HarvardX: PH125.9x Data Science: Capstone - Choose Your Own. Coronavirus Prediction

Carlos Mayora

5/19/2020

Contents

1 Introduction

Coronaviruses are a large group of viruses that can cause illness in animals and humans. Some coronaviruses commonly circulate in the air and usually cause upper respiratory symptoms such as cough or runny nose, although some can cause more serious illness.

The 2019 novel (new) coronavirus causes the illness Coronavirus disease (COVID-19), which is an infectious disease caused by a newly discovered coronavirus, it was identified in late 2019 and was declared a pandemic on March 11 2020. At the beginning, local hospitals in Wuhan City, Hubei Province, China, were reported unusual number of patients who comes with severe pneumonia without knowing cause and not responds to any kind of vaccine or medicine. Besides, these cases were further increased because of human to human transmission, and doctors confirmed that this unknown disease had similar epidemic of Severe Acute Respiratory Syndrome (SARS)2 in 2002 and the agent causing this disease was recognized as a coronavirus.

The disease started as a local epidemic of Wuhan, China, but it quickly escalated all over the world, being transmitted by international travelers, making it an international public health emergency. There is no scientific evidence for where it has originated although is believed to have originally occurred from animal-to-person contact and spreads person-to-person.

Coronaviruses like COVID-19 are most often spread through the air by coughing or sneezing, through close personal contact (including touching and shaking hands) or through touching your nose, mouth or eyes before washing your hands. This is a new disease and we are still learning about how it spreads and the severity of illness it causes.

Most people infected with the COVID-19 virus will experience mild to moderate respiratory illness and recover without requiring special treatment. Older people, and those with underlying medical problems like cardiovascular disease, diabetes, chronic respiratory disease, and cancer are more likely to develop serious illness.

COVID-19 has turned the world upside down, most of the countries are currently in some degree of “lock-down”, with restaurants, bars, shops, schools and gyms closed, and citizens required, or at least strongly encouraged, to stay home to avoid catching or spreading COVID-19. This has impacted everything, how we live and interact with each other, how we work and communicate, how we move around and travel. Every aspect of our lives has been affected.

Data is critical to understand the global COVID-19 pandemic. Decisions made now and in the upcoming months will be some of the most important made in generations. They will affect people all around the world for years to come. It is imperative that governments making those decisions have access to the best information available.

1.1 Project Goal

The aim of this report is to predict whether a COVID-19 patient in Colombia will recover or not. We will use data from the National Health Institute of Colombia (Instituto Nacional de Salud) from where we will create a subset of the data (training set) and train using machine learning algorithms to predict the outcome in the prediction set, this prediction set will be all the current active cases which we don’t know the outcome yet.

We will apply different machine learning algorithms and compare the accuracy, sensitivity and specificity of each one in order to select the best model to use in our final outcome prediction.

1.2 Dataset

For this project we will use the data provided by the National Health Institute of Colombia, which is available through a CSV file with all the COVID-19 cases in Colombia, basically this file contains the historic of all COVID-19 cases in Colombia.

We will download and read the CSV file, with the following information:

- ID de caso: unique identifier of the record.
- Fecha de notificacion: date when was notified to the National Health Institute.
- Codigo DIVIPOLA: municipality code.
- Ciudad de ubicacion: City where the patient is located.
- Departamento o Distrito: State where the patient is located.
- atencion: outcome including the of type of medical atention, like recovered or death, or UCI, home or hospital.
- Edad: age of the patient.
- Sexo: sex of the patient.
- Tipo: type and status of the case, imported from other country, in analysis.
- Estado: severity of the case.
- Pais de procedencia: contagion Country.
- FIS: date when the symptoms started. We will use this date as the start date for the case.
- Fecha de muerte: date of death (if applies).
- Fecha diagnostico: date of diagnostic, basically is the date when the case is confirmed by the medical lab analysis.
- Fecha recuperado: recovery date (if applies).
- fecha reporte web: date when the case was registered on the web site.
- Tipo recuperacion: recovery type, two possible values: PCR, second negative medical test and Tiempo which is considered recovered after 30 days without symptoms.
- Codigo Pais: contagion Country code.

In following sections we will work on transforming this dataset to make columns and values English readable.

2 Data Analysis

2.1 Data Ingestion

For purpose of this project we have already downloaded the .csv file provided by the National Health Institute of Colombia, next we will read and load the data in to a dataset so we can use it in the analysis, also we are installing the packages we will need through out the project.

```
################################
# Load libraries and data
################################

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(glmnet)) install.packages("glmnet", repos = "http://cran.us.r-project.org")
if(!require(rpart)) install.packages("rpart", repos = "http://cran.us.r-project.org")
if(!require(rattle)) install.packages("rattle", repos = "http://cran.us.r-project.org")
if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org")

# To make our coronavirus predictions we will use the data
# repository for Colombia COVID-19 cases at
# https://www.datos.gov.co/Salud-y-Protecci-n-Social/Casos-positivos-de-COVID-19-en-Colombia/gt2j-8ykr

coronavirus <- read.csv('Casos_positivos_de_COVID-19_en_Colombia.csv',
                        stringsAsFactors = FALSE, na.strings='')
```

2.2 Data Exploration

Before we start building out recovery predictions, we need to get familiar and understand the data structure of the dataset in order to build a better model. First let's get the number of rows and columns in the *coronavirus* dataset:

```
# coronavirus dataset rows and columns
dim(coronavirus)
```

```
## [1] 45212    18
```

Coronavirus dataset structure.

```
# coronavirus structure
str(coronavirus)
```

```
## 'data.frame':    45212 obs. of  18 variables:
##  $ ID.de.caso            : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Fecha.de.notificación  : chr  "2020-03-02T00:00:00.000" "2020-03-06T00:00:00.000" "2020-03-07T00:
##  $ Codigo.DIVIPOLA        : int  11001 76111 5001 5001 5001 5360 13001 11001 11001 11001 ...
##  $ Ciudad.de.ubicación    : chr  "Bogotá D.C." "Guadalajara de Buga" "Medellín" "Medellín" ...
##  $ Departamento.o.Distrito: chr  "Bogotá D.C." "Valle del Cauca" "Antioquia" "Antioquia" ...
##  $ atención               : chr  "Recuperado" "Recuperado" "Recuperado" "Recuperado" ...
##  $ Edad                   : int  19 34 50 55 25 27 85 22 28 36 ...
##  $ Sexo                   : chr  "F" "M" "F" "M" ...
##  $ Tipo                     : chr  "Importado" "Importado" "Importado" "Relacionado" ...
##  $ Estado                   : chr  "Leve" "Leve" "Leve" "Leve" ...
##  $ País.de.procedencia      : chr  "Italia" "España" "España" NA ...
##  $ FIS                      : chr  "2020-02-27T00:00:00.000" "2020-03-04T00:00:00.000" "2020-02-29T00:0
##  $ Fecha.de.muerte          : chr  NA NA NA NA ...
##  $ Fecha.diagnostico        : chr  "2020-03-06T00:00:00.000" "2020-03-09T00:00:00.000" "2020-03-09T00:0
##  $ Fecha.recuperado         : chr  "2020-03-13T00:00:00.000" "2020-03-19T00:00:00.000" "2020-03-15T00:0
##  $ fecha.reporte.web        : chr  "2020-03-06T00:00:00.000" "2020-03-09T00:00:00.000" "2020-03-09T00:0
##  $ Tipo.recuperación        : chr  "PCR" "PCR" "PCR" "PCR" ...
##  $ Código.País              : int  380 724 724 NA NA NA 840 724 724 724 ...
```

We can see that the columns are all in Spanish, we will change the column names to make them more readable, and check the structure one more time. The name translation will be as follows:

- ID de caso - id
- Fecha de notificacion - record_date
- Codigo DIVIPOLA - municipality_code
- Ciudad de ubicacion - city
- Departamento o Distrito - state
- atencion - outcome
- Edad - age
- Sexo - sex
- Tipo - contagion_type
- Estado - severity
- Pais de procedencia - origin_country
- FIS - symptoms_date
- Fecha de muerte - date_of_death
- Fecha diagnostico - diagnosis_date
- Fecha recuperado - recovery_date
- fecha reporte web - web_date
- Tipo recuperacion - recovery_type
- Codigo Pais - origin_country_code

```r
# Change column names
names(coronavirus) <- c('id','record_date','municipality_code','city','state','outcome',
                        'age','sex', 'contagion_type','severity','origin_country', "symptoms_date",
                        'date_of_death','diagnosis_date','recovery_date','web_date','recovery_type',
                        'origin_country_code')

# coronavirus structure
str(coronavirus)
```

```
## 'data.frame':    45212 obs. of  18 variables:
##  $ id                : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ record_date       : chr  "2020-03-02T00:00:00.000" "2020-03-06T00:00:00.000" "2020-03-07T00:00:00
##  $ municipality_code : int  11001 76111 5001 5001 5001 5360 13001 11001 11001 11001 ...
##  $ city              : chr  "Bogotá D.C." "Guadalajara de Buga" "Medellín" "Medellín" ...
##  $ state             : chr  "Bogotá D.C." "Valle del Cauca" "Antioquia" "Antioquia" ...
##  $ outcome           : chr  "Recuperado" "Recuperado" "Recuperado" "Recuperado" ...
##  $ age               : int  19 34 50 55 25 27 85 22 28 36 ...
##  $ sex               : chr  "F" "M" "F" "M" ...
##  $ contagion_type    : chr  "Importado" "Importado" "Importado" "Relacionado" ...
##  $ severity          : chr  "Leve" "Leve" "Leve" "Leve" ...
##  $ origin_country     : chr  "Italia" "España" "España" NA ...
##  $ symptoms_date      : chr  "2020-02-27T00:00:00.000" "2020-03-04T00:00:00.000" "2020-02-29T00:00:0
##  $ date_of_death      : chr  NA NA NA NA ...
##  $ diagnosis_date     : chr  "2020-03-06T00:00:00.000" "2020-03-09T00:00:00.000" "2020-03-09T00:00:0
##  $ recovery_date      : chr  "2020-03-13T00:00:00.000" "2020-03-19T00:00:00.000" "2020-03-15T00:00:0
##  $ web_date           : chr  "2020-03-06T00:00:00.000" "2020-03-09T00:00:00.000" "2020-03-09T00:00:0
##  $ recovery_type      : chr  "PCR" "PCR" "PCR" "PCR" ...
##  $ origin_country_code: int  380 724 724 NA NA NA 840 724 724 724 ...
```

Now we have changed the column names, let's get the first 6 rows of the *coronavirus* dataset.

```
# 6 first rows of coronavirus dataset including column names
head(coronavirus)
```

```
##   id             record_date municipality_code                city
## 1  1 2020-03-02T00:00:00.000             11001         Bogotá D.C.
## 2  2 2020-03-06T00:00:00.000             76111 Guadalajara de Buga
## 3  3 2020-03-07T00:00:00.000              5001            Medellín
## 4  4 2020-03-09T00:00:00.000              5001            Medellín
## 5  5 2020-03-09T00:00:00.000              5001            Medellín
## 6  6 2020-03-10T00:00:00.000              5360              Itagüí
##             state    outcome age sex contagion_type severity origin_country
## 1     Bogotá D.C. Recuperado  19   F      Importado     Leve         Italia
## 2 Valle del Cauca Recuperado  34   M      Importado     Leve         España
## 3       Antioquia Recuperado  50   F      Importado     Leve         España
## 4       Antioquia Recuperado  55   M    Relacionado     Leve           <NA>
## 5       Antioquia Recuperado  25   M    Relacionado     Leve           <NA>
## 6       Antioquia Recuperado  27   F    Relacionado     Leve           <NA>
##             symptoms_date date_of_death          diagnosis_date
## 1 2020-02-27T00:00:00.000          <NA> 2020-03-06T00:00:00.000
## 2 2020-03-04T00:00:00.000          <NA> 2020-03-09T00:00:00.000
## 3 2020-02-29T00:00:00.000          <NA> 2020-03-09T00:00:00.000
## 4 2020-03-06T00:00:00.000          <NA> 2020-03-11T00:00:00.000
## 5 2020-03-08T00:00:00.000          <NA> 2020-03-11T00:00:00.000
## 6 2020-03-06T00:00:00.000          <NA> 2020-03-11T00:00:00.000
##             recovery_date                web_date recovery_type
## 1 2020-03-13T00:00:00.000 2020-03-06T00:00:00.000           PCR
## 2 2020-03-19T00:00:00.000 2020-03-09T00:00:00.000           PCR
## 3 2020-03-15T00:00:00.000 2020-03-09T00:00:00.000           PCR
## 4 2020-03-26T00:00:00.000 2020-03-11T00:00:00.000           PCR
## 5 2020-03-23T00:00:00.000 2020-03-11T00:00:00.000           PCR
## 6 2020-03-26T00:00:00.000 2020-03-11T00:00:00.000           PCR
##   origin_country_code
## 1                 380
## 2                 724
## 3                 724
## 4                  NA
## 5                  NA
## 6                  NA
```

We can confirm that the dataset contains 18 columns, which we have described in the Dataset section.

The dataset is not in tidy format, so before we continue we need to change the columns classes to date and factor accordingly and also translate the levels values.

```r
# Changing columns class to date and translating levels values
coronavirus$record_date <- as.Date(coronavirus$record_date)
coronavirus$municipality_code <- as.integer(coronavirus$municipality_code)
coronavirus$date_of_death <- as.Date(coronavirus$date_of_death)
coronavirus$city <- as.factor(coronavirus$city)
coronavirus$state <- as.factor(coronavirus$state)
coronavirus[,'outcome'] <- ifelse(coronavirus[,'outcome'] == "Recuperado", "recovered",
                    ifelse(coronavirus[,'outcome'] == "Fallecido", "deceased",
                      ifelse(coronavirus[,'outcome'] == "Hospital UCI", "icu",
                        ifelse(coronavirus[,'outcome'] == "Hospital", "hospitalized",
                          ifelse(coronavirus[,'outcome'] == "Casa", "outpatientCare",
                                 'unknown')))))

coronavirus$outcome <- as.factor(coronavirus$outcome)
coronavirus$sex <- as.factor(coronavirus$sex)
coronavirus[,'contagion_type'] <- ifelse(coronavirus[,'contagion_type'] == "Importado",
                                         "travel","contact")
coronavirus$contagion_type <- as.factor(coronavirus$contagion_type)
coronavirus[,'severity'] <- ifelse(coronavirus[,'severity'] == "Leve", "low",
                      ifelse(coronavirus[,'severity'] == "Moderado", "medium",
                        ifelse(coronavirus[,'severity'] == "Grave", "high",
                          ifelse(coronavirus[,'severity'] == "Fallecido", "death",
                            ifelse(coronavirus[,'severity'] == "Asintomático", "asymptomatic", NA)))))
coronavirus$severity <- as.factor(coronavirus$severity)
coronavirus$origin_country <- as.factor(coronavirus$origin_country)
coronavirus$symptoms_date <- as.Date(coronavirus$symptoms_date)
coronavirus$date_of_death <- as.Date(coronavirus$date_of_death)
coronavirus$diagnosis_date <- as.Date(coronavirus$diagnosis_date)
coronavirus$recovery_date  <- as.Date(coronavirus$recovery_date)
coronavirus$web_date  <- as.Date(coronavirus$web_date)
coronavirus[,'recovery_type'] <- ifelse(coronavirus[,'recovery_type'] == "PCR", "negTest",
                            ifelse(coronavirus[,'recovery_type'] == "Tiempo" |
                              coronavirus[,'recovery_type'] == "TIEMPO", "time", NA))
coronavirus$recovery_type  <- as.factor(coronavirus$recovery_type)
coronavirus$origin_country_code <- as.integer(coronavirus$origin_country_code)

# coronavirus structure after tyding
str(coronavirus)
```

```
## 'data.frame':    45212 obs. of  18 variables:
##  $ id                 : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ record_date        : Date, format: "2020-03-02" "2020-03-06" ...
##  $ municipality_code  : int  11001 76111 5001 5001 5001 5360 13001 11001 11001 11001 ...
##  $ city               : Factor w/ 493 levels "Ábrego","Acacías",..: 51 185 256 256 256 203 78 51 51 5
##  $ state              : Factor w/ 37 levels "Amazonas","Antioquia",..: 7 35 2 2 2 2 13 7 7 7 ...
##  $ outcome            : Factor w/ 6 levels "deceased","hospitalized",..: 5 5 5 5 5 5 5 5 5 5 ...
##  $ age                : int  19 34 50 55 25 27 85 22 28 36 ...
##  $ sex                : Factor w/ 2 levels "F","M": 1 2 1 2 2 1 1 1 1 1 ...
##  $ contagion_type     : Factor w/ 2 levels "contact","travel": 2 2 2 1 1 1 2 2 2 2 ...
##  $ severity           : Factor w/ 5 levels "asymptomatic",..: 4 4 4 4 4 4 4 4 4 4 ...
##  $ origin_country     : Factor w/ 39 levels "Alemania","Antillas Neerlandesas",..: 25 17 17 NA NA NA
##  $ symptoms_date      : Date, format: "2020-02-27" "2020-03-04" ...
##  $ date_of_death      : Date, format: NA NA ...
##  $ diagnosis_date     : Date, format: "2020-03-06" "2020-03-09" ...
##  $ recovery_date      : Date, format: "2020-03-13" "2020-03-19" ...
##  $ web_date           : Date, format: "2020-03-06" "2020-03-09" ...
##  $ recovery_type      : Factor w/ 2 levels "negTest","time": 1 1 1 1 1 1 1 1 1 1 ...
##  $ origin_country_code: int  380 724 724 NA NA NA 840 724 724 724 ...
```

Now the data is ready for exploration and analysis, each row represents a specific COVID-19 case in Colombia.

