Edit and render Quarto notebooks

Guided exercise

Hélène Langet Zhihan Zhu

2025-03-03

Table of contents

1 Learning objectives

- Edit Quarto notebooks.
- Learn about visual editor and source editor.
- Run R code in notebook mode.
- Render Quarto documents in different formats, and execute R code within the text.

2 Editing text

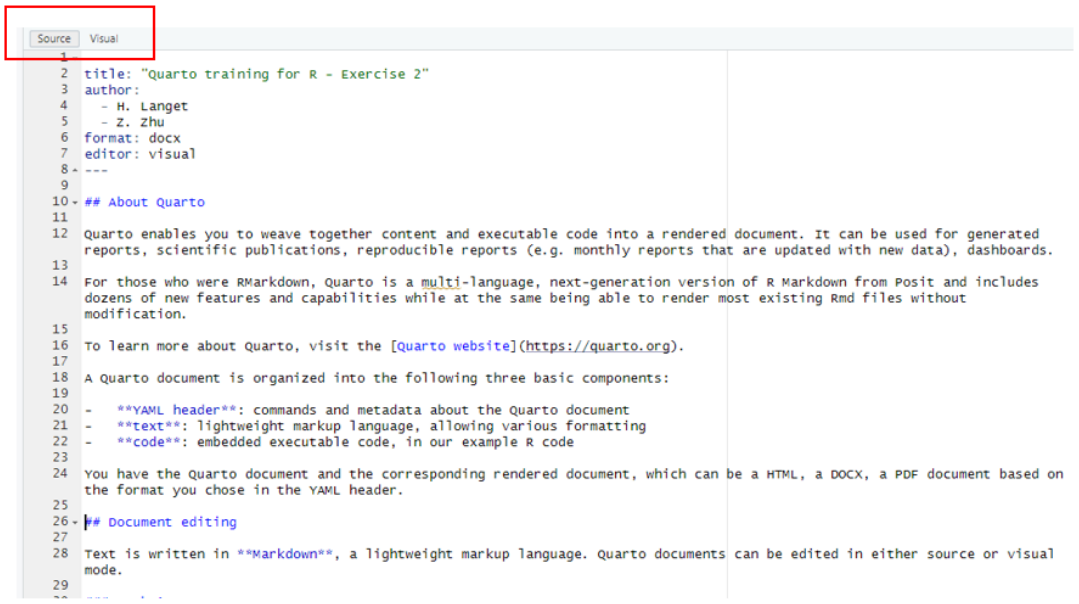
Useful resources

- Quarto Documentation - Visual Editing in RStudio
- Quarto Documentation - Markdown Basics

