Getting started with graphics

Andreas Alfons Pieter Schoonees

[1]Erasmus School of Economics, Erasmus Universiteit Rotterdam

[2]Rotterdam School of Management, Erasmus Universiteit Rotterdam

EQI Programming & Visualization for Business Analytics,
September 7, 2018

Course documents and software

- $\longrightarrow$ Canvas
- $\longrightarrow$ R package euR contains additional functions and data sets used in this course

Make sure to have the latest versions of the following software installed:

- R: `http://CRAN.R-project.org`
- RStudio: `http://RStudio.com`

About the lecturer

Andreas Alfons

- PhD in Statistics from Vienna University of Technology
- Assistant professor at the Econometric Institute, ESE
- Research interests:
  - Robust statistics
  - Computer-intensive methods
  - Predictive modeling
- R user and developer for 10+ years
- Austria

Lecture structure

- Materials: Lecture slides with R script; exercises with R script solutions (afterwards)

  $\longrightarrow$ Also, additional material from R for Data Science book

- Lecture for 20 min, exercises for 20 min, repeat

  $\longrightarrow$ Practice (at home) makes perfect!

- Lecturers can help if you get stuck

  $\longrightarrow$ Do not fall behind

- Learn by helping each other

  $\longrightarrow$ Also on the discussion board on Canvas

- Consult (1) course material and (2) book before other sources

  $\longrightarrow$ There are plenty of ways to skin a cat

Assignment

Two parts:

1. Midterm assignment after week 4

   $\longrightarrow$ Due 7 October 2018

2. Final assignment after week 7

   $\longrightarrow$ Due 28 October 2018

$\longrightarrow$ Similar to the in-class / take home exercises

Content

About R

About R

- Open source environment for statistical computing
- Free as in “free speech” and “free beer”
- Cross platform: Linux/Unix, Mac OS X, Microsoft Windows
- Fully developed and easy-to-use programming language
- Extensible by community contributed packages

  $\longrightarrow$ Roughly 16000 packages on CRAN, Bioconductor and Github
- Support for businesses by Microsoft:

  `https://www.microsoft.com/en-us/cloud-platform/r-server`

$\longrightarrow$ More information on `http://www.R-project.org/`

R books

- Wickham and Grolemund (2017): R for Data Science (read it online for free on `http://r4ds.had.co.nz/`)
- Kabacoff (2015): R in Action
- Wickham (2009): `ggplot2`: Elegant Graphics for Data Analysis
- Specialized topics:
  - Springer's Use R! series
  - Chapman & Hall/CRC's The R Series

R resources

- Interactive online learning platform DataCamp: `http://www.DataCamp.com`
- Tutorial collection Quick-R: `http://www.statmethods.net`
- Aggregate news and tutorial site R-bloggers: `http://www.R-bloggers.com`
- Scientific articles on R software:
  - Journal of Statistical Software: `http://www.jstatsoft.org`
  - R Journal: `http://journal.R-project.org/`
- R for Matlab users (comparison of R and Matlab commands): `http://mathesaurus.sourceforge.net/octave-r.html`

Design principles

- Interactive data analysis via command line interface (CLI)
  $\longrightarrow$ Enter command and resulting output is printed
- Easy transition from user to developer
  $\longrightarrow$ Integrated development environment (IDE): e.g., RStudio
- Vectorized arithmetic
  $\longrightarrow$ Scalars are vectors of length 1
- Data frames for storing variables of different types
- Formula interface for model specification: `y ~ x1 + x2`

What is R used for?

1. Collecting data
   - Scrape from the web, import from databases, ...
2. Preparing, exploring and cleaning data
   - Data wrangling, exploratory data analysis, plotting
3. Modelling
   - Regression, segmentation, machine learning, custom methods, ...
4. Model evaluation
   - Assessing model quality
5. Reporting results
   - Writing (dynamic) reports, visualization, creating dashboards, ...

$\longrightarrow$ And various other tasks

Who Uses R?

