Quarto workshop VVSOR

Tiny van Boekel

Em. Prof. Food Science, Wageningen University

WAGENINGEN
UNIVERSITY & RESEARCH

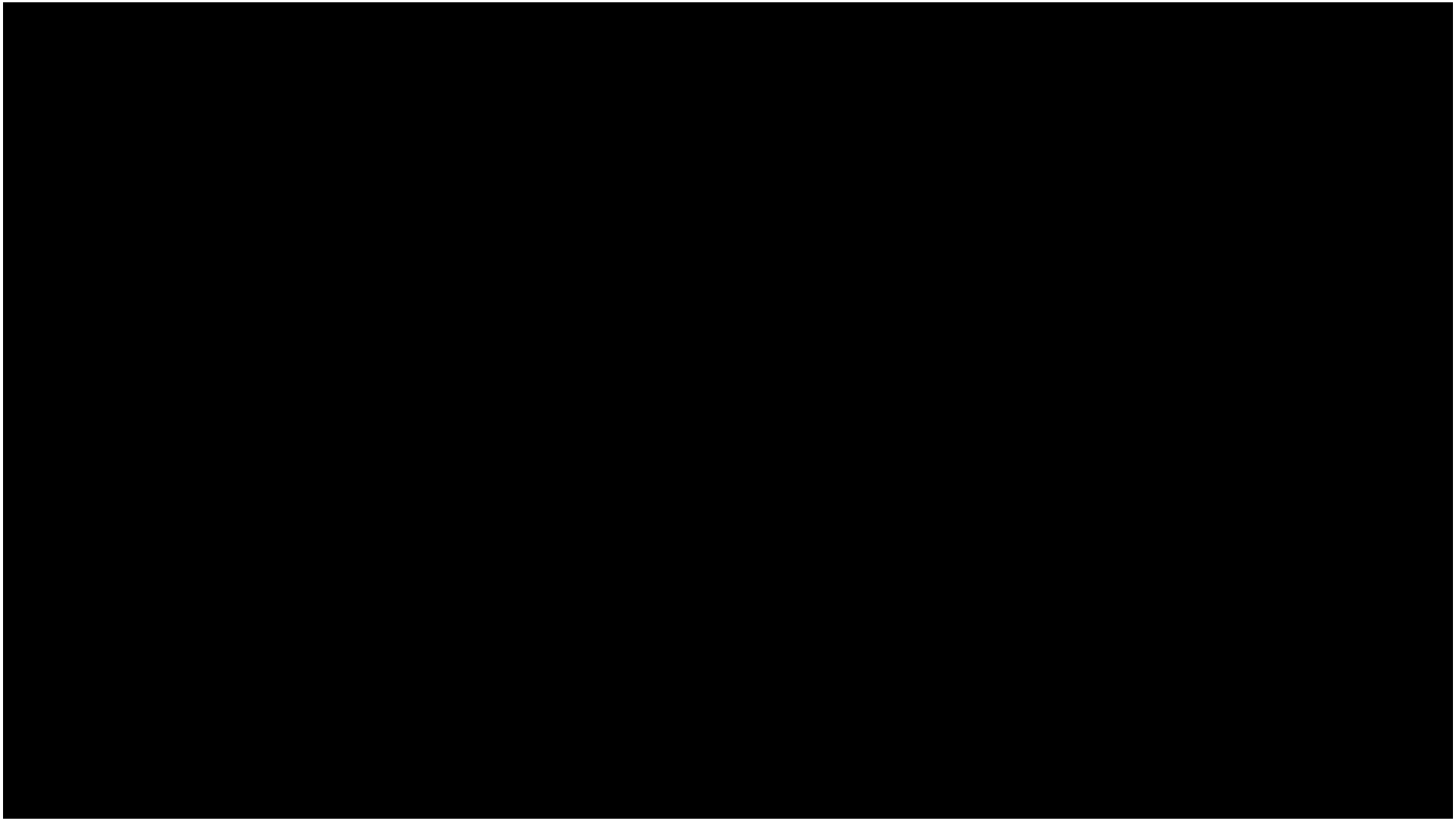
100years
1918 — 2018

Outline

- Getting started
  - Suggested workflow
- Scientific reports/papers
  - Markup language
  - R code for data wrangling and analysis
  - Tables
  - Figures
  - References
  - Possible output formats
- Exercises

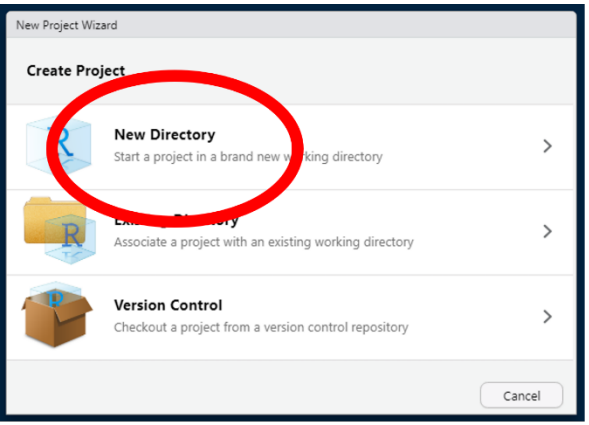
Why Quarto?

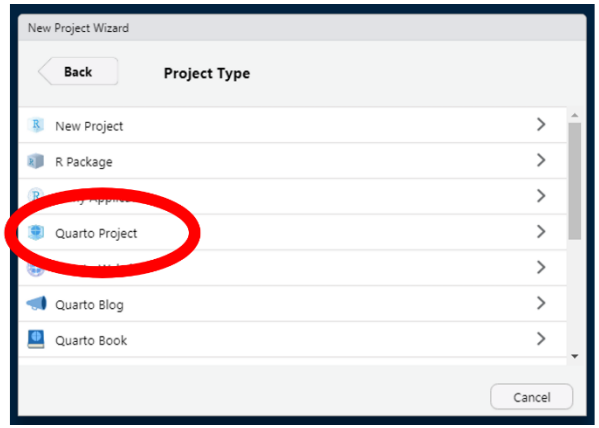
- Reproducible workflow!
- No more copying and pasting from various software programs
- No more errors resulting from copying and pasting

