Stats243 Problem Set 2

Mengling Liu

Sept 2017

Collaboration: I worked with Rui Chen and Fan Dong for this homework set.

1-a. Storage in ASCII plain text versus binary formats

The file size is 2000000 because sample function generates 10^6 letters as one vector and save into chars. Then write.table saves the chars into a csv file called tmp1.csv which contains 10^6 string characters and each one containing one letter. Each string is also followed by a change of row so that there are 10^6 rows in tmp1.csv. Each change of row occupies one byte as well as the letter. So there are in total 2×10^6 bytes.

The file size is 1000001 because paste function here paste the chars generated in a into one string and save into chars. Then write.table saves the chars into a csv file called tmp2.csv which contains one string character with 10^6 letters. There is one change of row at the end of the string. Each letter contains one byte as well as change of row so there are in total of $10^6 + 1 = 1000001$ bytes.

The file size is 7678109 because rnorm function generates 10^6 numbers following normal distribution. Save function turns nums to .Rda file i.e. a binary format which will load in R to global environment. It will save as 1000000 numbers with each number occupying about 7.6 bytes so the total file size mounts to 7.6×10^6 bytes.

The file size is 18160350 because nums generated are saved into csv text file, which contains 10^6 strings. Each string contains around 18 to 19 digits including negative signs, integer digit, decimal points with change of row and each digit is one byte. So in total around $18 \times 10^6 \times 1 = 18 \times 10^6$ bytes.

The file size is 5378678 because nums generated are rounded into two decimal points and saved into csv text file. There are total 10^6 strings and each string contains around 5 to 6 digits including negative signs, integer digit, decimal points with change of row and each digit is one byte. So in total around $5 \times 10^6 \times 1 = 5 \times 10^6$ bytes.

1-b. save function When use save function, the file is default set to compression when saving string to Rda format in R, so the file size is smaller. Because tmp7.Rda contains repetition of one character "a", so the compression level will be higher and therefore the file size much smaller than a variety of mixed letters.

```
#save(chars,file='tmp8.Rda',compress=FALSE)
#system('ls -l tmp8.Rda', intern=TRUE)
```

2. Google Scholar Problem a Output researcher's citation page and Google Scholar ID

```
ScholarsearchID<-function(firstname, lastname) {
#firstname <- "Geoffery"
#lastname <- "Hinton"
library(XML)
library(curl)
library(stringr)

#Call the function ScholarsearchID() by entering the first name
#and last name of the scholar
#Construct the valid search page URL by inserting first name
#and last name of the scholar
URL1_1 <- "http://scholar.google.com/scholar?q="
URL1 <- paste0(URL1_1,firstname,"+",lastname)
#URL1 <- "http://scholar.google.com/scholar?q=geoffrey+hinton"
#Read the HTML document and parse it into its components
scholar <- htmlParse(URL1)
#Extract the nodes in HTML with subnode name "//a" and have
#attribute "href"
listofnodes <- getNodeSet(scholar, "//a[@href]")
#Extract the corresponding value under attribute "href"
href <- sapply(listofnodes,xmlGetAttr,"href")

#Need to search for the scholar ID in the nodes column, where
#the corresponding value contains the scholar name. Pattern for
#scholar name is a string ending with last name of scholar.
#Pattern for scholar ID is a string starting with "user="
pattern <- paste0(lastname,"$")
results <- rep(0,length(href))
for (i in 1:length(href)){
  if (str_detect(sapply(listofnodes,xmlValue)[i], pattern) == TRUE){
    results[i] <- str_extract(href[i], 'user=[a-zA-Z0-9-_]+')
  }
}

#Extract the unique scholar ID and cut out the substring that
#only contains the ID characters
User <- unique(results[which(str_detect(results, 'user=[a-zA-Z0-9-_]+'
                                        )== TRUE)])
scholarID <- substring(User, 6, nchar(User))
```

```
#Contruct the URL for citation page by inserting the scholar ID
URL2_1 <- "https://scholar.google.com/citations?user="
URL2_2 <- "&hl=en&oi=ao"
URL2 <- paste0(URL2_1,scholarID,URL2_2)

#Read the citation page URL and parse it to HTML text
htmlraw <- readLines(URL2)
html<-htmlParse(htmlraw)

#Construct a list containing scholar ID and citaion page HTML text
return(list(scholarID=scholarID,html=html))

}
```

