R Coding Practice

Coding Skill Focus

Ralf Becker

June 2020

Aim for today

- Become familiar with some Covid related data
- Upload data from csv
- Undertake some basic data exploration
- Use time formats
- Create time-series graphs

In addition to this you will practice the three crucial coding skills of:

- Using the help function
- Searching the internet for solutions
- Trial and error
- Finding mistakes (debugging)

I assume that you have a good working knowledge of R, including some experience with tidyverse and ggplot.

Why, as economists, should we look at Covid-19

- Understanding the current and future needs are important for business and government for planning (toilet paper producers, fresh food importers, pasta retailers, NHS hospitals, etc)
- An event which allows us to reconsider the interplay between markets, government, and civil society

Data used

Today we will use two data sources.

1. Google mobility data, from https://www.google.com/covid19/mobility/.
2. Data from https://www.bsg.ox.ac.uk/research/research-projects/coronavirus-government-response-tracker. This dataset combines a range of Covid statistics, like the number of affected and the number of deseased.

Setup

Load the libraries needed.

```r
library(tidyverse)    # for almost all data handling tasks
library(ggplot2)      # plotting toolbox
```

Setup your working directory.

```
setwd("YOUR_WORKING_DIRECTORY")
```

or by using the menu - Session - Set Working Directory - To Source File Location

Import and examine Google mobility data

Go to https://www.google.com/covid19/mobility/ and download the “Global_Mobility_Report.csv” and save it in your working directory folder. We use the `read.csv` function to load the data andsave them into the `mob_data` dataframe. Also run the `str` function to see what variable are included.

```
mob_data <- Read.Csv("Global_Mobility_Report.csv")
str(mob_data)
```

When you run this code you will encounter an error message

```
could not find function "Read.Csv"
```

The error message tells you that R could not find the funcrion `Read.Csv`.

> Try to fix the code keeping in mind that R is case-sensitive. As you work with R you willoften get error messages and it is an important skill to use the information in these to help you find the problem. You could type `?Read.Csv` into the console. Typically as you type R will actually suggest available functions.

Once you fixed the code you should see this output.

```
'data.frame':   477322 obs. of  11 variables:
 $ country_region_code                               : Factor w/ 131 levels "AE","AF","AG",..: 1 1 1 1
 $ country_region                                    : Factor w/ 132 levels "Afghanistan",..: 124 124 12
 $ sub_region_1                                      : Factor w/ 1845 levels "","Ã-rebro County",..: 1
 $ sub_region_2                                      : Factor w/ 1716 levels "","Abbeville County",..:
 $ date                                              : Factor w/ 105 levels "01/03/2020","01/04/2020",.
 $ retail_and_recreation_percent_change_from_baseline: int  0 1 -1 -2 -2 -2 -3 -2 -1 -3 ...
 $ grocery_and_pharmacy_percent_change_from_baseline : int  4 4 1 1 0 1 2 2 3 0 ...
 $ parks_percent_change_from_baseline                : int  5 4 5 5 4 6 6 4 3 5 ...
 $ transit_stations_percent_change_from_baseline     : int  0 1 1 0 -1 1 0 -2 -1 -1 ...
 $ workplaces_percent_change_from_baseline           : int  2 2 2 2 2 1 -1 3 4 3 ...
 $ residential_percent_change_from_baseline          : int  1 1 1 1 1 1 1 1 1 1 ...
```

There are 477322 rows of data and 11 variables. We have geographical information and information on activity indices. More detail on the latter soon.

You can see that there is a `date` variable which contains date information.

```
head(mob_data$date)
```

```
[1] 15/02/2020 16/02/2020 17/02/2020 18/02/2020 19/02/2020 20/02/2020
105 Levels: 01/03/2020 01/04/2020 01/05/2020 02/03/2020 ... 31/03/2020
```

This is currently formatted as a `factor` variable, i.e. a categorical variable. Let’s use the `as.Date` function to convert this variable into a date format such that R recognises that these are dates.

