COVID19_São_Paulo_State

Marcio Campos

23/11/2020

1. Introduction

Data Analysis is a powerfull tool that is being more and more used to understand behaviors and patterns to help us to take decisions. Currently, data about almost everything is anyhow available. The most important step when we have a dataset in hands is to extract useful information from that dataset, and, to do that, many methods can be used. Understanding the information hidden in a given dataset allow us to make expectations about the future. Combining the knowledge acquired from a dataset with some suitable methods a good prediction can be done. The combination of different Machine Learning algorithms can provide us a good accuracy in the prediction.

The object of this report is the CoronaVirus pandemic, derived from a virus, so called sars-cov-2 or COVID19, which started by the end of 2019 and begin of 2020 and currently is spread around the world. This pandemic has changed many things in our lives, forcing us to re-think the way to live. Many huge datasets are available with pandemic data.The focus is to analyse the data from Corona virus in Brazil, available in the official Government website. Since the complete COVID19 dataset from Brazil is too large, and splitted in each State, we used part of the dataset of the most populated state in Brazil, the São Paulo State. The data are collected by the Brazilian Public health System in each health unit across the country and released in the Health Ministery website.

1.1 The Dataset

The whole Covid dataset from São Paulo State is released in four files, with 1 million entries and more than 300Mb each file. The dataset we analyse in this report is part of the file number 1 "dados-sp-1.csv", released by the goverment. It contains, originally, an notification ID, notification date, simthoms date, birthdate, simthoms, health professional, test date, test status, test results, gender, age, evolution of the disease, information about the location of the notification and origin of the patient. We selected, as target variable, the evolution of the disease, which was filtered in two categories: "death", for death cases, or "survive", for recovered cases. In order to garantee the balance between that two categorical outputs, death or survive, we implemented the downsampling in the dataset. This partial dataset, so called "covid.csv", has all information we need to analyse the features and implement some prediction methods.
This report presents analysis on each feature of the dataset and also some methods to predict the conclusion of each case. There are many studies on the features that can cause a bad evolution of the disease. In [1] a meta analysis was done on many articles to present the prevalence of commorbidities in COVID19 cases, highlighting the gender, country and the evolution of the disease. In [2] the disease mortality (IFR - Infection Fatality Ratio) was analysed against the age and country of the patients. In [3] the definition of IFR and CFR is shown, highlighting the potential bias that can lead to a wrong mortality estimation. As an exemple, the different timing when the case result in death compared to the cases that result in recovery. That is the reason to filter the output variable only into solved cases, "death" or "survive", as stated by [3].
Regarding the testing methodology, some test types were used, mainly the RT-PCR test which is one of the most used CoronaVirus test in the dataset. The RT-PCR test is known as a "Gold Standard" in the sars-cov-2 diagnose tests [4]. Many articles were published, also researches under review process, on the accuracy of RT-PCR test. According to [5] that test can have a higher sensitivity, which means low number of false-negatives, with two consecutive RT-PCR test instead of only one. In [6] a meta analisys in 16 test

types was doneat the beginning of the pandemic, highlighing the accuracy, sensitivity, specificity and other parameters. Also in [7] , a meta analisys was done on some studies and the specificity of PCR tests was 86%, and sensitivity 96%. also interesting is the statemend of [7] that the accuracy varies according to the prevalence of the disease in the observations. Despite the discussion on the tests accuracy around the world, we calculated those parameters considering the test result and the final classification, which one of the columns in the dataset and represents a confirmation of the first test result.

1.1.1 Downloading and Creating the Dataset

The original dataset was downloaded from the adress: https://opendatasus.saude.gov.br/dataset/casos-nacionais in 26/09/2020.
The daily new information on the pandemic makes the goverment dataset changing everyday. Due to this, we did not updated the file.

The original dataset was downloaded by this code:

```
#Data csv from COVID19 - São Paulo State, Brazil - 26set2020. File number 1.
#"https://opendatasus.saude.gov.br/dataset/casos-nacionais"

#downloading the dataset.
set.seed(1, sample.kind = "Rounding")
covid_full <- read_csv2("dados-sp-1.csv", locale(encoding = "ISO-8859-1"),
                        col_names = TRUE, col_types = NULL)

#checking the target variable
levels(as.factor(covid_full$evolucaoCaso))
covid_full %>% group_by(evolucaoCaso) %>% summarize(n = n())

#filtering the target variable
covid_full <-covid_full %>% filter(evolucaoCaso == "Cura" | evolucaoCaso == "Óbito")
mean(covid_full$evolucaoCaso == "Óbito")
#prevalence here is 0,28%
```

We filtered the target variable in only two categories: death ("Óbito") and survive ("Cura"). The other available categories of this variable means cases where the patient is in house still under health care, or the patient is in a hospital or ICU. Also there were many "NAs" in the target variable. That means, we removed "NAs" and cases where the disease is still under treatment. The two output categories indicates that the dataset is very inbalanced, with very low proportion of "death" cases. The percent of death cases was 0.28%. We downsampled the dataset in order to reach a better prediction without bias. The code is the following:

```
## Due to imbalancing of the outcome "Case evolution", lets downsample the dataset
covid_full <- mutate_if(covid_full, is.character, as.factor)
covid_file <- downSample(covid_full[,!colnames(covid_full) %in% "evolucaoCaso"] ,
                         y = covid_full$evolucaoCaso)
covid_file <-  covid_file %>% mutate( evolucaoCaso = Class) %>% select(-Class)

#saving the dataset
write_csv2(covid_file, "covid.csv", append = FALSE, col_names = TRUE)
rm(covid_full, covid_file)
```

We created the dataset "covid.csv" , which was the starting point for this report. This dataset is available in https://raw.githubusercontent.com/marciofcampos/Corona-Analysis/master/covid.csv.

1.2 The Goal of the Project and The Structure of the Report

The main objective of this project is to analyse the features of the dataset in order to bring insights related to this new disease. Also , the objective is to apply some machine learning techniques to predict the COVID19 evolution to death or survival, based on a set of predictors , with the higher accuracy as possible, given the nature of the dataset. Actually, since the outcome of the algorithm is categorical, it means, "death" or "survive" , the task is a classification and not a prediction. In orther to achieve this target the first step was to clean and tide the data, by selecting and reorganizing columns, translating and filtering. After that the data was analysed and some possible predictors, such as notification date, gender, age, symtoms, testtype, were evaluated. After this step, some models were implemented, trained and applied in the test set. Finally, the classification algorithms were compared considering suitable parameters.

The report is divided as follows. The section 2 describes the methodology of the data analysis, that means, the data tidying and cleaning steps, the analisys on each predictor, and the modelling concept. The train set was used to train the models and the test result was used to check the accuracy on each model. The section 3 describes the results of the report, it means, the accuracy of the prediction models applied on the test set. in the conclusion, a short resume of the report and the possible next steps are presented.

2. Methodology

The original dataset was not organized, at least not for our purposes. That is a table from the Public Health System of Brazilian Goverment, about the COVID19 data of São Paulo State. That database is currently used for many purposes, including research. In order to simplify and permit a correct analysis on specific possible predictors the dataset had to be cleaned.

2.1 Tidying the dataset

The first change we made was to analyse the structure of the dataset and select those columns that has important information to the proposed study.

The selected columns were:

- dataNotificacao - notification date - the date when the patient entered in a Hospital;
- dataInicioSintomas - symtoms date - the date when some symtom has appeared;
- sintomas - symtoms - shows one or more symtoms presented by the patient;
- condicoes - conditions - shows whether the patient had commorbidies or not;
- estadoTeste - test status - whether the test is concluded or not;
- dataTeste - test date - the date when the test material was collected;
- tipoTeste - test type - shows the test method;
- resultadoTeste - test result - shows the test result as positive or negative;
- sexo - gender - male or female;
- municipioNotificacao - notification city - city when the case was notified;
- idade - age - age of the patient;
- evolucaoCaso - case evolution - shows whether the case conclusion was death or recovery;
- classificacaoFinal - final classification - final classificaton of each test result;

We translated the column names to english, as short names, and the new structure is:

```r
#the structure after selecting the desired columns
str(covid)
```

```
## tibble [1,596 x 13] (S3: tbl_df/tbl/data.frame)
##  $ notdate   : chr [1:1596] "2020-07-24T03:00:00.000Z" "2020-06-05T18:41:14.387Z" "2020-06-07T03:00:0
##  $ symtomdate: chr [1:1596] "2020-07-24T03:00:00.000Z" "2020-06-02T03:00:00.000Z" "2020-06-02T03:00:0
##  $ symtom    : chr [1:1596] "Outros" "Febre, Tosse, Outros" "Febre, Tosse, Outros" "Febre, Tosse, Do
##  $ conditions: chr [1:1596] "null" "null" "null" "null" ...
##  $ teststatus: chr [1:1596] "Concluído" "null" "Concluído" "Concluído" ...
##  $ testdate  : chr [1:1596] "2020-07-24T03:00:00.000Z" "null" "2020-06-07T03:00:00.000Z" "2020-08-25T
##  $ testtype  : chr [1:1596] "TESTE RÁPIDO - ANTICORPO" "null" "RT-PCR" "RT-PCR" ...
##  $ testres   : chr [1:1596] "Negativo" "null" "Positivo" "Negativo" ...
##  $ gender    : chr [1:1596] "Masculino" "Masculino" "Masculino" "Masculino" ...
## $ city      : chr [1:1596] "Bauru" "Buritama" "Campinas" "Peruíbe" ...
## $ age       : num [1:1596] 39 30 34 4 32 21 59 30 58 35 ...
## $ evolution : chr [1:1596] "Cura" "Cura" "Cura" "Cura" ...
## $ finalclass: chr [1:1596] "Descartado" "Descartado" "Confirmado Laboratorial" "Síndrome Gripal Não
```

where:

- notdate -> notification date;
- symtomdate -> symtoms date;
- symtom -> symtoms;
- conditions -> patient condition (commorbidities);
- teststatus -> test status;
- testdate -> test date;
- testtype -> test type;
- testres -> test result;
- gender -> gender;
- city -> notification city;
- age -> age of the patient;
- finalclass -> final classificaton of test result;

After selecting the columns we translated and checked the balance of the target variable.

Var1	Freq
death	798
survive	798

The table above shows the same amount of samples for each category. The expected value of deaths is 0.5 , the standard error is 0.0125, the margin of error is 2.5, and the 95% confidence interval is 0.5245, 0.4755.

2.1.1 Timing Related Columns

We started the cleaning with the columns related to the timing. The timestamp from the column "notdate", notification date, was splitted in date and week, to allow further analysis on timing effects.

```r
#tyding the timestamp of notification date column, by separating the date and time:
datetime <- covid$notdate
datetime <- as.data.frame(datetime)%>%separate(datetime, c("date", "time"), sep = "T")
datetime$date <- as_datetime(datetime$date)
datetime <- datetime %>% mutate(notweek = week(date)) #Splitting the week
covid <- covid %>% mutate(notdate = datetime$date, notweek = datetime$notweek) %>%
  filter(!is.na(notdate)) #Replacing the dates in the dataset

#tyding the timestamp of symtom date column, by separating the date and time:
datetime <- covid$symtomdate
datetime <- as.data.frame(datetime)%>%separate(datetime, c("date", "time"), sep = "T")
datetime$date <- as_datetime(datetime$date)
covid <- covid %>% mutate(symtomdate = datetime$date) %>%
  filter(!is.na(symtomdate)) #Replacing the dates in the dataset

#tyding the timestamp of test date column, by separating the date and time:
datetime <- covid$testdate
datetime <- as.data.frame(datetime)%>%separate(datetime, c("date", "time"), sep = "T")
datetime$date <- as_datetime(datetime$date)
covid <- covid %>% mutate(testdate = datetime$date) %>%
  filter(!is.na(testdate)) #Replacing the dates in the dataset
```

The same was done in the columns “symtomdate” and “testdate”.

2.1.2 Checking the Test Status

After time/date arrangement, we checked the “teststatus” column. We found some status different than “concluded”. Since we are interested only in the concluded tests, we filtered the column “teststatus” by keeping only the rows with concluded tests. After this filtering, the “test status” column was removed.

```r
# Filtering only the concluded tests, removing nulls from test status and test results
covid <- covid %>% filter(teststatus == "Concluído") %>%
  select(-teststatus)
```

Since some columns has portuguese characters, we translated the columns “testres”, “testtype” and “gender” to english. Also, we filtered the dataset to remove undesirable entries.

2.1.3 Checking the Test Results

From the column “testres”, test results, the considered levels were:

```
## [1] "Negativo" "Positivo"
```

We translated the testresults to english and changed the class to factor.

```r
#Translating the testresults.
covid$testres <- str_replace_all(covid$testres,
                                c("Positivo" = "positive",
                                  "Negativo" = "negative")) %>%
  as.factor()
```

2.1.4 Cleaning the Column Test Type

From the column “testtype”, we can see that there are some test types in the dataset:

```
## [1] "Enzimaimunoensaio - ELISA IgM"
## [2] "Imunoensaio por Eletroquimioluminescência \023 ECLIA"
## [3] "Imunoensaio por Eletroquimioluminescência - ECLIA IgG"
## [4] "Quimioluminescência - CLIA"
## [5] "RT-PCR"
## [6] "TESTE RÁPIDO - ANTICORPO"
## [7] "TESTE RÁPIDO - ANTÍGENO"
```

We translated the tests to english. However, the RT-PCR was not translated.

```r
#Filtering and Translating the testtype. The RT-PCR was not traslated.
covid$testtype <- str_replace_all(covid$testtype,
        c("TESTE RÁPIDO - ANTÍGENO" = "RT-antigen",
          "TESTE RÁPIDO - ANTICORPO" = "RT-antibody",
          "Enzimaimunoensaio - ELISA IgM" = "ELISA",
          "Imunoensaio por Eletroquimioluminescência \023 ECLIA" = "ECLIA",
          "Imunoensaio por Eletroquimioluminescência - ECLIA IgG" = "ECLIA",
          "Quimioluminescência - CLIA" = "CLIA")) %>% as.factor()
```

We see below that some test types were more used than others.

```r
#number of each test
covid %>% group_by(covid$testtype) %>%
  summarize(n = n()) %>%
  knitr::kable()
```

covid$testtype	n
CLIA	1
ECLIA	3
ELISA	2
RT-antibody	363
RT-antigen	31
RT-PCR	978

We removed the tests with very low samples and focused on the RT-PCR, RT-antibody and RT-antigen tests.

```
#removing the tests with low samples
covid <- covid %>% filter(testtype == "RT-PCR" | testtype == "RT-antibody" | testtype == "RT-antigen")
```

2.1.5 Checking the Gender

From the column “gender” the levels are the following:

```
## [1] "Feminino"  "Masculino"
```

We translated the gender to english and changed from character to factor.