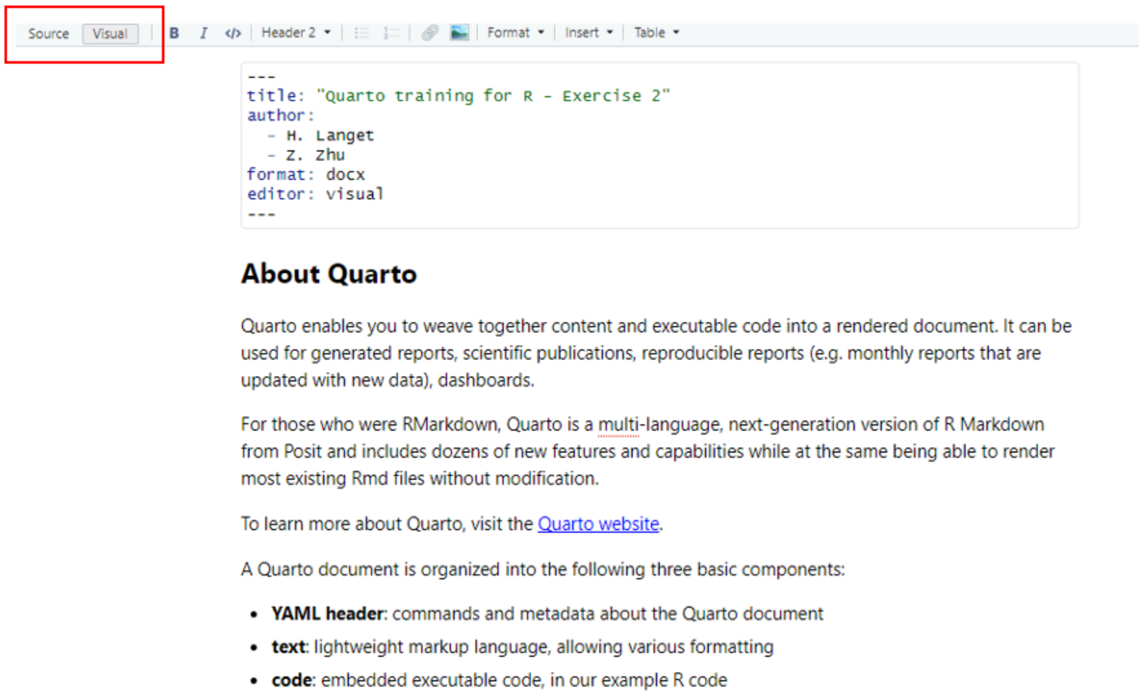
The **RStudio visual editor** is a relatively new feature designed to improve the editing experience by providing an intuitive interface. In the Visual Editor, you can preview your document in a format that closely resembles its final rendered appearance, similar to working in a "*What You See Is What You Get*" (WYSIWYG) languages, such as Microsoft Word, allowing for seamless content creation and editing.

This contrasts with the **RStudio source editor**, where content is written in **Markdown** syntax. The ability to switch between these two modes allows for flexibility, depending on your preference or task.

Below is an example of the same file viewed in both the source Editor and the visual Editor:



```
title: "Quarto training for R - Exercise 2"
author:
  - H. Langet
  - Z. Zhu
format: docx
editor: visual
---

## About Quarto

Quarto enables you to weave together content and executable code into a rendered document. It can be used for generated
reports, scientific publications, reproducible reports (e.g. monthly reports that are updated with new data), dashboards.

For those who were RMarkdown, Quarto is a multi-language, next-generation version of R Markdown from Posit and includes
dozens of new features and capabilities while at the same being able to render most existing Rmd files without
modification.

To learn more about Quarto, visit the [Quarto website](https://quarto.org).

A Quarto document is organized into the following three basic components:

-   **YAML header**: commands and metadata about the Quarto document
-   **text**: lightweight markup language, allowing various formatting
-   **code**: embedded executable code, in our example R code

You have the Quarto document and the corresponding rendered document, which can be a HTML, a DOCX, a PDF document based on
the format you chose in the YAML header.

## Document editing

Text is written in **Markdown**, a lightweight markup language. Quarto documents can be edited in either source or visual
mode.

```



```
---
title: "Quarto training for R - Exercise 2"
author:
  - H. Langet
  - Z. Zhu
format: docx
editor: visual
---
```

About Quarto

Quarto enables you to weave together content and executable code into a rendered document. It can be used for generated reports, scientific publications, reproducible reports (e.g. monthly reports that are updated with new data), dashboards.

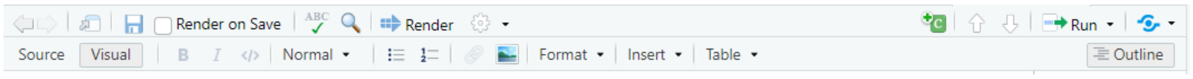
For those who were RMarkdown, Quarto is a multi-language, next-generation version of R Markdown from Posit and includes dozens of new features and capabilities while at the same being able to render most existing Rmd files without modification.

To learn more about Quarto, visit the Quarto website.

