R-work for Online Assessment, 2021/22

Instructions

You should work through the code below and complete it. Keep the completed code and all the resulting output. Next you should answer the questions in the online quiz. Every student will see a slightly different collection of questions (as we will randomly draw 10 questions from a pool of about 20 questions).

The questions are of four types.

1) Questions that merely ask you to report output from your analysis.
2) Some questions will ask you about R code. For example, you will see a lot of gaps (`XXXX`) in the code and questions may ask you how to complete the code to make the code work. Sometimes the `XXXX` will represent one word and on other occasions it will represent a full line (or two) of code. Other questions may ask you about the output to be produced by a particular bit of code.
3) The third type of questions will test your understanding of econometric issues. For example: "What is the meaning of an estimated coefficient?" "Is a particular coefficient statistically significant?"
4) The fourth type of question, if asked, will be on general programming issues. For example: what is the meaning of a particular error message, or, how would you search for a particular piece of information.

Preparing your workfile

Set the working directory

```
setwd("YOUR/WORKING/DIRECTORY")
```

```
setwd("C:/Rcode/RforQM/Regression")
```

We add the basic libraries needed for this week's work:

```
library(tidyverse)    # for almost all data handling tasks
library(ggplot2)      # to produce nice graphiscs
library(stargazer)    # to produce nice results tables
library(haven)        # to import stata file
library(AER)          # access to HS robust standard errors
library(countrycode)

source("stargazer_HC.r")  # includes the robust regression display
```

Introduction

The data are an extract from the OECD.

Task 1: Data Upload - and understanding data structure

Upload the data, which are saved in the two csv files `OECD_RoadAccidents.csv` (linked from here) and `OECD_Passenger_Transport.csv` (linked from here).

```
data_acc <- XXXX("OECD_RoadAccidents.csv")
data_pt <- XXXX("OECD_Passenger_Transport.csv")

names(data_acc)
names(data_pt)
```

```
data_acc <- read_csv("OECD_RoadAccidents.csv")
data_pt <- read_csv("OECD_Passenger_Transport.csv")
names(data_acc)
```

```
## [1] "LOCATION"  "INDICATOR" "SUBJECT"   "MEASURE"   "FREQUENCY" "TIME"
## [7] "Value"
```

```
names(data_pt)
```

```
## [1] "LOCATION"  "INDICATOR" "SUBJECT"   "MEASURE"   "FREQUENCY" "TIME"
## [7] "Value"
```

Look at the spreadsheets and understand the data structure.

Task 2: Cleaning Accident data

Run the following lines and use the information to understand the data structure.

```
data_acc[1,]
```

```
## # A tibble: 1 x 7
##   LOCATION INDICATOR SUBJECT MEASURE FREQUENCY  TIME Value
##   <chr>    <chr>     <chr>   <chr>   <chr>     <dbl> <dbl>
## 1 SVN      ROADACCID DEATH   NBR     A          1970   620
```

```
unique(data_acc$MEASURE)
```

```
## [1] "NBR"        "1000000HAB" "1000000VEH"
```

```
unique(data_acc$SUBJECT)
```

```
## [1] "DEATH"          "INJURE"         "ACCIDENTCASUAL"
```

In order to understand what these different measures and subjects represent you should consult the website linked above from which the data were downloaded.

