Old Dominion Univeristy

CS 495: Introduction to Web Science
Instructor: Michael L. Nelson, Ph.D
Fall 2014 4:20pm - 7:10pm R, ECSB 2120

Assignment # 5

GeorgeC. Micros UIN: 00757376

Honor Pledge
I pledge to support the Honor System of Old Dominion University. I will refrain from any form of academic dishonesty or deception, such as cheating or plagIiarism. I am aware that as a member of the academic community it is my responsibility to turn in all suspected violations of the Honor Code. I will report to a hearing if summoned.

Signed ______________________________
October 16, 2014

George C. Micros

Written Assignment 5

Fall 2014
CS 495: Introduction to Web Science
Dr. Michael Nelson

October 16, 2014

Contents

Chapter 1
Written Assignment 5

1.1 Question 1

1.1.1 The Question

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`

1.1.2 The Answer

```
#! /usr/bin/python

# -*- encoding: utf-8 -*-
from __future__ import unicode_literals
import requests
from requests_oauthlib import OAuth1
from urlparse import parse_qs
from pprint import pprint
import urllib2
import httplib2
import sys


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 = "fZJV8Ab0SPvE3RbELyok0vjfa"
CONSUMER_SECRET = "HmjPCwt5ysI51pYtCGbmQKJU5IqUtIqI8sL2fGpvKhMIYFHaq6"
OAUTH_TOKEN = "2822206502-dN9QiytM0BKSRrirhmzGYHLcGypaGMoa9X3vZvv"
OAUTH_TOKEN_SECRET = "cR0B9TgqWaKG00h0eGsG8lEFi1BtQvKczlTGXBEggqAa0"

def setup_oauth():
        """Authorize your app via identifier."""
        # Request token
        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]
        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)
        # Finally, Obtain the 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
# returns id num and list of links
def getURL(r):
    links = [];
    # traverse json object of multiple tweets
    for rs in r.json():
        temp = rs.get('entities').get('urls');
        num =  rs.get('id');
        # check if there were urls in tweet
        if len(temp) != 0:
            num =  rs.get('id');
            temp = str((temp[0])['url']);
            # attempt to get the
            try:
                h = httplib2.Http(".cache_httplib")
                h.follow_all_redirects = True
                h.force_exception_to_status_code = True
                resp = h.request(temp, "GET")[0]
                if resp['status'] == '200':
                    # this is the final redirected url
                    print resp['content-location']
                    links.append(resp['content-location'])
            except:
                pass
    return num, links

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()

                # initial variables
                numURLs = 1000;
                site = "https://api.twitter.com/1.1/friends/list.json?cursor="
                curs = "-1"
                usr = "&screen_name=phonedude_mln"
                othr = "&skip_status=true&include_user_entities=false&count="
                cnt = "200"

                print "name, count"
                r = requests.get("https://api.twitter.com/1.1/users/show.json?screen_name=
                    phonedude_mln", auth=oauth);
                name = ((r.json())['name']).encode('ascii', 'ignore');
                friends = (r.json())['friends_count'];
                print "\""+str(name)+"\" , "+str(friends)
                # initial request with count
                r = requests.get(url=site+curs+usr+othr+cnt, auth=oauth)
                while True:
                    for user in (r.json())['users']:
                        name =  (user['name']).encode('ascii', 'ignore')
                        count = user['friends_count']
                        print "\""+str(name)+"\" , " +str(count)
                        curs =  str((r.json())['next_cursor'])
                        if (curs == '0'):
                            break
                        else:
                            r = requests.get(url=site+curs+usr+othr+cnt, auth=oauth)
```

Listing 1: Python script that extracts twitter “friends” and the number of their “friends”