A Quarto document is organized into the following three basic components:

- **YAML header**: commands and metadata about the Quarto document
- **text**: lightweight markup language, allowing various formatting
- **code**: embedded executable code, in our example R code

In the Visual Editor, the toolbar inclues the most commonly used formatting commands:



In the menus you can find available options. For example, in **Format**, you can make text **Bold**, *Italic*, or underline; in **Insert**, you can insert a code chunk with available language options or a YAML block easily; and **Table** create much convenience for inserting a table.

3 Running R code in notebook mode

To write and execute code in Quarto, you will use **code chunks**.

Multilanguage

As the number of programming languages used for scientific discourse is very broad, Quarto was developed to be multilingual, beginning with R, Python, Observable JavaScript (OJS), and Julia, building on the RStudio (R) and Jupyter (Python, Julia) ecosystems which are very popular among data scientists. Stata is not a language supported by Quarto.

You can run **R code** in your Quarto document, by writing R commands within **code chunks** as is displayed below

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

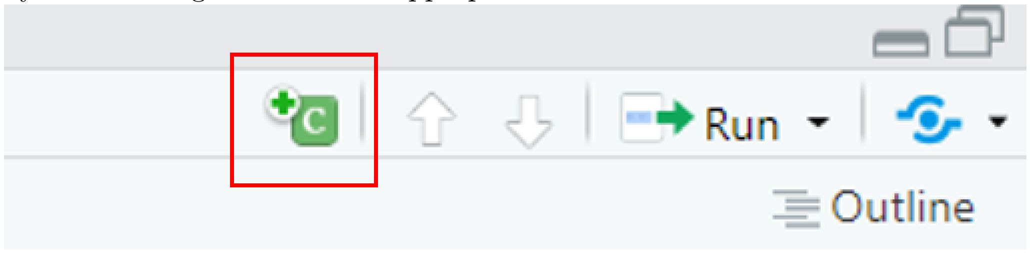
```
[1] 2
```

3.1 Create code chunks

Tip

Here are some tips for creating code chunks in RStudio

- **Backticks**: use three backticks to start and end a code chunk.
- **Toolbar icon**: you can also start a code chunk by clicking the appropriate icon in the toolbar.



- **Keyboard shortcut**: for a quicker method, use the keyboard shortcut `Ctrl` + `Alt` + `I`

3.2 Tables

Displaying data can be achieved with simple commands. For instance, to show the first 10 rows of the `iris` dataset, one can use:

~~~
```{r}
iris |>
  head(10)
```
~~~

```
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1           5.1         3.5          1.4         0.2  setosa
2           4.9         3.0          1.4         0.2  setosa
3           4.7         3.2          1.3         0.2  setosa
```

```
4          4.6        3.1         1.5         0.2  setosa
5          5.0        3.6         1.4         0.2  setosa
6          5.4        3.9         1.7         0.4  setosa
7          4.6        3.4         1.4         0.3  setosa
8          5.0        3.4         1.5         0.2  setosa
9          4.4        2.9         1.4         0.2  setosa
10         4.9        3.1         1.5         0.1  setosa
```

To improve the readability of tables, the `knitr::kable()` function provides a more structured and formatted output:

~~~
```{r}
iris |>
  head(10)  |>
  knitr::kable()
```
~~~

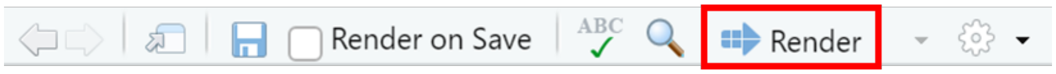
Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa

Tip

Additional R packages can be used for more advanced and aesthetically refined tables.

4 Rendering

When you click the **Render** button a document will be generated that includes both content and the output of embedded code.



In the case of this YAML header, the Quarto document will generate an HTML file because of the `format: html` setting.

> ⚠ Code errors
>
> If you attempt to render the document with code chunks that contain errors, the rendering process will fail. Therefore, it is essential to ensure that all code chunks run successfully in sequence before rendering the document.