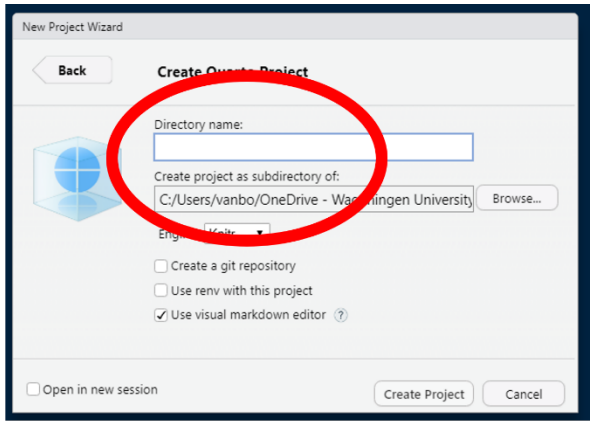


Suggested steps in your workflow -1

- Start with a Quarto project in a new or existing directory
  - do this always for every new topic!
  - "one folder – one project mentality", easy to do in RStudio

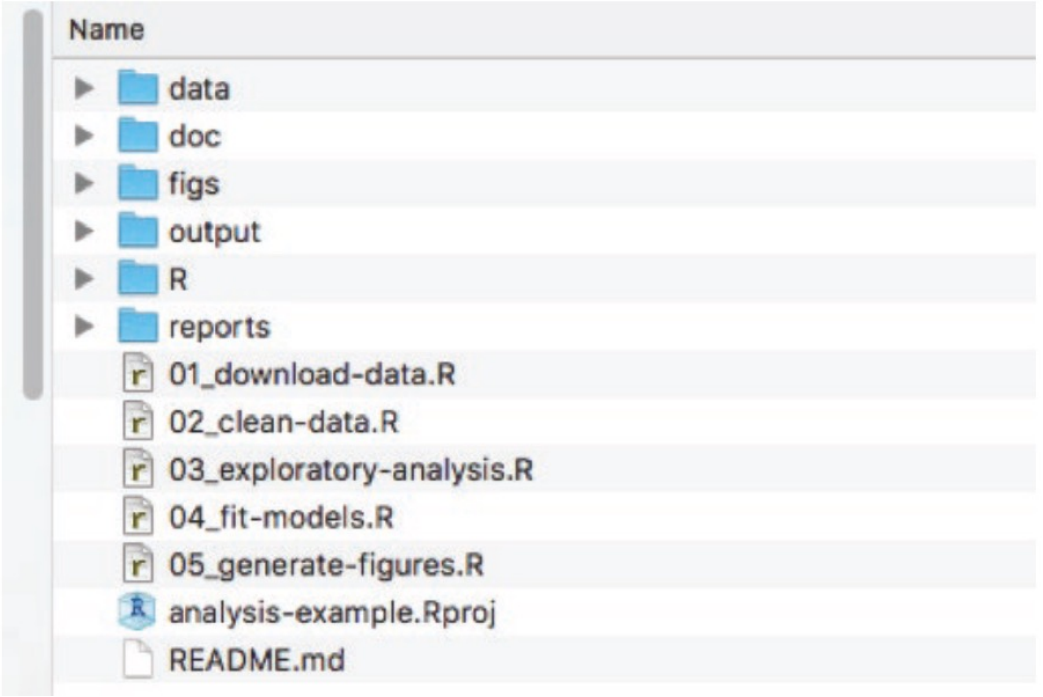






Workflow with subdirectories

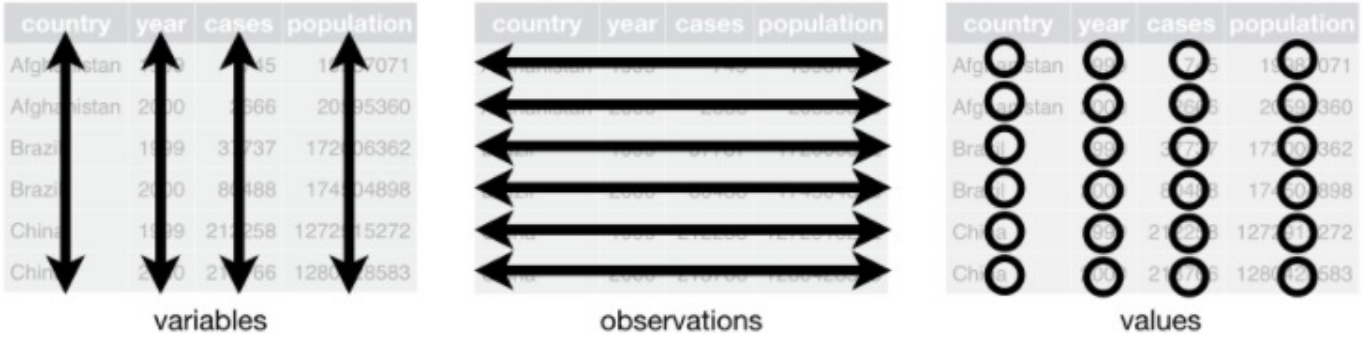
- Organise project directory into subdirectories, e.g.:
  - Data
  - Scripts
  - Figures
  - Images
  - References
  - ….



- Give readable names to your files!

Suggested steps in your workflow -2

- Organize your data in a tidy way:
  - One column for each variable
  - One row for each observation
  - Values (numeric, characters, dates,…) in cells



  - Save data in csv files
  - Import data in Quarto to process them but leave the raw data in a safe place and do not change them
- See also the article on data organisation in spreadsheets!

