DAY 1: INTRO TO R & RSTUDIO

BSTA 511/611 Fall 2023, OHSU

Meike Niederhausen, PhD

2023-09-27

INTRODUCTION TO R

Artwork by @allison_horst

WHAT IS R?

- A programming language
- Focus on statistical modeling and data analysis
  - import data, manipulate data, run statistics, make plots
- Useful for data science
- Great visualizations
- Also useful for most anything else you'd want to tell a computer to do
- Interfaces with other languages i.e. python, C++, bash

For the history and details: Wikipedia

- an interpreted language (run it through a command line)
- procedural programming with functions
- Why "R"?? Scheme inspired S (invented at Bell Labs in 1976) which inspired R since 1st letters of original authors (**free and open source!** in 2000)

WHAT IS RSTUDIO?

R is a programming language

RStudio is an integrated development environment (IDE)
= an interface to use R (with perks!)

R: Engine

RStudio: Dashboard

OPEN RSTUDIO ON YOUR COMPUTER (NOT R!)

1.1.2 Using R via RStudio

Recall our car analogy from earlier. Much as we don't drive a car by interacting directly with the engine but rather by interacting with elements on the car's dashboard, we won't be using R directly but rather we will use RStudio's interface. After you install R and RStudio on your computer, you'll have two new *programs* (also called *applications*) you can open. We'll always work in RStudio and not in the R application. Figure 1.2 shows what icon you should be clicking on your computer.

R: Do not open this	RStudio: Open this
	

FIGURE 1.2: Icons of R versus RStudio on your computer.

RSTUDIO ANATOMY

BuzzR

RStudio anatomy

https://buzzrbeeline.blog/

Emma Rand

Script file

Write code here

To run code put your cursor on the line and click the run button

Edit to correct errors

⇒ record of commands that worked

Save scripts with the **.R** extension

⇒ syntax will be highlighted

⇒ good practice

`<-` is the assignment operator

⇒ puts what is on the right in to the object on the left

⇒ Assign results if you want to use them again

Console

When you click run, code is sent to the console and executed

`>` is the prompt

⇒ do not type it

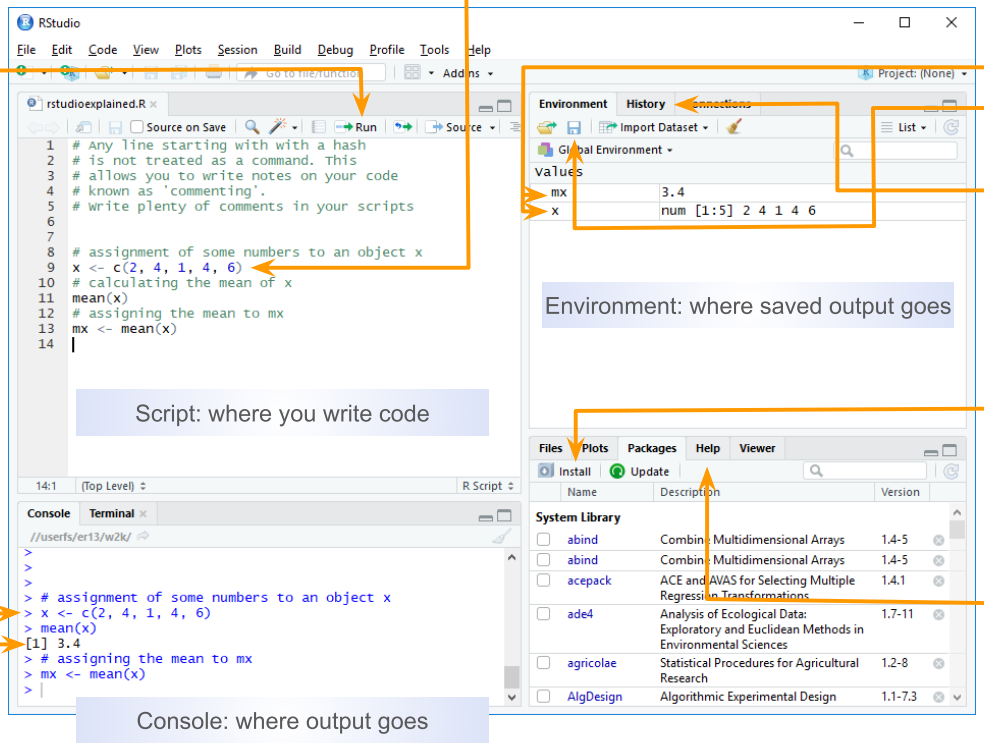
⇒ appears when R is ready for next command

Command output goes here by default

⇒ output is in a different colour

⇒ `[1]` indicates 3.4 is the first element of the output

⇒ many commands will not have output, the prompt just reappears



Environment

Name objects by assignment to use them again

All the objects you created in your session

Saving the environment saves all the objects, but not the code with a **.RData** extension

History

A history of every command you sent to the console, mistakes included.

