

Problem Set 1

Natalia Sarabia Vasquez

septiembre 07, 2021

1 Problem 1

Summary.

Memory hierarchy: Layers of memory: CPU can directly fetch instructions from cache memory. located on the processor chip. Cache memory must be loaded in from the main memory system (i.e. RAM). RAM only retains its contents when power is on.

Fastest speed - Cache Memory Slowest speed - Disk Memory

There is a trade off between speed and size, the faster the memory, the smallest it is.

Cache in depth Is one of the most important elements of CPU. There are both economic and physical limits to its maximum size.

2 Problem 2

2.1 Problem 2a

My first attempt is based on pure intuition. I created one row of the matrix, converted it to a data frame and exported into a csv file. Then, I checked the output's size. I thought that each row in a file had the same "contribution" to the total size of a CSV file.

```
# First attempt:
rows <- 1
# Create a dataframe with one row
df <- as.data.frame(matrix(runif(10*rows,0,100), nrow = rows, ncol = 10)) %>%
# Export and check the size
write_csv(file.path '..', 'PS1/data', 'df.csv'))
```

The CSV file has a size of 1KB. So I used a rule of three and the fact that 1MB = 1000KB to determine the number of rows I will need in order to have a CSV with size 100MB:

1 row → 1KB → 0.001MB

x rows → xKB → 100MB

And concluded that I needed $1e5$ rows:

```
# First attempt (continuation):
rows <- 1e5
df <- as.data.frame(matrix(runif(10*rows,0,100), nrow = rows, ncol = 10)) %>%
```

```
write_csv(file.path('..', 'PS1/data', 'df.csv'))
```

Unfortunately, my reasoning was not completely accurate. The final file had a size of 17.759MB, far less from 100MB.

In R, I encountered the function `object.size()`. This function provides an estimate of the memory that is being used to store an R object. I will try to understand first how R manages memory for objects. Probably, with my first attempt I was not looking the big picture.

I will generate different matrices, each with a different size (from 1 row, to 100 rows). Then, I will check if the relationship between size and the number of rows is steady (I guess not, otherwise I would have finished by doing my first attempt). I am also interested in finding a certain number from which the relationship is almost linear:

```
# Second attempt:
rows <- 100
sizes <- data.frame(matrix(NA,nrow=rows,ncol=2))
names(sizes) <- c("rows","size")

for(i in seq_len(rows)){
  m <- matrix(runif(10*i,0,100), nrow = i, ncol = 10)
  df <- data.frame(m)
  sizes[i,1] <- nrow(m)
  sizes[i,2] <- object.size(df)
}

sizes <- sizes %>%
  mutate(lag_size=lag(size),
         size_growth=size-lag_size) %>%
  select(rows,size, size_growth)
```

I built a quick graph of the size of each object versus the number of rows it has to have of better understanding of these apparent (random) numbers:

```
sizes %>%
  ggplot(aes(x=rows,y=size)) +
  geom_line()
```

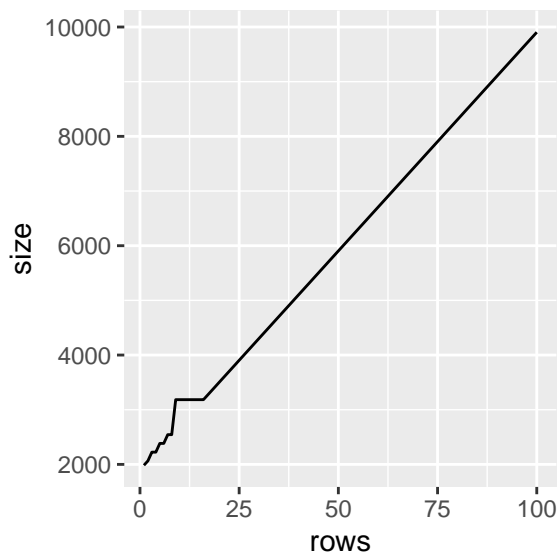


Figure 1: We show the size of an object in terms of the rows it has.

