Supplement 1

One nice aspect about using a programming language for analyzing data is that the script is at the same time documentation about what you have done. However, it is only a valuable documentation if it is readable for other humans and your future you. To keep a script readable there are some easy rules to follow (and in the future I will refuse to help if you do not follow these rules. I'm a biologist/data analyst, not a cryptologist ;-)).

- Packages are loaded at the beginning of a script, always! It is the very first thing you have to do. There is no valid reason to load packages somewhere half way. This makes it directly obvious which dependencies your script has. Other R scripts that belong to your project are sourced directly after these imports.
- A single line never exceeds 80 characters. If you have such a line, break it in multiple lines. As a help: You can add a vertical line in RStudio that indicates when you exceed 80 characters. Go to `Tools > Global Options... > Code > Display` and activate `Show margin` with margin column of `80`. Function calls can easily be broken over several lines.

```
function(
    argument1 = parameter,
    argument2 = other_parameter,
    argument3 = NULL
)
```

- Add comments to your code. R ignores everything in a line behind a `#`. Add comments in front of a block of code to explain what the code in that block will do. Short and precise but with enough details to be understandable. It is much easier to understand a code and how it works if it is known what it should do. You can also use comments to comment-out some code if you are trying around with different ways of achieving your task. This is much better than having different versions of the same file.
- Use blank lines to visually structure your code. Lines do not cost anything but hugely improves readability.
- Use meaningful variable names. `a` is a very short name for a variable but doesn't give the reader any idea what kind of data the variable holds. It is not easy to find good names but it's worth the effort.
- Put spaces around assignment operators and equal signs in argument specification. Don't break this rule to follow the second point!

```
# Bad.
mean(x=1:20,na.rm=TRUE,trim=.1)

# Good.
mean(1:20, na.rm = TRUE, trim = .1)
```

Every programming language has style guides which you should follow. One popular one for R can be found at `https://style.tidyverse.org/`. You do not need the packages mentioned on the first page. Keep it simple ;-)

Exercise 1

For the following exercises you have to modify the `analysis.R` file. Make sure to consider the above rules. Also make sure to keep everything in the correct order. When starting with a fresh R session everything you should have to do is to press `Source` on the upper right. This runs the complete script from top to bottom.
We will only analyze the data for whole countries, not for single regions. So we have to sum the data of all regions for one country. Also we have to modify the date of the last update as this date might differ between different regions of the same country. We will use the latest date for any of the regions of one country as a representative date for the whole country.

(a) First, we have to sum up the cases of all regions of one country. Use the function `aggregate` to sum up the number of confirmed cases, deaths, and recovered (look in the examples on the help page for `aggregate` to see how you can use several columns). Group the data by the country only. Store the result in a variable called `sum_country`.

(b) Second, we have to determine the date of the last update for one country. On the last sheet we have converted the respective column to dates, so we can do math with them and apply, for instance, functions like `max` to them. Use `aggregate` to determine the latest date for each country. Again, group by country only. Store the result in a variable called `update_country`.

(c) Merge the two new data frames using `merge`. Since the key column has the same name in both data frames you do not have to specify it.

(d) Save the result in a CSV file as before.

(e) Find out how many and which countries had more than 1000 confirmed cases. Hint: You can use `nrow` to determine the number of rows of a data frame. Why would `length` not work? What does `length` return?

*Exercise 2

This exercise is for the upcoming lecture to prepare you for the content or to sensibilize you for a certain aspect.
Think about some nice graphics that might be informative. Consider that for now we only read in data from two dates and we will not read in more by hand. So we can not show any temporal trends.