Data Visualisation and Manipulation

DATA VISUALISATION

In this document we are analysing the covid dataset. In order to do so the first step was to require ggplot2, and dplyr libraries.

```
library(readr)
library(ggplot2)
library(dplyr)
library(tidyr)
library(magrittr)
library(knitr)
library(MASS)
```

The second step was to upload the dataset, and to eliminate the index column that was not needed

```
covid <- read_csv("covid.csv")
data<- covid[,-1]
View(data)
```

Then, we inspected the dataset before visualizing it in order to understand what it was about.

```
data%>% glimpse
range(data$year)
data%>% distinct(year)
length(unique(data$country))
length(unique(data$continent))
data%>% distinct(year, cases)
sum(is.na(data$deaths))
sum(is.na(data$cases))
```

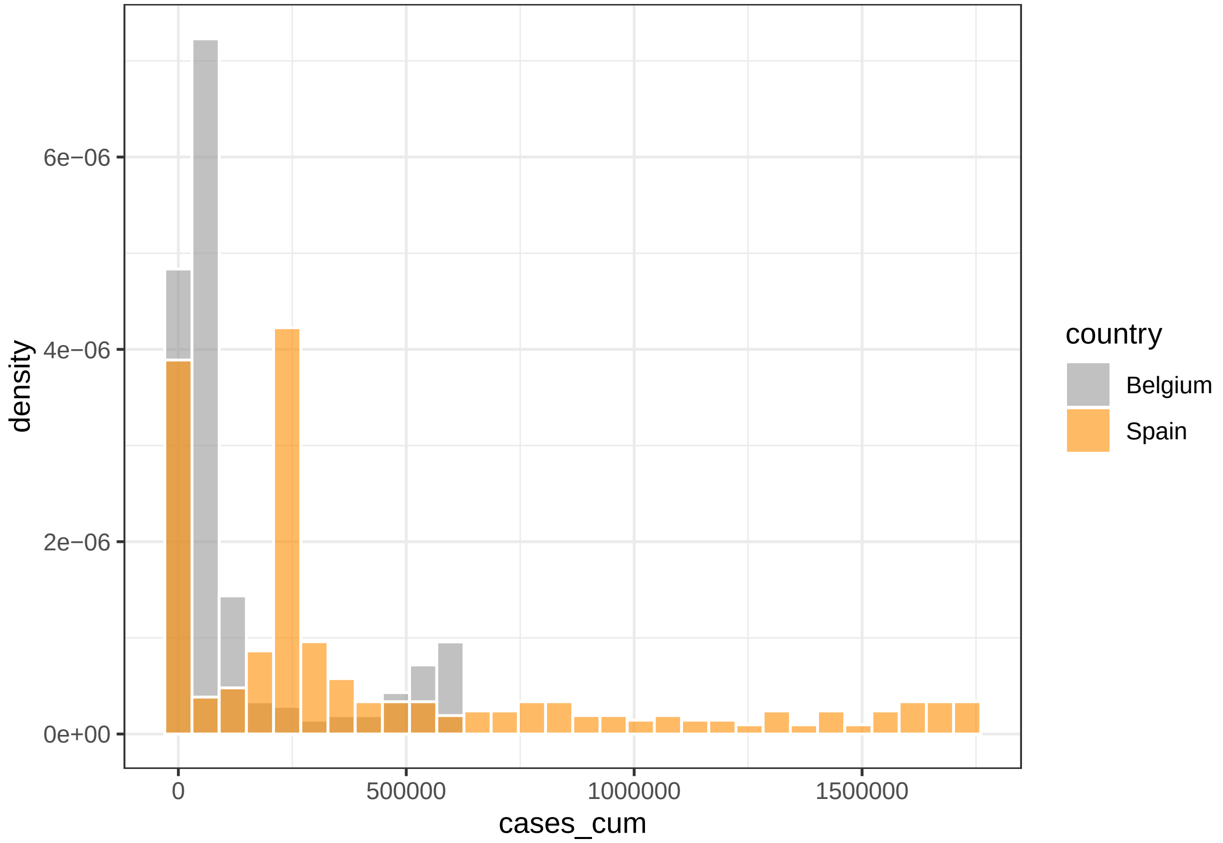
From the above code it emerged that: the data set is made of 61900 observations and 12 variables. The year inspected is from 31-12-2019 to 14-12-2020. There are 214 countries from 6 continents. We also checked that there were no missing values

