

Assignment 1

Jessica McConnell

February 12, 2015

Q1

We were required to get 10,000 tweets, map the links in each tweet, get the headers for each link, determine the unique and duplicate URIs and build a histogram of redirects and HTTP status codes.

Getting the tweets and headers

I started my script for getting tweets by using the code provided at <http://thomassileo.com/blog/2013/01/25/using-twitter-rest-api-v1-dot-1-with-python/> for twitter connectivity. I then customized it to pull tweets, extract the creation date and links from the tweet, filter for content and get the headers for the links within the tweet. I chose to use the search/tweets API to ensure I had a myriad of different topics and links in the tweets I pulled.

For each word in my wordlist the tweets.py script would perform a search returning the maximum of 100 tweets. The script applied some filters to help ensure the returned tweets were not inappropriate and to check that the tweet had at least one link. If the tweet did not pass the restrictions, the tweet was skipped. I ended up having a wordlist of more than 2000 words to ensure I could get 10000 tweets.

After getting the links from the tweet, 'tweets.py' uses subprocess to get all the headers by calling 'curl -I *link*'.

The URI count, tweet URIs, headers for the URIs and JSON from the tweet were saved for each tweet in the file 'tweetsURI'.

Final URIs

I wrote a python script 'finaluri.py' that reads through the 'tweetsURI' file. This script originally sets *link* to the URI found in the tweet and then continues updating *link* until it has reached the end of the headers for that

URI. It then checks to see if it is in the unique URI list. If not, it is added to the unique URI list and the all URI list. If it is, we add it to the duplicate URI list if it is not already in there and the all URI list. There were two URIs from all the tweets that had the twitter t.co link as the final link.

<http://t.co/ryFCdcfQQC>

HEADERS:

HTTP/1.1 200 OK

cache-control: no-cache,no-store,must-revalidate

content-length: 0

content-type: text/html; charset=utf-8

date: Fri, 30 Jan 2015 13:27:22 UTC

server: tsa_b

x-connection-hash: 66ab6a89cec1d5a8b920e1e3eda9980d

x-response-time: 12

x-xss-protection: 1; mode=block

<http://t.co/YwwhBJrvfx>

HEADERS:

HTTP/1.1 200 OK

cache-control: no-cache,no-store,must-revalidate

content-length: 0

content-type: text/html; charset=utf-8

date: Fri, 30 Jan 2015 17:28:53 UTC

server: tsa_b

x-connection-hash: 0cd8a3e079647045036b4ba7f77e3623

x-response-time: 11

x-xss-protection: 1; mode=block

The unique and duplicate URIs were outputted to the files ‘uniqueuri’ and ‘dupuri’ respectively.

URIs:

- All URIs: 10347
- Duplicate URIs: 1071
- Unique URIs: 7406

Redirects

I used a bash command to get the redirect code numbers from the headers.

Listing 1: bash code

```
status=$(egrep ^HTTP/1.1 tweetsURI | cut -d' ' -f2- | egrep ^3)
lower=$(echo "$status" | tr '[:upper:]' '[:lower:]')
echo "$lower" | sort | uniq -c | sort -nr > redirects
```

Redirect Codes:

- 19835 301 moved permanently
- 1201 302 found
- 540 302 moved temporarily
- 208 303 see other
- 95 301 ok
- 21 307 temporary redirect
- 15 302 object moved
- 11 301 apple webobjects
- 3 302
- 2 301 www redirect
- 2 301 moved permanently
- 1 302 please wait
- 1 301 <http://hee.nhs.uk/work-programmes/tel/hack>
- 1 301 found
- 1 301

