Problem Set 2

A.J. Torre

September 8, 2018

Problems

1. One way I incorporated what I learned is through adding comments in the beginning of the code that describe what the function does and who may be interested in using it. I also tried to add in some white space/ indentations to make the code easier to read and understand. I also tried to name variables/ objects things that made sense and are hopefully somewhat intuitive. Also, I used some "assertthat" statements in my first function in order to check if the input for the first and last name of the scholar are strings. I am still fairly new to coding, especially in R, and I think I will definitely work on trying to build these habits of writing understandable code.

2. a) The file.size function in R gives the size of the file in bytes. So, the csv file is 133887710 bytes and the rda file is 80000087 bytes. Binary files have that numeric characters will be 8 bytes/ character while in csv, this is not always the case. Since RDA is a binary format, we can divide the bytes by 8 to get how many numberic characters there are in the file; there are about 100,00 numeric characters and the 87 other bytes store metadata about the file. The csv file, which also has 100,000 numbers, is bigger because each separate character is a byte in ASCII file formats, including commas and new lines. So, for example, this table has 100 columns so that means that the csv file needs to store 99 commas per row as well as new line characters.

   b) The file size is unchanged as csvs count new lines as a character as well. So, even though we eliminate the commas, we still use as many bytes because now we are using new lines (1 byte each) to separate values.

   c) After looking at a couple articles online, I read a few that stated scan is quicker than using read table and thus, may be more convenient if dealing with a lot of data. Using colClasses, which helps specify what each column contains, makes it quicker for R to read in data as it doesn't have to go through eachh element in the column and check if it's a numeric value, which is what's happening between Comparison 1 and Comparison 2. When Looking at Comparison 1/2 and Comparsion 3, we see that it's much quicker for R to read in rda files and not csv. Rda's are R's native file formats so it makes it quicker to read and also takes less space when saving. The Rda file is also quicker for R to read as it's a binary file so its information is stored in the way that computers read in information.

   d) If we look at object b, we see that it's a vector that has 1e7 numbers (the same as a), but all these numbers are generated in the same way. It takes up less bytes/ space for R to repeat the same command to generate a number rather than each number coming from a different place and possibly being very different values, which is what is happening in object a. Rda can compress the same number more easily than different numnbers

3. a) To build this function, I first started with figuring out how to get from the search page to the scholar page, just through clicking on the query page to the User Profiles list page to the specific scholar profile page. I realized how I could use the first and last name of the scholar to get the "User Profiles" page and then used html nodes and attribute to get the link to the specific scholar's profile page. The important part of getting from the "User Profiles" page to the specific scholar's page is the Google Scholar ID, a 12 character string that comes after "=user" in the url and also within the html href. After getting this id (which is set to be the first or the only, depending on how many users there are with the same name), this id is then pasted with a base url to take us to the scholar profile. I first tried just going through the steps on how to do use Trevor Hastie's name to get his scholar id from through a query (using his name in the url) and then navigating using this id to navigate to his profile. After completing it for Trevor Hastie, I

was then able to use those steps to build a function for a general scholar. This function assumes that there is either only one scholar with this name or that the user wants the first scholar that is listed.

```
# this function is built to get the scholar id, url, and html
# text of the scholar profile page user inputs first and last
# name of a scholar for now, a limitation in this function is
# that it assumes the user wants the first/ only scholar with
# the name they entered
google.scholar.profile <- function(firstname, lastname) {
    assert_that(is.string(firstname))
    assert_that(is.string(lastname))
    firstname <- tolower(firstname)
    lastname <- tolower(lastname)
    baseurl <- "https://scholar.google.com/citations?view_op=search_authors&mauthors="
    fullname <- paste(firstname, "+", lastname, "&hl=en&oi=ao",
        sep = "")
    fullurl <- paste0(baseurl, fullname)
    html_query <- read_html(fullurl) %>% html_nodes("h3 a") %>%
        html_attr("href")
    user_code <- substring(html_query, 17, 28)
    general_profile_url <- "https://scholar.google.com/citations?user="
    scholar_profile <- paste0(general_profile_url, user_code,
        "&hl=en&oi=ao")
    scholar_profile <- scholar_profile[1]  #assume that either google search
    # only returns one scholar or user just wants the first
    # scholar
    if (scholar_profile == "https://scholar.google.com/citations?user=&hl=en&oi=ao") {
        stop("Sorry, there are no scholars with that name.")  #if no results come up
    }
    html_scholar <- read_html(scholar_profile)
    print(html_scholar)
    print(user_code)
    print(scholar_profile)
}
```

