



# Data Analytics in R

[github.com/Mathijs995/Data-Analytics-in-R](https://github.com/Mathijs995/Data-Analytics-in-R)

---

Mathijs de Jong

20 February 2020



1. Introducing R
2. Programming environment
3. R fundamentals
4. Data visualization using `ggplot2`
5. Assignments

# Introducing R

---



- Programming language & free software environment
- Statistical computing and data visualization
- First appeared: August 1993
- Written in **C**, **Fortran** and **R** itself



- Data manipulation
- Data visualization
- Analyzing data
  - Advanced statistical methods
  - Machine learning
- And much more!



- Very powerful for data analytics
- Simple and relatively easy to learn
- Popular both in academia and in the business world
- Open source & excellent package support
- Supported by main data analytics software including **Alteryx**, **Power BI** and **Tableau**.

## Programming environment

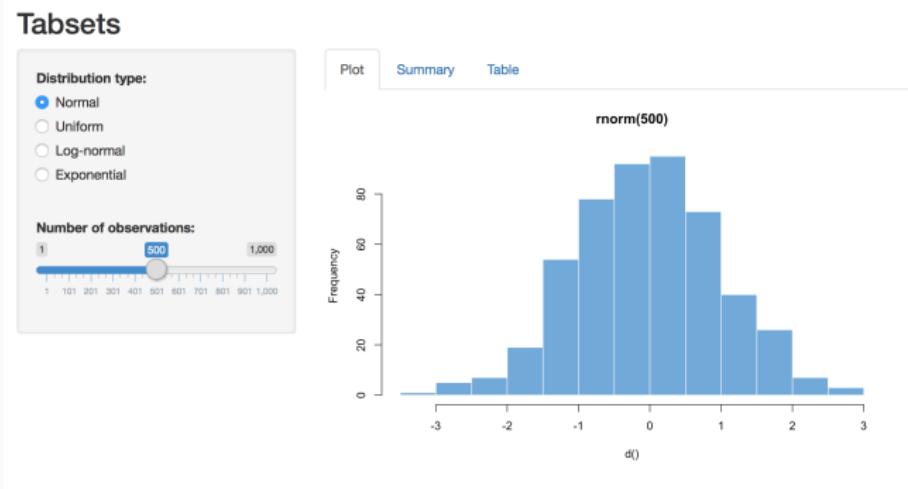
---



- Download RStudio or use RStudio Cloud
- Available on all computers at the EUR
- When you are comfortable with R, you can switch to
  - R Markdown
  - Jupyter notebooks
  - Shiny



- Interactive web applications
- Best library for dashboarding





~/Documents/r4ds/data-analysis - RStudio

mpg-plot.R

```
1 library(ggplot2)
2
3 ggplot(mpg, aes(x = displ, y = hwy)) +
4   geom_point(aes(colour = class))
5
```

**Editor**

1:1 (Top Level) R Script

Console ~ /Documents/r4ds/data-analysis/

```
> library(ggplot2)
> ggplot(mpg, aes(x = displ, y = hwy)) +
+   geom_point(aes(colour = class))
> |
```

**Console**

Environment History

Global Environment

Environment is empty

Files Plots Packages Help Viewer

Zoom Export

**Output**

highway (hwy)

displacement (displ)

class

- 2seater
- compact
- midsized
- minivan
- pickup
- subcompact
- suv

## R fundamentals

---



# Variables

- Building blocks used for storing information
- Dynamically typed, meaning that the type of data is determined when the statement is run

## Defining variables

```
1 # Text behind the # is ignored; these are comments.  
2 a = 10 # Store 10 in the variable a  
3 a <- 10 # Same as above, not recommended
```



## Data types (1/3)

- Data types:
  - **character** – String
  - **integer** – Integers
  - **numeric** – Integers + Fractions
  - **logical** – Boolean (true or false)
  - **factor** – Element in a given set



