

[About](#) [Projects](#) [Talks](#) [Blog](#) [Links](#)

Making Pretty PDFs with Quarto

Adding custom styling to documents makes them look more professional. This blog post will take you through the process of making a Quarto extension to create a reusable custom template for good-looking PDFs.

February 17, 2023

[Quarto](#) is an open-source scientific and technical publishing system that allows you to combine text with code to create fully reproducible documents in a variety of formats. One of those formats is PDF. The default outputs look reasonably good for academic articles, but if you're making professional reports, a CV, or sending a letter - you probably want something that just looks a bit *nicer*.

Before we begin, I do want to point out that PDFs are not accessible, and generally recommend using HTML documents instead. However, many places won't yet accept HTML documents - especially when it comes to uploading documents online. So, we might have to stick with PDF for now.

PRETTY PDFS WITH QUARTO

Quarto

Quarto enables you to weave together content and executable code into a finished document. To learn more about Quarto see <https://quarto.org>.

Running Code

When you click the **Render** button a document will be generated that includes both content and the output of embedded code. You can embed code like this:

```
1 + 1
```

[1] 2

You can add options to executable code like this

```
[1] 4
```

The `echo: false` option disables the printing of code (only output is displayed).



What are Quarto extensions?

[Quarto extensions](#) are used to modify and extend the behaviour of Quarto. Templates are just one type of Quarto extension.

Note that if you're using or developing Quarto extensions, you will most likely need to be using at least version 1.2 of Quarto.

Building a Quarto extension

Here, I've developed the Quarto extension in a GitHub repository with the following structure:

Copy

```

1 repository
2 └─ template.qmd
3 └─ _extensions
4     └─ PrettyPDF
5         └─ _extension.yml
6         └─ PrettyPDF.tex
7         └─ pagestyle.tex
8         └─ logo.png

```

The `template.qmd` is the file that will be copied over to a new directory when a user chooses to use the template, as well as the extension. Template files are optional, but it makes it easier when starting a new file if the YAML has already been filled in for a user. In this case, it's a `.qmd` file that's (almost) the same as the default template you get if you do `File --> New File --> Quarto Document` but with the format specified as `format: PrettyPDF-pdf` to make sure Quarto looks for our extension.

The `PrettyPDF.tex` file is where most of the work is being done, and is what I'll explain in the rest of this blog post. It contains LaTeX code which will be included in the header of the document, and implements the styling that we want. The (optional) `pagestyle.tex` file contains an extra little bit of LaTeX that needs to be included before the body of the document (but not in the header). The (optional) `logo.png` file is the image that will be included in the final document in the top right corner.

The `_extension.yml` file is where the extension specific information is stored. It looks a little bit like the YAML at the top of Quarto documents, in that it specifies the format (and any pre-set format options). But it also includes information on the extension - including the extension name, and the version number. For the PrettyPDF extension, it looks like this:

Copy

```

1 title: PrettyPDF
2 author: Nicola Rennie
3 version: 0.0.1
4 contributes:
5   formats:
6     pdf:
7       include-in-header:
8         - "PrettyPDF.tex"
9       include-before-body:
10        - "pagestyle.tex"
11     toc: false
12     code-block-bg: light
13     linkcolor: highlight
14     urlcolor: highlight

```

You can see where the `PrettyPDF.tex` and `pagestyle.tex` are referenced. These could be added directly in the template file, but it keeps the main `.qmd` YAML a bit tidier if they're loading in the background, and uses the same pre-sets for all documents of this type.

Let's talk about that `PrettyPDF.tex` file in a bit more detail...

Load some LaTeX packages

The first thing we need to do is load some LaTeX packages, that will allow us to implement the rest of the styling options.

Copy

```

1 % load packages
2 \usepackage{geometry}

```

```

3 \usepackage{xcolor}
4 \usepackage{eso-pic}
5 \usepackage{fancyhdr}
6 \usepackage{sectsty}
7 \usepackage{fontspec}
8 \usepackage{titlesec}

```

Now we can set up the page geometry. Here we specify that we want the paper size to be A4, and increase the margin on the right hand side. This is because I want to add a coloured bar on the right hand side of the page, and need to make sure the text doesn't overlap into that sidebar.

Copy

```

1 %% Set page size with a wider right margin
2 \geometry{a4paper, total={170mm,257mm}, left=20mm, top=20mm, bottom=20mm, right=50mm}

```

Let's also define some colours using the `xcolor` package, which takes hex colours as input (crucially, not including the `#` symbol)! Defining colours at the start makes it easier to change colours later, and easier to match up different elements with the same colour. Here, I've defined three colours: `light` (which is a pale purple) that will be used for the aforementioned sidebar and code block background; `highlight` (which is a brighter purple) that will be used for links; and `dark` which will be used for text.