Suggested steps in your workflow -3

- Use Quarto to
  - write text
  - import data
  - process the data and do calculations
  - make tables and graphs
  - insert citations via a bibliographic reference manager (Mendeley, Endnote, Zotero,…)
  - wrap up in a report, article, thesis, slides, …

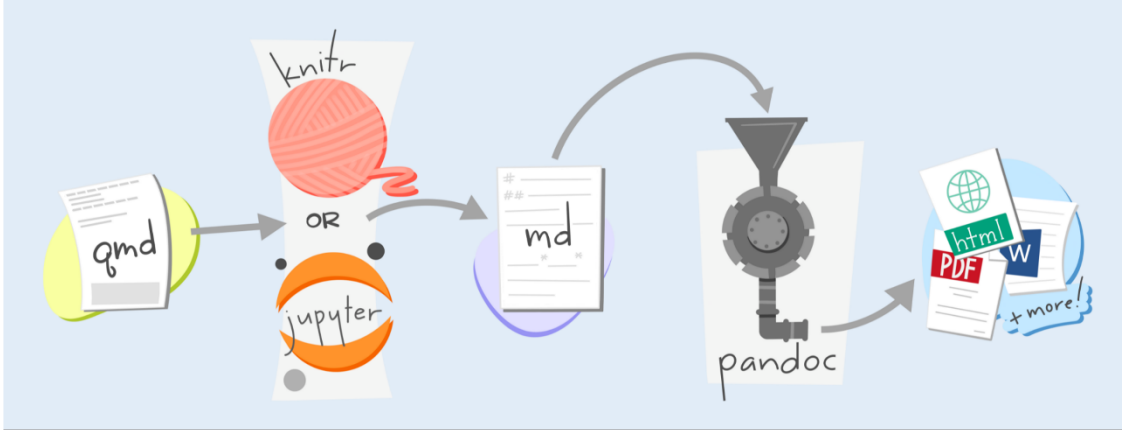
Suggested steps in your workflow -4

Learn how to use Git and GitHub:

- *Will not be discussed in this workshop but is highly recommended to learn by yourself*
- RStudio can be linked to Git and GitHub very easily
- Excellent source on Git and GitHub is the freely available e-book by Jenny Bryan: https://happygitwithr.com/
- Also very instructive is a series of blogs by Page Piccinini: https://datascienceplus.com/r-for-publication-by-page-piccinini-lesson-0-introduction-and-set-up/

Quarto: R code and markup mixed

- Tool to write documents, presentations or webpages that combines written text with R code (or Python, …)
- Markup language mixed with R code
  - R code is evaluated and output is pasted in the text
- Quarto is built upon pandoc

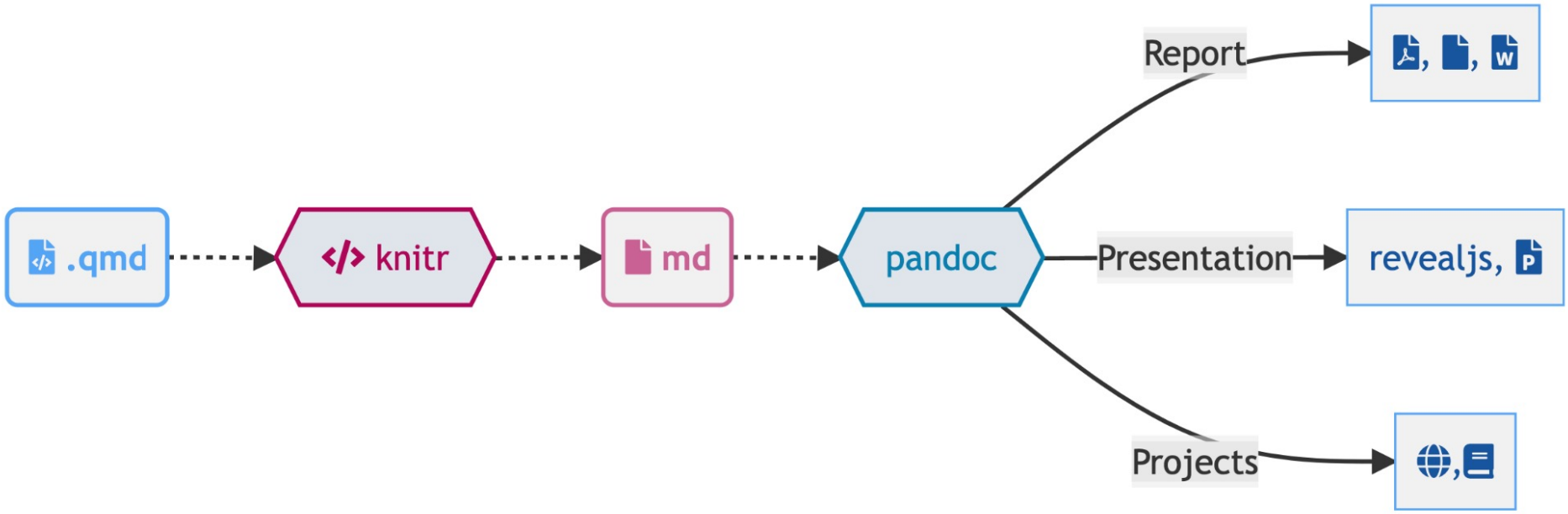


Source: mine çetinkaya-rundel, RStudio

What is Markup language?

- plain text file
- markup is added as extra code in between text
- similar principle as HTML or LaTex but easier syntax
  - more human readable
- The mix of R code and markup language is converted by knitr to markdown (.md)
- pandoc converts .md files into:
  - pdf, Word, html, …

From Quarto to output via Markdown



Source: Tom Mock, RStudio

Anatomy of a Quarto document

- Metadata (YAML)

```
---
format: html
---
```

- Code

````
```{r}
#| eval: true
library(dplyr)
mtcars %>%
  group_by(cyl) %>%
  summarize(mean = mean(mpg), .groups = "drop")
```
````

```
# A tibble: 3 x 2
    cyl  mean
  <dbl> <dbl>
1     4  26.7
2     6  19.7
3     8  15.1
```

- Text

```
# Heading 1
This is a sentence with some **bold text**, some *italic text* and an [image](image.png).
```

YAML header

- A recursive acronym for "YAML Ain't Markup Language"

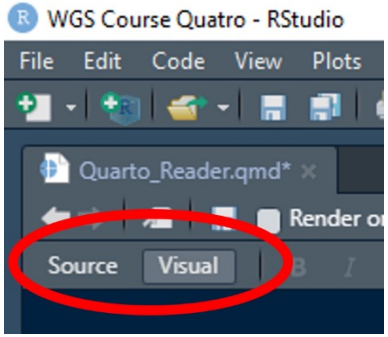
- Controls how the document is generated but does not control the content
    - Sets options for
        - Title/data/author
        - Font
        - Figure options
        - Controls type of output (word, pdf, html)
        - TOC (table of contents)
        - ...

