Introduction to Web Science: Assignment #5

Dr. Nelson

Alexander Nwala

Thursday, October 16, 2014

Contents

Problem 1 3

Problem 2 5

Problem 3 8

Problem 1

The "friendship paradox" (http://en.wikipedia.org/wiki/Friendship_paradox) says that your friends have more friends than you do.

Explore the friendship paradox for your Twitter account. Since Twitter has directional links (i.e., "followers" and "following"), we'll be investigating if the people you follow (Twitter calls these people "friends") follow more people than you. If you are following <50 people, use my twitter account "phonedude_mln" instead of your own.

Create a graph of the number of friends (y-axis) and the friends sorted by number of friends (x-axis). (The friends don't need to be labeled on the x-axis as "Bob", "Mary", etc. – just 1, 2, 3 ...) In other words, if you have 100 friends your x-axis will be 1..101 (100 + you), and the y-axis value will be number of friends that each of those friends has. The friend with the lowest number of friends will be first and the friend with the highest number of friends will be last.

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.

The appropriate part of the Twitter API to use is:
https://dev.twitter.com/rest/reference/get/friends/list

SOLUTION 1

The solution for this problem is outlined by the following steps:

1. **Establish connection between application and Twitter:** This was achieved through Tweepy [4] which provides a python wrapper for the Twitter API.
2. **Authenticate application:** This was achieved by registering the application and generating OAuth [2] access tokens.
3. **Fetch all friends:** With the use of Dr. Nelson's Twitter account (phonedude_mln) and the Twitter user.friends() method, all friends were retrieved.
4. **Fetch the count of friends of friends:** Finally, given the friends derived in the previous step, the user.friends_count() method was applied to get the count of friends of friends as outlined in Listing 1.

```
The file FoFCountTwitter.csv contains the
complete list of friends and their total friends count
```

Listing 1: Twitter: Get The Count Of Friends Of Friends

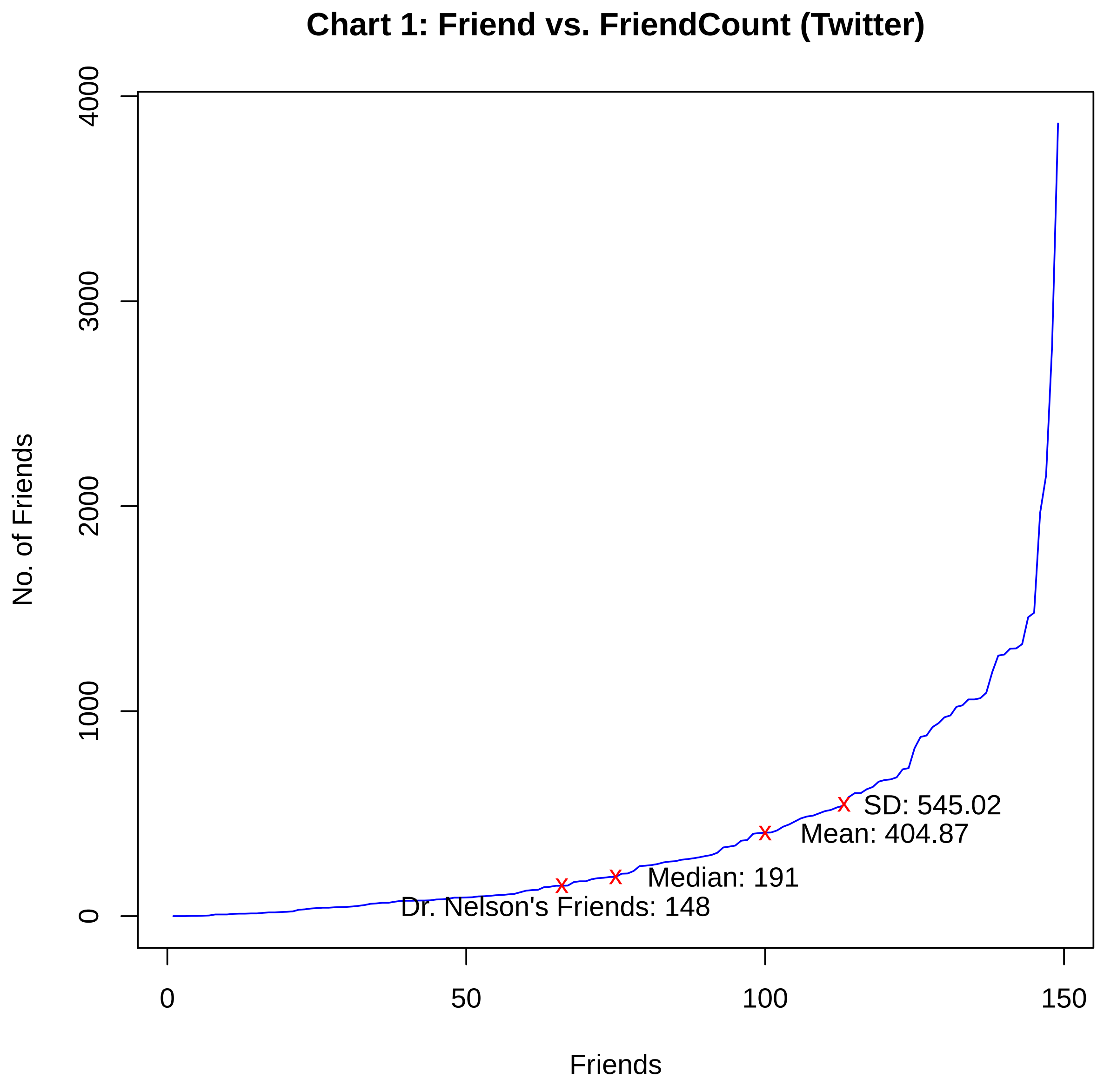
```
...
#Get the count of friends of friends
user = api.get_user(screenName)
friendsCount = str(user._json['friends_count'])
print screenName + ' has ' + friendsCount + ' friends, and '

friendCountFile.write('"USER", "FRIENDCOUNT"\n')
```