Since *symptoms_date* is the date of the onset of symptoms for each patient, we will consider this date to make any time analysis, so let's check the summary of the dataset, where we can see the min date (2020-02-27) and max date (2020-06-10), this represents the time frame of the COVID-19 cases in Colombia.

```
# Basic summary statistics
summary(coronavirus)
```

```
##        id          record_date          municipality_code
##  Min.   :    1   Min.   :2020-03-02   Min.   :    5
##  1st Qu.:11344   1st Qu.:2020-05-06   1st Qu.:11001
##  Median :22646   Median :2020-05-19   Median :11001
##  Mean   :22643   Mean   :2020-05-14   Mean   :28082
##  3rd Qu.:33949   3rd Qu.:2020-05-29   3rd Qu.:50001
##  Max.   :45252   Max.   :2020-06-10   Max.   :99001
##
##                    city                       state                 outcome
##  Bogotá D.C.        :14537   Bogotá D.C.        :14537   deceased      : 1488
##  Cartagena de Indias: 4303   Valle del Cauca    : 4411   hospitalized  : 2311
##  Barranquilla       : 4168   Cartagena D.T. y C.: 4303   icu           :  477
##  Cali               : 3886   Barranquilla D.E.  : 4168   outpatientCare:23072
##  Soledad            : 2233   Atlántico          : 3763   recovered     :17790
##  Leticia            : 1995   Amazonas           : 2075   unknown       :   74
##  (Other)            :14090   (Other)            :11955
##       age          sex        contagion_type          severity
##  Min.   :  0.00   F:20167   contact:44337   asymptomatic: 5125
##  1st Qu.: 26.00   M:25045   travel :  875   death       : 1482
##  Median : 36.00                             high        :  479
##  Mean   : 38.89                             low         :35697
##  3rd Qu.: 51.00                             medium      : 2360
##  Max.   :106.00                             NA's        :   69
##
##                     origin_country  symptoms_date        date_of_death
##  España                    :  261   Min.   :2020-02-27   Min.   :2020-03-16
##  Estados Unidos de América:  216   1st Qu.:2020-05-01   1st Qu.:2020-05-02
##  Ecuador                  :   54   Median :2020-05-15   Median :2020-05-20
##  México                   :   51   Mean   :2020-05-10   Mean   :2020-05-15
##  Brasil                   :   44   3rd Qu.:2020-05-25   3rd Qu.:2020-05-31
##  (Other)                  :  246   Max.   :2020-06-10   Max.   :2020-06-11
##  NA's                     :44340   NA's   :5125         NA's   :43683
##  diagnosis_date       recovery_date           web_date          recovery_type
##  Min.   :2020-03-06   Min.   :2020-03-13   Min.   :2020-03-06   negTest:10678
##  1st Qu.:2020-05-11   1st Qu.:2020-05-13   1st Qu.:2020-05-11   time   : 7112
##  Median :2020-05-26   Median :2020-05-31   Median :2020-05-26   NA's   :27422
##  Mean   :2020-05-20   Mean   :2020-05-22   Mean   :2020-05-20
##  3rd Qu.:2020-06-04   3rd Qu.:2020-06-03   3rd Qu.:2020-06-04
##  Max.   :2020-06-11   Max.   :2020-06-11   Max.   :2020-06-11
##  NA's   :319          NA's   :27511
##  origin_country_code
##  Min.   : 32.0
##  1st Qu.:484.0
##  Median :724.0
##  Mean   :610.9
##  3rd Qu.:840.0
##  Max.   :862.0
##  NA's   :44340
```

2.3 COVID-19 Situation In Colombia

All the world have been impacted by this pandemic, we can see the current situation in Colombia by getting the total confirmed cases number, total deaths, recovered and current active cases.

```r
# Let's get the number of cases per status
cat("The total number of confirmed cases is: ", nrow(coronavirus))
```

```
## The total number of confirmed cases is:  45212
```

```r
cat("The total number of deaths is: ", sum(coronavirus$outcome == "deceased"))
```

```
## The total number of deaths is:  1488
```

```r
cat("The total number of recovered is: ", sum(coronavirus$outcome == "recovered"))
```

```
## The total number of recovered is:  17790
```

```r
cat("The total number of active cases is: ", nrow(coronavirus) -
      sum(coronavirus$outcome == "deceased") - sum(coronavirus$outcome == "recovered") -
      sum(coronavirus$outcome == "unknown"))
```

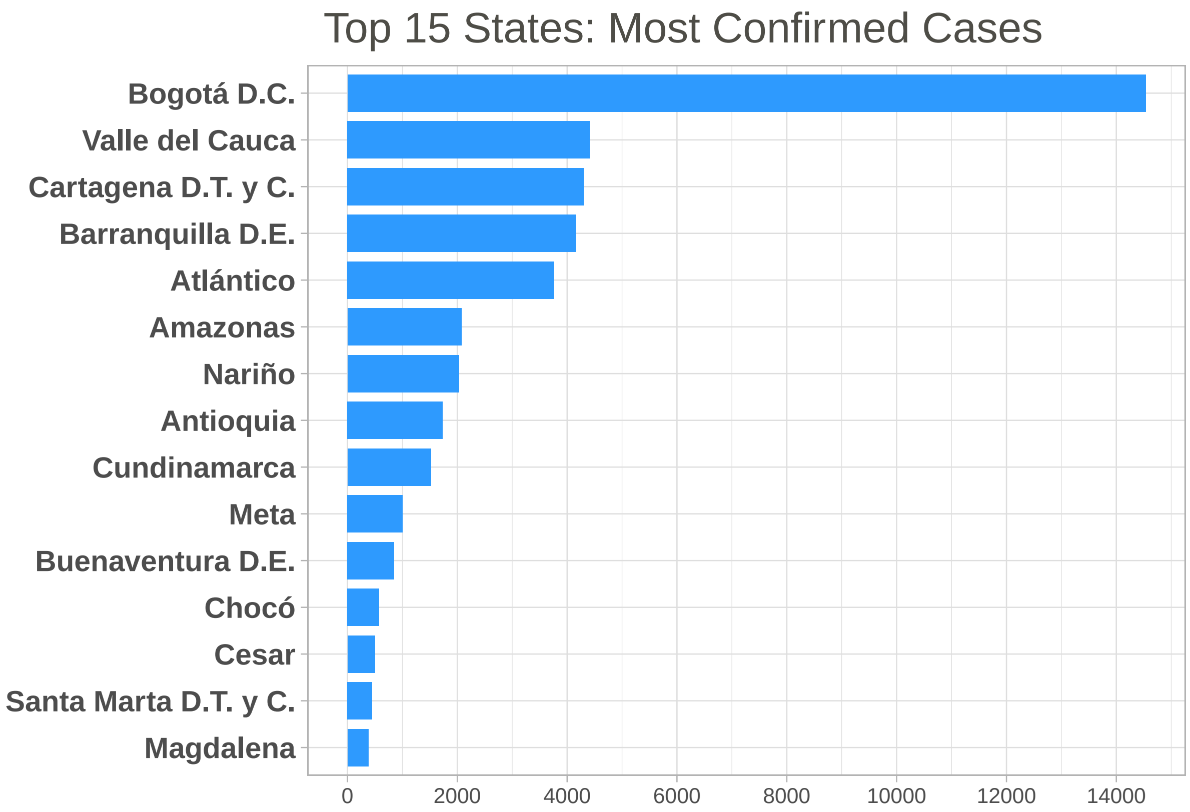
```
## The total number of active cases is:  25860
```

Let's see what states have been more impacted by COVID-19, first let's list and then plot the top 15 states with more confirmed cases.

```r
# Object with the total cases by state and type
totals_state <- coronavirus %>%
  group_by(state, outcome) %>%
  summarize(total = n())

# Let's get the list of the 15 most impacted states,
# those with most confirmed cases
totals_state %>% group_by(state) %>%
  summarize(total_cases = sum(total)) %>% arrange(-total_cases) %>% head(15)
```

```
## # A tibble: 15 x 2
##    state                 total_cases
##    <fct>                       <int>
##  1 Bogotá D.C.                  14537
##  2 Valle del Cauca               4411
##  3 Cartagena D.T. y C.           4303
##  4 Barranquilla D.E.             4168
##  5 Atlántico                     3763
##  6 Amazonas                      2075
##  7 Nariño                        2030
##  8 Antioquia                     1731
##  9 Cundinamarca                  1519
## 10 Meta                          1003
## 11 Buenaventura D.E.              847
## 12 Chocó                          576
## 13 Cesar                          499
## 14 Santa Marta D.T. y C.          451
## 15 Magdalena                      382
```

```
# Variable with colos we will use for the following plots
colors <- data.frame(
  type = c("confirmed", "active", "deceased", "recovered"),
  fill = c("#2E9AFE", "#F2F5A9", "#FA5858", "#81F79F"),
  color = c("#0404B4", "#E2D303", "#B40404", "#088A08"))

totals_state %>% group_by(state) %>%
  summarize(total_cases = sum(total)) %>% arrange(-total_cases) %>% head(15) %>%
  ggplot(aes(x = reorder(state,total_cases), y = total_cases)) +
  geom_bar(stat = "identity", fill  = colors$fill[colors$type == "confirmed"], width = 0.8) +
  scale_y_continuous(breaks = seq(0, 14000, by = 2000)) +
  coord_flip() +
  theme_light(base_size = 10) +
  labs(x = "", y = "", title = "Top 15 States: Most Confirmed Cases") +
  theme(axis.title = element_text(size = 14, colour = "black"),
        axis.text.y = element_text(size = 11, face = "bold"),
        plot.title = element_text(size= 16, hjust=0.1, color = "#4e4d47"))
```

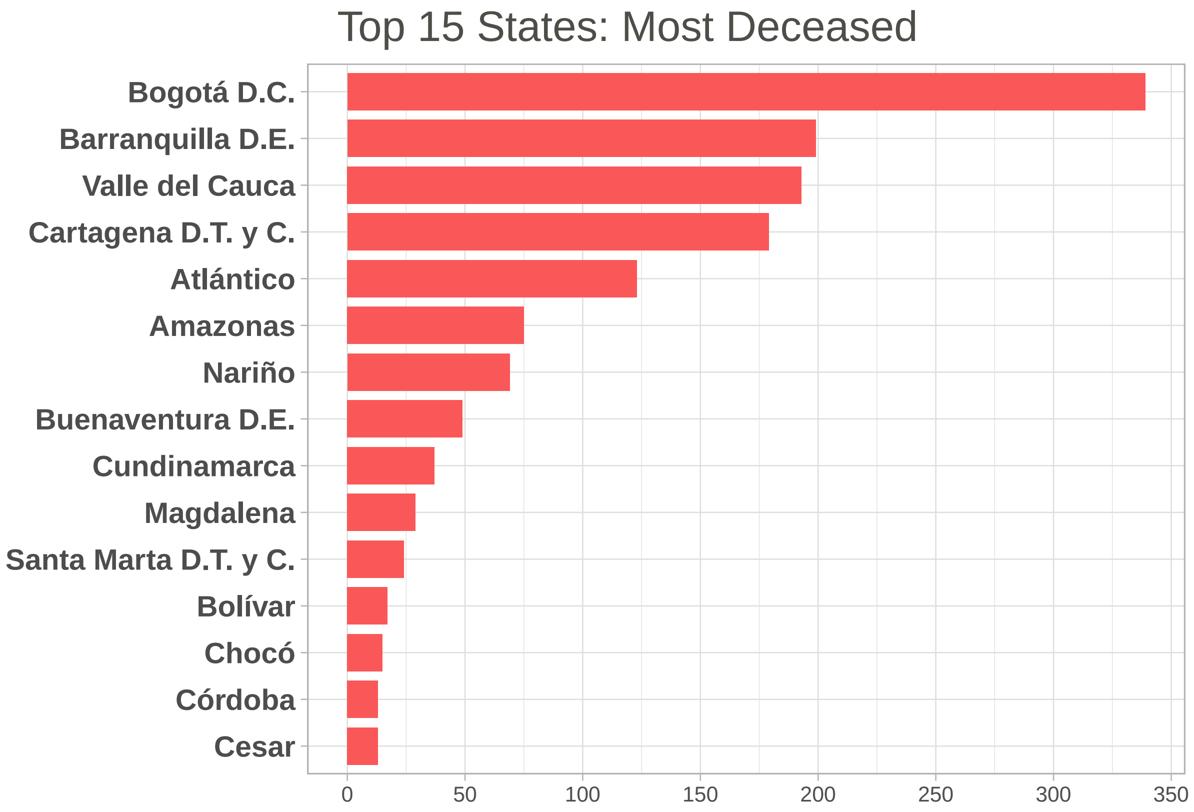


Top 15 states by death.

```r
# Let's get the list of the 15 states with most deaths
totals_state %>% filter(outcome == "deceased") %>%
  arrange(-total) %>% head(15)
```

```
## # A tibble: 15 x 3
## # Groups:   state [15]
##    state                outcome  total
##    <fct>                <fct>    <int>
##  1 Bogotá D.C.          deceased   339
##  2 Barranquilla D.E.    deceased   199
##  3 Valle del Cauca      deceased   193
##  4 Cartagena D.T. y C.  deceased   179
##  5 Atlántico            deceased   123
##  6 Amazonas             deceased    75
##  7 Nariño               deceased    69
##  8 Buenaventura D.E.    deceased    49
##  9 Cundinamarca         deceased    37
## 10 Magdalena            deceased    29
## 11 Santa Marta D.T. y C. deceased    24
## 12 Bolívar              deceased    17
## 13 Chocó                deceased    15
## 14 Cesar                deceased    13
## 15 Córdoba              deceased    13
```

```r
totals_state %>% filter(outcome == "deceased") %>%
  arrange(-total) %>% head(15) %>%
  ggplot(aes(x = reorder(state,total), y = total )) +
  geom_bar(stat = "identity", fill  = colors$fill[colors$type == "deceased"], width = 0.8) +
  scale_y_continuous(breaks = seq(0, 400, by = 50)) +
  coord_flip() +
  theme_light(base_size = 10) +
  labs(x = "", y = "", title = "Top 15 States: Most Deceased") +
  theme(axis.title = element_text(size = 14, colour = "black"),
        axis.text.y = element_text(size = 11, face = "bold"),
        plot.title = element_text(size= 16, hjust=0.1, color = "#4e4d47"))
```