DATA VISUALISATION WITH GGPLO2

For data visualization the ggplot2 library was used, sometimes together with some dyplr functions. The first plot compares the distribution of COVID cases by country, in this case we chose as example Belgium and Spain.

```
data%>%
  filter(country %in% c("Belgium", "Spain")) %>%
  ggplot(
    aes(x = cases_cum, y = ..density.., color = I("white"), fill = country)
  ) +
  geom_histogram(alpha = 0.6, position = "identity") +
  scale_fill_manual(values = c("grey60", "darkorange")) +
  theme_bw()
```

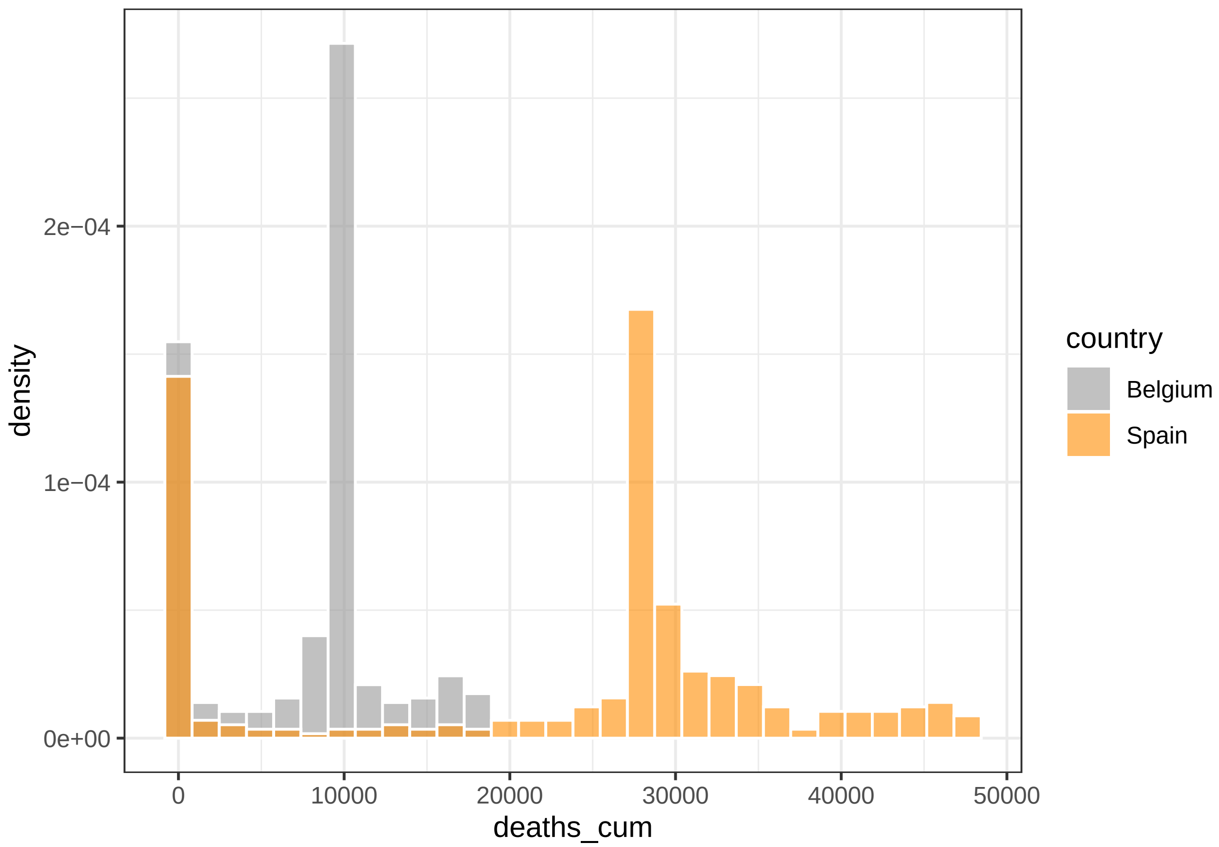
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



The same plot was then used to show the distribution of deaths for the same coutry. Note that the filter() function is part of the dplyr packaged and is used to extract the countries of interest

```
data %>%
  filter(country %in% c("Belgium", "Spain"))%>%
  ggplot(
    aes(x= deaths_cum, y= ..density.., color=I("white"), fill=country)
    )+
  geom_histogram(alpha=0.6, position="identity")+
  scale_fill_manual(values=c("grey60", "darkorange"))+
  theme_bw()
```

