

Introduction & Setup

Session 1

Mickaël CANOUIL, *Ph.D.* 

pro@mickael.canouil.dev

Independent Contractor

Saturday, the 7th of February, 2026

- About Me
- Session 1: Introduction & Setup
- Workshop Objectives
- Learning Objectives

Introduction

About Me



- 🏠 mickael.canouil.fr
- ✉ pro@mickael.canouil.dev
- 🐙 [@mcanouil](#)

- 🔗 [mickaelcanouil](#)
- 🦋 [@mickael.canouil.fr](#)
- 📧 @MickaelCanouil@fosstodon.org

- *I am Mickaël CANOUIL, I hold a Ph.D. in Biostatistics, with over a decade of academic experience in the genetics of type 2 diabetes and obesity.*
- *My research has contributed to understanding the genetic and molecular mechanisms underlying metabolic diseases, with publications in leading journals.*
- *Currently, I work as a consultant in biostatistics, applying my expertise to diverse projects in multi-omics and data analysis.*
- *I am also deeply involved in the Quarto ecosystem, developing extensions and tools that enhance reproducibility and scientific communication.*
- *My contributions, including [Quarto Wizard](#) and [various Quarto extensions](#), aim to streamline workflows for researchers and data scientists.*
- *You can explore my work on [GitHub](#), [my projects](#), and [my publications](#).*

Session 1: Introduction & Setup

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](#).

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.

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](#).

- **Quarto Projects**

- Project types: default, website, blog, book, manuscript.
- Creating projects with `quarto create project`.
- Writing with various editors (VS Code, RStudio, Jupyter, etc.).

- **Installation & Environment Setup**

- Installing Quarto (installer, Homebrew, Chocolatey).
- Verify installation with `quarto check`.
- Quarto as a command-line interface.

Workshop Objectives

Workshop Objectives

- **Familiarise** participants with the Quarto ecosystem.

Workshop Objectives

- **Familiarise** participants with the Quarto ecosystem.
- **Empower** attendees to create reproducible documents and interactive publications.

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.

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.

Learning Objectives

Learning Objectives

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

Learning Objectives

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

- Explain what Quarto is and how it relates to Pandoc.

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.

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.

Learning Objectives

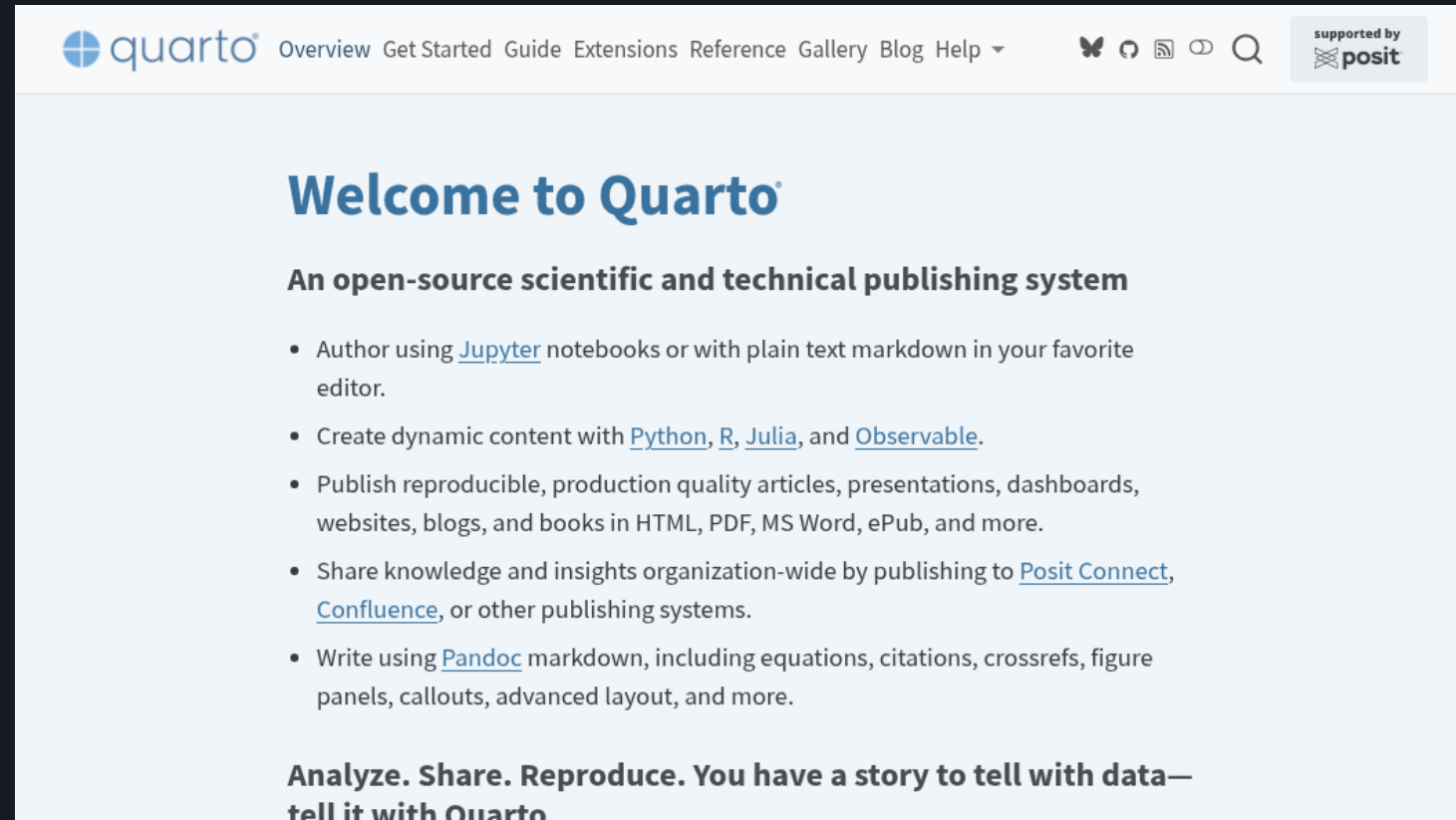
By the end of this session, participants will be able to:

- 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.

- What is Quarto?
- Purpose
- Key Strengths
- Reproducibility Benefits
- What is Pandoc?
- How does Quarto work?
- A Guide to Quarto Versions

What is Quarto? 🤔

What is Quarto? 🤔



The screenshot shows the Quarto website homepage. At the top is a navigation bar with the Quarto logo, links for Overview, Get Started, Guide, Extensions, Reference, Gallery, Blog, and Help, and a search icon. A 'supported by posit' badge is also present. The main content area features the heading 'Welcome to Quarto' followed by the subtitle 'An open-source scientific and technical publishing system'. Below this is a bulleted list of features: authoring with Jupyter or plain text, creating dynamic content with Python, R, Julia, and Observable, publishing reproducible articles in various formats, sharing knowledge via Posit Connect or Confluence, and writing with Pandoc for equations and citations. The page concludes with the tagline 'Analyze. Share. Reproduce. You have a story to tell with data—tell it with Quarto.'

quarto® Overview Get Started Guide Extensions Reference Gallery Blog Help 🔍 supported by **posit**

Welcome to Quarto

An open-source scientific and technical publishing system

- Author using [Jupyter](#) notebooks or with plain text markdown in your favorite editor.
- Create dynamic content with [Python](#), [R](#), [Julia](#), and [Observable](#).
- Publish reproducible, production quality articles, presentations, dashboards, websites, blogs, and books in HTML, PDF, MS Word, ePub, and more.
- Share knowledge and insights organization-wide by publishing to [Posit Connect](#), [Confluence](#), or other publishing systems.
- Write using [Pandoc](#) markdown, including equations, citations, crossrefs, figure panels, callouts, advanced layout, and more.

Analyze. Share. Reproduce. You have a story to tell with data—tell it with Quarto.

Purpose

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

Purpose

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

- Multi-language platform supporting Python, R, Julia, and Observable.js.

Purpose

⊕ 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.

Purpose

⊕ 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.

Key Strengths

Key Strengths

Multi-format output

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

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.

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.

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.

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.

Reproducibility Benefits

Reproducibility Benefits

Embedded computation

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

Reproducibility Benefits

Embedded computation

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

Version control friendly

Plain text format works seamlessly with Git.

