R getting started - session 1

Alina Ferecatu
Rotterdam School of Management
Erasmus University

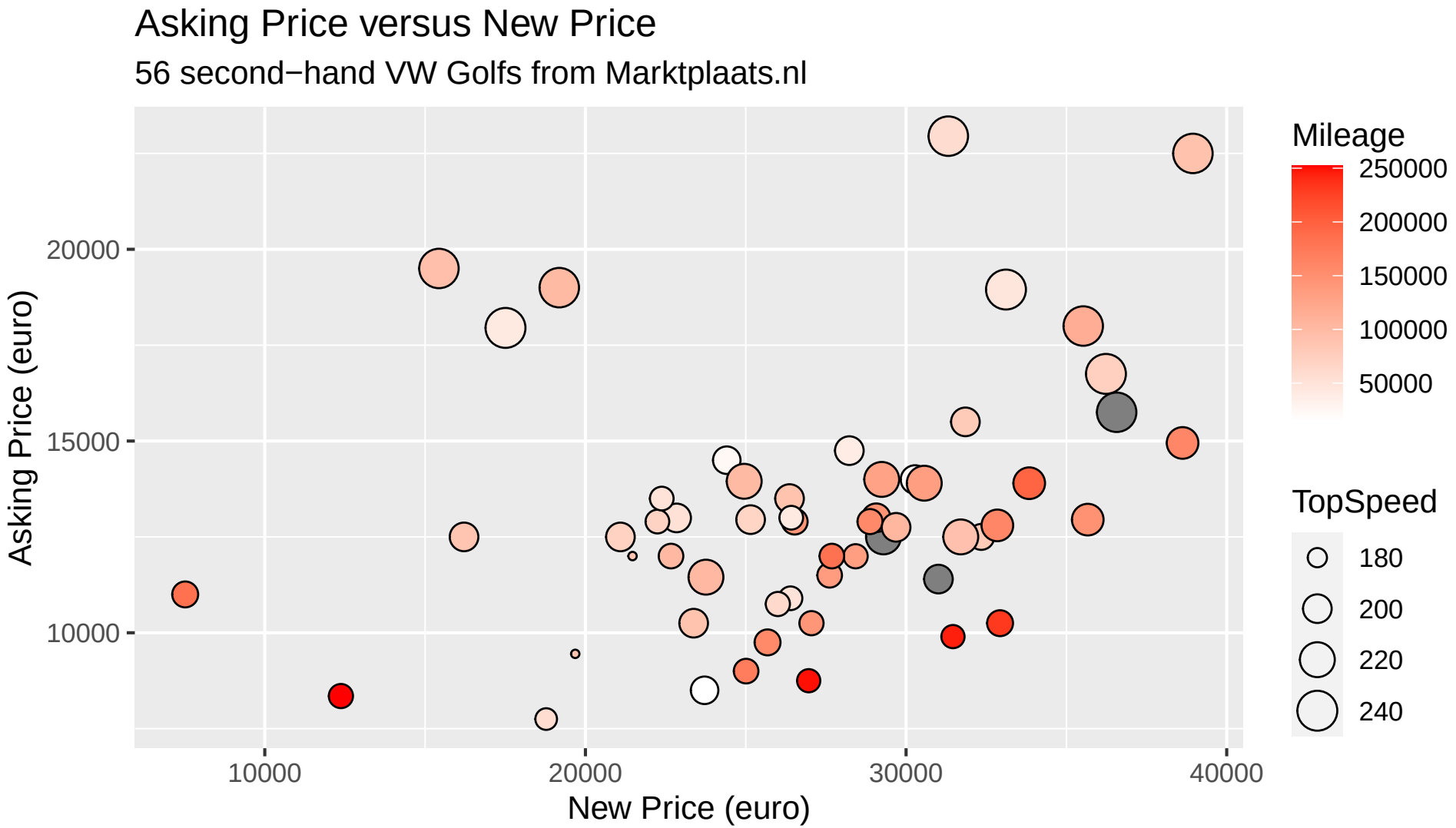
9/9/2020

Documents and software

Have the **latest version** of:

- R: https://CRAN.R-project.org
- RStudio Desktop: https://www.rstudio.com
- Installation instructions on Canvas

Target



−>After session 2 (+ statistics and data manipulations)

Today's lecture

Getting started with R:

- Overview of R ecosystem
- Load data
- Descriptive statistics
- Graphics

Overview of R

About R

- Open source (free) environment for statistical computing
- One of the most popular data science tools worldwide
- Runs on Linux/Unix, Mac OS X, Microsoft Windows
- Fully developed and easy-to-use programming language
- Extensible by community contributed packages
    - 13500+ packages on CRAN, Bioconductor and Github

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 interactive web applications, ...

And more (including everything SPSS can do)

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
- . . .

You too?

RStudio: four panels

- Top left: Script editor (if open)
- 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 filles 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

Example session 1

Please open RStudio, and open example1.R in the script editor

- Available on Canvas
- Execute the line in which your cursor is with Ctrl / Cmd + Enter
- You can also type the command directly in the console, but it is better to store your commands separately in a script (reproducibility)

Script files in RStudio

Create a new script file: File -> New file -> R Script

Save script file:

- Keyboard shortcut: *Ctrl / Cmd + S*
- 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

Some details

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

Install packages

1. 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

2. From command line

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

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:

```
library("tidyverse")
```

–> Install a package **once** on a computer

–> Load it in every new session

Data and descriptive statistics

Loading data

R can read all sorts of data:

- Native R data files: .RData or .rda
- Text files: .txt and .csv
- Excel spreadsheets: .xls(x)
- SPSS files: .sav and .por

. . . and many others

–> We will use the simplest for now: RData format

–> RStudio makes importing other common data types easy: File –> Import Dataset

Load RData files: RStudio

RStudio:

Click the Files tab in the lower right panel, navigate to the R data file and click on it

Or:

File –> Open File . . . and select the R data file in the dialog

–> R objects loaded into your session environment (workspace)

Load RData files: command line

Path relative to the current working directory:

```
load("Prestige.RData")
```

If the file is not in the working directory, specify the full path:

```
load("~/Documents/CMR/session_1/Prestige.RData")
```

Always use / and not \ (Windows!)

