

Introduction & Setup

Session 1

Mickaël CANOUIL, Ph.D.^{1*}

Saturday, the 7th of February, 2026

Affiliations

¹Independent Contractor, Lille, France

ORCID

 Mickaël CANOUIL, Ph.D.: [0000-0002-3396-4549](https://orcid.org/0000-0002-3396-4549)

*Corresponding author: pro@mickael.canouil.dev

Table of Contents

1. Introduction	3
1.1. Session 1: Introduction & Setup	4
1.2. Workshop Objectives	4
1.3. Learning Objectives	4
2. What is Quarto?	5
2.1. Key Strengths	6
2.2. Reproducibility Benefits	7
2.3. What is Pandoc?	7
2.4. How does Quarto work?	8
2.5. A Guide to Quarto Versions	8
3. Getting started	9
3.1. Downloading and installing Quarto	10
3.2. Using GitHub Codespaces	10
3.3. Quarto: a command line interface	11
3.4. Checking Quarto is installed	12
4. Quarto Projects	14
4.1. Creating a new Quarto project	15
4.2. Quarto Projects: <code>default</code>	15
4.3. Quarto Projects: <code>website</code>	16
4.4. Quarto Projects: <code>blog</code>	16
4.5. Quarto Projects: <code>book</code>	17
4.6. Quarto Projects: <code>manuscript</code>	17
4.7. Writing with your favourite editor	18
4.7.1. POSITRON	18
4.7.2. VSCODE	18
4.7.3. JUPYTER	19
4.7.4. RSTUDIO	19
4.7.5. NEOVIM	20
4.7.6. TEXT-EDITOR	20
4.8. Writing using the Visual Editor	21
4.9. POSITRON	21
4.10. VSCODE	21
4.11. RSTUDIO	22
5. Hands-On Exercise: Creating Your First Quarto Project	23
5.1. Exercise Overview	24
5.2. Part 1: Project Creation	24
5.3. Part 2: IDE Features Exploration	24
5.4. Part 3: Document Customisation	24
5.5. Part 4: Rendering	25
5.6. Success Criteria	25

1. Introduction

- 1.1. Session 1: Introduction & Setup
- 1.2. Workshop Objectives
- 1.3. Learning Objectives

1.1. Session 1: Introduction & Setup

- Welcome & Overview
 - Workshop objectives and participant goals.
 - Quarto overview: purpose, strengths, and reproducibility benefits.
 - What Quarto is and how it works with Pandoc.
 - Brief look at the [Quarto Guide](#).
- Installation & Environment Setup
 - Installing Quarto (installer, Homebrew, Chocolatey).
 - Verify installation with `quarto check`.
 - Quarto as a command-line interface.
- Quarto Projects
 - Project types: default, website, blog, book, manuscript.
 - Creating projects with `quarto create project`.
 - Writing with various editors (VS Code, RStudio, Jupyter, etc.).

1.2. Workshop Objectives

- Familiarise participants with the Quarto ecosystem.
- Empower attendees to create reproducible documents and interactive publications.
- Guide learners through embedding code, adjusting outputs, and publishing projects.
- Encourage hands-on practice with interactive sessions.

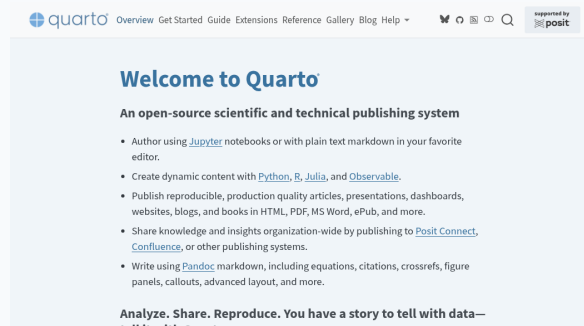
1.3. Learning Objectives

By the end of this session, participants will be able to:

- Explain what Quarto is and how it relates to Pandoc.
- Install Quarto and verify the installation.
- Create and render different Quarto project types.
- Set up an IDE for Quarto development.

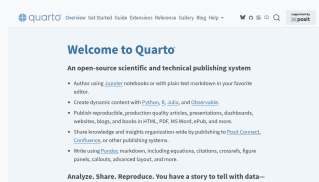
2. What is Quarto?

- 2.1. Key Strengths
- 2.2. Reproducibility Benefits
- 2.3. What is Pandoc?
- 2.4. How does Quarto work?
- 2.5. A Guide to Quarto Versions



Quarto is an open-source scientific and technical publishing system built on Pandoc:

- Multi-language platform supporting Python, R, Julia, and Observable.js.
- Creates dynamic, reproducible documents from code and narrative text.
- Single-source authoring for articles, websites, books, presentations, and dashboards.



Quarto is an open-source scientific and technical publishing system built on Pandoc:

- Multi-language platform supporting Python, R, Julia, and Observable.js.
- Creates dynamic, reproducible documents from code and narrative text.
- Single-source authoring for articles, websites, books, presentations, and dashboards.

2.1. Key Strengths

Multi-format output

Generate HTML, PDF, Word, ePub, and more from one source.

Advanced markdown

Cross-references, equations, citations, callouts, and complex layouts.

Tool flexibility

Works with VS Code, RStudio, JupyterLab, or any text editor.

Project system

Render document collections with shared configurations.

Interactive elements

Embed widgets, Shiny apps, and Observable JS for dynamic content.

2.2. Reproducibility Benefits

Embedded computation

Code and results stay together, eliminating copy-paste errors.

Version control friendly

Plain text format works seamlessly with Git.