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.

Reproducibility Benefits

Embedded computation

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

Version control friendly

Plain text format works seamlessly with Git.

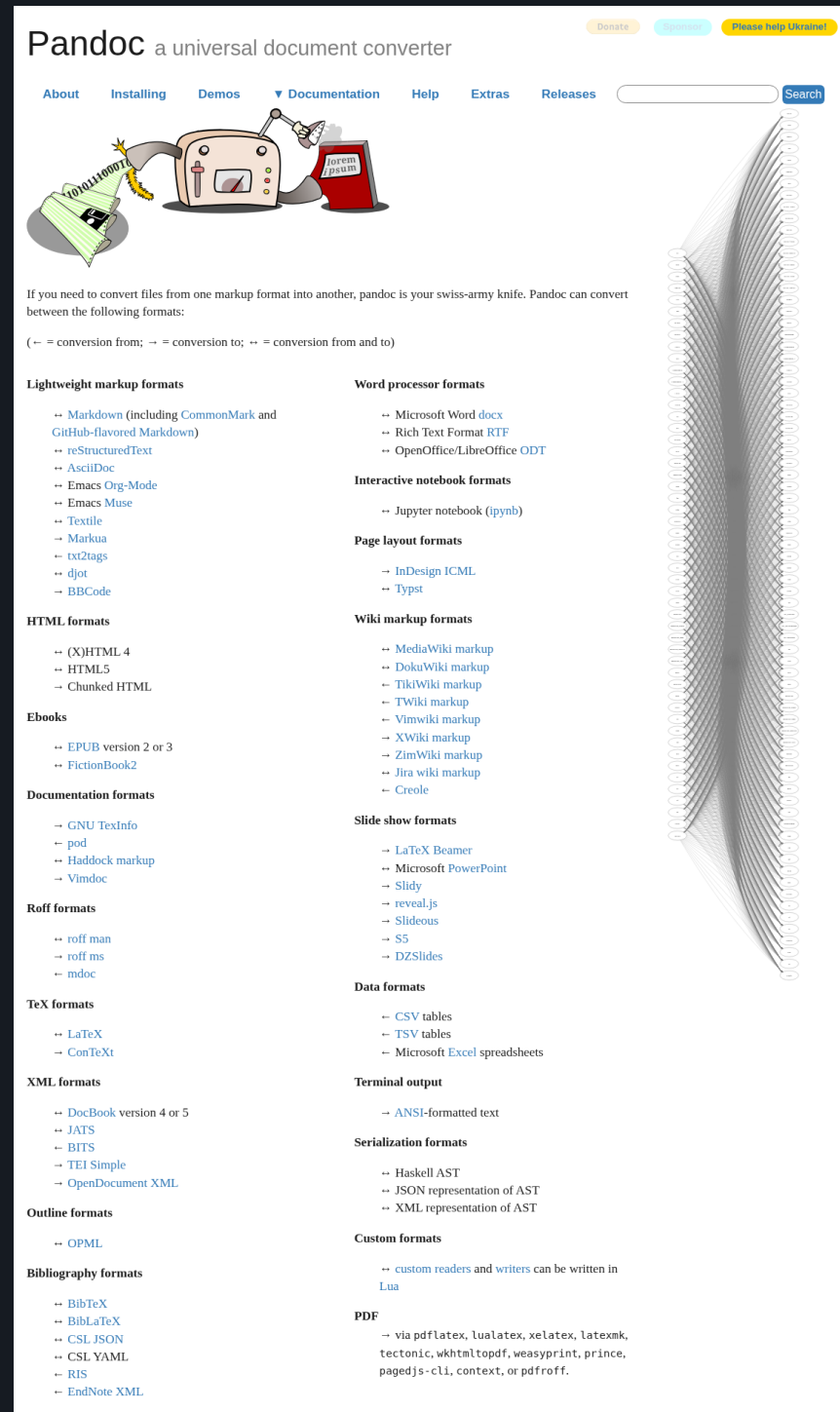
Transparent workflows

Complete audit trail from data to final output.

Collaborative research

Shared environments and dependencies ensure consistent results.

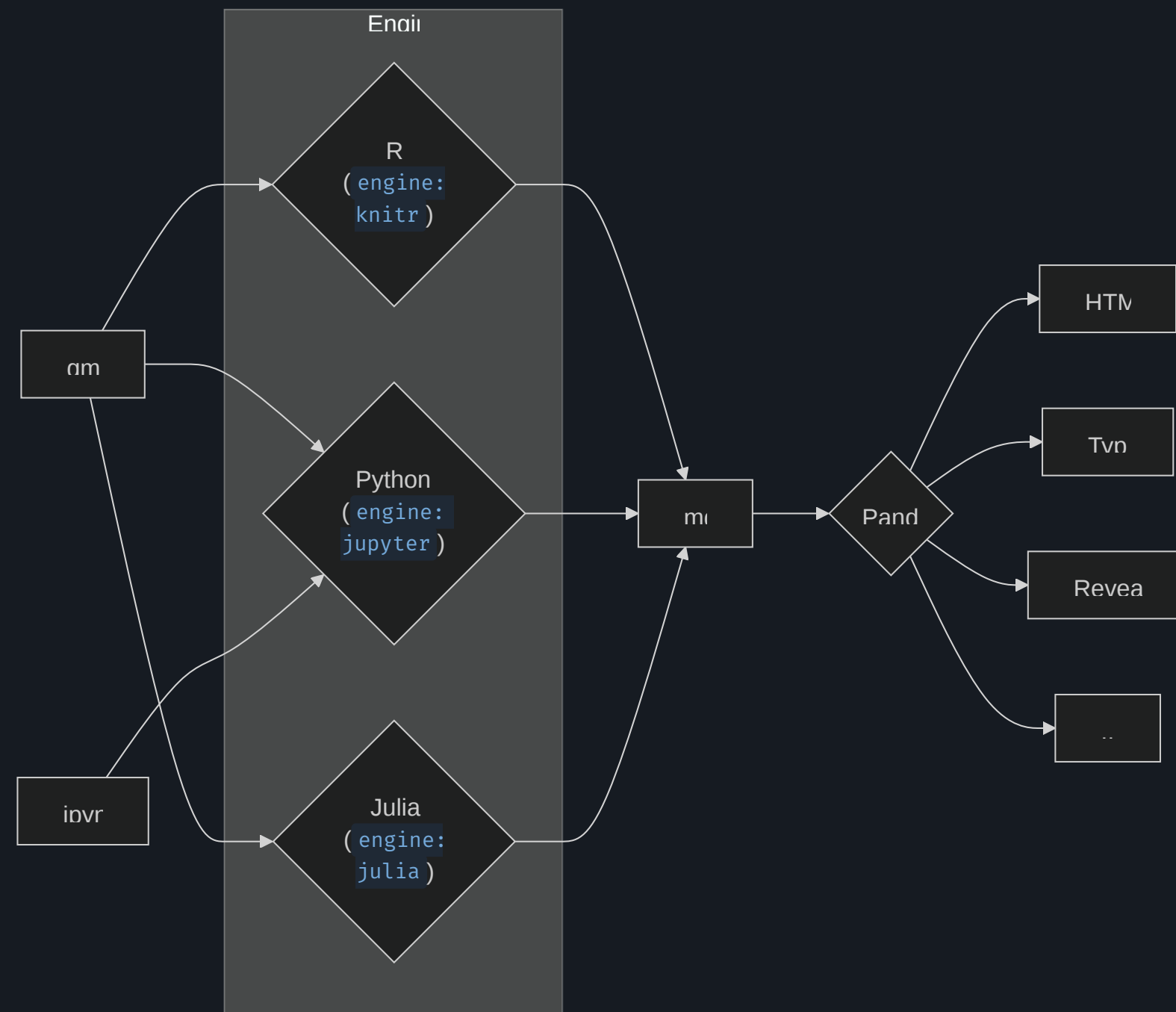
What is Pandoc? 🤔



Pandoc (pandoc.org):

- is a **free software**, released under the GPL by [John MacFarlane](https://pandoc.org).
- 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**.

How does Quarto work? 🤔



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.

