

Quarto Extensions

for the Julia Community

Patrick Altmeyer Ronny Bergmann

Friday, July 12, 2024



Overview

1. What is Quarto?
2. Julia-themed Quarto:
Simple Extensions
3. Quarto for Documentation
4. Quarto for JuliaCon
Proceedings

talk

July 12, 11:00-11:30

Quarto Extensions for the Julia Community

Patrick Altmeyer
Ronny Bergmann

Quarto is an open-source scientific and technical publishing system that was first presented at JuliaCon 2022. We propose new extensions and workflows that we hope will help the community embrace this promising new tool and boost developers' efforts toward effective communication and reproducibility.



 **juliacon
2024
Eindhoven**

Join the largest annual
conference organized
around the Julia
programming language!



Philips Stadion, Eindhoven, NL.
July 9th-13th, 2024
<https://juliacon.org/2024>

What is Quarto?

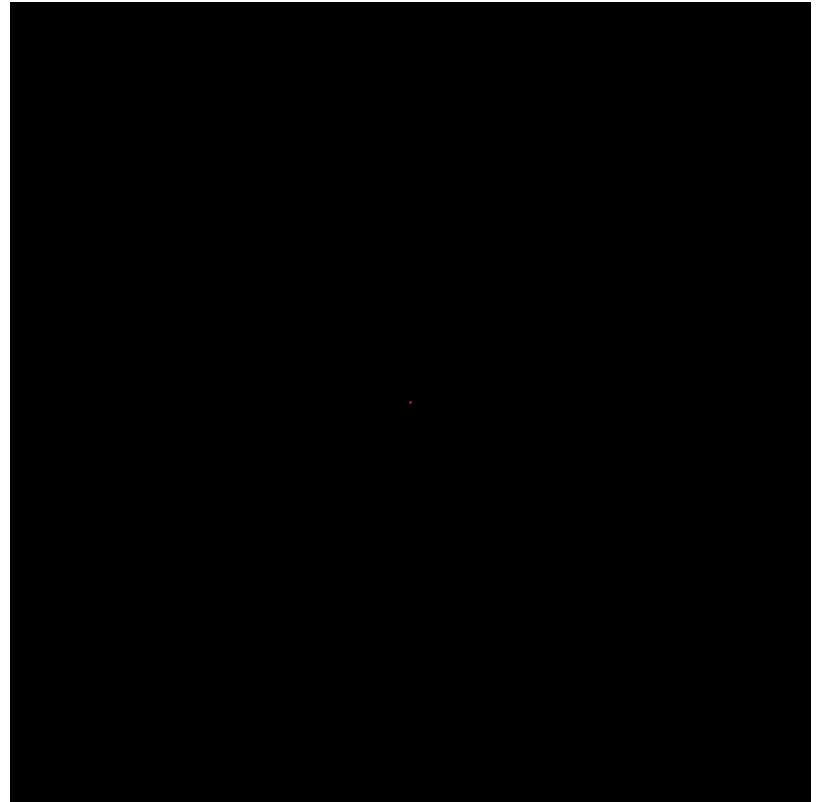
- Cross-platform open-source scientific publishing tool with a focus on reproducibility.
- Based on Markdown, which is easy to learn and write.
- Very flexible and can be extended with custom templates and styles.

```
1 ---
2 title: "Hello Quarto"
3 jupyter: julia-1.10
4 ---
5
6 ## Hello Quarto!
7
8 Blending Markdown and
9 Julia code:
10
11 ```{julia}
12 #| echo: true
13
14 println("Hello, Quarto")
15 ```
```

Julia-themed Quarto

Simple Quarto extension that adds a Julia-themed touch to your documents.

Extensions are a powerful way to modify and extend the behavior of Quarto.



Getting Started

To install the Julia-themed Quarto extensions, run:

```
1 quarto add pat-alt/quarto-julia
```

Usage

Simply adjust the Quarto header of your document:

```
1 ---
2 title: Julia-themed Quarto
3 format:
4   julia-html: default
5   julia-revealjs:
6     scrollable: true
7 author: pat-alt
8 date: last-modified
9 ---
```

Available Formats

- HTML (articles)
- Revealjs (presentations)

Preview and Render

```
1 quarto preview yourdoc.qmd
2 quarto render yourdoc.qmd
```

Fonts

1. [JuliaMono](#) font for `monospace` text and `code`.
2. [Barlow](#) font for headers and blockquotes (thanks [@cormullion](#)).
3. [Roboto](#) font for everything else.
4. Also available is [Bangla MN](#) by Muthu Nedumaran of [Murasu Systems](#), which is closely related to the official Julia logo font.