## Different data types

```
1 sentence = "Hello world!"  
2 class(sentence) # character  
3  
4 x = 1L # or as.integer(1)  
5 class(x) # integer  
6  
7 pi = 3.14159265359  
8 class(pi) # numeric  
9  
10 is_clear = T # or TRUE  
11 class(is_clear) # boolean
```



## Data types (3/3)

- Sometimes you want to restrict the number of options a variable can take
- R introduced the `factor` data type for this purpose
- Always check whether you are working with character or with factor variables

### Working with factors

```
1 my_gender = factor("female", levels = c("female", "male"))
2 my_gender = factor("unspecified", levels = c("female", "male"))
```



# Data structures

- Data structures:
  - **vector** – A collection of elements of the same class
  - **matrix** – A matrix (table) of elements of the same class
  - **data.frame** – A matrix (table) with columns containing elements of the same class
  - **list** – A collection of elements of different classes

Dimension	Homogeneous	Heterogeneous
1	Atomic vector	List
2	Matrix	Data frame
$n^1$	Array	

---

<sup>1</sup>For any  $n = 1, 2, 3, \dots$



- RStudio GUI
- Using the R console:

## Importing and exporting Excel data

```
1 # First make sure you are in the right directory using setwd() and getwd()  
2  
3 # Import a csv file and store it in data  
4 data = read.csv("SMT1920casus - data.csv", sep = ";")  
5  
6 # Install the required library  
7 install.packages("xlsx")  
8  
9 # Export a data structure as the Excel file 'output.xlsx'  
10 xlsx::write.xlsx(data, "output.xlsx")
```



- You will mostly be working with data frames to store your data
- Think of a data frame as a table in Excel
- It contains columns with a name (header) and an equal number of rows per column



## Commonly used functions

```
1 view(airquality)      # view the data frame  
2 class(airquality)    # get class  
3 sapply(airquality, class) # get class of all columns  
4 str(airquality)       # structure  
5 summary(airquality)   # summary of airquality  
6 head(airquality)      # view the first 6 obs  
7 fix(airquality)        # view spreadsheet like grid  
8 rownames(airquality)   # row names  
9 colnames(airquality)   # columns names  
10 nrow(airquality)      # number of rows  
11 ncol(airquality)      # number of columns
```



## Data frame operations

```
1 cbind(myDf1, myDf2) # columns append DFs with same no. rows  
2 rbind(myDf1, myDf1) # row append DFs with same no. columns  
3 myDf1$vec1 # vec1 column  
4 myDf1[, 1] # df[row.num, col.num]  
5 myDf1[, c(1,2)] # columns 1 and 3  
6 myDf1[c(1:5), c(2)] # first 5 rows in column 2  
7 myDf1[c(1:5), -c(2)] # first 5 rows in all columns but column 2
```



- To add logic to your workflow, you can use conditional statements
- The standard conditional statement is the If-Else condition:

## If-Else condition

```
1 if(checkConditionIfTrue) {  
2     ....statements..  
3     ....statements..  
4 } else { # place the 'else' in same line as '}'  
5     ....statements..  
6     ....statements..  
7 }
```



- Alternatively, you can use the `ifelse()` function for vector based conditional statements

### Application of the `ifelse()` function

```
1 possible_grades = 1:10  
2 pass = ifelse(possible_grades >= 6, TRUE, FALSE)
```



- Standard for-loop:

## For loop

```
1 for(counterVar in c(1:n)){  
2     .... statements..  
3 }
```



- The **apply** family
  - **apply()**: Apply FUN through a data frame or matrix by rows or columns.