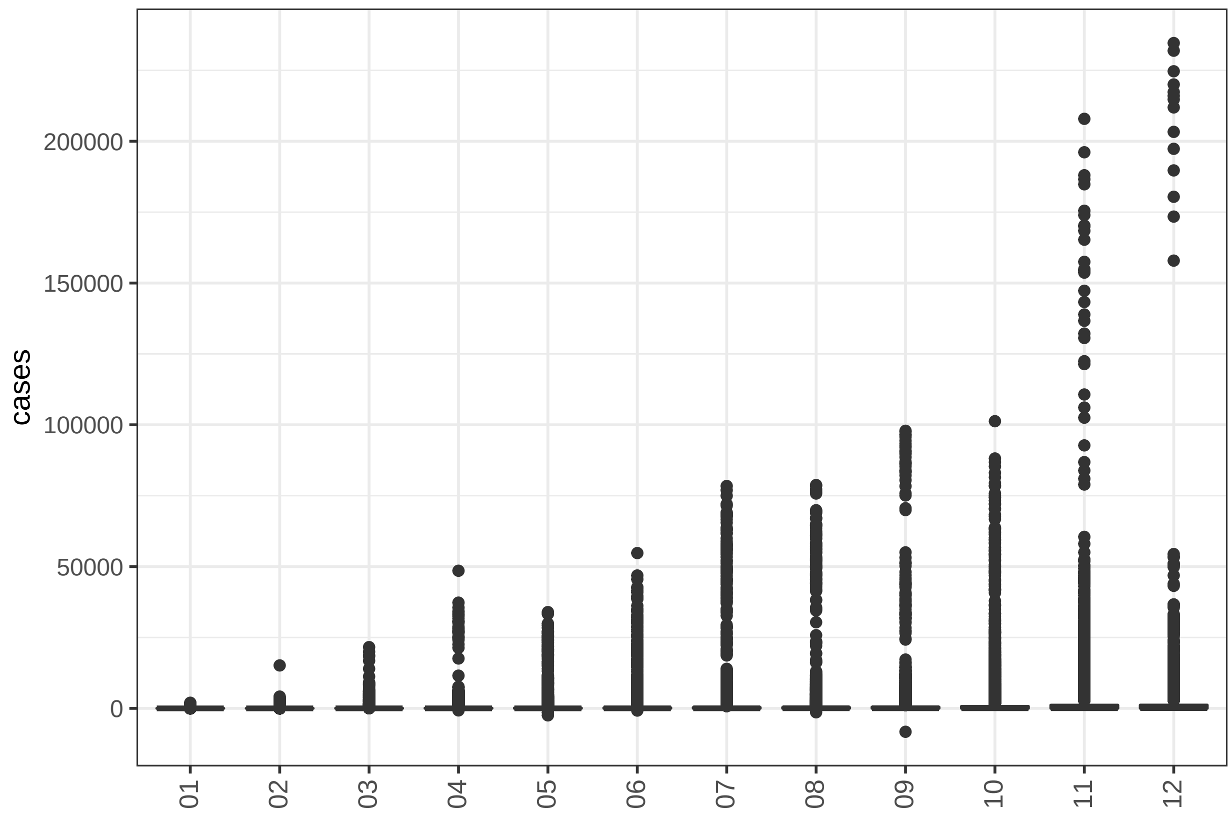
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



