Stat243: Problem Set 2, Due Monday Sept. 17

September 12, 2018

This covers material in Unit 3.
It's due **as PDF submitted to bCourses** and submitted via Github at 2 pm on Sep. 17.
Some general guidelines on how to present your problem set solutions:

1. Please use Rmd/Rtex as in problem set 1.
2. Your solution should not just be code - you should have text describing how you approached the problem and what the various steps were.
3. Your PDF submission should be the PDF produced from your Rmd/Rtex. Your Github submission should include the Rtex/Rmd file, any R or bash 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*.
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.
5. 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.**

Problems

1. Please read Unit 5 on good programming/project practices and incorporate what you've learned from that reading into your solutions (particularly for Problem 3). As your response to this question, briefly (a few sentences) note what you did in your code that reflects the material in Unit 5. You could also note anything in Unit 5 that you disagree with, if you have a different stylistic perspective.
2. This problem explores file sizes and file-reading speed and their relationship to the file format in light of understanding storage in ASCII plain text versus binary formats and the fact that numbers are stored as 8 bytes per number in binary format.

    (a) Explain the sizes of the various files. In discussing the CSV text file, how many characters do you expect to be in the file (i.e., you should be able to estimate this very accurately without using *wc* or any explicit program that counts characters).

```
n <- 1e7
a <- matrix(rnorm(n), ncol = 100)
a <- round(a, 10)

write.table(a, file = '/tmp/tmp.csv', quote=FALSE, row.names=FALSE,
            col.names = FALSE, sep=',')
save(a, file = '/tmp/tmp.Rda', compress = FALSE)

file.size('/tmp/tmp.csv')

## [1] 133890295

file.size('/tmp/tmp.Rda')

## [1] 80000087
```

(b) Now consider saving out the numbers one row per number. Given we no longer have to save all the commas, why is the file size unchanged?

```
b <- a
dim(b) <- c(1e7, 1)  ## change to one column by adjusting attribute
write.table(b, file = '/tmp/tmp-onecolumn.csv', quote=FALSE,
            row.names=FALSE, col.names = FALSE, sep=',')
file.size('/tmp/tmp-onecolumn.csv')

## [1] 133890295
```

(c) Consider the following ways of reading the data into R. Explain the difference in speed between the two situations in "First comparison", then for the difference in "Second comparison", and for "Third comparison". Side note: in this case *read_csv()* is rather faster than *read.csv()*.

```
## First comparison
system.time(a0 <- read.csv('/tmp/tmp.csv', header = FALSE))

##    user  system elapsed
##  35.748   0.264  36.014

system.time(a1 <- scan('/tmp/tmp.csv', sep = ','))

##    user  system elapsed
##   5.236   0.048   5.283

## Second comparison
system.time(a0 <- read.csv('/tmp/tmp.csv',header = FALSE,
                           colClasses = 'numeric'))

##    user  system elapsed
##   5.336   0.028   5.363

system.time(a1 <- scan('/tmp/tmp.csv', sep = ','))
```

```
##    user  system elapsed
##   5.236   0.028   5.266

## Third comparison
system.time(a1 <- scan('/tmp/tmp.csv', sep = ','))

##    user  system elapsed
##   5.244   0.024   5.267

system.time(load('/tmp/tmp.Rda'))

##    user  system elapsed
##   0.080   0.000   0.083
```

(d) Explain why *tmp.Rda* is so much bigger than *tmp2.Rda* given they both contain the same number of numeric values.

```
save(a, file = '/tmp/tmp1.Rda')
file.size('/tmp/tmp1.Rda')

## [1] 76778914

b <- rep(rnorm(1), 1e7)
save(b, file = '/tmp/tmp2.Rda')
file.size('/tmp/tmp2.Rda')

## [1] 116494
```

3. 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 "Trevor Hastie", 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.
Note: if you repeatedly query the Google Scholar site too quickly, Google will start returning "503" errors because it detects automated usage (see problem 4 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.

   (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) Based on the discussion in lab on Tuesday Sept. 4, 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 for all of your checks but please use it for at least one. Also write some test code that uses the *testthat* package to carry out a small number of tests of your function.

(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.

4. Look at the *robots.txt* file for Google Scholar and the references in Unit 3 on the ethics of webscraping. Write a paragraph or two discussing the ethics of scraping data from Google Scholar as we do in Problem 3. Do you have any concerns in terms of whether anything we are doing in Problem 3 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.)