Own Project: Spain Mortality

Anayansi Olivares

2/1/2021

Introduction

We all are aware of the many impacts that COVID-19 pandemic has had in our life. The world has faced confinements, economic losses, social distance, and many other sanitary measures to avoid the overload of the sanitary system and its consequences. One of the most dramatic fact is the increase of lives lost during this period. This drama is of course directly impacting the mortality data. Predicting metrics associated to COVID-19 pandemic, including *Number of Deaths*, has become a huge challenge for the Data Scientific Community during this period.

As part of the assignment of the course *HarvardX PH125.9x Data Science*, this report presents the process, the challenges, the insights and the results of a predictive model built around Spain's mortality data using *Number of Deaths* as the main metric. Machine learning strategies and algorithms have been used to produce the model. The metrics to evaluate its efficacy are:

1. ***Accuracy***
2. ***Root Mean Square Error (RMSE)***.

To comply with the course requirement, a linear regression model will be presented as baseline, and 3 more models will be built to improve prediction results. An additional objective will be to evaluate and analyze the impact of COVID-19 pandemic in the models built with previous mortality data. Therefore a first group of models will be built using data until 2019, and with those results, the same model will be applied, but with data that includes COVID-19 impact. These will be built to compere **Accuracy** and **RMSE** with the previous data models.

This document has four main sections:

1. **Data preparation:** This section shows the analysis applied to choose the data sources, the data transformation executed on the original raw data, and the construction of the training and test data sets. The purpose of the training data set is to train the machine learning model. Finally, the test_data set will be used to evaluate the algorithm and calculate the final Accuracy and RMSE.
2. **Data Analysis:** In this section, data exploration will be followed to better understand data behavior. Some insights to guide the decision of the Model approaches will be presented.
3. **Building and comparing models:** In this section, different models will be built and compared to each other in order to choose best options for previous COVID data. Due to technical restrictions in the hardware used, performance also has a role in the design of the final Model.
4. **Model selection, evaluation and conclusions:** This section shows the model generation using the previous COVID test_data, along with the final Accuracy and RMSE. Once the model is chosen, the same model will be run again with data that includes COVID impact. Some insights and final conclusions resulting from the project execution are also presented.

Data Preparation

The objective of this chapter is to produce four data sets: training and test, a pair with data until 2019, and another pair with data that includes COVID in 2020. Data preparation includes four sections:

1. Initialization
2. Data source selection
3. Data extraction and transformation
4. Data sets creation

Initialization

The initialization section includes the library calls required for the execution of the project. The libraries required are:

- **tidyverse:** widely used in all sections to transform data, manage data frames and produce graphics.
- **caret:** used to generate the data partition sets and to run models with different machine learning algorithms.
- **lubricate:** used to transform and manage time fields.
- **gam:** used for *Loess* algorithm.

```r
knitr::opts_chunk$set(
    echo = TRUE,
    message = FALSE,
    warning = FALSE
)
#  Library calls: Boolean function <<require>> is used to determine if the
#                 library has been installed. In case it has not, the proper
#                 <<install.package>> function is call to add it.
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project.org")
if(!require(gam)) install.packages("gam", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(caret)
library(lubridate)
library(gam)

#  Data visualization parameter
options(digits = 4)
```

Data source selection

The first challenge in this assignment was to search and select the data to be used. As mentioned in the previous section, the metric is *Number of Deaths* in Spain. Here the potential data sources analyzed:

1. **INE:** Stand for "Instituto Nacional de Estadística", which is the official statistics institution in Spain. The link address is as follows: https://ine.es/. These data sets were identified as potentially interesting:

- Spain official (final) number of deaths from 1975 to 2019. This data is classified by Year (annual data only), by Region, By Age and Sex. At first view, this data set is promising since it includes demographic data that can be used for prediction purposes.
- Spain official (final) number of deaths from 1975 to 2019. This data is classified by Year-Month (monthly data) and by Region. No further demographics details are provided.
- Spain provisional (not final) number of deaths for first semester 2020. In order to finalized the data, the INE follows a process that implies several months before achieving the "official" status. By the time this report was completed (beginning of 2021), only provisional data of 2020 for the first semester was found, and no data for the 2nd semester on their website. The data for the 1st semester of 2020 is classified by Month and Sex.

2. **The Human Mortality Database (HMD):** According to their statement, the HMD *".. was created to provide detailed mortality and population data to researchers, students, journalists, policy analysts, and others interested in the history of human longevity..".* They collect data from several countries, and currently they have published a group of data to be used for COVID analysis called ***"Short-term Mortality Fluctuations (STMF) data series"***. Their input data source can be found in the following link: https://www.mortality.org/Public/STMF/Inputs/STMFinput.zip. This zip data includes "ESPstmf.csv" which is the file for Spain with the following characteristics:

- Spain number of deaths from 2000 to 2021.
- This data is classified by Year, Week, Sex and Age-Interval (5-year age groups).
- The metadata explains that the input data source is **INE** and data from 2000 and 2018 is official, 2019 is provisional (not final), and 2020 data is estimated by INE. We are interested in real data and not estimations, therefore, 2020 with the COVID impact has been discarded as a potential data source.

3.- **MoMo:** It is an alert system created by the "Instituto de Salud Carlos III", an institution dedicated to the public health research in Spain. The system alert is feed on a daily basis from all computerized civil registries, which covers 93% of total Spanish data. This fact can be a constraint in comparing with other sources, since 7% is not available. The data can be found in the following: link https://momo.isciii.es/public/momo/data, with the following characteristics:

- Spain number of deaths from 2019 to 2021. The data available covers 2 years. For example on Feb 1st, data found will go from January 14th 2019 to January 31st 2021. Therefore data changed on daily basis not only to includes new data, but also delete older ones, to keep a 2 year window approach.
- The file contains real data (remember only 93%) and estimated data that they produce with their own models for alert purposes.
- The data is classified by Region, Sex, Date of death and Age in three groups (less than 65 years old, between 65 and 74, and older than 74 years old).

With the objective of choosing a model built with data before COVID, and then evaluating it with data from 2020, it is fundamental that the two sets are comparable between them. As mentioned before, the 2020 HMD has been discarded for being estimated and not real (observed) data.

The 2020 data of MoMo is very detailed, but it does not have an extensive history previous to 2020. The question is, will 2019 (with a few missing days) be enough for the model?. I think the final model should have some typical elements, but also the model will be totally biased by 2019 particularities. Therefore there is simply not enough historical data to establish seasonal behaviors, and make a reasonable model.

The next question is, will it make sense to use MoMO daily data of 2020, and join it with monthly or even annual data coming from INE? The fact that only 93% of the data is available, and the non-computerized civil register are distributed in multiples regions, implies that, either we estimate the gap in 2020, or decrease it in previous years to make them comparable. Both approaches compromise the accuracy of the COVID-19 evaluation established as an objective.

These constraints only leave us with the first semester of 2020 monthly data from INE, that can be easily joined with historical monthly data from 1975 to 2019, although Sex details will be lost, since it is not available in the second set (1975-2019). Please note that data (provisional) for the first wave of COVID-19, that deeply impacted Spain, is included, and therefore the objective can be achievable.

Data extraction and transformation

Now that our data sources have been chosen, two files should be extracted. To facilitate the extraction process, the original files from INE have been uploaded to Github (no changes in files has been done at this stage). The extraction and transformation process for each file is explained in the following sub-section.

Spain official (final) number of deaths from 1975 to 2019

The source information for the first file is presented bellow, along with the Github link.

- **File name:** 6562.csv
- **Link INE:** https://www.ine.es/dynt3/inebase/es/index.htm?padre=1132&capsel=1134
- **Options:** *"Por lugar de residencia. (Serie desde 1975). Total nacional y comunidades autónomas"* (From 1975 and by Region)
- **Link in Github:** https://raw.githubusercontent.com/anastoll/SpainMortalityData/main/6562.csv

The extraction process starts by opening a file connection that is then used to keep the file download from Github. At this moment the file is read and transformed into data frame R objects for manipulation purpose. An important point to consider is the usage of *fileEncoding = "UTF-8"*, to properly interpret the data, since it contains special characters that are part of Spanish Language (accents). Additionally, a number convention in Europe uses ", " as decimal separator and "." as thousands separator. This is indicated in reading time to avoid interpretation issues, for example read 1.000 as value one 1 instead of one thousand. The names of the columns are set in English.

```r
#  Data file:    Data file 6562.csv will be extracted from Github.
#   Open file connection
file <- tempfile()

#   Download the file input data and keep it as an R object.
download.file("https://raw.githubusercontent.com/anastoll/SpainMortalityData/main/6562.csv",
              file)

#   Reading file
spain_raw_ine_monthly <- read.csv(file, header = FALSE, sep = ";",
                                  col.names = c("Region", "Period", "Deaths"),
                                  dec=",",fileEncoding = "UTF-8")

#   Review of first rows
head(spain_raw_ine_monthly)
```

```
##       Region  Period Deaths
## 1  Autónomas Periodo  Total
## 2      Total 2019M12 36.739
## 3      Total 2019M11 34.706
## 4      Total 2019M10 32.770
## 5      Total 2019M09 29.916
## 6      Total 2019M08 31.671
```

The first row of the file contains the column names in Spanish, which needs to be deleted.

```
#   Eliminate column names in Spanish from file.
spain_raw_ine_monthly<-spain_raw_ine_monthly[-1,]
head(spain_raw_ine_monthly)
```

```
##   Region  Period Deaths
## 2  Total 2019M12 36.739
## 3  Total 2019M11 34.706
## 4  Total 2019M10 32.770
## 5  Total 2019M09 29.916
## 6  Total 2019M08 31.671
## 7  Total 2019M07 33.551
```

The transformation of decimal convention has to be done, please note that decimals are not expected in *number of deaths*"* observed metrics, as we can not have 0.5 deaths, but either 0 or 1. Therefore, we performed a replacement of "." into *void* string to eliminate the thousand separator, and finally convert to integer, with a quick verification that there are no NAs produced in the data.

```
#   Review of data type
str(spain_raw_ine_monthly)
```

```
## 'data.frame':    11340 obs. of  3 variables:
##  $ Region: chr  "Total" "Total" "Total" "Total" ...
##  $ Period: chr  "2019M12" "2019M11" "2019M10" "2019M09" ...
##  $ Deaths: chr  "36.739" "34.706" "32.770" "29.916" ...
```

```
#   Change European decimal standard to American standard in R
spain_raw_ine_monthly<- spain_raw_ine_monthly %>%
  mutate(Deaths= as.integer(str_replace_all(Deaths, "\\.", "")))

#   There are no Na values in Deaths.
sum(is.na(spain_raw_ine_monthly$Deaths))
```

```
## [1] 0
```

```
sum(spain_raw_ine_monthly$Deaths)
```

```
## [1] 31623656
```

The next transformation is on the Region field. The values extracted has Code and Name combined. Therefore, a separation is required. Region Code is numeric except for tow cases:

- *Extranjero:* Which refers to people who have died in Spain, but were officially resident in other countries. For these records we keep *Ex* as code, and the field will remain as characters.
- *Total:* Is the aggregation level for all regions. This value will be eliminated to avoid data duplication. For this, we will double check that *Total* sum of Deaths is equal to sum of Deaths in all Regions. Once this data verification is done, we simply filter non *Total* values. A reduction in number of records is expected (from 11340 to 1080). Finally, we verified that the sum of Deaths has the same value as *Total*.

```
#   Review of Region values
spain_raw_ine_monthly %>% distinct(Region)
```

```
##                              Region
## 1                             Total
## 2                      01 Andalucía
## 3                         02 Aragón
## 4       03 Asturias, Principado de
## 5                 04 Balears, Illes
## 6                       05 Canarias
## 7                      06 Cantabria
## 8                07 Castilla y León
## 9           08 Castilla - La Mancha
## 10                      09 Cataluña
## 11          10 Comunitat Valenciana
## 12                    11 Extremadura
## 13                        12 Galicia
## 14          13 Madrid, Comunidad de
## 15             14 Murcia, Región de
## 16 15 Navarra, Comunidad Foral de
## 17                    16 País Vasco
## 18                     17 Rioja, La
## 19                          18 Ceuta
## 20                        19 Melilla
## 21                        Extranjero
```

```
#   Separate Region code from Region Name
spain_raw_ine_monthly <- spain_raw_ine_monthly %>%
  mutate(RegionCode = str_sub(Region, 1, 2),
         RegionName = ifelse(Region=="Extranjero","Extranjero",
                             str_sub(Region, 4, str_length(Region))))

#   Verification of transformation
spain_raw_ine_monthly %>% distinct(Region, RegionCode, RegionName)
```

```
##                              Region RegionCode                 RegionName
## 1                             Total         To                         al
## 2                      01 Andalucía         01                  Andalucía
## 3                         02 Aragón         02                     Aragón
## 4       03 Asturias, Principado de         03     Asturias, Principado de
## 5                 04 Balears, Illes         04             Balears, Illes
## 6                       05 Canarias         05                   Canarias
## 7                      06 Cantabria         06                  Cantabria
## 8                07 Castilla y León         07            Castilla y León
## 9           08 Castilla - La Mancha         08       Castilla - La Mancha
## 10                      09 Cataluña         09                   Cataluña
## 11          10 Comunitat Valenciana         10       Comunitat Valenciana
## 12                    11 Extremadura         11                Extremadura
## 13                        12 Galicia         12                    Galicia
## 14          13 Madrid, Comunidad de         13       Madrid, Comunidad de
## 15             14 Murcia, Región de         14          Murcia, Región de
## 16 15 Navarra, Comunidad Foral de         15 Navarra, Comunidad Foral de
## 17                    16 País Vasco         16                 País Vasco
## 18                   17 Rioja, La         17                   Rioja, La
## 19                       18 Ceuta         18                       Ceuta
## 20                     19 Melilla         19                     Melilla
## 21                     Extranjero         Ex                  Extranjero
```

```
#   Eliminate "Total" Values in Region to avoid duplicates
spain_raw_ine_monthly %>% filter(Region=="Total") %>% summarise(Deaths=sum(Deaths))
```

```
##     Deaths
## 1 15811828
```

```
spain_raw_ine_monthly %>% filter(Region!="Total") %>% summarise(Deaths=sum(Deaths))
```

```
##     Deaths
## 1 15811828
```

```
#   Same sum value for "Total" and for  detailed data. There are 11340 records
#   before transformation.
spain_raw_ine_monthly<- spain_raw_ine_monthly %>% filter(Region!="Total")

#   Same total Deaths value is kept. And number of records downs to 10800
spain_raw_ine_monthly %>% summarise(Deaths=sum(Deaths))
```

```
##     Deaths
## 1 15811828
```

The next transformation is related to Year and Month values, which were combined in the file as *Period* column. The Year information is in a 4 digit format in the first 4 positions of the field. The month is represented with the letter *M* in the position 5 of the field, and two digits between position 6-7. We will ignore the position 5, and extract Month as a numeric value from the two last positions.

This transformation will complete the process for the first file *6562.csv*.

```
#   Review of Period values
head(spain_raw_ine_monthly %>% distinct(Period))
```

```
##    Period
## 1 2019M12
## 2 2019M11
## 3 2019M10
## 4 2019M09
## 5 2019M08
## 6 2019M07
```

```
#   Separate Year and Month from Period
spain_raw_ine_monthly <- spain_raw_ine_monthly %>%
  mutate(Year = as.integer(str_sub(Period, 1, 4)),
         Month = as.integer(str_sub(Period, 6, 7)))

#   Review of separation results
str(spain_raw_ine_monthly)
```

```
## 'data.frame':    10800 obs. of  7 variables:
##  $ Region    : chr  "01 Andalucía" "01 Andalucía" "01 Andalucía" "01 Andalucía" ...
##  $ Period    : chr  "2019M12" "2019M11" "2019M10" "2019M09" ...
##  $ Deaths    : int  6338 5849 5534 5013 5256 5458 5182 5804 5822 6265 ...
##  $ RegionCode: chr  "01" "01" "01" "01" ...
##  $ RegionName: chr  "Andalucía" "Andalucía" "Andalucía" "Andalucía" ...
##  $ Year      : int  2019 2019 2019 2019 2019 2019 2019 2019 2019 2019 ...
##  $ Month     : int  12 11 10 9 8 7 6 5 4 3 ...
```

```
#   Review of separation results Year
head(spain_raw_ine_monthly %>% distinct(Year))
```

```
##   Year
## 1 2019
## 2 2018
## 3 2017
## 4 2016
## 5 2015
## 6 2014
```

```
tail(spain_raw_ine_monthly %>% distinct(Year))
```

```
##    Year
## 40 1980
## 41 1979
## 42 1978
## 43 1977
## 44 1976
## 45 1975
```

```
#   Review of separation results Month
spain_raw_ine_monthly %>% distinct(Month)
```

```
##    Month
## 1     12
## 2     11
## 3     10
## 4      9
## 5      8
## 6      7
## 7      6
## 8      5
## 9      4
## 10     3
## 11     2
## 12     1
```

Spain provisional number of deaths for first semester 2020.

The source information for the second file is presented bellow, along with the Github link.

- **File name:** 02003.csv

- **Link INE:** https://www.ine.es/dyngs/INEbase/es/operacion.htm?c=Estadistica_C&cid=1254736177008&menu=resultados&idp=1254735573002#!tabs-1254736195546
- **Options:** "Defunciones por comunidad autónoma de residencia del fallecido y mes" (Region and month).
- **Link in Github:** https://raw.githubusercontent.com/anastoll/SpainMortalityData/main/02003.csv

The extraction process is similar to the previous on. It starts by opening a file connection that is then used to keep the file download from Github. The data type of the file read is data frame. The same consideration related to *fileEncoding = "UTF-8"*, and number convention in Europe for decimal "," have to be taken into account. The names of the columns are set in English, and first row with Spanaish column name is removed.

```
#  Data file:    Data file will be extracted from Github.

#   Open file connection
file2 <- tempfile()

#   Download the file input data and keep it as an R object.
download.file("https://raw.githubusercontent.com/anastoll/SpainMortalityData/main/02003.csv",
              file2)

#   Reading file
spain_raw_ine_2020 <- read.csv(file2,
                                 header = FALSE,
                                 sep = ";",
                                 col.names = c("Region", "MonthESP", "Sex","Deaths"),
                                 dec=",",
                                 fileEncoding = "UTF-8")

#   Review of first rows
head(spain_raw_ine_2020)
```

```
##                      Region MonthESP     Sex  Deaths
## 1  residencia del fallecido      Mes    Sexo   Total
## 2                     Total    Total   Total 262.373
## 3                     Total    Total Varones 131.326
## 4                     Total    Total Mujeres 131.047
## 5                     Total    Enero   Total  42.807
## 6                     Total    Enero Varones  21.401
```

```
#   Eliminate column names in Spanish from file.
spain_raw_ine_2020<-spain_raw_ine_2020[-1,]
head(spain_raw_ine_2020)
```

```
##   Region MonthESP     Sex  Deaths
## 2  Total    Total   Total 262.373
## 3  Total    Total Varones 131.326
## 4  Total    Total Mujeres 131.047
## 5  Total    Enero   Total  42.807
## 6  Total    Enero Varones  21.401
## 7  Total    Enero Mujeres  21.406
```

```
#   Review of data types
str(spain_raw_ine_2020)
```

```
## 'data.frame':    441 obs. of  4 variables:
##  $ Region  : chr  "Total" "Total" "Total" "Total" ...
##  $ MonthESP: chr  "Total" "Total" "Total" "Enero" ...
##  $ Sex     : chr  "Total" "Varones" "Mujeres" "Total" ...
##  $ Deaths  : chr  "262.373" "131.326" "131.047" "42.807" ...
```

The next transformation step is to eliminate the European number convention for thousands symbol ".", and convert Deaths into integer. A verification of NAs is done to ensure none are produced.

```
#   Change European decimal standard to American standard in R
spain_raw_ine_2020<- spain_raw_ine_2020 %>%
  mutate(Deaths= as.integer(str_replace_all(Deaths, "\\.", "")))

summary(spain_raw_ine_2020)
```

```
##     Region            MonthESP              Sex                Deaths
##  Length:441         Length:441         Length:441         Min.   :    15
##  Class :character   Class :character   Class :character   1st Qu.:   407
##  Mode  :character   Mode  :character   Mode  :character   Median :  1028
##                                                           Mean   :  4760
##                                                           3rd Qu.:  2970
##                                                           Max.   :262373
```

```
#   There are no Na values in Deaths.
sum(is.na(spain_raw_ine_2020$Deaths))
```

```
## [1] 0
```

```
sum(spain_raw_ine_2020$Deaths)
```

```
## [1] 2098984
```

In this file, the region data only includes the Region Name. There is not need to separate the Region Code as before. Additionally, *Total* aggregated value is identified, and it should be eliminated to avoid duplicate data.

```
#   Review of Region values
spain_raw_ine_2020 %>% distinct(Region)
```

```
##                         Region
## 1                        Total
## 2                    Andalucía
## 3                       Aragón
## 4      Asturias, Principado de
## 5               Balears, Illes
## 6                     Canarias
## 7                    Cantabria
## 8              Castilla y León
## 9           Castilla-La Mancha
## 10                    Cataluña
## 11        Comunitat Valenciana
## 12                 Extremadura
## 13                     Galicia
## 14        Madrid, Comunidad de
## 15           Murcia, Región de
## 16 Navarra, Comunidad Foral de
## 17                  País Vasco
## 18                   Rioja, La
## 19                       Ceuta
## 20                     Melilla
## 21                  Extranjero
```

```
#   In this case, there is not Region Code.

#   Eliminate "Total" Values in Region to avoid duplicates
spain_raw_ine_2020 %>% filter(Region=="Total") %>% summarise(Deaths=sum(Deaths))
```

```
##    Deaths
## 1 1049492
```

```
spain_raw_ine_2020 %>% filter(Region!="Total") %>% summarise(Deaths=sum(Deaths))
```

```
##    Deaths
## 1 1049492
```

```
#   Same sum value for "Total" and fo  details data. There are 441 records
#   before transformation.
spain_raw_ine_2020<- spain_raw_ine_2020 %>% filter(Region!="Total")

#   Same total Deaths value is kept. And number of records downs to 420
spain_raw_ine_2020 %>% summarise(Deaths=sum(Deaths))
```

```
##    Deaths
## 1 1049492
```