Dates can be formatted in number of ways, eg. 23 March 2020, 23/2/2020, 2020-03-23, can all stand for the same date. For teh `as.Date` function to work you will have to let it know how the data you are feeding in are formatted.

> Can you figure out which of the following three date formatting instructions fit to the above versions * %d %B %Y * %Y-%m-%d * %d/%m/%Y

In order to understand what these do you should use your favourite serach engine and find some help, e.g. “R as.Date date formats”. Look at the highest rated link and you should get some help on the meaning of these strings.

> Try which of these works for our dataset. Either by substituting one of the above for `XXXX` or by looking at the dataset to see how the dates are formatted (you need to keep the quotation marks!). Also have a look at the examples in the help entry for `as.Date` (by calling `?as.Date` from the command window).

```
mob_data$date <- as.Date(as.character(mob_data$date),"XXXX")
head(mob_data$date) # this just displays the first
```

Once you have done this you need to look at the dates. If you translated them correcty they will look like below. If you havn’t then you are likely to see `NA`s. If that is the case, then you will have to execute the aove line in which you imported the data again, as you have now removed the actual date information!

```
[1] "2020-02-15" "2020-02-16" "2020-02-17" "2020-02-18" "2020-02-19"
[6] "2020-02-20"
```

Let’s look at a small subset of the data, in particular we pick out three comparable city regions. The regional information is saved in `sub_region_1` and we create a list with the three regions, `region_sel`. We then filter all observations from `mob_data` which belong to one of these three regions. Do do so we use the `%in%` operator. In words this operator does something like “chose all values which match one of the values in the following list”.

> Search the internet to figure out how to use this operator given the remaining information, to filter out all the rows which belong to one of the regions in `region_sel`. Replace all `XXXX` in the following code chunk. If you do it correctly you should find approximately 315 observations in `mob_data_sel` (a few more if you downloaded the file later than 6 June 2020).

```
region_sel <- c("Greater Manchester", "Stockholm County", "Berlin")

mob_data_sel <- mob_data %>%
  filter(sub_region_1 XXXX XXXX)

nrow(mob_data)
nrow(mob_data_sel)
```

```
[1] 477322
```

```
[1] 315
```

Let’s look at some summary statistics.

```
summary(mob_data_sel)
```

```
 country_region_code             country_region                sub_region_1
 DE     :105         Germany            :105    Berlin              :105
 GB     :105         Sweden             :105    Greater Manchester   :105
 SE     :105         United Kingdom     :105    Stockholm County     :105
 AE     :  0         Afghanistan        :  0                         :  0
 AF     :  0         Angola             :  0    Ã-rebro County       :  0
 AG     :  0         Antigua and Barbuda:  0    Ã-stergÃ¶tland County:  0
 (Other):  0         (Other)            :  0    (Other)              :  0
           sub_region_2      date
                 :315   Min.   :2020-02-15
 Abbeville County:  0   1st Qu.:2020-03-12
 Acadia Parish   :  0   Median :2020-04-07
 Accomack County :  0   Mean   :2020-04-07
 Ada County      :  0   3rd Qu.:2020-05-03
 Adair County    :  0   Max.   :2020-05-29
 (Other)         :  0
 retail_and_recreation_percent_change_from_baseline
 Min.   :-90.00
 1st Qu.:-63.00
 Median :-30.00
 Mean   :-35.37
 3rd Qu.: -8.00
 Max.   : 11.00

 grocery_and_pharmacy_percent_change_from_baseline
 Min.   :-93.000
 1st Qu.:-17.000
 Median : -6.000
 Mean   : -9.381
 3rd Qu.:  0.000
 Max.   :107.000
 NA's   :3
 parks_percent_change_from_baseline
 Min.   :-63.000
 1st Qu.:-13.000
 Median :  6.000
 Mean   :  8.594
 3rd Qu.: 22.500
 Max.   :148.000

 transit_stations_percent_change_from_baseline
 Min.   :-82.00
 1st Qu.:-59.00
 Median :-43.00
 Mean   :-39.18
 3rd Qu.:-11.00
 Max.   :  6.00

 workplaces_percent_change_from_baseline
 Min.   :-87.00
 1st Qu.:-53.00
 Median :-35.00
 Mean   :-32.23
 3rd Qu.: -6.00
 Max.   :  2.00

 residential_percent_change_from_baseline
 Min.   :-1.00
 1st Qu.: 3.00
 Median :12.00
 Mean   :11.84
 3rd Qu.:18.00
 Max.   :32.00
```

