ps2_ans

Weijie Yuan

9/7/2018

Answers for Problem 1

- I use informative and consistent naming convention.
- I use proper indentation, whitespace and blank line.
- I split long lines to make the meaning of long syntax clear.
- I add useful documentation.
- I add tests if necessary.

Answers for Problem 2

(a)

The encoding format of .csv file is 'utf-8' and every ASCII character is one byte in .csv file. Function round(a,10) convert numeric character into a format that one digit before dicimal point and ten digits after dicimal point (majority situation). Suppose half of "a" is positive of which size is 12 bytes and the other half is negative of which size is 13 bytes('-' is also one byte). And every row has 99 commas as separator and 1 line break except the last row which is free from line break. So we get:

$$12.5 * 10^7 + 10^7 - 1 = 134999999$$

And some of elements of "a" can not have exactly 10 digits after dicimal point because R will automatically omit the last digit if it is 0. Suppose tenth of elements of "a" have 9 digits after dicimal point (some of them have less digits). Then we can get the approximated number of file size, which is very close to the number that 'file.size('/tmp/tmp.csv')' show considering some of elements of "a" have less digits than 9 after dicimal point.

$$134999999 - 10^6 = 133999999$$

By contrast, .Rda file is binary and every number occupy 8 bytes regardless its dicimal digits and its characteristic of positive or negative. So we can get the fact that the file size is about $8 * 10^7$ bytes. And the other 87 bytes may store some metadata of .Rda file.

(b)

Saving out the numbers one row per number just changes the commas to line breaks. If we regard both comma and line break as separators, the total number of separators does not change at all.

(c)

- First Comparison

'read.csv' and 'read.table' are not the right tool for reading .txt file and .csv file containing large matrices, especially those with many columns. They are designed to read dataframes which may have columns of very different classes considering that they read all columns as character first and then convert them to numeric

objects. This slows down the speed of reading to a large extent. By contrast, the 'scan' function reads the fields of data in the file as specified by the what option, with the default being numeric. That means that scan() function assumes all the elements in the file is of the same type like numeric by default and it read a matrix omitting lines at higher speed. All of them speeds up the reading process.

- Second Comparison

Change one parameter of 'read.csv' function by assigning "colClasses = 'numeric' ". That means 'read.csv' will regard all columns as the same type which is 'numeric' in this case instead of reading distinct values as different character strings. So there are not much difference between 'read.csv' and 'scan' function.

- Third Comparison

The file size and file format determine the speed of reading file. As we know in 2(a), the file size of .csv is much larger than .Rda file when they store almost same data. Also, the 'load' function can read a compressed file ('save') directly from a file or from a suitable connection. All of them benefits the speed of reading .Rda file by 'load'.

(d)

The 'save' function automatically compress the data if user does not assign 'compress = FALSE'. 'b' has much duplicate message because all the numeric object in it is exactly the same. As a result, compression is more effective to 'b' than 'a', which leads to the fact that tmp.Rda is so much bigger than tmp2.Rda although they both contain the same number of numeric values.

Answers for Problem 3

(a)

Self define a function whose input is a charater string of the name of the researcher and whose output is his/her google scholar ID and corresponding citation page.

```
retrieveCitation<-function(name){
  # convert the name to proper format
  name <- gsub(" ", "+", name)
  # assign the baseURL and suffix of URL which needs to be used later
  # according to interface information from the develope tool of broswer
  baseUrl = "https://scholar.google.fr/scholar?hl=fr&as_sdt=0%2C5&q="
  suffix = "&btnG="
  # paster several part to needed URL
  pageText <- paste0(baseUrl, name, suffix)
  # read the html page using read_html function
  page <- read_html(pageText)
  # select the needed part including google scholar id of the research
  # using ".gs_rt2" direct the needed link
  text <- html_nodes(page, ".gs_rt2")
  # using regex to extract the corresponding google scholar id of the researcher
  user_code <- sub(".*user=(.*?)&.*", "\\1", text)
  # form citation page for the researcher
  citationPage = paste0("https://scholar.google.com/citations?user=",
                        user_code, "&hl=en&oi=ao")
  citationInfo = read_html(citationPage)
  # store the citation page and scholar id in a list
```

```
  infoResult <- list("Google Scholar ID" = user_code,
                     "Citation Page" = citationInfo)
  return(infoResult)
}
# example
retrieveCitation("Trevor Hastie")
```

```
## $`Google Scholar ID`
## [1] "tQVe-fAAAAAJ"
##
## $`Citation Page`
## {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 ...
```