BOXPLOT

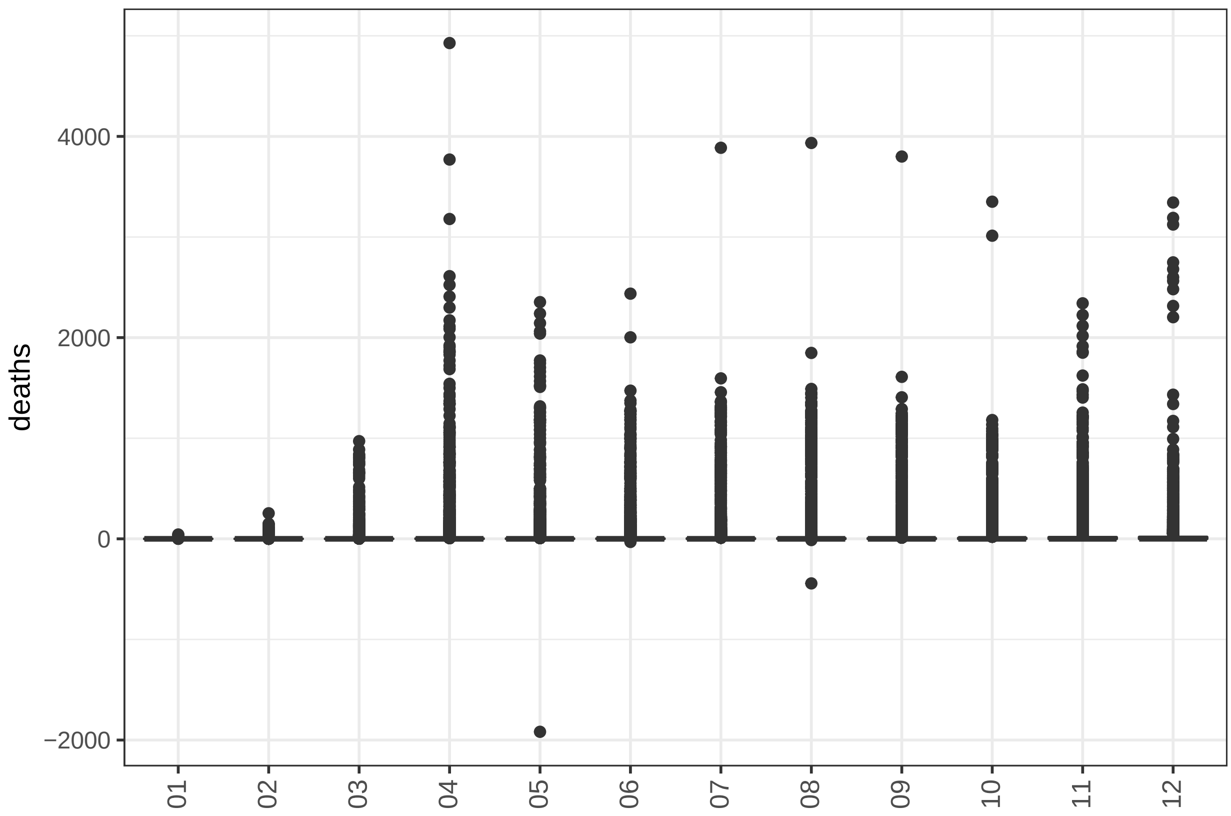
The following plot shows the distribution of cases by month

```
data %>%
  ggplot(aes(x = month, y = cases, fill = I("skyblue2"))) +
  geom_boxplot(position = "dodge") +
  labs(x = "") +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 90,  vjust = .4, size = 10))
```



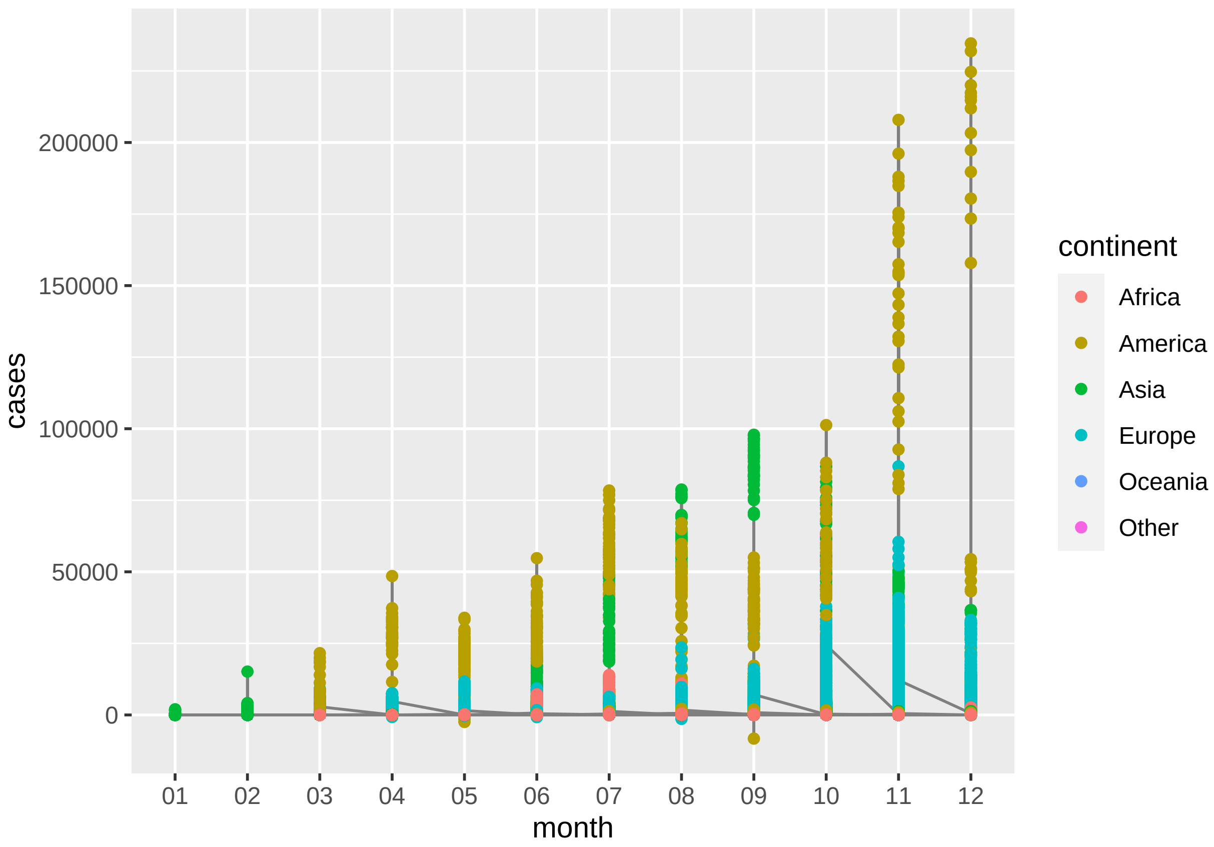
Same plot can be used to show the distribution of deaths.

```
data%>%
  ggplot(aes(x=month, y=deaths, fill=I("blue")))+
  geom_boxplot(position="dodge")+
  labs(x="")+
  theme_bw()+
  theme(axis.text.x=element_text(angle=90, vjust=.4, size=10))
```