You can see that there are 6 activity indices. Google extract these from the detailed user data they. For instance the `workplaces_percent_change_from_baseline` provides information on the extend to which they detected activities in workplaces. The numbers are percentage changes relative to a baseline. It is important to understand what the baseline is. This is a general point, you need to understand data definitions. On the https://www.google.com/covid19/mobility/data_documentation.html?hl=en you can find the following:

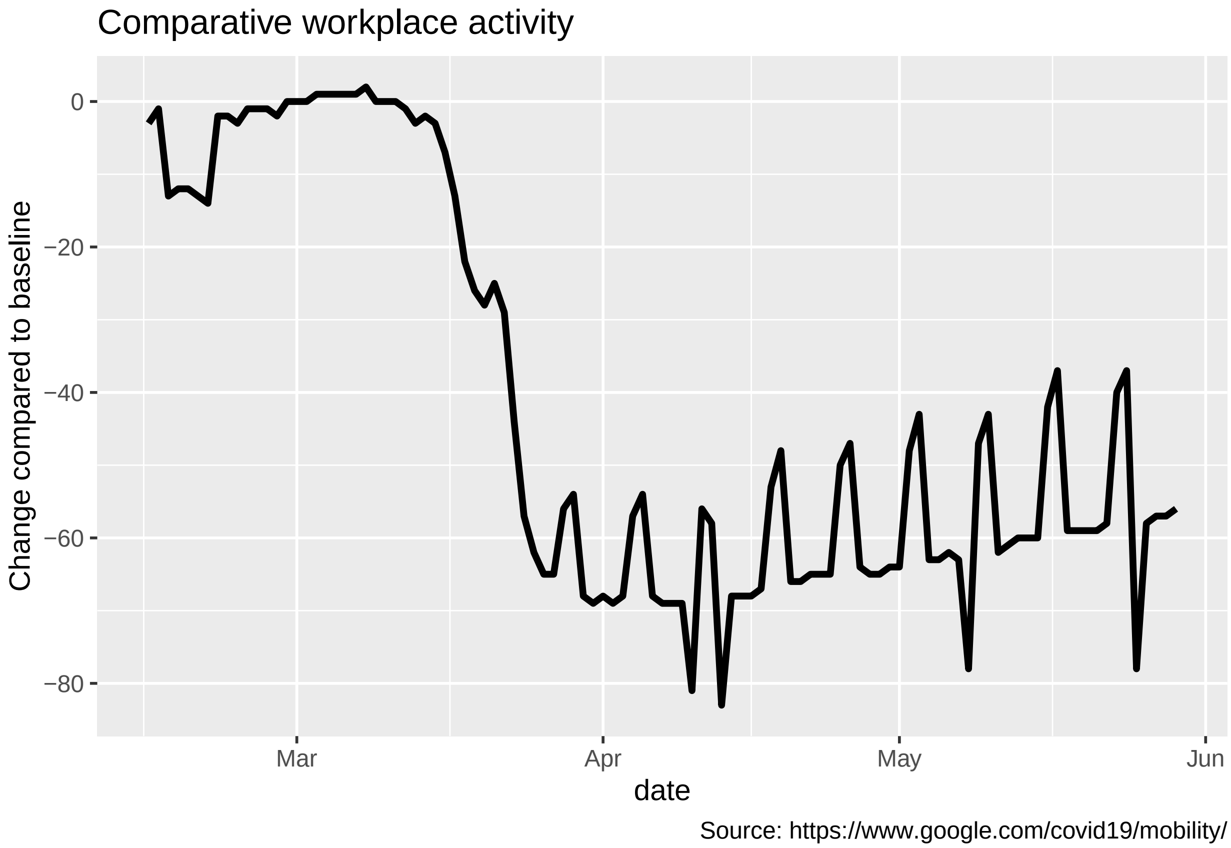
"Changes for each day are compared to a baseline value for that day of the week: The baseline is the median value, for the corresponding day of the week, during the 5-week period Jan 3–Feb 6, 2020.""

