CS 532: Assignment 2

Dinesh Kumar Paladhi

Spring 2016

Contents

1 Problem 1

1. Write a Python program that extracts 1000 unique links from Twitter. You might want to take a look at:

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

2. But there are many other similar resources available on the web. Note that only Twitter API 1.1 is currently available; version 1 code will no longer work.

3. Also note that you need to verify that the final target URI (i.e., the one that responds with a 200) is unique. You could have different shortened URIs for www.cnn.com (t.co, bit.ly, goo.gl, etc.).

4. You might want to use the search feature to find URIs, or you can pull them from the feed of someone famous (e.g., Tim O'Reilly).

   Hold on to this collection – we'll use it later throughout the semester.

1.1 Solution

Extracting 1000 unique links from Twitter was more complex endeavor than expected.

The following steps were taken in order to get the links

1. I did not know where to start,so first i referenced the link provided in the question. This link gave me good idea on how to solve the problem.

2. I got to know that we need to create an app in "https://apps.twitter.com/" and then from there i can get my keys in order to hit twitter and get urls.

3. I wrote the code such that i can search for some key words in all the tweets in twitter and get only those tweets which have my key word.

4. This Key word should be given every single time through command prompt and i have queried Subway,Walmart,Trump,McDonalds and Old Dominion to get 1000 urls.

5. I extracted each url using urllib library and saved urls which return status-code as 200 OK.

6. I have had a counter to count my urls and i saved the twitter id,tweet creation date,tweet text and the urls.

7. I preferred to store these files in json format so that it will be well structured and can be easily used in future.

8. I had gone through a lot of problems with the first program for writing data into json file.

9. There were many other errors coming and i managed to resolve few of those with the help of except block.

10. Results of my python code are shown below. The output is in the json format.

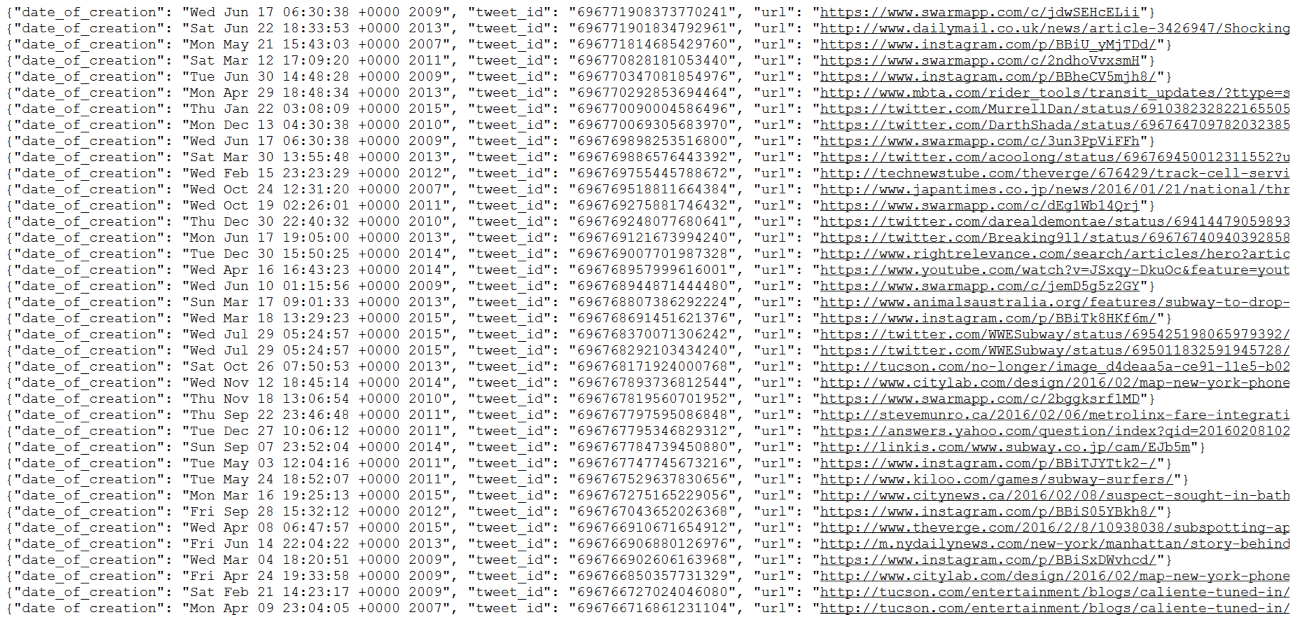
1.2 Code Listing

Here is the python code that is used to collect Thousand URL's from twitter.

```
# I have queried subway,walmart,trump,McDonalds,Old Dominion
import urllib
import json
import tweepy
import sys
import time

ACCESS_TOKEN = '118623489-QsSuqItzx8cnReRHI67ylffqpOPNs7z4Qp8hcOiI'  # Variables that
    contains the user credentials to access Twitter API
ACCESS_SECRET = 'PAPovgDO6QPy9QV8BbllM8p2MGWrcLLD8pesMHjXxTEMl'
CONSUMER_KEY = 'wxSZ8GSC7aRC7dAsM3m7UqgIg'
CONSUMER_SECRET = 'HuauQk780HuKWyQky9e4J6QM1DlwVxHXuvrLbgHGWhkmRXlvE4'
i=0

auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)   #Authentication is handled by
    the tweepy.AuthHandler class
auth.set_access_token(ACCESS_TOKEN, ACCESS_SECRET)


api = tweepy.API(auth) # Construct the API instance

var=sys.argv[1]                                                          #Takes input from
    command prompt
list_of_urls=set()                          #declaring url list as set to avoid duplicate url's
output_file=open('my_json_data','a')                              #
Final_results = tweepy.Cursor(api.search,q=var).items()
while True:
        try:
                status = Final_results.next()
                element=status._json
                one_element={} #  below code is to write data into json format

                i=i+1
                for url in element['entities']['urls']: # to get url's from each tweet
                        if len(list_of_urls) == 1000:  #url count
                                break
                        else:
                                response=urllib.urlopen(url['url'])
                                final_url=response.url
                                print response.getcode()
                                print final_url
                                print len(list_of_urls)
                                if response.getcode()==200: #checking where status code is
                                    200 or not
                                        if final_url in list_of_urls:
                                                break
                                        else:
                                                one_element['tweet_id'] = element['id_str']
                                                    #storing tweet id and date of creation
                                                    of it in json file
                                                one_element['date_of_creation'] = element['
                                                    user']['created_at']
                                                list_of_urls.add(final_url)
                                                one_element['url']=final_url
                                                output_file.write(json.dumps(one_element)+'\
                                                    n') #adding each element into json file

                if len(list_of_urls) == 1000:  #url count
                        break
        except tweepy.TweepError :
                time.sleep(60 * 1)
        continue
print len(list_of_urls)
```