Now find all the data which relate to Germany (country code: DEU) and Poland (country code: POL) in 2017.

```
data_acc %>% filter(LOCATION %in% c("DEU","POL"), TIME == "2017")
```

```
## # A tibble: 10 x 7
##    LOCATION INDICATOR SUBJECT        MEASURE    FREQUENCY  TIME    Value
##    <chr>    <chr>     <chr>          <chr>      <chr>     <dbl>    <dbl>
##  1 POL      ROADACCID INJURE         NBR        A          2017  39466
##  2 POL      ROADACCID ACCIDENTCASUAL NBR        A          2017  32760
##  3 POL      ROADACCID DEATH          1000000HAB A          2017     74.5
##  4 DEU      ROADACCID INJURE         NBR        A          2017 390312
##  5 DEU      ROADACCID DEATH          1000000HAB A          2017     38.5
##  6 DEU      ROADACCID ACCIDENTCASUAL NBR        A          2017 302656
##  7 POL      ROADACCID DEATH          NBR        A          2017   2831
##  8 DEU      ROADACCID DEATH          NBR        A          2017   3180
##  9 DEU      ROADACCID DEATH          1000000VEH A          2017     57.2
```

```
## 10 POL      ROADACCID DEATH          1000000VEH A          2017     95.5
```

You should see 10 rows of data.

We will only keep death data and only those data which measure the number of deaths by 1000000 inhabitants.

```
data_acc <- data_acc %>%  filter(SUBJECT == "DEATH") %>%
                          filter(MEASURE == "1000000HAB")
```

The only data left in the `Value` column is the deaths per 1,000,000 inhabitants variable. We therefore can now change the name of that variable to `Deaths_p1M`.

```
names(XXXX)[names(data_acc)=="XXXX"] <- "Deaths_p1M"
```

```
names(data_acc)[names(data_acc)=="Value"] <- "Deaths_p1M"
```

The datafile contains a few variables which are now redundant or we don't need any longer. We will only keep the variables we need, `LOCATION`, `TIME` and `Deaths_p1M`. The code below is faulty and you need to fix it

```
data_acc <- data_acc %>% Select(LOCATION, time, Deaths_p1M)
```

```
data_acc <- data_acc %>% select(LOCATION, TIME, Deaths_p1M)
```

The `LOCATION` variable represents the three digit country code of the respective observation. Let's add proper country names to our variables. For this we use the function `countrycode` from the `countrycode` package. The available information in the datafiles is in the `LOCATION` variable. It has a numeric ISO-3 format (`origin = "iso3c"`). We want to create a new variable in both of our datasets which is called `country` and contains the respective full English country name. You will have to consult the help for function `countrycode` to figure out what the destination format should be.

```
data_acc$country <- XXXX(data_acc$LOCATION, origin = "iso3c",
                         destination = "XXXX")
```

```
data_acc$country <- countrycode(data_acc$LOCATION, origin = "iso3c",
                                destination = "country.name")
```

Task 3: Cleaning Personal Transport data

Run the following lines and use the information to understand the data structure.

```
data_pt[1,]
```

```
## # A tibble: 1 x 7
##   LOCATION INDICATOR  SUBJECT MEASURE FREQUENCY  TIME  Value
##   <chr>    <chr>      <chr>   <chr>   <chr>     <dbl>  <dbl>
## 1 AUS      PASSTRANSP RAIL    MLN_PKM A          1970 12473.
```

```
unique(data_pt$MEASURE)
```

```
## [1] "MLN_PKM"
```

```
unique(data_pt$SUBJECT)
```

```
## [1] "RAIL"   "ROAD"   "INLAND"
```

What does `MLN_PKM` stand for? What do the different values in `SUBJECT` represent? You may need to refer to the website (see link above) to check. Note that the miles traveled are given as total distance. The information is not standardised by population size. We will do this later.

You may want to check your understanding by looking at the data for Germany and Poland in 2017.

```r
data_pt %>% filter(LOCATION %in% c("DEU","POL"), TIME == "2017")
```

```
## # A tibble: 6 x 7
##   LOCATION INDICATOR  SUBJECT MEASURE FREQUENCY  TIME   Value
##   <chr>    <chr>      <chr>   <chr>   <chr>     <dbl>   <dbl>
## 1 DEU      PASSTRANSP RAIL    MLN_PKM A          2017   95530
## 2 DEU      PASSTRANSP ROAD    MLN_PKM A          2017  977430
## 3 POL      PASSTRANSP RAIL    MLN_PKM A          2017   20319
## 4 POL      PASSTRANSP ROAD    MLN_PKM A          2017  257610
## 5 DEU      PASSTRANSP INLAND  MLN_PKM A          2017 1072960
## 6 POL      PASSTRANSP INLAND  MLN_PKM A          2017  277929
```

```r
# You should see that, in a particular year in a particular country,
# RAIL + ROAD = INLAND
```