Some data plots

Let's plot a few of the activity inices using the `ggplot` function.

First we wpick out one of the locations, `Greater Manchester` and plot `workplaces_percent_change_from_baseline`. We achieve this by first creating a new (temporary) dataset, `temp` which only contains data from Manchester. We call it temp as we don't expect to need that data file afterwards.

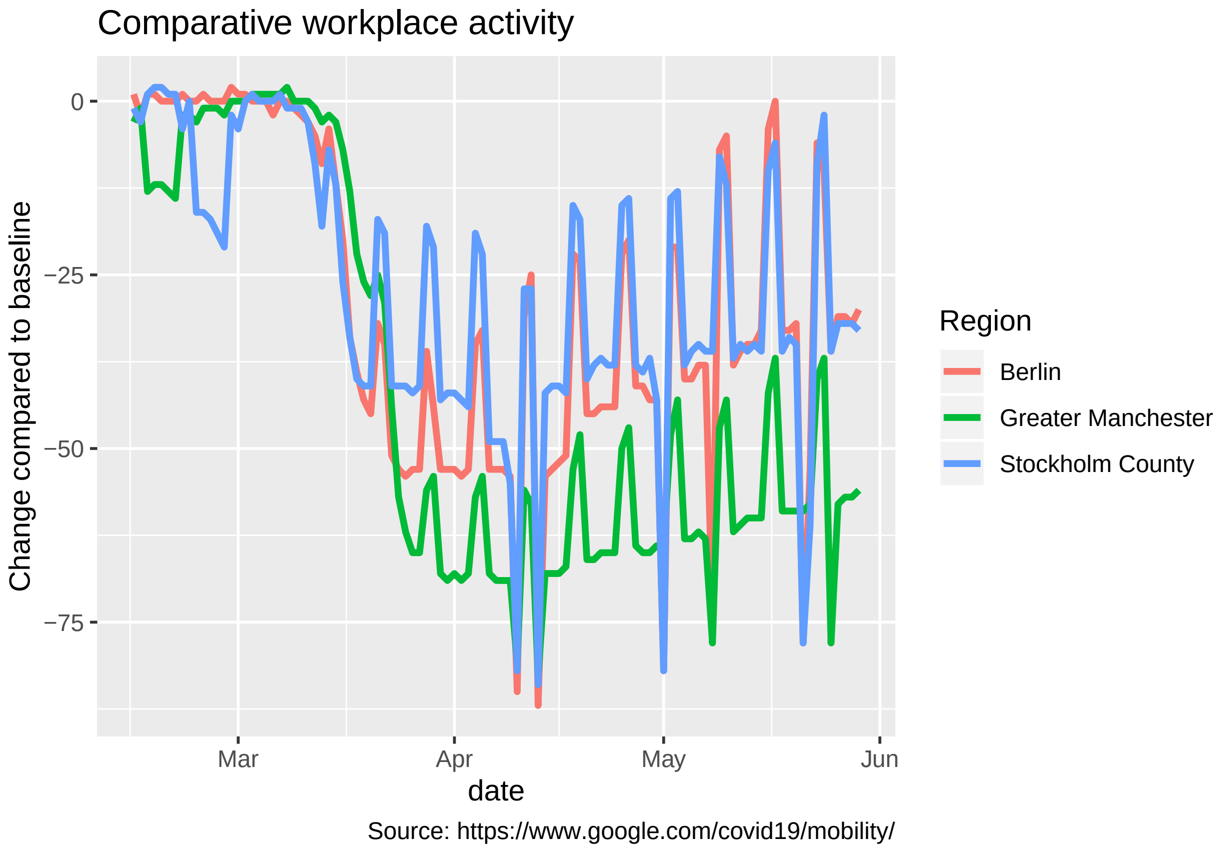
```r
temp <- mob_data_sel %>% filter(sub_region_1 == "Greater Manchester")
ggplot(temp,aes(x =date, y=workplaces_percent_change_from_baseline)) +
  geom_line(size = 1.2) +
  labs(title = "Comparative workplace activity",
       caption = "Source: https://www.google.com/covid19/mobility/") +
  ylab("Change compared to baseline")
```



