests_oauthlib import OAuth1
import urllib
import json
import codecs

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

def getUser(user, page):
  params = urllib.urlencode({'cursor' : page,'screename': user, 'count': 200, 'skip_status':'true', '
      include_user_entities':'false'}) #https://docs.python.org/2/library/urllib.html
  urli = "https://api.twitter.com/1.1/friends/list.json?"
  full_url = urli+params
  r = requests.get(url=full_url, auth=oauth)
  data = json.loads(r.text)
  slist = list(data[u'users'])
  parse_List(slist, user)

def parse_List(slist, user):
  ulist = []
  for i in range(len(slist)):
    ulist.append({'name':slist[i][u'name'], 'count': slist[i][u'friends_count']})
  # add the searched user
  ulist.append({'name': user, 'count': len(ulist)})
    #sort list by user count
  sorted_u = sorted(ulist, key=lambda k : k['count'])
  writeToFile(sorted_u)

def writeToFile(ulist):
  with codecs.open('counts.txt', 'w', encoding='utf-8') as O:
```

```
36      O.write('No.\tName\tCount\n')
37      for i in range(len(ulist)):
38        O.write(str(i+1)+'\t'+ulist[i][u'name']+'\t'+str(ulist[i][u'count'])+'\r')
39  #compute and save mean, median, and standard deviation
40    with open('calculations.txt', 'a') as O:
41      clist = []
42      O.write('Twitter\n')
43      for u in ulist:
44        clist.append(u['count'])
45      mean = calculate.getMean(clist)
46      O.write('Mean: ' + str(mean) + '\n')
47      median = calculate.getMedian(clist)
48      O.write('Median: ' + str(median) + '\n')
49      stdev = calculate.standardDev(clist)
50      O.write('Standard Deviation: ' + str(stdev) + '\n')
51
52  CONSUMER_KEY = "NlcSlE8LpTNQ6HOvyunMVx4EN"
53  CONSUMER_SECRET = "fB5hyOpBG9cklmPHvGVPyOJALX7UEG3SPWjjnOaRXq3TgzRGkx"
54
55  OAUTH_TOKEN = "2815818894-fhOZj1FomjECCvDc62EPpxyKJTOktOuk5cl7BOR"
56  OAUTH_TOKEN_SECRET = "GIMhewOb2ZMjGOMgEddRlHzuFPQFbfAWn8mecw1yNrboK"
57
58  oauth = get_oauth()
59  getUser('LilyMotoko', -1)
```

Listing 1: Twitter Friend Counts