- Downloading and installing Quarto
- Using GitHub Codespaces
- Quarto: a command line interface
- Checking Quarto is installed

Getting started

Downloading and installing Quarto

 Overview Get Started Guide Extensions Reference Gallery Blog Help ▾

 supported by 

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

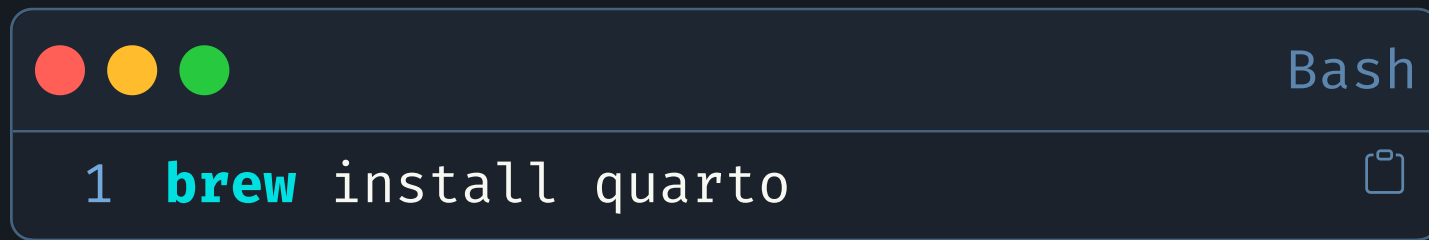
Platform	Download	Size	SHA-256
Ubuntu 18+/Debian 10+	quarto-1.8.27-linux-amd64.deb	121.16 MB	43678c1
Linux x86 Tarball	quarto-1.8.27-linux-amd64.tar.gz	119.26 MB	bdf689b
Linux Arm64	quarto-1.8.27-linux-arm64.deb	122.7 MB	21cc945
Linux Arm64 Tarball	quarto-1.8.27-linux-arm64.tar.gz	119.25 MB	1f2082e
Mac 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

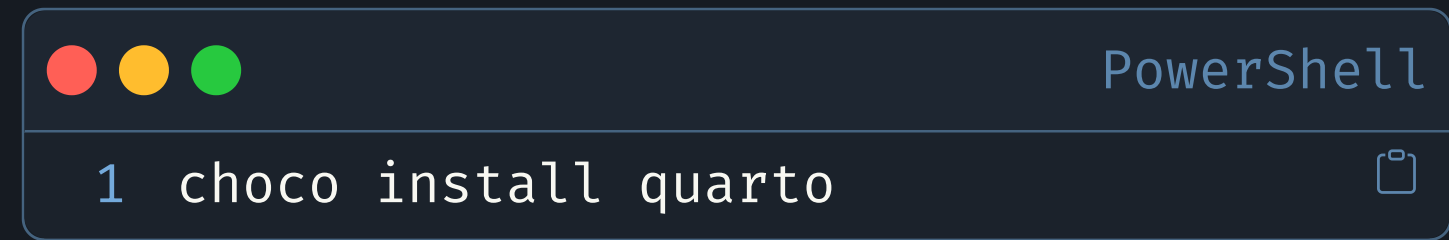
<https://quarto.org/>

Downloading and installing Quarto

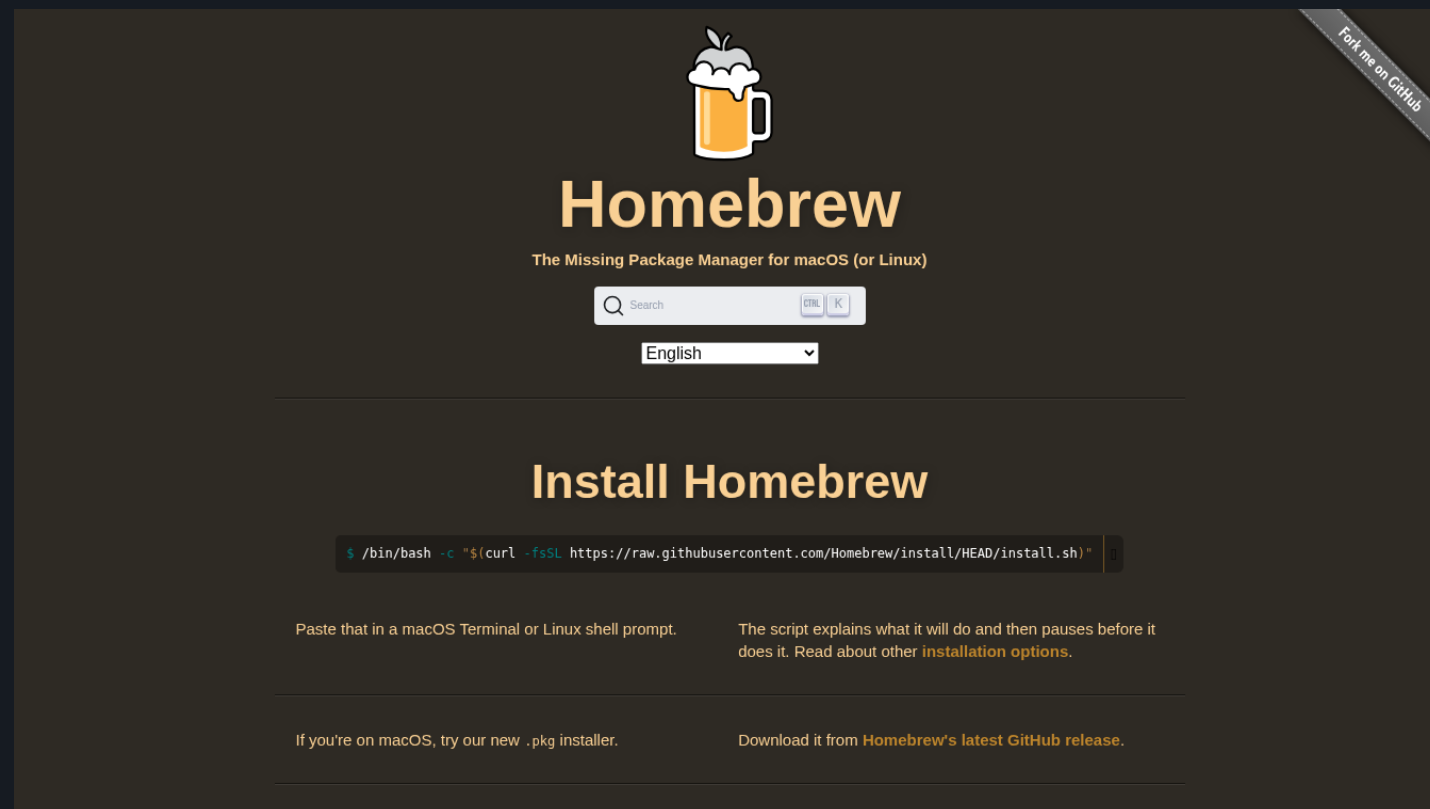
Additionally, you can download and install Quarto using:



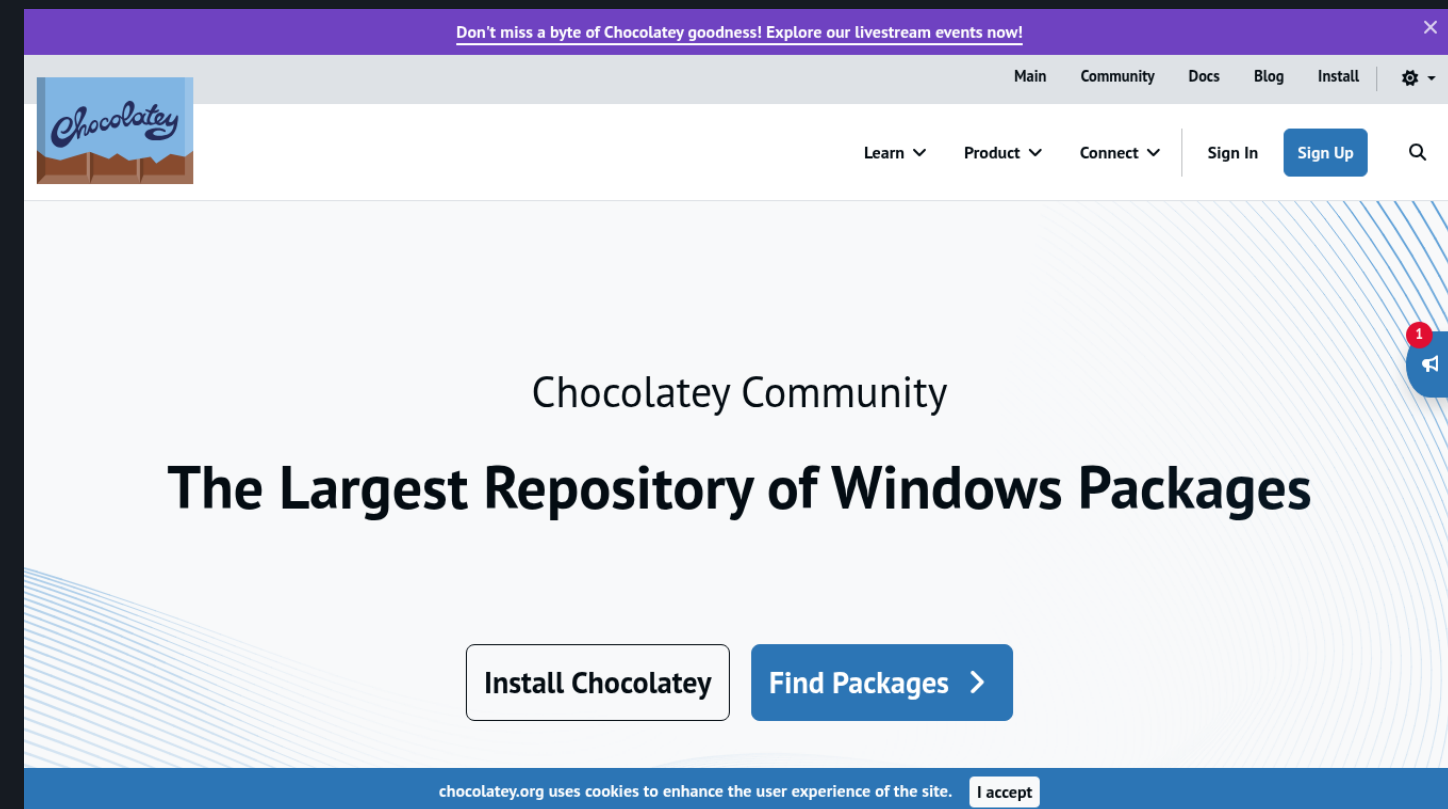
```
1 brew install quarto
```



```
1 choco install quarto
```



<https://brew.sh/>



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

Using GitHub Codespaces



Open in 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.

Quarto: a command line interface

BASH

1

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.

Checking Quarto is installed

```
1 quarto check

1 Quarto 1.8.24
2 [v] Checking environment information...
3     Quarto cache location: /home/vscode/.cache/quarto
4 [v] Checking versions of quarto binary dependencies...
5     Pandoc version 3.6.3: OK
6     Dart Sass version 1.87.0: OK
7     Deno version 2.3.1: OK
8     Typst version 0.13.0: OK
9 [v] Checking versions of quarto dependencies.....OK
10 [v] Checking Quarto installation.....OK
11     Version: 1.8.24
12     Path: /opt/quarto/bin
13
14 [v] Checking tools.....OK
15     TinyTeX: (external install)
16     Chromium: (not installed)
17
18 [v] Checking LaTeX.....OK
19     Using: TinyTex
20     Path: /home/vscode/.TinyTeX/bin/x86_64-linux
21     Version: 2025
22
```

Checking Quarto is installed

```
1 quarto check

1 Quarto 1.8.24
2 [v] Checking environment information...
3     Quarto cache location: /home/vscode/.cache/quarto
4 [v] Checking versions of quarto binary dependencies...
5     Pandoc version 3.6.3: OK
6     Dart Sass version 1.87.0: OK
7     Deno version 2.3.1: OK
8     Typst version 0.13.0: OK
9 [v] Checking versions of quarto dependencies.....OK
10 [v] Checking Quarto installation.....OK
11     Version: 1.8.24
12     Path: /opt/quarto/bin
13
14 [v] Checking tools.....OK
15     TinyTeX: (external install)
16     Chromium: (not installed)
17
18 [v] Checking LaTeX.....OK
19     Using: TinyTex
20     Path: /home/vscode/.TinyTeX/bin/x86_64-linux
21     Version: 2025
22
```

1 Quarto version and depende

2

3

Checking Quarto is installed

```
1 quarto check

[✓] Checking versions of quarto binary dependencies ...
5   Pandoc version 3.6.3: OK
6   Dart Sass version 1.87.0: OK
7   Deno version 2.3.1: OK
8   Typst version 0.13.0: OK
9 [✓] Checking versions of quarto dependencies.....OK
10 [✓] Checking Quarto installation.....OK
11   Version: 1.8.24
12   Path: /opt/quarto/bin
13
14 [✓] Checking tools.....OK
15   TinyTeX: (external install)
16   Chromium: (not installed)
17
18 [✓] Checking LaTeX.....OK
19   Using: TinyTex
20   Path: /home/vscode/.TinyTeX/bin/x86_64-linux
21   Version: 2025
22
23 [✓] Checking Chrome Headless.....OK
24   Using: Chrome found on system
25   Path: /usr/bin/google-chrome
26   Source: PATH
```

Tools installed by Quarto (e.g., **quarto** install).

2

3

4

Checking Quarto is installed

```
1 quarto check

9 [v] Checking versions of quarto dependencies.....OK
10 [v] Checking Quarto installation.....OK
11     Version: 1.8.24
12     Path: /opt/quarto/bin
13
14 [v] Checking tools.....OK
15     TinyTeX: (external install)
16     Chromium: (not installed)
17
18 [v] Checking LaTeX.....OK
19     Using: TinyTex
20     Path: /home/vscode/.TinyTeX/bin/x86_64-linux
21     Version: 2025
22
23 [v] Checking Chrome Headless.....OK
24     Using: Chrome found on system
25     Path: /usr/bin/google-chrome
26     Source: PATH
27
28 [v] Checking basic markdown render....OK
29
30 [v] Checking Python 3 installation....OK
31     Version: 3.13.7
```

LaTeX installation detected by Quarto.

Checking Quarto is installed

```
1 quarto check

14 [v] Checking tools.....OK
15     TinyTeX: (external install)
16     Chromium: (not installed)
17
18 [v] Checking LaTeX.....OK
19     Using: TinyTex
20     Path: /home/vscode/.TinyTeX/bin/x86_64-linux
21     Version: 2025
22
23 [v] Checking Chrome Headless.....OK
24     Using: Chrome found on system
25     Path: /usr/bin/google-chrome
26     Source: PATH
27
28 [v] Checking basic markdown render....OK
29
30 [v] Checking Python 3 installation....OK
31     Version: 3.13.7
32     Path: /usr/local/python/current/bin/python3
33     Jupyter: 5.8.1
34     Kernels: julia-1.11, python3
35
36 [v] Checking Jupyter engine render....OK
```

Chromium-based browser installation detected by Quarto.

Checking Quarto is installed

```
1 quarto check

21 Version: 2023
22
23 [v] Checking Chrome Headless.....OK
24     Using: Chrome found on system
25     Path: /usr/bin/google-chrome
26     Source: PATH
27
28 [v] Checking basic markdown render....OK
29
30 [v] Checking Python 3 installation....OK
31     Version: 3.13.7
32     Path: /usr/local/python/current/bin/python3
33     Jupyter: 5.8.1
34     Kernels: julia-1.11, python3
35
36 [v] Checking Jupyter engine render....OK
37
38 [v] Checking R installation.....OK
39     Version: 4.5.1
40     Path: /opt/R/4.5.1/lib/R
41     LibPaths:
42         - /home/vscode/R/x86_64-pc-linux-gnu-library/4.5
43         - /opt/R/4.5.1/lib/R/library
```

Python installation and computing dependencies detected by Quarto.

Checking Quarto is installed

```
1 quarto check

Source: PATH

26
27
28 [v] Checking basic markdown render....OK
29
30 [v] Checking Python 3 installation....OK
31   Version: 3.13.7
32   Path: /usr/local/python/current/bin/python3
33   Jupyter: 5.8.1
34   Kernels: julia-1.11, python3
35
36 [v] Checking Jupyter engine render....OK
37
38 [v] Checking R installation.....OK
39   Version: 4.5.1
40   Path: /opt/R/4.5.1/lib/R
41   LibPaths:
42     - /home/vscode/R/x86_64-pc-linux-gnu-library/4.5
43     - /opt/R/4.5.1/lib/R/library
44   knitr: 1.50
45   rmarkdown: 2.29
46
47 [v] Checking Knitr engine render.....OK
```

R installation and computing dependencies
detected by Quarto.

