Exercises

Lene Østvand

September 8, 2022

Fiddling with data

In the following exercises we will read the excel file containing observations and learn how to do different operations with them. First we need to import some libraries that allow us to use some useful functions.

1. Import the following libraries in R with the library function:
   - readxl
   - dplyr
   - tidyr
   - tidyverse
   - ggplot2
2. Read the excel file with observations with the function

   ```
   read_excel('<path/to/file/filename>')
   ```

   and store it in a variable called `obs`. View `obs` by clicking it in the Environment tab, or type `View(obs)` and press enter in the Console.
3. This may look a little strange. The headers we would like to have are in row 2. Set the names of obs to be the names in row 2 with

   ```
   colnames(obs) <- obs[2, ]
   ```

4. Delete the two first rows with

   ```
   obs <- obs[-(1:2), ]
   ```

5. Check the type of the column with the temperatures with the function `class`. We would like to be able to do calculations with the temperature. Do you think it is possible now? Feel free to try a mathematical operation, e.g.

```
sum(obs$`Dry_Bulb_Temperature(deg. C)`)
```

Hint: To extract one column from a data frame, use the $ sign, i.e.,

```
obs$<column name>
```

When the column name has special characters like spaces and brackets in it you need to use ` on each side of the column name.

6. Change the column with temperature to be numeric with the function as.numeric. Check the class again.

7. There are easier ways to deal with excel files that have rows that contain data we do not want to read. Type `?read_excel` in the Console and press Enter and read about the optional argument `skip`. Reread the file with the observations with the argument skip.

8. Check the class of the column with temperatures again.

9. The column names `Dry_Bulb_Temperature(deg. C)` and `3hourly rain(mm)` are a bit long and tedious to work with. Change them to `T2m` and `AccPcp3h` respectively.

10. Remove the column `T2m` with the following code

```
obs <- obs[colnames(obs) != "T2m"]
```

11. Oops, we actually need that column. Luckily, all the changes we have done are local in R, and the original file is unchanged. You can read the file again with the code you have written. If you were clever, you put the code in a script. If not, now is a good time to do so. Feel free to comment out any lines that are not needed with the # symbol in front of the line (short key `Ctrl+Shift+c`), and run the script by clicking Source.

12. We would like to have a column containing all the information about time and date. Create this with the following code

```
obs$datetime <- as.POSIXct(
  paste(obs$Year, obs$Month, obs$Day, obs$`Hour(UTC)`),
  format = "%Y %m %d %H",
  tz = "UTC")
```

13. You have now created a column that is a Date-Time class. If some dates are not actual dates, like February 30, the entry in the column will be `NA`. Check for `NA`s with the line

```
print(obs[is.na(obs$datetime), ])
```

Do you see why the datetime column has `NA`s? NB! This is something you might want to correct in your excel file.

14. We don't want to keep the rows with nonsensical dates. Remove them with

```
obs <- obs[!is.na(obs$datetime), ]
```

15. The filter function from the `dplyr` package can be quite useful. Read about it by typing `?dplyr::filter` and pressing Enter in the Console. At the bottom there are many useful examples. Try to use the filter function to keep only data from Dhaka, and store it in a new variable called `obs2`.

16. Plot the temperature with

```
ggplot(obs2, aes(x = datetime, y = T2m)) + geom_line()
```

17. Create a new variable called `obs3` filtered to contain data from Dhaka between the dates 2020-01-01 and 2020-01-07. Plot the temperature again. Hint: You need to tell R that the string 2020-01-01 is a date with the function `as.Date()` when comparing with the column `datetime` with inequality symbols.

18. Let's say that we want to find the monthly mean temperature for the full period for Dhaka.

    (a) Create a new variable named `obs4` that includes all the columns from `obs2` except `datetime` and `AccPcp3h`.

    (b) Read about the function `pivot_wider`, and learn what the arguments `names_from`, `values_from` and `names_prefix` mean.

    (c) Use `pivot_wider` on `obs4` to make columns with names from the months with prefix M and values from temperature. View the new object.

    (d) Use the `grep` function to find the column names that contain the letter M.

    (e) Use the function `colMeans` to find the mean for these columns. You should now know how to find information about functions and their arguments. Do `colMeans` have any useful arguments for dealing with missing values?

19. We now want to find the monthly mean temperature for all stations.

    (a) Do step a and c from exercise 18 and store the result in a variable called `obs5`.

(b) Find the monthly means by running the code

```
obs5 %>%
  group_by(Station) %>%
  summarise(
    across(starts_with("M"),
           ~ mean(.x, na.rm = TRUE)
    )
  )
```

20. A different way to find the monthly means is to use the `group_by` function on both Station and Month.

(a) Run the code

```
 monthly.means <- obs %>%
   group_by(Station, Month) %>%
   summarise(T2m = mean(T2m, na.rm = T))
```

(b) View the object `monthly.means`.

(c) Plot the temperature means with respect to month. Hint: you can plot the temperature means with different color for each station if you include `color = Station` in the `aes()`.

(d) The wide format can be useful if you want to store the result in a table. Transforrm `monthly.means` to wide format with `pivot_wider`.