Working directory

R follows a one directory per project philosophy

–> Location where R starts looking for files on the file system

In RStudio:

1. Session -> Set Working Directory -> Choose Directory ...
2. In the dialog, select the desired working directory

From the command line:

```
setwd("~/Documents/CMR/session_1")
```

You can automate this by using RStudio projects:

–> Opening the RStudio project restores the working directory

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:

–> Type the name of the data set to print it

–> Use the View() function to open RStudio's viewer

View data sets: command line

Prestige of occupations in Canada in a data frame called Prestige:

–> Too much output even for moderately sized data sets

```
Prestige
```

```
##                       education logincome women prestige type
## gov.administrators        13.11 13.592340 11.16     68.8 prof
## general.managers          12.26 14.659494  4.02     69.1 prof
## accountants               12.77 13.178509 15.70     63.4 prof
## purchasing.officers       11.42 13.113905  9.11     56.8 prof
## chemists                  14.62 13.036689 11.68     73.5 prof
## physicists                15.64 13.429145  5.13     77.6 prof
## biologists                15.09 13.011577 25.65     72.6 prof
## architects                15.44 13.789839  2.69     78.1 prof
## civil.engineers           14.52 13.473833  1.03     73.1 prof
## mining.engineers          14.64 13.428229  0.94     68.8 prof
## surveyors                 12.39 12.526988  1.91     62.0 prof
## draughtsmen               12.30 12.785248  7.83     60.0 prof
## computer.programers       13.83 13.040461 15.33     53.8 prof
```

View first rows of data: head()

–> Get overview of what the data looks like

```
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 the data:

```
tail(Prestige)
```

Summarize data: summary()

```
summary(Prestige)
```

```
##    education       logincome          women           prestige
##  Min.   : 6.380   Min.   : 9.255   Min.   : 0.000   Min.   :14.80
##  1st Qu.: 8.445   1st Qu.:12.003   1st Qu.: 3.592   1st Qu.:35.23
##  Median :10.540   Median :12.534   Median :13.600   Median :43.60
##  Mean   :10.738   Mean   :12.494   Mean   :28.979   Mean   :46.83
##  3rd Qu.:12.648   3rd Qu.:12.999   3rd Qu.:52.203   3rd Qu.:59.27
##  Max.   :15.970   Max.   :14.659   Max.   :97.510   Max.   :87.20
```

Basic Data Frame Functions

Dimensions using dim():

```
dim(Prestige)
```

```
## [1] 102   5
```

Number of rows and columns separately with nrow() and ncol():

```
nrow(Prestige)
```

```
## [1] 102
```

```
ncol(Prestige)
```

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

Column (variable) names:

```
colnames(Prestige)
```

```
## [1] "education" "logincome" "women"     "prestige"  "type"
```

Extracting Variables

One way of extracting variables from a data.frame is via the $ operator:

```
Prestige$education
```

```
##   [1] 13.11 12.26 12.77 11.42 14.62 15.64 15.09 15.44 14.52 14.64
##  [13] 13.83 14.44 14.36 14.21 15.77 14.15 15.22 14.50 15.97 13.62
##  [25] 15.94 14.71 12.46  9.45 13.62 15.21 12.79 11.09 12.71 11.44
##  [37] 11.32 10.64 11.36  9.17 12.09 11.04  9.22 10.07 10.51 11.20
##  [49] 11.00  9.84 11.13 10.05  9.62  9.93 11.60 11.09 11.03  9.47
##  [61]  8.50 10.57  9.46  7.33  7.11  7.58  6.84  8.60  8.88  7.54
##  [73]  7.42  6.69  6.74 10.09  8.81  8.40  7.92  8.43  8.78  8.76
##  [85]  8.10 10.10  6.67  9.05  9.93  8.24  6.92  6.60  7.81  8.33
##  [97]  8.49  7.58  7.93  8.37 10.00  8.55
```

–> Type Prestige$ and RStudio will bring up suggestions

Basic Mathematical Functions

Operator or function	*Operation*
+	addition x + y
-	subtraction x - y
*	multiplication x * y
/	division x / y
^	exponentiation x ^ y
abs()	absolute value abs(x)
sqrt()	square root sqrt(x)
log()	logarithm log(x)
exp()	exponential function exp(x)

Basic Math Example

women as a proportion:

```
Prestige$women / 100
```

```
##   [1] 0.1116 0.0402 0.1570 0.0911 0.1168 0.0513 0.2565 0.0269 0.0
##  [11] 0.0191 0.0783 0.1533 0.5731 0.4828 0.5477 0.0513 0.7710 0.3
##  [21] 0.1959 0.8378 0.4680 0.1056 0.0432 0.0691 0.9612 0.7614 0.8
##  [31] 0.7604 0.2103 0.1115 0.0813 0.9751 0.9597 0.6824 0.9176 0.7
##  [41] 0.8319 0.9286 0.0762 0.5227 0.9614 0.4706 0.5610 0.3917 0.6
##  [51] 0.0316 0.6782 0.0700 0.0369 0.1309 0.2444 0.2388 0.0000 0.0
##  [61] 0.1551 0.0601 0.9653 0.6931 0.3357 0.3008 0.0360 0.2775 0.0
##  [71] 0.1726 0.1726 0.7224 0.3136 0.3948 0.0150 0.0428 0.0230 0.0
##  [81] 0.0578 0.7454 0.0292 0.9067 0.0081 0.0078 0.0000 0.0134 0.0
##  [91] 0.0056 0.0052 0.0246 0.0061 0.0109 0.0058 0.0000 0.0947 0.0
## [101] 0.1358 0.7087
```

Adding a New Variable

Assign a value to a new name, using $ with <-:

```
Prestige$income <- 2 ^ Prestige$logincome
```

Check that it worked:

```
head(Prestige$income)
```

```
## [1] 12351 25879  9271  8865  8403 11030
```

```
head(Prestige)
```

```
##                        education logincome women prestige type incom
## gov.administrators         13.11  13.59234 11.16     68.8 prof  1235
## general.managers           12.26  14.65949  4.02     69.1 prof  2587
## accountants                12.77  13.17851 15.70     63.4 prof   927
## purchasing.officers        11.42  13.11390  9.11     56.8 prof   886
## chemists                   14.62  13.03669 11.68     73.5 prof   840
## physicists                 15.64  13.42915  5.13     77.6 prof  1103
```

Basic Data Analysis Functions: Vectors

Function	**Operation**
length(x)	Length
min(x)	Minimum
max(x)	Maximum
sum(x)	Summation (total)
range(x)	Minimum and maximum
quantile(x)	Quantiles
mean(x)	Mean
var(x)	Variance
sd(x)	Standard deviation
median(x)	Median
table(x)	Contingency table
table(x, y)	Cross-tabulation
cor(x, y)	Correlation
cov(x, y)	Covariance

Example

Minimum and maximum percentage of women:

```
min(Prestige$women)
```

```
## [1] 0
```

```
max(Prestige$women)
```

```
## [1] 97.51
```

```
range(Prestige$women)
```

```
## [1]  0.00 97.51
```

Mean, median and standard deviation of logincome:

```
mean(Prestige$logincome)
```

```
## [1] 12.49447
```

```
median(Prestige$logincome)
```

```
## [1] 12.53392
```

Functions

Frequency (contingency) table of type:

```
table(Prestige$type)
```

```
##
##   bc   wc prof
##   44   23   31
```

Missing values

You have to specify how R must handle missings (NA):

–> Depends on function; usually using the na.rm argument

```
table(Prestige$type, useNA = "always")
```

```
##
##    bc    wc prof <NA>
##    44    23    31    4
```

–> See the help page, for example, ?table and ?mean

–> Default behaviour depends on function

Pearson correlation between women and prestige:

```
cor(Prestige$women, Prestige$prestige)
```

```
## [1] -0.1183342
```

Recap: Special Values

NA	Not available (represents missing value)
NaN	Not a number (usually division 0/0)
Inf	Positive infinity
-Inf	Negative infinity
NULL	Represents undefined value
pi	

Practice

Consider the patents data in the file patents.RData

Information on patents granted in 2012 in each US state

Variables are described on the next slide

The data were scraped from StatsAmerica and Wikipedia

Variables

total The total number of granted patents.

utility The number of granted utility patents.

design The number of granted design patents.

plant The number of granted plant patents.

population The number of inhabitants.

area Land area in km2.

governor Party afiliation of the state governor.

area Land area divided into three categories: “small”, “medium” and “large”.

density The population density.

densitycat Population density divided into two categories: “low” and “high”.

logdensity Logarithm of population density.

logutility log(utility+1).

logdesign log(design+1).

logplant log(plant+1).

Exercises 1: Questions

1. What is the minimum and maximum number of total patents granted per state?
2. What is the Pearson correlation between logtotal and logdensity?
3. How many Republican governors were there in 2012? And how many Democrats? Use the governor variable.
4. How many Republican governors were there in small states? How many Democratic governors were there in large states? Use the areacat variable.
5. What is the mean proportion of the total patents per state that consist of utility patents? And the median and the standard deviation?

Basic plotting with package ggplot2

Graphics systems in R

R has several plotting systems and packages, including:

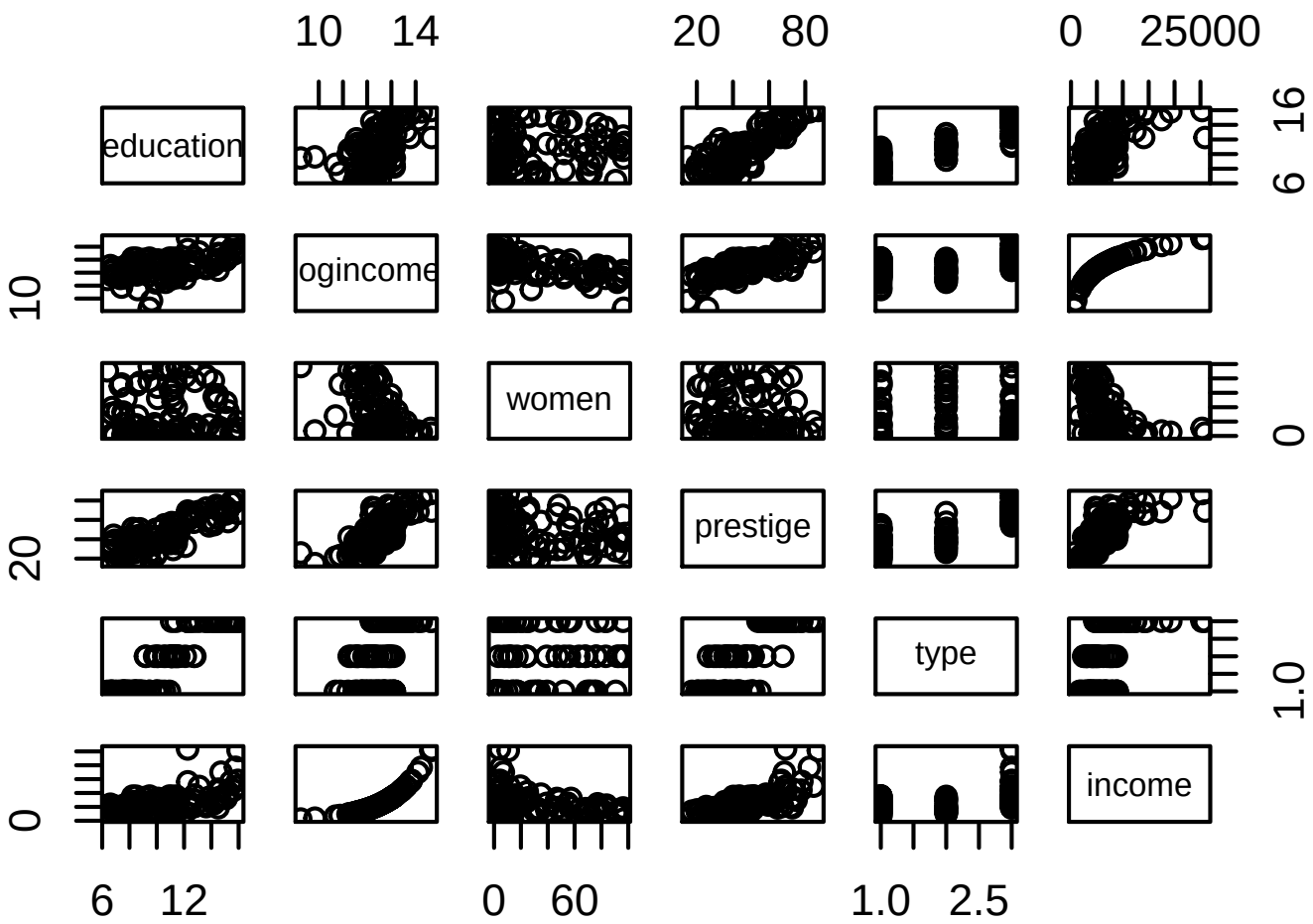
- Base graphics (e.g., *plot()*) is older but still very good
- Package ggplot2 delivers modern, cutting-edge capabilities:

–> We focus on ggplot2 graphics

–> Whatever analysis you do, always check if you can *plot()* the result

Scatterplot matrix

