Statistical analysis of clinical data

Marc Bañuls Tornero

3/3/2021

Contents

Abstract

In this report we are going to do an statistical analysis of clinical data, reporting and explaining the steps done and discussing different results of the data and an explanation of the code and packages used in the process.

Objectives

At first we will reformat the data to be more understandable, and for each analysis or representation of the data we will have to clean it or reorganize it properly.

Having a date of the arrival of patients, we want to do several temporal distributions, to check this way differences in some variables depending the time of the year or in which group it is categorized the patient.

Another thing that can be interesting is to study the data of the patients depending on their BMI.

After that, we want to check if there is a relation between different variables of the patient and if the patient exit the hospital as “Deceased” or “Alive”. We can do that using a logistic regression of the dataset.

Lastly, we jump to machine learning algorithms to try to create a model that can predict if the patient will be deceased or alive using only it’s data.

Methods and materials

The data to be used will be clinical data from a hospital.

The methods to be used are comprised in different R packages. Some are used in a daily basis and others are necessary to facilitate the data cleaning process. Other packages are more unusual, but very useful to achieve some of the objectives.
We will see thorough the report where we use the different packages and why. This way it will be more dynamic and makes more sense for the type of report I am trying to do.

Data obtention

The data are samples of patients provided by IIS La Princesa to do a brief Technical test for a job offer, and I thought it would be a good idea to reuse the data to extend the analysis.

Data selection and understanding

First we set our working directory where our dataset is stored:

```
setwd("D:/R_projects/Prueba_técnica_Proceso_06/2021")
```

Then we read our data. Knowing that is in xlsx format we can use the package *readxl*. This package let us read data stored as xlsx to R in an intuitive way.

```
data<- read_excel("Ejercicio Práctico.xlsx")
```

```
## Warning in read_fun(path = enc2native(normalizePath(path)), sheet_i = sheet, :
## Expecting numeric in AH1061 / R1061C34: got 'NA'
```

```
## Warning in read_fun(path = enc2native(normalizePath(path)), sheet_i = sheet, :
## Expecting numeric in AH1914 / R1914C34: got 'NA'
```

```
## Warning in read_fun(path = enc2native(normalizePath(path)), sheet_i = sheet, :
## Expecting numeric in AH2053 / R2053C34: got 'NA'
```

Reading the xlsx file we have a warning where it expects numeric cells but there are NA values. These missing values will be treated or removed later properly to ensure clean reports, so we can ignore the warnings.

We can check de data to understand the variables and to see what can be used for analysis:

```
str(data)
```

```
## tibble [2,314 x 38] (S3: tbl_df/tbl/data.frame)
##  $ N         : num [1:2314] 1 2 3 4 5 6 7 8 9 10 ...
##  $ Edad      : chr [1:2314] "[75,80)" "[60,65)" "[75,80)" "[65,70)" ...
##  $ Ingreso   : POSIXct[1:2314], format: "2020-02-05" "2020-02-06" ...
##  $ Sexo      : chr [1:2314] "H" "H" "H" "M" ...
##  $ Pais_Nac  : chr [1:2314] "ESPAÑA" "ESPAÑA" "ESPAÑA" "ESPAÑA" ...
##  $ Serv_Alta : chr [1:2314] "INFC" "INFC" "NRC" "NML" ...
##  $ Extranjero: chr [1:2314] "N" "N" "N" "N" ...
##  $ VMNI      : chr [1:2314] "NA" "NA" "NA" "S" ...
##  $ VMI       : chr [1:2314] "NA" "NA" "NA" "S" ...
##  $ UCI       : chr [1:2314] "S" "NA" "NA" "S" ...
##  $ Moti_Alta : chr [1:2314] "Alta con seguimiento en atención primaria" "Traslado a centro sociosani
##  $ D1        : chr [1:2314] "S06.6X0A" "M86.9" "C71.0" "J10.08" ...
##  $ D2        : chr [1:2314] "R09.2" "L89.154" "NA" "J13" ...
##  $ D3        : chr [1:2314] "K62.6" "B95.62" "J12.89" "J96.01" ...
##  $ D4        : chr [1:2314] "K62.5" "B95.4" "B97.29" "I21.3" ...
##  $ D5        : chr [1:2314] "N39.0" "B95.2" "I10" "I25.10" ...
```

```
##  $ D6        : chr [1:2314] "B96.20" "B96.89" "E78.5" "J20.9" ...
##  $ D7        : chr [1:2314] "Z16.12" "A41.50" "E11.9" "B96.1" ...
##  $ D8        : chr [1:2314] "B95.2" "N39.0" "E03.9" "Y95" ...
##  $ D9        : chr [1:2314] "F43.20" "B96.5" "NA" "Z16.12" ...
##  $ D10       : chr [1:2314] "J12.89" "A41.89" "NA" "I48.91" ...
##  $ D11       : chr [1:2314] "B97.29" "J98.8" "NA" "J95.811" ...
##  $ D12       : chr [1:2314] "F05" "B97.29" "NA" "G62.81" ...
##  $ D13       : chr [1:2314] "A04.72" "Y95" "NA" "E46" ...
##  $ D14       : chr [1:2314] "I69.254" "T82.7XXA" "NA" "A41.9" ...
##  $ D15       : chr [1:2314] "I69.298" "B96.1" "NA" "R65.20" ...
##  $ D16       : chr [1:2314] "H53.47" "Z16.12" "NA" "N17.9" ...
##  $ D17       : chr [1:2314] "I25.2" "Y84.8" "NA" "D68.9" ...
##  $ D18       : chr [1:2314] "I10" "R53.81" "NA" "J12.89" ...
##  $ D19       : chr [1:2314] "I71.4" "R68.83" "NA" "B97.29" ...
##  $ D20       : chr [1:2314] "Z93.6" "T40.2X5A" "NA" "B37.49" ...
##  $ PESO_D    : chr [1:2314] "NA" "102.9" "1.61" "55" ...
##  $ COLESTEROL: chr [1:2314] "176" "132" "160" "129" ...
##  $ LINFOCITOS: num [1:2314] 0.88 1.99 0.8 0.54 1.1 2.03 1.86 1.98 2.12 2.73 ...
##  $ ALBUMINA  : chr [1:2314] "3.9" "3.4" "3.2" "2.6" ...
##  $ TALLA_D   : chr [1:2314] "NA" "NA" "86.6" "NA" ...
##  $ CONUT     : chr [1:2314] "1" "2" "2" "2" ...
##  $ IMC       : chr [1:2314] "NA" "NA" "2.15" "NA" ...
```

We see that the dataset is treated as a tibble with 2314 rows and 38 column variables. As we can see most of the column names are acronyms or are in Spanish. First we'll explain every column and see if it can have a use for possible analyses. This way we can then translate properly the columns to English and remove redundant or unnecessary data.

