Data Visualization in R with ggplot2

Peter Pütz, CRC 990, University of Göttingen

August 13, 2019

“The simple graph has brought more information to the data analyst’s mind than any other device.”

John Tukey

Aims of this workshop & organisation

At the end of this R workshop, we should be able to…

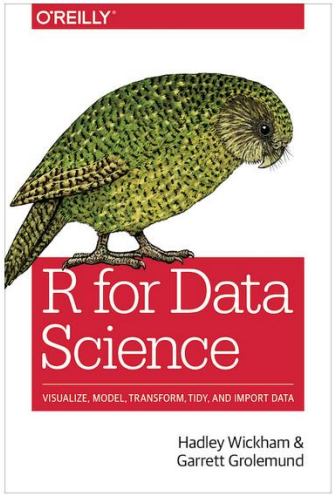
- … visualize data using the ggplot2-package
- … understand the basic grammar of graphics, the idea that you can build every graph from the same components

Organisational remarks:

- First part of the course: Lecture and exercises (alternating)
- Second part of the course: Exercises
- All R Code will be made available after the course

ggplot2

- Part of the tidyverse packages such as dplyr, readr etc.

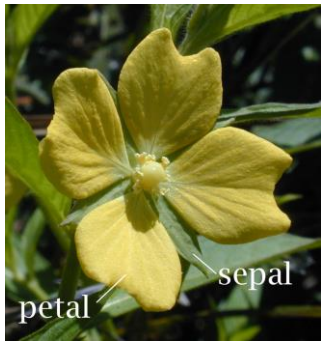


Available online: https://r4ds.had.co.nz/index.html

Dataset for today

- Edgar Anderson's famous Iris Data (base R dataset, https://en.wikipedia.org/wiki/Iris_flower_data_set)

- Dataset gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris.




```r
# Load packages
library(ggplot2)
library(dplyr)

# get familar with the dataset
?iris

dim(iris)
```

```
## [1] 150   5
```

```r
class(iris)
```

```
## [1] "data.frame"
```

```r
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

Exercise 1

i. Load the ggplot2 package.

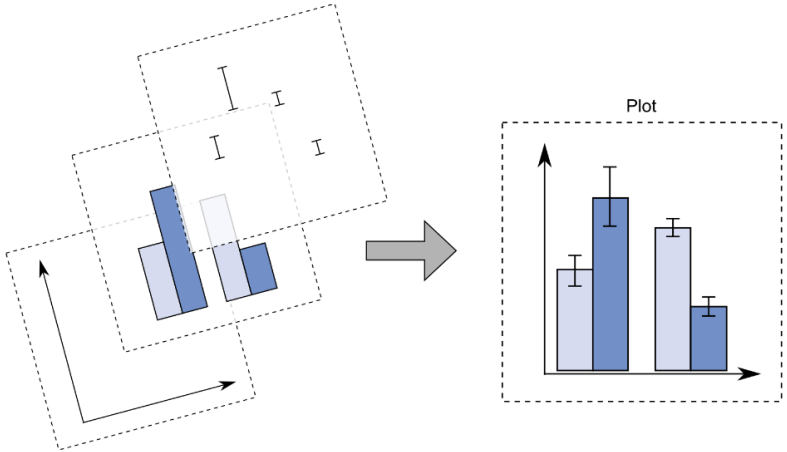
ii. Get familiar with the iris dataset.

General recommendations for solving the exercises:

- Use the provided cheat sheet, the internet and the R help.
- Ask your colleagues or the course instructor for help.

The philosophy of ggplot2

- ggplot2 uses “layered grammar of graphics”.
- Basic idea is that you can split a chart into components of a graphic such as data, coordinate system, etc. and think about them separately. When putting them all together, you get a complete chart.



The general form of a ggplot2 command

- In ggplot2, a graph is made up of a series of layers.
- *ggplot()* initializes a ggplot object. It is used to declare the input data frame for a graphic and to specify the set of plot aesthetics intended to be common throughout all subsequent layers unless specifically overridden.
- *ggplot()* creates a graph object, but no graphical elements. Using the "+" operator, various layers can be added, each with its own aesthetics.

```
MyObject <- ggplot(Dataset, aesthetic mapping ... ) +
  geom_function( ... )
```

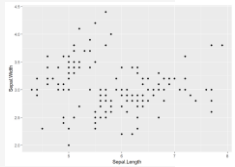
Geometrical objects and aesthetic mapping in ggplot2

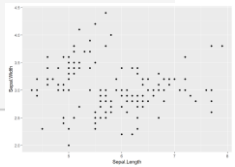
- *Geoms* are geometrical objects that a plot uses to represent data, e.g., points.
- Aesthetic mappings describe how variables in the data are mapped to visual properties (*aesthetics=aes*) of *geoms*.
- Part of the *aesthetics* is the mapping of x and y variables to the plot axes.
- These can be assigned as a default in *ggplot()* or repeatedly in each *geom* – layer.