## Demonstration of the **apply** function

```
1 myData = matrix(seq(1,16), 4, 4)    # make a matrix
2 apply(myData, 1, FUN=min)        # apply 'min' by rows
3 #=> [1] 1 2 3 4
4
5 apply(myData, 2, FUN=min)        # apply 'min' by columns
6 #=> [1] 4 8 12 16
```



- Reusable block of code

## Creating functions

```
1 myfunction = function(arg1, arg2, ... ){  
2     .... statements  
3     return(object)  
4 }
```



- Build-in support
- Google
- Source code

## Getting help

```
1 help(merge)    # get help page for 'merge'  
2 ?merge          # lookup 'merge' from installed pkgs  
3 ??merge         # vague search  
4 example(merge)  # show code examples  
5 help
```

# What is a R package?



- Packages are collections of
  - R functions
  - R data
  - compiled code
- Can be created by anyone
- Academics and companies often create and share their own packages



- Using the RStudio GUI
- From the R console:

## Installing and loading a package

```
1 install.packages("tidyverse") # install 'tidyverse' package (and its dependencies)
2 library(tidyverse) # initialize the 'tidyverse' package
3 require(tidyverse) # another way to initialize
```

## Data visualization using `ggplot2`

---



# Workflow of a data science project

- Workflow of a data science project:
  - Step 1:** Collect data
  - Step 2:** Manipulate data
  - Step 3:** Visualize data
  - Step 4:** Apply statistical methods
  - Step 5:** IF results are not satisfactory THEN return to **Step 2.**
- Why is R a better option for this type of projects than Excel? When would Excel be a better choice?



- `ggplot2` is a system for declaratively creating graphics, based on [The Grammar of Graphics](#)
- Best package for visualization available
- Check out the cheat [cheat sheets](#) using the link or the last two slides of this presentation!



- Let's get started!

## Install and load the parent library

```
1 # Run this block to install and load the tidyverse package  
2 install.packages("tidyverse")  
3 library(tidyverse)
```



- To create graphs using `ggplot2` you have to specify:
  - What do you want to use (preferably as a data frame)
  - Which visualization you want to use (`geom_function`)
  - How the data is mapped in the visualization (`aesthetics`)

## Graph structure

```
1 ggplot2::ggplot(data = <DATA>) +  
2   <GEOM_FUNCTION>(mapping = ggplot2::aes(<MAPPINGS>))
```



- Let's say we want to visualize the distribution of the gender of students in the Applied Business Methods (ABM, DUTCH: Statistische Methoden en Technieken) data set
- The code below creates a histogram of the distribution of the genders in the data set:

## Gender distribution plot

```
1 ggplot(data = data, mapping = aes(x = dGender))  
2   + geom_histogram(stat = "count")
```



- The best way to learn how to code is by trial and error!
- Start by working your way through the following tutorial:
  - [ggplot2 short tutorial](#)
- Afterwards, continue with the assignments on the next slides.
- **Optionally:** Go through the following tutorials:
  - [ggplot2 tutorial 1 - Introduction](#)
  - [ggplot2 tutorial 2 - Theme](#)
  - [ggplot2 tutorial 3 - Masterlist](#)

## Assignments

---



- Create a distribution plot for the year of birth of the students in the ABM data set using `ggplot2` with a histogram.
- Add a legend, title and rename the x-axis and the y-axis.



- Transform the gender column in the ABM data set into a factor column with levels "female" and "male".
- Transform the living conditions column in the ABM data set into a factor column with appropriate levels.
- Create a stacked column chart for the living conditions of the students grouped by gender.
- Add a legend, title and rename the x-axis and the y-axis.

## Assignment 3



- Transform the stacked column chart of the previous assignment into a grouped column chart.



- Interpret the code below.

