Thesis / Paper Title

Thesis / Paper Subtitle

Firstname Lastname

2024-01-01

Abstract

This is the Quarto template from the DH Lab of the University of Basel. It is suitable for Seminar Papers and Master's / PhD theses, but can be adapted to fit a variety of use cases. It was created by Stefan Freitag with great help of Eric Dubey, whose Master's Thesis laid the groundwork for this template.

The abstract of your paper / thesis goes right here. It will appear on the cover sheet (very first page) of the PDF, together with the title, subtitle, author name and date.

If you do not want your document to have an abstract, you can also simple delete the 'abstract:' part with all its content and the final document will be rendered without it.

University of Basel
Digital Humanities Lab
Switzerland

Table of Contents

1 The Basics

What is Quarto? Quarto markets itself as "An open-source scientific and technical publishing system". In short, it offers a way to write articles, books or (in this case) papers and theses in markdown language. It enables a direct preview of your work and uses LaTeX for creating the final output, making it a powerful tool, if used correctly. In some ways, it is superior to writing exclusively in LaTeX, especially when working with code in the document.

Note: This is **not** a guide on how to write a paper / thesis. If you need guidance on those topics, check the appropriate DH Lab resources or ask a supervisor.

Instead, this template is meant to introduce you to what Quarto can do and give you a starting point to use it on your own. You can use Quarto via VS Code, JupyterLab and RStudio, among others. This template covers the usage of Quarto with VS Code and JupyterLab, RStudio will not be covered, but information on that topic is available here. All of those options share the same features with slight differences in usage, so the choice is mostly a matter of personal preference.

If you have questions that are not addressed within this template, Quarto offers a tutorial as well as a comprehensive guide. As you can already see in this chapter, you can put clickable links in your document.

2 Prerequisites

Here, it is assumed you are starting fresh with basically nothing installed on your machine. If you have already installed Python, VSCode, Jupyter etc., you can simply skip the corresponding step.

2.1 General requirements

- Download and install Python
- Download and install Quarto
- Install the jupyter package for Python

You can check whether everything so far is working correctly by using the command `quarto check jupyter` in your terminal.

To enable preview and export of your document as PDF, you also need to install TinyTeX:

- Run the command `quarto install tinytex` in your terminal.

2.2 Using Quarto with VS Code

- Download and install VSCode
- Install the VSCode Python Extension. You can also find it by searching the extensions in VSCode for `Python`.
- Install the VSCode Quarto Extension. You can also find it by searching the extensions in VSCode for `Quarto`.
- Install the VSCode Jupyter Extension. You can also find it by searching the extensions in VSCode for `Jupyter`.
- Open the `Template.qmd` or `Template.ipynb` file with VS Code, depending on your preferred formatting of the document file.

For further information on using VS Code to write documents with Quarto (document preview, preview formats etc.), check this chapter of the guide.

Note: The preview inside VS Code is **not** live! To see the impact of changes you have made to your document, you need to save the document file. Then, the preview will be rendered and refreshed. Also, if there are any issues with running commands including the document file, try giving the absolute path.

2.3 Using Quarto with JupyterLab

- Launch JupyterLab using the command `jupyter lab` or `python3 -m jupyter lab` (only on Mac) in the terminal. **Note**: On some Windows installations this command might not work. Try `jupyter-lab` or `py -m jupyter lab` instead.
- Open the `Template.ipynb` file with JupyterLab.

For further information on using JupyterLab to write documents with Quarto (document preview, preview formats etc.), check this chapter of the guide.

Note: The preview inside JupyterLab is **not** live! To see the impact of changes you have made to your document, you need to save the document file. Then, the preview will be rendered and refreshed. Also, if there are any issues with running commands including the document file, try giving the absolute path.

2.4 Switching between file formats

If you, for example, start to write your document in `.qmd` with but then realize you like the look of a `.ipynb` file and want to switch, you do not have to start over. You can convert a `.qmd` file to `.ipynb` or vice versa using this command: `quarto convert X`. Replace the `X` with the file of your document you wish to convert.

Note: Everything possible in one format is also possible in the other, the conversion is **lossless**!

