Covid 19 - some Challenges - some Data

Introduction

This lab has been written in the first week of April 2020.

In this lab we will investigate some of the data which can help inform issues around the Covid 19 Pandemic.

Worldwide there is a huge effort being undertaken by specialists of all colours to understand and reduce the threat of this pandemic and of course there aare huge efforts undertaken by many specialists to help us all live under these new conditions.

Here we will illuminate some of the challenges and questions to which people with data skills can contribute.

There are a number of great place to start any such journey:

- https://coronavirus.jhu.edu/ at the John Hopkins University. There you can find a collation of data but also articles on the topic. have to achieve the following tasks/learning outcomes:
- https://ourworldindata.org/coronavirus has a dedicated Covid-19 page where they review some of the latest numbers. A particularly interesting element of this page is that they provide a discussion of why they use the daily updates provided by the https://www.ecdc.europa.eu/en/geographical-distribution-2019-ncov-cases, rather than other sources. This is a particularly good case of careful datawork.
- The data competition site Kaggle has a dedicated https://www.kaggle.com/covid19 section of challenges.

In fact we start with the latter site. At the time of writing the structure their data challenges into three categories. "Use Natural Language Processing to answer key questions from the scientific literature", "Forecast COVID-19 cases and fatalities to help understand what drives transmission rates", "Curate COVID-19 related datasets to further research" and "Use exploratory analysis to answer research questions that support frontline responders".

Exploring the latter of these you will find that 12 particular tasks were posed:

- Which populations are at risk of contracting COVID-19?
- What is the incidence of infection with coronavirus among cancer patients?
- Which patient populations pass away from COVID-19?
- Are hospital resources being diverted from providing oncology care to support the COVID-19 response?
- How is the implementation of existing strategies affecting the rates of COVID-19 infection?
- Which populations have contracted COVID-19 and require ventilators?
- Which populations have contracted COVID-19 who require the ICU?
- What is the change in turnaround time for routine lab values for oncology patients?
- Which populations of clinicians are most likely to contract COVID-19?
- Which populations assessed should stay home and which should see an HCP?
- Which populations of clinicians and patients require protective equipment?
- How are patterns of care changing for current patients (i.e. cancer patients)?

Quoting from the Kaggle website: "The tasks associated with this dataset were developed and evaluated by global frontline healthcare providers, hospitals, suppliers, and policy makers. They represent key research questions where insights developed by the Kaggle community can be most impactful in the areas of at-risk population evaluation and capacity management."

Some exploratory data analysis

Let's do some exploratory analysis using a dataset published by the ECDC. This dataset is used by the Our World in Data page and is also part of the dataset in the Kaggle challenge. ## Preparing your workfile

We add the basic libraries needed for this week's work:

```r
library(tidyverse)    # for almost all data handling tasks
library(ggplot2)      # to produce nice graphiscs
library(stargazer)    # to produce nice results tables
library(AER)          # access to HS robust standard errors
library(readxl)       # enable the read_excel function
library(ggplot2)      # plotting toolbox
```

Data Upload

Very helpfully the ECDC webpage through which you can https://www.ecdc.europa.eu/en/publications-data/download-todays-data-geographic-distribution-covid-19-cases-worldwide provides an R script which allows you to download the most current dataset. This is the script replicated in the next code block. You could of course download the datset to your computer and then upload the excel or csv file, but here the ECDC has build a direct pipeline into their data.

```r
#these libraries need to be loaded
library(utils)
library(httr)

#download the dataset from the ECDC website to a local temporary file
GET("https://opendata.ecdc.europa.eu/covid19/casedistribution/csv", authenticate(":", ":", type="ntlm")
```

```
## Response [https://opendata.ecdc.europa.eu/covid19/casedistribution/csv/]
##   Date: 2020-04-07 10:30
##   Status: 200
##   Content-Type: application/octet-stream
##   Size: 476 kB
## <ON DISK>  C:\Users\msassrb2\AppData\Local\Temp\RtmpuO0FiV\file41fc3aac78b1.csv
```

```r
#read the Dataset sheet into "R". The dataset will be called "data".
data <- read.csv(tf)
```

Some data cleaning

Let's look at the structure of this dataset. We wnat to make sure we understand all the variables and give them sensible names we want to work with.

```r
str(data)
```

```
## 'data.frame':    9310 obs. of  10 variables:
##  $ dateRep                : Factor w/ 99 levels "01/01/2020","01/02/2020",..: 28 24 20 16 12 8 4 98
##  $ day                    : int  7 6 5 4 3 2 1 31 30 29 ...
##  $ month                  : int  4 4 4 4 4 4 4 3 3 3 ...
##  $ year                   : int  2020 2020 2020 2020 2020 2020 2020 2020 2020 2020 ...
##  $ cases                  : int  38 29 35 0 43 26 25 27 8 15 ...
##  $ deaths                 : int  0 2 1 0 0 0 0 0 1 1 ...
##  $ countriesAndTerritories: Factor w/ 204 levels "Afghanistan",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ geoId                  : Factor w/ 203 levels "AD","AE","AF",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ countryterritoryCode   : Factor w/ 201 levels "","ABW","AFG",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ popData2018            : int  37172386 37172386 37172386 37172386 37172386 37172386 37172386 3717
```

Some of the variables have obvious meaning, sich as `day`, `month`, `year`, `countriesAndTerritories` and `popData2018`, the latter giving the population of the respective country in 2018. `geoId` and `countryterritoryCode` are common appreviations for the respective country.

For starters we want to shorten the name of `countriesAndTerritories` to `country` and `countryterritoryCode` to `countryCode` and `dateRep` to `dates`.

```
names(data)[names(data) == "countriesAndTerritories"] <- "country"
names(data)[names(data) == "countryterritoryCode"] <- "countryCode"
names(data)[names(data) == "dateRep"] <- "dates"
```

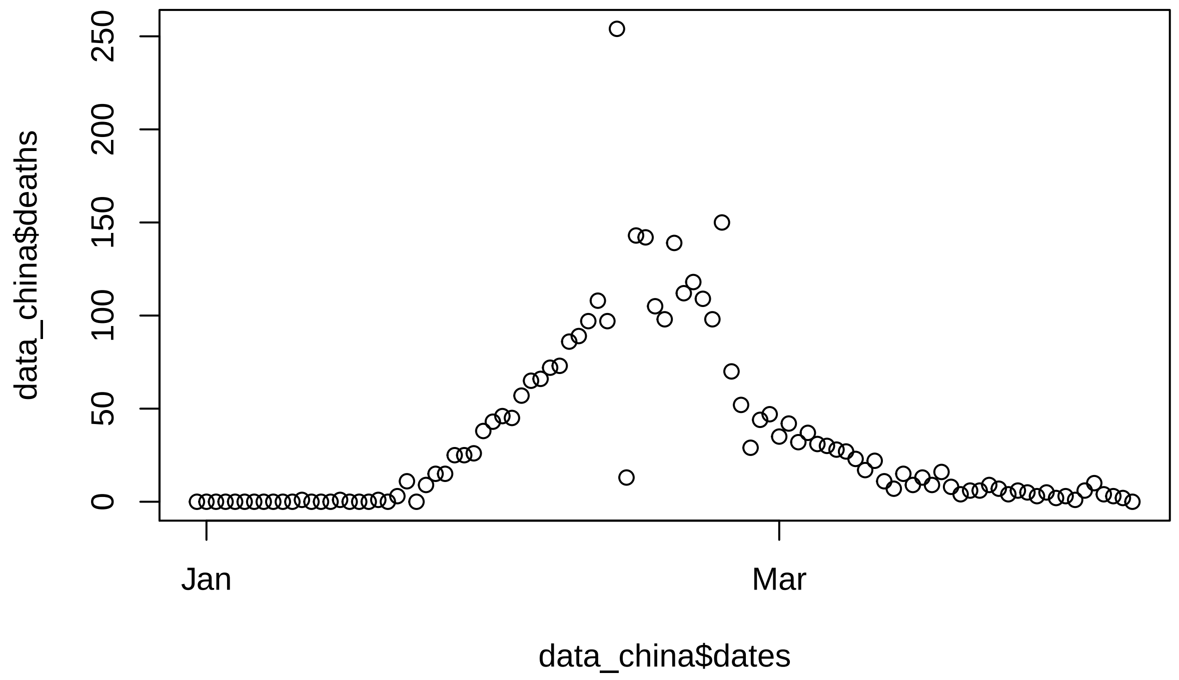
The first variable is a factor variable which includes the entire date string. At this stage R does not recognise this as a date. So let's change this as it will be useful for R to know that this variable represents a date. Dates are of the format 24/01/2020. In the `format` option we specify this date such that R knows how to translate to dates (see https://www.stat.berkeley.edu/~s133/dates.html to understand how to specify the format string).

```
data$dates <- as.Date(as.character(data$dates),format = "%d/%m/%Y")
```

