Rcode for GV900 Homework 1

GV900 Political Explanation

My student ID: 2107091

13-11-2021

```
# Firstly, it will be advisable to clean the console and the environment from
previously used data and values by using the following functions:

rm(list=ls(all=TRUE))
cat("\014")
```

```r
# Then, I load the ggplot2 package in order to enable the following code to draw graphs by using the "library" function since I have already installed it:

library(ggplot2)
```

1. Load the data set

```r
# Now, I can import the data set in my console by using the "read.csv" function to read the Comma Separated Values file. Moreover, I will use the "paste0" function to concatenate all elements without a separator (source: https://r-lang.com/paste0-function-in-r-with-example/). The path to find the data is stored in an object called "myPath":

world.data <- read.csv(paste0(myPath, "world.csv"))
```

2. Create frequency table

```r
# To create a frequency table, I will use  "data.frame" and "table" functions, and I will assign the name "ft.oecd" to it. Indeed, the data.frame function creates data frames. Hence:

ft.oecd <- data.frame(table(world.data$oecd))

# However, the previous frequency table has only two columns, namely "Var1" and "Freq", but not the percentage column. Therefore, it is essential to add it by using the "prop.table" function. First of all, it is necessary to calculate the relative frequencies using the "prop.table" function "prop.table(ft.oecd$Freq)".Then, we can convert the resulting relative frequencies into percentages by multiplying them by 100 "prop.table(ft$oecd)*100". Finally, it is important to insert the new column Percentage into the already existing ft.oecd frequency table. Thus, the final code:

ft.oecd$Percentage <- prop.table(ft.oecd$Freq) * 100

# It should be noticed that, in the assignment, there is no request to round the percentage results. However, it would be advisable to do so in order to have a more understandable table by using the "round" function. Namely:

ft.oecd$Percentage <- round(ft.oecd$Percentage, digits = 2)

# Besides, the name "Var1" in the first column does not help to understand the meaning of the variable, thus, to change it, I will use the function "colnames":

colnames(ft.oecd)[colnames(ft.oecd) == "Var1"] <- "OECD Member?"
```

```r
# Finally, I will use the "print" function to display the frequency table on
my console and file:

print (ft.oecd)
```

```
##        OECD Member? Freq Percentage
## 1        Not member  161      84.29
## 2 OECD Member state   30      15.71
```

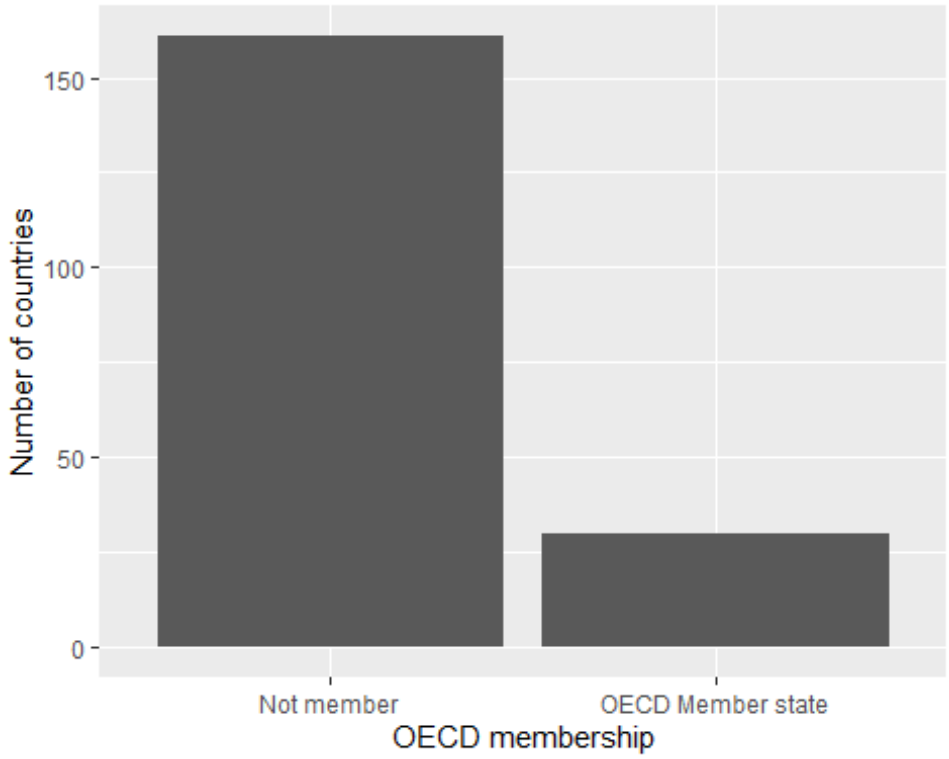
3. Questions about ft.oecd

```r
# (A) 30 countries are OECD Member States (in R: "ft.oecd[2,2]")

# (B) 161 countries are not OECD Member States (in R: "ft.oecd[1,2]")

# (C) 15.71% of countries are OECD Member States (in R: "ft.oecd[2,3]")

# (D) 84.29% of countries are not OECD Member States (in R: "ft.oecd[1,3]")
```