File can be saved but usually you just need the script

Packages

Many functions come with R

A huge amount of extra functionality is available in packages

Packages can be installed by clicking the Install button

Help

Access to manual pages for all installed packages

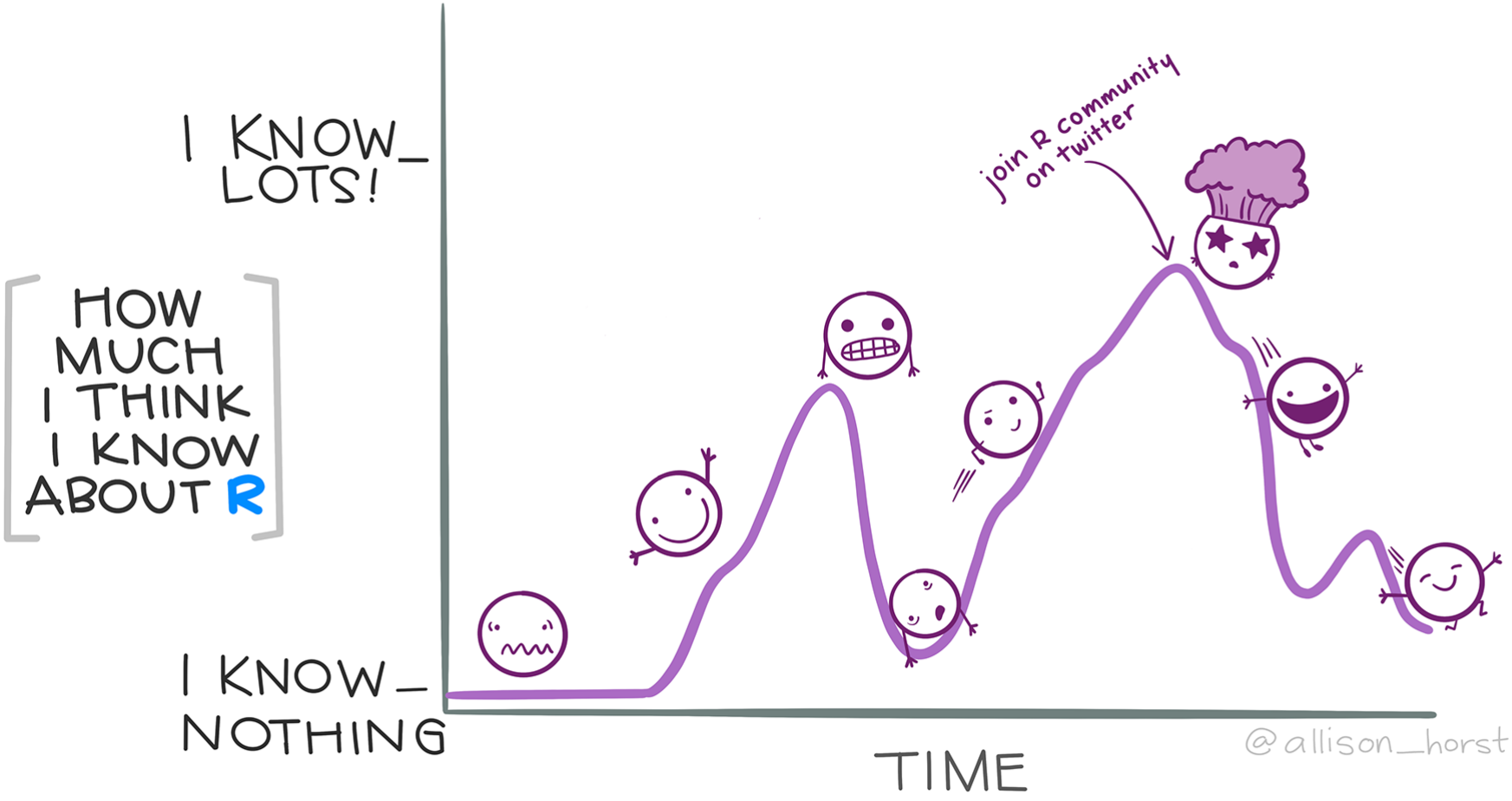
Plots

Figure output appears here

Emma Rand

Read more about RStudio's layout in Section 3.4 of "Getting Used to R, RStudio, and R Markdown" (Ismay and Kennedy 2016)

LET'S CODE! R BASICS



Artwork by @allison_horst

CODING IN THE CONSOLE

When you first open R, the console should be empty.

Console Terminal Jobs

~/Google Drive/BERD R Classes/berd_r_courses_github/

```
R version 4.0.2 (2020-06-22) -- "Taking Off Again"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin17.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
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.

>
```

Typing and executing code in the console

- Type code in the console (blue text)
- Press **return** to execute the code
- Output shown below in black

```
> 7
[1] 7
> 3 + 5
[1] 8
> "hello"
[1] "hello"
> # this is a comment, nothing happens
> # 5 - 8
> # separate multiple commands with ;
> 3 + 5; 4 + 8
[1] 8
[1] 12
>
```

MATH CALCULATIONS USING R

- Rules for order of operations are followed
- Spaces between numbers and characters are ignored

```
10^2
```

```
[1] 100
```

```
3 ^ 7
```

```
[1] 2187
```

```
6/9
```

```
[1] 0.6666667
```

```
9-43
```

```
[1] -34
```

```
4^3-2* 7+9 /2
```

```
[1] 54.5
```

The equation above is computed as

$$4^3 - (2 \cdot 7) + \frac{9}{2}$$

VARIABLES (SAVED R OBJECTS)

Variables are used to store data, figures, model output, etc.

- Can assign a variable using either = or <–
  - **Using <– is preferable**

Assign just one value:

```
x = 5
x
```

```
[1] 5
```

```
x <- 5
x
```

```
[1] 5
```

Assign a **vector** of values

- Consecutive integers using **:**

```
a <- 3:10
a
```

```
[1]  3  4  5  6  7  8  9 10
```

- **Concatenate** a string of numbers

```
b <- c(5, 12, 2, 100, 8)
b
```

```
[1]   5  12   2 100   8
```

DOING MATH WITH VARIABLES

Math using variables with just one value

```
x <- 5
x
```

```
[1] 5
```

```
x + 3
```

```
[1] 8
```

```
y <- x^2
y
```

```
[1] 25
```

Math on vectors of values: **element-wise** computation

```
a <- 3:6
a
```

```
[1] 3 4 5 6
```

```
a+2; a*3
```

```
[1] 5 6 7 8
[1]  9 12 15 18
```

```
a*a
```

```
[1]  9 16 25 36
```

VARIABLES CAN INCLUDE TEXT (CHARACTERS)

```r
hi <- "hello"
hi
```

```
[1] "hello"
```

```r
greetings <- c("Guten Tag", "Hola", hi)
greetings
```

```
[1] "Guten Tag" "Hola"      "hello"
```

USING FUNCTIONS

- `mean()` is an example of a function
- functions have "arguments" that can be specified within the `()`
- `?mean` in console will show help file for `mean()`

Function arguments specified by name:

```
mean(x = 1:4)
```

```
[1] 2.5
```

```
seq(from = 1, to = 12, by = 3)
```

```
[1]  1  4  7 10
```

```
seq(by = 3, to = 12, from = 1)
```

```
[1]  1  4  7 10
```

Function arguments not specified, but listed in order:

```
mean(1:4)
```

```
[1] 2.5
```

```
seq(1, 12, 3)
```

```
[1]  1  4  7 10
```