Then, we can visualise changes in the number of cases in each continent on different months

```
ggplot(data, aes(month, cases))+
  geom_line(aes(group= continent), colour="grey50")+
  geom_point(aes(colour=continent))
```



Then we decided to summarise the statistics. In this case to inspect the number of deaths

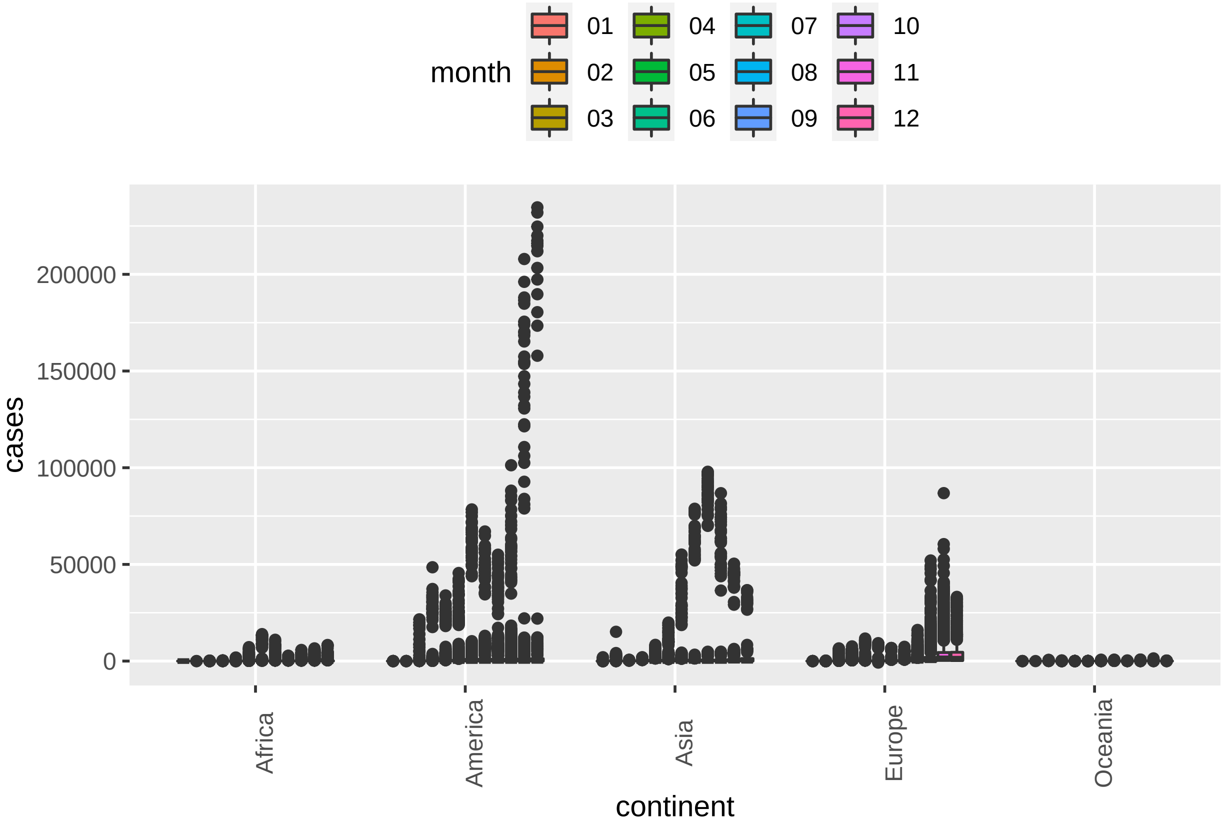
```
by(data$cases, data$deaths, summary)
by(data$cases, data$deaths, sd)

data %>%
  group_by(deaths) %>%
  summarise(
    Minimum = min(cases),
    Q1 = quantile(cases, probs = .25),
    Median = quantile(cases, probs = .5),
    Mean = mean(cases),
    Q3 = quantile(cases, probs = .75),
    Maximum = max(cases),
    SD = sd(cases),
    CV = abs(Mean)/SD,
    IQR = Q3 - Q1
  )
```