Use `?labs` to figure out what the last few lines in the code did.

Let's add the same information for Berlin and Stockholm. Hence we are using `mob_data_sel`.

```r
ggplot(mob_data_sel,aes(x =date, y=workplaces_percent_change_from_baseline, color=sub_region_1)) +
  geom_line(size = 1.2) +
  labs(title = "Comparative workplace activity",
       caption = "Source: https://www.google.com/covid19/mobility/") +
  ylab("Change compared to baseline") +
  scale_color_discrete(name="Region")
```



Source: https://www.google.com/covid19/mobility/

Which part of the above code chunk created three lines with differnt colors?

You can clearly see the dip due to lockdowns, the gradual increase of the activity since and the weekly seasonality pattern.

Import policy and outcome data

Let's use another dataset which contains measures of how stringet a countrie's policies were to restrict the spread of the pandemic, but also contains some basic health indicators. Go to the https://www.bsg.ox.ac.uk/research/research-projects/coronavirus-government-response-tracker page of Blavatnik Schoool of Government (Uni of Oxford) and download the latest available data into your working directory.

```
policy_data <- XXXX("OxCGRT_latest.csv")
```

When done correctly you should have a new datafile with 42 variables.

Let's restrict ourselves to the three countries which correspond to the above cities (`country_sel`).

```
country_sel <- c("United Kingdom", "Sweden", "Germany")

policy_data_sel <- XXXX %>%
  XXXX(CountryName XXXX country_sel)
names(policy_data_sel)
```

```
 [1] "CountryName"
 [2] "CountryCode"
 [3] "Date"
 [4] "C1_School.closing"
 [5] "C1_Flag"
 [6] "C2_Workplace.closing"
 [7] "C2_Flag"
 [8] "C3_Cancel.public.events"
 [9] "C3_Flag"
[10] "C4_Restrictions.on.gatherings"
[11] "C4_Flag"
[12] "C5_Close.public.transport"
[13] "C5_Flag"
[14] "C6_Stay.at.home.requirements"
[15] "C6_Flag"
[16] "C7_Restrictions.on.internal.movement"
[17] "C7_Flag"
[18] "C8_International.travel.controls"
[19] "E1_Income.support"
[20] "E1_Flag"
[21] "E2_Debt.contract.relief"
[22] "E3_Fiscal.measures"
[23] "E4_International.support"
[24] "H1_Public.information.campaigns"
[25] "H1_Flag"
[26] "H2_Testing.policy"
[27] "H3_Contact.tracing"
[28] "H4_Emergency.investment.in.healthcare"
[29] "H5_Investment.in.vaccines"
[30] "M1_Wildcard"
[31] "ConfirmedCases"
[32] "ConfirmedDeaths"
[33] "StringencyIndex"
[34] "StringencyIndexForDisplay"
[35] "StringencyLegacyIndex"
[36] "StringencyLegacyIndexForDisplay"
[37] "GovernmentResponseIndex"
[38] "GovernmentResponseIndexForDisplay"
[39] "ContainmentHealthIndex"
[40] "ContainmentHealthIndexForDisplay"
[41] "EconomicSupportIndex"
[42] "EconomicSupportIndexForDisplay"
```

Dates are in the `Date` variable. They are formatted as `20200521` for 21 May 2020. Let's translate these into date format.