COMMON CONSOLE ERRORS (1/2)

Incomplete commands

- When the console is waiting for a new command, the prompt line begins with >
  - If the console prompt is +, then a previous command is incomplete
  - You can finish typing the command in the console window

Example:

```
> 3 + (2*6
+ )
```

```
[1] 15
```

COMMON CONSOLE ERRORS (2/2)

Object is not found

- This happens when text is entered for a non-existent variable (object)

Example:

```
hello
```

```
Error in eval(expr, envir, enclos): object 'hello' not found
```

- Can be due to missing quotes

```
install.packages(dplyr)
```

```
Error in install.packages(dplyr): object 'dplyr' not found
```

```
# correct code is: install.packages("dplyr")
```

SAVING YOUR WORK WITH QUARTO

or, creating reproducible reports

Artwork by @allison_horst

EXAMPLE: CREATING AN HTML FILE

.qmd file

Untitled2*

Render on Save | Render | Run

Source | Visual | Normal | Format | Insert | Table

```yaml
---
title: "My first Quarto file"
author: "Meike"
format: html
editor: visual
---
```

Quarto

Quarto enables you to weave together content and executable code into a finished document. To learn more about Quarto see https://quarto.org.

Running Code

When you click the **Render** button a document will be generated that includes both content and the output of embedded code. You can embed code like this:

```{r}
1 + 1
```

You can add options to executable code like this

```{r}
#| echo: false
2 * 2
```

The `echo: false` option disables the printing of code (only output is displayed).

html output

My first Quarto file

AUTHOR
Meike

Quarto

Quarto enables you to weave together content and executable code into a finished document. To learn more about Quarto see https://quarto.org.

Running Code

When you click the **Render** button a document will be generated that includes both content and the output of embedded code. You can embed code like this:

```
1 + 1
```

```
[1] 2
```

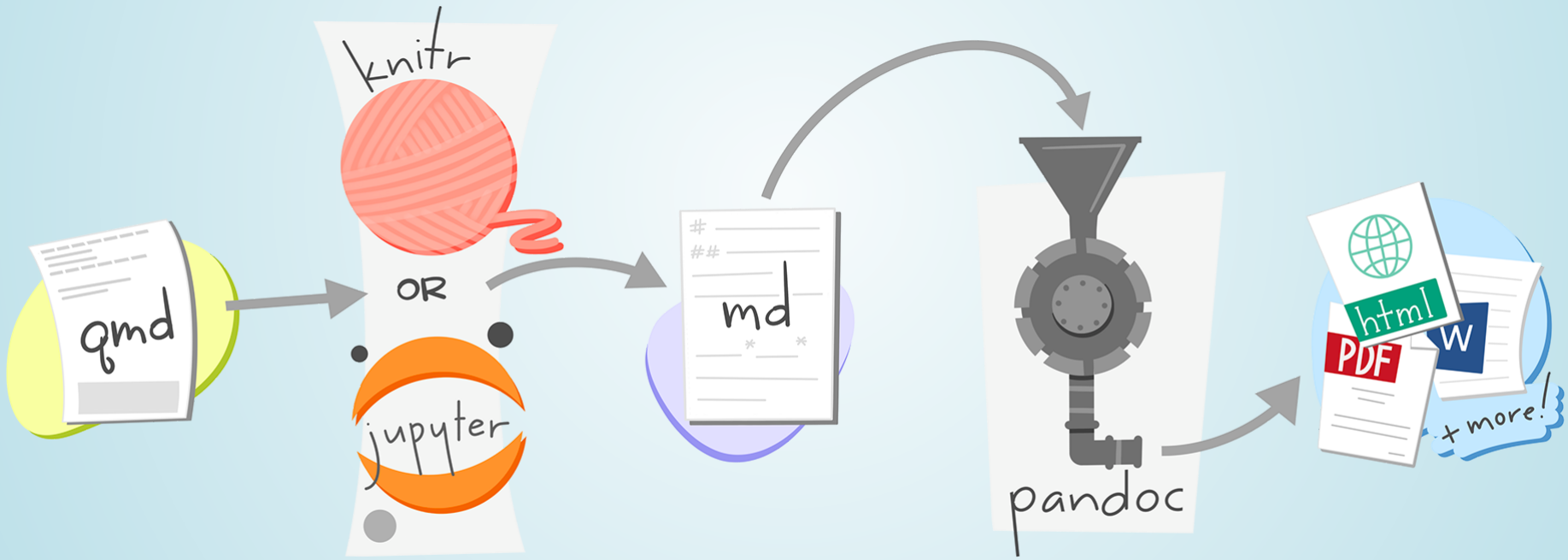
You can add options to executable code like this

```
[1] 4
```

The `echo: false` option disables the printing of code (only output is displayed).

QUARTO = `.qmd` FILE = CODE + TEXT

`knitr` is a package that converts `.qmd` files containing code + markdown syntax to a plain text `.md` markdown file, and then to other formats (html, pdf, Word, etc)



Artwork from “Hello, Quarto” keynote by Julia Lowndes and Mine Çetinkaya-Rundel, presented at RStudio Conference 2022. Illustrated by Allison Horst.

BASIC QUARTO EXAMPLE

Artwork from “Hello, Quarto” keynote by Julia Lowndes and Mine Çetinkaya-Rundel, presented at RStudio Conference 2022. Illustrated by Allison Horst.

