R Programming

In this project you will write a Shiny app which visualises air pollution data collected across multiple sites in the Czech Republic from 2013 to 2019.

The data is available on Moodle as a collection of CSV files, as downloaded from European Environmental Agency (see https://www.eea.europa.eu/themes/air). The file `Stations.csv` contains information about each station. There is one CSV file for each station and each pollutant (PM2.5, PM10, SO_2 and NO_2) measured at the station (not every pollutant is measured at every station). The filenames are of the form `<EoICode>_<PollutantCode>.csv`. Each row of these files contains hourly measurements of that pollutant at that station. For some stations there only is data for a subset of the time points.

The current air quality standards in the EU are:

Fine particulates (PM2.5) Yearly average of at most $25\mu g/m^3$.

Particulates (PM10) Daily average exceeding $50\mu g/m^3$ observed on at most 35 days a year, and yearly average of at most $40\mu g/m^3$.

Sulphur dioxide (SO_2) Hourly concentration exceeding $350\mu g/m^3$ for at most 24 hours per year, and average daily concentration exceeding $125\mu g/m^3$ on at most 3 days per year.

Nitrogen dioxide (NO_2) Hourly concentration exceeding $200\mu g/m^3$ for at most 18 hours per year, and average yearly concentration of at most $40\mu g/m^3$.

Your task consists of developing a Shiny app visualising the data with the following functionality.

- The core functionality of the app is to produce time series plots of relevant summary statistics against time.
- The user should be able to select for which pollutant and for which stations (at most 3) the quantities should be shown.
- The user should either be informed if the pollutant is not available for this measuring station or, after selecting the pollutant, the app should only show stations at which the pollutant was measured.
- The user should be able to decide what aggregation to plot. This should include the following aggregations:
  - Raw hourly data (no aggregation);
  - Daily averages;
  - Daily maxima;
  - Number of hours per day for which a given threshold is exceeded; and
  - Number of hours per year for which a given threshold is exceeded;

  For aggregations involving a threshold, the user should be able to choose this threshold.

- The user should be able to choose how time is to be handled and what the x-Axis should represent. Choices should include
  - Calendar time;
  - Date within the year (going from Jan 1st to Dec 31st);
  - Day or hour within the week (going from 0 to 7 (days) or 0 to 168 (hours)); and
  - Hour in the day (going from 0 to 24 (hours)).

  For calendar time the plot should be a line plot, but for the other three choices the plot should be a scatter plot.
  If data is to be aggregated over years then the latter three plots are not meaningful. The app should handle this suitably.
- If the choice of pollutant and aggregation matches a criterion from the EU air quality standard, then the plot should show a horizontal line at the threshold for that criterion. For example, if the user wants to plot the raw hourly data for SO_2, then a horizontal line should be drawn at $y = 350$.
- Make sure that suitable axis labels and legends are used.
- The app should also show a second plot with the location of the measuring station(s). You can either use the package `maps` or `ggmaps` for this (or any other package you may find suitable). The code below shows the location of all measuring stations together with a map of the Czech Republic (assuming the data from `Stations.csv` is stored in a data frame `stations`).

```
maps::map("world","Czech Republic")
points(stations$Longitude, stations$Latitude, pch=16, col="red")
```

- Below the plot there should the a table showing the data used in the plot in wide format (i.e. columns should correspond to measuring stations).
- The user should be able to download the table as a CSV file and the plots and table together as a Word document (generated using `rmarkdown`).

You can either use classical R plots or `ggplot2` (or any other plotting library of your choosing).

Coding hints

- You might not have enough memory to read all the data into R in one go. One way to deal with this is to read in the data only when it is needed. Once the user has chosen the pollutant and the measuring stations, at most four CSV files need to be read into memory. To minimise the storage space taken up by the data on disk, you can convert each CSV file to an RData file.
- Once the package `lubridate` has been loaded a date can be converted to a decimal date using

```
library(lubridate)
decimal_date(ymd(paste(2019, 12, 19, sep="-")))
```