After seeing this graph, I realized that: 1. After a number of rows, the size is (almost) a linear function. 2. the size of the dataframe in R is not the same that the one of the output 3. the size of one row in the csv is not equal to the size of the file divided by the number of rows

So, my first reasoning was not totally wrong. I just needed more trials. The function stabilizes after 20 rows. Then, I tried with 10000 rows:

```
# Third attempt:
rows <- 100
df <- as.data.frame(matrix(runif(10*rows,0,100), nrow = rows, ncol = 10)) %>%
write_csv(file.path '..', 'PS1/data', 'df.csv'))
```

The CSV output has a size of 1776KB. We now can use the rule of three approach described earlier:

```
# Third attempt:
rows_needed <- (100*100) / (18*0.001)
```

I got that the number of rows should be between 555555 and 555556.

```
# Third attempt:
rows <- 555555
df <- as.data.frame(matrix(runif(10*rows,0,100), nrow = rows, ncol = 10)) %>%
write_csv(file.path '..', 'PS1/data', 'df.csv'))
```

2.2 Problem 2b

Reading just the first 100,000 observations:

```
#Using read.csv()
system.time(read.csv(file.path '..', 'PS1/data', 'df.csv'),nrows=100000))
```

```
## user system elapsed
## 2.27 0.22 2.65
```

After reading it at least four consecutive times, I found that the first time was the slowest.

Now, passing the type of columns as a parameter, considerably decreases the amount of time it takes to read the first 100,000 lines:

```
#Using read.csv()
system.time(read.csv(file.path('..', 'PS1/data', 'df.csv'), nrow=100000, colClasses=rep("numeric", 10)))

##      user      system elapsed
##      1.38       0.14       1.77
```

2.3 Problem 2c

I read the third chunk of 100,000 observations from the file. I found that the time it takes in order to read each chunk really depends on the chunk one is reading. My computer took less time on reading the first chunk than the third one. It is not an efficient way of reading the file.

```
#Using read.csv()
system.time(read.csv(file.path('..', 'PS1/data', 'df.csv'), nrow=100000, skip=300000, colClasses=rep("numeric", 10)))

##      user      system elapsed
##      2.80       0.45       3.55
```

2.4 Problem 2d

If we read the data using a connection, reading each of the chunks takes almost the same time. This is because, we already opened a connection. The time it takes is lesser than compared with incise c.

```
jump <- 555555/111111

con <- file(file.path('..', 'PS1/data', 'df.csv'), "r")

for(i in seq_len(jump)){
  print(system.time(read.csv(con, nrow=111111)))
}

##      user      system elapsed
##      3.05       0.19       3.44
##      user      system elapsed
##      1.67       0.19       1.96
##      user      system elapsed
##      1.71       0.08       1.91
##      user      system elapsed
##      2.12       0.17       2.54
##      user      system elapsed
##      2.43       0.21       3.36

close(con)
```

2.5 Problem 2e

R reserves space not only for the object but also to process it. This is the reason why, when applying the `object.size()` function we got a larger different number from the one of the exported file.

```
rows <- 555555
df <- as.data.frame(matrix(runif(10*rows, 0, 100), nrow = rows, ncol = 10)) %>%
save(file=file.path('..', 'PS1/data', 'df.rda'), compress=FALSE)
```



```

table <- "http://mldb.org/search?mq=yellow&si=2&mm=0&ob=1" %>%
  read_html() %>%
  html_nodes("#thelist") %>%
  html_table() %>%
  as.data.frame()

```

4.2 Problem 4b

The idea is to have a main function `getMyLyrics()` that calls some modular functions.