Once Region transformation has been completed, it is the turn for Month data. In this file, month data is presented with its name in Spanish. Therefore a transformation to numeric needs to be done. Additionally, *Total* aggregated value is also found. We will start by eliminating this value, and then we will simply translate the name of the month to its correspondence number (January to 1, February to 2, etc).

```
#   Review of Month values
spain_raw_ine_2020 %>%
  group_by(MonthESP) %>%
  summarise(n=n())
```

```
## # A tibble: 7 x 2
##   MonthESP     n
##   <chr>    <int>
## 1 Abril       60
## 2 Enero       60
## 3 Febrero     60
## 4 Junio       60
## 5 Marzo       60
## 6 Mayo        60
## 7 Total       60
```

```
#   Eliminate "Total" Values in MonthESP to avoid duplicates
spain_raw_ine_2020 %>% filter(MonthESP=="Total") %>% summarise(Deaths=sum(Deaths))
```

```
##   Deaths
## 1 524746
```

```
spain_raw_ine_2020 %>% filter(MonthESP!="Total") %>% summarise(Deaths=sum(Deaths))
```

```
##   Deaths
## 1 524746
```

```
#   Same sum value for "Total" and fo  details data. There are 420 records
#   before transformation.
spain_raw_ine_2020<- spain_raw_ine_2020 %>% filter(MonthESP!="Total")

#   Same total Deaths value is kept. And number of records downs to 360
spain_raw_ine_2020 %>% summarise(Deaths=sum(Deaths))
```

```
##   Deaths
## 1 524746
```

```
#   We need to transform Spanish month name into numbers
spain_raw_ine_2020 <- spain_raw_ine_2020 %>%
  mutate( Month = ifelse(MonthESP=="Enero",1,
                       ifelse(MonthESP=="Febrero",2,
                              ifelse(MonthESP=="Marzo",3,
                                      ifelse(MonthESP=="Abril",4,
                                             ifelse(MonthESP=="Mayo",5,6))))))



#   Verify of Month values
spain_raw_ine_2020 %>%
  group_by(Month,MonthESP) %>%
  summarise(n=n())
```

```
## # A tibble: 6 x 3
## # Groups:   Month [6]
##   Month MonthESP     n
##   <dbl> <chr>    <int>
## 1     1 Enero       60
## 2     2 Febrero     60
## 3     3 Marzo       60
## 4     4 Abril       60
## 5     5 Mayo        60
## 6     6 Junio       60
```

```r
str(spain_raw_ine_2020)
```

```
## 'data.frame':    360 obs. of  5 variables:
##  $ Region  : chr  "Andalucía" "Andalucía" "Andalucía" "Andalucía" ...
##  $ MonthESP: chr  "Enero" "Enero" "Enero" "Febrero" ...
##  $ Sex     : chr  "Total" "Varones" "Mujeres" "Total" ...
##  $ Deaths  : int  7602 3902 3700 6294 3173 3121 7014 3574 3440 6608 ...
##  $ Month   : num  1 1 1 2 2 2 3 3 3 4 ...
```

The last transformation of the file is related to the Sex columns. As in the Month case, the Sex data is presented as word in Spanish. A simple transformation to value *m* for male and *f* for female will be done. This field also has a *Total* aggregated value that should be removed.

```r
#   Review of Sex values
spain_raw_ine_2020 %>%
  group_by(Sex) %>%
  summarise(n=n())
```

```
## # A tibble: 3 x 2
##   Sex         n
##   <chr>   <int>
## 1 Mujeres   120
## 2 Total     120
## 3 Varones   120
```

```r
#   Eliminate "Total" Values in Sex to avoid duplicates
spain_raw_ine_2020 %>% filter(Sex=="Total") %>% summarise(Deaths=sum(Deaths))
```

```
##   Deaths
## 1 262373
```

```r
spain_raw_ine_2020 %>% filter(Sex!="Total") %>% summarise(Deaths=sum(Deaths))
```

```
##   Deaths
## 1 262373
```

```r
#   Same sum value for "Total" and fo  details data. There are 360 records
#   before transformation.
spain_raw_ine_2020<- spain_raw_ine_2020 %>% filter(Sex!="Total")

#   Same total Deaths value is kept. And number of records downs to 240
spain_raw_ine_2020 %>% summarise(Deaths=sum(Deaths))
```

```
##   Deaths
## 1 262373
```

```r
#   Change values "Varones" by "m" (male) and "Mujeres" by "f" (female)
spain_raw_ine_2020<- spain_raw_ine_2020 %>%
  mutate(Sex=ifelse(Sex=="Varones","m","f"))
```

```
#   Verification
spain_raw_ine_2020 %>%
  group_by(Sex) %>%
  summarise(n=n())
```

```
## # A tibble: 2 x 2
##   Sex       n
##   <chr> <int>
## 1 f       120
## 2 m       120
```

Unification of historical data (1975-2019) with first semester 2020.

The unified data set has the following columns: *RegionName, Year, Month, Deaths*. The first step is to verify that all data is compatible.

For RegionName, a verification of value is done detecting issues in *"Castilla-La Mancha"* value due to blank spaces. This value is replaced for *"Castilla - La Mancha"*.

For Year values, we know they have a numeric data type in the historical data set (1975-2019). There is not *Year* column in the second file, but we know that all records corresponds to the year 2020. A Year field with the value 2020 value needs to be added. For Month data, we know that in both cases they are numeric from 1 to 12. No further transformations are required before the unification.

The data is unified in a single data set that includes records from 1975 to 2020/1 (first semester). Having 2020 incomplete (only first semester) makes it impossible to carry out the analysis by Year. Therefore, a calculated field with the semester value is added to facilitate the analysis phase.

```
#   Review if Region Name are the same. We should have 20 items
spain_raw_ine_2020 %>% mutate(RegionName=Region) %>%
  distinct(RegionName) %>% inner_join(spain_raw_ine_monthly%>%distinct(RegionName),by ="RegionName")
```

```
##                        RegionName
## 1                       Andalucía
## 2                          Aragón
## 3      Asturias, Principado de
## 4                 Balears, Illes
## 5                        Canarias
## 6                       Cantabria
## 7                Castilla y León
## 8                        Cataluña
## 9         Comunitat Valenciana
## 10                    Extremadura
## 11                        Galicia
## 12         Madrid, Comunidad de
## 13            Murcia, Región de
## 14 Navarra, Comunidad Foral de
## 15                     País Vasco
## 16                      Rioja, La
## 17                          Ceuta
## 18                        Melilla
## 19                     Extranjero
```

```r
#   Value related to Region Castilla-La Mancha is missing. There is blank space
#   defference that should be transformed.
spain_raw_ine_2020<- spain_raw_ine_2020 %>%
  mutate(RegionName=ifelse(Region=="Castilla-La Mancha",
                           "Castilla - La Mancha",
                           Region))

#   Verify if Region Name are the same. We should have 20 items
spain_raw_ine_2020 %>% distinct(RegionName) %>%
  inner_join(spain_raw_ine_monthly%>%distinct(RegionName),by ="RegionName")
```

```
##                     RegionName
## 1                    Andalucía
## 2                       Aragón
## 3      Asturias, Principado de
## 4               Balears, Illes
## 5                     Canarias
## 6                    Cantabria
## 7              Castilla y León
## 8         Castilla - La Mancha
## 9                     Cataluña
## 10        Comunitat Valenciana
## 11                 Extremadura
## 12                     Galicia
## 13        Madrid, Comunidad de
## 14           Murcia, Región de
## 15 Navarra, Comunidad Foral de
## 16                  País Vasco
## 17                   Rioja, La
## 18                       Ceuta
## 19                     Melilla
## 20                  Extranjero
```

```r
#  Let's unify monthly data from 1975 until 2019 with the first semester of 2020
#  and call it Covid impact (since it is the only data we have that includes
#  deaths for Covid). Fields to be combined: RegionName, Year, Month and Deaths.

#   First steps is add Year to 2020 data
spain_raw_ine_2020<- spain_raw_ine_2020 %>%
  mutate(Year=2020)

#   Bind all rows for year-months combinations avalaible
spain_covid_impact <- bind_rows(spain_raw_ine_monthly%>%select(RegionName, Year, Month, Deaths),
                                spain_raw_ine_2020%>%select(RegionName, Year, Month, Deaths))


#   Review bind result
str(spain_covid_impact)
```

```
## 'data.frame':    11040 obs. of  4 variables:
##  $ RegionName: chr  "Andalucía" "Andalucía" "Andalucía" "Andalucía" ...
##  $ Year      : num  2019 2019 2019 2019 2019 ...
##  $ Month     : num  12 11 10 9 8 7 6 5 4 3 ...
##  $ Deaths    : int  6338 5849 5534 5013 5256 5458 5182 5804 5822 6265 ...
```

```
#  Since we have only first semester, let's add the semester to help analysis.
spain_covid_impact <- spain_covid_impact %>%
  mutate(Semester = ifelse(Month<7,1,2))
```

Data sets creation

This is the final part of the Data preparation phase. Here we will create 4 data sets:

1. *Data from 1975-2019:* This data will be used to select the design of the model. Two partitions will be created:

- *Training data set:* A partition with 80% of the data. It will be used for data analysis, for training the model, and for cross validation.
- *Testing data set:* A partition with 20% of the data. It will be used for validation. *Accuracy* and *RMSE* will be calculated with this data.

2. *Data from 1975-2020/1:* This data will be used to evaluate the model selected on this set of data. Two partitions will be created:

- *Training data set:* A partition with 80% of the data. It will be used for data analysis and for re-training the model. However the tuning parameter will be the one selected in the original model.
- *Testing data set:* A partition with 20% of the data. It will be used for validation. *Accuracy* and *RMSE* will be calculated with this data.

```
#  Models with Monthly Data until 2019
#    Setting seed 1. if you are not using R 3.6 or later, please remove
#    sample.kind parameter in seed definition
set.seed(1, sample.kind = "Rounding")

#  Variable to be predicted
y1 <- spain_raw_ine_monthly$Deaths

# Creation of 2 partitions, 80% of data (training) and the test one with 20%
test_index1 <- createDataPartition(y1, times = 1, p = 0.2, list = FALSE)

train_set_monthly <- spain_raw_ine_monthly %>% slice(-test_index1)
test_set_monthly <- spain_raw_ine_monthly %>% slice(test_index1)

#  Models with Monthly Data until First semester 2020

#    Setting seed 1. if you are not using R 3.6 or later, please remove
#    sample.kind parameter in seed definition
set.seed(1, sample.kind = "Rounding")

#  Variable to be predicted
y2 <- spain_covid_impact$Deaths

# Creation of 2 partitions, 80% of data (training) and the test one with 20%
test_index2 <- createDataPartition(y2, times = 1, p = 0.2, list = FALSE)
```

```r
train_set_covid <- spain_covid_impact %>% slice(-test_index2)
test_set_covid <- spain_covid_impact %>% slice(test_index2)
```

Data analysis

Even though the number of predictors in the data selected is relative low, there are many machine learning algorithms and strategy choices. A way to structure this search is to start by exploring the data, getting to better understand its behavior and identify potential opportunities and issues.

During the data analysis process, questions and hypotheses will be raised, some of which will be discarded, and others which will guide the model approach. This analysis will be performed only on the *train* data sets.

Data from 1975-2019 (previous to COVID-19)

The analysis will be done with in the following order:

1. General distribution
2. Semester analysis
3. Month analysis
4. Region

General distribution

To better understand the mortality data, the first step is to review its distribution.

```
#  Let's start by reviewing the mean and quartiles
summary(train_set_monthly)
```

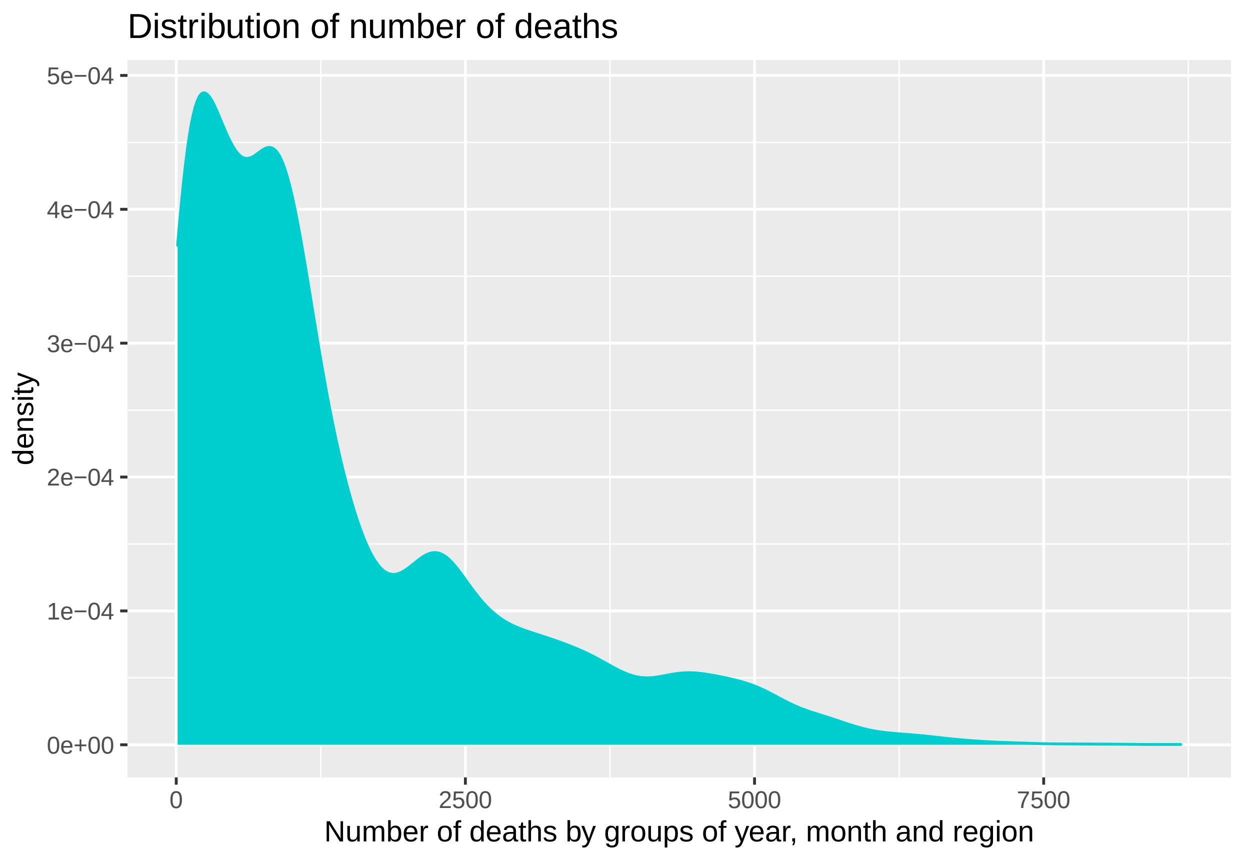
```
##     Region             Period              Deaths      RegionCode
##  Length:8639        Length:8639        Min.   :  12   Length:8639
##  Class :character   Class :character   1st Qu.: 398   Class :character
##  Mode  :character   Mode  :character   Median : 958   Mode  :character
##                                        Mean   :1460
##                                        3rd Qu.:2174
##                                        Max.   :8685
##   RegionName             Year          Month
##  Length:8639        Min.   :1975   Min.   : 1.0
##  Class :character   1st Qu.:1986   1st Qu.: 4.0
##  Mode  :character   Median :1997   Median : 6.0
##                     Mean   :1997   Mean   : 6.5
##                     3rd Qu.:2008   3rd Qu.: 9.0
##                     Max.   :2019   Max.   :12.0
```

```
#  Standard deviation
sd(train_set_monthly$Deaths)
```

```
## [1] 1438
```

The result shows that for Year-month and Region groups, the *Deaths* data goes from interval values 12 to 8685, the mean is *1460*, a median (percentile 50%) at value *958*, lower than the mean, which indicates a right-skewed distribution. The standard deviation is 1438, which is quite high, indicating that the data is dispersed. To confirm this visually, the distribution plot is presented bellow.

```
#   Distribution graph
train_set_monthly %>%
  ggplot(aes(x=Deaths)) +
  geom_density(color="cyan3", fill="cyan3") +
  xlab("Number of deaths by groups of year, month and region")+
  ggtitle("Distribution of number of deaths")
```



In effect, the data is concentrated bellow 2174 (percentile 75) and it is disperse in the last quartile.

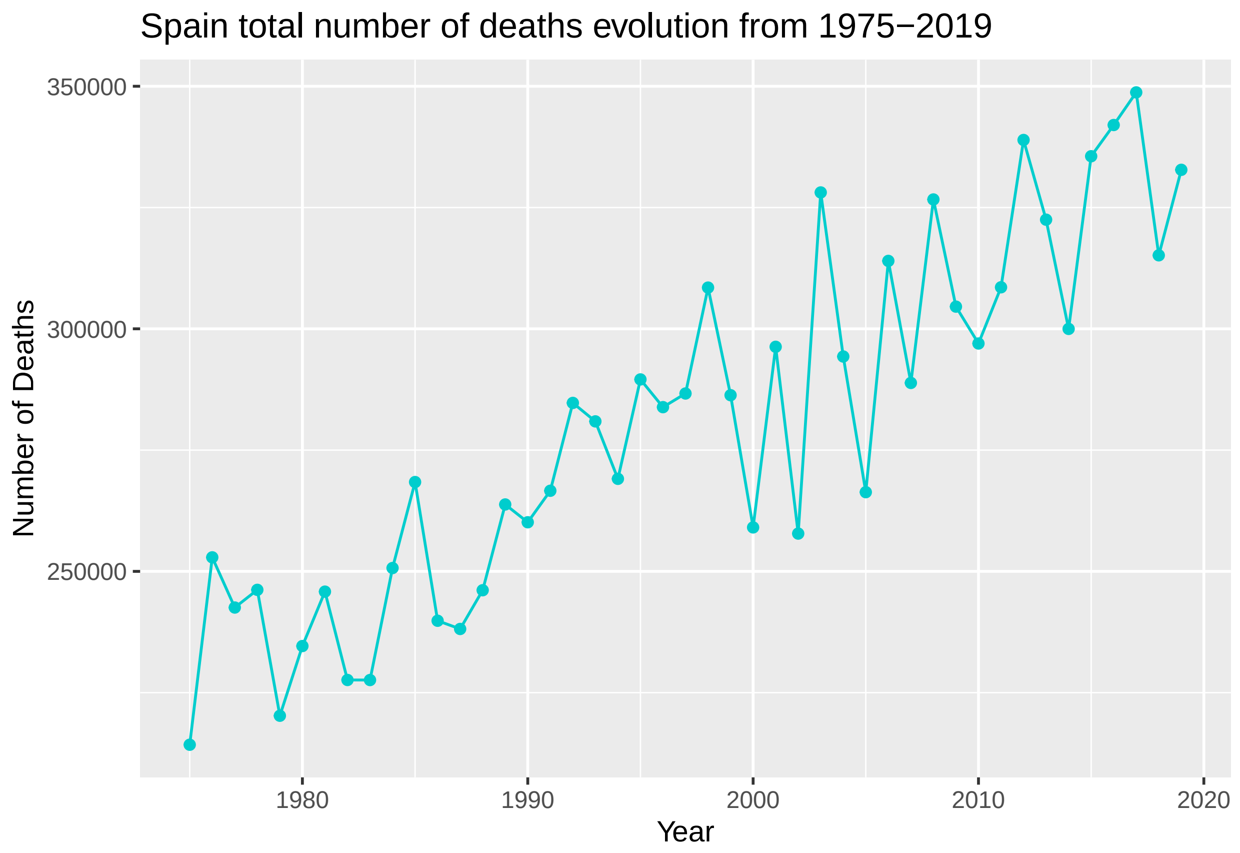
Now, let's deep dive in the predictors analysis. It will be done in the following order:

1. Year analysis
2. Month analysis
3. Region

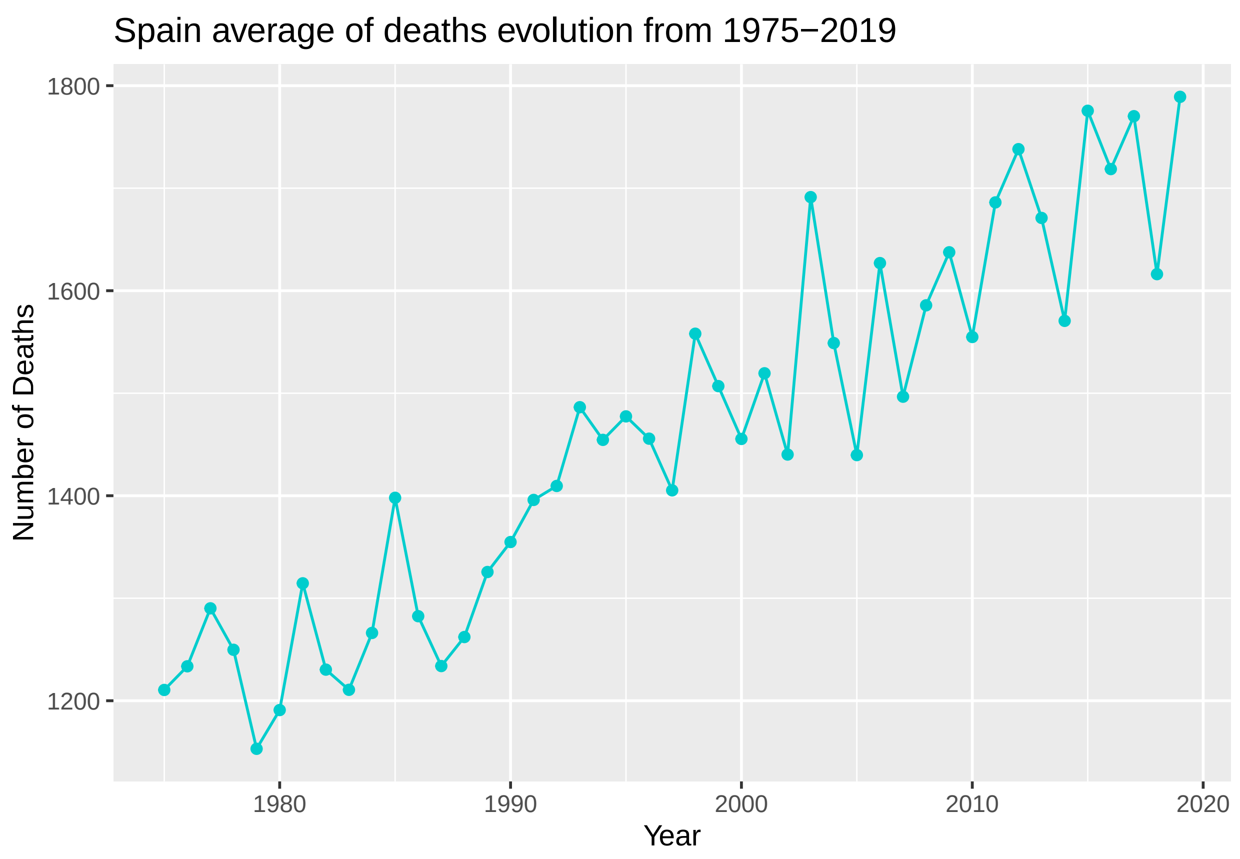
Year analysis

The first step is to review the evolution of Deaths from 1975 to 2019.

```
## [1] 0.1158
```