GROUPED BOXPLOT

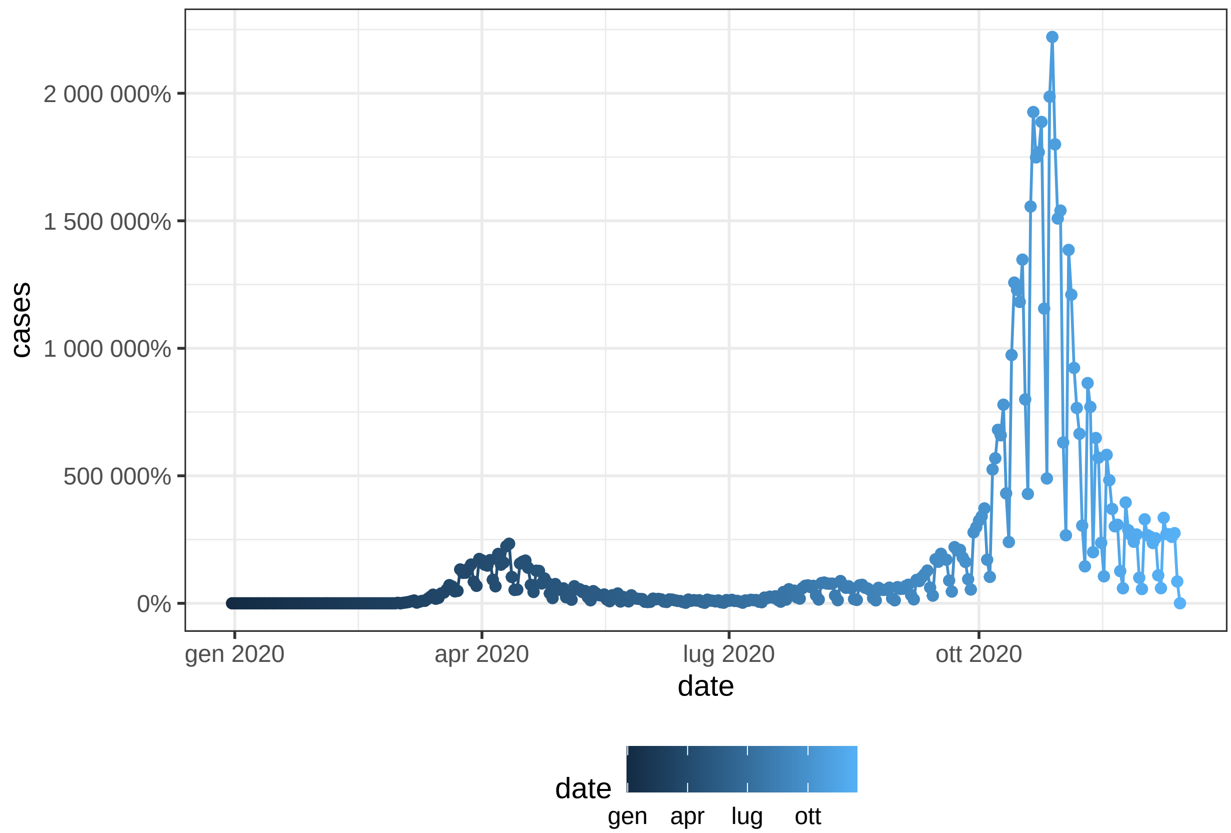
With the filter function we created a time series to show the distirbution of cases in each continent

```
data%>%
  filter(
    month %in% c("01", "02", "03", "04", "05","06", "07", "08", "09", "10", "11", "12"),
    grepl("*[aeiouy]$", country)
  ) %>%
  ggplot(aes(x = continent, y = cases, fill = month)) +
  geom_boxplot(positon = "dodge") +
  theme(legend.position = "top", axis.text.x = element_text(angle = 90))
```