Code

Inline code looks like this `print("hello 🌍")`. Code blocks look like `this` (Revealjs not affected):

```
1 using CounterfactualExplanations, TaijaData
2
3 # Data and Classifier:
4 counterfactual_data = CounterfactualData(load_linearly_separable()).
5 M = fit_model(counterfactual_data, :Linear)
6
7 # Select random sample:
8 target = 2
9 factual = 1
10 chosen = rand(findall(predict_label(M, counterfactual_data) .== fac
11 x = select_factual(counterfactual_data, chosen)
12
13 # Counterfactual search:
14 generator = GenericGenerator()
15 ce = generate_counterfactual(x, target, counterfactual_data, M, gen
```

Callouts

Note

This is a note in [julia_blue](#). Icons are deactivated in Revealjs.

Tip

This is a tip in [julia_green](#).

Warning

This is a warning in [julia_purple](#).

Caution

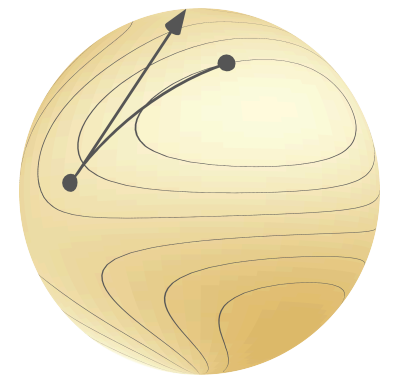
This is a caution callout in [julia_purple](#).

Important

This is an important callout in [julia_red](#).

Documentation

Write tutorials in Quarto that fully integrate into the rendered documentation.



Manopt.jl

Goals

Tutorials that are

- integrated in the documentation (`Documenter.jl`)
 - Integrated Links within the documentation
 - include mathematical formulae
- render into Markdown
- show code and results
- reproducible
- run on CI (cached)

General Workflow

1. write tutorials (`.qmd`) for a package in `tutorials/`
2. run their code when generating the documentation
3. render them (as `.md`) into `docs/src/tutorials/`

```
1  ---
2  title: "A short example"
3  ---
4
5  ```{julia}
6  #| echo: false
7  #| output: false
8  cd(@__DIR__); # activate
9  using Pkg; Pkg.activate(".")
10  ```
11
12  Let's start with loading
13  necessary packages
14
15  ```{julia}
16  using Manopt, Manifolds
17  M = Sphere(2)
18  ```
```

Workflow II: Configure Quarto

`_quarto.yml` in
`tutorials/`:

- one file:
`quarto render`
`file.qmd`
- `quarto render` in
`tutorials/` renders all
but !
- cached in
`tutorials/_freeze`

```
1 project:
2   title: "My Tutorials"
3   output-dir: ../docs/src
4   render: # specify some
5     - "*.qmd"
6     - "!NotThisFile.qmd"
7
8   format: #render to md
9     commonmark:
10       variant: -tex_math_do
11       wrap: preserve
```

Tip. Use the
[Quarto VS Code extension](#)
and the command
`Quarto Preview`

Workflow III: Documenter

- Use `CondaPkg.jl` to handle/install Python, creating a `CondaPkg.toml` specifying the Python version

```
1 [deps]
2 jupyter = ""
3 python = "3.11"
```

And include rendering the tutorials in your `make.jl`

```
1 using CondaPkg
2 CondaPkg.withenv() do
3     tutorials_folder = (@__DIR__) * "../tutorials"
4     run(`quarto render $(tutorials_folder)`)
5 end
```

Workflow IV: GitHub Action

On CI: Set up caches:

```
1 - name: Cache Quarto
2   uses: actions/cache@v4
3   with:
4     path: tutorials/_freeze # Quarto Cache
5     key: ${{ runner.os }}-${{ env.cache-name }}-${{ hashFiles('tuto
6     restore-keys: |
7       ${{ runner.os }}-${{ env.cache-name }}-
8 - name: Cache Documenter
9   uses: actions/cache@v4
10  with:
11    path: docs/src/tutorials # Cache rendered tutorials
12    key: ${{ runner.os }}-${{ env.cache-name }}-${{ hashFiles('tuto
13    restore-keys: |
14      ${{ runner.os }}-${{ env.cache-name }}-
```

There we are!