2.1.4 Cleaning the Age

After that, from the column “age”, we filtered the age to higher than -1 and lower than 91 years old to avoid wrong entries.

```
#filtering the age to remove entries higher than 90 years old and lower than 0.
covid <- covid %>% filter(age < 91 & age > -1)
```

2.1.5 Checking the Symtoms

Next step was to check the column: “symtom”. There are many symtom combinations. Those symtoms was described by the patient and checked by the health proffesional during the entrance in one of the public health units. We translated and splitted the symtoms in different columns in order to analise them separately. However we kept the column “Symtom” with each entry sorted alfabetically to avoid double accounting on the symtom combination (i.e. “fever, cought” and “cought, fever”). The “symtom” column was analysed later regarding to the symtoms combinations.
Below we can see part of the combinated simtoms:

```
##  [1] "Assintomático"                      "Coriza"
##  [3] "Coriza, Dor de Garganta, Tosse"     "Dispneia"
##  [5] "Dispneia, Dor de Garganta"          "Dispneia, Dor de Garganta, Outros"
##  [7] "Dispneia, Febre"                    "Dispneia, Febre, Outros"
##  [9] "Dispneia, Febre, Tosse"             "Dispneia, Febre, Tosse, Outros"
## [11] "Dispneia, Outros"                   "Dispneia, Outros, Febre"
## [13] "Dispneia, Outros, Febre, Tosse"     "Dispneia, Outros, Tosse"
## [15] "Dispneia, Tosse"                    "Dispneia, Tosse, Febre"
## [17] "Dispneia, Tosse, Febre, Outros"     "Dispneia, Tosse, Outros"
## [19] "Dispneia,Febre"                     "Dispneia,Febre,Tosse"
```

We checked and translated to english all symtoms , created separeted columns for each one, and sorted each entry of the column “symtom” alfabetically.

```
#Translate the symtoms
covid$symtom <- str_replace_all(covid$symtom,
                                c("Tosse" = "cought",
```

```r
                                  "Febre" = "fever",
                                  "Dor de Garganta" = "sore_throat",
                                  "Dor de Cabeça" = "headache",
                                  "Dispneia" = "dyspnoea",
                                  "Distúrbios Gustativos" = "taste_disorder",
                                  "Distúrbios Olfativos" = "olfactory_disorder",
                                  "Coriza" = "runny_nose",
                                  "Assintomático" = "asymtomatic",
                                  "Outros" = "others"))


#Splitting the symtoms in separated columns and filtering "NAs" entries
covid <- covid %>% mutate(cought = str_detect(covid$symtom,"cought"),
                      fever = str_detect(covid$symtom,"fever"),
                      sorethroat = str_detect(covid$symtom,"sore_throat"),
                      headache = str_detect(covid$symtom,"headache"),
                      dyspnoea = str_detect(covid$symtom,"dyspnoea"),
                      tastedisorder = str_detect(covid$symtom,"taste_disorder"),
                      olfactorydisorder = str_detect(covid$symtom,"olfactory_disorder"),
                      runnynose = str_detect(covid$symtom,"runny_nose"),
                      asymtomatic = str_detect(covid$symtom, "asymtomatic"),
                      others = str_detect(covid$symtom,"others")) %>%
  filter(!is.na(symtom))

#creating the column "symtom" with symtoms in the same order
covid$symtom <- sapply(strsplit(covid$symtom, ","), function(x){
  paste(sort(trimws(x)), collapse = ", ")
  })
```

2.1.6 Checking the Final Classification

From the column “finalclass”, the final classification of each case, the levels of the final classification are:

```
## [1] "Confirmação Laboratorial"          "Confirmado Clínico-Epidemiológico"
## [3] "Confirmado Clínico-Imagem"         "Confirmado Laboratorial"
## [5] "Confirmado por Critério Clínico"   "Descartado"
## [7] "Síndrome Gripal Não Especificada"
```

we translated to english, removed “NAs”, and changed the column class from character to factor. We categorized the final classification in two levels: confirmed, and discarted.

```r
#translating the classification
covid$finalclass <- str_replace_all(covid$finalclass,
                                c("Confirmação Laboratorial" = "confirmed",
                                  "Confirmado Clínico-Epidemiológico" = "confirmed",
                                  "Confirmado Clínico-Imagem" = "confirmed",
                                  "Confirmado Laboratorial" = "confirmed",
                                  "Confirmado por Critério Clínico" = "confirmed",
                                  "Descartado" = "discarted",
                                  "Síndrome Gripal Não Especificada" = "discarted")) %>%
  as.factor()

covid <- covid %>% filter(!is.na(finalclass))
```

2.1.7 Checking the Column Conditions

From the column "conditions", we found many combinations of previous commorbidities or conditions that is known as key factor for the disease evolution, i.e. [1]. The first 10 levels we found were:

```
##  [1] "Diabetes"
##  [2] "Diabetes, Doenças cardíacas crônicas"
##  [3] "Diabetes, Doenças renais crônicas em estágio avançado (graus 3, 4 ou 5)"
##  [4] "Diabetes, Doenças respiratórias crônicas descompensadas"
##  [5] "Diabetes, Imunossupressão, Doenças cardíacas crônicas"
##  [6] "Diabetes, Portador de doenças cromossômicas ou estado de fragilidade imunológica"
##  [7] "Diabetes,Doenças cardíacas crônicas"
##  [8] "Doenças cardíacas crônicas"
##  [9] "Doenças cardíacas crônicas, Diabetes"
## [10] "Doenças cardíacas crônicas, Diabetes, Doenças renais crônicas em estágio avançado (graus 3, 4 
```

We removed "NAs", translated to the english and organized each row in alfabetic order with the code below:

```
#translating the previous conditions
covid <- covid %>% filter(!is.na(conditions))
covid$conditions <- str_replace_all(covid$conditions,
      c("Diabetes" = "diabetes",
        "Doenças cardíacas crônicas" = "heart_disease",
        "Doenças renais crônicas em estágio avançado \\(graus 3, 4 ou 5\\)" =
          "kidney_disease",
        "Doenças respiratórias crônicas descompensadas" = "respiratory_disease",
        "Imunossupressão" = "immunodeficiency",
        "Portador de doenças cromossômicas ou estado de fragilidade imunológica" =
          "immunodeficiency",
        "Obesidade" = "weight",
        "Gestante" = "pregnant",
        "null" = "no_previous_conditions"))

#Ordering "conditions" in the alfabetic order.
covid$conditions <- sapply(strsplit(covid$conditions, ","), function(x){
  paste(sort(trimws(x)), collapse = ", ")
})
```

The 10 first levels are:

```
##  [1] "diabetes"
##  [2] "diabetes, heart_disease"
##  [3] "diabetes, heart_disease, immunodeficiency"
##  [4] "diabetes, heart_disease, kidney_disease"
##  [5] "diabetes, heart_disease, respiratory_disease"
##  [6] "diabetes, heart_disease, weight"
##  [7] "diabetes, immunodeficiency"
##  [8] "diabetes, immunodeficiency, respiratory_disease"
##  [9] "diabetes, kidney_disease"
## [10] "diabetes, respiratory_disease"
```

2.2 Creating the Working dataset.

THe working dataset was created after some cleaning and translating steps. Two more steps were necessary. We changed the class of the predictors from character to factor, from logical to numeric and from integer to numeric, in order to allow the usage of some modelling strategies that requires such classes.

```r
#Change the class of the predictors from character to factor, and from logical and integer to numeric
covid <- mutate_if(covid, is.character, as.factor)
covid <- mutate_if(covid, is.logical, as.numeric)
covid <- mutate_if(covid, is.integer, as.numeric)
```

Finally, we checked again the proportion of deaths and survivals.

```r
#Number of recoveries and deaths
table(covid$evolution) %>% knitr::kable()
```

Var1	Freq
death	621
survive	657

We see that the target output remains balanced after the tidying process. Now the dataset is ready to be analysed. We, then, created the "workset" as starting point of the analysis.

```r
#creating the workset
workset <- covid
```

2.3 Analysing The Dataset

The workset has 23 columns, in which 10 of them are the symtoms. Also there are the columns related to the gender, age, notification date, test type, test result, conditions and final classification of each case. We analysed each one to understand the relationship between the column information and the evolution of the disease, the target variable.

2.3.1 Gender

We started with the gender. The amount of males and females in the workset is shown in the table below:

```r
#analysis by gender
n_female <- workset %>% filter(gender == "female") %>% nrow() #amount of female
n_male <- workset %>% filter(gender == "male") %>% nrow() #amount of male
p_female <- round((n_female / nrow(workset)*100), 0)  #pecent of female
p_male <- round((n_male / nrow(workset)*100), 0)  #percent of male

#creating a data frame with the distribution by gender.
gender_sp<- data.frame( c(n_female, p_female), c(n_male, p_male))
colnames(gender_sp) <- c("Female", "Male")
rownames(gender_sp) <- c("number of cases [un]", "percentage [%]")
gender_sp %>% knitr::kable()
```

	Female	Male
number of cases [un]	648	630
percentage [%]	51	49

Ther are few more females than males in the dataset. Let's analise the COVID19 evolution by gender. The proportion of deaths per gender is the following:

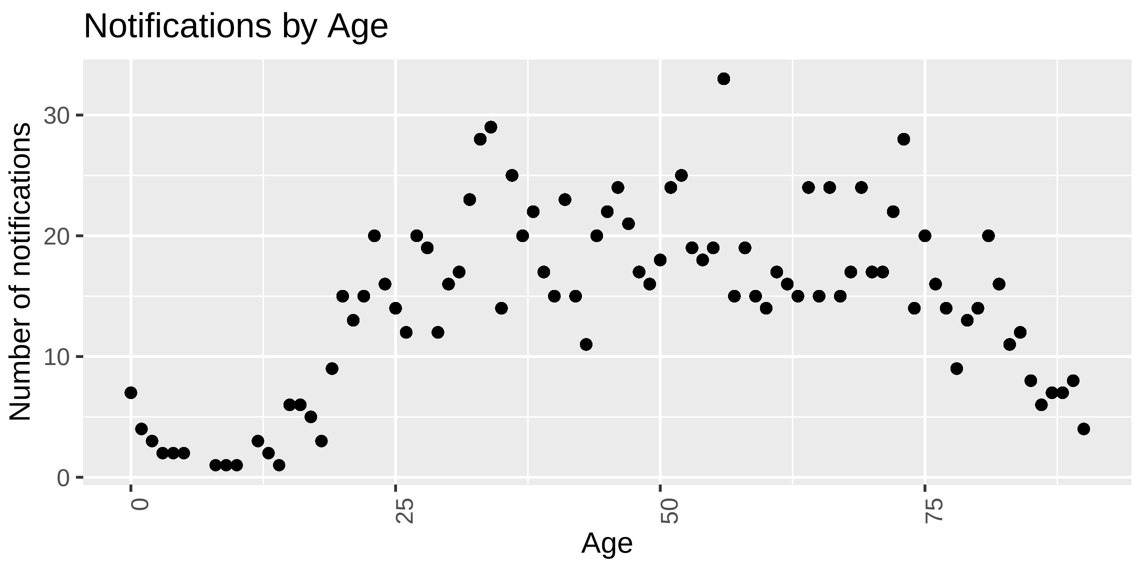
```r
#checking the evolution of the COVID19 for each gender:
workset %>% group_by(gender) %>%
  summarize(death = round(mean(evolution == "death"), 2),
            survive = round(mean(evolution =="survive"), 2)) %>%
  knitr::kable()
```

gender	death	survive
female	0.41	0.59
male	0.57	0.43

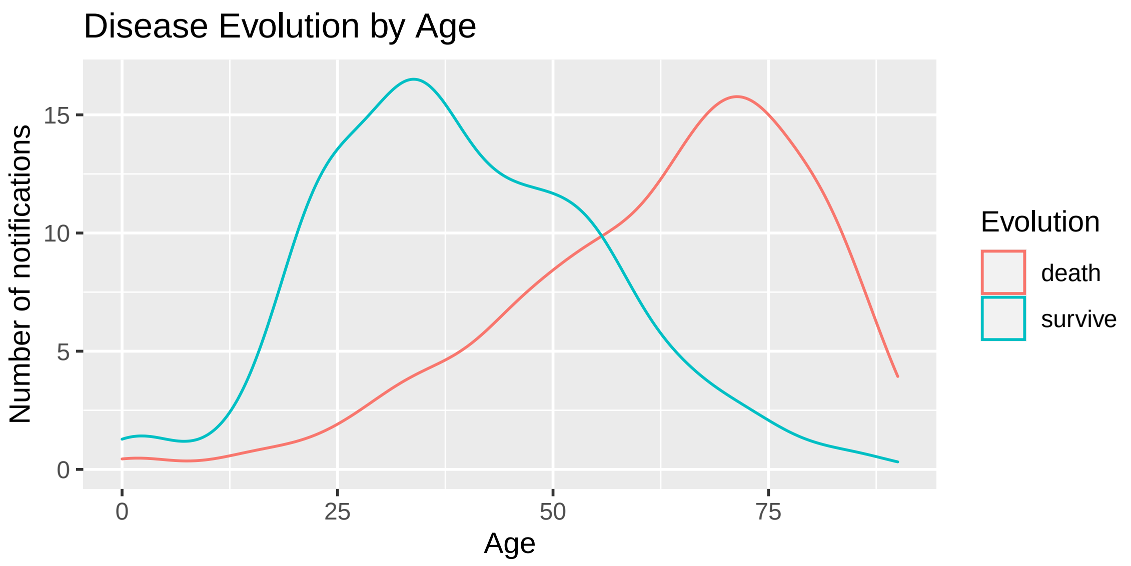
The are few mor deaths in males than females. This proportion does not represent the total population because we downsampled the dataset. However, it could be important for prediction models.

2.3.2 Age

Analysing the number of the cases by age, the number of notifications is lower in the extremes of the age range. The ages between 25 and 75 years old have high number of notifications, as we can see in the picture below.

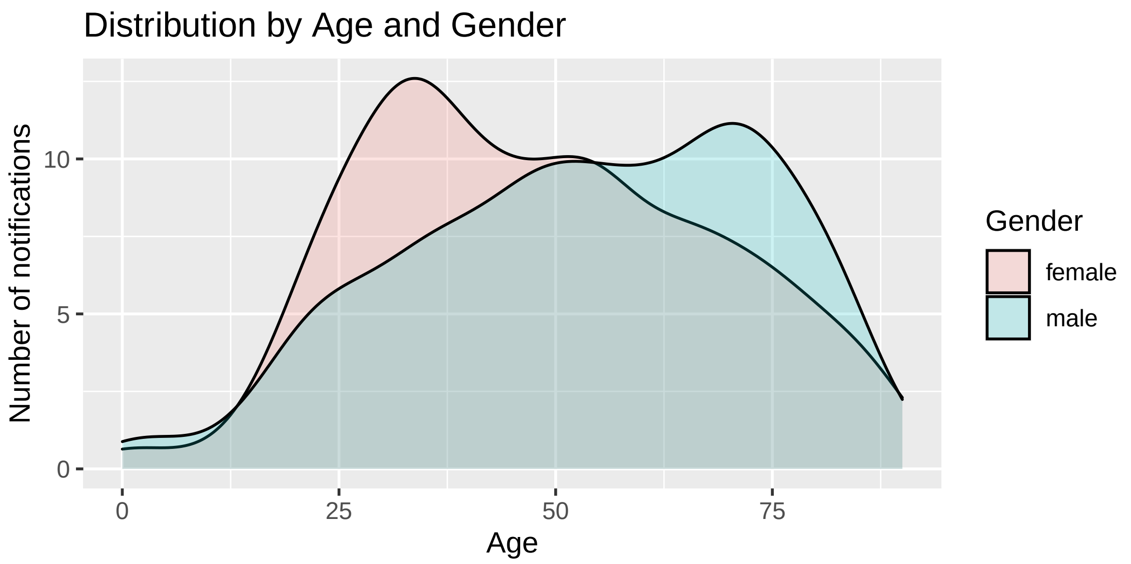


By checking the evolution of the disease by age, we see the picture below:



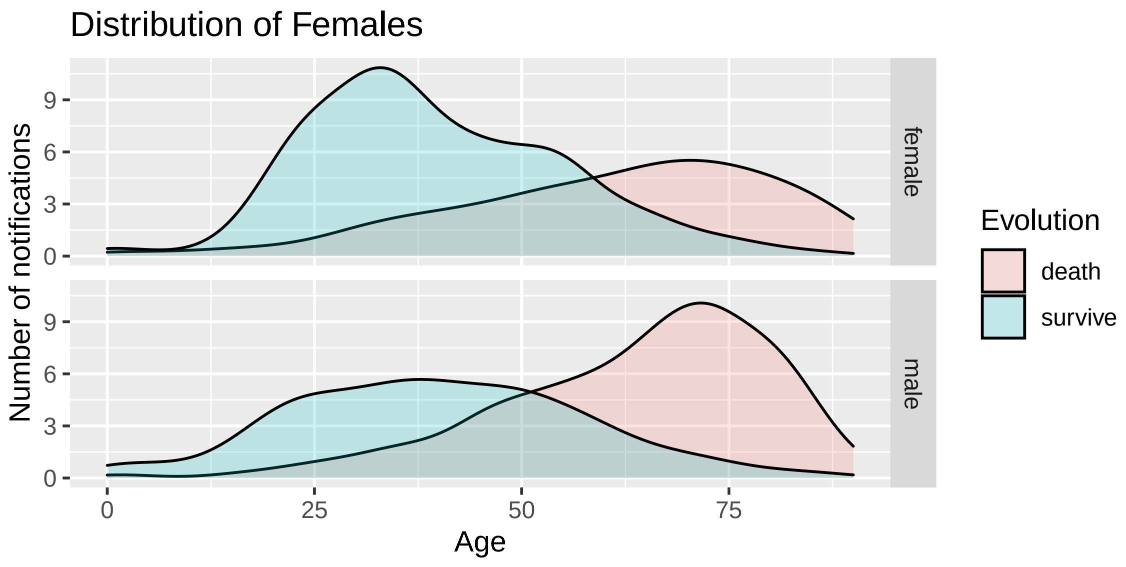
The distribution of the disease evolution on the age shows that the proportion of deaths is higher in older people, as stated previously in the gender analysis.
As an example, the proportion of deaths below 50 years old 0.22 , and the proportion of deaths over 50 years old is 0.73. The odds ratio is 3.3. This feature "age" was defined as one of the most important predictors.