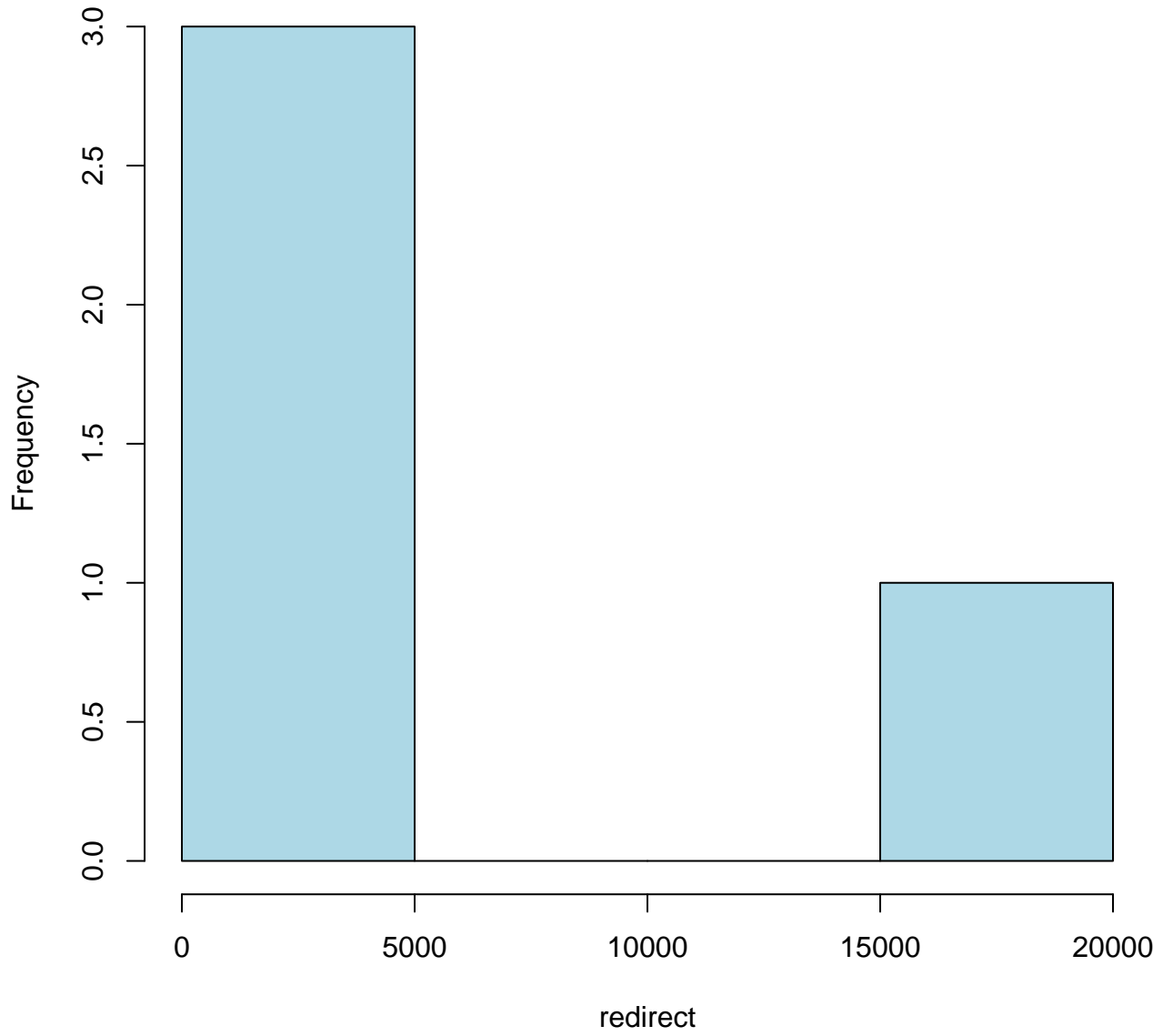
Histogram:

NOTE: For the histogram I consolidated the above information into groups of individual code numbers instead of the specific textual responses.

Status Codes

I used a bash command to get the status code numbers from the headers.

