Webscraping with Bash

Rustie Lin

December 3, 2017

Introduction

In my previous post, titled *UNIX and Bash Fundamentals*, I went over how to pair both UNIX and Bash tools with R to expand one's toolbox to become a more flexible data analyst. In that post, we mainly went over how to run data preprocessing locally, on already pre-downloaded files. This post will continue with the knowledge we learned in the previous post, and extend our operability to finessing online data. In other words, we will be learning how to web scrape with Bash tools, and how to feed this data into R for more powerful data processing.

The reason why we might want to use Bash tools for web scraping instead of R tools such as *rvest* is because Bash is much more prevalent that full installs of R (and R Studio). While every computer might not have R and R Studio installed, they probably have access to a Bash shell, or at least a Bash emulator. It is also important to understand proper data flow between Bash commands, especially since they are the building blocks to much of the advanced functionality that R and R libraries for web scraping are built off of.

Fetching the Necessary Dependencies

For this demo, we will be using standard Bash tools in conjunction with the web tools *cURL* and *lynx*. I will go over what each of these commands does in the next sections, but for now, you can download them using your favorite package manager. I personally use the package manager *apt*, so I run the following commands:

```
# installing curl and lynx
apt-get install curl lynx
```

Software installation might require super-user access on your system, in which case you can simply append the command *sudo* to the beginning of the above command. Check your computer to see which package manager you have installed if you are unaware. Mac users can install *brew*, and Windows users with the Linux subshell enabled can use *apt* by default, or any other installed package manager. It is also possible to install *cURL* and *lynx* from source, instructions for which are out of the scope of this post, but documentation for which exists in excess online thanks to the wonders of open source software.

cURL: transfer a URL

From the man pages:

```
# man curl
curl is a tool to transfer data from or to a server, using one of the supported pro-
tocols (DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS,  POP3,
POP3S,  RTMP, RTSP, SCP, SFTP, SMB, SMBS, SMTP, SMTPS, TELNET and TFTP). The command
is designed to work without user interaction.
```

Basically what this means is that it lets us download files and web pages in our case.

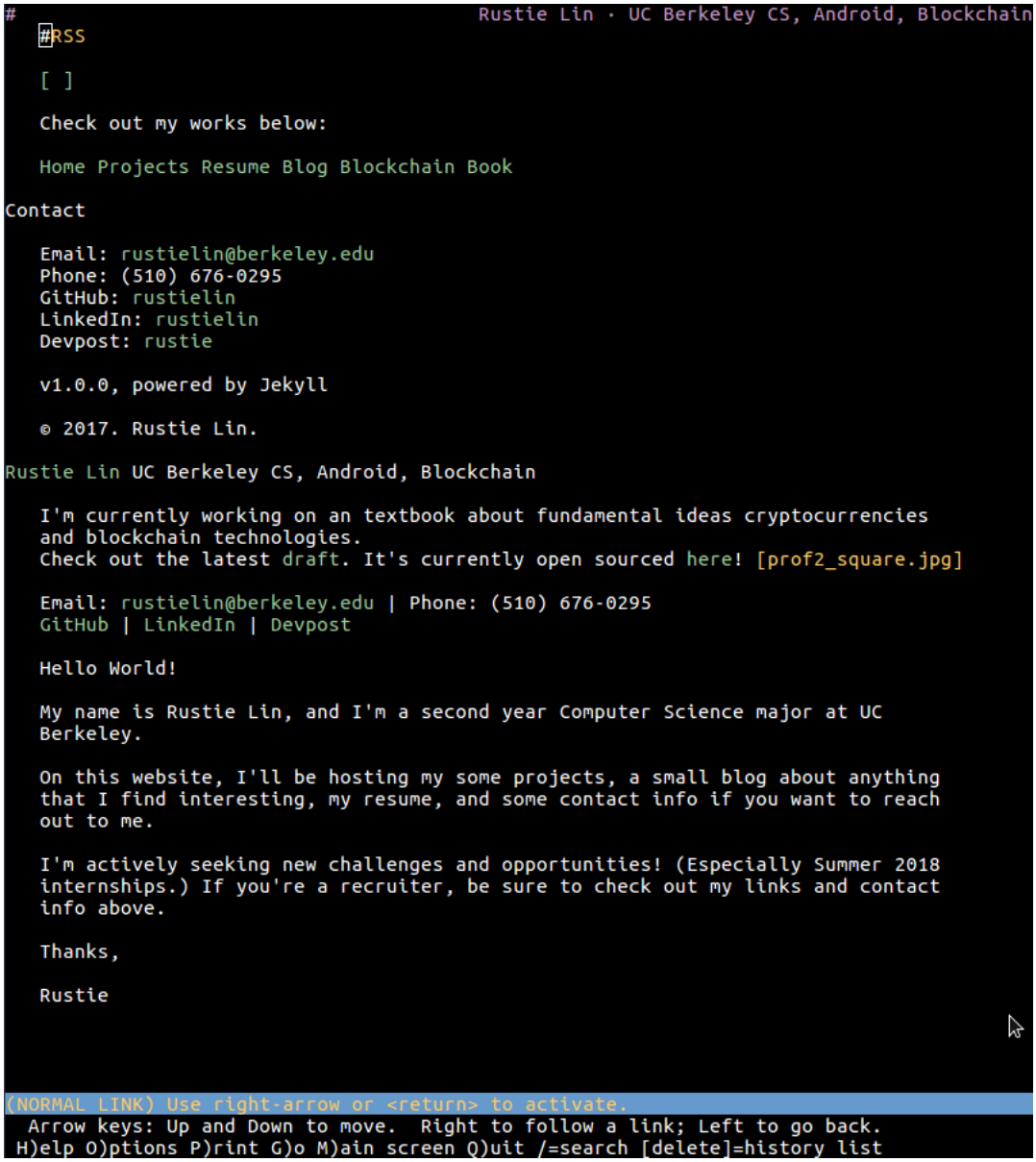
Let's try it out. To download the page http://rustielin.me, simply execute the following command.

```
# save to a file
curl http://rustielin.me > data/rustie.html

# see what's in the html
head data/rustie.html
```

```
##   % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
##                                  Dload  Upload   Total   Spent    Left  Speed
##
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100  5136  100  5136    0     0  16361      0 --:--:-- --:--:-- --:--:-- 16408
## <!DOCTYPE html>
## <html lang="en-us">
##
##   <head style="position: fixed">
##   <link href="http://gmpg.org/xfn/11" rel="profile">
##   <meta http-equiv="X-UA-Compatible" content="IE=edge">
##   <meta http-equiv="content-type" content="text/html; charset=utf-8">
##
##   <!-- Enable responsiveness on mobile devices-->
##   <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1">
```