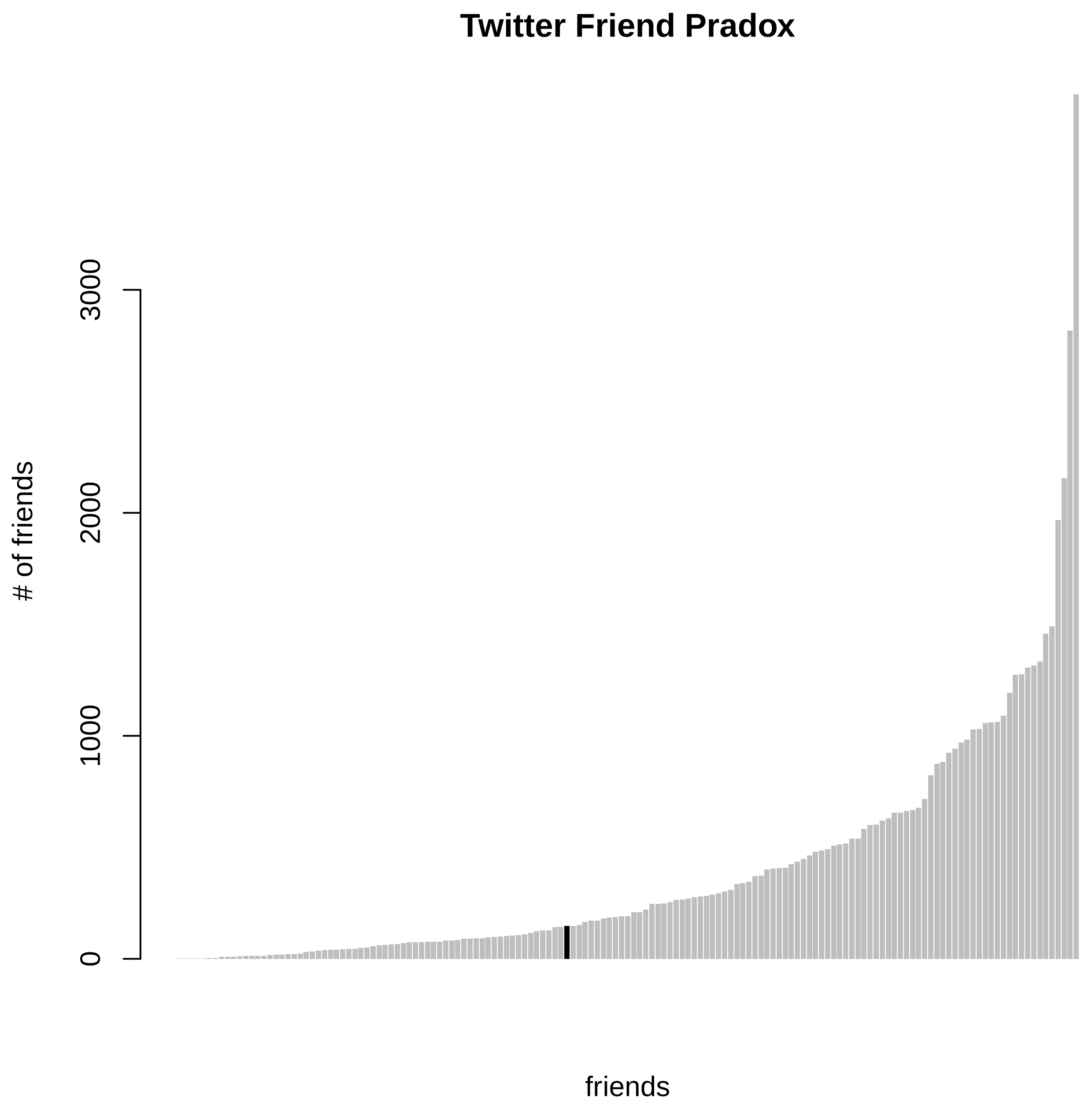


Fig. 1.1: mean = 405.51, std = 546.97

```
#! /usr/bin/Rscript

d <- read.table("temp", header=TRUE, sep=",", as.is=TRUE, strip.white=TRUE)

mean(d[,2])
sd(d[,2])


names <-d[,1]
num <- sort(d[,2]);
names <- names[order(d[,2])];

```

```
cols <- c("grey", "black")[(names=="Michael L. Nelson")*1+1];

pdf("twitter.pdf")
barplot(num, col=cols, main="Twitter Friend Pradox", xlab="friends", ylab="# of friends", border=
    NA);
dev.off()
```