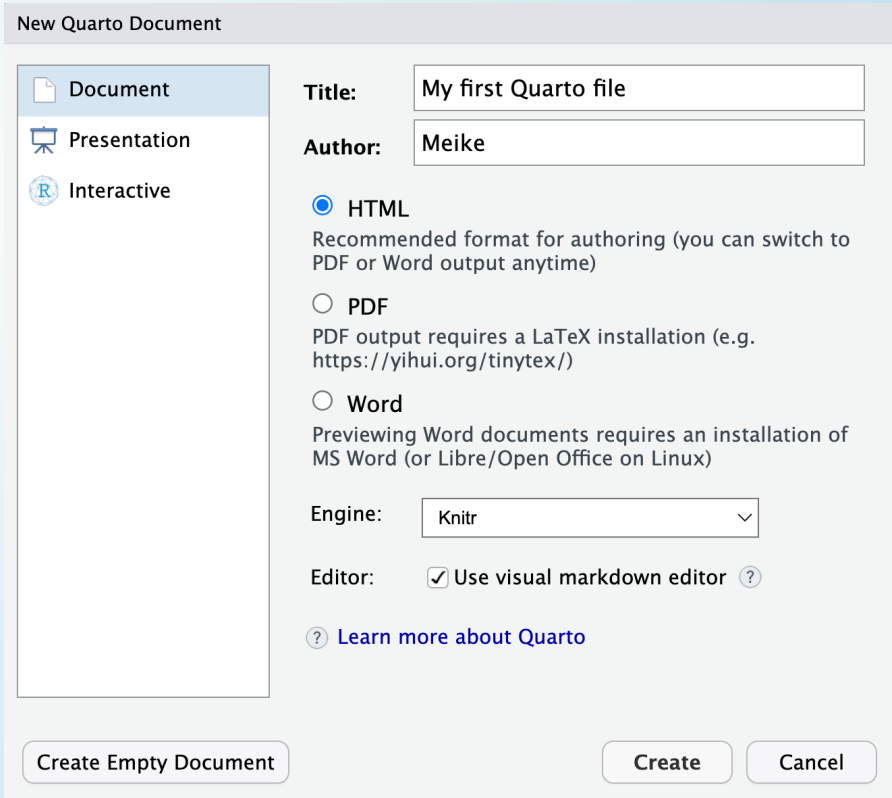
1. CREATE A QUARTO FILE (.qmd)

Two options:

1. click on File → New File → Quarto Document... → OK,
2. or in upper left corner of RStudio click on  → 

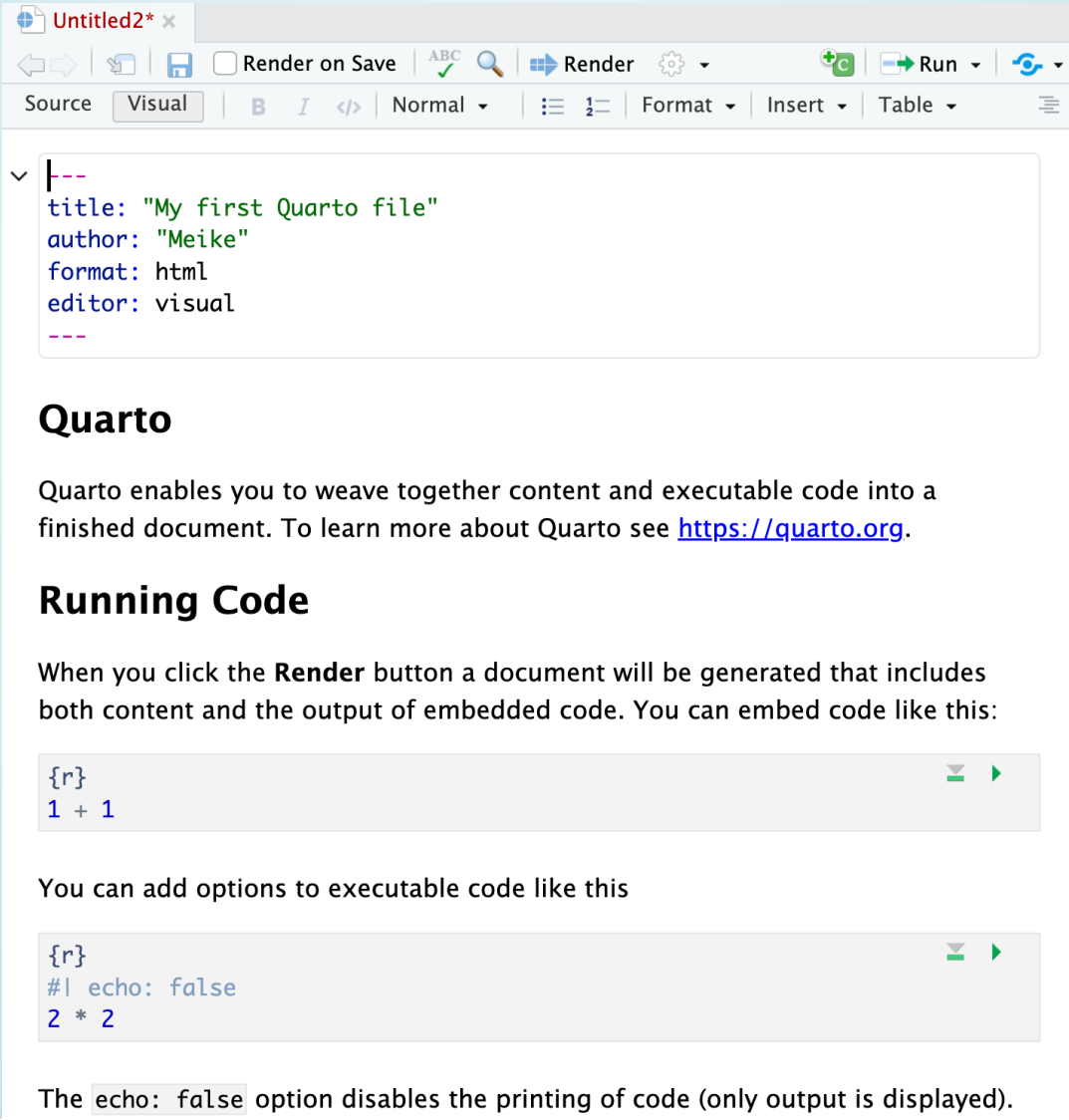
Pop-up window selections:

- Enter a title and your name
- Select `HTML` output format (default)
- Engine: select `Knitr`
- Editor: Select `Use visual markdown editor`
- Click `Create`



2. CREATE A QUARTO FILE (.qmd)

- After clicking on `Create`, you should then see the following in your editor window:



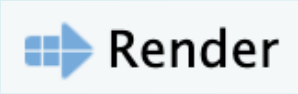
3. SAVE THE QUARTO FILE (`.qmd`)

- **Save the file** by
  - selecting `File -> Save`,
  - or clicking on  (towards the left above the scripting window),
  - or keyboard shortcut
    - PC: *Ctrl* + *s*
    - Mac: *Command* + *s*
- You will need to specify
  - a **filename** to save the file as
    - ALWAYS use **.qmd** as the filename extension for Quarto files
  - the **folder** to save the file in

4. CREATE HTML FILE

We create the html file by **rendering** the .qmd file.