The "N" column keeps a numeric order of the data, but doesn't give useful information about the patients, so we can remove it.
"Edad" means "Age" and it give us an age range of each patient, so it can be interesting to use for future analyses.
"Ingreso" indicates the date of admission of the patient in the hospital. This can be used for future analyses.
"Sexo" tells if the addmited patient is male "H" or female "M", a useful data to use later. "Pais_Nac" indicates the nationality of the admitted patient.
"Serv_Alta" tells us which service is used to discharge the patient. This variable doesn't seem useful so we can remove it later.
"Extranjero" tells us if the admitted patient is living in the hospital's country (Spain) or is from a foreign country. This data can be seen as redundant with "Pais_Nac", but there are people with other nationalities that aren't foreign from Spain, so we keep this data to use later.
"VMNI" and "VMI" are spanish acronyms for noninvasive mechanical ventilation and invasive mechanical ventilation respectively, so these two columns can be useful. In english the acronym would be NIMV and IMV respectively.
"UCI" is another spanish acronym that means Intensive Care Unit (ICU in English), so this column can also be interesting to keep.
"Moti_Alta" tells the reason of discharge of each patient. This data is also useful but to simplify the data we can use this variable to see if the reason of discharging the patient is because is deceased ("Exitus") or because it's well enough to be discharged.
The twenty following columns (from "D1" to "D20") tells the diagnosis code of each patient. As this data is very specific and almost each patient of this dataset has a different one, it's better to discard all these columns for the sake of simplicity.
"PESO_D" indicates the weight in Kg of the patient, that is complemented with the height in cm of the patient in the column "TALLA_D" and achieves its completion with the "IMC" column, that uses the previous two variables to give the BMI of the patient.
The columns "COLESTEROL", "LINFOCITOS" and "ALBUMINA" gives the amount of cholesterol (in

mg/dL) lymphocytes (in thousands of units/mL) and albumin (in g/dL) respectively, being interesting data to analyse later.
Lastly the "CONUT" column is an acronym for Nutritional Control, that gives a number depending on the quantities obtained in the variables of cholesterol lymphocytes and albumin. This number indicates the risk of malnutrition of the patient, and it can be helpful to do a summary of the previous variables.

Data cleaning

Now that we know the meaning of the data and what can be useful or not, we can first remove the columns that we won't use.

```
# Removing all the ICD data, that starts with "D"
data <- data[, -grep("^D.*", colnames(data))]

# Removing the other columns not needed by name
data_clean <- subset(data, select = -c(N,Serv_Alta))
```

Now we can check again the format of the data to see if it needs changes or it's good to use already.

```
str(data_clean)
```

```
## tibble [2,314 x 16] (S3: tbl_df/tbl/data.frame)
##  $ Edad      : chr [1:2314] "[75,80)" "[60,65)" "[75,80)" "[65,70)" ...
##  $ Ingreso   : POSIXct[1:2314], format: "2020-02-05" "2020-02-06" ...
##  $ Sexo      : chr [1:2314] "H" "H" "H" "M" ...
##  $ Pais_Nac  : chr [1:2314] "ESPAÑA" "ESPAÑA" "ESPAÑA" "ESPAÑA" ...
##  $ Extranjero: chr [1:2314] "N" "N" "N" "N" ...
##  $ VMNI      : chr [1:2314] "NA" "NA" "NA" "S" ...
##  $ VMI       : chr [1:2314] "NA" "NA" "NA" "S" ...
##  $ UCI       : chr [1:2314] "S" "NA" "NA" "S" ...
##  $ Moti_Alta : chr [1:2314] "Alta con seguimiento en atención primaria" "Traslado a centro sociosani
##  $ PESO_D    : chr [1:2314] "NA" "102.9" "1.61" "55" ...
##  $ COLESTEROL: chr [1:2314] "176" "132" "160" "129" ...
##  $ LINFOCITOS: num [1:2314] 0.88 1.99 0.8 0.54 1.1 2.03 1.86 1.98 2.12 2.73 ...
##  $ ALBUMINA  : chr [1:2314] "3.9" "3.4" "3.2" "2.6" ...
##  $ TALLA_D   : chr [1:2314] "NA" "NA" "86.6" "NA" ...
##  $ CONUT     : chr [1:2314] "1" "2" "2" "2" ...
##  $ IMC       : chr [1:2314] "NA" "NA" "2.15" "NA" ...
```

Now we only have 16 columns, a far smaller pool of variables to work. We see that most of the values are treated as characters. Some of the variables are really numerical data, and other variables that have characters can be reformatted to factors with different levels, to easily classify the repeated data.

To facilitate the understanding of the variables we'll change its names to english:

```
colnames(data_clean) <- c("Age", "Admission_date", "Gender", "Nationality", "Foreign",
                          "NIMV", "IMV", "ICU", "Discharge", "Weight", "Cholesterol",
                          "Lymphocytes", "Albumin", "Height", "CONUT", "BMI")
```

We first reformat the numerical variables:

```
# Columns to reformat to numeric
numerics <- c("Weight", "Cholesterol", "Lymphocytes", "Albumin", "Height", "CONUT", "BMI")

# Code used to convert the variables to numeric
data_clean[numerics] <- lapply(data_clean[numerics], as.numeric)
```

```
## Warning in lapply(data_clean[numerics], as.numeric): NAs introducidos por
```

```
## coerción

## Warning in lapply(data_clean[numerics], as.numeric): NAs introducidos por
## coerción

## Warning in lapply(data_clean[numerics], as.numeric): NAs introducidos por
## coerción

## Warning in lapply(data_clean[numerics], as.numeric): NAs introducidos por
## coerción

## Warning in lapply(data_clean[numerics], as.numeric): NAs introducidos por
## coerción

## Warning in lapply(data_clean[numerics], as.numeric): NAs introducidos por
## coerción
```

The NAs introduced by coercion are only introduced were there already was a NA value, so don't have to worry about it.

```
str(data_clean)
```

```
## tibble [2,314 x 16] (S3: tbl_df/tbl/data.frame)
##  $ Age           : chr [1:2314] "[75,80)" "[60,65)" "[75,80)" "[65,70)" ...
##  $ Admission_date: POSIXct[1:2314], format: "2020-02-05" "2020-02-06" ...
##  $ Gender        : chr [1:2314] "H" "H" "H" "M" ...
##  $ Nationality   : chr [1:2314] "ESPAÑA" "ESPAÑA" "ESPAÑA" "ESPAÑA" ...
##  $ Foreign       : chr [1:2314] "N" "N" "N" "N" ...
##  $ NIMV          : chr [1:2314] "NA" "NA" "NA" "S" ...
##  $ IMV           : chr [1:2314] "NA" "NA" "NA" "S" ...
##  $ ICU           : chr [1:2314] "S" "NA" "NA" "S" ...
##  $ Discharge     : chr [1:2314] "Alta con seguimiento en atención primaria" "Traslado a centro socios
##  $ Weight        : num [1:2314] NA 102.9 1.61 55 NA ...
##  $ Cholesterol   : num [1:2314] 176 132 160 129 195 161 188 133 129 156 ...
##  $ Lymphocytes   : num [1:2314] 0.88 1.99 0.8 0.54 1.1 2.03 1.86 1.98 2.12 2.73 ...
##  $ Albumin       : num [1:2314] 3.9 3.4 3.2 2.6 3.9 4.3 4.4 4.6 2.4 3.8 ...
##  $ Height        : num [1:2314] NA NA 86.6 NA NA NA 158 165 NA NA ...
##  $ CONUT         : num [1:2314] 1 2 2 2 3 1 2 1 2 1 ...
##  $ BMI           : num [1:2314] NA NA 2.15 NA NA ...
```

We can see now that there are a lot of missing values, but this is common with clinical data, so for the moment we will keep formating the variables.

