Problem Set 2

Hsieh Cheng Han

September 15, 2017

Problem Set 2

0.1 1. a)

```
chars <- sample(letters, 1e6, replace = TRUE)
# Create a character vector with 1000000 elements,
# each is a lower alphabet from a to z.
write.table(chars, file = 'tmp1.csv', row.names = FALSE, quote = FALSE,
col.names = FALSE)
# Create a csv file called 'tmp1', which accepts elements from chars.
system('ls -l tmp1.csv', intern = TRUE)
```

```
## [1] "-rw-r--r--  1 henry50618  staff  2000000  9 15 14:34 tmp1.csv"
```

```
# list details of tmp1.csv and capture the result as a character vector.
# It shows 2000000 bytes since a comma also counts for 1 byte.
chars <- paste(chars, collapse = '')
# concatenate elements in chars without other characters
# like whitespaces or commas.
write.table(chars, file = 'tmp2.csv', row.names = FALSE, quote = FALSE,
col.names = FALSE)
# Again create a csv file called 'tmp2', which accepts elements from chars,
# which has been concatenated.
system('ls -l tmp2.csv', intern = TRUE)
```

```
## [1] "-rw-r--r--  1 henry50618  staff  1000001  9 15 14:34 tmp2.csv"
```

```
# list details of tmp2.csv and capture the result as a character vector.
# It shows 1000001 bytes since there is no comma among the
# 1000000-characters string(but has one after the string).
nums <- rnorm(1e6)
# Create a numeric vector with 1000000 numeric elements from N(0,1).
save(nums, file = 'tmp3.Rda')
# Save nums as a Rda file.
system('ls -l tmp3.Rda', intern = TRUE)
```

```
## [1] "-rw-r--r--  1 henry50618  staff  7678102  9 15 14:34 tmp3.Rda"
```

```
# list details of tmp3.Rda and capture the result as a character vector.
# It shows approximate 7678000 bytes since each numeric element is 8 bytes in R.
# The reason that actual size is less than 8000000 bytes is
# sharing-memory mechanism used by R.
write.table(nums, file = 'tmp4.csv', row.names = FALSE, quote = FALSE, col.names = FALSE, sep = ',')
# Create a csv file called 'tmp4', which accepts elements from nums.
system('ls -l tmp4.csv', intern = TRUE)
```

```
## [1] "-rw-r--r--  1 henry50618  staff  18160358  9 15 14:34 tmp4.csv"

# list details of tmp4.Rda and capture the result as a character vector.
# It shows approximate 18159000 bytes.
# Check tmp4.csv we observe that each number is stored as 17-characters string(or 18, with negative sig
# As a result, the total bytes should be approximate (18+1)*1000000 = 19000000 bytes when considering c
write.table(round(nums, 2), file = 'tmp5.csv', row.names = FALSE,
quote = FALSE, col.names = FALSE, sep = ',')
# round the elements in nums to two digits and save it as a csv file.
system('ls -l tmp5.csv', intern = TRUE)

## [1] "-rw-r--r--  1 henry50618  staff  5378240  9 15 14:34 tmp5.csv"

# list details of tmp5.csv and capture the result as a character vector.
# It shows approximate 5378000 bytes.
# Each number is 4 or 5 bytes, so the total size is no more than 6000000 bytes.
```

0.2 1. b)

```
chars <- sample(letters, 1e6, replace = TRUE)
chars <- paste(chars, collapse = '')
save(chars, file = 'tmp6.Rda')
system('ls -l tmp6.Rda', intern = TRUE)

## [1] "-rw-r--r--  1 henry50618  staff  635267  9 15 14:34 tmp6.Rda"

# list details of tmp6.Rda and capture the result as a character vector.
# It shows approximate 630000 bytes.
# Each character is 1 byte, so the total bytes are less 1000000 bytes.
chars <- rep('a', 1e6)
chars <- paste(chars, collapse = '')
save(chars, file = 'tmp7.Rda')
system('ls -l tmp7.Rda', intern = TRUE)

## [1] "-rw-r--r--  1 henry50618  staff  1056  9 15 14:34 tmp7.Rda"

# list details of tmp7.Rda and capture the result as a character vector.
# It shows 1056 bytes.
# We used replicate function and thus R creates a pointer toward 'a'.
# So the actual byte is the pointer's byte, which is relatively small.
```

0.3 2. a)

```
readscholar <- function(Scholarname){
  name <- gsub(" ","",Scholarname)
  # Remove blankspaces among scholar's name.
  URL <- paste0("https://scholar.google.com/citations?view_op=search_authors&mauthors=",name,"&hl=en&oi
  # Put the name into the hyperlink
  html <- readLines(URL)
  # Read HTML files from URL into "html"
  links <- getHTMLLinks(html)
```

```
  # Get the links from HTML file "html"
  num <- grep("user=",links)[1]
  ID <- str_extract(links[num],"=[[:alnum:]]+&")
  # Find scholar's ID from links.
  # The ID will follow "=" and be followed by "&".
  ID <- str_replace_all(ID,"=(.*)&","\\1")
  # Remove "=" and "&" from ID, "\\1" means retaining the (.*) part.
  # Notice that this function is done under the package "stringr".
  citationURL <- paste0("https://scholar.google.com",links[num])
  # Use ID to create a hyperlink towards the scholar's citaiton page.
  citationweb = readLines(citationURL)
  # Read HTML files from citationURL into "citationweb"
  info <- c(citationweb,ID)
  return(info)
}
```