Copy

```

1 %% Let's define some colours
2 \definecolor{light}{HTML}{E6E6FA}
3 \definecolor{highlight}{HTML}{800080}
4 \definecolor{dark}{HTML}{330033}

```

You *could* also define a colour for the page background but I'd recommend against that most of the time. Some people still print PDFs - and you don't want to be the person who sends in their CV with a dark background colour and only half of gets printed because there's not enough ink in the printer...

So let's get onto adding that sidebar! This LaTeX code adds a coloured bar (with the `light` colour) on the right hand side which spans the entire height of the page, and is 3cm wide.

Copy

```

1 %% Let's add the border on the right hand side
2 \AddToShipoutPicture{%
3   \AtPageLowerLeft{%
4     \put(\LenToUnit{\dimexpr\paperwidth-3cm},0){%
5       \color{light}\rule{3cm}{\LenToUnit\paperheight}%
6     }%
7   }%
8 }

```

If we want to add a logo in the (top right) corner, we can edit the above code instead to be:

Copy

```

1 %% Let's add the border on the right hand side and the logo in the top right corner
2 \AddToShipoutPicture{%
3   % Right bar
4   \AtPageLowerLeft{%
5     \put(\LenToUnit{\dimexpr\paperwidth-3cm},0){%
6       \color{light}\rule{3cm}{\LenToUnit\paperheight}%
7     }%
8   }%
9   % Logo

```

```

10 \AtPageLowerLeft{%
11     \put(\LenToUnit{\dimexpr\paperwidth-2.25cm},27.2cm){%
12         \color{light}\includegraphics[width=1.5cm]{_extensions/nrennie/PrettyPDF/logo.png}
13     }%
14 }%
15 }

```

The second part of this LaTeX code, adds the file `logo.png` to the top right corner, 2.25cm from the side of the page, and is 1.5cm wide (which ensures it's centered on the 3cm sidebar). Here, the file extension is relative to where the main `.qmd` file is after the extension has been installed by a user.

Since I have a sidebar, I'd prefer if the page numbers were in that sidebar, rather than in the middle of the page (where they normally are by default). This LaTeX code defines a new page style that pushes the page number to the right hand side, and also increase how far it is from the bottom of the page.

Copy

```

1 %% Style the page number
2 \fancypagestyle{mystyle}{
3 \fancyhf{}
4 \renewcommand\headrulewidth{0pt}
5 \fancyfoot[R]{\thepage}
6 \fancyfootoffset{3.5cm}
7 }
8 \setlength{\footskip}{20pt}

```

To get this to work, we need to include `\pagestyle{mystyle}` just before our document content - that's what's stored in the `pagestyle.tex` file. Again, it could be included in the `template.qmd` file, but this approach means users don't see what they don't need to.

Finally, I want to deal with the font: colours and font family. Let's start of with changing the colour of all the section title font to our dark colour. I also added an underline to subsection (`##`) headings.

Copy

```

1 %% style the chapter/section fonts
2 \chapterfont{\color{dark}\fontsize{20}{16.8}\selectfont}
3 \sectionfont{\color{dark}\fontsize{20}{16.8}\selectfont}
4 \subsectionfont{\color{dark}\fontsize{14}{16.8}\selectfont}
5 \titleformat{\subsection}
6 {\sffamily\Large\bfseries}{\thesection}{1em}{}[\titledrule[0.8pt]]

```

I also want to left align the title, subtitle, and author fields, since it looks better with the sidebar on the right:

Copy

```

1 % left align title
2 \makeatletter
3 \renewcommand{\maketitle}{\bgroup\setlength{\parindent}{0pt}
4 \begin{flushleft}
5 {\sffamily\huge\textbf{\MakeUppercase{\@title}}} \vspace{0.3cm} \newline
6 {\Large {\@subtitle}} \newline
7 \@author
8 \end{flushleft}\egroup
9 }
10 \makeatother