Problem 1 continued on next page. . .

```
friendCountFile.write(user.name.encode('utf-8') + ', ' + friendsCount + '\n')

for friend in user.friends(count=int(friendsCount)):
     friendsCount = str(api.get_user(friend.screen_name).friends_count)
     print friend.name + ' has ' + friendsCount + ' friends'
     friendCountFile.write(friend.name.encode('utf-8') + ', ' + friendsCount + '\n
        ')
...
```

CONCLUSION 1

Based on Chart 1, it is clear that the friendship paradox holds; Dr. Nelson has less friends than his friends.



Problem 2

Using your facebook account, repeat question #1 (if you have >50 friends).
Start at: https://developers.facebook.com/docs/graph-api/reference/v2.1/user/friends
or perhaps:
http://socialnetimporter.codeplex.com/

SOLUTION 2

The Facebook API restricts seeing the friends of friends; only the friends of people who use the Facebook API and have granted permission to the third-party application can be retrieved. Consequently, I used the Selenium [3] web browser automation/testing tool to retrieve the friends of my friends (the count) for those who share this information.
The solution for this problem is outlined by the following steps:

1. **Login into Facebook:** Using my Facebook account, I was able to login into Facebook with selenium by supplying my credentials to the HTML username and password elements as outlined in Listing 2.

Listing 2: Facebook Login With Selenium

```
...
#Login into Facebook
myFirefoxBrowser = webdriver.Firefox()
myFirefoxBrowser.implicitly_wait(3)
# or you can use Chrome(executable_path="/usr/bin/chromedriver")
myFirefoxBrowser.get("http://www.facebook.org")
assert "Facebook" in myFirefoxBrowser.title

elem = myFirefoxBrowser.find_element_by_id("email")
elem.send_keys(userFaceBookEmail)
elem = myFirefoxBrowser.find_element_by_id("pass")
elem.send_keys(userFaceBookPassword)
elem.send_keys(Keys.RETURN)
...
```

2. **Open the friends page and scroll to the bottom of page:** Following authentication, the application subsequently opened my friends page (https://www.facebook.com/friends/). But since only a subset of my friends are viewable, the application scrolls to the end of the page in order to load all my friends data as outlined in Listing 3.

