PS2

Mengying Yang

September 15, 2017

0.1 1. a)

(1) Because the file store as a csv file, therefore, it's format of the file is ASCII text. Each letter store in ASCII format occupies 1 bytes in computer. There are 1e6 letters, totally need 1e6 bytes. Also because there is a '/n' after each letter (1e6 '/n's in total, and 1 byte each), which will cost 1e6 bytes totally. That's why there are 2000000 bytes need.

(2) When we pasted all letters together without new lines after each letter, the total letters will still need 1e6 bytes for storage. In the end of all letters, there will be an end-of-line character (1 byte), so totally 1000001 bytes

(3) In the third case, we stored 1e6 numbers in binary format. The Rda(binary format) will zip the file therefore, it will less than ASCII format file. In addition to that, each number will length 17 to 18(if include a negative sign) which will need 8 bytes, therefore, the total bytes were needed are around 8000000 bytes.

(4) In this case, we store the numbers in ASCII text again and then each number or decimal point will need 1 byte. The total bytes of an 18-length number is 18 bytes. Because include new line and negative sign, the total bytes needed are (0.5)*1e6 +17*1e6 +1*1e6 = 18.5*1e6 (represent negative sign, number + decimal point, /n respectively)

(5) if we use round (num, 2), then we round the number to 2 decimals. Including the decimal point, the length of a positive number is 4 and negative number is 5 (maybe more than that, but less probability and small influence). If we include the new line, the length will be 5-6. As we know, each number is 1 bytes in ASCII text. The total bytes is around 550000 bytes.

1. b) because 'save' function can save file and compress it in binary much better (the default of the function's compress option is true), so the 1e6 letters storage size become smaller. And the repetition of 'a' can compress even more, so is much smaller than 1e6 letters.

0.2 2.a)

I searched several google scholar's website and I found the URL of the researcher's profile is construct with the researcher's name and the rest are same. Therefore through construct a URL by input, we could find the html. Through the getHTMLLinks function we can find the hyperlink inside of the html and therefore get the researchers' profile URL and extract their scholarID

```
#Loading required library
library(stringr)
library(XML)
library(curl)
readpage <- function(researchername){

  #seperate the input to two part based on the space between the input and
  #paste the two part with a +
  nameindex <- paste0(unlist(strsplit(researchername, ' ')), collapse = '+')

  #plugin the nameindex to the URL so that we could get the URL of the search
  #result in Google Scholar based on the input
```

```
  URL <- paste0("https://scholar.google.com/scholar?hl=en&q=", nameindex, "&btnG=&as_sdt=1%2C5&as_sdtp=

  #read the html document based on URL
  html <- readLines(URL)

  # read all links in the html document
  links <- getHTMLLinks(html)

  #I found the 36th link contain the scholarID and I construct the URL of the
  #Google scholar's profile
  URL1 <- paste0("https://scholar.google.com",links[36])

  #read the html document and save it as html1, this is basicly what we want
  html1 <- readLines(URL1)

  #extract the scholarID from the URL
  scholarID <- str_extract(URL1, "=[[:alnum:]-]+&")
  scholarID <- str_replace_all(scholarID, "=(.*)&", "\\1")

  #To let the function return all the information we need,
  #I constructed this information to a list. The first element is the
  #html text of the researcher's profile, the second is the scholar IDD
  citationinfo = c(html1, scholarID)

  return(citationinfo)
}
```

0.3 2. b)

In this part, I Created a 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.

When I used readHTMLTable function, I found the author name, jounal information, title cannot be seperate. Therefore, I used regular expression to extract the useful information, like title of papers, authors and journal information. I still used readHTMLTable function for the last two column(number of citation and year), because I found it will be easier to gain the information for this part. Then I combine the five information vectors to a table and return it.

```
tablef <- function(researchername){

  # gain the html text we want through the funciton we construct in 2 a)
  new <- readpage(researchername)

  #use regular expression to extract the string contain title
  title <- str_extract_all(new[1], "gsc_a_at\">.*?</a><div")
  title <- unlist(title)
  title <- str_replace_all(title, "gsc_a_at\">(.*?)</a><div", "\\1")

  #similar as above to find authors and journal information
  author <- str_extract_all(new[1], "gs_gray\">.*?gs_gray\"")
  author <- unlist(author)
  author <- str_replace_all(author, "gs_gray\">(.*?)</div><div class=\"gs_gray\"", "\\1")
```

```
  journal <- str_extract_all(new[1], "iv><div class=\"gs_gray\">.*?<span")
  journal <- unlist(journal)
  journal <- str_replace_all(journal, "iv><div class=\"gs_gray\">(.*?)<span", "\\1")

  #use readHTMLTable to get the citation number and year.(we only need the 2nd and 3rd column)
  tt<-readHTMLTable(new[1],which = 2)
  NumCite <- tt[,c(2)]
  NumCite <- as.character(NumCite)
  NumCite <- as.vector(NumCite)

  year <- tt[,c(3)]
  year <- as.character(year)
  year <- as.vector(year)

  #combine them to one table
  finaltable <- cbind(title, author, journal, NumCite, year)
  return(finaltable)
}
```

