# ps2

*Yihuan Song*

*9/9/2018*

## question 1

For my code in question 3, the good programming practices learned from unit 5 includes: the use of indentation and white space(blank lines) to make the structure of functions clear and readable; the use of header for the functions in order to specify the input, output and intention of the function; the use of comments to explain what the code is doing and summarize a block of code; careful use of naming style to avoid conflicts with functions that already exist(for example, there is a function named "citation" in r, therefore the function I wrote should not be named as "citation").

## question 2

(a) For the CSV text file, since the file size is 133887710, we know that the characters in CSV files are in ascii format, and one character takes one byte, so 133887710 characters are expected to be in the file. (The data is stored in character as "x.xxxxxxxxx,", so it takes 13 character to store one data, so it takes 13 bytes to store one data). For the Rda file, since the file size is 80000087, and the data is in binary, so we know that the number of data is 80000000/8 = 10000000. The size difference between the csv file and the rda file is because that the data in the csv file is stored in ascii format and the data the the rda file is in binary format, so the rda file has a smaller size.

(b) The file size is unchanged, because the first csv file is in matrix form, which is comma delimited, and each comma is used to seperate each piece of data. Changing the data from a matrix to a single column would not change the number of characters of the file, because changing to one column of data would just substitute the commas by the same number of new line characters, so there is no less file size.

(c) For the first comparison, the speed is different between read.csv() and scan(), because the function scan() just reads numeric data one by one ignoring lines, which is faster and more efficient than read.csv(). For the second comparison, the speed difference between read.csv() and scan() is much smaller, because specifying the "colClasses = 'numeric' " would help the read.csv() function identify the data to be read in as numeric format, so that it increases the efficiency for reading data. For the third comparison, the speed is different between using scan() to read in the csv file and using load() to read in the rda file, and the speed to load the rda file(binary format) is much faster than reading in the csv file(ascii format). The rda file in binary format is easier for R to process the data, and has less file size, so it reads data faster.

(d) tmp1.Rda is much bigger than tmp2.Rda, because the data in the temp.Rda is generated using a matirx format, while the data in the tmp2.Rda is generated using rep(), so it is in general numeric format, not stored as a matrix, which takes up less space. Therefore, the tmp1.Rda is larger than tmp2.Rda because of the different formatting of data.

## question 3

3.

(a) In the function "researcher_page", the function would take the researcher's name as input string, and return the citation page and Google Scholar ID of the researcher. First, the function would create the url to the searching page of the scholar. By using the developer's tool on Google Scholar's webpage, the "read_html" and "html_nodes" functions would provide a list of nodes with attribute "href" at

a specific div ID that provides the citation page information in the scholar-searching page. After inspecting the list of nodes, we could find out the Google scholar ID and the link to citation page for the specified scholar. After that, we could again use the "read_html" to get the html text for the citation page, and print out the page url and scholar's id.

```
#load required packages for the rest of the question
library(xml2)
library(rvest)
library(testthat)
library(assertthat)
library(knitr)
library(kableExtra)

researcher_page <- function(name) {
    # search for the citation page on Google Scholar of a specified researcher
    #
    # Arguments:
    #     researcher: a character string of the name of the researcher
    #
    # Returns:
    #     The html text corresponding to the researcher's citation page
    #     and the Google scholar ID of the researcher

  #reformat user input to create url for the search page of a scholar
    nm<-sub(" ", "+", name)
    baseurl1_1 <- "https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q="
    baseurl1_2 <- "&btnG="
    baseurl2 <- "https://scholar.google.com"
    url1<-paste0(baseurl1_1, nm, baseurl1_2)

  #using the html of the scholar's search page to find out the url and user-ID on the citation page
    listOfNodes <- read_html(url1) %>% html_nodes("div.gs_r") %>% html_nodes("a") %>% html_attr('href')
    Sys.sleep(3)
    url2 <- paste(baseurl2, strsplit(listOfNodes, " ")[[2]], sep="")
    userID <- strsplit(url2, "=")[[1]][2]

  #get the html and user id on the citation page, printing out the url and user ID
    research_html <- read_html(url2)
    cat("The researcher's Google Scholar ID is", userID)
    cat("\nThe researcher's citation page is\n", url2)
    return(research_html)
}

resch_html <- researcher_page("Trevor Hastie")
```

```
## The researcher's Google Scholar ID is tQVe-fAAAAAJ&hl
## The researcher's citation page is
##  https://scholar.google.com/citations?user=tQVe-fAAAAAJ&hl=en&oe=ASCII&oi=ao
```