Listing 3: Scroll To the Bottom of my Friends Page

```
...
#open friends page
friendsLink = 'https://www.facebook.com/friends/'
myFirefoxBrowser.get(friendsLink)
myFirefoxBrowser.maximize_window()

#scroll to bottom of page
previousCountOfFriends = -1
while True:

```

Problem 2 continued on next page. . .

```
        myFirefoxBrowser.execute_script("return window.scrollTo(0, document.body.
            scrollHeight);")
        html = myFirefoxBrowser.page_source.encode('utf-8')

        soup = BeautifulSoup(html)
15      parentOfUIProfileBlockContent = soup.findAll('div', { 'class' : '
            uiProfileBlockContent' })

        #lastIndexOfFriends = html.rfind('<div class="uiProfileBlockContent">')
        lastIndexOfFriends = len(parentOfUIProfileBlockContent)

20      #'Friends' not found
        if( lastIndexOfFriends == -1 ):
            break

        #No new entry
25      if( previousCountOfFriends == lastIndexOfFriends ):
            htmlOutputFile.write(html)
            break
        else:
            previousCountOfFriends = lastIndexOfFriends
30
        sleepTime = randint(3,7)
        print "...sleeping for", sleepTime, "seconds"
        time.sleep(sleepTime)
...
```

3. **Download HTML content:** Having loaded all friends data, the application saves the html data into a file as outlined in Listing 3, Line 26.

4. **Process HTML content to retrieve the count of the friends of my friends:** With the use of BeautifulSoup [1], data of form

   `<friend, friendCount>`

   was written into the file **FoFCountFacebook.csv** as outlined by Listing 4.

Listing 4: Facebook: Get The Count Of Friends Of Friends

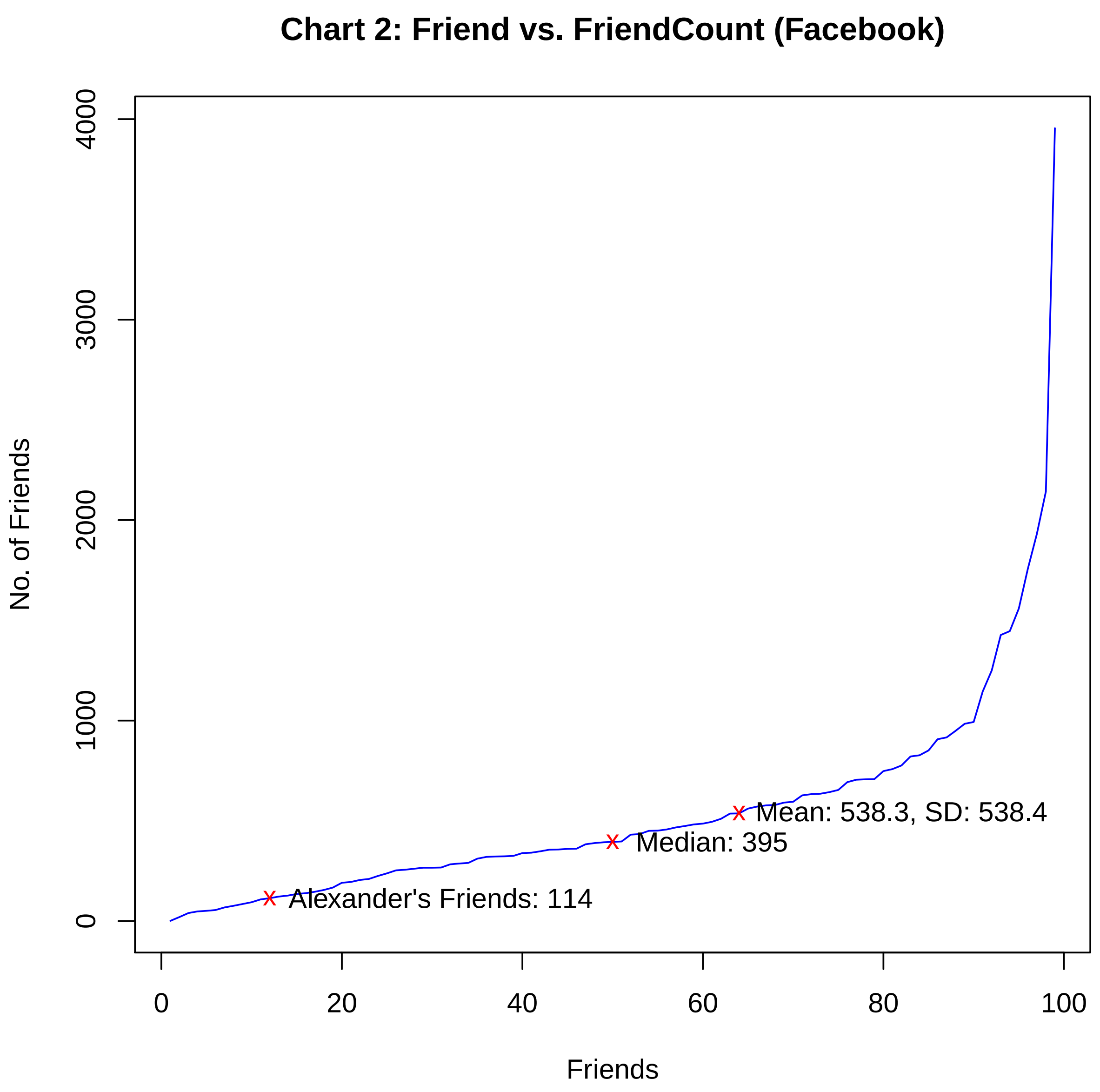
```
#writes tuples <friend, friendCount> into globalCSVOutputFile
def getFriendOfFriendsFromHtml(htmlText):
goAheadFlag = False
if( len(htmlText) > 0 ):
    try:
        outputFile = codecs.open(globalCSVOutputFile, 'w', 'utf-8')
        outputFile.write('"USER", "FRIENDCOUNT"\n')
    except:
        exc_type, exc_obj, exc_tb = sys.exc_info()
        fname = os.path.split(exc_tb.tb_frame.f_code.co_filename)[1]
        print fname, exc_tb.tb_lineno, sys.exc_info()
        return


    soup = BeautifulSoup(htmlText)
```