```
# Create a scatterplot to check the relation between Sepal.Length and Sepal.Width

# Configure x and y axis as a default for further layers
Myplot <- ggplot(data = iris, aes(x = Sepal.Length,
                                  y = Sepal.Width)) +
  geom_point()
Myplot

# Create the same plot while mapping x and y axis only to one layer
Myplot <- ggplot(data = iris) +
  geom_point(aes(x = Sepal.Length,
                 y = Sepal.Width))
Myplot
```





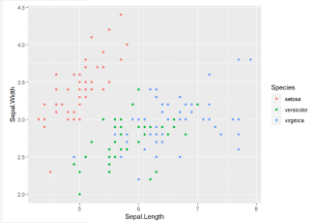
Exercise 2

i. Using *geom_point()*: Explore the relation between the variables **Petal.Length** and **Petal.Width** graphically. What can you see?

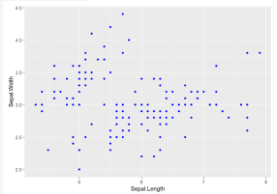
Colours and the difference between *Mapping* and *Setting*

- Again: Aesthetic mappings describe how variables in the data are mapped to visual properties (*aesthetics*) of *geoms*.
- Colour and size of points can also be dependent on a variable.
- Assigning all points to one color is not a aesthetic mapping as the command is not connected to variables. Instead, the color can be specified as a *Setting* in the corresponding *geom*.

```
# A Scatterplot of Sepal.Length and Sepal.Width
# Color dependent on Species
Myplot <- ggplot(data = iris) +
  geom_point(aes(x = Sepal.Length,
                 y = Sepal.Width,
                 color = Species))
Myplot
```



```
# A Scatterplot of Petal.Length and Petal.Width
# Color "Blue"
Myplot <- ggplot(data = iris) +
  geom_point(aes(x = Sepal.Length,
                 y = Sepal.Width),
             color = "Blue")
Myplot
```



Exercise 3

i. Create (again) a scatterplot with the variables **Petal.Length** and **Petal.Width**. Use the variable **Species** to color the points. What can you see?

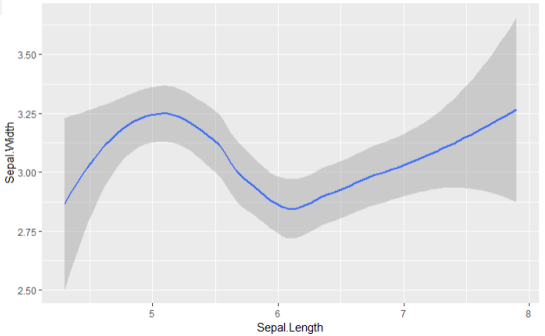
Geoms in ggplot2

- Again: Geometric objects (*geoms*) are geometrical objects to represent the data.

- Apart from *geom_point()*, ggplot2 contains multiple different types of *geoms*. Some popular ones are:

  - *geom_text()* / *geom_label()* – adds text/labels instead of e.g. points
  - *geom_bar()* – creates a bar chart
  - *geom_line()* – creates a line diagram, connecting observations by x-value
  - *geom_boxplot()* – creates a Box-and-whisker plot
  - *geom_smooth()* – to add a smoothed conditional mean

