Case Study 2

AKSTA Statistical Computing

Konstantinos Vakalopoulos 12223236

2023-05-11

Introduction

Before answering the tasks for the case study, the libraries dplyr and readr are introduced and are used for data manipulation and data import, respectively. Also, the working directory is set, which must be modified. Note that all the dplyr functions are used with the principal function provided by the magrittr package, which is %>%, or what's called the "pipe" operator.

```
library(dplyr)
library(readr)
library(tidyverse)
setwd("C:/Users/vaka1/Desktop/Case Study 2") #This must be modified
```

Task 1

From https://datahub.io/core/country-codes obtain the csv containing information related to the countries and load it in R. Keep only the columns containing official country name in English, ISO 3166 country codes with 2 and 3 alpha-numeric characters, development status (developing vs developed) and region and sub-region.

```
#read the csv
country_code <- read.csv("country-codes_csv.csv")

#str(country_code) #get an idea of the data

#select the proper columns
data_code <-country_code %>% select(official_name_en,ISO3166.1.Alpha.2,ISO3166.1.Alpha.3,
                     Developed...Developing.Countries, Region.Name, Sub.region.Name)
```

First, we read the csv file using the command read.csv() and pass it to the variable country_code. Using the function str(), we inspect the data. Due to problems during the knitting, the command str(country_code) has been commented. Finally, from the dplyr package, the function select() is used in order to keep only the columns: official country name in English, ISO 3166 country codes with 2 and 3 alpha-numeric characters, development status (developing vs developed) and region and sub-region.

Task 2

Load in R the following data sets which you can find in TUWEL. For each data set, ensure that missing values are read in properly, that column names are unambiguous. Each data set should contain at the end only two columns: country and the variable.

- rawdata_343.txt which contains the (estimated) median age per country. Pay attention! The delimiter is 2 or more white spaces (one space would not work as it would separate country names which contain a space); you have to skip the first two lines. Hint: you can look into function read.fwf or the readr corresponding function. It might also be useful to use tidyr functions to unite some columns back or separate them.

- rawdata_373.csv which contains the (estimated) youth unemployment rate (15-24) per country

```
#Read the txt file using the command read_fwf()
rawdata_343 <- read_fwf("rawdata_343.txt", fwf_empty("rawdata_343.txt"),
                               skip = 2)

#For convenient purposes, transform it into data frame
rawdata_343 <- as.data.frame(rawdata_343)

#Get an idea of the data
#str(rawdata_343)

#Keep the variables X2 and X5 (country name and median age) and change the name of X2 and X5
rawdata_343 <- rawdata_343 %>%
  select(X2,X5) %>%
  rename("Country_Names" = "X2", "Median_Age" = "X5")

#Change the name of X2 and X5
#names(rawdata_343) <- c("Country_Names","Median_Age")

#The median age is character column. Also the observation 198 has a problem
rawdata_343$Median_Age <- substr(rawdata_343$Median_Age, 1, 4)

#Transform it into numeric
rawdata_343$Median_Age <- as.numeric(rawdata_343$Median_Age)

#Read the csv
rawdata_373 <- read.csv("rawdata_373.csv")

#Remove spaces from the column country_name
rawdata_373$country_name <- trimws(rawdata_373$country_name)
```

First we read the txt file ("rawdata_343.txt") using the command read_fwf() from the readr library/package and assigned to the variable rawdata_343. The fwf_empty("rawdata_343.txt") is an argument specifying the width of each column in the file. The fwf_empty() function creates an empty fixed-width format specification for a file with the given name, and read_fwf() uses this specification to determine the width of each column. Furthermore, skip = 2 is an optional argument that tells read_fwf() to skip the first two lines of the file.

Then, for convenient purposes, the variable rawdata_343 is transformed into data frame and using the str() function, we get an idea about the data. Consequently, the columns X2 and x5 are kept, which they represent the country name and the median age, respectively. Also, the names of the 2 columns has been changed using the command rename() for the dplyr package. By observing the data, the column for the median age is a character column. Despite that, the element 198 is problematic. Thus, the function substr() is used for the observation 198 in order to keep only the 4 first characters, i.e. the median age. Finally, the column is transformed into numeric.

Lastly, we read the rawdata_373.csv and it is observed that the column country_name contains plenty of spaces. Thus, the trimws() function is used in order to remove the unnecessary spaces.