Cached, reproducible tutorials within `Documenter.jl`.

Challenges

- Cache vs. breaking versions of the package
- Recommendation: Maybe print package versions
- Quarto replaces spaces in markdown links
`[A](@ref B)` with `[A](@ref%20B)`.

These have to be “escaped”:

``[A](@ref B)`{=commonmark}`

- Due to pandoc:
for now write math in `$...$` and not ``...``

Documenter Summary

- tutorials with reproducible, executed code
- cached and rendered on CI
- integrated: we can use
 - `@ref` from `Documenter.jl`
 - `[citekey](@cite)` from `DocumenterCitations.jl`
 - `@extref` from `DocumenterInterLinks.jl`
- Full example: [Manopt.jl: Optimize](#) | [source](#)
- longer tutorial at [Julia Forem](#)

Outlook / Soon

- the `QuartoNotebookRunner.jl` should provide native rendering in pure Julia
- Pluto notebooks might be used as [input there as well](#)

JuliaCon Proceedings

The [quarto-juliacon-proceedings](#) extension adds support for writing a JuliaCon Proceedings article in Quarto.



Getting Started

To install the [JuliaCon Proceedings extension](#), run:

```
1 quarto add pat-alt/quarto-juliacon-proceedings
```

Disclaimer

PDF version resembles the official JuliaCon Proceedings format almost exactly but it is not officially endorsed by the JuliaCon Proceedings team.

See this [issue](#).

Supports both PDF and HTML:

```
1 ---
2 title: JuliaCon Proceedings in Quarto
3 format:
4   juliacon-proceedings-pdf:
5     keep-tex: true
6   juliacon-proceedings-html: default
7 author:
8   - name: Patrick Altmeyer
9     affiliations:
10      - Delft University of Technology
11     orcid: 0000-0003-4726-8613
12 ---
```

Motivation

JuliaCon Proceedings to set an example for reproducibility:

- Code, results and text in one document.
- Executable code blocks serve as a form of testing.
- Same document can be rendered into HTML, PDF, ...

Benefits for the authors:

- Essentially zero maintenance for the authors.
- LaTeX option still available for those who prefer it.

Showcase

The following two example articles were rendered using the extension:


- Official extension template: [\[pdf, html\]](#)
- Altmeyer, Deursen, et al. (2023): [\[pdf, html\]](#)

Julia-themed Quarto

Explaining Black-Box Models through Counterfactuals

Explaining Black-Box Models through Counterfactuals

AUTHORS

Patrick Altmeyer 
Arie van Deursen
Cynthia C. S. Liem

AFFILIATION

Delft University of Technology
Delft University of Technology
Delft University of Technology

ABSTRACT

We present `CounterfactualExplanations.jl`: a package for generating Counterfactual Explanations (CE) and Algorithmic Recourse (AR) for black-box models in Julia. CE explain how inputs into a model need to change to yield specific model predictions. Explanations that involve realistic and actionable changes can be used to provide AR: a set of proposed actions for individuals to change an undesirable outcome for the better. In this article, we

Comparison

Proposed Version

1 / 10 — + ↺



Explaining Black-Box Models

Patrick Altmeyer¹, Arie van Driel

¹Delft University of Technology

ABSTRACT

Published Version

1 / 10 — + ↺



Explaining Black-Box Models

Patrick Altmeyer¹, Arie van Driel

¹Delft University of Technology

ABSTRACT

More Information

- Official Quarto docs: <https://quarto.org/>
- Blog post on Quarto extensions:
<https://tinyurl.com/quarto-extensions>
- Blog post on Documeter integration:
<https://tinyurl.com/manopt-tutorials>
- Example tutorial written in Quarto:
<https://tinyurl.com/quarto-example>