Problem Set 2

Natalia Sarabia Vasquez

September 17, 2021

1 Problem 1

1.1 Original functions

```
setLink <- function(arg1){
# Creates a URL to navigate http://mldb.org/
# baseURL: Main website.
# arg1: 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
  baseURL <- "http://mldb.org/search?mq="
  rest_options<- "&si=0&mm=0&ob=1"
  arg1 <- str_replace_all(arg1," ","+")
  URL <- paste0(baseURL,arg1,rest_options)

  assert_that(url.exists(URL), msg = "URL does not exist")

  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 searches
# in the All words and by title
# elements is the element type we want from the HMTL node.
# Possible values: "td a:nth-of_type(1)", "td:nth-of_type(2) a",
# "td:nth-of-type(2)", "td:nth-of-type(1)"
# #thelist is the name of the table where the website saves the info of the search
  assert_that(url.exists(URL),msg="URL does not exist")
  assert_that(is.string(element),msg="Element must be string, check admitted values")

  table <- URL %>%
    read_html() %>%
    html_nodes("#thelist") %>%
    html_nodes("tr")  %>%
    html_node(element) #%>%

  return(table)
}
```

```r
# I've already checked in the function that call this one,
# that the artist name and the song name are strings, so I will not check that again

getLyricsLink <- function(artist_name,song_name){
  artist_name <- artist_name %>%
    str_to_lower()

  song_name<- song_name%>%
    str_to_lower()

  link <- setLink(song_name)

# If the match is exact, the website redirects you to the page with the lyrics,
# If it is not exact, it takes you to the results. We will check what happens
# with the search and act accordingly

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

# Case 1. The page does not redirect, and the song does exist}

  if((not_empty(aux)==TRUE) & (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=paste0("http://mldb.org/",song_link)) %>%
      head(n=1) %>%
      pull()

    return(lyricsLink)
```

```r
# Case 2. The match is exact and the site redirects you.
# 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 if((HEAD(link)$url != link)){
    return(HEAD(link)$url )
  }
    else {
# Case 3: Any other case. It includes when the song does not exists
    return(-1)
  }

}

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

  # Check if the input is valid
  assert_that(url.exists(lyricsLink),msg="Link does not exist")

  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){
  # Managing some exemptions
  # Checking for invalid inputs
  assert_that(is.string(user_song))
  assert_that(is.string(user_artist))

  # In order to continue, check if the user has internet connection
  if(pingr::is_online()){
    # Case when the lyrics of this song does not exist
      if(getLyricsLink(user_artist,user_song)==-1){
        message("We don't have the lyrics of this song.")
    # Case when the user enters a song with exact match
```

```
      } else if(length(getLyricsLink(user_artist,user_song))==0){
        link <- setLink(user_song)

        songInfo = getSongInfo(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")
    # Case when there is more than one option for the song
      } else {
        link = getLyricsLink(user_artist,user_song)

        songInfo = getSongInfo(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 {
    warning("There is no internet connection. Try again later.")
  }
}
```

1.2 Tests

1.2.1 Tests for function setLink()

For this function, I would have added an assert_that() to check that input (song_name) is a string. However, as I call this function from the main function GetMYLyrics(), and this function handles the case when the input is different from an string, I decided not to add the assertion. It will be repetitive.

I also intended to add a test function that checked if the returned URL exists or not. It always exists. This is mainly because of the manner I programmed the function and also because of how the website handles its searches. The website searches for whatever you enter, and in the worst case, just return a message if the song is not found. However, this is not a problem for my function as I address the different responses I can get from the website in my function.

1.2.2 Tests for function getData()

Some tests that verify that the function does not work with invalid URLs or values for the argument element.

```
test_that("getData handles wrong inputs", {

  expect_error(getData("SDFs","td:nth-of-type(1)"),
               "URL does not exist")
  expect_error(getData("http://mldb.org/search?mq=%3C&si=0&mm=0&ob=1",4),
               "Element must be string, check admitted values",fixed=TRUE)
})
```

```
## Test passed
```

1.2.3 Tests for function getSongInfo()

Tests that check what could happen if the function recieves a wrong link.

```
test_that("getSongInfo handles wrong inputs", {
  expect_error(getSongInfo("123"),
               "Link does not exist")
})
```

```
## Test passed
```