Listing 1: Python program for acquiring 1000 unique links for a given keyword

1.3 Results



```
{"date_of_creation": "Wed Jun 17 06:30:38 +0000 2009", "tweet_id": "696771908373770241", "url": "https://www.swarmapp.com/c/jdwSEHcELii"}
{"date_of_creation": "Sat Jun 22 18:33:53 +0000 2013", "tweet_id": "696771901834792961", "url": "http://www.dailymail.co.uk/news/article-3426947/Shocking
{"date_of_creation": "Mon May 21 15:43:03 +0000 2007", "tweet_id": "696771814685429760", "url": "https://www.instagram.com/p/BBiU_yMjTDd/"}
{"date_of_creation": "Sat Mar 12 17:09:20 +0000 2011", "tweet_id": "696770828181053440", "url": "https://www.swarmapp.com/c/2ndhoVvxsmH"}
{"date_of_creation": "Tue Jun 30 14:48:28 +0000 2009", "tweet_id": "696770347081854976", "url": "https://www.instagram.com/p/BBheCV5mjh8/"}
{"date_of_creation": "Mon Apr 29 18:48:34 +0000 2013", "tweet_id": "696770292853694464", "url": "http://www.mbta.com/rider_tools/transit_updates/?ttype=s
{"date_of_creation": "Thu Jan 22 03:08:09 +0000 2015", "tweet_id": "696770090004586496", "url": "https://twitter.com/MurrellDan/status/691038232822165505
{"date_of_creation": "Mon Dec 13 04:30:38 +0000 2010", "tweet_id": "696770069305683970", "url": "https://twitter.com/DarthShada/status/696764709782032385
{"date_of_creation": "Wed Jun 17 06:30:38 +0000 2009", "tweet_id": "696769898253516800", "url": "https://www.swarmapp.com/c/3un3PpViFFh"}
{"date_of_creation": "Sat Mar 30 13:55:48 +0000 2013", "tweet_id": "696769886576443392", "url": "https://twitter.com/acoolong/status/696769450012311552?u
{"date_of_creation": "Wed Feb 15 23:23:29 +0000 2012", "tweet_id": "696769755445788672", "url": "http://technewstube.com/theverge/676429/track-cell-servi
{"date_of_creation": "Wed Oct 24 12:31:20 +0000 2007", "tweet_id": "696769518811664384", "url": "http://www.japantimes.co.jp/news/2016/01/21/national/thr
{"date_of_creation": "Wed Oct 19 02:26:01 +0000 2011", "tweet_id": "696769275881746432", "url": "https://www.swarmapp.com/c/dEg1Wb14Qrj"}
{"date_of_creation": "Thu Dec 30 22:40:32 +0000 2010", "tweet_id": "696769248077680641", "url": "https://twitter.com/darealdemontae/status/69414479059893
{"date_of_creation": "Mon Jun 17 19:05:00 +0000 2013", "tweet_id": "696769121673994240", "url": "https://twitter.com/Breaking911/status/69676740940392858
{"date_of_creation": "Tue Dec 30 15:50:25 +0000 2014", "tweet_id": "696769007701987328", "url": "http://www.rightrelevance.com/search/articles/hero?artic
{"date_of_creation": "Wed Apr 16 16:43:23 +0000 2014", "tweet_id": "696768957999616001", "url": "https://www.youtube.com/watch?v=JSxqy-DkuOc&feature=yout
{"date_of_creation": "Wed Jun 10 01:15:56 +0000 2009", "tweet_id": "696768944871444480", "url": "https://www.swarmapp.com/c/jemD5g5z2GY"}
{"date_of_creation": "Sun Mar 17 09:01:33 +0000 2013", "tweet_id": "696768807386292224", "url": "http://www.animalsaustralia.org/features/subway-to-drop-
{"date_of_creation": "Wed Mar 18 13:29:23 +0000 2015", "tweet_id": "696768691451621376", "url": "https://www.instagram.com/p/BBiTk8HKf6m/"}
{"date_of_creation": "Wed Jul 29 05:24:57 +0000 2015", "tweet_id": "696768370071306242", "url": "https://twitter.com/WWESubway/status/695425198065979392/
{"date_of_creation": "Wed Jul 29 05:24:57 +0000 2015", "tweet_id": "696768292103434240", "url": "https://twitter.com/WWESubway/status/695011832591945728/
{"date_of_creation": "Sat Oct 26 07:50:53 +0000 2013", "tweet_id": "696768171924000768", "url": "http://tucson.com/no-longer/image_d4deaa5a-ce91-11e5-b02
{"date_of_creation": "Wed Nov 12 18:45:14 +0000 2014", "tweet_id": "696767893736812544", "url": "http://www.citylab.com/design/2016/02/map-new-york-phone
{"date_of_creation": "Thu Nov 18 13:06:54 +0000 2010", "tweet_id": "696767819560701952", "url": "https://www.swarmapp.com/c/2bggksrf1MD"}
{"date_of_creation": "Thu Sep 22 23:46:48 +0000 2011", "tweet_id": "696767797595086848", "url": "http://stevemunro.ca/2016/02/06/metrolinx-fare-integrati
{"date_of_creation": "Tue Dec 27 10:06:12 +0000 2011", "tweet_id": "696767795346829312", "url": "https://answers.yahoo.com/question/index?qid=20160208102
{"date_of_creation": "Sun Sep 07 23:52:04 +0000 2014", "tweet_id": "696767784739450880", "url": "http://linkis.com/www.subway.co.jp/cam/EJb5m"}
{"date_of_creation": "Tue May 03 12:04:16 +0000 2011", "tweet_id": "696767747745673216", "url": "https://www.instagram.com/p/BBiTJYTtk2-/"}
{"date_of_creation": "Tue May 24 18:52:07 +0000 2011", "tweet_id": "696767529637830656", "url": "http://www.kiloo.com/games/subway-surfers/"}
{"date_of_creation": "Mon Mar 16 19:25:13 +0000 2015", "tweet_id": "696767275165229056", "url": "http://www.citynews.ca/2016/02/08/suspect-sought-in-bath
{"date_of_creation": "Fri Sep 28 15:32:12 +0000 2012", "tweet_id": "696767043652026368", "url": "https://www.instagram.com/p/BBiS05YBkh8/"}
{"date_of_creation": "Wed Apr 08 06:47:57 +0000 2015", "tweet_id": "696766910671654912", "url": "http://www.theverge.com/2016/2/8/10938038/subspotting-ap
{"date_of_creation": "Fri Jun 14 22:04:22 +0000 2013", "tweet_id": "696766906880126976", "url": "http://m.nydailynews.com/new-york/manhattan/story-behind
{"date_of_creation": "Wed Mar 04 18:20:51 +0000 2009", "tweet_id": "696766902606163968", "url": "https://www.instagram.com/p/BBiSxDWvhcd/"}
{"date_of_creation": "Fri Apr 24 19:33:58 +0000 2009", "tweet_id": "696766850357731329", "url": "http://www.citylab.com/design/2016/02/map-new-york-phone
{"date_of_creation": "Sat Feb 21 14:23:17 +0000 2009", "tweet_id": "696766727024046080", "url": "http://tucson.com/entertainment/blogs/caliente-tuned-in/
{"date_of_creation": "Mon Apr 09 23:04:05 +0000 2007", "tweet_id": "696766716861231104", "url": "http://tucson.com/entertainment/blogs/caliente-tuned-in/
```

Figure 1: Sample Url's

2 Problem 2

```
Download the TimeMaps for each of the target URIs.  We'll use the ODU
Memento Aggregator, so for example:

URI-R = http://www.cs.odu.edu/

URI-T = http://mementoproxy.cs.odu.edu/aggr/timemap/link/1/http://www.cs.odu.edu/

Create a histogram* of URIs vs. number of Mementos (as computed from
the TimeMaps).  For example, 100 URIs with 0 Mementos, 300 URIs
with 1 Memento, 400 URIs with 2 Mementos, etc.

* = https://en.wikipedia.org/wiki/Histogram
```

2.1 Solution

1. First i played with the memento aggregator in the command prompt and i was getting timemaps for each url and i found mementos in each time map.
2. Now i got to know what to do so i started to write a python code to calculate mementos with urls given as input.
3. I got a lot of errors and had a hard time working with those. So,I was searching for any other way to find the mementos.
4. I took the URI-T and pasted it in the browser and ran it.Surprisingly a file downloaded and i opened it in notepad++ and was happy to see all the time maps in it.
5. So now i switched to this method and used urllib library to run all the urls in the browser and got their respective time maps.
6. I got 404 error for some URI's which indicated that they have no mementos.I had written one function called getmementos(url) in my code to get urls having mementos.
7. Now my next task was to find and count the number of mementos. I used regular expression which finds all the expressions like rel=mementos,rel=first memento and rel=memento last from the time maps..
8. I ran a for loop and counted the number of mementos for each url and saved the url and number of mementos in json format.
9. I observed that out of 1000 only 177 urls had 1 or more mementos and remaining 823 urls had 0 mementos.
10. I wrote code to get 3 output files,one with urls and number of mementos,one csv file only with the number of mementos which is used to give input to R and one json file with urls and number of mementos where number of mementos is greater than zero.
11. The last file is used to give input to the third question because we need to calculate age for only those urls which have mementos greater than or equal to one.

2.2 Code Listing

Code for counting mementos for each url

```
import re
import urllib2
import json
import sys

def getmementos(url):
        mem_prefix = 'http://mementoproxy.cs.odu.edu/aggr/timemap/link/1/' + url    #memento aggregator is concatenated with the url for which mementos should be found out
        try:
                response = urllib2.urlopen(mem_prefix)
                time_map = response.read()
        except urllib2.HTTPError:
                time_map = None
        return time_map

find_memento = re.compile(r'rel.*?=.*?"memento".*?')  # To find memento using regular expression
my_urls = open('my_json_data','r+')  #This file contains 1000 urls their tweets,tweet ids and created dates
output_file = open('mem_and_links.json','a')  # This file stores number of mementos for each url
output_file2 = open('only_count.csv','a')
output_file_carbon = open('mem_grt0.json','a')
one_element={}
count_of_mems = [] #array is created to store count
for line in my_urls.readlines(): #reads line by line
        each_line = json.loads(line)
        url = each_line['url']
        memento_data = getmementos(url)

        #print memento_data
        if memento_data == 'Null':
                count = 0
                one_element['num_of_mems'] = count
                one_element['url'] = url
                output_file.write(json.dumps(one_element)+'\n') #adding each element into json file
                #print count,"    ",url
        else:
                count = len(find_memento.findall(str(memento_data))) #forms an array where "memento"" is found and finds the length of that array
                # a=find_memento.findall(str(memento_data))
                # print a
                one_element['num_of_mems'] = count
                one_element['url'] = url
                output_file.write(json.dumps(one_element)+'\n') #adding each element into json file
                output_file2.write("%s\n" % (count))
                if one_element['num_of_mems'] != 0:
                        output_file_carbon.write(json.dumps(one_element)+'\n') # for getting urls and mementos for mementos > 0
                #output_file2.write('\r\n')
                #print count,"    ",url
output_file.close()
output_file2.close()
output_file_carbon.close()
```