```
1
2   import math
3
4   def standardDev(clist):
5     mean = getMean(clist)
6     var = 0
7     for c in clist:
8       v = pow(c-mean, 2)
9       var += v
10    var = var/len(clist)
11    stdev = math.sqrt(var)
12    return stdev
13
14
15  def getMean(clist):
16    sum = float(0)
17    for c in clist:
18      sum += c
19    mean = sum/len(clist)
20    return mean
21
22  def getMedian(clist):
23    median = 0
24    # if list has even number of elements:
25    if len(clist) % 2 == 0:
26      m1 = clist[len(clist)/2]
27      m2 = clist[(len(clist)/2)+1]
28      median = (m1 + m2) /2
29    else:
30      median = clist[(len(clist)/2)+1]
31    return median
```

Listing 2: Calculations

2 FACEBOOK FRIENDS

Using your facebook account, repeat question #1 (if you have > 50 friends).

2.1 SOLUTION

I was able to extract my Facebook friends' friends using the code, "seleniumScrapePB.py", provided by classmate Alexander Nwala. Firefox is required to use it and gave me an error because I was working on a Centos server without a display. I used Xvfb to create an "X server that doesn't require connection to a physical display." [1] A Python program was written to parse and sort the resulting CSV file, "facebookFriendFriendsCountTuples.txt." [2] I added myself to the list and wrote the result to a file to be used to create the friendship graph. The output file "facebook.txt" is included with this report. Additionally, the mean, median, and standard deviation were computed using the same calculate program used for Twitter. The result of the calculations was appended to the same file with the Twitter calculations. Several of my friends and family members have over a thousand Facebook friends, which accounts for the high standard deviation. I calculated that I am in the tenth percentile, which is well below the average and median. Listing 3 is the python program used to convert the csv into the tabular, sorted text file and perform the calculations. Table 2.1 shows the calculation results for Facebook.

Mean	484.42
Median	290
Std. Dev	697.54

Table 2.1: Facebook Statistics

R was used to create the graph from the output file. A large dot was placed on the graph with the text "Me" to denote my position on it, shown in Figure 2.1. [7] I am at position 9 of 95, so most of my friends have more friends than I do. The majority of my friends with fewer number of friends are older family members whose only friends are other family members, so it makes sense. The code used to create the graph is also in file "a5.r", included in this report.

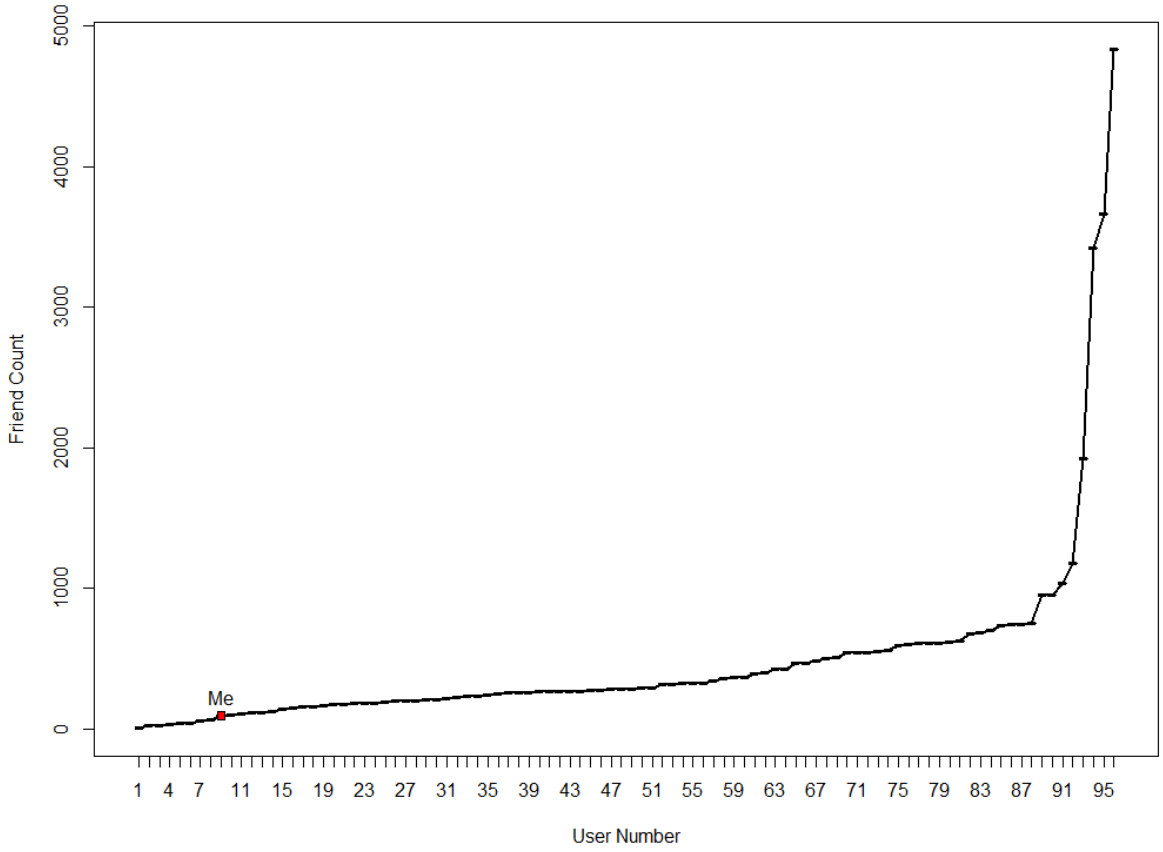

Figure 2.1: Facebook Number of Friends per Friend

2.2 PYTHON CODE

