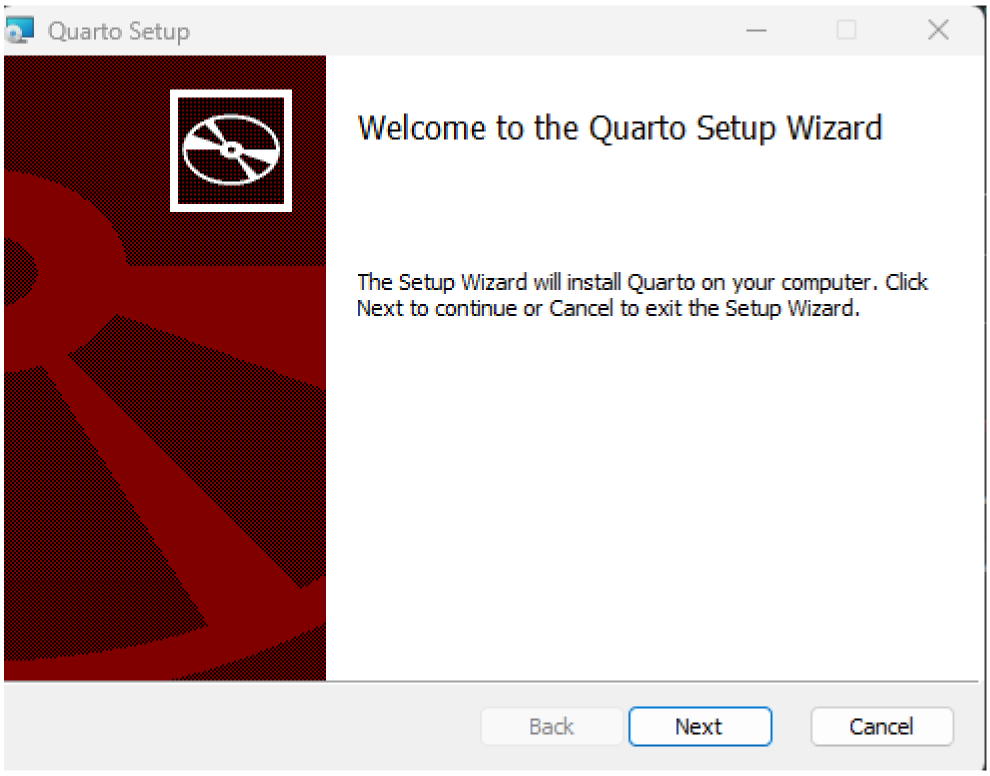
QUARTO BASICS

Table of contents

Getting Started with Quarto

Installation

1. Download Quarto installation wizard & install.



The Setup Wizard will install Quarto on your computer. Click Next to continue or Cancel to exit the Setup Wizard.

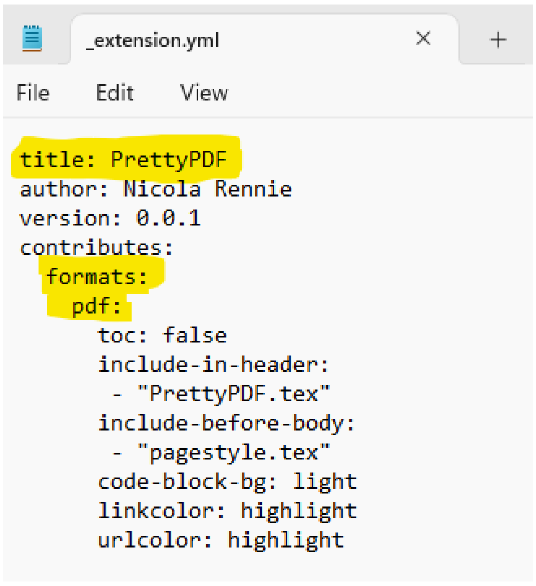
Figure 1: Quarto download wizard

2. Close all `cmd`, `powershell` and `jupyter` instances before using Quarto.
3. Create a Jupyter notebook (e.g. `hello.ipynb`).
4. Open a `cmd` window, navigate to the directory in which your Jupyter notebook is located.
5. Run command `quarto render {notebook_name} --to {file_type}`, e.g. where `notebook_name` is the name of the notebook, and `file_type` is a valid tile type (e.g. html, pdf) or extension name (e.g. `PrettyPDF-pdf`). When rendering our notebook `hello.ipynb` to pdf the full command would be as follows: `quarto render hello.ipynb --to pdf`.