A time series was then created for the country Belgium.

```
data %>% filter(country == "Belgium") %>%
  ggplot(aes(x =date, y = cases, color = date)) + geom_point() + geom_line() +
  scale_y_continuous(labels = scales::percent) +
  theme_bw() +
  theme(legend.position = "bottom")
```



## DATA MANUPULATION WITH DPLYR

Now we perform some data manipulation with the dplyr package. Suppose that we want to create a new dataframe with some of the variables form our original data. We use the select() function, and name the new dataset "data2". We added dplyr::select in order to avoid conflicts with the MASS library.

```
data2<- data%>%
  dplyr::select(deaths)
data2
```

```
## # A tibble: 61,900 x 1
##    deaths
##     <dbl>
##  1      0
##  2      0
##  3      0
##  4      0
##  5      0
##  6      0
##  7      0
##  8      0
##  9      0
## 10      0
## # ... with 61,890 more rows
```

Now suppose we want to rename the column deat with the new name being "KO". We used for this purpose the rename() function.

```
data3<- rename(data2, KO= deaths)
data3
```

```
## # A tibble: 61,900 x 1
##       KO
##    <dbl>
##  1     0
##  2     0
##  3     0
##  4     0
##  5     0
##  6     0
##  7     0
##  8     0
##  9     0
## 10     0
## # ... with 61,890 more rows
```

Then we used the filter() function in order to choose a subset of the original data, and retain only the values in which the variable death is equal to ten. Then we added the number "30" to retain also the cases in which the number of deaths was 30.

```
data4<-filter(data, deaths == "10")
data4
```

```
## # A tibble: 488 x 12
##    date       day   month  year cases deaths country     code  population continent
##    <date>     <chr> <chr> <dbl> <dbl>  <dbl> <chr>       <chr>      <dbl> <chr>
##  1 2020-04-27 27    04     2020    68     10 Afghanistan AF      38041757 Asia
##  2 2020-07-27 27    07     2020   106     10 Afghanistan AF      38041757 Asia
##  3 2020-08-13 13    08     2020    76     10 Afghanistan AF      38041757 Asia
##  4 2020-08-25 25    08     2020    71     10 Afghanistan AF      38041757 Asia
##  5 2020-09-16 16    09     2020    40     10 Afghanistan AF      38041757 Asia
##  6 2020-11-14 14    11     2020    66     10 Afghanistan AF      38041757 Asia
##  7 2020-12-06 06    12     2020   234     10 Afghanistan AF      38041757 Asia
##  8 2020-12-11 11    12     2020    63     10 Afghanistan AF      38041757 Asia
##  9 2020-11-27 27    11     2020   656     10 Albania     AL       2862427 Europe
## 10 2020-04-23 23    04     2020    99     10 Algeria     DZ      43053054 Africa
## # ... with 478 more rows, and 2 more variables: cases_cum <dbl>,
## #   deaths_cum <dbl>
```

```
data5<- filter(data, deaths %in% c("10", "30"))
data5
```

```
## # A tibble: 604 x 12
##    date       day   month  year cases deaths country     code  population continent
##    <date>     <chr> <chr> <dbl> <dbl>  <dbl> <chr>       <chr>      <dbl> <chr>
##  1 2020-04-27 27    04     2020    68     10 Afghanistan AF      38041757 Asia
##  2 2020-06-08 08    06     2020   791     30 Afghanistan AF      38041757 Asia
##  3 2020-07-27 27    07     2020   106     10 Afghanistan AF      38041757 Asia
##  4 2020-08-13 13    08     2020    76     10 Afghanistan AF      38041757 Asia
##  5 2020-08-25 25    08     2020    71     10 Afghanistan AF      38041757 Asia
##  6 2020-09-16 16    09     2020    40     10 Afghanistan AF      38041757 Asia
##  7 2020-11-14 14    11     2020    66     10 Afghanistan AF      38041757 Asia
##  8 2020-12-06 06    12     2020   234     10 Afghanistan AF      38041757 Asia
##  9 2020-12-11 11    12     2020    63     10 Afghanistan AF      38041757 Asia
## 10 2020-11-27 27    11     2020   656     10 Albania     AL       2862427 Europe
## # ... with 594 more rows, and 2 more variables: cases_cum <dbl>,
## #   deaths_cum <dbl>
```