(b) The create_table function takes the html prepared by part a) and creates an R data frame that contains the article title, authors, journal information, year of publication, and number of citations. Specifically, it first created 5 empty lists corresponding to the 5 columns of information required. Then the lists are filled by for loop going through each citation, selecting information from the html by "html_nodes" and "html_text", where the path to the node is generated by using the "copy as select" in Google's developer tools. After generating the 5 lists, combine them to a dataframe by "as.data.frame". The

printing of tables used “kable” in the “knitr” package, and a second scholar’s citation table is generated in order to provide more confidence that the function works properly.

```
citation_table <- function(html) {

    #Usage:
    #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
    #
    # Arguments:
    #     html: the html file stored in part a)
    #
    # Returns:
    #     The dataframe containing information for each citation

  #creating lists to be filled by citaion information
    author <- list()
    title <- list()
    journal <- list()
    year <- list()
    cited <- list()

  #looping for each citation on the citation page to create lists with information on
  #author,title, journal information, year of publication and number of citations

    for (i in 1:20){
    #using developer's tool, find out the selection path for each citation
        path_auth <- paste0("#gsc_a_b > tr:nth-child(",i,") > td.gsc_a_t > div:nth-child(2)")
        path_tit <- paste0("#gsc_a_b > tr:nth-child(",i,") > td.gsc_a_t > a")
        path_jour <- paste0("#gsc_a_b > tr:nth-child(",i,") > td.gsc_a_t > div:nth-child(3)")
        path_year <- paste0("#gsc_a_b > tr:nth-child(",i,") > td.gsc_a_y > span")
        path_num <- paste0("#gsc_a_b > tr:nth-child(",i,") > td.gsc_a_c > a")
    #by specifying selection path, find the required information inside the html
        auth <- html %>% html_node(path_auth) %>% html_text()
        tit <- html %>% html_node(path_tit) %>% html_text()
        jour <- html %>% html_node(path_jour) %>% html_text()
        yr <- html %>% html_node(path_year) %>% html_text()
        num <- html %>% html_node(path_num) %>% html_text()
    #put each citation's information in the lists
        author[[i]] <- auth
        title[[i]] <- tit
        journal[[i]] <- jour
        year[[i]] <- yr
        cited[[i]] <- num

    }

    #create the data frame by combining the 5 lists
    resch_table <- as.data.frame(cbind(author, title, year, cited, journal))
    return(resch_table)
}

#sample citation dataframe for "Trevor Hastie"
```

```
table_researcher1<-citation_table(resch_html)
kable(table_researcher1) %>%  kable_styling(latex_options="scale_down")
```