Last, but most importantly, there are two variables called `cases` and `deaths`. These are daily data. So without further explanation it is not obvious whether these are the cumulative data (e.g. all the Covid-19 cases which have been identified in a country by a certain day, or whether these are the cases identified on that particular day). YOu could either go back to the data source to find an explanation orwe could use our understanding of the problem at hand. The two series should look very different.

To investigate this let's look at one country in particular, say China, where this particular virus was first identified. We use the plot function which provides the standard build-in R plotting facility. Later we will look at using `ggplot` to produce nicer plots.

```
data_china <- data %>%  filter(country == "China")
plot(data_china$dates,data_china$deaths)  # specifies variable for x and y axis
```



We can clearly see that after an increase in the numbers of fatalities in China in January and February we see a decrease in numbers in March. If these were cumulaive numbers we would not see a decrease. So these are the deaths which occured on a particular day. You can find the same for cases.

Before continuing we may also want to highlight a particularity in these data. YOu can see that in the middle of March there is one day (13 Feb 2020) on which almost 100 more deaths have been reported than at any other day. And in fact, the day before tehre were only 13 reported deaths. It turns out that these irregularities are the result of changes in data definitions as reported, for instance, in this https://www.cnbc.com/2020/02/26/confusion-breeds-distrust-china-keeps-changing-how-it-counts-coronavirus-cases.html.

Knowing that we are talking about daily statistics for new confirmed infections and daily deaths, we may also want to calculate the accumulated infections and deaths. This is achieved with the `cumsum` (cumulative sum) command. This takes a vector of data and keeps adding these up.

To illustrate what this command does, let's use an example.

```
test <- c(0,0,2,4,9,2)
cumsum(test)
```

```
## [1]  0  0  2  6 15 17
```

Before we apply this to our data in `data`, we need to make sure that we only accumulate by country (`group_by(country)`) and that the data are arranged by date (`arrange(dates)`) before we apply the `cumsum` function.

```
data <- data %>% group_by(country) %>%
          arrange(dates) %>%
          mutate(c_cases = cumsum(cases), c_deaths = cumsum(deaths)) %>%
        ungroup()
```

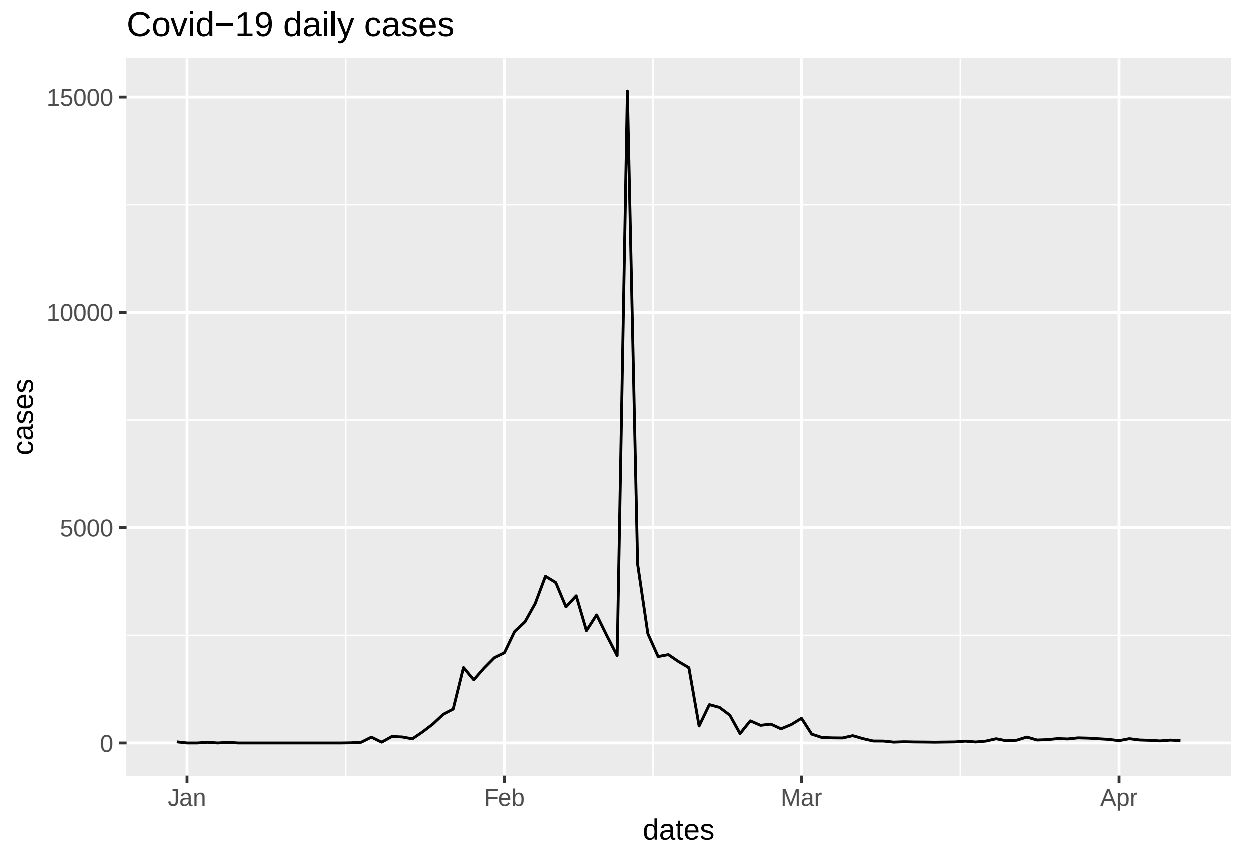
In the variables `c_cases` and `c_deaths` we now have the accumulated cases and deaths by country. At the end we undid the `group_by` by calling `ungroup()`.

Some country graphs

Let's create some nicer graphs to describe the development of the pandemic in different countries. Let's first continue with the Chinese data.

First we replicate the above Figure but using the `ggplot` function which produces much nicer graphs.