lynx: text-based web browser



lynx

I mean the heading says it all. *lynx* eats html files and processes them in human-readable formats, and this will come in really handy when we actually start web scraping. *lynx* is pretty powerful, and you can actually use it to view and interact with web pages (seen in the image above), but we'll only be using it to get a dump of the web page we're interested in. To get an idea of what we'll be using *lynx* for, check out the following commands, where we make our previously downloaded html file a bit more readable.

```
# dump lynx output to a file
lynx -dump data/rustie.html > data/rustie.txt

# see what's in the txt
cat data/rustie.txt
```

```
##    #[1]RSS
##
##    [ ]
##
##    Check out my works below:
##
##    [2]Home [3]Projects [4]Resume [5]Blog [6]Blockchain Book
##
## Contact
##
##    Email: [7]rustielin@berkeley.edu
##    Phone: (510) 676-0295
##    GitHub: [8]rustielin
##    LinkedIn: [9]rustielin
##    Devpost: [10]rustie
##
##    v1.0.0, powered by Jekyll
##
##    © 2017. Rustie Lin.
##
## [11]Rustie Lin UC Berkeley CS, Android, Blockchain
##
##    I'm currently working on an textbook about fundamental ideas
##    cryptocurrencies and blockchain technologies.
##    Check out the latest [12]draft. It's currently open sourced [13]here!
##    [prof2_square.jpg]
##
##    Email: [14]rustielin@berkeley.edu | Phone: (510) 676-0295
##    [15]GitHub | [16]LinkedIn | [17]Devpost
##
##    Hello World!
##
##    My name is Rustie Lin, and I'm a second year Computer Science major at
##    UC Berkeley.
##
##    On this website, I'll be hosting my some projects, a small blog about
##    anything that I find interesting, my resume, and some contact info if
##    you want to reach out to me.
##
##    I'm actively seeking new challenges and opportunities! (Especially
##    Summer 2018 internships.) If you're a recruiter, be sure to check out
##    my links and contact info above.
##
##    Thanks,
##
##    Rustie
##
## References
##
##    1. file:///atom.xml
##    2. file:///
##    3. file:///projects/
##    4. https://drive.google.com/file/d/0B46VW36KYd_NcDZUbmJEVWFmRm8/view
##    5. file:///blog
##    6. file:///blockchain_textbook.pdf
##    7. mailto:rustielin@berkeley.edumailto:rustielin@berkeley.edu
##    8. https://github.com/rustielin
##    9. https://www.linkedin.com/in/rustielin/
##   10. https://devpost.com/rustie
##   11. file:///
##   12. file:///blockchain_textbook.pdf
##   13. https://github.com/rustielin/Blockchain-Notes
##   14. mailto:rustielin@berkeley.edumailto:rustielin@berkeley.edu
##   15. https://github.com/rustielin
##   16. https://www.linkedin.com/in/rustielin/
##   17. https://devpost.com/rustie
```