The variables that aren't numeric (except Admission_date") can be reformatted to factors due to being repeated values, but we can change some of the parameters of each variable to facilitate the understanding of the data or for simplicity.

The variable Age is divided in ranges of five years, and these ranges can be interpreted as factors. Also, due to the large amount of factors it can be difficult to analyse the data, so we can relevel the factors to wider ranges of age. One good approach is to divide the ranges in objective categories of age. We will divide the ages in 4 categories, young-young adult (from 15 to 25), adult (from 25 to 40), middle aged adult (from 40 to 65) and older adult (from 65 to 110).
To do the reformatting of the Age variable, we will use the package forcats, a package that allow us to collapse the different ranges of age predetermined to the bigger range that we stated before.

```
# "Age" factor levels manually named
data_clean$Age <- factor(data_clean$Age, levels = c("[15,20)", "[20,25)", "[25,30)",
```

```
                                                    "[30,35)", "[35,40)", "[40,45)", "[45,50)", "[50,55)",
                                                    "[60,65)", "[65,70)", "[70,75)", "[75,80)", "[80,85)",
                                                    "[90,95)", "[95,100)", "[100,105)", "[105,110)"))
```

```
# Relevel of factors to group bigger ranges of age
data_clean$Age <- fct_collapse(data_clean$Age,
                               "[15,25)" = c("[15,20)", "[20,25)"),
                               "[25,40)" = c("[25,30)", "[30,35)", "[35,40)"),
                               "[40-65)" = c("[40,45)", "[45,50)",
                                             "[50,55)", "[55,60)", "[60,65)"),
                               "[65,110)" = c("[65,70)", "[70,75)", "[75,80)",
                                              "[80,85)", "[85,90)", "[90,95)",
                                              "[95,100)", "[100,105)", "[105,110)"))
```

The other columns can be reformatted to factors without major changes.

```
# Variables a convertir en factor
factors <- c("Gender", "Nationality", "Discharge", "Foreign", "NIMV", "IMV", "ICU")

# Conversión de factores
data_clean[factors] <- lapply(data_clean[factors], factor)
```

Now we can check again the data:

```
str(data_clean)
```

```
## tibble [2,314 x 16] (S3: tbl_df/tbl/data.frame)
##  $ Age            : Factor w/ 4 levels "[15,25)","[25,40)",..: 4 3 4 4 4 4 3 2 3 3 ...
##  $ Admission_date: POSIXct[1:2314], format: "2020-02-05" "2020-02-06" ...
##  $ Gender         : Factor w/ 2 levels "H","M": 1 1 1 2 1 1 2 2 1 1 ...
##  $ Nationality    : Factor w/ 46 levels "ARMENIA","BELGICA",..: 5 5 5 5 5 19 5 5 5 5 ...
##  $ Foreign        : Factor w/ 2 levels "N","S": 1 1 1 1 1 2 1 1 1 1 ...
##  $ NIMV           : Factor w/ 2 levels "NA","S": 1 1 1 2 1 1 1 2 1 1 ...
##  $ IMV            : Factor w/ 2 levels "NA","S": 1 1 1 2 2 1 1 1 2 2 ...
##  $ ICU            : Factor w/ 2 levels "NA","S": 2 1 1 2 2 1 1 1 2 2 ...
##  $ Discharge      : Factor w/ 10 levels "Alta con seguimiento en atención primaria",..: 1 9 2 6 6 10 1
##  $ Weight         : num [1:2314] NA 102.9 1.61 55 NA ...
##  $ Cholesterol    : num [1:2314] 176 132 160 129 195 161 188 133 129 156 ...
##  $ Lymphocytes    : num [1:2314] 0.88 1.99 0.8 0.54 1.1 2.03 1.86 1.98 2.12 2.73 ...
##  $ Albumin        : num [1:2314] 3.9 3.4 3.2 2.6 3.9 4.3 4.4 4.6 2.4 3.8 ...
##  $ Height         : num [1:2314] NA NA 86.6 NA NA NA 158 165 NA NA ...
##  $ CONUT          : num [1:2314] 1 2 2 2 3 1 2 1 2 1 ...
##  $ BMI            : num [1:2314] NA NA 2.15 NA NA ...
```

Now we can see that the factor levels aren't very understandable (partly due to the data aimed to be interpreted in spanish). To fix this, we change and translate the symbols to be interpreted in english. In the case of NIMV and IMV we notice that when they aren't used is interpreted as NA, so to prevent misunderstandings we change it to N (No).

```
# Gender factor change name
levels(data_clean$Gender)[levels(data_clean$Gender) == "M"] <- "F"
levels(data_clean$Gender)[levels(data_clean$Gender) == "H"] <- "M"

# Foreign factor change name
levels(data_clean$Foreign)[levels(data_clean$Foreign) == "S"] <- "Y"

# NIMV and IMV factor change name
```

```r
levels(data_clean$NIMV)[levels(data_clean$NIMV) == "S"] <- "Y"
levels(data_clean$NIMV)[levels(data_clean$NIMV) == "NA"] <- "N"

# IMV factor change name
levels(data_clean$IMV)[levels(data_clean$IMV) == "S"] <- "Y"
levels(data_clean$IMV)[levels(data_clean$IMV) == "NA"] <- "N"

# ICU factor change name
levels(data_clean$ICU)[levels(data_clean$ICU) == "S"] <- "Y"
levels(data_clean$ICU)[levels(data_clean$ICU) == "NA"] <- "N"
```

