STAT 243 PS 2

Junyuan Gao(SID:26484653)

September 16, 2017

1 Q1

a

```
## save letters in text format
chars <- sample(letters, 1e6, replace = TRUE)
write.table(chars, file = 'tmp1.csv', row.names = FALSE, quote = FALSE,
col.names = FALSE)
system('ls -l tmp1.csv', intern = TRUE)

## Error in system("ls -l tmp1.csv", intern = TRUE): 'ls' not found

## [1] "-rw-r--r-- 1 paciorek scfstaff 2000000 Sep 11 07:38 tmp1.csv" --(1)
chars <- paste(chars, collapse = '')
write.table(chars, file = 'tmp2.csv', row.names = FALSE, quote = FALSE,
col.names = FALSE)
system('ls -l tmp2.csv', intern = TRUE)

## Error in system("ls -l tmp2.csv", intern = TRUE): 'ls' not found

## [1] "-rw-r--r-- 1 paciorek scfstaff 1000001 Sep 11 07:38 tmp2.csv" --(2)
## save in binary format
nums <- rnorm(1e6)
save(nums, file = 'tmp3.Rda')
system('ls -l tmp3.Rda', intern = TRUE)

## Error in system("ls -l tmp3.Rda", intern = TRUE): 'ls' not found

## [1] "-rw-r--r-- 1 paciorek scfstaff 7678421 Sep 11 07:38 tmp3.Rda" --(3)
## save in text format
write.table(nums, file = 'tmp4.csv', row.names = FALSE, quote = FALSE,
col.names = FALSE, sep = ',')
system('ls -l tmp4.csv', intern = TRUE)

## Error in system("ls -l tmp4.csv", intern = TRUE): 'ls' not found
```

```
## [1] "-rw-r--r-- 1 paciorek scfstaff 18158912 Sep 11 07:38 tmp4.csv" --(4)
write.table(round(nums, 2), file = 'tmp5.csv', row.names = FALSE,
quote = FALSE, col.names = FALSE, sep = ',')
system('ls -l tmp5.csv', intern = TRUE)

## Error in system("ls -l tmp5.csv", intern = TRUE): 'ls' not found

## [1] "-rw-r--r-- 1 paciorek scfstaff 5379375 Sep 11 07:38 tmp5.csv" --(5)
```

Explanations for (1) to (5)
(1) In text format, each character cost 1 byte of storage, and since the seperate method is "Space" or "Return Carriage", this will also cost 1 byte of storage, so totally cost 2*1e6= 2000000 bytes.

(2) For similar reason in (1), each character take 1 byte and at the end the space take another 1 byte, so 1e6+1= 1000001

(3) In binary format, each number take 8 byte of storage, so 8*1e6= 8000000. However, save() function use gzip to compress the duplicated items in file, so the final result is 7678421 which is less than 8000000.

(4) In text format, one number takes at most 19 bytes(1 byte for digits, 1 byte for space, 1 byte for decimal point, 15 bytes for 15 decimals and 1 byte for negative sign) and at least 18 bytes(same as before without 1 byte for negative sign). Thus, the result 18158912 is between 18*1e6 and 19*1e6 since some but not all numbers are negative.

(5) Similar to 4, one number rounded to 2 decimals takes at most 6 bytes(1 byte for digits, 1 byte for space, 1 byte for decimal point, 2 bytes for 2 decimals and 1 byte for negative sign) and at least 5 bytes(same as before without 1 byte for negative sign). Thus, the result 5379375 is between 5*1e6 and 6*1e6 since some but not all numbers are negative.

b.