- Creating a new Quarto project
- Quarto Projects: default
- Quarto Projects: website
- Quarto Projects: blog
- Quarto Projects: book
- Quarto Projects: manuscript
- Writing with your favourite editor
- Writing using the Visual Editor

Quarto Projects

Creating a new Quarto project



Bash

```
1 quarto create project
```



Creating a new Quarto project



Bash

```
1 quarto create project
```



BASH

```
1  ? Type
2  > default
3  website
4  blog
5  manuscript
6  book
7  confluence
```



Quarto Projects: default

```
Bash
1 quarto create project default Default
_demo/Default
├─ Default.qmd
└─ _quarto.yml
0 directories, 2 files
```

Quarto Projects: default

```
Bash
1 quarto create project default Default
_demo/Default
├─ Default.qmd
└─ _quarto.yml
0 directories, 2 files
```

Default

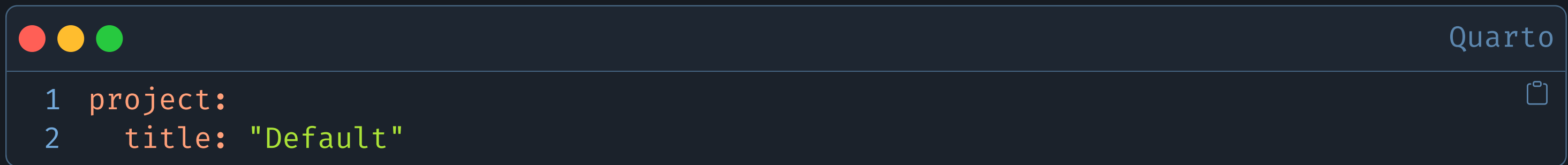
Quarto

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

Quarto Projects: default

Quarto Projects: **default**

- `_quarto.yml`: project configuration file.



```
1 project:
2   title: "Default"
```

Quarto Projects: `default`

- `_quarto.yml`: project configuration file.

```
1 project:
2   title: "Default"
```

- `Default.qmd`: default Quarto document.

```
1 ---
2 title: "Default"
3 ---
4
5 ## Quarto
6
7 Quarto enables you to weave together content and executable code into a finished doc
```

Quarto Projects: **website**

```
Bash
1 quarto create project website Website
_demo/Website
├─ _quarto.yml
├─ about.qmd
├─ index.qmd
└-- styles.css

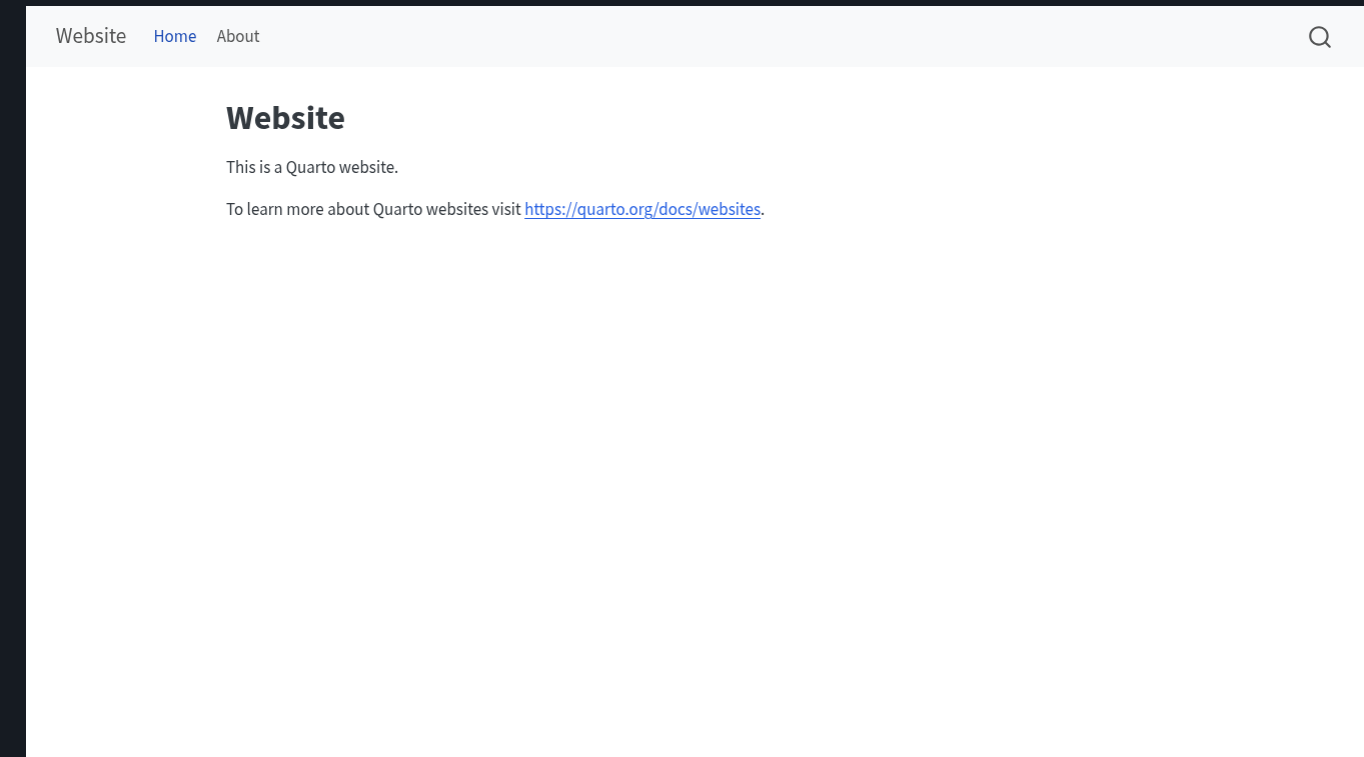
0 directories, 4 files
```

Quarto Projects: **website**

```
Bash
1 quarto create project website Website

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

0 directories, 4 files
```



Quarto Projects: **blog**

```
Bash
1 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

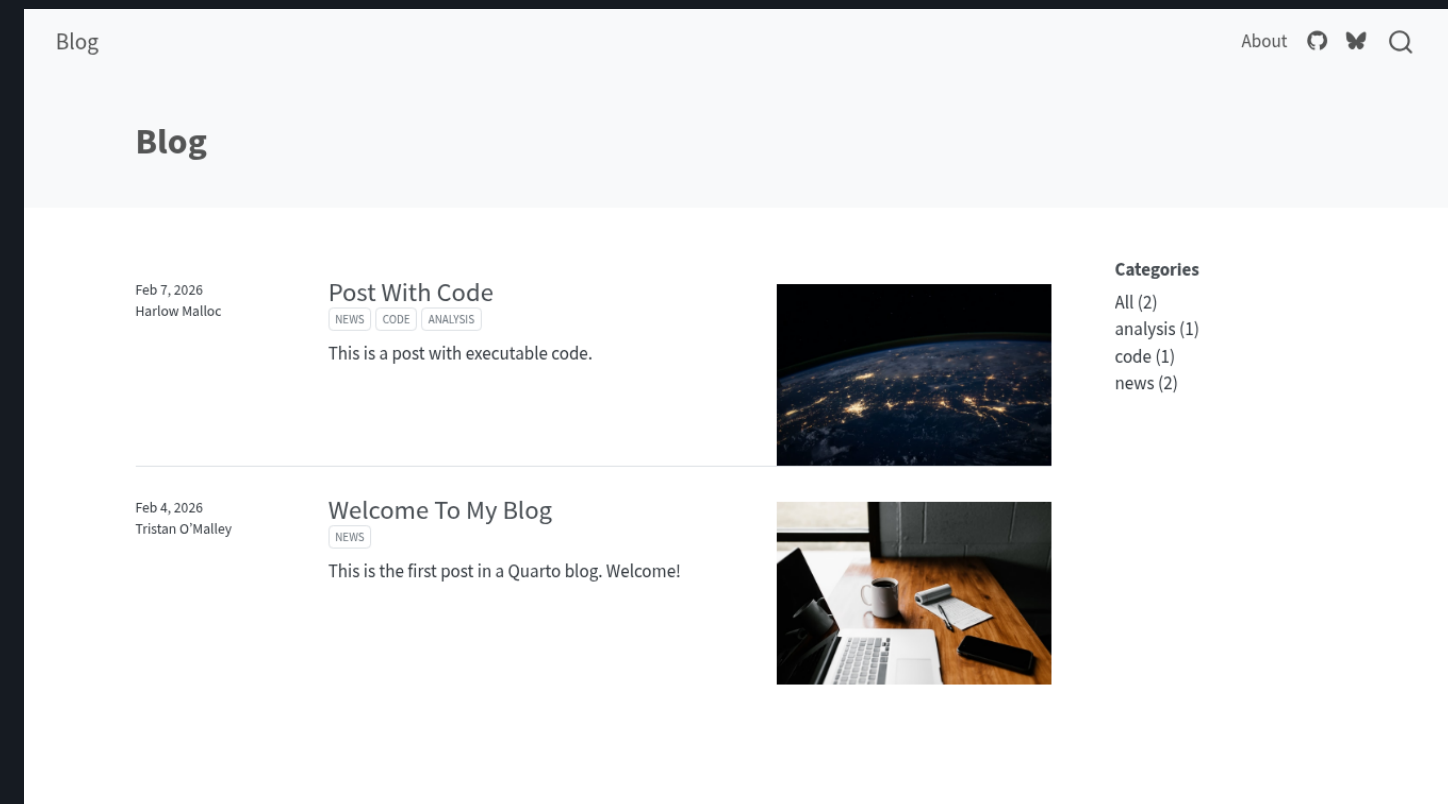
3 directories, 10 files
```