Redirects Status Count



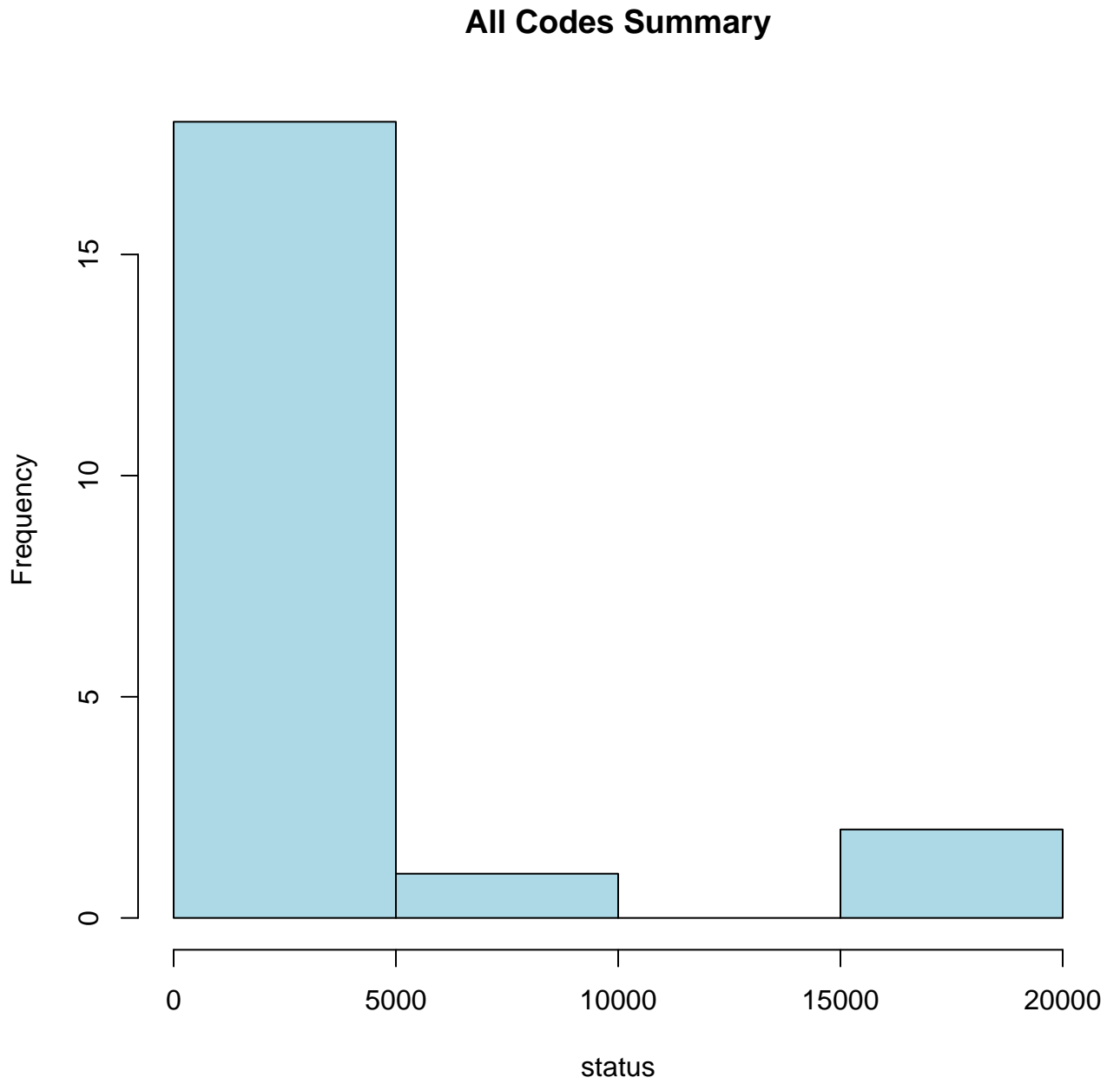
Listing 2: bash code

```
status=$(egrep ^HTTP/1.1 tweetsURI | cut -d' ' -f1,2)
echo "$status" | sort | uniq -c | sort -nr > statuses
```

Status codes:

- 19947 HTTP/1.1 301
- 9782 HTTP/1.1 200
- 1757 HTTP/1.1 302
- 208 HTTP/1.1 303
- 191 HTTP/1.1 405
- 107 HTTP/1.1 404
- 47 HTTP/1.1 503
- 46 HTTP/1.1 403
- 42 HTTP/1.1 406
- 21 HTTP/1.1 307
- 16 HTTP/1.1 500
- 7 HTTP/1.1 416
- 3 HTTP/1.1 508
- 3 HTTP/1.1 400
- 3 HTTP/1.1 302
- 1 HTTP/1.1 504
- 1 HTTP/1.1 502
- 1 HTTP/1.1 501
- 1 HTTP/1.1 410
- 1 HTTP/1.1 301
- 1 HTTP/1.1 204

Histogram:



Q2

We were required to get the carbon date for all the links in the tweets, get the deltas between the age of the tweet and age of the link and perform statistical operations on the deltas.

Running the Carbon Date Utility

Since I had several duplicate URIs I decided to perform the carbon date on the final URIs instead of the links in the tweets directly. I assigned each URI a number and created a directory for it in the carbondate directory. Within the numbered directory for each individual URI I house the results from ‘python local.py *url*’.

I wrote two scripts to perform my carbon date lookups. The main script is getTimes.sh. It is called by `./getTimes.sh folder url`. This creates the numbered folder for the URI and calls ‘python local.py *url*’ outputting the results to *folder/cd*. The second script is called getAll.sh. This runs through a list of URIs and assigned folder numbers and calls the getTimes.sh script with the information provided in each line.

I also received several ‘sh: 1: Syntax error: Unterminated quoted string’ errors. I am not sure if this affected receiving responses for dates from the carbon date utility.

There were 4 URIs that were hung when I used my scripts. I ended up using the carbon date utility provided at <http://cd.cs.odu.edu/cd>. By doing this I learned that they were having issues because of some of the characters that were in the URIs. I fixed this by changing the characters from ascii to url format. For example: I changed ‘ ’ to %27.