4. Create a bar chart

```r
# To draw a bar chart, I will use the "ggplot" function to tell R in which
data frame the variables are stored, in this case, in world.data. Besides, in
order to make things simpler, I will store the graph in an object called "b":

b <- ggplot(world.data)

# Then, I have to tell R what variables to plot inside the chart thanks to
the "aes" function, in this case, membership into OECD. Further, it is
significant to remember to insert "b + ..." when we modify the object;
otherwise, the same-called object will be rewritten, and we will lose all the
previous data stored in it:

b <- b + aes(x = oecd)

# Then, I use the geom_bar function to tell R what graph I want, in this
case, a bar chart:

b <- b + geom_bar()

# I will use the "ylab" function to change the name of the y-axis in the bar
graph previously created:

b <- b + ylab("Number of countries")

# Further, I will use the "xlab" function to change the name of the X-axis:

b <- b + xlab("OECD membership")
```

```
# Finally, I will use the "print" function to display the graph

print(b)
```



5. Name countries

```
# OECD Member States:
# a) United Kingdom
# b) Italy
# c) United States

# Non-democratic countries:
# d) Egypt
# e) Iran
# f) Qatar
```

6. Description of per capita GDP variable

```
# To statistically describe the per capita GDP variable (stored inside
world.data as "gdp_10_thou"), I can use the "summary" function, which will
display in the console the minimum value of the called variable, the maximum
value, the median, the mean, and the first and third quartile.

summary(world.data$gdp_10_thou)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  0.0090  0.0503  0.1897  0.6018  0.6320  4.7354      14

# Minimum: 0.0090
# Maximum: 4.7354
# Median: 0.1897
# Mean: 0.6018
# 1st Quartile: 0.0503
# 3rd Quartile: 0.6320

# To facilitate the use of the summary measurements, I could store it into an
object called "summary_gdp", thus:

summary_gdp <- summary(world.data$gdp_10_thou)

# To calculate standard deviation, I will use the "sd" function. However, as
shown in the previous summary, there are 14 missing values in the data set.
These missing values can alter the measurement of standard deviation. Hence,
to eliminate them from the calculation, I will insert "na.rm = TRUE", meaning
that R will now display the measurements excluding the observations with
missing values:

sd(world.data$gdp_10_thou, na.rm = TRUE)

## [1] 0.9433982

# Standard deviation: 0.9433982

# To facilitate the use of the standard deviation measurement, I could store
it into an object called "sd_gdp", thus:

sd_gdp <- sd(world.data$gdp_10_thou, na.rm = TRUE)

# Besides and more than asked in this assignment, I may want to calculate the
summary without including the NA values. To do this, I will first create a
subset of gdp_10_thou, named "capita_gdp", where the NA values are excluded:

capita_gdp <- world.data[!is.na(world.data$gdp_10_thou),]

# Then, I will use the "summary" function on the new subset formed by 177
objects:

summary(capita_gdp$gdp_10_thou)

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0090  0.0503  0.1897  0.6018  0.6320  4.7354
```