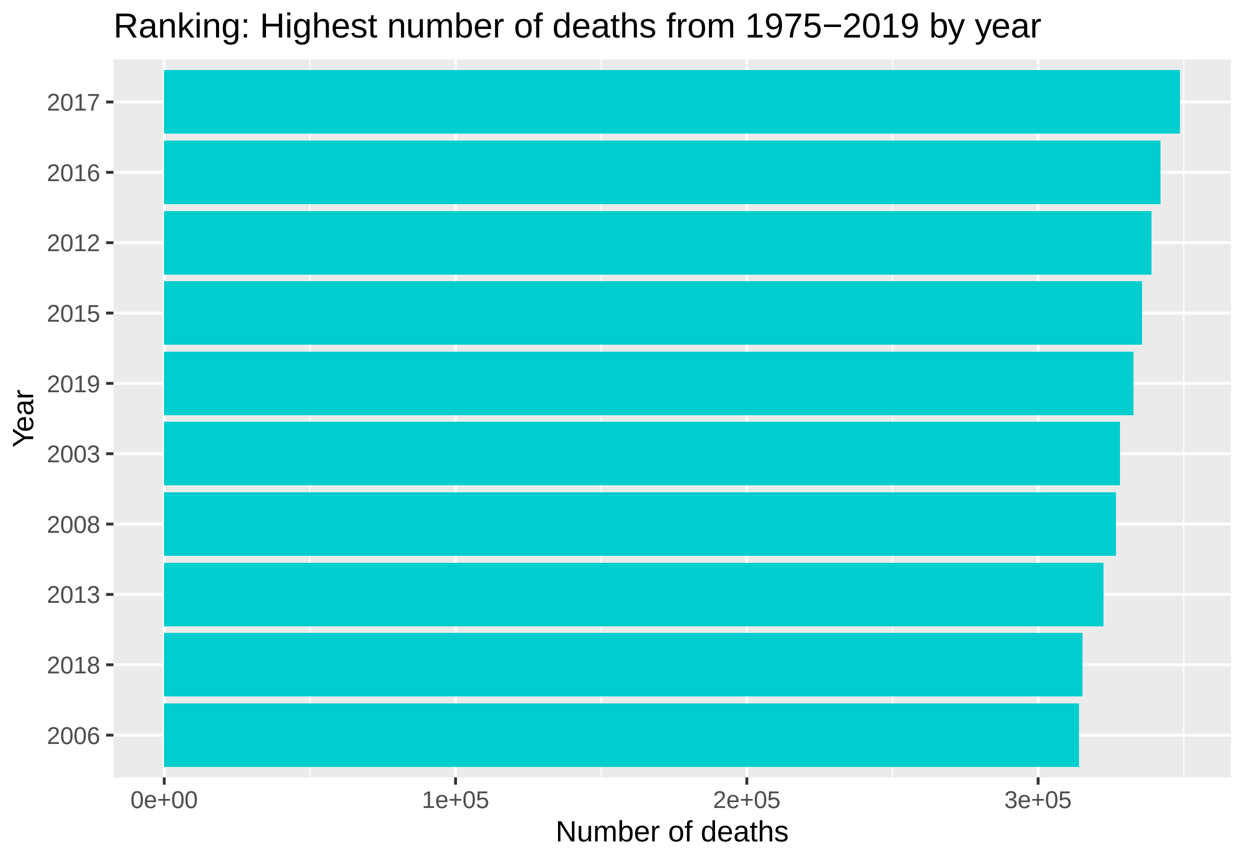
The correlation result is not particular high. The graphs shows a global incremental tendency, but with variance (jumps), especially from the year 1998 to 2006. Let's have a look at to the *mean* evolution.



As expected, the same global incremental tendency is clearly shown, with some jumps after 1998. The lowest *mean* is presented in Year 1979 Year, and the highest in 2019. The evolution between 1987 and 1993 is quite linear.

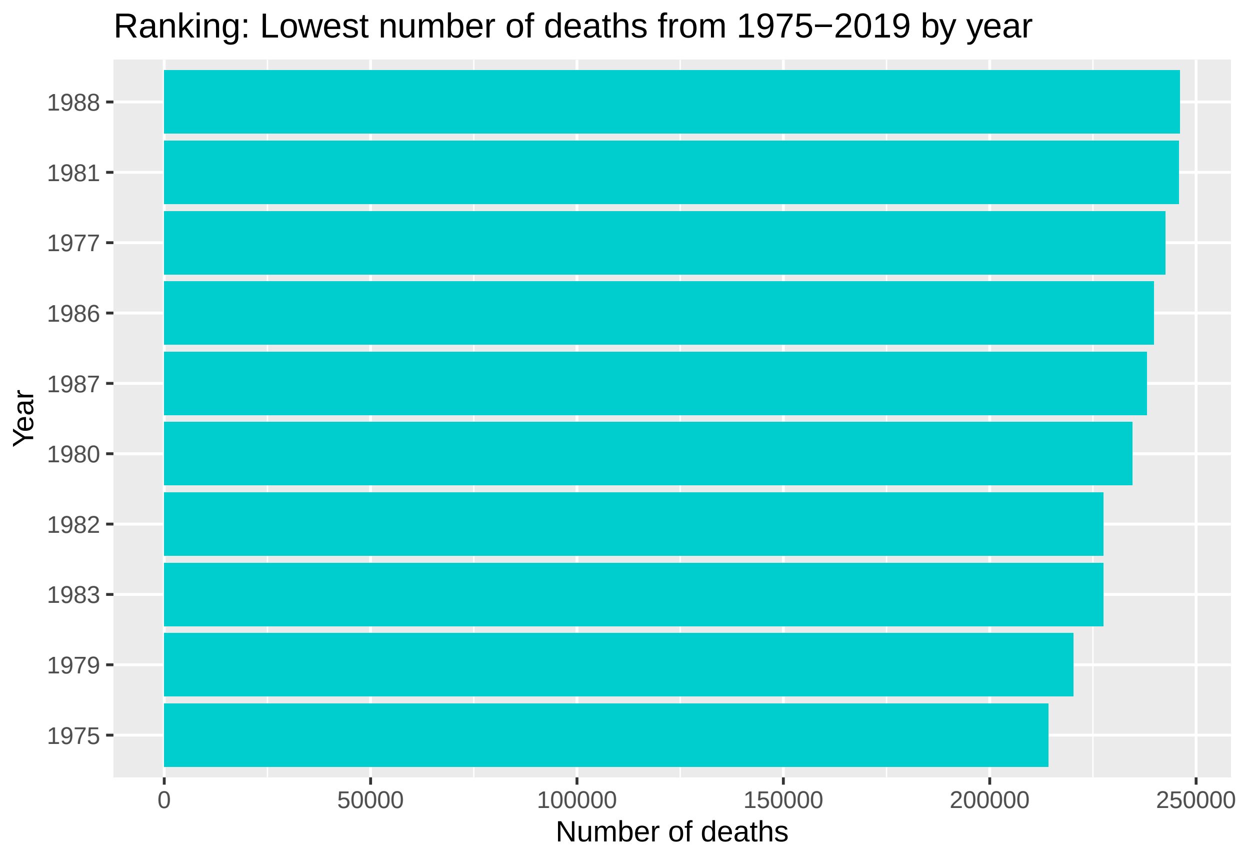
Now let's have a closer view at the top ten.

```
#    Ranking by year - 10 highest
train_set_monthly%>%
  group_by(Year) %>%
  summarise(Deaths=sum(Deaths)) %>%
  mutate(YearRanking=fct_reorder(factor(Year), Deaths))  %>%
  arrange(desc(Deaths)) %>%
  slice(1:10) %>%
  ggplot(aes(x =YearRanking, y=Deaths)) +
  geom_bar(stat = "identity", fill="cyan3") +
  xlab("Year") +
  ylab("Number of deaths") +
  coord_flip() +
  ggtitle("Ranking: Highest number of deaths from 1975-2019 by year")
```



The last 5 years are included in the ranking (2015-2019), which is expected due to the tendency shown in the evolution chart. A few early 2000's years, such as 2003 (in 6th position) and 2006 (in 10th position) are included in the top 10. Finanlly, the difference between the ranked years seems small.

Now let's have continue with the bottom 10.



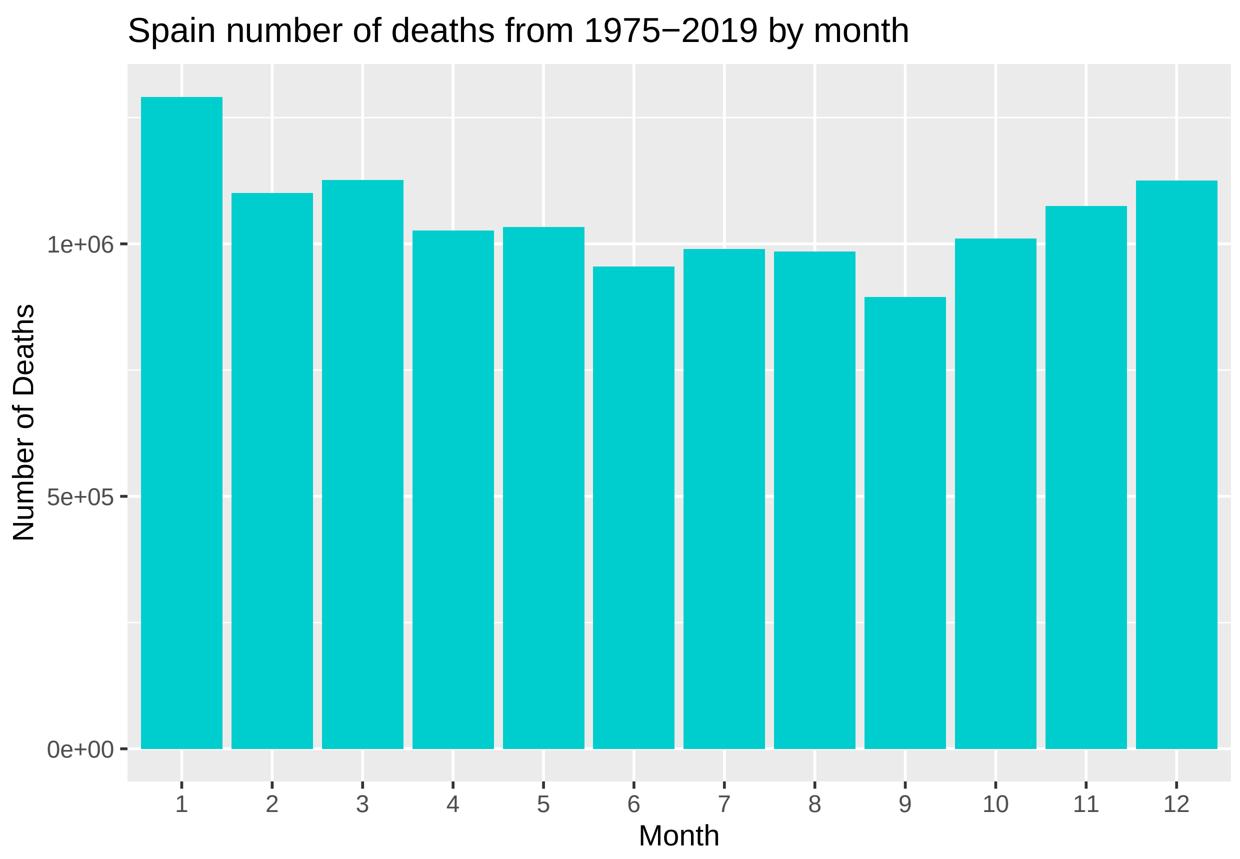
The year 1975 is the lowest, followed by 1979 and 1983. Years 1976 and 1978 are not in the bottom 10, neither are 1984 and 1985. The latest year included in the ranking, and with the highest number, is 1988.

As a conclusion, we can state that even if the relation between Year and number of Deaths is not "totally direct", there is a clear tendency to increase the number of deaths across the years.

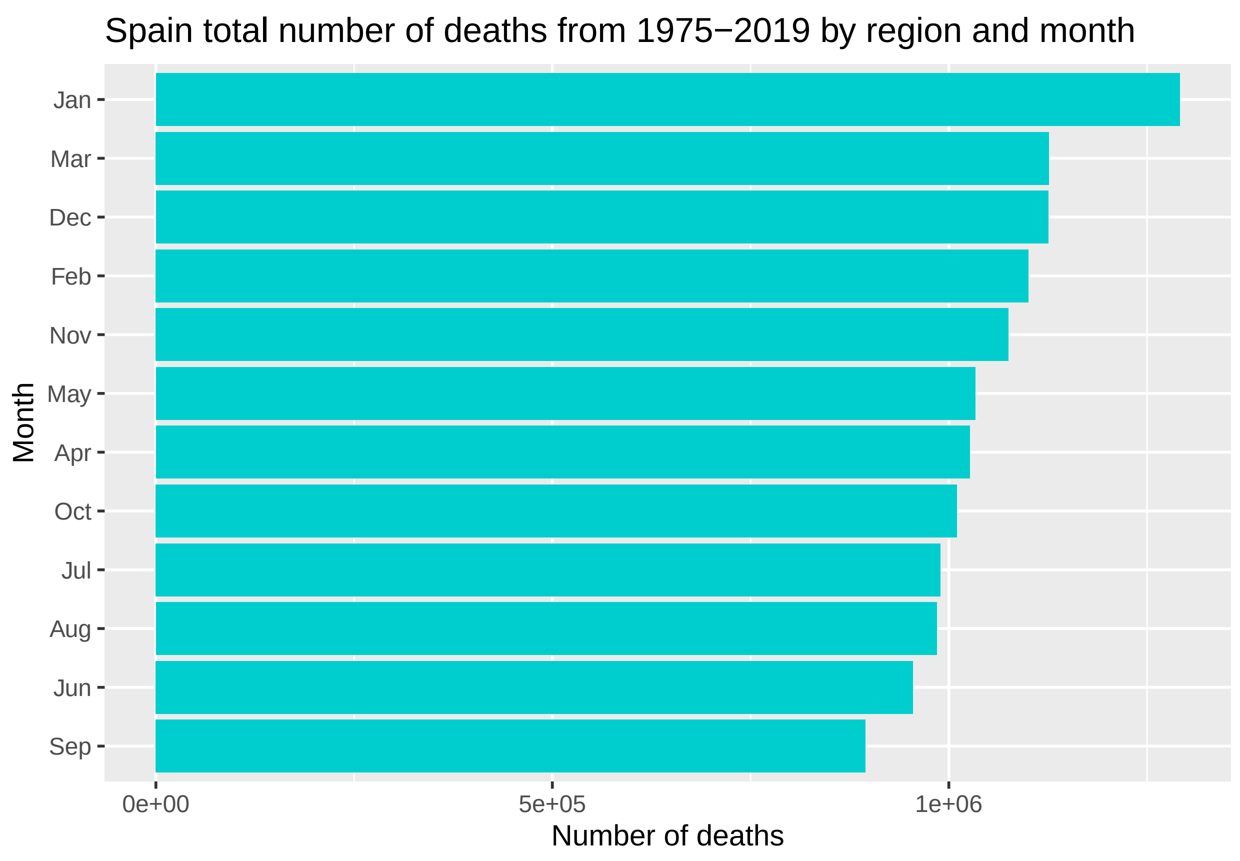
Month analysis

In order to carry out a monthly analysis, we will start by reviewing if there is any seasonal behavior.

```r
#    Seasonal review
train_set_monthly%>%
  group_by(Month) %>%
  summarise(Deaths=sum(Deaths)) %>%
  ggplot(aes(x =factor(Month), y=Deaths)) +
  geom_bar(stat = "identity", fill="cyan3") +
  xlab("Month") +
  ylab("Number of Deaths") +
  ggtitle("Spain number of deaths from 1975-2019 by month")
```



The accumulated number of deaths since 1975 by month goes from January (month 1) which has the highest number of death in the year, followed by March (month 3) and December (month 12). This last point will be interesting, since March 2020 is the first month when the COVID-19 pandemic hit in Spain, and therefore we will see later how much this data was impacted. The lowest number of deaths is in September (month 9). The following graph will show it more clearly.



```
## # A tibble: 12 x 4
##    Month avg_Deaths     n  Deaths
##    <int>      <dbl> <int>   <int>
##  1     1      1803.   716 1291159
##  2    12      1605.   701 1125233
##  3     3      1602.   703 1126405
##  4     2      1516.   726 1100458
##  5    11      1475.   729 1075165
##  6     5      1425.   725 1033111
##  7     4      1422.   722 1026413
##  8    10      1403.   720 1010311
##  9     7      1369.   723  989583
## 10     8      1366.   721  984707
## 11     6      1297.   736  954911
## 12     9      1248.   717  894842
```

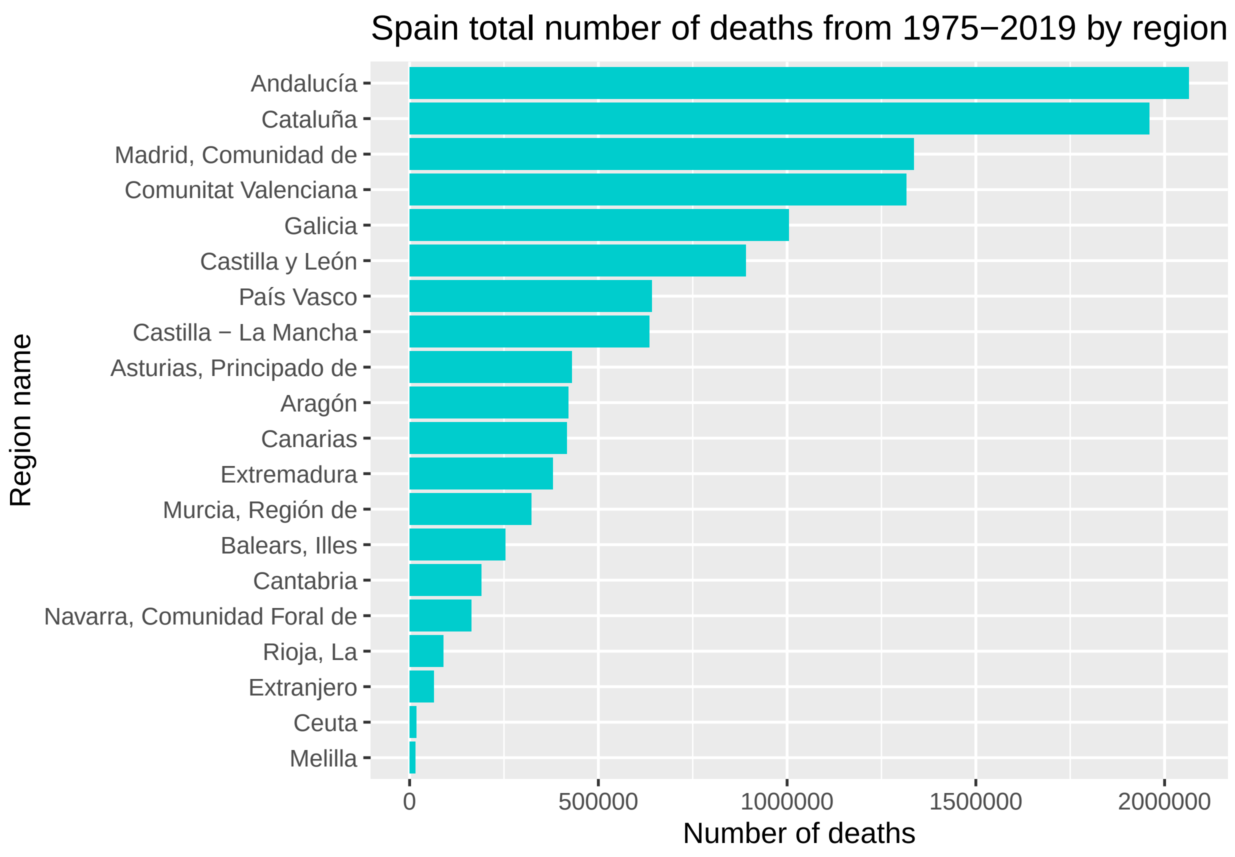
Region analysis

Spain has 17 large Regions, know as *"comunidad autonoma"* that are equivalent to States in most countries. Additionally, there are two 2 States Cities known as *"cuidad autonoma"* (Ceuta and Melilla), that have a similar territorial organization as the large regions, even though they are cities. Finally, inside the data the value " Extranjero" is found for those cases of people who died in Spain but officially resided in another country.