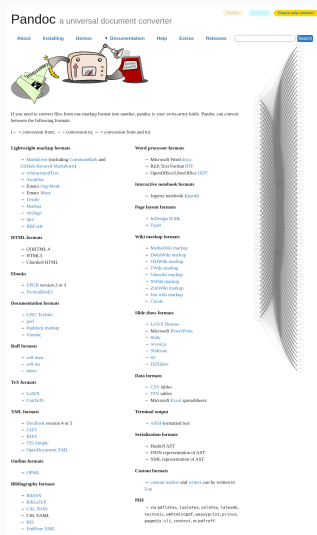
Transparent workflows

Complete audit trail from data to final output.

Collaborative research

Shared environments and dependencies ensure consistent results.

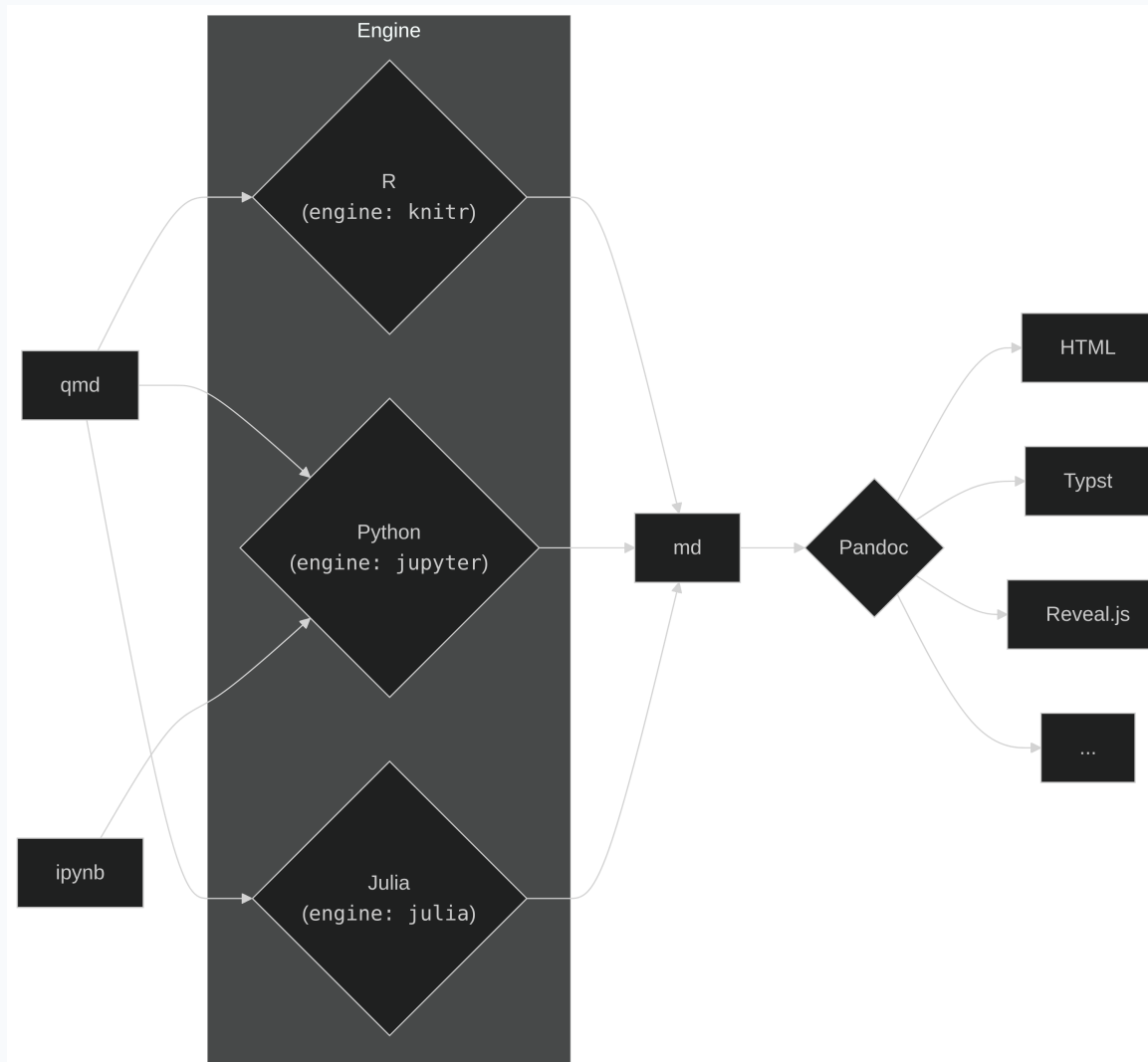
2.3. What is Pandoc?



Pandoc (pandoc.org):

- is a free software, released under the GPL by [John MacFarlane](#).
- is a command line interface (CLI).
- allows to convert to and between multiple formats.
- understands Markdown syntax.
- understands LaTeX math and macros.
- handles citations and bibliographies.

2.4. How does Quarto work?



2.5. A Guide to Quarto Versions

Quarto follows a versioning scheme similar to [Semantic Versioning](#) with three release types: Pre-release, Release Candidate (RC), and Release.

