CS 595: Assignment #5

Due on Thursday, October 16, 2014

Dr. Nelson 4:20pm

Holly Harkins

Contents

Problem 1 **3**

Problem 2 **7**

Problem 3 **11**

Problem 1

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

1. 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 labelled 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.

Listing 1: Python Part1

```
# -*- encoding: utf-8 -*-

from __future__ import unicode_literals
import requests
from requests_oauthlib import OAuth1
from urlparse import parse_qs

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 = "QX9UsnHmngd6vD20LgEBXtvoN"
CONSUMER_SECRET = "Kkvt4i7x2pglyl8qhSlzdoHc2vjexNlrW8xGNZZeaFY2QjavQx"

OAUTH_TOKEN = "2825370151-dNfwsYgzC12FUyZjM4MhoXu4D7hMmG1RguUd3q0"
OAUTH_TOKEN_SECRET = "pZV2GtrPOPG9v3V0ugTWYqgm0KpyKrGlFQTj5djQu5I8C"

##Save Result to File
file1 = open('twitterFriends.txt','w')

def setup_oauth():
    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]
```

Problem 1 continued on next page...

```
    resource_owner_secret = credentials.get('oauth_token_secret')[0]

    ##Authorize
    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)

    ##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:
        oauth = get_oauth()
        r=requests.get(url="https://api.twitter.com/1.1/friends/list.json?cursor=-1&
            screen_name=phonedude_mln&skip_status=true&include_user_entities=false&
            count=200", auth=oauth)
        p=r.json()["users"]
        file1.write("Screen_Name, Number_Friends")
        file1.write("\n")

        for f in p:
            name=f["screen_name"]
            count=f["friends_count"]
            file1.write(name + "," + str(count) + "\n")

file1.close()
```

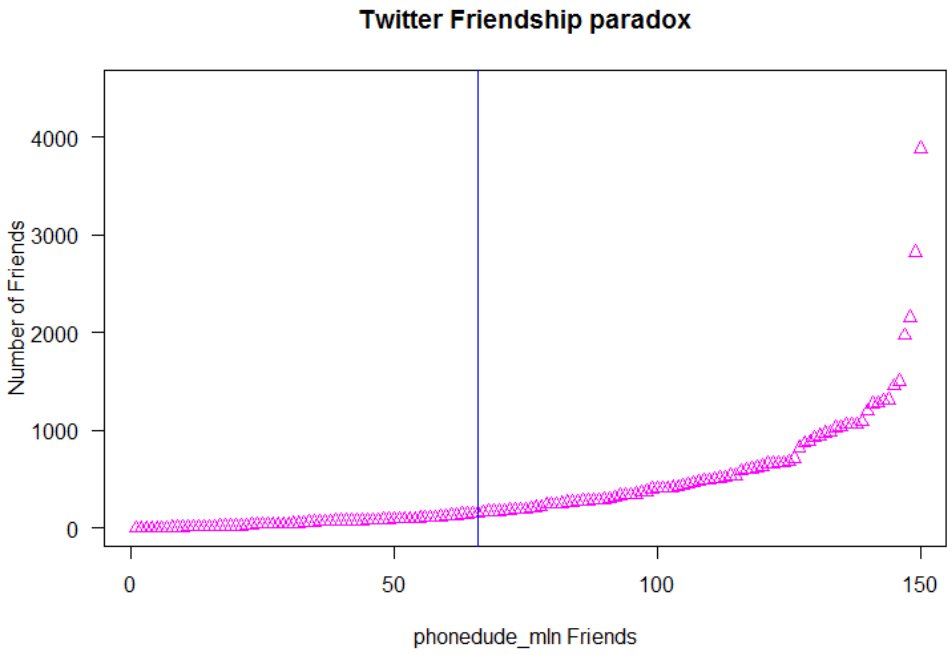

Figure 1: Twitter Friendship Paradox

Computed Mean, Median, Std Dev

Mean	397.8145695
Median	191
Standard Deviation	539.0136474

Figure 2: Computed Mean, Median, and Std Dev

I choose to use Twitter account phonedude_mln to explore the friendship paradox. I used part1.py to retrieve the list of friends.
I used a spread sheet to calculate mean, median, and std deviation.
I assigned a unique ID to each member and made a scatter plot using R.

As shown by the graph, the "friendship paradox" that says your friends have more friends than you, appears true for this twitter account.