4.1 Options

You can add options to **a specific code chunk** as is shown below:

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

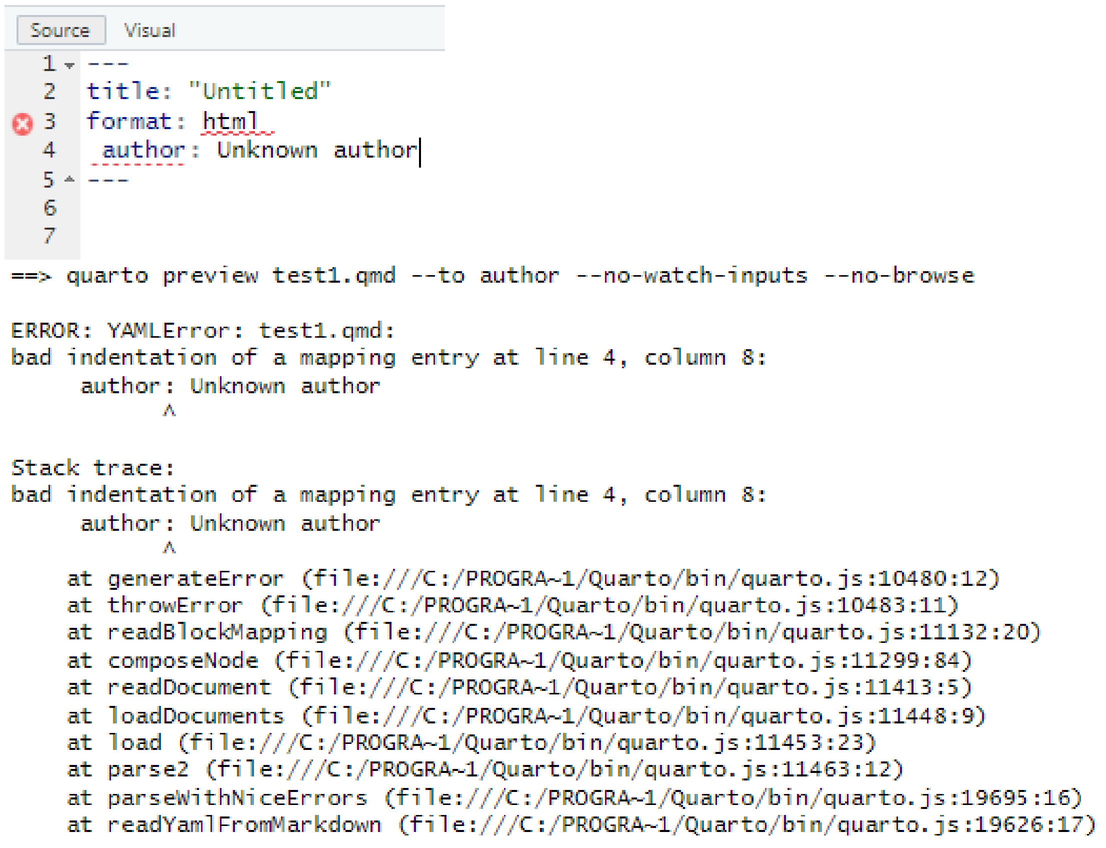
The `echo: false` option hides the code chunk in the rendered output document (only code output is displayed). In this example, rendered output document will only print:

```
[1] 4
```

Additionally, you can apply such options **globally** by specifying them in the YAML header, which configures settings for the entire Quarto document.

```
---
title: "Quarto training for R - Exercise 2"
author: Unknown author
format: html
editor: visual
execute:
  echo: true
---
```