0.4 2. b)

```
papers <- function(html){
  # Observe the html file (We could open it on browser like Chrome),
  # the articles' name are encompassed by "gsc_a_at"" and "</a><div"
  title <- str_extract_all(html,"gsc_a_at\">.*?</a><div")
  # Extract the substrings satisfying the pattern.
  # (.*?) is the article's name, "?" is used to prevent greedy matching.
  title <- title[[1]]
  # x is a list with length 1, extract it.(Or using unlist function)
  title <- str_replace_all(title,"gsc_a_at\">(.*?)</a><div","\\1")
  # remove unnecessary strings.

  # Similar to titles, authors are encompassed by two "gs_gray"s
  author <- str_extract_all(html,"gs_gray\">.*?gs_gray\"")
  author <- author[[1]]
  author <- str_replace_all(author,"gs_gray\">(.*?)</div><div class=\"gs_gray\"","\\1")

  # Similarily, years are encompassed by "gsc_a_h">" and "</span>"
  year <- str_extract_all(html,"gsc_a_h\">[[:digit:]]{4}</span>")
  year <- year[[1]]
  year <- str_replace_all(year,"gsc_a_h\">([[:digit:]]{4})</span>","\\1")

  # Similarily, journal information are encompassed by "div><div class="gs_gray"> and "<span"
  journal <- str_extract_all(html,"div><div class=\"gs_gray\">.*?<span")
  journal <- journal[[1]]
  journal <- str_replace_all(journal,"div><div class=\"gs_gray\">(.*?)<span","\\1")

  # Unfortunately, the methods above cannot be used for citation times
  # since some of the numbers are deleted-lined and don't appear on html file.
  links <- readHTMLTable(html)
  # We use readHTMLTable() to deal with the problem.
  citation <- links[2]$gsc_a_t$`Cited by`
  # Extract citation times in links.
  # Some of the citation times have asterisks which we need to remove.
  citation <- as.character(citation)
```

```
  # Transform them to strings in order to be deal with.
  citation <- gsub("[*]","",citation)
  # Remove asterisks. Notice that "*" should be put in square brackets.

  info <- cbind(title,author,journal,year,citation)
  return(info)
}
```

0.5 2. c)

I modify the code in (a) and add the functionality to detect the invalid input names.

```
checkname <- function(Scholarname){
  # First we should check whether the name is a valid string.
  if(is.character(Scholarname) == FALSE ){
    warningsignal <- "Please enter a valid name."
    return (warningsignal)
  # Prompt a warning signal to remind user of error.

  # Followings are similar to (a)
  name <- gsub(" ","",Scholarname)
  URL <- paste0("https://scholar.google.com/citations?view_op=search_authors&mauthors=" ,name,"&hl=en&o
  html <- readLines(URL)
  links <- getHTMLLinks(html)
  num <- grep("user=",links)[1]
  citationURL <- paste0("https://scholar.google.com",links[num])

  # Then check if the input name is in the google scholar base,
  # If not, return an error message.
   if(!str_detect(S)(citationURL, "citations\\?user=")){
    warningsignal2 <- "There is no corresponding scholar profile based on your input."
    return(warningsignal2)
   }

  # Similar to remaining of (a)
  ID <- str_extract(links[num],"=[[:alnum:]]+&")
  ID <- str_replace_all(ID,"=(.*)&","\\1")
  citationweb = readLines(citationURL)
  info <- c(citationweb,ID)
  return(info)
}
```

Using testthat package to construct some cases to test whether the function works successfully or not.

```
library(testthat)

# Test if we can show error the message when an invalid input is entered.
test_that("Input is invalid", {
  expect_equal(checkfunction(427294), "Please enter a valid name.")
})

# Test whether the error message appears when there is no corresponding scholar.
test_that("URL is invalid",{
```

```
  expect_equal(checkfunction("lalallalala"), "There is no corresponding scholar profile based on your i
})

# Test if the result of the scholarID is correct.
test_that("ScholarID is incorrect",{
  expect_equal(readscholar("Geoffrey Hinton")[2], "JicYPdAAAAAJ")
  expect_equal(readscholar("Amanda Nickerson")[2], "16nAfxIAAAAJ")
})

# Test whether the table has five columns.
test_that("Correct column of the table",{
  expect_equal(ncol(papers("Geoffrey Hinton")),5)
})
```

0.6 2.d)

Similar to question (b), the difference is that we want to show all articles but only first 20. If we click the next page, the URL is somewhat different on the index, which we could use to acquire information from other pages. We use loop function to change the index of the URL till the last page.

```
fulltable <-function(Scholarname){
  index = 0
  # The index is the difference of each URL.
  ID <- readscholar(Scholarname)[2]
  # Get the ID of the scholar.
  signal <- TRUE
  # Used in while loop to check if the loop will continue or stop.
while(signal){

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

  # Similar to (b), create the information table.
  title <- str_extract_all(html,"gsc_a_at\">.*?</a><div")
  title <- title[[1]]
  title <- str_replace_all(title,"gsc_a_at\">(.*?)</a><div","\\1")

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

  year <- str_extract_all(html,"gsc_a_h\">[[:digit:]]{4}</span>")
  year <- year[[1]]
  year <- str_replace_all(year,"gsc_a_h\">([[:digit:]]{4})</span>","\\1")

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

  links <- readHTMLTable(html)
  citation <- links[2]$gsc_a_t$`Cited by`
  citation <- as.character(citation)
```

```
  citation <- gsub("[*]","",citation)

  info <- cbind(title,author,journal,year,citation)


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

  if(dim(info)[1]<20){
      signal <- FALSE
  # If the number of rows is smaller than 20, stop the while loop.
  }

  index <- index + 20
  # Add 20 to this index so that in next loop, we contruct a new URL.
  Sys.sleep(2)
  # Prevent Google from detecting automated usage.
}
  return(finalinfo)
}
```