We basically have three variables here, `RAIL` travel kilometers (or better miles), `ROAD` travel and total travel (`INLAND`). We want to keep these three variables. To do so we use the `pivot_wider` command which is part of the tidyverse package.

```r
data_pt <- data_pt %>% pivot_wider(names_from = SUBJECT,
                                   values_from = Value)
```

This is a useful function and you may want to remember that this functionality exists for your later work.

Display again the data for Germany and Poland in 2017 to see the difference of the datafile's setup after this operation. You should see the following:

```r
data_pt %>% filter(LOCATION %in% c("DEU","POL"), TIME == "2017")
```

```
## # A tibble: 2 x 8
##   LOCATION INDICATOR  MEASURE FREQUENCY  TIME  RAIL   ROAD  INLAND
##   <chr>    <chr>      <chr>   <chr>     <dbl> <dbl>  <dbl>   <dbl>
## 1 DEU      PASSTRANSP MLN_PKM A          2017 95530 977430 1072960
## 2 POL      PASSTRANSP MLN_PKM A          2017 20319 257610  277929
```

```r
# You should see that, in a particular year in a particular country,
# RAIL + ROAD = INLAND
```

The datafile contains a few variables which are now redundant or we don't need any longer. We will only keep the variables we need, `LOCATION`, `TIME`, `RAIL`, `ROAD` and `INLAND`.

```r
data_pt <- XXXX %>% XXXX(LOCATION, TIME, RAIL, ROAD, INLAND)
```

```r
data_pt <- data_pt %>% select(LOCATION, TIME, RAIL, ROAD, INLAND)
```

Now add the country name as above for `data_acc`.

```r
data_pt$XXXX <- XXXX(data_pt$XXXX, origin = "iso3c",
                        destination = "XXXX")
```

```r
data_pt$country <- countrycode(data_pt$LOCATION, origin = "iso3c",
                               destination = "country.name")
```

Task 4: Merging dataset

We will want to merge the two datasets `data_acc` and `data_pt` together. Before merging it is useful to review the variable names of both files and the number of observations in each.

```r
nrow(data_acc)
```

```
## [1] 1368
```

```r
names(data_acc)
```

```
## [1] "LOCATION"   "TIME"       "Deaths_p1M" "country"
```

```r
nrow(data_pt)
```

```
## [1] 2598
```

```r
names(data_pt)
```

```
## [1] "LOCATION" "TIME"     "RAIL"     "ROAD"     "INLAND"   "country"
```

The files have three variables in common, `LOCATION`, `TIME` and `country`. This is the information on the basis of which we want to match the data. So for instance we will want one row of data for Germany in 2017 and that row of data should contain `Deats_p1M` from the `data_acc` dataset and `RAIL`, `ROAD` and `INLAND` from the `data_pt` dataset.