Algorithm:

1. Build the link for the search using the song that the user entered.
2. Follow that link, and webscrape the results of the search.
3. Form a table with 4 columns: the name of the song, the link of the song, the name of the artist and the link of the artist.
4. Search the song and the artist the user entered in the table of point 4.
5. Retrieve the value of the column `song_link` for the matching row (in case of more than one match, it only returns the first one).
6. Print the name of the album, the artist and the lyrics.

```

setLink <- function(baseURL,args1,args2,args3,args4,arg5){
  # Creates a URL to navigate http://mldb.org/
  # baseURL: Main website.
  # args1: "search?" if we are going to search a song,
  # args2: mq: name of the song

  # To navigate through the search options:
  # args3: si: 3-Fulltext, 2-Title, 1-Artist, 0-Artist and title
  # args4: mm: 2-Exact phrase, 1-Any word, 0-All words
  # args5: ob: 2-Rating, 1-Relevance

  URL <- paste0(baseURL,args1,args2,args3,args4,arg5)

  return(URL)
}

getData <- function(URL,element){
  # Retrieves the database that is a result of webscraping
  # URL: is the search on the website of the song x, by default it search in all words and by title
  # #thelist is the name of the table where the website saves the info of the search
  table <- URL %>%
    read_html() %>%
    html_nodes("#thelist") %>%
    html_nodes("tr") %>%
    html_node(element) %>%
    return(table)
}

getLyricsLink <- function(artist_name,song_name){
  # Obtains the link to search for a specific song from an author
  # artist_name is the name of the artist whose song one is searching for
  # song_name is the name of the song one is searching for

```

```

# Clean data. Makes the search non-case sensitive
artist_name <- artist_name %>%
  str_to_lower()

song_name<- song_name%>%
  str_to_lower()
# Concatenate the elements of the search and creates a link for the search
link <- setLink("http://mldb.org/", "search?", paste0("mq=",song_name,"&"),"si=2&","mm=0&","ob=1")

# Checks if the search returns one or more results.
# First, creates the data frame with the result of the search and then
# retrieves artist, artist's link, song and song's name.
if(url.exists(link)){

  artist <- getData(link,"td:nth-of-type(1)") %>%
    html_text() %>%
    trimws()

  song <- getData(link,"td:nth-of-type(2)") %>%
    html_text() %>%
    str_squish() %>%
    trimws()

  artist_link <- getData(link,"td a:nth-of-type(1)") %>%
    html_attr("href")

  song_link <- getData(link,"td:nth-of-type(2) a") %>%
    html_attr("href")

  # Clean the data, filter the target song
  lyricsLink <- data.frame(artist,artist_link,song,song_link) %>%
    mutate(artist = str_to_lower(artist),
           song = str_to_lower(song)) %>%
    filter(is.na(song)==FALSE,
           artist == artist_name,
           song == song_name) %>%
    select(song_link) %>%
    unique() %>%
    mutate(song_link=setLink("http://mldb.org/",song_link,"","","","")) %>%
    head(n=1) %>%
    pull()

  return(lyricsLink)

} else {
  return(-1) # In case the link does not exist, it returns -1
}

}

getSongInfo <- function(lyricsLink){
# Using the lyrics link, it retrieves the info of the artist, the album and the song.
  tables <- lyricsLink %>%

```

```

    read_html() %>%
    html_nodes("table")

info <- tables[[7]] %>%
    html_table()

artist <- info[1,2]
album <- info[2,2]

lyrics <- lyricsLink %>%
    read_html() %>%
    html_nodes("p") %>%
    html_text()

output<-list(artist,album,lyrics)
return(output)
}

getMyLyrics <- function(user_song,user_artist){
  #user_song = readline(prompt = "Enter the song's name : ")
  #user_artist = readline(prompt = "Enter the name of the artist : ")

  link = getLyricsLink(user_artist,user_song)

  if(length(link)>0){
    songInfo = getSngInfo(link)

    cat("The artist's name is:", pull(songInfo[[1]]), "\n")
    cat("The album where you can find this song is:", pull(songInfo[[2]]), "\n")
    cat("Enjoy your song! Here are the lyrics:", "\n")
    cat(as.vector(songInfo[[3]]),sep="\n")

  } else {
    cat("Look, I couldn't find info. Please, check the website and DIY!")
  }
}

# My function is not case sensitive!
getMyLyrics("yellow","COLdplay")

## The artist's name is: Coldplay
## The album where you can find this song is: Parachutes
## Enjoy your song! Here are the lyrics:
## Look at the stars,
## Look how they shine for you,
## And everything you do,
## Yeah, they were all yellow.
## I came along,
## I wrote a song for you,
## And all the things you do,
## And it was called "Yellow".
## So then I took my turn,