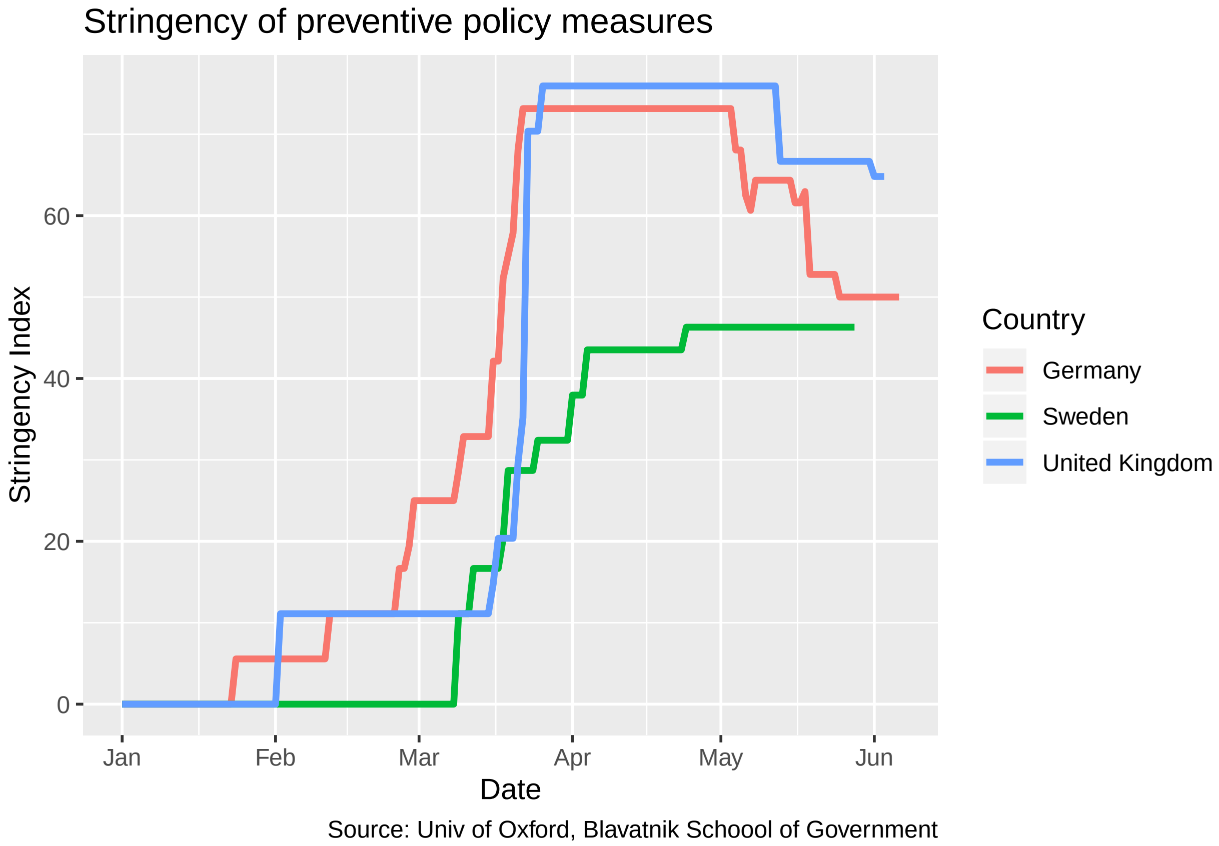
> You will need to let the `as.Date` function now how the date information is formatted.

```
policy_data_sel$Date <- as.Date(as.character(policy_data_sel$Date),"XXXX")
```

> Check that the date conversion worked and you can still see the date information in `Date`. If it didn't work you will see `NA`s.

The variable `StringencyIndex` contains an index describing the severity of the policy measures imposed.