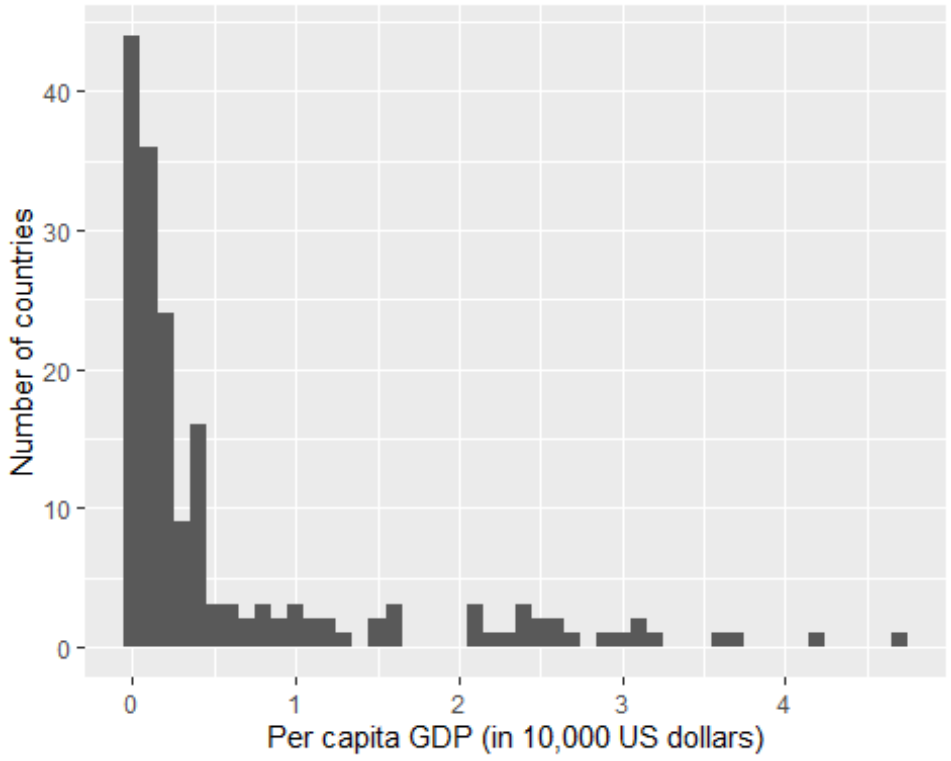
7. Skewness of the per capita GDP

```
# The median is the "value of the case that sits at the exact centre of our cases when we rank the values of a single variable from the smallest to the largest observed values" (Kellstedt and Whitten, 2018).
# The mean is the "arithmetical average of a variable equal to the sum of all values of Y across individual cases of Y, Y_i, divided by the total number of cases" (Kellstedt and Whitten, 2018).
# Following these definitions, it is reasonable to assume that the majority of the values (as the number of samples but with low absolute value) are on the left, nearer zero. On the other hand, the minority of values are much greater in absolute terms than the others, increasing the average value of the variable.
# Therefore, it can be assumed that the variable is positively skewed since the mean is to the right of the median. (source: https://www.newtraderu.com/2020/08/27/positive-skew-vs-negative-skew/; https://opentextbc.ca/introbusinessstatopenstax/chapter/skewness-and-the-mean-median-and-mode/#M06_Ch02_fig003)
```

8. Histogram of per capita GDP

```
#To draw a histogram, firstly, I will use the "ggplot" function to tell R in which data frame the variables are stored, in this case, world.data. Besides, in order to make things simpler, I will store the graph in an object called "h":

h <- ggplot(world.data)

# Then, I have to tell R what variables to plot inside the chart thanks to the "aes" function, in this case, per capita GDP.

h <- h + aes(gdp_10_thou)

# Then, I use the "geom_histogram" function to tell R what graph I want, in this case, a histogram chart. Moreover, I changed the binwidth (width of each bin along the x-axis) of the histogram in order to display the data more coherently. I chose 0.1 as binwidth in order to expose the countries in groups with a range of 1000 USD.Furthermore, I added the function "na.rm" to remove all countries without a measurement of per capita GDP from the histogram

h <- h + geom_histogram(binwidth = 0.1, na.rm = TRUE)

# Besides, I will use the "ylab" function to change the name of the y-axis in the histogram previously created:

h <- h + ylab("Number of countries")

# Further, I will use the "xlab" function to change the name of the X-axis:
```

```r
h <- h + xlab("Per capita GDP (in 10,000 US dollars)")

# Finally, I will use the "print" function to display the histogram

print(h)
```