We use the merge function to achieve this. As you can see from the above information, the two datasets contain different numbers of rows. This is because not all the sources have complete information. To demonstrate that, let's look at Ukraine (UKR) and find the information for Ukraine from both datasets.

```r
data_acc %>% filter(LOCATION == "UKR")
```

```
## # A tibble: 24 x 4
##    LOCATION  TIME Deaths_p1M country
##    <chr>    <dbl>      <dbl> <chr>
##  1 UKR       1994       146. Ukraine
##  2 UKR       1995       146. Ukraine
##  3 UKR       1996       130. Ukraine
##  4 UKR       1997       118. Ukraine
##  5 UKR       1998       110. Ukraine
##  6 UKR       1999       106. Ukraine
##  7 UKR       2000       105. Ukraine
##  8 UKR       2001       123. Ukraine
##  9 UKR       2002       124. Ukraine
## 10 UKR       2003       150. Ukraine
## # ... with 14 more rows
```

```r
data_pt %>% filter(LOCATION == "UKR")
```

```
## # A tibble: 31 x 6
##    LOCATION  TIME  RAIL  ROAD INLAND country
##    <chr>    <dbl> <dbl> <dbl>  <dbl> <chr>
##  1 UKR       1990 76038    NA     NA Ukraine
##  2 UKR       1991 70968    NA     NA Ukraine
##  3 UKR       1992 76196    NA     NA Ukraine
##  4 UKR       1993 75896    NA     NA Ukraine
##  5 UKR       1994 70882    NA     NA Ukraine
##  6 UKR       1995 63752    NA     NA Ukraine
##  7 UKR       1996 59080    NA     NA Ukraine
##  8 UKR       1997 54433    NA     NA Ukraine
##  9 UKR       1998 49938    NA     NA Ukraine
## 10 UKR       1999 47600    NA     NA Ukraine
## # ... with 21 more rows
```

You should see that the information for Ukraine starts in 1990 in the `data_pt` datafile but only in 1994 in the `data_acc` datafile. You can also see that the Ukraine only has information on rail travel, but not road travel.

You could decide, as we are merging data, to only keep information which is available from both datafiles, or to keep all information (Year-country combinations) which are available in at least one of the datasets. Here we decide that we want to do the latter, i.e. keep all information for now, even if it cannot be matched from the other file. So in the above case that means that we do want the information from 1990 to 1993 from Ukraine included. In effect that means that our new dataset, `data_merge` should have at least 2598 rows.

Which of the following commands does that? (Note that, as the information on the basis of which we match, country and year, is saved in identically named columns in both datafiles, we do not have to use the `by.x` and `by.y` options in the merge function).

```
data_merge <- merge(data_pt,data_acc, all.x = TRUE, all.y = TRUE)
data_merge <- merge(data_pt,data_acc, all.x = TRUE, all.y = FALSE)
data_merge <- merge(data_pt,data_acc, all.x = FALSE, all.y = FALSE)
data_merge <- merge(data_pt,data_acc, all.x = FALSE, all.y = TRUE)
```

```
data_merge <- merge(data_pt,data_acc, all.x = TRUE, all.y = TRUE)
```

Task 5: Calculate country averages

We now have multiple years of data for the countries in our dataset, `data_avg`. The next step is to calculate the averages for our four substantial variables (`Deaths_p1M`, `RAIL`, `ROAD` and `INLAND`).

```
data_avg <- XXXX %>% XXXX(country,LOCATION) %>%
  XXXX(avg_rail = mean(XXXX,na.rm = XXXX),
            avg_road = XXXX,
            avg_inland = XXXX,
            avg_deaths_p1M = XXXX)
```

```
data_avg <- data_merge %>% group_by(country,LOCATION) %>%
  summarise(avg_rail = mean(RAIL,na.rm = TRUE),
            avg_road = mean(ROAD,na.rm = TRUE),
            avg_inland = mean(INLAND,na.rm = TRUE),
            avg_deaths_p1M = mean(Deaths_p1M,na.rm = TRUE))
```

