Template for contribution to Computo

Example dedicated to `Julia` users

Jane Doe   Statistics, Name of Affiliation one
John Doe   Computer Science, Name of Afficiliation two

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur posuere vestibulum facilisis. Aenean pretium orci augue, quis lobortis libero accumsan eu. Nam mollis lorem sit amet pellentesque ullamcorper. Curabitur lobortis libero eget malesuada vestibulum. Nam nec nibh massa. Pellentesque porttitor cursus tellus. Mauris urna erat, rhoncus sed faucibus sit amet, venenatis eu ipsum.

Keywords: key1, key2, key3

Contents

1 Introduction

1.1 About this document

This document, accompanied with the [hopefully finely tuned git repos](), provides a template for writing contributions to **Computo** (Computo Team 2020). We show how `Julia` code can be included and how the repository can be set up for triggering github actions for rendering the document, with dependencies handled by the built-in `Pkg` manager.

1.2 Setup a github repository for preparing your submission

You can start by clicking the "use this template" button, on the top of the page of the [github repository associated to this document](). Of course, you can set your repository private during the preparation of your manuscript.

1.3 Quarto

You need [quarto]() installed on your system and the [Computo extension]() to prepare your document. For the latter, once quarto is installed, run the following to install the extension in the current directory (it creates a `_extension` directory which is ignored by git thanks to `.gitignore` by default):

```
quarto add computorg/computo-quarto-extension
```

2 Formatting

This section covers basic formatting guidelines for [Quarto](), a versatile formatting system for authoring document integrating markdown, LaTeX and various code block interpreted either via Jupyter or Knitr (and thus dealing with Python, R or Julia). It relies on the [Pandoc]() document converter.

To render a document, run `quarto render`. By default, both PDF and HTML documents are generated:

```
quarto render template-computo-julia.qmd # will render both to html and PDF
```

> 💡 **Note**
>
> To check the syntax of the formatting below, you can use the `</> source` button at the top left of this document.

2.1 Basic markdown formatting

Bold text or *italic*

- This is a list
- With more elements
- It isn't numbered.

But we can also do a numbered list

1. This is my first item
2. This is my second item
3. This is my third item

2.2 Mathematics

2.2.1 Mathematical formulae

LaTeX code is natively supported[1], which makes it possible to use mathematical formulae:

$$f(x_1, \ldots, x_n; \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{n} (x_i - \mu)^2\right)$$

It is also posible to cross-reference an equation, see Equation 1:

$$\begin{aligned} D_{x_N} &= \frac{1}{2} \begin{bmatrix} x_L^\top & x_N^\top \end{bmatrix} \begin{bmatrix} L_L & B \\ B^\top & L_N \end{bmatrix} \begin{bmatrix} x_L \\ x_N \end{bmatrix} \\ &= \frac{1}{2}(x_L^\top L_L x_L + 2x_N^\top B^\top x_L + x_N^\top L_N x_N), \end{aligned} \tag{1}$$

2.2.2 Theorems and other amsthem-like environments

Quarto includes a nice support for theorems, with predefined prefix labels for theorems, lemmas, proposition, etc. see this page. Here is a simple example:

Theorem 2.1 (Strong law of large numbers)*. The sample average converges almost surely to the expected value:*

$$\overline{X}_n \xrightarrow{a.s.} \mu \qquad when\ n \to \infty.$$

See Theorem 2.1.

2.3 Julia Code

Quarto uses either Jupyter or knitr to render code chunks. This can be triggered in the yaml header. In this tutorial, we use `Jupyter`, (`Julia` and `Jupyter` must be installed on your computer)

---

[1]We use lualatex for this purpose.

```
---
title: "My Document"
author "Jane Doe"
jupyter: julia-1.8
---
```

`julia` code chunks may be embedded as follows:

```
using Random
rng = MersenneTwister(1234)
x = randn(rng, ComplexF32, (10,1))
x
```

```
10×1 Matrix{ComplexF32}:
   0.61330706f0 - 0.63762915f0im
  -0.34964928f0 - 0.6384568f0im
   0.61122406f0 + 1.5640336f0im
    0.3767559f0 - 0.19214594f0im
    0.3552041f0 - 0.36556262f0im
   -0.3963343f0 - 0.013641349f0im
   0.09055523f0 + 1.3101153f0im
  -0.58531713f0 + 0.077849716f0im
  -0.17760806f0 + 0.26142728f0im
   0.05099396f0 - 1.0630851f0im
```