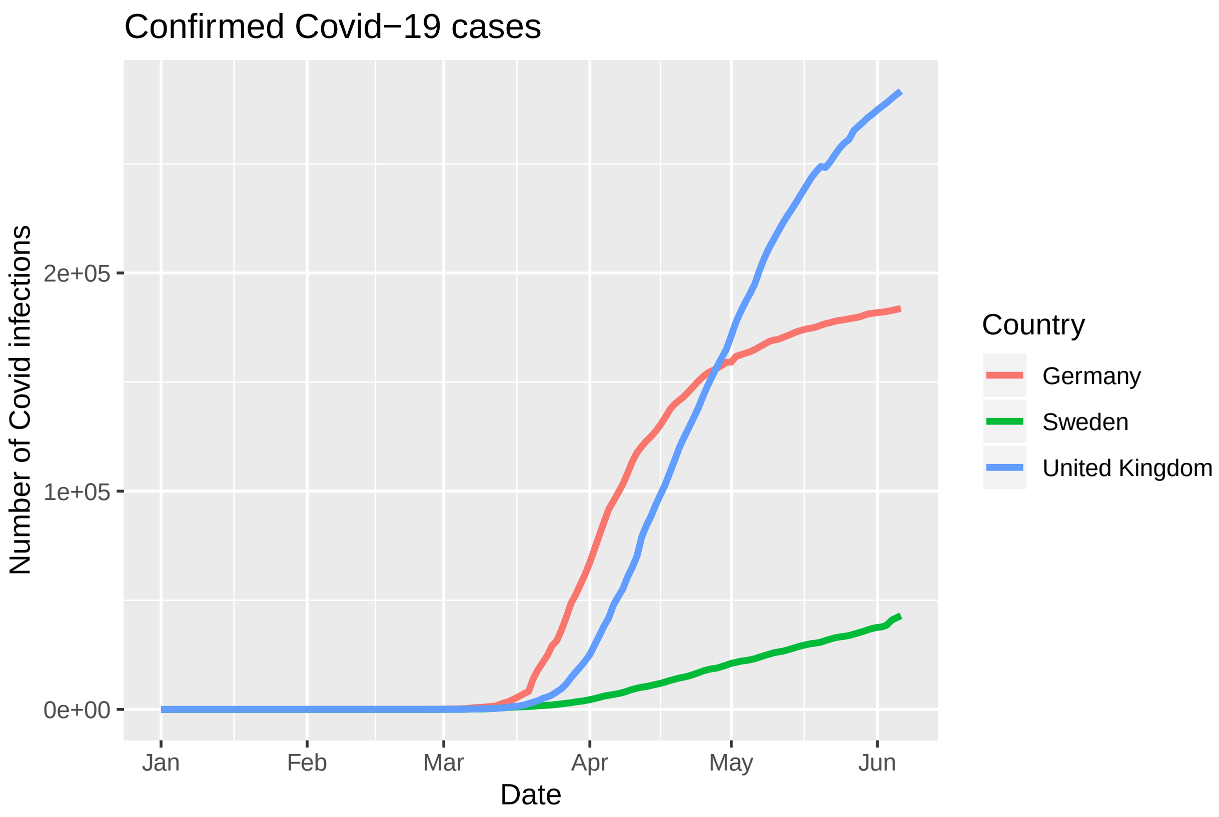
```r
ggplot(policy_data_sel,aes(x =Date, y=StringencyIndex, color=CountryName)) +
  geom_line(size = 1.2) +
  labs(title = "Stringency of preventive policy measures",
       caption = "Source: Univ of Oxford, Blavatnik Schoool of Government") +
  ylab("Stringency Index") +
  scale_color_discrete(name="Country")
```



The `+ scale_color_discrete(name="Country")` part of the above code changes the title to the legend which is automatically added to the plot as soon as you use the `color` aesthetic. Admittedly, the naming of this is not obvious but if you googled "r how to change legend title" you would quickly find examples which will tell you how to do this.