```



```
## Oh what a thing to have done,
## And it was all "Yellow."
## Jeg sked,
## Ååh ja jeg sked en bums,
## Turn into something beautiful,
## You know, you know I love you so,
## You know I love you so.
## I swam across,
## I jumped across for you,
## Oh what a thing to do.
## Cos you were all "Yellow",
## I drew a line,
## I drew a line for you,
## Oh what a thing to do,
## And it was all "Yellow."
## Jeg sked,
## Ååh ja jeg sked en bums,
## Turn into something beautiful,
## And you know,
## For you I'd bleed myself dry,
## For you I'd bleed myself dry.
## It's true,
## Look how they shine for you,
## Look how they shine for you,
## Look how they shine for,
## Look how they shine for you,
## Look how they shine for you,
## Look how they shine.
## Look at the stars,
## Look how they shine for you,
## And all the things that you do.
```

4.3 Problem 4c

```
# Same function as above, this time I consider the case when there is an exact match f
# or the response
getLyricsLink <- function(artist_name,song_name){
  artist_name <- artist_name %>%
    str_to_lower()

  song_name<- song_name%>%
    str_to_lower()

  link <- setLink("http://mldb.org/", "search?", paste0("mq=",song_name,"&"),"si=2&","mm=0&","ob=1")

  if((url.exists(link)) & (HEAD(link)$url == link)){

    artist <- getData(link,"td:nth-of-type(1)") %>%
      html_text() %>%
      trimws()

    song <- getData(link,"td:nth-of-type(2)") %>%
      html_text() %>%
```

```

    str_squish() %>%
    trimws()

    artist_link <- getData(link,"td a:nth-of_type(1)" ) %>%
      html_attr("href")

    song_link <- getData(link,"td:nth-of_type(2) a" ) %>%
      html_attr("href")

    lyricsLink <- data.frame(artist,artist_link,song,song_link) %>%
      mutate(artist = str_to_lower(artist),
             song = str_to_lower(song)) %>%
      filter(is.na(song)==FALSE,
             artist == artist_name,
             song == song_name) %>%
      select(song_link) %>%
      unique() %>%
      mutate(song_link=setLink("http://mldb.org/",song_link,"","","")) %>%
      head(n=1) %>%
      pull()

    return(lyricsLink)

  } else if((url.exists(link)) & (HEAD(link)$url != link)){
    return(HEAD(link)$url ) # This accounts for the case when there is one exact match
# resulting from the search. The link of the search redirects to the song one,
# so I checked if both links are the same.
  }
  else {
    return(-1)
  }
}

getMyLyrics <- function(user_song,user_artist){
  # I tried to read from console but RMarkdown does not allow this
#user_song = readline(prompt = "Enter the song's name : ")
#user_artist = readline(prompt = "Enter the name of the artist : ")

  #Managing some exemptions
  # Checking for invalid inputs
  if(suppressWarnings(is.na(as.numeric(user_song))==FALSE)){
    stopifnot(is.character(as.numeric(user_song)))
  }

  if(suppressWarnings(is.na(as.numeric(user_artist))==FALSE)){
    stopifnot(is.character(as.numeric(user_artist)))
  }

  if((getLyricsLink(user_artist,user_song))==1){
    cat("We don't have the lyrics of this song :(. Try again!")
    # Using stop would not allow to compile the complete pdf.
    #stop("We don't have the lyrics of this song :(. Try again!")
  }
}