```
1
2  import csv
3  import calculate
4
5  def writeToFile(ulist):
6    #write facebook friends and counts
7    with open('facebook.txt', 'w') as O:
8      O.write('No.\tName\tCount\n')
9      for i in range(len(ulist)):
10       O.write(str(i+1)+'\t'+ulist[i]['name']+'\t'+str(ulist[i]['count'])+'\n')
11   #compute and save mean, median, and standard deviation
12   with open('calculations.txt', 'a') as O:
13     clist = []
14     O.write('Facebook\n')
15     for u in ulist:
16       clist.append(u['count'])
17     mean = calculate.getMean(clist)
18     O.write('Mean: ' + str(mean) + '\n')
19     median = calculate.getMedian(clist)
20     O.write('Median: ' + str(median) + '\n')
21     stdev = calculate.standardDev(clist)
22     O.write('Standard Deviation: ' + str(stdev) + '\n')
23
24 with open('facebookFriendFriendsCountTuples.txt', 'r') as csvfile:
25   ulist = []
26   reader = csv.reader(csvfile)
27   for row in reader:
28     if row[0] == "USER":
29       continue
30     else:
31       c = row[1].rstrip()
32       c = int(c)
33       ulist.append({'name':row[0], 'count':c})
34   ulist.append({'name':'me', 'count':len(ulist)})
35   sorted_u = sorted(ulist, key=lambda k : k['count'])
36   writeToFile(sorted_u)
```

Listing 3: Convert Output

3 LINKEDIN CONNECTIONS

Using your linkedin account, repeat question #1 (if you have > 50 connections).

3.1 SOLUTION

There are caveats to the Linkedin Connections API. It is easy to retrieve your own connection numbers, but a first-degree connection must have a third-party accessible profile. Additionally, Linkedin puts a cap on the number of connections reported. If a person has more then 500 connections, 500 is the number given. [6] I found that I had only 39 connections and 34 do not have a private profile. Five connections were retrieved with 500, so they have 500 or more. Although this is under the 50 requirement, I still wanted to see if it followed the friendship paradox. Only four of my connections have fewer connections, and two of those four are recent college graduates. This puts me in the thirteenth percentile. The mean, median, and standard deviation were computed again and the result was appended to the same file with the Twitter and Facebook calculations. Listing 4 is the python program connect to the Linkedin API, create the sorted text file, and perform the calculations. [4] Listing 5 is the python program used to set up the initial Oauth with the API. [5] Table 3.1 shows the statistics results for Linkedin.

Mean	211.55
Median	175
Std. Dev	157

Table 3.1: Linkedin Statistics

R was used to create the graph from the output file. A large dot was placed on the graph with the text "Me" to denote my position on it, shown in Figure 3.1. [7] I am at position 5 of 39, so as expected, most of my connections have more connections than I do. The code used to create the graph was also added to file "a5.r," which is included in this report.

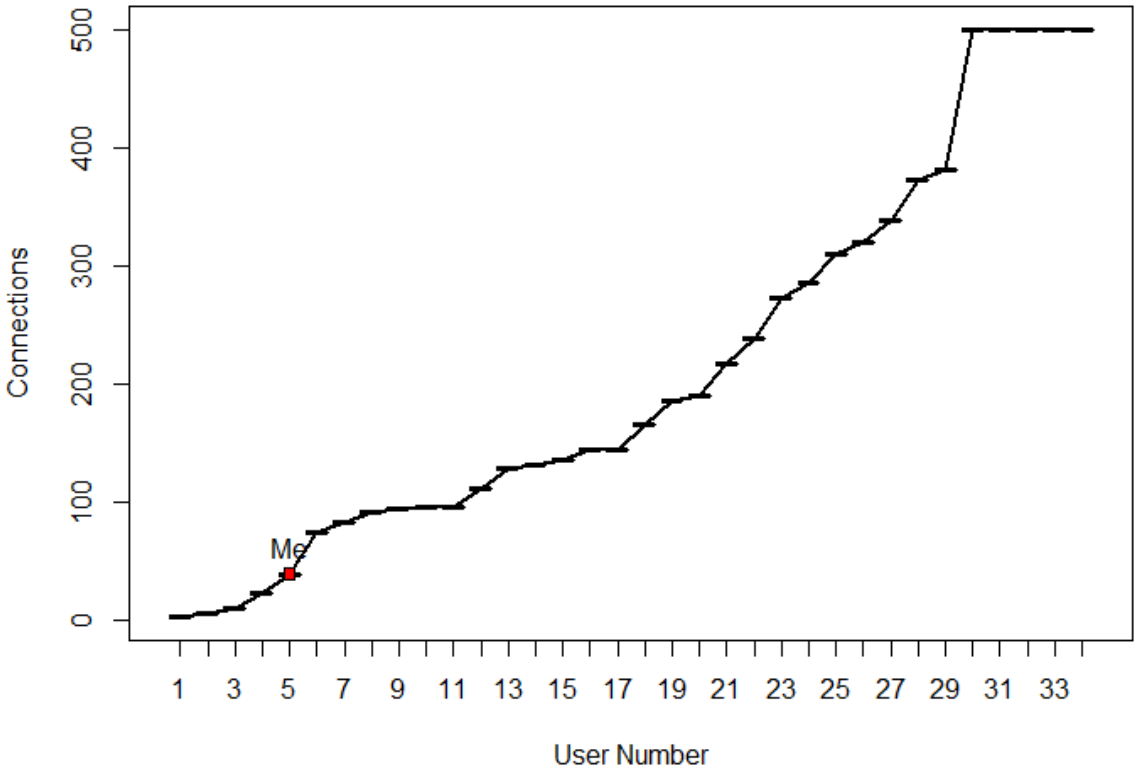

Figure 3.1: Linkedin Number of Friends per Friend

3.2 Python Code

