R-Bootcamp (day 2)

Dr. Matteo Tanadini, Claude Renaux & Co.

31 January - 3 February 2022

Outline

1. Graphics I

2. Graphics II

3. Graphics III

Section 1

Graphics I

Graphics I (1)

Any statistical analysis involves displaying the data and the results of the modelling phase in a graphical way.

Graphs are created for several reasons:

- inspect the correctness of the data
- formulate hypotheses to be tested
- decide on the methods to analyse data
- visualise the results of the modelling phase
- to better communicate results of an analysis
- ...

Graphics I (2)

R comes with several graphical functions:

- `plot()`
- `boxplot()`
- `hist()`
- `pairs()`
- ...

In addition to basic-**R** graphical functions, there are hundreds of graphical add-on packages.

Graphics I (3)

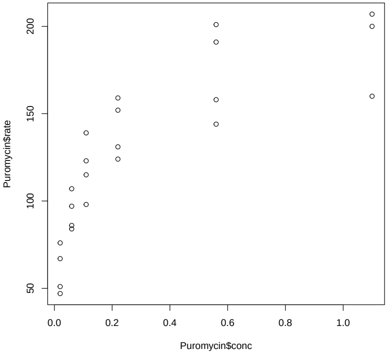
The `plot()` function

```r
head(Puromycin)

  conc rate   state
1 0.02   76 treated
2 0.02   47 treated
3 0.06   97 treated
4 0.06  107 treated
5 0.11  123 treated
6 0.11  139 treated

## see ?Puromycin
```

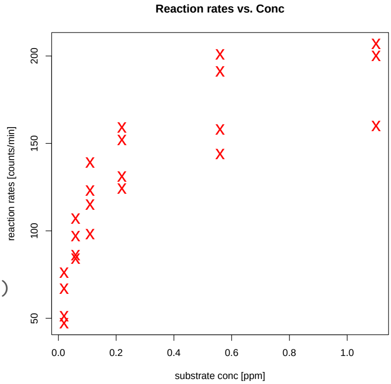
```r
## a simple scatterplot
plot(y = Puromycin$rate,
     x = Puromycin$conc)
```



Graphics I (4)

The plot() function

```r
## a more complex scatterplot
plot(y = Puromycin$rate,
     x = Puromycin$conc,
     pch = "x",
     col = "red",
     cex = 2,
     main = "Reaction rates vs. Conc",
     xlab = "substrate conc [ppm]",
     ylab = "reaction rates [counts/min]")
```



Graphics I (5)

Graphical parameters[1]

- "y" and "x": dimension to display
- "pch": plotting character
- "col": colour
- "cex": expansion factor
- "main": main title
- "ylab": label of the y-axis
- ...

[1] Note that these arguments are to be found in many other graphical functions, not only in `plot()`.

Graphics I (6)

Graphical parameters

- arguments can be set to a non-default values
  - using named values (e.g. $lty = "dashed")$
  - using numbers (e.g. $lty = 2$)
- named values are preferred over number as they more explicit. This makes the code easier to read
- arguments can take vectors (e.g. $col = Puromycin\$state$)
- Google is you best friend...

Graphics I (7)

The formula interface:

```r
plot(y = Puromycin$rate,
     x = Puromycin$conc,
     col = Puromycin$state)
##
## same as
plot(rate ~ conc, data = Puromycin,
     col = state)
```

The formula interface is available in most graphical AND modelling functions.

☞ Go to Democode ☞

☞ Go to Series 2 exercise 1 ☞

Section 2

Graphics II

Graphics II (1)

In base R there are 5 types of graphical functions

- high-level plotting functions
- low-level plotting functions
- control functions (e.g. `par()`)
- device control functions (e.g. `jpeg()`)
- interactive functions

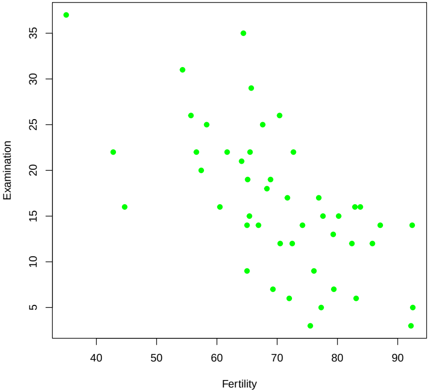
Graphics II (2)

