Stat243: Problem Set 1, Due Wednesday Sept. 11

August 31, 2019

Comments:

- This covers material in Units 2 and 3 as well as practice with some of the tools we'll use in the course (knitr/R Markdown).
- It's due at 2 pm on September 11, **both submitted as a PDF to bCourses as well as committed to your Github repository** as documented in the *howtos/submitting-electronically.txt* file.
- Please note my comments in the syllabus about when to ask for help and about working together. In particular, **please give the names of any other students that you worked with on the problem set and indicate in comments any ideas or code you borrowed from another student.**

Formatting requirements

1. Your electronic solution should be in the form of an R markdown file named *ps1.Rmd* or a LaTeX+knitr file named *ps1.Rtex*, with bash code chunks included in the file (or read in from a separate code file). Please see the *dynamic documents* tutorial or Lab 1 materials for more information on how to do this.
2. Your PDF submission should be the PDF produced from your Rmd/Rtex. Your Github submission should include the Rtex/Rmd file, any R code files containing chunks that you read into your Rtex/Rmd file, and the final PDF, all named according to the guidelines in *howtos/submitting-electronically.txt*.
3. Your solution should not just be code - you should have text describing how you approached the problem and what the various steps were.
4. Your code should have comments indicating what each function or block of code does, and for any lines of code or code constructs that may be hard to understand, a comment indicating what that code does. You do not need to show exhaustive output but in general you should show short examples of what your code does to demonstrate its functionality.

Problems

1. As preparation for the next few units, please read the sections titled "Memory hierarchy" and "Cache in depth" in the following brief piece that talks about the difference between the CPU cache, main memory (RAM) and disk. You don't need to follow all the technical details in the "Cache in depth" section - just try to get the big picture of what the cache is and a bit about how it works.

2. Please read Unit 3 on good programming/project practices and incorporate what you've learned from that reading into your solution for Problem 4. As your response to this question, briefly (a few sentences) note what you did in your code that reflects the material in Sections 1.2 and 1.3 of Unit 3. Please also note anything in Unit 3 that you disagree with, if you have a different stylistic perspective.

3. This problem explores file sizes and their relationship to the file format. The *system()* function in R calls out to the command line and in this case executes "ls -l", which (among other output) shows the file size in bytes as the 5th element of information.

   (a) Based on understanding storage in ASCII plain text versus binary formats and on the fact that numbers are (usually) stored as 8 bytes per number in binary format, explain the results below. If there are any that seem inconsistent with the class notes and our discussion in class, note that.

```
## save letters in text format
chars <- sample(letters, 1e6, replace = TRUE)
write.table(chars, file = 'tmp1.csv', row.names = FALSE,
            quote = FALSE, col.names = FALSE)
system('ls -l tmp1.csv', intern = TRUE)

## [1] "-rw-r--r-- 1 paciorek scfstaff 2000000 Aug 31 09:44 tmp1.csv"

chars <- paste(chars, collapse = '')
write.table(chars, file = 'tmp2.csv', row.names = FALSE,
            quote = FALSE, col.names = FALSE)
system('ls -l tmp2.csv', intern = TRUE)

## [1] "-rw-r--r-- 1 paciorek scfstaff 1000001 Aug 31 09:44 tmp2.csv"

## save in binary format
nums <- rnorm(1e6)
save(nums, file = 'tmp3.Rda')
system('ls -l tmp3.Rda', intern = TRUE)

## [1] "-rw-r--r-- 1 paciorek scfstaff 7678419 Aug 31 09:44 tmp3.Rda"

## save in text format
write.table(nums, file = 'tmp4.csv', row.names = FALSE,
            quote = FALSE, col.names = FALSE, sep = ',')
system('ls -l tmp4.csv', intern = TRUE)

## [1] "-rw-r--r-- 1 paciorek scfstaff 18159771 Aug 31 09:44 tmp4.csv"

write.table(round(nums, 2), file = 'tmp5.csv',
            row.names = FALSE, quote = FALSE,
            col.names = FALSE, sep = ',')
system('ls -l tmp5.csv', intern = TRUE)

## [1] "-rw-r--r-- 1 paciorek scfstaff 5377299 Aug 31 09:44 tmp5.csv"
```