Two options:

1. click on the Render icon 

at the top of the editor window,
2. or use keyboard shortcuts
    - Mac: *Command+Shift+K*
    - PC: *Ctrl+Shift+K*

- A new window will open with the html output.
- You will now see both .qmd and .html files in the folder where you saved the .qmd file.

Note

- The template .qmd file that RStudio creates will render to an html file by default.
- The output format can be changed to create a Word doc, pdf, slides, etc.

.QMD FILE VS. ITS HTML OUTPUT

.qmd file

Untitled2*

Render on Save | Render | Run

Source | Visual | Normal | Format | Insert | Table

```yaml
---
title: "My first Quarto file"
author: "Meike"
format: html
editor: visual
---
```

Quarto

Quarto enables you to weave together content and executable code into a finished document. To learn more about Quarto see https://quarto.org.

Running Code

When you click the **Render** button a document will be generated that includes both content and the output of embedded code. You can embed code like this:

```{r}
1 + 1
```

You can add options to executable code like this

```{r}
#| echo: false
2 * 2
```

The `echo: false` option disables the printing of code (only output is displayed).

html output

My first Quarto file

AUTHOR
Meike

Quarto

Quarto enables you to weave together content and executable code into a finished document. To learn more about Quarto see https://quarto.org.

Running Code

When you click the **Render** button a document will be generated that includes both content and the output of embedded code. You can embed code like this:

```
1 + 1
```

```
[1] 2
```

You can add options to executable code like this

```
[1] 4
```

The `echo: false` option disables the printing of code (only output is displayed).

3 TYPES OF QUARTO CONTENT

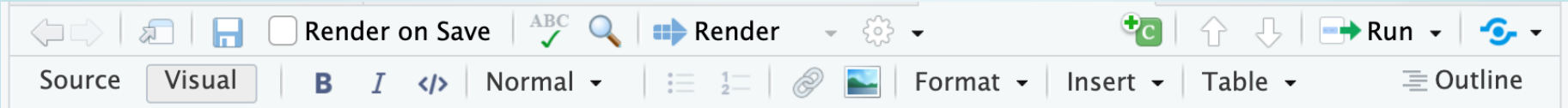
1. *Text*, lists, images, tables, links
2. Code chunks
3. YAML metadata

Illustration by Alison Hill and Allison Horst, for RStudio.

FORMATTING TEXT

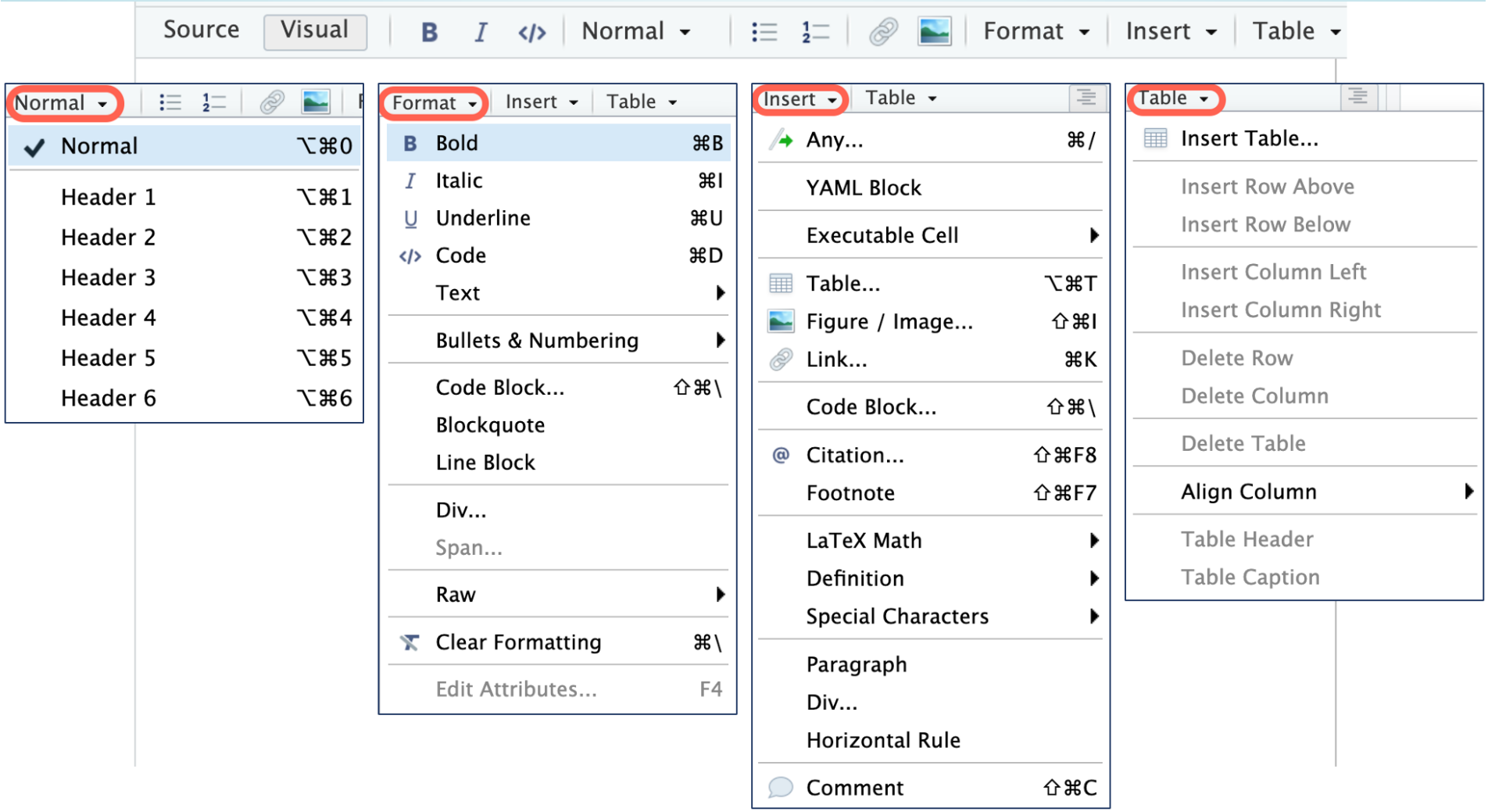
- **bold**, *italics*, super^{scripts} & sub$_{\text{scripts}}$, ~~strikethrough~~, `verbatim`, etc.