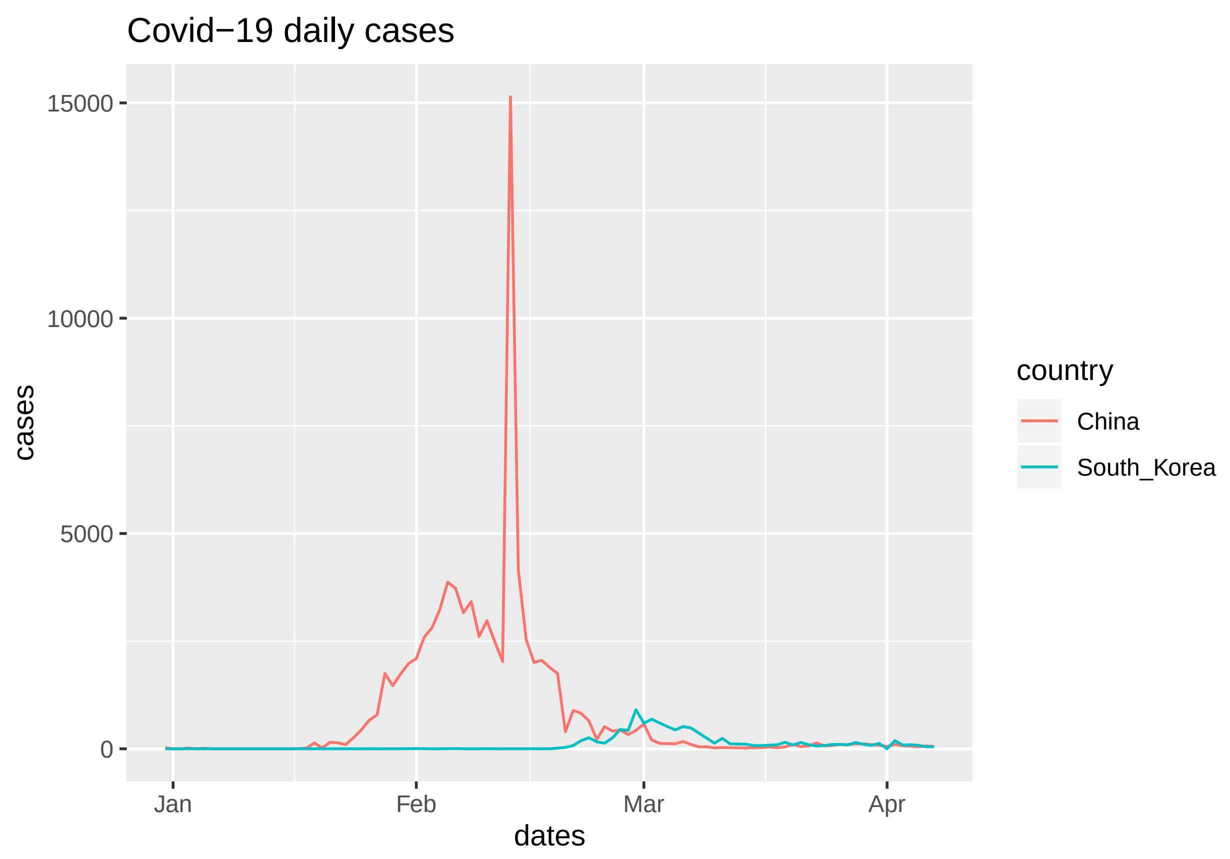
```
g1 <- ggplot(subset(data, country == "China"), aes(x=dates,y=cases)) +
      geom_line() +
      ggtitle("Covid-19 daily cases")
g1
```



As you can see we didn't create a special Chinese dataset first, but we used the subset function instead.

Let's overlay the daily cases for two countries, say China and South Korea.

```
sel_countries <- c("China", "South_Korea")
g2 <- ggplot(subset(data, country %in% sel_countries),
             aes(x=dates,y=cases, color = country)) +
      geom_line() +
      ggtitle("Covid-19 daily cases")
g2
```

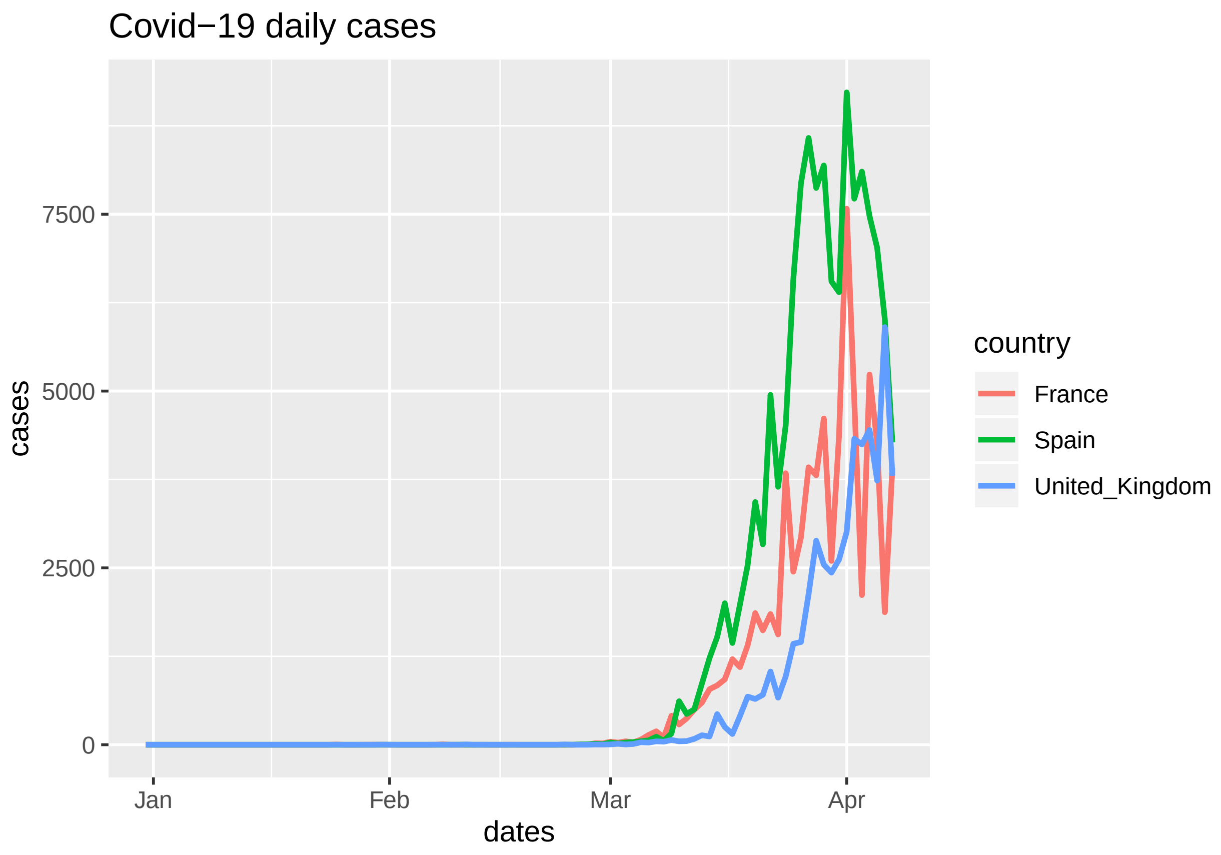


Here you can see the much praised ability by South Korea to suppress the numbers of infections effectively. However, you might argue that

> Explore: How do these look like if you were to look at per capita infection rates?