Problem 2 continued on next page. . .

```
    parentOfUIProfileBlockContent = soup.findAll('div', { 'class' : '
        uiProfileBlockContent' })

    for profile in parentOfUIProfileBlockContent:

            friendName = profile.find('div', { 'class' : 'fsl fwb fcb' })
            potentialFriendsCount = profile.find('a', { 'class' : 'uiLinkSubtle
                ' })

            #potentialFriendsCount: x (f)riends | x mutual friends, etc, so
                split
            if( potentialFriendsCount is not None ):

                potentialFriendsCount = potentialFriendsCount.text.split(' ')

                if( len(potentialFriendsCount) > 1 ):
                    if( len(potentialFriendsCount[1]) > 0):
                        if( potentialFriendsCount[1][0].lower() == 'f' ):

                            friendCount = potentialFriendsCount[0].replace(
                                ',','')

                            stringToWrite = friendName.text + ', ' +
                                friendCount + '\n'
                            outputFile.write(stringToWrite)
                            goAheadFlag = True


    outputFile.close()
return goAheadFlag
```

CONCLUSION 2

Based on Chart 2, it is clear that the friendship paradox holds; I have less friends than my friends.



Problem 3

Using your linkedin account, repeat question #1 (if you have <50 connections).
Start at: https://developer.linkedin.com/apis

SOLUTION 3

Similar to Facebook, the LinkedIn API restricts seeing the 2nd degree connections (friends of friends). Consequently, I used the Selenium web browser automation/testing tool to retrieve the 2nd degree connections (the count).
The solution for this problem is outlined by the following steps:

Problem 3 continued on next page...

1. **Login into LinkedIn:** Using Jose Antonio Olvera Caizares' account, I was able to login into LinkedIn with selenium by supplying my credentials to the HTML username and password elements as outlined in Listing 5.