Now let's analyse age and gender together. The distribution of age by gender is the following:



We see that there are more older males and females, and more younger females than males.
The distribution of deaths and recoveries among gender and age is shown below:

```
#distribution of disease evolution by age and gender

workset %>% ggplot(aes(age, y=..count.. , fill = evolution)) +
  geom_density(alpha=0.2) +
  ggtitle("Distribution of Females") +
  ylab("Number of notifications") +
  xlab("Age") +
  labs(fill="Evolution") +
  facet_grid(gender~.)
```



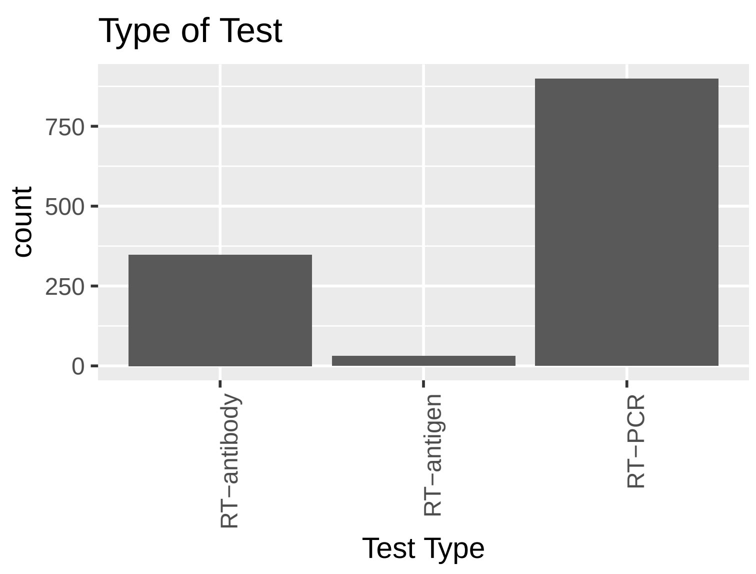
```
#proportion of deaths in males older than 50:
d_males <- workset %>% filter(gender == "male" & age > 50) %>%
  summarize(mean = mean(evolution == "death")) %>% round(2)

#proportion of deaths in females older than 50:
d_females <- workset %>% filter(gender == "female" & age > 50) %>%
  summarize(mean = mean(evolution == "death")) %>% round(2)
```

From the picture above, we see clearly the trend to have more deaths in older people, and among those people, more deaths in males than females. Also, for youngers, more recoveries in females than males. Considering a cutoff in 50 years old the proportion of male deaths is 0.79. And for females, the proportion of deaths over 50s is 0.66.
The gender column was also kept in the workset as a possible predictor.

2.3.3 Test Type

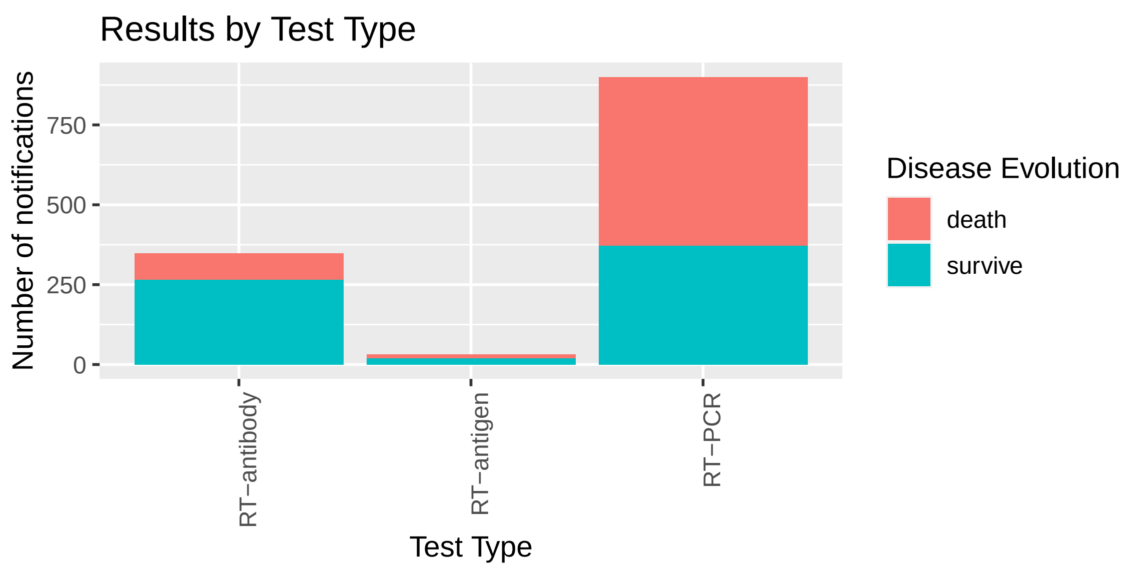
By checking the test result according the type of test we can see the distribution below:

```
#analysing the type of test
#how many tests per type
workset %>% ggplot(aes(testtype)) +
  geom_bar(stat = "count") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ggtitle("Type of Test") +
  xlab("Test Type")
```



The three selected tests are the RT-antibody, and RT-antigen, which analyse the blood, and the RT-PCR, which analyse the virus presence in the upper respiratory system, collecting material through the nose. The disease evolution according to the test type is shown below:

```r
#disease evolution per test type:
workset %>%
  group_by(testtype) %>%
  ggplot(aes(testtype, fill = evolution)) +
  geom_bar(stat = "count", show.legend = TRUE) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ggtitle("Results by Test Type") +
  ylab("Number of notifications") +
  xlab("Test Type") +
  labs(fill="Disease Evolution")
```



```r
#Table format
workset %>% group_by(testtype) %>%
  summarize(death = sum(evolution == "death"), survive = sum(evolution == "survive")) %>%
  knitr::kable()
```

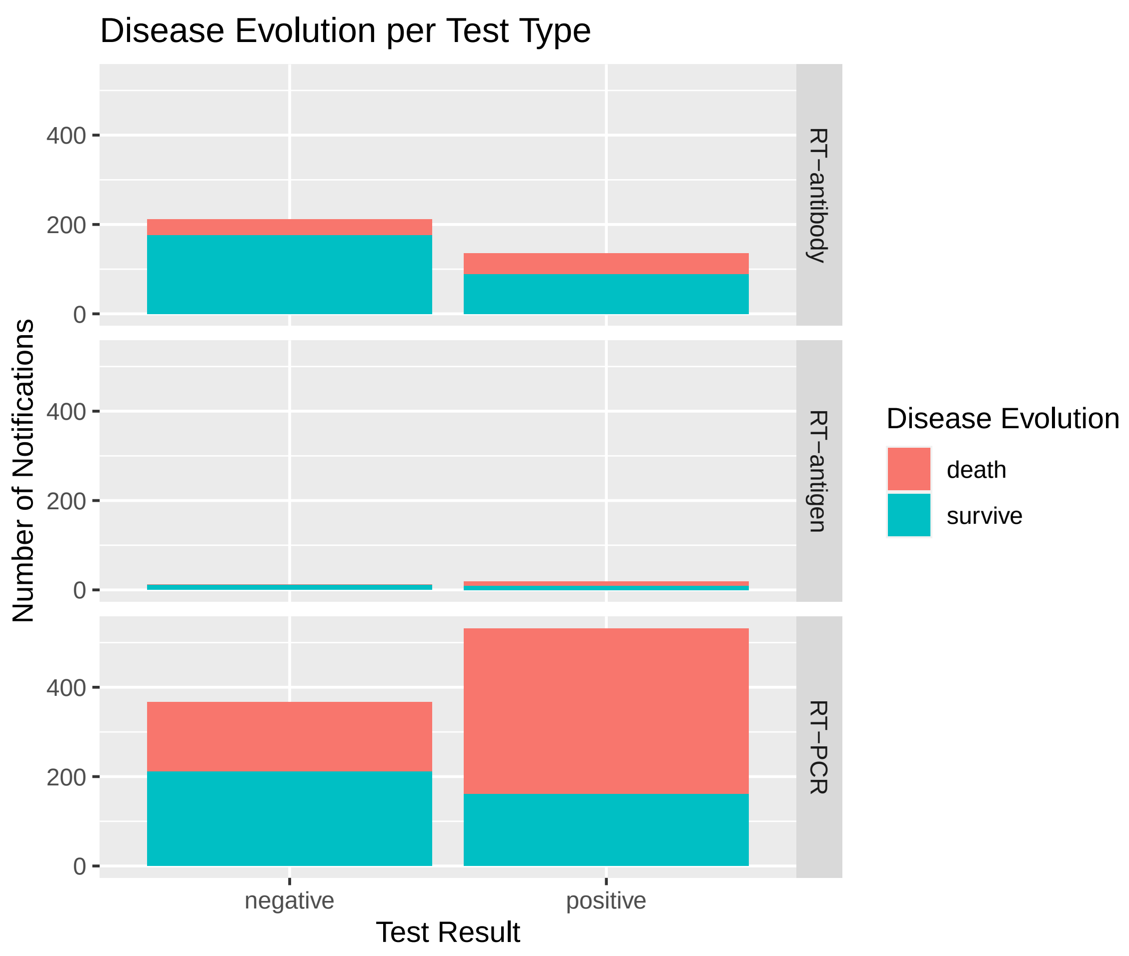
testtype	death	survive
RT-antibody	83	265
RT-antigen	11	20
RT-PCR	527	372

As we can see, considering only the three most used tests, there were less deaths when RT-antibody test was applied. That is because that RT-antibody test, as we can see later, has a recommendation to be applied after the 7th day of the disease, later than the RT-PCR [7]. The RT-PCR is performed mostly between 3rd and 5th day after the simtoms [7], and at this moment the evolution of the disease is still not clear.

2.3.4 Test Results

From the column "testres" we have the results of each type of test. Considering the tes results, test type and the evolution of the COVID19, the picture below shows that there is more deaths whe the test is RT-PCR. Again, that is related to the period when the test is performed, at the first days after the symtoms, when there is no clear definition on how will be the disease evolution.

```
#test result related to disease evolution and test type
workset %>% ggplot(aes(testres, fill =  evolution)) +
  geom_histogram(binwidth = 10, stat = "count") +
  xlab("Test Result") +
  ylab("Number of Notifications") +
  labs(fill="Disease Evolution") +
  ggtitle("Disease Evolution per Test Type") +
  facet_grid(testtype~.)
```



2.3.5 Symtoms

The main symtoms is declared by the patient, checked by the health proffesional, and noted in the patient formular. The most common symtoms of that dataset are *cought*, *fever*, and *sorethroat* , as well as the most common entry *others*, as we can see in the table below:

```
#proportion of appearance for each symtom
workset %>% gather(symtoms, appearance, `cought`:`others`, na.rm = TRUE) %>%
  select(symtoms, appearance, testres, evolution) %>%
  group_by(symtoms) %>%
  summarize(proportion = round((mean(appearance)), 2)) %>%
  arrange(desc(proportion)) %>%
  knitr::kable()
```

symtoms	proportion
others	0.59
cought	0.58
fever	0.41
dyspnoea	0.34
sorethroat	0.25
asymtomatic	0.01
headache	0.01
runnynose	0.01

symtoms	proportion
olfactorydisorder	0.00
tastedisorder	0.00

Cought, fever, dyspnoea and sore throat were the most commom symtoms in the dataset. Other point are the small cases of asymtomatic patients in this dataset, which can be far higher "in the streets", but anyhow they were checked by the public health system. That could be the case of a person who wants to check if he is infected of not, after having contact with a parent or somebody who tested positive. Or a health professional who has to repeat the test periodically. Most of the entries has more than one symtom together. The most 20 symtom combination appearance is the following:

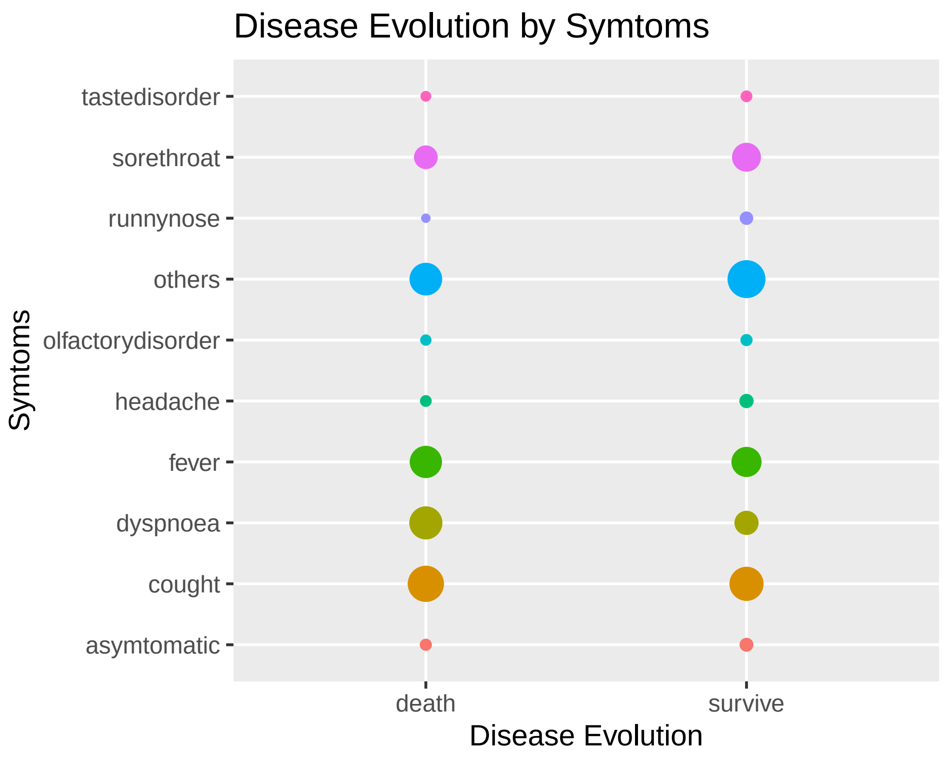
```r
#number of each symtom combination
workset %>% select(symtom) %>%
  filter(!is.na(symtom)) %>%
  group_by(symtom) %>%
  summarize( n = n()) %>%
  arrange(desc(n)) %>%
  head(20) %>%
  knitr::kable()
```

symtom	n
others	248
cought, others	82
cought, fever, others	79
cought, dyspnoea, fever	73
cought	66
cought, dyspnoea	65
cought, fever	50
fever, others	48
cought, dyspnoea, others	47
cought, fever, sore_throat	42
cought, sore_throat	36
dyspnoea	36
cought, dyspnoea, fever, others	35
cought, others, sore_throat	31
dyspnoea, others	29
others, sore_throat	29
cought, fever, others, sore_throat	28
dyspnoea, fever	27
cought, dyspnoea, fever, others, sore_throat	26
cought, dyspnoea, fever, sore_throat	25

Again, we see many entries with *others*, perhaps not common symtoms for a flu-like disease. The most common symtom combination are among the *cought*, *fever*, *dyspnoea* and *sorethroat*.

Considering each simtom against the disease we can plot the picture below.

```r
#relation between symtom and evolution of disease
workset %>% gather(symtoms, appearance, `cought`:`others`, na.rm = TRUE) %>%
  select(symtoms, appearance, evolution) %>%
  group_by(symtoms, evolution) %>%
  summarize(n = sum(appearance == 1)) %>%
  ggplot(aes(evolution, symtoms, color = symtoms, size = n)) +
  geom_point(show.legend = FALSE) +
  ggtitle("Disease Evolution by Symtoms") +
  ylab("Symtoms") +
  xlab("Disease Evolution")
```



We see that *fever* , *cought* and *dyspnoea* are related with mora cases of death than the other symtoms. Now in table format, the amount of deaths and survivals by symtom is shown below:

```
#table format
workset %>% gather(symtoms, appearance, `cought`:`others`, na.rm = TRUE) %>%
  select(symtoms, appearance, evolution) %>%
  group_by(symtoms, evolution) %>%
  summarize(n = sum(appearance == 1)) %>%
  knitr::kable()
```

symtoms	evolution	n
asymtomatic	death	4
asymtomatic	survive	11
cought	death	398
cought	survive	338
dyspnoea	death	310
dyspnoea	survive	120
fever	death	288
fever	survive	237
headache	death	3
headache	survive	13
olfactorydisorder	death	2
olfactorydisorder	survive	4
others	death	299

symtoms	evolution	n
others	survive	449
runnynose	death	0
runnynose	survive	9
sorethroat	death	112
sorethroat	survive	209
tastedisorder	death	1
tastedisorder	survive	3

```
#cought fever and dyspnoea indicates more deaths than survivals
```

The analysis of isolated symtoms say that *dyspnoea* is the symtom more related to the death cases, folowed by $fever$ and *cought*. And those three are also the most common ones.