> ! Important
>
> Indentation is essential for defining the structure of YAML contents. It is important to note that tabulations are not recognised as valid indentation, but the YAML language is whitespace-sensitive. The recommended practice is therefore to use **two spaces** per indentation level to ensure consistency and avoid errors.



```
1 ---
2 title: "Untitled"
3 format: html
4  author: Unknown author
5 ---
6
7
```

```
==> quarto preview test1.qmd --to author --no-watch-inputs --no-browse

ERROR: YAMLError: test1.qmd:
bad indentation of a mapping entry at line 4, column 8:
     author: Unknown author
           ^

Stack trace:
bad indentation of a mapping entry at line 4, column 8:
     author: Unknown author
           ^
    at generateError (file:///C:/PROGRA~1/Quarto/bin/quarto.js:10480:12)
    at throwError (file:///C:/PROGRA~1/Quarto/bin/quarto.js:10483:11)
    at readBlockMapping (file:///C:/PROGRA~1/Quarto/bin/quarto.js:11132:20)
    at composeNode (file:///C:/PROGRA~1/Quarto/bin/quarto.js:11299:84)
    at readDocument (file:///C:/PROGRA~1/Quarto/bin/quarto.js:11413:5)
    at loadDocuments (file:///C:/PROGRA~1/Quarto/bin/quarto.js:11448:9)
    at load (file:///C:/PROGRA~1/Quarto/bin/quarto.js:11453:23)
    at parse2 (file:///C:/PROGRA~1/Quarto/bin/quarto.js:11463:12)
    at parseWithNiceErrors (file:///C:/PROGRA~1/Quarto/bin/quarto.js:19695:16)
    at readYamlFromMarkdown (file:///C:/PROGRA~1/Quarto/bin/quarto.js:19626:17)
```

Here you may want to edit the title of your document.

4.2 MS Word outputs

You can also render a MS Word document by modifying the global format option in the YAML header to `format: docx`

```
---
title: "Quarto training for R - Exercise 2"
author: Unknown author
format: docx
editor: visual
---
```

4.3 PDF outputs

You can also render a PDF document by modifying the global format option in the YAML header to `format: pdf`

```
---
title: "Quarto training for R - Exercise 2"
author: Unknown author
format: pdf
editor: visual
---
```

While PDF documents are also able to be created, they require installing LaTeX, which can sometimes be complicated to install. `TinyTeX` is a custom LaTeX distribution that is easy to install with R. It is needed to compile R Markdown or Quarto documents to PDF.

Install the `tinytex` package, and install `TinyTeX` from the `tinytex` package:

~~~
```{r}
install.packages("tinytex")
tinytex::install_tinytex()
```
~~~

If specific LaTeX packages (e.g., `fancyhdr`, `lastpage`, `babel`, `tocbibind`, `worldflags`) or styles are missing, they can be installed using the following commands:

~~~
```{r}
tinytex::tlmgr_install("fancyhdr")
```
~~~

~~~
```{r}
tinytex::parse_install(
    text = "! LaTeX Error: File `ulem.sty' not found."
)
```
~~~

In order to avoid the warning *No hyphenation patterns were preloaded for the language French into the format.*, please consider installing the package `hyphen-french`

~~~
```{r}
tinytex::tlmgr_install("hyphen-french")
```
~~~

4.4 Execute R code within the text

Inline code allows you to execute code within markdown, e.g. to automatically use the most up-to-date computations in narrative. Quarto provides an inline code syntax that works across all three engines (Jupyter, Knitr and OJS).

The syntax for inline code is similar to code blocks, except you use a single tick (‘) rather than triple ticks (“‘), and you can use it in the middle of markdown. Here is an example storing the variable in a code block and then print it in a sentence using inline code:

~~~
```{r}
comment <- "AWESOME"
```
~~~

You can print the sentence like this:

```
This lecture is `{r} comment`!
```

The result of the commented code is the following output:

> This lecture is AWESOME!

Inline expressions are always evaluated when rendering and previewing `.qmd` files.

- Chunk Execution Order: Run code chunks in the correct order to avoid errors due to missing objects or incomplete definitions. Use clear and consistent chunk labels for cross-referencing.

5 References

- The Epidemiologist R Handbook
- Analytically reproducible documents