```

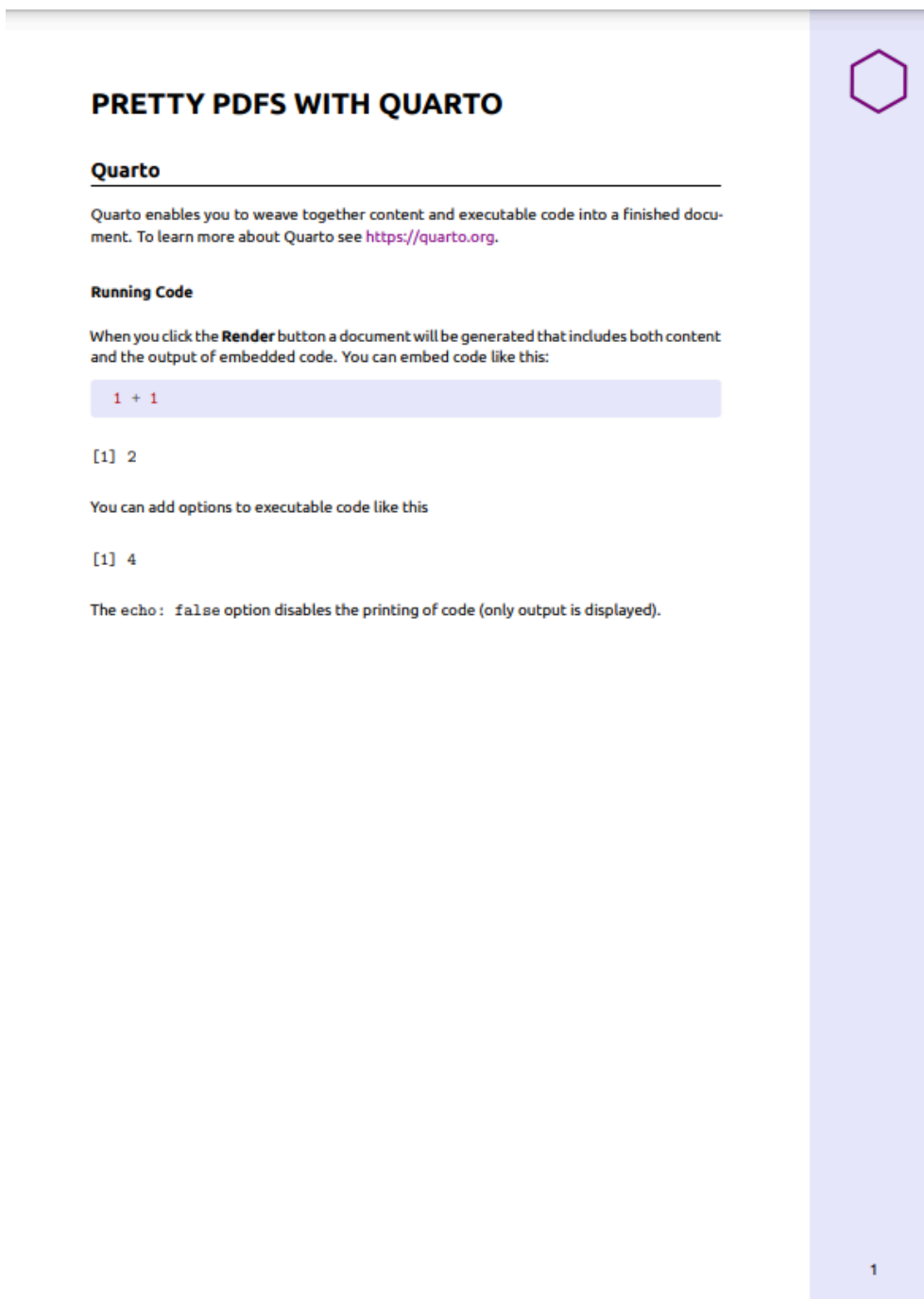
Now we just need to change the fonts that are used. Here, I've used the Ubuntu font which I've downloaded and stored in the `_extensions/Ubuntu` directory. LaTeX has some built-in font rules, so here I've set the

default sans serif font to be Ubuntu, as well as the main font. Again, the file paths here are relative to the main .qmd file that a user will be editing.

Copy

```
1 %% Use some custom fonts
2 \setsansfont{Ubuntu}[
3   Path=_extensions/nrennie/PrettyPDF/Ubuntu/,
4   Scale=0.9,
5   Extension = .ttf,
6   UprightFont=*-Regular,
7   BoldFont=*-Bold,
8   ItalicFont=*-Italic,
9 ]
10
11 \setmainfont{Ubuntu}[
12   Path=_extensions/nrennie/PrettyPDF/Ubuntu/,
13   Scale=0.9,
14   Extension = .ttf,
15   UprightFont=*-Regular,
16   BoldFont=*-Bold,
17   ItalicFont=*-Italic,
18 ]
```

And that's everything that's in the `PrettyPDF.tex` file! This gives us a PDF that looks a bit like this:



You can download a copy of the template PDF [here](#), and view the source code on [GitHub](#).