Listing 2: Python program for counting mementos

R code for histogram

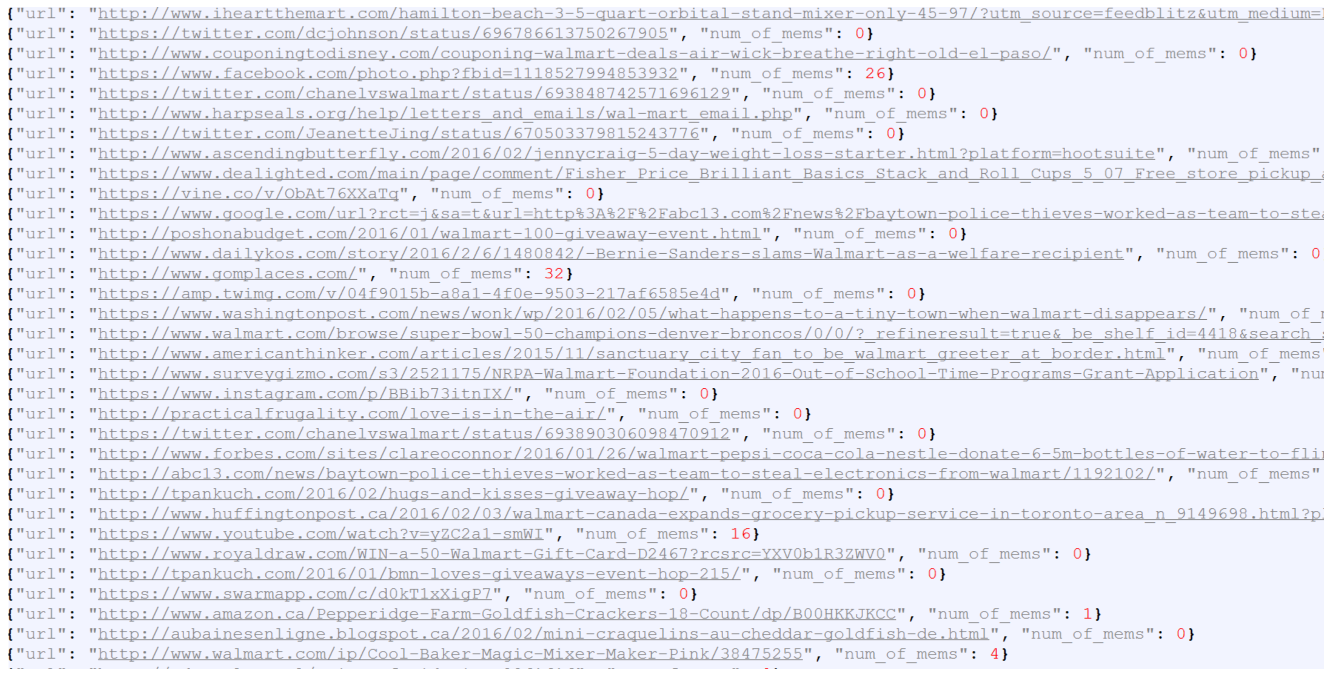
```

R version 3.2.3 (2015-12-10) -- "Wooden Christmas-Tree"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: i386-w64-mingw32/i386 (32-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> hist_mementos = scan("only_count.csv")
Read 1000 items
> hist(hist_mementos,xlab= "Number of Mementos" , ylab= "Number of URI's",main ="Histogram
    of momento/URI's",ylim=c(0,800),las=1,col=3)
> hist(hist_mementos,xlab= "Number of Mementos" , ylab= "Number of URI's",main ="Histogram
    of momento/URI's",xlim=c(0,150),ylim=c(0,200),las=1,breaks=100000,col=3)
>
```

Listing 3: R program for generating the histograms for Question 2

1. I had gone through the R tutorial once before plotting histogram with this data.
2. Then i passed my mementos count to R. Code used for R is shown above.
3. Histogram takes only one input and the frequency of the input is automatically taken on y-axis.
4. My first histogram graph is shown below. This graph shows that more than 800 urls have 0 mementos and very small number of urls have 1 or more mementos.
5. I wanted to elaborate this graph to show the number of urls that have greater than or equal to 1 memento. So, I scaled the histogram.
6. X-axis is scaled from 0 to 150 using xlim and y-axis is scaled from 0 to 200 using ylim. This graph is shown below and named as Histogram 2.

2.3 Results

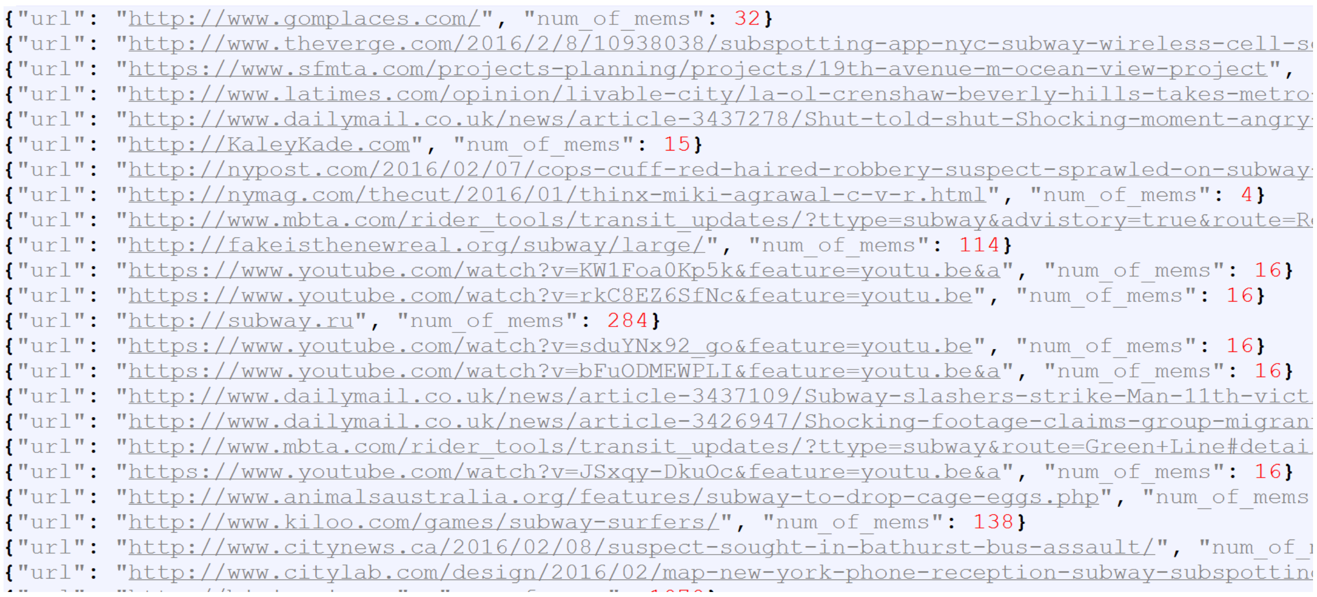
Output for the getting mementos



```
{"url": "http://www.iheartthemart.com/hamilton-beach-3-5-quart-orbital-stand-mixer-only-45-97/?utm_source=feedblitz&utm_medium=
{"url": "https://twitter.com/dcjohnson/status/696786613750267905", "num_of_mems": 0}
{"url": "http://www.couponingtodisney.com/couponing-walmart-deals-air-wick-breathe-right-old-el-paso/", "num_of_mems": 0}
{"url": "https://www.facebook.com/photo.php?fbid=1118527994853932", "num_of_mems": 26}
{"url": "https://twitter.com/chanelvswalmart/status/693848742571696129", "num_of_mems": 0}
{"url": "http://www.harpseals.org/help/letters_and_emails/wal-mart_email.php", "num_of_mems": 0}
{"url": "https://twitter.com/JeanetteJing/status/670503379815243776", "num_of_mems": 0}
{"url": "http://www.ascendingbutterfly.com/2016/02/jennycraig-5-day-weight-loss-starter.html?platform=hootsuite", "num_of_mems"
{"url": "https://www.dealighted.com/main/page/comment/Fisher_Price_Brilliant_Basics_Stack_and_Roll_Cups_5_07_Free_store_pickup_
{"url": "https://vine.co/v/ObAt76XXaTq", "num_of_mems": 0}
{"url": "https://www.google.com/url?rct=j&sa=t&url=http%3A%2F%2Fabc13.com%2Fnews%2Fbaytown-police-thieves-worked-as-team-to-ste
{"url": "http://poshonabudget.com/2016/01/walmart-100-giveaway-event.html", "num_of_mems": 0}
{"url": "http://www.dailykos.com/story/2016/2/6/1480842/-Bernie-Sanders-slams-Walmart-as-a-welfare-recipient", "num_of_mems": 0
{"url": "http://www.gomplaces.com/", "num_of_mems": 32}
{"url": "https://amp.twimg.com/v/04f9015b-a8a1-4f0e-9503-217af6585e4d", "num_of_mems": 0}
{"url": "https://www.washingtonpost.com/news/wonk/wp/2016/02/05/what-happens-to-a-tiny-town-when-walmart-disappears/", "num_of_
{"url": "http://www.walmart.com/browse/super-bowl-50-champions-denver-broncos/0/0/?_refineresult=true&_be_shelf_id=4418&search_
{"url": "http://www.americanthinker.com/articles/2015/11/sanctuary_city_fan_to_be_walmart_greeter_at_border.html", "num_of_mems
{"url": "http://www.surveygizmo.com/s3/2521175/NRPA-Walmart-Foundation-2016-Out-of-School-Time-Programs-Grant-Application", "nu
{"url": "https://www.instagram.com/p/BBib73itnIX/", "num_of_mems": 0}
{"url": "http://practicalfrugality.com/love-is-in-the-air/", "num_of_mems": 0}
{"url": "https://twitter.com/chanelvswalmart/status/693890306098470912", "num_of_mems": 0}
{"url": "http://www.forbes.com/sites/clareoconnor/2016/01/26/walmart-pepsi-coca-cola-nestle-donate-6-5m-bottles-of-water-to-fli
{"url": "http://abc13.com/news/baytown-police-thieves-worked-as-team-to-steal-electronics-from-walmart/1192102/", "num_of_mems"
{"url": "http://tpankuch.com/2016/02/hugs-and-kisses-giveaway-hop/", "num_of_mems": 0}
{"url": "http://www.huffingtonpost.ca/2016/02/03/walmart-canada-expands-grocery-pickup-service-in-toronto-area_n_9149698.html?p
{"url": "https://www.youtube.com/watch?v=yZC2a1-smWI", "num_of_mems": 16}
{"url": "http://www.royaldraw.com/WIN-a-50-Walmart-Gift-Card-D2467?rcsrc=YXV0b1R3ZWV0", "num_of_mems": 0}
{"url": "http://tpankuch.com/2016/01/bmn-loves-giveaways-event-hop-215/", "num_of_mems": 0}
{"url": "https://www.swarmapp.com/c/d0kT1xXigP7", "num_of_mems": 0}
{"url": "http://www.amazon.ca/Pepperidge-Farm-Goldfish-Crackers-18-Count/dp/B00HKKJKCC", "num_of_mems": 1}
{"url": "http://aubainesenligne.blogspot.ca/2016/02/mini-craquelins-au-cheddar-goldfish-de.html", "num_of_mems": 0}
{"url": "http://www.walmart.com/ip/Cool-Baker-Magic-Mixer-Maker-Pink/38475255", "num_of_mems": 4}
```