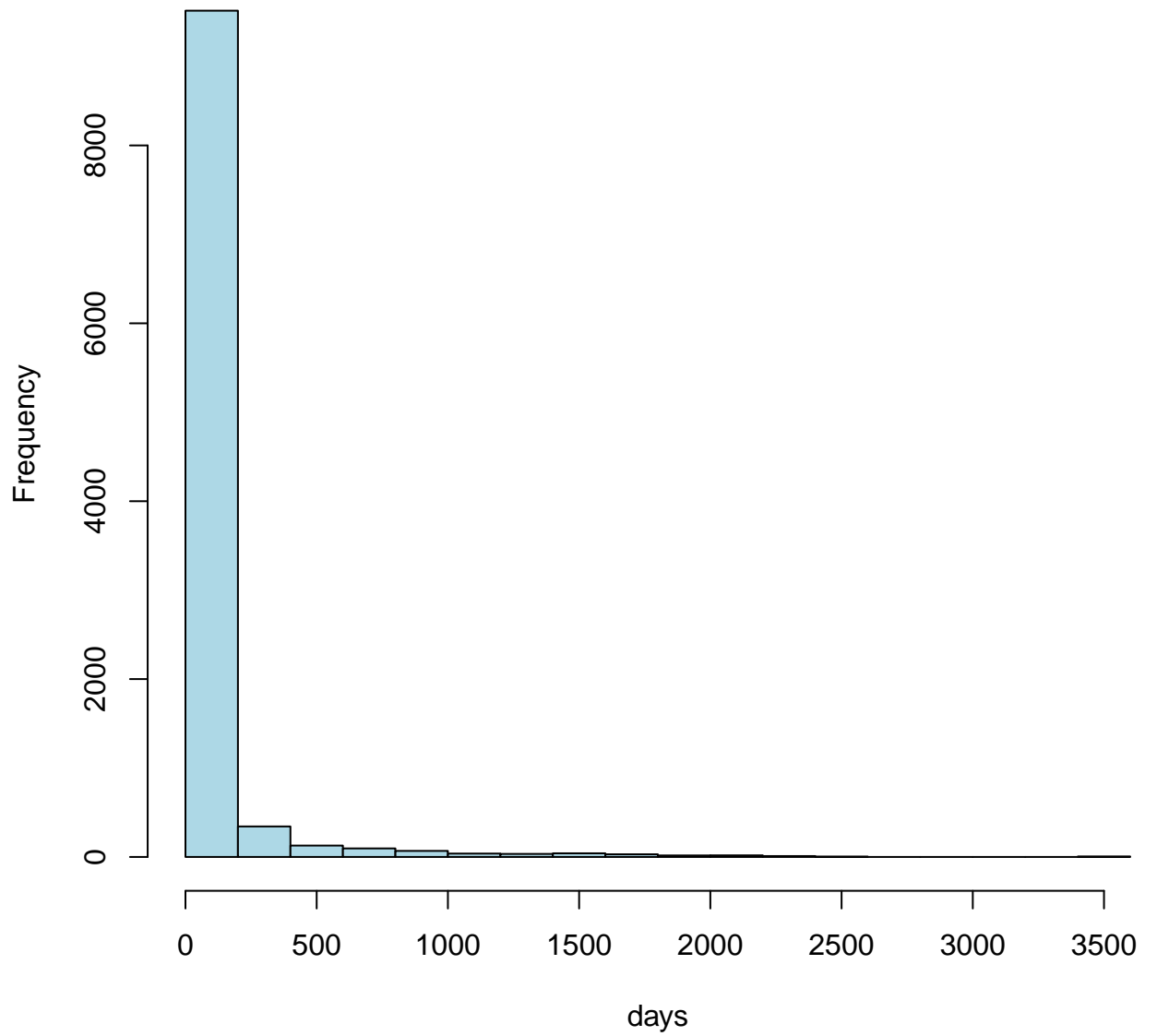
Determining the deltas between tweet age and link age

I wrote a script called ‘addDatesToUri.py’ that concatenated each URI and the date found to be earliest by the carbon date utility. There were several URIs that I came across that had no dates in the response from carbon date and were skipped. The results were placed in a file called ‘uniqueuridates’. The file ended up having 4345 entries with a date given from carbon date out of 7406 unique URIs.

To compute the difference in the ages I wrote a script named ‘timedif.py’. This script starts by pulling in the information in ‘uniqueuridates’. After that it begins to work through the tweetsURI file line by line. If it sees a line that starts with CREATED_AT: it changes the format of the date and saves it as tweetdate. If it sees a line that starts with URL or LOCATION, it saves the result in link. Note this only saves the last one found before the end of the headers. Once the script reaches the end of the headers for the twitter link it searches for the final link in the list of known URIs. If it is there we perform a time difference between the tweetdate and urldate. Else we make the time difference 0 saying twitter was the first known time instance of the link.

Histogram:

Time difference between tweet and URL creation



Statistics on Deltas

I wrote a python program to perform my statistics operations named time/stats.py. It returned the following information on my deltas.

- SUM OF ALL DELTAS: 685375
- NUMBER OF DELTAS: 10347
- MEAN OF ALL DELTAS: 66.23900647530685
- MEDIAN OF ALL DELTAS: 0
- STD ERR OF ALL DELTAS: 2.543419215641349
- STD DEV OF ALL DELTAS: 258.71712270023767