The Region structure should be reflected in the data, since the population of Andalucía is around 100 times bigger than Ceuta or Melilla, and therefore have relevance in the number of deaths. Please note that population data has not been included in the data set.

To start the Region behavior analysis, we will present a ranking of the accumulated deaths from 1975 to 2019 by each Region.

```r
#    Deaths ranking by region
train_set_monthly%>%
  group_by(RegionName) %>%
  summarise(Deaths=sum(Deaths)) %>%
  mutate(RegionRanking=fct_reorder(RegionName, Deaths))  %>%
  ggplot(aes(x =RegionRanking, y=Deaths)) +
  geom_bar(stat = "identity", fill="cyan3") +
  xlab("Region name") +
  ylab("Number of deaths") +
  coord_flip() +
  ggtitle("Spain total number of deaths from 1975-2019 by region")
```



The variability between regions is quite high, for example, big or populated Regions as Andalucia, Cataluna o Madrid are at the top, while much smaller Rehions as Cantabria, Navarra, La Rioja and of course Ceuta and Melilla are at the bottom.

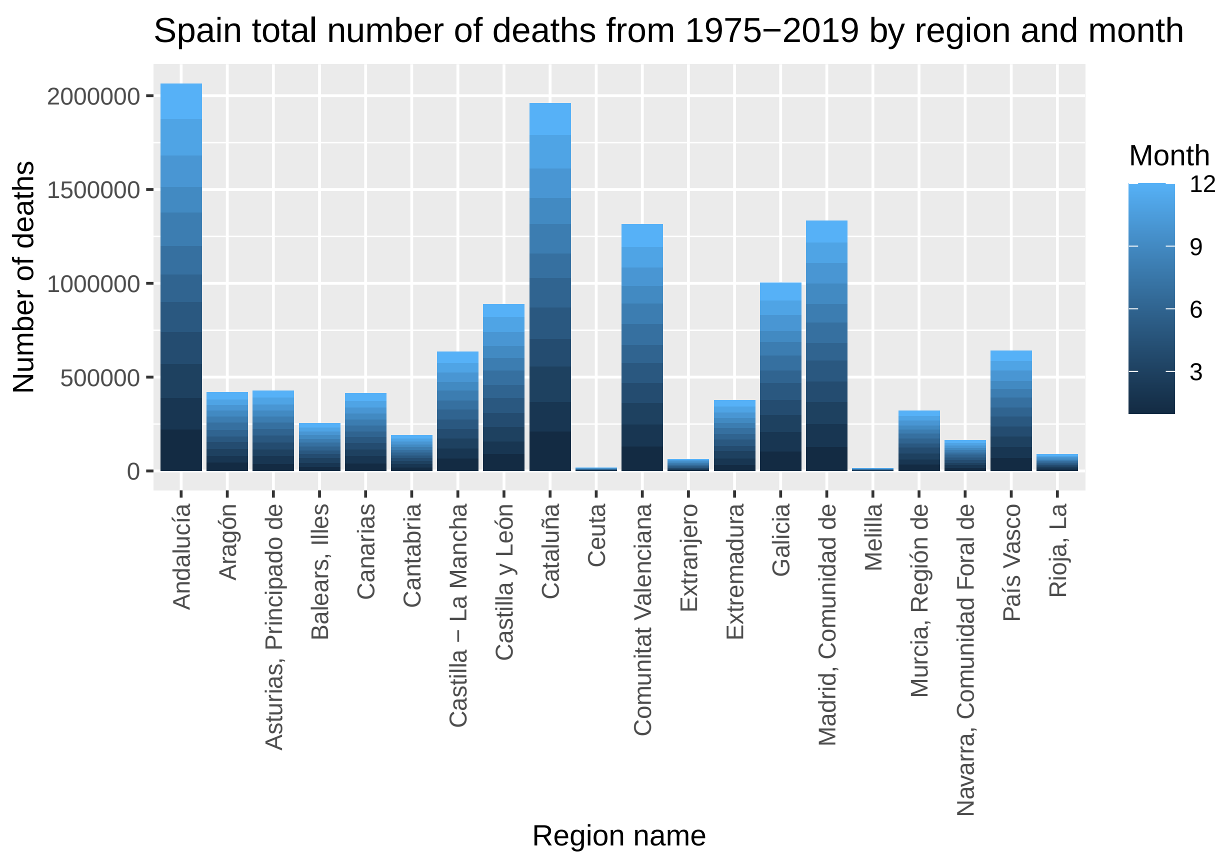
Let's review the average deaths by Region.

```r
#   Ranking by month table
train_set_monthly%>%
  group_by(RegionName) %>%
  summarise(Average=mean(Deaths), SD=sd(Deaths), Deaths=sum(Deaths)) %>%
  arrange(desc(Average))
```

```
## # A tibble: 20 x 4
##    RegionName                   Average    SD  Deaths
##    <chr>                          <dbl> <dbl>   <int>
##  1 Andalucía                      4904.  849.  2064415
##  2 Cataluña                       4505.  820.  1959868
##  3 Madrid, Comunidad de           3049.  583.  1335381
##  4 Comunitat Valenciana           3019.  518.  1316338
##  5 Galicia                        2359.  344.  1005113
##  6 Castilla y León                2086.  291.   890635
##  7 País Vasco                     1466.  257.   642218
##  8 Castilla - La Mancha           1408.  213.   634845
##  9 Aragón                         1028.  156.   420476
## 10 Asturias, Principado de         977.  155.   429872
## 11 Canarias                        957.  224.   416177
## 12 Extremadura                     873.  128.   378978
## 13 Murcia, Región de               758.  143.   322887
## 14 Balears, Illes                  577.   97.4  254082
## 15 Cantabria                       418.   75.8  189893
## 16 Navarra, Comunidad Foral de     393.   62.4  163260
## 17 Rioja, La                       216.   37.0   89797
## 18 Extranjero                      153.   59.0   64631
## 19 Ceuta                            41.4   7.95  18071
## 20 Melilla                          35.0   7.97  15361
```

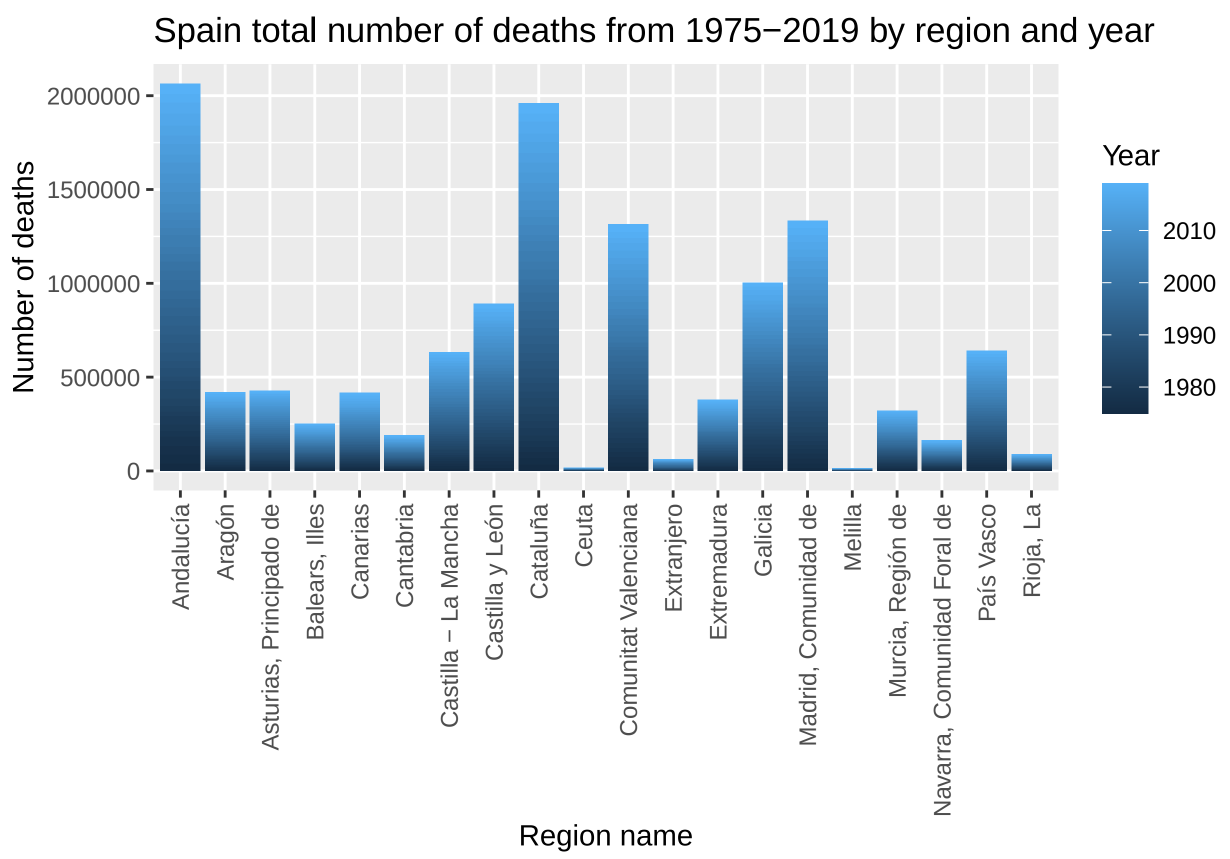
The average deaths by Region shows the number by groups by year-month. Let's know review how the seasonal effect identified with Months is equally distributed by Regions.

```
#    Region by months
train_set_monthly%>%
  group_by(RegionName, Month) %>%
  summarise(Deaths=sum(Deaths)) %>%
  ggplot(aes(x =RegionName, y=Deaths, fill=Month)) +
  geom_bar(stat = "identity") +
  xlab("Region name") +
  ylab("Number of deaths") +
  ggtitle("Spain total number of deaths from 1975-2019 by region and month")+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```



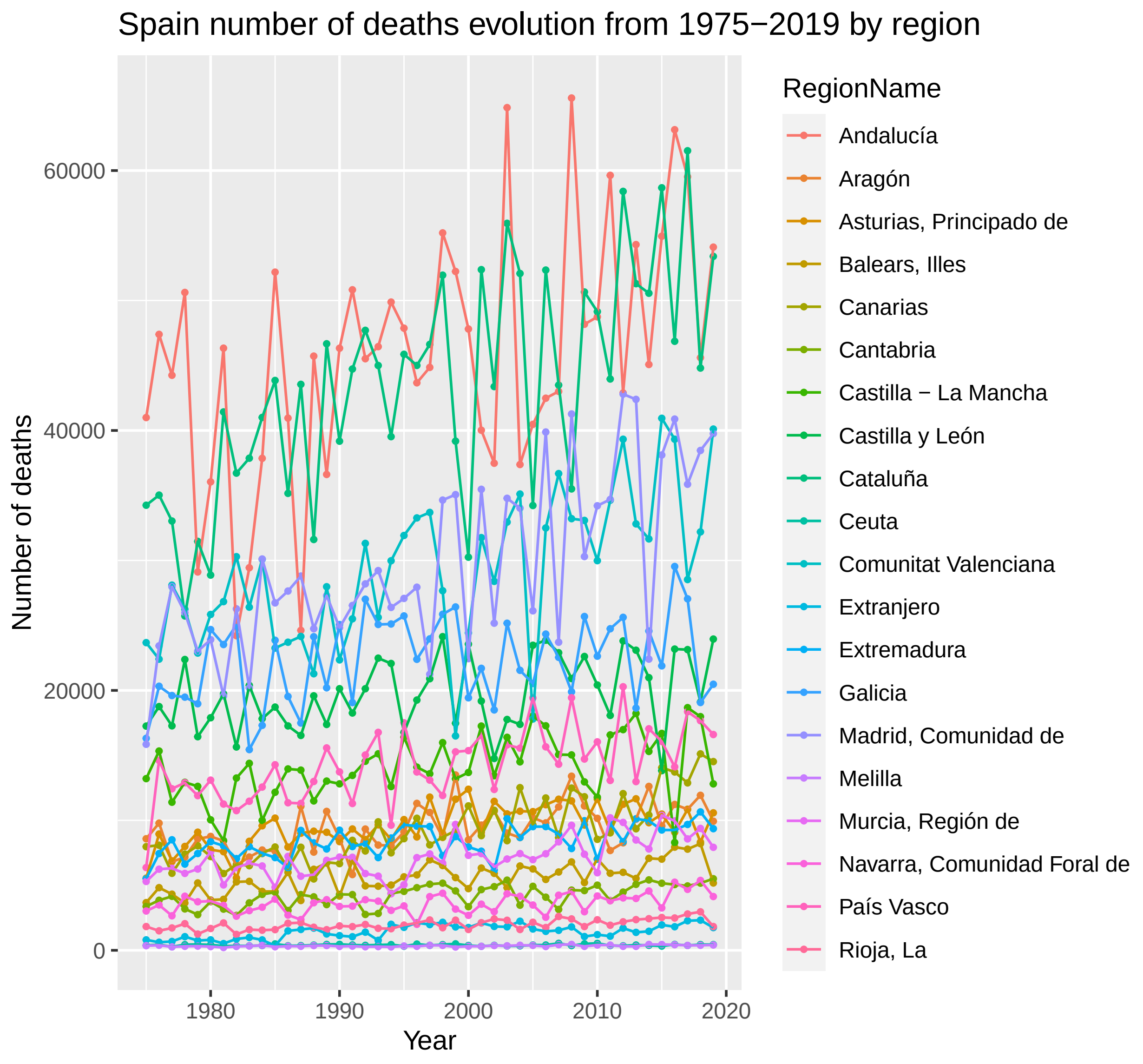
The effect of having higher numbers in January (darkest color) and December (lighter color), seems to be equally distributed across all the regions, and transitions are quite smooth, explained by having a low variability between month. Now let's review how the increase effect in Year is distributed by Region.

```
#    Region by year
train_set_monthly%>%
  group_by(RegionName, Year) %>%
  summarise(Deaths=sum(Deaths)) %>%
  ggplot(aes(x =RegionName, y=Deaths, fill=Year)) +
  geom_bar(stat = "identity") +
  xlab("Region name") +
  ylab("Number of deaths") +
  ggtitle("Spain total number of deaths from 1975-2019 by region and year")+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```



The proportion of lighters colors, corresponding to the latest years, is bigger than those related to dark colors. But again, transitions are smooth, without jumps or drastic changes. Let's deep dive in the Region by Year relation, by showing an evolution graph.

```
#   Evolution by region
train_set_monthly%>%
  group_by(Year, RegionName) %>%
  summarise(Deaths=sum(Deaths)) %>%
  ggplot(aes(x =Year, y=Deaths, color=RegionName)) +
  geom_point(size=0.75)  +
  geom_line() +
  ylab("Number of deaths") +
  ggtitle("Spain number of deaths evolution from 1975-2019 by region")
```



We can clearly see how the Region with higher number of deaths, also shows higher variability between years, for example *Andalucía*, *Cataluña*, *Madrid* or *Comunitat Valenciana*, and getting more stable in lower raking Regions. The incremental tendency is also shown more clearly in Regions with higher number of deaths, and it is almost lost in the lowest Regions.

Data from 1975-2020/1 (post COVID-19)

As explained earlier, the data covers only the first semester of 2020, which shows the impact of the first wave in Spain's mortality due to the pandemic.

The analysis will be done in the following order:

1. General distribution
2. Semester analysis
3. Month analysis

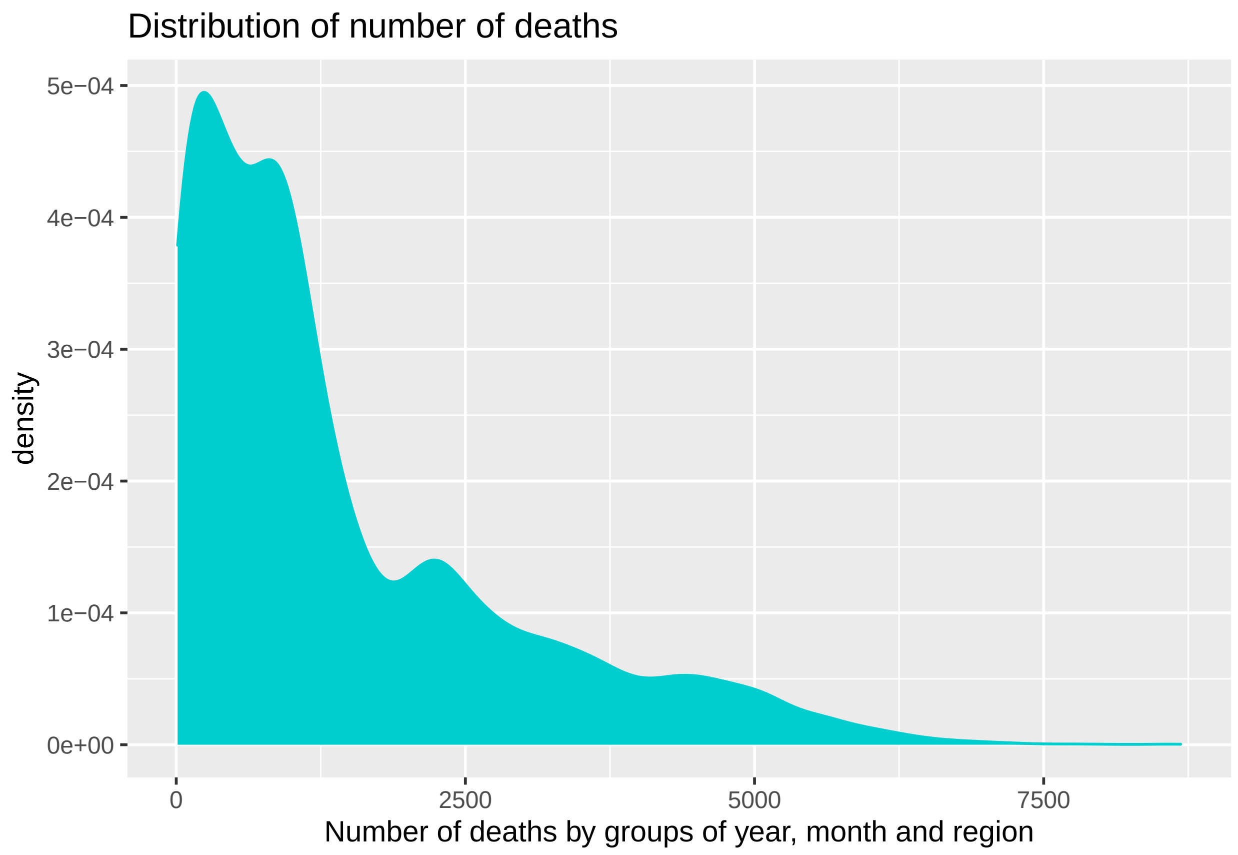
General distribution

The first step is the distribution.

```
#  Let's start by reviewing distribution
summary(train_set_covid)
```

```
##   RegionName             Year          Month           Deaths
##  Length:8831        Min.   :1975   Min.   : 1.00   Min.   :  12
##  Class :character   1st Qu.:1986   1st Qu.: 3.00   1st Qu.: 396
##  Mode  :character   Median :1997   Median : 6.00   Median : 948
##                     Mean   :1998   Mean   : 6.42   Mean   :1451
##                     3rd Qu.:2009   3rd Qu.: 9.00   3rd Qu.:2158
##                     Max.   :2020   Max.   :12.00   Max.   :8685
##     Semester
##  Min.   :1.00
##  1st Qu.:1.00
##  Median :1.00
##  Mean   :1.49
##  3rd Qu.:2.00
##  Max.   :2.00
```

```
train_set_covid %>%
  ggplot(aes(x=Deaths)) +
  geom_density(color="cyan3", fill="cyan3") +
  xlab("Number of deaths by groups of year, month and region")+
  ggtitle("Distribution of number of deaths")
```



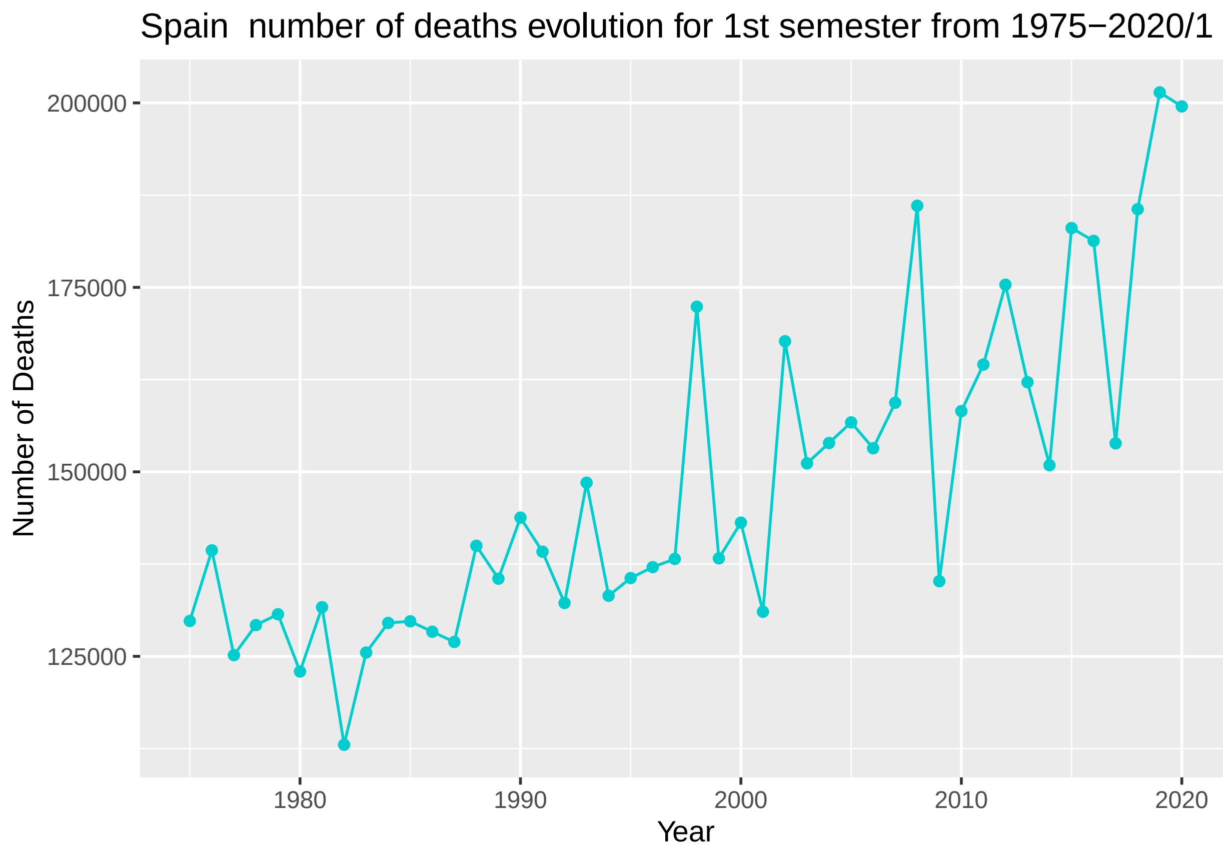
The the peak is a little higher than the peak in the distribution until 2019. We can also verify the data goes from 1975 to 2020, where we have 12 months as expected. A interesting point is the *mean* of post COVID-19 data is actually a little bit lower (1451) than the pre COVID-19 *mean* (1460). The min and max number of death by groups of year-month and region is the same (12 and 8685 respectively).