Lastly we change the Discharge factor levels to check only if the reason of the discharge is the person being deceased or well enough to be discharged (Alive or Deceased).

```r
# Grouping discharge reason in "Alive" or "Éxitus"
data_clean$Discharge <- ifelse(data_clean$Discharge == "Éxitus",0,1)
data_clean$Discharge<- factor(data_clean$Discharge)
levels(data_clean$Discharge)<- c("Deceased", "Alive")
```

```r
str(data_clean)
```

```
## tibble [2,314 x 16] (S3: tbl_df/tbl/data.frame)
##  $ Age           : Factor w/ 4 levels "[15,25)","[25,40)",..: 4 3 4 4 4 4 3 2 3 3 ...
##  $ Admission_date: POSIXct[1:2314], format: "2020-02-05" "2020-02-06" ...
##  $ Gender        : Factor w/ 2 levels "M","F": 1 1 1 2 1 1 2 2 1 1 ...
##  $ Nationality   : Factor w/ 46 levels "ARMENIA","BELGICA",..: 5 5 5 5 5 19 5 5 5 5 ...
##  $ Foreign       : Factor w/ 2 levels "N","Y": 1 1 1 1 1 2 1 1 1 1 ...
##  $ NIMV          : Factor w/ 2 levels "N","Y": 1 1 1 2 1 1 1 2 1 1 ...
##  $ IMV           : Factor w/ 2 levels "N","Y": 1 1 1 2 2 1 1 1 2 2 ...
##  $ ICU           : Factor w/ 2 levels "N","Y": 2 1 1 2 2 1 1 1 2 2 ...
##  $ Discharge     : Factor w/ 2 levels "Deceased","Alive": 2 2 2 1 1 2 2 1 2 2 ...
##  $ Weight        : num [1:2314] NA 102.9 1.61 55 NA ...
##  $ Cholesterol   : num [1:2314] 176 132 160 129 195 161 188 133 129 156 ...
##  $ Lymphocytes   : num [1:2314] 0.88 1.99 0.8 0.54 1.1 2.03 1.86 1.98 2.12 2.73 ...
##  $ Albumin       : num [1:2314] 3.9 3.4 3.2 2.6 3.9 4.3 4.4 4.6 2.4 3.8 ...
##  $ Height        : num [1:2314] NA NA 86.6 NA NA NA 158 165 NA NA ...
##  $ CONUT         : num [1:2314] 1 2 2 2 3 1 2 1 2 1 ...
##  $ BMI           : num [1:2314] NA NA 2.15 NA NA ...
```

The data is correclty ordered, letting us see clearly the data available.

Now we can do a first summary of the data, to analyse in a general way the data to be treated:

```r
summary(data_clean)
```

```
##        Age       Admission_date                Gender
##  [15,25) :  15   Min.   :2020-02-05 00:00:00   M:1266
##  [25,40) : 116   1st Qu.:2020-03-24 00:00:00   F:1048
##  [40-65) : 836   Median :2020-04-04 00:00:00
##  [65,110):1347   Mean   :2020-05-26 08:55:10
##                  3rd Qu.:2020-08-31 00:00:00
##                  Max.   :2020-10-25 00:00:00
##
##                Nationality   Foreign  NIMV     IMV      ICU
##  ESPAÑA              :1730   N:2261   N:2231   N:2222   N:2162
##  REPUBLICA DEL ECUADO: 134   Y:  53   Y:  83   Y:  92   Y: 152
```

```
##  REPUBLICA DE PERU   : 115
##  REPUBLICA DE BOLIVIA:  40
##  REPUBLICA DE COLOMBI:  38
##  REPUBLICA DOMINICANA:  34
##  (Other)             : 223
##     Discharge        Weight        Cholesterol     Lymphocytes
##  Deceased: 362   Min.   :  0.30   Min.   : 60.0   Min.   :  0.14
##  Alive   :1952   1st Qu.: 63.10   1st Qu.:134.0   1st Qu.:  0.92
##                  Median : 75.00   Median :165.0   Median :  1.37
##                  Mean   : 75.01   Mean   :166.8   Mean   :  1.78
##                  3rd Qu.: 85.00   3rd Qu.:195.0   3rd Qu.:  2.05
##                  Max.   :150.00   Max.   :476.0   Max.   :206.57
##                  NA's   :1841     NA's   :155     NA's   :3
##     Albumin          Height           CONUT            BMI
##  Min.   :1.60   Min.   :  1.64   Min.   :1.000   Min.   : 1.92
##  1st Qu.:3.50   1st Qu.:158.00   1st Qu.:1.000   1st Qu.:23.50
##  Median :4.00   Median :166.50   Median :2.000   Median :26.37
##  Mean   :3.97   Mean   :164.42   Mean   :1.633   Mean   :27.10
##  3rd Qu.:4.40   3rd Qu.:174.00   3rd Qu.:2.000   3rd Qu.:30.39
##  Max.   :5.40   Max.   :195.00   Max.   :3.000   Max.   :56.00
##  NA's   :129    NA's   :1967     NA's   :468     NA's   :2009
```

In the factor data format variables we can see the number of patients in each level of the variable. In a quick look we can see that most of the patients are older adults, followed by middle-aged adults.

As we said previously, due to being clinical data there are a significant number of NA values, and we can see them in some of the variables of the dataset. The variable with most of the NA values are in the weight and height of the patient, reflected in the BMI (if one of these two variables isn't present in the patient the BMI can't be calculated either). Besides these variables, we see that a small percentage of the datasets have NA values in the cholesterol lymphocytes and albumin variables, The variable CONUT has a significant number of NA due to being necessary that the patient must have data about cholesterol lymphocytes and albumin (if one these variables doesn't have data CONUT isn't calculable). We need to be careful with these missing values when we start analysing the data, and we can try to remove the patient with the variables of interest with missing values, omit directly the values, standarize them, etc.

Analyses

Temporal distribution

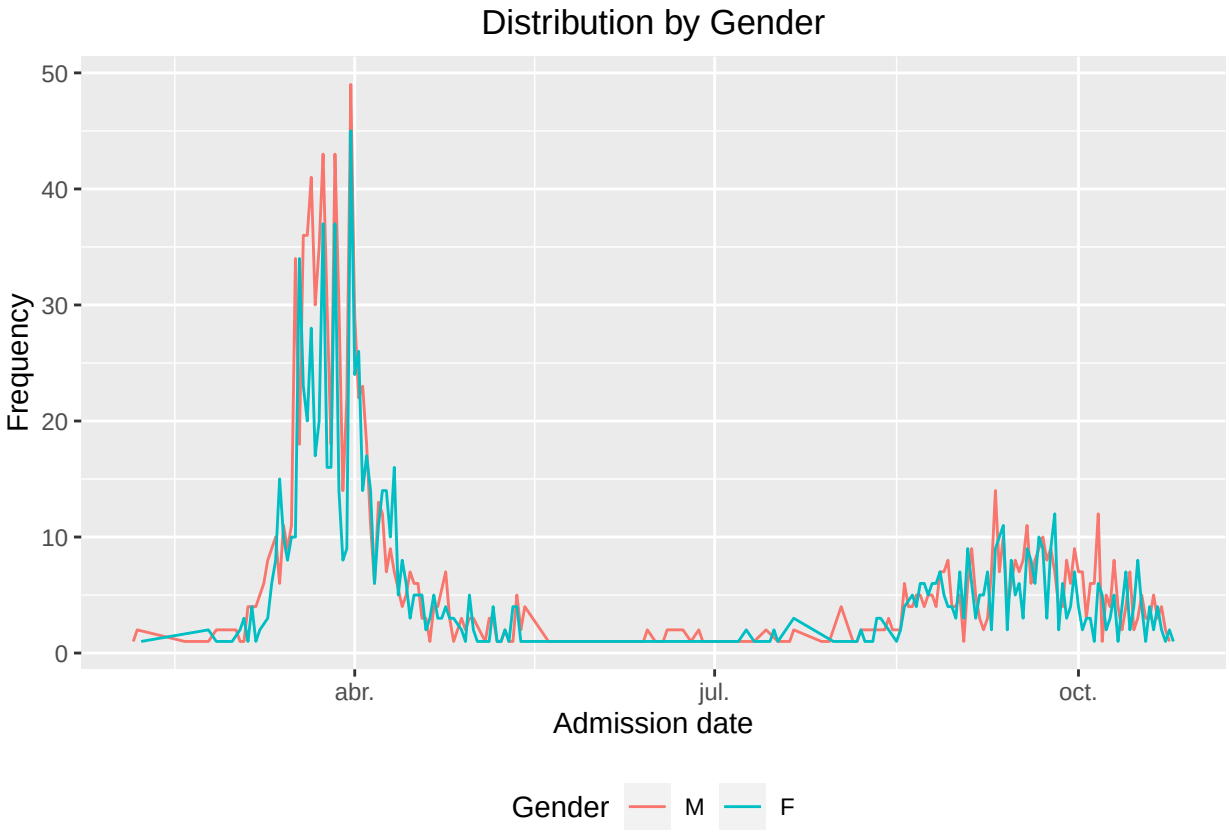
Due to having admission date data of each patient we can use it to see the flow of patients treating different variables. One good way to do this is making a graphical representation of different variables with a temporal distribution. We chose in this report to make graphical visualizations of the Gender, Nationality and Age. To make the temporal distribution we will use the ggplot2 package, that will allow us to do these graphics in an intuitive way.