9. Coutries with more than 40,000 USD per capita GDP

```r
# To display the name of the two countries, I will create a subset of
world.data. To do so, I first create an object "high_capita_gdp" to store the
values of the subset.

high_capita_gdp <- world.data[world.data$gdp_10_thou > 4, ]

# Then, I will use the "print" function to display only the column "country"
for the high_capita_gdp subset:

print(high_capita_gdp$country)
```

```
##  [1] NA          NA          NA          NA          NA
##  [6] NA          NA          "Luxembourg" NA          NA
## [11] NA          "Norway"    NA          NA          NA
## [16] NA

# Therefore, the two countries are Luxembourg and Norway
```

10. Standard error of the mean

```
# Firstly, to calculate the standard error, I will store the mean value as an
object named "pe" (point estimate), which represents the central tendency of
the variable (Kellstedt and Whitten, 2018).
# Moreover, in order to eliminate all NA values, I will also insert the
"na.rm" parameter inside the object:

pe <- mean(world.data$gdp_10_thou, na.rm = TRUE)

# Then, it is necessary to recall the standard deviation measure. However, I
have already stored it in the object "sd_gdp", so I will only recall it:

# sd_gdp

# (I inserted this recall as a comment in order to not interfere with the
coding)

# Then, it is necessary to measure the square root of the number of samples
in the variable. To calculate them, I will use the "length" function However,
it is also necessary to eliminate all the NA values from the variable.
Fortunately, I have already done that in question 6 and stored them in the
"capita_gdp" object. Thus, I will only recall it, use the length function,and
store it in the object called "n":

n <- length(capita_gdp$gdp_10_thou)

# Finally, the standard error is (standard deviation)/(squared root of the
number of samples). Besides, I will store the standard deviation in the
object "se", hence:

se <- sd_gdp/sqrt(n)

# I will also display it using the "print" function:

print(se)
```

```
## [1] 0.07091015
```

11. 95% confidence interval

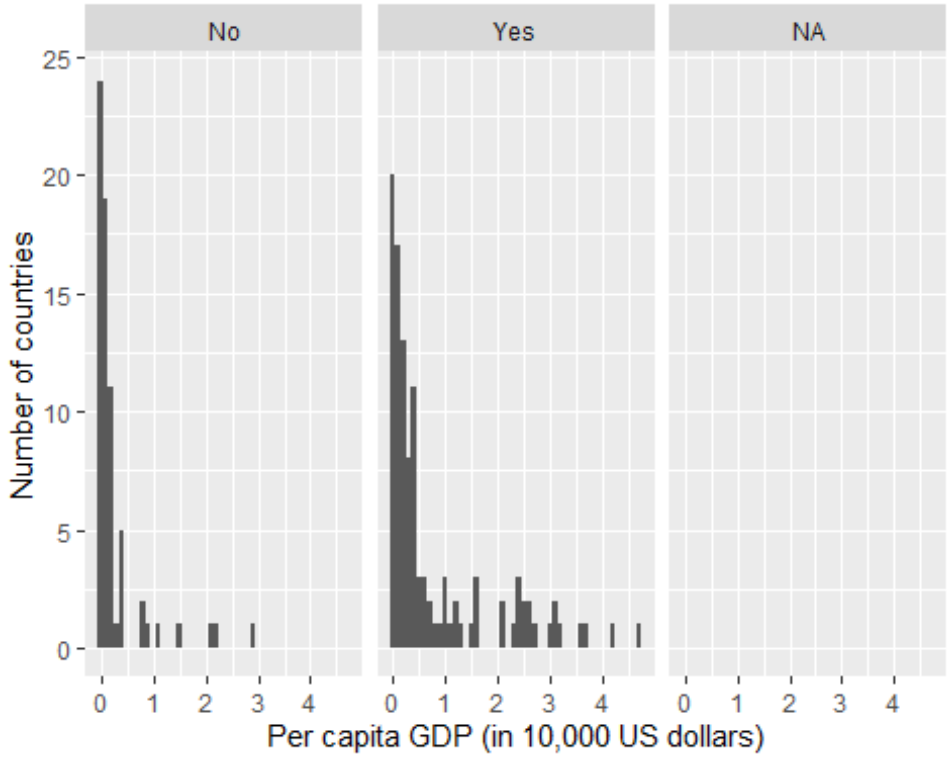
```
# The 95% confidence interval has an upper and lower bound, calculated as the
point estimates minus/plus two standard errors. Therefore, if the point
estimate is filed in the object pe, and the standard error is saved in the
object se, then the lower bound (stored as an object named "lb_gdp") would be
the point estimate minus two standard errors:

lb_gdp <- pe - 2 * se
```

```
# Then, the upper bound (stored as an object named "ub_gdp") would be the
point estimate plus two standard errors:

ub_gdp <- pe + 2 * se

# Therefore, I will print them using the "print" function, first the lower
bound:

print(lb_gdp)

## [1] 0.4599983

# Then, the upper bound:

print(ub_gdp)

## [1] 0.7436389

# Finally, I will print the result as:

print(paste("The mean of gdp_10_thou is", pe,
            "with a 95% confidence interval of [", lb_gdp, ",", ub_gdp, "]"))

## [1] "The mean of gdp_10_thou is 0.601818644067797 with a 95% confidence
interval of [ 0.459998340237962 , 0.743638947897631 ]"
```