1.2.4 Tests for function getMyLyrics()

Verification of the inputs of the main function, and what could happen if the song does not exist. I added a verification of internet connection in my function. Therefore, I did not add a new verification in the test.

```
test_that("getMyLyrics handles wrong inputs", {
# check for wrong input
 expect_message(getMyLyrics("","Coldplay"),
             "We don't have the lyrics of this song")
  expect_error(getMyLyrics(1,1),
               "user_song is not a string (a length one character vector).",fixed=TRUE)
  expect_error(getMyLyrics("dark necessities",1),
               "user_artist is not a string (a length one character vector).",fixed=TRUE)
# Check for messages when there is no such song
  expect_message(getMyLyrics("dark necessities","red hot chilli peppers"),
                 "We don't have the lyrics of this song.")
})
```

```
## Test passed
```

1.2.5 Tests for function getLyricsLink()

I did not add special tests for this function. I handle cases that can appear, in my own function. I did not test for the inputs to be strings, as this function is called in my main function that already tests that. There is not a unique data type that this function could return (I coded it in way to handle special cases). Therefore, I also did not add a test to check the type of the output as it may differ.

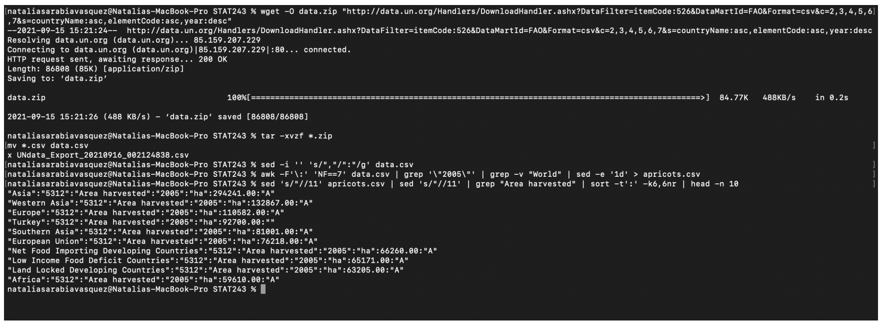
2 Problem 2

I tried many things in order to use bash commands embedded in my .Rmd. I tried using the option bash at the beginning of every chunk, but I had some problems with the quotations. I also tried to upload my job to Rcloud, but the command to unzip my file (which perfectly works in my computer) does not work there. For these reasons, I will be including images of how I entered the instructions in the command line and the results I got. I also will add, as comments, some of the commands I ran. I use some pipes and if I want to add a line break, the code will not work.

IMPORTANT. I acknowledge that I can improve some of my commands. However, I tried to follow the workflow instructed in the PS instructions. What I mean is that I can have finished in one line but instead decided to work step by step.

```
# Download the data for apricots
wget -O data.zip "http://data.un.org/Handlers/DownloadHandler.ashx?DataFilter=itemCode:526&DataMartId=F

# wget -O data.zip "http://data.un.org/Handlers/DownloadHandler.ashx?
# DataFilter=itemCode:526&DataMartId=FAO&Format=csv&c=2,3,4,5,6,7&s=
# countryName:asc,elementCode:asc,year:desc"
# Extract the data (it returns a .csv)
tar -xvzf *.zip
mv *.csv data.csv

# Change de delimiters of the file, from ',' to ':'
sed -i '' 's/","/":"/g' data.csv

# Exclude the metadata and the global total in another file
# Keep just the rows with 7 fields, this will remove the footnotes!
awk -F'\:' 'NF==7' data.csv | grep '\"2005\"' | grep -v "World" | sed -e '1d' > apricots.csv

# awk -F'\:' 'NF==7' data.csv | grep '\"2005\"' | grep -v "World" |
# sed -e '1d' > apricots.csv

# Based on the "area harvested" determine the ten regions/countries using
# the most land to produce apricots.
sed 's/"//11' apricots.csv | sed 's/"//11' | grep "Area harvested" | sort -t':' -k6,6nr | head -n 10

# sed 's/"//11' apricots.csv | sed 's/"//11' | grep "Area harvested" |
# sort -t':' -k6,6nr | head -n 10
```

```
include_graphics(file.path('..','PS2/PS2a.png'))
```