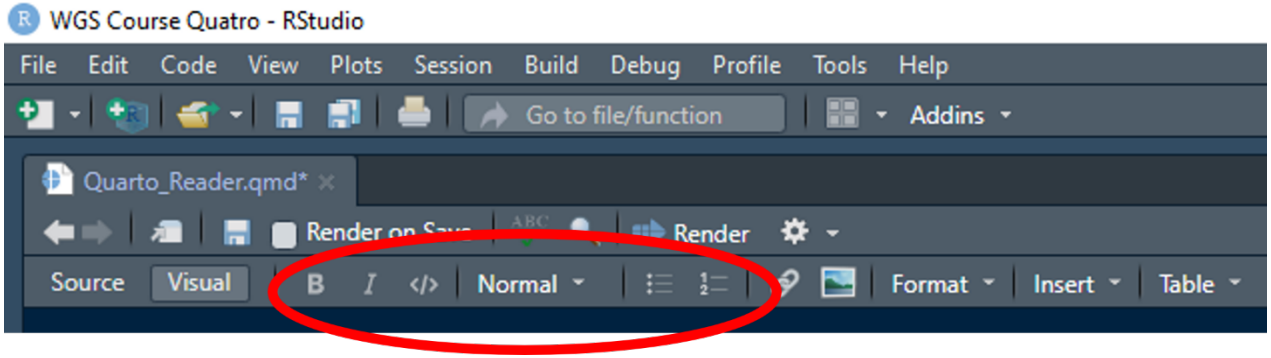
Adding text

- Quarto can be used as a word processor
  - Plain text with possibilities to use italics, bold, links, different fonts, unicode characters….
  - Spelling checker
  - Markdown syntax can be used to format text
  - But: RStudio makes life easier with WYSIWYG

Markdown

- With Quarto and Rstudio in Visual mode it is also possible by clicking on relevant buttons





syntax

```
Plain text
End a line with two spaces to start a new paragraph.
*italics* and _italics_
**bold** and __bold__
superscript^2^
~~strikethrough~~
[link](www.rstudio.com)

# Header 1

## Header 2

### Header 3

#### Header 4

##### Header 5

###### Header 6
```

becomes

Plain text
End a line with two spaces to start a new paragraph.
italics and *italics*
bold and **bold**
superscript^2
~~strikethrough~~
link

Header 1

Header 2

Header 3

Header 4

Header 5

Header 6

syntax

```
endash: --
emdash: ---
ellipsis: ...
inline equation: $A = \pi*r^{2}$
image: ![](path/to/smallorb.png)

horizontal rule (or slide break):

***

> block quote

* unordered list
* item 2
    + sub-item 1
    + sub-item 2

1. ordered list
2. item 2
    + sub-item 1
    + sub-item 2

Table Header  | Second Header
------------- | -------------
Table Cell    | Cell 2
Cell 3        | Cell 4
```

becomes

endash: –

emdash: —

ellipsis: …

inline equation: $A = \pi * r^2$

image: 

horizontal rule (or slide break):

> block quote

- unordered list
- item 2
  - sub-item 1
  - sub-item 2

1. ordered list
2. item 2
   - sub-item 1
   - sub-item 2

Table Header	Second Header
Table Cell	Cell 2
Cell 3	Cell 4

Greek and math symbols: LaTex style

Syntax	becomes
β	β
ξ	ξ
Σ	Σ
log	log
$\log$	$\log$
$\sqrt$	$\sqrt{}$
∞	∞
y^2	y^2
y_i	y_i
y^{a+b}	y^{a+b}

Exercise 1

- Let's start to write an exercise report
- The report 'penguin_paper.pdf' is what the end result could look like, starting from scratch (but you can take a look at the pdf to see what the end result will be)
- Start with **exercise 1**:
    - Start defining a project in which you are going to build your own report
    - Modify the YAML as indicated in the reader
    - Save the project

Exercise 2: text and formatting

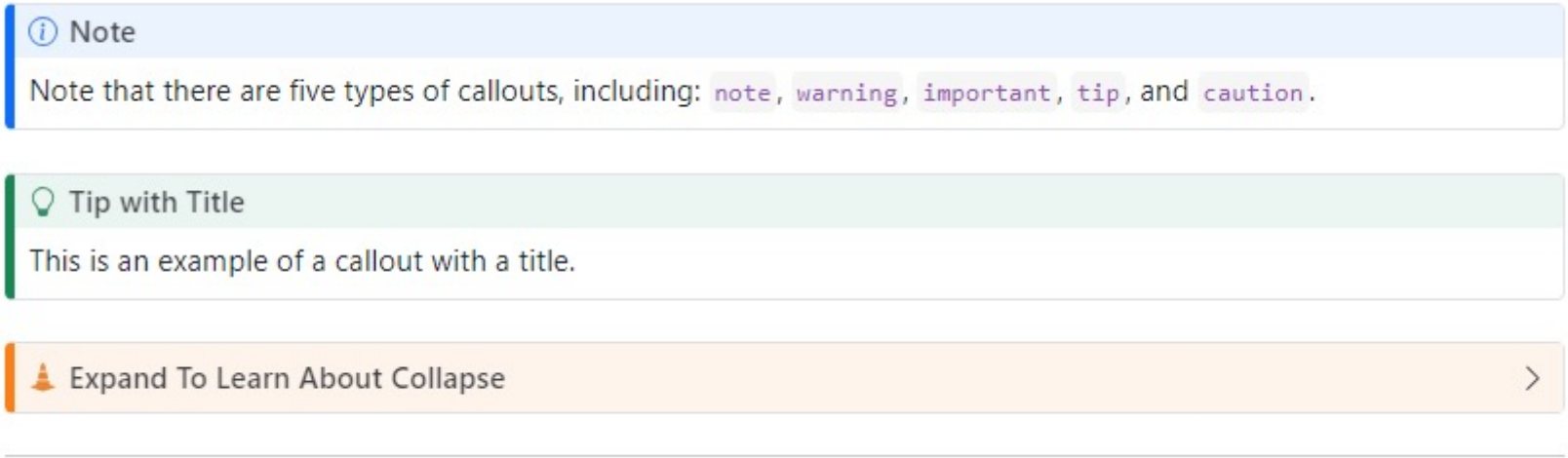
- Continue with **exercise 2a**:
  - Start with some text, see example in reader for exercise 2a
  - use the various formatting options (lists, italics, bold, footnotes, …)
  - render the file and study the output
  - start with html, if output is OK, switch to pdf if that is what you want (*in that case: make sure that tinytex is installed*)