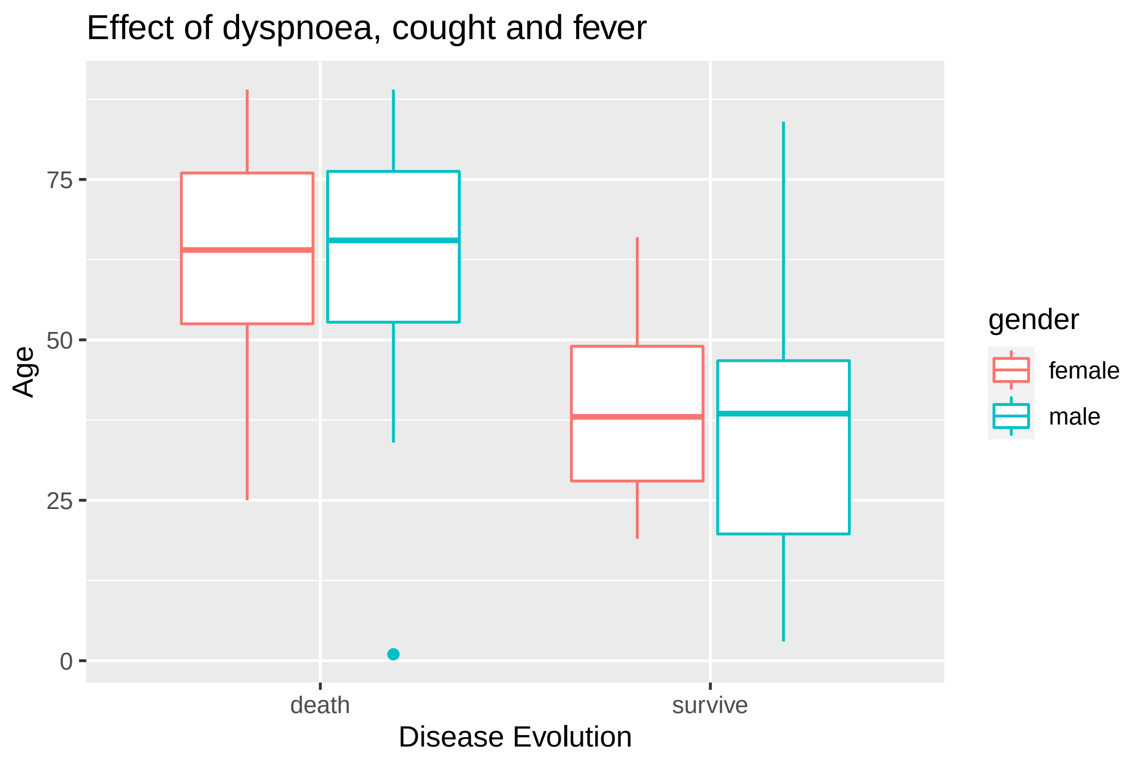
Analysing the combination of simtoms, colected when the patients were attended, against the disease evolution, we can see below that we have higher death proportion when the combination of the symtoms were among those three already commented: *dyspnoea*, $fever$ and *cought*. The 20 symtom combinations with highest death proportion is shown below:

```
#top 20 symtom combinations with higest death proportion
workset %>% select(symtom, evolution) %>%
  group_by(symtom) %>%
  summarize(n= n(), deathprop = round(mean(evolution =="death"), 2)) %>%
  arrange(desc(deathprop)) %>%
  head(20) %>%
  knitr::kable()
```

symtom	n	deathprop
cought, Dispnéia, fever	2	1.00
cought, fever, headache, others	1	1.00
cought, fever, olfactory_disorder	1	1.00
cought, headache, olfactory_disorder, others, taste_disorder	1	1.00
dyspnoea, sore_throat	2	1.00
dyspnoea	36	0.92
dyspnoea, fever	27	0.89
cought, dyspnoea	65	0.85
cought, dyspnoea, fever	73	0.84
cought, dyspnoea, fever, sore_throat	25	0.80
cought, dyspnoea, fever, others	35	0.74
dyspnoea, others	29	0.72
cought, dyspnoea, sore_throat	18	0.67
dyspnoea, fever, others	16	0.62
cought, fever, sore_throat	42	0.57
cought, fever	50	0.56
cought, fever, others	79	0.52
fever	23	0.52
cought, dyspnoea, others	47	0.51
cought, dyspnoea, others, sore_throat	14	0.50

We have to skip the entries with one or two appearances. Despite of that, the symtoms combinations with dyspnoea, cought and fever presented very high proportion of deaths.
By filtering those symtoms and comparing with gender and age we have the picture below:

```r
#Filtering dyspnoea, cought and fever, compared with gender and age
workset %>% select(evolution, gender, age, dyspnoea, cought, fever) %>%
  filter(dyspnoea == 1 & cought == 1 & fever == 1) %>%
  group_by(evolution, age, gender) %>%
  summarize(n= n()) %>%
  ggplot(aes(evolution, age, color = gender)) +
  geom_boxplot() +
  ggtitle("Effect of dyspnoea, cought and fever") +
  ylab("Age") +
  xlab("Disease Evolution") +
  labs(fill = "Gender")
```

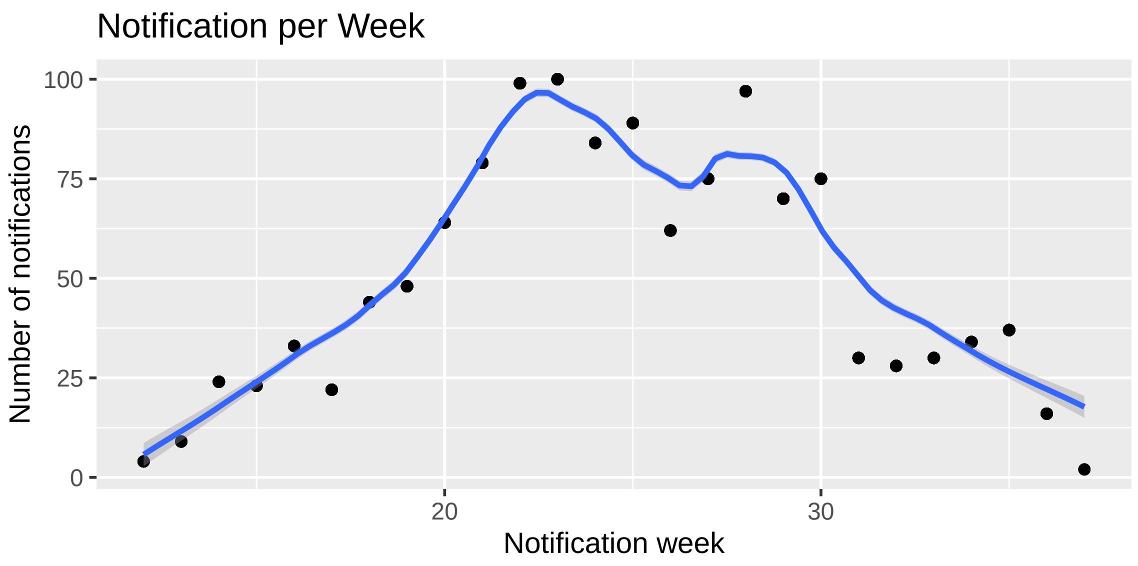


The picture shows that those three symtoms are very related with deaths of older people, no matter the gender. The symtoms are important predictors.

2.3.6 Timeline

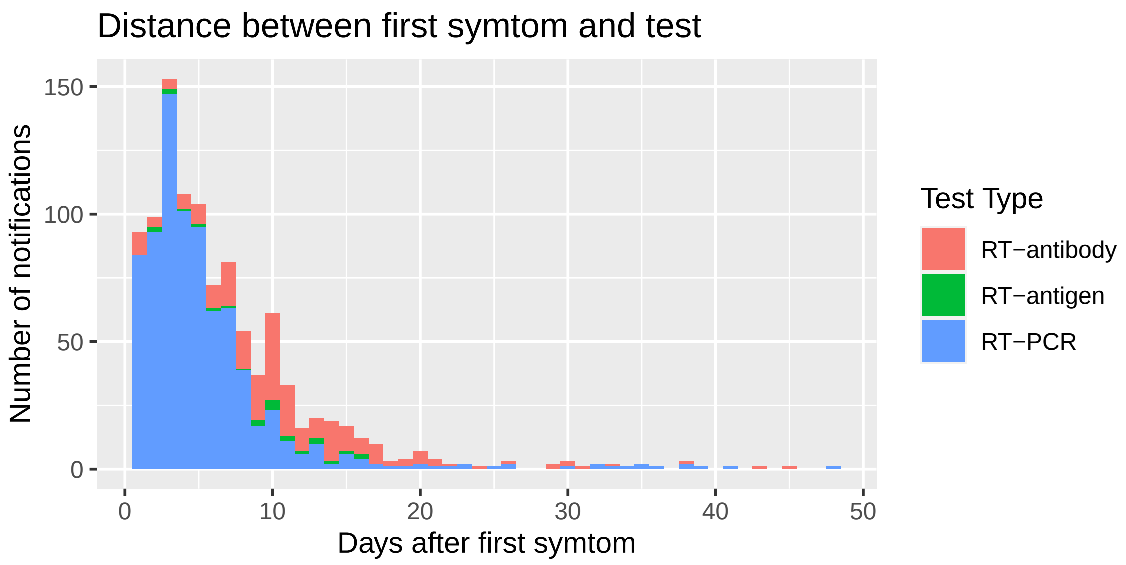
Now, considering the date when the cases were notified, the distribution of the notifications on the weeks is the following:

```r
# Notifications per week
workset %>% group_by(notweek) %>%
  mutate(n = length(notweek)) %>%
  ggplot(aes(notweek, n)) +
  geom_point(na.rm = TRUE) +
  geom_smooth(method = "loess", span = 0.3, method.args = list(degree = 1)) +
  ggtitle("Notification per Week") +
  ylab("Number of notifications") +
  xlab("Notification week")
```



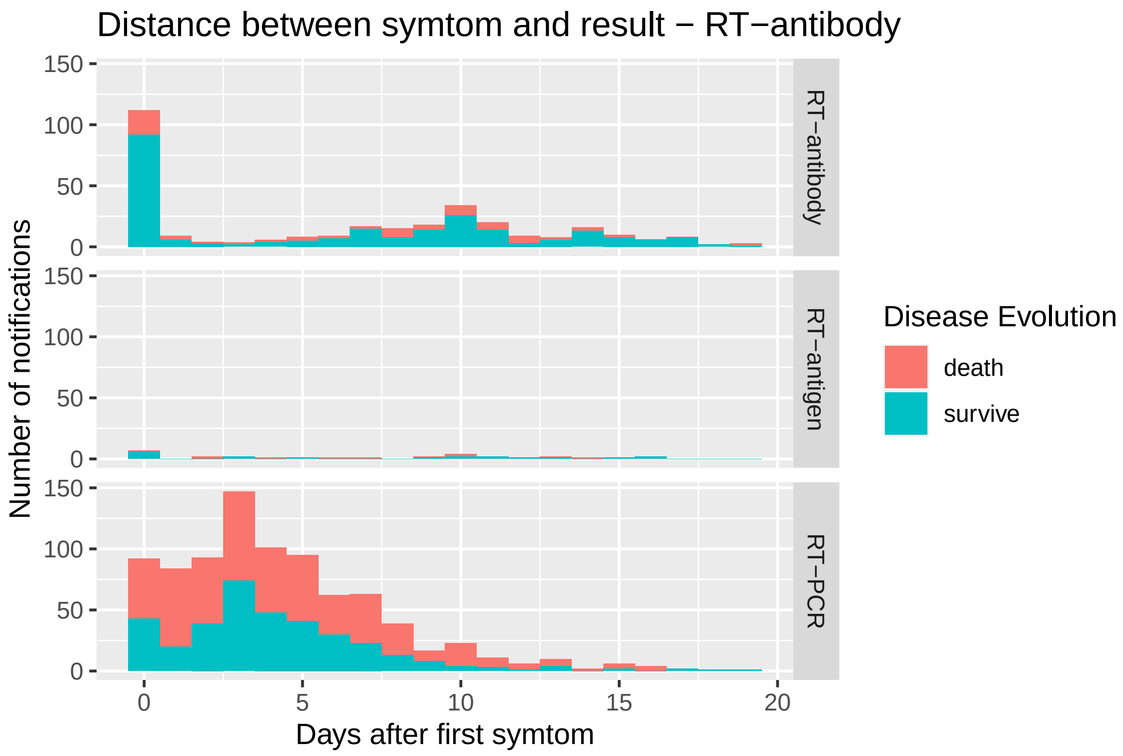
We see the characteristic increasing of notifications until week 20 to 25. After that, the trend is to decreasing the number of notifications per week. this behavior is a feature of this partial dataset. The most important parameter to perform the tests is the time when the material is colected, related to the symtom appearance. This time window can affect the test result, as a consequence affects also the correct treatment, and at the end, the evolution of the disease. Considering the three most used tests, RT-antigen, RT-antibody and RT-PCR, the distance between the symtoms starting date and the test date, in days, is shown below.

```
#Distance between symtom and test.
workset %>% mutate(distance = as.numeric(difftime(testdate, symtomdate, unit="days"))) %>%
  group_by(distance) %>%
  filter(distance < 50 & distance > 0) %>%
  ggplot(aes(distance, fill = testtype)) +
  geom_histogram(binwidth = 1) +
  ggtitle("Distance between first symtom and test") +
  ylab("Number of notifications") +
  xlab("Days after first symtom") +
  labs(fill = "Test Type")
```



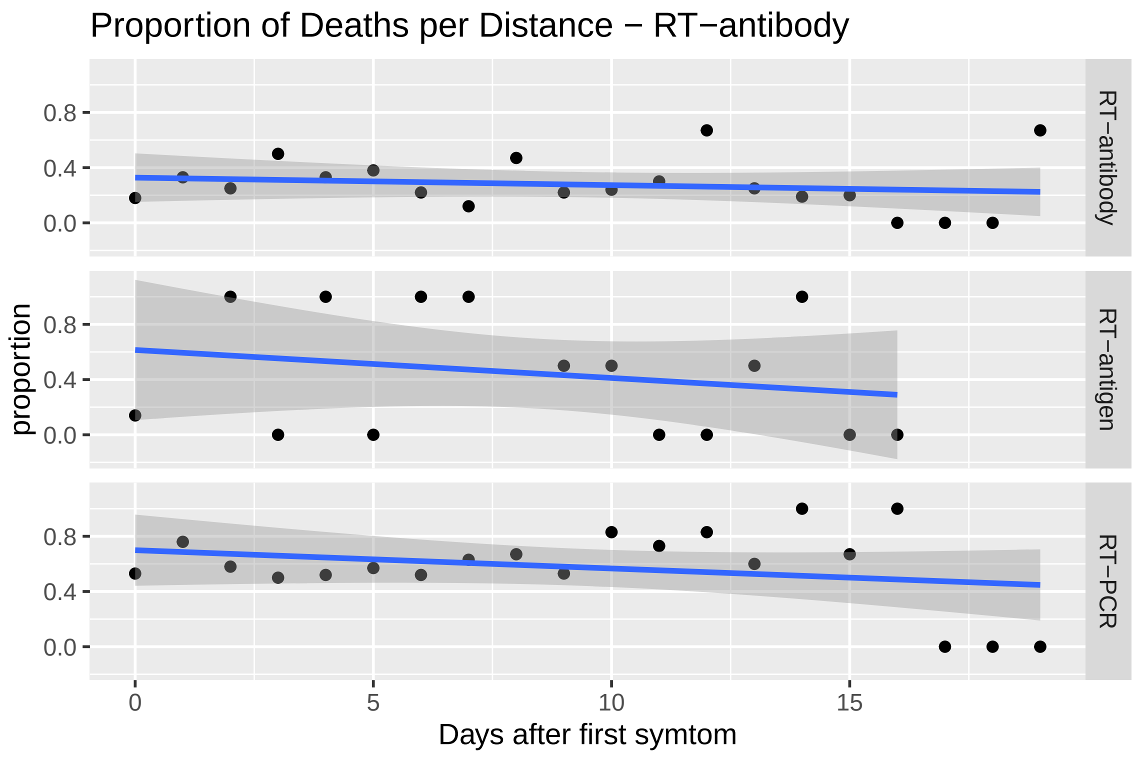
We see that, in most of the cases, both test were done in the first days after the symtoms, and part of the RT-antibody test were done later than the Rt-PCR.
According to [7] there is a diagnostic window to collect specimen for RT-PCR test. The results has better accuracy when the material is collected and the test is performed between 2nd and 5th day after the symtoms. Also according to [7], in case of serological tests, i.e. RT-antibody, it should be done not less than seven days after the symtoms.
We can see in the pictures below that, for the RT-PCR, the proportion of deaths in the first days is higher than in the RT-antibody test. The case where the test is done at the beginning of the disease has no clear indication on how will be the disease evolution. Since the RT-PCR is done at the beginning, more death cases appears here than other test which is done more days after the symtoms appearance.

```r
#Distance between symtom and test considering the disease evolution
workset %>% mutate(distance = as.numeric(difftime(testdate, symtomdate, unit="days"))) %>%
  group_by(distance) %>%
  filter(distance < 20 & distance > -1) %>%
  ggplot(aes(distance, fill = evolution)) +
  geom_histogram(binwidth = 1) +
  ggtitle("Distance between symtom and result - RT-antibody") +
  ylab("Number of notifications") +
  xlab("Days after first symtom") +
  labs(fill="Disease Evolution") +
  facet_grid(testtype~.)
```