Now we look at some European countries.

```r
sel_countries <- c("Spain", "France", "United_Kingdom")
g3 <- ggplot(subset(data, country %in% sel_countries),
             aes(x=dates,y=cases, color = country)) +
      geom_line(size = 1) +   # size controls the line thickness
      ggtitle("Covid-19 daily cases")
g3
```

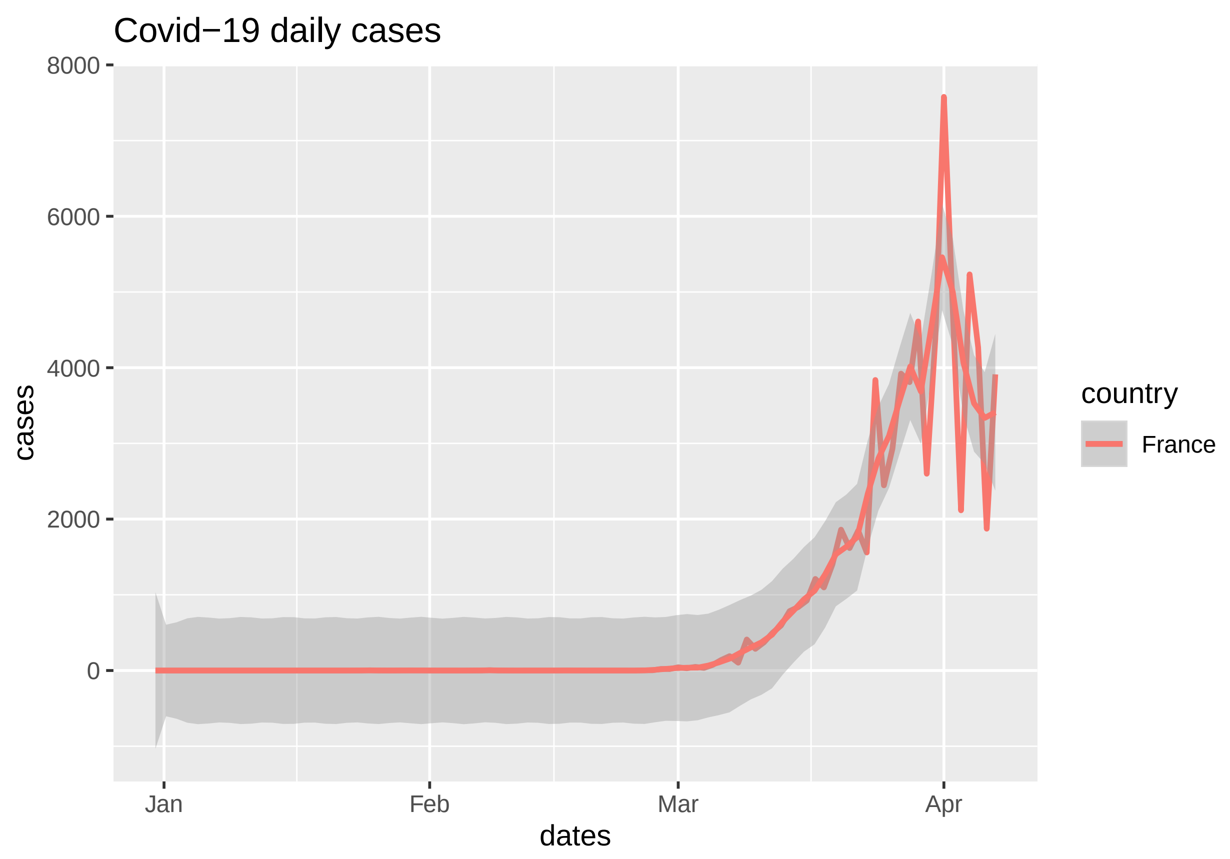


You can see that especially the French data are very variable. It is important to understand that the number of newly identified cases is also a function of the testing effort on a particular day and if that is variable then we may well expect a huge variability.

In fact that much variation that it was difficult to separate the signal from the noise. As this is a typical issue with high frequency data we often see a smoothed version of these data. `ggplot` has a build-n function which allows you to see smoothed versions of the data, `geom_smooth(method = "loess")`. This averages data in some local (time) window. We will not discuss the details of the algorithm here.

We start by focusing on the French data and add the smoothed version.

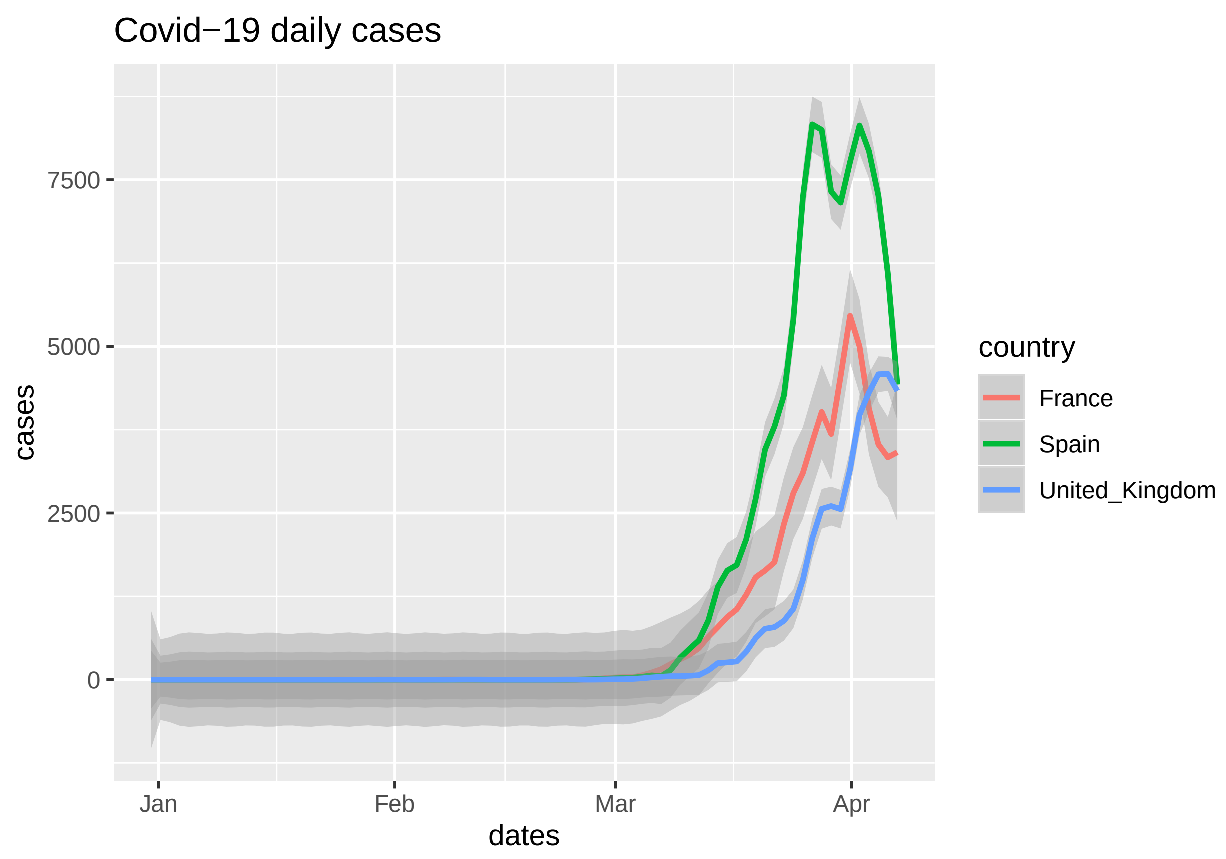
```
sel_countries <- c("France")
g4 <- ggplot(subset(data, country %in% sel_countries),
             aes(x=dates,y=cases, color = country)) +
      geom_line(size = 1) +    # size controls the line thickness
      geom_smooth(method = "loess", span = 0.1) + # smoothed data
      ggtitle("Covid-19 daily cases")
g4
```



You can see that the smoothed version clearly shows the exponential trend. However, the algorithm used also results in the smoothed version temporarily becoming negative. Change the values of the `span` option to figure out how it changes the results.