   (b) Now consider the additional results below. Can you explain what is going on and understand what R is doing when it saves an object using *save()*? You might experiment with options to the

save() function to determine if your explanations are correct.

```
chars <- sample(letters, 1e6, replace = TRUE)
chars <- paste(chars, collapse = '')
save(chars, file = 'tmp6.Rda')
system('ls -l tmp6.Rda', intern = TRUE)

## [1] "-rw-r--r-- 1 paciorek scfstaff 635277 Aug 31 09:44 tmp6.Rda"

chars <- rep('a', 1e6)
chars <- paste(chars, collapse = '')
save(chars, file = 'tmp7.Rda')
system('ls -l tmp7.Rda', intern = TRUE)

## [1] "-rw-r--r-- 1 paciorek scfstaff 1066 Aug 31 09:44 tmp7.Rda"
```

4. Go to https://scholar.google.com and enter the name (including first name to help with disambiguation) for a researcher whose work interests you. (If you want to do the one that will match the problem set solutions, you can use "Robert Tibshirani", who is a well-known statistician and machine learning researcher.) If you've entered the name of a researcher that Google Scholar recognizes as having a Google Scholar profile, you should see that the first item returned is a "User profile". Next, if you click on the hyperlink for the researcher under "User profiles for <researcher name>", you'll see that brings you to a page that provides the citations for all of the researcher's papers.
IMPORTANT: if you repeatedly query the Google Scholar site too quickly, Google will start returning "503" errors because it detects automated usage (see problem 5 below). So, if you are going to run code from a script such that multiple queries would get done in quick succession, please put something like `Sys.sleep(2)` in between the calls that do the HTTP requests. Also when developing your code, once you have the code in part (a) working to download the HTML, use the downloaded HTML to develop the remainder of your code and don't keep re-downloading the HTML as you work on the remainder of the code.

   (a) Now, based on the information returned by your work above, including the various URLs that your searching and clicking generated, write R code that will programmatically return the citation page for your researcher. Specifically, write a function whose input is the character string of the name of the researcher and whose output is the HTML (as an object of class 'xml_document') corresponding to the researcher's citation page as well as the researcher's Google Scholar ID.

   Hint: you will need to use some string processing functions to extract the Scholar ID (and possibly for other things) from the output of the various *rvest* functions. I recommend functions from the *stringr* package (we'll see these in Unit 5 and you can find information in Section 2.1.2 of the *string processing* tutorial), in particular: *str_detect()*, *str_extract()*, *str_split()*, and *str_replace()*. You should NOT need to use regular expressions (which we'll cover in Unit 5), but you can if you want/know how to. Finally if you get an error message like this "`Syntax error in regexp pattern. (U_REGEX_RULE_SYNTAX)`", it means the string processing function is interpreting the characters you are looking for as a regular expression. Simply wrap the string you are looking for in the *fixed()* function, e.g., "`fixed('sdf?=sd34')`" so the characters are simply interpreted as regular characters (i.e., interpreted literally).

(b) Create a second function to process the resulting HTML to create an R data frame that contains the article title, authors, journal information, year of publication, and number of citations as five columns of information. Try your function on a second researcher to provide more confidence that your function is working properly.

(c) Include checks in your code so that it fails gracefully if the user provides invalid input or Google Scholar doesn't return a result. You don't have to use the *assertthat* package as we won't cover that until the Lab on Tuesday September 10, but you can if you want.

(d) (Extra credit) Fix your function so that you get all of the results for a researcher and not just the first 20. Hint: figure out what query is made when you click on "Show More" at the bottom of the page.

Hint: as you are trying to understand the structure of the HTML, one option is to use *write_xml()* on the result of *read_html()* and then view the file that is produced in a text editor.

Note: For simplicity you can either assume only one User Profile will be returned by your initial search or that you can just use the first of the profiles.

5. Look at the *robots.txt* file for Google Scholar and the references in Unit 2 on the ethics of webscraping. Write a paragraph or two discussing the ethics of scraping data from Google Scholar as we do in Problem 4. Do you have any concerns in terms of whether anything we are doing in Problem 4 might violate Google's rules or the general ethical considerations in the references? (I think what we are doing is in the spirit of ethical webscraping, or I wouldn't have assigned it, but there is a grey zone here, and I'd like you to think about the ethical issues within a specific context.)

6. This problem is actually just the formatting of this document.

   (a) Have the document be correctly formatted according to the requirements above. You don't need to explicitly answer this sub-problem.

   (b) (extra credit) The *reticulate* package and R Markdown allow you now to have Python and R chunks in a document that interact with each other. Demonstrate the ability to use this functionality, in particular sending data from R to Python and back to R, with some processing done in Python (it doesn't have to be complicated processing). There's a blog post here to get you started: http://feedproxy.google.com/~r/RBloggers/~3/76l-uQEOoiU/?utm_source=feedburner&utm_medium=email .