Listing 2: R script to do the maths and plot

1.2 Question 2

1.2.1 The Question

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/`

1.2.2 The Answer

```
#! /bin/bash

fbcmd FQL "SELECT uid, name, friend_count FROM user WHERE uid = me() OR uid IN (SELECT uid2 FROM
    friend WHERE uid1 = me()) AND friend_count!=0"
```

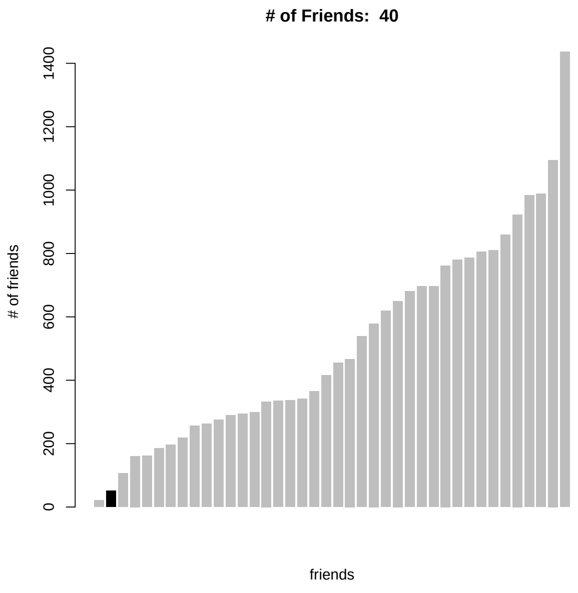
Listing 3: Bash script that fetchs the friends list and counts

```
#! /bin/bash

grep -E '^ *name*' $1 | cut -d ' ' -f18- > temp
grep -E '^ *friend_count*' $1 | cut -d ' ' -f10-  > cnts

rm names

while read  line
do
    echo "\"$line\" , " >> names
done < temp

echo "names,  count"
paste names cnts


rm temp cnts names
```

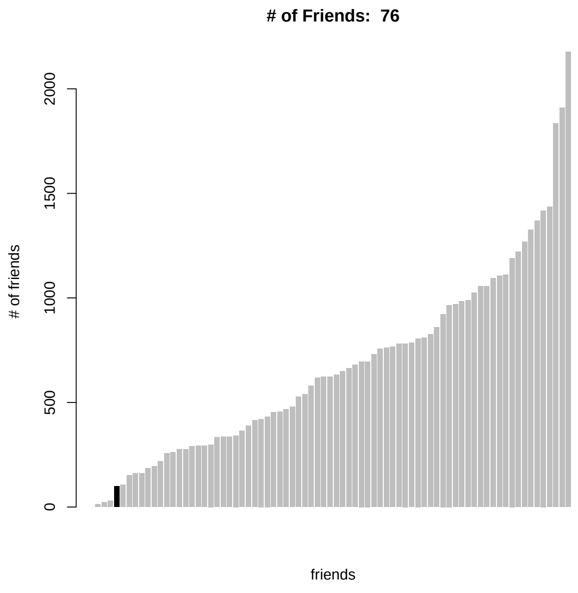
Listing 4: Bash script to parse friend list into CSV file

```
#! /usr/bin/Rscript

args <- commandArgs(trailingOnly=TRUE)
input <- args[1]
fn <- unlist(strsplit(input, "/"))
name <- paste(fn[3], ".pdf", sep="")
name <- paste("./figs/", name, sep="")

print(name)

d <- read.table(input, header=TRUE, sep=",", as.is=TRUE, strip.white=TRUE)

names <-d[,1]
num <- sort(d[,2]);
names <- names[order(d[,2])];

mean(num)
sd(num)

cols <- c("grey", "black")[(names=="George C Micros")*1+1];

nF = paste("# of Friends: ", length(num));

pdf(name)
barplot(num, col=cols, main=nF, xlab="friends", ylab="# of friends", border=NA);
dev.off()
```

