# STAT243-PS2

Jinhui Xu

September 2017

## 1 other students

I discuss some problems with Xin Shi.

## 2 Question 1

### 2.1 (a)

When save letters in text format, every letter is stored as one byte.In the first r-chunk,1000000 letters and about 1000000 separators are stored while the second r-chunk only store 1000000 letters.So tmp1.csv is smaller.

```
 ## 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)

## [1] "-rw-r--r--  1 xujinhui  staff  2000000  9 15 09:23 tmp1.csv"
```

```
 ## save letters in text format
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)

## [1] "-rw-r--r--  1 xujinhui  staff  1000001  9 15 09:23 tmp2.csv"
```

As numbers are stored as 8 bytes in binary format,so 1000000 numbers takes about 8MB.

```
nums <- rnorm(1e6)
save(nums, file = 'tmp3.Rda')
system('ls -l tmp3.Rda', intern = TRUE)

## [1] "-rw-r--r--  1 xujinhui  staff  7678281  9 15 09:23 tmp3.Rda"
```

For example,123.232123 would take 10 bytes in ASCII.So a .csv is generally larger than .Rda(binary)

```
write.table(nums, file = 'tmp4.csv', row.names = FALSE, quote = FALSE,
col.names = FALSE, sep = ',')
system('ls -l tmp4.csv', intern = TRUE)

## [1] "-rw-r--r--  1 xujinhui  staff  18159317  9 15 09:23 tmp4.csv"
```

round(nums,2) makes every number shorter,so it saves many bytes in ASCII.

```
write.table(round(nums, 2), file = 'tmp5.csv', row.names = FALSE,
quote = FALSE, col.names = FALSE, sep = ',')
system('ls -l tmp5.csv', intern = TRUE)

## [1] "-rw-r--r--  1 xujinhui  staff  5376847  9 15 09:23 tmp5.csv"
```

### 2.2 (b)

According to r-chunk6 and r-chunk7, I find Rda is actually a compressed file. The option of save() shows that it can use gzip. bzip2 or xz to compress files.

```
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 xujinhui  staff  635215  9 15 09:23 tmp6.Rda"
```

```
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 xujinhui  staff  1056  9 15 09:23 tmp7.Rda"
```

## 3 Qustion 2

### 3.1 (a)

The first part of url is always https://scholar.google.com/, so I need to get the rest part of url.

First,I view the html element via chrome,and understand the structure of html.Then find the class corresponding to the researchers's citation page.It is in "h4 class". Finally just find url I need and paste two url.

The second part of url contains ID,so locate the ID and substr the url.

```
library(XML)
library(curl)
library(stringr)
```

```
get_html=function(name){
  url=gsub(" ","",paste("https://scholar.google.com/scholar?hl=en&q=",name))
  html=htmlParse(readLines(url))
  content=getNodeSet(html,'//h4[@class="gs_rt2"]')                    #get the class we need
if(content[1]=="NULL") {                                             #check whether the name exists
    result="Sorry,can not find scholar name"
    return(result)
```

```
  }
  else
  url_1=as.character(xmlAttrs(content[[1]][[1]]))                    #get the useful part of content
  urlfinal=gsub(" ","",paste("https://scholar.google.com/",url_1))
  userid=substr(url_1,str_locate(url_1,"user")[1,2]+2,str_locate(url_1,"&")[1,1]-1)  #locate the ID
  result=list(html=htmlParse(readLines(urlfinal)),userID=userid)
return(result)
}
name="Albert Einstein"
get_html(name)$userID

## Warning in readLines(url):  'https://scholar.google.com/scholar?hl=en&q=AlbertEinstein'
## Warning in readLines(urlfinal):  'https://scholar.google.com//citations?user=qc6CJjYAAAAJ&hl=en&oe=A

## [1] "qc6CJjYAAAAJ"
```

## 3.2 (b)

View the element of html and find exactly what we need.

The information about articles is contained in "tr class". Therefore extract that part and store them in vector.

In final transfer five vectors into a dataframe.

```
get_article_inf=function(html){
  content=getNodeSet(html,'//tr[@class]')        #get the part we need
  article_title=c()
  authors=c()
  journal_information=c()
  citations_number=c()
  year_publication=c()
  for(i in 1:20){                                #store data in vectors
    article_title[i]=xmlValue(content[[i]][[1]][[1]])                          #titles
    authors[i]=xmlValue(content[[i]][[1]][[2]])                                #authors
    journal_information[i]=xmlValue(content[[i]][[1]][[3]][[1]])               #journal
    year_publication[i]=gsub(",","",xmlValue(content[[i]][[1]][[3]][[2]]))     #year of publication
    citations_number[i]=xmlValue(content[[i]][[2]][[1]])                       #number of citations
    data=cbind(article_title,authors,journal_information,year_publication,citations_number)
    data=as.data.frame(data)                     #transfer into dataframe
  }
return(data)
}
```

```
name="Albert Einstein"              #try the function.but the dataframe is large, so do not show result
get_article_inf(get_html(name)$html)
```

## 3.3 (c)

I test whether the id I get from the first function is right and whether the function can give a graceful recommend when the input is invalid.

```
library(testthat)
test_that("wrong input",{
expect_equal(get_html('abc'),"Sorry,can not find this name in Google Scholar")
expect_equal(get_html('Albert Einstein'),"qc6CJjYAAAAJ")
}
)
```

### 3.4 (d)

Through the network option of the html source, I find that the request of Show more just add "&cstart=20&pagesize=80" to url.Therefore, loop is able to help me get all information of articles. Let pagesize=20.

```
get_all_article=function(name){
  article_title=c()                          #     set five vectors to store data
  authors=c()
  journal_information=c()
  citations_number=c()
  year_publication=c()
  url0=gsub(" ","",paste("https://scholar.google.com/scholar?hl=en&q=",name))   #get initial url
  html0=htmlParse(readLines(url0))
  content0=getNodeSet(html0,'//h4[@class="gs_rt2"]')
    if(content0[1]=="NULL") {                                                    #check whether the name exists
      result="Sorry,can not find scholar name"
      return(result)
    }
  url_1=as.character(xmlAttrs(content0[[1]][[1]]))
  urlfinal=gsub(" ","",paste("https://scholar.google.com/",url_1))   #get url that can get 20 articles
  length=20;j=0;
  #set length=length(content),when it is smaller than 20,show that we have get all data
  while(length==20){
  #every time we get 20 new articles data
    url=gsub(" ","",paste(urlfinal,"&cstart=",20*j,"&pagesize=20"))
    html=htmlParse(readLines(url))
    content=getNodeSet(html,'//tr[@class]')
   length=length(content)
    for(i in 1:length){                                                  #store the data
      article_title[i+20*j]=xmlValue(content[[i]][[1]][[1]])
      authors[i+20*j]=xmlValue(content[[i]][[1]][[2]])
      journal_information[i+20*j]=xmlValue(content[[i]][[1]][[3]][[1]])
      year_publication[i+20*j]=gsub(",","",xmlValue(content[[i]][[1]][[3]][[2]]))
      citations_number[i+20*j]=xmlValue(content[[i]][[2]][[1]])
      }
  j=j+1
  }
data=cbind(article_title,authors,journal_information,year_publication,citations_number)
data=as.data.frame(data)
return(data)
}
```