0.4 2.c)

There are some situation the function may not work, such as the input is invalid, Google scholar cannot find the researcher and so on. Therefore, I rewrite the 2 a) so that it can give some error message if these situation happened.

```
checkfunction <- function(researchername){

  #first check if the input is valid string
  if(!is.character(researchername)){
    return("please input a valid researcher name")
  }

  #similar with a)
  nameindex <- paste(unlist(strsplit(researchername, ' ')), collapse = '+')
  URL <- paste0("https://scholar.google.com/scholar?hl=en&q=", nameindex, "&btnG=&as_sdt=1%2C5&as_sdtp=
  html <- readLines(URL)
  links <- getHTMLLinks(html)

  #second check if there is google scholar base on the input name, if not reture the error message
  URL1 <- paste0("https://scholar.google.com",links[36])
  if(!str_detect(S)(URL1, "citations\\?user=")){
    return("There is Google scholar profile was found based on provided researcher name")
  }

  #redo the same thing in 2(a)
  html1 <- readLines(URL1)
  scholarID <- str_extract(URL1, "=[[:alnum:]-]+&")
  scholarID <- str_replace_all(scholarID, "=(.*)&", "\\1")
  citationinfo = c(html1, scholarID)
  return(citationinfo)
}
```

Then I used the testthat function to setup several test-cases to test if the functions works well.

```
library(testthat)

#test if we can show the error message if the input is not valid
test_that("Input is invalid", {
  expect_equal(checkfunction("12"), "please input a valid researcher name")
})

#test if the URL is not valid, will the function can detect it and give error message
test_that("URL is not valid error message",{
  expect_equal(checkfunction("lalallalala"), "There is Google scholar profile was found based on provid
})

#test if the result of the scholarID is correct
test_that("ScholarID is incorrect",{
  expect_equal(readpage("Bin yu")[2], "xT19JcOAAAAJ")
  expect_equal(readpage("geoffrey hinton")[2], "JicYPdAAAAAJ")
})

#test if the table is five columns
test_that("correct column of the table",{
  expect_equal(ncol(tablef("geoffrey hinton")),5)
})
```

0.5 2.d)

This part is very similar with 2b, but, instead of just obtain the first citation page table, we want all of the paper citation informations related to the google scholar and then output a table based on the five column.

If we click the next page, I found the URL is different. And the different part is a index which we could use and change it to gain different page's information. Therefore, we need a loop to change the index of the URL until the last page.

```
fulltable <-function(){
  index = 0 # the index is the difference of each URL
  scholarID <- (readpage(researchername))[2]
  judge <- T # judge is used in while loop to check if the loop will continue or stop. T the while loop
while(judge){

  # construct the URL
  URL <- paste0("https://scholar.google.com/citations?user=",scholarID,"&hl=en&oi=ao&cstart=",index,"&p

  #read and save the html text
  html<-readLines(URL)

  #gain table elements in this page
  title <- str_extract_all(html, "gsc_a_at\">.*?</a><div")
  title <- unlist(title)
  title <- str_replace_all(title, "gsc_a_at\">(.*?)</a><div", "\\1")

  author <- str_extract_all(html, "gs_gray\">.*?gs_gray\"")
  author <- unlist(author)
  author <- str_replace_all(author, "gs_gray\">(.*?)</div><div class=\"gs_gray\"", "\\1")

  journal <- str_extract_all(html, "iv><div class=\"gs_gray\">.*?<span")
```

```
  journal <- unlist(journal)
  journal <- str_replace_all(journal, "iv><div class=\"gs_gray\">(.*?)<span", "\\1")

  tt<-readHTMLTable(html,which = 2)
  NumCite <- tt[,c(2)]
  NumCite <- as.character(NumCite)
  NumCite <- as.vector(NumCite)

  year <- tt[,c(3)]
  year <- as.character(year)
  year <- as.vector(year)

  table <- cbind(title, author, journal, NumCite, year)

  #if is the first page, paste the table to finaltable,
  #if not the first page combine the new table to last finaltable and save them as finaltable
  if(index == 0){
    finaltable <- table
  }else{
    finaltable <- rbind(finaltable, table)
  }

  #check if the number of rows of the new table smaller than 20,
  #let judge equal to F which means we arrive the last page.
  if(dim(table)[1]<20){
      judge <-F
  }

  #add 20 to this index so that in next loop, we contruct a new URL
  index <- index +20
  Sys.sleep(2)
}
  #return the final table
  return(finaltable)
}
```