b) To build the table, I again started with just trying it for Trevor Hastie first. I originally attempted the html table function, which does provide a nice table from the webpage, but does not have the data sorted in the way the problem asks for it. So, what I found easiest was to get the information by columns as each column (mostly) has its own html node and attribute. The only two categories that I couldn't find a way to separate through html nodes and attributes was the authors and journal information. For this, I gathed the info from the html text together and then used a sequence of odd and even to separate the data. For the years and citations data, I also took off the first two indices of these character vectors as the actual years and numbers of citations didn't start until index 3. Once the I gathered all the data separately, I then used cbind to get a table with 5 columns. The most challenging part of 3.a) and b) was trying to look for the right nodes and attributes to get the information I was looking for.

```
# this function input will be from the url output of the
# previous function.
google.scholar.table <- function(scholar_profile) {
    title <- read_html(scholar_profile) %>% html_nodes(".gsc_a_t") %>%
        html_nodes(".gsc_a_at") %>% html_text()
```

```r
    authors_journals <- read_html(scholar_profile) %>% html_nodes(".gsc_a_t") %>%
        html_nodes(".gs_gray") %>% html_text()
    years <- read_html(scholar_profile) %>% html_nodes(".gsc_a_y") %>%
        html_text()
    cites <- read_html(scholar_profile) %>% html_nodes(".gsc_a_c") %>%
        html_text()
    authors <- authors_journals[seq(1, 40, 2)]  #splits the authors_journals character vector
    # to get the authors as every odd character
    journals <- authors_journals[seq(2, 40, 2)]  #splits the authors_journals character vector
    # to get the journals as every even character
    years <- years[c(-1, -2)]  #to get off the first 2 characters, that are the blank ' '
    # and 'Year' characters so all our columns are the same
    # length
    cites <- cites[c(-1, -2)]
    scholar_table <- cbind(title, authors, journals, years, cites)
    print(scholar_table)
}

# example table for Trevor Hastie
google.scholar.table("https://scholar.google.com/citations?user=tQVe-fAAAAAJ&hl=en&oi=ao")
```

```
##       title
##  [1,] "Unsupervised learning"
##  [2,] "Generalized additive models"
##  [3,] "Gene expression patterns of breast carcinomas distinguish tumor subclasses with clinical impli
##  [4,] "Regularization and variable selection via the elastic net"
##  [5,] "Least angle regression"
##  [6,] "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder
##  [7,] "Regularization paths for generalized linear models via coordinate descent"
##  [8,] "An introduction to statistical learning"
##  [9,] "Estimating the number of clusters in a data set via the gap statistic"
## [10,] "The elements of statistical learning"
## [11,] "The Dantzig selector: Statistical estimation when p is much larger than n"
## [12,] "Sparse inverse covariance estimation with the graphical lasso"
## [13,] "Statistical models in S"
## [14,] "A statistical