Using this extension

If you want to use this extension in your own projects, please feel free to do so! You can install the extension using the command line:

To install the Quarto extension, create a directory, and use the template file:

Copy

```
1 quarto use template nrennie/PrettyPDF
```

To use the extension in an existing project without installing the template file:

Copy

```
1 quarto install extension nrennie/PrettyPDF
```

Note that you will need to update the output format of your existing .qmd file to format: PrettyPDF-pdf to enable use of the extension. You can add any additional PDF options to the PrettyPDF-pdf using, for example:

Copy

```
1 format:
2   PrettyPDF-pdf:
3     keep-tex: true
```

Adapting this extension

If you want to update this template to use, for example, different colours or a different logo, you have two options:

- Install the extension using the instructions above, then edit the `_extensions/nrennie/PrettyPDF/PrettyPDF.tex` file.
- Make a fork of the [GitHub repository](#), and update the `_extensions/PrettyPDF/PrettyPDF.tex` file. You can then install the extension from your own GitHub.

Changing the logo

Either replace the `logo.png` file in the `_extensions` directory with a new file of your choosing (with the same name), or change the file path on line 28 of `PrettyPDF.tex` to point to a different logo file. Note that the file path is relative to your .qmd file. You can remove lines 25-30, if you'd rather not have a logo at all.

Changing the colours

Lines 14-16 of `PrettyPDF.tex` define three colours used in the template: `light`, `dark`, and `highlight`. Change the hex colours in these lines to update the colours. The `light` colour changes the sidebar and code block background colours. The `dark` colour changes the text colour, and the `highlight` colour changes the link colours.

There will almost certainly be some elements of the default Quarto PDF theme that I haven't adjusted here, simply because I haven't had the need to yet. Feel free to make your own adjustments!

Further resources

If you're looking for more Quarto extensions, I'd highly recommend checking out the [Awesome Quarto](#) repository - it has links to lots of Quarto resources including talks, tools, examples, and articles.

If you want to know how to distribute your Quarto extension as part of an R package, instead of through GitHub, this [blog post](#) from [Spencer Schien](#) will be very useful.

This [blog post](#) from [Meghan Hall](#) gives some great tips for customising Quarto PDFs, including parameterised reports so you can change the styling of your report based on parameters, and your data.

Now you're ready to go and create some beautiful looking PDFs with Quarto!



Image: giphy.com

For attribution, please cite this work as:

Making Pretty PDFs with Quarto.

Nicola Rennie. February 17, 2023.

nrennie.rbind.io/blog/making-pretty-pdf-quarto

Licence: creativecommons.org/licenses/by/4.0

19 Comments - powered by *utteranc.es*

collioud commented on 20. Feb. 2023

Hello Nicola,

Thanks for this pretty PrettyPDFs extension, which looks really nice!

Just a quick note : I think you have to switch the "quarto use template nrennie/PrettyPDF" and the "quarto install extension nrennie/PrettyPDF" code lines to fit to the text...

Cheers,

Arnaud

nrennie commented on 20. Feb. 2023

Owner

Thanks Arnaud! I'm not quite sure what you mean by "fit to the text"? Using the `install` extension copies the `_extensions` folder, whereas `use template` does the same thing but additionally copies the `template.qmd` folder into a new directory for you.

collioud commented on 20. Feb. 2023

My mistake... I thought that "quarto install" was for (globally) installing a template/extension and "quarto

use" for using it in a project. Now your text makes perfectly sense to me!

ibecav commented on 6. März 2023

Just a quick thanks for a great post and a great extension. The sincerest form of flattery -- I was able to follow along put it to use and then modify a version for my own tastes!



nrennie commented on 8. März 2023

Owner

I'm glad it was useful!

fjodor commented on 28. März 2023

Thanks, very helpful and appreciated!

Are you also printing reveal.js slides to PDF using R and quarto? I enthusiastically switched from xaringan to quarto presentations and really liked reveal.js in html format. However, I need a PDF version as well, and had too many ugly slides (e. g. some images missing in an html table in PDF, others shown) that I decided to move back to xaringan for the time being, until quarto and reveal.js will hopefully play more nicely to PDF.