b Create R data frame that contains the article title, authors, journal information, year of publication and number of citation

```
Scholarinfo <- function(firstname,lastname ) {
#Call scholarinfo() function by entering first name and last name
#of scholar
#The function will call on the ScholarresearchID() function to
#obtain the citation HTML text
result <- ScholarsearchID(firstname,lastname)

#Article information are under nodes that are named "tr"
#and have attribute class  i.e.NODE <- getNodeSet(result$html,
#"//tr[@class]"). Obtain artile title by locating subnodes that
#are named "a" and have attribute class='gsc_a_at' which is
#under nodes named "td" and have attribute class='gsc_a_t'
#Extract corresponding value using xmlValue to obtain the title
listofnodes2 <- getNodeSet(result$html, "//tr[@class]//
td[@class='gsc_a_t']//a[@class='gsc_a_at']")
articletitle <- sapply(listofnodes2,xmlValue)
#Obtain author name by locating first subnodes that are
#named "div" and have attribute class='gs_gray' which is
#under nodes named "td" and have attribute class='gsc_a_t'
#Extract corresponding value using xmlValue to obtain the
#author name
listofnodes3 <- getNodeSet(result$html, "//tr[@class]//
td[@class='gsc_a_t']//div[@class='gs_gray'][1]")
authors <- sapply(listofnodes3,xmlValue)
#Obtain journal information by locating second subnodes that
#are named "div" and have attribute class='gs_gray' which is
#under nodes named "td" and have attribute class='gsc_a_t'
#Extract corresponding value using xmlValue to obtain the
```

```
#journal information
listofnodes4 <- getNodeSet(result$html, "//tr[@class]//
td[@class='gsc_a_t']//div[@class='gs_gray'][2]")
journal <- sapply(listofnodes4,xmlValue)
#Obtain year by locating subnodes that are named "span" and
#have attribute class='gsc_a_h' which is under nodes named
#"td" and have attribute class='gsc_a_y'
#Extract corresponding value using xmlValue to obtain the year
listofnodes5 <- getNodeSet(result$html, "//tr[@class]
//td[@class='gsc_a_y']//span[@class='gsc_a_h']")
year <- sapply(listofnodes5,xmlValue)
#Obtain number of citation by locating subnodes that are
#named "a" and have either attribute class='gsc_a_ac' or
#class='gsc_a_ac gsc_a_acm' which is under
#nodes named "td" and have attribute class='gsc_a_c'
#Extract corresponding value using xmlValue to obtain
#the number of citation
listofnodes6 <- getNodeSet(result$html, "//tr[@class]
//td[@class='gsc_a_c']//a[@class='gsc_a_ac' or @class=
'gsc_a_ac gsc_a_acm']")
numcitation <- sapply(listofnodes6,xmlValue)

#Create a data frame with data in article title, authors,
#journal, year and number of citation
scholar.data <- data.frame(articletitle, authors, journal,
                           year, numcitation)
colnames(scholar.data) <- c('Article Title','Authors',
'Journal','Year','Number of Citation')
return(scholar.data)

}
```