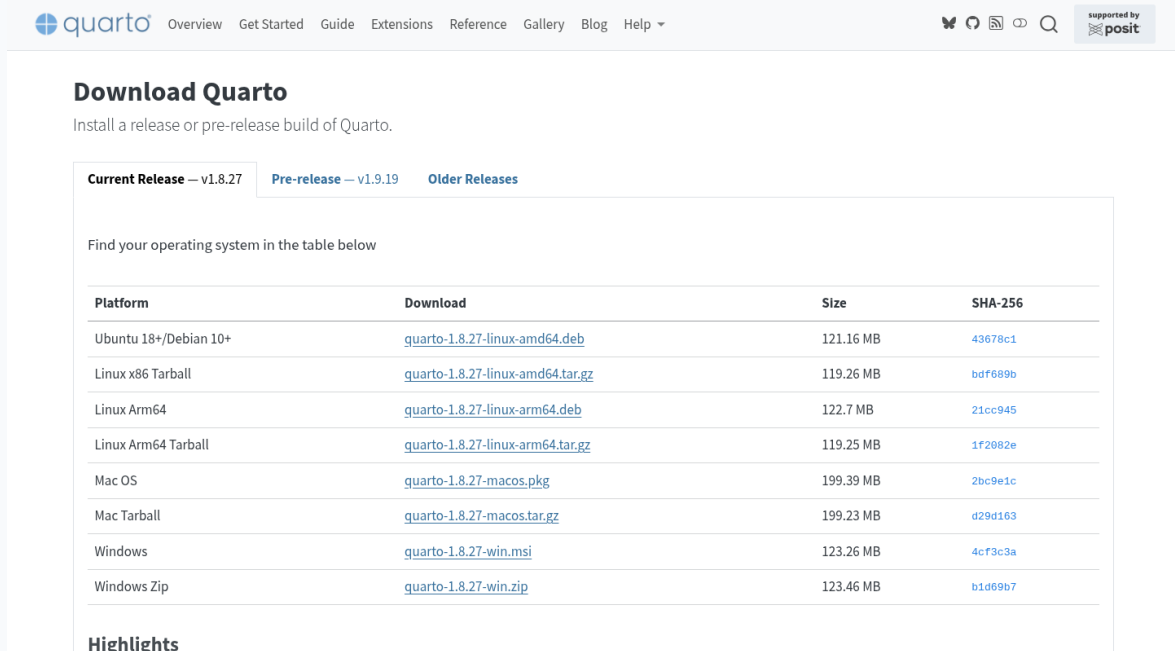
💡 Tip

Always use the latest stable release for production work. See the [Quarto Changelog](#) for the full release history.

3. Getting started

- 3.1. Downloading and installing Quarto
- 3.2. Using GitHub Codespaces
- 3.3. Quarto: a command line interface
- 3.4. Checking Quarto is installed

3.1. Downloading and installing Quarto



Download Quarto
Install a release or pre-release build of Quarto.

Current Release — v1.8.27 Pre-release — v1.9.19 Older Releases

Find your operating system in the table below

Platform	Download	Size	SHA-256
Ubuntu 18+/Debian 10+	quarto-1.8.27-linux-amd64.deb	121.16 MB	43678c1
Linux x86 Tarball	quarto-1.8.27-linux-amd64.tar.gz	119.26 MB	bdf689b
Linux Arm64	quarto-1.8.27-linux-arm64.deb	122.7 MB	21cc945
Linux Arm64 Tarball	quarto-1.8.27-linux-arm64.tar.gz	119.25 MB	1f2082e
Mac OS	quarto-1.8.27-macos.pkg	199.39 MB	2bc9e1c
Mac Tarball	quarto-1.8.27-macos.tar.gz	199.23 MB	d29d163
Windows	quarto-1.8.27-win.msi	123.26 MB	4cf3c3a
Windows Zip	quarto-1.8.27-win.zip	123.46 MB	b1d69b7

Highlights

<https://quarto.org/>

Additionally, you can download and install Quarto using:

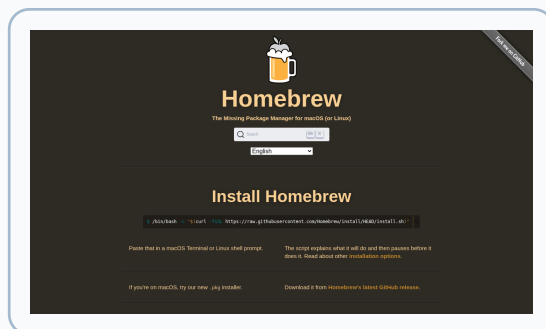
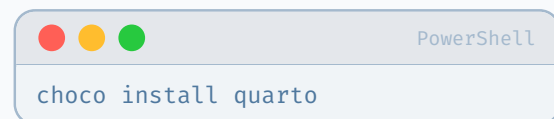
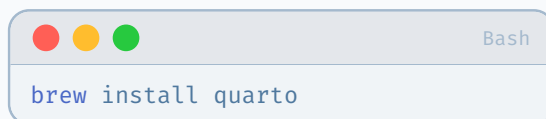


Figure 3.1: <https://brew.sh/>

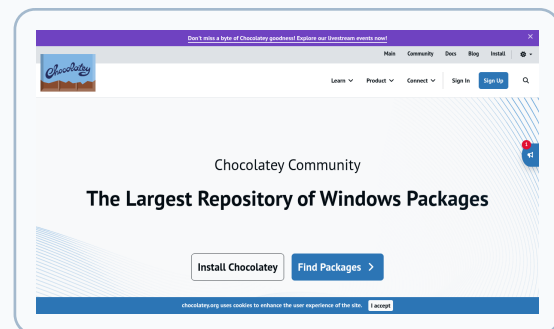


Figure 3.2: <https://community.chocolatey.org/>

3.2. Using GitHub Codespaces



A pre-configured development environment is available via GitHub Codespaces with all required tools pre-installed: Quarto, R, Python, Julia, TinyTeX, and supporting utilities.

3.3. Quarto: a command line interface



BASH

```
quarto --help
```

Usage: quarto

Version: 1.9.18

Description:

Quarto CLI

Options:

-h, --help - Show this help.
-V, --version - Show the version number for this program.