(b)

Self define a function whose input is a charater string of the name of the researcher and whose output is his/her corresponding citation information including article title, author, journal information, years of publication and number of citation. And this function is builded based on the result html text retrieved from "retrieveCitation()" function self-defined in 3(a). Respectively, each colume of information are retrieved by 'xpath' after reading the sourses code of html.

```
retrieveInfo<-function(name){
  # retrieve html text from the result of retrieveCitation() function
  pageText <- retrieveCitation(name)[['Citation Page']]

  # retireve articleTitle from html text by setting xpath = "//a[@class='gsc_a_at']"
  # and using xml_text() and pipe to extract the text information
  articleTitle <- pageText %>%
    xml_nodes(xpath = "//a[@class='gsc_a_at']") %>% xml_text()
  # retireve author from html text by setting xpath = "//div[@class='gs_gray']"
  author <- pageText %>%
    xml_nodes(xpath = "//div[@class='gs_gray']") %>% xml_text()
  # check the result and find above xpath return information about author and
  # journal as well. Extract the author information by assigning particular index
  author <- author[seq(from = 1, to = 40, by = 2)]
  # retireve journal information from html text by setting
  # xpath = "//div[@class='gs_gray']"
  # Extract the journal information by assigning particular index
  journalInformation <- pageText %>%
    xml_nodes(xpath = "//div[@class='gs_gray']")%>% xml_text()
  journalInformation <- journalInformation[seq(from = 2, to = 40, by = 2)]
  # retireve year of publication by setting
  # xpath = "//span[@class='gsc_a_h gsc_a_hc gs_ibl']"
  yearsofPublication <- pageText %>%
    xml_nodes(xpath = "//span[@class='gsc_a_h gsc_a_hc gs_ibl']") %>% xml_text()
  # retireve number of citation by setting
  # xpath = "//a[@class='gsc_a_ac gs_ibl']"
  numberofCitation <- pageText %>%
    xml_nodes(xpath = "//a[@class='gsc_a_ac gs_ibl']") %>% xml_text()
```

```r
  # put all information mentioned above into a dataframe and return it as result
  dataFrame <- data.frame('article title' = articleTitle,
                          'author' = author,
                          'journal information' = journalInformation,
                          'years of publication' = yearsofPublication,
                          'number of citation' = numberofCitation,
                          check.names = FALSE)
  return(dataFrame)
}
# second example to check the function is working properly
retrieveInfo("Emmanuel Saez")
```

```
##
## 1                                                                    Income Inequality in the United
## 2                                                                          Top incomes in the l
## 3  The role of information and social interactions in retirement plan decisions: Evidence from a ran
## 4                    Where is the land of opportunity? The geography of intergenerational mobility in
## 5                                                          Using elasticities to derive optima
## 6                                 How does your kindergarten classroom affect your earnings? Evidence
## 7                                                    The elasticity of taxable income: evidenc
## 8                                The elasticity of taxable income with respect to marginal tax rates:
## 9                                             The evolution of top incomes: a historical and interna
## 10                         Wealth inequality in the United States since 1913: Evidence from capitaliz
## 11                                                                       Do taxpayers bun
## 12                                  Optimal income transfer programs: intensive versus extensive labo
## 13                                                   The top 1% in international and hist
## 14                                        Unwilling or unable to cheat? Evidence from a tax audit exp
## 15                   Participation and investment decisions in a retirement plan: The influence of c
## 16                                      Dividend taxes and corporate behavior: Evidence from the 200
## 17                                               Optimal taxation of top labor