- Text is formatted through a markup language called `Markdown` (Wikipedia)
  - Other markup languages include html (webapges) and LaTeX (math)
  - All text formatting is specified via *code*
  - "Markdown is a plain text format that is designed to be easy to write, and, even more importantly, easy to read" [1]
- Newer versions of RStudio include a `Visual editor` as well that makes formatting text similar to using a word processor.



FORMATTING TEXT: `Visual editor`

- Using the `Visual editor` is similar to using a wordprocessor, such as Word
- Keyboard shortcuts usually work as well (*shown for Mac below*)



PRACTICE

1. Part 1
    1. Using the visual editor, practice formatting text in your qmd file, such as making text **bold**, *italicized*, and in `code` format.
    2. Add 1st, 2nd, and 3rd level headers
    3. Add a list with a
        - sub-list (bullet and/or numbered)
    4. Add a table
    5. Add whatever else you are interested in!
2. Part 2
    1. Switch back to the `Source` editor and examine the markdown code that was used for the formatting.

Questions:

1. What went smoothly?
2. What hurdles did you encounter?

FORMATTING TEXT: Markdown

Markdown:	**Output:**
`*This text is in italics*, but _so is this text_.`	*This text is in italics*, but *so is this text*.
`**Bold** also has __2 options__`	**Bold** also has **2 options**
`~~Should this be deleted?~~`	~~Should this be deleted?~~
`Need^super^ or~sub~ scripts?`	Need^{super} or$_{\text{sub}}$ scripts?
`` `Code is often formatted as verbatim` ``	`Code is often formatted as verbatim`
`>This is a block quote.`	> This is a block quote.

HEADERS

- Organize your documents using headers to create sections and subsections
- Use # at the beginning of the line to create headers

Text in editor:

```
# Header 1

## Header 2

### Header 3

#### Header 4

##### Header 5

###### Header 6
```

Output:

Header 1

Header 2

Header 3

Header 4

Header 5

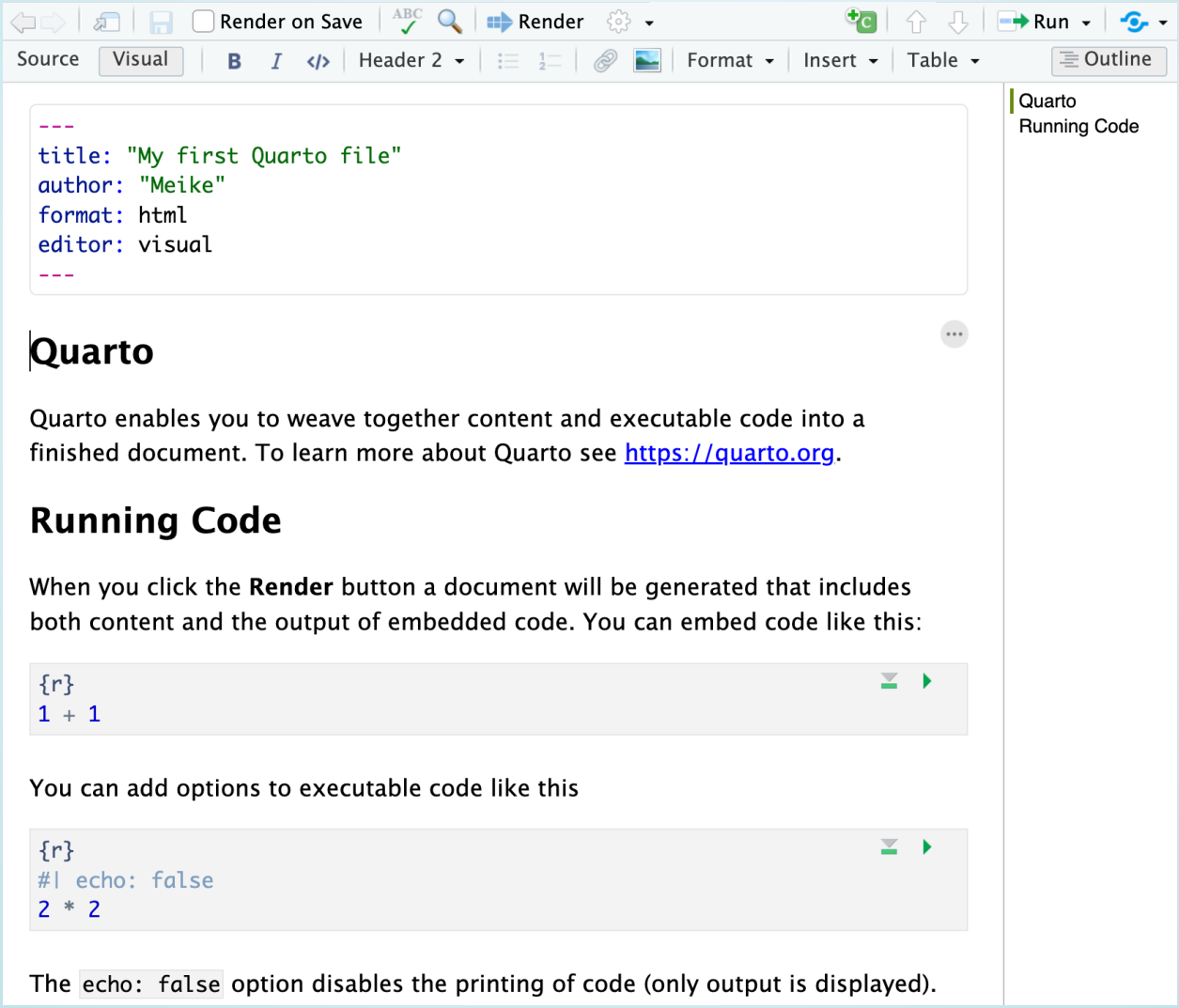
Header 6

Important

Make sure there is no space before the #, and there IS a space after the # in order for the header to work properly.