We can use the function summarise() and take the mean and the median of the variable death.

```
summarise(data2, d_mean= mean(deaths), d_med= median(deaths))
```

```
## # A tibble: 1 x 2
##   d_mean d_med
##    <dbl> <dbl>
## 1   26.1     0
```

We can use the function group by() and summarise at() to create another dataframe with the selected variables

```
a<- data%>%group_by(deaths)%>%
  summarise_at(vars(cases, population), funs(n(), mean(., na.rm=TRUE)))
```

```
## Warning: ‘funs()‘ was deprecated in dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with ‘tibble::lst()‘:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_warnings()‘ to see where this warning was generated.
```

```
a
```

```
## # A tibble: 1,049 x 5
##    deaths cases_n population_n cases_mean population_mean
##     <dbl>   <int>        <int>      <dbl>           <dbl>
##  1  -1918       1            1     -372          46937060
##  2   -443       1            1      237           6415851
##  3    -31       1            1      577          60359546
##  4    -12       1            1      218           1798506
##  5     -5       1            1      466           4904240
##  6     -3       1            1       75          10650000
##  7     -2       1            1     3172          46937060
##  8     -1       1            1      121          10650000
##  9      0   36728        36728       26.0        25743880.
## 10      1    4464         4464      124.         29333910.
## # ... with 1,039 more rows
```

The min_rank() function is a function that returns the same values as rank when the ties_method is set to “min”, that is, ties are assigned the minimum ranking possible

```
rank<- data %>% group_by(deaths)%>%filter(min_rank(desc(population))==50)%>%
  dplyr::select(deaths, year, population)
rank
```

```
## # A tibble: 56 x 3
## # Groups:   deaths [20]
##    deaths  year population
##     <dbl> <dbl>      <dbl>
##  1     20  2020   44780675
##  2     38  2020   10047719
##  3     38  2020   10047719
##  4     38  2020   10047719
##  5     47  2020   11455519
##  6     45  2020   11455519
##  7     45  2020   11455519
##  8     31  2020   18952035
##  9     42  2020   18952035
## 10     42  2020   18952035
## # ... with 46 more rows
```

Finally, we can use the join () function. Specifically,inner_join()returns rows when there is a match in both tables. In this case, I am merging rank and data4 with deaths as primary key

```
d<- inner_join(rank,data4, by="deaths")
d
```

```
## # A tibble: 1,952 x 14
## # Groups:   deaths [1]
##    deaths year.x population.x date       day   month year.y cases country  code
##     <dbl>  <dbl>        <dbl> <date>     <chr> <chr>  <dbl> <dbl> <chr>    <chr>
##  1     10   2020    100388076 2020-04-27 27    04      2020    68 Afghani~ AF
##  2     10   2020    100388076 2020-07-27 27    07      2020   106 Afghani~ AF
##  3     10   2020    100388076 2020-08-13 13    08      2020    76 Afghani~ AF
##  4     10   2020    100388076 2020-08-25 25    08      2020    71 Afghani~ AF
##  5     10   2020    100388076 2020-09-16 16    09      2020    40 Afghani~ AF
##  6     10   2020    100388076 2020-11-14 14    11      2020    66 Afghani~ AF
##  7     10   2020    100388076 2020-12-06 06    12      2020   234 Afghani~ AF
##  8     10   2020    100388076 2020-12-11 11    12      2020    63 Afghani~ AF
##  9     10   2020    100388076 2020-11-27 27    11      2020   656 Albania  AL
## 10     10   2020    100388076 2020-04-23 23    04      2020    99 Algeria  DZ
## # ... with 1,942 more rows, and 4 more variables: population.y <dbl>,
## #   continent <chr>, cases_cum <dbl>, deaths_cum <dbl>
```