incomes: A tale of
## 18                 Earnings inequality and mobility in the United States: evidence from social securi
## 19                                               Inequality at work: The effect of peer salaries o
## 20                 Striking it richer: The evolution of top incomes in the United States (update wi
##                                                      author
## 1                                              T Piketty, E Saez
## 2                                   AB Atkinson, T Piketty, E Saez
## 3                                                E Duflo, E Saez
## 4                               R Chetty, N Hendren, P Kline, E Saez
## 5                                                      E Saez
## 6  R Chetty, JN Friedman, N Hilger, E Saez, DW Schanzenbach, D Yagan
## 7                                               J Gruber, E Saez
## 8                                  E Saez, J Slemrod, SH Giertz
## 9                                              T Piketty, E Saez
## 10                                              E Saez, G Zucman
## 11                                                     E Saez
## 12                                                     E Saez
## 13                          F Alvaredo, A Atkinson, T Piketty, E Saez
## 14             HJ Kleven, MB Knudsen, CT Kreiner, S Pedersen, E Saez
## 15                                               E Duflo, E Saez
## 16                                              R Chetty, E Saez
## 17                                T Piketty, E Saez, S Stantcheva
## 18                                      W Kopczuk, E Saez, J Song
## 19                                D Card, A Mas, E Moretti, E Saez
## 20                                                     E Saez
```

```
##                                                    journal information
## 1                      Quarterly Journal of Economics 118 (1), 1-39, 2003
## 2                       Journal of Economic Literature 49 (1), 3-71, 2011
## 3                    Quarterly Journal of Economics 118 (3), 815-842, 2003
## 4                  Quarterly Journal of Economics 129 (4), 1553-1623, 2014
## 5                              Review of Economic Studies 68, 205-229, 2001
## 6                  Quarterly Journal of Economics 126 (4), 1593-1660, 2011
## 7                           Journal of public Economics 84 (1), 1-32, 2002
## 8                        Journal of economic literature 50 (1), 3-50, 2012
## 9  American Economic Review: Papers and Proceedings 96 (2), 200-205, 2006
## 10                   Quarterly Journal of Economics 131 (2), 519-578, 2016
## 11        American economic Journal: economic policy 2 (3), 180-212, 2010
## 12                 Quarterly Journal of Economics 117 (3), 1039-1073, 2002
## 13                     Journal of Economic Perspectives 27 (3), 3-20, 2013
## 14                                      Econometrica 79 (3), 651-692, 2011
## 15                       Journal of public Economics 85 (1), 121-148, 2002
## 16                   Quarterly Journal of Economics 120 (3), 791-833, 2005
## 17        American Economic Journal: Economic Policy 6 (1), 230-271, 2014
## 18                The Quarterly Journal of Economics 125 (1), 91-128, 2010
## 19                       American Economic Review 102 (6), 2981-3003, 2012
## 20
##    years of publication number of citation
## 1                  2003               3510
## 2                  2011               2057
## 3                  2003               1205
## 4                  2014               1104
## 5                  2001               1035
## 6                  2011                989
## 7                  2002                957
## 8                  2012                952
## 9                  2006                912
## 10                 2016                803
## 11                 2010                747
## 12                 2002                742
## 13                 2013                683
## 14                 2011                654
## 15                 2002                623
## 16                 2005                571
## 17                 2014                566
## 18                 2010                546
## 19                 2012                536
## 20                 2009                518
```