- Google
- Twitter
- Facebook
- New York Times
- John Deere
- Deloitte
- Credit Suisse
- Novartis
- eBay
- Ford Motor Company
- Kickstarter
- Uber
- Airbnb
- Booking.com
- Bank of America
- McKinsey & Company
- FourSquare

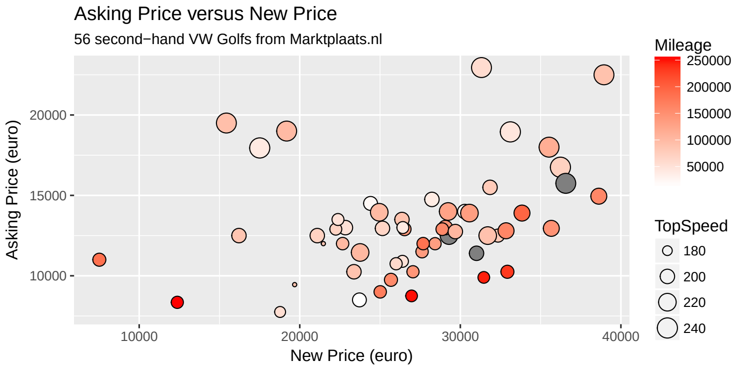
$\longrightarrow$ You too?

Who Uses R?

- Google
- Twitter
- Facebook
- New York Times
- John Deere
- Deloitte
- Credit Suisse
- Novartis
- eBay
- Ford Motor Company
- Kickstarter
- Uber
- Airbnb
- Booking.com
- Bank of America
- McKinsey & Company
- FourSquare

$\longrightarrow$ You too?

Target

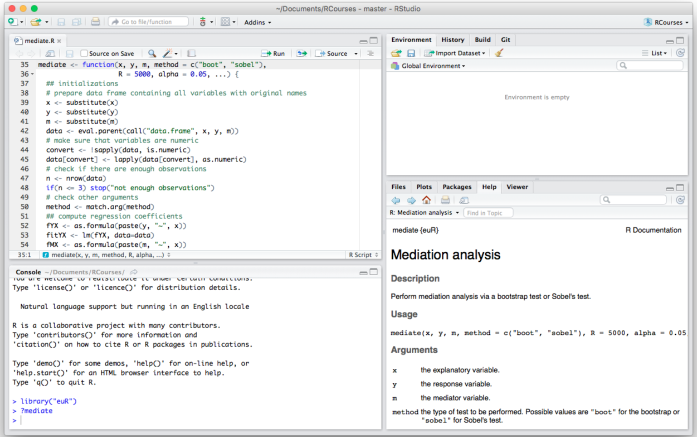


⟶ After week 2

Getting started with R and RStudio

RStudio

⟶ Available from `http://RStudio.com`



```r
mediate <- function(x, y, m, method = c("boot", "sobel"),
                    R = 5000, alpha = 0.05, ...) {
  ## initializations
  # prepare data frame containing all variables with original names
  x <- substitute(x)
  y <- substitute(y)
  m <- substitute(m)
  data <- eval.parent(call("data.frame", x, y, m))
  # make sure that variables are numeric
  convert <- !sapply(data, is.numeric)
  data[convert] <- lapply(data[convert], as.numeric)
  # check if there are enough observations
  n <- nrow(data)
  if(n <= 3) stop("not enough observations")
  # check other arguments
  method <- match.arg(method)
  ## compute regression coefficients
  fYX <- as.formula(paste(y, "~", x))
  fitYX <- lm(fYX, data=data)
  fMX <- as.formula(paste(m, "~", x))
```

```
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> library("euR")
> ?mediate
>
```

mediate {euR}   R Documentation

Mediation analysis

Description

Perform mediation analysis via a bootstrap test or Sobel's test.

Usage

`mediate(x, y, m, method = c("boot", "sobel"), R = 5000, alpha = 0.05`

Arguments

`x`	the explanatory variable.
`y`	the response variable.
`m`	the mediator variable.
`method`	the type of test to be performed. Possible values are `"boot"` for the bootstrap or `"sobel"` for Sobel's test.