```
# A smooth relationship between Sepal.Length and Sepal.Width
Myplot <- ggplot(data = iris) +
  geom_smooth(aes(x = Sepal.Length,
                  y = Sepal.Width))
Myplot
```



Exercise 4

i. Using the provided *geoms*, create a boxplot of the variable **Petal.Width** for all types of **Species**. Color all types of species differently.

Themes and appearance

- ggplot2 uses a default theme that can be changed by adding further layers.
- Themes can be chosen as pre-made, (e.g. with *theme_minimal()*, *theme_dark()*) or created by yourself using *theme()*.
- Labels and legends can be changed outside of themes, e.g. using functions like *ggtitle*, *xlab()*, *ylab()*.

Exercise 5

i. Look for preexisting themes and change the theme of your boxplot from exercise 4. Again: The cheat sheet might help.

ii. Improve your boxplot by changing labels of the axis using *xlab()* and *ylab*.

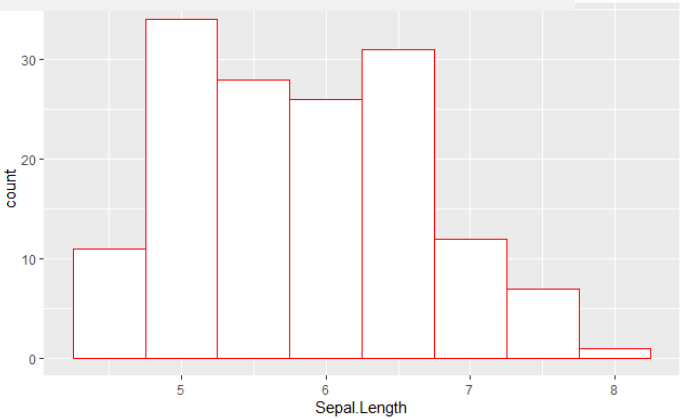
iii. Add a title using *ggtitle()*.

iv. Remove the guide on the right side of your boxplot using *guides()*.

Stats in the ggplot environment

- Sometimes, it is more convenient to visualize transformations of the chosen data.
- Each *stat* computes such a transformation and creates the resulting variable.
- E.g., *geom_histogram()* produces a histogram by using counts of a numerical variable within bins, i.e. by dividing the x axis into bins and counting the number of observations in each bin.
- Stat functions can also be explicitely called, e.g. using *stat_bin()*.

```
# A histogram using counts of a numerical variable within
bins, i.e., creating automatically a count variable
Myplot <- ggplot(data = iris,aes(x = Sepal.Length)) +
  geom_histogram(binwidth=0.5,fill="white",colour="red")
Myplot
```



Exercise 6

i. Create a histogram of the variable **Petal.Width** using both *geom_histogram()* and *stat_bin()*.

Faceting

- With *facet_grid()*, plots can be divided into subplots using one (or more) discrete variables.
- This way, subsets of the data can be plotted next to each other.
- To be able to properly compare different variables, it is important to pay attention to the scaling in a grid.

Exercise 7

i. Recreate your plot from exercise 3. Using *geom_smooth()* add an additional layer visualizing a linear regression of the two variables. Mark the line black and don't show confidence intervals.

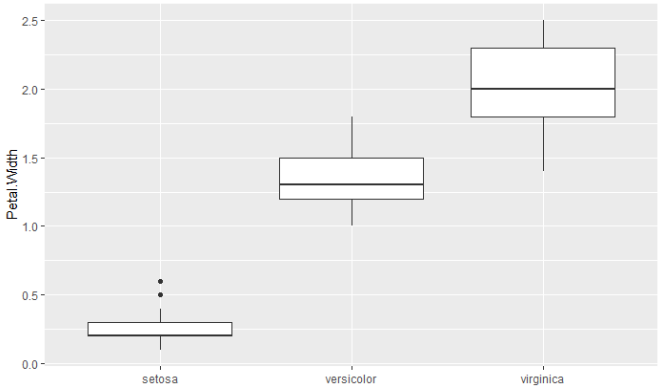
ii. Using *facet_grid()* divide the plot from i. into subplots, dependent on the variable **Species**. What changed?

iii. Plot the same plot again, this time allowing free scales in the grids. What changed? When and where would you use the two plots?

What else about the ggplot2 package?

- *qplot() is* very convenient to generate non-complex plots if you're used to base *plot()*.
- There is much more what the ggplot2 package and *ggplot* can do.
- For your specific problem and the exercises, the cheat sheet might help: https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf

```
# use qplot for a boxplot
qplot(data = iris, factor(Species),Petal.Width, geom = c("boxplot"),xlab="")
```



Extensions for ggplot2

- Various extensions for ggplot2 exist in the form of packages. Worth to mention are:
  - *gganimate* – for animated ggplots (e.g., https://gganimate.com/)
  - *ggthemes* – for extra geoms, scales and themes
  - *ggraph* – for plotting graph-like data structures (e.g. networks, trees)
  - *esquisse* – for interactively exploring data without writing code
  - More can be found here: https://www.ggplot2-exts.org/gallery/

Summary and miscellaneous

- ggplot2 provides a coherent system (grammar) to visualize data.
- Default choices in ggplot2 often adequate (but you can customize!).
- More generic and elegant in comparison to base R plots (e.g., plot, histogram, etc.).
- Always combine graphical tools with simple descriptive statistics
  - Summary statistics
  - Outlier and missing value analysis
  - …

References and useful links

- https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf
- https://www.bioinformatics.babraham.ac.uk/training/ggplot_course/Introduction%20to%20ggplot.pdf
- http://www.cookbook-r.com/Graphs/
- https://ggplot2.tidyverse.org/