Table of contents

Documenting code

Documenting your code is crucial for both your future self and anyone else who might want to work with your code. You want to document your code with as much detail as you would fill out a lab book as your documentation will help others (and your future self) understand the purpose, functionality, and usage of your code.

A Guide to Reproducible Code in Ecology and Evolution gives detailed information on how to organize project folders and how to write clear and reproducible code. The examples are mainly based on R but are general enough to apply to other computational languages (and scientific disciplines).

> **Note**
>
> The information in this section is not part of the actual tutorial but was added to give you a starting point for how to document your code.
> If you follow the in-person tutorial you might want to start by recording your notes using a plain text editor but feel free to explore the more advanced options after the tutorial.

Choose Your Editor

Plain text editor

When documenting code, its best to avoid visual editors, such as Word, as these editors are not designed for writing code and easily destroy the formatting by for example changing ‘ to ’, which when writing code is quite a big difference.

Instead you can use a plain text editor, such as TextEdit (Mac) or Notepad (Windows). This is the easiest to get started but you will loose some functionality, such as separating the code from comments, adding headers or writing text in bold.

Alternatives, that offer more functionality, are for example RStudio or VScode.

Rmarkdown in RStudio

RMarkdown is an extension of Markdown (more on Markdown in a second) that allows you to integrate R code directly into your documentation.

If you have not installed R and Rstudio, follow these instructions.

In RStudio you can create an R Markdown File by:

- In RStudio, go to File -> New File -> R Markdown
- Choose a title, author, and output format
- After you are done writing your documentation you can knit the document into an HTML, PDF or word document:
  - Click the "Knit" button to render your R Markdown document into the chosen output format.

For more information visit the RMarkdown tutorial.

Quarto in Rstudio

Quarto is an alternative to RMarkdown for creating dynamic documents in RStudio that can be read by other editors, such as VScode. Compared to RMarkdown it provides enhanced features for document creation and includes many more built in output formats (and many more options for customizing each format).

It is installed by default on newer R installations. If you do not have R and Rstudio installed, follow these instructions.

- In RStudio, go to File -> New File -> Quarto document
- Choose a title, author, and output format
- Render the Document:
  - Click the "Render" button to render your R Markdown document into the chosen output format.

For more information (and more functionality) visit the Quarto website.

VSCode

Visual Studio Code (VSCode) is a versatile and user-friendly code editor. It provides excellent support for various programming languages, extensions, and a built-in terminal but might take a bit of work to setup to work with different computational languages. VSCode might take a bit longer to setup than RStudio but offers more flexibility due to various extensions that users can install.

1. Installation:
    - Download and install VSCode from here.
2. Extensions:
    - Install extensions relevant to your programming language (e.g., Python, R). These extensions enhance code highlighting and provide additional features.

Markdown for Documentation

Markdown is a lightweight markup language that is easy to read and write. It allows you to add formatting elements, such as headers, to plain text documents.

Headers:

Use `#` to add a header and separate different sections of your documentation. The more `#` symbols you use after each other, the smaller the header will be. When writing a header make sure to always put a space between the `#` and the header name.

```
# Main Header
## Subheader
```

Lists:

Use `-` or `*` for unordered lists and numbers for ordered lists.

Ordered lists are created by using numbers followed by periods. The numbers do not have to be in numerical order, but the list should start with the number one.

```
1. First item
2. Second item
3. Third item
4. Fourth item
```

```
1. First item
2. Second item
3. Third item
    1. Indented item
    2. Indented item
4. Fourth item
```

Unordered lists are created using dashes (`-`), asterisks (`*`), or plus signs (`+`) in front of line items. Indent one or more items to create a nested list.

```
- First item
- Second item
- Third item
- Fourth item
```

```
 - First item
- Second item
- Third item
    - Indented item
    - Indented item
- Fourth item
```

You can also combine ordered with unordered lists:

```
1. First item
2. Second item
3. Third item
    - Indented item
    - Indented item
4. Fourth item
```

Code Blocks:

Enclose code snippets in triple backticks followed by the computational language, i.e. bash or python, used.

```
```bash
grep "control" downloads/Experiment1.txt
```
```

```
```

Links:

You can easily add links to external resources or within your documentation as follows:

```
[Link Text](https://www.example.com)
```

Emphasis:

You can use `*` or `_` to write italic and `**` or `__` for bold text.

```
*italic*
**bold**
```

Pictures

You can also add images to your documentation as follows:

```
![Alt Text](path/to/your/image.jpg)
```

Here, replace `Alt Text` with a descriptive alternative text for your image, and `path/to/your/image.jpg` with the actual path or URL of your image.

Tables

Tables can be useful for organizing information. Here's a simple table:

```
| Header 1 | Header 2 |
| ---------| ---------|
| Content 1| Content 2|
| Content 3| Content 4|
```