12. Histograms of per capita GDP in democracies and per capita GDP in non-democracies

```
# To draw histograms of per capita GDP based on the political regime in the
countries, I will use the "facet_wrap" function and store the histogram for
democracies as "d" and that for autocracies as "a".
# Firstly, I will use the "ggplot" function to tell R in which data frame the
variables are stored, in this case, world.data:

d <- ggplot(world.data)

# Then, I have to tell R what variables to plot inside the chart thanks to
the "aes" function, in this case, per capita GDP:

d <- d + aes(gdp_10_thou)

# Then, I use the "geom_histogram" function to tell R what graph I want, in
this case, a histogram chart. Moreover, I changed the binwidth of the
histogram in order to display the data more coherently. I chose 0.1 as
binwidth in order to expose the countries in groups with a range of 1000 USD.
Furthermore, I added the function "na.rm" to remove all countries without a
measurement of per capita GDP from the histogram. Hence:
```

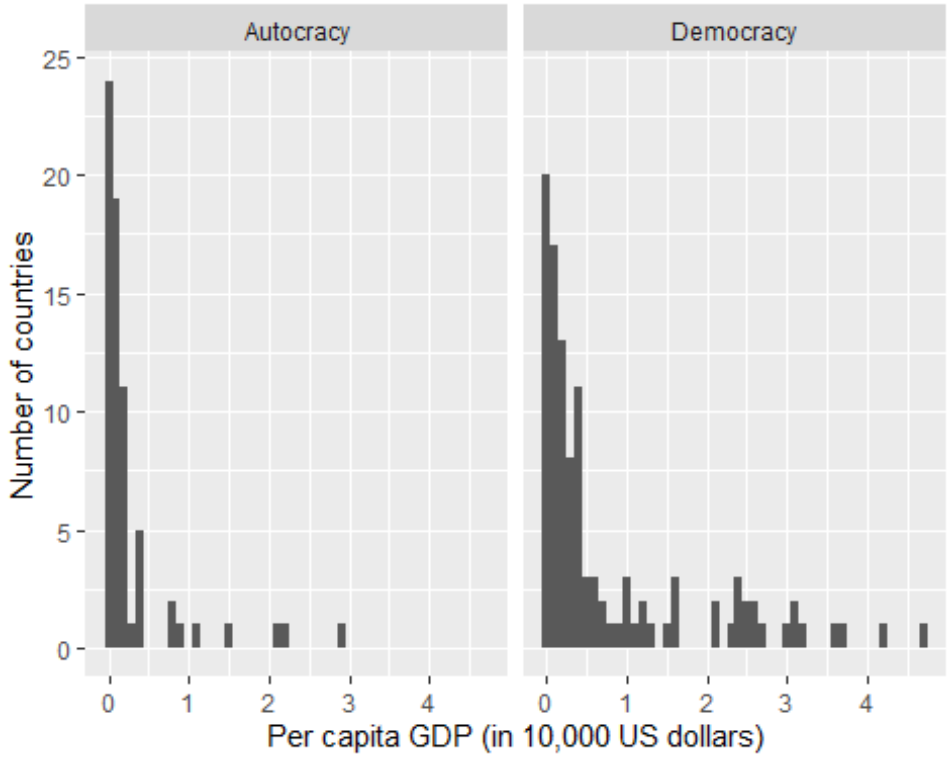
```r
d <- d + geom_histogram(binwidth = 0.1, na.rm = TRUE)

# Besides, I will use the "ylab" function to change the name of the y-axis in
the histogram previously created:

d <- d + ylab("Number of countries")

# Further, I will use the "xlab" function to change the name of the X-axis:

d <- d + xlab("Per capita GDP (in 10,000 US dollars)")

# To differentiate the graph, I will use the "facet_wrap" option, linking it
to the "democ_regime" variable in World.data.

d <- d + facet_wrap( ~ democ_regime)

# Finally, I use the "print" function to display the histogram

print(d)
```