In the two pictures below we can see the proportion of deaths over the time for the two most used test types. The trend is to decrease. this is explained by the fact that a test done after many days from the symtoms appearance, the worst period of the symtoms already passed and the risk to recover increases.

```r
# variation of the proportion of deaths by distance

workset %>%  mutate(distance = as.numeric(difftime(testdate, symtomdate, unit="days"))) %>%
  filter(distance < 20 & distance > -1) %>%
  group_by(distance, testtype) %>%
  summarize(proportion = round((mean(evolution =="death")), 2)) %>%
  ggplot(aes(distance, proportion, testtype), alpha = 0.2) +
  geom_point() +
  geom_smooth(method = "lm") +
  ggtitle("Proportion of Deaths per Distance - RT-antibody") +
  xlab("Days after first symtom") +
  facet_grid(testtype~.)
```

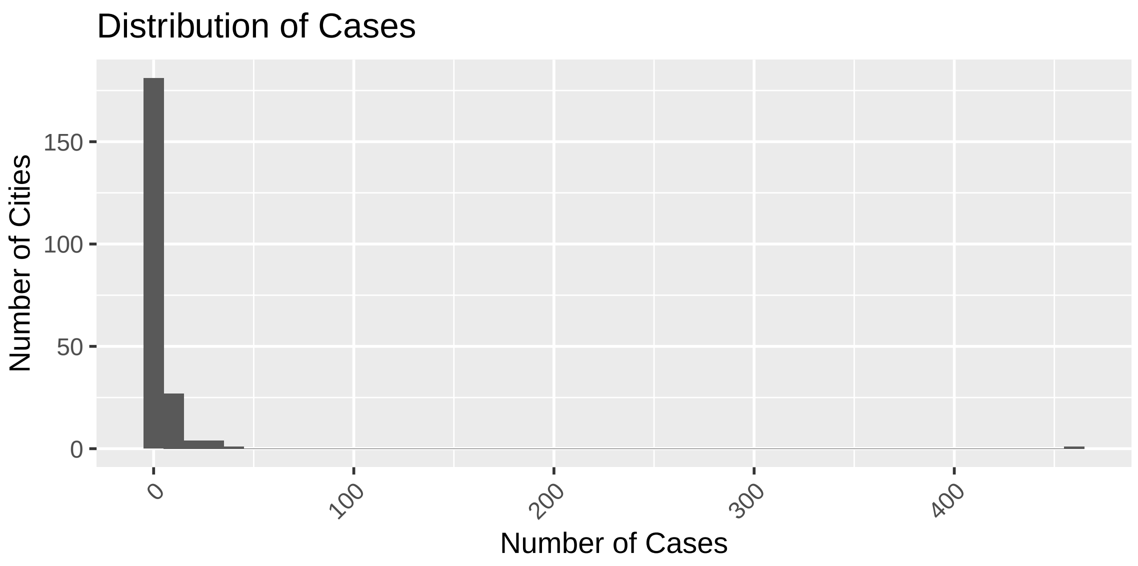


The timing related features seems to be not a good predictor.

2.3.7 Notification City

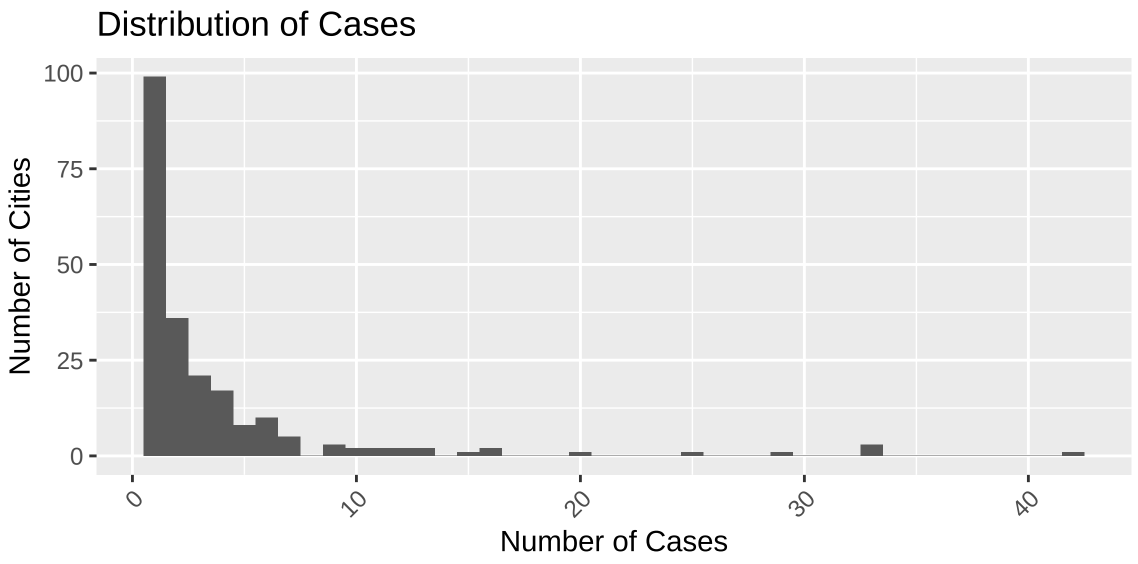
Now we analysed the city when the patient was notified. From the column "city" we can see distribution of the cases over the cities.

```
#distribution of the number of cases and cities
workset %>% group_by(city) %>%
  summarize(n = length(city)) %>%
  arrange(desc(n)) %>%
  ggplot(aes(n)) +
  geom_histogram(binwidth = 10) +
  ggtitle("Distribution of Cases") +
  ylab("Number of Cities") +
  xlab("Number of Cases") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



The picture above shows the distribution of the cities per cases. We see that most of the cities has less than 50 cases. Only one city, probabily São Paulo, has more than 400 cases. By removing this entry we have the picture below:

```
#distribution of the number of cases and cities, less than 400 cases
workset %>% group_by(city) %>%
  summarize(n = length(city)) %>%
  filter(n < 400) %>%
  arrange(desc(n)) %>%
  ggplot(aes(n)) +
  geom_histogram(binwidth = 1) +
  ggtitle("Distribution of Cases") +
  ylab("Number of Cities") +
  xlab("Number of Cases") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



That is the absolute number of cases and not the cases per million, because the dataset has no information about the total number of citizens for each city. Most of the cities has low number of cases, less than 10. We can see below the 20 cities with more cases.

```r
#20 cities with more cases
workset %>% group_by(city) %>%
  summarize(n = length(city))  %>%
  arrange(desc(n)) %>%
  head(20) %>%
  knitr::kable()
```

city	n
São Paulo	460
Santo André	42
Campinas	33
Ribeirão Preto	33
São Bernardo do Campo	33
São José do Rio Preto	29
Diadema	25
Mogi das Cruzes	20
Guarulhos	16
Sorocaba	16
Santos	15
Cubatão	13
Lençóis Paulista	13
Botucatu	12
Suzano	12
Piracicaba	11
Praia Grande	11
Bauru	10
Itapetininga	10
Araçatuba	9

Ordering according the number of deaths we have:

```r
#20 cities with more deaths
workset %>% select(city, evolution) %>%
  group_by(city) %>%
  summarize(cases = n(),
            deaths = sum(evolution == "death"),
            deathprop = round(mean(evolution == "death"), 2)) %>%
  arrange(desc(deaths)) %>%
  head(20) %>%
  knitr::kable()
```

city	cases	deaths	deathprop
São Paulo	460	310	0.67
Santo André	42	34	0.81
Diadema	25	25	1.00
São Bernardo do Campo	33	19	0.58
Sorocaba	16	9	0.56
Suzano	12	9	0.75
Cubatão	13	7	0.54
Ibiúna	7	7	1.00

city	cases	deaths	deathprop
Santos	15	7	0.47
Caraguatatuba	7	6	0.86
Ipuã	6	6	1.00
Itapetininga	10	6	0.60
Ribeirão Preto	33	6	0.18
Botucatu	12	5	0.42
Guarujá	6	4	0.67
Praia Grande	11	4	0.36
Sumaré	6	4	0.67
Birigui	3	3	1.00
Campos do Jordão	3	3	1.00
Francisco Morato	4	3	0.75

taking again the cities with most cases, now checking the proportion of deaths and positive results, we see below that there is no clear correlation between the size of the city and the proportions.

```r
#20 cities with more cases and evolution of the disease.
cities <- workset %>% select(city, testres, evolution) %>%
  group_by(city) %>%
  summarize(n = length(city),
         prop_death = round(mean(evolution =="death"), 2),
         prop_pos = round(mean(testres =="positive"),2)) %>%
  arrange(desc(n)) %>%
  head(20)
cities %>% knitr::kable()
```

city	n	prop_death	prop_pos
São Paulo	460	0.67	0.68
Santo André	42	0.81	0.38
Campinas	33	0.03	0.48
Ribeirão Preto	33	0.18	0.55
São Bernardo do Campo	33	0.58	0.45
São José do Rio Preto	29	0.00	0.10
Diadema	25	1.00	0.52
Mogi das Cruzes	20	0.10	0.10
Guarulhos	16	0.00	0.19
Sorocaba	16	0.56	0.75
Santos	15	0.47	0.73
Cubatão	13	0.54	0.77
Lençóis Paulista	13	0.00	0.08
Botucatu	12	0.42	0.25
Suzano	12	0.75	0.42
Piracicaba	11	0.00	0.45
Praia Grande	11	0.36	0.91
Bauru	10	0.10	0.20
Itapetininga	10	0.60	0.40
Araçatuba	9	0.11	0.67

The correlation between the number of cases of agiven city and the proportion of positive test results is 0.202 . And the correlation between the number of cases of a given city and the proportion of deaths is 0.242. The information about the city seems to be not a good predictor.

2.3.8 Conditions

Let's check the previous conditions of each patient. Considering the column "conditions" categorized in "yes when the patient has any previous disease or commorbities, and"no" when there is no previous conditions, we have:

```
#disease evolution related with previous conditions
workset %>% mutate(commorbidities = ifelse(conditions =="no_previous_conditions", "no", "yes")) %>%
  group_by(commorbidities) %>%
  summarize(proportion_of_death = mean(evolution =="death") %>% round(2),
            proportion_of_survival = mean(evolution =="survive") %>% round(2)) %>%
  knitr::kable()
```

commorbidities	proportion_of_death	proportion_of_survival
no	0.36	0.64
yes	0.77	0.23

The previous conditions or diseases increases the change of death. The most common comorbidity combinations in the dataset is shown below.

```
#Appearance of each commorbities combination
workset %>% select(conditions) %>%
  group_by(conditions) %>%
  summarize( n = n()) %>%
  arrange(desc(n)) %>%
  head(10) %>%
  knitr::kable()
```

conditions	n
no_previous_conditions	893
heart_disease	109
diabetes, heart_disease	76
diabetes	72
respiratory_disease	45
immunodeficiency	24
kidney_disease	12
diabetes, kidney_disease	9
heart_disease, respiratory_disease	7
diabetes, respiratory_disease	6

We see that heart_disease, diabetes and respiratory disease are the most common. Considering the COVID19 evolution, filtering commorbidities combination with more than 10 appearances, and ordering from the highest proportin of deaths, we have the following:

```
#commorbities combination versus COVID19 evolution
workset %>% select(conditions, evolution) %>%
  group_by(conditions) %>%
  summarize(n = n(),
            prop_death = mean(evolution =="death") %>% round(2),
            prop_survival = mean(evolution =="survive") %>% round(2)) %>%
  arrange(desc(prop_death)) %>%
  filter(n > 10) %>%
  head(10) %>%
  knitr::kable()
```

conditions	n	prop_death	prop_survival
diabetes, heart_disease	76	0.88	0.12
kidney_disease	12	0.83	0.17
immunodeficiency	24	0.79	0.21
diabetes	72	0.76	0.24
heart_disease	109	0.69	0.31
respiratory_disease	45	0.64	0.36
no_previous_conditions	893	0.36	0.64

Again, we see that previous disease increases the risk of death by COVID19. The column "conditions" was considered an important predictor.

2.3.9 Final Classification

From the column "finalclass" we have the final classification of each case, whether is valid or not.
The final classification was resumed in two possibilities: "confirmed", as positive, or "discarted", if negative. The table below shows the confusion matrix between the test result and the final classification. The numbers before the test results guarantee the correct order of the table rows.

```
#checking the final classification of each case.
workset %>% select(testres, finalclass) %>%
  mutate(testres = ifelse(testres=="positive", "1_positive", "2_negative")) %>%
  table() %>%
  knitr::kable()
```

	confirmed	discarted
1_positive	684	3
2_negative	43	548

From the table above we can get some important indicators. Before to explore the indicators we remembered the meaning of each cell. The TP(True Positive) cases are those who were tested positive and confirmed as positive later. The FP(False Positive) cases are those who were tested positive and discarted later, that means, there was not a positive case. The FN(False Negative) cases are those who were tested negative and confirmed as positive later. This is the whorst situation for the public health, when a given patient has the disease and the test result does not indicate this. The TN(True Negative) cases are those who were tested negative and discarted later, that means, it was really a negative case.
The table with the definitions above is the following:

```
#table with cases definition
cmtable <- data.frame(c("disease","notdisease"), c("TP", "FN"), c("FP", "TN"))
colnames(cmtable) <- c("", "confirmed", "discarted")
cmtable %>% knitr::kable()
```

	confirmed	discarted
disease	TP	FP
notdisease	FN	TN

First indicator is the *accuracy*, which is an important indicator when we are predicting categorical data. The formula measures the proportion of correct predictions regarding to the whole data:

$$\text{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN}$$

However, only the accuracy may not give the correct evaluation of the prediction model mainly in case of biased data, when for example we have more samples of one output than another, in a given pair of possible outputs. in order to improve our analysis, we checked also the sensitivity and specificity. The sensitivity , or TPR(True Positive Rate), also called *Recall*, means the proportion of predicted true positive cases among all confirmed as positive, and is given by the formula below:

$$\text{Sensitivity} = \frac{TP}{TP + FN}$$

The specificity, or TNR (True Negative Rate) means the proportion of predicted true negative cases among all discarted cases, and is given by the formula below:

$$\text{Specificity} = \frac{TN}{FP + TN}$$

The specificity is also called precision, or PPV(Positive Prediction Value) and can be calculated also as follow:

$$\text{Precision} = \frac{TP}{TP + FP}$$

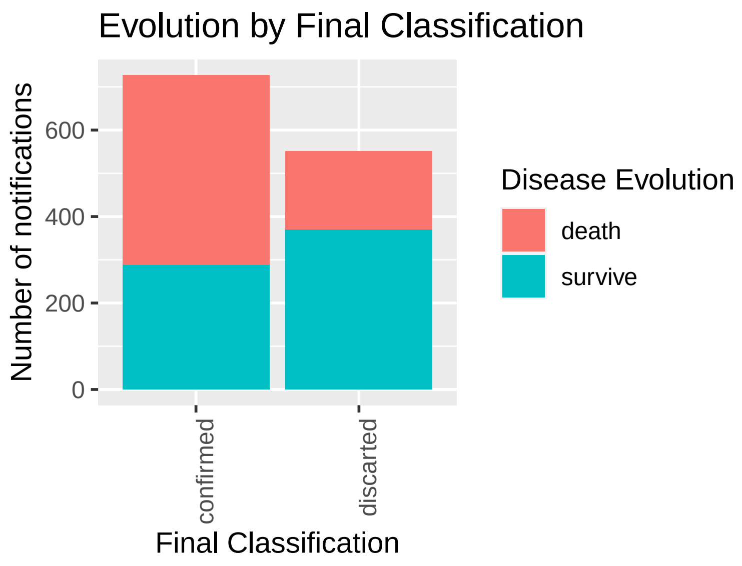
The accuracy, sensitivity, specificity, and precision for each test type was calculated by the code below:

```r
#calculating the accuracy, sensitivity, specificity and precision for each test
workset %>% select(testres, testtype, finalclass) %>%
  group_by(testtype) %>%
  summarize(accuracy = round(((sum(finalclass =="confirmed" & testres=="positive") +
          sum(finalclass =="discarted" & testres =="negative"))/length(testtype)), 4),
            sensitivity = round((sum(finalclass =="confirmed" & testres=="positive")/
                            sum(finalclass =="confirmed")), 4),
            specificity = round((sum(finalclass =="discarted" & testres=="negative")/
                            sum(finalclass =="discarted")), 4),
            precision = round((sum(finalclass =="confirmed" & testres=="positive")/
                          sum(testres =="positive")), 4)) %>%
  knitr::kable()
```

testtype	accuracy	sensitivity	specificity	precision
RT-antibody	0.9569	0.9116	0.9900	0.9853
RT-antigen	0.9355	0.9048	1.0000	1.0000
RT-PCR	0.9677	0.9499	0.9971	0.9981

Looking at the numbers we can say that the results are good. However this is a result of a downsized dataset focused on the evolution of the CODIV19. The distribution of death and survivals among the cases classification is show below:

```r
#evolution od the cases according to the final classification
workset %>%
  group_by(finalclass) %>%
  ggplot(aes(finalclass, fill = evolution)) +
  geom_bar(stat = "count", show.legend = TRUE) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  ggtitle("Evolution by Final Classification ") +
  ylab("Number of notifications") +
  xlab("Final Classification") +
  labs(fill="Disease Evolution")
```



We see that there are deaths among the negative cases. That amount of deaths can be nothing related to the COVID19? That dataset cannot answer this question. The column "finalclass" was removed from the workset.
In this report we checked the accuracy, sensitivity and specificity of the prediction models as comparison indicators.