Semester analysis

The year analysis cannot be done since data from 2020 is not complete, and if we try to graph a year evolution we will see a drop in 2020 that doesn't represent the yearly behavior. Therefore, we can review the evolution of the the first semester with a seasonal approach. Then, we can also evaluate the combination of Year-Semester and start by making a ranking of the top 15.

Let's start with the evolution view.

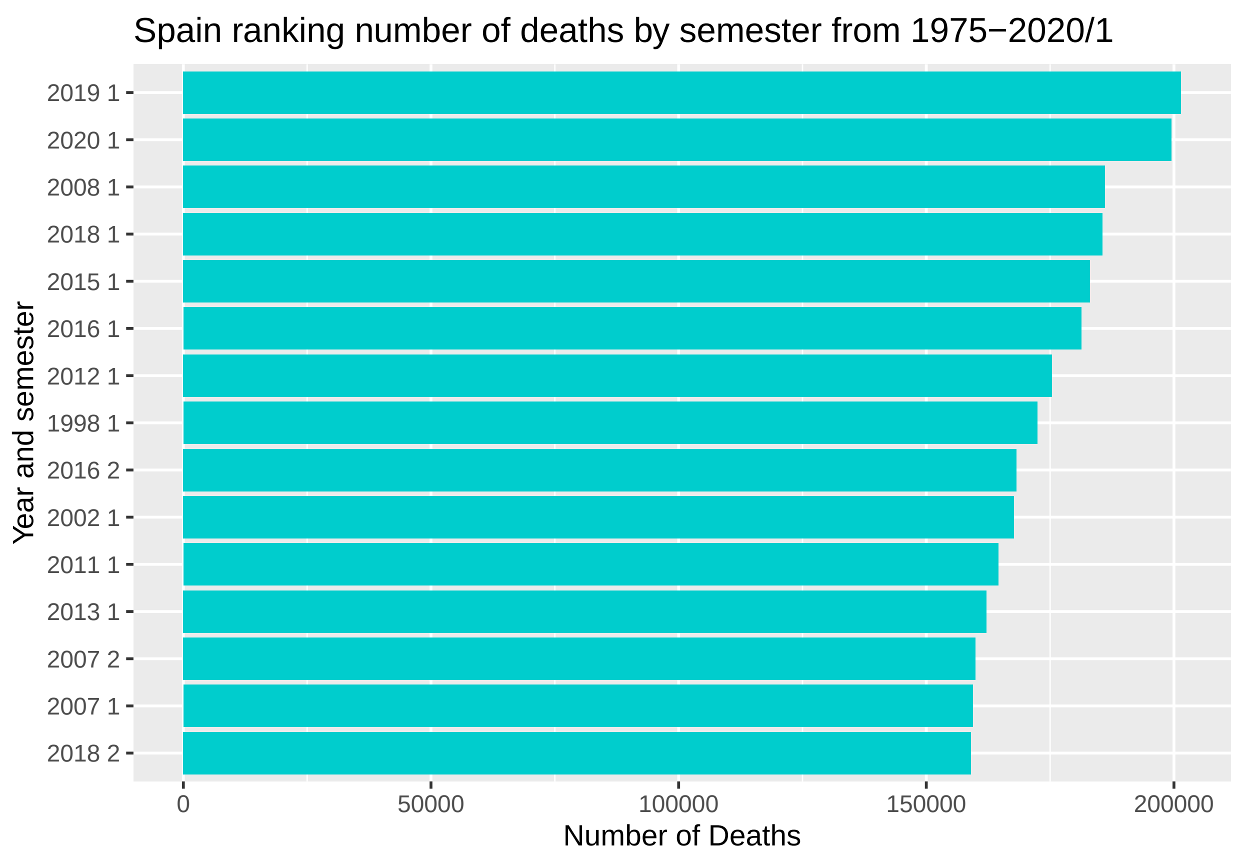
```r
#   Evolution Semester analysis
train_set_covid %>%
  filter(Semester==1) %>%
  group_by(Year) %>%
  summarise(Deaths=sum(Deaths)) %>%
  ggplot(aes(x =Year, y=Deaths)) +
  geom_point(color="cyan3",size=1.5)  +
  geom_line(color="cyan3") +
  ylab("Number of Deaths") +
  ggtitle("Spain  number of deaths evolution for 1st semester from 1975-2020/1")
```



The same incremental *tendency* we see in the Year analysis, is presented in the First Semester by Year evolution. However, we can notice a great deal of variability in short periods like: 1997 to 1999, and 2007 to 2009. This is shown in the graph with jumps, being the most important one between 2008 and 2009. On the other hand, the variability between 2019 and 2020 is quite small.

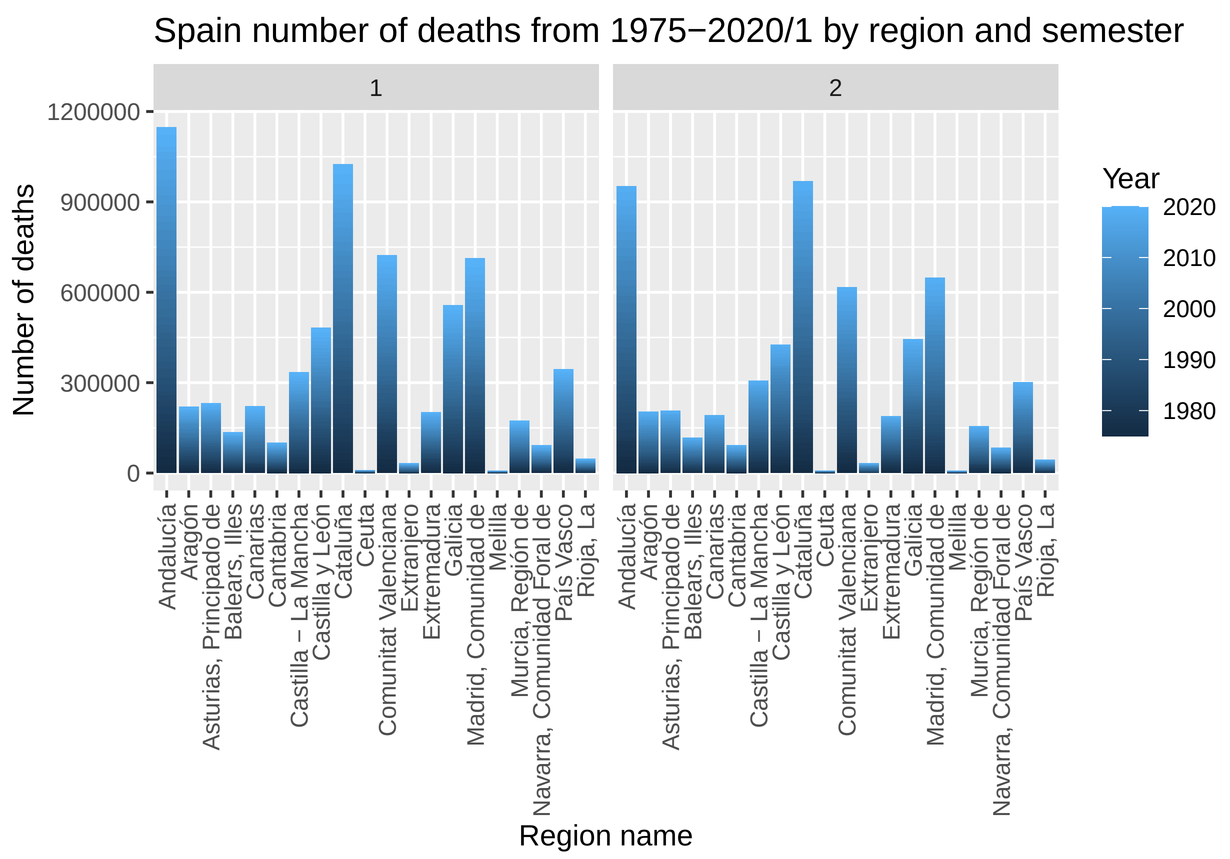
Now let's review the ranking results.

```
#   Ranking Semester analysis
train_set_covid%>%
  mutate(YearSem = paste(Year,Semester)) %>%
  group_by(YearSem) %>%
  summarise(Deaths=sum(Deaths)) %>%
  mutate(YearSemRanking=fct_reorder(YearSem, Deaths))  %>%
  arrange(desc(Deaths)) %>%
  slice(1:15) %>%
  ggplot(aes(x =YearSemRanking, y=Deaths)) +
  geom_bar(stat = "identity", fill="cyan3") +
  coord_flip() +
  xlab("Year and semester") +
  ylab("Number of Deaths") +
  ggtitle("Spain ranking number of deaths by semester from 1975-2020/1")
```



The data in the training set shows that the first semester of 2019 is higher than that of 2020, which is kind of surprising. We will deep dive into this point in the Month analysis, but before getting there, let's review the Region distribution by Semester.

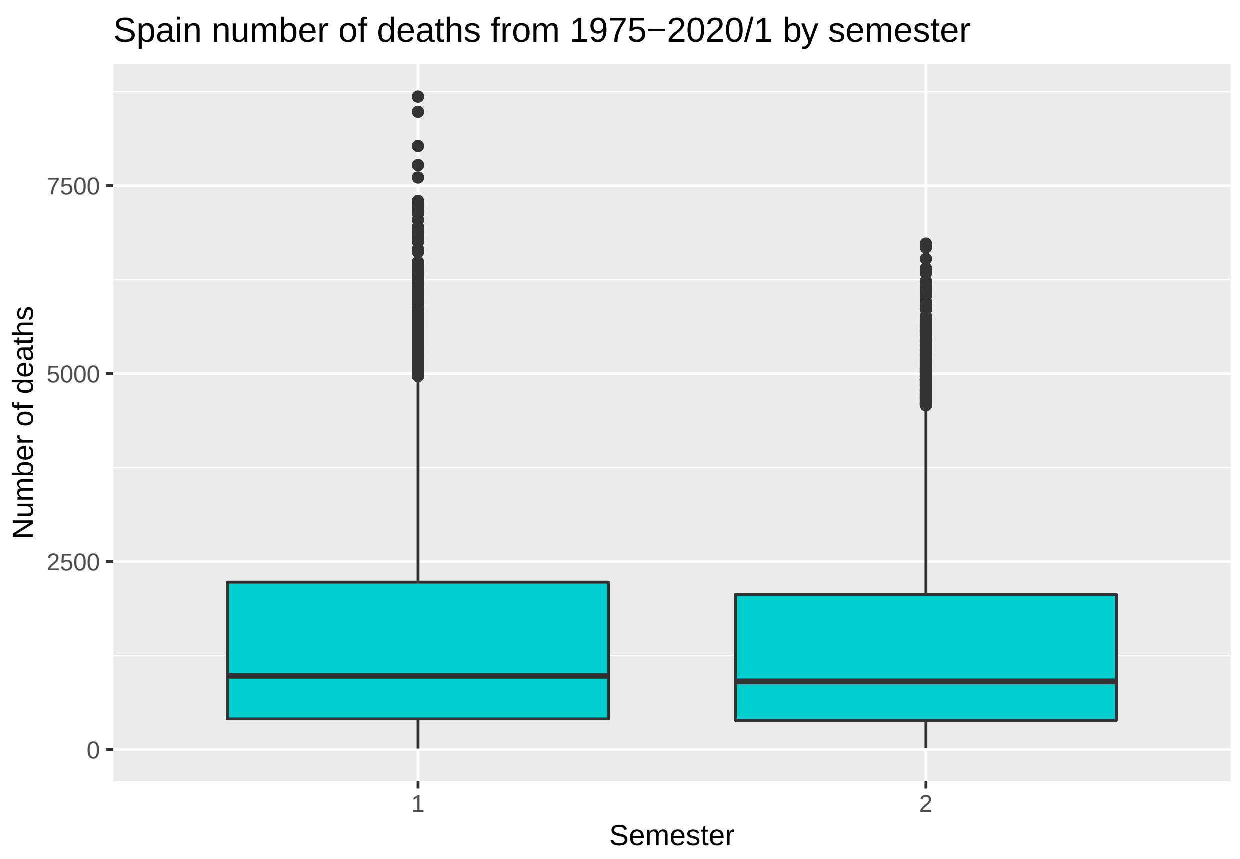
```
#    Region by semester
train_set_covid%>%
  group_by(Semester, RegionName, Year) %>%
  summarise(Deaths=sum(Deaths)) %>%
  ggplot(aes(x =RegionName, y=Deaths, fill=Year)) +
  geom_bar(stat = "identity") +
  xlab("Region name") +
  ylab("Number of deaths") +
  ggtitle("Spain number of deaths from 1975-2020/1 by region and semester")+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) +
  facet_wrap(.~Semester)
```



In general, we see higher numbers in semester 1, which is expected, taking into account the additional semester in 2020, but there are no significant changes in the distribution by Region. Therefore at semester level, 2020 does not include a change in the general behavior by Region.

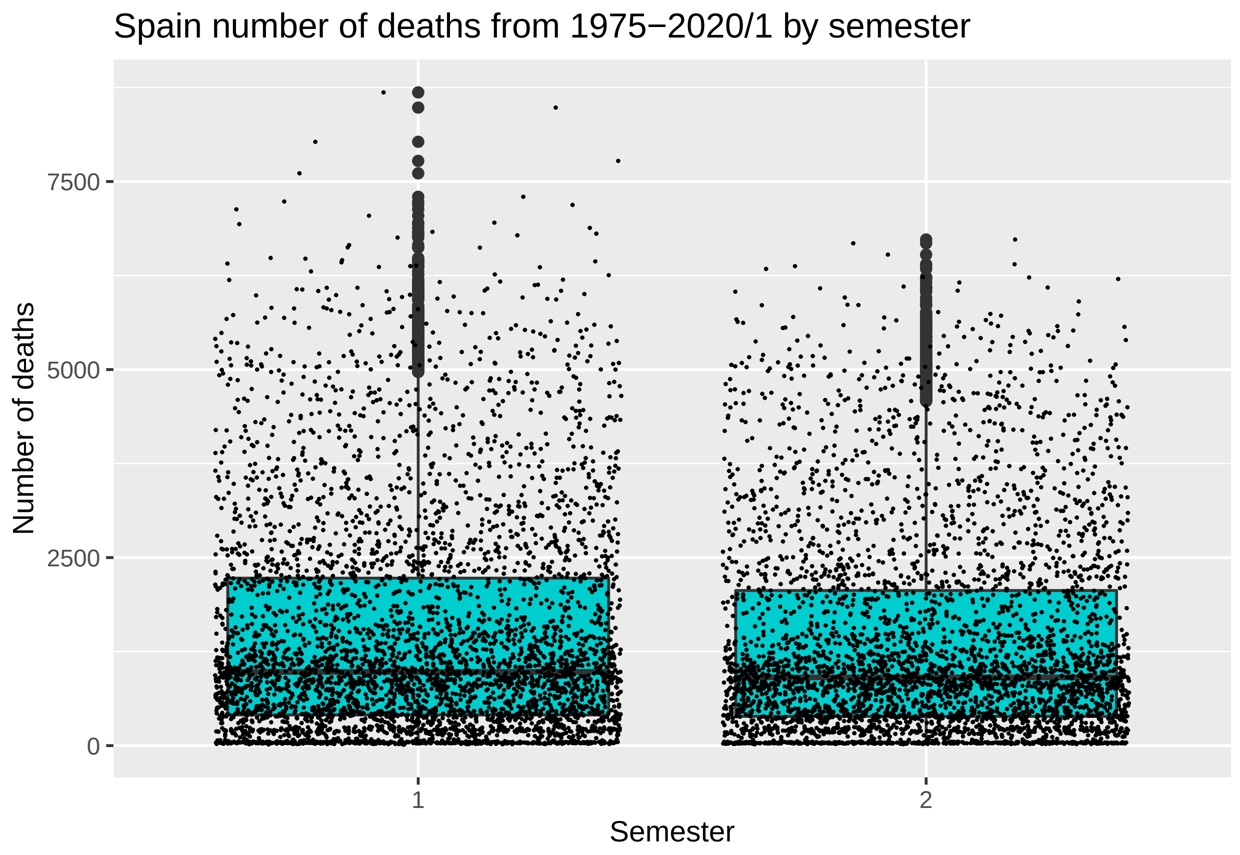
For the next step, let's review the data dispersion by semester.

```
#   Semester analysis - data dispersion
train_set_covid %>%
  ggplot(aes(x= factor(Semester), y= Deaths)) +
  geom_boxplot(fill="cyan3") +
  xlab("Semester") +
  ylab("Number of deaths") +
  ggtitle("Spain number of deaths from 1975-2020/1 by semester")
```



The mean and 3rd quartile of Semester 1 are a little bit higher than Semester 2. However the most remarkable point is the higher level of dispersion in the first semester. Let's add a jitter layer to review how many *outliners* are producing this effect.

```
#   Semester analysis with jitter
train_set_covid %>%
  ggplot(aes(x= factor(Semester), y= Deaths)) +
  geom_boxplot(fill="cyan3") +
  geom_jitter(size=0.1) +
  xlab("Semester") +
  ylab("Number of deaths") +
  ggtitle("Spain number of deaths from 1975-2020/1 by semester")
```



We can see that even in the share intervals, semester 2 data is more compact showing more "green spaces" above the mean than semester 1. Additionally, the number of groups above 3rd quatile is high, but only a few in semester 1 go beyond the top values of semester 2.

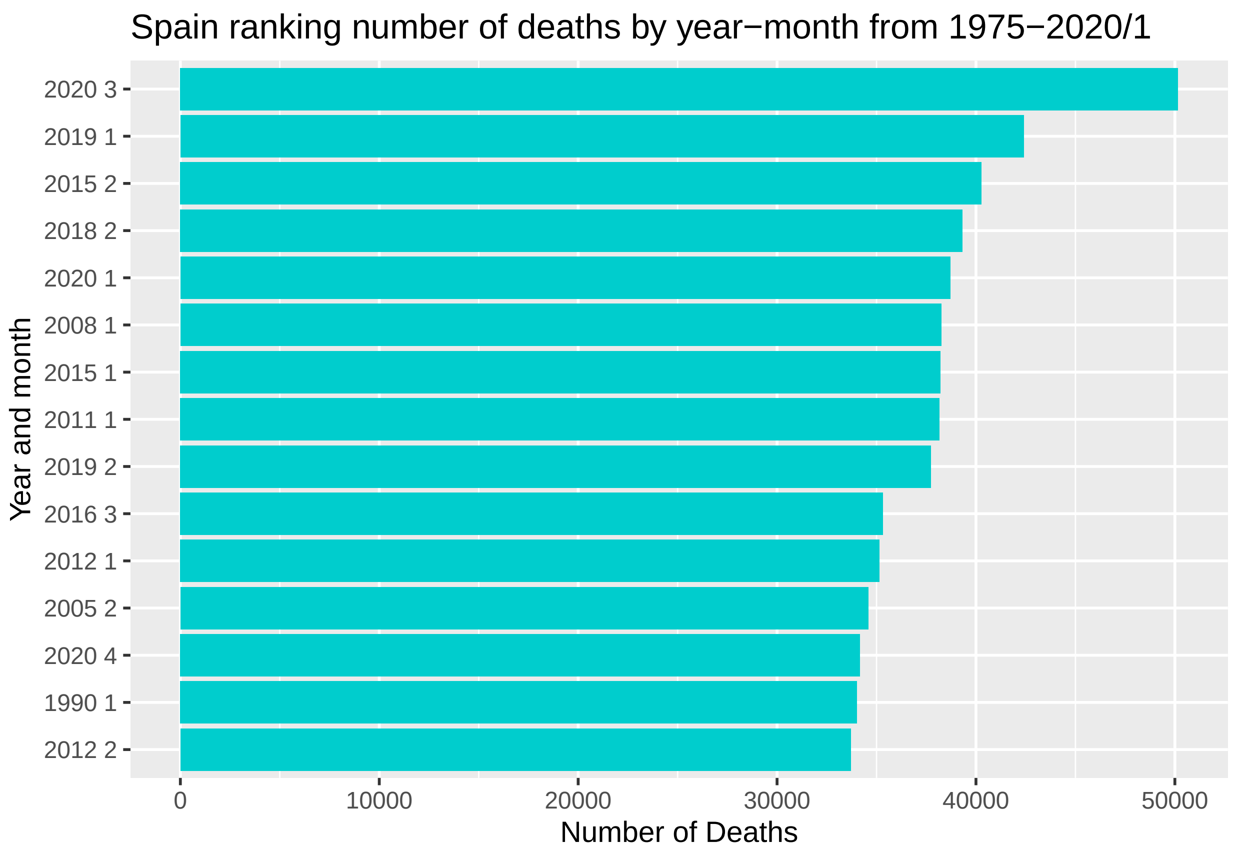
Month analysis

Let's deep dive into the semester behaviors. We know, from news and other data that the first COVID-19 wave hit Spain in March 2020, where the peak of daily deaths was reached in April. In May, after 2 months of confinement, the wave was almost finished, getting to its end by June (https://www.worldometers.info/coronavirus/country/spain/).