Listing 5: LinkedIn Login With Selenium

```
...
#Login into LinkedIn
userLinkedInEmail = userLinkedInEmail.strip()
userLinkedInPassword = userLinkedInPassword.strip()

myFirefoxBrowser = webdriver.Firefox()
myFirefoxBrowser.implicitly_wait(3)
# or you can use Chrome(executable_path="/usr/bin/chromedriver")
myFirefoxBrowser.get("http://www.linkedin.com/")
assert 'LinkedIn' in myFirefoxBrowser.title

elem = myFirefoxBrowser.find_element_by_id('session_key-login')
elem.send_keys(userLinkedInEmail)
elem = myFirefoxBrowser.find_element_by_id('session_password-login')
elem.send_keys(userLinkedInPassword)
elem.send_keys(Keys.RETURN)

all_cookies = myFirefoxBrowser.get_cookies()
pleaseSleep()
...
```

2. **Open/download/process the Connections page:** Following authentication, the application subsequently opened Jose's Connections page (https://www.linkedin.com/people/connections?trk=nav_responsive_tab_network). This page was downloaded and serves as the initial input to retrieve 2nd degree connections as outlined in Listing 6. From this page, with the use of BeautifulSoup, the count of the connections of Jose's connection (2nd degree connection) were extracted as outlined in Listing 6.

Listing 6: LinkedIn: Get The Count Of Friends Of Friends

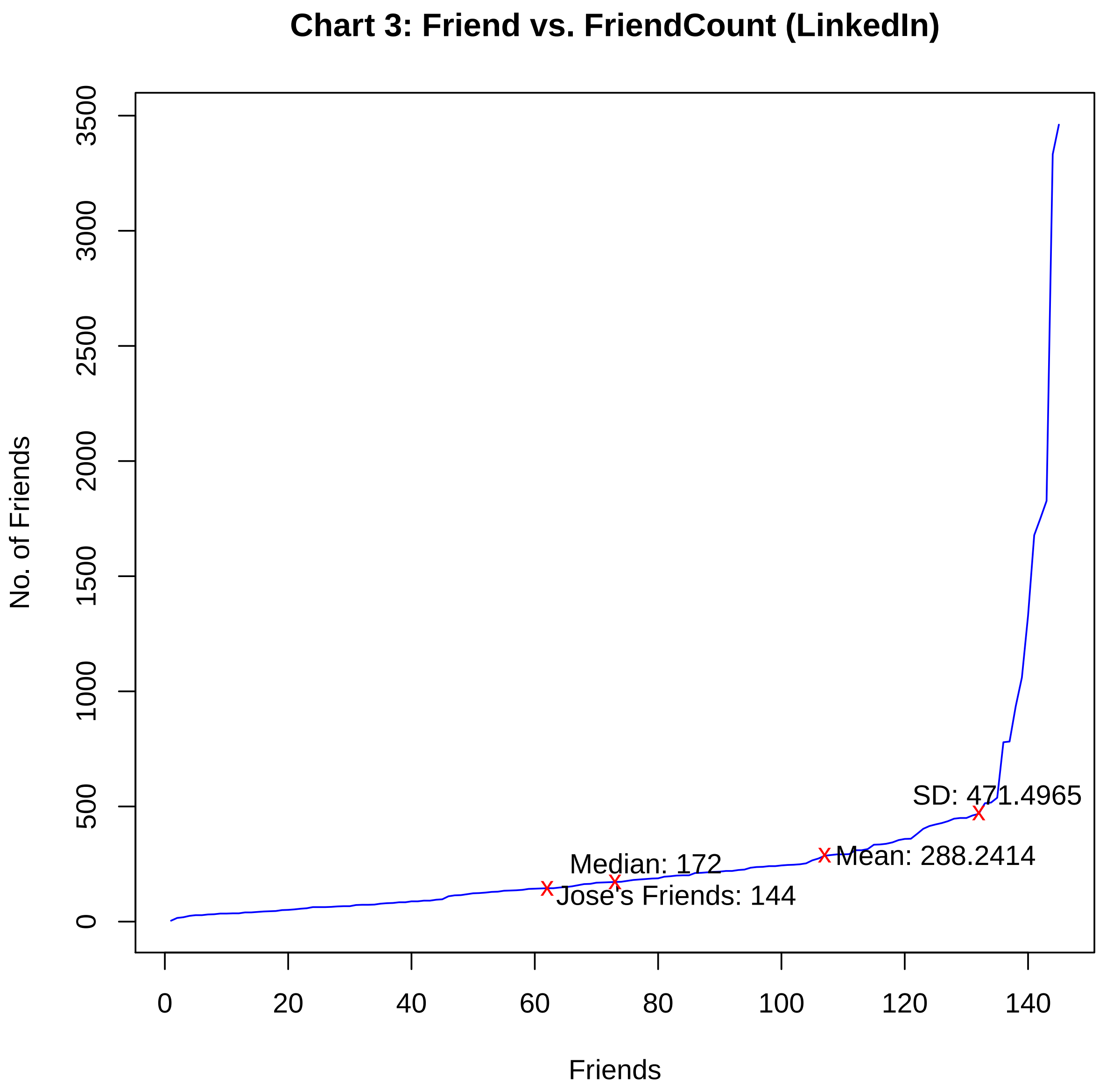
```
#write <connection, connectionCount> tuples into globalCSVOutputFile
#returns connection element ids of connections with 500+ connections to be
    dereferenced
def get2ndDegreeConnectionsFromHTML(connectionsHtml):
...
indexOfConnection = 0