RStudio: four panels

Top left Script editor

Bottom left R console

Top right Two tabs

- Environment: list of objects used in the session
- History: allows to re-run previous commands

Bottom right Five tabs

- Files: browse through files on the computer
- Plots: graphics are displayed here
- Packages: list of installed packages
- Help: R help files are displayed here
- Viewer: local web content created in the session

How do I use R with RStudio?

Type commands into an R script and execute from there:

1. Create objects
   - Typically contain data, or statistics derived from data
   - For example, data loaded from a spreadsheet
   - You have to pick (informative) names
2. Manipulate these objects to create new ones
   - E.g., transform data, fit a model or create a plot
   - All actions are performed by functions
3. Interpret and report your results

Example session 1

Open RStudio, and open `my_first_script.R` in the script editor

⟶ Available on Blackboard

⟶ Execute the line in which your cursor is with *ctrl+enter*

⟶ You can also type the command directly in the console, but it is better to store your commands in a script (reproducibility)

Using R

To use R (efficiently) you need to understand:

1. Different types of objects and how they behave
2. Various functions and what they do

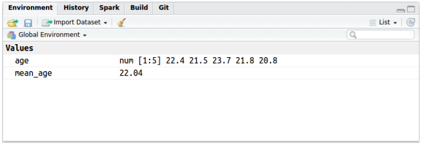
$\longrightarrow$ This takes time, but it is worth it!

Functions & objects

- All actions are performed by functions:
  - We used `c()`, `mean()` and '<-' (an operator function)
  - Takes an object (or objects), and returns a transformed result
- Functions are pieces of code that perform specific actions
  - For example, `mean()` calculates the mean
  - Function usage is described in its help file (for example, `?mean`)
- Functions accept objects as arguments
  - Data to compute on and other options that control output
  - For example, age is a vector object containing ages

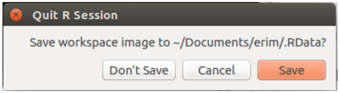
Workspace

Created objects are available in your workspace, also known as the Global Environment:



```
Values
  age                num [1:5] 22.4 21.5 23.7 21.8 20.8
  mean_age           22.04
```

You can save the workspace when you exit R:



Save workspace image to ~/Documents/erim/.RData?

⟶ Better not to: store the code that created the objects

Some details

- R is case sensitive
- The + prompt means R is waiting for you to complete the command
- Press Esc to cancel the command being evaluated
- Use the Tab key for code completion
- Remember to close your parentheses ()
- Press the up and down arrows to browse the command history in the console

Script editor

- ⟶ Idea: perform an analysis and save the commands in a script
- ⟶ Execute current line or selection on R console with keyboard shortcut *ctrl+enter*

Advantages and disadvantages:

- + Reproducibility
- + Easy to perform the same analysis with different data
  - ⟶ Just change the input data
  - ⟶ Particularly useful, e.g., for periodic reports
- – Steeper learning curve than GUI

Script files

Create new script file: *File* → *New file* → *R Script*

Save script file:

- Keyboard shortcut: *ctrl+S* (Windows/Linux), *cmd+S* (Mac)
- *File* → *Save As...* and enter the file name in the dialog

⟶ Use file extension .R

Open existing script file:

- *File* → *Open File...* and select the script file in the dialog
- Click the *Files* tab in the lower right panel, navigate to the script file and click on it

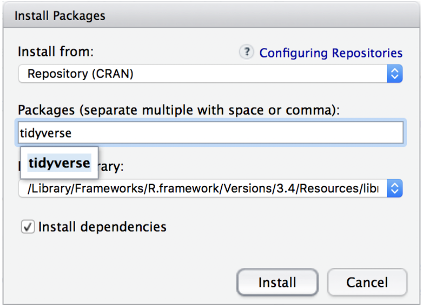
Packages, data and help

Install packages from CRAN

Tools → *Install Packages...*

In the dialog:

- In the *Install from:* box, select *Repository*
- Type the names of the packages into the text box (suggestions are shown as you type)
- Make sure that *Install dependencies* is checked
- Click *Install*


Install Packages

Install from: Repository (CRAN)