```
plot(Prestige)
```



The grammar of graphics

- Implemented in package ggplot2
- 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
- The visual representation should fit the data

–> Must explicitly specify what variables to use and how to plot them

Ref: Wickham (2009): ggplot2: Elegant Graphics for Data Analysis

Basic usage of ggplot2

Add together two basic elements:

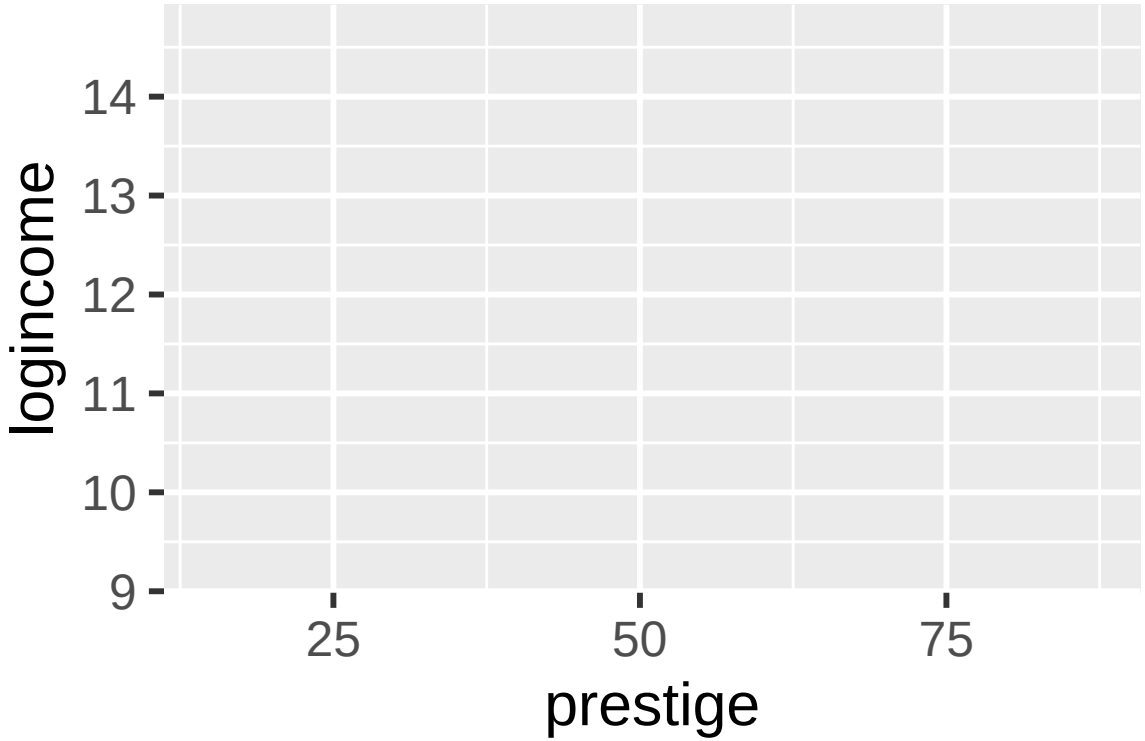
1. Scaffolding deffned by ggplot()
   - Selects the data set
   - Defines the variables to be used (the aesthetic mapping): function aes()
2. Any number of visual representations of the data, known as geoms
   - Define the visual representation (the geometric objects): function family geom_x()
   - Different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

Load the package:

```
library("ggplot2")
```

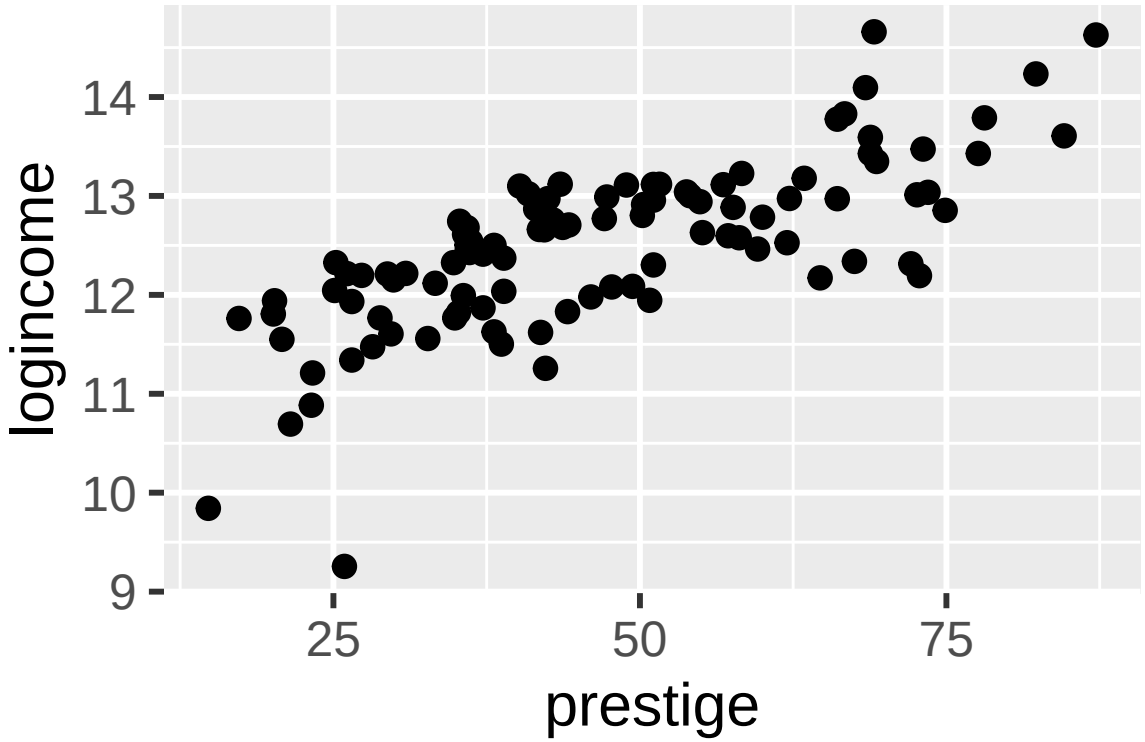
Scatterplot: Scaffolding

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



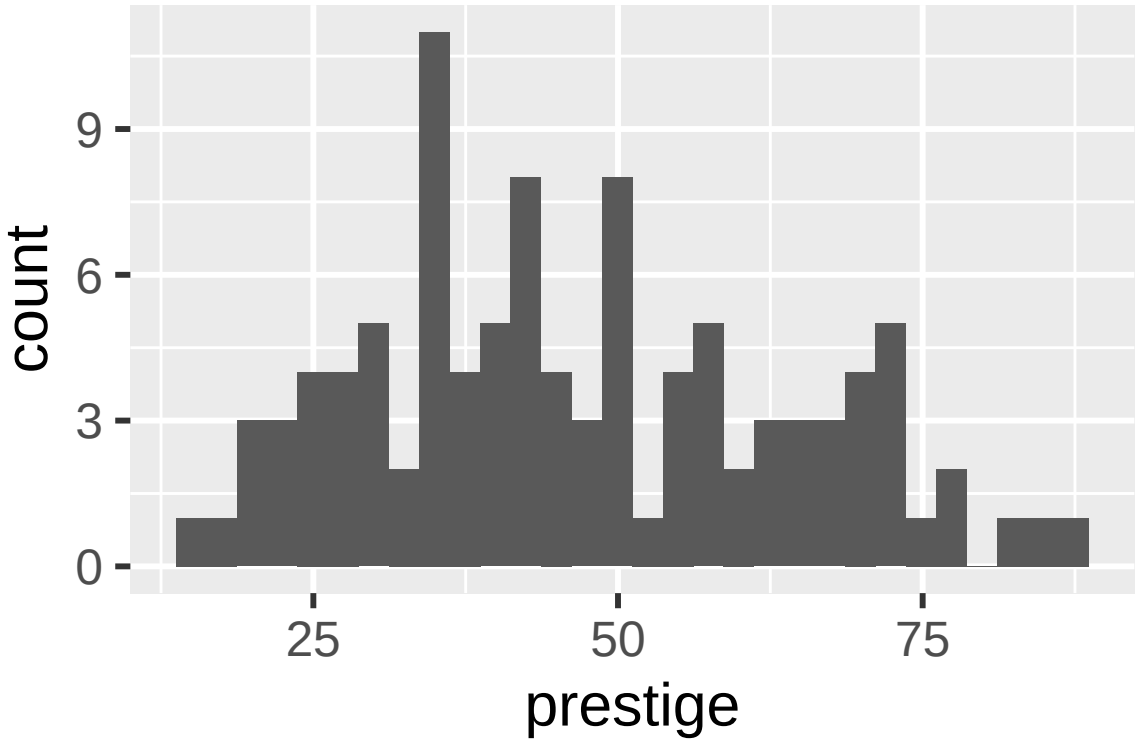
Scatterplot: Scaffolding + points

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



Histogram

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



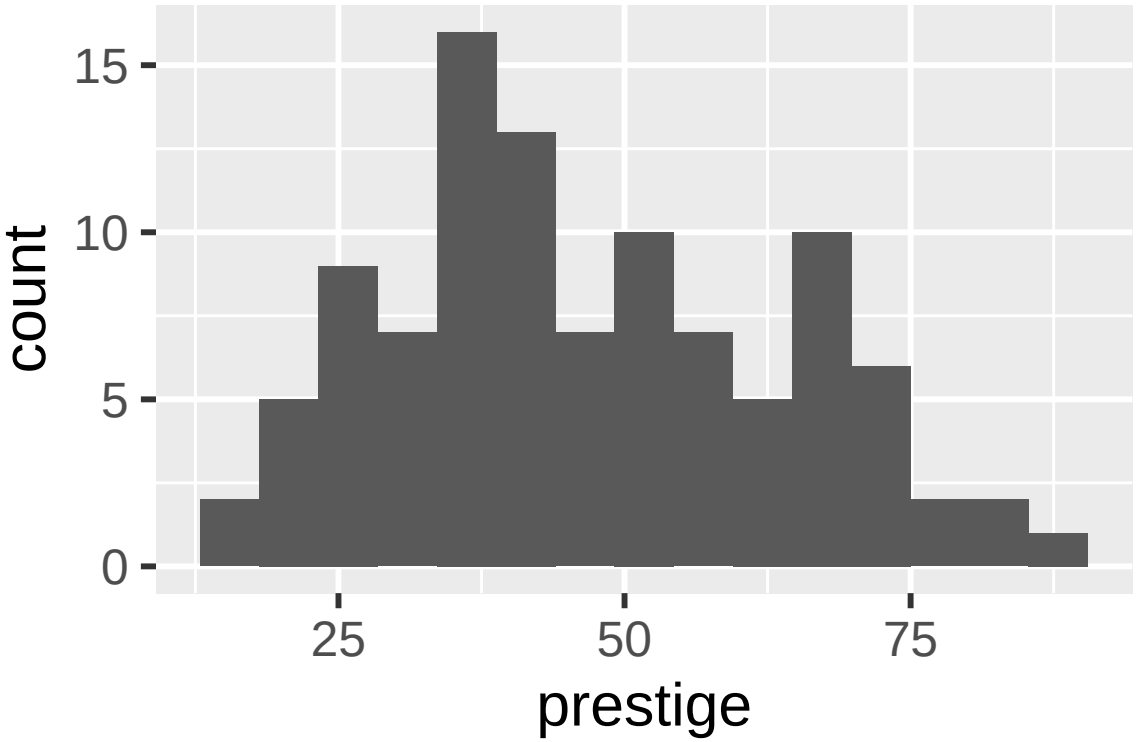
Histogram: number of bins

–> For histograms, it is always a good idea to play with the number of bins

- Number of bins can be specified with argument bins
- Bin width can be specified with argument binwidth

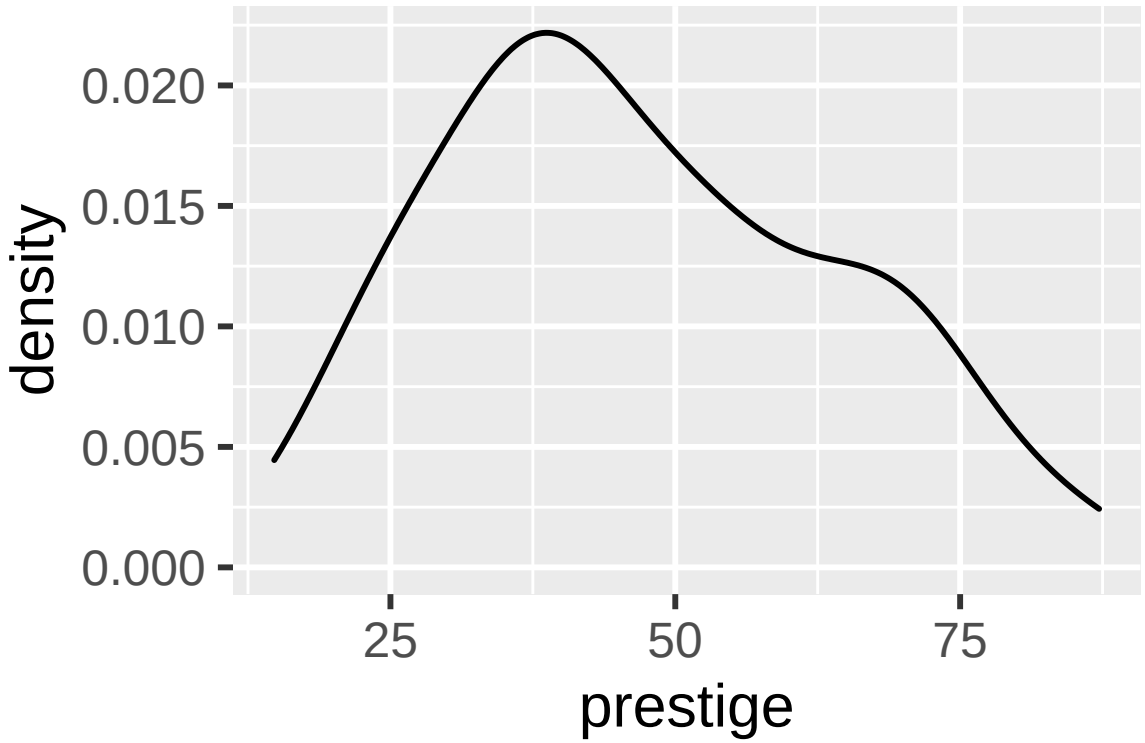