Problem 2

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

Start at:
https://developers.facebook.com/docs/graph-api/reference/v2.1/user/friends

or perhaps:

http://socialnetimporter.codeplex.com/

Listing 2: Facebook Scrape

```
#Mechanize and Beautiful soup can't inteface with the javascript used for the infinite
    scroll: I was able to login, see friends but not able
#to scroll to see more friends

#pip install -U selenium
#sources:
#https://gist.github.com/leostera/3535568
#https://pypi.python.org/pypi/selenium
#cookies problem: http://stackoverflow.com/questions/7854077/using-a-session-cookie-
    from-selenium-in-urllib2

from selenium import webdriver
from selenium.webdriver.common.keys import Keys
import time
from selenium.webdriver.common.by import By
import os, sys

#dirty code, consider user behaviour simulation, not found exceptions, etc
#output file: allFacebookFriends.html
def getHtmlOfAllFriends(userFaceBookEmail, userFaceBookPassword, LastFriendName):

    if( len(userFaceBookEmail) > 0 and len(userFaceBookPassword) > 0 and len(
        LastFriendName) > 0 ):
        pass
    else:
        print "one input length is bad"
        return

    try:
        htmlOutputFile = open('allFacebookFriends.html', 'w')
    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

    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)

    #http://stackoverflow.com/questions/7854077/using-a-session-cookie-from-selenium-in-urllib2
    all_cookies = myFirefoxBrowser.get_cookies()
    #cookies = {}
    #for s_cookie in all_cookies:
    #    cookies[s_cookie["name"]]=s_cookie["value"]


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


    #scroll to bottom of page
    for i in range(0, 20):
        myFirefoxBrowser.execute_script("window.scrollTo(0, document.body.scrollHeight);")
        html = myFirefoxBrowser.page_source.encode('utf-8')


        if( html.find(LastFriendName) > -1 ):
            htmlOutputFile.write(html)
            print "found"
            break

        time.sleep(5)


    myFirefoxBrowser.close()


usr = ''
pwd = ''
lastFriendName = ''
getHtmlOfAllFriends(usr, pwd, lastFriendName)
```

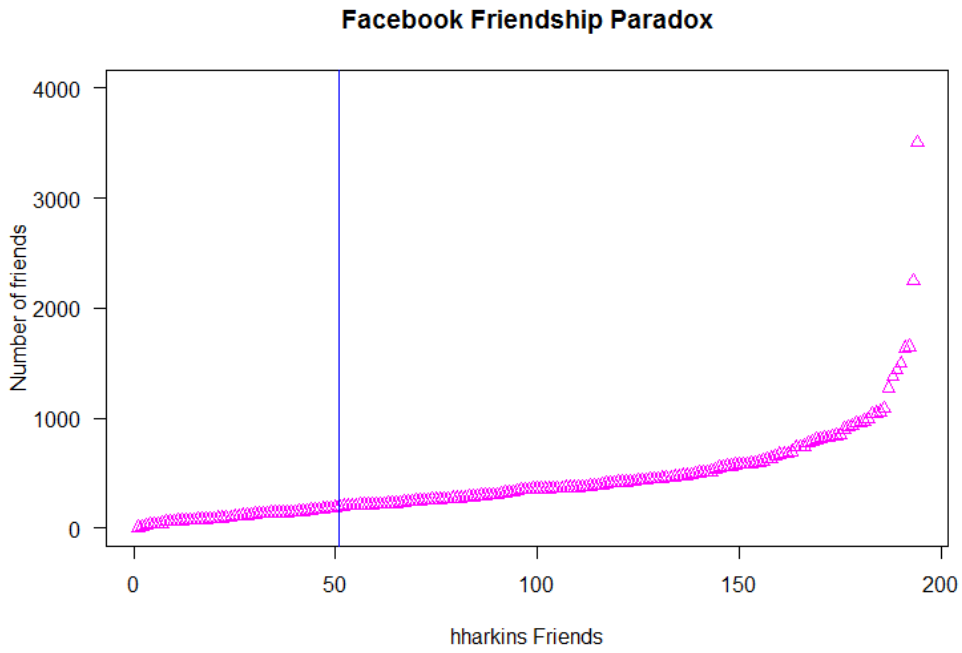

Figure 3: Facebook Friendship Paradox

Computed Mean, Median, Std Dev

Mean	431.5128205
Median	353
Standard Deviation	407.4813937

Figure 4: Computed Mean, Median, and Std Dev

Problem 2 continued on next page...

I choose to use my facebook account to explore the friendship
paradox. I used seleniumScrapeFB.py by Alex Nwala to retrieve my friends list.
I used a spread sheet to calculate mean, median, and std deviation.
I assigned a unique ID to each member and made a scatter plot using R.

As shown by the graph, the "friendship paradox" that says your
friends have more friends than you, appears true in my case.

Problem 3

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

Start at: https://developer.linkedin.com/apis
```

References

[1] http://thomassileo.com/blog/2013/01/25/using-twitter-rest-api-v1-dot-1-with-python/

[2] http://stackoverflow.com/questions/3453695/adding-text-to-a-plot/

[3] http://www.craigaddyman.com/mining-all-tweets-with-python/

[4] author = Alex Nwala, title = Facebook Scrape, year = 2014, https://github.com/anwala/cs595-f14/tree/master/scrapeFacebook