Introduction

Methods of Scientific Working (for Crop Sciences) (3502-440)

21 Jan 2025

Table of contents

The purpose of this computer lab is to demonstrate two aspects of data visualization:

1. The advantage of visualization over looking at the data directly
2. A comparison of the two methods for making graphs:

- base graph, which is the original plotting system of R
- ggplot2, which is based on the concept of a 'Grammar of graphics'

To demonstrate these concepts, we use four data sets that consist of eleven data points each of x and y values.

Furthermore, your task is to find out about some of the functions to complete the exercises for yourself.

The dataset

The goal of this exercise is to format the dataset as a table and then to find out in what respect the datasets differ from each other.

A very useful tool to format dataframes as nice table is the `kable` function which is part of the `knitr` package. If the `knitr` library is not installed, you need to do it with `install.packages("knitr")`.

````
```{r}
library(knitr)
```
````

The data set we are going to analyse is already installed in the regular R installation and it is called `anscombe`.

````
```{r}
anscombe
```
````

Let us reorder the column names:

```
```{r}
df <- anscombe[c("x1", "y1", "x2", "y2", "x3", "y3", "x4", "y4")]
df
```
```

1 Exercise:

- Make the table a bit more prettier with the `kable()` function.
- Carefully look at the data: Do you see any differences between the data? Can you tell from the numbers, how the datasets differ from each other?

Properties of the data

The next step is to analyse the properties of the datasets. We focus on calculating the mean and the standard deviation.

2 Exercise

- Use the functions `mean()` and `sd()` to calculate the means and the standard deviations for all `x` and `y` variables in the four datasets. For the `x` values of dataset 1, the command is `mean(df$x1)`, etc.
- What do the datasets have in common and how do they differ?

The base plot system of R.

We next use the plotting system of base R. The goal is to construct a scatter plot in order to visualize differences between the plots and then to run a linear model on the data to test whether `x` and `y` are correlated.

For all functions mentioned below, please use the help function, *e.g.*, `?plot()`

3 Exercise

1. Create a separate plot for each `x` and `y` variables of each data set using the `plot()` function of base R.
2. Add a header to each dataset with the `main` parameter of the `plot()` function.
3. Combine the plots in a single panel using the `par(mfrow = c(2,2))` function.
4. Describe the difference between the datasets in words.
5. Also look at the other parameters of the `plot()` function. Can you *e.g.* figure out how to change the axis labels, the symbols used for plotting the points, use colours or change the sizes?

Statistical analysis of the data sets.

To find out whether the different datasets follow the same relationship between `x` and `y`, run a linear model with the `lm()` function.

A linear model has the form

$$y = a + bx$$

and the parameters a (= Intercept) and b are estimated as well as a p-value which shows whether there is a significant relationship between both variables.

The example calculation for data set 1 is shown in the following code chunk.

```{r}
fit1 <- lm(y1 ~ x1, data = df)
```

4 Exercises

- Calculate the linear model for each data set. Given the four plots: Do you expect that slope, intercept and significance are the same or different between the datasets?
- Compare the summary of the linear model of each data set and compare the outcome.

Plotting the slope of the line

In order to visualize the outcome of the linear model, you can add the line which indicates the slope parameter, b, to the plot. To do this, just run the function `abline(fit1)` after the function for the plot.

```{r}
plot(df$x1, df$y1)
abline(fit1, col = "red")
```

5 Exercise

- Plot the slope of the line for the other three datasets as well.

Plot the data using ggplot2 and the grammar of graphics

The graphics library `ggplot2` implements the **grammar of graphics**, which is defined as a 'coherent system for describing and building graphs. For a concise and very accessible introduction into the grammar of graphics, please consult http://vita.had.co.nz/papers/layered-grammar.pdf.

The key difference is that plots are build by subsequently adding functions to the plot-generating function, which as a consequence add additional properties to the plots. The commands that construct the plot are easy to understand and are also flexible to use.

To use ggplot2 and dependent packages, we need the meta-package `tidyverse`. If it is not installed on your system yet, install it with `install.packages("tidyverse")`.

```{r}
library(tidyverse)
```

Then, create the first plot with dataset 1

```{r}
ggplot(data = df) +
  geom_point(mapping = aes(x = x1, y = y1))
```

The ggplot2 commands have the same basic outline:

```
```{r, eval=FALSE}
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```
```

6 Exercise

- Run `ggplot(data = df)`. What do you see?
- Create the plot for the other datasets as well.

Statistical analysis

With ggplot2, statistical analyses of the data and the visualization of the results is relatively easy.

These are implemented in so called `geoms`, which are geometrical objects that a plot uses to represent data. Different `geoms` can be combined to show different aspects of the data.

For example, `geom_smooth` provides a statistical smoothing of the data.

```
```{r}
ggplot(data = df) +
  geom_smooth(mapping = aes(x = x1, y = y1))
```
```

Two `geoms` can be easily combined with the + operator.

```
```{r}
ggplot(data = df) +
  geom_point(mapping = aes(x = x1, y = y1)) +
  geom_smooth(mapping = aes(x = x1, y = y1))
```
```

The `geom_smooth()` function (synonymous with the `stat_smooth()` function) also allows to select the smoothing function, *i.e.* a linear model. Note: If the mapping function is applied within the `ggplot` function, it is also used for the subsequent `geoms` and has to be only defined once.

```
```{r}
ggplot(data = df, mapping = aes(x = x1, y = y1)) +
  geom_point() +
  stat_smooth(method = "lm", col = "red")
```
```

7 Exercise

- By looking at the help function or an internet search, find out what the blue line and gray area in the `geom_smooth` function indicate for the linear model and in the default setting.
- Create the corresponding plot for the other datasets as well.
- Change the label of the plot with the `labs` function.

Creating publication-ready plots with ggplot2

There are several packages available to contribute to the publication ready plot.

For example, the package `cowplot` allows to create nicely formatted panels of plots. You find a nice introduction to this package here:
https://cran.r-project.org/web/packages/cowplot/vignettes/introduction.html

First install the package and load it.

```
```{r}
#install.packages("cowplot")
library(cowplot)
```
```

Then create the plots for the four datasets again and assign a variable to each plot.

```
```{r}
plot1 <- ggplot(data = df, mapping = aes(x = x1, y = y1)) +
  geom_point() +
  stat_smooth(method = "lm", col = "red")
```
```

Now, using cowplot the datasets can be combined to make a nice grid:

```
```{r}
plot_grid(plot1, plot1, labels = c("A", "B"))
```
```

8 Exercises

- Create a 2x2 grid with cowplot for all four datasets.
- Save the grid plot as a pdf with `save_plot()`.

Further reading

- https://r4ds.had.co.nz/data-visualisation.html : Excellent introduction into `ggplot2`
- A theoretical introduction into the grammar of graphics:
  http://vita.had.co.nz/papers/layered-grammar.pdf