| author | title | year | cited | journal |
|---|---|---|---|---|
| T Hastie, R Tibshirani, J Friedman | Unsupervised learning | 2009 | 39960 | The elements of statistical learning, 485-585, 2009 |
| TJ Hastie | Generalized additive models | 2017 | 15762 | Statistical models in S, 249-307, 2017 |
| T Sorlie, CM Perou, R Tibshirani, T Aas, S Geisler, H Johnsen, T Hastie, ... | Gene expression patterns of breast carcinomas distinguish tumor subclasses with clinical implications | 2001 | 11880 | Proceedings of the National Academy of Sciences 98 (19), 10869-10874, 2001 |
| H Zou, T Hastie | Regularization and variable selection via the elastic net | 2005 | 7994 | Journal of the Royal Statistical Society: Series B (Statistical Methodology ..., 2005 |
| B Efron, T Hastie, I Johnstone, R Tibshirani | Least angle regression | 2004 | 7836 | The Annals of statistics 32 (2), 407-499, 2004 |
| J Friedman, T Hastie, R Tibshirani | Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors) | 2000 | 6255 | The annals of statistics 28 (2), 337-407, 2000 |
| J Friedman, T Hastie, R Tibshirani | Regularization paths for generalized linear models via coordinate descent | 2010 | 5396 | Journal of statistical software 33 (1), 1, 2010 |
| G James, D Witten, T Hastie, R Tibshirani | An introduction to statistical learning | 2013 | 3269 | springer, 2013 |
| R Tibshirani, G Walther, T Hastie | Estimating the number of clusters in a data set via the gap statistic | 2001 | 3248 | Journal of the Royal Statistical Society: Series B (Statistical Methodology ..., 2001 |
| J Friedman, T Hastie, R Tibshirani | The elements of statistical learning | 2001 | 2888 | Springer series in statistics 1 (10), 2001 |
| E Candes, T Tao | The Dantzig selector: Statistical estimation when p is much larger than n | 2007 | 2886 | The Annals of Statistics 35 (6), 2313-2351, 2007 |
| J Friedman, T Hastie, R Tibshirani | Sparse inverse covariance estimation with the graphical lasso | 2008 | 2861 | Biostatistics 9 (3), 432-441, 2008 |
| JM Chambers, TJ Hastie | Statistical models in S | 1992 | 2822 | Wadsworth & Brooks/Cole Advanced Books & Software, 1992 |
| J Elith, SJ Phillips, T Hastie, M Dudík, YE Chee, CJ Yates | A statistical explanation of MaxEnt for ecologists | 2011 | 2793 | Diversity and distributions 17 (1), 43-57, 2011 |
| R Tibshirani, T Hastie, B Narasimhan, G Chu | Diagnosis of multiple cancer types by shrunken centroids of gene expression | 2002 | 2708 | Proceedings of the National Academy of Sciences 99 (10), 6567-6572, 2002 |
| O Troyanskaya, M Cantor, G Sherlock, P Brown, T Hastie, R Tibshirani, ... | Missing value estimation methods for DNA microarrays | 2001 | 2700 | Bioinformatics 17 (6), 520-525, 2001 |
| J Elith, JR Leathwick, T Hastie | A working guide to boosted regression trees | 2008 | 2261 | Journal of Animal Ecology 77 (4), 802-813, 2008 |
| H Zou, T Hastie, R Tibshirani | Sparse principal component analysis | 2006 | 2032 | Journal of computational and graphical statistics 15 (2), 265-286, 2006 |
| T Hastie, R Tibshirani | Varying-coefficient models | 1993 | 1850 | Journal of the Royal Statistical Society. Series B (Methodological), 757-796, 1993 |
| T Hastie, R Tibshirani | Classification by pairwise coupling | 1998 | 1651 | Advances in neural information processing systems, 507-513, 1998 |

```
#sample citation dataframe for "Hui Zou"
resch_html2 <- researcher_page("Hui Zou")
```

```
## The researcher's Google Scholar ID is dJfEfJgAAAAJ&hl
## The researcher's citation page is
##  https://scholar.google.com/citations?user=dJfEfJgAAAAJ&hl=en&oe=ASCII&oi=ao
```

```
table_researcher2<-citation_table(resch_html2)
kable(table_researcher2) %>% kable_styling(latex_options="scale_down")
```