Pandoc Divs and Spans

- Can be used to apply identifiers and styles to a block of a document for special effects
  - to produce coloured words or lines
  - to change font size
  - ….
- Define classes: css (cascading style sheet) files produced as txt files (RStudio -> New File -> Text file)
- Store with extension .css
- Call the .css file from the YAML
- Write a fenced Div in the text
- **Exercise 2b**: follow the instructions in the reader

Callouts (fenced Dics)

- Callouts are ready-to-use fenced Divs (5 types)



- **Exercise 2c**: produce some callouts

Spans

- Spans are used for special effects in in-line text
- Highlight the word or line with [] followed by the desired style in {}
    - This word is in [red]{style="color:red"}
- Spans in visual mode, select text and then: Format -> Span…

- **Exercise 2d**: use Span on a word or line

Text in columns

- It is possible to use columns for your text. For instance, suppose you want your text in 2 columns. Fenced Divs are used:

:::: columns

::: {.column}

Your text here for the first column

::: {.column}

Your text here for the second column

:::

::::

- **Exercise 2e**

Tabsets

- You can organize your text in different clickable tabs *(only for format html!)*
- This is, again, achieved using Divs:

::: {.panel-tabset}

Title of Panel 1

Content of panel 1

Title of Panel 2

Content of panel 2

:::

- **Exercise 2f**: Introduce tab panels in the exercise doc, one with some text and another one with a different text

R code chunks

- Used to import data, to do calculations and statistical analysis
    - (incl all proceeding steps)
- R code chunks produce
    - Text output, tables, graphics
- Contain R code which is executed when document is generated

R code chunks

- Distinct begin ```{r}
- Distinct end ```
- Can have name to refer to (#| label: name)
- Inside the chunk use 'normal' R code
- Many code chunk options via 'hash pipe' #|

````
```{r}
#| label: example
#| echo: true
#| eval: true
#| warning: false
#| include: true
#| fig-width: 60
#| fig-height: 80
#| fig-cap: caption to figure

some code here

```
````

Chunk options

option	default	effect
eval	TRUE	Whether to evaluate the code and include its results
echo	TRUE	Whether to display code along with its results
warning	TRUE	Whether to display warnings
error	FALSE	Whether to display errors
message	TRUE	Whether to display messages
tidy	FALSE	Whether to reformat code in a tidy way when displaying it
results	"markup"	"markup", "asis", "hold", or "hide"
cache	FALSE	Whether to cache results for future renders
comment	"##"	Comment character to preface results with
fig.width	7	Width in inches for plots created in chunk
fig.height	7	Height in inches for plots created in chunk

Options can also be put in yaml

```yaml
execute:
  echo: false
  warning: false
  message: false
```

- Options placed in the YAML under execute will be executed for all code chunks
- These general settings can be overruled in individual code chunks

Exercise 3

- Continue with **exercise 3** in your doc
  - Insert code chunks to load libraries and data
  - Use the slash (/) character in Visual mode (the fastest way) to start code chunks (Ctrl-Alt-I in source mode)
  - Give code chunks a name and make use of chunk options with the hash pipe (#|)
- Chunk names:
  - #| label: chunkname
  - #| label: chunk-name
  - #| label: chunkname1
  - NOT: #| label: my_chunkname (no underscores)
  - NOT: #| label: chunk name (no spaces)

Inline code

- Numbers etc. may be needed in a text paragraph

To predict the trait fruity, ten metabolites have been selected explaining up to 82 % of the variance. The biggest contribution comes from metabolites 994 and 1480. Trait spicy is predicted by just one metabolite (1036) with 31 % explained

- Code chunk not suited for this
- Use inline code
  - single backticks `r *RCodeGoesHere*`

```
34
35 The model has an explained variance of `r round(summary(MyModel)$r.squared,2)`.
36
```

The model has an explained variance of 0.26.

Exercise 4

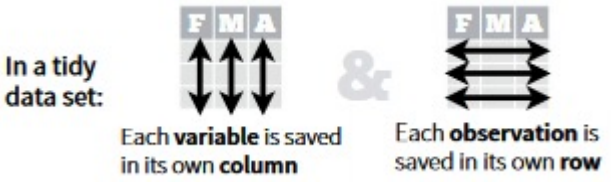
- Continue with **exercise 4**:
  - Produce some inline code (instructions in reader)

Tables and graphs

- The basis for tables and graphs is formed by tidy data in dataframes or tibbles
- Data wrangling may be needed to get the data in the right format
- The R package tidyverse is developed for that
- Tables and graphs can be automatically numbered and referenced by:
    - #| label: fig-name
    - #| fig-cap: This is figure X
    - #| label: tbl-name
    - #| tbl-cap: This is Table X
- Reference in text: @fig-name, or @tbl-name

Tidyverse – once again

- Consistent ecosystem of packages for statistics with R
- Modelled on the concept of tidy data
- Tidy data:
  - Every column is a variable
  - Every row is an observation
  - Every cell is a single value



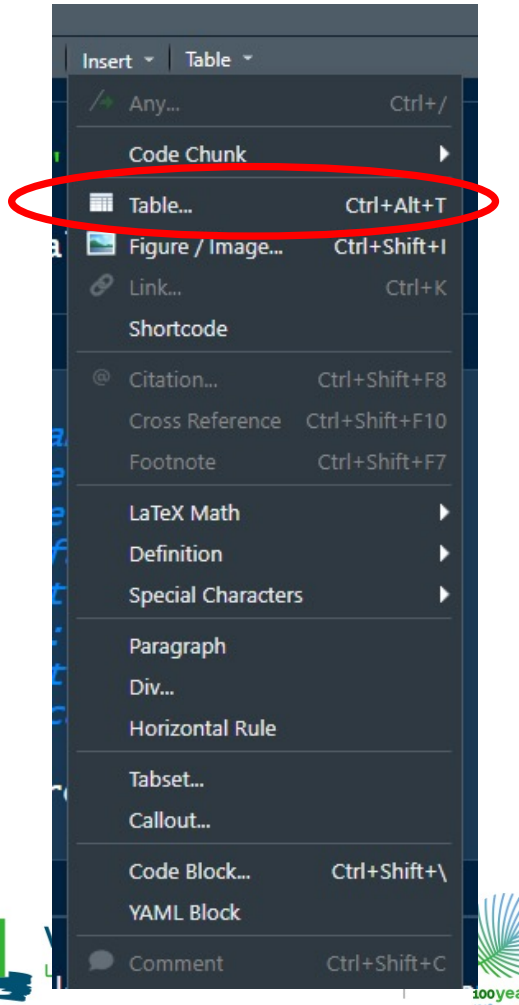
- Store your data like this and you'll spent less time fighting with R and more time on your analysis