By gender

To be able to count the number of people in each level of the variable we have to sum the number of patients of each level separately of the dataset. This can be done grouping the gender variable with the admission date, and then using this reformatted dataset to plot the result.

```
# Data grouping by gender and admission date
genderplot <- data_clean %>%
  group_by(Admission_date,Gender) %>%
  tally()
```

```
ggplot(data = genderplot, aes(x = Admission_date, y = n, colour = Gender)) +
  geom_line() +
  xlab("Admission date") +
  ylab("Frequency") +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5)) +
  ggtitle("Distribution by Gender")
```



We can see clearly the number of males or females being admitted at the hospital at different rates during the different weeks and months. In the graphic it's visible that there are a greater number of males than females in average being seen clearly between march and april. Nevertheless the difference in gender doesn't seem to be high enough to obtain accurate conclusions of this graphic.

By nationality

We saw in the summary that there are a great number of levels, indicating that a lot of nationalities are being considered. If we do a specific summary of this variable we can see in more detail the nationalities.

```
summary(data_clean$Nationality)
```

```
##              ARMENIA              BELGICA               BRASIL
##                    1                    3                    4
##          DESCONOCIDO              ESPAÑA ESTADOS UNIDOS DE AM
##                    6                 1730                    5
## ESTADOS UNIDOS MEXIC     FEDERACION RUSA FRANCIA, METROPOLITA
##                    6                    2                    4
##              HOLANDA               INDIA              IRLANDA
##                    1                    1                    1
```

```
##                ITALIA                 KENYA             MARRUECOS
##                     2                     1                    12
##             NO EXISTE               POLONIA              PORTUGAL
##                     2                     3                     4
##           REINO UNIDO REPUBLICA ARABE SIRI  REPUBLICA ARGENTINA
##                     7                     1                    12
## REPUBLICA DE BANGLAD REPUBLICA DE BOLIVIA   REPUBLICA DE CHILE
##                     4                    40                     8
##    REPUBLICA DE CHINA REPUBLICA DE COLOMBI    REPUBLICA DE CUBA
##                     2                    38                    14
## REPUBLICA DE FILIPIN REPUBLICA DE GUATEMA  REPUBLICA DE GUINEA
##                    19                     1                     3
## REPUBLICA DE HONDURA REPUBLICA DE INDONES    REPUBLICA DE IRAQ
##                    11                     1                     2
## REPUBLICA DE NICARAG REPUBLICA DE PARAGUA    REPUBLICA DE PERU
##                     7                    10                   115
## REPUBLICA DEL ECUADO REPUBLICA DEL SALVAD REPUBLICA DOMINICANA
##                   134                     3                    34
##    REPUBLICA FRANCESA   REPUBLICA GABONESA              RUMANIA
##                     3                     1                    19
##                 SUIZA        UKRAINIAN SSR              URUGUAY
##                     3                     6                     1
##             VENEZUELA
##                    27
```

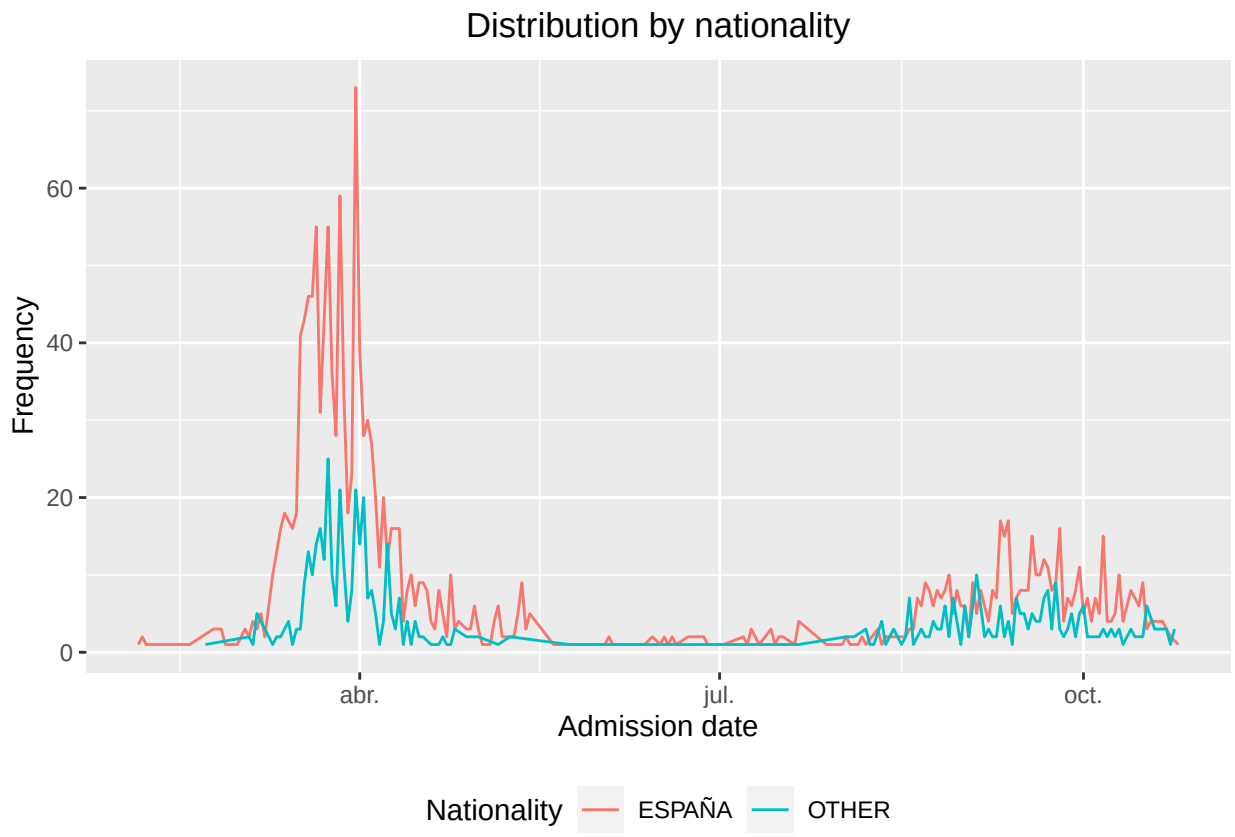
As expected, most patients are from "ESPAÑA" (Spain) and there are a few people in different nationalities, being the most notable after spain ECUADOR and PERU which have 134 and 115 persons respectively. In contrast, we have 1 or 2 persons of other nationalities that, when plotted, won't be seen and will only make the graphic look messy. To have an interesting plot while maintaining the data, we can group all the nationalities outside of Spain in one, so this way it will be more signficant.

```
# Name change of countries outside of Spain to "OTHER"
countrydata <- data_clean
countrydata$Nationality <- as.character(countrydata$Nationality)
countrydata$Nationality[countrydata$Nationality != "ESPAÑA"] <- "OTHER"
countrydata$Nationality <- as.factor(countrydata$Nationality)
```