2.3.10 Selecting the Predictors

After checking all columns of the dataset and extracting the important information for prediction purposes we selected the columns that could be a good predictor. Before, we checked the correlatinon or association between the features. First step was to check the binary features. We removed the non binary columns and checked features against the target variable "evolution". We applied the chi-squared test, calculated the Phi Coefficient, and also calculated the correlation with Pearson method.

```
# check the association/correlation between the binary predictors
#select only columns with binary information
chworkset <- workset %>% select(-notdate, -symtomdate, -symtom, -conditions, -testdate,
                                -testtype, -city, -age, - finalclass, -notweek)

#change factors to numeric
chworkset <- mutate_if(chworkset, is.factor, as.numeric)

#chisq test and phi calculation
chisq <- matrix(1, 13, 5)
name <- colnames(chworkset)
for(i in 1:13){
  tab <- table(chworkset$evolution, as.matrix(chworkset[,i]))
  mat <- matrix(as.numeric(tab), nrow = nrow(tab), ncol = ncol(tab))
  info <- chisq.test(mat, correct = FALSE)
  chisq[i, 1] <- "evolution"
  chisq[i, 2] <- name[i]
  chisq[i, 3] <- info$p.value
  chisq[i, 4] <- sqrt(info$statistic/nrow(chworkset))
  chisq[i, 5] <- cor(chworkset$evolution, chworkset[,i])
}
colnames(chisq) <- c("target", "feature", "p-value", "phi-coef", "pearson")
chisq %>% knitr::kable()
```

target	feature	p-value	phi-coef	pearson
evolution	testres	4.0402554734806e-26	0.295713233554019	-0.295713233554019
evolution	gender	6.35942047668541e-09	0.16243812259874	-0.16243812259874
evolution	evolution	6.82592539157279e-280	1	1
evolution	cought	4.845285967196e-06	0.127873824923831	-0.127873824923831
evolution	fever	0.000182508956116166	0.104675911671977	-0.104675911671977
evolution	sorethroat	1.3829173383445e-08	0.158758863832194	0.158758863832194
evolution	headache	0.0162450267690964	0.0672285972538465	0.0672285972538465
evolution	dyspnoea	5.0943123899972e-33	0.334837254077309	-0.334837254077309
evolution	tastedisorder	0.344394775436769	0.0264486867670881	0.0264486867670881
evolution	olfactorydisorder	0.453522639695486	0.0209670917693575	0.0209670917693575
evolution	runnynose	0.00342276594188937	0.0818754234227394	0.0818754234227394
evolution	asymtomatic	0.0874458729764639	0.0478061313433051	0.0478061313433051
evolution	others	2.42104280242961e-13	0.204850108789656	0.204850108789656

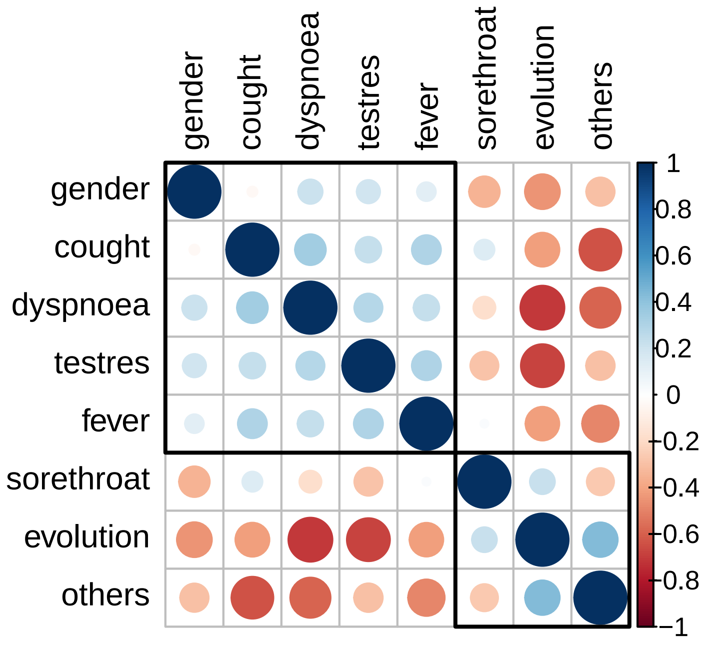
We see that the Pearson correlation values are equivalent as the phi-coefficient. The purpose is to decide if the correlation is low, medium or high. That means we can use a correlation plot as an aproximation of the correlation between binary variables. By checking the proportion of appearance of some binary features, mainly the symtoms, we see than in some cases the column has almost no "1's". We removed those features: headache, tastedisorder, olfactorydisorder, and asymtomatic.

```
#select only columns with binary information
chworkset <- workset %>% select(-notdate, -symtomdate, -symtom, -conditions,
                                -testdate, -testtype, -city, -age, - finalclass,
                                -notweek)

#change factors to numeric
chworkset <- mutate_if(chworkset, is.factor, as.numeric)

#checking and removing non common features
colMeans(chworkset) %>% knitr::kable()
```

	x
testres	1.5375587
gender	1.4929577
evolution	1.5140845
cought	0.5758998
fever	0.4107981
sorethroat	0.2511737
headache	0.0125196
dyspnoea	0.3364632
tastedisorder	0.0031299

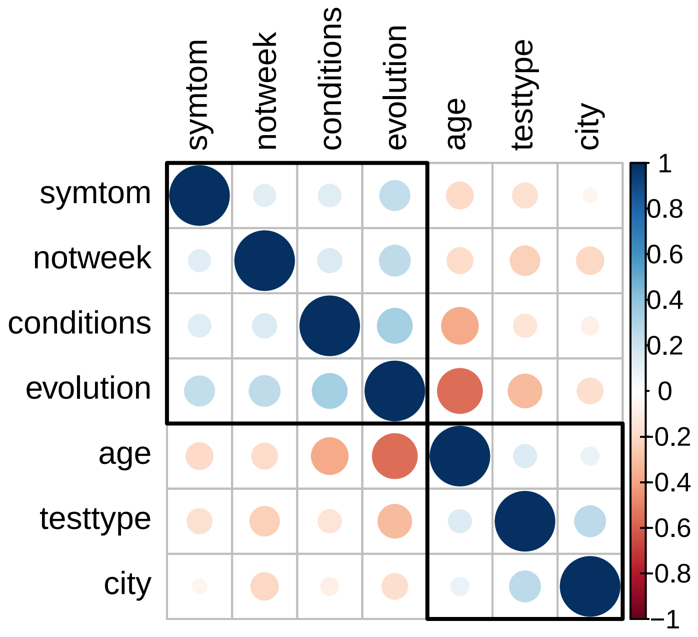
	x
olfactorydisorder	0.0046948
runnynose	0.0070423
asymtomatic	0.0117371
others	0.5852895

```
chworkset <- chworkset %>% select(-headache, -tastedisorder, -olfactorydisorder,
                                  - runnynose, -asymtomatic)

#correlation for binary predictors approximated by pearson.
T <- cor(chworkset)
corrplot::corrplot(cor(T), method = "circle", order = "hclust", addrect = 2,
                   tl.col = "black", hclust.method = "average")
```



Most of the features presents moderate to weak relationship. Some of them presented no relationship, and those will be not consider in the prediction models.
For non binary features, we checked a correlation plot only to certify the low/medium association with the target varilable.

```
#Correlation Matrix for non binary predictors
T <- mutate_if(workset, is.factor, as.numeric)
T <- mutate_if(T, is.POSIXct, as.numeric)
T <- T %>% select(symtom, conditions, testtype, city, age, notweek, evolution)
corrplot::corrplot(cor(T), method = "circle", order = "hclust", addrect = 2,
                   tl.col = "black", hclust.method = "average")
```



As we stated during the features analysis, we will not consider the features related to the timing, the "city", the column "symtom", and the following symtoms: "headache", "tastedisorder", "olfactorydisorder", "runnynose", "asymtomatic" and "others".Also the column "finalclass

```
#removing predictors not correlated with the target variable
workset <- workset %>% select(-notdate, -symtomdate, -symtom, -testdate, -city, -notweek,
                              -headache, -tastedisorder, -olfactorydisorder, -runnynose,
                              -asymtomatic, -others, -finalclass)
```

The following features remained in the dataset: conditions, testtype, testres, gender, age, the target variable "evolution", and the four symtoms: cought, fever, sorethroat and dyspnoea.

With the analysis above we covered all columns of the workset. We analysed the test results against the gender, age, test type, symtoms, timeline issues, city, previous conditions, and the final classification of the test. Also we selected predictors and removed not used columns to help us to achieve better predictions. That is the topic of the next sections.

2.4 The Train and Test sets

Considering what was discussed earlier, we divided the workset in two sets: the trainset and the testset. The trainset was created to be the reference for model training, where we know the results we want to predict: the test results. The testset was created to be the checking dataset, where the proposed models was tested against the target variable, by checking the accuracy, sensitivity, specificity and others. Also we splitted the output variable from the testset:

```
#Creating the train and test sets
set.seed(1, sample.kind="Rounding")
test_index <- createDataPartition(workset$evolution, times = 1, p = 0.5, list = FALSE)
trainset <- workset[-test_index,]
testset <- workset[test_index,]
```

```r
#filtering the test set
testset <- testset %>%
  semi_join(trainset, by = "conditions") %>%
  semi_join(trainset, by = "testtype") %>%
  semi_join(trainset, by = "testres") %>%
  semi_join(trainset, by = "gender") %>%
  semi_join(trainset, by = "age") %>%
  semi_join(trainset, by = "evolution") %>%
  semi_join(trainset, by = "cought") %>%
  semi_join(trainset, by = "fever") %>%
  semi_join(trainset, by = "sorethroat") %>%
  semi_join(trainset, by = "dyspnoea")

#splitting the predictors and the target variable
testset.y <- testset$evolution
testset.x <- testset %>% select(-evolution)
```

The splitting was defined as 50% for training and 50% for testing, with aproximatelly 640 observations each. This splitting guarantees a good size for training and test sets, allowing the trainset splitting for cross validation, for example. Other splitting percentages, 80%-20% or 70%-30% i.e., could lead a low number of observations in testset, which could lower the accuracy.

2.5 Model Fitting

Based on the previous sections, some features were selected as predictors. In order to evaluate each model, the accuracy, sensitivity ans specificity were checked.

2.5.1 Guessing

The first try was to guess the outcome, the disease evolution. The forecast is to reach an accuracy close than 50%, because there is a guessing. The parameters Accuracy, Sensitivity, Specificity, Precision and F1 Score was shown in a table.

```r
# 1 - Guessing the outcome
set.seed(1, sample.kind = "Rounding")
y_hat_guess <- sample(c("death", "survive"), nrow(testset), replace = TRUE)  %>%
  factor(levels = levels(testset.y))
acc <- confusionMatrix(y_hat_guess, testset.y)$overall["Accuracy"] %>% round(4)
sens <- confusionMatrix(y_hat_guess, testset.y)$byClass["Sensitivity"] %>% round(4)
spec <- confusionMatrix(y_hat_guess, testset.y)$byClass["Specificity"] %>% round(4)
prec <- confusionMatrix(y_hat_guess, testset.y)$byClass["Precision"] %>% round(4)
f1 <- confusionMatrix(y_hat_guess, testset.y)$byClass["F1"] %>% round(4)

#creating an output table
test_results <- tibble(method = "Guessing", Accuracy = acc, Sensitivity = sens,
                       Specificity = spec,
                       Precision = prec, F1 = f1)
test_results %>% knitr::kable()
```

method	Accuracy	Sensitivity	Specificity	Precision	F1
Guessing	0.4801	0.5017	0.4599	0.4648	0.4825

A expected, the accuracy is close to 50%.

2.5.2 Logistic Regression

Next step was to apply the logistic regression on the trainset, which is one of the most reccomended method for categorical binary prediction. The first try here, was considering the most correlated features, which is age and dyspnoea. The results table was updated with the logistic regression results.

```
#2 - logistic regession with more correlated predictors: age and dyspnoea
set.seed(1, sample.kind = "Rounding")
fit_glm <- glm(evolution ~ age + dyspnoea, data = trainset, family = "binomial")
p_hat_glm <- predict(fit_glm, testset.x, type = "response")
y_hat_glm <- ifelse(p_hat_glm < 0.5, "death", "survive") %>%
  factor(levels = levels(testset.y))
acc <- confusionMatrix(y_hat_glm, testset.y)$overall["Accuracy"] %>% round(4)
sens <- confusionMatrix(y_hat_glm, testset.y)$byClass["Sensitivity"] %>% round(4)
spec <- confusionMatrix(y_hat_glm, testset.y)$byClass["Specificity"] %>% round(4)
prec <- confusionMatrix(y_hat_glm, testset.y)$byClass["Precision"] %>% round(4)
f1 <- confusionMatrix(y_hat_glm, testset.y)$byClass["F1"] %>% round(4)


#updating the output table
test_results <- bind_rows(test_results,
                          data_frame(method="Logistic Regression with Age and Dyspnoea ",
                                     Accuracy = acc, Sensitivity = sens,
                                     Specificity = spec,
                                     Precision = prec, F1 = f1))
test_results[2,] %>% knitr::kable()
```

method	Accuracy	Sensitivity	Specificity	Precision	F1
Logistic Regression with Age and Dyspnoea	0.7719	0.7492	0.7932	0.7721	0.7605

We see that the accuracy is 0.77.

The second try with Logistic regression was to consider all features of the trainset:

```
#3- logistic regession with  all predictors.
set.seed(1, sample.kind = "Rounding")
fit_glm_all <- glm(evolution~., data = trainset, family = "binomial")
p_hat_glm_all <- predict(fit_glm_all, testset.x, type = "response")
y_hat_glm_all <- ifelse(p_hat_glm_all<0.5,"death","survive") %>%
  factor(levels=levels(testset.y))
acc <- confusionMatrix(y_hat_glm_all, testset.y)$overall["Accuracy"] %>% round(4)
sens <- confusionMatrix(y_hat_glm_all, testset.y)$byClass["Sensitivity"] %>% round(4)
spec <- confusionMatrix(y_hat_glm_all, testset.y)$byClass["Specificity"] %>% round(4)
prec <- confusionMatrix(y_hat_glm_all, testset.y)$byClass["Precision"] %>% round(4)
f1 <- confusionMatrix(y_hat_glm_all, testset.y)$byClass["F1"] %>% round(4)


#updating the output table
test_results <- bind_rows(test_results,
                          data_frame(method="Logistic Regression with All Features",
                                     Accuracy = acc, Sensitivity = sens,
                                     Specificity = spec,
                                     Precision = prec, F1 = f1))
test_results[3,] %>% knitr::kable()
```

method	Accuracy	Sensitivity	Specificity	Precision	F1
Logistic Regression with All Features	0.8134	0.7855	0.8395	0.8207	0.8027

The paramenters increased a little. Accuracy reached 0.81. We implemented, in both Logistic regression tries, a cuttoff point at 0.5, defining that if the odd is lower than 0.5 that means a "death", and "survive" otherwise.