Let's look at the three countries again and only show the smoothed versions.

```
sel_countries <- c("Spain", "France", "United_Kingdom")
g5 <- ggplot(subset(data, country %in% sel_countries),
             aes(x=dates,y=cases, color = country)) +
      geom_smooth(method = "loess", span = 0.1) + # smoothed data
      ggtitle("Covid-19 daily cases")
g5
```

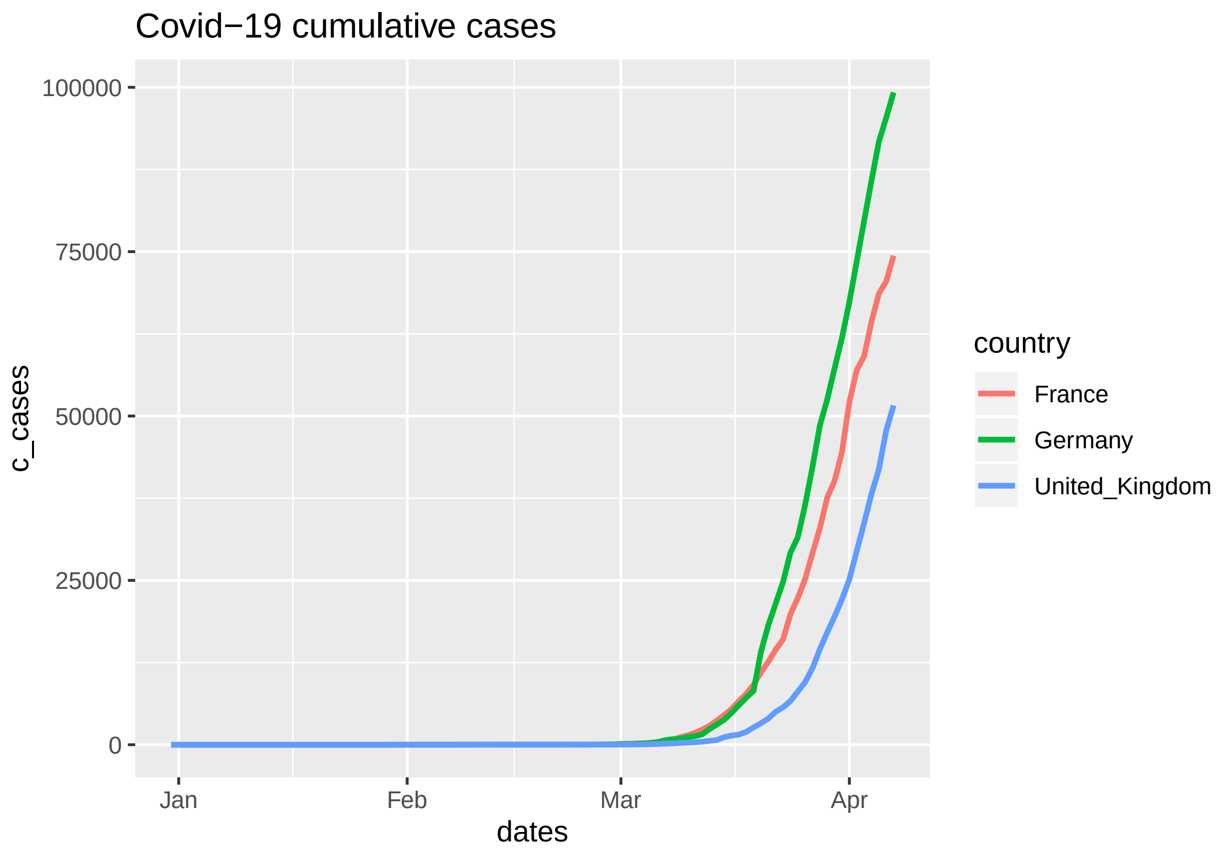


The grey areas around the smoothed data versions are confidence intervals. They recognise that the data are noisy and it is impossible to precisely get the underlying trend. As the French data are noisier than those of the UK and Spain the grey error bands are somewhat wider than the others, indicating larger uncertainty about the trend.

But smoothing can be useful. Here for instance we can perhaps see that France and Spain may have reached the peak of new registered cases, where, at the time of writing (5 April 2020), the UK numbers were still on a clearly positive trend.

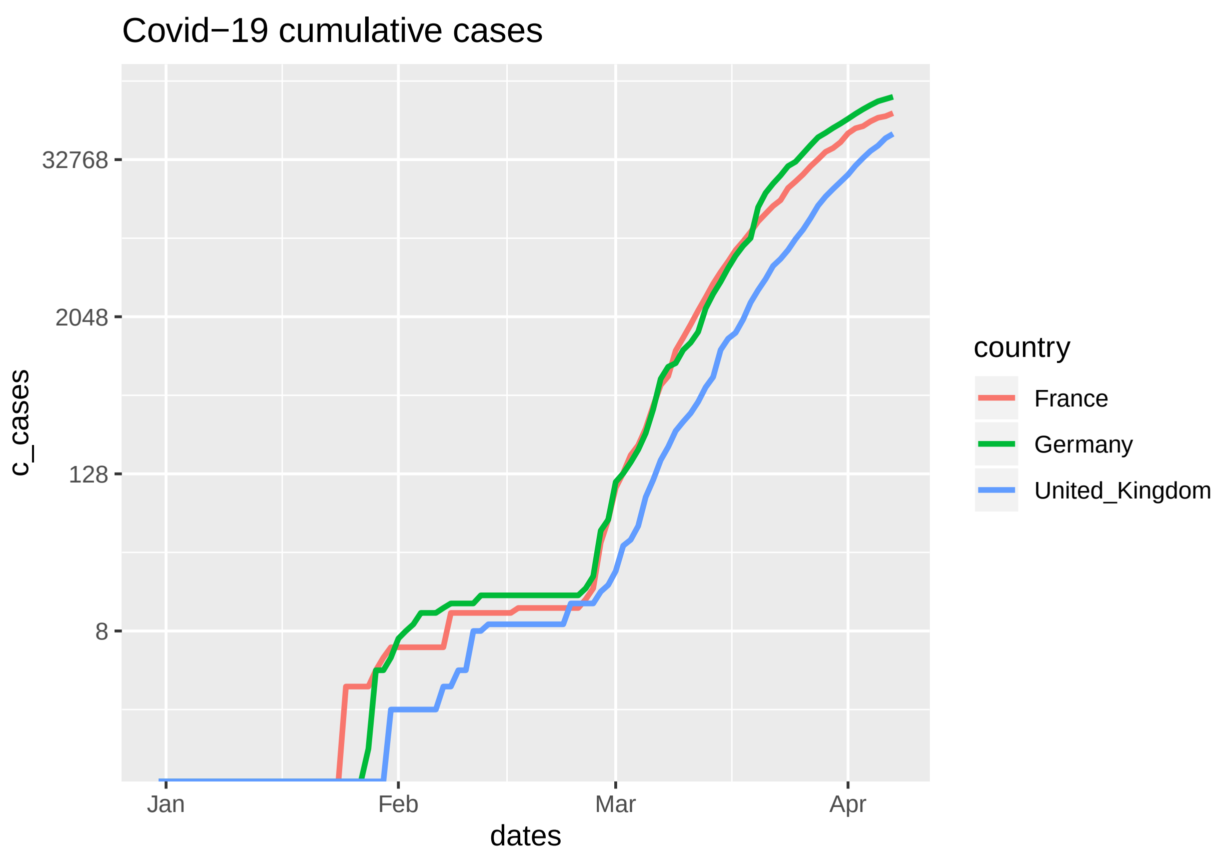
Let's also look at the cumulative case numbers.

```
sel_countries <- c("Germany", "France", "United_Kingdom")
g6 <- ggplot(subset(data, country %in% sel_countries),
             aes(x=dates,y=c_cases, color = country)) +
      geom_line(size = 1) +
      ggtitle("Covid-19 cumulative cases")
g6
```