Any thoughts / experiences?

nrennie commented on 14. Apr. 2023

Owner

Thanks @fjodor! Yes, I do quite often make a PDF version of HTML slides (always useful to have an backup!). I've found some issues converting to PDF when I have self-contained/embed-resources set to true (unsure why), so might be worth toggling between true/false to see if that helps. Another thing I've found useful is using a fixed width/height for revealjs presentations.

jstrappa commented on 25. Apr. 2023

Thank you for this extension and for the easy-to-follow explanation. The default style for PDFs is just ugly. :) I can't understand why Quarto doesn't have themes for the PDF output, maybe it has to do with the accessibility issues? Is it that PDFs are falling into disuse in favor of HTML? But I'm not sure how I'd use HTML for documents that need to be self-contained. I tried that once and the recipients were unable to open the file.

nrennie commented on 25. Apr. 2023

Owner

Thanks! It might be a combination of accessibility, interactivity, and perhaps that PDF output was maybe originally aimed at academics?

You can set `self-contained: true` (or the slightly newer `embed-resources: true`) in the YAML to make your HTML document self contained. They don't usually preview well in email apps, but if recipients download the file it should open in a web browser. I haven't had any issues sharing self-contained HTML files with Quarto yet

chonasson commented on 27. Apr. 2023

This is really helpful, thank you Nicola.

I was wondering how I would have to proceed if I were to create a template for a multi-page document, such as a report. Do you have any tips or a link to more advanced resources?

nrennie commented on 4. Mai 2023

Owner

Thanks! This template will also work for multipage documents - it will apply the same styling to each page. There are LaTeX functions that can change to styling for odd and even pages if you want put e.g., the page number on different sides - this would be a reasonably small change to this template. The other thing that's common is to also include a differently styled title page - there's some examples here: <https://www.overleaf.com/gallery/tagged/title-page> that you can look at the source code for.

♥ 1

myaseen208 commented on 27. Mai 2023

Thank you for the helpful template. I have two questions: firstly, I would like to know how to have separate page numbers for the frontmatter and mainmatter sections; and secondly, I am interested in learning how to increase the spacing between the chapter number and chapter title. I appreciate your assistance.

aito123 commented on 10. Juli 2023

Hi Nicola, thanks so much for the explanation and the resources at the end. You really make the data science path more fun!
Take care.

♥ 1

nrennie commented on 19. Juli 2023


Owner

@myaseen208 Thanks! For the different page numbering, you could use something like `\pagenumbering{roman}` for the frontmatter, and then switch to a different type for your main document. There's some more details here: https://www.overleaf.com/learn/latex/Page_numbering

For the chapter number spacing, I'm not too sure. You probably want to do something like `\titleformat{\chapter}{hang}{\thechapter\hspace{3pt}}` You could maybe adapt this code for styling


`\section{main{chapter}[hang]{\renewcommand{\chapter}{\chapter{ope}}}` `\renewcommand{\chapter}{\chapter{ope}}` `\renewcommand{\chapter}{\chapter{ope}}`

chapter numbers: <https://tex.stackexchange.com/questions/139366/chapter-header-with-super-huge-numbers>

 1


dmpimentel commented on 6. Dez. 2023

Hi! This template is so pretty! Thank you for sharing. I'm going to use your explanations and try to make one my own. Thank you :)

 1

MarkHanly commented on 8. Jan. 2024

So useful, thank you for sharing!

 1

DavidPatShuiFong commented on 30. Jan. 2024

Thanks Nicola! this is a really nice template, and I am planning to use them as a basis for monitoring/evaluation reports for a non-profit child development organisation.

Is there a way to add author affiliations details? I tried to modify/add the code of `PrettyPDF.tex` using `authblk` as per [Why do affiliations not show up anywhere in the pdf output of quarto?](#), but I wasn't successful!

nrennie commented 3 months ago

Owner

Thanks @DavidPatShuiFong! You would need to add that in a separate .tex file and pass it into the yaml as:

```
template-partial:
  - title.tex
```

The PrettyPDF.tex file isn't a LaTeX template file so it goes in the yaml as:

```
include-in-header:
  - PrettyPDF.tex
```

Hope that helps!

 1

sebastianmarinc commented a month ago

Hi Nicole, thanks so much for creating this extension! I'm actually having trouble with installation on my Windows machine. For some reason when trying to knit my doc, there's a latex error: it can't find `eso-pic.sty'. Have you encountered this issue? Any help would be much appreciated!

Write

Preview

Sign in to comment

 Styling with Markdown is supported

Sign in with GitHub

[← What's new in {PrettyCols} 1.0.1? Scraping London Marathon data with {rvest} →](#)

© 2024 Nicola Rennie. Made with [Hugo Apéro](#).