JuliaCon Proceedings in Quarto

Patrick Altmeyer[1]

[1]Delft University of Technology

ABSTRACT

This document is a template demonstrating the `juliacon-proceedings` format. It is a standard Quarto document composed of Markdown, LaTeX and code chunks. It is rendered to PDF and HTML. The PDF is rendered using the `juliacon-proceedings-pdf` format, which is based on the `juliacon-proceedings` LaTeX class. The HTML is rendered using the `juliacon-proceedings-html` format, which is based on the `juliacon-proceedings` HTML template.

Keywords

Template, Demo, Quarto, JuliaCon

1. Introduction

This is a template for writing a JuliaCon Proceedings article in Quarto. This is a proof-of-concept for how we could use Quarto for JuliaCon proceedings. For current submissions, please ignore this repo and follow the official instructions here.

1.1 What is Quarto?

Quarto makes it easy to write reproducible documents that can be rendered to PDF, HTML, Word and more. It is based on Markdown, which is easy to learn and write. It also supports LaTeX, which is useful for more advanced formatting. As this extension demonstrates, Quarto is also very flexible and can be extended with custom templates and styles.

1.2 Why Quarto?

By embracing Quarto, JuliaCon Proceedings can set an example for how to write reproducible documents. We would not only make it easier for authors to write their submissions but also open the door for more advanced features such as interactive figures and executable code blocks in HTML documents.

2. About this template

This template is based on the existing JuliaCon Proceedings LaTeX template. The rendered versions can be found here: pdf, html.

2.1 Basic Usage

To use the Quarto extension that provides this template, you can create a new project as follows:

```
quarto use template pat-alt/quarto-juliacon-proceedings
```

Alternatively, you can add the extension to an existing project:

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

Then, add the format to your document options:

```yaml
format:
  juliacon-proceedings-pdf: default
```

3. Code

There are various options to present code using this template. We could leverage Quarto's existing support for executable, cross-referenceable code. Alternatively, we could use the existing `lstlisting` environment for Julia code.

3.1 Executable Listings (version $\geq$ 1.4.0)

Relying on Quarto's support for executable listings is the most flexible option as it allows us to use all of Quarto's features. It comes with numerous advantages:

1. We can use the same code chunks for both PDF and HTML output (and more) with consistent formatting.
2. Using executable code ensures that things stay up-to-date and potential errors in the code are identified early: as you render your document, the code will be executed and the output will be inserted into the document.
3. The output of executable code can be cross-referenced. For example, we can reference a figure that is generated by a code chunk and the reference will be updated automatically when the figure number changes.
4. Code chunks can also be hidden. In HTML, there is additional support for code folding.
5. The code itself can be cross-referenced.

From the Quarto Docs:

> To create cross-referenceable code listings from executable code blocks, use `lst-label` and `lst-cap`. — Quarto Docs

For example, the following code block will be labelled as `lst-1` and calling `@lst-1` will render as Listing 1. The figure it generates will be labelled as `fig-1` and calling `@fig-1` will render as Figure 1.

Listing 1 A listing caption.

```
using Plots

x = -3.0:0.01:3.0
y = rand(length(x))
plot(x, y)
```

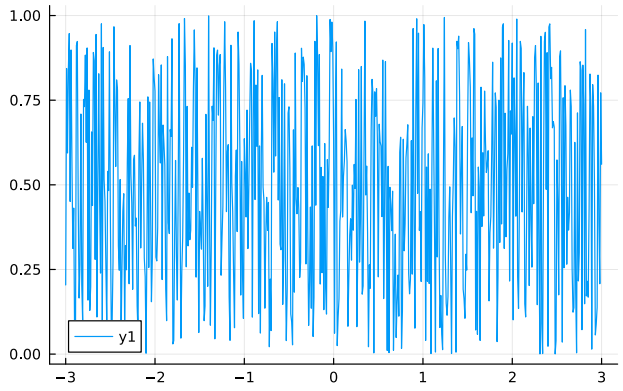

Fig. 1: A figure caption.

> **Current Limitations**
>
> Support for executable listings is only available in the development version of Quarto (version $\geq$ 1.4.0). It is currently not clear how to support custom formatting, for example through the existing `jlcode.sty` environment.

3.2 Static Listings

The special environment that is already defined for Julia code is still compatible with this template.

```
\begin{lstlisting}[
    language = Julia,
    numbers=left,
    label={lst:exmplg},
    caption={Example Code Block.}
]
using Plots

x = -3.0:0.01:3.0
y = rand(length(x))
plot(x, y)
\end{lstlisting}
```

Code 1: Example Code Block.

```
1 using Plots
2
3 x = -3.0:0.01:3.0
4 y = rand(length(x))
5 plot(x, y)
```

Table 1. : This is a table caption.

Age	Frequency
18–25	15
26–35	33
36–45	22

4. Tables

As with code, there are various options to present tables. Using standard Markdown syntax is the most flexible option because it is compatible with various output formats. Alternatively, standard LaTeX syntax can be used.

4.1 Markdown Tables

Markdown tables are rendered to LaTeX using the `booktabs` package. This is the recommended way to create tables because it is compatible with various output formats. Tables can be cross-referenced and the table caption will be rendered as a LaTeX table caption.

```
| Default | Left | Right | Center |
|---------|:-----|------:|:------:|
| 12      | 12   |    12 |   12   |
| 123     | 123  |   123 |  123   |
| 1       | 1    |     1 |   1    |

: Demonstration of pipe table syntax {#tbl-1}
```

> **Current Limitations**
>
> This seems to currently mess with executable listing (Section 3.1).

4.2 LaTeX Tables

Standard LaTeX syntax can also be used to create tables. To use standard Quarto cross-referencing, the LaTeX syntax needs to be wrapped in a `div` with the `tbl` class. The table caption can be added as a paragraph. The following creates Table 1 which can be cross-referenced as `@tbl-1`.

```
::: {#tbl-1}

\begin{tabular}{|l|l|}\hline
Age & Frequency \\ \hline
18--25  & 15 \\
26--35  & 33 \\
36--45  & 22 \\ \hline
\end{tabular}

This is a table caption.

:::
```