explanation of MaxEnt for ecologists"
## [15,] "Diagnosis of multiple cancer types by shrunken centroids of gene expression"
## [16,] "Missing value estimation methods for DNA microarrays"
## [17,] "A working guide to boosted regression trees"
## [18,] "Sparse principal component analysis"
## [19,] "Varying-coefficient models"
## [20,] "Classification by pairwise coupling"
##       authors
##  [1,] "T Hastie, R Tibshirani, J Friedman"
##  [2,] "TJ Hastie"
##  [3,] "T Sørlie, CM Perou, R Tibshirani, T Aas, S Geisler, H Johnsen, T Hastie, ..."
##  [4,] "H Zou, T Hastie"
##  [5,] "B Efron, T Hastie, I Johnstone, R Tibshirani"
##  [6,] "J Friedman, T Hastie, R Tibshirani"
##  [7,] "J Friedman, T Hastie, R Tibshirani"
##  [8,] "G James, D Witten, T Hastie, R Tibshirani"
##  [9,] "R Tibshirani, G Walther, T Hastie"
## [10,] "J Friedman, T Hastie, R Tibshirani"
## [11,] "E Candes, T Tao"
## [12,] "J Friedman, T Hastie, R Tibshirani"
## [13,] "JM Chambers, TJ Hastie"
## [14,] "J Elith, SJ Phillips, T Hastie, M Dudík, YE Chee, CJ Yates"
## [15,] "R Tibshirani, T Hastie, B Narasimhan, G Chu"
## [16,] "O Troyanskaya, M Cantor, G Sherlock, P Brown, T Hastie, R Tibshirani, ..."
## [17,] "J Elith, JR Leathwick, T Hastie"
## [18,] "H Zou, T Hastie, R Tibshirani"
## [19,] "T Hastie, R Tibshirani"
## [20,] "T Hastie, R Tibshirani"
##       journals
##  [1,] "The elements of statistical learning, 485-585, 2009"
##  [2,] "Statistical models in S, 249-307, 2017"
##  [3,] "Proceedings of the National Academy of Sciences 98 (19), 10869-10874, 2001"
##  [4,] "Journal of the Royal Statistical Society: Series B (Statistical Methodology ..., 2005"
##  [5,] "The Annals of statistics 32 (2), 407-499, 2004"
##  [6,] "The annals of statistics 28 (2), 337-407, 2000"
##  [7,] "Journal of statistical software 33 (1), 1, 2010"
##  [8,] "springer, 2013"
##  [9,] "Journal of the Royal Statistical Society: Series B (Statistical Methodology ..., 2001"
## [10,] "Springer series in statistics 1 (10), 2001"
## [11,] "The Annals of Statistics 35 (6), 2313-2351, 2007"
## [12,] "Biostatistics 9 (3), 432-441, 2008"
## [13,] "Wadsworth & Brooks/Cole Advanced Books & Software, 1992"
## [14,] "Diversity and distributions 17 (1), 43-57, 2011"
## [15,] "Proceedings of the National Academy of Sciences 99 (10), 6567-6572, 2002"
## [16,] "Bioinformatics 17 (6), 520-525, 2001"
## [17,] "Journal of Animal Ecology 77 (4), 802-813, 2008"
## [18,] "Journal of computational and graphical statistics 15 (2), 265-286, 2006"
## [19,] "Journal of the Royal Statistical Society. Series B (Methodological), 757-796, 1993"
## [20,] "Advances in neural information processing systems, 507-513, 1998"
##       years  cites
##  [1,] "2009" "40041"
##  [2,] "2017" "15769"
##  [3,] "2001" "11890*"
##  [4,] "2005" "8007"
##  [5,] "2004" "7843"
##  [6,] "2000" "6260"
##  [7,] "2010" "5405"
##  [8,] "2013" "3286"
##  [9,] "2001" "3252"
## [10,] "2001" "2895"
## [11,] "2007" "2889"
## [12,] "2008" "2867"
## [13,] "1992" "2822"
## [14,] "2011" "2798"
## [15,] "2002" "2709"
## [16,] "2001" "2703"
## [17,] "2008" "2264"
## [18,] "2006" "2036"
## [19,] "1993" "1850"
## [20,] "1998" "1652"
```