After that we can do the same as the gender plot to group the data and plot it

```
# Data grouping by nationality and admission date
countryplot <- countrydata %>%
  group_by(Admission_date,Nationality) %>%
  tally()
```

```
ggplot(data = countryplot, aes(x = Admission_date, y = n, colour = Nationality)) +
  geom_line() +
  xlab("Admission date") +
  ylab("Frequency") +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5)) +
  ggtitle("Distribution by nationality")
```



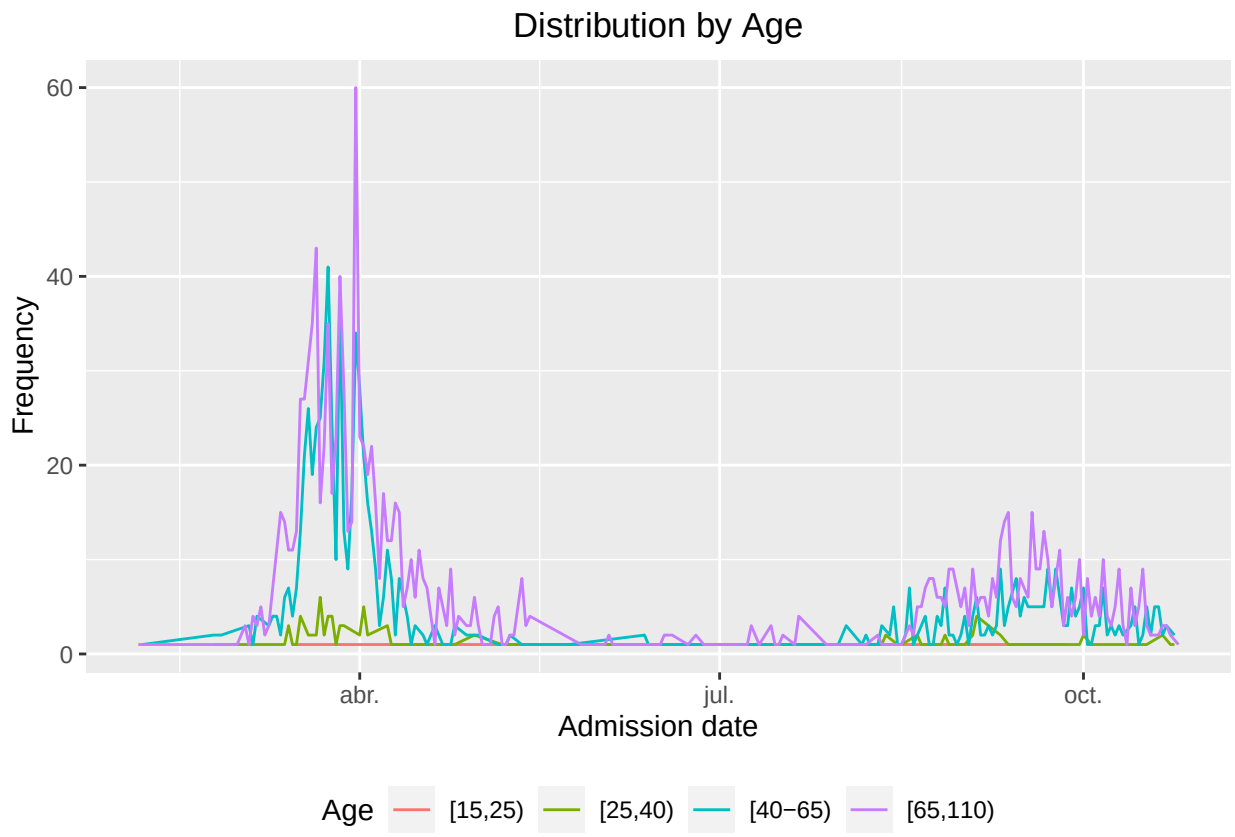
As expected, there is a big difference between the number of patients from other nationalities in a spanish hospital but interestingly, in the range of September to November the differences in nationalities have a decrease, being in some weeks a similar number of spanish patients and patients of other nationality. This may be due to the lowering of restrictions during these months, allowing people from other countries to travel more freely these dates.

By Age

As we stated previously, we grouped the ranges of age in four groups, allowing us to make this plot easily readable and giving us information more accurately. We make the plot in a similar manner than in the previous plots, grouping the patients with the same range that are admitted in the same date to see its frequency.

```
# Data grouping by age and admission date
ageplot <- data_clean %>%
  group_by(Admission_date, Age) %>%
  tally()
```

```
ggplot(data = ageplot, aes(x = Admission_date, y = n, colour = Age)) +
  geom_line() +
  xlab("Admission date") +
  ylab("Frequency") +
  theme(legend.position = "bottom", plot.title = element_text(hjust = 0.5)) +
  ggtitle("Distribution by Age")
```



In this temporal distribution we see the lack of young adult and adult people, being visible that the patients usually are old adults or middle-aged adults being the first one more frequent than the second one. Objectively speaking, is logical that the patients tend to be of older age (the older the more frequent is to have medical complications).

BMI study

One variable that can be useful to study about is the BMI of the patients. As we said previously, there are a lot of missing values in the height and weight variables, so consequently that affects the IBM missing values.

```
table(is.na(data_clean$BMI))
```

```
##
## FALSE  TRUE
##   305  2009
```

Specifically there are 2009 NA values in the BMI variable, implying that we can only have 305 patients with both height and with values. With that amount of NA values it's better to work with only the 305 patients because it would be difficult to normalise or predict that amount of missing data. This way although we don't use all the data of the dataset, we can have accurate results of the available data.