Commands:

render	[input] [args ...]	- Render files or projects to various document types.
preview	[file] [args ...]	- Render and preview a document or website project.
serve	[input]	- Serve a Shiny interactive document.
create	[type] [commands ...]	- Create a Quarto project or extension
use	<type> [target]	- Automate document or project setup tasks.
add	<extension>	- Add an extension to this folder or project
update	[target ...]	- Updates an extension or global dependency.
remove	[target ...]	- Removes an extension.
convert	<input>	- Convert documents to alternate representations.
pandoc	[args ...]	- Run the version of Pandoc embedded within Quarto.
typst	[args ...]	- Run the version of Typst embedded within Quarto.
run	[script] [args ...]	- Run a TypeScript, R, Python, or Lua script.
list	<type>	- Lists an extension or global dependency.
install	[target ...]	- Installs a global dependency (TinyTex or Chromium).
uninstall	[tool]	- Removes an

extension.	
tools	- Display the status of Quarto installed
dependencies	
publish [provider] [path]	- Publish a document or project to a
provider.	
check [target]	- Verify correct functioning of Quarto
installation.	
call	- Access functions of Quarto subsystems such
as its rendering engines.	
help [command]	- Show this help or the help of a sub-
command.	

3.4. Checking Quarto is installed

```
quarto check
```

```
Quarto 1.8.24
[✓] Checking environment information...
    Quarto cache location: /home/vscode/.cache/quarto
[✓] Checking versions of quarto binary dependencies...
    Pandoc version 3.6.3: OK
    Dart Sass version 1.87.0: OK
    Deno version 2.3.1: OK
    Typst version 0.13.0: OK
[✓] Checking versions of quarto dependencies.....OK
[✓] Checking Quarto installation.....OK
    Version: 1.8.24
    Path: /opt/quarto/bin ①

[✓] Checking tools.....OK
    TinyTeX: (external install)
    Chromium: (not installed) ②

[✓] Checking LaTeX.....OK
    Using: TinyTeX
    Path: /home/vscode/.TinyTeX/bin/x86_64-linux
    Version: 2025 ③

[✓] Checking Chrome Headless.....OK
    Using: Chrome found on system
    Path: /usr/bin/google-chrome
    Source: PATH ④
```

```
[✓] Checking basic markdown render....OK

[✓] Checking Python 3 installation....OK
  Version: 3.13.7
  Path: /usr/local/python/current/bin/python3
  Jupyter: 5.8.1
  Kernels: julia-1.11, python3 ⑤

[✓] Checking Jupyter engine render....OK

[✓] Checking R installation.....OK
  Version: 4.5.1
  Path: /opt/R/4.5.1/lib/R
  LibPaths:
    - /home/vscode/R/x86_64-pc-linux-gnu-library/4.5
    - /opt/R/4.5.1/lib/R/library
  knitr: 1.50
  rmarkdown: 2.29 ⑥

[✓] Checking Knitr engine render.....OK
```

- ① Quarto version and dependencies.
- ② Tools installed by Quarto (e.g., `quarto install`).
- ③ LaTeX installation detected by Quarto.
- ④ Chromium-based browser installation detected by Quarto.
- ⑤ Python installation and computing dependencies detected by Quarto.
- ⑥ R installation and computing dependencies detected by Quarto.

4. Quarto Projects

- 4.1. Creating a new Quarto project
- 4.2. Quarto Projects: `default`
- 4.3. Quarto Projects: `website`
- 4.4. Quarto Projects: `blog`
- 4.5. Quarto Projects: `book`
- 4.6. Quarto Projects: `manuscript`
- 4.7. Writing with your favourite editor
 - 4.7.1. POSITRON
 - 4.7.2. VSCODE
 - 4.7.3. JUPYTER
 - 4.7.4. RSTUDIO
 - 4.7.5. NEOVIM
 - 4.7.6. TEXT-EDITOR
- 4.8. Writing using the Visual Editor
- 4.9. POSITRON
- 4.10. VSCODE
- 4.11. RSTUDIO

4.1. Creating a new Quarto project

```
Bash
quarto create project
```

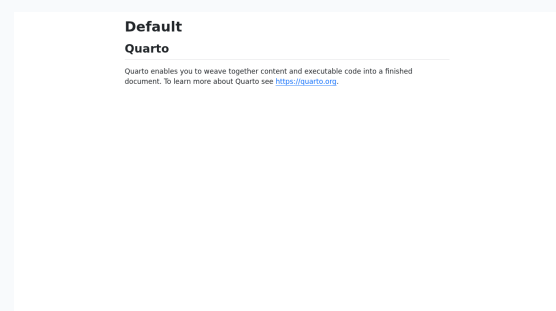
```
BASH
? Type
> default
  website
  blog
  manuscript
  book
  confluence
```

4.2. Quarto Projects: default

```
Bash
quarto create project default
Default
```

```
_demo/Default
├─ Default.qmd
└─ _quarto.yml

0 directories, 2 files
```



- `_quarto.yml` : project configuration file.

```
Quarto
project:
  title: "Default"
```

- `Default.qmd` : default Quarto document.

```
Quarto
---
title: "Default"
---

## Quarto
```

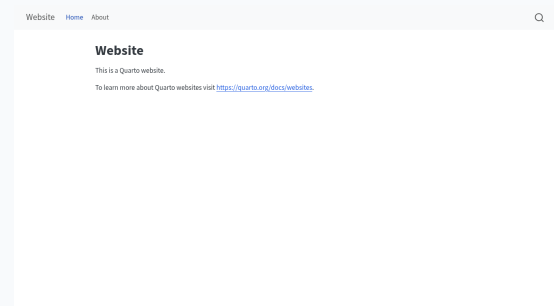
Quarto enables you to weave together content and executable code into a finished document. To learn more about Quarto see <https://quarto.org>.