Packages (separate multiple with space or comma): tidyverse

/Library/Frameworks/R.framework/Versions/3.4/Resources/libı

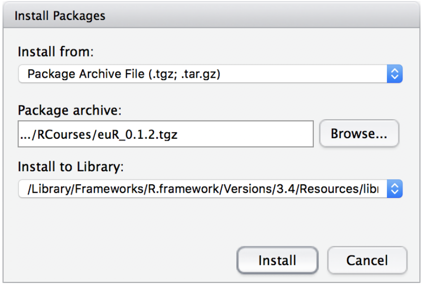
☑ Install dependencies

Install local packages

Tools → *Install Packages...*

In the dialog:

- In the *Install from:* box, select *Package Archive File*
- Click *Browse* and select the package file
- Click *Install*


Install Packages

Install from: Package Archive File (.tgz; .tar.gz)

Package archive: .../RCourses/euR_0.1.2.tgz

Install to Library: /Library/Frameworks/R.framework/Versions/3.4/Resources/libı

Install packages: command line

From CRAN:

```
R> install.packages("tidyverse")
```

Local packages:

Windows:

```
R> install.packages("euR_0.1.2.zip", repos = NULL)
```

Mac:

```
R> install.packages("euR_0.1.2.tgz", repos = NULL)
```

Linux/Unix:

```
R> install.packages("euR_0.1.2.tar.gz", repos = NULL)
```

$\longrightarrow$ If the package is not in the working directory, the full path needs to be specified

Load installed packages

In RStudio:

1. In the lower right panel, click the *Packages* tab
2. Check the box next to the packages to be loaded

On the command line:

```
R> library("euR")
```

$\longrightarrow$ Install a package once on a computer

$\longrightarrow$ Load it in every new session

R help

- $\longrightarrow$ R comes with built-in help facility to get more information on functionality
- $\longrightarrow$ Help topic is usually the name of a function, data set or package

Contributed packages:

- $\longrightarrow$ Help files are required for packages on CRAN
- $\longrightarrow$ Package needs to be loaded to view its help files
- $\longrightarrow$ Overview of all help files within a package is available

View R help files

In RStudio:

1. In the lower right panel, click the *Help* tab
2. Type the topic into the text box on the right (suggestions are shown as you type)

View R help files from the command line

Help for the `help()` function is available:

```
R> ?help
R> help("help")
```

List all help topics within a package:

```
R> help(package = "euR")
```

Run examples from a help file:

```
R> example("mean")
```

Load data sets from packages

From a package that is already loaded:

```
R> data("Prestige")
```

From a package that is installed, but not loaded:

```
R> data("Prestige", package = "euR")
```

List all data sets of an installed package:

```
R> data(package = "euR")
```

View loaded data set objects

In RStudio:

1. In the top right panel, click the *Environment* tab
2. Click on the data set object name

On the command line:

$\longrightarrow$ Type the name of the data set to print it

$\longrightarrow$ Use the `View()` function to open RStudio's viewer

View data sets: command line

⟶ Example: prestige of occupations in Canada

```
R> Prestige
                   education logincome women prestige type
gov.administrators     13.11  13.59234 11.16     68.8 prof
general.managers       12.26  14.65949  4.02     69.1 prof
accountants            12.77  13.17851 15.70     63.4 prof
-------------- (output removed from slides) --------------
longshoremen            8.37  12.21462  0.00     26.1   bc
typesetters            10.00  12.65777 13.58     42.2   bc
bookbinders             8.55  11.82058 70.87     35.2   bc
```

⟶ Too much output even for moderately sized data sets

View first rows of data: `head()`

$\longrightarrow$ Get overview of what the data look like

```
R> head(Prestige)
                    education logincome women prestige type
gov.administrators      13.11  13.59234 11.16     68.8 prof
general.managers        12.26  14.65949  4.02     69.1 prof
accountants             12.77  13.17851 15.70     63.4 prof
purchasing.officers     11.42  13.11390  9.11     56.8 prof
chemists                14.62  13.03669 11.68     73.5 prof
physicists              15.64  13.42915  5.13     77.6 prof
```