Task 3

Merge the two data sets containing raw data using dplyr function on the unique keys. Keep the union of all observations in the two tables. What key are you using for merging?

```
#The key is country names
merge_data <- rawdata_343 %>% full_join(rawdata_373, by=c("Country_Names"="country_name"))

#sort based on country names
merge_data <- merge_data %>% arrange(Country_Names)
```

In order to merge the data, the command full_join() is used. The reason why the full join is used is that we desire to keep the union of all observations in the two tables. The country names is the key that is used for merging. Also, we sort the data based on the country names using the arrange() command.

Task 4

Merge the resulting data set above with the data set containing country information (from point 1) using dplyr functions on the unique keys. Name this new object df_vars.

- Inspect the country names and check if it would be a reliable variable for matching. Why or why not?
- In TUWEL you have the file CIA_factbook_matching_table_iso.xlsx which provides reliable matching information based in ISO codes. Use this for the final merging of the data sets.

The data set df_vars is created by merging the data from task 1 and task 3. The command inner_join() is used in order to merge the 2 data sets and also to keep the same observational units. The key that is used is the country names. However, by inspecting the data, country names is not a reliable variable for matching because the data sets could contain the same country but the name could be different. This difference can be observed in the country Federated States of Micronesia. As can be seen, in the data sets from task 1 and 3, the country of Micronesia exists and in the df_vars data set does not (the command any() is used for that purpose.). Thus, the names of each country is not the proper variable for merging the data sets.

```
#merge using inner_join() due to unique keys
df_vars <- merge_data %>% inner_join(data_code, by=c("Country_Names"="official_name_en"))

any(merge_data$Country_Names == "Micronesia, Federated States of")
```

```
## [1] TRUE
```

```
any(data_code$official_name_en == "Micronesia (Federated States of)")
```

```
## [1] TRUE
```

```
any(df_vars$Country_Names == "Micronesia (Federated States of)")
```

```
## [1] FALSE
```

```
any(df_vars$Country_Names == "Micronesia, Federated States of")
```

```
## [1] FALSE
```

Therefore, more reliability is required. For that reason, the file CIA_factbook_matching_table_iso.xlsx, which provides reliable matching information based in ISO codes, is used. First, the library "readxl" is introduced, in order to read the xlsx file. Then, the data from task 1 is merged with the data from task 3 according to the country names using the command left_join(). Therefore, the data set temp_data is created, which contains the country names, the median age, the youth unemployment rate, the 2 ISO codes the region and sub region names and the development status. Finally, the df_vars data set is created by merging the iso data set and temp_data using as key the 2 ISO codes. Once again, the left_join() is used. Note that the country variable is removed from the final data set (df_vars) because it is insignificant to keep 2 columns with the same country names.

```
library("readxl")
```

```
## Warning: package 'readxl' was built under R version 4.2.3
```

```
iso <- read_excel("CIA_factbook_matching_table_iso.xlsx")
iso <- as.data.frame(iso)

temp_data <- data_code %>%
  left_join(merge_data, by = c("official_name_en" = "Country_Names"))

df_vars <- temp_data %>%
  left_join(iso, by=c("ISO3166.1.Alpha.2"="ISO 3166 2",
                      "ISO3166.1.Alpha.3"="ISO 3166 3")) %>%
  select(-Country)
```

Task 5

Discuss on the tidyness of the data set df_vars. What are the observational units, what are the variables? What can be considered fixed vs measured variables? Tidy the data if needed.

In order to inspect the data, the head() command is used.

```
head(df_vars)
```

```
##   official_name_en ISO3166.1.Alpha.2 ISO3166.1.Alpha.3
## 1                                 TW               TWN
## 2      Afghanistan                AF               AFG
## 3          Albania                AL               ALB
## 4          Algeria                DZ               DZA
## 5   American Samoa                AS               ASM
## 6          Andorra                AD               AND
##   Developed...Developing.Countries Region.Name Sub.region.Name Median_Age
## 1                                                                     NA
## 2                       Developing        Asia   Southern Asia       19.5
## 3                        Developed      Europe Southern Europe       34.3
## 4                       Developing      Africa Northern Africa       28.9
## 5                       Developing     Oceania       Polynesia       27.2
## 6                        Developed      Europe Southern Europe       46.2
##   youth_unempl_rate
## 1                NA
## 2              17.6
## 3              31.9
## 4              39.3
## 5                NA
## 6                NA
```