4.3. Quarto Projects: website

```
Bash
quarto create project website
Website
```

```
_demo/Website
├─ _quarto.yml
├─ about.qmd
├─ index.qmd
└─ styles.css

0 directories, 4 files
```

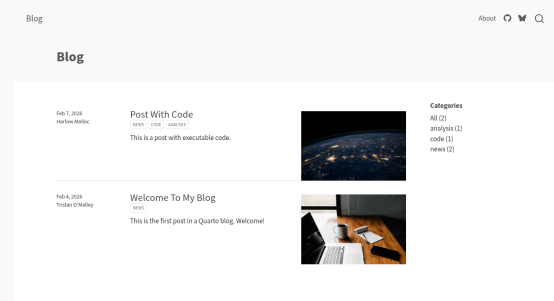


4.4. Quarto Projects: blog

```
Bash
quarto create project blog Blog
```

```
_demo/Blog
├─ _quarto.yml
├─ about.qmd
├─ index.qmd
├─ posts
│   ├── _metadata.yml
│   ├── post-with-code
│   │   ├── image.jpg
│   │   └─ index.qmd
│   └─ welcome
│       ├── index.qmd
│       └─ thumbnail.jpg
├─ profile.jpg
└─ styles.css

3 directories, 10 files
```

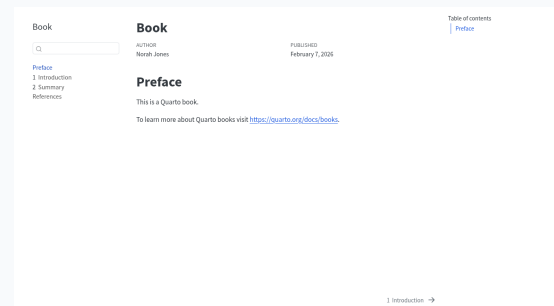


4.5. Quarto Projects: book

```
Bash
quarto create project book Book

_demo/Book
├─ _quarto.yml
├─ cover.png
├─ index.qmd
├─ intro.qmd
├─ references.bib
├─ references.qmd
└-- summary.qmd

0 directories, 7 files
```

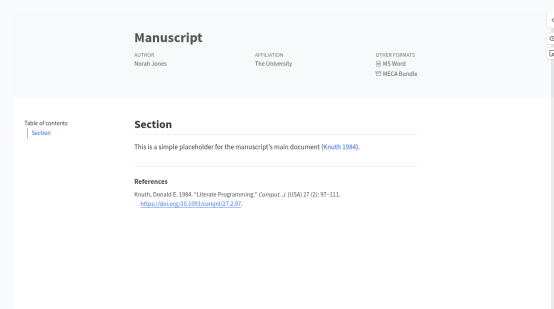


4.6. Quarto Projects: manuscript

```
Bash
quarto create project manuscript
Manuscript

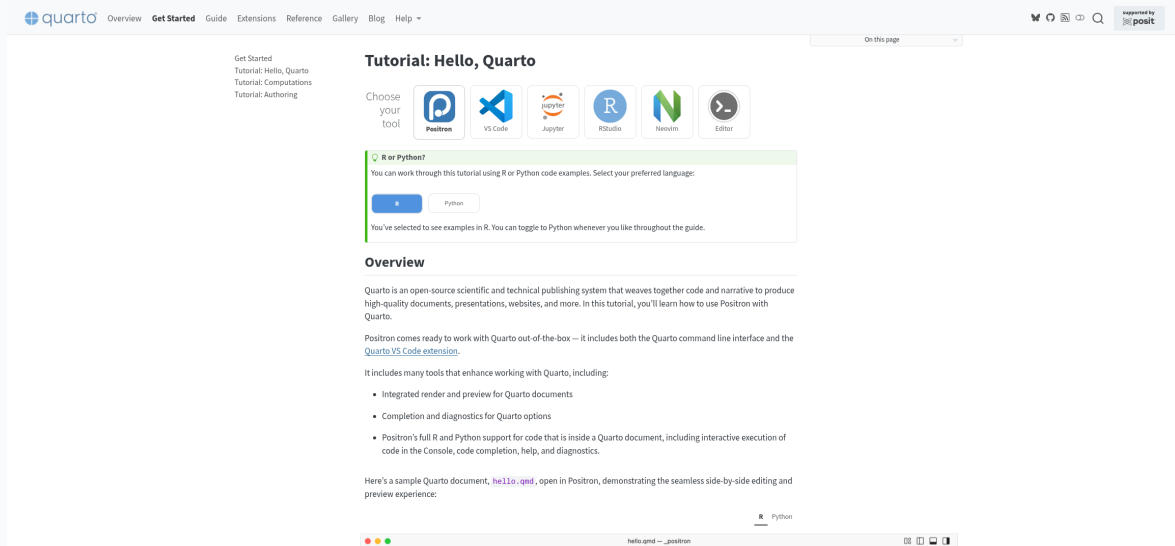
_demo/Manuscript
├─ _quarto.yml
├─ index.qmd
└-- references.bib

0 directories, 3 files
```