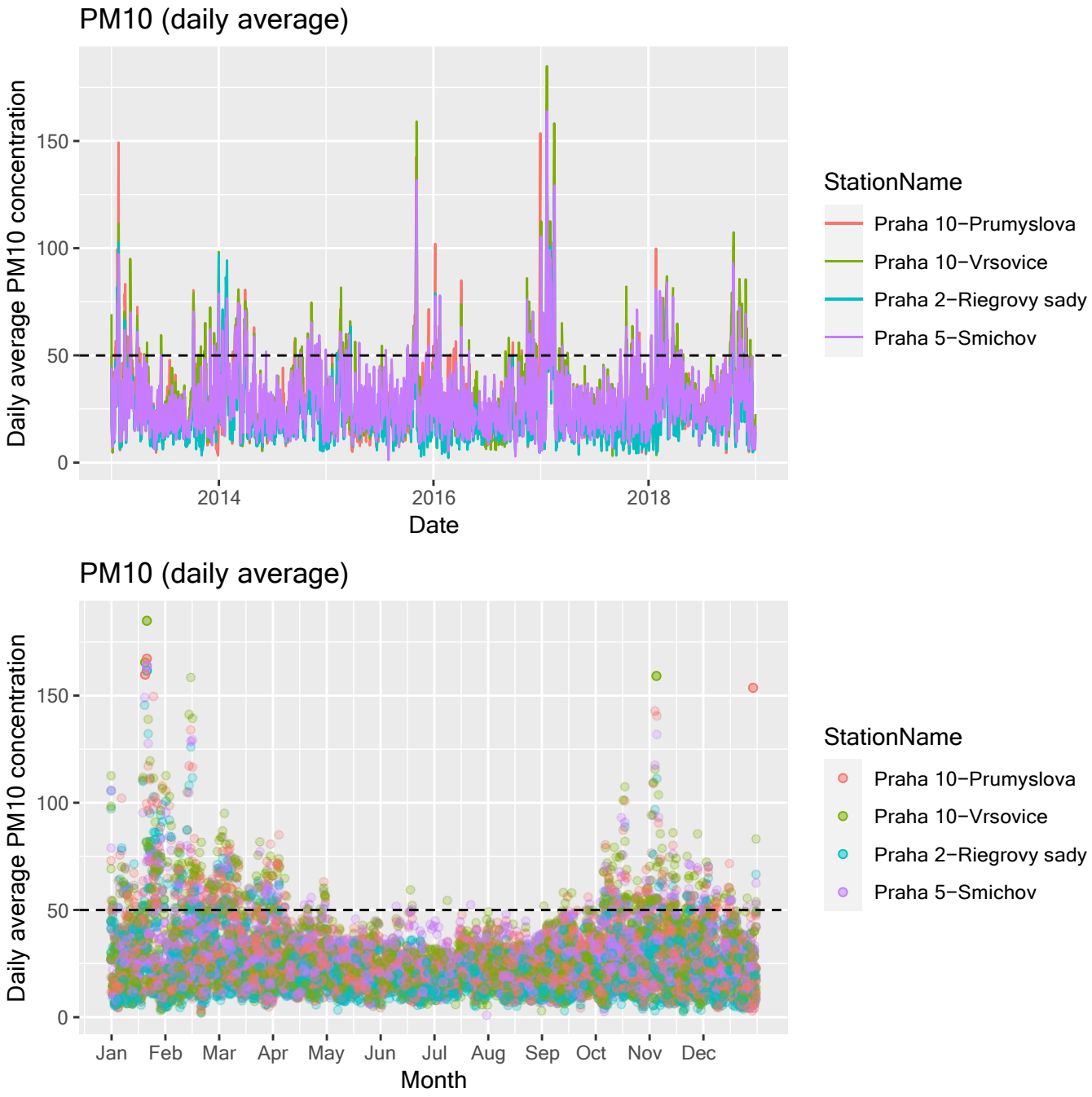
- The code below renders the file 'report.Rmd' (in the same directory as your Shiny app) as a Word document and makes it available to download.

```
output$downloadReportButton <- downloadHandler(
    filename = "report.docx",
    content = function(file) {
        render("report.Rmd", output_format="word_document",
               output_file=file, params=list(a=input$a))
#                                ^^^^^^^^^^^^^^^^^^^^^^
#                                input$a now available as params$a
#                                in rmarkdown
#                                (remember to define parameters in
#                                 header of Rmd file)
    }
)
```

  Instead of passing on objects to `rmarkdown` using `params` you can also store them in a temporary RData file in your Shiny app and then load this file in the `rmarkdown` document.

  If there are functions you want to use both in the Shiny app and in your `rmarkdown` report, consider placing them in an separate R file. This R file can then be included using `source`. (Inside the Shiny app you have to use `source` with the additional argument `local=TRUE`).

Examples of plots



What will be assessed?

I will assess the functionality of the app and the code implementing it. The following scheme will be used to assess your submission.

A bonus 2 marks are available for advanced formatting and aesthetics. It is important to note that your Shiny app first and foremost should be able to perform the key functionality and this should be your priority.

Please note, the maximum number of marks that can be achieved for this assignment is 20 marks.

A Moodle forum will be set up for you to raise any questions on the project. Here, you may ask general questions on Shiny implementation and high level questions on data aggregation/plotting. Please **do not** share any code directly related to your app or any code which answers components of the assessment.

Criterion / “Unit test”	Marks if correct
Does the user interface have the required functionality?	6 marks
Aggregations calculated correctly?	4 marks
Main plot shown as described?	3 marks
Plot of map shown as described?	1 mark
Data shown in table and downloadable?	2 marks
Report generation	2 marks
Code clear and well organised?	2 marks
BONUS - Use of advanced aesthetic features or functionality	2 marks