Histogram: number of bins

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



Density plot

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



Density plot: kernel and bandwidth

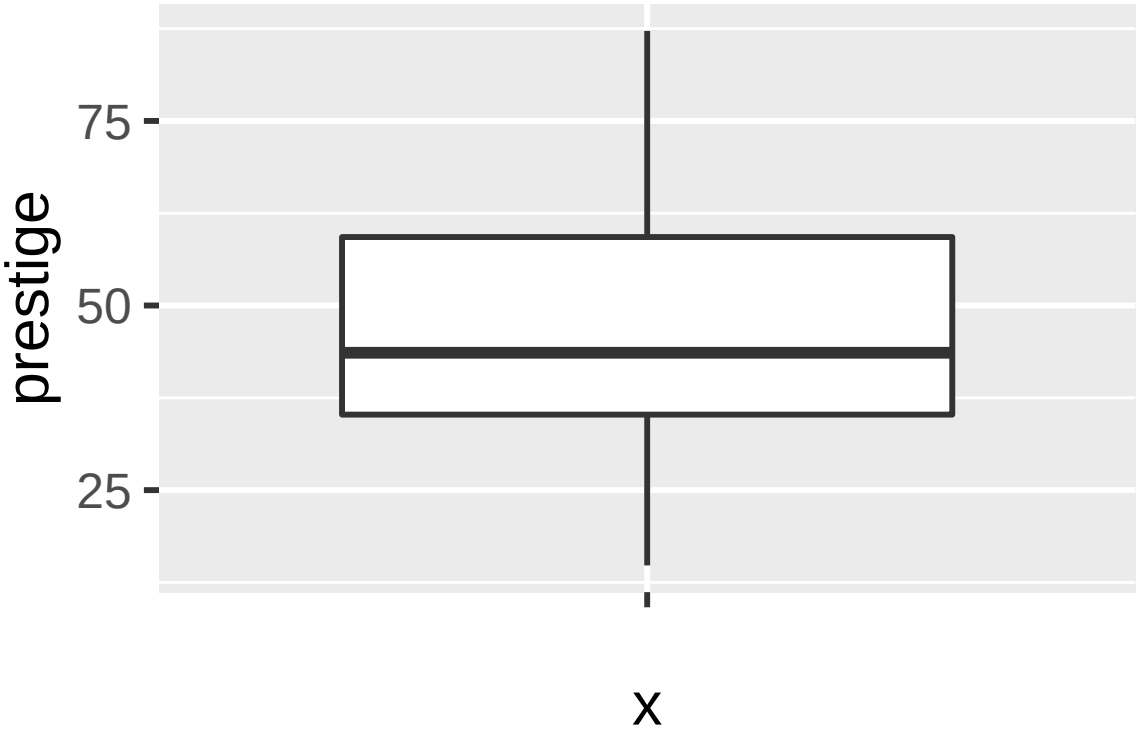
Density estimate depends on the kernel and smoothing bandwidth

Default Gaussian kernel is symmetric and therefore not optimal for asymetric distributions

–> Still useful to get an insight on the shape of the distribution, but be aware of those issues

Boxplot

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



Boxplot statistics

Upper whisker: Largest point still within 1.5IQR 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 1.5IQR of the lower quartile