Note especially how *lynx* collects all the links in the webpage at the very bottom in a section titled *References*! See where we're going with this?

Actually doing the web scraping

Alright so now we know how to download a web page using *cURL*, and how to use *lynx* to process the web page into something easy to read, and thus easy to scrape. In this section, we'll be scraping from the following website, which has some sample CSV files: https://support.spatialkey.com/spatialkey-sample-csv-data/

(We are using a sample web page here, but imagine using the power of Bash scrpting to recursively download all the CSV (or any other format) files from a given domain.)

```
# download the web page with cURL
curl -s https://support.spatialkey.com/spatialkey-sample-csv-data/ -o data/sample.html

# process the web page with lynx
lynx -dump data/sample.html > data/sample.txt
```

We only really care about the CSV files hosted on this page, and not the rest of the page. We can extract the links to the CSV files using the *grep* string search command, which has support for Regular Expressions!

```
# get the urls
cat data/sample.txt | grep -e '\.csv$'
```

```
##   13. http://samplecsvs.s3.amazonaws.com/Sacramentorealestatetransactions.csv
##   15. http://samplecsvs.s3.amazonaws.com/SalesJan2009.csv
##   16. http://samplecsvs.s3.amazonaws.com/TechCrunchcontinentalUSA.csv
##   18. http://samplecsvs.s3.amazonaws.com/SacramentocrimeJanuary2006.csv
```

Nice, we got the URLs. But they have numbers in front of those. We can clean our URLs up using the *cut* command, whose functionality is self explanatory. We're interested in each line of the previous output, starting from the 7th character onwards.

```
# same command as before, piped into cut, and saved to a file
cat data/sample.txt | grep -e '\.csv$' | cut -c 7- > data/urls.txt

# see what we got
cat data/urls.txt
```

```
## http://samplecsvs.s3.amazonaws.com/Sacramentorealestatetransactions.csv
## http://samplecsvs.s3.amazonaws.com/SalesJan2009.csv
## http://samplecsvs.s3.amazonaws.com/TechCrunchcontinentalUSA.csv
## http://samplecsvs.s3.amazonaws.com/SacramentocrimeJanuary2006.csv
```

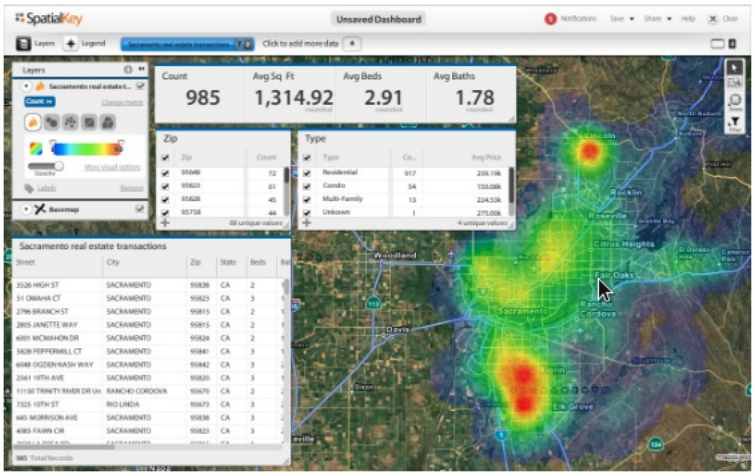
Awesome. Nowe we have to get the CSVs from these URLs somehow. *cURL* lets us download these files. We'll only be downloading from the first URL for simplicity.

```
# get the first url and download with crl
head -n 1 data/urls.txt | xargs curl -so data/realestate.csv
```

For reference, here is the description of the dataset we just downloaded. It's a screenshot of the website as viewed from a modern web browser (Google Chrome), but you could have gotten the same information from the output of *lynx*, which was saved into the file `data/sample.txt`.