```

```

} else if(length(getLyricsLink(user_artist,user_song))==0){
  link <- setLink("http://mldb.org/", "search?", paste0("mq=",user_song,"&"),"si=2&","mm=0&","ob=1")
} else {
  link = getLyricsLink(user_artist,user_song)
}

songInfo = getSngInfo(link)

cat("The artist's name is:", pull(songInfo[[1]]),"\n")
cat("The album where you can find this song is:", pull(songInfo[[2]]),"\n")
cat("Enjoy your song! Here are the lyrics:", "\n")
cat(as.vector(songInfo[[3]]),sep="\n")
}

```

Example of the function behavior when the result of the search is exact:

```
getMyLyrics("caLIifornication","red hot chilli peppers")
```

```

## The artist's name is: Red Hot Chili Peppers
## The album where you can find this song is: Californication
## Enjoy your song! Here are the lyrics:
## Psychic spies from China
## Try to steal your mind's elation
## Little girls from Sweden
## Dream of silver screen quotations
## And if you want these kind of dreams
## It's Californication
## It's the edge of the world
## And all of western civilization
## The sun may rise in the East
## At least it settles in the final location
## It's understood that Hollywood
## sells Californication
## Pay your surgeon very well
## To break the spell of aging
## Celebrity skin is this your chin
## Or is that war your waging
## Chorus:
## First born unicorn
## Hard core soft porn
## Dream of Californication
## Dream of Californication
## Marry me girl be my fairy to the world
## Be my very own constellation
## A teenage bride with a baby inside
## Getting high on information
## And buy me a star on the boulevard
## It's Californication
## Space may be the final frontier
## But it's made in a Hollywood basement
## Cobain can you hear the spheres
## Singing songs off station to station
## And Alderon's not far away

```

```
## It's Californication
## Born and raised by those who praise
## Control of population everybody's been there and
## I don't mean on vacation
## Chorus
## Destruction leads to a very rough road
## But it also breeds creation
## And earthquakes are to a girl's guitar
## They're just another good vibration
## And tidal waves couldn't save the world
## From Californication
## Pay your surgeon very well
## To break the spell of aging
## Sicker than the rest
## There is no test
## But this is what you're craving
## Chorus
```

Result of function when there is no match for the entered song:

```
getMyLyrics("dark necessities","red hot chilli peppers")
```

```
## We don't have the lyrics of this song :(. Try again!
```

```
## Error in getSngInfo(link): object 'link' not found
```

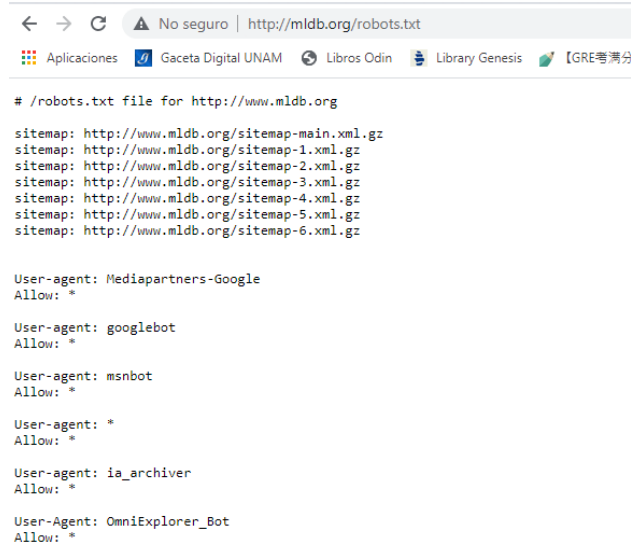
Result of the function when the user inputs' are not the correct type:

```
getMyLyrics("dark necessities","1")
```

```
## Error in getMyLyrics("dark necessities", "1"): is.character(as.numeric(user_artist)) is not TRUE
```