```
from linkedin import linkedin # pip install python-linkedin
import json
import codecs
import calculate

# Define CONSUMER_KEY, CONSUMER_SECRET,
# USER_TOKEN, and USER_SECRET from the credentials
# provided in your LinkedIn application

def writeToFile(ulist):
  with codecs.open('linkedin.txt', 'w', encoding='utf-8') as O:
    O.write('No.\tName\tCount\n')
    for i in range(len(ulist)):
      O.write(str(i+1)+'\t'+ulist[i]['name']+'\t'+str(ulist[i]['count'])+'\n')
#compute and save mean, median, and standard deviation
  with open('calculations.txt', 'a') as O:
    clist = []
    O.write('Linkedin\n')
    for u in ulist:
      clist.append(u['count'])
    mean = calculate.getMean(clist)
    O.write('Mean: ' + str(mean) + '\n')
    median = calculate.getMedian(clist)
    O.write('Median: ' + str(median) + '\n')
    stdev = calculate.standardDev(clist)
    O.write('Standard Deviation: ' + str(stdev) + '\n')

CONSUMER_KEY = '776k21mop4bbh5'
CONSUMER_SECRET = 'QK7bWW9O7eAk4Wte'
```

```
OAUTH_TOKEN = '2780f3f9-132e-4228-8077-9500b0d9933d'
OAUTH_TOKEN_SECRET = '858813cb-f727-46bd-b730-aeddbcf97129'

RETURN_URL = 'http://localhost:8000' # Not required for developer authentication

# Instantiate the developer authentication class
auth = linkedin.LinkedInDeveloperAuthentication(CONSUMER_KEY, CONSUMER_SECRET,
                                                OAUTH_TOKEN, OAUTH_TOKEN_SECRET,
                                                RETURN_URL,
                                                permissions=linkedin.PERMISSIONS.enums.values())