We also know that *January* is seasonally, the month with the highest number of deaths, follow by *March* and *December*.

Let's start with a quick ranking using the 1975-2020/1 data set.

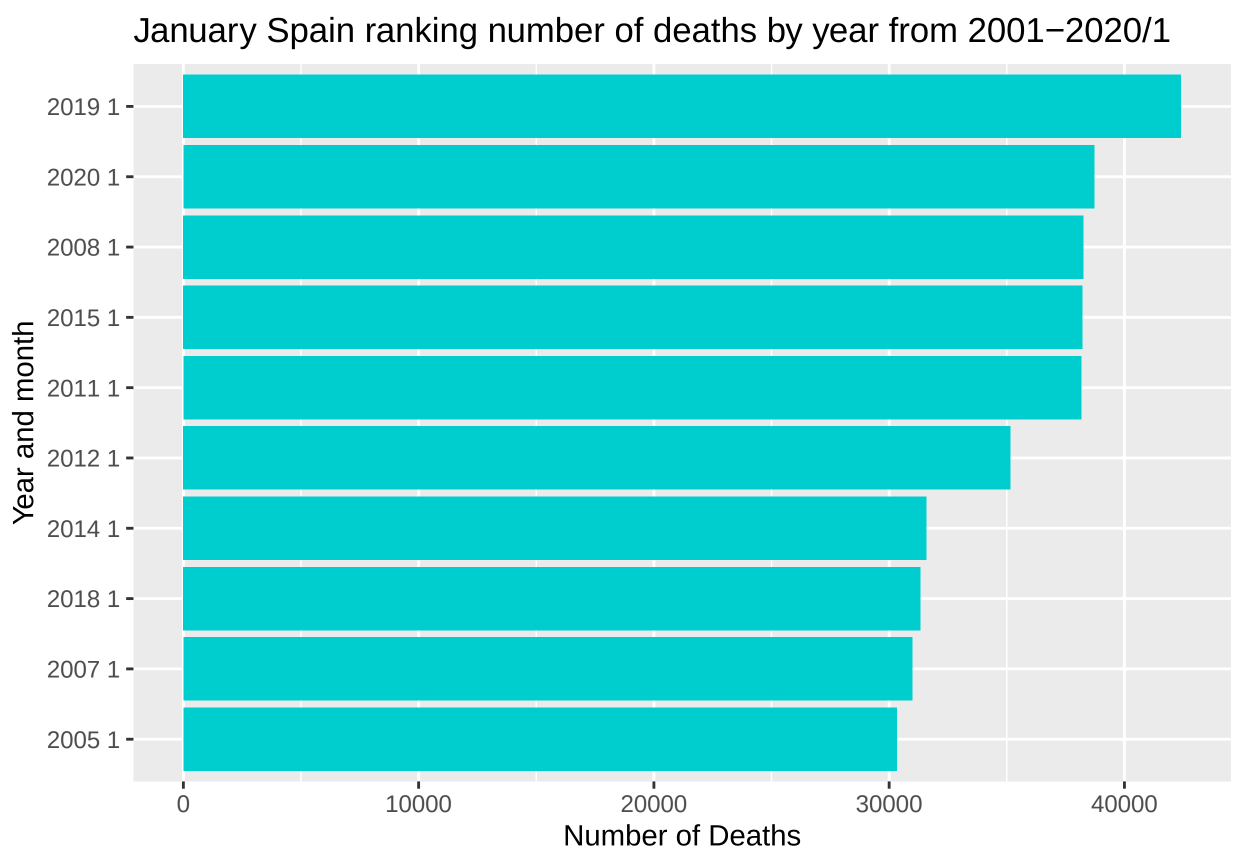
```
#   Ranking by total death by month graph post covid
train_set_covid%>%
  filter(Semester==1) %>%
  mutate(YearMonth = paste(Year,Month)) %>%
  group_by(YearMonth) %>%
  summarise(Deaths=sum(Deaths)) %>%
  mutate(YearMonth=fct_reorder(YearMonth, Deaths))  %>%
  arrange(desc(Deaths)) %>%
  slice(1:15) %>%
  ggplot(aes(x =YearMonth, y=Deaths)) +
  geom_bar(stat = "identity", fill="cyan3") +
  coord_flip() +
  xlab("Year and month") +
  ylab("Number of Deaths") +
  ggtitle("Spain ranking number of deaths by year-month from 1975-2020/1")
```



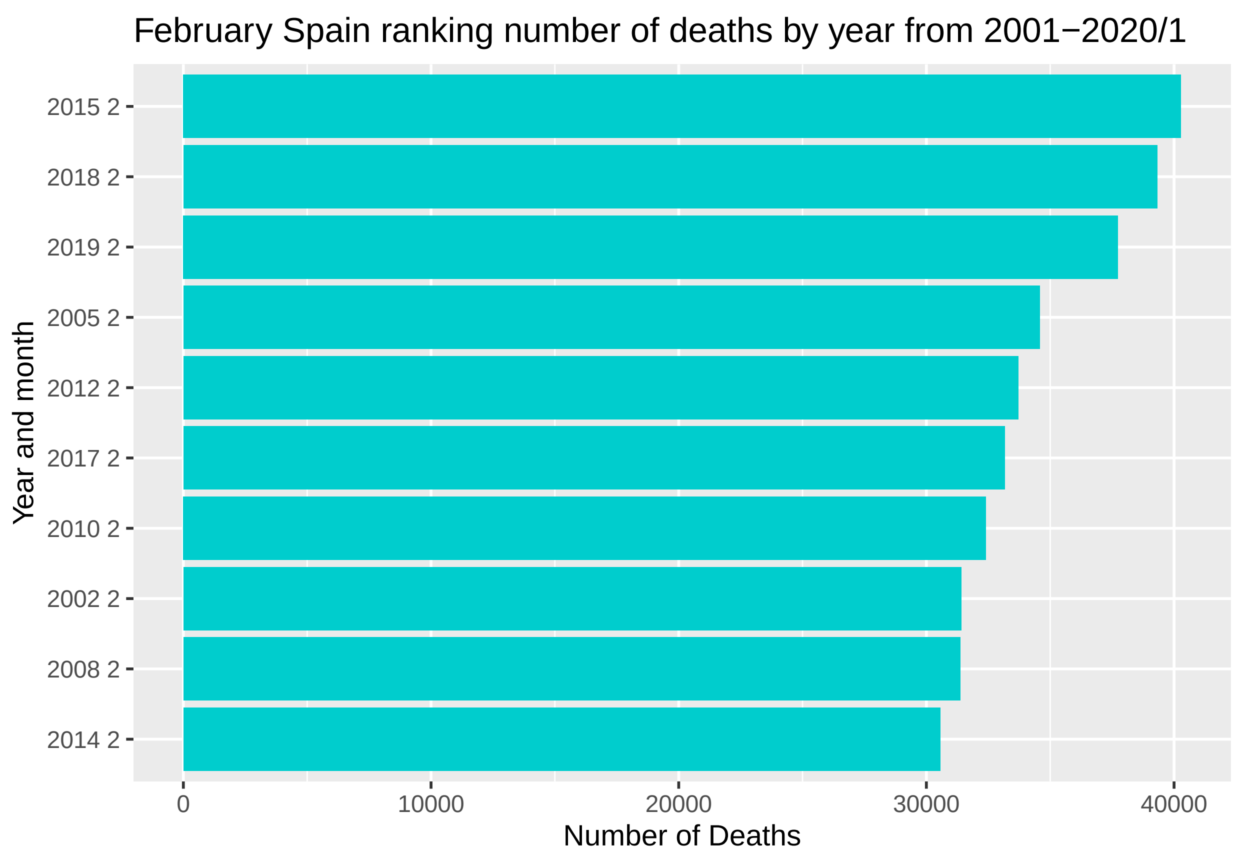
March 2020 is at the top of the ranking, with a clear difference in terms of number of death from other months. This behavior is kind of expected knowing the dates of the first wave. However April 2020 is only at the 13th position, and no other months in 2020 appears on the top 15.

Lest do the analysis month by month to consider know the influence of the seasonality.

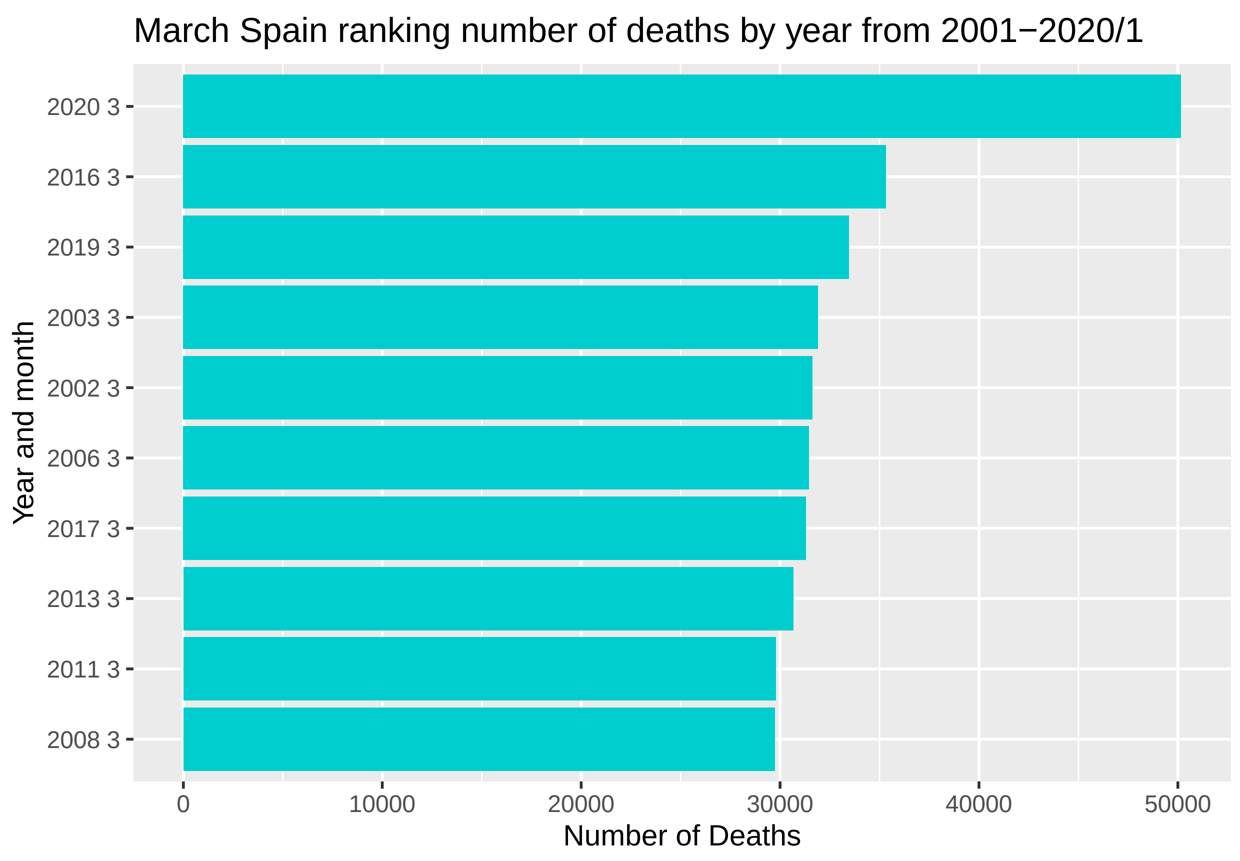
```
#    January
train_set_covid%>%
  filter(Semester==1 & Year > 2000 & Month==1) %>%
  mutate(YearMonth = paste(Year,Month)) %>%
  group_by(YearMonth) %>%
  summarise(Deaths=sum(Deaths)) %>%
  mutate(YearMonth=fct_reorder(YearMonth, Deaths))  %>%
  arrange(desc(Deaths)) %>%
  slice(1:10) %>%
  ggplot(aes(x =YearMonth, y=Deaths)) +
  geom_bar(stat = "identity", fill="cyan3") +
  coord_flip() +
  xlab("Year and month") +
  ylab("Number of Deaths") +
  ggtitle("January Spain ranking number of deaths by year from 2001-2020/1")
```



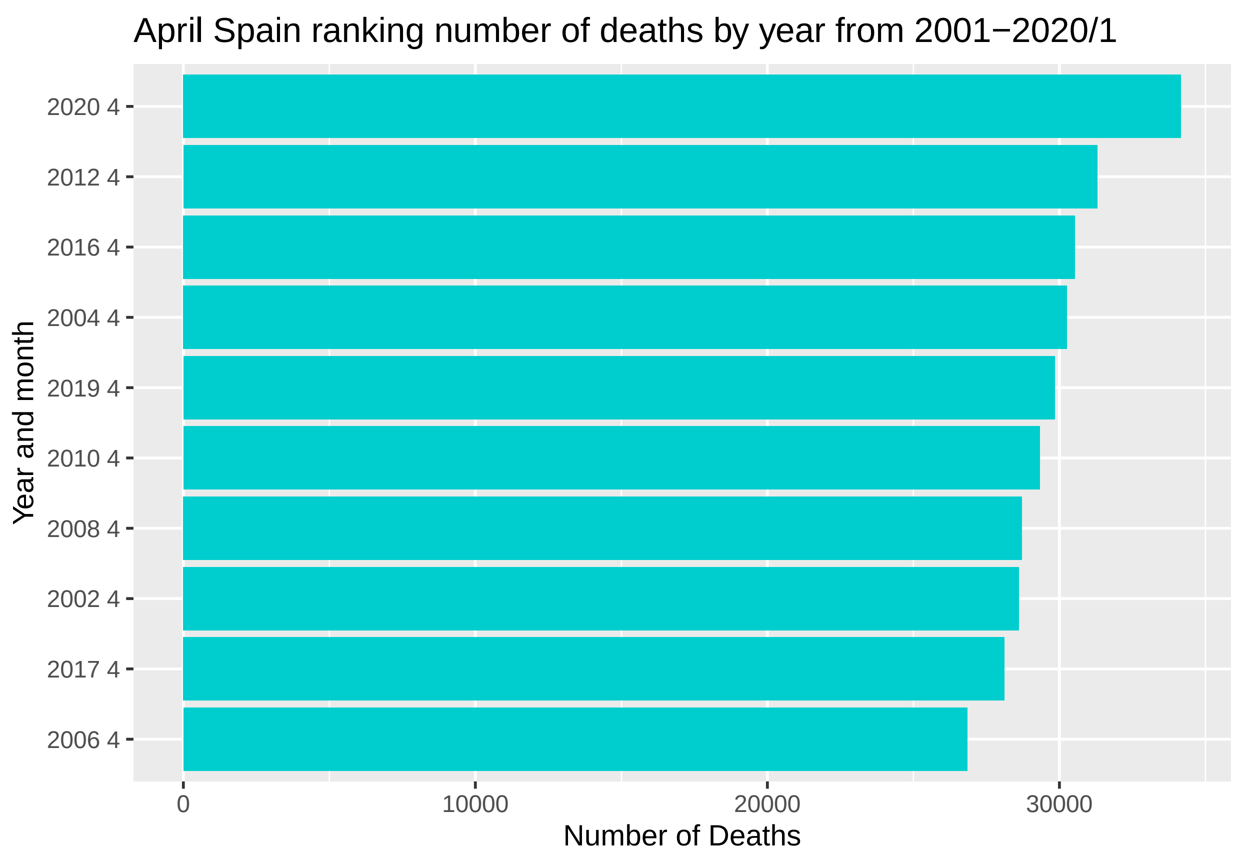
```
#    February
train_set_covid%>%
  filter(Semester==1 & Year > 2000 & Month==2) %>%
  mutate(YearMonth = paste(Year,Month)) %>%
  group_by(YearMonth) %>%
  summarise(Deaths=sum(Deaths)) %>%
  mutate(YearMonth=fct_reorder(YearMonth, Deaths))  %>%
  arrange(desc(Deaths)) %>%
  slice(1:10) %>%
  ggplot(aes(x =YearMonth, y=Deaths)) +
  geom_bar(stat = "identity", fill="cyan3") +
  coord_flip() +
  xlab("Year and month") +
  ylab("Number of Deaths") +
  ggtitle("February Spain ranking number of deaths by year from 2001-2020/1")
```



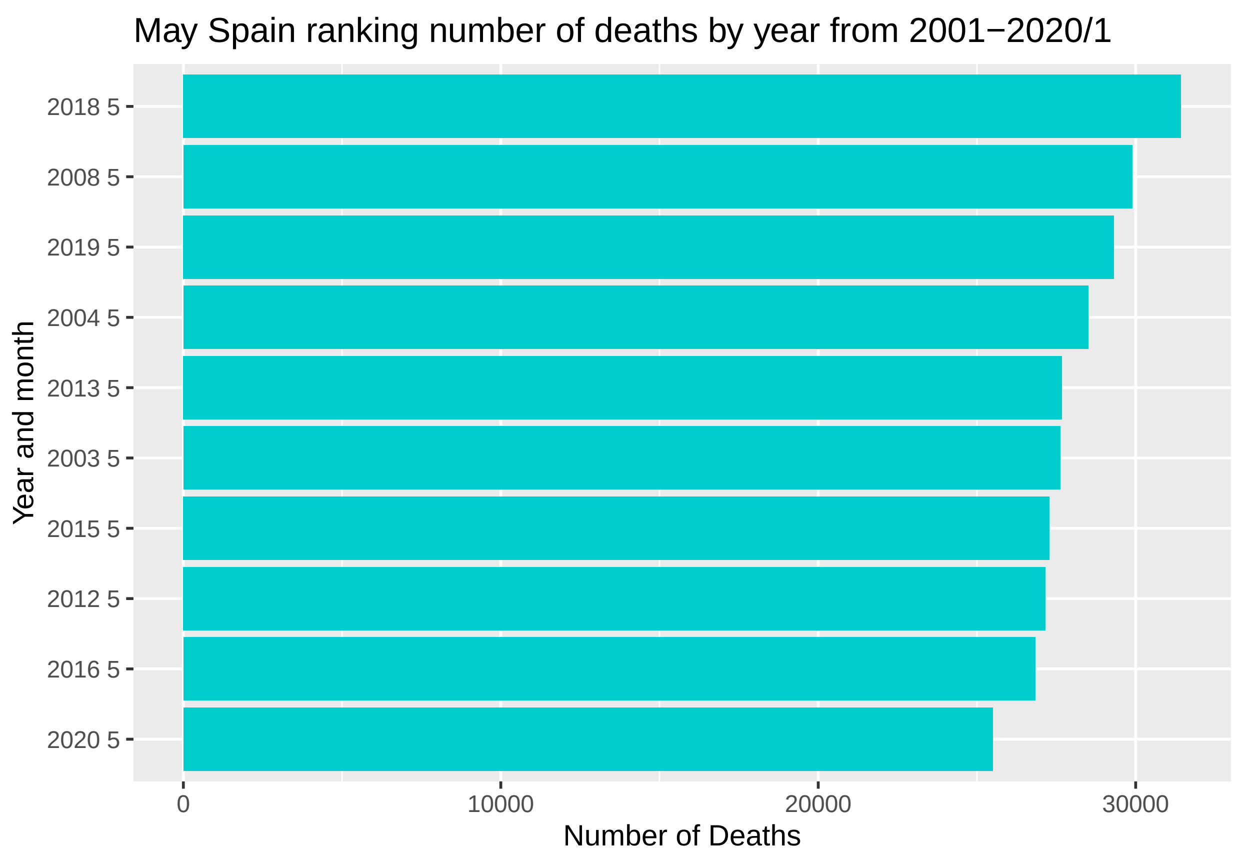
```
#    March - first wave starts
train_set_covid%>%
  filter(Semester==1 & Year > 2000 & Month==3) %>%
  mutate(YearMonth = paste(Year,Month)) %>%
  group_by(YearMonth) %>%
  summarise(Deaths=sum(Deaths)) %>%
  mutate(YearMonth=fct_reorder(YearMonth, Deaths))  %>%
  arrange(desc(Deaths)) %>%
  slice(1:10) %>%
  ggplot(aes(x =YearMonth, y=Deaths)) +
  geom_bar(stat = "identity", fill="cyan3") +
  coord_flip() +
  xlab("Year and month") +
  ylab("Number of Deaths") +
  ggtitle("March Spain ranking number of deaths by year from 2001-2020/1")
```