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

## Error in system("ls -l tmp6.Rda", intern = TRUE): 'ls' not found

## [1] "-rw-r--r-- 1 paciorek scfstaff 635237 Sep 11 07:38 tmp6.Rda"    ---(1)
chars <- rep('a', 1e6)
chars <- paste(chars, collapse = '')
save(chars, file = 'tmp7.Rda')
system('ls -l tmp7.Rda', intern = TRUE)

## Error in system("ls -l tmp7.Rda", intern = TRUE): 'ls' not found

## [1] "-rw-r--r-- 1 paciorek scfstaff 1056 Sep 11 07:38 tmp7.Rda"      ---(2)
```

explanation for (1) and (2)

(1) Similar to a(3), the upper bound of storage is 1e6. Since save() compress duplicated data in some ratio, the final result is 635237 that less than 1e6.

(2)The function save() use gzip as default compressing method, and gzip will have a lower compression ratio if the number of duplicates is small and have a higher compression ratio if the number of duplicate is big. Since in this case, the whole file is character "a", so the compression ratio is very high.

2 Q2

(a)

```
ScholarScraper <- function(userName){
  library(XML)
  library(RCurl)
  #Get the address of the searching page that match to particular scholar's name(input)
  address <- readLines(paste0("https://scholar.google.com/scholar?hl=en&q=",
              gsub(" ", "+", userName), "&btnG=&as_sdt=1%2C5&as_sdtp=&oq=g"))

  #Get all the links in this page, I find that the first link that have "?user=" pattern
  #has a scholar ID of that scholar inside
  #(e.g. "/citations?user=xT19JcOAAAAJ&hl=en&oe=ASCII&oi=ao")
  #save this as part of the address to the user Profile Page.
  temp = getHTMLLinks(address)
  idSegment <- temp[grep("?user=",temp)][1]

  #putting Sys.sleep(2) in between the calls that do the HTTP requests
  #to avoid 503 error(automated usage)
  Sys.sleep(2)

  #get link to scholar's profile page. htmlParse() returns a nicely formatted text of
  #scholar's page
  final_address <- readLines(paste0("https://scholar.google.com", idSegment))
  doc<- htmlParse(final_address,encoding="utf-8")

  #extract scholar ID from idSegment and combining the html text to make the output
  userID <- substr(idSegment, 17, 28)
  result <- c(userID, doc)
  return(result)
}
```

(b)

```
ScholarScraper2 <- function(userName){
  library(XML)
```

```r
library(RCurl)
library(rvest)

#Modified code from 2(a) with same usage
address <- readLines(paste0("https://scholar.google.com/scholar?hl=en&q=",
            gsub(" ", "+", userName), "&btnG=&as_sdt=1%2C5&as_sdtp=&oq=g"))
temp = getHTMLLinks(address)
idSegment <- temp[grep("?user=",temp)][1]
Sys.sleep(2)
final_address <- readLines(paste0("https://scholar.google.com", idSegment))
web<- htmlParse(final_address)

#create an empty data frame to store the user information
# system.sleep() same usage as above
df<- data.frame()
Sys.sleep(2)

#the title of paper comes with "gsc_a_at" class
#use this code to extract titles in the page
title1 <- getNodeSet(web, '//*[@class="gsc_a_at"]')
title2<- sapply(title1, xmlValue)

#Both authors and journal information of papers comes with "gs_gray" class
Sys.sleep(2)
info1 <- getNodeSet(web, '//*[@class="gs_gray"]')
info2 <- sapply(info1, xmlValue)

#Number of citations comes with "gsc_a_ac" class
Sys.sleep(2)
citeNum1<- getNodeSet(web, '//*[@class="gsc_a_ac"]')
citeNum2<- sapply(citeNum1, xmlValue)

#Year of publication comes with "gsc_a_h" class
Sys.sleep(2)
year1 <-getNodeSet(web, '//*[@class="gsc_a_h"]')
year2<- sapply(year1, xmlValue)

for (i in 1: length(title2)){
  # Since info2 stores both author and journal info
  # (author in odd index, journal info in even index)
  # extract them to 2 attributes
  author_index <- seq(1,length(info2),2)
  journal_index <- seq(2,length(info2),2)
  author2 <- info2[author_index]
  journal2 <- info2[journal_index]
```