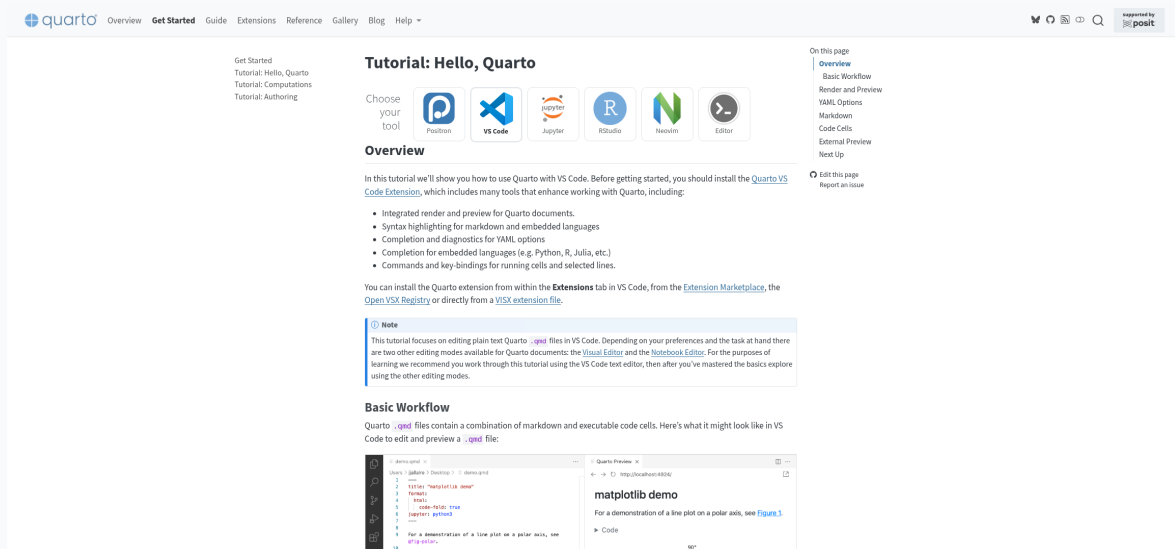


4.7. Writing with your favourite editor

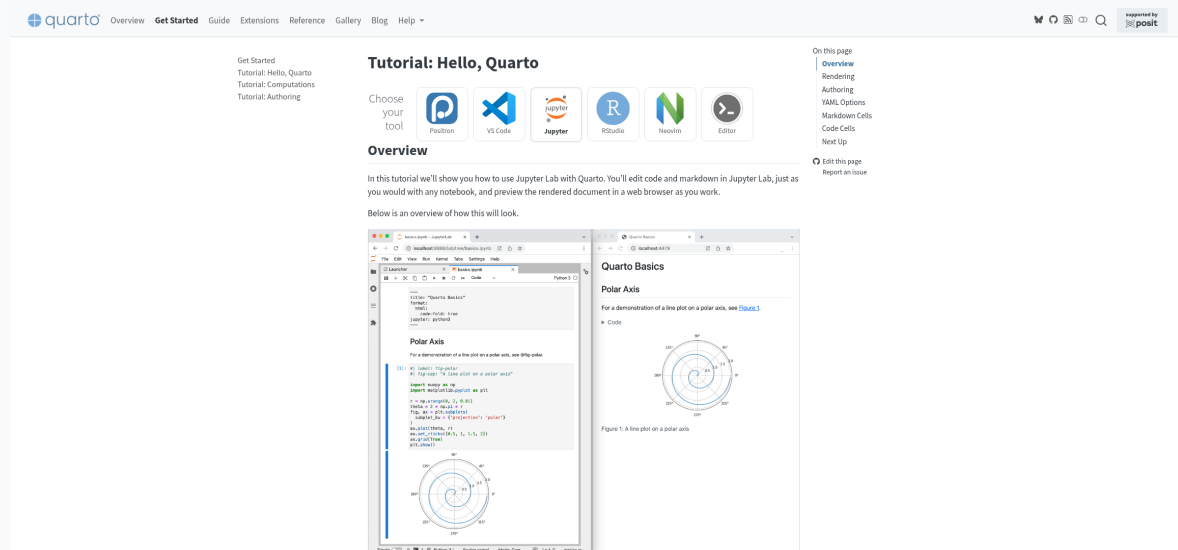
4.7.1. POSITRON



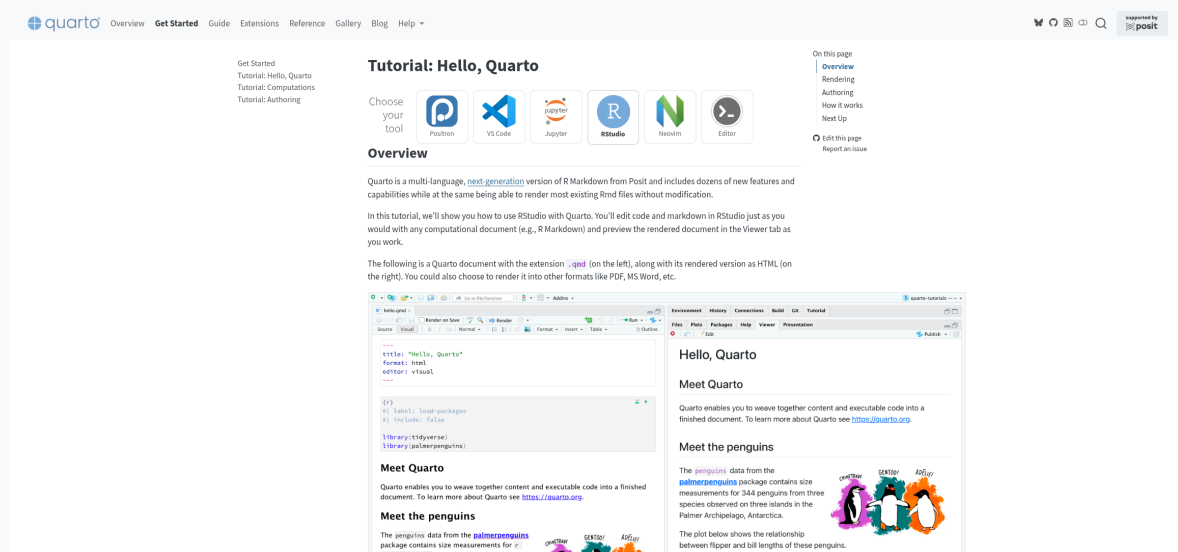
4.7.2. VSCODE



4.7.3. JUPYTER



4.7.4. RSTUDIO



4.7.5. NEOVIM

quarto

OverviewGet StartedGuideExtensionsReferenceGalleryBlogHelp

supported by

posit

Get Started

Tutorial: Hello, Quarto

Tutorial: Computations

Tutorial: Authoring

Tutorial: Hello, Quarto

Choose your tool

Positron

VS Code

Jupyter

RStudio

Neovim

Editor

Overview

In this tutorial we'll show you how to use Quarto with Neovim. While we cover the basics here, you will also want to review the article on [Using Neovim with Quarto](#) to learn more about installing, using, and customizing Neovim for Quarto.

If you already have Neovim configured to your liking, you may only want to add the [quarto-nvim](#) plugin and only refer to this guide for inspiration and seeing the possibilities. But if you are entirely new to Neovim or want to simply try out a configuration already set up for data science with Quarto, you should head over to this [kickstarter configuration](#). This is also what we will be using for this tutorial.

Note

Neovim is a highly customizable editor. So much so that Neovim core member TJ Devries has recently coined the term Personal Development Environments (PDEs) to separate the concept from Integrated Development Environments (IDEs) such as VS Code and RStudio.

Out of the box neovim is fairly minimal. To work efficiently and get all the nice features, you have to configure it. You have to make it your own. If this approach sounds enticing to you, read on. Welcome to the rabbit hole. :)

You can also watch [this video](#) for a quick guide to getting started with the kickstarter configuration alongside this write-up.

A coffee with quarto and neovim - part 1

Quarto Nvim Kickstarter

On this page

Overview

Basic Workflow

Render and Preview

YAML Options

Code Cells

Next Up

Next Up

Edit this page

Report an issue

4.7.6. TEXT-EDITOR

quarto

OverviewGet StartedGuideExtensionsReferenceGalleryBlogHelp

supported by

posit

Get Started

Tutorial: Hello, Quarto

Tutorial: Computations

Tutorial: Authoring

Tutorial: Hello, Quarto

Choose your tool

Positron

VS Code

Jupyter

RStudio

Neovim

Editor

Overview

In this tutorial we'll show you how to use your favorite text editor with Quarto. You'll edit plain text `.qmd` files and preview the rendered document in a web browser as you work.

Below is an overview of how this will look.

quarto

Quarto Basics

Polar Axis

For a demonstration of a line plot on a polar axis, see [Figure 1](#).

Code

Figure 1: A line plot on a polar axis

On this page

Overview

Editor Modes

Rendering

Authoring

YAML Options

Markdown

Code Cells

Next Up

Next Up

Edit this page

Report an issue

<https://quarto.org/>

4.8. Writing using the Visual Editor

4.9. POSITRON