5 Problem 5

```
include_graphics(file.path('.', 'PS1/MLDb_robots.png'))
```



```

# /robots.txt file for http://www.mldb.org

sitemap: http://www.mldb.org/sitemap-main.xml.gz
sitemap: http://www.mldb.org/sitemap-1.xml.gz
sitemap: http://www.mldb.org/sitemap-2.xml.gz
sitemap: http://www.mldb.org/sitemap-3.xml.gz
sitemap: http://www.mldb.org/sitemap-4.xml.gz
sitemap: http://www.mldb.org/sitemap-5.xml.gz
sitemap: http://www.mldb.org/sitemap-6.xml.gz

User-agent: Mediapartners-Google
Allow: *

User-agent: googlebot
Allow: *

User-agent: msnbot
Allow: *

User-agent: *
Allow: *

User-agent: ia_archiver
Allow: *

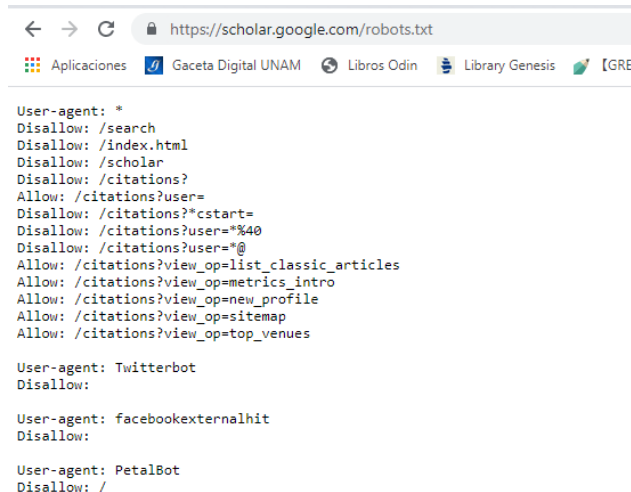
User-Agent: OmniExplorer_Bot
Allow: *

```

Figure 3: Are we allowed to webscrap the website?

After inspecting the robots.txt file (<http://mldb.org/robots.txt>), it seems that there is no restrictions regarding the content that can be accessed via webscraping. In our case, we are using a User-agent that is not one of the list, so we are allowed to access the info in the website.

```
include_graphics(file.path('.', 'PS1/Scholar_robots.png'))
```



```

User-agent: *
Disallow: /search
Disallow: /index.html
Disallow: /scholar
Disallow: /citations?
Allow: /citations?user=
Disallow: /citations?*cstart=
Disallow: /citations?user=%40
Disallow: /citations?user=%@
Allow: /citations?view_op=list_classic_articles
Allow: /citations?view_op=metrics_intro
Allow: /citations?view_op=new_profile
Allow: /citations?view_op=sitemap
Allow: /citations?view_op=top_venues

User-agent: Twitterbot
Disallow:

User-agent: facebookexternalhit
Disallow:

User-agent: PetalBot
Disallow: /

```

Figure 4: Are we allowed to webscrap the website?

In the case of Google Scholar (<https://scholar.google.com/robots.txt>) it all depends in the info we want to access and the user agent we use. For instance, the User-agent PetalBot cannot access any part of the website.

6 Problem 6

I will import some data in R, modify the data in Python and plot the resulting data in R:

```
library(reticulate)
library(datasets)

# Import data in R
data(iris)
X <- iris[,1:4]
y <- as.data.frame(as.character(iris[,5]))
names(y) <- c("var")

# Subsetting the element from R in Python and creating a new variable
z=(r.X["Petal.Length"])

# Plotting the result of the Python response
plot(py$z)
```

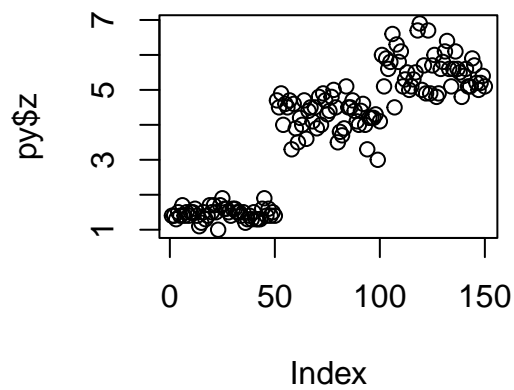


Figure 5: Python object graph in R