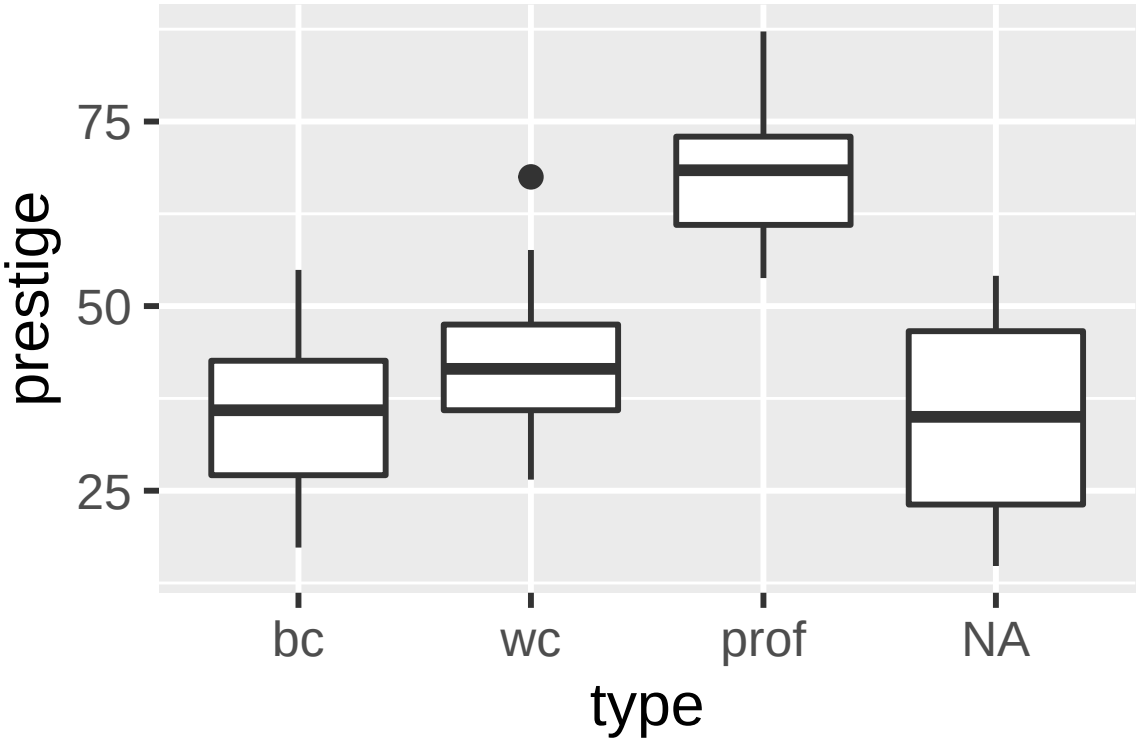
IQR: Interquartile range (i.e., difference between upper and lower quartile)

–> No assumption about statistical distribution

–> But: definition of whiskers assumes some degree of symmetry

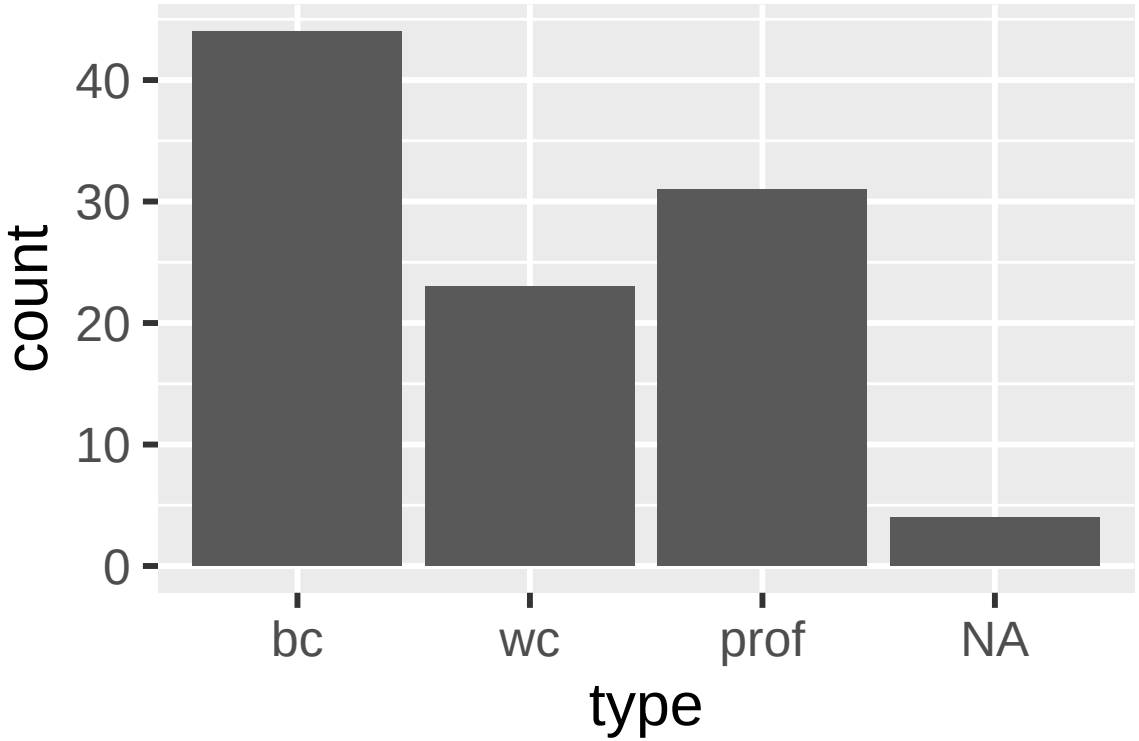
Conditional boxplot

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



Barplot

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



Time series plot

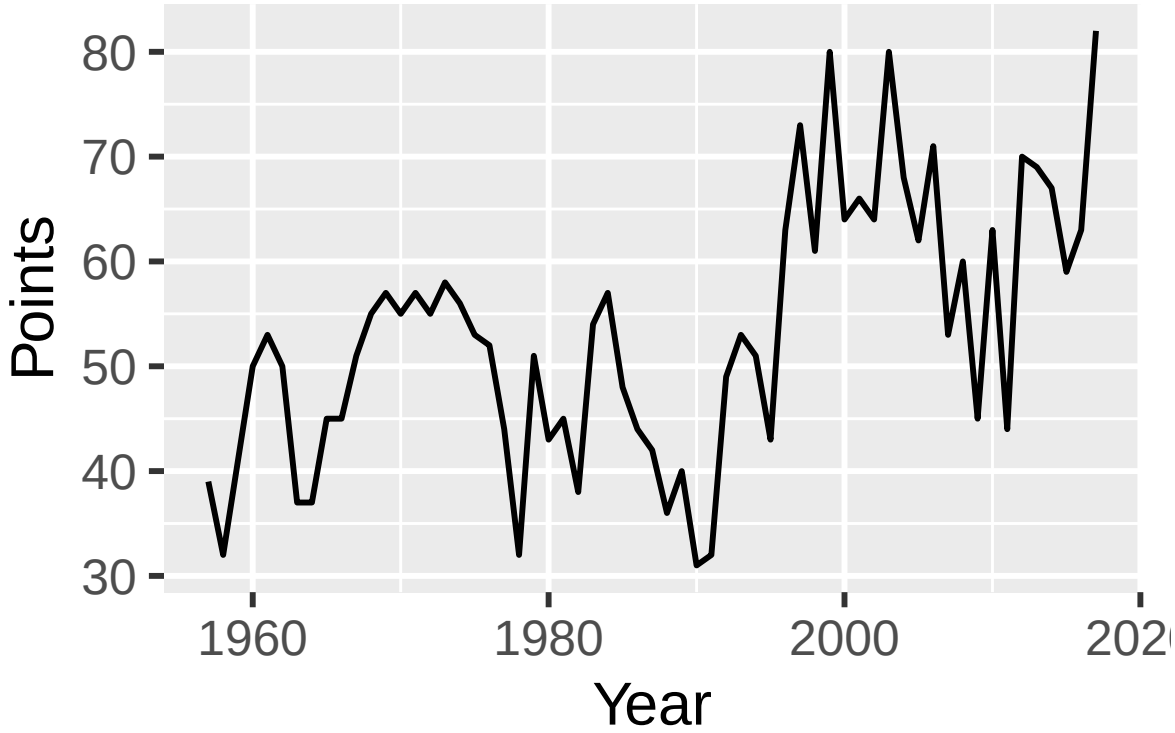
Simply use geom line() instead of geom point() to draw connected line instead of scattered points

–> Example: Eredivisie points of Feyenoord