View last rows of data: `tail()`

```
R> tail(Prestige)
                education logincome women prestige type
train.engineers      8.49  13.11065  0.00     48.9   bc
bus.drivers          7.58  12.44139  9.47     35.9   bc
taxi.drivers         7.93  12.04439  3.59     25.1   bc
longshoremen         8.37  12.21462  0.00     26.1   bc
typesetters         10.00  12.65777 13.58     42.2   bc
bookbinders          8.55  11.82058 70.87     35.2   bc
```

Summarize data: summary()

⟶ Get overview of the marginal distributions of the variables

```
R> summary(Prestige)
   education        logincome           women
 Min.   : 6.380   Min.   : 9.255   Min.   : 0.000
 1st Qu.: 8.445   1st Qu.:12.003   1st Qu.: 3.592
 Median :10.540   Median :12.534   Median :13.600
 Mean   :10.738   Mean   :12.494   Mean   :28.979
 3rd Qu.:12.648   3rd Qu.:12.999   3rd Qu.:52.203
 Max.   :15.970   Max.   :14.659   Max.   :97.510
    prestige        type
 Min.   :14.80   bc  :44
 1st Qu.:35.23   wc  :23
 Median :43.60   prof:31
 Mean   :46.83   NA's: 4
 3rd Qu.:59.27
 Max.   :87.20
```

Loading .RData files

- $\longrightarrow$ RStudio: Click on the data file in the *Files* tab
- $\longrightarrow$ Command line: Use `load("path/to/file")`

Working directory

R follows a one directory per project philosophy

$\longrightarrow$ Location where R starts looking for files on the file system

In RStudio:

1. *Session* $\rightarrow$ *Set Working Directory* $\rightarrow$ *Choose Directory...*
2. In the dialog, select the desired working directory

You can automate this by using RStudio projects:

$\longrightarrow$ Opening the RStudio project restores the working directory

Working directory

On the command line:

```
R> setwd("../foo/bar")
```

⟶ Better to use relative path (with respect to working directory) than absolute path (e.g., `C:/Users/andreas/foo/bar`)

⟶ Go back to parent directory with `..`

⟶ Always use / as path separator (instead of \ on Windows)

Exercises

Download and open the “Graphics-Exercises.pdf” file from Blackboard, and do Exercises 1.1 and 1.2

Built-in R graphics versus package `ggplot2`

The usual suspect

Function `plot()`:

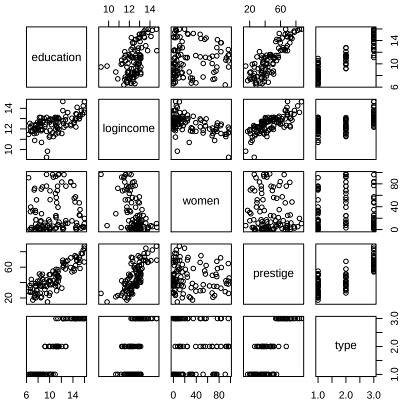
$\longrightarrow$ Scatterplot matrix for data frame

$\longrightarrow$ Works with many other objects, e.g., density estimates, linear models

$\longrightarrow$ Whatever analysis you do, always check if you can `plot()` the result

Scatterplot matrix

```
R> plot(Prestige)
```



Built-in R graphics

+ Allow the user to create quick plots for exploring the data
+ Easy to add elements to an existing plot
+ Fine tuning to produce high-quality graphics for publications

– Designed in the 1970s/80s
– Sometimes inconsistencies in usage or behavior
– Customization via cryptic graphical parameters (see `?par`)

$\longrightarrow$ Murrell (2011): R Graphics

The grammar of graphics

- Designed with recent research on data visualization and human perception in mind
- Focused on coherence between geometry of the data and geometry of the plot

$\longrightarrow$ The visual representation should fit the data

$\longrightarrow$ Always need to explicitly specify what variables to use and how to plot them

$\longrightarrow$ Implemented in package `ggplot2`

$\longrightarrow$ Wickham (2009): `ggplot2`: Elegant Graphics for Data Analysis

Package `ggplot2`

+ Coherent approach to graphics
+ Highly flexible and customizable via options and layers
+ Pretty plots (subjective)

– Steeper learning curve than built-in R graphics
– Often not straightforward to add elements to the plot
– Slow even for moderately sized data sets

