thoughtfulbloke aka David Hood

Getting and Cleaning week 2

Another week where you are learning how your own OS and network affect getting various kinds of data, but if you are checking the results of each step as you go, you should be fine.

- Part 1 : General Advice (https://thoughtfulbloke.wordpress.com/2015/08/31/hello-world/)
- Part 2 : Getting and Cleaning Week 1 (https://thoughtfulbloke.wordpress.com/2015/09/09/getting-and-cleaning-week-1/)
- Part 3 : Getting and Cleaning Week 2 (https://thoughtfulbloke.wordpress.com/2015/09/09/getting-and-cleaning-week-2/)
- Part 4 : Getting and Cleaning Week 3 (https://thoughtfulbloke.wordpress.com/2015/09/09/getting-and-cleaning-week-3/)
- Part 5 : Getting and Cleaning Week 4 (https://thoughtfulbloke.wordpress.com/2015/09/09/getting-and-cleaning-week-4/)
- Part 6 : Getting and Cleaning the Assignment (https://thoughtfulbloke.wordpress.com/2015/09/09/getting-and-cleaning-the-assignment/)

Lectures

You do not need to install and configure MySQL to complete the course. There are no quiz questions that use it. Also, since the lectures were recorded RMySQL now comes with its own drive for Windows and Mac so does not need the installs.

Google Scholar has changed its web page generation since the lecture was recorded, so the Reading from the Web lecture example cannot directly be copied. Just think of it as illustrating a general approach.

Quiz 2 advice

Question 1

My interpretation of Quiz 2 Question 1 is the learning objective of practicing the fairly common situation of "I need to get access to a data resource. I've googled and found some example code. And it does something similar to what I want to do, but not quite, so I need to work out what is going on and make changes to it" (this is our chance to practice these kinds of situations under controlled conditions where there is a known answer so we know if we actually did it OK). In this case the question is assuming that we googled and found the example code for getting the rate limit information and we want to adapt that code to bringing back jtleek's repo information, then understand the JSON that we brought back. But this is the first time a lot of people have done something like this so find it hard (mind you, this is pretty representative of APIs. They are that horrible).

I will acknowledge you can get the answer without doing everything the question specifies. Like a number of other questions in the course (like the dealing with Excel question) the intent is to make you actually do something that is a useful skill for data scientists rather than just watch it in a lecture. In the case of github, the data is available from other means (github limits the rate at which non-authorised apps can obtain information more than what information is available), but because people already are using github for this course there is no added privacy burden of getting a github developer account.

If it helps, this would be the way I would tell the story of doing this question:

- We are using the httr library, so there is the option of checking the documentation.
- We need to register for making an app (an application) this application is the script we will be using to get priviledged access to github. The web address in Hadley Wickam's example code, http://localhost:1410 (http://localhost:1410/) (don't put any added spaces into the form), is the web address we use as that means the script running in r will receive the data. Application Name- in theory doesn't matter so long as you use the same application name in your code, so for the example code to work it would be github. Homepage URL- really doesn't matter. Description- totally up to you
- We need the up to date httr library, so don't forget to library/require it. And probably the httpuv library to.
- we basically reproduce Hadley Wickam's example code, since it is part of the information given for the question I am happy to reproduce it

myapp <- oauth_app("github",key="add_your_registered_key",secret = "add_your_secret") github_token <- oauth2.0_token(oauth_endpoints("github"), myapp) gtoken <- config(token = github_token)

- Now we need to vary the code a little, as we want to use our privileged developer access on Jeff Leeks repository, so the req line will still be configured with our github token, it is just that the page we req (request) is different
- storing the content() of the req, we find it is a bit messy, so we might want a fromJSON(toJSON(messy_stuff)) to wind up with a fairly clean data frame we can query for the question result.

Running the script and the authentication step

Some people, when running the script, see the message

```
Waiting for authentication in browser...
Press Esc/Ctrl + C to abort
```

Then things stall. In a previous session, Nathan Moser gave a good write-up of what should be happening here which I a borrowing from for here:

The reason listing "http://localhost:1410 (http://localhost:1410/)" as your callback URL in your GitHub application is crucial, is that when you execute the oauth2.0_token() function, the R runtime

1. opens network port 1410 and listens on it, then
2. opens your default browser on your system and opens the oauth authorize endpoint URL in your brower, so that
3. once you log in successfully, your browser is redirected to open http://localhost:1410 (http://localhost:1410/) on your host, and then
4. the success/fail authentication and authorization result is communicated over the 'localhost' network on your host to the R runtimeYou can verify this using the 'netstat' command (available on MacOS terminal, Windows cmd.exe, and Linux command lines); doing a "netstat -a | grep LISTEN" while the R command window is displaying

Waiting for authentication in browser…

You should see an entry that looks similar to

```
TCP 127.0.0.1:1410 nate-pc:0 LISTENING
```

This port will be closed when the oauth2.0_token() returns with a success/fail message.