You can clearly see the exponential growth in the number of cases. Sometimes you will see these graphs on a logarithmic scale.

```
sel_countries <- c("Germany", "France", "United_Kingdom")
g7 <- ggplot(subset(data, country %in% sel_countries),
             aes(x=dates,y=c_cases, color = country)) +
      geom_line(size = 1) +
      scale_y_continuous(trans='log2') +
      ggtitle("Covid-19 cumulative cases")
g7
```



From this you can clearly see some early dynamics which is not visible from the natural scale. But looking at the logarithmic scale takes away the dramatic impression the natural scale shows.

Some country maps

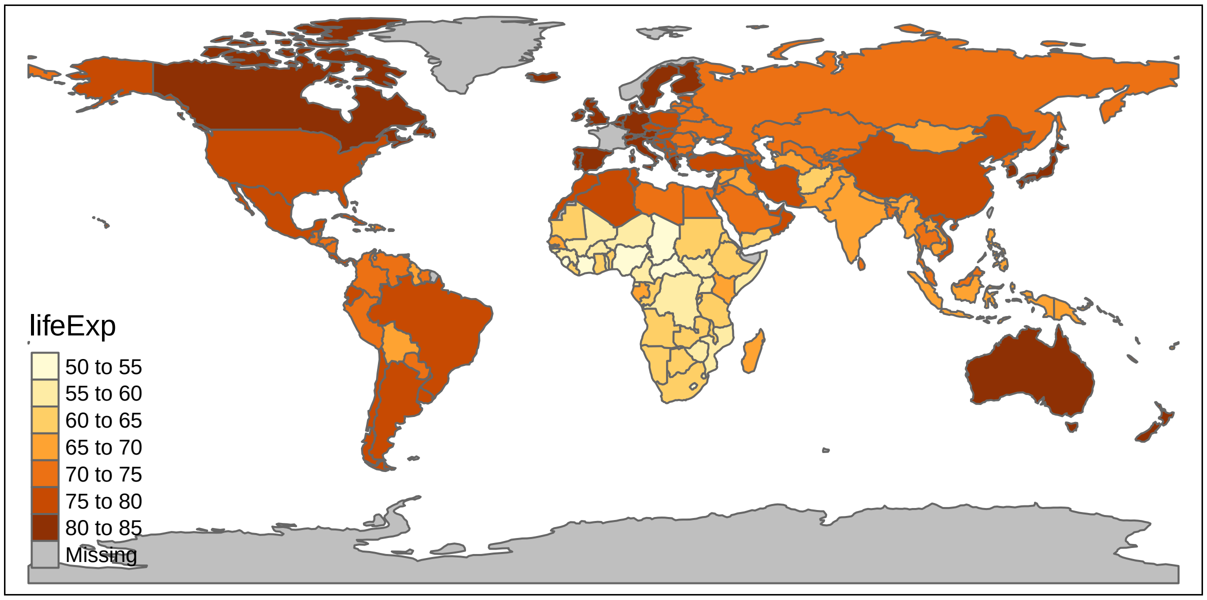
Maps are a great tool to illustrate the geographic distribution of any statistics.

We need a few more packages at this stage.

```
library(sf)
library(raster)
library(dplyr)
library(spData)
library(tmap)
```

Let's create a map first and then we will find out how to manipulate the map to display what we want.

```
# Add fill and border layers to world shape
tm_shape(world) + tm_polygons(col = "lifeExp")
```



Wow, one line of code and you get a world map which shows which countries have the highest and lowest life expectation. Amazing! What did the code do? We saved the map in a new object `m1`. We used the `tmap` package (Check out https://geocompr.robinlovelace.net/ by Robin Loveace for an introduction into this package and geocomputing in general). It has a range of map information at its fingertips and `tm_shape(world)` basically called up shape information on the world's countries with some basic information. Then we add the information which should determine the colors (`+ tm_fill(col = "lifeExp")`)and finally we add borders (`+ tm_polygons()`). Clearly we will want to replace the information on life expectatncy with information on Covid-19 cases. TO do this we first need to understand where tmaps stores the data on life expectatncy.

Hence we need to slightly deconstruct the above code

```
m2 <- tm_shape(world)
str(m2)
```

```
## List of 1
##  $ tm_shape:List of 12
##   ..$ shp_name   : chr "world"
##   ..$ shp        :Classes 'sf', 'tbl_df', 'tbl' and 'data.frame':    177 obs. of  11 variables:
##   .. ..$ iso_a2   : chr [1:177] "FJ" "TZ" "EH" "CA" ...
##   .. ..$ name_long: chr [1:177] "Fiji" "Tanzania" "Western Sahara" "Canada" ...
##   .. ..$ continent: chr [1:177] "Oceania" "Africa" "Africa" "North America" ...
##   .. ..$ region_un: chr [1:177] "Oceania" "Africa" "Africa" "Americas" ...
##   .. ..$ subregion: chr [1:177] "Melanesia" "Eastern Africa" "Northern Africa" "Northern America" ..
##   .. ..$ type     : chr [1:177] "Sovereign country" "Sovereign country" "Indeterminate" "Sovereign c
##   .. ..$ area_km2 : num [1:177] 19290 932746 96271 10036043 9510744 ...
##   .. ..$ pop      : num [1:177] 8.86e+05 5.22e+07 NA 3.55e+07 3.19e+08 ...
##   .. ..$ lifeExp  : num [1:177] 70 64.2 NA 82 78.8 ...
##   .. ..$ gdpPercap: num [1:177] 8222 2402 NA 43079 51922 ...
##   .. ..$ geom     :sfc_MULTIPOLYGON of length 177; first list element: List of 3
##   .. .. ..$ :List of 1
##   .. .. .. ..$ : num [1:8, 1:2] 180 180 179 179 179 ...
##   .. .. ..$ :List of 1
##   .. .. .. ..$ : num [1:9, 1:2] 178 178 179 179 178 ...
##   .. .. ..$ :List of 1
##   .. .. .. ..$ : num [1:5, 1:2] -180 -180 -180 -180 -180 ...
##   .. .. ..- attr(*, "class")= chr [1:3] "XY" "MULTIPOLYGON" "sfg"
##   .. ..- attr(*, "sf_column")= chr "geom"
##   .. ..- attr(*, "agr")= Factor w/ 3 levels "constant","aggregate",..: NA NA NA NA NA NA NA NA NA NA
##   .. .. ..- attr(*, "names")= chr [1:10] "iso_a2" "name_long" "continent" "region_un" ...
##   ..$ name       : NULL
##   ..$ is.master  : logi NA
##   ..$ projection : NULL
##   ..$ bbox       : NULL
##   ..$ unit       : NULL
##   ..$ simplify   : num 1
##   ..$ point.per  : logi NA
##   ..$ line.center: chr "midpoint"
##   ..$ filter     : NULL
##   ..$ check_shape: logi TRUE
##  - attr(*, "class")= chr "tmap"
```

So, this looks complicated. `m2` is a list with one element called `sm_shape`. That in turn is a list with 12 elements. Importantly one of these elements is called `shp` and this contains information on the respective countries.