for connection in allConnections:

     person = connection.find('input', { 'type' : 'checkbox' })
     #person is of form: <input type="checkbox" value="Marc Abramo Serr"/>

     if( person is not None ):
          person = str(person)
          connectionName = person.split('"')

          #get connection count, name is 3rd position
          if( len(connectionName) > 2 ):
               connectionCount = connection.find('div', { 'class' : 'conn-count'
                   })
               connectionCount = connectionCount.text

               #get id
               idOfElement = str(connection)
               indexIdOfElement = idOfElement.find('id="')

               if( indexIdOfElement > -1 ):
                    indexIdOfElement = indexIdOfElement + 4
                    indexIdOfElementClosingQuotes =    idOfElement.find('"',
                        indexIdOfElement)
                    connectionHTMLElementID = idOfElement[indexIdOfElement :
                        indexIdOfElementClosingQuotes]

                    #print connectionName[3], connectionCount,
                        connectionHTMLElementID + '\n'
                    connectionCount = connectionCount.strip()
                    if( connectionCount == '500+' ):
                         dictionaryOfConnectionsToDereference[connectionName[3]] =
                              connectionHTMLElementID
                    else:
                         connectionsOutputFile.write( connectionName[3].decode('
                             utf-8') + ', ' + connectionCount +'\n')
...
```