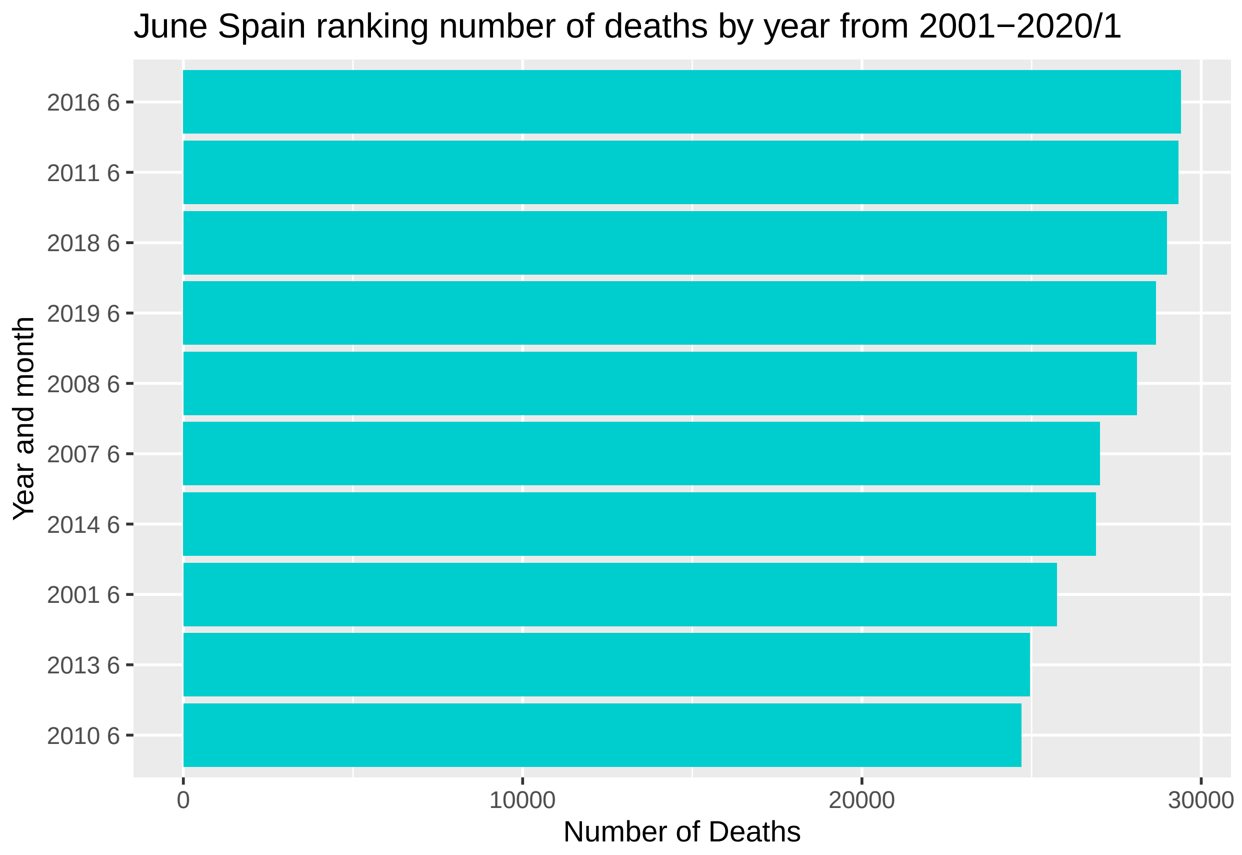
```r
#    April - first wave peak on daily deaths
train_set_covid%>%
  filter(Semester==1 & Year > 2000 & Month==4) %>%
  mutate(YearMonth = paste(Year,Month)) %>%
  group_by(YearMonth) %>%
  summarise(Deaths=sum(Deaths)) %>%
  mutate(YearMonth=fct_reorder(YearMonth, Deaths))  %>%
  arrange(desc(Deaths)) %>%
  slice(1:10) %>%
  ggplot(aes(x =YearMonth, y=Deaths)) +
  geom_bar(stat = "identity", fill="cyan3") +
  coord_flip() +
  xlab("Year and month") +
  ylab("Number of Deaths") +
  ggtitle("April Spain ranking number of deaths by year from 2001-2020/1")
```



```
#    May - first wave ends
train_set_covid%>%
  filter(Semester==1 & Year > 2000 & Month==5) %>%
  mutate(YearMonth = paste(Year,Month)) %>%
  group_by(YearMonth) %>%
  summarise(Deaths=sum(Deaths)) %>%
  mutate(YearMonth=fct_reorder(YearMonth, Deaths))  %>%
  arrange(desc(Deaths)) %>%
  slice(1:10) %>%
  ggplot(aes(x =YearMonth, y=Deaths)) +
  geom_bar(stat = "identity", fill="cyan3") +
  coord_flip() +
  xlab("Year and month") +
  ylab("Number of Deaths") +
  ggtitle("May Spain ranking number of deaths by year from 2001-2020/1")
```



```
#    June
train_set_covid%>%
  filter(Semester==1 & Year > 2000 & Month==6) %>%
  mutate(YearMonth = paste(Year,Month)) %>%
  group_by(YearMonth) %>%
  summarise(Deaths=sum(Deaths)) %>%
  mutate(YearMonth=fct_reorder(YearMonth, Deaths))  %>%
  arrange(desc(Deaths)) %>%
  slice(1:10) %>%
  ggplot(aes(x =YearMonth, y=Deaths)) +
  geom_bar(stat = "identity", fill="cyan3") +
  coord_flip() +
  xlab("Year and month") +
  ylab("Number of Deaths") +
  ggtitle("June Spain ranking number of deaths by year from 2001-2020/1")
```



In the month by month analysis using the training data, we found:

- *January* 2019 is at the top of the ranking, followed by 2020, even before the first official COVID-19 death was declared in February. However, knowing that data has a *tendency* to grow by year, this sounds about right.
- *February*, 2020 is not even in the top 10.
- *March* 2020 highlight in comparison to previous years, being the closest one being 2016 with almost a 15000 difference in the number of deaths.
- *April* 2020 is also at the top. Even though, the difference between 2020 and the closest year, 2012, is smaller than in *March*, it is still bigger that other year to year differences.
- *May* 2020 is in the 10th position.
- *June* does not even appear in the top 10 graph.

Therefore, the relevant COVID-19 impact is shown only in the month of *March* and *April*, and even though the difference is quite big in the month by month review, the effect is lost at the semester level.

Building and comparing models

This phase has two objectives, the first one is to find a model for the 1975-2019 (pre COVID-19) data that improves **Accuracy** and **RMSE** of the Linear regression, that will be used as a baseline. This objective will cover the course requirement.

Once this is achieved, a new training will be executed, using the 1975-2020/1 (post COVID-19) data set, but keeping the same parameters used in the selected model for the 1975-2019 data set. Then, we will compare

Accuracy and **RMSE** of both models to determine if the COVID-19 influence on mortality has a relevant impact in the model designed before COVID-19, or not.

Following are the functions definitions that will calculated the metrics **Accuracy** and **RMSE**.

```
# Function to compare methods

rmse_f <- function(y, y_hat){
  sqrt(mean((y - y_hat)^2))
}
accuracy_f<-function(y, y_hat){
  mean(y==y_hat)
}
```

Model for the 1975-2019 (pre COVID-19)

The models to be built are:

- Linear regression (lm)
- Loess
- KNN (N neighbors)
- Random Forest

Linear regression (lm)

In order to establish the baseline, we need to find which predictor, or which combination of predictors, produce the best results for the *lm* model. The predictors available in our data are:

- Region
- Year and
- Month

We will choose the model with the lowest RMSE value produce by in the train function as a baseline. Let's start with the *all predictors* combination, that includes: Region, Year and Month.

```
#   Lm: Including all predictors
#    Seed setting for sampling replication
set.seed(1, sample.kind = "Rounding")

#    Training algorithm
train_lm_monthly <- train(Deaths ~ RegionName + Year + Month,
                          method = "lm",
                          data = train_set_monthly)
#    Review of training results
train_lm_monthly
```

```
## Linear Regression
##
## 8639 samples
##    3 predictor
##
```

```
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 8639, 8639, 8639, 8639, 8639, 8639, ...
## Resampling results:
##
##   RMSE   Rsquared  MAE
##   302.4  0.9559    190.2
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
#    Keep RMSE results
result_lm <- data.frame(Type = "Monthly 1975-2019",
                        Method = "lm: all predictors",
                        RMSE_training = train_lm_monthly$results$RMSE)
result_lm
```

```
##                Type             Method RMSE_training
## 1 Monthly 1975-2019 lm: all predictors         302.4
```

We reach a RMSE of 302.4. Let's see if other combinations have better results. Let's continue with Region.

```
#   Lm: using Region
#    Seed setting for sampling replication
set.seed(1, sample.kind = "Rounding")

#    Training algorithm
train_lm_monthly <- train(Deaths ~ RegionName ,
                          method = "lm",
                          data = train_set_monthly)
#    Review of training results
train_lm_monthly
```

```
## Linear Regression
##
## 8639 samples
##    1 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 8639, 8639, 8639, 8639, 8639, 8639, ...
## Resampling results:
##
##   RMSE   Rsquared  MAE
##   354.5  0.9394    198.8
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
#    Keep RMSE results
result_lm<-bind_rows(result_lm,
                  data_frame(Type = "Monthly 1975-2019",
                             Method = "lm: Region",
                             RMSE_training = train_lm_monthly$results$RMSE))

result_lm
```

```
##                 Type            Method RMSE_training
## 1 Monthly 1975-2019 lm: all predictors         302.4
## 2 Monthly 1975-2019         lm: Region         354.5
```

The result with Region is 354.5, higher than *all predictors*. Let's try now Year

```
#   Lm: using Year
#    Seed setting for sampling replication
set.seed(1, sample.kind = "Rounding")

#    Training algorithm
train_lm_monthly <- train(Deaths ~ Year ,
                          method = "lm",
                          data = train_set_monthly)

#    Review of training results
train_lm_monthly
```

```
## Linear Regression
##
## 8639 samples
##    1 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 8639, 8639, 8639, 8639, 8639, 8639, ...
## Resampling results:
##
##   RMSE  Rsquared  MAE
##   1430  0.01322   1125
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
#    Keep RMSE results
result_lm<-bind_rows(result_lm,
                     data_frame(Type = "Monthly 1975-2019",
                                Method = "lm: Year",
                                RMSE_training = train_lm_monthly$results$RMSE))

result_lm
```

```
##                 Type            Method RMSE_training
## 1 Monthly 1975-2019 lm: all predictors         302.4
## 2 Monthly 1975-2019         lm: Region         354.5
## 3 Monthly 1975-2019           lm: Year        1430.3
```

A RMSE of 1430.3 is very high. Region alone is a much better predictor than Year alone. Let's try Month.

```
#   Lm: using Month
#    Seed setting for sampling replication
set.seed(1, sample.kind = "Rounding")
```

```
#    Training algorithm
train_lm_monthly <- train(Deaths ~ Month ,
                          method = "lm",
                          data = train_set_monthly)

#    Review of training results
train_lm_monthly
```

```
## Linear Regression
##
## 8639 samples
##    1 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 8639, 8639, 8639, 8639, 8639, 8639, ...
## Resampling results:
##
##   RMSE  Rsquared  MAE
##   1438  0.001966  1128
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
#    Keep RMSE results
result_lm<-bind_rows(result_lm,
                  data_frame(Type = "Monthly 1975-2019",
                             Method = "lm: Month",
                             RMSE_training = train_lm_monthly$results$RMSE))

result_lm
```

```
##                Type             Method RMSE_training
## 1 Monthly 1975-2019 lm: all predictors         302.4
## 2 Monthly 1975-2019         lm: Region         354.5
## 3 Monthly 1975-2019           lm: Year        1430.3
## 4 Monthly 1975-2019          lm: Month        1438.4
```

Months alone produces a worse result 1438.4 than Year. Let's try Year and Month combined.

```
#   Lm: using Year + Month
#    Seed setting for sampling replication
set.seed(1, sample.kind = "Rounding")

#    Training algorithm
train_lm_monthly <- train(Deaths ~ Year + Month ,
                          method = "lm",
                          data = train_set_monthly)
#    Review of training results
train_lm_monthly
```

```
## Linear Regression
##
```

```
## 8639 samples
##    2 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 8639, 8639, 8639, 8639, 8639, 8639, ...
## Resampling results:
##
##   RMSE  Rsquared  MAE
##   1429  0.01463   1124
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
#    Keep RMSE results
result_lm<-bind_rows(result_lm,
                     data_frame(Type = "Monthly 1975-2019",
                                Method = "lm: Year + Month",
                                RMSE_training = train_lm_monthly$results$RMSE))

result_lm
```

```
##                Type              Method RMSE_training
## 1 Monthly 1975-2019 lm: all predictors         302.4
## 2 Monthly 1975-2019          lm: Region         354.5
## 3 Monthly 1975-2019            lm: Year        1430.3
## 4 Monthly 1975-2019           lm: Month        1438.4
## 5 Monthly 1975-2019    lm: Year + Month        1429.2
```

The result is still quite high 1429.2. Let's try Region and Year.

```
#   Lm: using Region + Year
#    Seed setting for sampling replication
set.seed(1, sample.kind = "Rounding")

#    Training algorithm
train_lm_monthly <- train(Deaths ~ RegionName + Year ,
                          method = "lm",
                          data = train_set_monthly)
#    Review of training results
train_lm_monthly
```

```
## Linear Regression
##
## 8639 samples
##    2 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 8639, 8639, 8639, 8639, 8639, 8639, ...
## Resampling results:
##
##   RMSE   Rsquared  MAE
##   309.8  0.9537    193.2
```

```
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
#    Keep RMSE results
result_lm<-bind_rows(result_lm,
                     data_frame(Type = "Monthly 1975-2019",
                             Method = "lm: Region + Year",
                             RMSE_training = train_lm_monthly$results$RMSE))

result_lm
```

```
##                 Type             Method RMSE_training
## 1 Monthly 1975-2019 lm: all predictors         302.4
## 2 Monthly 1975-2019         lm: Region         354.5
## 3 Monthly 1975-2019           lm: Year        1430.3
## 4 Monthly 1975-2019          lm: Month        1438.4
## 5 Monthly 1975-2019   lm: Year + Month        1429.2
## 6 Monthly 1975-2019  lm: Region + Year         309.8
```

The combination of Region and Year reaches a value of 309.9. The Combination of *all predictors* is still better. Let's try now Region and Month.

```
#   Lm: using Region + Month
#    Seed setting for sampling replication
set.seed(1, sample.kind = "Rounding")

#    Training algorithm
train_lm_monthly <- train(Deaths ~ RegionName + Month ,
                          method = "lm",
                          data = train_set_monthly)
#    Review of training results
train_lm_monthly
```

```
## Linear Regression
##
## 8639 samples
##    2 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 8639, 8639, 8639, 8639, 8639, 8639, ...
## Resampling results:
##
##   RMSE   Rsquared  MAE
##   348.1  0.9416    203
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
#    Keep RMSE results
result_lm<-bind_rows(result_lm,
                     data_frame(Type = "Monthly 1975-2019",
                                Method = "lm: Region + Month",
                                 RMSE_training = train_lm_monthly$results$RMSE))

result_lm %>% arrange(RMSE_training)
```

```
##                Type             Method RMSE_training
## 1 Monthly 1975-2019 lm: all predictors         302.4
## 2 Monthly 1975-2019  lm: Region + Year         309.8
## 3 Monthly 1975-2019 lm: Region + Month         348.1
## 4 Monthly 1975-2019         lm: Region         354.5
## 5 Monthly 1975-2019   lm: Year + Month        1429.2
## 6 Monthly 1975-2019           lm: Year        1430.3
## 7 Monthly 1975-2019          lm: Month        1438.4
```

The result with Region and Month is 348.1. The best option is *all predictors*, with a RMSE of 302.4. It now becomes our baseline.

Now that we have chosen the baseline model with the lowest RMSE, let's calculate the accuracy using the test data set. For that we will train the algorithm using the 1975-2019 data set, and Region, Year and Month as predictors. Then we will use the train model to calculate the $\hat{Y}$.

```
#   Calculate accuracy and RMSE against test data
#    Seed setting for sampling replication
set.seed(1, sample.kind = "Rounding")

#    Training algorithm  Including all predictors
train_lm_monthly <- train(Deaths ~ RegionName + Year + Month,
                          method = "lm",
                          data = train_set_monthly)

#   Review of training results
train_lm_monthly
```

```
## Linear Regression
##
## 8639 samples
##    3 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 8639, 8639, 8639, 8639, 8639, 8639, ...
## Resampling results:
##
##   RMSE   Rsquared  MAE
##   302.4  0.9559    190.2
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
#   Calculate y_hat
y_hat_lm_monthly <- predict(train_lm_monthly, test_set_monthly)

#  Quick review of results
y_hat_lm_monthly[1:10]
```

```
##    1    2    3    4    5    6    7    8    9   10
## 5114 5133 5152 5120 5233 5271 5290 5182 5257 5244
```

```
test_set_monthly$Deaths[1:10]
```

```
##  [1] 5849 5534 5013 5378 5918 7234 8673 5229 6484 6088
```

Having a look at the results of $\hat{Y}$, we see decimal values predicted, which in reality is wrong, since we cannot have 0.5 deaths, we have 0, 1, 2 . . . number of deaths being a discrete variable with integer values. We will simply round the value resulting from the *predict* function, and use that as our $\hat{Y}$. Finally, the calculation of accuracy will be done.

```
#  Calculate round values of y_hat
y_hat_lm_monthly <- round(y_hat_lm_monthly)

# Result table to keep Model results
result <- data.frame(Type = "Monthly 1975-2019",
                     Method = "lm: all predictors",
                     Accuracy = accuracy_f(test_set_monthly$Deaths,y_hat_lm_monthly),
                     RMSE_test = rmse_f(test_set_monthly$Deaths,y_hat_lm_monthly),
                     RMSE_training = train_lm_monthly$results$RMSE)

result
```

```
##                Type             Method Accuracy RMSE_test RMSE_training
## 1 Monthly 1975-2019 lm: all predictors 0.002314     349.8         302.4
```

The RMSE in the test data set goes up to 349.8. The accuracy of 0.002314 is quite low. Nevertheless, this is the baseline.

Loess

The next step is to build and evaluate a model based on the Loess algorithm. We will use *all predictors* combination in the model, as in Linear Regression one. As for the tuning parameters:

- *Degree*: we will use $degree = 1$ as a fixed value.
- *Span*: We will use the cross-validation facility of the *Caret package* in the *train* function, to find the best span value. The vector to be tried goes from 0.25 to 0.95, and has 10 values.

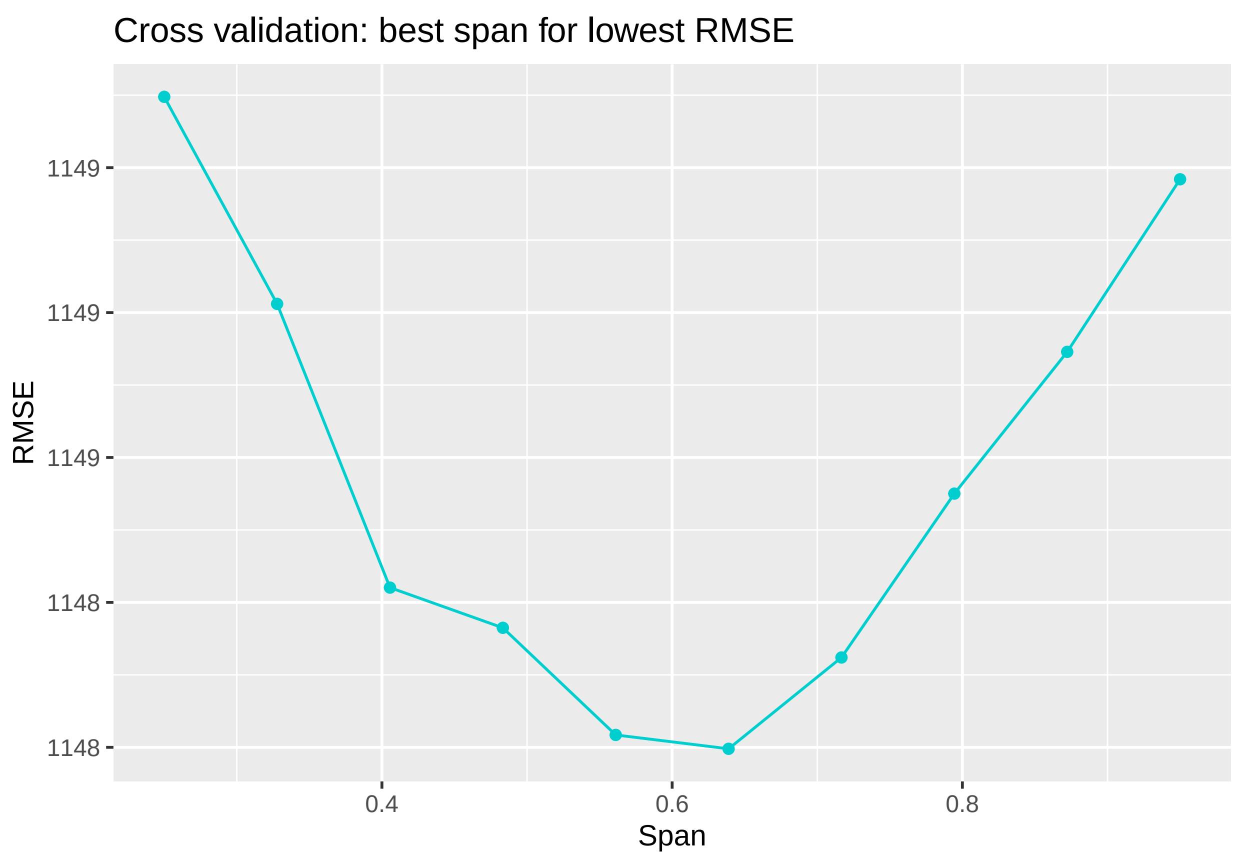
Let’s train the model and see the cross validation results.

```
#   Seed setting for sampling replication
set.seed(1, sample.kind = "Rounding")

#   Definition of span parameters to be tested in the model.
grid <- expand.grid(span = seq(0.25, 0.95, len = 10), degree = 1)

#   Training algorithm
train_loess <- train(Deaths ~ RegionCode+ Year + Month,
                          method = "gamLoess",
                          tuneGrid = grid,
```

```r
                             data = train_set_monthly)

#   Review training results
train_loess
```

```
## Generalized Additive Model using LOESS
##
## 8639 samples
##    3 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 8639, 8639, 8639, 8639, 8639, 8639, ...
## Resampling results across tuning parameters:
##
##   span    RMSE  Rsquared  MAE
##   0.2500  1149  0.3577    752.4
##   0.3278  1149  0.3581    751.8
##   0.4056  1149  0.3587    749.9
##   0.4833  1148  0.3587    749.8
##   0.5611  1148  0.3590    749.1
##   0.6389  1148  0.3590    748.7
##   0.7167  1148  0.3588    748.5
##   0.7944  1149  0.3585    748.6
##   0.8722  1149  0.3582    748.4
##   0.9500  1149  0.3579    748.4
##
## Tuning parameter 'degree' was held constant at a value of 1
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were span = 0.6389 and degree = 1.
```

```r
#   Graphic cross validation
train_loess$results %>% ggplot(aes(x=span, y=RMSE)) +
  geom_point(color="cyan3") +
  geom_line(color="cyan3") +
  xlab("Span")+
  ylab("RMSE")+
  ggtitle("Cross validation: best span for lowest RMSE")
```



The best *Span* is 0.6389 for RMSE 1148. Now let's calculate $\hat{Y}$ and the accuracy and RMSE in the test data set.

```
#   Obtain value of the lowest RMSE in the model
lowest_rmse<-train_loess$results$RMSE[which.min(train_loess$results$RMSE)]
span <- train_loess$results$span[which.min(train_loess$results$RMSE)]


#   Calculate y_hat
y_hat_loess <- predict(train_loess, test_set_monthly)

#   Quick review of results
y_hat_loess[1:10]
```

```
##    1    2    3    4    5    6    7    8    9   10
## 1790 1715 1650 1703 1752 1927 2020 1647 1914 1901
```

```
#   Transform decimal to integer values
y_hat_loess <- round(y_hat_loess)

#   Keep result table
result<-bind_rows(result,
                  data_frame(Type = "Monthly 1975-2019",
                             Method = paste("loess: span =",round(span,4)) ,
                             Accuracy = accuracy_f(test_set_monthly$Deaths,
                                                   y_hat_loess),
                             RMSE_test  = rmse_f(test_set_monthly$Deaths,
                                                 y_hat_loess),
                             RMSE_training = lowest_rmse))

result
```

```
##                 Type               Method Accuracy RMSE_test RMSE_training
## 1 Monthly 1975-2019   lm: all predictors 0.002314     349.8         302.4
## 2 Monthly 1975-2019 loess: span = 0.6389 0.003239    1070.1        1148.2
```

Even though the accuracy is a little bit better than the Linear Regression, the RMSE is much worse. Therefore, this model is discarded.