You got it right, if your result, `data_avg` has 59 eows and you can replicate the following information:

```
head(data_avg,10)
```

```
## # A tibble: 10 x 6
## # Groups:   country [10]
##    country                LOCATION avg_rail avg_road avg_inland avg_deaths_p1M
##    <chr>                  <chr>       <dbl>    <dbl>      <dbl>          <dbl>
##  1 Albania                ALB          246.     6075.      6177.           96.9
##  2 Argentina              ARG         7765.    34058.     49191.          127.
##  3 Armenia                ARM           44.8    1945.      1987.           99.4
##  4 Australia              AUS        11658.   220449.    232107.           67.6
##  5 Austria                AUT         8792.    55652.     62935.           89.9
##  6 Azerbaijan             AZE         1304.    12856.     13838.           98.2
##  7 Belarus                BLR        11235.      NaN        NaN           134.
##  8 Belgium                BEL         8097.    98049.    106104.          100.
##  9 Bosnia & Herzegovina BIH            691.      NaN        NaN            89.4
## 10 Bulgaria               BGR         4748.    24748.     22303.          113.
```

As you can see, not all countries will have information on all variables.

As mentioned above, the information coming from the passenger transport file are not standardised by population size (`avg_rail`, `avg_road` and `avg_inland`), while `avg_deaths_p1M` already is. We now want to standardise the three variables coming from the passenger transport file by population size. For that we need to import a file with population information. That information (for more than 200 countries) is in `CountryInfo.csv`. Import that file and merge all the available country info into `data_avg` only keeping the 59 rows which we have in `data_avg`.

Note that you should match by the three letter country code which in `CountryInfo.csv` is labelled as `countryCode`. You will therefore, now have to use the `by.x` and `by.y` options in the `merge` function.

```
data_countries = read_csv("XXXX", na = "XXXX" )
data_avg <- merge(data_avg, XXXX,
                  by.x = "XXXX", by.y = "XXXX",
                  all.x = XXXX, all.y = XXXX)
```

```
data_countries = read_csv("CountryInfo.csv", na = "NA" )
data_avg <- merge(data_avg, data_countries,
                  by.x = "LOCATION", by.y = "countryCode",
                  all.x = TRUE, all.y = FALSE)
```

Now check the names of the variables in your datafile.

```
names(data_avg)
```

```
##  [1] "LOCATION"       "country"        "avg_rail"       "avg_road"
##  [5] "avg_inland"     "avg_deaths_p1M" "popData2019"    "continentExp"
##  [9] "Land_Area_sqkm" "HealthExp"      "GDPpc"          "Obese_Pcent"
## [13] "Over_65s"       "Diabetis"
```

You should have 14 variables. Also confirm that you do have 59 rows of data. All the new country data are from the year 2019.

Task 6: Standardisation

We now need to calculate the standardised personal transport variables.

```
data_avg <- data_avg %>% XXXX(
                            avg_rail_pp = avg_rail/popData2019 * 1000000,
                            avg_road_pp = XXXX,
                            avg_inland_pp = XXXX)
)
```

```
data_avg <- data_avg %>% mutate(
                            avg_rail_pp = avg_rail/popData2019 * 1000000,
                            avg_road_pp = avg_road/popData2019 * 1000000,
                            avg_inland_pp = avg_inland/popData2019 * 1000000)
```

You get it right if you can replicate the following.

```
data_avg %>%  filter(LOCATION %in% c("DEU","POL")) %>%
                select(LOCATION, avg_rail_pp, avg_road_pp, avg_inland_pp)
```

```
##   LOCATION avg_rail_pp avg_road_pp avg_inland_pp
## 1      DEU    752.3575    9082.167      9834.525
## 2      POL    843.5536    4885.718      5595.204
```

So the average number of miles traveled by rail per person in Germany is 752 miles per year. The average number of miles traveled on road per person in Poland is 4886. Convince yourself that the above calculation (`avg_rail_pp = avg_rail/popData2019 * 1000000`) did deliver miles per person. recall that the travel distance coming from the passenger transport datafile was measured in Millions of miles. The population is the actual population. If we didn't multiply by 1,000,000 we would get a measure of average millions of miles traveled per person.

Lastly, we will rescale the `GDPpc` variable. It is currently defined in USD, but to simplify later analysis we want to express it in 1000 USD. So for instance Albania's GDPpc is USD 5224, but we want to express it as 5.224 [1000 USD].

```
data_avg <- data_avg %>% mutate(GDPpc = GDPpc/1000)
```

Look at the dataset to confirm that your operation was successful.