Let's just look at that to understand what it looks like.

```r
temp <- m2$tm_shape$shp
names(temp)
```

```
##  [1] "iso_a2"    "name_long" "continent" "region_un" "subregion" "type"
##  [7] "area_km2"  "pop"       "lifeExp"   "gdpPercap" "geom"
```

This is really a traditional data frame with country specific information and you can see that one of the pieces of information here is life expectation (`lifeExp`). This is where `tmap` got the info from. We will insert into this spreadsheet the information on cases and then use that to display the data. Importantly, `iso_a2` is a variable with a country appreviation. As we have this information also in our `data` set we will use that to merge the data.

```r
temp_mergein <- data %>% filter(dates == "2020-04-04") %>%
                          select(geoId, cases, c_cases, deaths, c_deaths)
```

When you run this you are likely to get the following error message

```
Error in (function (classes, fdef, mtable)  :
  unable to find an inherited method for function ‘select’ for signature ‘"tbl_df"’
```

I had to google the error message and found out that the select function stops working like this as the `raster` function also has a `select` function (type `?select` into the command window to see this). When we loaded the `raster` package the `select` function of the `dplyr` package (automatically loaded with the `tidyverse`) which is what we want.

So when we want to run the above command we have to tell R that we want the select function of the dplyr package (`dlyr::select`).

```
temp_mergein <- data %>% filter(dates == "2020-04-04") %>%
                            dplyr::select(geoId, cases, c_cases, deaths, c_deaths)
```

We only selected the actual variables we are interested in and the `geoId` variable as this will be the variable which will be matched with `iso_a2`.

As it turns out the country code for the United Kingdom in `data` (and hence in `temp_mergein`) is `UK` and in the shape file it is `GB`. So we need to change on of them to enable a match. If you didn't do this you would find that the map shows missing data for the United Kingdom.

```
temp_mergein$geoId <- as.character(temp_mergein$geoId)
temp_mergein$geoId[temp_mergein$geoId == "UK"] <- "GB"
```

Now we will `merge` this info into `m2` such that we have our COVID-19 data available to map. We specify the respective variables used to match the data (`by.x = "iso_a2"`, `by.y = "geoId"`) and also ensure that we keep all of our original country info, even if there is no COVID-19 information (`all.x = TRUE`).

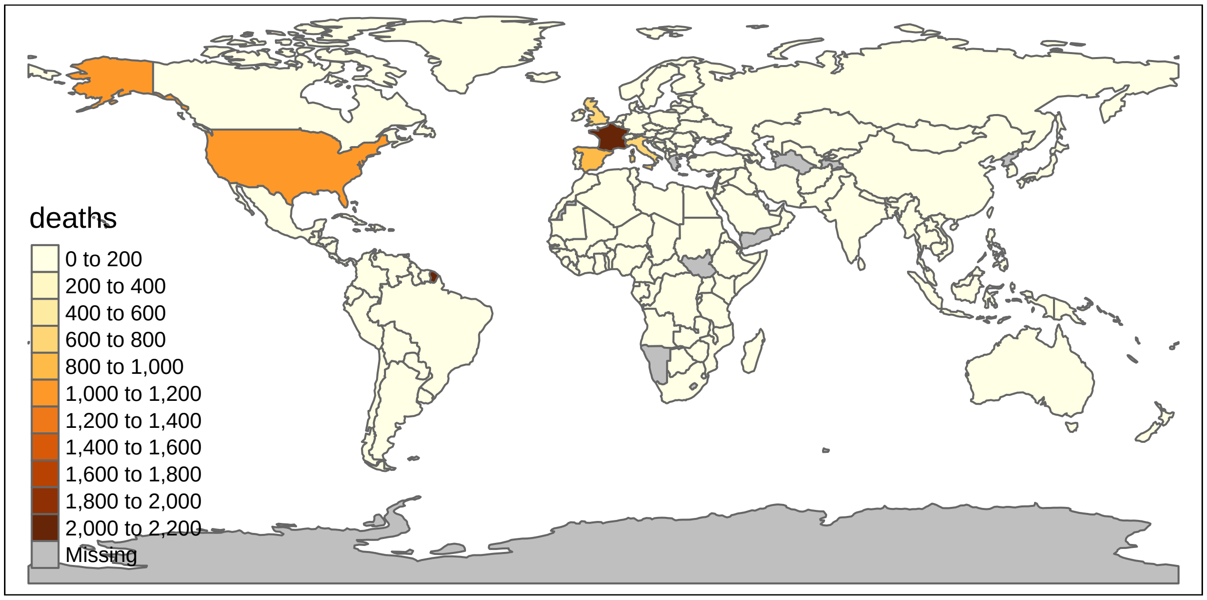
```
temp <- merge(temp, temp_mergein, by.x = "iso_a2", by.y = "geoId", all.x = TRUE)
```

Now that we manipulated the datafile at the core of the mapping operation we need to insert it back into the `m2` file into exactly the same spot where we found that datafile in the first place (`m2$tm_shape$shp`).

```
m2$tm_shape$shp <- temp
```

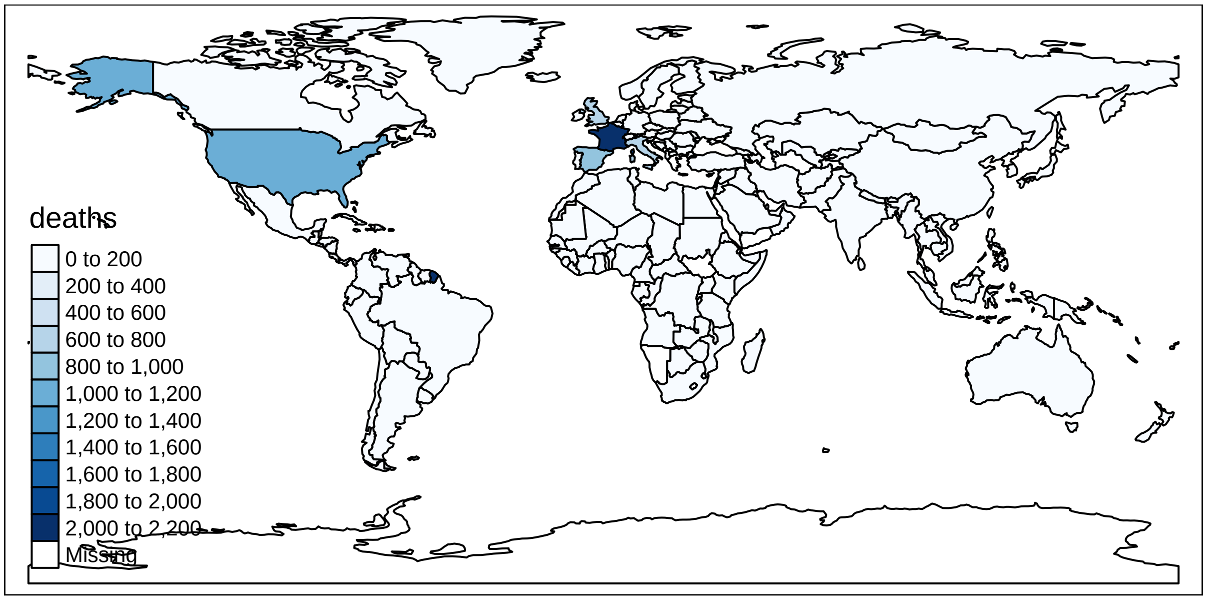
Now we can return to the mapping code

```
# Add polygons layer to world shape
m2 + tm_polygons(col = "deaths", n=10)   # n = 10 controls the number of categories
```