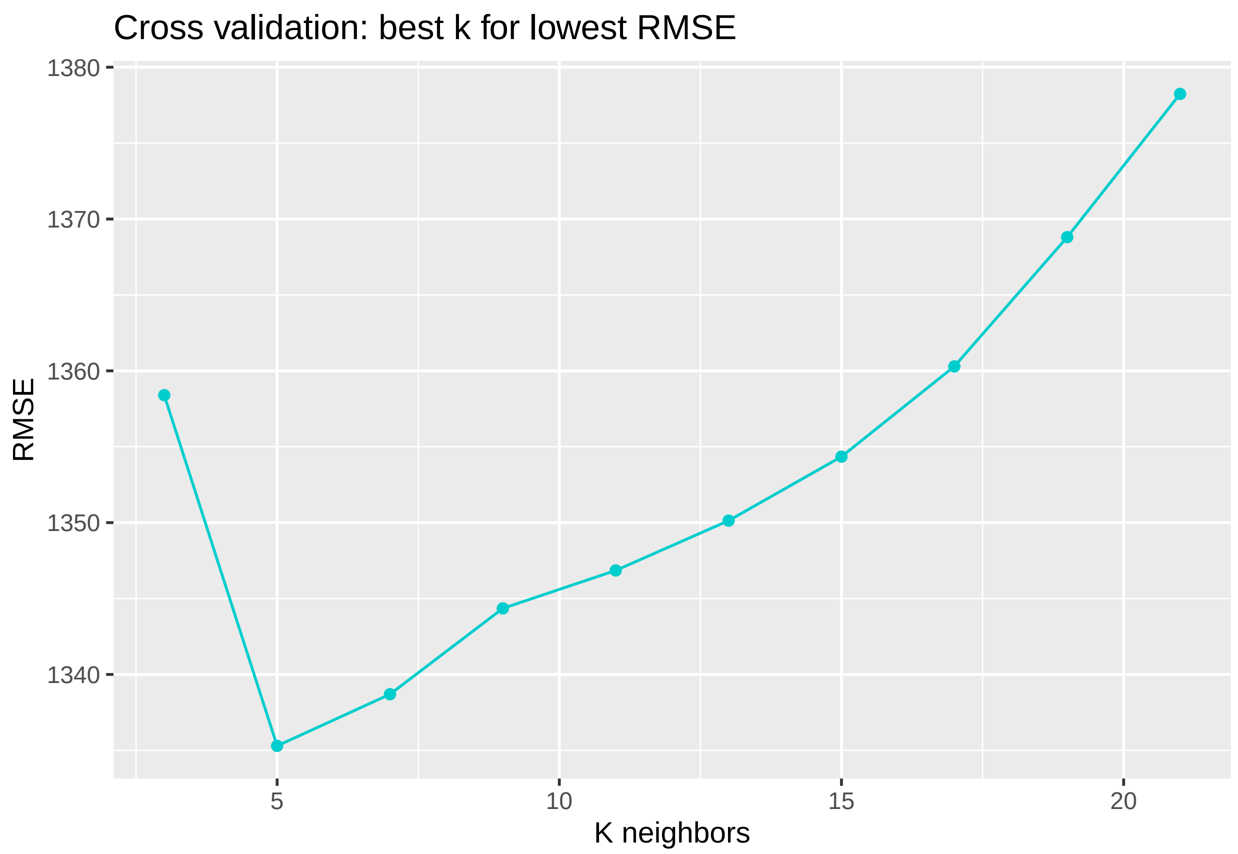
KNN (N neighbors)

Now it is time for the KNN algorithm. We will keep using *all predictors* in the model. As for the tuning parameter, we must find k, which represents the number of neighbors. For that we will create a vector that goes from 3 to 21, using only odd numbers. Again, we will use the cross validation facility of the *Caret package* to determine the best k value.

```
#  Model KNN method. We will cross-validation in the train model to find the
#  best K (number of neighbor). All predictors are used.

#   Seed setting for sampling replication
set.seed(1, sample.kind = "Rounding")

#   Training algorithm
train_knn <- train(Deaths ~ RegionCode+ Year + Month,
                   method = "knn",
                   tuneGrid = data.frame(k = seq(3, 21, 2)),
                   data = train_set_monthly)

#   Review of cross-validation results. Best K =5
train_knn
```

```
## k-Nearest Neighbors
##
## 8639 samples
##    3 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 8639, 8639, 8639, 8639, 8639, 8639, ...
## Resampling results across tuning parameters:
##
##   k   RMSE  Rsquared  MAE
##    3  1358  0.2866     970.5
##    5  1335  0.2365    1042.5
##    7  1339  0.2099    1084.3
##    9  1344  0.2011    1100.0
##   11  1347  0.1977    1104.0
##   13  1350  0.1914    1106.1
##   15  1354  0.1820    1107.9
##   17  1360  0.1677    1110.2
##   19  1369  0.1470    1114.0
##   21  1378  0.1241    1118.0
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 5.
```

```
#   Graphic cross validation
train_knn$results %>% ggplot(aes(x=k, y=RMSE)) +
  geom_point(color="cyan3") +
  geom_line(color="cyan3") +
  xlab("K neighbors")+
  ylab("RMSE")+
  ggtitle("Cross validation: best k for lowest RMSE")
```



The best RMSE is 1335 using $k = 5$. These are not good results, but let's confirm accuracy against the test data set.

```
#   Obtain value of the lowest RMSE in the model
lowest_rmse<-train_knn$results$RMSE[which.min(train_knn$results$RMSE)]
k_value <- train_knn$results$k[which.min(train_knn$results$RMSE)]

#   Calculate y_hat
y_hat_knn <- predict(train_knn, test_set_monthly)
```

```
#   Transform decimal to integer values
y_hat_knn <- round(y_hat_knn)

#  Keep result table
result<-bind_rows(result,
                  data_frame(Type = "Monthly 1975-2019",
                             Method = paste("knn with k= ",k_value),
                             Accuracy = accuracy_f(test_set_monthly$Deaths,
                                                    y_hat_knn),
                             RMSE_test = rmse_f(test_set_monthly$Deaths,
                                                y_hat_knn),
                             RMSE_training = lowest_rmse))


result
```

```
##                Type               Method  Accuracy RMSE_test RMSE_training
## 1 Monthly 1975-2019   lm: all predictors 0.0023137     349.8         302.4
## 2 Monthly 1975-2019 loess: span = 0.6389 0.0032392    1070.1        1148.2
## 3 Monthly 1975-2019       knn with k=  5 0.0004627    1290.1        1335.3
```

The RMSE in the test data set is a little better than the one in training data set, but still too high. Accuracy is much worst than that obtained in the Linear Regression results.

Random Forest

We will continue with the Random Forest algorithm. We will keep using *all predictors* combination in the model. As for the tuning parameter, we must find *mtry*, which represents the number of variables for splitting at each tree node, with the caeat that the bigger it, the more complex the processing. While executing this model, we faced performance issues due to long time calculations. Therefore, we will present here a reduced version of the vector, that will included only 3 values: $(5, 10, 15)$. Again, we will use the cross validation facility of the *Caret package* to determine the best *mtry* value.

```
#   Seed setting for sampling replication
set.seed(1, sample.kind = "Rounding")

date()
```

```
## [1] "Sun Feb 07 20:12:37 2021"
```

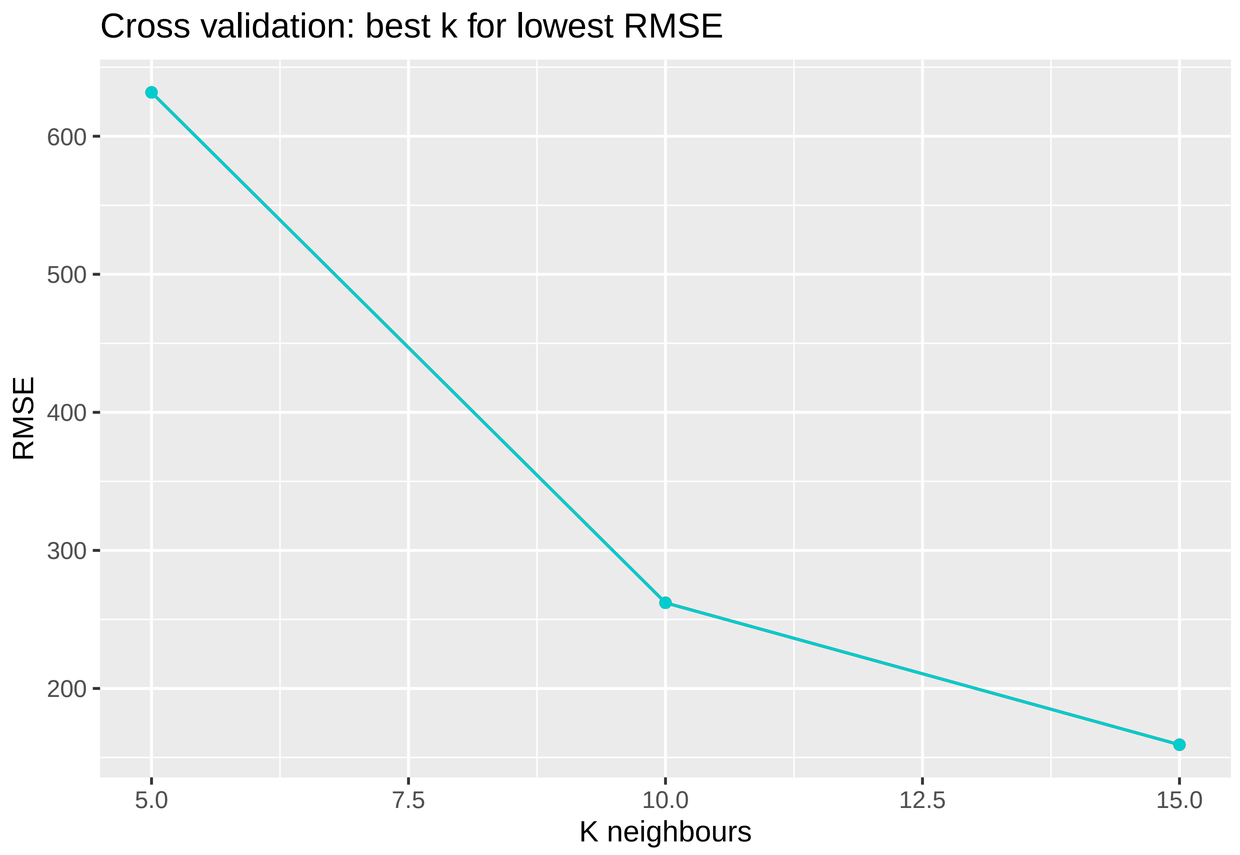
```
#   Training algorithm
train_rf <- train(Deaths ~ RegionName+ Year + Month,
                  method = "rf",
                  tuneGrid = data.frame(mtry = c(5,10,15)),
                  data = train_set_monthly)
date()
```

```
## [1] "Sun Feb 07 20:55:50 2021"
```

```r
#   Review of cross-validation results.
train_rf
```

```
## Random Forest
##
## 8639 samples
##    3 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 8639, 8639, 8639, 8639, 8639, 8639, ...
## Resampling results across tuning parameters:
##
##   mtry  RMSE   Rsquared  MAE
##    5    631.8  0.8184    321.02
##   10    262.1  0.9700    139.44
##   15    159.2  0.9879     84.02
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was mtry = 15.
```

```r
#   Graphic cross validation
train_rf%>% ggplot(aes(x=k, y=RMSE)) +
  geom_point(color="cyan3") +
  geom_line(color="cyan3") +
  xlab("K neighbours")+
  ylab("RMSE")+
  ggtitle("Cross validation: best k for lowest RMSE")
```



The best *mtry* value is 15. Since the graph does not show a parabola, but a descending line, it is likely that a higher value for *mtry* will produce an even better result. However, as indicated before, due to performance constrains, we will stop the optimization at this point.

```
#   Obtain value of the lowest RMSE in the model
lowest_rmse<-train_rf$results$RMSE[which.min(train_rf$results$RMSE)]
mtry_value <- train_rf$results$mtry[which.min(train_rf$results$RMSE)]


#   Review best tune
train_rf$bestTune
```

```
##   mtry
## 3   15
```

```
#   Obtain variable importance in the model
imp<- varImp(train_rf)
imp
```

```
## rf variable importance
##
##   only 20 most important variables shown (out of 21)
##
##                                         Overall
## RegionNameCataluña                      100.000
## RegionNamePaís Vasco                   31.973
## RegionNameMadrid, Comunidad de         27.698
## RegionNameComunitat Valenciana         27.233
## RegionNameCastilla - La Mancha         17.273
## RegionNameAragón                       11.888
## RegionNameCastilla y León              11.337
## RegionNameGalicia                       9.679
## Year                                    5.725
## RegionNameAsturias, Principado de       5.487
## RegionNameMelilla                       5.342
## RegionNameCeuta                         5.143
## RegionNameCanarias                      4.062
## RegionNameRioja, La                     3.517
## RegionNameExtranjero                    3.398
## RegionNameNavarra, Comunidad Foral de   2.635
## RegionNameCantabria                     2.213
## RegionNameBalears, Illes                1.897
## RegionNameExtremadura                   1.402
## RegionNameMurcia, Región de             0.514
```

```
#   Calculate y_hat
y_hat_rf <- predict(train_rf, test_set_monthly)

#   Transform decimal to integer values
y_hat_rf <- round(y_hat_rf)

#  Keep result table
result<-bind_rows(result,
                  data_frame(Type = "Monthly 1975-2019",
                             Method = paste("rf with mtry= ",mtry_value),
                             Accuracy = accuracy_f(test_set_monthly$Deaths,
                                                   y_hat_rf),
                             RMSE_test = rmse_f(test_set_monthly$Deaths,
                                                y_hat_rf),
                             RMSE_training = lowest_rmse))


result
```

```
##                Type               Method  Accuracy RMSE_test RMSE_training
## 1 Monthly 1975-2019   lm: all predictors 0.0023137     349.8         302.4
## 2 Monthly 1975-2019 loess: span = 0.6389 0.0032392    1070.1        1148.2
## 3 Monthly 1975-2019       knn with k=  5 0.0004627    1290.1        1335.3
## 4 Monthly 1975-2019    rf with mtry=  15 0.0143452     183.7         159.2
```

The results shows an **Accuracy** of 0.014, which is a great improvement over 0.0023 of the linear regression result. The RMSE also shows an important improvement reducing the value from 349.8 to 159.2.

Additionally, the *importance* review helps us visualize what combinations are more relevant in the Random Forest prediction. The top 3 are the following Regions: *País Vasco*, *Madrid* and *Comunitat Valenciana*. In the Region analysis, we found *País Vasco* ranked as number 7 in the number of deaths, *Madrid* in the 3rd position and *Comunitat Valenciana* in the 4th.

This model improves significantly the linear regression results.

Model selection, evaluation and conclusions

For the mortality data from 1975-2019, the model that produces better **Accuracy** and **RMSE** is the one built using **Random Forest** algorithm, **all predictors** (RegionName, Year and Month), and using tuning parameter $mtry = 15$.

Now let's evaluate the selected model with the 1975-2020/1 data set. This means that we should train the model again with the post COVID-19 data set, but keeping the model designed selected. We will start by comparing with the linear regression model as the baseline.

```
# Compere model with Covid data (first semester 2020).
#  Model Lm
#    Seed setting for sampling replication
set.seed(1, sample.kind = "Rounding")

#    Training algorithm  Including all predictors
train_lm_covid <- train(Deaths ~ RegionName + Year + Month,
                          method = "lm",
                          data = train_set_covid)

#   Review of training results
train_lm_covid
```

```
## Linear Regression
##
## 8831 samples
##    3 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 8831, 8831, 8831, 8831, 8831, 8831, ...
## Resampling results:
##
##   RMSE   Rsquared  MAE
##   330.1  0.9473    195.5
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
#   Calculate y_hat
y_hat_lm_covid <- predict(train_lm_covid, test_set_covid)

#   Transform decimal to integer values
y_hat_lm_covid <- round(y_hat_lm_covid)

# Result table to keep Model results
result<-bind_rows(result,
                  data_frame(Type = "Monthly 1975-2020/1",
                             Method = "lm: all predictors",
                             Accuracy = accuracy_f(test_set_covid$Deaths,y_hat_lm_covid),
                             RMSE_test = rmse_f(test_set_covid$Deaths,y_hat_lm_covid),
                             RMSE_training = train_lm_covid$results$RMSE))

result
```

```
##                 Type              Method  Accuracy RMSE_test RMSE_training
## 1   Monthly 1975-2019   lm: all predictors 0.0023137     349.8         302.4
## 2   Monthly 1975-2019 loess: span = 0.6389 0.0032392    1070.1        1148.2
## 3   Monthly 1975-2019       knn with k=  5 0.0004627    1290.1        1335.3
## 4   Monthly 1975-2019    rf with mtry=  15 0.0143452     183.7         159.2
## 5 Monthly 1975-2020/1   lm: all predictors 0.0031689     379.5         330.1
```

The *Accuracy* is 0.0031689, which is actually better for the 1975-2020/1 data set. The RMSE of 379.5, on the other hand, is not better than the one with pre COVID-19 data.

The result of having better *Accuracy* using linear regression in 1975-2020/1 is quite surprising. However, the *RMSE* is actually worse. Giving more relevance to *RMSE*, we can conclude that the model in general performs worse than the one with pre COVID-19 data.

Now let's review the results using Random Forest.

```r
#  Model Random Forest method "rf" with Covid Data and mtry=15 found in previous model.

#   Seed setting for sampling replication
set.seed(1, sample.kind = "Rounding")

#   Training algorithm
train_rf_covid <- train(Deaths ~ RegionName+ Year + Month,
                  method = "rf",
                  tuneGrid = data.frame(mtry = 15),
                  data = train_set_covid)

#   Review of cross-validation results.
train_rf_covid
```

```
## Random Forest
##
## 8831 samples
##    3 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 8831, 8831, 8831, 8831, 8831, 8831, ...
## Resampling results:
##
##   RMSE   Rsquared  MAE
##   180.2  0.9844    89.07
##
## Tuning parameter 'mtry' was held constant at a value of 15
```

```r
#   Obtain value of the lowest RMSE in the model
lowest_rmse<-train_rf_covid$results$RMSE[which.min(train_rf_covid$results$RMSE)]
mtry_value <- train_rf_covid$results$mtry[which.min(train_rf_covid$results$RMSE)]

#   Calculate y_hat
y_hat_rf_covid <- predict(train_rf_covid, test_set_covid)

#   Transform decimal to integer values
y_hat_rf_covid <- round(y_hat_rf_covid)
```

```r
#  Keep result table
result<-bind_rows(result,
                  data_frame(Type = "Monthly 1975-2020/1",
                             Method = paste("rf with mtry= ",mtry_value),
                             Accuracy = accuracy_f(test_set_covid$Deaths,
                                                   y_hat_rf_covid),
                             RMSE_test = rmse_f(test_set_covid$Deaths,
                                                y_hat_rf_covid),
                             RMSE_training = lowest_rmse))
result
```

```
##                    Type               Method  Accuracy RMSE_test RMSE_training
## 1   Monthly 1975-2019   lm: all predictors 0.0023137     349.8         302.4
## 2   Monthly 1975-2019 loess: span = 0.6389 0.0032392    1070.1        1148.2
## 3   Monthly 1975-2019        knn with k=  5 0.0004627    1290.1        1335.3
## 4   Monthly 1975-2019    rf with mtry=  15 0.0143452     183.7         159.2
## 5 Monthly 1975-2020/1   lm: all predictors 0.0031689     379.5         330.1
## 6 Monthly 1975-2020/1    rf with mtry=  15 0.0117700     218.8         180.2
```

The **Accuracy** results of 0.0117 shows a decrease of 18%. The **RMSE** of 218.8 implies an increase (negative result) of 19% compare with the model for 1975-2019. We can concluded that our model performs worse using the post COVID-19 data, that includes the impact of the first COVID-19 wave (a 100 year event).

It is also true, that the decrease in performance is relevant but the results of Random Forest post COVID-19 are still better than Linear Regression for pre COVID-19 , making the model useful, even though the predictability lost is around 18-19%.