III. Graphics with ggplot2 (exercises)

Data Science Lab, University of Copenhagen

20 May, 2022

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 Tidyverse exercise. 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. Save the plot in an object, for example `p`.
3. Add the command `facet_wrap(~station)` to the saved plot object from above, and update the plot. New instructions are added with `+`, see the ggplot lecture. 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.

Scatter plot II: labels

5. Now, let's label each point with the month it corresponds to. This is done by using the geom `geom_text`. Have a look at the help:

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

```
?geom_text
```

You need to tell the aesthetics mapping which column contains your labels, just like when you define which column is on the x-axis or which column to color by. The relevant aes keyword is `label`. You can add it to the geom_text or in the `ggplot` call on top where you define the rest of the aes keywords:

```
#pseudo code:
#... + geom_text(aes(label = ???))
```

6. You might find that your labels are on top of your points and not very readable. Let's put them a bit higher by adding `nudge_y = 15` to the geom_text.
7. Change `geom_text` to `geom_label`.

Graphic files

7. 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

```
getwd()
```

If necessary change the working directory via the function `setwd`, see presentation 1.

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

Scatter plot III: error bars

10. Calculate the average and standard deviation for sunshine in each month and save it to a table called `summary_stats`. You will need group_by and summarize.

If you can't remember how, the command is written here:

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

11. Make a scatter plot of the summary_stats with month on the x-axis, and the average number of sunshine hours on the y-axis.
12. Add error bars to the plot, which represent the average number of sunshine hours plus/minus the standard deviation of the observations. The relevant geom is called `geom_errorbar`.

Hint:

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

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

Line plot (also known as a spaghetti plot)

14. Make a line plot of the rainfall observations over time, such that observations from the same station are connected in one line. If you have trouble with this, have a look at the ggplot lecture. Put month on the x-axis. Colour the lines according to weather station as well.

15. 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?

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

Layering

We can add several geoms to the same plot to show several things at once.

17. Use `geom_point()` to add the monthly rainfall data points to the line plot above.
18. 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?
19. 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. Colour the boxes according to weather station.

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.

24. Move the legend with `theme(legend.position="top")` and add a different legend title with `labs(fill = "Custom Title")`.
25. Change the theme of the ggplot grid. Suggestions: `theme_minimal()`, `theme_bw()`, `theme_dark()`, `theme_void()`.
26. Instead of the colours automatically chosen when you use `fill = station`, 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?
27. Change the boxplot to a **violin plot**. Add the sunshine observations as scatter points to the plot. Include a boxplot inside the violin plot, use `geom_boxplot(width=.1)` to do this.

Histogram

28. Run the code

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

What does this plot show?

29. 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.

30. Add some colour to the histogram by inserting `colour = "grey50"` and `fill = "white"` 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 per month. *HINT:* use `geom_col()`. You might get a better result by treating the month as a factor (instead of numeric). You can add a factor cast right inside the aes, i.e.

```
#pseudocode
#ggplot(climate, aes(y=as.factor(month), x = ...))
```

33. Colour, i.e. divide the bars according to weather station.

34. For better comparison, place the bars for each station next to each other instead of stacking them.

35. 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: Sorting bars

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

40. Sort the stations in accordance to rainfall, either ascending or descending. This was shown in the ggplot lecture. Hint: You will need to first make a new dataframe with summed up rain data per station and sort it by that sum. Then re-arrange the 'station' column by making it a factor whose levels correspond to the order in the new dataframe.

41. Add labels to each bar that state the sum of the rainfall. You can do this by using the summarized dataframe created above and passing it to `geom_label` as data, together with a suitable aes mapping. I.e. if you named it `annual_rain` you can write:

```
#+ geom_label(mapping = aes(x = station, y = rain, label = rain), data = annual_rain)
```

42. Adjust the label positions so that the labels are positioned immediately above the bars.