Introduction to R + Python + Quarto

Aaron Geller

Northwestern | INFORMATION TECHNOLOGY

What is Quarto and why should I use it?

Quarto is an open-source, multi-language, multi-format, technical publishing system. For those familiar with R, Quarto is a more generalized R Markdown.

Why use Quarto?

- Language-agnostic document creation (e.g., Python + R in same doc).
- Reproducibility and dependency management with your package manager (e.g., conda, renv, etc)
- Multiple output formats: e.g. (HTML, PDF, MS Word, and more)
- Many available themes (from Bootwatch via bslib)
- Publishing and sharing (e.g., on GitHub pages, and many others)

Installation

1. **Install Quarto:**

- Documentation from Quarto can be found here and here. (I downloaded the latest version from their GitHub repo here. Then I added the executable from <base_dir>/src/bin/ to my PATH variable.)

1. **Install Python and R and related libraries**

- I recommend that you use conda and create an environment for this.

```
conda create --name quarto-env
conda activate quarto-env
conda config --add channels conda-forge
conda config --set channel_priority strict
conda install python=3.10 r-base=4.1.3 pandas matplotlib numpy seaborn r-rmarkdown r-
   reticulate
```

Basic Code Structure

doc.qmd

````
---
# yaml header
---

## Markdown text

```{r}
# R code chunk
```

```{python}
# Python code chunk
```
````

Basic Code Structure

doc.qmd

````
---
# yaml header
---

## Markdown text

```{r}
# R code chunk
```

```{python}
# Python code chunk
```
````

- Include document-wide settings

```yaml
---
title: "Quarto Basics"
author: "Aaron M. Geller"
date: "12/12/2023"
number-sections: true
format:
    html:
        code-fold: false
        code-tools: true
theme: darkly
toc: true
---
```

Basic Code Structure

doc.qmd

````
---
# yaml header
---

## Markdown text

```{r}
# R code chunk
```

```{python}
# Python code chunk
```
````

- Include document-wide settings
- Throughout the doc you can add any text using Markdown

Basic Code Structure

doc.qmd

````
---
# yaml header
---

## Markdown text

```{r}
# R code chunk
```

```{python}
# Python code chunk
```
````

- Include document-wide settings
- Throughout the doc you can add any text using Markdown
- R code can be included

````
```{r}
#| label: r-code-1

library(reticulate)

# define a simple variable as a test
x <- 123

```
````

Basic Code Structure

doc.qmd

````
---
# yaml header
---

## Markdown text

```{r}
# R code chunk
```

```{python}
# Python code chunk
```
````

- Include document-wide settings
- Throughout the doc you can add any text using Markdown
- R code can be included
- Python code can be included

````
```{python}
#| label: python-code-1

# multiply the x value from r by 2
y = r.x*2.

```
````

How to compile the doc

Option 1 : From your terminal in the directory with your .qmd file

```
quarto render quarto_example.qmd --to html
```

Option 2 : From RStudio

- Launch RStudio and then open your application file (e.g., "`doc.qmd`")
- Click the Render button.

Examples

1. Basics of how to combine Python and R code
2. More in depth example exploring Valentine's Day data

Example 1

Basics of how to combine Python and R code

- To reference R variables in Python and vice versa:
  - Load the `reticulate` R library.
  - Reference R variables in Python using `r.<variable_name>`
  - Reference Python variables in R using `py$<variable_name>`
  - (Replace `<variable_name>` with the actual name.)

theme

using R variables in Python

using Python variables in R

```yaml
---
title: "Quarto Basics"
author: "Aaron M. Geller"
date: "12/12/2023"
number-sections: true
format:
    html:
        code-fold: false
        code-tools: true
theme: darkly
toc: true
---
```

Start in R
Run Cell | Run Next Cell

```{r}
#| label: r-code-1

library(reticulate)

# define a simple variable as a test
x <- 123

```

Use variables from R in Python
Run Cell | Run Next Cell | Run Above

```{python}
#| label: python-code-1

# multiply the x value from r by 2
y = r.x*2.

```

Use variables from Python in R
Run Cell | Run Next Cell | Run Above

```{r}
#| label: r-code-2

# add to the y variable from python
z <- py$y + 7
```

- **Exercise 1.1:** change theme picking from these options (recompile)
- **Exercise 1.2:** add a new Python and/or R code section (recompile)