Quarto Projects: **blog**

```
Bash
1 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
```



Quarto Projects: **book**

```
Bash
1 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
```

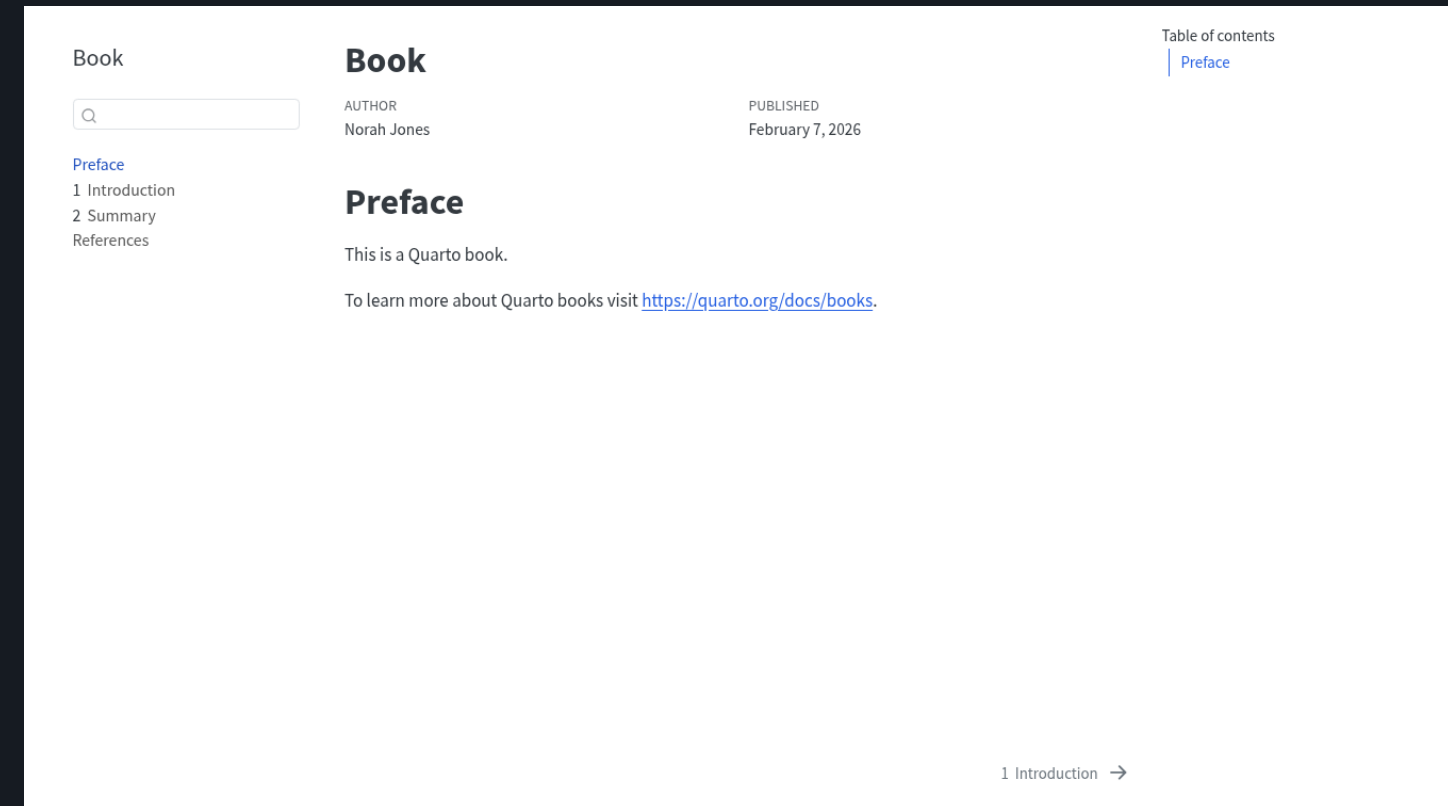
Quarto Projects: **book**

```
Bash

1 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
```



Quarto Projects: **manuscript**

```
Bash
1 quarto create project manuscript Manuscript

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

0 directories, 3 files
```