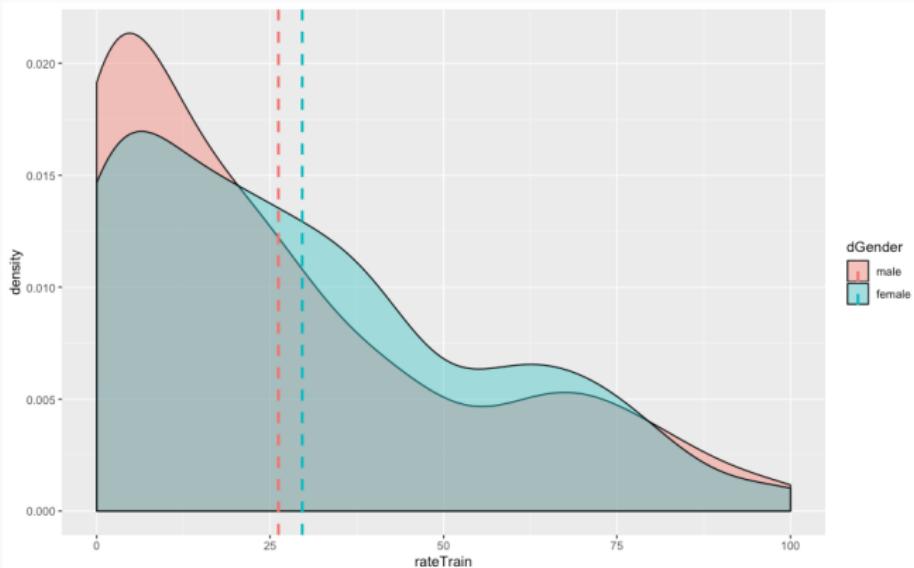


## Distribution of train ratings by gender

```
1 # Distribution of train ratings by gender
2 library(plyr)
3 mu = ddply(data, "dGender", summarise, grp.mean=mean(rateTrain))
4 ggplot(data = data, mapping = aes(x = rateTrain, fill = dGender)) +
   → geom_density(alpha=0.4) +
5   geom_vline(data = mu, aes(xintercept=grp.mean, color = dGender),
   →   linetype="dashed", size=1)
```

- Have a look at the graph produced by the code above on the next slide.
- Adapt the code above to display the distribution of airplane ratings by gender.

# Assignment 4 (3/3)





- Go through the resources for using facets.
- Think about where using facets could be helpful.
- Choose one of the situations you thought of above and create a graph using a facet.

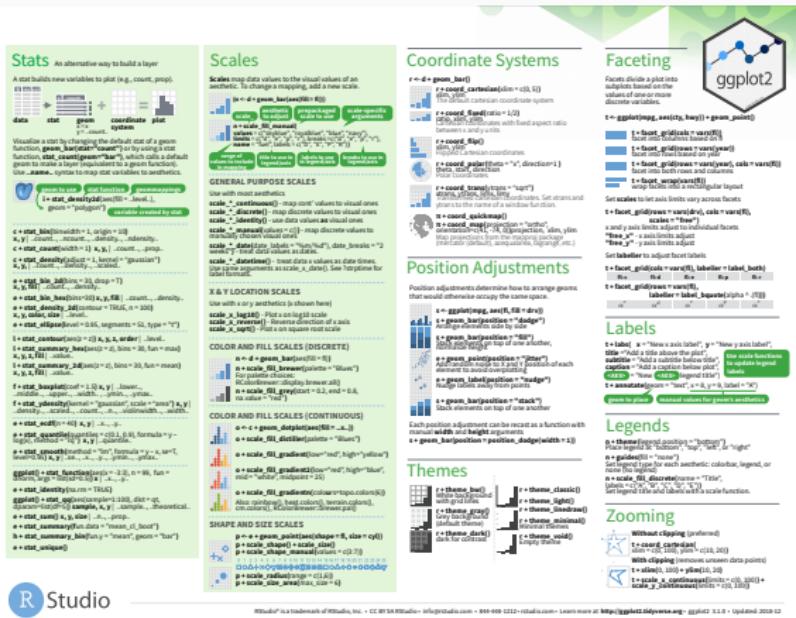
**Questions?**

# Thank you!

For questions, you can always contact me at  
[mathijs@accelerytics.com](mailto:mathijs@accelerytics.com).



## ggplot2 cheat sheets (2/2)



**Figure 2:** Page 2 of the cheat sheet. Source:

<https://github.com/rstudio/cheatsheets/blob/master/data-visualization-2.1.pdf>