Figure 2: Links along with the number of mementos

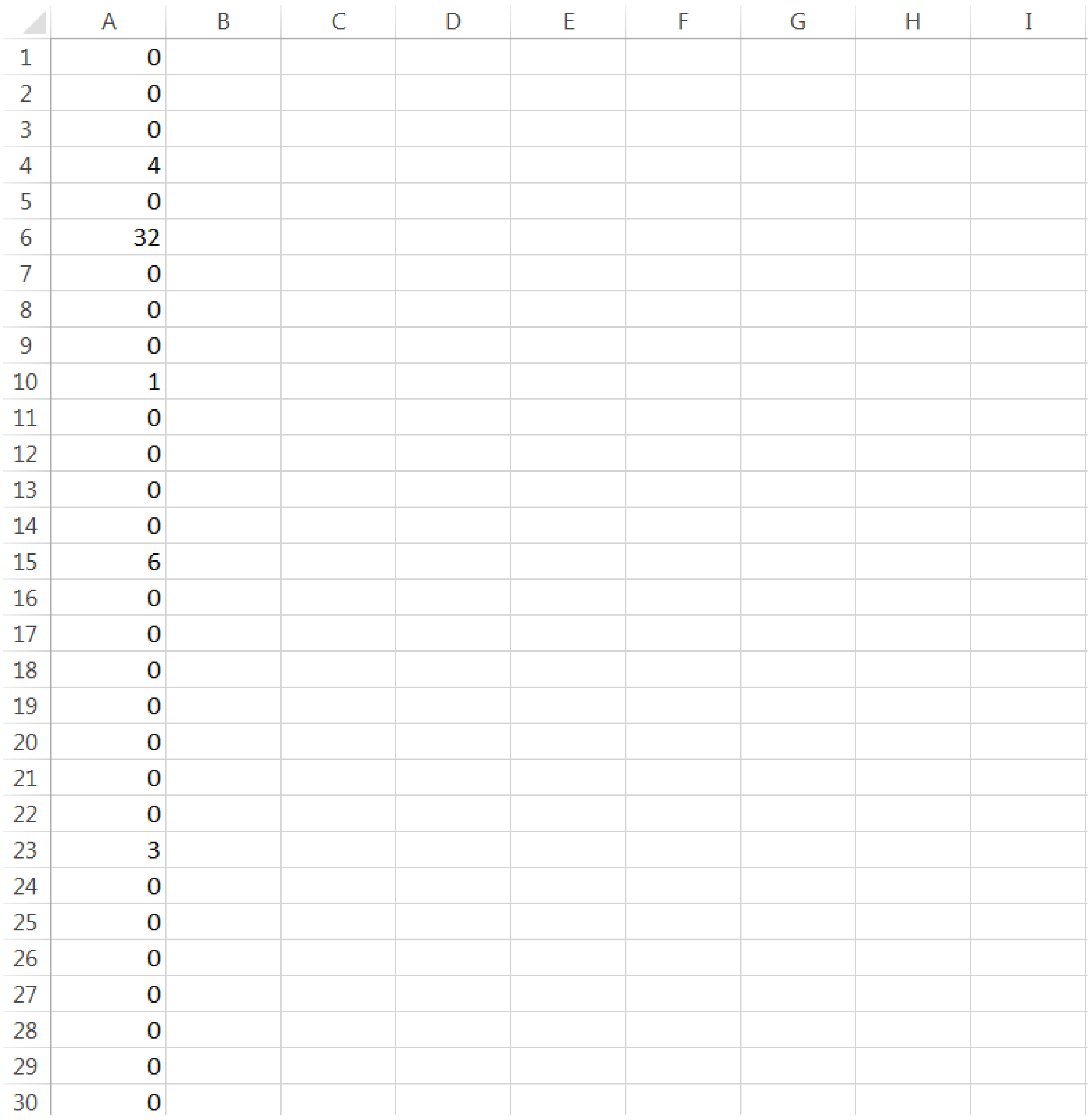
Urls with Mementos greater Zero



```
{"url": "http://www.gomplaces.com/", "num_of_mems": 32}
{"url": "http://www.theverge.com/2016/2/8/10938038/subspotting-app-nyc-subway-wireless-cell-s
{"url": "https://www.sfmta.com/projects-planning/projects/19th-avenue-m-ocean-view-project",
{"url": "http://www.latimes.com/opinion/livable-city/la-ol-crenshaw-beverly-hills-takes-metro
{"url": "http://www.dailymail.co.uk/news/article-3437278/Shut-told-shut-Shocking-moment-angry
{"url": "http://KaleyKade.com", "num_of_mems": 15}
{"url": "http://nypost.com/2016/02/07/cops-cuff-red-haired-robbery-suspect-sprawled-on-subway
{"url": "http://nymag.com/thecut/2016/01/thinx-miki-agrawal-c-v-r.html", "num_of_mems": 4}
{"url": "http://www.mbta.com/rider_tools/transit_updates/?ttype=subway&advistory=true&route=R
{"url": "http://fakeisthenewreal.org/subway/large/", "num_of_mems": 114}
{"url": "https://www.youtube.com/watch?v=KW1Foa0Kp5k&feature=youtu.be&a", "num_of_mems": 16}
{"url": "https://www.youtube.com/watch?v=rkC8EZ6SfNc&feature=youtu.be", "num_of_mems": 16}
{"url": "http://subway.ru", "num_of_mems": 284}
{"url": "https://www.youtube.com/watch?v=sduYNx92_go&feature=youtu.be", "num_of_mems": 16}
{"url": "https://www.youtube.com/watch?v=bFuODMEWPLI&feature=youtu.be&a", "num_of_mems": 16}
{"url": "http://www.dailymail.co.uk/news/article-3437109/Subway-slashers-strike-Man-11th-vict
{"url": "http://www.dailymail.co.uk/news/article-3426947/Shocking-footage-claims-group-migran
{"url": "http://www.mbta.com/rider_tools/transit_updates/?ttype=subway&route=Green+Line#detai
{"url": "https://www.youtube.com/watch?v=JSxqy-DkuOc&feature=youtu.be&a", "num_of_mems": 16}
{"url": "http://www.animalsaustralia.org/features/subway-to-drop-cage-eggs.php", "num_of_mems
{"url": "http://www.kiloo.com/games/subway-surfers/", "num_of_mems": 138}
{"url": "http://www.citynews.ca/2016/02/08/suspect-sought-in-bathurst-bus-assault/", "num_of_
{"url": "http://www.citylab.com/design/2016/02/map-new-york-phone-reception-subway-subspottin
```