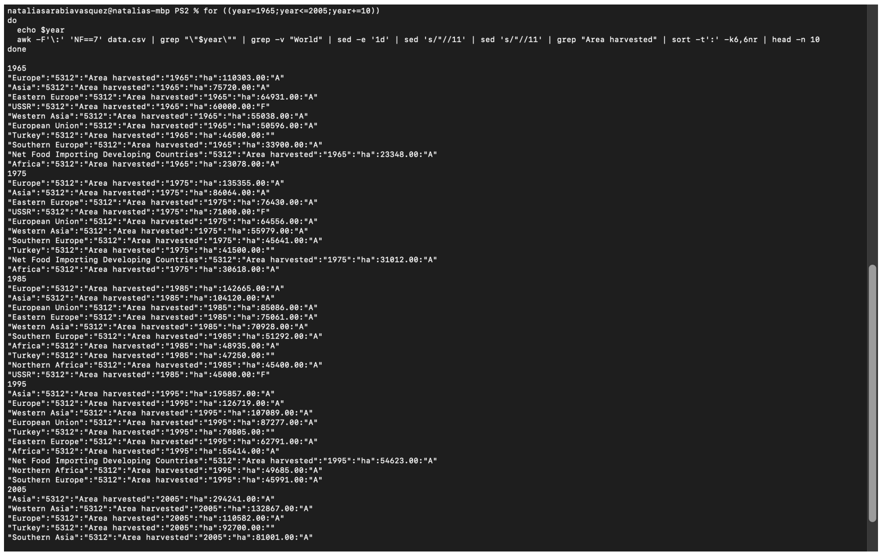


```
nataliasarabiavasquez@Natalias-MacBook-Pro STAT243 % wget -O data.zip "http://data.un.org/Handlers/DownloadHandler.ashx?DataFilter=itemCode:526&DataMartId=FAO&Format=csv&c=2,3,4,5,6
,7&s=countryName:asc,elementCode:asc,year:desc"
--2021-09-15 15:21:24--  http://data.un.org/Handlers/DownloadHandler.ashx?DataFilter=itemCode:526&DataMartId=FAO&Format=csv&c=2,3,4,5,6,7&s=countryName:asc,elementCode:asc,year:desc
Resolving data.un.org (data.un.org)... 85.159.207.229
Connecting to data.un.org (data.un.org)|85.159.207.229|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 86808 (85K) [application/zip]
Saving to: ‘data.zip’

data.zip                                100%[===========================================================================================>]  84.77K   488KB/s    in 0.2s

2021-09-15 15:21:26 (488 KB/s) - ‘data.zip’ saved [86808/86808]

nataliasarabiavasquez@Natalias-MacBook-Pro STAT243 % tar -xvzf *.zip
mv *.csv data.csv
x UNdata_Export_20210916_002124838.csv
nataliasarabiavasquez@Natalias-MacBook-Pro STAT243 % sed -i '' 's/","/":"/g' data.csv
nataliasarabiavasquez@Natalias-MacBook-Pro STAT243 % awk -F'\:' 'NF==7' data.csv | grep '\"2005\"' | grep -v "World" | sed -e '1d' > apricots.csv
nataliasarabiavasquez@Natalias-MacBook-Pro STAT243 % sed 's/"//11' apricots.csv | sed 's/"//11' | grep "Area harvested" | sort -t':' -k6,6nr | head -n 10
"Asia":"5312":"Area harvested":"2005":"ha":294241.00:"A"
"Western Asia":"5312":"Area harvested":"2005":"ha":132867.00:"A"
"Europe":"5312":"Area harvested":"2005":"ha":110582.00:"A"
"Turkey":"5312":"Area harvested":"2005":"ha":92700.00:""
"Southern Asia":"5312":"Area harvested":"2005":"ha":81001.00:"A"
"European Union":"5312":"Area harvested":"2005":"ha":76218.00:"A"
"Net Food Importing Developing Countries":"5312":"Area harvested":"2005":"ha":66260.00:"A"
"Low Income Food Deficit Countries":"5312":"Area harvested":"2005":"ha":65171.00:"A"
"Land Locked Developing Countries":"5312":"Area harvested":"2005":"ha":63205.00:"A"
"Africa":"5312":"Area harvested":"2005":"ha":59610.00:"A"
nataliasarabiavasquez@Natalias-MacBook-Pro STAT243 %
```