Important: if you are saving a `.qmd` file to PDF while having a `.ipynb` with the same name as the `.qmd`, the `.ipynb` file **will** be deleted, so be careful!

3 Structure of a document

The most important file when using Quarto is your **.qmd** (Quarto markdown) or **.ipynb** (Jupyter Notebook) file. This is the file in which you write the text for your paper / thesis. Both formats are functionally identical, but look different when editing. The choice is a matter of preference.

Each file starts with a header, in which you can change metadata and other settings of the document. Images, links, references, code etc. can be embedded within the document and will be rendered when the document is output. Pages are auto-numbered, starting after the table of contents.

Let's go over the basic structure of a typical document:

4 This is a section

4.1 This is a sub-section

4.1.1 This is a sub-sub-section

And so on. Up to 5 sublevels are possible. The table of contents will be automatically generated (including proper numbering and page numbers) upon exporting your document.

You do not need to write it yourself.

5 Formatting your document

Quarto supports a variety of formatting options:

You can write in italics.

Or bold.

Or both at the same time.

You can also superscript$^{\text{text}}$ or subscript$_{\text{text}}$

You can also write lists (ordered or unordered):

- Example list

1. first
2. second
3. third

and use tables (with various formatting options):

Right	Left	Default	Center
12	12	12	12
123	123	123	123
1	1	1	1

It is possible to automatically generate a list of tables at the beginning of the document. To do this, set the `lot:` option in the document header to `true`.

You can also insert page breaks like this:

These are only some examples, please see the full section in the guide for more.

6 Using images

With Quarto, images are embedded within the text with a caption and a path to the image. This requires the images to be a local file on your computer. In this template, all images go into the `assets` folder.

This is how an image is inserted into the document:

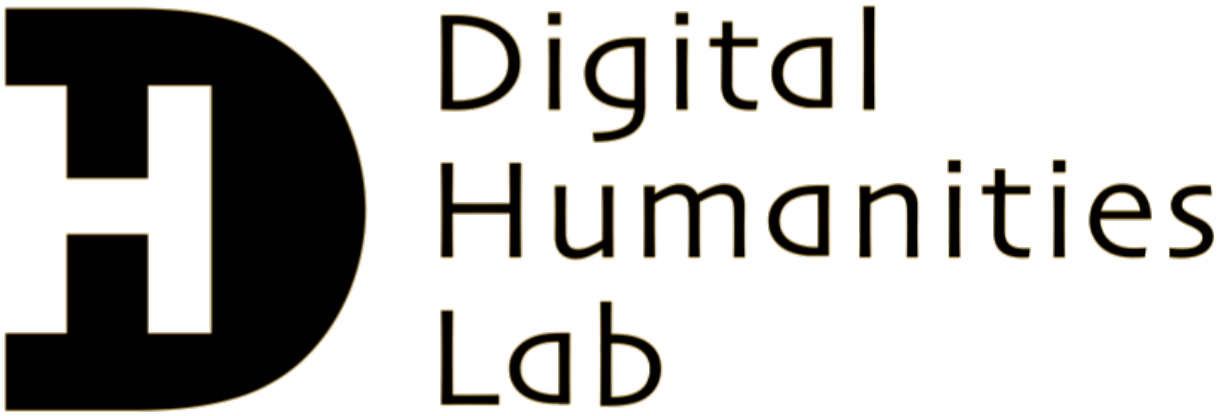

Figure 1: Logo of the DHLab

You can also resize images:

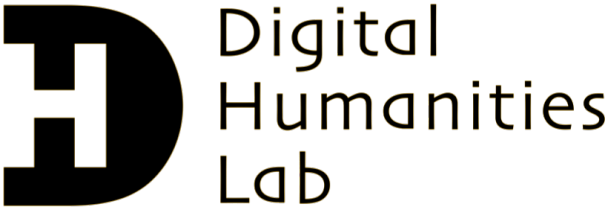

Figure 2: Logo of the DHLab, but smaller

As you can see, images with captions are numbered automatically. It is possible to generate a list of figures at the beginning of the document. To do this, set the `lof:` option in the document header to `true`.