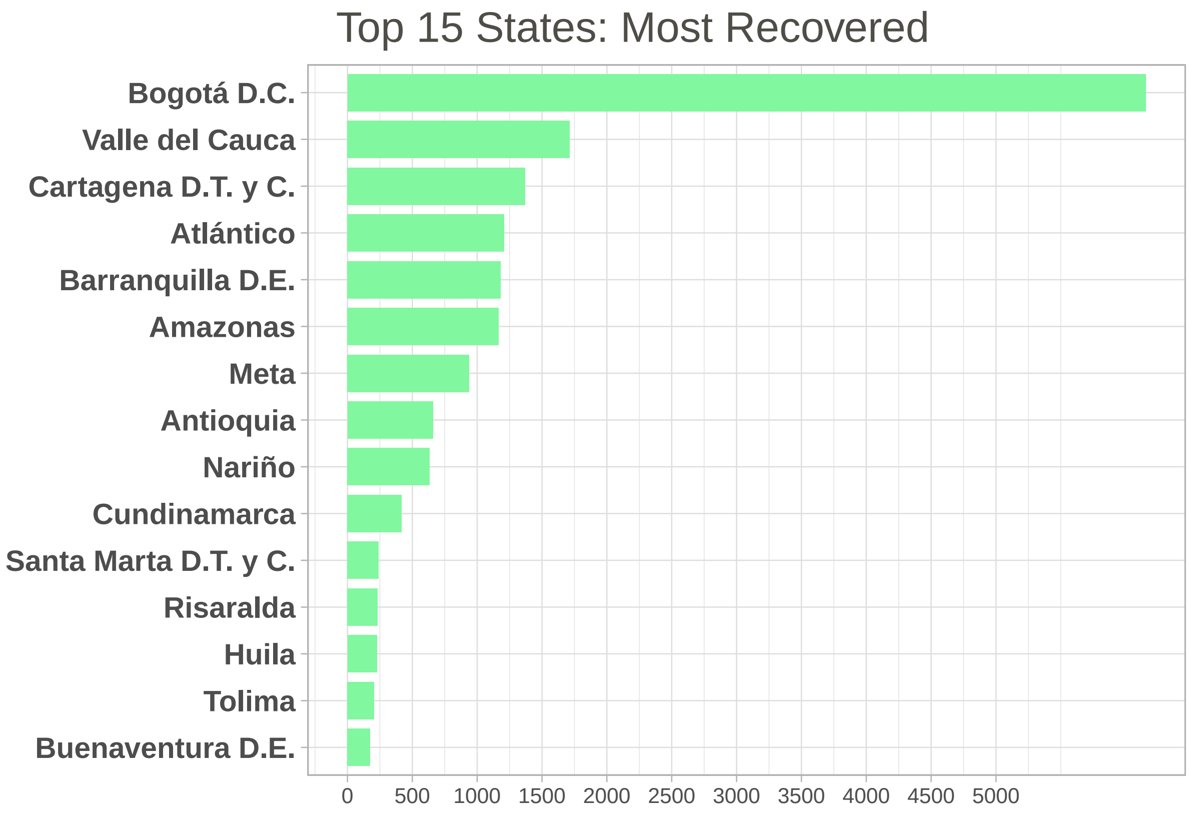


Top 15 states with most recovered patients.

```r
# Let's get the list of the 15 states with most recovered cases
totals_state %>% filter(outcome == "recovered") %>%
  arrange(-total) %>% head(15)
```

```
## # A tibble: 15 x 3
## # Groups:   state [15]
##    state                 outcome   total
##    <fct>                 <fct>     <int>
##  1 Bogotá D.C.           recovered  6155
##  2 Valle del Cauca       recovered  1715
##  3 Cartagena D.T. y C.   recovered  1371
##   4 Atlántico              recovered  1208
##   5 Barranquilla D.E.      recovered  1182
##   6 Amazonas               recovered  1167
##   7 Meta                   recovered   939
##   8 Antioquia              recovered   659
##   9 Nariño                 recovered   633
##  10 Cundinamarca           recovered   419
##  11 Santa Marta D.T. y C. recovered   240
##  12 Risaralda              recovered   233
##  13 Huila                  recovered   227
##  14 Tolima                 recovered   206
##  15 Buenaventura D.E.      recovered   176
```

```
totals_state %>% filter(outcome == "recovered") %>%
  arrange(-total) %>% head(15) %>%
  ggplot(aes(x = reorder(state,total), y = total )) +
  geom_bar(stat = "identity", fill  = colors$fill[colors$type == "recovered"], width = 0.8) +
  scale_y_continuous(breaks = seq(0, 5000, by = 500)) +
  coord_flip() +
  theme_light(base_size = 10) +
  labs(x = "", y = "", title = "Top 15 States: Most Recovered") +
  theme(axis.title = element_text(size = 14, colour = "black"),
        axis.text.y = element_text(size = 11, face = "bold"),
        plot.title = element_text(size= 16, hjust=0.1, color = "#4e4d47"))
```

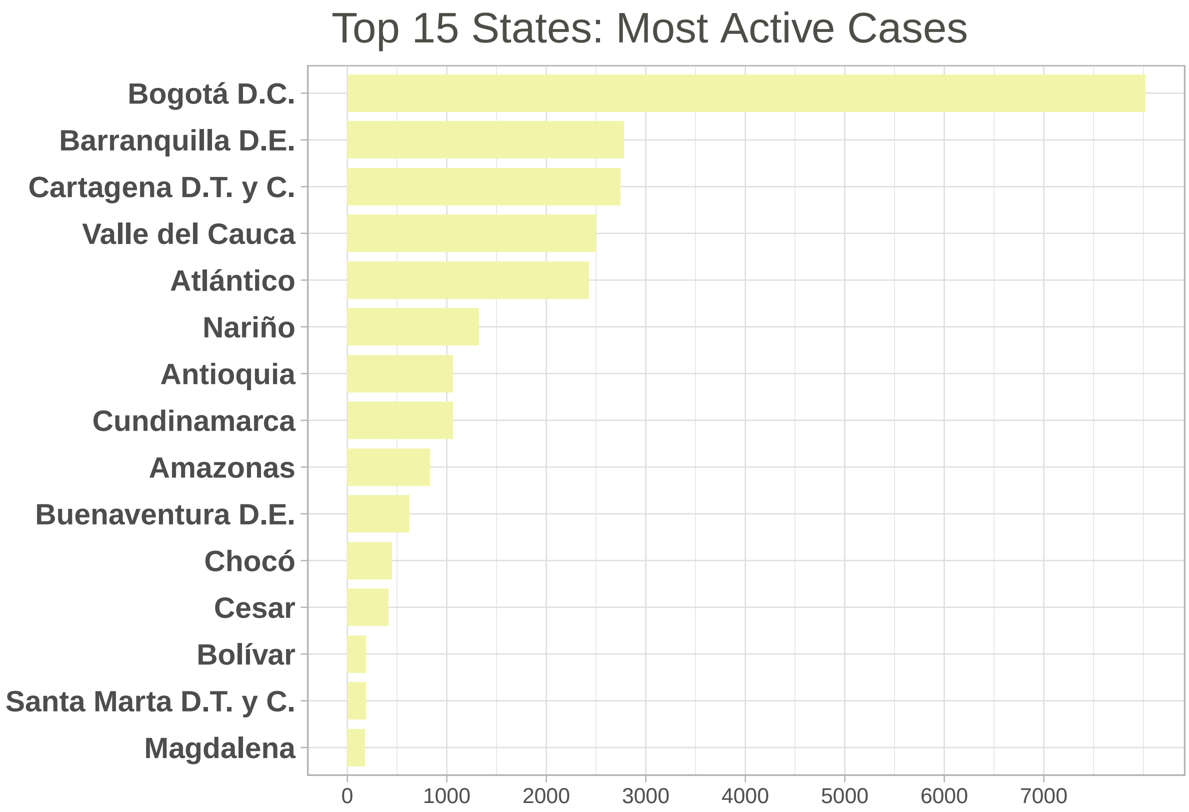


Top 15 states with most active cases.

```r
# Let's get the list of the 15 states with most active cases
totals_state %>% filter(outcome != "deceased" & outcome != "recovered" & outcome != "unknown") %>%
  group_by(state) %>% summarize(total_cases = sum(total)) %>%
  arrange(-total_cases) %>% head(15)
```

```
## # A tibble: 15 x 2
##    state                  total_cases
##    <fct>                        <int>
##  1 Bogotá D.C.                   8020
##  2 Barranquilla D.E.             2782
##  3 Cartagena D.T. y C.           2747
##  4 Valle del Cauca               2500
##  5 Atlántico                     2429
##  6 Nariño                        1325
##  7 Antioquia                     1062
##  8 Cundinamarca                  1059
##  9 Amazonas                       832
## 10 Buenaventura D.E.              622
## 11 Chocó                          450
## 12 Cesar                          414
## 13 Bolívar                        188
## 14 Santa Marta D.T. y C.          185
## 15 Magdalena                      179
```

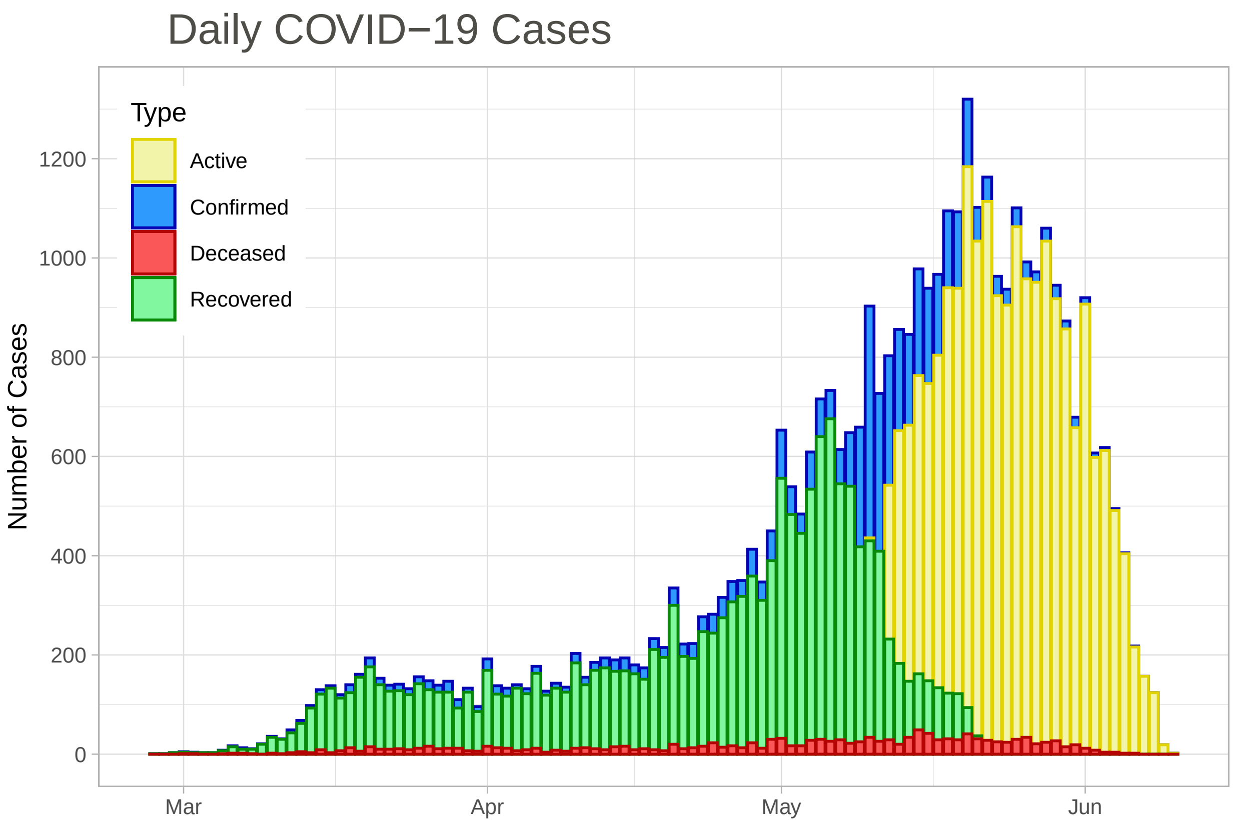
```r
totals_state %>% filter(outcome != "deceased" & outcome != "recovered" & outcome != "unknown") %>%
  group_by(state) %>% summarize(total_cases = sum(total)) %>%
  arrange(-total_cases) %>% head(15) %>%
  ggplot(aes(x = reorder(state,total_cases), y = total_cases )) +
  geom_bar(stat = "identity", fill  = colors$fill[colors$type == "active"], width = 0.8) +
  scale_y_continuous(breaks = seq(0, 7000, by = 1000)) +
  coord_flip() +
  theme_light(base_size = 10) +
  labs(x = "", y = "", title = "Top 15 States: Most Active Cases") +
  theme(axis.title = element_text(size = 14, colour = "black"),
        axis.text.y = element_text(size = 11, face = "bold"),
        plot.title = element_text(size= 16, hjust=0.1, color = "#4e4d47"))
```



From the above set of plots we can see that Bogota is the most impacted city in the country, with a huge difference against the others, this makes sense, if we think about this is the biggest city and with the highest population density.

Now let's look at how the disease has evolved in Colombia with 2 plots, the first one is a time plot showing the daily new cases by type.

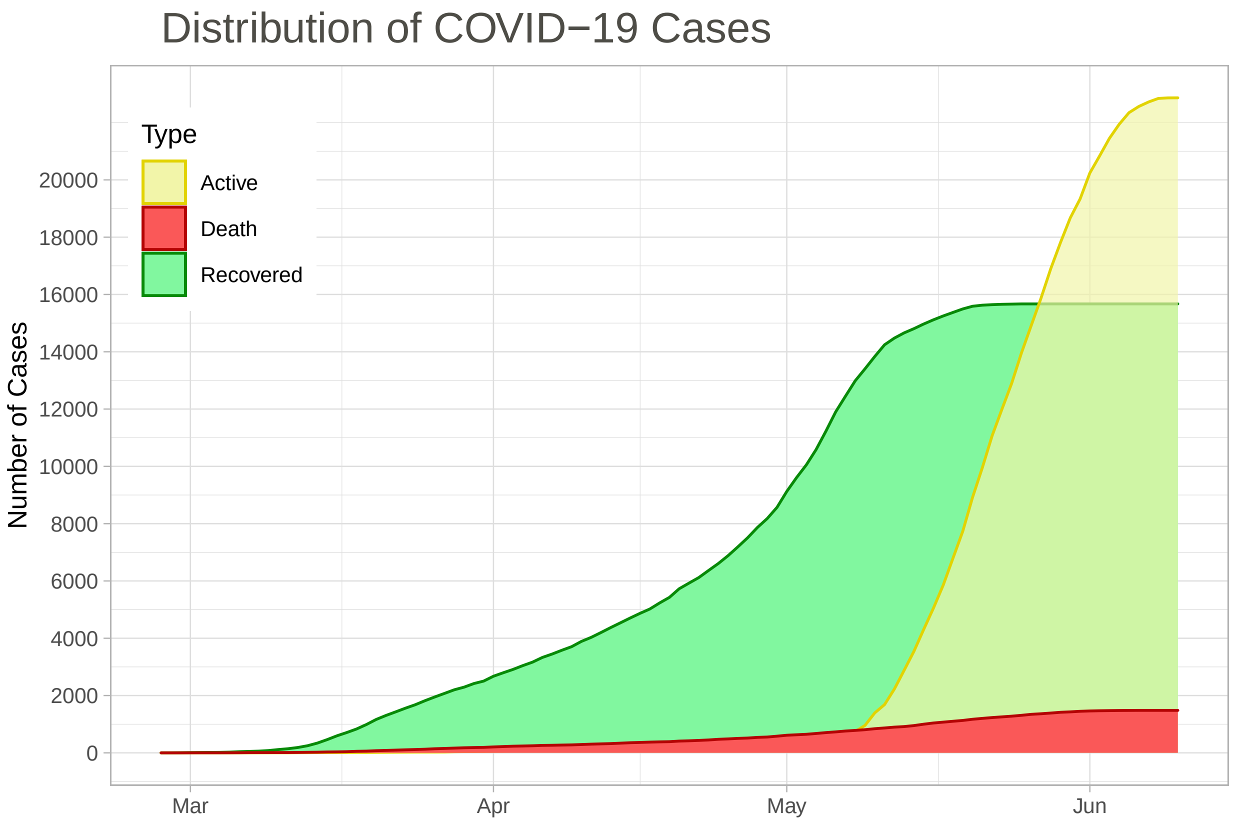
```
# Object with total cases by date
totals <- coronavirus %>% filter(!is.na(symptoms_date)) %>%
  group_by(outcome, symptoms_date) %>%
  summarise(total_cases = n()) %>%
  ungroup() %>%
  arrange(symptoms_date) %>%
  pivot_wider(names_from = outcome, values_from = total_cases,
              values_fill = list(total_cases = 0)) %>%
  mutate(confirmed = rowSums(.[,-1]), active = icu+hospitalized+outpatientCare)

# Daily COVID-19 cases
ggplot(data = totals, aes(x = symptoms_date)) +
  geom_bar(aes(y = confirmed, color = "confirmed_col", fill = "confirmed_col"),
           position = "identity", stat = "identity") +
  geom_bar(aes(y = active, color = "active_col", fill = "active_col"),
           position = "identity", stat = "identity") +
  geom_bar(aes(y = recovered, color = "recovered_col", fill = "recovered_col"),
           position = "identity", stat = "identity") +
  geom_bar(aes(y = deceased, color = "deceased_col", fill = "deceased_col"),
           position = "identity", stat = "identity") +
  theme_light(base_size = 10) +
  scale_y_continuous(breaks = seq(0, 1200, by = 200)) +
  scale_color_manual(name = "Type",
            values = c( "confirmed_col"=paste(colors$color[colors$type == "confirmed"], sep=""),
                        "active_col"=paste(colors$color[colors$type == "active"], sep=""),
                        "deceased_col"=paste(colors$color[colors$type == "deceased"], sep=""),
                        "recovered_col"=paste(colors$color[colors$type == "recovered"], sep="")),
            labels = c("Active", "Confirmed", "Deceased", "Recovered")) +
  scale_fill_manual(name = "Type",
            values = c( "confirmed_col"=paste(colors$fill[colors$type == "confirmed"], sep=""),
                        "active_col"=paste(colors$fill[colors$type == "active"], sep=""),
                        "deceased_col"=paste(colors$fill[colors$type == "deceased"], sep=""),
                        "recovered_col"=paste(colors$fill[colors$type == "recovered"], sep="")),
            labels = c("Active", "Confirmed", "Deceased", "Recovered")) +
  theme( #plot.margin = margin(0, 0, 0, 0, "pt"),
         legend.position = c(0.1, 0.8),
         plot.title = element_text(size= 16, hjust=0.1, color = "#4e4d47")
  ) +
  xlab("") +
  ylab("Number of Cases") +
  ggtitle("Daily COVID-19 Cases")
```