(c)

Firstly, we can add a assert_that function to check if the user provides a character string.

```
retrieveCitation<-function(name){

  #check the input is a character string
  assert_that(is.character(name))

  name <- gsub(" ", "+", name)
```

```
  baseUrl = "https://scholar.google.fr/scholar?hl=fr&as_sdt=0%2C5&q="
  suffix = "&btnG="
  pageText <- paste0(baseUrl, name, suffix)
  page <- read_html(pageText)
  text <- html_nodes(page, ".gs_rt2")
  user_code <- sub(".*user=(.*?)&.*", "\\1", text)
  citationPage = paste0("https://scholar.google.com/citations?user=",
                        user_code, "&hl")
  citationInfo = read_html(citationPage)
  infoResult <- list("Google Scholar ID" = user_code,
                     "Citation Page" = citationInfo)
  return(infoResult)
}
# example
retrieveCitation(123)
```

```
## Error: name is not a character vector
```

And then, we can self-define a function to check if Google Scholar doesn't return a result.

```
# check if the length of x is zero
isNull <- function(x){
  assert_that(is.character(x))
  length(x) != 0
}

# customize the error message
on_failure(isNull) <- function(call, env) {
  "Google Scholar can not find proper information about this researcher."
}

retrieveCitation<-function(name){

  #check the input is a character string
  assert_that(is.character(name))

  name <- gsub(" ", "+", name)
  baseUrl = "https://scholar.google.fr/scholar?hl=fr&as_sdt=0%2C5&q="
  suffix = "&btnG="
  pageText <- paste0(baseUrl, name, suffix)
  page <- read_html(pageText)
  text <- html_nodes(page, ".gs_rt2")
  user_code <- sub(".*user=(.*?)&.*", "\\1", text)

  # use self-define assert_that function on user_code
  assert_that(isNull(user_code))

  citationPage = paste0("https://scholar.google.com/citations?user=",
                        user_code, "&hl")
  citationInfo = read_html(citationPage)
  infoResult <- list("Google Scholar ID" = user_code,
                     "Citation Page" = citationInfo)
  return(infoResult)
}
# example
```

```
retrieveCitation("Weijie Yuan")
```

```
## Error: Google Scholar can not find proper information about this researcher.
```

As for test_that package, we can use expect function to check if the length of output information and the colnames of output dataframe are proper.

```
retrieveInfo<-function(name){
  pageText <- retrieveCitation(name)[['Citation Page']]

  articleTitle <- pageText %>%
    xml_nodes(xpath = "//a[@class='gsc_a_at']") %>% xml_text()
  # check if the length of information is equal to 20
  expect_length(articleTitle, 20)
  author <- pageText %>%
    xml_nodes(xpath = "//div[@class='gs_gray']") %>% xml_text()
  author <- author[seq(from = 1, to = 40, by = 2)]
  # check if the length of information is equal to 20
  expect_length(author, 20)
  journalInformation <- pageText %>%
    xml_nodes(xpath = "//div[@class='gs_gray']")%>% xml_text()
  journalInformation <- journalInformation[seq(from = 2, to = 40, by = 2)]
  # check if the length of information is equal to 20
  expect_length(journalInformation, 20)
  yearsofPublication <- pageText %>%
    xml_nodes(xpath = "//span[@class='gsc_a_h gsc_a_hc gs_ibl']") %>% xml_text()
  # check if the length of information is equal to 20
  expect_length(yearsofPublication, 20)
  numberofCitation <- pageText %>%
    xml_nodes(xpath = "//a[@class='gsc_a_ac gs_ibl']") %>% xml_text()
  # check if the length of information is equal to 20
  expect_length(numberofCitation, 20)

  dataFrame <- data.frame('article title' = articleTitle,
                          'author' = author,
                          'journal information' = journalInformation,
                          'years of publication' = yearsofPublication,
                          'number of citation' = numberofCitation,
                          check.names = FALSE)

  # check if the colnames of result dataframe is in right order
  expect_equal(colnames(dataFrame), c("article title", "author",
                                      "journal information",
                                      "years of publication",
                                      "number of citation"))
  return(dataFrame)
}
```

(d)

Self define a function whose input a character string of the name of researcher who users are interested in and whose output is all of his/her citation information including article title, author, journal information, years of publication and number of citation.

When I click on "Show More", the header of network in developed tool of broswer shows that "&cstart=20&pagesize=80", "&cstart=100&pagesize=100" and so on. So in this function, I set pagesize=20, which means in each query, this function can download html text including 20 citation information.