Figure 3: Links along with the mementos greater than zero

Mementos count



	A	B	C	D	E	F	G	H	I
1	0								
2	0								
3	0								
4	4								
5	0								
6	32								
7	0								
8	0								
9	0								
10	1								
11	0								
12	0								
13	0								
14	0								
15	6								
16	0								
17	0								
18	0								
19	0								
20	0								
21	0								
22	0								
23	3								
24	0								
25	0								
26	0								
27	0								
28	0								
29	0								
30	0								

Figure 4: Excel sheet with count of number of mementos

Histograms

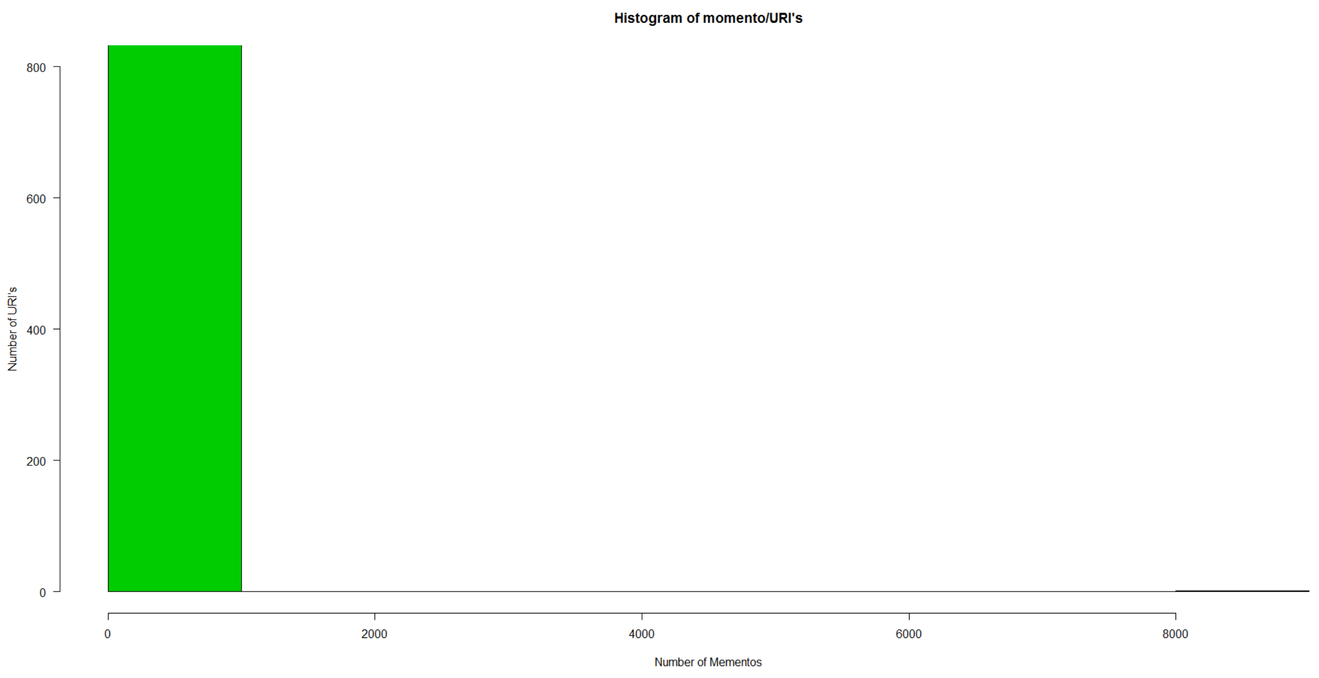


Figure 5: Histogram 1

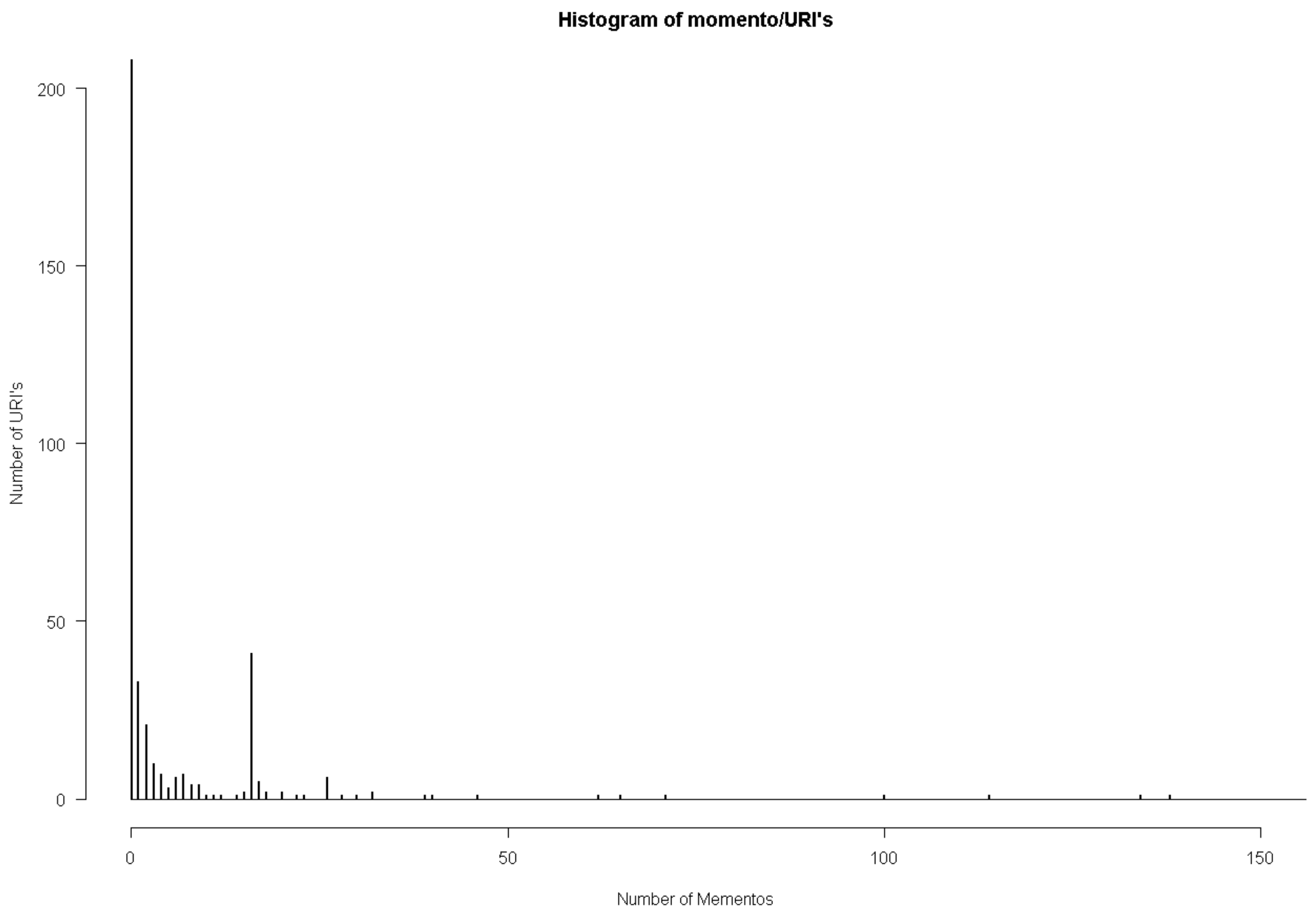


Figure 6: Histogram 2

3 Problem 3

```
Estimate the age of each of the 1000 URIs using the ``Carbon Date'' tool:

http://ws-dl.blogspot.com/2014/11/2014-11-14-carbon-dating-web-version-20.html

Note: you'll should download the library and run it locally; don't
try to use the web service.

For URIs that have > 0 Mementos and an estimated creation date,
create a graph with age (in days) on one axis and number of mementos
on the other.

Not all URIs will have Mementos, and not all URIs will have an estimated
creation date.  State how many fall into either categories
```

3.1 Solution

1. Aim of this question is to find out the age of each url in days. But only those urls should be considered which have more than 0 mementos.
2. I already filtered all those urls with more than 0 mementos from my previous code and that file is given as input to the code written for this question.
3. I referred the link given in the question and downloaded the carbon date tool zip file. From the link i got to know that there are 3 methods to find the created date of each link.
4. First one was web service which would take a whole lot of time to process my 177 urls. Other two were local.py and server.py.
5. I randomly picked local.py and understood the code. There was already one function defined which takes url as argument and give creation date and many other parameters as output.
6. Now I opened my file with 177 urls and ran a loop which took each url and called the above function and each time i appended my created date for each url to my output file.
7. Figure 7 shows my output json file which has url and its respective creation date.
8. Now my task was to write code for calculation age of the url and this can be done by subtraction the creation date of the url from the present date.
9. Here i got stuck up for a lot of time while subtracting date there was constantly one error called some kind of valueError.
10. I searched for this error in Google and got to know that i should use strptime function in order to modify my creation date format into present date's format. This command eliminated 'T' variable which was present in our creation date.
11. Now i was successful in finding the age of each url in days and saved that to a csv file which shown below in Figure 8. I had also got number of mementos into a separate csv file so that i can easily pass my inputs to R.
12. Number of mementos saved in csv file is shown in Figure 9.
13. From the second question i already got some grip on R, so now i was able to pass the age and number of mementos as inputs to R which gave me a scatter plot as output.
14. Now this scatter plot shows that higher the number of mementos for a particular url higher is the age of that url. I had a maximum of 8999 mementos for 1 url and maximum age for 1 url was 6922 days.
15. The scatter plot is shown below in Figure 10.

