

III. Graphics with ggplot2 (exercises)

Data Science Lab, University of Copenhagen

August 2025

Before you proceed with the exercises in this document, make sure to run the command `library(tidyverse)` in order to load the core **tidyverse** packages (including **ggplot2**).

The data set used in these exercises, **climate.xlsx**¹, was compiled from data downloaded in 2017 from the website of the UK's national weather service, the *Met Office*.

The spreadsheet contains data from five UK weather stations in 2016. The following variables are included in the data set:

Variable name	Explanation
<i>station</i>	Location of weather station
<i>year</i>	Year
<i>month</i>	Month
<i>af</i>	Days of air frost
<i>rain</i>	Rainfall in mm
<i>sun</i>	Sunshine duration in hours
<i>device</i>	Brand of sunshine recorder / sensor

The data set is the same as the one used for the exercises **Working with data in R**. If you have already imported the data, there is no need to import it again (unless you have made changes to the data assigned to **climate** since the original data set was imported).

Scatter plot I

1. Make a scatter plot of **rain** against **sun**.
2. Colour the points in the scatter plot according to weather station.
3. Add the command `facet_wrap(~station)` to the code for the scatter plot, and update the plot. (Remember to put a `+` in between the old and the new code.) What happens?
4. Is it necessary to have a legend in the faceted plot? You can remove the legend by adding `theme(legend.position = "none")` to your code.
5. The gray background is fine for graphics on the computer screen, but might not be desirable for plots on paper. You can change to a black-white *theme* (the overall layout) by adding `theme_bw()` to your code. Please note that `theme_bw()` overwrites all theme-settings. Thus, if you also want to remove the legend, then `theme(legend.position = "none")` should be added after `theme_bw()`. There are other build-in themes, e.g. `theme_classic()`.

¹Contains public sector information licensed under the Open Government Licence v3.0.

Graphic files

1. Make sure that the working directory is pointing to the place, where you want to save the graphic files. You may check this by executing the code

```
getwd()
```

If necessary you may change the working directory via the function `?setwd`.

2. Use `ggsave(file="weather.jpeg")` to remake the last gg-plot as a jpeg-file. Locate this file on your computer and open it.
3. Use `ggsave(file="weather.png",width=10,height=8,units="cm")` to remake the last gg-plot as a png-file. What do the three other options do? Look at the help page `?ggsave` to get an overview of the possible options.

Line plot (also known as a spaghetti plot)

1. Make a line plot of the rainfall observations over time, in which observations from the same station are connected. (Put month on the x-axis.) Colour the lines according to weather station as well.
2. The **month** variable was read into R as a numerical variable. Run the following command, which overwrites the numerical **month** variable in the climate data set with a factor (categorical variable) of the same name.

```
climate <- mutate(climate, month = factor(month))
```

Make the line plot again. What has changed? Note: When the x-axis is categorical you need to specify the aesthetic `group=station` in order to connect the points in the line.

3. Use `theme(legend.position = "top")` to move the colour legend to the top of the plot.
4. Use `geom_point()` to add the monthly rainfall data points to the line plot.
5. Now, insert `geom_hline(yintercept = mean(climate$rain), linetype = "dashed")` at the end of your code for the line plot, and update the plot again. What have you just added to the plot?
6. Finally, try adding

```
labs(x = "X", y = "Y", colour = "COL")
```

to the code and update the plot. What changed? Replace X, Y, and COL with some more suitable (informative) text.

Box plot

1. Make a box plot of the sunshine observations by weather station.
2. Add `geom_boxplot(fill = "lightgreen")` to your code to colour the boxes light green.
3. Try colouring the boxes according to weather station instead, by using `geom_boxplot(aes(fill = station))`. Remove the superfluous plot legend.

Histogram

1. Run the code

```
ggplot(climate, aes(x = rain)) +  
  geom_histogram()
```

What does this plot show?

2. R suggests that you choose a different number of bins/binwidth for the histogram. Use `geom_histogram(binwidth = ...)` to experiment with different values of binwidth in place of ..., e.g. 5, 15, 25, and 35. See how the histogram changes.
3. Add some colour to the histogram by inserting `colour = "white"` and `fill = "orange"` as additional arguments to `geom_histogram`. Try making the border of the boxes red instead.

Bar chart I - a plot often used in papers!

1. Run the following command

```
climate %>%
  group_by(month) %>%
  summarize(sun_avg = mean(sun), sun_sd = sd(sun))
```

Make sure you understand the output, and save it as `summary_stats`.

2. Use the `summary_stats` data to make a bar chart with month on the x-axis, where the heights of the bars represent the average number of sunshine hours.
3. Colour the bars light blue.
4. Add the following layer to the plot:

```
geom_errorbar(aes(ymin = sun_avg - sun_sd, ymax = sun_avg + sun_sd), width = 0.2)
```

What has been added to the plot?

5. Experiment with different widths for the ends of the whiskers. For example, try widths of 0.1, 0.5, and 1.

Scatter plot II

1. Make a scatter plot with month on the x-axis, and the average number of sunshine hours on the y-axis.
2. Add error bars to the plot, which represent the average number of sunshine hours plus/minus the standard deviation of the observations.
3. Could you have made a plot with horizontal error bars instead? How?

Bar chart II

1. Make a bar chart which visualizes the total number of sunshine hours recorded at the five weather stations during each month of the year 2016.
2. Colour the bars according to weather station. Hint: The colour inside the bars is controlled by the `fill` aesthetic (and not the `colour` aesthetic).
3. Make the axis labels and legend title of the plot more informative by customizing them like you did for the line plot above.

Bar chart III

1. Make a bar chart showing the annual rainfall recorded at each weather station.
2. Run the following code:

```
climate %>%
  group_by(station) %>%
  summarize(rain = sum(rain)) %>%
  arrange(rain)
```

What is the connection between the output and the bar chart? Save the output as `annual_rain`.

3. Order the bars in the bar chart according to total annual rainfall.
4. Now, try adding

```
geom_text(mapping = aes(x = station, y = rain, label = rain), data = annual_rain)
```

to the code for the bar chart, and update the plot. Understand what has been added to the plot (what the labels represent, and how they have been positioned).

5. Adjust the label positions so that the labels are positioned immediately above the bars.
6. Replace `geom_text` with `geom_label`, and see how the labels change.

Scatter plot III

1. Run the following code, which makes a simple example data set.

```
example_data <- tibble(X = 1:5, Y = 2*X, Z = letters[1:5])
example_data
```

Make a simple scatter plot of Y against X using the code below.

```
ggplot() +
  geom_point(mapping = aes(x = X, y = Y), data = example_data)
```

Note that the data set and aesthetics are specified “locally” for use with the `geom_point` function.

2. Run the following code and observe what changes in the plot. In the code, `geom_point` has been replaced by `geom_text`, and an additional aesthetic `label = Z` has been inserted.

```
ggplot() +
  geom_text(mapping = aes(x = X, y = Y, label = Z), data = example_data)
```

3. What happens when `nudge_y = 0.3` is added as an argument to `geom_text` (as in the code below)?

```
ggplot() +
  geom_text(mapping = aes(x = X, y = Y, label = Z), data = example_data, nudge_y=0.3)
```

End of exercise