Introduction to Tables

- Writing tables in markdown format is very rudimentary and not further discussed in this workshop
- Many packages to produce tables from data have been developed over the years
- The kable function from the package knitr (and its extension kableExtra) was recommended for RMarkdown
- A relatively new package gt developed by the Rstudio team: gt = grammar of tables
- With the advent of Quarto, every table package can easily be integrated via code chunks

Tables in Quarto

- A simple table (not necessarily based on data) can be directly inserted in the text when in Visual mode:



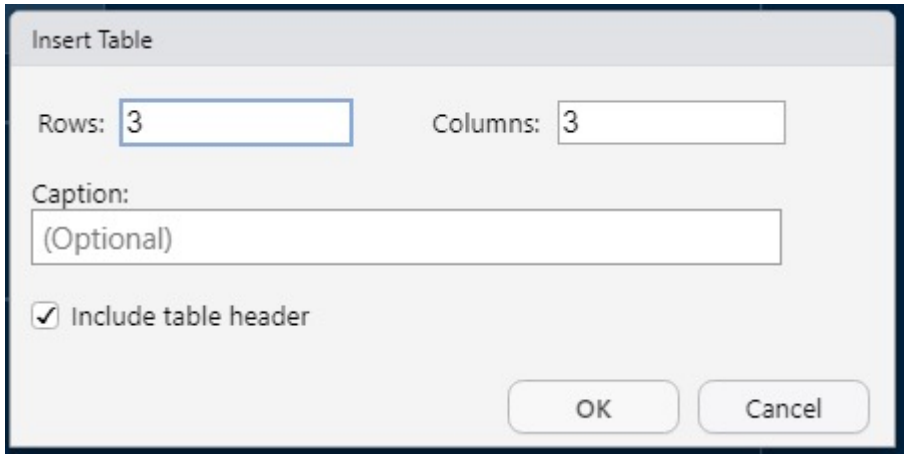


Table needs to be filled in by hand,

not transparent when used for data!

Demo: mtcars data

- Motor Trend Car Road Tests
- The data was extracted from the 1974 Motor Trend US magazine and comprises fuel consumption and 10 aspects of automobile design and performance for 32 automobiles (1973–74 models).

```
> mtcars
                   mpg cyl  disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
```

- The data set is available in base R
- Let’s produce some tables from this dataframe

Table packages: kable

Tables based on data: use packages in code chunks

```r
{r}
#| label: tbl-kable
#| tbl-cap: "Table produced by kable"

knitr::kable(mtcars[1:6,1:3])
```

Table 1: Table produced by kable

	mpg	cyl	disp
Mazda RX4	21.0	6	160
Mazda RX4 Wag	21.0	6	160
Datsun 710	22.8	4	108
Hornet 4 Drive	21.4	6	258
Hornet Sportabout	18.7	8	360
Valiant	18.1	6	225

Table packages: kable + kableExtra

```{r}
#| label: tbl-kExtra
#| tbl-cap: "Using kable and kableExtra to show the first 6 rows and first 3 columns of mtcars"

library(kableExtra)

knitr::kable(mtcars[1:6,1:3], booktabs=TRUE) %>%
kable_styling(position="center", full_width = F)
```

Table 2: Using kable and kableExtra to show the first 6 rows and first 3 columns of mtcars

	mpg	cyl	disp
Mazda RX4	21.0	6	160
Mazda RX4 Wag	21.0	6	160
Datsun 710	22.8	4	108
Hornet 4 Drive	21.4	6	258
Hornet Sportabout	18.7	8	360
Valiant	18.1	6	225

Table package gt

```{r}
#| label: tbl-gt
#| tbl-cap: "Table produced using the package gt"

library(gt)

head(mtcars[,1:6]) %>%
  tibble::rownames_to_column("car") %>%
  gt()
```

Table 3: Table produced using the package gt

car	mpg	cyl	disp	hp	drat	wt
Mazda RX4	21.0	6	160	110	3.90	2.620
Mazda RX4 Wag	21.0	6	160	110	3.90	2.875
Datsun 710	22.8	4	108	93	3.85	2.320
Hornet 4 Drive	21.4	6	258	110	3.08	3.215
Hornet Sportabout	18.7	8	360	175	3.15	3.440
Valiant	18.1	6	225	105	2.76	3.460

Packages built upon gt

- Packages gtextra and gtsummary produce publication-ready tables:
  - From a regression model
  - Summary table from a dataframe/tibble

```r
{r}
#| label: tbl-gtsummary
#| tbl-cap: "Summary table produced by gtsummary"

library(gtsummary)

tbl_summary(mtcars)
```

Table 2: Summary table produced by gtsummary

Characteristic	N = 32[1]
mpg	19.2 (15.4, 22.8)
cyl	
4	11 (34%)
6	7 (22%)
8	14 (44%)
disp	196 (121, 326)
hp	123 (96, 180)
drat	3.70 (3.08, 3.92)
wt	3.33 (2.58, 3.61)
qsec	17.71 (16.89, 18.90)
vs	14 (44%)
am	13 (41%)
gear	
3	15 (47%)
4	12 (38%)
5	5 (16%)
carb	
1	7 (22%)
2	10 (31%)
3	3 (9.4%)
4	10 (31%)
6	1 (3.1%)
8	1 (3.1%)

[1] Median (IQR); n (%)

Table packages

- There is a variety of packages to produce tables:
  - kable + kableExtra
  - gt + gtsummary + gtextra
  - stargazer
  - flextable
  - huxtable
- Whichever you choose:
  - Use the prefix ‘tbl-’ so that Quarto knows it’s a table
  - Label with #| label: tbl-name
  - Caption with #| tbl-cap: “table caption”
  - Refer to tables in text with @tbl-name