There is all sorts of things you can change about these maps.

```
m2 + tm_polygons(col = "deaths", n=10) +  # n = 10 controls the number of categories
      tm_style("col_blind")
```



More useful maps

The above maps were created for one particular day (“04/04/2020”). Could we create multiple maps for each day in 2020 so that we can visualise the spread of the pandemic. The answer is yes. Let’s start by creating the base world map again.

```
m3 <- tm_shape(world) # create a new shape file from scratch
temp3 <- m3$tm_shape$shp # extract the data frame with the data
```

First we need to ensure that we insert the daily information into `m2` just as we inserted the info for the one day. What we will do is to select one day a week, starting with the 15th of Jan 2020. This is done in `date_sel` using the sequence (`seq` function). Then we select all the information available for these dates from `data`.

```
# prepare the data from data
date_sel <- seq(as.Date("2020-01-15"), as.Date("2020-04-04"), 7)
temp_mergein <- data %>% filter(dates %in% date_sel) %>%
                  dplyr::select(geoId, dates, cases, c_cases, deaths, c_deaths) %>%
                  arrange(geoId,dates)
temp_mergein$geoId <- as.character(temp_mergein$geoId)
temp_mergein$geoId[temp_mergein$geoId == "UK"] <- "GB"
```

So far we extracted all our data, just for more dates. We are now also keeping the `dates` variable.

Now we need to merge the new data into `temp3`.

```
temp3 <- merge(temp3, temp_mergein, by.x = "iso_a2", by.y = "geoId", all.x = TRUE)
temp3$dates <- as.character(temp3$dates)  # tmaps doesn't like date formats I think
temp3 <- temp3 %>%  filter(!is.na(dates)) # delete countries with no data
```

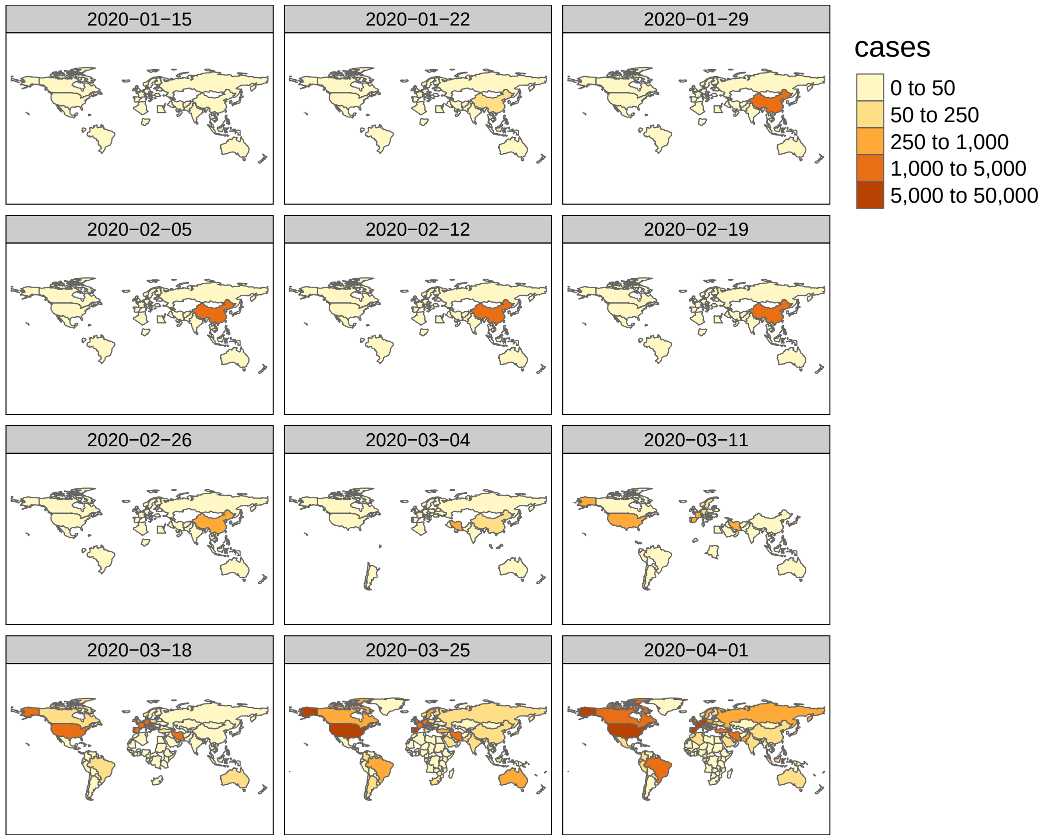
Here we turned the dates into the character format (as `tmap` seemed to dislike date formats) and we now delete countries with no available observations as these would otherwise introduce an extra `na` date.

This updated data frame is now inserted back into the shape file.

```
m3$tm_shape$shp <- temp3
```

Now we need to instruct `tmap` to produce the maps, but now a separate map for every day. The `tmap` command which is useful here is `tm_facets(by = "dates", free.coords = FALSE)`. The `by = "dates"` option selects the `dates` variable to be the facet variable, i.e. the variable for which we calculate separate maps for every possible outcome (i.e. day).

```
covid_fac <- m3 + tm_polygons(col = "cases",
                              style = "fixed", # use fixed categories
                              breaks = c(0,50,250,1000,5000,50000)) + # set category boundaries
  tm_facets(by = "dates", free.coords = FALSE)
covid_fac
```



The result of this calculation was saved in `covid_fac`. As you can see, we do not have an observation from every country on all days. If we are missing an observation then, for the particular day the country will not appear on the map.

Correlate Covid-19 data with other indicators

In this section we will use a range of other country-wide indicators andcheck whether they.

The datasets we will look at here are

- number of international tourism arrivals
- percentage of population with access to hand-washing facilities
- number of hospital beds
- number of nurses and midwives

The first dataset mainly as it may allow us to find a proxy for how the pandemic spreads around the world. The others as they relate to how well a country may either pprevent the spread of the virus (handwashing facilities) and how well their health system may be able to cope with any significant outbreak.

International tourism travel

Let's load the data

```
data_tourism <- read_csv("international-tourism-number-of-arrivals.csv")
names(data_tourism)
```

```
## [1] "Entity" "Code"   "Year"   "X4"
```