Quarto Extensions

Using Extensions in Quarto

1. Create a directory `_extensions` in your Quarto project folder>
2. Download the folder containing the extension files into the `_extensions` folder. E.g. download files from PrettyPDF into folder `_extensions/nrennie/PrettyPDF`.
3. To use the extension for rendering, open a `cmd` window, navigate to the the Quarto project folder (the one in which you created the `_extensions` folder), and then run the command `quarto install extension {src_dir}`. Thereby, `src_dir` is a sub directory of the `_extensions` folder in which the `_extension.yml` for the extension is located. E.g. in our example `nrennie/PrettyPDF.tex`.
4. In a `cmd` window, navigate to the directory in which your Jupyter notebook (e.g. `hello.ipynb`) is located.
5. Run command `quarto render {notebook_name} --to {extension_name}-{format}`. Thereby {extension_name} is the name of the extension (the attribute **title** in the `_extension.yml`), and {format} one of the formats defined in the attribute **formats** in the `_extension.yml`. In our example, e.g. `quarto render hello.ipynb --to PrettyPDF-pdf`.



```
title: PrettyPDF
author: Nicola Rennie
version: 0.0.1
contributes:
  formats:
    pdf:
      toc: false
      include-in-header:
        - "PrettyPDF.tex"
      include-before-body:
        - "pagestyle.tex"
      code-block-bg: light
      linkcolor: highlight
      urlcolor: highlight
```

Figure 2: Quarto extensions.yml

TEX Definitions for Custom LaTeX Environments in Quarto

To incorporate custom LaTeX commands and environments into a Quarto-rendered PDF, it's essential to define these environments within your LaTeX template or preamble. Typically, the LaTeX template is specified in the extension's configuration file, referenced under the `include-in-header` attribute in the `_extension.yml` file. For instance, in the `PrettyPDF` extension, this is usually a `.tex` file named `PrettyPDF.tex`.

Setting Up a `warning` Environment Below is an example of how you can set up a `warning` environment in your LaTeX template:

```
\usepackage{mdframed}
\newmdenv[linecolor=red,backgroundcolor=yellow!20]{warning}
```

This code snippet uses the `mdframed` package to create a new environment named `warning`, characterized by a red line border and a light yellow background.

Utilizing the `warning` Environment in a Markdown Cell After establishing the `warning` environment, you can use it within a markdown cell in your Quarto document as follows:

```
\begin{warning}
Simple warning text.
\end{warning}
```

This markdown syntax instructs Quarto to process the enclosed text as LaTeX, rendering it within the defined `warning` environment. The resulting output in the PDF will be a styled box containing your warning message, visually distinguishing it from the d effectiveness of their documents.:nts.:

Simple warning text.

TEX Fonts

TeX fonts can be downloaded from here.

Matplotlib

For a demonstration of a line plot on a polar axis, see Figure 3.

```
import numpy as np
import matplotlib.pyplot as plt

r = np.arange(0, 2, 0.01)
theta = 2 * np.pi * r
fig, ax = plt.subplots(
  subplot_kw = {'projection': 'polar'}
)
ax.plot(theta, r)
ax.set_rticks([0.5, 1, 1.5, 2])
ax.grid(True)
plt.show()
```