Table exercises

- **Exercise 5**:
  - produce a table in the mock report via 'insert table' in Visual mode (this produces actually a markdown table that you can see if you switch to source mode)
  - Reproduce tables 1, 2, 3 from the penguins manuscript using code chunks and kable and kableExtra (or another package if you like)

Figures in Quarto

- There are several ways to incorporate figures in a Quarto document
  - Imported from an outside source (e.g., a logo)
  - Graphs produced with base R plot functions
  - Graphs produced with ggplot

Importing figures

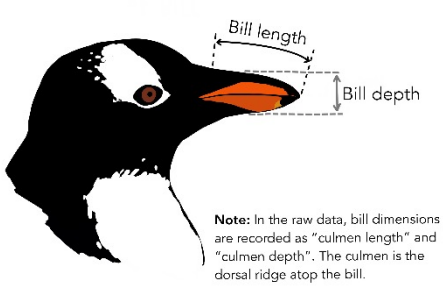
- Two options:
  - Outside a code chunk: ![caption](link)
  - With figure reference: ![caption](link){#fig-name}
  - Inside a code chunk with the knitr function include_graphics():

```
```r
#| label: fig-knitrfunction
#| fig-cap: Use of knitr to include files from outside
Quarto
knitr::include_graphics("filename.png")
```
```

- Use the prefix 'fig-' to label code chunks and to add captions so that Quarto knows it is a figure
- Refer to in text with @fig-name

Exercise 6

- Import figures in the exercise manuscript
  - Try the two methods: without and with a code chunk
  - Refer to the figures in the text



ggplot2

- With ggplot2, it's easy to:
  - produce handsome, publication-quality plots, with automatic legends created from the plot specification
  - superpose multiple layers (points, lines, maps, tiles, box plots to name a few) from different data sources, with automatically adjusted common scales
  - add customisable smoothers that use the powerful modelling capabilities of R, such as loess, linear models, generalised additive models and robust regression
- Use the package patchwork to combine graphs in the document

Components of graphics

- data : the data used (data frame)
- coordinate system : 2d space used (cartersian, polar, …)
- geoms : type of plot (points, lines, polygons, …
- aesthetics : visual characteristics (size, color, shape, fill, …)
- scales : conversion to display values (log, color, size, …)
- stats : stat transformation (counts, mean, median, regression lines)
- facets : how is data split in subsets with own small graphs

Overview

(short) overview in cheat sheet ggplot2

Two Variables

Continuous X, Continuous Y

f <- ggplot(mpg, aes(cty, hwy))

f + **geom_blank()**

f + **geom_jitter()**
x, y, alpha, color, fill, shape, size

f + **geom_point()**
x, y, alpha, color, fill, shape, size

f + **geom_quantile()**
x, y, alpha, color, linetype, size, weight

f + **geom_rug**(sides = "bl")
alpha, color, linetype, size

f + **geom_smooth**(model = lm)
x, y, alpha, color, fill, linetype, size, weight

f + **geom_text**(aes(label = cty))
x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

Continuous Bivariate Distribution

i <- ggplot(movies, aes(year, rating))

i + **geom_bin2d**(binwidth = c(5, 0.5))
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size, weight

i + **geom_density2d()**
x, y, alpha, colour, linetype, size

i + **geom_hex()**
x, y, alpha, colour, fill size

Continuous Function

j <- ggplot(economics, aes(date, unemploy))

j + **geom_area()**
x, y, alpha, color, fill, linetype, size

j + **geom_line()**
x, y, alpha, color, linetype, size

j + **geom_step**(direction = "hv")
x, y, alpha, color, linetype, size

Visualizing error

df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)

k <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))

k + **geom_crossbar**(fatten = 2)
x, y, ymax, ymin, alpha, color, fill, linetype, size

k + **geom_errorbar()**
x, ymax, ymin, alpha, color, linetype, size, width (also **geom_errorbarh()**)

k + **geom_linerange()**
x, ymin, ymax, alpha, color, linetype, size

k + **geom_pointrange()**
x, y, ymin, ymax, alpha, color, fill, linetype, shape, size

Discrete X, Continuous Y

g <- ggplot(mpg, aes(class, hwy))

g + **geom_bar**(stat = "identity")
x, y, alpha, color, fill, linetype, size, weight

g + **geom_boxplot()**
lower, middle, upper, x, ymax, ymin, alpha, color, fill, linetype, shape, size, weight

g + **geom_dotplot**(binaxis = "y", stackdir = "center")
x, y, alpha, color, fill

g + **geom_violin**(scale = "area")
x, y, alpha, color, fill, linetype, size, weight

Maps

data <- data.frame(murder = USArrests$Murder, state = tolower(rownames(USArrests)))

map <- map_data("state")

l <- ggplot(data, aes(fill = murder))

l + **geom_map**(aes(map_id = state), map = map) + **expand_limits**(x = map$long, y = map$lat)
map_id, alpha, color, fill, linetype, size

Discrete X, Discrete Y

h <- ggplot(diamonds, aes(cut, color))

h + **geom_jitter()**
x, y, alpha, color, fill, shape, size

WAGENINGEN UNIVERSITY & RESEARCH

100years 1918 — 2018

Exercise 7: graphs in the exercise report

- Reproduce Figures 2, 3, 4, 5, and 6 in the penguins report using ggplot
- Give captions to the figures
- Use the R package patchwork to combine figures
- Write some text in the manuscript what the figures represent and refer to them in the text

Inserting equations

- Ways to introduce equations in a document
  - Inline equations in text:

The famous Einstein equation is $E = mc^2$ that everyone knows.