Basic usage of `ggplot2`

Add together two basic elements:

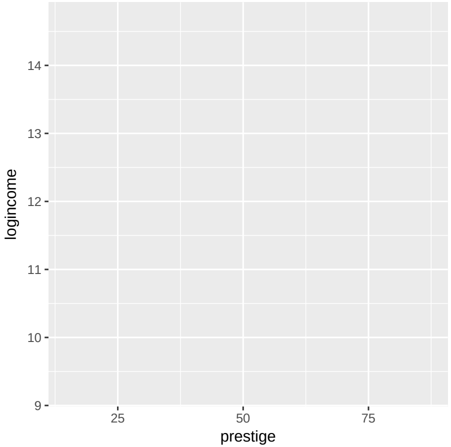
1. Scaffolding defined by `ggplot()`
   - Selects the data set
   - Defines the variables to be used (the aesthetic mapping): function `aes()`
2. Any number of visual representations of the data, known as geoms
   - Define the visual representation (the geometric objects): function family `geom_xxx()`
   - Different elements are added to the plot using the + operator

```
R> library("ggplot2")
```

Basic plots

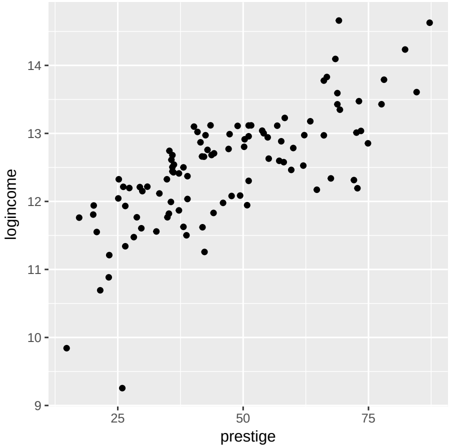
Scatterplot: Scaffolding

```
R> ggplot(Prestige, aes(x = prestige, y = logincome))
```



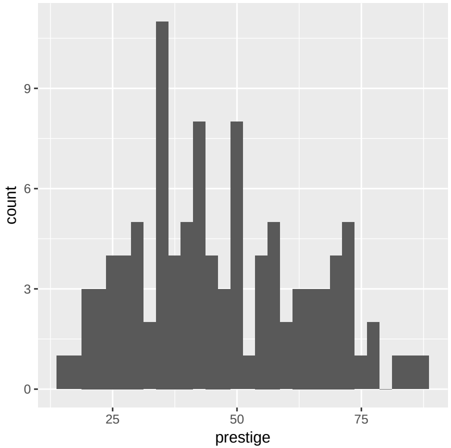
Scatterplot: Scaffolding + points

```
R> ggplot(Prestige, aes(x = prestige, y = logincome)) + geom_point()
```



Histogram

```
R> ggplot(Prestige, aes(x = prestige)) + geom_histogram()
```



Histogram: number of bins

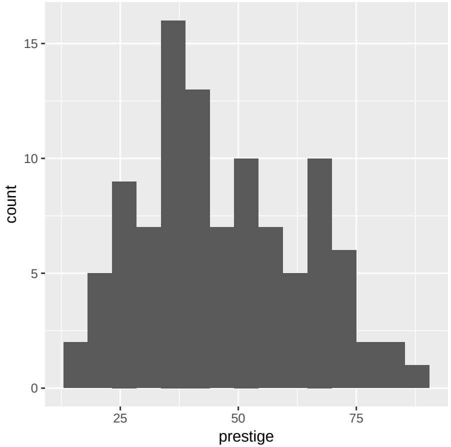
$\longrightarrow$ For histograms, it is always a good idea to play with the number of bins