Task 7: Creating League Tables

Create a league table of the countries in your dataset where people travel most and least by rail and road. You want to display the 10 countries with the most and the 10 countries with the least average travel per person in both categories.

Here is the code for the table for the top 10 rail travel nations.

```
task_7a <- data_avg %>%
select(country, avg_rail_pp) %>% arrange(desc(avg_rail_pp))
head(task_7a,10)
```

```
##        country avg_rail_pp
## 1        Japan   2925.0936
## 2  Switzerland   1589.8252
## 3       Latvia   1371.0047
## 4       Russia   1318.4358
## 5      Belarus   1188.9556
## 6      Ukraine   1153.9672
## 7       France   1116.1391
## 8      Hungary   1075.6537
## 9      Austria    987.6995
## 10  Kazakhstan    915.7539
```

Repeat this for the Top 10 road travel nations and equally find similar tables for the nations travelling least (but non-zero amounts) by rail and road.

```
task_7b <- data_avg %>%
select(country, avg_road_pp) %>% arrange(desc(avg_road_pp)) %>% print()
task_7c <- data_avg %>%
select(country, avg_rail_pp) %>% arrange(avg_rail_pp) %>% print()
task_7d <- data_avg %>%
select(country, avg_road_pp) %>% arrange(avg_road_pp) %>% print()
```

Task 8: Estimate regression models - Version 1

We shall estimate the following three regression models (`mod1`)

$$avg\_deaths\_p1M = \gamma_0 + \gamma_1\ GDPpc + w$$

and (`mod2`)

$$avg\_deaths\_p1M = \alpha_0 + \alpha_1 \; avg\_rail\_pp + v$$

and (mod3)

$$avg\_deaths\_p1M = \beta_0 + \beta_1 \; avg\_road\_pp + \beta_2 \; GDPpc + u$$

```
mod1 <- lm(XXXX ~ XXXX, data = data_avg)
mod2 <- lm(XXXX)
mod3 <- lm(XXXX)
stargazer_HC(mod1,mod2,mod3,omit.stat = "f")
```

```
mod1 <- lm(avg_deaths_p1M ~ GDPpc, data = data_avg)
mod2 <- lm(avg_deaths_p1M ~ avg_rail_pp, data = data_avg)
mod3 <- lm(avg_deaths_p1M ~ avg_road_pp+GDPpc, data = data_avg)
stargazer_HC(mod1,mod2,mod3,type_out="latex",omit.stat = "f")
```

% Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harvard.edu
% Date and time: Mon, Mar 14, 2022 - 09:52:20

Table 1:

	Dependent variable:		
	avg_deaths_p1M		
	(1)	(2)	(3)
avg_road_pp			0.003
			(0.002)
GDPpc	−0.718***		−1.218***
	(0.202)		(0.245)
avg_rail_pp		−0.002	
		(0.011)	
Constant	122.825***	103.763***	117.284***
	(6.962)	(6.241)	(10.919)
Observations	55	53	42
R^2	0.288	0.001	0.369
Adjusted R^2	0.274	−0.019	0.336
Residual Std. Error	29.346 (df = 53)	33.680 (df = 51)	29.034 (df = 39)
Note:			*p<0.1; **p<0.05; ***p<0.01
			Robust standard errors in parenthesis

If you have done this correctly, you will find that that your estimated constant for `mod1` is 122.82, the estimated constant for `mod2` is 103.763 and that `mod3` is estimated with 42 observations.