2.4 Figures

Plots can be generated as follows:

```
using Plots

plot(sin,
     x->sin(2x),
     0,
     2 ,
     leg=false,
     fill=(0,:lavender))
```
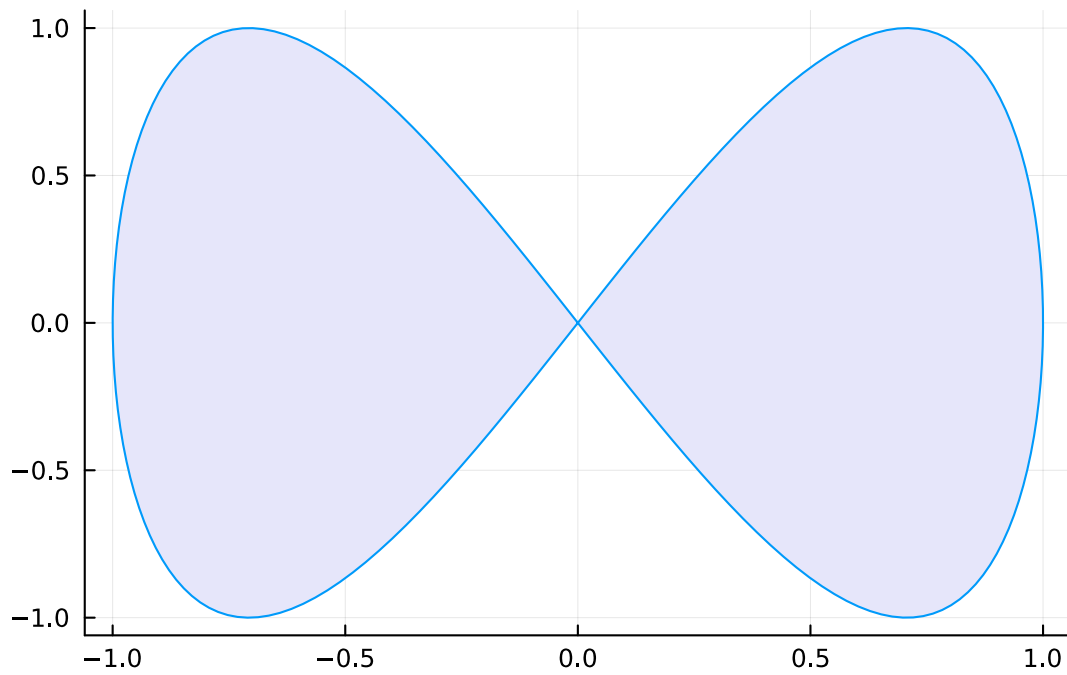
Figure 1: Parametric Plots

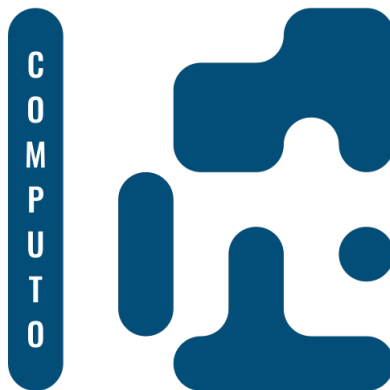It is also possible to create figures from static images:

Figure 2: Computo logo (label)

2.5 Tables

Tables (with label: `@tbl-mylabel` renders Table 1) can be generated with markdown as follows

Table 1: my table caption

Tables	Are	Cool
col 1 is	left-aligned	$1600
col 2 is	centered	$12
col 3 is	right-aligned	$1

2.6   Handling references

2.6.1   Bibliographic references

References are displayed as footnotes using [BibTeX](#), e.g. `[@computo]` will be displayed as (Computo Team 2020), where `computo` is the bibtex key for this specific entry. The bibliographic information is automatically retrieved from the `.bib` file specified in the header of this document (here: `references.bib`).

2.6.2   Other cross-references

As already (partially) seen, Quarto includes a mechanism similar to the bibliographic references for sections, equations, theorems, figures, lists, etc. Have a look at [this page](#).

2.7   Advanced formatting

Advanced formatting features are possible and documented (including interactive plots, pseudo-code, (Tikz) diagrams, Lua filters, mixing R + Python in the same document), but are beyond the scope of this simple introduction. We point several entries in this direction.

> ⚠️ More information
>
> - The [Quarto web site](#) for comprehensive documentation.
> - The [template distributed with the Computo Quarto extension](#), which uses such advanced features.
> - [Our mock version of the t-SNE paper](#), a full and advanced example using Python and the Jupyter kernel.
> - [The previously published papers in Computo](#) can be used as references.

3   Finalize your submission

3.1   Handle `Julia` dependencies with `Pkg`

To make your work reproducible, you need to fix the packages and environment used to run your analysis. In `Julia`, the built-in `Pkg` package manager is a method of choice.

https://towardsdatascience.com/how-to-setup-project-environments-in-julia-ec8ae73afe9c.

In this tutorial, we simply need to add the `Plots` library and its dependencies. Once added via the `Pkg` manager of Julia, the files `Project.toml` and `Manifest.toml` in the `julia_env` directory contains in information required. We then use the command

```
julia --project=.
```

In the CI to set up the environment ready for reproductibility.

3.2   Continuous integration

The repository associated with this template is pre-configure to trigger an action on push that performs the following:

1. Check out repository on the `ubuntu-latest` machine
2. Install quarto and dependencies, including the Computo extension
3. Install Julia (1.8.1) and dependencies based on the files `Project.toml` and `Manifest.toml` in `julia_env`
4. Render your .qmd file and Publish the results on a gh-page (both HTML and PDF)

The file .github/workflows/build_n_publish.yml is largely inspired from this file.

Once this is successful, you are ready to submit your manuscript to the Computo submission platform.

> ⚠️ **Warning**
>
> The first time, you possibly need to create the branch for the action to work. This can be done by running the following command from your computer, in your git repository:
>
> ```
> quarto publish gh-pages
> ```
>
> Then, set the branch `gh-page` as the source of your github page, and trigger the action to check that everything works fine.

References

Computo Team. 2020. "Computo: Reproducible Computational/Algorithmic Contributions in Statistics and Machine Learning."