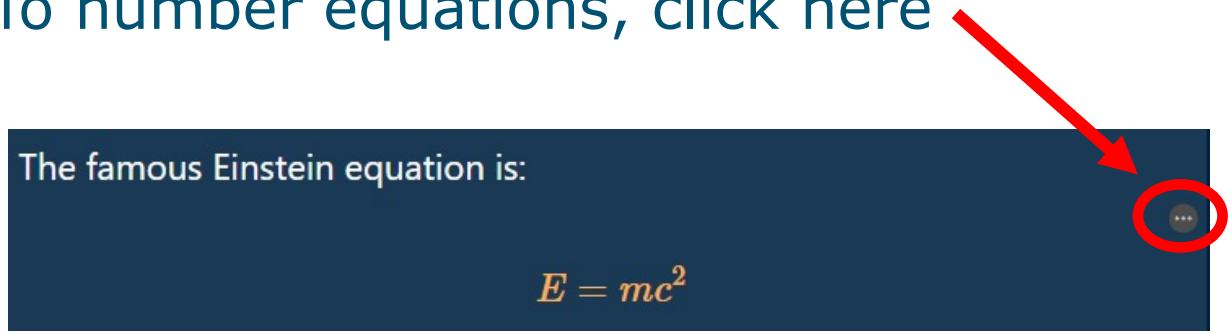
  - Equations separated from text on a separate line:

The famous Einstein equation is:

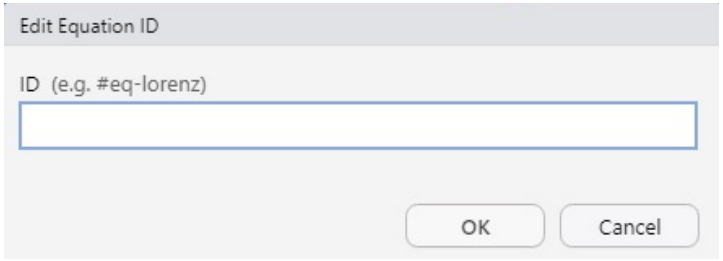
$$E = mc^2$$

Inserting equations

- To number equations, click here

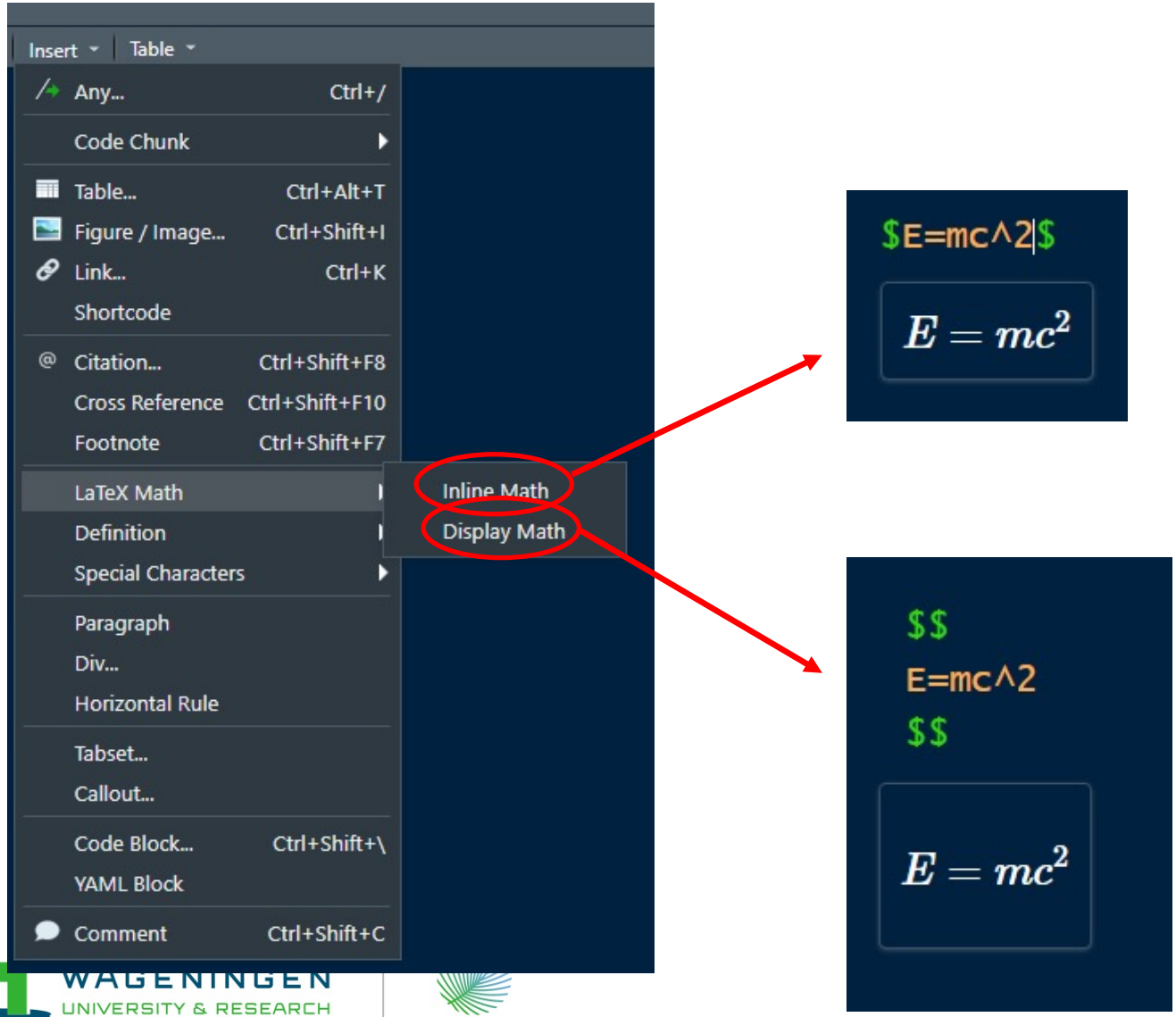


- Give a name:



- Refer to equation in text as @eq-name

Equations are added in Latex style



Exercise 8

- Practice with writing equations LaTex style
- Follow the instructions in the reader
- See LaTex cheat sheets on internet if codes are needed

Cross references overview

- With cross references, Quarto will number Figures, Tables, Equations and Sections according to the order they appear in the text

Element	Label prefix	Cross reference in text
Figure	fig-name (in chunk)	@fig-name becomes Figure ...
Table	tbl-name (in chunk)	@tbl-name becomes Table ...
Equation	#eq-name (after equation)	@eq-name becomes Equation ....
Section	#sec-name (after section)	@sec-name becomes Section ...

Exercise 9

- Produce a toc (table of contents) for the exercise doc
- Number sections and lists of tables and figures

End of workshop Part 1

- Possibility to do some homework to practice what you learned today
- Use own data if you have them
- Otherwise: two datasets are available
    - Wine data
    - Pottery data
- Produce two short reports, one for each data set
- See reader for instructions
- Will be used in Part 2 to compile the three docs into a booklet