Real estate transactions (download .csv file)

The Sacramento real estate transactions file is a list of 985 real estate transactions in the Sacramento area reported over a five-day period, as reported by the Sacramento Bee. Note that this file has address level information that you can choose to geocode, or you can use the existing latitude/longitude in the file.



real estate data

Now we have a workable CSV! Time to import into R for some fun. We'll graph this with ggplot.

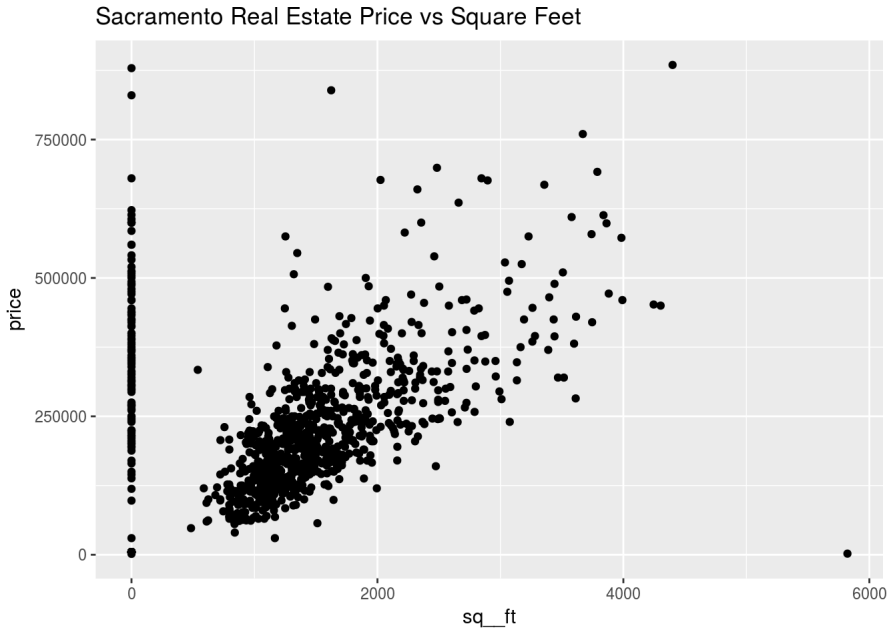
```
library(ggplot2)

# importing the CSV to show that we actually got a valid CSV
dat <- read.csv("data/realestate.csv")

# see what we got
colnames(dat)
```

```
##  [1] "street"    "city"      "zip"       "state"     "beds"
##  [6] "baths"     "sq__ft"    "type"      "sale_date" "price"
## [11] "latitude"  "longitude"
```

```
# a sample plot
ggplot(dat, aes(x = sq__ft, y  = price)) +
  ggtitle("Sacramento Real Estate Price vs Square Feet") + geom_point()
```



Remarks

I used *lynx* in this demo because it's an easy way to visualize web pages. It might not be standard in some software repositories, and may be hard to get one's hands on. Instead of using *lynx*, one could just use *grep* and *sed* to get CSV files. This method requiries a bit more mastery of Regular Expressions, but it's still doable. *grep* and *sed* are much more standard than *lynx*, and are preintalled on most modern UNIX and GNU/Linux based operating sytetms.

```
# get the urls
cat data/sample.html | grep -oe 'http.*\.csv"' | sed 's/\"$//g'
```

```
## http://samplecsvs.s3.amazonaws.com/Sacramentorealestatetransactions.csv
## http://samplecsvs.s3.amazonaws.com/SalesJan2009.csv
## http://samplecsvs.s3.amazonaws.com/TechCrunchcontinentalUSA.csv
## http://samplecsvs.s3.amazonaws.com/SacramentocrimeJanuary2006.csv
```

Conclusion

In this post, we learned how to use some Bash tools to web scrape CSV files to pipe into R. While we could have just used R libraries to web scrape, using Bash still has its merits. Firstly, the person web scraping and the person running the data analysis could be different. Imagine a team workflow where I'm tasked with getting the data for my teammate to run analysis on. My computer might not have R installed, etc. Bash is also faster than R in terms of sketching together a demo or proof-of-concepts. Finally, Bash and UNIX are tools that everyone should have at least some experience with, as I explained in my previous post (*UNIX and Bash Fundamentals*) I alluded to in the introduction.

Hopefully this post has inspired you the reader to play around with Bash tools yourself, and to expand your data analysis/web scrapiing tool-set. By mastering the Bash shell, UNIX, as well as R and other tools, one can become a very effective and flexible data analyst.

References

- https://support.spatialkey.com/spatialkey-sample-csv-data/
- My previous post titled *UNIX and Bash Fundamentals*
- Lecture material from Stat133