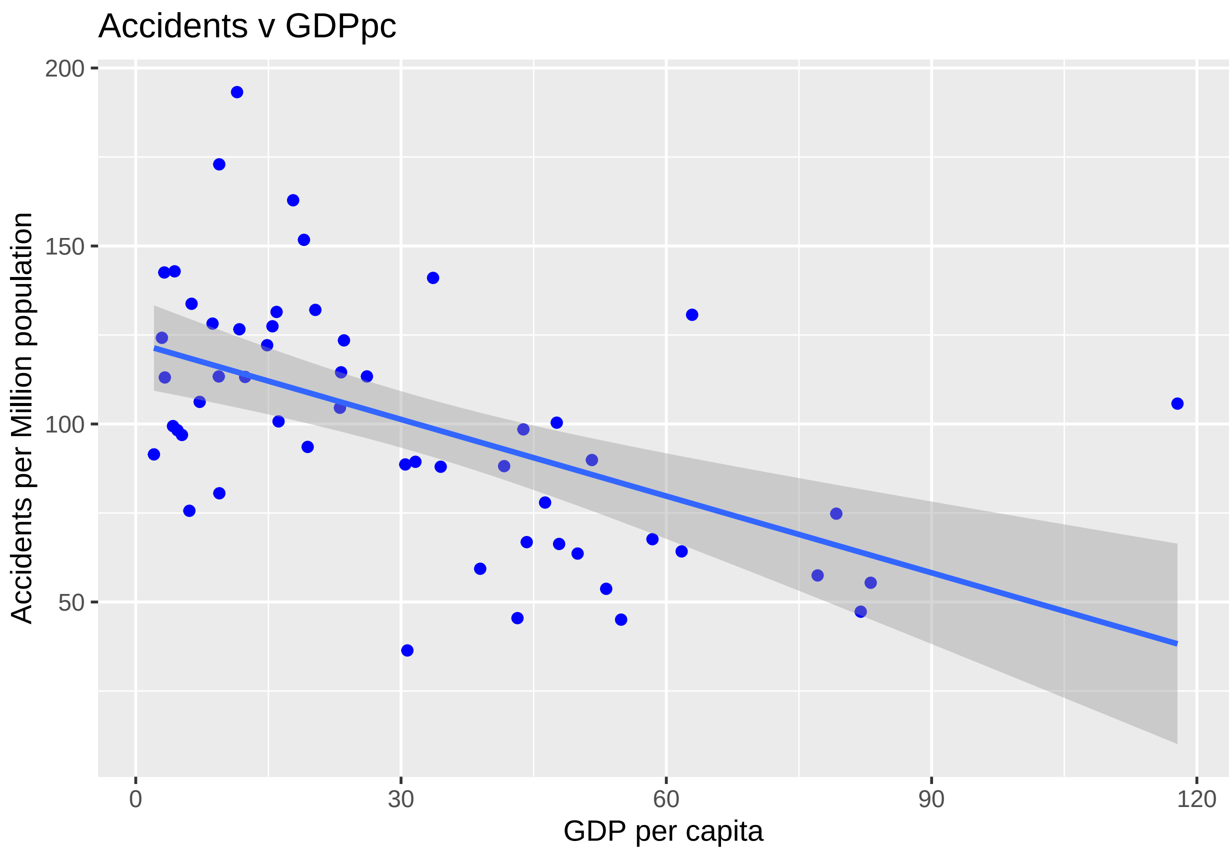
Task 9: Plot the data

Now we want to plot the `GDPpc` versus the `avg_deaths_p1M` data in a scatter plot. Replicate the graph below.

