Tutorial 3: Running Analysis

Executing code and rendering reports

2026-01-28

Introduction

In this final tutorial, we will run the analysis code provided in the lecture files. We will use **Quarto**, a scientific publishing system that allows us to mix text, code, and results in one reproducible document.

Prerequisite: Before starting this tutorial, ensure you have completed Tutorial 2: Getting Started with Positron and have the `Math495Spring26` folder open as a project in Positron.

Understanding the Document

Open `lecture1_data_management.qmd`. You will see three types of content:

1 **YAML Header**

The block at the very top between `---` lines (called "front matter"). It contains metadata about the document:

- `title`: The document title
- `author`: The author's name
- `date`: The date (often set to `last-modified`)
- `format`: Output format settings (HTML, PDF, etc.)
- `execute`: Options for code execution (echo, warning, etc.)

Example:

```yaml
---
title: "Lecture 1: Data Management"
author: "Your Name"
date: last-modified
format: html
---
```

Note: Don't modify the YAML header unless instructed.

2 Markdown Text

The narrative content written in Markdown format. Markdown is a lightweight markup language that converts to formatted text.

Common Markdown syntax:

- `# Heading 1`, `## Heading 2` - Headings
- `**bold**` or `__bold__` - Bold text
- `*italic*` or `_italic_` - Italic text
- `- Item` or `* Item` - Bulleted lists
- `1. Item` - Numbered lists
- `[text](url)` - Links
- `![alt](image.png)` - Images
- `> quote` - Blockquotes
- `` `code` `` - Inline code
- `---` or `***` - Horizontal rules

Tip: The text you're reading right now is written in Markdown!

3 Code Chunks

Blocks of executable code wrapped in triple backticks with language specification.

Syntax:

````
```{r eval=FALSE}
# R code here
```
````

````
```{python eval=FALSE}
# Python code here
```
````

Code chunk options (in curly braces after language):

- `echo=FALSE` - Hide code, show only output

- `eval=FALSE` - Don't execute code
- `warning=FALSE` - Suppress warnings
- `message=FALSE` - Suppress messages
- `fig.width=7, fig.height=5` - Set plot dimensions

Example with options:

```
```{r echo=FALSE, warning=FALSE, eval=FALSE}
library(tidyverse)
data <- read_csv("data.csv")
```
```

Tip: Code chunks are executed in order when you render the document.

Running Code

You can run code interactively to test it before creating the final report.

1 **Run a Single Chunk**

Look for the green **Play icon** ( ) at the top right of any code chunk. Click it to run that specific block of code.

Keyboard shortcuts:

- `Ctrl+Enter` (Windows/Linux) or `Cmd+Enter` (Mac) - Run current chunk
- `Alt+Enter` (Windows/Linux) or `Option+Enter` (Mac) - Run chunk and move to next
- `Ctrl+Shift+Enter` (Windows/Linux) or `Cmd+Shift+Enter` (Mac) - Run all chunks above

2 **View Output**

The result will appear directly below the code chunk in the editor.

- **Text output**: Tables, summary statistics, or printed results
- **Plots**: Visualizations appear inline
- **Errors/Warnings**: Displayed in red with details

Tip: You can clear output by clicking the broom icon ( ) at the top right of the output area.

3 **Run Multiple Chunks**

To run all code chunks in order:

1. Click the **Run** menu at the top of Positron
2. Select **Run All** (or **Run All Chunks**)
3. Alternatively, use the keyboard shortcut `Ctrl+Alt+R` (Windows/Linux) or `Cmd+Option+R` (Mac)

Important: Code chunks often depend on previous ones (e.g., loading data before analyzing it). Always run chunks in order, starting with the setup chunk at the top that loads libraries and data.

Rendering the Report

"Rendering" means converting the `.qmd` file into a polished HTML, PDF, or Word document.