The screenshot shows the Quarto Positron Visual Editor interface. On the left is a sidebar with a navigation menu including Guide, Authoring, Computations, Tools, JupyterLab, RStudio IDE, Positron, Positron Basics, Visual Editor, and Notebook Editor. The main content area is titled "Visual Editing in Positron" and "Overview". It explains that the Quarto VS Code Extension includes a visual markdown editor that supports all of Quarto's markdown syntax including tables, citations, cross-references, footnotes, divs/spans, definition lists, attributes, raw HTML/TeX, and more. A code editor shows a snippet of Quarto markdown: `---
title: "Libraries"
format: html
execute:
 echo: fenced
---`. Below the code editor, the "Overview" section states there are three types of library you'll generally use with OJS: 1. [Observable core libraries](#) automatically available in every document. 2. Third-party JavaScript libraries from [npm](#) and [ObservableHQ](#). 3. Custom libraries you and/or your colleagues have created. It also mentions that in this document, a high-level overview of the core libraries and some examples of using third-party libraries ([D3](#) and [Arquero](#)) are provided, and that creating your own libraries is covered in the article on [Code Reuse](#). At the bottom, it notes that you can switch between visual and source mode at any time and can even edit documents concurrently in both modes.

4.10. VSCODE

The screenshot shows the Quarto VS Code Visual Editor interface. On the left is a sidebar with a navigation menu including Guide, Authoring, Computations, Tools, JupyterLab, RStudio IDE, Positron, VS Code, VS Code Basics, Visual Editor, and Notebook Editor. The main content area is titled "Visual Editing in VS Code" and "Overview". It explains that the Quarto VS Code Extension includes a visual markdown editor that supports all of Quarto's markdown syntax including tables, citations, cross-references, footnotes, divs/spans, definition lists, attributes, raw HTML/TeX, and more. A code editor shows a snippet of Quarto markdown: `---
title: "Libraries"
format: html
---`. Below the code editor, the "Overview" section states there are three types of library you'll generally use with OJS: 1. [Observable core libraries](#) automatically available in every document. 2. Third-party JavaScript libraries from [npm](#) and [ObservableHQ](#). 3. Custom libraries you and/or your colleagues have created. It also mentions that in this document, a high-level overview of the core libraries and some examples of using third-party libraries ([D3](#) and [Arquero](#)) are provided, and that creating your own libraries is covered in the article on [Code Reuse](#). At the bottom, it notes that you can switch between visual and source mode at any time and can even edit documents concurrently in both modes. It also provides instructions: 1. Use the `⇧ F4` keyboard shortcut. 2. Use the context menu from anywhere in a document.