3. **Dereference connections with 500+ connections:** connections who have more than 500 connections have there connections count expressed as 500+. Consequently, Listing 7 dereferences this 500+ link to get the exact number, for connections with over 500 connections. This was achieved by retrieving the HTML IDs of connections which fulfilled this criteria. Subsequently, with Selenium, a search query was submitted (Listing 7, Line 16), in order to retrieve the page which contains the exact number of connections. After dereferencing these, the result was appended to **FoFCountLinkedIn.csv**

Listing 7: LinkedIn: Get The Count Of Friends Of Friends For Friends With 500+ Friends

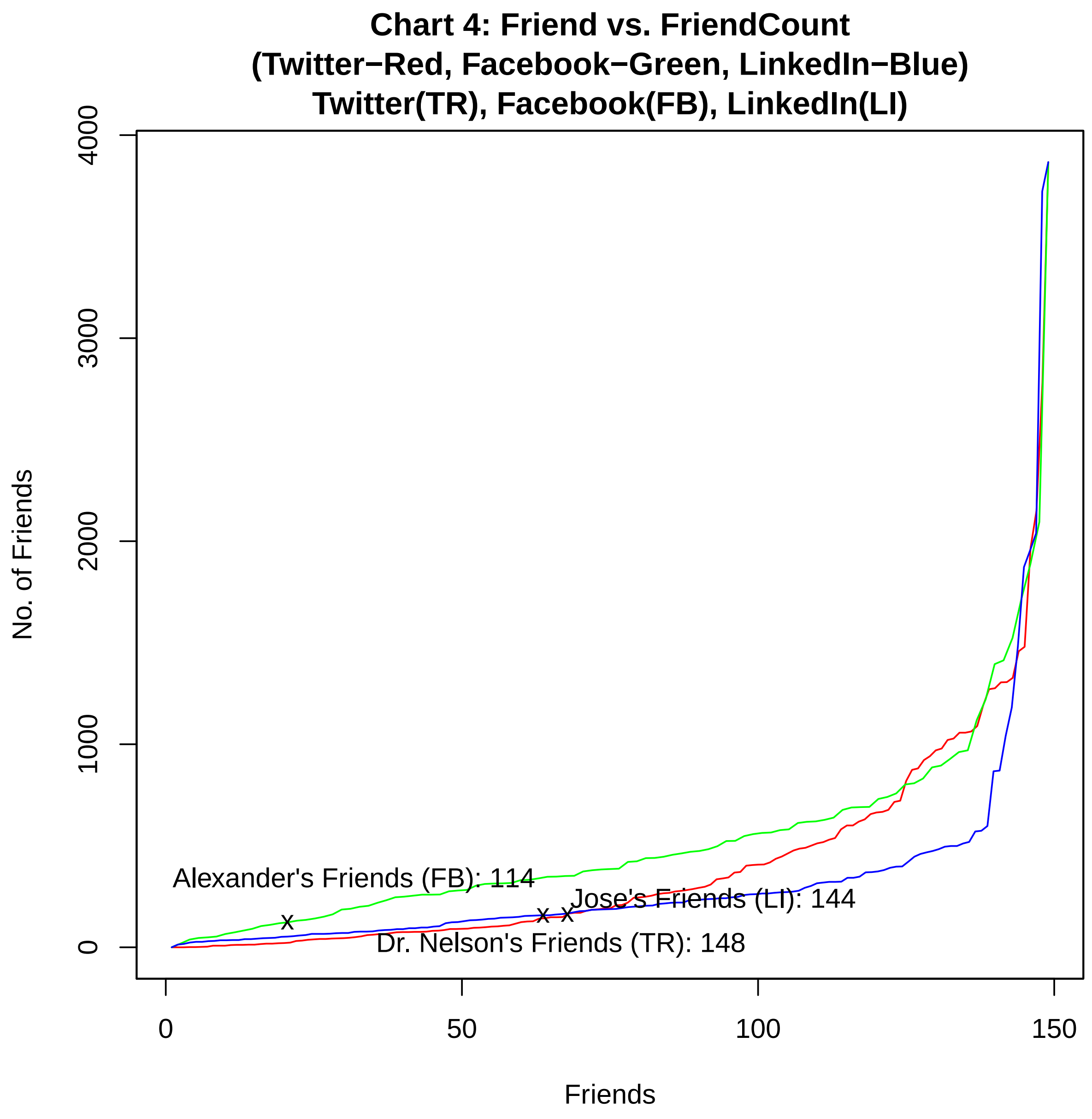
```
   #Deference the count of connections for connections with 500+ connections
   dictionaryOfConnectionsToDereference = get2ndDegreeConnectionsFromHTML(html)
   ...
   #call function to get <friend, friendCount> and get ids of 500+ connection count
       to be dereferenced
 5 if( len(dictionaryOfConnectionsToDereference) > 0 ):
        try:
             connectionsOutputFile = codecs.open(globalCSVOutputFile, 'a', 'utf-8')
        except:
             exc_type, exc_obj, exc_tb = sys.exc_info()
10           fname = os.path.split(exc_tb.tb_frame.f_code.co_filename)[1]
             print fname, exc_tb.tb_lineno, sys.exc_info()
             return
        count = len(dictionaryOfConnectionsToDereference)
        for connection, conElementID in dictionaryOfConnectionsToDereference.items():
15           pleaseSleep()
             linkToGet = 'http://www.linkedin.com/profile/connections?id=' +
                 conElementID
             myFirefoxBrowser.get(linkToGet)
```

```
            html = myFirefoxBrowser.page_source.encode('utf-8')
            connectionCount = getConnectionCountFor500PlusConnection(html)

            #append globalCSVOutputFile
            print "deref:", connection, connectionCount, count
            connectionsOutputFile.write( connection.decode('utf-8') + ', ' +
                connectionCount +'\n')
            count = count - 1
...
```



CONCLUSION 3

Based on Chart 3, it is clear that the friendship paradox holds; Jose has less friends than his friends.



References

[1] Beautifulsoup. http://www.crummy.com/software/BeautifulSoup/bs4/doc/. Accessed: 2014-10-15.

[2] Oath. https://dev.twitter.com/oauth. Accessed: 2014-10-14.

[3] Selenium. http://www.seleniumhq.org/. Accessed: 2014-10-15.

[4] Twitter api. http://tweepy.readthedocs.org/en/v2.3.0/. Accessed: 2014-10-14.