# Pass it in to the app...
app = linkedin.LinkedInApplication(auth)
# Use the app...
#app.get_profile()

my_conn = app.get_profile(selectors=['num-connections'])
num = my_conn['numConnections']

connections = app.get_connections()
people = connections['values']

ulist = []
for i in range(len(people)):
  conn_id = connections['values'][i]['id']
  if conn_id == 'private':
    continue  #user has profile set to private
  conn_num = app.get_profile(member_id=conn_id, selectors=['num-connections'])
  n = conn_num['numConnections']
  conn_name = connections['values'][i]['lastName']
  ulist.append({'name':conn_name, 'count':n})

ulist.append({'name':'me', 'count':num})
sorted_u = sorted(ulist, key=lambda k : k['count'])
writeToFile(sorted_u)
```

Listing 4: Linkedin Connections

```
# https://developer.linkedin.com/documents/getting-oauth-token-python

import oauth2 as oauth
import urlparse

consumer_key            = '776k21mop4bbh5'
consumer_secret         = 'QK7bWW9O7eAk4Wte'
consumer = oauth.Consumer(consumer_key, consumer_secret)
client = oauth.Client(consumer)

request_token_url       = 'https://api.linkedin.com/uas/oauth/requestToken'
resp, content = client.request(request_token_url, "POST")
if resp['status'] != '200':
    raise Exception("Invalid response %s." % resp['status'])

request_token = dict(urlparse.parse_qsl(content))

authorize_url =      'https://api.linkedin.com/uas/oauth/authorize'
print "Go to the following link in your browser:"
print "%s?oauth_token=%s" % (authorize_url, request_token['oauth_token'])
print

accepted = 'n'
while accepted.lower() == 'n':
    accepted = raw_input('Have you authorized me? (y/n) ')
oauth_verifier = raw_input('What is the PIN? ')

access_token_url = 'https://api.linkedin.com/uas/oauth/accessToken'
token = oauth.Token(request_token['oauth_token'], request_token['oauth_token_secret'])
token.set_verifier(oauth_verifier)
client = oauth.Client(consumer, token)

resp, content = client.request(access_token_url, "POST")
access_token = dict(urlparse.parse_qsl(content))
```

```
35
36 print "Access Token:"
37 print "    - oauth_token        = %s" % access_token['oauth_token']
38 print "    - oauth_token_secret = %s" % access_token['oauth_token_secret']
39 print
40 print "You may now access protected resources using the access tokens above."
41 print
```

Listing 5: Linkedin Oauth

REFERENCES

[1] cjc. How can i run firefox on centos with no display? http://serverfault.com/questions/363827/how-can-i-run-firefox-on-centos-with-no-display. Accessed: 2014-10-14.

[2] Csv file reading and writing. https://docs.python.org/2/library/csv.html. Accessed: 2014-10-14.

[3] Get friends list. https://dev.twitter.com/rest/reference/get/friends/list. Accessed: 2014-10-10.

[4] Mining the social web, 2nd edition. http://nbviewer.ipython.org/github/furukama/Mining-the-Social-Web-2nd-Edition/blob/master/ipynb/Chapter%203%20-%20Mining%20LinkedIn.ipynb. Accessed: 2014-10-11.

[5] Getting an oauth token in python. https://developer.linkedin.com/documents/getting-oauth-token-python. Accessed: 2014-10-11.

[6] Profile fields. https://developer.linkedin.com/documents/profile-fields#profile. Accessed: 2014-10-11.

[7] Frank McCown. Producing simple graphs with r. http://www.harding.edu/fmccown/r/#linecharts. Accessed: 2014-10-10.

[8] Phillip. How to write unicode strings into a file? http://stackoverflow.com/questions/5483423/how-to-write-unicode-strings-into-a-file. Accessed: 2014-10-10.

[9] Standard deviation formulas. http://www.mathsisfun.com/data/standard-deviation-formulas.html. Accessed: 2014-10-15.