We can clearly see how at the beginning the confirmed cases and recovered cases number almost match, but from mid May is quite different, since we can see how active and confirmed cases are growing almost at the same pace, this is because of all the current cases that do not have an outcome yet, those are the cases we are going to attempt to predict.

The second plot is also a time plot that shows the cumulative number of cases per type.

```r
# Updating totals object with cumulative total cases by date
totals <- totals %>%
  mutate(active_total = cumsum(active),
         recovered_total = cumsum(recovered),
         deceased_total = cumsum(deceased))

# Distribution of COVID-19 Cases
ggplot(data = totals, aes(x = symptoms_date)) +
  geom_density(aes(y = recovered_total, color = "recovered_col", fill = "recovered_col"),
               position = "identity", stat = "identity") +
  geom_density(aes(y = active_total, color = "active_col", fill = "active_col"),
               position = "identity", stat = "identity", alpha=.7) +
  geom_density(aes(y = deceased_total, color = "deceased_col", fill = "deceased_col"),
               position = "identity", stat = "identity") +
  theme_light(base_size = 10) +
  scale_y_continuous(breaks = seq(0, 20000, by = 2000)) +
  scale_color_manual(name = "Type",
                values = c( "active_col"=paste(colors$color[colors$type == "active"], sep=""),
                            "deceased_col"=paste(colors$color[colors$type == "deceased"], sep=""),
                            "recovered_col"=paste(colors$color[colors$type == "recovered"], sep="")),
                labels = c("Active", "Death", "Recovered")) +
  scale_fill_manual(name = "Type",
                values = c( "active_col"=paste(colors$fill[colors$type == "active"], sep=""),
                            "deceased_col"=paste(colors$fill[colors$type == "deceased"], sep=""),
                            "recovered_col"=paste(colors$fill[colors$type == "recovered"], sep="")),
                labels = c("Active", "Death", "Recovered")) +
  theme(
    legend.position = c(0.1, 0.8),
    plot.title = element_text(size= 16, hjust=0.1, color = "#4e4d47")
  ) +
  xlab("") +
  ylab("Number of Cases") +
  ggtitle("Distribution of COVID-19 Cases")
```



In both plots we can see how the cases peak starts around mid April, also we can notice that the curve seems to be stable and not growing up, which is promising.

2.4 Features

Now we are going to analyze some of the features we have in the *coronavirus* dataset, we won't include all the features, only those we consider can be more important to predict.

2.4.1 Age

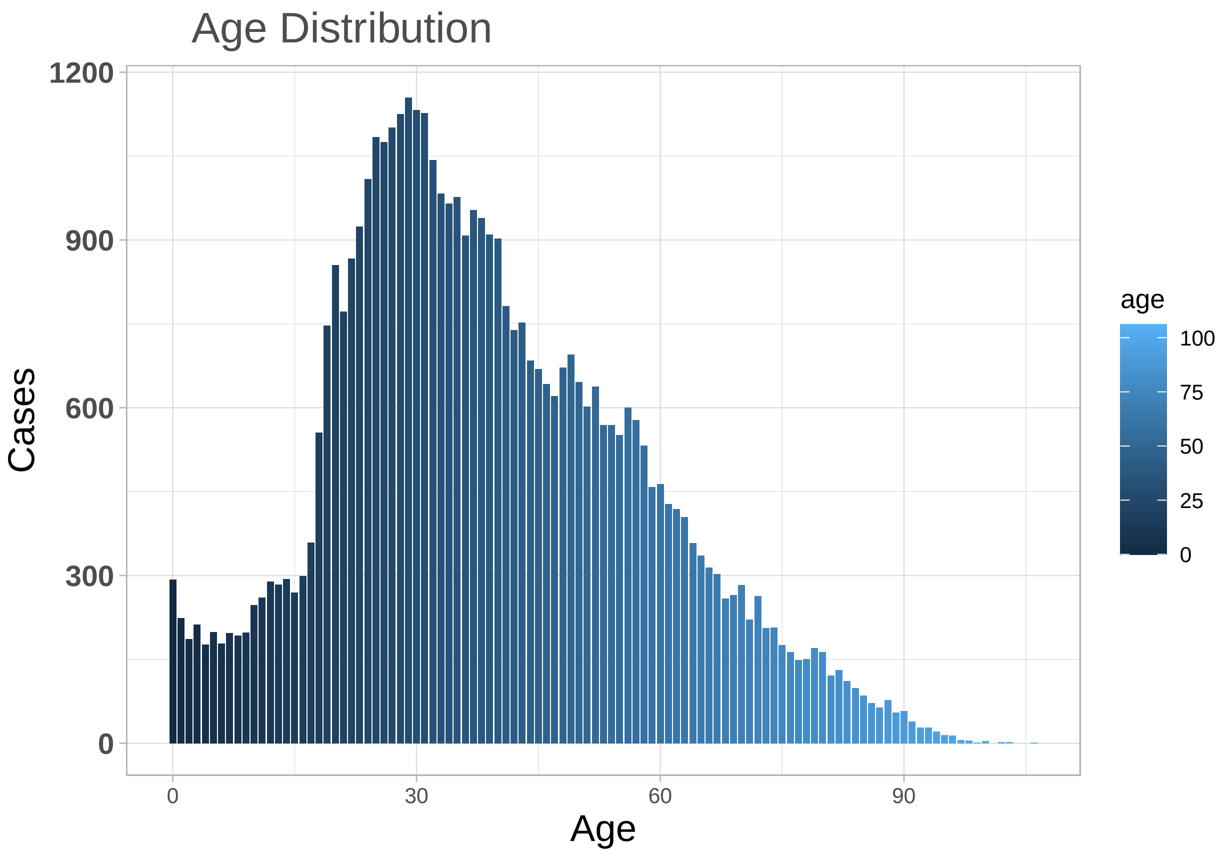
This is an important feature, knowing that COVID-19 has hit older adults harder than other age groups because they are more likely to already have underlying conditions such as cardiovascular disease, diabetes, or respiratory illness — comorbidities that we now know raise the risk of severe COVID-19 and COVID-19-related death. In addition, a likely weaker immune system makes it harder for older adults to fight off infection.

Let's take a look at how is the COVID-19 cases distribution by age.

```
# Set with total cases by age and type of outcome
totals_age <- coronavirus %>%
  group_by(outcome, age) %>%
  summarise(total_cases = n()) %>%
  ungroup() %>%
  arrange(age) %>%
  pivot_wider(names_from = outcome, values_from = total_cases,
              values_fill = list(total_cases = 0)) %>%
  mutate(confirmed = rowSums(.[,-1]), death_rate = (deceased*100)/confirmed)

# List of top 15 ages with more confirmed cases
totals_age %>% arrange(-confirmed) %>% head(15)
```

```
## # A tibble: 15 x 9
##       age deceased hospitalized   icu outpatientCare recovered unknown confirmed
##     <int>    <int>        <int> <int>          <int>     <int>   <int>     <dbl>
##  1    29        5           22     1            603       524       0      1155
##  2    30        4           27     4            610       486       1      1132
##  3    31        2           39     6            622       457       1      1127
##  4    28        1           23     3            636       461       1      1125
##  5    27        4           29     1            592       475       0      1101
##  6    25        3           27     2            592       460       0      1084
##  7    26        2           18     3            589       463       0      1075
##  8    32        2           36     7            565       433       0      1043
##  9    24        3           20     4            554       428       0      1009
## 10    33        6           32     1            537       407       0       983
## 11    35        4           33     3            540       396       1       977
## 12    34        5           30     4            514       412       0       965
## 13    37        9           31     4            517       392       1       954
## 14    38        4           23    10            503       398       1       939
## 15    23        1           11     3            531       378       0       924
## # ... with 1 more variable: death_rate <dbl>
```

```
# Age distribution
totals_age %>% arrange(-confirmed) %>%
  ggplot(aes(x = age, y = confirmed, fill=age)) +
  geom_bar(stat = "identity", width = 0.8) +
  theme_light(base_size = 10) +
  labs(x = "Age", y = "Cases", title = "Age Distribution") +
  theme(
    axis.title = element_text(size = 14, colour = "black"),
    axis.text.y = element_text(size = 11, face = "bold"),
    plot.title = element_text(size= 16, hjust=0.1, color = "#4e4d47"))
```

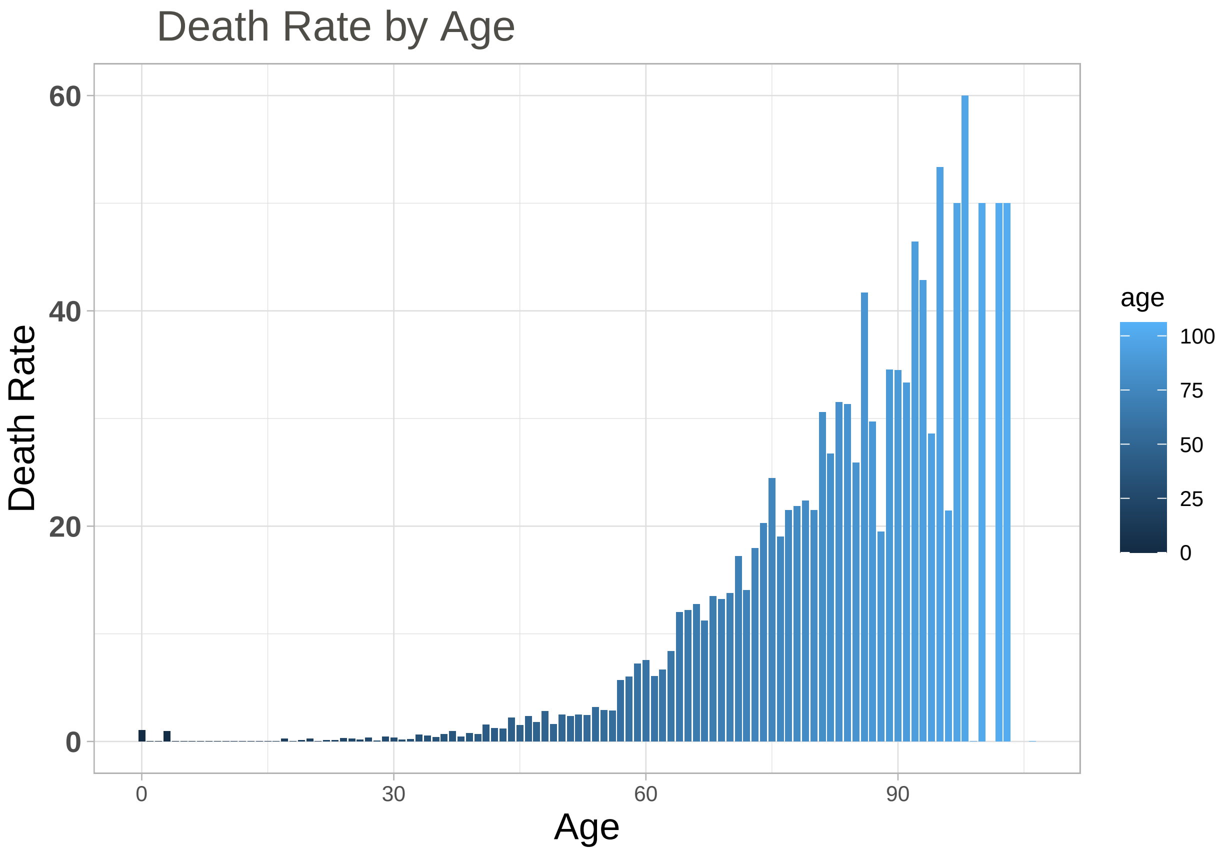


We can see that the age range with more cases are between 25 and 40 years, but are these also the most impacted by deaths? Let's check.

```
# Top 15 ages by death rate
totals_age %>% arrange(-death_rate) %>% head(15)
```

```
## # A tibble: 15 x 9
##      age deceased hospitalized   icu outpatientCare recovered unknown confirmed
##    <int>    <int>        <int> <int>          <int>     <int>   <int>     <dbl>
##  1    98        3            0     0              1         1       0         5
##  2    95        8            0     0              4         2       1        15
##  3    97        3            1     0              2         0       0         6
##  4   100        2            0     1              1         0       0         4
##  5   102        1            0     0              1         0       0         2
##  6   103        1            0     0              0         1       0         2
##  7    92       13            5     0              6         4       0        28
##  8    93       12            0     0              8         8       0        28
##  9    86       30           12     4             13        11       2        72
## 10    89       19            9     0             15        10       2        55
## 11    90       20            6     0             14        16       2        58
## 12    91       13            6     0              9         9       2        39
## 13    83       35           18     5             29        23       1       111
## 14    84       31           12     3             21        32       0        99
## 15    81       37           13     4             38        28       1       121
## # ... with 1 more variable: death_rate <dbl>
```

```
# Death rate by age
totals_age %>% arrange(-death_rate) %>%
  ggplot(aes(x = age, y = death_rate, fill=age)) +
  geom_bar(stat = "identity", width = 0.8) +
  theme_light(base_size = 10) +
  labs(x = "Age", y = "Death Rate", title = "Death Rate by Age") +
  theme(
    axis.title = element_text(size = 14, colour = "black"),
    axis.text.y = element_text(size = 11, face = "bold"),
    plot.title = element_text(size= 16, hjust=0.1, color = "#4e4d47"))
```



We can clearly see how the death rate is greater for older people, so answering the above question: no, the 25-40 age range is not also the most impacted by deaths.