3.2 Code Listing

Python code for calculating creation date for each url

```
from checkForModules import checkForModules
import json
from ordereddict import OrderedDict
#import simplejson
import urlparse
import re

from getBitly import getBitlyCreationDate
from getArchives import getArchivesCreationDate
from getGoogle import getGoogleCreationDate
from getBacklinks import *
from getLowest import getLowest

from getLastModified import getLastModifiedDate
#Topsy service is no longer available
#from getTopsyScrapper import getTopsyCreationDate
from htmlMessages import *
from pprint import pprint

from threading import Thread
import Queue
import datetime

import os,sys, traceback



one_element={}
def cd(url, backlinksFlag = False):

    #print 'Getting Creation dates for: ' + url


    #scheme missing?
    parsedUrl = urlparse.urlparse(url)
    if( len(parsedUrl.scheme)<1 ):
        url = 'http://'+url


    threads = []
    outputArray =['','','','','','']
    now0 = datetime.datetime.now()


    lastmodifiedThread = Thread(target=getLastModifiedDate, args=(url, outputArray, 0))
    bitlyThread = Thread(target=getBitlyCreationDate, args=(url, outputArray, 1))
    googleThread = Thread(target=getGoogleCreationDate, args=(url, outputArray, 2))
    archivesThread = Thread(target=getArchivesCreationDate, args=(url, outputArray, 3))

    if( backlinksFlag ):
        backlinkThread = Thread(target=getBacklinksFirstAppearanceDates, args=(url,
            outputArray, 4))

    #topsyThread = Thread(target=getTopsyCreationDate, args=(url, outputArray, 5))


    # Add threads to thread list
    threads.append(lastmodifiedThread)
    threads.append(bitlyThread)
    threads.append(googleThread)
    threads.append(archivesThread)

    if( backlinksFlag ):
        threads.append(backlinkThread)

    #threads.append(topsyThread)


    # Start new Threads
    lastmodifiedThread.start()
    bitlyThread.start()
    googleThread.start()
    archivesThread.start()

    if( backlinksFlag ):
        backlinkThread.start()

    #topsyThread.start()


    # Wait for all threads to complete
    for t in threads:
        t.join()

    # For threads
    lastmodified = outputArray[0]
    bitly = outputArray[1]
    google = outputArray[2]
    archives = outputArray[3]

    if( backlinksFlag ):
        backlink = outputArray[4]
    else:
        backlink = ''

    #topsy = outputArray[5]

    #note that archives["Earliest"] = archives[0][1]
    try:
        #lowest = getLowest([lastmodified, bitly, google, archives[0][1], backlink, topsy])
            #for thread
        lowest = getLowest([lastmodified, bitly, google, archives[0][1], backlink]) #for
            thread
    except:
       print sys.exc_type, sys.exc_value , sys.exc_traceback



    result = []

    result.append(("URI", url))
    result.append(("Estimated Creation Date", lowest))
    result.append(("Last Modified", lastmodified))
    result.append(("Bitly.com", bitly))
    result.append(("Topsy.com", "Topsy is out of service"))
    result.append(("Backlinks", backlink))
    result.append(("Google.com", google))
    result.append(("Archives", archives))
    values = OrderedDict(result)
    r = json.dumps(values, sort_keys=False, indent=2, separators=(',', ': '))

    now1 = datetime.datetime.now() - now0


    #print "runtime in seconds: "
    #print now1.seconds
    #print r
    print 'runtime in seconds:  ' + str(now1.seconds) + '\n' + r + '\n'
    file_1 = open('url_cdate.json','a')
    one_element['url'] = url
    one_element['created_date'] = lowest
    file_1.write(json.dumps(one_element)+'\n')
    file_1.close
    return r



q2_data=open("mem_grt0.json",'r')
for each_line in q2_data.readlines():
    json_line=json.loads(each_line)
    url=json_line['url']
    cd(url)
```

Listing 4: Python program for getting creation date for URI's

Python Code for Calculation age

```
from datetime import datetime
import json


file_1=open('url_cdate.json','r')
output_file=open('age.csv','w')
file_2=open('mem_grt0.json','r')
output_file2=open('num_mems.csv','w')
                #file_2=open('mem_grt0.json','r')
                #one_element={}
today_date=datetime.now()
#print today_date
#print (today_date-datetime.timedelta(2016,02,8,20,46,40)).days
#print "hello"
i=0

for each_line in file_1.readlines():
        dumy_line=json.loads(each_line)
        created_date = dumy_line['created_date']
        print created_date
        convert_date=datetime.strptime(created_date,"%Y-%m-%dT%H:%M:%S")
                #print convert_date
        age=(today_date - convert_date).days
                #print age
        #age_int=int(age)
        print age
        print i
        i=i+1
        output_file.write("%s\n" % (age))
for every_line in file_2.readlines():
        dumy_ev_line=json.loads(every_line)
        mems=dumy_ev_line['num_of_mems']
        output_file2.write("%s\n" % (mems))


output_file.close()
output_file2.close()
file_2.close()
file_1.close()
```

Listing 5: Python program for calculating the age of URI's

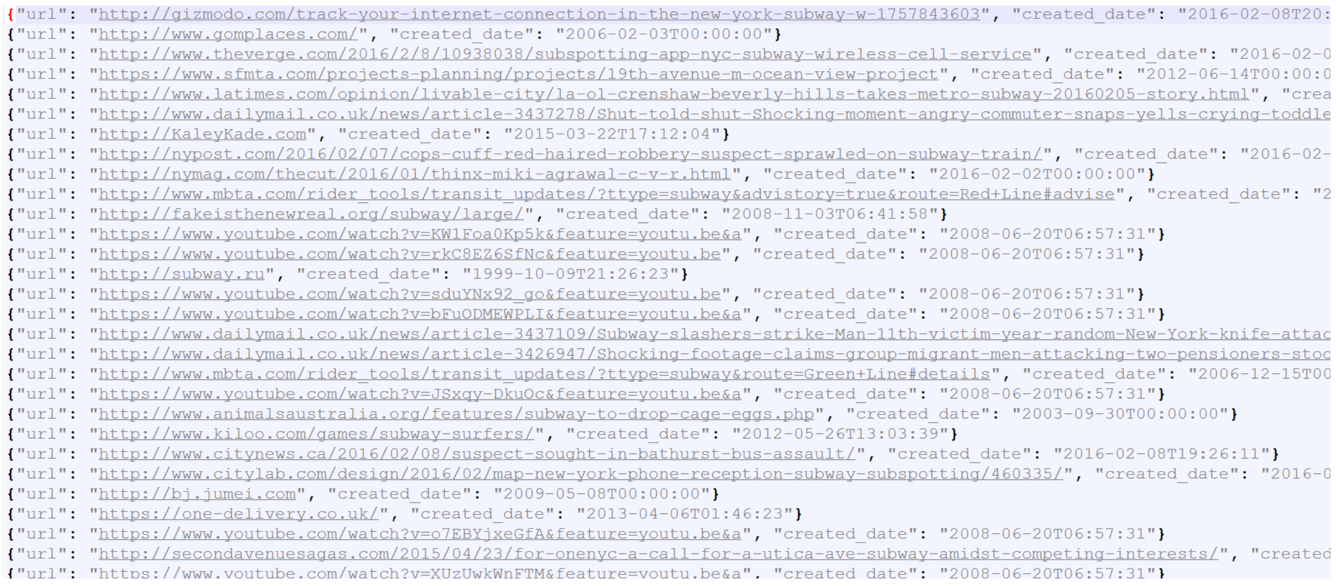
R code for scatter plot

```
R version 3.2.3 (2015-12-10) -- "Wooden Christmas-Tree"
 Copyright (C) 2015 The R Foundation for Statistical Computing
 Platform: i386-w64-mingw32/i386 (32-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
 You are welcome to redistribute it under certain conditions.
 Type 'license()' or 'licence()' for distribution details.

   Natural language support but running in an English locale

R is a collaborative project with many contributors.
 Type 'contributors()' for more information and
 'citation()' on how to cite R or R packages in publications.

 Type 'demo()' for some demos, 'help()' for on-line help, or
 'help.start()' for an HTML browser interface to help.
 Type 'q()' to quit R.

> age=scan("age.csv")
 Read 177 items
> mementos=scan("num_mems.csv")
 Read 177 items
> plot(mementos,age,xlab="Number of mementos",ylab="Age in days",main="Scatter plot for
     Mementos/Days",las=1)
>
```