Quarto Projects: manuscript

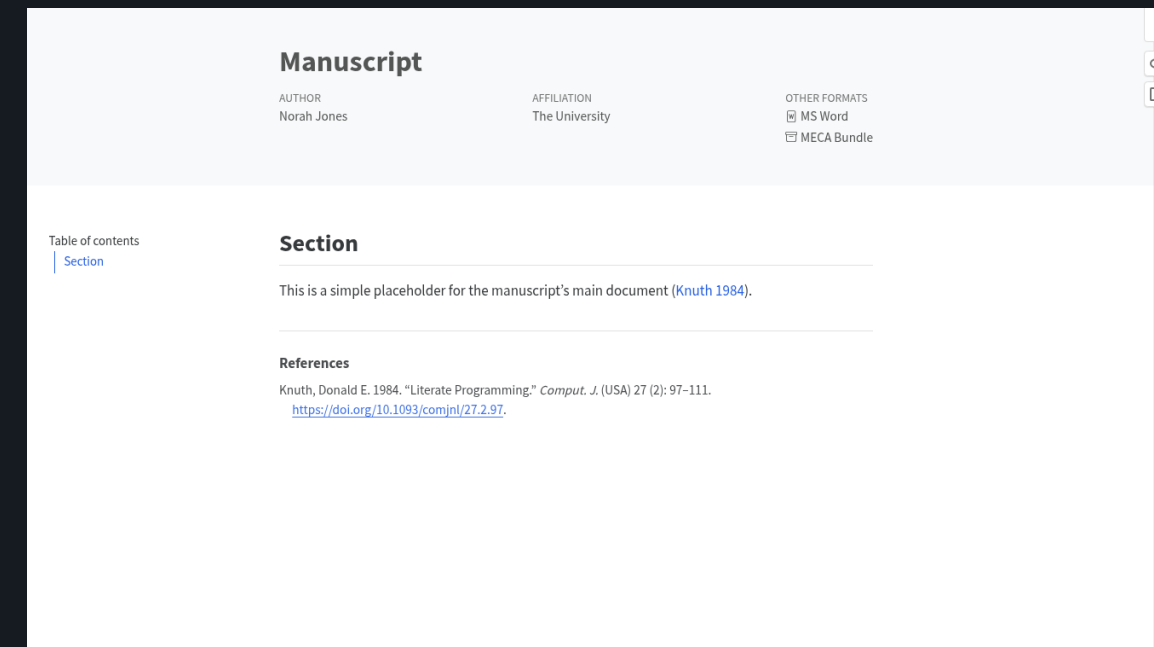


Bash

```
1 quarto create project manuscript Manuscript
```

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

0 directories, 3 files
```



Writing with your favourite editor

POSITRON

VSCODE

JUPYTER

RSTUDIO

NEOVIM

TEXT-EDITOR

quarto

OverviewGet StartedGuideExtensionsReferenceGalleryBlogHelp

On this page

supported byposit

Get Started

Tutorial: Hello, Quarto

Tutorial: Computations

Tutorial: Authoring

Tutorial: Hello, Quarto

Choose your tool

Positron

VS Code

Jupyter

RStudio

Neovim

Editor

R or Python?

You can work through this tutorial using R or Python code examples. Select your preferred language:

R

Python

You've selected to see examples in R. You can toggle to Python whenever you like throughout the guide.

Overview

Quarto is an open-source scientific and technical publishing system that weaves together code and narrative to produce high-quality documents, presentations, websites, and more. In this tutorial, you'll learn how to use Positron with Quarto.

Positron comes ready to work with Quarto out-of-the-box — it includes both the Quarto command line interface and the [Quarto VS Code extension](#).

It includes many tools that enhance working with Quarto, including:

- Integrated render and preview for Quarto documents
- Completion and diagnostics for Quarto options
- Positron's full R and Python support for code that is inside a Quarto document, including interactive execution of code in the Console, code completion, help, and diagnostics.

Here's a sample Quarto document, `hello.qmd`, open in Positron, demonstrating the seamless side-by-side editing and preview experience:

RPython

hello.qmd — _positron

https://quarto.org/

mickael.canouil.fr | License: CC BY-NC-SA 4.0

Mickaël
CANOUIL

Writing with your favourite editor

POSITRON

VSCODE

JUPYTER

RSTUDIO

NEOVIM

TEXT-EDITOR

quarto

OverviewGet StartedGuideExtensionsReferenceGalleryBlogHelp

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 VS Code. Before getting started, you should install the [Quarto VS Code Extension](#), which includes many tools that enhance working with Quarto, including:

- Integrated render and preview for Quarto documents.
- Syntax highlighting for markdown and embedded languages
- Completion and diagnostics for YAML options
- Completion for embedded languages (e.g. Python, R, Julia, etc.)
- Commands and key-bindings for running cells and selected lines.

You can install the Quarto extension from within the **Extensions** tab in VS Code, from the [Extension Marketplace](#), the [Open VSX Registry](#) or directly from a [VISX extension file](#).

Note

This tutorial focuses on editing plain text Quarto `.qmd` files in VS Code. Depending on your preferences and the task at hand there are two other editing modes available for Quarto documents: the [Visual Editor](#) and the [Notebook Editor](#). For the purposes of learning we recommend you work through this tutorial using the VS Code text editor, then after you've mastered the basics explore using the other editing modes.

Basic Workflow

Quarto `.qmd` files contain a combination of markdown and executable code cells. Here's what it might look like in VS Code to edit and preview a `.qmd` file:

demo.qmd

Quarto Preview

On this page

Overview

Basic Workflow

Render and Preview

YAML Options

Markdown

Code Cells

External Preview

Next Up

Edit this page

Report an issue

supported by

posit

Users > jllalrre > Desktop > demo.qmd

1
2 title: "matplotlib demo"
3 format:
4 html:
5 code-fold: true
6 jupyter: python3
7
8
9 For a demonstration of a line plot on a polar axis, see
10 @fig-polar.

matplotlib demo

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

► Code

90°

https://quarto.org/

mickael.canouil.fr | License: CC BY-NC-SA 4.0

Mickaël
CANOUIL

Writing with your favourite editor

POSITRON

VSCODE

JUPYTER

RSTUDIO

NEOVIM

TEXT-EDITOR

quarto

OverviewGet StartedGuideExtensionsReferenceGalleryBlogHelp

On this page

OverviewRenderingAuthoringYAML OptionsMarkdown CellsCode CellsNext Up

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 Jupyter Lab with Quarto. You'll edit code and markdown in Jupyter Lab, just as you would with any notebook, and preview the rendered document in a web browser as you work.

Below is an overview of how this will look.

basics.ipynb - JupyterLab

localhost:8888/lab/tree/basics.ipynb

File Edit View Run Kernel Tabs Settings Help

Launcher basics.ipynb Python 3

title: "Quarto Basics"

format: html

code-fold: true

jupyter: python3

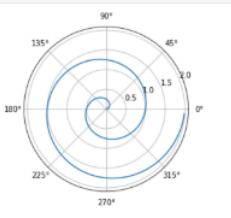
Polar Axis

For a demonstration of a line plot on a polar axis, see @fig-polar.

```
[1]: # label: fig-polar
# fig-cap: "A line plot on a polar axis"

import numpy as np
import matplotlib.pyplot as plt

r = np.arange(0, 2, 0.01)
theta = 2 * np.pi * r
fig, ax = plt.subplots(
    subplot_kw={'projection': 'polar'}
)
ax.plot(theta, r)
ax.set_rticks([0.5, 1, 1.5, 2])
ax.grid(True)
plt.show()
```



Quarto Basics

Polar Axis

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

Code

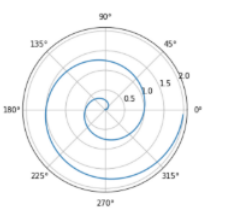


Figure 1: A line plot on a polar axis

https://quarto.org/

Writing with your favourite editor

POSITRON

VSCODE

JUPYTER

RSTUDIO

NEOVIM

TEXT-EDITOR

quarto

OverviewGet StartedGuideExtensionsReferenceGalleryBlogHelp

On this page

OverviewRenderingAuthoringHow it worksNext Up

Edit this pageReport an issue

Get Started

Tutorial: Hello, Quarto

Tutorial: Computations

Tutorial: Authoring

Tutorial: Hello, Quarto

Choose your tool

PositronVS CodeJupyterRStudioNeovimEditor

Overview

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.

In this tutorial, we'll show you how to use RStudio with Quarto. You'll edit code and markdown in RStudio just as you would with any computational document (e.g., R Markdown) and preview the rendered document in the Viewer tab as you work.

The following is a Quarto document with the extension `.qmd` (on the left), along with its rendered version as HTML (on the right). You could also choose to render it into other formats like PDF, MS Word, etc.

hello.qmd

SourceVisual

```
---
title: "Hello, Quarto"
format: html
editor: visual
---

{r}
#| label: load-packages
#| include: false

library(tidyverse)
library(palmerpenguins)
```

Meet Quarto

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

Meet the penguins

The `penguins` data from the [palmerpenguins](#) package contains size measurements for 344 penguins from three species observed on three islands in the Palmer Archipelago, Antarctica.

EnvironmentHistoryConnectionsBuildGitTutorial

FilesPlotsPackagesHelpViewerPresentation

Edit

Hello, Quarto

Meet Quarto

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

Meet the penguins

The `penguins` data from the [palmerpenguins](#) package contains size measurements for 344 penguins from three species observed on three islands in the Palmer Archipelago, Antarctica.

The plot below shows the relationship between flipper and bill lengths of these penguins.

https://quarto.org/

Writing with your favourite editor

POSITRON

VSCODE

JUPYTER

RSTUDIO

NEOVIM

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 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 (PDE)¹ 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-lspconfig

dependencies = {

"johndr/otter.nvim",

"neovim/nvim-lspconfig",

},

config = function()

require "quarto-nvim".setup {

logFeatures = {

enabled = true,

},

languages = { "r", "python", "julia" },

diagnostics = {

enabled = true,

triggers = { "BufWrite" }

},

completion = {

enabled = true

}

}

end

)

end vim code from python/r/julia documents to a text file

scripts

gritpane

move queries to quarto-nvim

2 days ago

0 forks

LICENSE

babel config

7 months ago

README.md

update readme

yesterday

api.lua

refactor

yesterday

lazy-lsp.j...

add keymap for settings

yesterday

Releases

Release the otter 1.0

4 days ago

0 forks

Quarto Nvim Kickstarter

Compersion to <https://github.com/quarto-dev/quarto-nvim>

Setup

Package

no package published

Push your first package

Languages

See 100.0%

Writing with your favourite editor

POSITRON

VSCODE

JUPYTER

RSTUDIO

NEOVIM

TEXT-EDITOR

quarto

OverviewGet StartedGuideExtensionsReferenceGalleryBlogHelp

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.

hello.qmd

