III. Graphics with ggplot2 (exercises)

Data Science Lab, University of Copenhagen

01 September, 2021

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

The data set is the same as the one used for the exercises **II. 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).

Need a little help? Consult the ggplot2 cheat sheet here: https://rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf

Scatter plot I

1. Make a scatter plot of **rain** against **sun**. HINT: Try `geom_point`.
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.

Graphic files

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

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

```
getwd()
```

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

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

Scatter plot II

8. 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 to a vector named `summary_stats`.

9. Make a scatter plot with month on the x-axis, and the average number of sunshine hours on the y-axis.
10. Add error bars to the plot, which represent the average number of sunshine hours plus/minus the standard deviation of the observations. Try:

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

11. Could you have made a plot with horizontal error bars instead? How?

Scatter plot III

12. 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
```

13. 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.

14. 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)
```

15. 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)
```

Line plot (also known as a spaghetti plot)

16. 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.

17. 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?

18. Use `theme(legend.position = "top")` to move the colour legend to the top of the plot.

19. Use `geom_point()` to add the monthly rainfall data points to the line plot.

20. 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?

21. 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 I

22. Make a box plot of the sunshine observations by weather station.

23. Add `geom_boxplot(fill = "lightgreen")` to your code to colour the boxes light green.

24. Try colouring the boxes according to weather station instead, by using `geom_boxplot(aes(fill = station))`. Remove the superfluous plot legend.

Box plot II - Aesthetics

There are many ways in which you can manipulate the look of your plot. For this we will use the boxplot you made in the exercise above.

25. Move the legend with `theme(legend.position="top")` and add a different legend title with `labs(fill = "Custom Title")`.

26. Change the theme of the ggplot grid. You could try `theme_minimal()`, `theme_dark()`, or `theme_void()`.

27. Instead of the colours automatically chosen when you use `fill = station`, try so pick your own colors. Use the `scale_fill_manual()`. You will need five colors, one for each station. What happens if you choose too few colours?

28. Change the boxplot to a **violin plot**. Add the sunshine observations as scatter points to the plot. Include the a boxplot inside the violin plot, use `geom_boxplot(width=.1)` to do this.

Histogram

29. Run the code

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

    What does this plot show?

30. R suggests that you choose a different number of bins/bin width for the histogram. Use `geom_histogram(binwidth = ...)` to experiment with different values of bin width in place of `...`, e.g. 5, 15, 25, and 35. See how the histogram changes.

31. 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

32. Make a bar chart which visualizes the sunshine hours recorded at the five weather stations during each month of the year 2016. *HINT:* use `geom_col()`.
33. Colour the bars according to weather station.
34. 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 II

35. Use the `summary_stats` data (from Scatter plot II section above) to make a bar chart with month on the x-axis, where the heights of the bars represent the average number of sunshine hours.
36. Colour the bars light yellow.
37. Add the following layer to the plot:

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

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

Bar chart III

39. Make a new bar chart showing the annual rainfall recorded at each weather station.
40. 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`.

41. Use the `annual_rain` data to make a bar chart of the annual rainfall recorded at each weather station (just like in question 1.) but this time add:

```
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.

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