- high-level plotting functions such as `plot()` generate a new graph
- low-level plotting functions such as `abline()` add elements to an existing graph
- control functions such as `par()` allow us to control the visual aspect of a graph
- device control functions such as `jpeg()` allow us to save graphs as files
- interactive functions allow us to interact with graphs

Graphics II (3)

High-level plotting functions

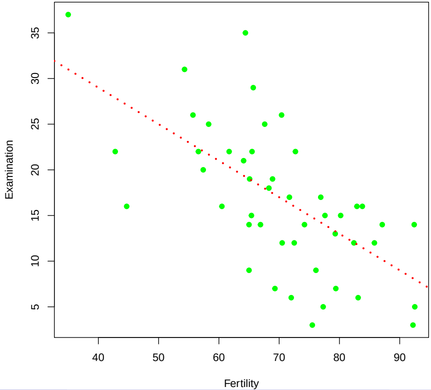
```
plot(Examination ~ Fertility, data = swiss,
     col = "green", pch = 19)
```



Graphics II (4)

Low-level plotting functions

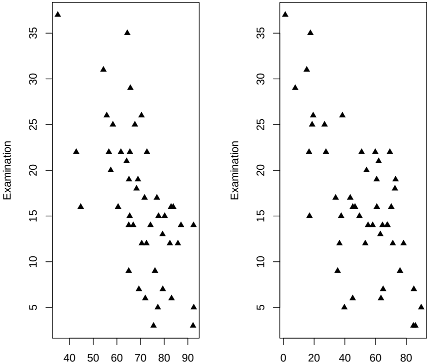
```
plot(Examination ~ Fertility, data = swiss,
     col = "green", pch = 19)
abline(a = 45, b = -0.4,
       col = "red", lty = "dotted", lwd = 3)
```



Graphics II (5)

Control functions

```
par(mfrow = c(1, 2), ## two  graphs in one device
    pch =  17) ## all graphs with triangles
##
plot(Examination ~ Fertility, data = swiss)
plot(Examination ~ Agriculture, data = swiss)
```



Graphics II (6)

Device control functions

```r
jpeg("MyPlotsFor_Bootcamp.jpeg")
par(mfrow = c(1, 2), ## two  graphs in one device
    pch =  17) ## all graphs with triangles
##
plot(Examination ~ Fertility, data = swiss)
plot(Examination ~ Agriculture, data = swiss)
dev.off()
```

Graphs can also be exported via the Rstudio interface (see ”Export” in the ”Plots” pane).

Nowadays graphs are stored automatically when ”Dynamic Documents” are used (more comes later).

☞ Go to Democode ☞
☞ Go to Series 2 exercise 2 ☞

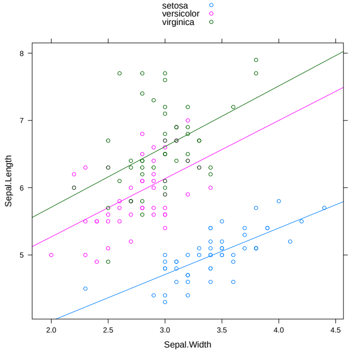
Section 3

Graphics III

Graphics III (1)

Base **R** graphical functions are very powerful, however, creating even simple graphs can sometimes be very time-consuming.

```
library(lattice)
xyplot(Sepal.Length ~ Sepal.Width,
       groups = Species,
       type = c("p", "r"),
       data = iris,
       auto.key = TRUE)
```



Graphics III (2)

The add-on packages {`lattice`} and {`ggplot2`} allow us to create beautiful graphics in a very code-efficient way. Among the most important functionalities of these package we mention:

- panelling
- grouping
- adding summary statistics (e.g. regression lines, smoothers)

Take Home Messages: Graphics III

- the base **R** graphical functions are versatile and very powerful
- however, sometimes even simple graphs require a long and cumbersome code
- add-on graphical packages are extremely powerful and extremely user friendly
- currently the best add-on graphical packages are {`lattice`} and {`ggplot2`}[2]

☞ Go to Democode ☞

☞ Go to Series 2 exercise 3 ☞

☞ See fancy example with Sarah ☞

[2] Both these packages come with an excellent companion book.