PS2

Hangyu Huang

14 September 2017

```
chars <- sample(letters, 1e6, replace = TRUE)
write.table(chars, file = 'tmp1.csv', row.names = FALSE, quote = FALSE, col.names = FALSE)
```

Save letters in text format 1

In the tmp1.csv, there are 1e6 strings. Each string has one letter (1byte). Each string has a character %%/n for newline. Therefore the total bytes is 2x1e6=2e6

```
chars <- paste(chars, collapse = '')
write.table(chars, file = 'tmp2.csv', row.names = FALSE, quote = FALSE, col.names = FALSE)
```

Save letters in text format 2

In chars <- paste(chars, collapse = ‘’), collapse =”means it becomes 1 string instead of 1e6 strings. so the total size is 1x1e6+1=1000001

```
nums <- rnorm(1e6)
save(nums, file = 'tmp3.Rda')
```

Save letters in binary format 1

As the numbers are saved in binary base, each number uses around 8 bytes. therefore, the total bytes are around 8x1e6=8000000

```
write.table(nums, file = 'tmp4.csv', row.names = FALSE, quote = FALSE, col.names = FALSE, sep = ',')
sum(nchar(nums))
```

```
## [1] 17158719
```

Save letters in text format 3

The total character of ‘muns’ is 17160347. As in the text file, each character has 1 bytes. and every ‘,’ need 1 bytes. Therefore, the total bytes are around 17160347+1e6=18160347

```
write.table(round(nums, 2), file = 'tmp5.csv', row.names = FALSE, quote = FALSE, col.names = FALSE, sep
```

Save letters in text format 4

round(nums, 2) means all the numbers in ‘nums’ has only 2 decimal places. Therefore the bytes of each number is 4bytes. As a result, the total bytes will be 4x1e6+1x1e6 = 5e6(approximately)

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

saves objects using save()

All 1e6 letters are saved as one string without line break. The byte size should be 1000001. However, 'chars' are saved as tmp6.Rda using save(). Based on the usage of save(..., list = character(), file = stop(" 'file' must be specified"), ascii = FALSE, version = NULL, envir = parent.frame(), compress = isTRUE(!ascii), compression_level, eval.promises = TRUE, precheck = TRUE), we can see that the default value of ASCII is false, and the compress is true when it is not ASCII. Therefore, the size is smaller than 1000001 depanding on how well it compresses.Defaults to 6 for gzip compression.

```
chars <- rep('a', 1e6)
# 1e6 of "a" "a"
chars <- paste(chars, collapse = '')
#"a....a" total 1e6 of a
save(chars, file = 'tmp7.Rda')
```

saves objects using save() with repeated characters

In the file tmp7.Rta, the string is "aaaaa...aaaaa", total 1e6 of a. As shown in the previous paragraph, save() will compress it automatically. And the compress level will be much higher if the characters are repeated.

```
##example URL:
##https://scholar.google.com/citations?view_op=search_authors&mauthors=Geoffrey+Hinton&hl=en&oi=ao
## https://scholar.google.com/citations?user=JicYPdAAAAAJ&hl=en&oi=ao
Citations <- function(name){
baseURL <- "http://scholar.google.com/"
Name <- sub("\ ", "+", name)
filter <- paste0(Name, "&hl=en&oi=ao")
args1 <- "citations?view_op=search_authors&mauthors="
url <- paste0(baseURL, args1, filter)
library(XML)
scholar <- htmlParse(url)
listofANode <- getNodeSet(scholar, "//h3[@class]")
head(listofANode)
a <- sapply(listofANode, xmlChildren)
args2 <- sapply(a, xmlGetAttr, "href")
args2
args3 <- sub("ASCII", "ao", args2)
args3
url4 <- paste0(baseURL, args3)
url4
citation <- htmlParse(url4)
ID <- sub(".*user=","\\1", args2)
ID <- sub("&.*", "\\1", ID)
cat('Google Scholar ID:', ID)
print(url4)
print(citation)
}
```

The way to approch this problem

1. To find the special pattern of the url of citation page
2. Use sub() and paste0()to get the correct url for the citation page1.
3. find the Google scholar ID in the html file of the citation page1.

4. Use getNodeSet() and sapply() to get the google ID and get the final citation url folloring step2
5. print out the citation html and Google scholar ID.

```
table <- function(url4)
{
doc <- htmlParse(url4)
table <- getNodeSet(doc, "//td[@class='gsc_a_t']")
head(table)
CITATIONS <-sapply(table, xmlValue)[1:20]
class(CITATIONS)
CITATIONS
require(plyr)
dataframe <- read.delim(textConnection(CITATIONS),header = FALSE, sep = ",", strip.white = TRUE, fill =
dataframe
colnames(dataframe) <- c('title', 'author', 'journal information', 'year')
table2 <- getNodeSet(doc, "//td[@class='gsc_a_c']")
table2
Nocitations <- sapply(table2,xmlValue)
Nocitations
NoCitation <- data.frame(list(Nocitations), row.names = NULL)
colnames(NoCitation) <- c('No of Citations')
dataframe
NoCitation
citation_list <- cbind(dataframe, NoCitation)

print(citation_list)
}
```

the way to approch this problem

1. use getnodeset() to download the information of citations.
2. then use sapply() to apply function xmlvalue to it to find the first five values.
3. USe read.delim() to read the file in table format and creat the data frame from it.
4. change the column names

Q2.3

Q2.4

https://scholar.google.com/citations?user=JicYPdAAAAAJ&hl=en&oi=ao&cstart=100&pagesize=100

the main difference is the url. “&cstart=100&pagesize=100” is the additional pattern for “show more” button.