2.5.3 KNN

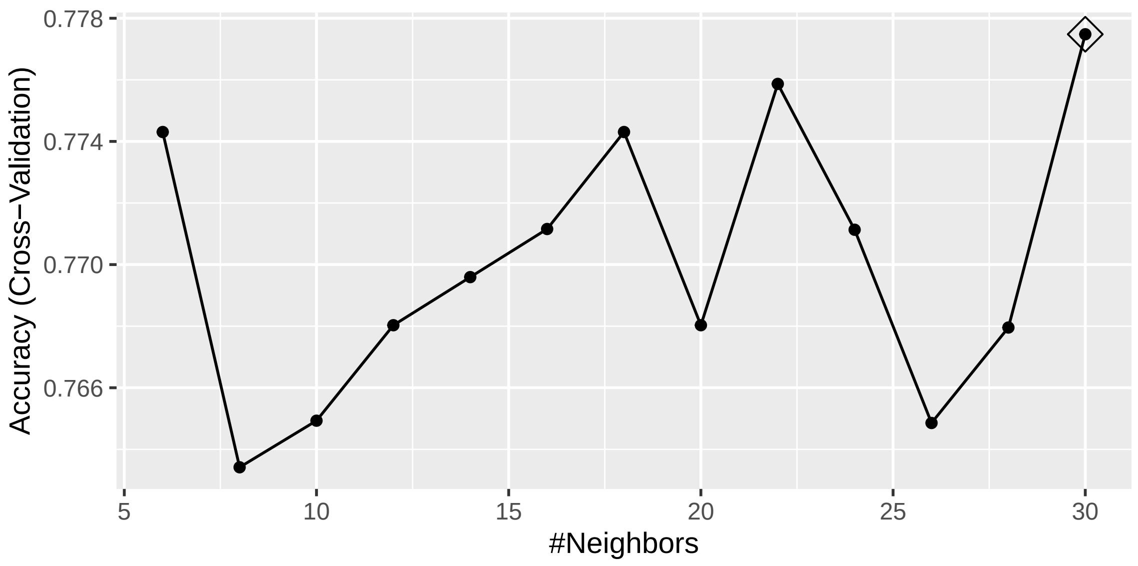
Now we try K-Nearest Neighbors algorithm with caret package.

```r
# 4 - KNN
set.seed(1, sample.kind = "Rounding")
knn_fit <- train(evolution~., method = "knn" , data = trainset)
y_hat_knn <- predict(knn_fit, testset.x, type = "raw")
acc <- confusionMatrix(y_hat_knn, testset.y)$overall["Accuracy"] %>% round(4)
sens <- confusionMatrix(y_hat_knn, testset.y)$byClass["Sensitivity"] %>% round(4)
spec <- confusionMatrix(y_hat_knn, testset.y)$byClass["Specificity"] %>% round(4)
prec <- confusionMatrix(y_hat_knn, testset.y)$byClass["Precision"] %>% round(4)
f1 <- confusionMatrix(y_hat_knn, testset.y)$byClass["F1"] %>% round(4)

#updating the output table
test_results <- bind_rows(test_results,
                          data_frame(method="K-Nearest Neighbors",
                                     Accuracy = acc, Sensitivity = sens,
                                     Specificity = spec,
                                     Precision = prec, F1 = f1))
test_results[4,] %>% knitr::kable()
```

method	Accuracy	Sensitivity	Specificity	Precision	F1
K-Nearest Neighbors	0.7943	0.7261	0.858	0.8271	0.7733

The parameters values decreased a little, except by the Specificity which reaches 0.84. Implementing a Cross Validation to tune the KNN algorithm we have the following results:

```r
#5 - Cross Validation to choose the best k parameter
set.seed(1, sample.kind = "Rounding")
control <- trainControl(method = "cv", number = 10, p = .9)
knn_fit <- train(evolution~., method = "knn", data = trainset,
                 tuneGrid = data.frame(k = seq(6, 30, 2)),
                 trControl = control)
ggplot(knn_fit, highlight = TRUE) # k = 22 is the best choice
```



```r
knn_fit$bestTune #22
```

```
##     k
## 13 30
```

```r
y_hat_knn_cv<- predict(knn_fit, testset.x, type = "raw")
acc <- confusionMatrix(y_hat_knn_cv, testset.y)$overall["Accuracy"] %>% round(4)
sens <- confusionMatrix(y_hat_knn_cv, testset.y)$byClass["Sensitivity"] %>% round(4)
spec <- confusionMatrix(y_hat_knn_cv, testset.y)$byClass["Specificity"] %>% round(4)
prec <- confusionMatrix(y_hat_knn_cv, testset.y)$byClass["Precision"] %>% round(4)
f1 <- confusionMatrix(y_hat_knn_cv, testset.y)$byClass["F1"] %>% round(4)

#updating the output table
test_results <- bind_rows(test_results,
                          data_frame(method="K-Nearest Neighbors Cross Validation",
                                     Accuracy = acc, Sensitivity = sens,
                                     Specificity = spec,
                                     Precision = prec, F1 = f1))
test_results[5,] %>% knitr::kable()
```

method	Accuracy	Sensitivity	Specificity	Precision	F1
K-Nearest Neighbors Cross Validation	0.7624	0.6667	0.8519	0.808	0.7306

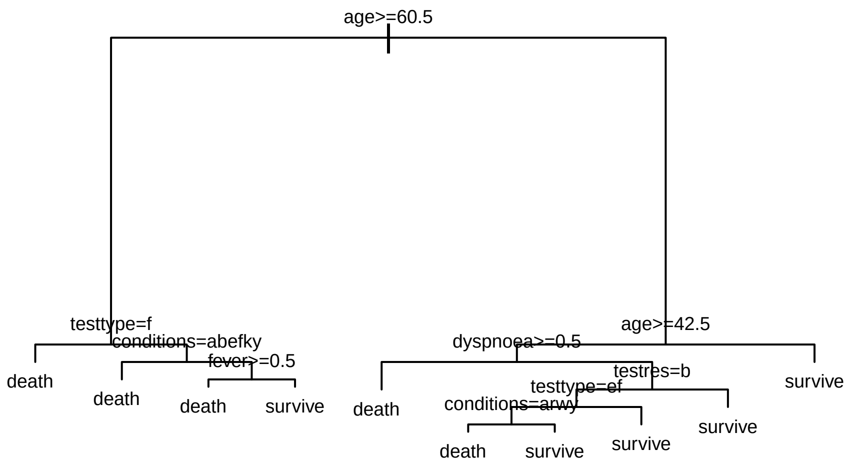
The best K value was 30, however the results did not changed much.

2.5.4 Classification Tree - rpart

Next try was to implement some Classification Algorithms. A Classification Tree with rpart function was set up manually with a complexity parameter cp = 0.08 and minsplit 28:

```r
#6 - Classification Tree (rpart)
set.seed(1, sample.kind = "Rounding")
fit_tree <- rpart(evolution ~ ., data = trainset,
                  control = rpart.control(cp = 0.008, minsplit = 28), method ="class")
p_hat_tree <- as.data.frame(predict(fit_tree, testset.x))
```

```r
y_hat_tree <- p_hat_tree %>%mutate(pred = ifelse(death > 0.5, "death", "survive")) %>%
  select(pred)
y_hat_tree <- as.matrix(y_hat_tree)
y_hat_tree <- as.factor(y_hat_tree)
plot(fit_tree, margin = 0.1)
text(fit_tree, cex = 0.6)
```

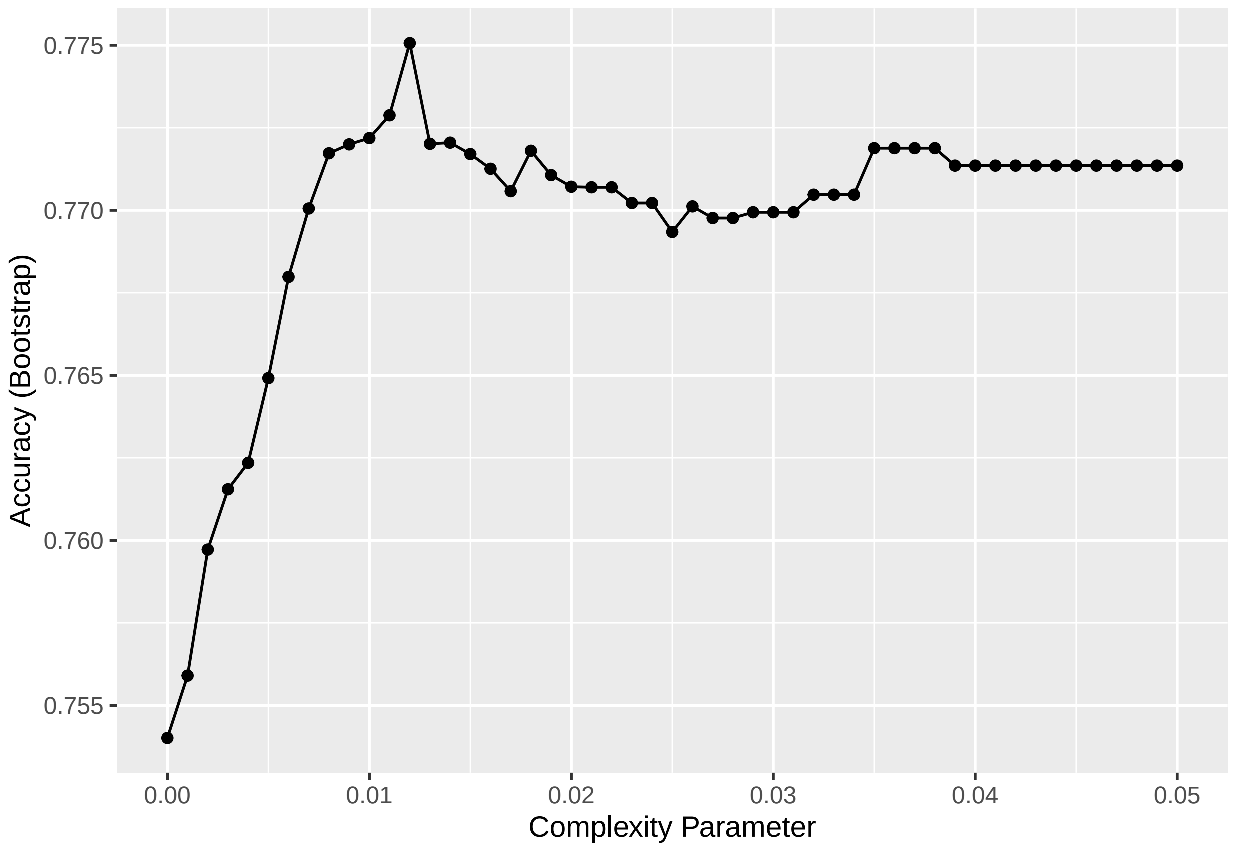


```r
#calculating the output parameters
acc <- confusionMatrix(y_hat_tree, testset.y)$overall["Accuracy"] %>% round(4)
sens <- confusionMatrix(y_hat_tree, testset.y)$byClass["Sensitivity"] %>% round(4)
spec <- confusionMatrix(y_hat_tree, testset.y)$byClass["Specificity"] %>% round(4)
prec <- confusionMatrix(y_hat_tree, testset.y)$byClass["Precision"] %>% round(4)
f1 <- confusionMatrix(y_hat_tree, testset.y)$byClass["F1"] %>% round(4)

#updating the output table
test_results <- bind_rows(test_results,
                          data_frame(method="Classification Tree - rpart()",
                                     Accuracy = acc, Sensitivity = sens,
                                     Specificity = spec,
                                     Precision = prec, F1 = f1))
test_results[6,] %>% knitr::kable()
```

method	Accuracy	Sensitivity	Specificity	Precision	F1
Classification Tree - rpart()	0.7927	0.7525	0.8302	0.8057	0.7782

We see tha the results decreases a little. Applying the Classification Tree from the caret package and tuning the complexity parameter we have the following:

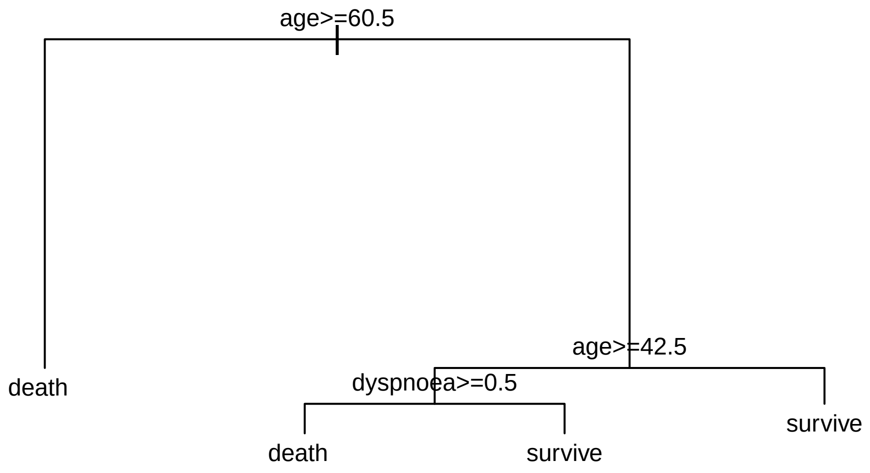
```
#7 - Classification Tree (Caret)
set.seed(1, sample.kind = "Rounding")
fit_tree_caret <- train(evolution ~ ., data = trainset, method = "rpart",
                  tuneGrid = data.frame(cp = seq(0, 0.05, 0.001)),
                  control = rpart.control(minsplit = 20, minbucket = 7))
ggplot(fit_tree_caret)
```



```
fit_tree_caret$bestTune
```

```
##       cp
## 13 0.012
```

```
y_hat_tree_caret <- predict(fit_tree_caret, testset.x)
plot(fit_tree_caret$finalModel, margin = 0.1)
text(fit_tree_caret$finalModel, cex = 0.75)
```



```r
#calculating the output parameters
acc <- confusionMatrix(y_hat_tree_caret, testset.y)$overall["Accuracy"] %>% round(4)
sens <- confusionMatrix(y_hat_tree_caret, testset.y)$byClass["Sensitivity"] %>% round(4)
spec <- confusionMatrix(y_hat_tree_caret, testset.y)$byClass["Specificity"] %>% round(4)
prec <- confusionMatrix(y_hat_tree_caret, testset.y)$byClass["Precision"] %>% round(4)
f1 <- confusionMatrix(y_hat_tree_caret, testset.y)$byClass["F1"] %>% round(4)

#updating the output table
test_results <- bind_rows(test_results,
                          data_frame(method="Classification Tree - caret",
                                     Accuracy = acc, Sensitivity = sens,
                                     Specificity = spec,
                                     Precision = prec, F1 = f1))
test_results[7, ] %>% knitr::kable()
```

method	Accuracy	Sensitivity	Specificity	Precision	F1
Classification Tree - caret	0.7671	0.7063	0.8241	0.7897	0.7456

The results of the classification trees did not increased the accuracy of the prediction. We see that the algorithm considered “death” after dividing the age in older than 60.5, and then from the younger than 60.5, splitted the older than 42.5 with dyspnoea.