Furthermore, it doesn't make sense that a citation information does not include article title. So this function use this variable as criterion of while() loop. Once the returned article title is a character(0) object, then this function stop webscraping. Last but not least, because the result is too large to show, I use "eval = FALSE" in r chunk and write the whole result into a .csv file and upload it to my github.

```
retrieveAllInfo <- function(name){
  # retieve the information from the home page
  # use the same method as that in retieveInfo function
  text <- read_html(paste0("https://scholar.google.com/citations?user=",
                    retrieveCitation(name)[['Google Scholar ID']],
                    "&hl=en&oi=ao"))

  articleTitle <- text %>%
    xml_nodes(xpath = "//a[@class='gsc_a_at']") %>% xml_text()
  author <- text %>%
    xml_nodes(xpath = "//div[@class='gs_gray']") %>% xml_text()
  author <- author[seq(from = 1, to = 40, by = 2)]
  journalInformation <- text %>%
    xml_nodes(xpath = "//div[@class='gs_gray']")%>% xml_text()
  journalInformation <- journalInformation[seq(from = 2, to = 40, by = 2)]
  yearsofPublication <- text %>%
    xml_nodes(xpath = "//span[@class='gsc_a_h gsc_a_hc gs_ibl']") %>% xml_text()
  numberofCitation <- text %>%
    xml_nodes(xpath = "//a[@class='gsc_a_ac gs_ibl']") %>% xml_text()

  # set the start index=20 and pageszie=20
  # for larger information, one can increase the pagesize of each retrieve
  start = 20
  pagesize = 20
  articleTitleAdd <- articleTitle

  # multiple queries until article title is null
  while(length(articleTitleAdd)!=0){
    # paste the start and pagesize parameter into url variable
    url <- paste0(
      "https://scholar.google.com/citations?user=",
      retrieveCitation(name)[['Google Scholar ID']],
      "&hl=en&oi=ao&cstart=", start, "&pagesize=",pagesize)

    # prepare for the next query
    start = start + pagesize
    pageText <- read_html(url)

    # use the same method as that in retieveInfo function
    articleTitleAdd <- pageText %>%
      xml_nodes(xpath = "//a[@class='gsc_a_at']") %>% xml_text()
    # bind the new information to the old one
    articleTitle <- c(articleTitle, articleTitleAdd)
    authorAdd <- pageText %>%
      xml_nodes(xpath = "//div[@class='gs_gray']") %>% xml_text()
    authorAdd <- authorAdd[seq(from = 1, to = 40, by = 2)]
```

```
    author <- c(author, authorAdd)
    journalInformationAdd <- pageText %>%
    xml_nodes(xpath = "//div[@class='gs_gray']")%>% xml_text()
    journalInformationAdd <- journalInformationAdd[seq(from = 2, to = 40, by = 2)]
    journalInformation <- c(journalInformation, journalInformationAdd)
    yearsofPublicationAdd <- pageText %>%
      xml_nodes(xpath = "//span[@class='gsc_a_h gsc_a_hc gs_ibl']") %>% xml_text()
    yearsofPublication <- c(yearsofPublication, yearsofPublicationAdd)
    numberofCitationAdd <- pageText %>%
      xml_nodes(xpath = "//a[@class='gsc_a_ac gs_ibl']") %>% xml_text()
    numberofCitation <- c(numberofCitation, numberofCitationAdd)

    # set system sleep between the calls in case Google detects automated usage
    Sys.sleep(0.5)
}

len = length(articleTitle)
# put all information into a single dataframe and return it as result
dataFrame <- data.frame('article title' = articleTitle,
                        'author' = author[1:len],
                        'journal information' = journalInformation[1:len],
                        'years of publication' = yearsofPublication[1:len],
                        'number of citation' = numberofCitation[1:len],
                        check.names = FALSE)
return(dataFrame)
}
# it is difficult and verbose to show so much information in .pdf
# so I write all the information into .csv file and upload it to github
write.csv(retrieveAllInfo("Trevor Hastie"), file = "Result.csv")
```

Answer for Problem 4

Firstly, I think that what I am doing in Problem 3 is ethical and my behavior is proper. I request the data at a reasonable rate by setting system sleep time. I only scrape what I need from Google Scholar. I respect the information and data I keep. I try to select the useful information to return value. I am creating a new value rather than earn something commercially by using scraped content. Then, I look over the Robots.txt file of Google Scholar. It is obvious that I violate one of the rules "Disallow: /citations?*cstart=". It can be one of my concerns that I may diminish value for Google Scholar.

In some specific context, for example, the owner of a website monetizes its content or data by setting some ads before a visitor can access the whole content. In contrast, webscraping skips all the interface made for human, which means scraper acquire the content at lower price. It certainly leads to diminishing value for the owner of website. Another side effect of webscaping is that it potentially increases the chance of revealing individual privacy or organizational privacy. Last but not least, webscraping may overload or damage a web server, which is a great loss for the web owner.