Figure 1: Bash output 2a)

```
# Now automate your analysis and examine the top five regions/countries for
# 1965, 1975, 1985, 1995, and 2005.

# Same idea as above, use a for to iterate through the years
for ((year=1965;year<=2005;year+=10))
do
  echo $year
  awk -F'\:' 'NF==7' data.csv | grep "\"$year\"" | grep -v "World" | sed -e '1d' | sed 's/"//11' | sed
done


# Inside of the for:
# awk -F'\:' 'NF==7' data.csv | grep "\"$year\"" | grep -v "World" |
# sed -e '1d' | sed 's/"//11' | sed 's/"//11' | grep "Area harvested" |
# sort -t':' -k6,6nr | head -n 10
```

```
include_graphics(file.path('..','PS2/PS2a.1.png'))
```



```
nataliasarabiavasquez@natalias-mbp PS2 % for ((year=1965;year<=2005;year+=10))
do
  echo $year
  awk -F'\:' 'NF==7' data.csv | grep "\"$year\"" | grep -v "World" | sed -e '1d' | sed 's/"//11' | sed 's/"//11' | grep "Area harvested" | sort -t':' -k6,6nr | head -n 10
done

1965
"Europe":"5312":"Area harvested":"1965":"ha":110303.00:"A"
"Asia":"5312":"Area harvested":"1965":"ha":75720.00:"A"
"Eastern Europe":"5312":"Area harvested":"1965":"ha":64931.00:"A"
"USSR":"5312":"Area harvested":"1965":"ha":60000.00:"F"
"Western Asia":"5312":"Area harvested":"1965":"ha":55038.00:"A"
"European Union":"5312":"Area harvested":"1965":"ha":50596.00:"A"
"Turkey":"5312":"Area harvested":"1965":"ha":46500.00:""
"Southern Europe":"5312":"Area harvested":"1965":"ha":33900.00:"A"
"Net Food Importing Developing Countries":"5312":"Area harvested":"1965":"ha":23348.00:"A"
"Africa":"5312":"Area harvested":"1965":"ha":23078.00:"A"
1975
"Europe":"5312":"Area harvested":"1975":"ha":135355.00:"A"
"Asia":"5312":"Area harvested":"1975":"ha":86064.00:"A"
"Eastern Europe":"5312":"Area harvested":"1975":"ha":76430.00:"A"
"USSR":"5312":"Area harvested":"1975":"ha":71000.00:"F"
"European Union":"5312":"Area harvested":"1975":"ha":64556.00:"A"
"Western Asia":"5312":"Area harvested":"1975":"ha":55979.00:"A"
"Southern Europe":"5312":"Area harvested":"1975":"ha":45641.00:"A"
"Turkey":"5312":"Area harvested":"1975":"ha":41500.00:""
"Net Food Importing Developing Countries":"5312":"Area harvested":"1975":"ha":31012.00:"A"
"Africa":"5312":"Area harvested":"1975":"ha":30618.00:"A"
1985
"Europe":"5312":"Area harvested":"1985":"ha":142665.00:"A"
"Asia":"5312":"Area harvested":"1985":"ha":104120.00:"A"
"European Union":"5312":"Area harvested":"1985":"ha":85086.00:"A"
"Eastern Europe":"5312":"Area harvested":"1985":"ha":75061.00:"A"
"Western Asia":"5312":"Area harvested":"1985":"ha":70928.00:"A"
"Southern Europe":"5312":"Area harvested":"1985":"ha":51292.00:"A"
"Africa":"5312":"Area harvested":"1985":"ha":48935.00:"A"
"Turkey":"5312":"Area harvested":"1985":"ha":47250.00:""
"Northern Africa":"5312":"Area harvested":"1985":"ha":45400.00:"A"
"USSR":"5312":"Area harvested":"1985":"ha":45000.00:"F"
1995
"Asia":"5312":"Area harvested":"1995":"ha":195857.00:"A"
"Europe":"5312":"Area harvested":"1995":"ha":126719.00:"A"
"Western Asia":"5312":"Area harvested":"1995":"ha":107089.00:"A"
"European Union":"5312":"Area harvested":"1995":"ha":87277.00:"A"
"Turkey":"5312":"Area harvested":"1995":"ha":70805.00:""
"Eastern Europe":"5312":"Area harvested":"1995":"ha":62791.00:"A"
"Africa":"5312":"Area harvested":"1995":"ha":55414.00:"A"
"Net Food Importing Developing Countries":"5312":"Area harvested":"1995":"ha":54623.00:"A"
"Northern Africa":"5312":"Area harvested":"1995":"ha":49685.00:"A"
"Southern Europe":"5312":"Area harvested":"1995":"ha":45991.00:"A"
2005
"Asia":"5312":"Area harvested":"2005":"ha":294241.00:"A"
"Western Asia":"5312":"Area harvested":"2005":"ha":132867.00:"A"
"Europe":"5312":"Area harvested":"2005":"ha":110582.00:"A"
"Turkey":"5312":"Area harvested":"2005":"ha":92700.00:""
"Southern Asia":"5312":"Area harvested":"2005":"ha":81001.00:"A"
```