13. Adjusted histograms

```r
# In the democ_regime variable, there are missing values. First of all, I
will use the "is.na" option to eliminate them and store the new data set into
a data frame called "dem.gdp". Therefore, in the dem.gdp data set there are
```

```
189 countries (Monaco and Tuvalu are excluded):

dem.gdp <- world.data[!is.na(world.data$democ_regime), ]

# To insert the label "Democracy" and "Autocracy" in the graph, I will create
a new factor, called "dem.dum", where I will insert the new names of the
labels:

dem.gdp$dem.dum <- factor(dem.gdp$democ_regime,
                  levels = c("No", "Yes"),
                  labels = c("Autocracy", "Democracy"))

# Finally, I will re-create the new histogram, storing it in the object "a":

a <- ggplot(dem.gdp)
a <- a + aes(gdp_10_thou)
a <- a + geom_histogram(binwidth = 0.1, na.rm = TRUE)
a <- a + xlab("Per capita GDP (in 10,000 US dollars)")
a <- a + ylab("Number of countries")
a <- a + facet_wrap (~ dem.dum)

# Finally, I print the histograms:

print(a)
```



14. Mean of per capita GDP for democracies

```
# Firstly, to calculate the mean of per capita GDP for democracies, I will
purify the gdp_10_thou from all missing values NA, and I will store it in an
object called "data.gdp10" that will contain 177 countries:

data.gdp10 <- world.data[!is.na(world.data$gdp_10_thou), ]

# Then, I will create a factor that will label  all the countries in
data.gdp10 based if they are democratic or not in accordance to the
democ_regime variable. I will store the factor in an object called "f.gdp10":

f.gdp10 <- factor(data.gdp10$democ_regime,
                  levels = c("No", "Yes"),
                  labels = c("Autocracy", "Democracy"))

# Then, I will use the "split" function to separate the data of per capita
GDP values for democracies from the data of per capita GDP values for
autocracies.Further, I will store the slipt into an object named "split.gdp":

split.gdp <- split(data.gdp10$gdp_10_thou, f = f.gdp10)

# Indeed, the split function divides the input data into different groups
(source: https://r-coder.com/split-r/). In this case, the split function
separated the GDP values for democracies from GDP values for autocracies
thanks to the instruction given by factor f.gdp10. Indeed, I inserted as
input only the column regarding my studied variable (gdp_10_thou) inside the
purified dataset data.gdp10 (source:
https://stackoverflow.com/questions/33426973/r-split-data-frame-and-save-to-
different-
files#:~:text=To%20split%20data%20frame%2C%20use%20split%20%28df4%2C%20df4%24
location%29.,everything%20in%20current%20R%20session%20into%20filename.RData%
20file.)
# Now, I can use the split data from my subset to calculate the mean of per
capita GDP for democracies. I will first store the value of the mean as an
object named "pe_dem" (point estimate), which represents the central tendency
of the variable.

pe_dem <- mean(split.gdp[["Democracy"]])

# In this case, I did not insert the function "na.rm" since I have already
eliminated all missing values before. Therefore, I will print it by using the
"print" function:

print(pe_dem)

## [1] 0.8013927

# Then, to calculate the standard error, it is necessary to calculate the
standard deviation. I will use the "sd" function, and I will store the value
```

```r
as an object called "sd_dem":

sd_dem <- sd(split.gdp[["Democracy"]])