According to Handley Wickham, the data are tidy because the column headers are not values. Also, in each column only one variable is stored and the variables are not stored in both rows and columns. In addition, each type of observational unit forms a table and each observation is represented by a row. More specific, in the current data set (df_vars), the observational units are the countries across regions and the variables are: "Country_Names", "Median_Age", "youth_unempl_rate", "ISO 3166 2", "ISO 3166 3", "Developed...Developing.Countries", "Region.Name" and "Sub.region.Name". Finally, the fixed variables are: Country_Names", "ISO 3166 2", "ISO 3166 3", "Developed...Developing.Countries", "Region.Name" and "Sub.region.Name". The measured variables are: "Median_Age" and "youth_unempl_rate".

Task 6

Count the number of developing vs. developed countries in the merged data set.

```
df_vars %>%
  filter(!(Developed...Developing.Countries == "")) %>%
  count(Developed...Developing.Countries)
```

```
##   Developed...Developing.Countries   n
## 1                        Developed  66
## 2                       Developing 183
```

The function count() is used. Also, it is observed that some information is missing from the data regarding the variable which describes the development of a country. For that reason, the command filter() is used.

Task 7

Count how many countries per region does the merged data set contain.

```
df_vars %>%
  filter(!(Region.Name == "")) %>%
  count(Region.Name)
```

```
##   Region.Name  n
## 1      Africa 60
## 2    Americas 57
## 3        Asia 51
## 4      Europe 52
## 5     Oceania 29
```

The function count() is used. The same thing applies for the variable Region.Name, as in task 6.

Task 8

Count the number of developing vs. developed countries for each region.

```
df_vars %>%
  filter(!(Region.Name == ""), !(Developed...Developing.Countries == "")) %>%
  group_by(Region.Name) %>%
  count(Developed...Developing.Countries)
```

```
## # A tibble: 8 x 3
## # Groups:   Region.Name [5]
##   Region.Name Developed...Developing.Countries     n
##   <chr>       <chr>                            <int>
## 1 Africa      Developing                          60
## 2 Americas    Developed                            5
## 3 Americas    Developing                          52
## 4 Asia        Developed                            3
## 5 Asia        Developing                          48
## 6 Europe      Developed                           52
## 7 Oceania     Developed                            6
## 8 Oceania     Developing                          23
```

The function group_by() and count() are used.

Task 9

Create a table of average values and the standard deviation for both median age and youth unemployment rate separated into developing and developed countries (hint: eliminate observations with missing development status beforehand). Comment briefly on the results.

```
df_vars %>%
  filter(!is.na(Median_Age),!is.na(youth_unempl_rate),
         !(Developed...Developing.Countries == "")) %>%
  group_by(Developed...Developing.Countries) %>%
  summarize(average_median_age = mean(Median_Age),
            average_unemployment_rate=mean(youth_unempl_rate),
            sd_median_age = sd(Median_Age),
            sd_unemployment_rate=sd(youth_unempl_rate))
```

```
## # A tibble: 2 x 5
##   Developed...Developi~1 average_median_age average_unemployment~2 sd_median_age
##   <chr>                               <dbl>                  <dbl>         <dbl>
## 1 Developed                            42.4                   16.5          3.92
## 2 Developing                           26.9                   18.6          6.78
## # i abbreviated names: 1: Developed...Developing.Countries,
## #   2: average_unemployment_rate
## # i 1 more variable: sd_unemployment_rate <dbl>
```

First, the observations with missing values are removed using the filter() command. Then, we group the observations by the development status using the command group_by() and finally a table of average values and the standard deviation for both median age and youth unemployment rate separated into developing and developed countries is created using the command summarize().

Looking at the results, we can see that the average median age in developed countries is significantly higher than in developing countries. The average median age in developed countries is 42.36889, while in developing countries it is 26.87818 Also, the standard deviation differs a lot (Developed countries = 3.915692 and Developing Countries = 6.775785) which indicates the fact that there are developing countries that have high median age compared to the rest.

Surprisingly, the difference in average unemployment rates is not noticeable. However, for the second time, the standard deviation differs significantly.