```
load("~/Documents/CMR/session_1/Feyenoord.RData")
```

Time series plot

```
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_[h/v]line(): Horizontal or vertical line

geom_bar() Bars

geom_boxplot() Box and whiskers plot

geom_density() Density estimate

geom_smooth() Fitted regression line

geom_text/label Text

geom_tile() Rectangles for heat maps

–> Use appropriate geoms!

Exercise 2

Use again the patents data:

1. Plot total vs density, then plot logtotal vs logdensity. What do you observe?
2. Produce histograms of total and logtotal to compare the distribution of the number of granted patents before and after the log-transformation. Play with the number of bins to get a more complete picture of the distributions.
3. Produce a boxplot of logdensity. Do you find any outliers?
4. Produce conditional boxplots of total and logtotal with observations grouped by population density category (densitycat).
5. Produce barplots of the factors governor and areacat.

Acknowledgements

Thank you Pieter Schoonees and Andreas Alfons

for contributing to these materials!

R extra-credit assignment and hackathon

R extra-credit assignment

Create a new R script (that is, a .R file) in RStudio.

Replicate all analyses posted for the CMR sessions 2, 3, 4, and 5, in the *.html* files.

It is important that your R script contains no errors and is easy to read.

Remember to include code that loads the required packages.

Include comments (lines that start with a #) in this text file so that it is clear which code replicates which analysis.

Save your work under a file name containing Yourname_studentNumber_Rassignment.R

Remember to resave regularly.

Submit the file (one file for all sessions) on Canvas, under *R extra-credit* assignment

Deadline: Oct 8 2020 at 1PM, before the workshop

R hackathon

Did you spot an error in the code running analyses for the CMR sessions 2, 3, 4, and 5?

–> Add a comment startig with #ERROR before the analysis

–> Propose a correction, and be very specific for which analysis you are proposing the correction

Do you have a more elegant solution to replicate an analysis?

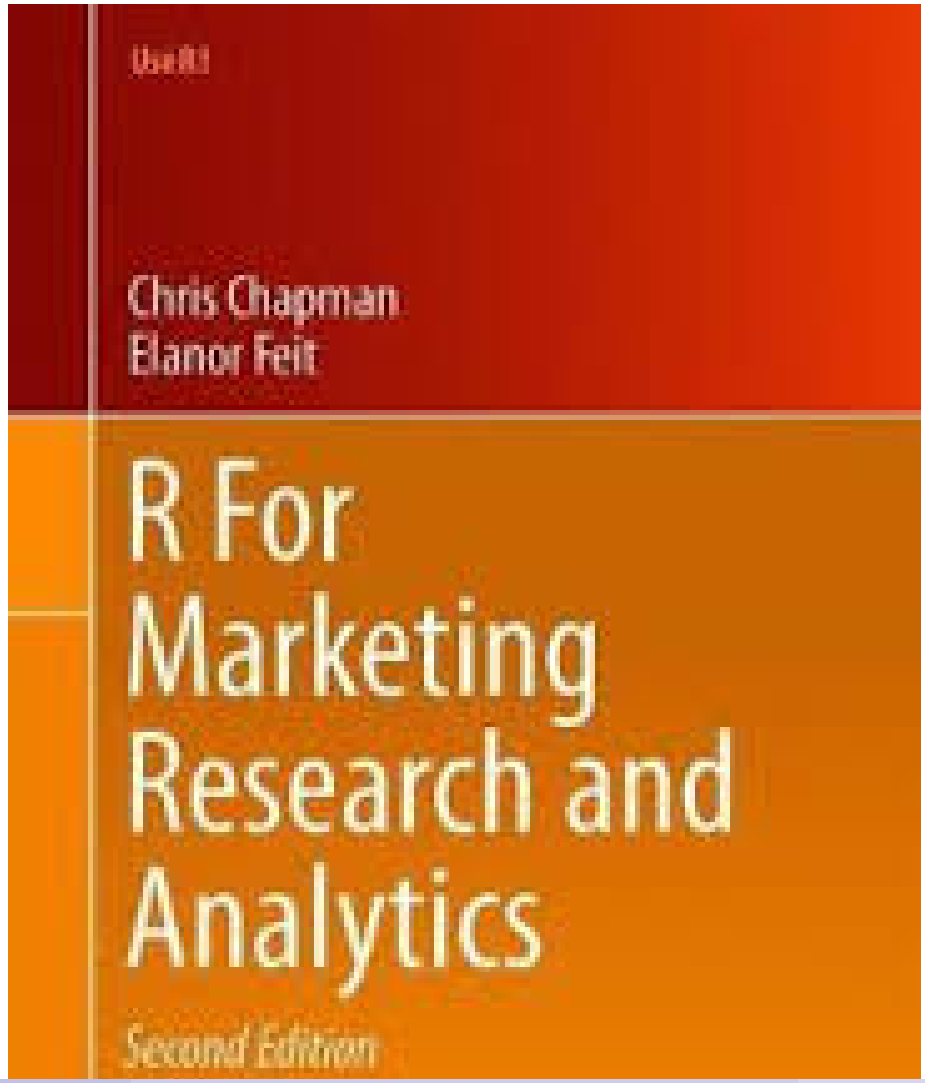
–> Add a comment startig with #SUGGESTION before the analysis

–> Suggest new code for the analysis, and be very specific for which analysis you are making the suggestion

To participate, add your name in the CMR sign-up document, on the sheet named *R hackathon*

R hackathon winner and prise

The student reporting most errors and suggestions (#ERROR+#SUGGESTION) wins the R hackathon.



R help

R help

R comes with built-in help for more information on functionality:

- Help topic is usually the name of a function, data set or package
- Help files are required for packages on CRAN
- Overview of all help files within a package is available

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: command line

Help for the help() function is available:

```
?help
help("help")
```

List all help topics within a package:

```
help(package = "ggplot2")
help("help")
```

Run examples from a help file:

```
example("mean")
```

```
##
## mean> x <- c(0:10, 50)
##
## mean> xm <- mean(x)
##
## mean> c(xm, mean(x, trim = 0.10))
## [1] 8.75 5.50
```