# Then, I will calculate the number of samples by using the "length"
function, and I will store the result as an object called "n_dem":

n_dem <- length(split.gdp[["Democracy"]])

# Finally, I will calculate the standard error as the standard deviation
divided by the square root of the number of samples, and I will store the
result as an object called "se_dem", namely:

se_dem <- sd_dem/sqrt(n_dem)

# If the point estimate is stored in the object "pe_dem",and the standard
error is stored in the object "se_dem", then the lower bound (stored as an
object named "lb_dem") would be the point estimate minus two standard errors,
in R language:

lb_dem <- pe_dem - 2 * se_dem

# Then, the upper bound (stored as an object named "ub_dem") would be the
point estimate plus two standard errors:

ub_dem <- pe_dem + 2 * se_dem

# Therefore, I will print them using the "print" function, first the lower
bound:

print(lb_dem)

## [1] 0.5942867

# Then, the upper bound:

print(ub_dem)

## [1] 1.008499

# Finally, I will print the result as:

print(paste("The mean of gdp_10_thou for democratic countries is", pe_dem,
            "with a 95% confidence interval of [", lb_dem, ",", ub_dem, "]"))

## [1] "The mean of gdp_10_thou for democratic countries is 0.801392660550459
with a 95% confidence interval of [ 0.594286678454443 , 1.00849864264647 ]"
```

15. Mean of per capita GDP for autocracies

```
# Following the previous stages, I will first store the value of the mean of
per capita GDP for autocracies as an object named "pe_aut" (point estimate),
which represent the central tendency of the variable.

pe_aut <- mean(split.gdp[["Autocracy"]])

# In this case, I did not insert the function "na.rm" since I have already
eliminated all missing values before. Therefore, I will print it by using the
"print" function:

print(pe_aut)

## [1] 0.2819132

# Then, to calculate the standard error, it is necessary to calculate the
standard deviation. I will use the "sd" function, and I will store the value
as an object called "sd_aut":

sd_aut <- sd(split.gdp[["Autocracy"]])

# Then, I will calculate the number of samples by using the "length"
function, and I will store the result as an object called "n_aut":

n_aut <- length(split.gdp[["Autocracy"]])

# Finally, I will calculate the standard error as the standard deviation
divided by the square root of the number of samples, and I will store the
result as an object called "se_aut", namely:

se_aut <- sd_aut/sqrt(n_aut)

# If the point estimate is stored in the object pe_aut,and the standard error
is stored in the object se_aut, then the lower bound (stored as an object
named "lb_gdp_aut") would be the point estimate minus two standard errors:

lb_aut <- pe_aut - 2 * se_aut

# Then, the upper bound (stored as an object named "ub_gdp") would be the
point estimate plus two standard errors:

ub_aut <- pe_aut + 2 * se_aut

# Therefore, I will print them using the "print" function, first the lower
bound:

print(lb_aut)

## [1] 0.1523983
```

```r
# Then, the upper bound:

print(ub_aut)
```

```
## [1] 0.4114282
```

```r
# Finally, I will print the result as:

print(paste("The mean of gdp_10_thou for autocracies is", pe_aut,
            "with a 95% confidence interval of [", lb_aut, ",", ub_aut, "]"))
```

```
## [1] "The mean of gdp_10_thou for autocracies is 0.281913235294118 with a
95% confidence interval of [ 0.152398258488598 , 0.411428212099638 ]"
```

End of assignment “Homerwork 1”