After analyzing the age data we can infer that this is an important feature to predict COVID-19 outcome.

2.4.2 Gender

We have heard on the news that the novel coronavirus, COVID-19, tends to affect men more severely than it does women. Though nobody can yet explain the oddity, potential reasons run the gamut from biology to bad habits.

The World Health Organization (WHO) has reported that around 60% of deaths related to COVID-19 in Europe have been among men.

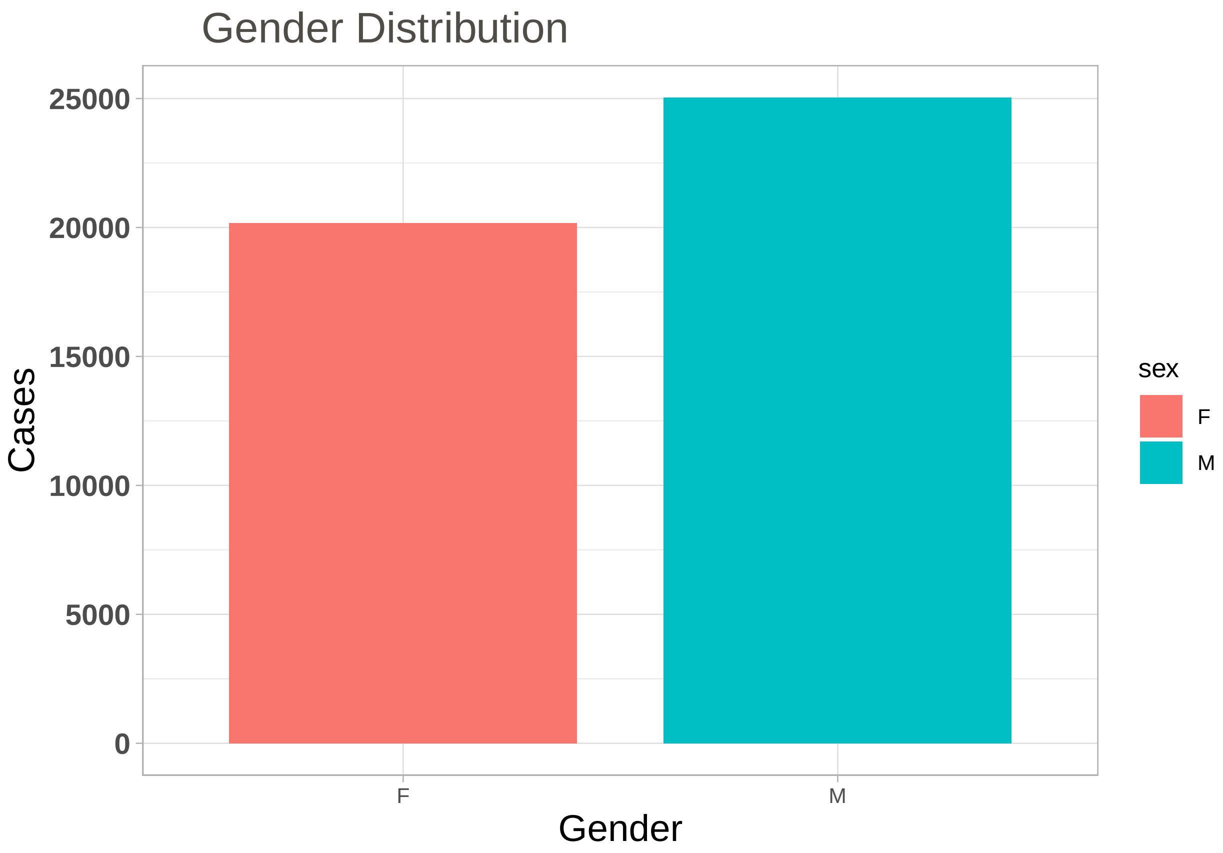
Some of the underlying reasons why COVID-19 may be more deadly for men than women may include the fact that heart disease is more common in elderly men than in elderly women, Dr. Stephen Berger. Genetics may also play a big role, Berger said, Women, because of their extra X chromosome, have a stronger immune system and response to infections than men. Berger also said that it's possible that men are more at risk because they tend to expose themselves more to larger crowds and social exchanges, including things like handshaking and sporting events

Let's now explore the Colombian situation and see whether men have been more impacted or not.

```r
# Object with total cases by sex and type of outcome
totals_sex <- coronavirus %>%
  group_by(outcome, sex) %>%
  summarise(total_cases = n()) %>%
  ungroup() %>%
  arrange(sex) %>%
  pivot_wider(names_from = outcome, values_from = total_cases,
              values_fill = list(total_cases = 0)) %>%
  mutate(confirmed = rowSums(.[,-1]), death_rate = (deceased*100)/confirmed)

# Confirmed cases by gender
totals_sex %>% arrange(-confirmed)
```

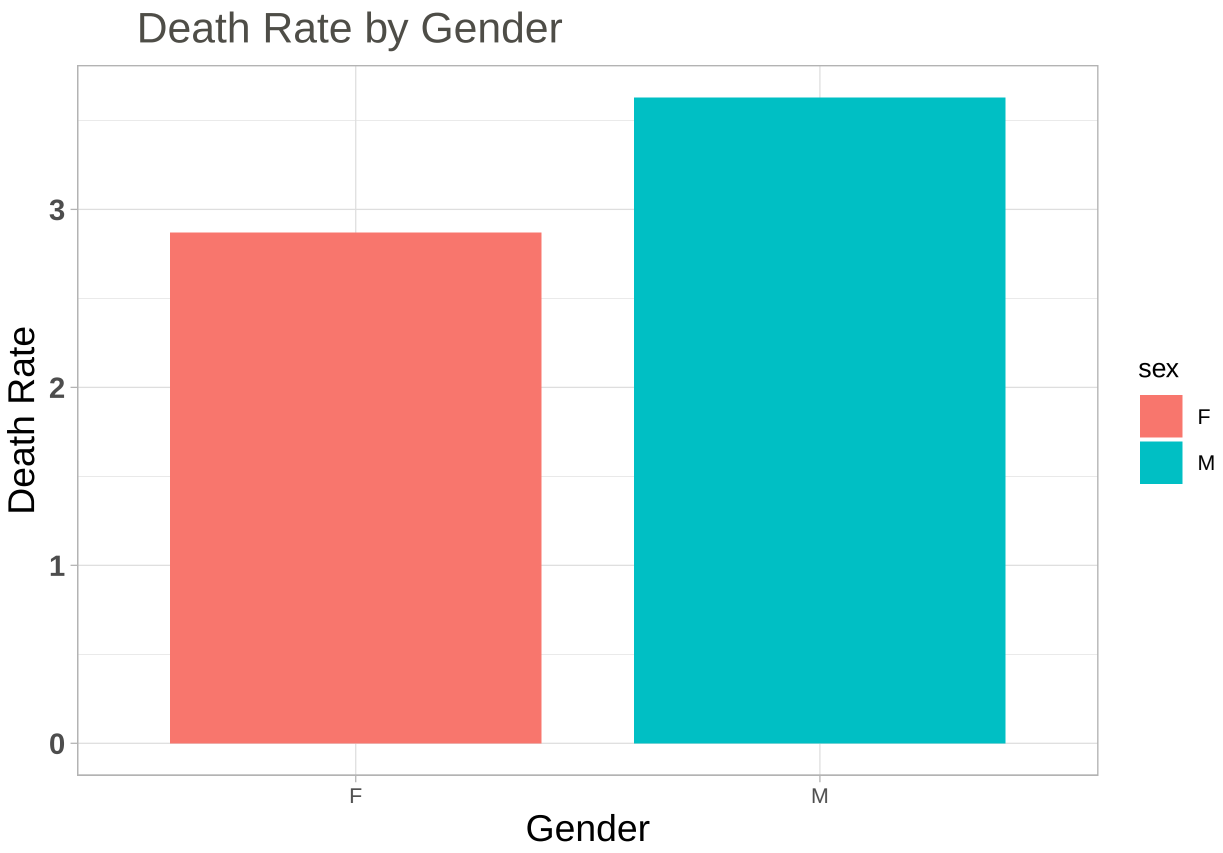
```
## # A tibble: 2 x 9
##   sex   deceased hospitalized   icu outpatientCare recovered unknown confirmed
##   <fct>    <int>        <int> <int>          <int>     <int>   <int>     <dbl>
## 1 M          909         1338   312          12724      9719      43     25045
## 2 F          579          973   165          10348      8071      31     20167
## # ... with 1 more variable: death_rate <dbl>
```

```r
# Gender distribution
totals_sex %>% arrange(-confirmed) %>%
  ggplot(aes(x = sex, y = confirmed, fill=sex)) +
  geom_bar(stat = "identity", width = 0.8) +
  theme_light(base_size = 10) +
  labs(x = "Gender", y = "Cases", title = "Gender Distribution") +
  theme(
    axis.title = element_text(size = 14, colour = "black"),
    axis.text.y = element_text(size = 11, face = "bold"),
    plot.title = element_text(size= 16, hjust=0.1, color = "#4e4d47"))
```



From the data and the plot we can see that men have about of 55.39% of the confirmed cases, but let's check the death rate and see how are men impacted.

```
# Death rate by sex
totals_sex %>% arrange(-death_rate) %>%
  ggplot(aes(x = sex, y = death_rate, fill=sex)) +
  geom_bar(stat = "identity", width = 0.8) +
  theme_light(base_size = 10) +
  labs(x = "Gender", y = "Death Rate", title = "Death Rate by Gender") +
  theme(
    axis.title = element_text(size = 14, colour = "black"),
    axis.text.y = element_text(size = 11, face = "bold"),
    plot.title = element_text(size= 16, hjust=0.1, color = "#4e4d47"))
```



This data confirms the world's trend of men being more affected by COVID-19, with a 3.63% of death rate for men and 2.87% for women.

3 Data Pre-Processing

We will predict if a person infected by COVID-19 in Colombia will die or live, using the features from the *coronavirus* dataset like age, sex, contagion type and more, and we will include others features like number of days until the patient gets an outcome and number of days from the symptoms date until confirmation date from lab analysis (diagnosis date), these two we will calculate using the data in the *coronavirus* dataset.

3.1 Features Creation

3.1.1 Outcome Time

We will include the time, in days, it takes a patient to get an outcome from the COVID-19 disease, whether died or recovered.

Knowing that recovering from COVID-19 may take up to 14 days, we think this is an important feature, since the number of days may indicate if the patient had any complication that took more than expected to get recovered, taking him under ICU or hospitalized for a period of time.

Let's add the feature and explore the data.

```
# Include outcome_time feature with number of days
# until patient gets an outcome
coronavirus <- coronavirus %>%
  mutate(outcome_time = ifelse(!is.na(date_of_death), difftime(date_of_death,
                                    symptoms_date, units="days"),
                            ifelse(!is.na(recovery_date), difftime(recovery_date,
                                          symptoms_date, units="days"),NA)))

# Let's get the number of deceased and recovered with outcome_time
# greater and smaller than 14
cat("Number of deceased with outcome_time greater than 14: ",
    coronavirus %>% filter(outcome_time > 14 & outcome == "deceased") %>% nrow())
```

```
## Number of deceased with outcome_time greater than 14:  560
```

```
cat("Number of recovered with outcome_time greater than 14: ",
    coronavirus %>% filter(outcome_time > 14 & outcome == "recovered") %>% nrow())
```

```
## Number of recovered with outcome_time greater than 14:  14513
```

```
cat("Number of deceased with outcome_time less than 14: ",
    coronavirus %>% filter(outcome_time < 14 & outcome == "deceased") %>% nrow())
```

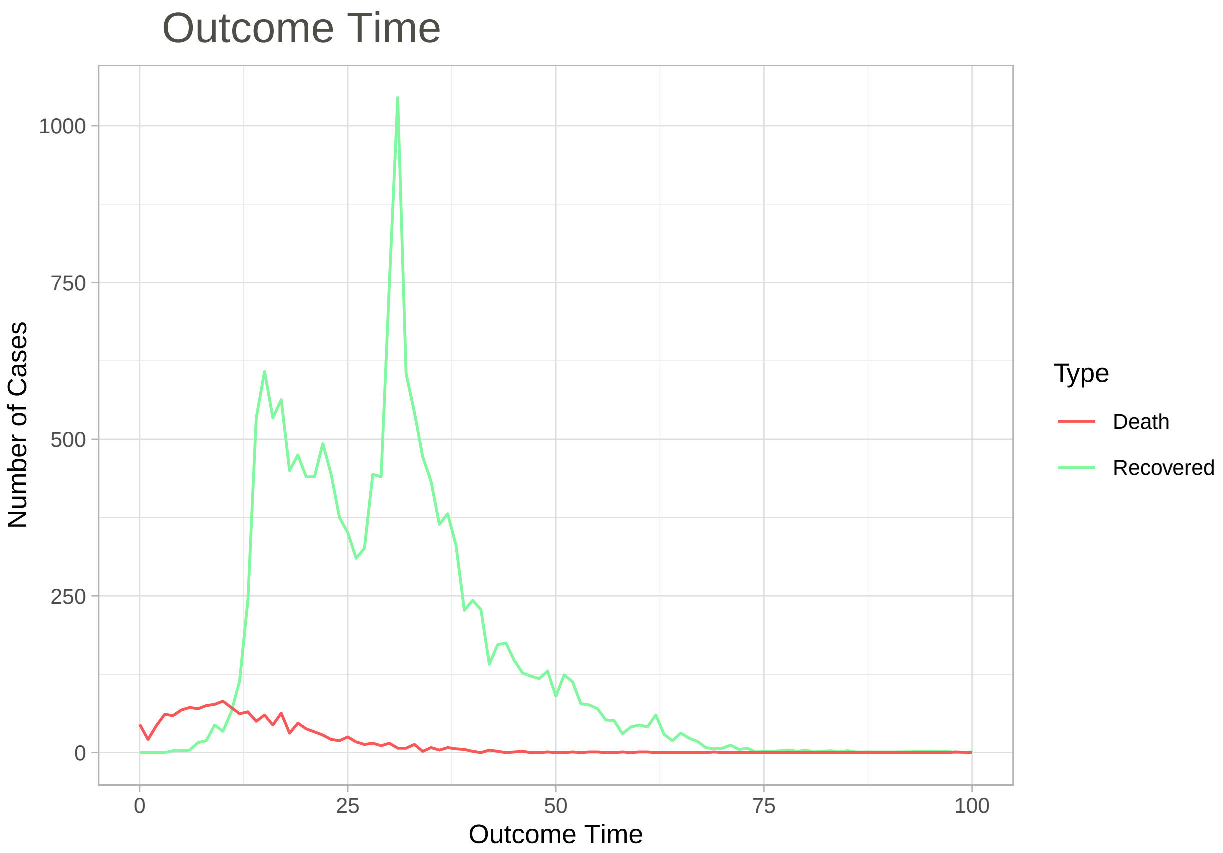
```
## Number of deceased with outcome_time less than 14:  872
```

```
cat("Number of recovered with outcome_time less than 14: ",
    coronavirus %>% filter(outcome_time < 14 & outcome == "recovered") %>% nrow())
```

```
## Number of recovered with outcome_time less than 14:  546
```

From the data above we can see that there is more deaths when the outcome time is less than 14 days, also we can see that a lot more people get recovered when outcome time is greater than 14 days. Let's visualize this in the next plot.

```r
# Let's visualize the outcome_time feature with this plot
coronavirus %>% filter((outcome=="recovered" | outcome=="deceased") & !is.na(outcome_time)) %>%
  group_by(outcome, outcome_time) %>%
  summarise(total_cases = n()) %>%
  ungroup() %>%
  arrange(outcome_time) %>%
  pivot_wider(names_from = outcome, values_from = total_cases,
              values_fill = list(total_cases = 0)) %>%
  ggplot(aes(x = outcome_time)) +
  geom_line(aes(y = recovered, color = "recovered_col"),
               position = "identity", stat = "identity") +
  geom_line(aes(y = deceased, color = "deceased_col"),
               position = "identity", stat = "identity") +
  theme_light(base_size = 10) +
  scale_color_manual(name = "Type",
                     values = c( "deceased_col"="#FA5858",
                                 "recovered_col"="#81F79F"),
                     labels = c("Death", "Recovered")) +
  theme(
    plot.title = element_text(size= 16, hjust=0.1, color = "#4e4d47")
  ) +
  xlab("Outcome Time") +
  ylab("Number of Cases") +
  ggtitle("Outcome Time")
```



With this plot we have confirmed that most of the deaths happen in the first 10 days of the disease, while most of the recovered are between 12 and 35 days after the symptoms date.