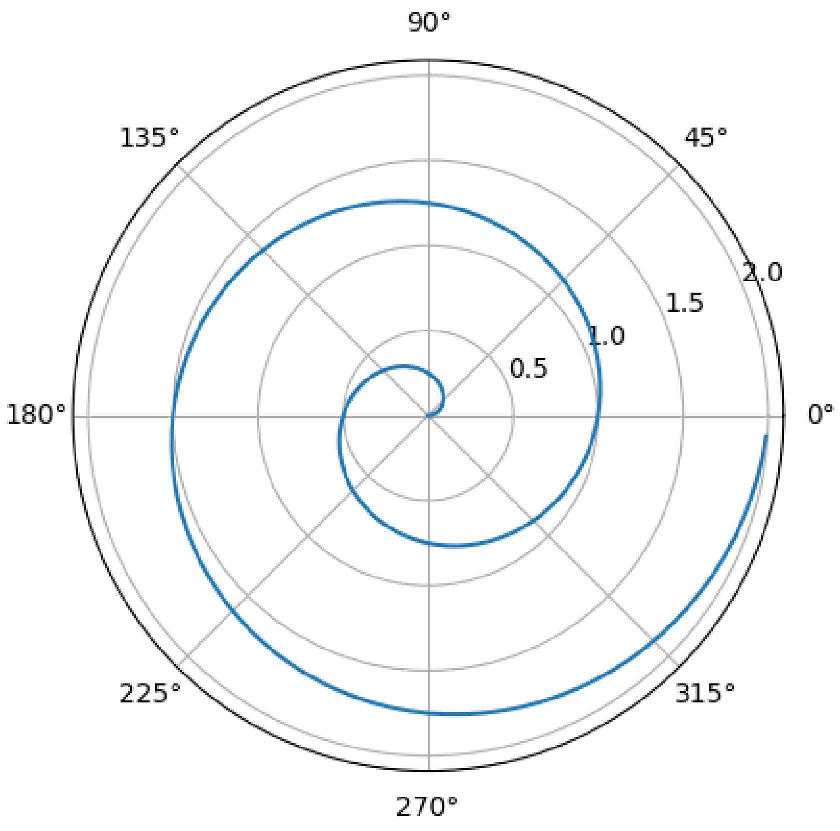


Figure 3: A line plot on a polar axis

Plotly

Pre-execution Requirement for Jupyter Notebooks

When incorporating Plotly charts into a Quarto-rendered document from a Jupyter Notebook, it is essential to first execute the notebook in **Jupyter Lab** underlinenot *Jupyter Notebook*. This pre-execution step ensures that all Plotly charts are fully rendered and available for inclusion in the final output.

Key Steps:

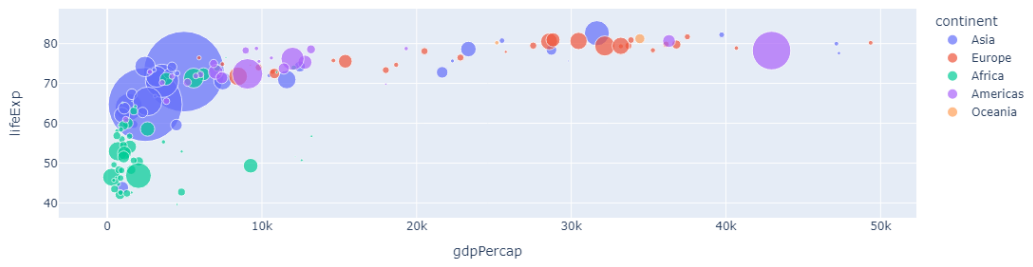
1. Open your Jupyter Notebook containing the Plotly charts in **Jupyter Lab**.
2. Run all cells to ensure that the Plotly charts are rendered within the notebook.
3. Save the notebook with the executed cells and rendered outputs.
4. Proceed with Quarto rendering of the notebook to your desired format (e.g., PDF, HTML).

By following these steps, Plotly charts are correctly embedded in the Quarto-rendered document, preserving their intended visual representation.

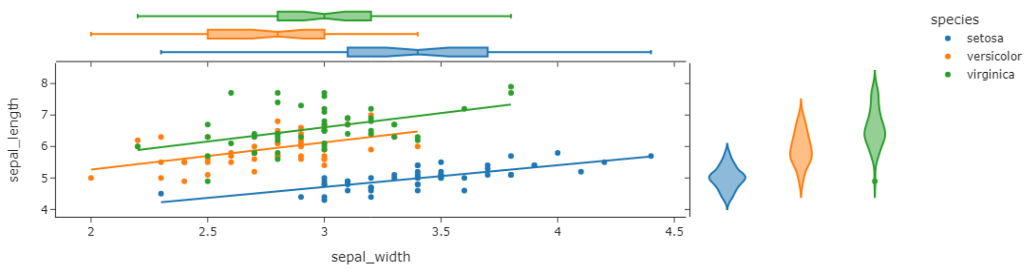
```
import plotly.express as px
import plotly.io as pio
```

```
gapminder = px.data.gapminder()
gapminder2007 = gapminder.query("year == 2007")
fig = px.scatter(gapminder2007,
                 x="gdpPercap", y="lifeExp", color="continent",
                 size="pop", size_max=60,
                 hover_name="country")
fig.show()
```

```
Unable to display output for mime type(s): text/html
```



```
df = px.data.iris()
fig = px.scatter(df, x="sepal_width", y="sepal_length",
                 color="species",
                 marginal_y="violin", marginal_x="box",
                 trendline="ols", template="simple_white")
fig.show()
```


species
setosa
versicolor
virginica
sepal_length
sepal_width