$\longrightarrow$ Number of bins can be specified with argument `bins`

$\longrightarrow$ Bin width can be specified with argument `binwidth`

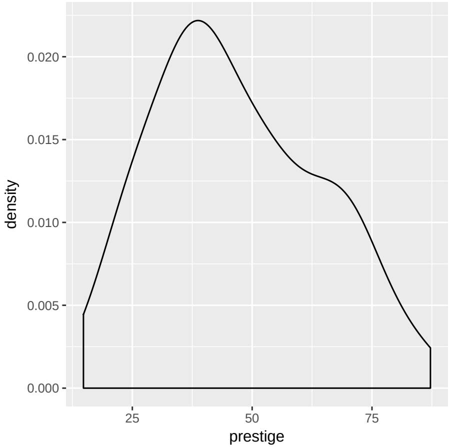
Histogram: number of bins

```
R> ggplot(Prestige, aes(x = prestige)) + geom_histogram(bins = 15)
```



Density plot

```
R> ggplot(Prestige, aes(x = prestige)) + geom_density()
```



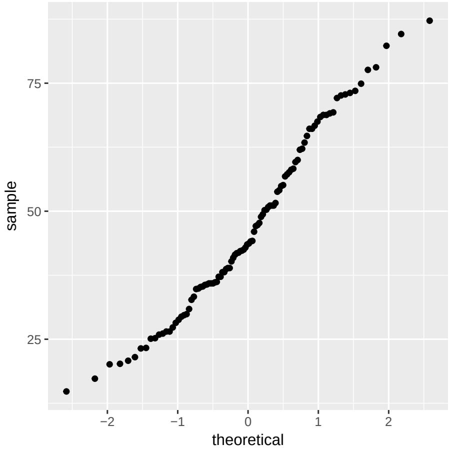
Density plot: kernel and bandwidth

- Density estimate depends on the kernel and smoothing bandwidth
- Default Gaussian kernel is symmetric and therefore not optimal for asymetric distributions

$\longrightarrow$ Still useful to get an insight on the shape of the distribution, but be aware of those issues

Quantile-quantile plot

```
R> ggplot(Prestige, aes(sample = prestige)) + geom_qq()
```



Quantile-quantile plot: straight line?

$\longrightarrow$ Plot sample quantiles against theoretical quantiles

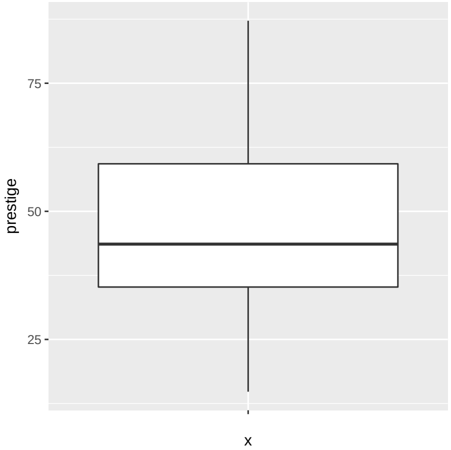
$\longrightarrow$ If the distibutional assumption holds, the points form almost a straight line

$\longrightarrow$ By default the normal distribution is used

$\longrightarrow$ Distribution can be specified with argument `distribution`

Boxplot

```
R> ggplot(Prestige, aes(x = "", y = prestige)) + geom_boxplot()
```



Boxplot statistics

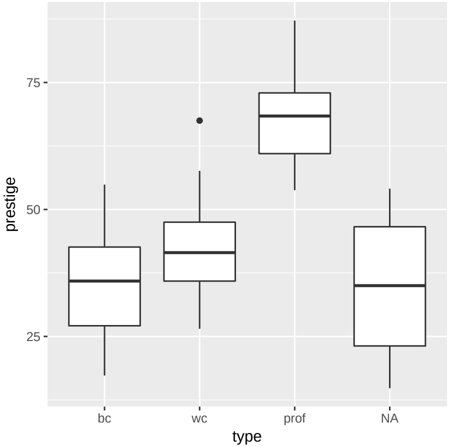
Upper whisker	Largest point still within $1.5 \cdot IQR$ of the upper quartile
Top of box	Upper quartile (i.e., 75% quantile)
Middle line	Median (i.e., 50% quantile)
Bottom of box	Lower quartile (i.e., 25% quantile)
Lower whisker	Smallest point still within $1.5 IQR$ of the lower quartile
IQR	Interquartile range (i.e., difference between upper and lower quartile)