3.1.2 Diagnosis Time

Now let's create one more feature which contains the time, in days, it takes a patient to get lab test results taking as start date, the symptoms date.

```
# Include diagnosis_time feature with number of days
# until patient gets the lab test results
coronavirus <- coronavirus %>%
  mutate(diagnosis_time = ifelse(!is.na(diagnosis_date),
                                 difftime(diagnosis_date, symptoms_date, units="days"),NA))
```

Let's analyze this feature by getting the top 6 cases ordered by recovered or deceased.

```
# Subset with recovered and deceased cases
# grouped by outcome and diagnosis_time
totals_diagnosis_time <- coronavirus %>%
  filter((outcome=="recovered" | outcome=="deceased") & !is.na(diagnosis_time)) %>%
  group_by(outcome, diagnosis_time) %>%
  summarise(total_cases = n()) %>%
  ungroup() %>%
  arrange(diagnosis_time) %>%
  pivot_wider(names_from = outcome, values_from = total_cases,
               values_fill = list(total_cases = 0))

# Top 6 diagnosis_time by recovered
top_recovered <- totals_diagnosis_time %>% arrange(-recovered) %>% head()
top_recovered
```

```
## # A tibble: 6 x 3
##   diagnosis_time recovered deceased
##            <dbl>     <int>    <int>
## 1              6      1261       87
## 2              8      1257      105
## 3              7      1224      108
## 4              5      1166       82
## 5              9      1125      103
## 6             10      1046      118
```

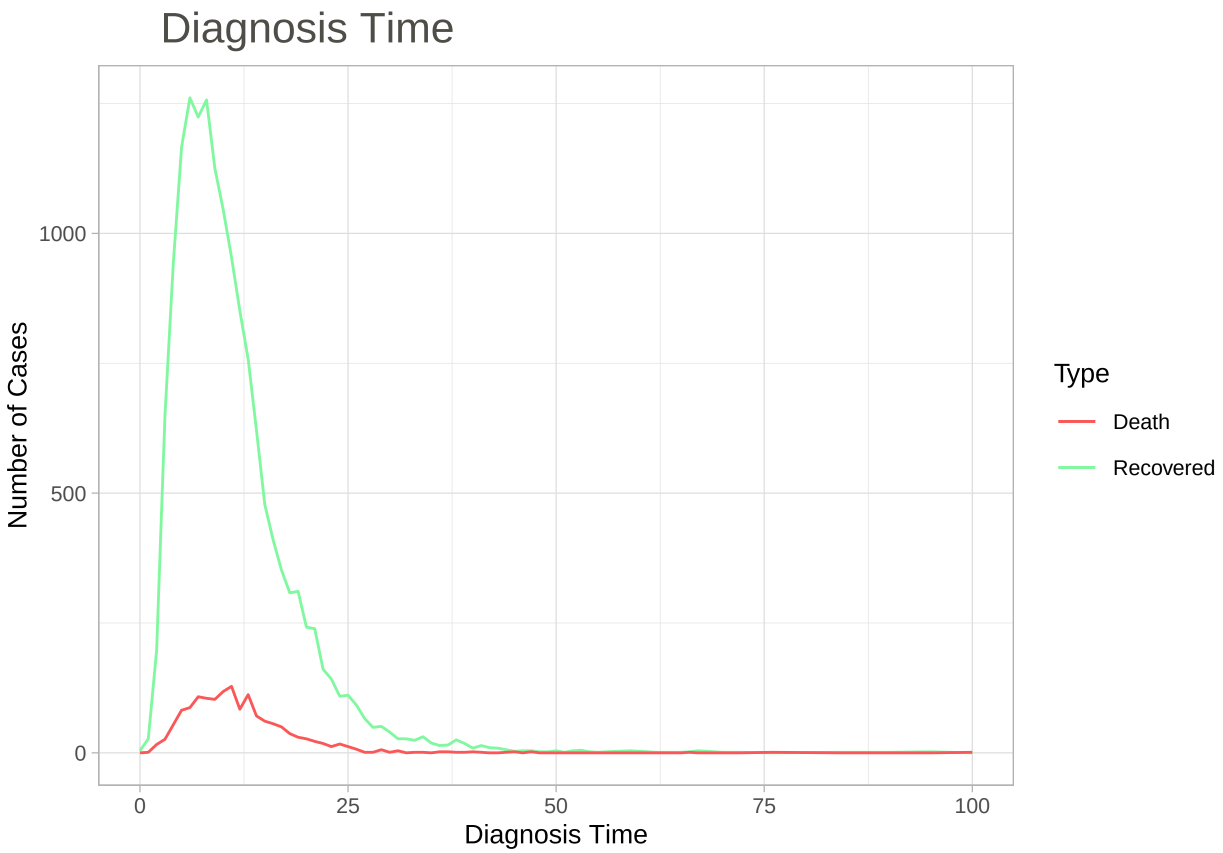
```
# Top 6 diagnosis_time by deceased
top_deceased <- totals_diagnosis_time %>% arrange(-deceased) %>% head()
top_deceased
```

```
## # A tibble: 6 x 3
##   diagnosis_time recovered deceased
##            <dbl>     <int>    <int>
## 1             11       954      128
## 2             10      1046      118
## 3             13       758      112
## 4              7      1224      108
## 5              8      1257      105
## 6              9      1125      103
```

From the above data we can see that most recovered cases are between 5 and 10 days, and for deceased cases are between 7 and 13 days, there is a few days difference between recovered and death cases, this can help us in our predictions.

Let's plot this feature.

```
# Let's visualize the diagnosis_time feature with this plot
ggplot(data = totals_diagnosis_time, aes(x = diagnosis_time)) +
  geom_line(aes(y = recovered, color = "recovered_col"),
            position = "identity", stat = "identity") +
  geom_line(aes(y = deceased, color = "deceased_col"),
            position = "identity", stat = "identity") +
  theme_light(base_size = 10) +
  scale_color_manual(name = "Type",
                     values = c( "deceased_col"="#FA5858",
                                 "recovered_col"="#81F79F"),
                     labels = c("Death", "Recovered")) +
  theme(
    plot.title = element_text(size= 16, hjust=0.1, color = "#4e4d47")
  ) +
  xlab("Diagnosis Time") +
  ylab("Number of Cases") +
  ggtitle("Diagnosis Time")
```



```r
# Cleaning objects we won't use
rm(totals_diagnosis_time, top_recovered, top_deceased)
```

3.2 Data Preparation

3.2.1 Data Cleaning

Here we are going to prepare our dataset to start modeling, first we are going to remove the columns we won't use to predict because they do not represent important data, like, *municipality_code*, *record_date*, *web_date*, etc.

```r
# Select only the columns we are interested in,
# the columns we will use for predicting
coronavirus <- coronavirus %>%
  select(id, state, age, sex, contagion_type, symptoms_date, diagnosis_date,
         diagnosis_time, outcome_time, outcome)
```

Next, we will change the *outcome* level values to make NA those statuses where a patient is still an active case, with no definitive outcome yet. For example if outcome value is "icu", this means the patient is still under care because of COVID-19. So, basically we will consider "icu", "hospitalized" and "outpatientCare" as NA and only keep as possible outcomes "recovered" or "deceased".

```r
# Mutate outcome to change to NA those cases we don't know the outcome yet
coronavirus[,'outcome'] <- ifelse(coronavirus[,'outcome'] == "recovered", "recovered",
                              ifelse(coronavirus[,'outcome'] == "deceased", "deceased",
                                ifelse(coronavirus[,'outcome'] == "icu", NA,
                                  ifelse(coronavirus[,'outcome'] == "hospitalized", NA,
                                    ifelse(coronavirus[,'outcome'] == "outpatientCare", NA, NA)))))

# Factorizing outcome to the 2 only possible values
coronavirus[,'outcome'] <- factor(coronavirus$outcome, levels = c("recovered","deceased"))

# Let's check the dataset dimension and structure
# before any cleanup
str(coronavirus)
```

```
## 'data.frame':    45212 obs. of  10 variables:
##  $ id            : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ state         : Factor w/ 37 levels "Amazonas","Antioquia",..: 7 35 2 2 2 2 13 7 7 7 ...
##  $ age           : int  19 34 50 55 25 27 85 22 28 36 ...
##  $ sex           : Factor w/ 2 levels "F","M": 1 2 1 2 2 1 1 1 1 1 ...
##  $ contagion_type: Factor w/ 2 levels "contact","travel": 2 2 2 1 1 1 2 2 2 2 ...
##  $ symptoms_date : Date, format: "2020-02-27" "2020-03-04" ...
##  $ diagnosis_date: Date, format: "2020-03-06" "2020-03-09" ...
##  $ diagnosis_time: num  8 5 9 5 3 5 9 5 4 6 ...
##  $ outcome_time  : num  15 15 15 20 15 20 15 15 16 15 ...
##  $ outcome       : Factor w/ 2 levels "recovered","deceased": 1 1 1 1 1 1 1 1 1 1 ...
```

```
dim(coronavirus)
```

```
## [1] 45212    10
```

Now, we will remove all the rows containing NAs in all the columns but *outcome_time* and *outcome*, as we will try to predict the outcome for all this NAs.

```
# Number of rows before cleaning
dim <- dim(coronavirus)[1]

# Remove NAs from all columns but outcome_time and outcome,
# since we will keep them to predict
coronavirus <- coronavirus[complete.cases(coronavirus[ , c(1:8)]),]

dim(coronavirus)
```

```
## [1] 39768    10
```

```
cat("Removed rows after cleaning NAs: ",
    dim - dim(coronavirus)[1])
```

```
## Removed rows after cleaning NAs:  5444
```

```
# Number of rows after cleaning NAs
dim <- dim(coronavirus)[1]

# Filter out rows with and outcome and no outcome_time nor diagnosis_time
```

```
coronavirus <- coronavirus %>%
  filter((!is.na(outcome) & !is.na(outcome_time) & !is.na(diagnosis_time)) |
           is.na(outcome))

dim(coronavirus)
```

```
## [1] 39689    10
```

```
cat("Removed rows after filtering outcome_time NAs: ",
    dim - dim(coronavirus)[1])
```

```
## Removed rows after filtering outcome_time NAs:  79
```

```
# Cleaning objects we won't use
rm(dim)
```

3.2.2 Train and Test Sets

First, we are going to split the *coronavirus* dataset in two, *training* and *prediction* sets, where the *prediction* set will contain all the active cases (outcome with NA value) and we will use it for our final prediction, the *training* set will be all cases with a known outcome, and will be used for modeling and choosing the best algorithm which we will apply to the *prediction* set in the final prediction.

```
# training & prediction sets
# Predict outcome of Active cases is.na(outcome) prediction set
training <- coronavirus %>% filter(!is.na(outcome))
prediction <- coronavirus %>% filter(is.na(outcome))

dim(training)
```

```
## [1] 16945    10
```

```
dim(prediction)
```

```
## [1] 22744    10
```

Now, we will randomly split the *training* set in two, *train_set* and *test_set* sets, which we will use for training the models and evaluate them. The train set will be 80% of *training* data and the test set will be the remaining 20%.

```
# train and test sets
# train 80% of training set
# test 20% of training set
set.seed(19, sample.kind="Rounding")
index <- createDataPartition(training$outcome, times = 1, p = 0.2, list=FALSE)
train_set <- training[-index,]
test_set  <- training[index,]

# Cleaning objects we won't use
rm(index)
```

Finally we are going to prepare our train and test sets, by removing columns we won't use for modeling, but are useful to keep for reference in the final result, these columns are: *id*, *symptoms_date* and *diagnose_date*, we replaced the dates with *outcome_time* and *diagnosis_time*.

```r
# Remove symptoms_date and diagnosis_date
# we won't use the dates for prediction,
# we use instead outcome_time and diagnosis_time
train_set <- train_set %>% select(-id,-symptoms_date,-diagnosis_date)

test_set <- test_set %>% select(-id,-symptoms_date,-diagnosis_date)

dim(train_set)
```

```
## [1] 13555     7
```

```r
dim(test_set)
```

```
## [1] 3390    7
```

4 Modeling

For modeling we will use the *caret* package, caret is short for Classification And REgression Training, and is a comprehensive framework for building machine learning models in R, *caret* helps to find the optimal model in the shortest possible time.

It integrates all activities related to model development in a streamlined workflow. For nearly every major machine learning algorithm available in R.

With R having so many implementations of machine learning algorithms, it can be challenging to keep track of which algorithm resides in which package. Thanks to *caret* no matter which package the algorithm resides, caret will remember that for you and it will just prompt you to run install.package for that particular algorithm's package.

For training we will use the caret *train* function, which lets us train different algorithms using similar syntax.

For evaluating the algorithms we will use the Confusion Matrix, which tabulates each combination of prediction and actual value, it is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. It allows the visualization of the performance of an algorithm. From the confusion matrix we will compare the following values to select the best model:

- Sensitivity: also known as the true positive rate or recall, is the proportion of actual positive outcomes correctly identified as such.
- Specificity: also known as the true negative rate, is the proportion of actual negative outcomes that are correctly identified as such.
- Overall Accuracy: the proportion of cases that were correctly predicted in the test set.

We will also examine the variables importance of each model. To define a variable importance we count how often a predictor is used. The caret package includes the function *varImp* that extracts variable importance from any model in which the calculation is implemented.

4.1 Logistic Regression

Linear regression is a model that assumes a linear relationship between the input variables (x) and the single output variable (y). More specifically, that y can be calculated from a linear combination of the input variables (x). Linear Regression serves as a baseline approach: if you can't beat it with a more complex approach, you probably want to stick to linear regression.

Logistic regression is useful when you are predicting a binary outcome from a set of continuous predictor variables. It differs from linear regression model because it only accepts dichotomous (binary) input as a dependent variable (i.e., a vector of 0 and 1).

In R, we can fit the logistic regression model with the function *glm*: generalized linear models. This function is more general than logistic regression so we need to specify the model we want through the family parameter.

Now, let's apply the logistic regression model to our data.

```r
# Change outcome to binary, 0 and 1
train_set_glm <- train_set %>% mutate(outcome = as.numeric(outcome == "recovered"))

# Fit logistic regression model
train_glm <- glm(outcome ~ ., data = train_set_glm, family = "binomial")

# Variable of importance
imp <- as.data.frame(varImp(train_glm))
imp <- data.frame(overall = imp$Overall,
                   names    = rownames(imp))
imp %>% arrange(-overall) %>% head()
```

```
##     overall           names
## 1 31.840981    outcome_time
## 2 27.641583             age
## 3 21.317045  diagnosis_time
## 4  6.003628            sexM
## 5  4.759431  stateAntioquia
## 6  4.741916      stateHuila
```

```
p_hat <- predict(train_glm, test_set)
y_hat <- factor(ifelse(p_hat > 0.5, "recovered", "deceased")) %>%
  factor(levels = c("recovered","deceased"))

# Confusion Matrix
cm <- confusionMatrix(y_hat, test_set$outcome)

# Results
cm_results <- bind_rows(tibble(Model = "Logistic Regression",
                                  Accuracy = cm$overall["Accuracy"],
                                  Sensitivity = cm$byClass["Sensitivity"],
                                  Specificity = cm$byClass["Specificity"]))
# Print the results
cm_results
```

```
## # A tibble: 1 x 4
##   Model               Accuracy Sensitivity Specificity
##   <chr>                  <dbl>       <dbl>       <dbl>
## 1 Logistic Regression    0.965       0.983       0.780
```

```
# Cleaning objects we won't use
rm(train_set_glm, p_hat)
```

We got an accuracy of 0.965 which is good, let's compare against other models and see if we can do better. The most important variables in the model were *outcome_time*, *age* and *diagnosis_time*.