| author | title | year | cited | journal |
|---|---|---|---|---|
| H Zou, T Hastie | Regularization and variable selection via the elastic net | 2005 | 7994 | Journal of the Royal Statistical Society: Series B (Statistical Methodology ..., 2005 |
| H Zou | The adaptive lasso and its oracle properties | 2006 | 4175 | Journal of the American statistical association 101 (476), 1418-1429, 2006 |
| H Zou, T Hastie, R Tibshirani | Sparse principal component analysis | 2006 | 2032 | Journal of computational and graphical statistics 15 (2), 265-286, 2006 |
| H Zou, R Li | One-step sparse estimates in nonconcave penalized likelihood models | 2008 | 893 | Annals of statistics 36 (4), 1509, 2008 |
| H Zou, T Hastie, R Tibshirani | On the "degrees of freedom" of the lasso | 2007 | 760 | The Annals of Statistics 35 (5), 2173-2192, 2007 |
| J Zhu, H Zou, S Rosset, T Hastie | Multi-class adaboost | 2009 | 690 | Statistics and Its Interface 2 (3), 349-360, 2009 |
| H Zou, HH Zhang | On the adaptive elastic-net with a diverging number of parameters | 2009 | 419 | Annals of statistics 37 (4), 1733, 2009 |
| H Zou, M Yuan | Composite quantile regression and the oracle model selection theory | 2008 | 314 | The Annals of Statistics 36 (3), 1108-1126, 2008 |
| H Zou, Y Yang | Combining time series models for forecasting | 2004 | 236 | International journal of Forecasting 20 (1), 69-84, 2004 |
| L Wang, J Zhu, H Zou | The doubly regularized support vector machine | 2006 | 199 | Statistica Sinica 16 (2), 589, 2006 |
| B Kai, R Li, H Zou | New efficient estimation and variable selection methods for semiparametric varying-coefficient partially linear models | 2011 | 185 | Annals of statistics 39 (1), 305, 2011 |
| L Wang, J Zhu, H Zou | Hybrid huberized support vector machines for microarray classification and gene selection | 2008 | 171 | Bioinformatics 24 (3), 412-419, 2008 |
| L Xue, H Zou | Regularized rank-based estimation of high-dimensional nonparanormal graphical models | 2012 | 166 | Annals of Statistics 40 (5), 2541-2571, 2012 |
| B Kai, R Li, H Zou | Local composite quantile regression smoothing: an efficient and safe alternative to local polynomial regression | 2010 | 136 | Journal of the Royal Statistical Society: Series B (Statistical Methodology ..., 2010 |
| J Fan, L Xue, H Zou | Strong Oracle Optimality of Folded Concave Penalized Estimation | 2014 | 126 | Annals of Statistics 42 (3), 819-849, 2014 |
| Q Mai, H Zou, M Yuan | A Direct Approach to Sparse Discriminant Analysis in Ultra-high Dimensions. | 2012 | 110 | Biometrika 99 (1), 29-42, 2012 |
| H Zou, M Yuan | The F-infinity-norm support vector machine | 2008 | 98 | Statistica Sinica 18, 379-398, 2008 |
| L Xue, S Ma, H Zou | Positive Definite $\ell_1$ Penalized Estimation of Large Covariance Matrices* | 2012 | 89 | Journal of the American Statistical Association, 2012 |
| M Yuan, VR Joseph, H Zou | Structured variable selection and estimation | 2009 | 77 | The Annals of Applied Statistics, 1738-1757, 2009 |
| S Ma, L Xue, H Zou | Alternating Direction Methods for Latent Variable Gaussian Graphical Model Selection | 2013 | 75 | Neural Computation 25, 2172-2198, 2013 |