4.11. RSTUDIO

The screenshot displays the Quarto Visual Editor interface. The top navigation bar includes the Quarto logo and links for Overview, Get Started, Guide, Extensions, Reference, Gallery, Blog, and Help. The left sidebar contains a table of contents with categories like Guide, Position, Documents, and Manuscripts. The main content area is titled 'Visual Editing in RStudio' and includes an 'Overview' section. The overview text states: 'The Quarto visual editor provides a WYSIWYM editing interface for all of Pandoc markdown, including tables, citations, cross-references, footnotes, divs/spans, definition lists, attributes, raw HTML/TeX, and more. The visual editor also includes support for executing code cells and viewing their output inline:'. Below this text is a code editor showing a R Markdown snippet for 'Filtering Joins'. The snippet includes a title, a description of filtering joins, a table of join types, a graphical representation of a semi-join, and a code cell with R code and a plot. The table of join types is as follows:

Join Type	Condition	Description
semi_join(x, y)	$x \approx y$	Keeps all observations in x that have a match in y
anti_join(x, y)	$x \not\approx y$	Drops all observations in x that have a match in y

The graphical representation shows two Venn diagrams. The left diagram shows the intersection of two sets, x and y , with the intersection shaded. The right diagram shows the same two sets, but the intersection is unshaded, representing a semi-join. Below the diagrams is a table with the following data:

key	val_x
1	x1
2	x2

<https://quarto.org/>

5. Hands-On Exercise: Creating Your First Quarto Project

- 5.1. Exercise Overview
- 5.2. Part 1: Project Creation
- 5.3. Part 2: IDE Features Exploration
- 5.4. Part 3: Document Customisation
- 5.5. Part 4: Rendering
- 5.6. Success Criteria

5.1. Exercise Overview

Objective: Create a personal Quarto document that demonstrates basic features, project setup skills, and IDE integration

Example Code: [Exercises](#)

```
tar -xzf "01-exercises.tar.gz" -C "01-introduction-setup"
```

💡 IDE Configuration

Before starting, ensure your development environment is ready:

- VS Code: Install the Quarto extension (`Ctrl/Cmd + Shift + X` , search “Quarto”).
- Positron: Built-in Quarto support is included (no additional extensions needed).
- Open the Command Palette (`Ctrl/Cmd + Shift + P`) and type “Quarto” to verify available commands.

5.2. Part 1: Project Creation

1. Choose Your Adventure: Create a project using the CLI or your IDE.
2. Explore the project structure in the IDE's file explorer panel.

5.3. Part 2: IDE Features Exploration

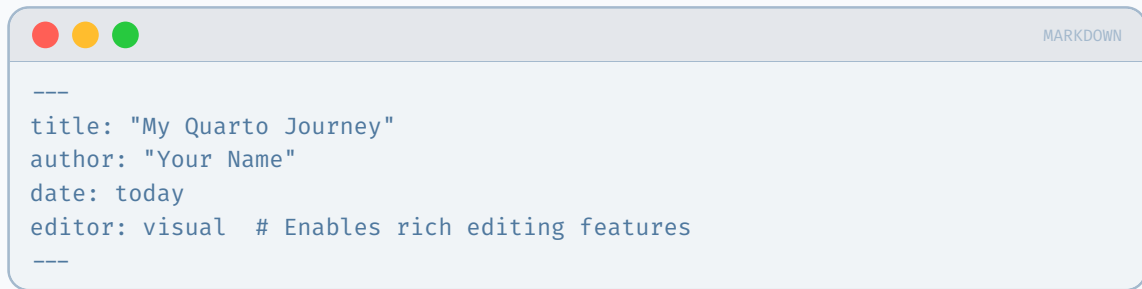
Discover your IDE's Quarto capabilities:

- File recognition: Notice `.qmd` files have distinctive icons and syntax highlighting.
- Document outline: Open a `.qmd` file and check the Outline/Structure panel (shows headings and sections).
- Quarto assist panel: Look for a Quarto-specific panel that provides contextual help.
- Command palette power: `Ctrl/Cmd + Shift + P` → type “Quarto” to see all available commands.
- Integrated features: Visual editor mode, preview panels, terminal integration.
- Smart editing: Try auto-completion in YAML headers, fenced divs, and code blocks.

5.4. Part 3: Document Customisation

Modify the main document (`index.qmd`) to include:

1. Update the YAML header with your information:



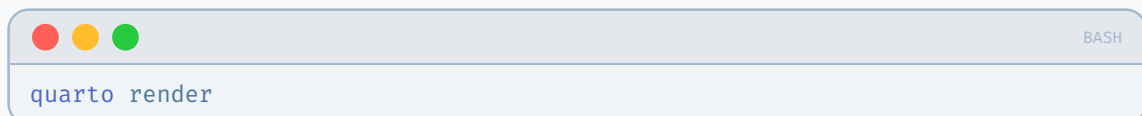
```
---  
title: "My Quarto Journey"  
author: "Your Name"  
date: today  
editor: visual # Enables rich editing features  
---
```

2. Add content to the document body.

5.5. Part 4: Rendering

Try different rendering approaches:

1. Command Line (classic method):



```
quarto render
```

2. IDE Integration (modern workflow):
 - Command Palette (`Ctrl/Cmd + Shift + P`) - search for “Quarto Preview” and/or “Quarto Render” commands.
 - Use keyboard shortcut if you configured one
 - Notice IDE-specific feedback and error handling

5.6. Success Criteria

- ☒ You've successfully completed the exercise if you can:
- Configure your IDE for optimal Quarto development.
 - Create a new Quarto project using the CLI or your IDE.
 - Navigate project files efficiently in your IDE.
 - Utilise IDE features like syntax highlighting and command palette.
 - Modify documents with IDE assistance (auto-completion, outline view).
 - Render documents using both CLI and IDE methods.