Let's also look at some infection numbers.

```r
ggplot(policy_data_sel,aes(x =XXXX, y=XXXX, color=XXXX)) +
  geom_line(size = 1.2) +
  labs(XXXX = "Confirmed Covid-19 cases",
       caption = "Source: Univ of Oxford, Blavatnik Schoool of Government") +
  XXXX("Number of Covid infections") +
  scale_color_discrete(name="Country")
```

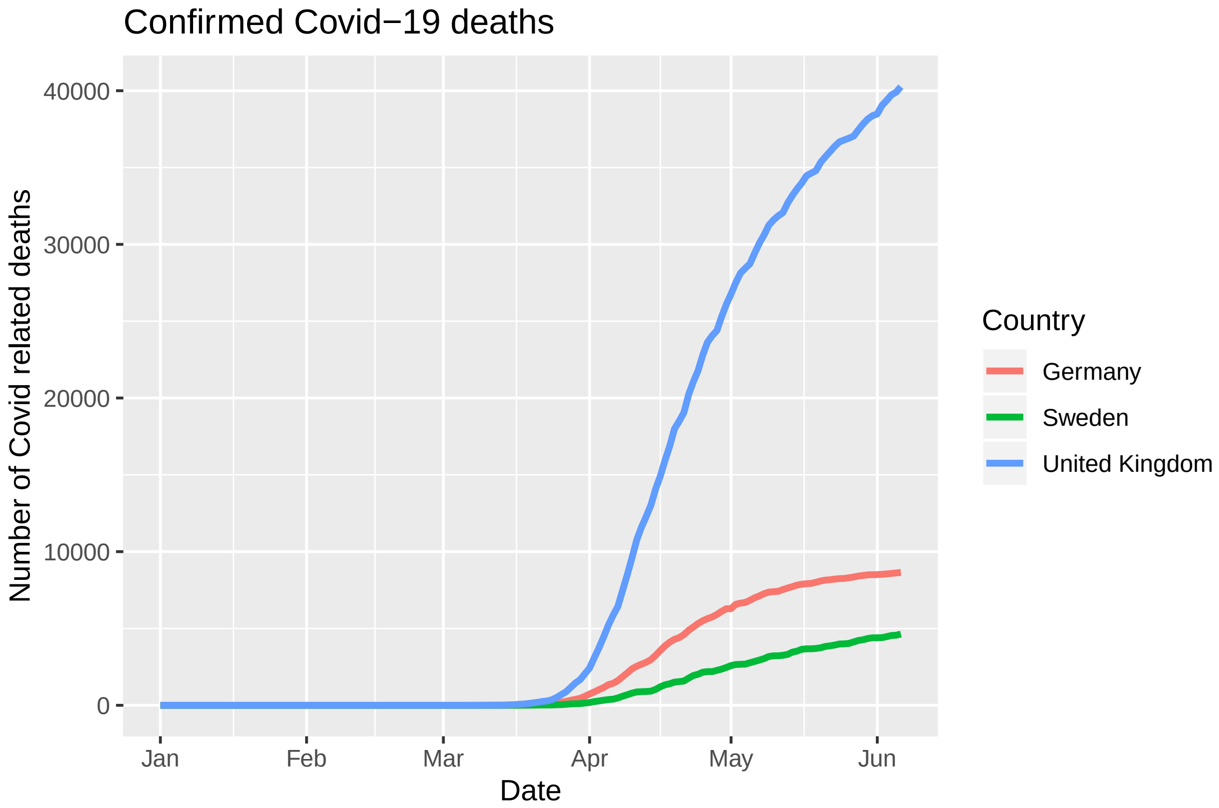


Confirmed Covid−19 cases

Source: Univ of Oxford, Blavatnik Schoool of Government

Or now the number of confirmed deaths.

```
ggplot(XXXX) +
  geom_XXXX(size = 1.2) +
  labs(title = "Confirmed Covid-19 deaths",
       caption = "Source: Univ of Oxford, Blavatnik Schoool of Government") +
  XXXX("Number of Covid related deaths") +
  XXXX(name="Country")
```



Source: Univ of Oxford, Blavatnik Schoool of Government

When looking at the number of deaths one would have to concluded that the UK has fared worse so far. This, however, does not take the size of the population into account. While the UK population is about 66 million, that of Sweden is about 10 million. If you adjust for this, then, in terms of deaths Sweden and the UK have about similar numbers.