c) When coding, I added assertthat in the first function to test whether the input of the user is strings, which will return an error if it's not. I also tested the function for the table with an "expect equal" to for the length of table when Trevor Hastie's profile url is the input. I also tried other expect commands outside the function, like the below, which again tests that the arguments of the first function are strings. While writing the actual function, I did informal testing, such as trying each/ every few steps of the function using Trevor Hastie's scholar id. I could definitely implement more formal testing using the assert_that and test_that package in R. I also found the assertthat function very handy for testing string inputs and was probably the most convenient code I found for testing the inputs of the function.

```r
expect_error(google.scholar.profile(12, 34))  #test for string inputs only
```

Here is the function for the google table with Trevor Hastie as the input and one extra line of code for expect equal for the length of the entries:

```r
# google table to test length of table assumes that
# researcher has full 20 entries on their profile
google.scholar.table <- function(scholar_profile) {
    title <- read_html(scholar_profile) %>% html_nodes(".gsc_a_t") %>%
        html_nodes(".gsc_a_at") %>% html_text()
    authors_journals <- read_html(scholar_profile) %>% html_nodes(".gsc_a_t") %>%
        html_nodes(".gs_gray") %>% html_text()
    years <- read_html(scholar_profile) %>% html_nodes(".gsc_a_y") %>%
        html_text()
    cites <- read_html(scholar_profile) %>% html_nodes(".gsc_a_c") %>%
        html_text()
    authors <- authors_journals[seq(1, 40, 2)]  #splits the authors_journals character vector
    # to get the authors as every odd character
    journals <- authors_journals[seq(2, 40, 2)]  #splits the authors_journals character vector
    # to get the authors as every even character
    years <- years[c(-1, -2)]  #to get off the first 2 characters, that are the blank ' '
    # and 'Year' characters so all our columns are the same
    # length
    cites <- cites[c(-1, -2)]
    scholar_table <- cbind(title, authors, journals, years, cites)
    print(scholar_table)
    expect_equal(length(scholar_table), 100)  #assuming first 20 entries are full for researcher
}
```

d) Extra credit- unable to complete.

4. Before reading these articles and learning about webscraping in this course, I wasn't very familiar with webscraping and what it entails. In fact, this assignment is the first time I've written code that collects data from the internet. Through looking at different blog posts and news articles webscraping, I was able to gain a better sense of what webscraping is and why there are ethical debates around it. I think one of the most interesting webscraping incidents I learned about was involving QVC and a shopping app called Resultly. In this case, Resultly was webscraping info off of QVC's sites, which led to them overloading and crashing QVC's servers. QVC sought legal action as they estimated that their server issue cost them around 2 million dollars. Before reading this case, I didn't have a solid grasp as to why webscraping can be considered unethical, but this legal case really caught my attention and showed a very powerful real-life example of the issues with webscraping. I also read about a couple other cases involving airline companies seeking legal actions on third-party apps or websites that webscrape in order to get the lowest deal/ compare flight prices between companies. One issue with webscraping is that some companies usewebscraping/ data from other companies for their own profit/ business ventures, like in the airlines cases. What I learned from reading about these real-life

cases of webscraping is that the actual laws and legality of webscraping still seem fuzzy and it may not be fully clear what is and isn't legal, although most sites recommend contacting a website's owner if you're planning on webscraping a lot of data.

I also looked at the Google Scholar robot.txt, which outlines what a bot/ crawler is allowed and not allowed to do on their website. From reading this txt, it looks like generic robots/ crawlers have some guidelines to follow while Twitter and Facebook bot's have free access. I think an interesting point is that Google Scholar does have regulations on webscraping, but I know there is indeed an R package specifically for getting data from Google Scholar. From reading posts on how to webscrape ethically, it seems like there are a key guidelines, such as not using the data to profit (like not taking data off of Craigslist for you to make your own shopping app), contacting the website owner about the webscraping, and waiting between crawling/ scraping in order to not overload the servers. Overall, I think webscraping seems really interesting and innovative, but I definitely see the ethical and legal issues with it, as illustrated by real-life legal cases.