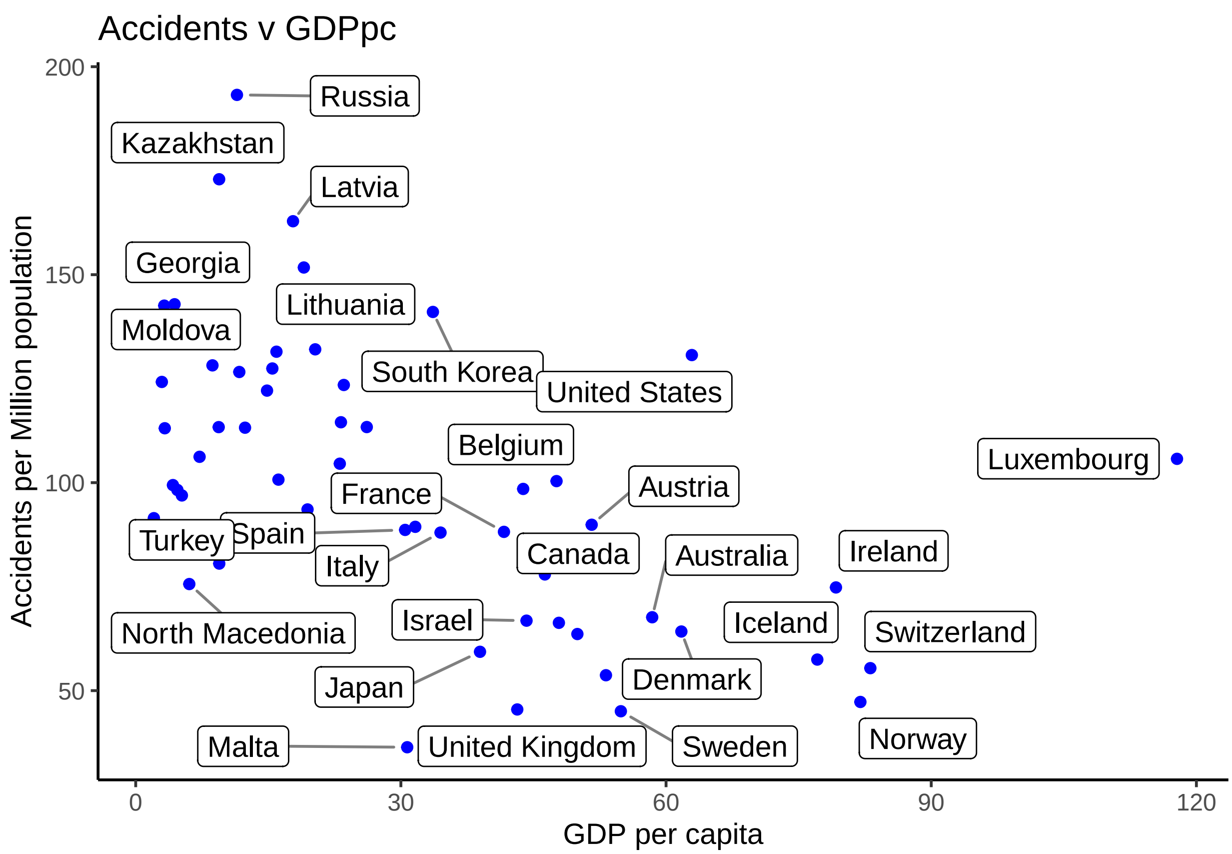
You should change the Axis labels and add a title as shown below. If you google you should find the appropriate commands to do so. (Think carefully about the search terms.) Also, what does the `geom_smooth(method='lm')` line achieve? Figure that out by running the code without that line.

```
ggplot(data_avg, aes(x=XXXX,y=XXXX)) +
  geom_point(color = "blue") +
  geom_smooth(method='lm') +
  XXXX +
  XXXX +
  XXXX
```

```
ggplot(data_avg, aes(x=GDPpc,y=avg_deaths_p1M)) +
  geom_point(color = "blue") +
  geom_smooth(method='lm') +
  ggtitle("Accidents v GDPpc") +
  ylab("Accidents per Million population") +
  xlab("GDP per capita")
```



Finally another searching challenge for you. The picture below has added data labels to the above plot. Find a way to achieve this. It is not important that the graph looks exactly like the one below, but you do want to find out how to add data labels. Dr. Google is your friend!



And does anyone know what is going on in Luxembourg?

END OF INSTRUCTIONS