$\longrightarrow$ No assumption about statistical distribution

$\longrightarrow$ But: definition of whiskers assumes some degree of symmetry

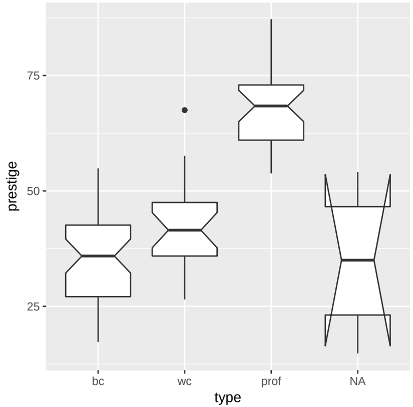
Conditional boxplot

```
R> ggplot(Prestige, aes(x = type, y = prestige)) + geom_boxplot()
```



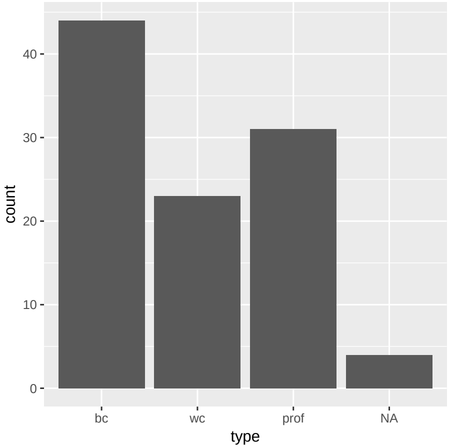
Conditional boxplot: significant differences?

```
R> ggplot(Prestige, aes(x = type, y = prestige)) +
+    geom_boxplot(notch = TRUE)
```



Barplot

```
R> ggplot(Prestige, aes(x = type)) + geom_bar()
```



Time series plot

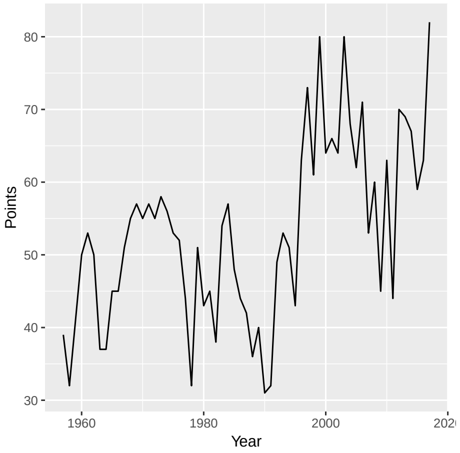
⟶ Simply use `geom_line()` instead of `geom_point()` to draw connected line instead of scattered points

⟶ Example: Eredivisie points of Feyenoord

```
R> data("Feyenoord")
```

Time series plot

```
R> ggplot(Feyenoord, aes(x = Year, y = Points)) + geom_line()
```



Some geoms

For a complete list of geoms, click here. Important ones include:

`geom_point()`	Points
`geom_line()`	Lines / time series
`geom_hline()`	Horizontal lines
`geom_vline()`	Vertical lines
`geom_bar()`	Bars
`geom_boxplot()`	Box and whiskers plot
`geom_density()`	Density estimate
`geom_smooth()`	Fitted regression line
`geom_text()`	Text
`geom_label()`	Text within rectangle
`geom_tile()`	Rectangles for heat maps

$\longrightarrow$ Use appropriate geoms!

Exercises

Load the `Patents` data from the `patents.RData` file, and do Exercise 2.1.

Conclusions

Conclusions

- R/RStudio are open source solutions for statistical analysis with a large community of users
- Command line interface has steeper learning curve, but offers greater flexibility

- Basic function `ggplot()` to initialize the plot
- Function `aes()` to define the variables to be used
- Function family `geom_xxx()` to define the visual representation
- Use scripts for reproducibility of the plots

References

R.I. Kabacoff. **R in Action**. Manning, 2nd edition, 2015.

P. Murrell. **R Graphics**. Chapman & Hall/CRC, 2nd edition, 2011.

H. Wickham. `ggplot2`**: Elegant Graphics for Data Analysis**. Springer-Verlag, 2009.

H. Wickham and G. Grolemund. **R for Data Science**. O'Reilly, 2017.