We can for example make a linear regression to see how affects some of the different variables to the IMC. Some interesting variables to study in relation of the BMI would be the amount of cholesterol lymphocytes and albumin of the patient.

```
BMI_lm <- lm(BMI ~ Cholesterol + Lymphocytes + Albumin, data = data_clean)
```

Now we can check the linear regression:

```r
summary(BMI_lm)
```

```
##
## Call:
## lm(formula = BMI ~ Cholesterol + Lymphocytes + Albumin, data = data_clean)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -24.0990  -3.7118  -0.6381   3.1044  29.1428
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 21.132738   2.045986  10.329   <2e-16 ***
## Cholesterol  0.007485   0.008017   0.934   0.3512
## Lymphocytes  0.007438   0.027294   0.273   0.7854
## Albumin      1.195299   0.551042   2.169   0.0309 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.606 on 295 degrees of freedom
##   (2015 observations deleted due to missingness)
## Multiple R-squared:  0.02966,    Adjusted R-squared:  0.01979
## F-statistic: 3.005 on 3 and 295 DF,  p-value: 0.03071
```

One thing to note is that we didn't remove the rows that had missing values so the linear regression did it automatically for us, being noted below the residual standard error line. We can see that the multiple R squared and adjusted R-squared are very low, indicating how bad the linear regression is. This may be due to the low amount of data taking in account the amount of variables needed to do the linear regression.

Logistic regression model.

We can do a logistic regression with the clean data to see how the survivability of the patient is related to the concentrations of cholesterol, lymphocytes and albumin.

```r
logit_model <- glm(Discharge ~ Cholesterol + Lymphocytes + Albumin,
                   data = data_clean, family = "binomial")
```

```r
summary(logit_model)
```

```
##
## Call:
## glm(formula = Discharge ~ Cholesterol + Lymphocytes + Albumin,
##     family = "binomial", data = data_clean)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3018   0.4508   0.5220   0.6142   1.1839
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.730830   0.380737  -1.920   0.0549 .
## Cholesterol  0.003771   0.001590   2.372   0.0177 *
## Lymphocytes -0.009881   0.007462  -1.324   0.1855
## Albumin      0.473481   0.108079   4.381 1.18e-05 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1839.0  on 2146  degrees of freedom
## Residual deviance: 1794.7  on 2143  degrees of freedom
##   (167 observations deleted due to missingness)
## AIC: 1802.7
##
## Number of Fisher Scoring iterations: 4
```

We can see first that some observations have been deleted due to missingness, suposing that the model have found NA values. Using the dataset "clean" we have some interesting information, indicating that the concentration of albumin affects significantly the survival of the patient. Also, the cholesterol levels seems to have a significant effect in the survivality of the patient. Due to being a logistic regression, the summary is seen differently to a linear regression. In this case, to interpret the significance of the Albumin, for example, we have to check the Estimate of this Coefficient. Specifically, in this summary it says that for every unit change in the concentration of albumin, the chance (log-odd) of survival decreases in a 0.47. In case of the cholesterol, the chance of survival decreases in a 0.003, confirming the low correlation significance.

Note: As you can see, the chance of survival is inverted as it is said in the summary. This is due to the factor levels of the variable "Discharge" being first Alive and second Deceased:

```
str(data_clean$Discharge)
```

```
##  Factor w/ 2 levels "Deceased","Alive": 2 2 2 1 1 2 2 1 2 2 ...
```

Neural network model

Lastly we would like to create a neural network model using the R package neuralnet, that let us train and test the dataset efficiently, and give us a result of how good or bad is the model created with the data. But to create this model we need to prepare again the data.

Data processing

To process the data we will use the cleaned dataset and change it to our necessity.

```
data_nn <- data_clean
```

In a neural network model we can't have factor variables, and the recommended way to prepare the data is to change the factor classes to TRUE and FALSE in separarate and new variables.

```
str(data_nn)
```

```
## tibble [2,314 x 16] (S3: tbl_df/tbl/data.frame)
##  $ Age           : Factor w/ 4 levels "[15,25)","[25,40)",..: 4 3 4 4 4 4 3 2 3 3 ...
##  $ Admission_date: POSIXct[1:2314], format: "2020-02-05" "2020-02-06" ...
##  $ Gender        : Factor w/ 2 levels "M","F": 1 1 1 2 1 1 2 2 1 1 ...
##  $ Nationality   : Factor w/ 46 levels "ARMENIA","BELGICA",..: 5 5 5 5 5 19 5 5 5 5 ...
##  $ Foreign       : Factor w/ 2 levels "N","Y": 1 1 1 1 1 2 1 1 1 1 ...
##  $ NIMV          : Factor w/ 2 levels "N","Y": 1 1 1 2 1 1 1 2 1 1 ...
##  $ IMV           : Factor w/ 2 levels "N","Y": 1 1 1 2 2 1 1 1 2 2 ...
##  $ ICU           : Factor w/ 2 levels "N","Y": 2 1 1 2 2 1 1 1 2 2 ...
##  $ Discharge     : Factor w/ 2 levels "Deceased","Alive": 2 2 2 1 1 2 2 1 2 2 ...
##  $ Weight        : num [1:2314] NA 102.9 1.61 55 NA ...
##  $ Cholesterol   : num [1:2314] 176 132 160 129 195 161 188 133 129 156 ...
##  $ Lymphocytes   : num [1:2314] 0.88 1.99 0.8 0.54 1.1 2.03 1.86 1.98 2.12 2.73 ...
##  $ Albumin       : num [1:2314] 3.9 3.4 3.2 2.6 3.9 4.3 4.4 4.6 2.4 3.8 ...
```

```
##  $ Height        : num [1:2314] NA NA 86.6 NA NA NA 158 165 NA NA ...
##  $ CONUT         : num [1:2314] 1 2 2 2 3 1 2 1 2 1 ...
##  $ BMI           : num [1:2314] NA NA 2.15 NA NA ...
```

In this case we will only use the numerical data and change the Discharge factor to be in two variables, "Alive" and "deceased"

```
data_nn$Alive <- data_nn$Discharge
data_nn$Deceased <- data_nn$Discharge
```

In the variable "Alive" when the patient had a factor Alive in the previous variable "Discharge", it will be wrote TRUE, and FALSE if otherwise. We will do the contrary in the "Deceased" variable.

```
data_nn$Alive <- sapply(data_nn$Alive, function(x) {ifelse(x == "Alive", TRUE, FALSE)})
data_nn$Deceased <- sapply(data_nn$Deceased, function(x) {ifelse(x == "Deceased", TRUE, FALSE)})
```

We also need to take care of missing values in the variables we are going to use, so we have to check these variables for NA values and remove them. To do so we need to remove the rows of patients that don't have values for these variables. We use the tidyr package to drop the patients that have missing values in these variables.

```
data_nn <- data_nn %>% drop_na(c(Cholesterol, Lymphocytes, Albumin))
```

Normalizing numeric data

As we said previously, we will only do predictions with numeric data, so we will only use Cholesterol, Lymphocytes, and Albumin for the predictions. We could use wheight and height, but doesn't seem relevant, and in case we want to use it we would prioritize the BMI variable, that joins these two variables together.