RSTUDIO TIP

You can easily navigate through your .qmd file if you use headers to outline your text



3 TYPES OF QUARTO CONTENT

1. Text, lists, images, tables, links
2. *Code chunks*
3. YAML metadata

Illustration by Alison Hill and Allison Horst, for RStudio.

CODE CHUNKS

.qmd file

Untitled2*

Render on Save | Render | Run

Source | Visual | Normal | Format | Insert | Table

```yaml
---
title: "My first Quarto file"
author: "Meike"
format: html
editor: visual
---
```

Quarto

Quarto enables you to weave together content and executable code into a finished document. To learn more about Quarto see https://quarto.org.

Running Code

When you click the **Render** button a document will be generated that includes both content and the output of embedded code. You can embed code like this:

```{r}
1 + 1
```

You can add options to executable code like this

```{r}
#| echo: false
2 * 2
```

The `echo: false` option disables the printing of code (only output is displayed).

html output

My first Quarto file

AUTHOR
Meike

Quarto

Quarto enables you to weave together content and executable code into a finished document. To learn more about Quarto see https://quarto.org.

Running Code

When you click the **Render** button a document will be generated that includes both content and the output of embedded code. You can embed code like this:

```
1 + 1
```

```
[1] 2
```

You can add options to executable code like this

```
[1] 4
```

The `echo: false` option disables the printing of code (only output is displayed).

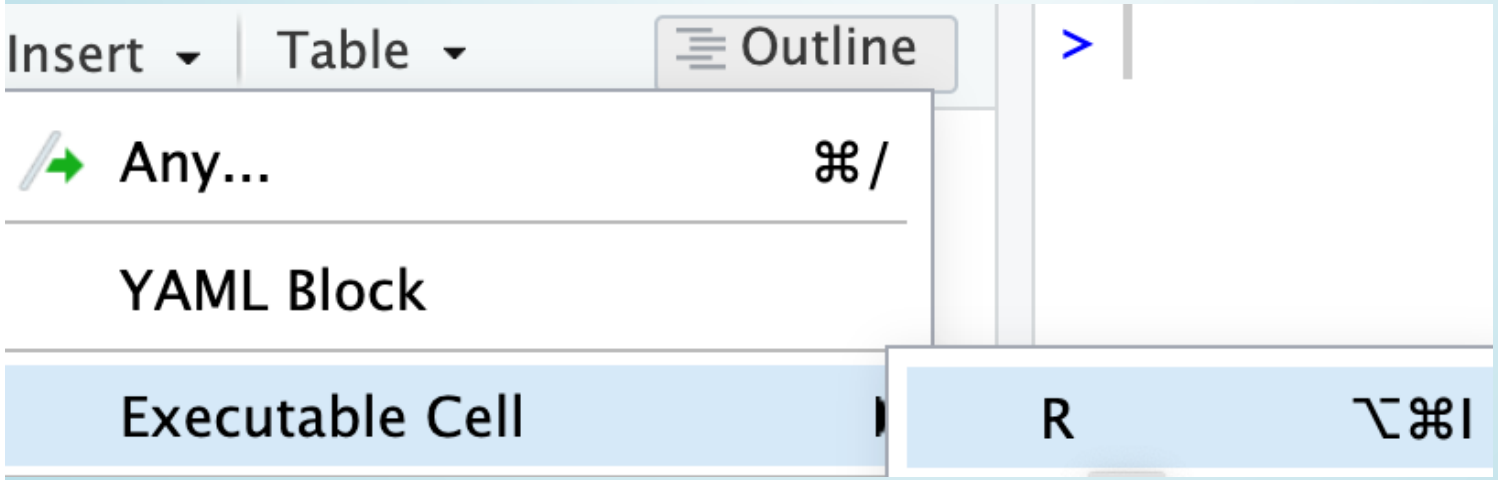
CREATE A CODE CHUNK

3 options to create a code chunk

1. Click on  at top right of the editor window, or
2. **Keyboard shortcut**

Mac	*Command + Option + I*
PC	*Ctrl + Alt + I*

3. `Visual editor`: Select `Insert` -> `Executable Cell` -> `R`



WHAT DOES A CODE CHUNK LOOK LIKE?

An empty code chunk looks like this:

Visual editor

```
{r}
```

Source editor

```
```{r}

```
```

Important

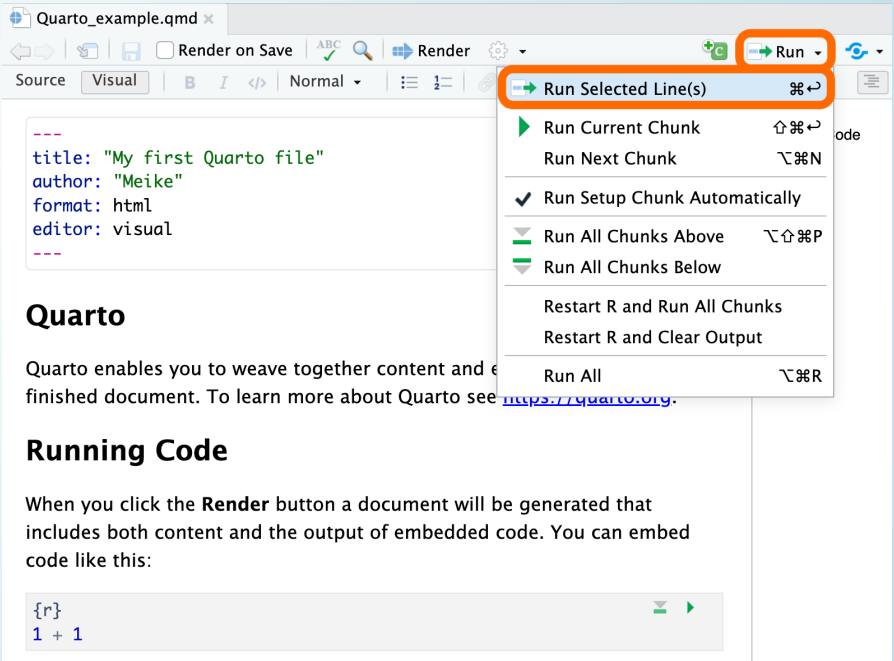
Note that a code chunks start with `` ```{r} `` and ends with `` ``` ``. Make sure there is no space before `` ``` ``.