If people are stalling at the browser window opening, this could be because the callback URL isn't set when you registered the application, or it could be a local firewall thing blocking the return message, or it could be a web browser extension conflicting, or it could be web browser not being very cooperative. This last one is actually the easiest to check/fix. The process should open a new web browser window, and if the browser isn't cooperating about that (possibly because there is already a window under that technical name open from an earlier attempt), one trick can be to close web browser before running the commands. This should cause your default browser to open with a new window to do the authentication with.

Think of this question a being a practical exercise as dealing with the issues like API enrolment so that people can practice under controlled conditions and ask questions if they are stuck and being able to check what they are getting along the way.

The “I have some general advice and some semi-related code from the web” is a very common situation, and the exercise gives the chance to pick through getting practice at working out how parts fit together. from previous sessions, having had experience with this one there are people who wanted to give a go analysing Facebook, or Twitter, or Flickr, all of which really need the api process to do anything useful (if people are wanting to discuss these kinds of things, I suggest starting a new thread in the General area rather than putting the discussion in this question specific thread).

The quizzes in the course are mostly the equivalent of practical example homework exercises rather than in class tests of lecture knowledge.

As well as the above some people get a bit lost investigating the returned JSON so I will talk through interrogating data with str() and subsetting() kind of things.

If I had example code like Hadley’s, I might well read the package documentation if I wasn’t familiar with it (by googling cran httr, I can find the page on cran and there is both the Reference Manual and the Quick Start Guide(I could read up in the help also, but I like reading through all the options in an old fashioned kind of document way). Either way I get to the point where I have made a req and used content to get the processed contents, for the sake of argument stored in raw_result. So I check it out

```
str(raw_result)
```

and that doesn’t work very well, mostly because json comes through as a repeating list of json elements. Possible I just look up on the internet what format github sends (though I am used enough to seeing json I recognise it) or otherwise check it looks like json (I might for example have read up on how the GET command generates a response object, so I go back and check it out with str(), in there I can see that headers – content-type is json, and utf-8 character set, if I want to print it out by itself because of the hypen I do need to use quote marks so might go req$headers$”content-type” but I can see it in str() so don’t really need it separate). Either way, I am confident I am dealing with json at this point.

As I side note, if you are using an R package regularly, I strongly suggest actually reading the documentation and vignettes for seeing how it works and what you can do.

Anyway I need something like the jsonlite package at this point and I might try

```
> jsoned <- fromJSON(raw_result)
Error: Argument 'txt' must be a JSON string, URL or path to existing file.
```

Clearly that didn't work, so I need to make it better configured JSON for R, so use toJSON(). Anyone who stops to check what this has produced with str() will see it has turned it from a fairly complex list of lists to something that is formally of class "json" and so can be used by fromJSON. Now, if I had stored the results of fromJSON, so I have got something to work with, and if I have looked at the help for fromJSON **where the examples at the bottom basically given me the solution to how to approach this question** I can use names(thingIStoredItIn) to know how to reference parts. then I do a simple subset using == to say give me the created_at entry equivalent to the entry where the name is data sharing.

401 errors

401 is pretty much has to be a problem with the secret, typing out the details and mistyping it, missing characters when copying and pasting, or forgetting to put in the quote marks around them. A good starting point is pasting the secret directly in your code, so that it is right there for debugging purposes. You will probably also want to add cache=FALSE to the github token step to clear away the previous problem settings when you fix this..

500 errors

Error 500 is normally a form of authentication error- people in the past have triggered it by not filling out the setup form on github properly (in particular for example having extra spaces, often after the http://localhost:1410 (http://localhost:1410/) in the callback. Going through the creation process again and being really careful about what they are putting into the creation form and copied into the script have normally fixed it for those people.

website not found

Just in last session (it hasn't happened before) people were finding that even if they did everything else correct (had the callback set to localhost, loaded httpuv to listen) they were getting an error (this seems to be on Windows. The culprit seems to be that httpuv depends on Rcpp, and a recent change in Windows means that if your Rcpp library is older than version 11.5 it needs to be updated.

You can get yourself in a tangle here- if you try and update a package that is actively in use by R (for example loaded with a library command), it will not be able to remove all of the old package (because it is using it), but it will remove what it can (so you can't use the old package). Because it can't get rid of the old package, you get an error and can't install the new package (so you can't use the new package). To avoid this happening, update it from R when the package is definitely not loaded:

- In RStudio go into Tools – Global Options – General Tab, and take the tick out of the Load .RData at startup option. By having this unticked, it means that when we restart RStudio it starts fresh rather than reloads things.
- Restart RStudio, then do `install.packages("Rcpp")` before running anything that uses the package.

tackling it in stages

You can approach the problem in steps- If you can get Hadley's code working to get the rate limit via the API, then you have a win you can build on and start modifying the code to get the data we want to answer this quiz question.

If you are reading this after getting in a tangle, we can have a discussion about what to do down in the comments. But please include actual details about your computer system, versions of R and packages, and what you have actually tried.