4.2 K-nearest neighbor

K-nearest neighbor (KNN) is a machine learning algorithm that classifies a new data point into the target class, depending on the features of its neighboring data points, so basically it is mainly based on feature similarity. KNN checks how similar a data point is to its neighbor and classifies the data point into the class it is most similar to.

The algorithm calculates the euclidean distance of all predictors, then for any point $(x1, .., xp)$ in the multi-dimensional space that we want to predict, the algorithm determines the distance to k points. The k nearest points is refereed as neighborhood.

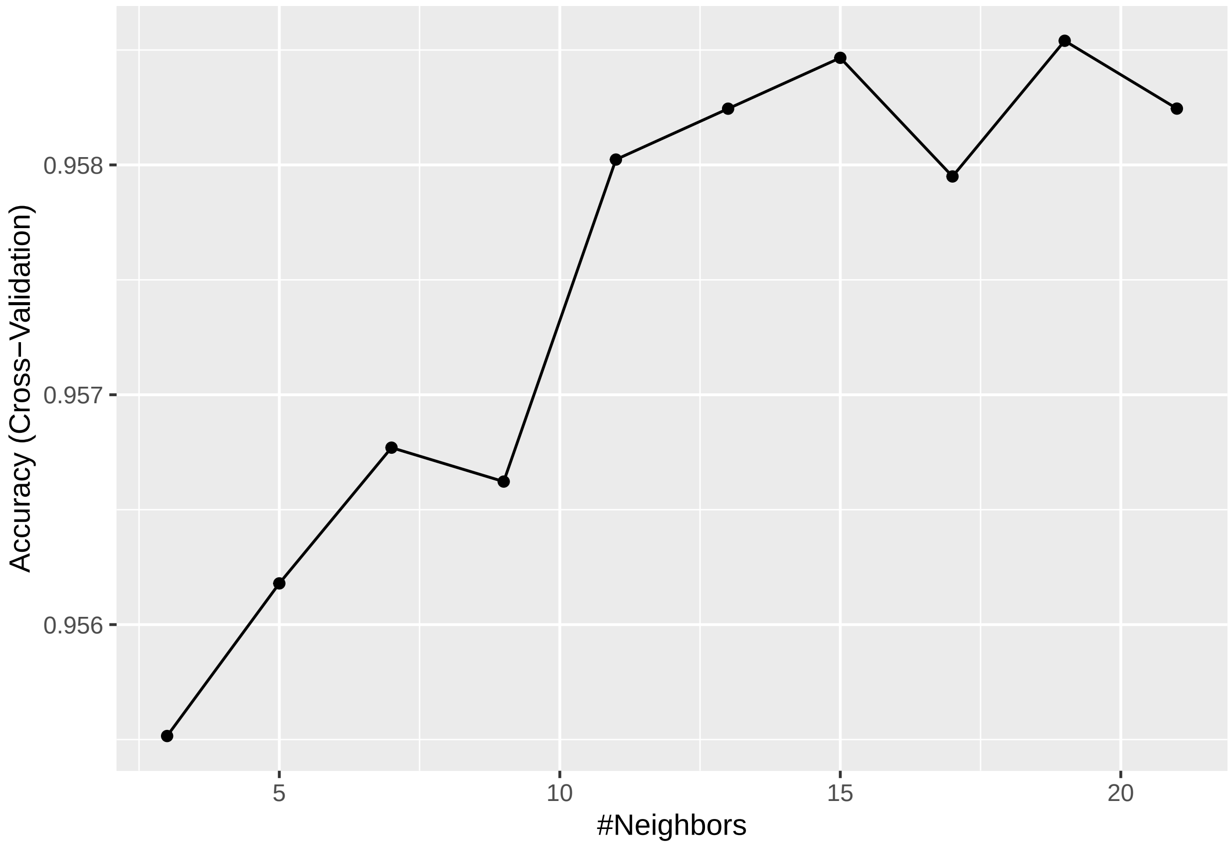
For $k = 1$ the algorithm finds the distance to a single neighbor, k is a tuning parameter that can be calculated running the algorithm for several values of k and picking the result with highest accuracy.

Let's fit our model and compare the results.

```
set.seed(19, sample.kind="Rounding")

# Fit knn model
control <- trainControl(method = "cv", number = 10, p = .9)
train_knn <- train(outcome ~ .,
                   data = train_set,
                   method = "knn",
                   tuneGrid = data.frame(k = seq(3, 21, 2)),
                   trControl = control)

# Final value of k used in the model
train_knn$bestTune
```

```
##    k
## 9 19
```

```
ggplot(train_knn)
```



```
y_hat <- predict(train_knn, test_set, type="raw")

# Confusion Matrix
cm <- confusionMatrix(y_hat, test_set$outcome)

# Results
```

```
cm_results <- bind_rows(cm_results,
                        tibble(Model = "k-nearest neighbors",
                               Accuracy = cm$overall["Accuracy"],
                               Sensitivity = cm$byClass["Sensitivity"],
                               Specificity = cm$byClass["Specificity"]))
# Print the results
cm_results
```

```
## # A tibble: 2 x 4
##   Model               Accuracy Sensitivity Specificity
##   <chr>                  <dbl>       <dbl>       <dbl>
## 1 Logistic Regression    0.965       0.983       0.780
## 2 k-nearest neighbors    0.963       0.993       0.659
```

With KNN we have obtained similar accuracy than with Logistic Regression model, 0.963, let's see how much can we improve in next models.

4.3 Classification and Regression Trees

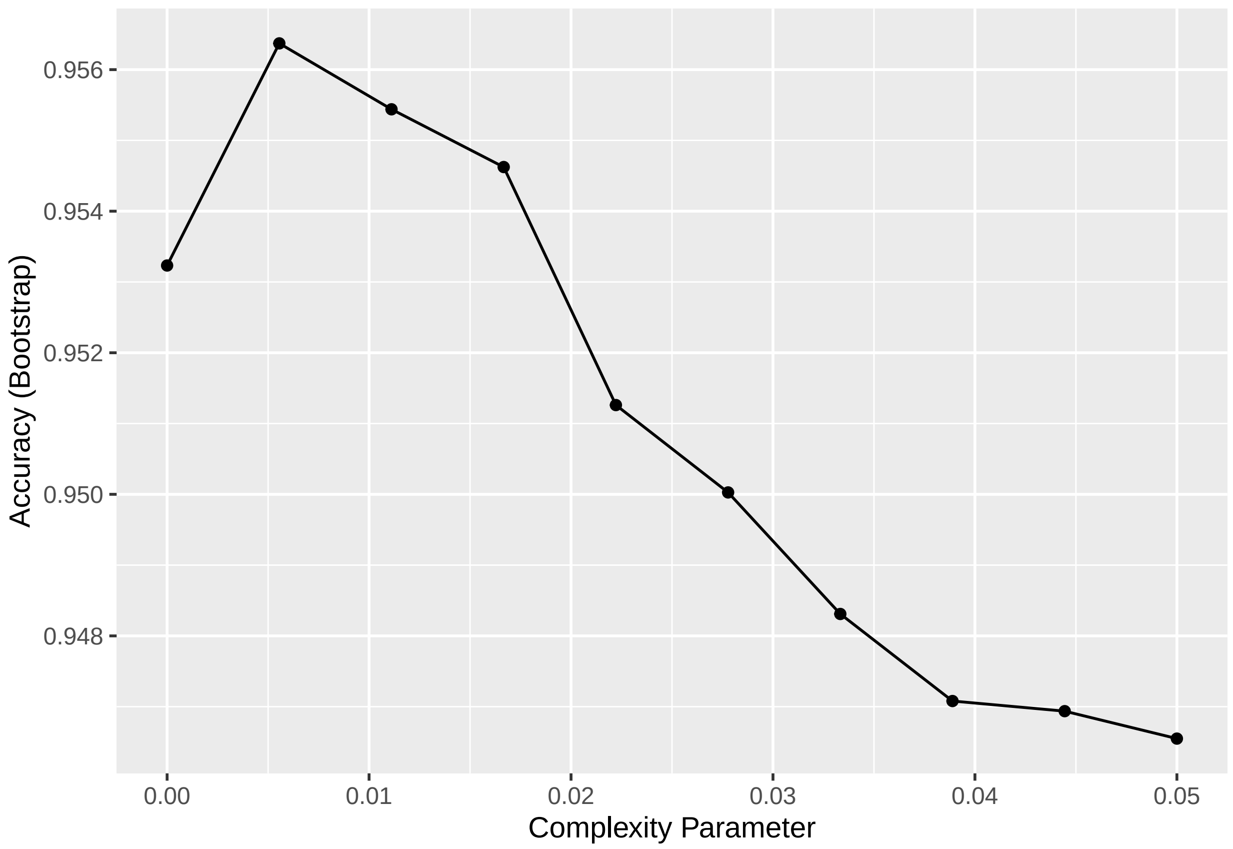
A tree is basically a flow chart of yes or no questions. A Regression or Decision Tree is a supervised learning predictive model that uses a set of binary rules to calculate a target value. It is used for either classification (categorical target variable) or regression (continuous target variable). Hence, it is also known as CART (Classification & Regression Trees).

Regression and decision trees operate by predicting an outcome variable Y by partitioning predictors.

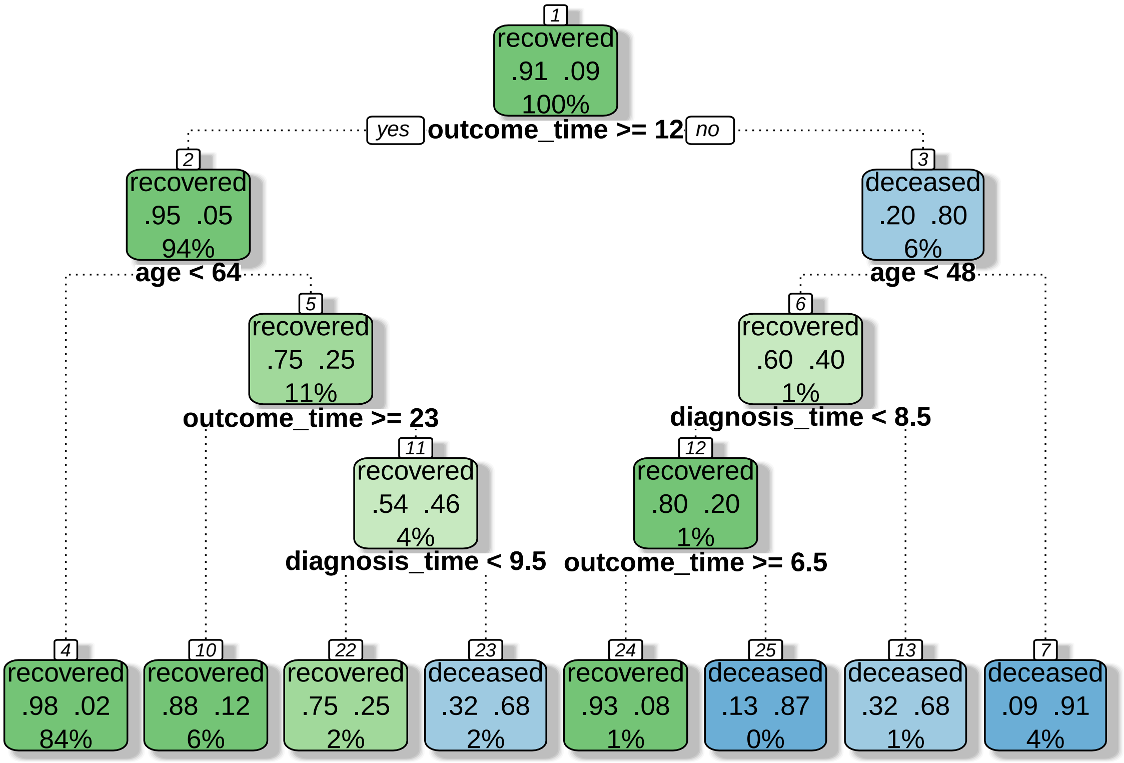
```
set.seed(19, sample.kind="Rounding")

# Fit rpart model
train_rpart <- train(outcome ~ .,
                     method = "rpart",
                     tuneGrid = data.frame(cp = seq(0, 0.05, len = 10)),
                     data = train_set)

# Best tuning parameter and plot
train_rpart$bestTune
```

```
##            cp
## 2 0.005555556
```

```
ggplot(train_rpart)
```



```
#To see the resulting tree we access the finalModel and plot it:
fancyRpartPlot(train_rpart$finalModel, space=0, yspace=0, tweak=1, under.cex=1, gap=0)
```



Rattle 2020−Jun−12 18:41:09 cmayora

```
# Variable of importance
imp <- as.data.frame(varImp(train_rpart)$importance)
imp <- data.frame(overall = imp$Overall,
                  names   = rownames(imp))
imp %>% arrange(-overall) %>% head()
```

```
##      overall                  names
## 1 100.000000           outcome_time
## 2  67.252606                    age
## 3  12.975059         diagnosis_time
## 4   5.000463       stateBogotá D.C.
## 5   1.107088 stateBarranquilla D.E.
## 6   1.102296    contagion_typetravel
```

```
y_hat = predict(train_rpart, test_set)

# Confusion Matrix
cm <- confusionMatrix(y_hat, test_set$outcome)

# Results
cm_results <- bind_rows(cm_results,
                        tibble(Model = "Regression Trees",
                               Accuracy = cm$overall["Accuracy"],
                               Sensitivity = cm$byClass["Sensitivity"],
                               Specificity = cm$byClass["Specificity"]))
```

```
# Print the results
cm_results
```

```
## # A tibble: 3 x 4
##   Model               Accuracy Sensitivity Specificity
##   <chr>                  <dbl>       <dbl>       <dbl>
## 1 Logistic Regression    0.965       0.983       0.780
## 2 k-nearest neighbors    0.963       0.993       0.659
## 3 Regression Trees       0.958       0.986       0.662
```

Again we got a similar accuracy, 0.958, having the same three important variables in the following order: *outcome_time*, *age* and *diagnosis_time*. Next let's evaluate a Random Forest model and see how much can we improve.

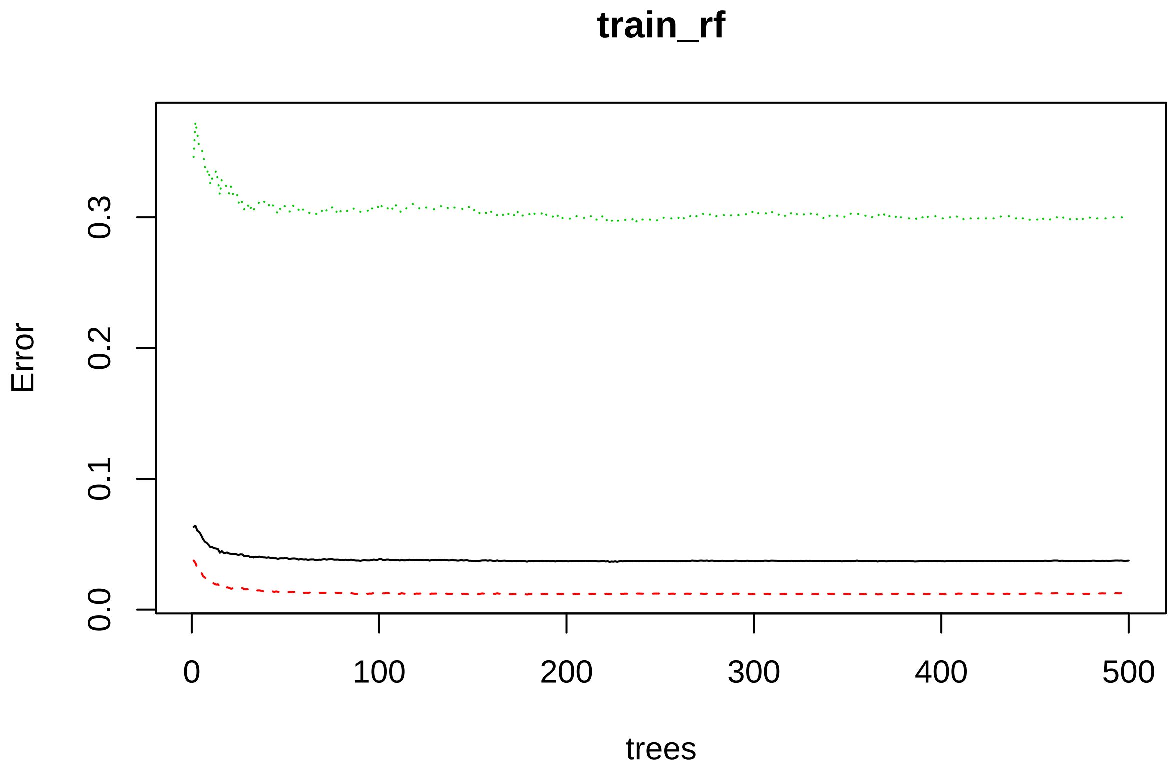
4.4 Random Forest

Random forests are a very popular machine learning approach that addresses the shortcomings of decision trees. The goal is to improve prediction performance and reduce instability by averaging multiple decision trees (a forest of trees constructed with randomness). The general idea of random forests is to generate many predictors, each using regression or classification trees, and then forming a final prediction based on the average prediction of all these trees. To assure that the individual trees are not the same, we use the bootstrap to induce randomness.