c write some test code using testthat package

```
# Paste out the code from 2a first, and then
#insert assert_that statement to test the input names

ScholarsearchID2<-function(firstname, lastname) {

library(testthat)
library(XML)
library(curl)
library(stringr)
library(assertthat)
```

```
#If the input name contains characters other than
#letters, then output error message "Input first name
#and last name can only contain letters"
assert_that(str_detect(firstname, '^[A-Za-z]+$') |
              str_detect(lastname, '^[A-Za-z]+$'),
            msg="Input first name and last name can
            only contain letters")

URL1_1 <- "http://scholar.google.com/scholar?q="
URL1 <- paste0(URL1_1,firstname,"+",lastname)
scholar <- htmlParse(URL1)
listofnodes <- getNodeSet(scholar, "//a[@href]")
href <- sapply(listofnodes,xmlGetAttr,"href")

pattern <- paste0(lastname,"$")

results <- rep(0,length(href))
for (i in 1:length(href)){
  if (str_detect(sapply(listofnodes,xmlValue)[i],
                 pattern) == TRUE){
    results[i] <- str_extract(href[i],
                              'user=[a-zA-Z0-9-_]+')
  }
}

User <- unique(results[which(str_detect(results,
                      'user=[a-zA-Z0-9-_]+')== TRUE)])
scholarID <- substring(User, 6, nchar(User))

#If the returned scholar ID is empty, then return an
#error message "Please enter a valid scholar name"
assert_that(length(scholarID) != 0, msg=
              "Please enter a valid scholar name")

URL2_1 <- "https://scholar.google.com/citations?user="
URL2_2 <- "&hl=en&oi=ao"
URL2 <- paste0(URL2_1,scholarID,URL2_2)
URL2
htmlraw <- readLines(URL2)
html<-htmlParse(htmlraw)

return(list(scholarID=scholarID,html=html))

}
```

```
library(testthat)
#Use test_that to test below three situations:
#Situation 1: if the input name involves numbers,
#then should incur error message
test_that('if input first name and last name involves numbers',{
  firstname="abc123"
  lastname="abc345"
  expect_error(ScholarsearchID2(firstname,lastname))
}
)

#Situation 2: if the input name does not form a valid google
#scholar name, then should incur error message
test_that('if input first name and last name does not
form a valid google scholar name',{
  firstname="Mengling"
  lastname="Liu"
  expect_error(ScholarsearchID2(firstname,lastname))
}
)

#Situation 3: if the data frame is a 20*5 matrix table,
#check the dimention of the output, if same as 20*5, then
#expect to be equal
test_that('if the data frame is a 20*5 matrix table',{
  firstname="Geoffrey"
  lastname="Hinton"
  expect_equal(dim(Scholarinfo(firstname,lastname))[1],20)
  expect_equal(dim(Scholarinfo(firstname,lastname))[2],5)
}
)
```

```
#Build in a while loop in the function to loop over the
#pagesize in URL.
#n here in the function represents the total number of
#articles want to output.

Scholarinfonew<-function(firstname, lastname, n) {

library(XML)
library(curl)
library(stringr)
library(assertthat)

assert_that(str_detect(firstname, '^[A-Za-z]+$') |
```

```
              str_detect(lastname, '^[A-Za-z]+$'),
            msg="Input first name and last name can
            only contain letters")

URL1_1 <- "http://scholar.google.com/scholar?q="
URL1 <- paste0(URL1_1,firstname,"+",lastname)
scholar <- htmlParse(URL1)
listofnodes <- getNodeSet(scholar, "//a[@href]")
href <- sapply(listofnodes,xmlGetAttr,"href")

pattern <- paste0(lastname,"$")

results <- rep(0,length(href))
for (i in 1:length(href)){
  if (str_detect(sapply(listofnodes,xmlValue)[i],
                 pattern) == TRUE){
    results[i] <- str_extract(href[i],
                              'user=[a-zA-Z0-9-_]+')
  }
}

User <- unique(results[which(str_detect(results,
                      'user=[a-zA-Z0-9-_]+')== TRUE)])
scholarID <- substring(User, 6, nchar(User))

#If the returned scholar ID is empty, then return an
#error message "Please enter a valid scholar name"
assert_that(length(scholarID) != 0, msg=
              "Please enter a valid scholar name")

#While loop starts here to loop over "csstart" and "pagesize"
#in the URL.
#n here represents the total number of articles need to output.
x <- 0
articletitleall <- 0
authorsall <- 0
journalall <- 0
yearall <-0
numcitationall <- 0

while (x < n){
URL3_1 <- "https://scholar.google.com/citations?user="
URL3_2 <- paste0("&hl=en&oi=ao&cstart=",x,"&pagesize=",x+100)
URL3 <- paste0(URL3_1,scholarID,URL3_2)
htmlrawnew <- readLines(URL3)
```

```
htmlnew <- htmlParse(htmlrawnew)
listofnodes_1 <- getNodeSet(htmlnew, "//tr[@class]//
td[@class='gsc_a_t']//a[@class='gsc_a_at']")
articletitle_d <- sapply(listofnodes_1,xmlValue)
articletitleall <- c(articletitleall, articletitle_d)
#print(articletitleall)
listofnodes_2 <- getNodeSet(htmlnew, "//tr[@class]//
td[@class='gsc_a_t']//div[@class='gs_gray'][1]")
authors_d <- sapply(listofnodes_2,xmlValue)
authorsall <- c(authorsall, authors_d)
#print(authorsall)
listofnodes_3 <- getNodeSet(htmlnew, "//tr[@class]//
td[@class='gsc_a_t']//div[@class='gs_gray'][2]")
journal_d <- sapply(listofnodes_3,xmlValue)
journalall <- c(journalall, journal_d)
#print(journalall)
listofnodes_4 <- getNodeSet(htmlnew, "//tr[@class]
//td[@class='gsc_a_y']//span[@class='gsc_a_h']")
year_d <- sapply(listofnodes_4,xmlValue)
yearall <- c(yearall, year_d)
#print(yearall)
listofnodes_5 <- getNodeSet(htmlnew, "//tr[@class]
//td[@class='gsc_a_c']//a[@class='gsc_a_ac' or @class=
'gsc_a_ac gsc_a_acm']")
numcitation_d <- sapply(listofnodes_5,xmlValue)
numcitationall <- c(numcitationall, numcitation_d)
print(numcitationall)
x <- x+100
}

scholar.dataall <- data.frame(articletitleall, authorsall,
                        journalall, yearall, numcitationall)
colnames(scholar.dataall) <- c('Article Title','Authors',
'Journal','Year','Number of Citation')
return(scholar.dataall)

}
```