```
1 ---
2 title: "Quarto Basics"
3 format:
4   html:
5     code-fold: true
6   jupyter: python3
7 ---
8
9 # Polar Axis
10
11 For a demonstration of a line plot on a polar axis, see @fig-polar.
12
13 ... (python)
14 # label: fig-polar
15 # fig-cap: "A line plot on a polar axis"
16
17 import numpy as np
18 import matplotlib.pyplot as plt
19
20 r = np.arange(0, 2, 0.01)
21 theta = 2 * np.pi * r
22 fig, ax = plt.subplots(
23   subplot_kw={'projection': 'polar'})
24
25 ax.plot(theta, r)
26 ax.set_ticks([0.5, 1, 1.5, 2])
27 ax.grid(True)
28 plt.show()
29
30
31
```

Quarto Basics

Polar Axis

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

Code

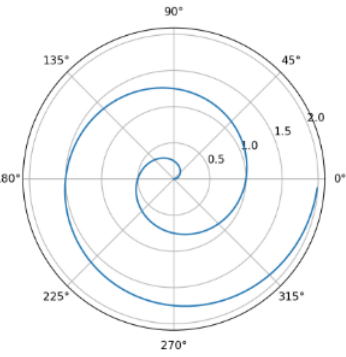


Figure 1: A line plot on a polar axis

On this page

Overview

Editor Modes

Rendering

Authoring

YAML Options

Markdown

Code Cells

Next Up

Edit this page

Report an issue

The notebook on the left is *rendered* into the HTML version you see on the right. This is the basic model for Quarto publishing—take a source document (in this case a notebook) and render it to a variety of *output formats*, including

https://quarto.org/

≡

mickael.canouil.fr | License: CC BY-NC-SA 4.0

Mickaël
CANOUIL

Writing using the Visual Editor

POSITRON

VSCODE

RSTUDIO

quarto

OverviewGet StartedGuideExtensionsReferenceGalleryBlogHelp

Guide

Authoring

Computations

Tools

JupyterLab

RStudio IDE

Positron

Positron Basics

Visual Editor

Notebook Editor

VS Code

Neovim

Text Editors

Documents

Presentations

Dashboards

Websites

Books

Manuscripts

Interactivity

Publishing

Projects

Advanced

Guide > Tools > Positron > Visual Editor

Visual Editing in Positron

Overview

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:

libraries.qmd

SourceVisualRender on SavePreview

libraries.qmd

NormalBIt</>ListTableLinkImageFormatInsertTable

title: "Libraries"

format: html

execute:

echo: fenced

Overview

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

In this document we'll provide a high-level overview of the core libraries and some examples of using third-party libraries ([D3](#) and [Arquero](#)). Creating your own libraries is covered in the article on [Code Reuse](#).

Stdlib

You can switch between visual and source mode at any time and can even edit documents concurrently in both modes. To switch between visual and source mode:

On this page

Overview

Keyboard Shortcuts

Insert Anything

Editor Toolbar

Editor Options

Zotero Citations

Markdown Output

Edit this page

Report an issue

supported by

posit

https://quarto.org/

Writing using the Visual Editor

POSITRON

VSCODE

RSTUDIO

quarto

OverviewGet StartedGuideExtensionsReferenceGalleryBlogHelp

Guide

Authoring

Computations

Tools

JupyterLab

RStudio IDE

Positron

VS Code

VS Code Basics

Visual Editor

Notebook Editor

Neovim

Text Editors

Documents

Presentations

Dashboards

Websites

Books

Manuscripts

Interactivity

Publishing

Projects

Advanced

Guide > Tools > VS Code > Visual Editor

Visual Editing in VS Code

Overview

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:

libraries.qmd M

quarto-web > docs > interactive > ojs > libraries.qmd

Normal | B | I | <> | | | | | Format | Insert | Table

title: "Libraries"

format: html

Overview

There are three types of library you'll generally use with OJS:

- [Observable core libraries](#) automatically available in every document.
- Third-party JavaScript libraries from [npm](#) and [ObservableHQ](#).
- Custom libraries you and/or your colleagues have created

In this document we'll provide a high-level overview of the core libraries and some examples of using third-party libraries ([D3](#) and [Arquero](#)). Creating your own libraries is covered in the article on [Code Reuse](#).

You can switch between visual and source mode at any time and can even edit documents concurrently in both modes. To switch between visual and source mode:

- Use the `⌘ ^ F4` keyboard shortcut.
- Use the context menu from anywhere in a document:

On this page

Overview

Keyboard Shortcuts

Insert Anything

Editor Toolbar

Editor Options

Zotero Citations

Markdown Output

Edit this page

Report an issue

supported by

posit

https://quarto.org/

Writing using the Visual Editor

POSITRON

VSCODE

RSTUDIO

quarto

Overview

Get Started

Guide

Extensions

Reference

Gallery

Blog

Help

Guide

Authoring

Computations

Tools

JupyterLab

RStudio IDE

RStudio Basics

Visual Editor

Editor Basics

Technical Writing

Content Editing

Shortcuts & Options

Markdown Output

Positron

VS Code

Neovim

Text Editors

Documents

Presentations

Dashboards

Websites

Books

Manuscripts

Interactivity

Publishing

Projects

Advanced

Guide > Tools > RStudio IDE > Visual Editor

Visual Editing in RStudio

Overview

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:

relational-data.qmd

Render on Save

Render

Source

Visual

B

I

<>

Normal

Format

Insert

Table

Outline

Filtering Joins

Filtering joins match observations in the same way as [mutating joins](#), but affect the observations, not the variables. There are two types:

<code>semi_join(x, y)</code>	$x \ltimes y$	Keeps all observations in <code>x</code> that have a match in <code>y</code>
<code>anti_join(x, y)</code>	$x \rhd y$	Drops all observations in <code>x</code> that have a match in <code>y</code>

Graphically, a semi-join looks like this:

```
{r::join-semi}~
#|echo:~false~
#|out-width:~"6"~
~
knitr::include_graphics("diagrams/join-semi.png")
```

On this page

Overview

Switching Modes

Getting Started

Using the Editor

Learning More

Edit this page

Report an issue

<https://quarto.org/>

- Exercise Overview
- Part 1: Project Creation
- Part 2: IDE Features Exploration
- Part 3: Document Customisation
- Part 4: Rendering
- Success Criteria

Hands-On Exercise: Creating Your First Quarto Project

Exercise Overview

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

Example Code:  [Exercises](#)

BASH

```
1 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.

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.

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.

Part 3: Document Customisation

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

1. **Update the YAML header** with your information:

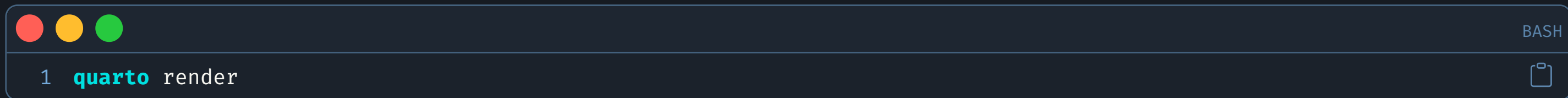
▶ Code

2. **Add content** to the document body.

Part 4: Rendering

Try different rendering approaches:

1. Command Line (classic method):

A terminal window with a dark background. The title bar shows three colored circles (red, yellow, green) on the left and the word 'BASH' on the right. The main area shows a prompt '1' followed by the command 'quarto render' in a light blue font. A small clipboard icon is visible on the right side of the terminal area.

```
1 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

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.



Mickaël CANOUIL, *Ph.D.*

Independent Contractor

pro@mickael.canouil.dev



Website

mickael.canouil.fr



GitHub

[@mcanouil](https://github.com/mcanouil)



LinkedIn

[MickaelCanouil](https://www.linkedin.com/in/MickaelCanouil)



Bluesky

[@mickael.canouil.fr](https://bsky.app/profile/@mickael.canouil.fr)



Mastodon

[@MickaelCanouil](https://mastodon.social/@MickaelCanouil)