Week 1: Introduction to Quarto for R and Python

Joan Robinson

Table of contents

1 Learning Goals **1**
1.1 Intro to Quarto 1
1.2 My personal TLDR about Quarto 2
1.3 Installing 2
1.4 Quick tutorial for RStudio 2
1.5 A minimal example of Quarto for Python 3

1 Learning Goals

The purpose of this notebook is to show case the use of `Quarto` notebooks. It will cover:

- Brief introduction to Quarto Notebooks
- Running R with Quarto Notebooks
- Running Python with Quarto Notebooks + `reticulate`

See Quarto's website for a more in-depth coverage of the framework.

1.1 Intro to Quarto

Quarto is the next-generation version of R Markdown. If you're an R Markdown user, you will see how Quarto is just an extension of the capabilities that were previously provided by R Markdown. Now, instead of `.rmd` files, we have `.qmd` files.

As in R Markdown, Quarto unified authoring framework. It combines **code, results, and text**. These combinations can be rendered as PDFs, Word Files, HTMLs, and more. Quarto use case. Quarto use cases include:

- Technical reports that demonstrate or utilize the features of a current code base.
- Slide decks that provide an overview of up-to-date data and are generated on a regular basis.
- Sharing exploratory or prototype analyses with co-authors or collaborators. Writing blogs for blogging platforms that accept .md files (markdown format).
- Authoring websites, like our course website.

1.2 My personal TLDR about Quarto

Quarto was developed by Posit (the new name of RStudio). It represents the effort of RStudio to become a language agnostic IDE for Data Science.

Quarto main promise is to allow data scientists to run many different languages using the same notebook structure.

For the purpose of our course, we will keep iterating between Jupyter and Quarto Notebooks. This is important because Jupyter Notebooks are dominant among Python developers.

However, as you are learning Python and R at the same time throughout your DSPP courses, I strongly advice you to be confortable with Quarto and use it the same tool to run both R and Python.

1.3 Installing

Quarto is the notebook framework for Rstudio. You can also use Quarto inside of JupyterLab, but that seems absolutely inneficient for me.

To use Quarto, you should then just install RStudio:

- RStudio

1.4 Quick tutorial for RStudio

This is what RStudio looks like when you open it for the first time.

```
knitr::include_graphics("images/rstudio_.png")
```

```
Error in knitr::include_graphics("images/rstudio_.png"): Cannot find the file(s): "images/rst
```

1.4.0.1 Top left pane (input/script)

This is your code editor. Here you enter code in any file type (.py, .r, .qmd) you are working on. If not working with notebooks, this is just gonna be a plain text file but with a extension that run the commands.

For example, enter `2 + 2` in your script and run a line of code by pressing `command + enter` (Mac) or `Ctrl + enter` (PC). This is a huge advantage of Rstudio over Jupyter. You can run your code line by line, instead of running the entire cell.

1.4.0.2 Bottom left pane (output/console)

This is the console. It is pretty much like when you open Python/R from the Command line.

In the console, the prompt `>` looks like a greater than symbol. If your prompt begins to look like a `+` symbol by mistake, simply click in your console and press the `esc` key on your keyboard as many times as necessary to return to the prompt.

Rstudio uses `+` when code is broken up across multiple lines and is still expecting more code. A line of code does not usually end until Rstudio finds an appropriate stop parameter or punctuation that completes some code such as a closed round parenthesis `)`, square bracket `]`, curly brace `}`, or quotation mark `'`.

If the output in your console gets too messy, you can clear it by pressing `control + l` on both Mac and PC. This will not erase any saved data - it will simply make your console easier to read.

1.4.0.3 Top right pane (global environment)

This is your environment pane. All objects you create will be displayed here.

1.4.0.4 Bottom right pane (files, plots, packages, and help)

Here you find useful tabs for navigating your file system, displaying plots, installing packages, and viewing help pages. Press the `control` key and a number (1 through 9) on your keyboard to shortcut between these panes and tabs.

1.5 A minimal example of Quarto for Python

```
---
title: "Example Report"
author: "Joan Robinson"
format: pdf
```

````
toc: true
number-sections: true
jupyter: python3
---

## Polar Axis

For a demonstration of a line plot on a polar axis, see @fig-polar.

::: {.cell}

```{.python .cell-code}
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()
````
````

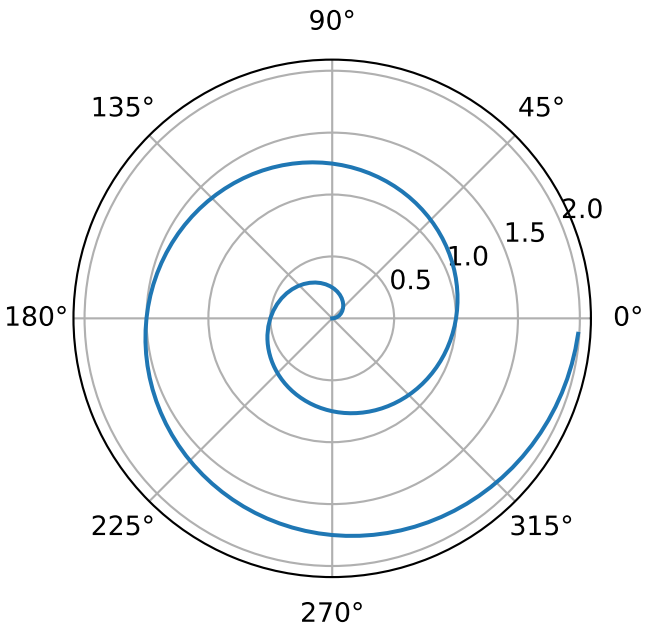

Figure 1: A line plot on a polar axis

:::