Figure 2: Bash output 2a.1)

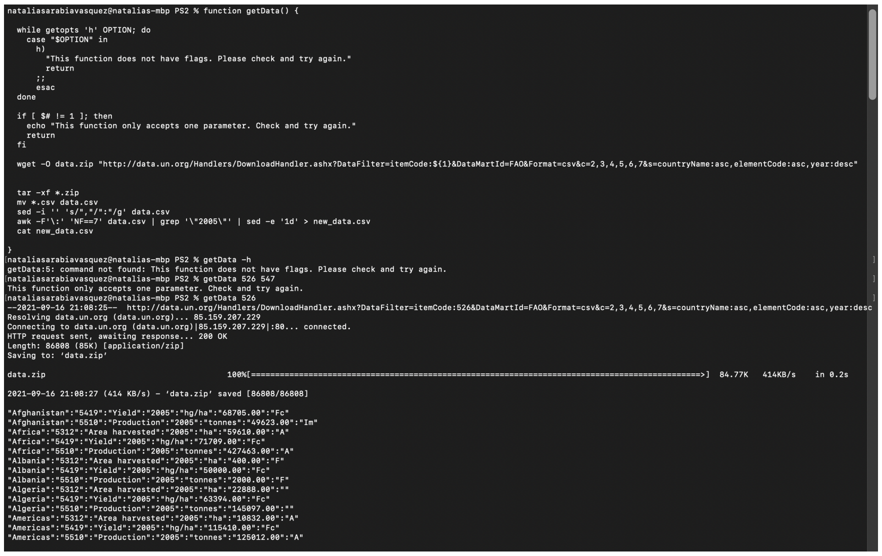
First, I handle the case when the user enters the flag -h. Then, I check if the number of arguments is different of one. Finally, I download, unzip, remove headers and metadata, change delimiters and print the data.

```
function getData() {

  while getopts 'h' OPTION; do
    case "$OPTION" in
      h)
        "This function does not have flags. Please check and try again."
        return
      ;;
      esac
  done

  if [ $# != 1 ]; then
    echo "This function only accepts one parameter. Check and try again."
    return
  fi

  wget -O data.zip "http://data.un.org/Handlers/DownloadHandler.ashx?DataFilter=itemCode:${1}&DataMartI

# wget -O data.zip "http://data.un.org/Handlers/DownloadHandler.ashx?Data
# Filter=itemCode:${1}&DataMartId=FAO&Format=csv&c=2,3,4,5,6,7&s=country
# Name:asc,elementCode:asc,year:desc"

  tar -xf *.zip
  mv *.csv data.csv
  sed -i '' 's/","/":"/g' data.csv
  awk -F'\:' 'NF==7' data.csv | sed -e '1d' > new_data.csv
  cat new_data.csv
```

```
}
```