The name random forest derives from the random process of splitting the data and creating many trees, or a forest.

Let's check the model.

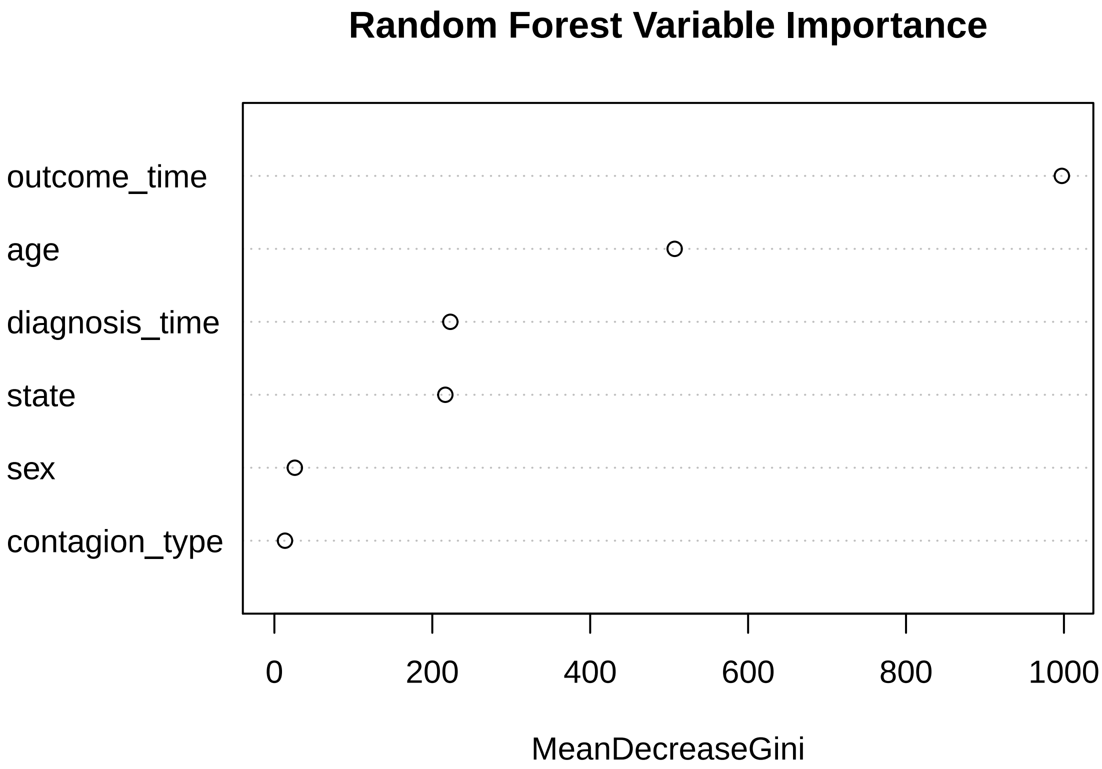
```
# Fit Random Forest model
train_rf <- randomForest(outcome ~ ., data=train_set)

plot(train_rf)
```



```
# Variable of importance
imp <- as.data.frame(varImp(train_rf))
imp <- data.frame(overall = imp$Overall,
                  names   = rownames(imp))
imp[order(imp$overall,decreasing = T),]
```

```
##     overall          names
## 6 997.34901   outcome_time
## 2 506.97608            age
## 5 222.87629 diagnosis_time
## 1 216.51650          state
## 3  25.91052            sex
## 4  13.45569 contagion_type
```

```
varImpPlot(train_rf, main = "Random Forest Variable Importance")
```



```
y_hat <- predict(train_rf, test_set)

# Confusion Matrix
cm <- confusionMatrix(y_hat, test_set$outcome)

# Results
cm_results <- bind_rows(cm_results,
                        tibble(Model = "Random Forest",
                               Accuracy = cm$overall["Accuracy"],
                               Sensitivity = cm$byClass["Sensitivity"],
                               Specificity = cm$byClass["Specificity"]))
# Print the results
cm_results
```

```
## # A tibble: 4 x 4
##   Model               Accuracy Sensitivity Specificity
##   <chr>                  <dbl>       <dbl>       <dbl>
## 1 Logistic Regression    0.965       0.983       0.780
## 2 k-nearest neighbors    0.963       0.993       0.659
## 3 Regression Trees       0.958       0.986       0.662
## 4 Random Forest          0.968       0.991       0.726
```

```
# Cleaning objects we won't use
rm(train_glm, control, train_knn, train_rpart, train_rf, y_hat)
```

In the first plot, we can see that the accuracy improves as we add more trees until about 50 trees where accuracy stabilizes. In general we have improved our accuracy from previous models, 0.968 as well sensitivity and specificity values. Checking the variables of importance we have the same three, in the same order *outcome_time*, *age* and *diagnosis_time*.

5 Final Prediction

From the results table above, we can see that Random Forest model got the best accuracy and sensitivity values. So, now we are going to apply this model using the *training* set to train the model and the *prediction* set to predict the outcome of all the active cases.

First we need to, prepare our data sets by removing columns we won't use for modeling, *id*, *symptoms_date* and *diagnose_date*. Also, we will calculate the *outcome_time* value in the *prediction* set, with the number of days, counting from the symptoms date to the current date, so we can predict considering the outcome time at the moment we run the prediction.

```r
# Remove id, symptoms_date and diagnosis_date
training <- training %>% select(-id, -symptoms_date, -diagnosis_date)

# Remove all the cases with an outcom_time and no outcome
# as this could be bad data
# calculate the outcome_time with current system date
prediction <- prediction %>% filter(is.na(outcome_time)) %>%
  mutate(outcome_time = as.integer(difftime(Sys.Date(), symptoms_date, units="days")))

# prediction str
str(prediction)
```

```
## 'data.frame':    22706 obs. of  10 variables:
##  $ id            : int  187 415 594 716 802 931 961 1171 1172 1181 ...
##  $ state         : Factor w/ 37 levels "Amazonas","Antioquia",..: 35 7 35 13 35 35 35 35 35 35 ...
##  $ age           : int  74 62 27 49 54 57 64 62 49 57 ...
##  $ sex           : Factor w/ 2 levels "F","M": 2 2 2 1 2 1 2 1 2 2 ...
##  $ contagion_type: Factor w/ 2 levels "contact","travel": 1 1 1 1 1 1 1 1 2 1 ...
##  $ symptoms_date : Date, format: "2020-03-09" "2020-03-14" ...
##  $ diagnosis_date: Date, format: "2020-03-21" "2020-03-24" ...
##  $ diagnosis_time: num  12 10 12 12 24 14 20 14 20 22 ...
##  $ outcome_time  : int  95 90 88 86 97 86 92 84 90 92 ...
##  $ outcome       : Factor w/ 2 levels "recovered","deceased": NA NA NA NA NA NA NA NA NA NA ...
```

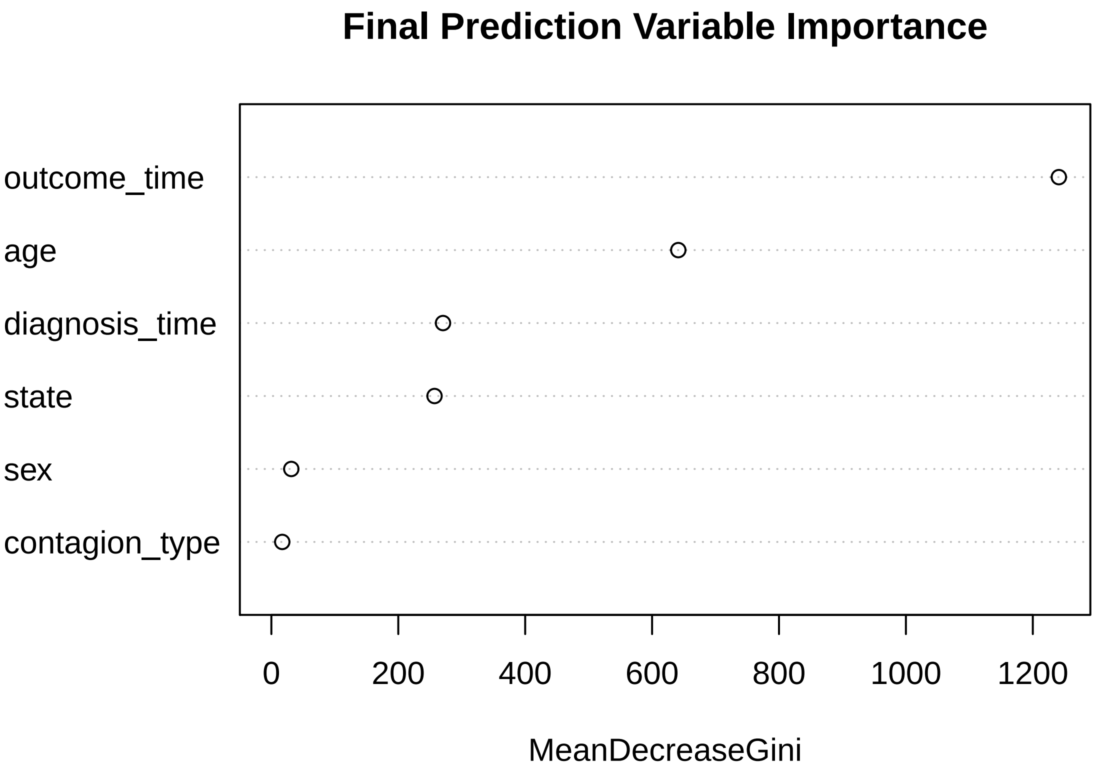
```r
# Remove id, symptoms_date and diagnosis_date
prediction_f <- prediction %>% select(-id,-symptoms_date,-diagnosis_date)
```

Now, let's run Random Forest and check the variables of importance.

```r
# Random Forest was selected
# train with whole training dataset
# removing the outcome column
final_train_rf <- randomForest(outcome ~ ., data=training)

# Variable of importance
imp <- as.data.frame(varImp(final_train_rf))
imp <- data.frame(overall = imp$Overall,
                  names   = rownames(imp))
imp[order(imp$overall,decreasing = T),]
```

```
##      overall          names
## 6 1241.03621   outcome_time
## 2  641.29765            age
## 5  270.48593 diagnosis_time
## 1  257.13544          state
## 3   31.41925            sex
## 4   17.16778 contagion_type
```

```
varImpPlot(final_train_rf, main = "Final Prediction Variable Importance")
```



As we can see we have the same variables of importance we got during our modeling: *outcome_time*, *age* and *diagnosis_time*. Now, we are ready to run our prediction and populate the outcome column in our *prediction* data set.

```
prediction$outcome <- factor(levels(prediction_f$outcome)[predict(final_train_rf, prediction_f[,-7])])

# Cleaning objects we won't use
rm(prediction_f, imp)

# 6 first rows of prediction dataset including column names
head(prediction)
```

```
##    id                state age sex contagion_type symptoms_date diagnosis_date
## 1 187      Valle del Cauca  74   M        contact    2020-03-09     2020-03-21
## 2 415          Bogotá D.C.  62   M        contact    2020-03-14     2020-03-24
## 3 594      Valle del Cauca  27   M        contact    2020-03-16     2020-03-28
## 4 716 Cartagena D.T. y C.  49   F        contact    2020-03-18     2020-03-30
## 5 802     Valle del Cauca  54   M        contact    2020-03-07     2020-03-31
## 6 931     Valle del Cauca  57   F        contact    2020-03-18     2020-04-01
##   diagnosis_time outcome_time   outcome
## 1             12           95 recovered
## 2             10           90 recovered
## 3             12           88 recovered
## 4             12           86 recovered
## 5             24           97 recovered
## 6             14           86 recovered
```

Now we have our prediction ready let's check some important data.

```
# Prediction analysis
cat("Death rate in training data: ",
    round((sum(training$outcome == "deceased")*100)/nrow(training), digits=2), "%", sep="")
```

```
## Death rate in training data: 8.71%
```

```
cat("Death rate in prediction data: ",
    round((sum(prediction$outcome == "deceased")*100)/nrow(prediction), digits=2), "%", sep="")
```

```
## Death rate in prediction data: 10.7%
```

```
cat("Recovered rate in training data: ",
    round((sum(training$outcome == "recovered")*100)/nrow(training), digits=2), "%", sep="")
```

```
## Recovered rate in training data: 91.29%
```

```
cat("Recovered rate in prediction data: ",
    round((sum(prediction$outcome == "recovered")*100)/nrow(prediction), digits=2), "%", sep="")
```

```
## Recovered rate in prediction data: 89.3%
```

We can see we have a difference of about 2% between our prediction and our known outcome data.

6 Conclusion

From the results table we can see how good the different models have predicted the outcome of the COVID-19 cases in Colombia, all with similar values, and with the same first three variable of importance, *outcome_time*, *age* and *diagnosis_time*, so, our calculated features helped in predicting the outcome, the other important feature is the age, which matches with what we have heard about the impact of COVID-19 in older population.

We have predicted and outcome for all the current active cases in Colombia, having as result death rate of 10.7% and recovered rate of 89.3%, representing a difference of about 2% against our known data. For this prediction we have splitted our data in *training* and *prediction* sets, being the *prediction* set all the current active cases in Colombia. The *training* set was used to train and evaluate four machine learning algorithms, Logistic Regression, K-nearest neighbor, Classification and Regression Trees and Random Forest.

At the end, our best two algorithms were Logistic Regression and Random Forest and we selected Random Forest for our final prediction which had an accuracy improvement of 0.27% and sensitivity improvement of 0.82%

With these results we can say that we can make use of machine learning algorithms to predict the outcomes of COVID-19 or any other disease.

7 References

- https://www.datos.gov.co/Salud-y-Protecci-n-Social/Casos-positivos-de-COVID-19-en-Colombia/gt2j-8ykr
- https://www.ins.gov.co/Noticias/Paginas/Coronavirus.aspx
- https://www.who.int/health-topics/coronavirus
- https://covid19.who.int/
- https://covid19.ncdhhs.gov/about-covid-19
- https://www.sciencedirect.com/science/article/pii/S1684118220300980
- https://unstats.un.org/unsd/ccsa/documents/covid19-report-ccsa.pdf
- https://www.medicalnewstoday.com/articles/the-impact-of-the-covid-19-pandemic-on-older-adults#Old-age-and-preexisting-health-conditions
- http://www.euro.who.int/en/health-topics/health-emergencies/coronavirus-covid-19/weekly-surveillance-report
- https://www.healthline.com/health-news/men-more-susceptible-to-serious-covid-19-illnesses
- https://www.gideononline.com/about/team/
- https://rafalab.github.io/dsbook/caret.html
- https://howtoteachdatascience.github.io/JSM2018/lectures/09-machine-learning-2.html
- https://www.machinelearningplus.com/machine-learning/caret-package/