1 **Find the Render Button**

Look for the **Render** button in the toolbar above the editor. It typically:

- Shows a preview icon ( ) or says "Render"
- May have a dropdown to choose output format (HTML, PDF, Word)
- Is located near the file name tabs

Alternative methods:

- **Keyboard shortcut**: `Ctrl+Shift+K` (Windows/Linux) or `Cmd+Shift+K` (Mac)
- **Command Palette**: Press `Ctrl+Shift+P` (Windows/Linux) or `Cmd+Shift+P` (Mac), type "Quarto: Render", and select it
- **Right-click**: Right-click in the editor and select "Render"

2 **Monitor the Rendering Process**

When you click Render:

1. Quarto processes the document from top to bottom
2. All code chunks are executed in sequence
3. Output is generated (tables, plots, etc.)
4. The final document is assembled

Watch the progress:

- **Terminal/Output pane**: Shows detailed progress messages
- **Background Jobs**: May show a progress indicator
- **Status bar**: Bottom-right shows "Rendering…"

Note: The first render may take longer as packages are loaded and caches are built.

3 **View and Use the Output**

Once rendering completes:

- **HTML files**: Open in Positron's Viewer pane or your default browser
- **PDF files**: Open in your system's PDF viewer
- **Word files**: Open in Microsoft Word or compatible software

Output location:

Rendered files are saved in the same directory as the `.qmd` file with the same base name but different extension (e.g., `lecture1_data_management.html`).

Tip: You can refresh the rendered output by re-rendering after making changes.

Troubleshooting Common Issues

"File not found" Error

This usually means Quarto can't find a data file or resource referenced in your code.

Solutions:

1. **Verify project setup**: Ensure you opened the `Math495Spring26` folder as a project in Positron (Tutorial 2).
2. **Check file paths**: Use relative paths from your project root:
   - Correct: `"MATH495_Spring26_ConsultingCourse/data/processed/poultry_litter_data.csv"`
   - Wrong: `"C:/Users/.../poultry_litter_data.csv"` (absolute path)
3. **Verify file exists**: Check the file exists at the expected location.
4. **Case sensitivity**: On Linux/Mac, file names are case-sensitive (`Data.csv` ≠ `data.csv`).

To check: Run this in the R Console:

```
file.exists("MATH495_Spring26_ConsultingCourse/data/processed/poultry_litter_data.csv")
```

It should return `TRUE`.

"Package not found" Error

You need to install required packages before using them.

For R packages:

```r
# Install a package (do once)
install.packages("tidyverse")

# Install multiple packages
install.packages(c("tidyverse", "ggplot2", "dplyr"))

# Load a package (do in each session)
library(tidyverse)
```

For Python packages:

```
# Install a package
pip install pandas seaborn

# Install from requirements (if provided)
pip install -r requirements.txt
```

Note: Package installation typically requires internet access.

"Quarto not found" or Rendering Fails

Solutions:

1. **Verify Quarto installation**: Run `quarto --version` in Terminal.
2. **Restart Positron**: Sometimes a restart helps.
3. **Check PATH**: Ensure Quarto is in your system PATH.
4. **Reinstall Quarto**: Download and install the latest version.

Code Runs Slowly or Gets Stuck

Solutions: 1. **Check for infinite loops**: Ensure your code has proper termination conditions. 2. **Large datasets**: Processing large files takes time. Be patient. 3. **Memory issues**: Close other applications to free memory. 4. **Use caching**: Add `cache=TRUE` to code chunk options to save results.

Getting Help

If you're stuck:

1. **Read error messages carefully**: They often contain specific solutions.
2. **Check the Quarto documentation**: quarto.org/docs/
3. **Search online**: Copy the error message into a search engine.
4. **Ask for help**: Contact your instructor or classmates.

Congratulations! You've completed all three tutorials. You now have: 1. All necessary software installed and verified 2. Familiarity with the Positron IDE interface 3. Understanding of Quarto documents and how to run code 4. Ability to render reports and troubleshoot common issues

You're ready to explore the lecture materials and analyze the poultry litter data!

Next Tutorial: GitHub + Positron Integration →