I checked that my function returns a message if the flag -h is entered. It also checks if there is more than one argument. Finally, I attached the screenshot with part of the desired output (trying to add everything would include more than 2 screenshots, so I decided to just add the first lines.).

```
include_graphics(file.path('..','PS2/PS2b_total.png'))
```



```
nataliasarabiavasquez@natalias-mbp PS2 % function getData() {

  while getopts 'h' OPTION; do
    case "$OPTION" in
      h)
        "This function does not have flags. Please check and try again."
        return
      ;;
      esac
  done

  if [ $# != 1 ]; then
    echo "This function only accepts one parameter. Check and try again."
    return
  fi

  wget -O data.zip "http://data.un.org/Handlers/DownloadHandler.ashx?DataFilter=itemCode:${1}&DataMartId=FAO&Format=csv&c=2,3,4,5,6,7&s=countryName:asc,elementCode:asc,year:desc"


  tar -xf *.zip
  mv *.csv data.csv
  sed -i '' 's/","/":"/g' data.csv
  awk -F'\:' 'NF==7' data.csv | grep '\"2005\"' | sed -e '1d' > new_data.csv
  cat new_data.csv

}
nataliasarabiavasquez@natalias-mbp PS2 % getData -h
getData:5: command not found: This function does not have flags. Please check and try again.
nataliasarabiavasquez@natalias-mbp PS2 % getData 526 547
This function only accepts one parameter. Check and try again.
nataliasarabiavasquez@natalias-mbp PS2 % getData 526
--2021-09-16 21:08:25--  http://data.un.org/Handlers/DownloadHandler.ashx?DataFilter=itemCode:526&DataMartId=FAO&Format=csv&c=2,3,4,5,6,7&s=countryName:asc,elementCode:asc,year:desc
Resolving data.un.org (data.un.org)... 85.159.207.229
Connecting to data.un.org (data.un.org)|85.159.207.229|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 86808 (85K) [application/zip]
Saving to: ‘data.zip’

data.zip                                 100%[==================================================================================================>]  84.77K   414KB/s    in 0.2s

2021-09-16 21:08:27 (414 KB/s) - ‘data.zip’ saved [86808/86808]

"Afghanistan":"5419":"Yield":"2005":"hg/ha":"68705.00":"Fc"
"Afghanistan":"5510":"Production":"2005":"tonnes":"49623.00":"Im"
"Africa":"5312":"Area harvested":"2005":"ha":"59610.00":"A"
"Africa":"5419":"Yield":"2005":"hg/ha":"71709.00":"Fc"
"Africa":"5510":"Production":"2005":"tonnes":"427463.00":"A"
"Albania":"5312":"Area harvested":"2005":"ha":"400.00":"F"
"Albania":"5419":"Yield":"2005":"hg/ha":"50000.00":"Fc"
"Albania":"5510":"Production":"2005":"tonnes":"2000.00":"F"
"Algeria":"5312":"Area harvested":"2005":"ha":"22888.00":""
"Algeria":"5419":"Yield":"2005":"hg/ha":"63394.00":"Fc"
"Algeria":"5510":"Production":"2005":"tonnes":"145097.00":""
"Americas":"5312":"Area harvested":"2005":"ha":"10832.00":"A"
"Americas":"5419":"Yield":"2005":"hg/ha":"115410.00":"Fc"
"Americas":"5510":"Production":"2005":"tonnes":"125012.00":"A"
```

Figure 3: Bash output 2b)

3 Problem 3

3.1 Problem 3a)

I read Chapter 11 of Christensen et al. Transparent and Reproducible Social Science Research.

The document provides useful guidelines to have a reproducible workflow in social science research. I believe that most of the mentioned practices are useful in other fields as well. During my years of professional experience, I have been involved in projects that, because of its nature, were audited. These experiences taught me some good practices. For instance, mainly for organization, I keep separated directories for the output, code and "untouched data". Similarly, I try to keep a daily journal about any change I make. However, I personally find very compelling to follow some of the proposed suggestions. For instance, working in multidisciplinary teams makes it difficult (although I recognize the huge advantage it could bring) using tools as GitHub. From my point of view, researchers do not use many of these guidelines because most of the times the projects have a very long duration. Therefore, trying to follow all these steps tend to be cumbersome. I think the most used is the one related to organization and naming of directories. Also, keeping track of the version of R, packages and libraries may take some time. Maybe if the research does not require to do so, using github could be more oriented to software engineering than the most of the other recommendations.

3.2 Problem 3b)

Strengths:

- The files are well documented (they contain many useful comments). The parts of the code where the authors collected, cleaned, calculated various statistics and produced their plots is clear after reading the paper and the README file.
- The name of the functions and variables make the code easier to read.

- The code has proper indentation.
- Sometimes, they provide examples of how to use an specific function. I find this very useful.
- All modifications made on the data are well explained in GitHub. The authors present cleaned and prepared data, in case the user wants to start from an specific point or does not want to make some assumptions.
- They explain how to settle the file organization in order to the codes to run.
- The titles of each sections are clear enough to understand which part is trying to replicate.
- They included a readme file that has clear information about the process of reproducing the content of the paper.

Weaknesses:

- Personally, I find diagrams easier to read. I think that a diagram explaining the process could have been very helpful. An update to this, I was able to find the diagram later on but it is not in the main page of the GitHub repository.
- The distribution of the files in GitHub is not very well organized from my point of view. You are presented with a bunch of files that you then have to organize.
- The specific details of the versions of R and the packages used in the code are not given. Sometimes, if a function becomes deprecated it would be useful to know where it was before, and what was its name.
- Although they explain how the five directories and file paths must be set, I believe that a .zip could also have been very helpful. For instance, the ones created by the professor for the tutorials do not require to set up anything.
- People who is not very familiar with Github could have some problems finding the information.
- Some of the functions are susceptible to errors from the user. Enter a wrong type, etc.