Listing 6: R program for generating the scatter plot for Question 3

3.3 Results

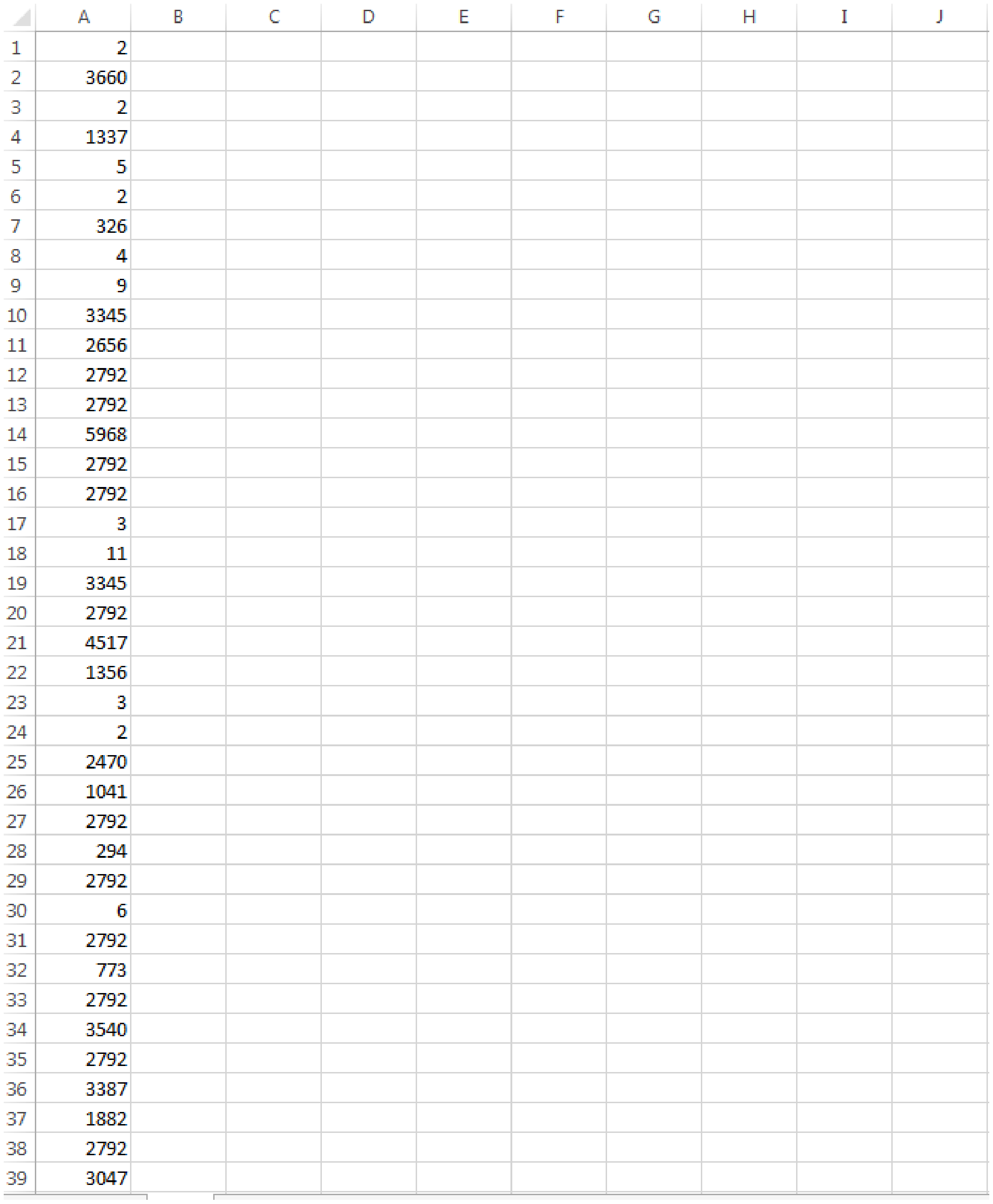
Creation date of each URL



```
{"url": "http://gizmodo.com/track-your-internet-connection-in-the-new-york-subway-w-1757843603", "created_date": "2016-02-08T20:
{"url": "http://www.gomplaces.com/", "created_date": "2006-02-03T00:00:00"}
{"url": "http://www.theverge.com/2016/2/8/10938038/subspotting-app-nyc-subway-wireless-cell-service", "created_date": "2016-02-0
{"url": "https://www.sfmta.com/projects-planning/projects/19th-avenue-m-ocean-view-project", "created_date": "2012-06-14T00:00:0
{"url": "http://www.latimes.com/opinion/livable-city/la-ol-crenshaw-beverly-hills-takes-metro-subway-20160205-story.html", "crea
{"url": "http://www.dailymail.co.uk/news/article-3437278/Shut-told-shut-Shocking-moment-angry-commuter-snaps-yells-crying-toddle
{"url": "http://KaleyKade.com", "created_date": "2015-03-22T17:12:04"}
{"url": "http://nypost.com/2016/02/07/cops-cuff-red-haired-robbery-suspect-sprawled-on-subway-train/", "created_date": "2016-02-
{"url": "http://nymag.com/thecut/2016/01/thinx-miki-agrawal-c-v-r.html", "created_date": "2016-02-02T00:00:00"}
{"url": "http://www.mbta.com/rider_tools/transit_updates/?ttype=subway&advistory=true&route=Red+Line#advise", "created_date": "2
{"url": "http://fakeisthenewreal.org/subway/large/", "created_date": "2008-11-03T06:41:58"}
{"url": "https://www.youtube.com/watch?v=KW1Foa0Kp5k&feature=youtu.be&a", "created_date": "2008-06-20T06:57:31"}
{"url": "https://www.youtube.com/watch?v=rkC8EZ6SfNc&feature=youtu.be", "created_date": "2008-06-20T06:57:31"}
{"url": "http://subway.ru", "created_date": "1999-10-09T21:26:23"}
{"url": "https://www.youtube.com/watch?v=sduYNx92_go&feature=youtu.be", "created_date": "2008-06-20T06:57:31"}
{"url": "https://www.youtube.com/watch?v=bFuODMEWPLI&feature=youtu.be&a", "created_date": "2008-06-20T06:57:31"}
{"url": "http://www.dailymail.co.uk/news/article-3437109/Subway-slashers-strike-Man-11th-victim-year-random-New-York-knife-attac
{"url": "http://www.dailymail.co.uk/news/article-3426947/Shocking-footage-claims-group-migrant-men-attacking-two-pensioners-stoc
{"url": "http://www.mbta.com/rider_tools/transit_updates/?ttype=subway&route=Green+Line#details", "created_date": "2006-12-15T00
{"url": "https://www.youtube.com/watch?v=JSxqy-DkuOc&feature=youtu.be&a", "created_date": "2008-06-20T06:57:31"}
{"url": "http://www.animalsaustralia.org/features/subway-to-drop-cage-eggs.php", "created_date": "2003-09-30T00:00:00"}
{"url": "http://www.kiloo.com/games/subway-surfers/", "created_date": "2012-05-26T13:03:39"}
{"url": "http://www.citynews.ca/2016/02/08/suspect-sought-in-bathurst-bus-assault/", "created_date": "2016-02-08T19:26:11"}
{"url": "http://www.citylab.com/design/2016/02/map-new-york-phone-reception-subway-subspotting/460335/", "created_date": "2016-0
{"url": "http://bj.jumei.com", "created_date": "2009-05-08T00:00:00"}
{"url": "https://one-delivery.co.uk/", "created_date": "2013-04-06T01:46:23"}
{"url": "https://www.youtube.com/watch?v=o7EBYjxeGfA&feature=youtu.be&a", "created_date": "2008-06-20T06:57:31"}
{"url": "http://secondavenuesagas.com/2015/04/23/for-onenyc-a-call-for-a-utica-ave-subway-amidst-competing-interests/", "created
{"url": "https://www.youtube.com/watch?v=XUzUwkWnFTM&feature=youtu.be&a", "created_date": "2008-06-20T06:57:31"}
```

