Introduction to
R + Python + Quarto

Aaron Geller

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. For Python folks, it might be easiest to follow the instructions to downloaded the latest .tar.gz version from their GitHub repo (see instructions), and then add the executable from <base_dir>/src/bin/ to your PATH variable. For R folks, you can do the installation all within RStudio.

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

● For R folks, this can be done within RStudio.

● For Python folks, I recommend that you use conda and create an environment.

```
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
```{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.

Render

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

```
1   ---
2   title: "Quarto Basics"
3   author: "Aaron M. Geller"
4   date: "12/12/2023"
5   number-sections: true
6   format:
7       html:
8           code-fold: false
9           code-tools: true
10  theme: darkly
11  toc: true
12  ---
13
14  💡 Start in R
    ▷ Run Cell | Run Next Cell
15  ```{r}                You, 44 minutes ago • added basic example
16  #| label: r-code-1
17
18  library(reticulate)
19
20  # define a simple variable as a test
21  x <- 123
22
23  ```
24
25  ## Use variables from R in Python
    ▷ Run Cell | Run Next Cell | Run Above
26  ```{python}
27  #| label: python-code-1
28
29  # multiply the x value from r by 2
30  y = r.x*2.
31
32  ```
33
34
35  ## Use variables from Python in R
    ▷ Run Cell | Run Next Cell | Run Above
36  ```{r}
37  #| label: r-code-2
38
39  # add to the y variable from python
40  z <- py$y + 7
41  ```
42
```

- **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


```
43  ## Markdown using variables from both R and Python.
44
45  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 $z = y + 7 = `r z`$.          You, 2 hours ago • added
46
47
48  ## Create a plot in Python using the iris dataset from R.
49
50  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.)
51
52  ### Load the data in R
    ▷ Run Cell | Run Next Cell | Run Above
53  ```{r}
54  #| label: r-code-3
55
56  library(datasets)
57
58  # load the iris dataset and print the first 10 rows
59  data(iris)
60  head(iris)
61
62  ```
63
64
65  ### Create the plot in Python
66
67     ▷ Run Cell | Run Above
68  ```{python}
69  #| label: fig-corner
70  #| fig-cap: "Corner plot created with seaborn showing iris flower
    data."
71
72  # make a plot of the iris data
73  import seaborn as sns
74
75  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))
76
77  ```
```

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!

- **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.)

Code is initially collapsed; can be expanded by user

Initial data prep and plotting in Python

Statistical analysis in R

Do Hershey's Stock Prices Increase Around Valentine's Day?

</> Code

AUTHOR
Aaron M. Geller

PUBLISHED
December 12, 2023

Import libraries in R and Python

▶ Code

▶ Code

Use Python to load, prepare and visualize the data (in Figure 1 and Figure 2)
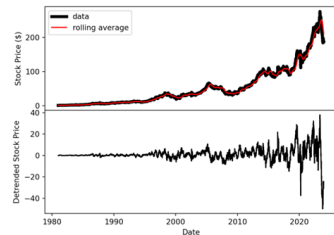
▶ Code

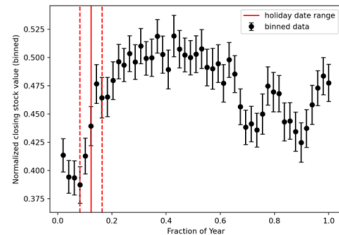Figure 1: Hershey's stock price over time

▶ Code

Figure 2: Mean binned Hershey's stock price over one year periods

Use R to perform a few statistical checks to see if the stock price increases after Valentine's Day

▶ Code

Statistical results:

- p-value for a 2-sample T-test on values before vs. after Valentine's Day : 0.0083071.
- p-value for a 2-sample K-S test on values before vs. after Valentine's Day : 0.002605.

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](#)