2.5.5 XGBoost

As a final try we implemented the XGBoost algorithm. First, few arrangments in the trainset ans testset. IT was necessary to change the features class to numeric and also splitting the target variable from the others.

```
# splitting the target variable, train and testset
trainset <- mutate_if(trainset, is.factor, as.numeric)
testset <- mutate_if(testset, is.factor, as.numeric)
train.x <- trainset %>% select(-evolution)
train.y <- trainset %>% select(evolution) %>% mutate(evolution = as.numeric(evolution))
test.x <- testset %>% select(-evolution)
test.y <- testset %>% select(evolution) %>% mutate(evolution = as.numeric(evolution))

#change the labels in binary data
train.y$evolution <- ifelse(train.y$evolution == 1, 0, 1)
test.y$evolution <- ifelse(test.y$evolution == 1, 0, 1)
```

Then , the algorithm was implemented as follows:

```
#8 - xgboost

#fitting the model
set.seed(1, sample.kind = "Rounding")
fit_xgboost <- xgboost(data = as.matrix(sapply(train.x, as.numeric)),
                       label = as.matrix(train.y),
                       max.depth = 4,
                       nrounds = 7,
                       early_stopping_rounds = 5,
                       objective = "reg:logistic",
                       eta = 0.4,
                       gamma = 3)
```

```
## [1]  train-rmse:0.423430
## Will train until train_rmse hasn't improved in 5 rounds.
##
## [2]  train-rmse:0.389842
## [3]  train-rmse:0.371325
## [4]  train-rmse:0.360953
## [5]  train-rmse:0.356324
## [6]  train-rmse:0.352842
## [7]  train-rmse:0.349624
```

```
p_hat_xgboost <- predict(fit_xgboost, as.matrix(sapply(test.x, as.numeric)))
y_hat_xgboost <- ifelse(p_hat_xgboost<0.5,"death","survive") %>%
  factor(levels=levels(testset.y))

#calculating theoutput parameters
acc <- confusionMatrix(y_hat_xgboost, testset.y)$overall["Accuracy"] %>% round(4)
sens <- confusionMatrix(y_hat_xgboost, testset.y)$byClass["Sensitivity"] %>% round(4)
spec <- confusionMatrix(y_hat_xgboost, testset.y)$byClass["Specificity"] %>% round(4)
prec <- confusionMatrix(y_hat_xgboost, testset.y)$byClass["Precision"] %>% round(4)
f1 <- confusionMatrix(y_hat_xgboost, testset.y)$byClass["F1"] %>% round(4)

#updating the output table
test_results <- bind_rows(test_results,
                          data_frame(method="XGBoost",
```

```r
                                          Accuracy = acc, Sensitivity = sens,
                                          Specificity = spec,
                                          Precision = prec, F1 = f1))
test_results[8,] %>% knitr::kable()
```

method	Accuracy	Sensitivity	Specificity	Precision	F1
XGBoost	0.8118	0.7558	0.8642	0.8388	0.7951

We see that the results increased in comparison with other classification trees algorithms.

3. Results

We have applied some machine learning algorithms on the dataset and collected importante parameters to compare then each other. We clusetered the results in the table below.

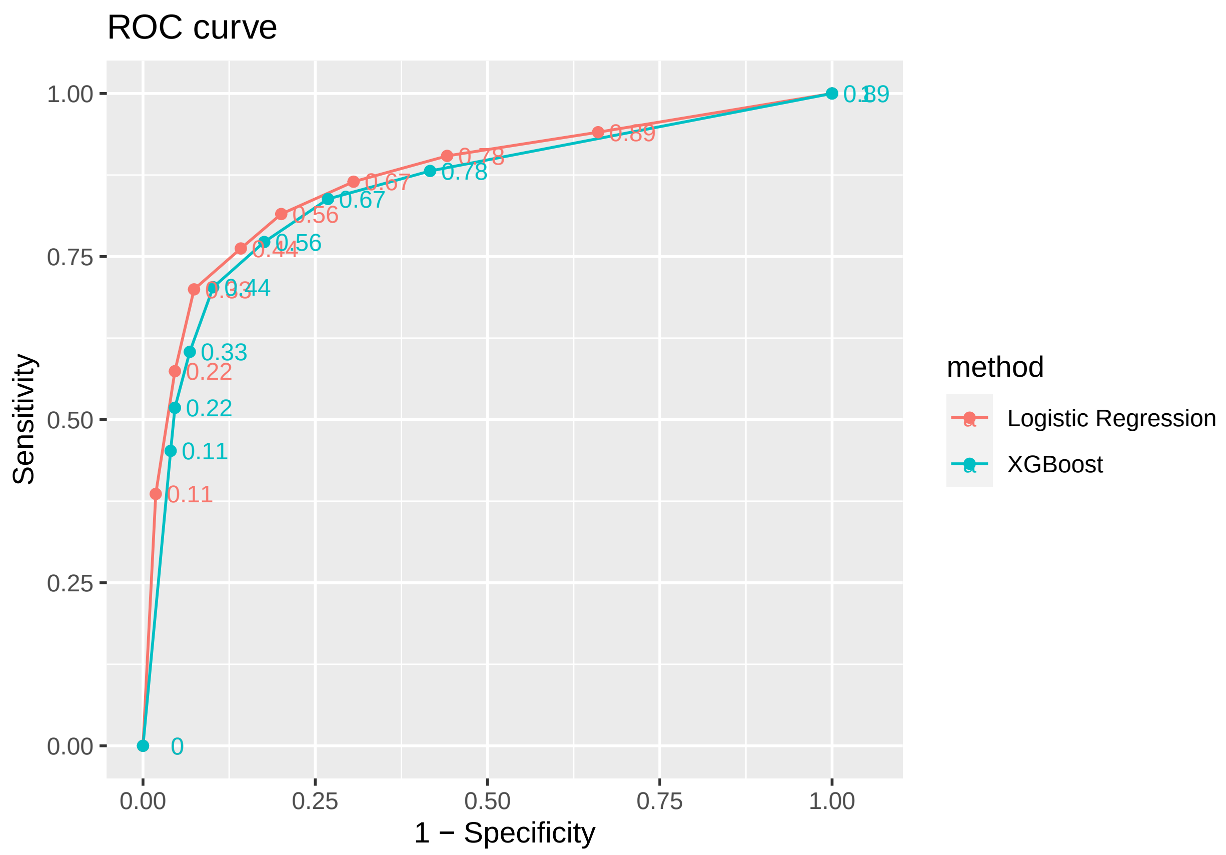
```r
#Overall results
test_results %>% knitr::kable()
```

method	Accuracy	Sensitivity	Specificity	Precision	F1
Guessing	0.4801	0.5017	0.4599	0.4648	0.4825
Logistic Regression with Age and Dyspnoea	0.7719	0.7492	0.7932	0.7721	0.7605
Logistic Regression with All Features	0.8134	0.7855	0.8395	0.8207	0.8027
K-Nearest Neighbors	0.7943	0.7261	0.8580	0.8271	0.7733
K-Nearest Neighbors Cross Validation	0.7624	0.6667	0.8519	0.8080	0.7306
Classification Tree - rpart()	0.7927	0.7525	0.8302	0.8057	0.7782
Classification Tree - caret	0.7671	0.7063	0.8241	0.7897	0.7456
XGBoost	0.8118	0.7558	0.8642	0.8388	0.7951

In terms of accuracy, the most efficient apgorithms were the logistic regression applied on all features. Looking at the F-score, also the Logistic Regression reached the highest value. Considering the nature of the prediction the most important parameter is the Sensitivity, where a False Negative result in the disease evolution prediction can leads to a lack of care in a situation that would demand maximum attention due to the high risck of death. Thus, the higher sensitivity value is 0.7855, reached with logistic regression on all features. This value is very low, considering the COVID19.

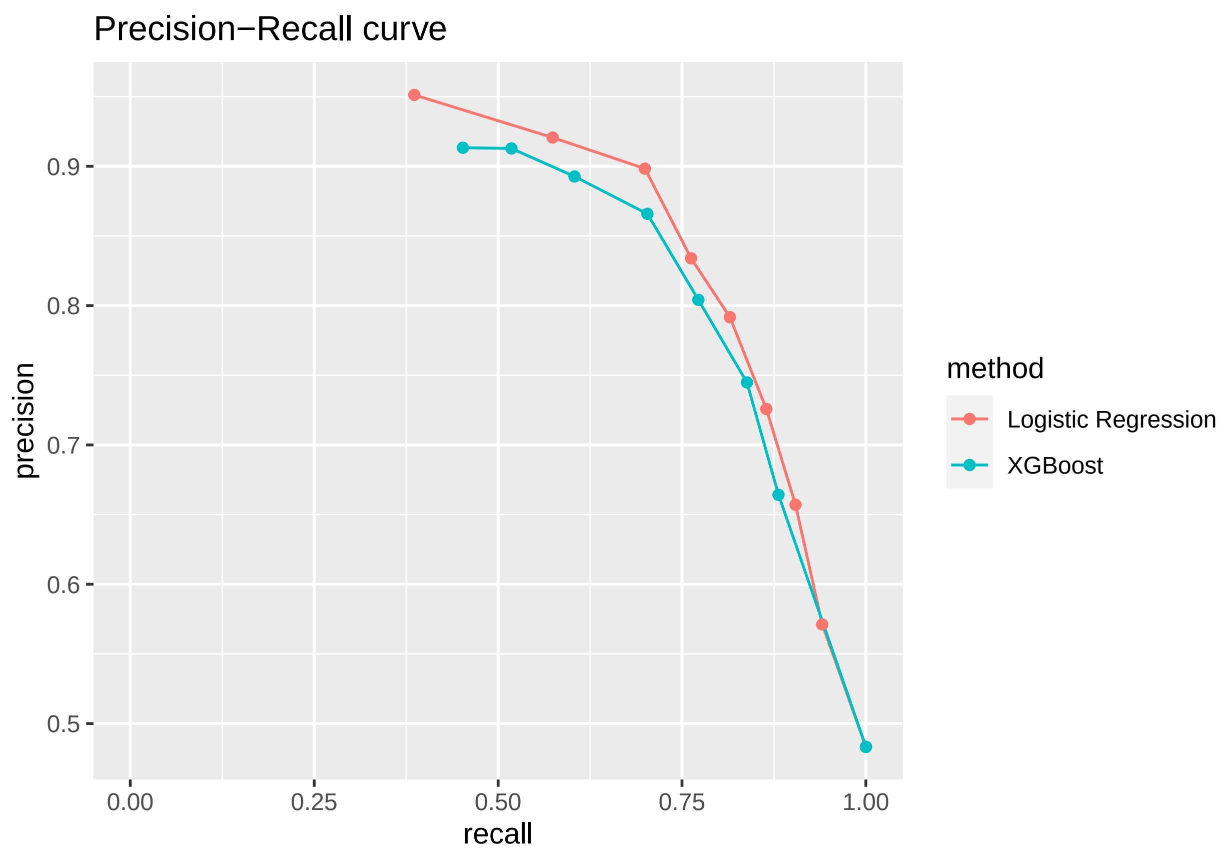
Let's check the two methods with better values, according to the previous table, which are the Logistic regression and teh XGBoost. The ROC curve foth both methods is the following:

```r
### checking the ROC curve for Logistic Regression and XGBoost.
probs <- seq(0, 1, length.out = 10)

glm_probs <- map_df(probs, function(p){
  y_hat <- ifelse(p_hat_glm_all< p,"death","survive") %>%
    factor(levels=levels(testset.y))
  list(method = "Logistic Regression",
       cutoff = round(p, 2),
       FPR = 1 - specificity(y_hat, testset.y),
       TPR = sensitivity(y_hat, testset.y))
})

XGboost_probs <- map_df(probs, function(p){
  y_hat <- ifelse(p_hat_xgboost< p ,"death","survive") %>%
    factor(levels=levels(testset.y))
  list(method = "XGBoost",
       cutoff = round(p, 2),
       FPR = 1 - specificity(y_hat, testset.y),
       TPR = sensitivity(y_hat, testset.y))
})

#Plotting the ROC
bind_rows(glm_probs, XGboost_probs) %>%
  ggplot(aes(FPR, TPR, label = cutoff, color = method)) +
  geom_line() +
  geom_point() +
  geom_text(size = 3, nudge_x = 0.05) +
  xlab("1 - Specificity") +
  ylab("Sensitivity") +
  ggtitle("ROC curve")
```



We see that both curves are almost the same, which very small advantage for logistic regression.
Despite this dataset is downsampled in a way that the prevalence of each output is equivalent, we plotted the Precision-Recall curve:

```r
### Checking the Precision-Recall curve for Logistic regression and XGBosst
probs <- seq(0, 1, length.out = 10)

glm_probs <- map_df(probs, function(p){
  y_hat <- ifelse(p_hat_glm_all< p,"death","survive") %>%
    factor(levels=levels(testset.y))
  list(method = "Logistic Regression",
       recall = sensitivity(y_hat, testset.y),
       precision = precision(y_hat, testset.y))
})

XGboost_probs <- map_df(probs, function(p){
  y_hat <- ifelse(p_hat_xgboost< p ,"death","survive") %>%
    factor(levels=levels(testset.y))
  list(method = "XGBoost",
       recall = sensitivity(y_hat, testset.y),
       precision = precision(y_hat, testset.y))
})

#plotting the curve
bind_rows(glm_probs, XGboost_probs) %>%
  ggplot(aes(recall, precision, color = method)) +
  geom_line() +
  geom_point() +
  ggtitle("Precision-Recall curve")
```



We see that the curve is similar, due to the balanced dataset. Also there is a small advantage to Logistic regression results. We see that to reach high sensitivity values, the precision decreases. The more a "positive" result is predicted, the more is the chance to have False Positives, which at the end, lower the precision value.

4. Conclusion

This report analysed a COVID19 dataset from São Paulo State, released by the Brazilian Goverment in the official Public Health website. The goal was to check the data in order to get some insigths on how that disease affect some popupation features, like gender, age, previous diseases, and others. We got important information from the data, for example it is widelly known, that the older population is more sucettible to death from COVID19. Among this older population, mortality of males is higher than females. Also, previous diseases affect de evolution of the CODIV19, leading to a high mortality than a group with no previous disease. Regarding to the symtoms, we have seen three of them, *cought*, $ fever$, and *dyspnoea*, mainly this last one, related with high proportion of deaths. Grouping all that information we could implement some machine learning algorithm to predict the CODIV19 evolution. Some algorithms presented higher accuracy than others, however available information on the dataset could not lead to a good accuracy. As a further investigation on the topic, considering this same dataset we would remove some more entries with low appearances, like the RT-antigen test, followed by increasing the amount of valid observations. Also, we would work on algorithms tunning, mainly the Xboost algorithm, to reach better performance. Another investigation is to analyse other datasets, from other states and countries, looking for features which allows better prediction of COVID19 evolution.

5. References

[1] Espinosa O.A., Zanetti A.S., Antunes E.F., Longhi F.G., Matos T.A., Battaglini, P.F. , *Prevalence of comorbidities in patients and mortality cases affected by SARS-CoV2: a systematic review and meta-analysis.* Rev. Inst. Med. trop. S. Paulo vol.62 São Paulo 2020 Epub June 22, 2020. https://doi.org/10.1590/s1678-9946202062043 , acessed in 19/Nov/2020.

[2]O'Driscoll M., Santos G.R., Wang L., Cummings D.A.T, Azman A.S., Paireau J., Fontanet A., Cauchemez S., Salje H., *Age-specific mortality and immunity patterns of SARS-CoV-2 infection in 45 countries*, https://doi.org/10.1101/2020.08.24.20180851 (preprint without peer review), acessed in 19/Nov/2020.

[3] https://www.who.int/news-room/commentaries/detail/estimating-mortality-from-covid-19, acessed in 19/Nov/2020.

[4] https://www.nature.com/articles/d41586-020-02140-8, acessed in 29/Sep/2020.

[5] Williams T.C., Wastnedge E., McAllister G., Bhatia R., Cuschieri K., Kefala K., Hamilton F., Johannessen I., Laurenson I.F., Shepherd J., Stewart A., Waters D., Wise H., Templeton K.E., *Sensitivity of RT-PCR testing of upper respiratory tract samples for SARS-CoV-2 in hospitalised patients: a retrospective cohort study.* https://doi.org/10.1101/2020.07.13.20152439 (preprint without peer review) . acessed in 29/Sep/2020.

[6] Castro R., Luz P. M., Wakimoto M.D., Veloso V.G., Grinsztejn B., Perazzo H., *COVID19: a meta-analysis of diagnostic test accuracy of commercial assays registered in Brazil*, The Brazilian Journal of Infectius Diseases, 2020,24(2),180-187. https://doi.org/10.1016/j.bjid.2020.04.003, acessed in 29/Sep/2020.

[7] Floriano I., Silvinato A., Bernardo W.M., Reis J., Soledade G., *Accuracy of the Polymerase Chain Reaction (PCR) test in the diagnosis of acute respiratory syndrome due to coronavirus: a systematic review and meta-analysis.*,Rev. Assoc. Med. Bras. vol.66 no.7 São Paulo July 2020 Epub Aug 24, 2020 , https://doi.org/10.1590/1806-9282.66.7.880 , acessed in 26/Oct/2020.