Task 10

Repeat the analysis in the previous task for each development status and region combination.

```
options(dplyr.summarise.inform = FALSE)
df_vars %>%
  filter(!is.na(Median_Age),!is.na(youth_unempl_rate),
         !(Developed...Developing.Countries == "")) %>%
  group_by(Developed...Developing.Countries,Region.Name) %>%
  summarize(average_median_age = mean(Median_Age), average_unemployment_rate=mean(youth_unempl_rate),
            sd_median_age = sd(Median_Age), sd_unemployment_rate=sd(youth_unempl_rate))
```

```
## # A tibble: 8 x 6
## # Groups:   Developed...Developing.Countries [2]
##   Developed...Developing~1 Region.Name average_median_age average_unemployment~2
##   <chr>                    <chr>                    <dbl>                 <dbl>
## 1 Developed                Americas                  42.7                  20.2
## 2 Developed                Asia                      39.0                  10.3
## 3 Developed                Europe                    42.9                  17.0
## 4 Developed                Oceania                   37.4                  11.6
## 5 Developing               Africa                    21.6                  21.0
## 6 Developing               Americas                  31.9                  17.0
## 7 Developing               Asia                      29.2                  15.5
## 8 Developing               Oceania                   28.0                  21.5
## # i abbreviated names: 1: Developed...Developing.Countries,
## #   2: average_unemployment_rate
## # i 2 more variables: sd_median_age <dbl>, sd_unemployment_rate <dbl>
```

The same thing, as in task 9, is repeated in task 10. This time the group is done according the development status and region.

Based on the results, it is worth noting that there are not developing countries in Europe. Regarding the average median the difference between the developed and developing countries is noticeable. However, the standard deviation especially in Asia is quite high. Regarding the youth unemployment rate, it is worth mentioning that the average and the standard deviation of unemployment rate in the developed countries in America and Europe reaches the rate of the developing countries. This leads to the result that, although Europe and the US are considered developed regions, the unemployment rate remains high.

Task 11

In df_vars create two additional indicator variable above_average_median_age which contains a yes is the country's median age lies above the region average and no otherwise. Create another above_average_yu which contains the same information but for the youth unemployment variable.

```r
#create a data set which contains the average median age and unemployment rate per region.
average_region <- df_vars %>%
  filter(!is.na(Median_Age),!is.na(youth_unempl_rate)) %>%
  group_by(Region.Name) %>%
  summarize(average_median_age = mean(Median_Age),
            average_unemployment_rate=mean(youth_unempl_rate))

#initialize the final dataset
final_dataset <- data.frame()

#using the for loop, we take each region at a time
for (region in average_region$Region.Name){

  #filter the data per region
  filtered_data_per_region <- df_vars %>% filter(Region.Name==region)

  #take the average values per region
  filtered_average <- average_region %>% filter(Region.Name==region)

  #create a temporary data set
  #create the two variables using the mutate command
  #use the case_when function for the "yes" and "no"
  temporary <- filtered_data_per_region %>%
    mutate(
      above_average_median_age = case_when(
      Median_Age > filtered_average$average_median_age ~ "yes",
      Median_Age <= filtered_average$average_median_age ~ "no"),

      above_average_yu = case_when(
        youth_unempl_rate > filtered_average$average_unemployment_rate ~ "yes",
        youth_unempl_rate <= filtered_average$average_unemployment_rate ~ "no"
      ))

  final_dataset <- rbind(final_dataset,temporary)
}
```

First a table (average_region) is created, which contains the average median age and unemployment rate per region. We filter any NAs that is contained in the data set df_vars. Then, the final data frame is initialized. A for loop is used in order to take each region at a time. Inside the for loop, a new data frame is introduced (filtered_data_per_region), which is a subset of the df_vars data frame filtered by region. After that, we obtain the average median and unemployment rate for each region and assign it to filtered_average. Finally, a data frame (named temporary) is created in order firstly to create the two new variables, in the subset filtered_data_per_region, using the mutate() command and secondly to use the case_when() function to assign "yes" or "no" whether the median age or the unemployment rate is lower or higher than the region rate. Finally, the rbind() function is used in order to add the rows of the subset to out final data frame (named: final_dataset).

Task 12

Export the final data set to a csv with ";" separator and "." as a symbol for missing values; no rownames should be included in the csv. Upload the .csv to TUWEL together with your .Rmd and PDF.

```
write.table(final_dataset, file = "Final_data.csv", sep = ";", na = ".",
            row.names = FALSE,col.names = TRUE)
```

The final data set is exported to a csv file, named "Final_data.csv", using the command write.table with ";" separator and "." as a symbol for missing values. Also, no rownames are included.