```
    title= title2[i]
    author= author2[i]
    Journal_Info= journal2[i]
    Cite_Num= citeNum2[i]
    year= year2[i]

    tempdf <- data.frame(Title= title, Authors= author, Journal_Information= Journal_Info,
                         Number_of_Citation= Cite_Num, Publication_Year= year)
    # append new information after the existing array
    df <- rbind(df, tempdf)
  }
  return(df)
}
```

(c) I found two type of possible errors: (1)invalid input(not a string); (2) No such scholar page

```
testScraper <- function(userName){
  library(XML)
  library(RCurl)
  library(stringr)

  #Error detection code with error message for case (1)
    if(!is.character(userName)){
      return(paste("ERROR:",
                   " Invalid scholar name, please enter a string ",
                   "e.g. 'David Aldous' or 'Peter Bartlett' etc.", sep = ""))
    }


  # begin modified function in 2(a)

  address <- readLines(paste0("https://scholar.google.com/scholar?hl=en&q=",
              gsub(" ", "+", userName), "&btnG=&as_sdt=1%2C5&as_sdtp=&oq=g"))
  temp = getHTMLLinks(address)
  idSegment <- temp[grep("?user=",temp)][1]

  #Error detection for case (2)
  if(is.na(idSegment) | ! str_detect(idSegment, "user=")){
    #if the profile page did not exist,
    #idSegment would either return NA
    #or some other link(not profile page link).
    return(paste0("ERROR: Can't find the profile for ",
          userName, ". In addition:",
    "There may not be a profile with the input name."
```

```
    ))
  }

  Sys.sleep(2)
  final_address <- readLines(paste0("https://scholar.google.com", idSegment))
  doc<- htmlParse(final_address,encoding="utf-8")
  userID <- substr(idSegment, 17, 28)
  result <- c(userID, doc)
  return(result)
}
```

Now I write my test cases:

```
#test 2(a) and 2(c)
library(testthat)
#test 2(a)
expect_equal(ScholarScraper("Geoffrey Hinton")[[1]][1], "JicYPdAAAAAJ")
#test 2(c)
expect_equal(ScholarScraper("Geoffrey Hinton")[[1]][1], "JicYPdAAAAAJ")
expect_equal(testScraper("Leo Neymar"),
paste0("ERROR:",
" ERROR: Can't find the profile for Leo Neymar. ",
"In addition: There may not be a profile with the input name.")
expect_equal(testScraper(123)),
paste0("ERROR: Invalid scholar name, please enter a string ",
              "e.g. 'David Aldous' or 'Peter Bartlett' etc.")

#test 2(b)
test<- ScholarScraper2("Peter Bartlett")
head(test)

#output:
#Title
#1   Boosting the margin: A new explanation for the effectiveness of voting methods
#2   New support vector algorithms
#3   Learning the kernel matrix with semidefinite programming
#4   Neural network learning: Theoretical foundations
#5   Rademacher and Gaussian complexities: Risk bounds and structural results
#6   The sample complexity of pattern classification with neural networks: the size of the
#                                                                    Authors
#1                      RE Schapire, Y Freund, P Bartlett, WS Lee
#2       B Sch<c3><b6>lkopf, AJ Smola, RC Williamson, PL Bartlett
#3 GRG Lanckriet, N Cristianini, P Bartlett, LE Ghaoui, MI Jordan
#4                                          M Anthony, PL Bartlett
#5                                       PL Bartlett, S Mendelson
```

```
#6                                                    PL Bartlett
#                                              Journal_Information Number_of_Citation Publica
#1              The annals of statistics 26 (5), 1651-1686, 1998                2746
#2                    Neural computation 12 (5), 1207-1245, 2000                2569
#3     Journal of Machine learning research 5 (Jan), 27-72, 2004                2298
#4                                cambridge university press, 2009                1240
#5   Journal of Machine Learning Research 3 (Nov), 463-482, 2002                1021
#6 IEEE transactions on Information Theory 44 (2), 525-536, 1998                 893

## Error:  <text>:11:1:  unexpected symbol
## 10:  "In addition:  There may not be a profile with the input name.")
## 11:  expect_equal
##    ^
```