Example 1

Basics of how to combine Python and R code

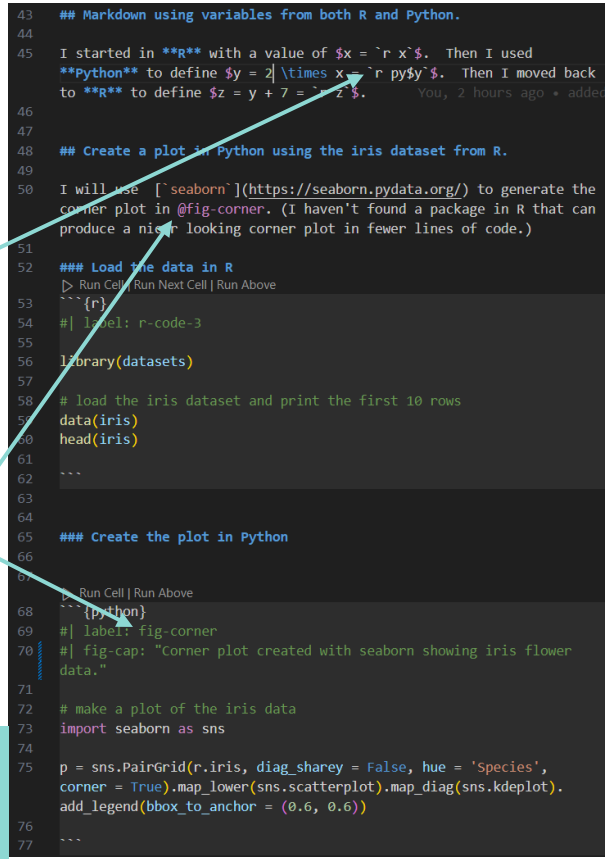
- To reference variables in Markdown:
  - (Load the `reticulate` R library.)
  - Use the same syntax as for using variables in code, but wrap in an r code statement, e.g. `` `r x` `` or `` `r py$y` ``
- To label figures and reference them in Markdown
  - include a `#| label fig-<name>` in the code cell that contains the figure
  - reference that label using `@fig-<name>`
  - Replace `<name>` with a single word (no dashes)

- **Exercise 1.3:** create a new plot in either R or Python, label it and reference the figure in Markdown (recompile)

using R/Python variables in Markdown

labeling figures and referencing them in Markdown

(Code continued)



````
## Markdown using variables from both R and Python.

I started in **R** with a value of $x = `r x`$.  Then I used
**Python** to define $y = 2 \times x = `r py$y`$.  Then I moved back
to **R** to define $z = y + 7 = `r z`$.

## Create a plot in Python using the iris dataset from R.

I will use [`seaborn`](https://seaborn.pydata.org/) to generate the
corner plot in @fig-corner. (I haven't found a package in R that can
produce a nicer looking corner plot in fewer lines of code.)

### Load the data in R

```{r}
#| label: r-code-3

library(datasets)

# load the iris dataset and print the first 10 rows
data(iris)
head(iris)
```

### Create the plot in Python

```{python}
#| label: fig-corner
#| fig-cap: "Corner plot created with seaborn showing iris flower
data."

# make a plot of the iris data
import seaborn as sns

p = sns.PairGrid(r.iris, diag_sharey = False, hue = 'Species',
corner = True).map_lower(sns.scatterplot).map_diag(sns.kdeplot).
add_legend(bbox_to_anchor = (0.6, 0.6))
```
````

Example 2

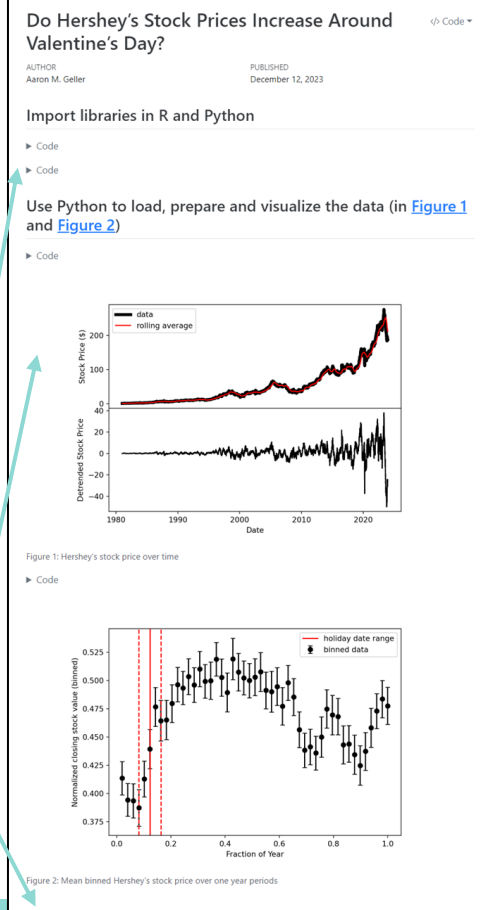
Exploring Valentine's Day data

- A few reasons why you may want to use both Python and R:
  - You prefer Python for data manipulation, but R has well verified stats packages
  - You prefer R for most things, but you want to run a simulation that is available in Python
  - You find a plotting or table generation package that is particularly easy to use in one language (e.g., Python's `seaborn`) but prefer data manipulation in the other language
  - You want an opportunity to learn R/Python!

Code is initially collapsed; can be expanded by user

Initial data prep and plotting in Python

Statistical analysis in R



- **Exercise 2.1:** add a new code cell in either Python or R and print the first few lines of the dataframe.
- **Exercise 2.2:** render this to a different format (e.g., html, pdf, etc.)

Further

- Interactive figures, e.g. using `plotly`
- Widgets controlling interactive figures, e.g., using `Shiny`
- Complex layouts are possible, e.g., to create dashboards
- Many more possibilities on the Quarto documentation