(c) Based on the function in part a), I included an "if" statement to first check if the user gives a string in the input. Then, by using the assertthat package, I created "on_failure" function and used "assert_that" to make sure the function fails with reminder message if there is no scholar found in Google Scholar accoring to the user's input. I also used the testthat package to write test functions to check the function returning the citation page("citation_assert") and the function creating citation table("citation_table") returns reasonable results.

```
citation_assert <- function(name) {
  #adding "if" statement to check user inputs a string
  if (class(name) != "character"){
    print("Invalid input, please input a scholar's full name")
  } else{
  #same as the "researcher_page" function in part a), despite some assertions added
    nm<-sub(" ", "+", name)
    baseurl1_1 <- "https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q="
    baseurl1_2 <- "&btnG="
    baseurl2 <- "https://scholar.google.com"
    url1<-paste0(baseurl1_1, nm, baseurl1_2)
  #Developer's tool to find and select the node to the scholar's profile page in the html
    selector <- "#gs_res_ccl_mid > div:nth-child(1) > h3 > a"
    test_node <- read_html(url1) %>% html_node(selector)
    Sys.sleep(3)
```

```
  #test to see if the scholar's profile(the test node) exists
    test <- function(test_node){
    length(test_node) != 0
    }
  #provide message to user if the scholar cannot be found
  on_failure(test) <- function(call, env) {
     paste0(name, " is not in Google Scholar")
  }
  #assert the tested node, quit if scholar not found
  assert_that(test(test_node))
  Sys.sleep(3)
  #rest is same as part a)
  listOfNodes <- read_html(url1) %>% html_nodes("div.gs_r") %>% html_nodes("a") %>% html_attr('href')
  Sys.sleep(3)
  url2 <- paste(baseurl2, strsplit(listOfNodes, " ")[[2]], sep="")
  research_html <- read_html(url2)
  userID <- strsplit(url2, "=")[[1]][2]
  cat("\nThe researcher's Google Scholar ID is", userID)
  cat("\nThe researcher's citation page is:\n", url2)
  return(research_html)
  }
}

#sample results: "fake name" should throw error with specific message to user,
#"Trevor Hastie" should function properly, and inputting number should fail and show message to user
citation_assert(1)
```

```
## [1] "Invalid input, please input a scholar's full name"
```

```
citation_assert("fake name")
```

```
## Error: fake name is not in Google Scholar
```

```
citation_assert("Trevor Hastie")
```

```
##
## The researcher's Google Scholar ID is tQVe-fAAAAAJ&hl
## The researcher's citation page is:
##  https://scholar.google.com/citations?user=tQVe-fAAAAAJ&hl=en&oe=ASCII&oi=ao
```

```
## {xml_document}
## <html>
## [1] <head>\n<title>Trevor Hastie - Google Scholar Citations</title>\n<me ...
## [2] <body><div id="gs_top" onclick="">\n<style>#gs_md_s,.gs_md_wnw{z-ind ...
```

```
#test if the functions' results are in appropriate class and length
test_that("citation_assert() returns a non-empty html text", {

  expect_true(length(citation_assert("Trevor Hastie")) != 0 )
  expect_is(citation_assert("Trevor Hastie"), 'xml_node')
})
```

```
##
## The researcher's Google Scholar ID is tQVe-fAAAAAJ&hl
## The researcher's citation page is:
##  https://scholar.google.com/citations?user=tQVe-fAAAAAJ&hl=en&oe=ASCII&oi=ao
## The researcher's Google Scholar ID is tQVe-fAAAAAJ&hl
```

```
## The researcher's citation page is:
##  https://scholar.google.com/citations?user=tQVe-fAAAAAJ&hl=en&oe=ASCII&oi=ao
```

```
test_that("citation_table() returns a non-empty dataframe", {

  expect_true(length(citation_table(resch_html)) != 0 )
  expect_is(citation_table(resch_html), 'data.frame')
})
```

## question 4

From the robots.txt file for Google Scholar, we see that the file specifies "Allow: /citations?user=", which means that the kind of data scraping of the citation page for a specific researcher can be considered as ethical. The robots.txt also specifies "Disallow: /scholar", which is the scrapping I use to find out the user ID, but since our purpose is not to request actural data from the search page for scholars, we are not actually violating Google's rules in problem 3. Furthermore, while I am scraping the citation page, my function makes consecutive quries in a short time, so in order not to be considered as a robot, I used Sys.sleep() to make sure that there are delays between successive requests, which is also avoiding the violation of general scraping ethics for sending large volume of requests at a short time.