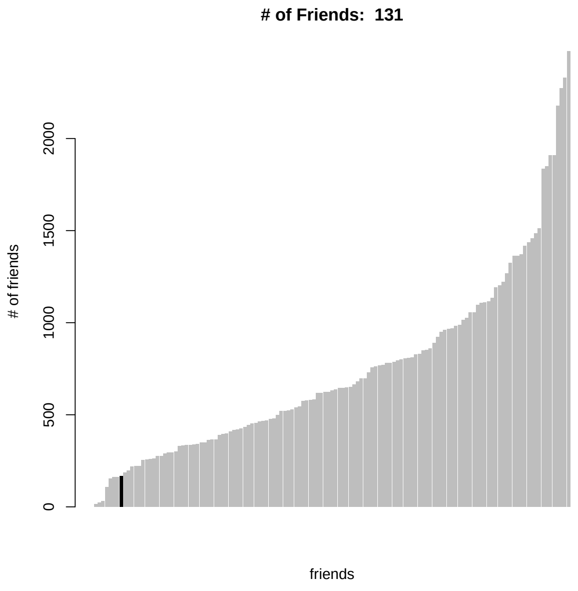
Listing 5: R script to do maths and make plots



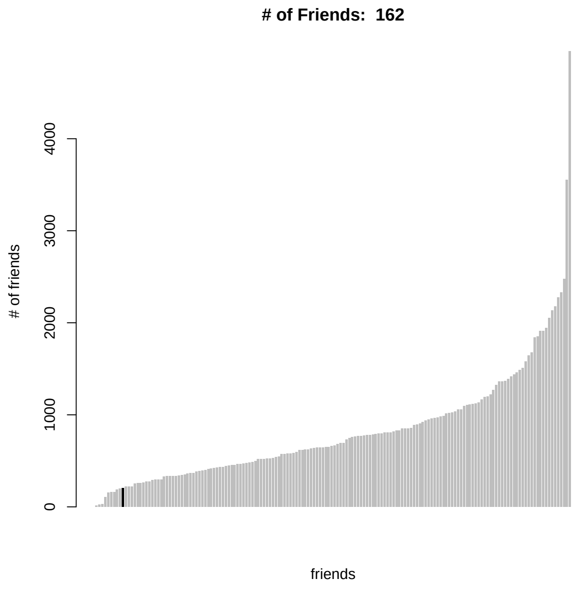
(a) mean =513.05, std = 323.84



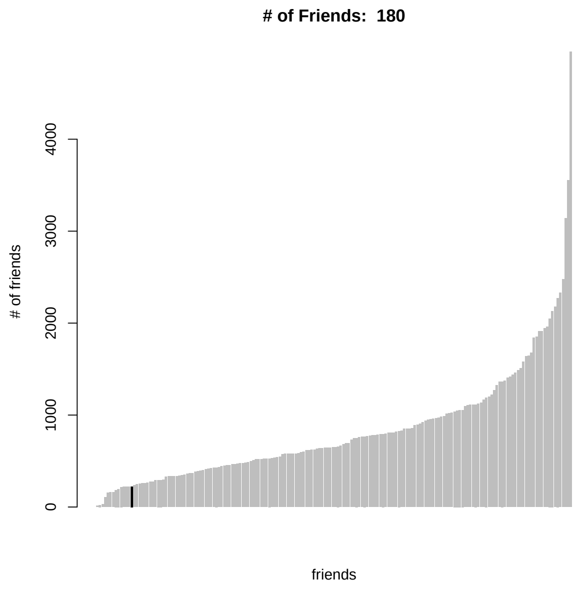
(b) mean = 677.01, std = 452.36



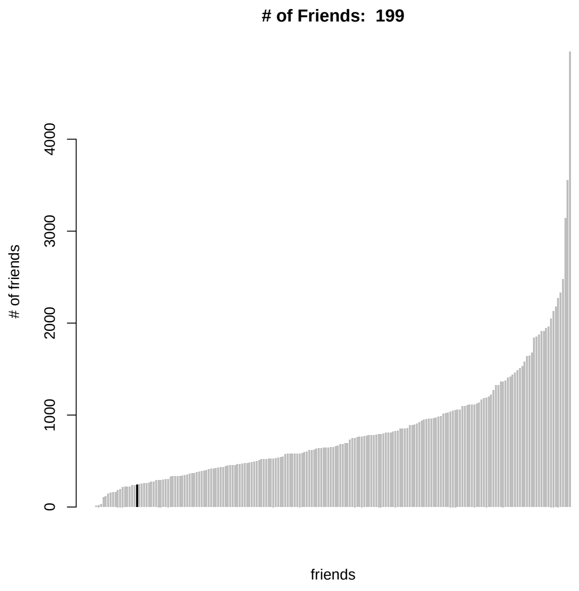
(a) mean = 737.23, std = 498.40



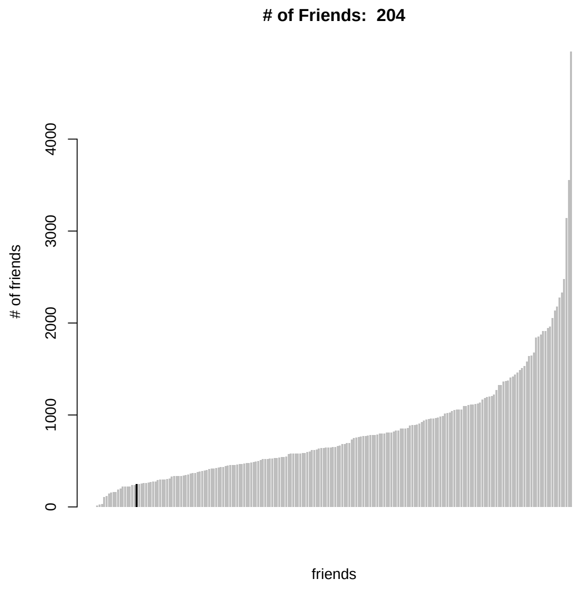
(b) mean = 826.42, std = 639.16



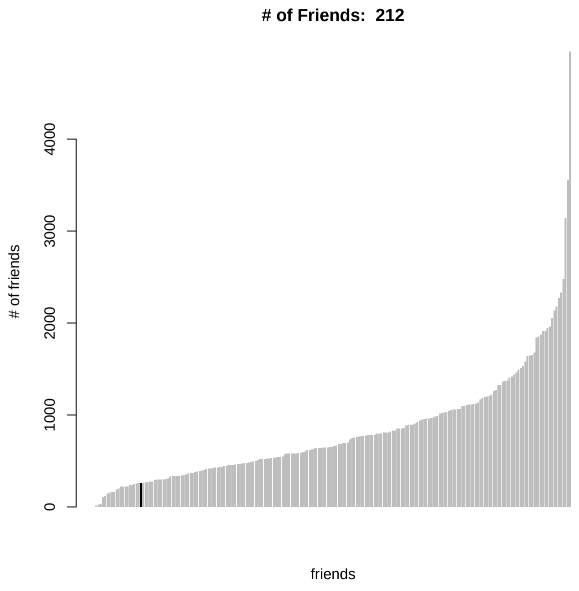
(a) mean = 831.67, std = 646.77



(b) mean = 823.67, std = 633.63



(a) mean = 819.97, std = 628.37



(b) mean = 822.39, std = 621.89

References

1. https://stat.ethz.ch/pipermail/r-help//2012-August/333656.html
2. http://digitalpbk.com/perl/perl-script-check-google-pagerank
3. http://www.google.com
4. http://jakevdp.github.io/blog/2012/10/14/scipy-sparse-graph-module-word-ladders/
5. http://curl.haxx.se/docs/httpscripting.html
6. http://www.crummy.com/software/BeautifulSoup/bs4/doc/
7. http://www.rmi.net/~lutz/
8. http://www.cs.cornell.edu/home/kleinber/networks-book/
9. http://thomassileo.com/blog/2013/01/25/using-twitter-rest-api-v1-dot-1-with-python/
10. http://www.cs.odu.edu/~mklein/cs796/lecture/