Figure 7: Each Url along with its creation date

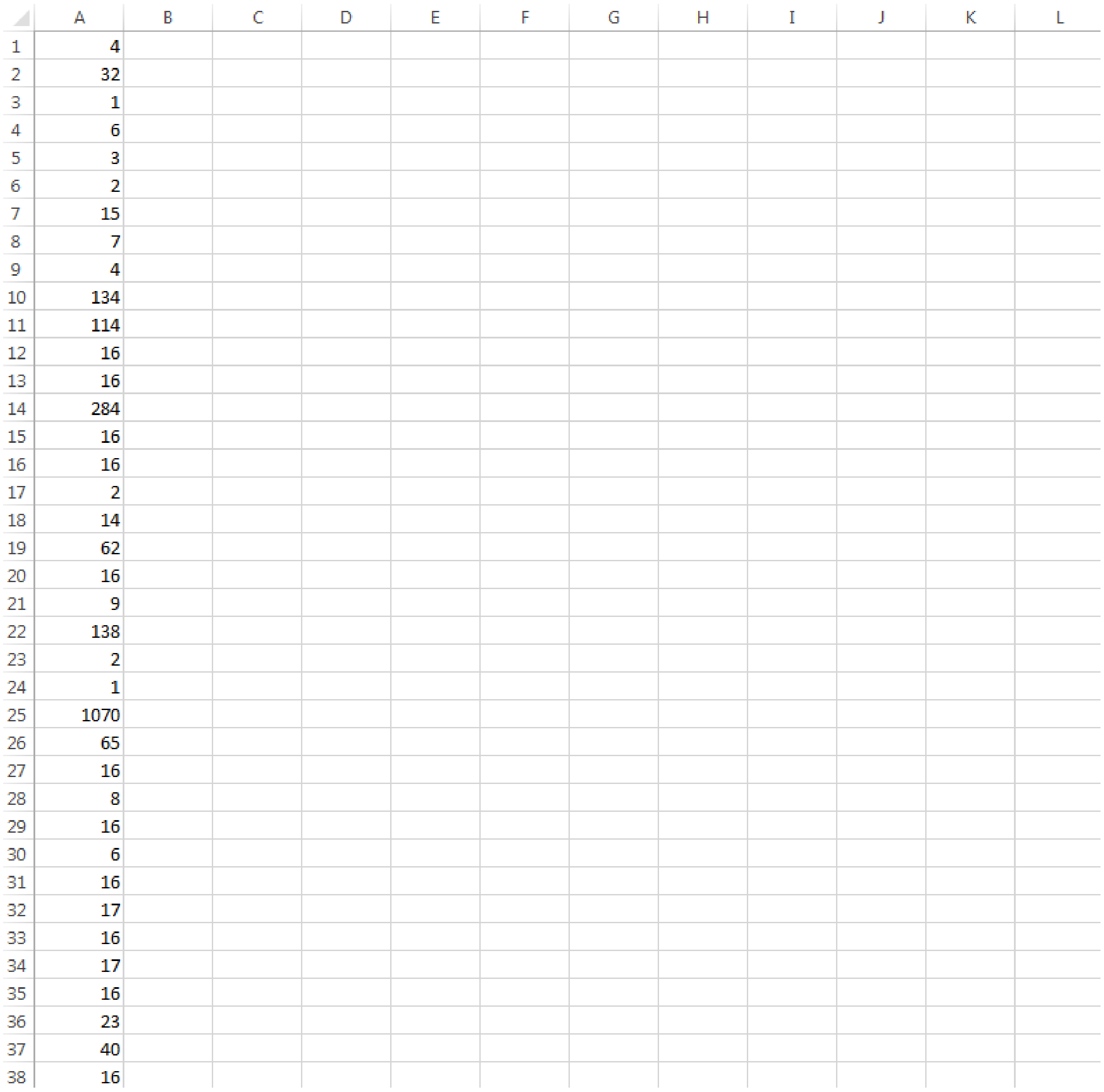
Age of each url



	A	B	C	D	E	F	G	H	I	J
1	2									
2	3660									
3	2									
4	1337									
5	5									
6	2									
7	326									
8	4									
9	9									
10	3345									
11	2656									
12	2792									
13	2792									
14	5968									
15	2792									
16	2792									
17	3									
18	11									
19	3345									
20	2792									
21	4517									
22	1356									
23	3									
24	2									
25	2470									
26	1041									
27	2792									
28	294									
29	2792									
30	6									
31	2792									
32	773									
33	2792									
34	3540									
35	2792									
36	3387									
37	1882									
38	2792									
39	3047									

Figure 8: Age of each url

Number of Mementos for each url



	A	B	C	D	E	F	G	H	I	J	K	L
1	4											
2	32											
3	1											
4	6											
5	3											
6	2											
7	15											
8	7											
9	4											
10	134											
11	114											
12	16											
13	16											
14	284											
15	16											
16	16											
17	2											
18	14											
19	62											
20	16											
21	9											
22	138											
23	2											
24	1											
25	1070											
26	65											
27	16											
28	8											
29	16											
30	6											
31	16											
32	17											
33	16											
34	17											
35	16											
36	23											
37	40											
38	16											

Figure 9: Number of Mementos for each url

Q3-scatterplot

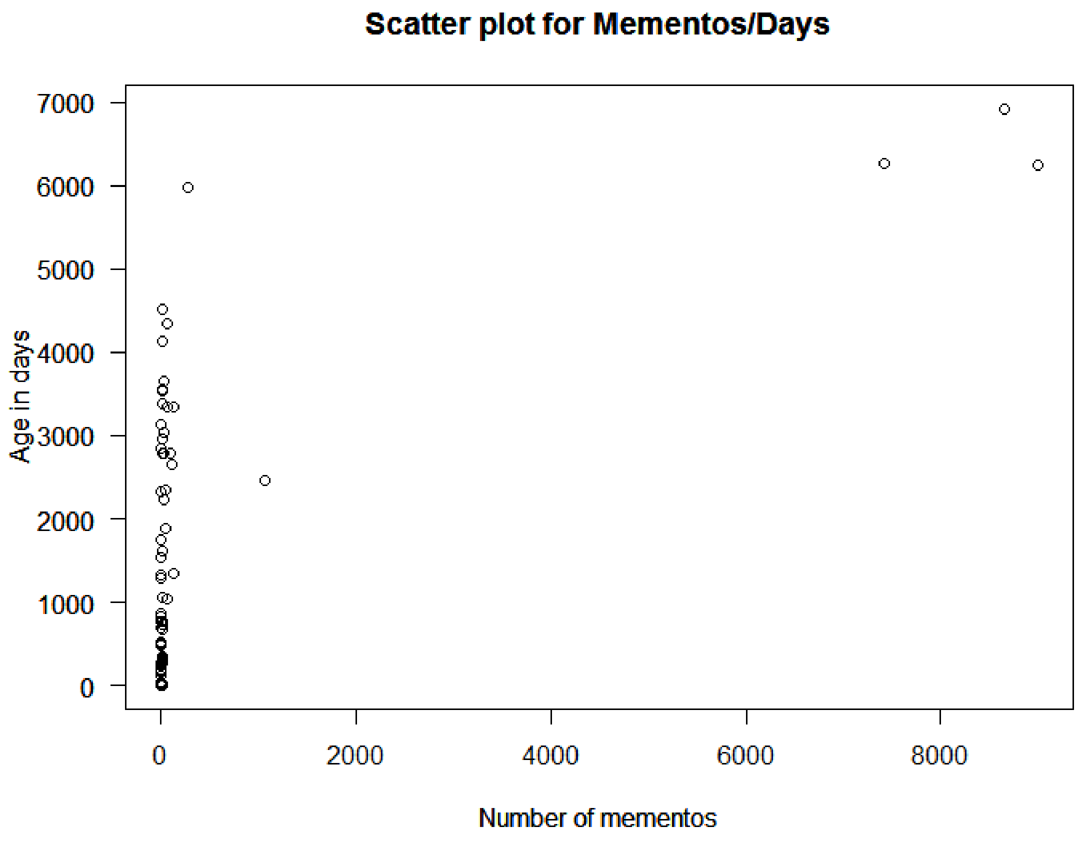


Figure 10: ScatterPlot

Bibliography

[1] Carbon dating tool:. http://ws-dl.blogspot.com/2014/11/2014-11-14-carbon-dating-web-version-20.html.

[2] Checking format of json:. http://jsonlint.com/.

[3] Mememto aggregator:. http://mementoproxy.cs.odu.edu/aggr/timemap/link/1/http://www.cs.odu.edu/.

[4] Processing twitter user data:. http://social-metrics.org/twitter-user-data/.

[5] Producing simple graphs in r:. http://www.harding.edu/fmccown/r/.

[6] Re library:. https://docs.python.org/2/library/re.html.

[7] Tweepy library:. http://docs.tweepy.org/en/latest/getting-started.html.

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

[9] Writing json data to file:. http://stackoverflow.com/questions/12309269/how-do-i-write-json-data-to-a-file-in-python.