ENTER AND RUN CODE (1/N)

- **Type R code** inside code chunks
- **Select code** you want to run, by
  - placing the cursor in the line of code you want to run,
  - ***or*** highlighting the code you want to run
- **Run selected code** by
  - clicking on the Run button in the top right corner of the scripting window and choosing `Run Selected Line(s)`,
  - or typing one of the following key combinations:

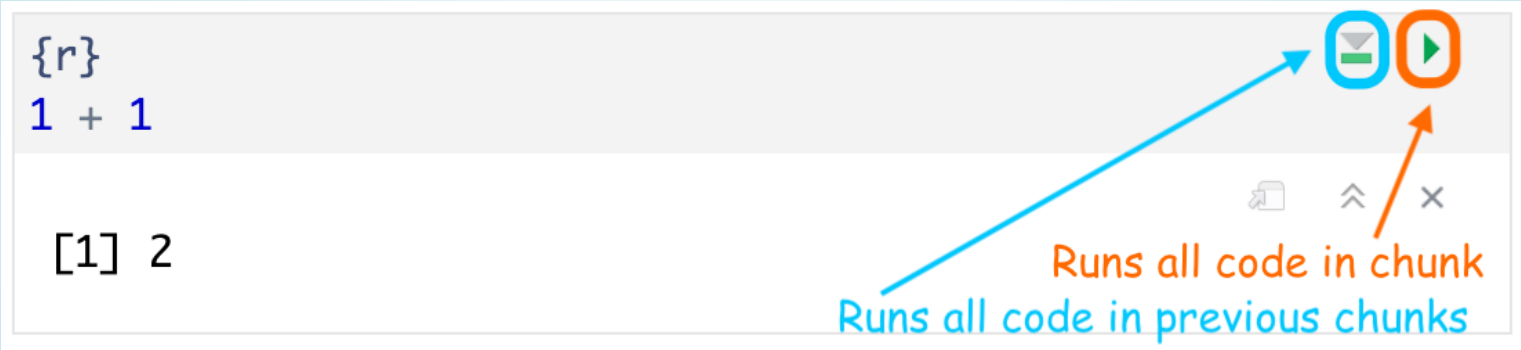
Mac	**ctrl + return**
PC	**command + return**

- *Where does the output appear?*



ENTER AND RUN CODE (2/N)

- **Run all code** in a chunk by
  - by clicking the play button in the top right corner of the chunk
- The code output appears below the code chunk



Note

- The output should also appear in the Console.
- Settings can be changed so that the output appears only in the Console and not below the code chunk:
  - Select (to right of Render button) and then *Chunk Output in Console*.

USEFUL KEYBOARD SHORTCUTS

Full list of keyboard shortcuts

action	mac	windows/linux
Run code in qmd (or script)	cmd + enter	ctrl + enter
`<-`	option + -	alt + -
interrupt currently running command	esc	esc
in console, retrieve previously run code	up/down	up/down
keyboard shortcut help	option + shift + k	alt + shift + k

PRACTICE

Try typing code below in your qmd (with shortcut) and evaluating it:

```
y <- 5
y
```

3 TYPES OF QUARTO CONTENT

1. Text, lists, images, tables, links
2. Code chunks
3. *YAML metadata*

Illustration by Alison Hill and Allison Horst, for RStudio.

YAML METADATA

Many output options can be set in the **YAML metadata**, which is the *first set of code in the file starting and ending with* ---.

- It sets the configuration specifications for the output file
- YAML is an acronym for
  - *yet another markup language*, or
  - *YAML ain't markup language*

SIMPLE YAML EXAMPLE

- The default YAML includes a **title** and **author** that appear at the top of the output file. In the example below, I also added in a **date** option

YAML:

```yaml
---
title: "My first Quarto file"
author: "Meike"
date: "9/25/2023"
format: html
editor: visual
---
```

Output:

My first Quarto file

AUTHOR	PUBLISHED
Meike	September 25, 2023

Important

- The YAML **must** start and end with 3 dashes ---.
- The first set of --- **must** be on the very first line.

CHANGE THE OUTPUT FILE TYPE

- The YAML specifies the format of the output file:
  - html, Word, pdf, slides, website, book, etc.
- This is done by changing the `format:` option

Illustration by Alison Hill and Allison Horst, for RStudio.

```yaml
---
title: "My first Quarto file"
author: "Meike"
date: "9/25/2023"
format: html
editor: visual
---
```

Output format	YAML
html	`format: html`
Word	`format: docx`
pdf[1]	`format: pdf`
html slides	`format: revealjs`
PPT slides	`format: pptx`

Artwork by @allison_horst

YOU WILL GET FRUSTRATED WHILE LEARNING R!

From Garrett Grolemund's Prologue of his book *Hands-On Programming with R*[1]:

> As you learn to program, you are going to get frustrated. You are learning a new language, and it will take time to become fluent. But frustration is not just natural, it's actually a positive sign that you should watch for. Frustration is your brain's way of being lazy; it's trying to get you to quit and go do something easy or fun. If you want to get physically fitter, you need to push your body even though it complains. If you want to get better at programming, you'll need to push your brain. Recognize when you get frustrated and see it as a good thing: you're now stretching yourself. Push yourself a little further every day, and you'll soon be a confident programmer.

RESOURCES

- Official Quarto guide: https://quarto.org/docs/guide/
  - Markdown basics: https://quarto.org/docs/authoring/markdown-basics.html
    - Text formatting, headings, linnks, images, lists, tables, equations, diagrams, page breaks, keyboard shortcuts, and more!
  - Code blocks: https://quarto.org/docs/computations/r.html#code-blocks
    - Chunk options: https://quarto.org/docs/computations/r.html#chunk-options
- Mine Çetinkaya-Rundel's Quarto tip a day: https://mine-cetinkaya-rundel.github.io/quarto-tip-a-day/
- Hadley Wickham's R for Data Science: https://r4ds.hadley.nz/ _ See Chapter 29 for Quarto