Once again, check the corresponding guide section for further information!

7 Using code

When writing a paper or thesis in the field of Digital Humanities, it is likely that you will need to also use code within your document. Quarto supports this too, in the form of code cells:

```python
# This is a fully functional code cell
for n in range(3):
  print('This is a fully executable Python code cell! \n')
```

```
This is a fully executable Python code cell!


This is a fully executable Python code cell!


This is a fully executable Python code cell!
```

Code written this way can be fully executed within the .qmd or .ipynb file. When rendering the document, the code is included, together with its outputs, in the resulting document.

Note: This behavior can be changed globally by adding `output: false` to the `pdf:` part of the document header. This way, you can have fully executable code cells in the document file without adding the output to the final PDF.

You can also include code in this form (not executable):

```python
# This is not a functional code cell, but still recognized as Python
for n in range(3):
  print('This is not an executable code cell! \n')
```

Bear in mind that doing it this way will still give you automatic syntax highlighting! You can also do it like this to disable syntax highlighting:

```
# This is not a functional code cell and without highlighting
for n in range(3):
  print('This is not an executable code cell! \n')
```

Quarto supports some languages natively, but others will not have proper syntax highlighting after rendering. For that, you will have to use a `.xml` file, that defines the syntax for the render. This file goes into the `assets` folder and must be included in the header of the document. This is done by assigning these files to `syntax-definitions` in the header. In this template, XML files for Python and R are used to give you an example on how it works. XML files for a variety of languages can be found via GitHub.

For further information on working with code and computations, please refer to this section of the guide.

8 References

Citations and references to literature, websites and other things work like this:

Authors X and Y say a lot of things about many topics (Abdullah Lajam & Ahmed Helmy, 2021).

These guys also talk about interesting stuff on certain pages (Alniamy & Taylor, 2020, pp. 136–139).

This style of citation (footnotes) is also possible[1].

The same without further text in the footnote[2].

You can also reference other (sub) chapters from your document[3]. This goes backwards **and** forwards, so you can reference chapters after the current one, like this[4]. This of course only works if the chapters you are referring to are named correctly.

The way references are formatted in the final document depends on the chosen `.csl` file in the document header. This template uses the APA-standard by default.

Citations and references using literature are handled by the `.bib` (bibliography) file within the `assets` folder. It contains all your literature (in BibTeX format), which can then be referenced within the text.

This file is also used to automatically generate the `References` section at the end of the final document. By default, it only includes literature that is actually referenced within the text. However, in this template, this default behavior is changed to include every entry in the `.bib` file, even those that are never referenced in the document itself.

You can also include citations and references from Zotero in your document, as `zotero: true` is written in the header. This of course requires Zotero on your device.

For more information about citations and references (and changing the behavior of the `References` section), see this part of the guide.

9 Exporting your document & further options

After each save of the `.qmd` or `.ipynb` file, the document is rendered and saved. Normally, the finished PDF should be located within the `_output` folder. If it is not, check the main folder.

[1] This guy (Gauthier, 2019) also talks about stuff
[2] Gauthier (2019)
[3] See for example Using Images
[4] Exporting your document & further options

You can also download the PDF directly from the preview, as you would do with any other PDF opened in a browser.

Alternatively, you can also create a PDF by using this command:

`quarto render Filename.qmd --to pdf`. If you are using a `.ipynb` file, change the command appropriately.

As always, the guide can help you with further questions. There are also many options to change the look and behavior of the resulting PDF. These go into the header of the document. You can find them here.

References

Abdullah Lajam, O., & Ahmed Helmy, T. (2021). Performance Evaluation of IPFS in Private Networks. *2021 4th International Conference on Data Storage and Data Engineering*, 77–84. https://doi.org/10.1145/3456146.3456159

Alniamy, A., & Taylor, B. D. (2020). Attribute-based Access Control of Data Sharing Based on Hyperledger Blockchain. *Proceedings of the 2020 The 2nd International Conference on Blockchain Technology*, 135–139. https://doi.org/10.1145/3390566.3391688

Gauthier, F. (2019). *Religion, Modernity, Globalisation: Nation-State to Market*. Routledge.