Question 2 and 3

These questions use sqldf not RMySQL. One technical issue is that if you did install MySQL and load RMySQL, by default sqldf is going "oh, you have MySQL active, you must want me to talk to MySQL databases" You can explicitly set it to talk to data frames by setting the driver, but if you aren't using MySQL at the moment, you can take the tick out of the loaded library tick box in the packages pane of R then when you load sqldf it will just use data frames. To formally set the driver, when using sqldf use the drv='SQLite' option.

Mac Specfic issue sqdf has an external dependency on xWindows, so to get it working on recent versions of OSX, you may need to install a x-windows environment installed, you can get one by going to http://xquartz.macosforge.org/landing (http://xquartz.macosforge.org/landing) and getting the XQuartz dmg (restart the machine after installing).

Another way of dealing with tcltk problems is

```
options(gsubfn.engine = "R") # as per R FAQ #5 use R code rather than tcltk
```

Question 4

Because people in previous courses have had trouble with finding information between lots of different threads on particular topics, I am creating some initial threads for particular quiz questions to give people an obvious place to look for help.

For this question, pay attention to the Reading from the Web lecture. There is more than one way of reading in a web-page covered in that lecture, and each suits different circumstances. One way talked about in the lecture was reading individual lines, the other way was the structure. This question is not reading from the structure.

Once every few hundred thousand students, someone has a virus or intrusive Interent Service Provider that is adding tracking code to webpages they are retrieving.

The first nine lines, if your web page traffic is not being interfered with, are:

```
> htmlCode[1:9]
[1] "<!DOCTYPE html PUBLIC \"-//W3C//DTD XHTML 1.0 Strict//EN\" \"http://www
[8] "<meta name=\"Keywords\" content=\"Johns Hopkins University, Bloomberg S
[9] "<meta http-equiv=\"Content-Type\" content=\"text/html; charset=iso-8859
```

Question 5

My interpretation of Quiz 2 Question 5 is that it is one of these programming assignment kind of questions where it doesn't matter exactly how your get to the end, so you are using this Hacking Skills mentioned in the Toolbox prerequisite course to look things up. This is not a quiz question to watch a lecture and a repeat the steps in a quiz, this is much more a homework assignment where you are getting familiar with what is involved with dealing with data where you have some hints about the file, so need to find information about how to read it in (except that being a quiz question you will know if you have done it correctly, unlike the real world where you need to be the one checking your work).

If you visually inspect the data (and you might use a text editor, or a spreadsheet program, or just read individual lines from the file into R), and I strongly suggest you do inspect data when trying to read it in, you should see why your can't use commands like read.table: On line 6 of the file Nino34 SST is separate from Nino34 SSTA by a space (which would work with read.table) but on line 7 the place where that space is has a negative sign there (which means there is no gap for read.table to work). You might want to use your hacking skill to search out how to read a fixed width file into R.

Another thing, this question is using a cached copy of the data on coursera (cloudfront link) and the questions also cites the original data source for people who want to go and investigate it further. Do not try and use the original data source to answer the question, it is being updated with real data so the answer changes. Use the cloudfront link, not the citation.

And, thought people find it hard to believe, yes you do get files like this, and yes there is no better plan than using a text editor that can tell you how far across the line characters are (I suggest professional level text editors, such as the freely downloadable Notepad++ for Windows (not the same as notepad), TextWrangler for the Mac, or GEdit for Linux, but there are a lot of other good ones as well.

In reality most people google to figure out the appropriate function for reading in a particular kind of data (and that works fine, and is a perfectly reasonable strategy when in this situation in the future) I thought it worth reminding people that while

```
?whatever
```

is a search for a function called whatever

```
??whatever
```

is a search of the text of all the help files for the word whatever. If you have some kind of distinctive word describe the data you are wanting to load in, you could search for that term through all the help pages, and if you use quote marks, you can look for an exact phrase

```
??"whatever secondterm"
```

But googling is fine.

widths

One thing people can get a bit stuck on, after they have researched (aka googled) reading fixed width files in R, is that there is a widths arguement to the command and people can find the description a bit opaque.

Now we can't discuss the widths to use in the question file (assuming we were skipping the non data lines at the top as part of the read in) but it is certainly fine to discuss an example file unrelated to the quiz question (a test case for using the function), and working through an example where you can see what is going on is an excellent way of figuring it out in this case. And if we create a new text file, called "fix.txt" that only contains these three lines;

```
1234567890
1234567890
12345 7890
```

We could see and discuss how various width settings might work on this file. Assuming it is in the working directory, have a think (by example) about what these width options are doing:

```
("fix.txt", widths=c(2,2,2,2,2))
("fix.txt", widths=c(2,2,2))
("fix.txt", widths=c(5,5))
("fix.txt", widths=c(-3,2,-3,2))
("fix.txt", widths=c(3,4,3))
```

I will also note that all the columns came through as numbers, except the last one where column V2 was text as it could not be interpreted as a number. So scrutinising this we can see that we are providing the groups of characters that R will turn into columns, that negative numbers are used to show how many characters to skip, and that R will trim extra whitespace from the left and right, but not the middle.

September 9, 2015 thoughtfulbloke Coursera, Getting and Cleaning, R