Now we need to normalize the numeric data to prevent scaling problems.

```
normalize <- function(x) {return(((x-min(x)) / (max(x) - min(x))))}
```

```
data_nn_norm <- as.data.frame(lapply(data_nn[11:13], normalize))
```

We can check the normalization of the data comparing it with the previous dataset:

```
summary(data_nn$Cholesterol)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    60.0   134.0   165.0   166.9   195.0   476.0
```

```
summary(data_nn_norm$Cholesterol)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.1779  0.2524  0.2569  0.3245  1.0000
```

We can see that in the neural network dataset the numeric data is normalized correctly. Now we join the classifier variables in this normalized data.

```
data_nn_norm$Alive <- data_nn$Alive
data_nn_norm$Deceased <- data_nn$Deceased
```

Creating the training and test datasets

Now that we have the data prepared to be used, we separate the data in a training and test datasets. For the training dataset we will use 2/3 of the data samples, and the 1/3 left will be used as a test dataset.

```
# We set a seed to keep reproducibility
set.seed(123)
# Sampling of the rows in the normalized dataset
```

```r
sampling_data_nn <- sample(1:nrow(data_nn_norm), nrow(data_nn_norm) * 0.67, replace = FALSE)

# Train and Test dataset parititon of rows using the previous sampling
train_data_nn <- data_nn_norm[sampling_data_nn,]
test_data_nn <- data_nn_norm[-sampling_data_nn,]
```

We can check both datasets now to see if the code works as intended:

```r
str(train_data_nn)
```

```
## 'data.frame':    1438 obs. of  5 variables:
##  $ Cholesterol: num  0.115 0.274 0.32 0.144 0.298 ...
##  $ Lymphocytes: num  0.00354 0.00872 0.00669 0.00247 0.00882 ...
##  $ Albumin    : num  0.421 0.553 0.711 0.526 0.737 ...
##  $ Alive      : logi  FALSE TRUE TRUE TRUE FALSE FALSE ...
##  $ Deceased   : logi  TRUE FALSE FALSE FALSE TRUE TRUE ...
```

```r
str(test_data_nn)
```

```
## 'data.frame':    709 obs. of  5 variables:
##  $ Cholesterol: num  0.2404 0.3245 0.3077 0.3221 0.0264 ...
##  $ Lymphocytes: num  0.0031 0.00455 0.00824 0.00397 0.06904 ...
##  $ Albumin    : num  0.421 0.605 0.737 0.553 0.263 ...
##  $ Alive      : logi  TRUE FALSE TRUE TRUE FALSE TRUE ...
##  $ Deceased   : logi  FALSE TRUE FALSE FALSE TRUE FALSE ...
```

We can see here that the train dataset has 1438 rows while the test dataset has 709 rows, and all the variables have the expected classes and values.

Training the Neural Network model

To train a predictive model we use the package neuralnet, and to see the best parameters we can use to have a model with great accuracy and sensitivity, we have to try them and see the results. At first we will use one hidden node, and we add a parameter stating that the variables doesn't affect linearly the outcome.

```r
set.seed(123)
data_nn_model <- neuralnet(Alive+Deceased ~ Cholesterol + Lymphocytes + Albumin,
                           data = train_data_nn, hidden = 1, linear.output = FALSE)
```

Now we can see how the model works:

```r
plot(data_nn_model)
```

At this plot we observe the weights the model uses to predict the outcome using the three variables we stated at the model. It also shows the "steps" needed to create the model, showing us that it wasn't very computational intensive. We also see that the model has a high error, meaning that the model might perform poorly.

Model evaluation

To check how well the model predicts data we evaluate its prediction capacity, using for that the test dataset

```r
nn_model_result <- compute(data_nn_model, test_data_nn[,1:3])
```

```r
predicted_data_nn <- nn_model_result$net.result
```

In this data we have two vectors with a percentage of the predictions in each row (if the row patient is alive or deceased). With this data we can separate the two vectors to obtain the prediction of the two outcomes:

```r
pred_alive <- predicted_data_nn[,1]
pred_deceased <- predicted_data_nn[,2]
```

```r
t_pred_alive <- sapply(pred_alive, function(x) {ifelse (x > 0.5, TRUE, FALSE)} )
t_pred_deceased <- sapply(pred_deceased, function(x) {ifelse (x > 0.5, TRUE, FALSE)} )
```

Now we can create a confusion matrix of one of the two outcome predictions (due to being complementary outcomes we don't need to visualize both predictions). To make a confusion matrix easily we can use the caret package, which has a function that does exactly this.

```r
confusionMatrix(as.factor(t_pred_alive), as.factor(test_data_nn[,4]), positive = "TRUE")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE     1    1
##      TRUE     94  613
##
##                Accuracy : 0.866
##                  95% CI : (0.8387, 0.8902)
##     No Information Rate : 0.866
##     P-Value [Acc > NIR] : 0.5273
##
##                   Kappa : 0.0152
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.99837
##             Specificity : 0.01053
##          Pos Pred Value : 0.86704
##          Neg Pred Value : 0.50000
##              Prevalence : 0.86601
##          Detection Rate : 0.86460
##    Detection Prevalence : 0.99718
##       Balanced Accuracy : 0.50445
##
##        'Positive' Class : TRUE
##
```

In the confusion matrix is visible that the model is very sensible, but it has a very low specificity. Also, the very low kappa indicates that there is very few agreement between the classification of the model and the true values. This model then predicts poorly with the cholesterol albumin and lymphocyte values if the patient will end alive or deceased. This may be due to a significant difference between the number of deceased/alive ratio. This can lead to an overfitting of the data with values usually coming from alive patients. Also, it matters to the model creation that there are only 3 variables that it can use to do the prediction, taking in account that in the previous logistic regression it was showed that the value of lymphocytes didn't have a direct correlaction with the outcome of the patient. Another thing to state is the low amount of deceased patients in the train and test dataset, having less amount of data to use for the creation of the model and its testing.

We could add more hidden nodes (just changing the number of nodes at the neuralnet function works), but due to the overfitting already stated here, the accuracy of the model won't go higher, instead it will likely start to overfit the data even more and lower the accuracy of the model.

Conclusion

We have seen that most of the patients are adults and older adults, usually more men than women, and almost all patients are spaniards, being able to see the peak of patients that enters the hospital between march and april, where the CoVID19 was at its peak. Also, we found a correlation of the BMI of the patients with the concentration of albumin. In the logistic regression we found out that higher concentrations of cholesterol and albumin lead to a decrease in survivality of the patients studied. Lastly, after trying to do a neural network model we found that with the data available it's difficult to train it properly. A higher pool of patients with complete data of its variables, and variables that could be more useful to the outcome (Deceased or Alive) could help to construct a model with higher predictive power. Also we could try bootstraping the data, but this goes out of the scope and length of this report.

Disclaimer

I don't plan in discovering anything relevant here or designing original pipelines/workflows. My sole purpose is to practise analysing data with R and (maybe) this could be of help to someone that is more novice in statistics or R than me. If you find something that could be better or more efficient (I'm pretty sure that there may be flaws in my interpretations or thoughts in how to do the analyses or data processing) don't hesitate in contacting me. Also, if you have any questions or I didn't explain correctly my thoughts, I'm always available to discuss it thoroughly.