Getting Started with Quarto

Tyler J. Loewenstein* Ryan T. Moore†

2025-08-16

This is a short introduction to using Quarto (`.qmd` files) on problem sets and labs. Before using this guide, read the *Using the Statistical Software R* guide available at ryantmoore.com/teaching.html.

1 Background

Quarto is a useful format for integrating plain text information and code into one compiled document. It can be used to create reports, articles, slide presentations, and a variety of other formats for scientific communication. Quarto (`.qmd`) files can be rendered to create documents in HTML, PDF, or Microsoft Word. For this class, we recommend a package called `{tinytex}` to create the PDF files that you will submit to Canvas. Instructions for installing `{tinytex}` can be found via at http://www.ryantmoore.org/files/ht/httinytex.pdf.

1.1 `.R` files versus `.qmd` files

It can be easy to confuse the two types of files used in RStudio. Both `.R` and `.qmd` (or `.Rmd`) files can be used to write and run code, but they will each play a different role when performing data analysis.

A `.R` file will ...

- Accept only functions and scripts written in R
- Ignore anything that is commented out using the `#` symbol.
- Keep a record of your code as you work
- Allow you to run **one or more lines** of code at a time

A `.qmd` or `.Rmd` file will ...

- Accept both code and regular text
- Evaluate R code written in gray chunks (more on this below)
- Allow you to format the document around your code
- Compile **the entire document** before outputting your work

*American University

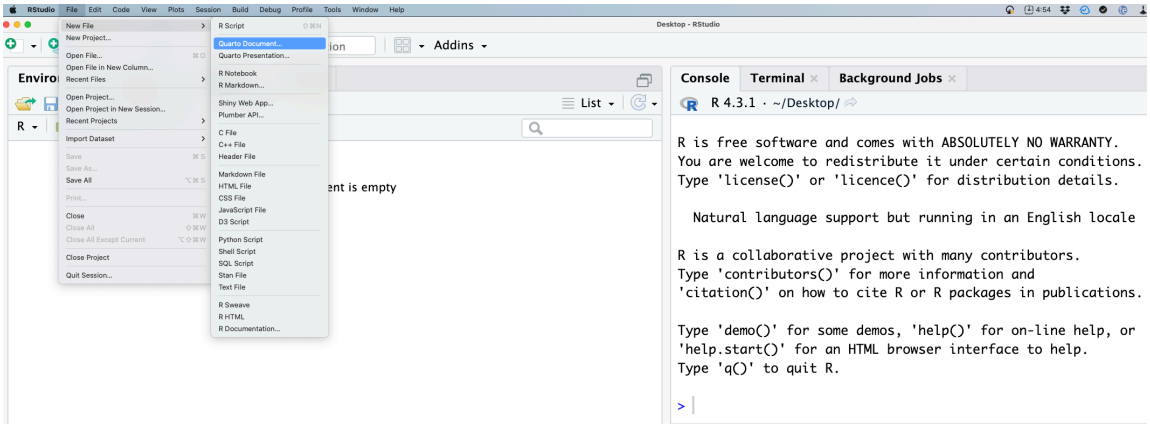
†Department of Government, American University, Kerwin Hall 234, 4400 Massachusetts Avenue NW, Washington DC 20016-8130. tel 202.885.6470; rtm (at) american (dot) edu; http://ryantmoore.org.

2 Using Quarto

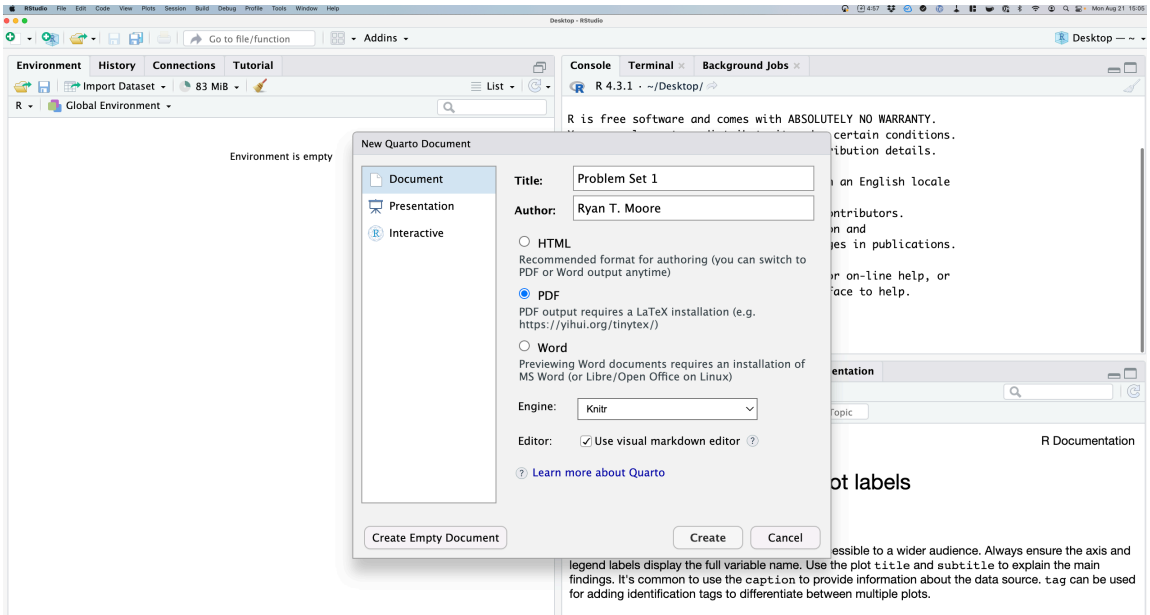
Now that you know the differences between a `.R` file and a `.qmd`/`.Rmd` file, it is important to recognize how they can be integrated into your workflow. This section will illustrate that process and explain how the two file types can work together.

2.1 Creating a `.qmd` file

Creating a `.qmd` is similar to creating a `.R` file. To begin, open RStudio and select File > New File > Quarto Document. (If you are creating a file for class, start by opening your `.Rproj` file, *then* creating a new Quarto document.)



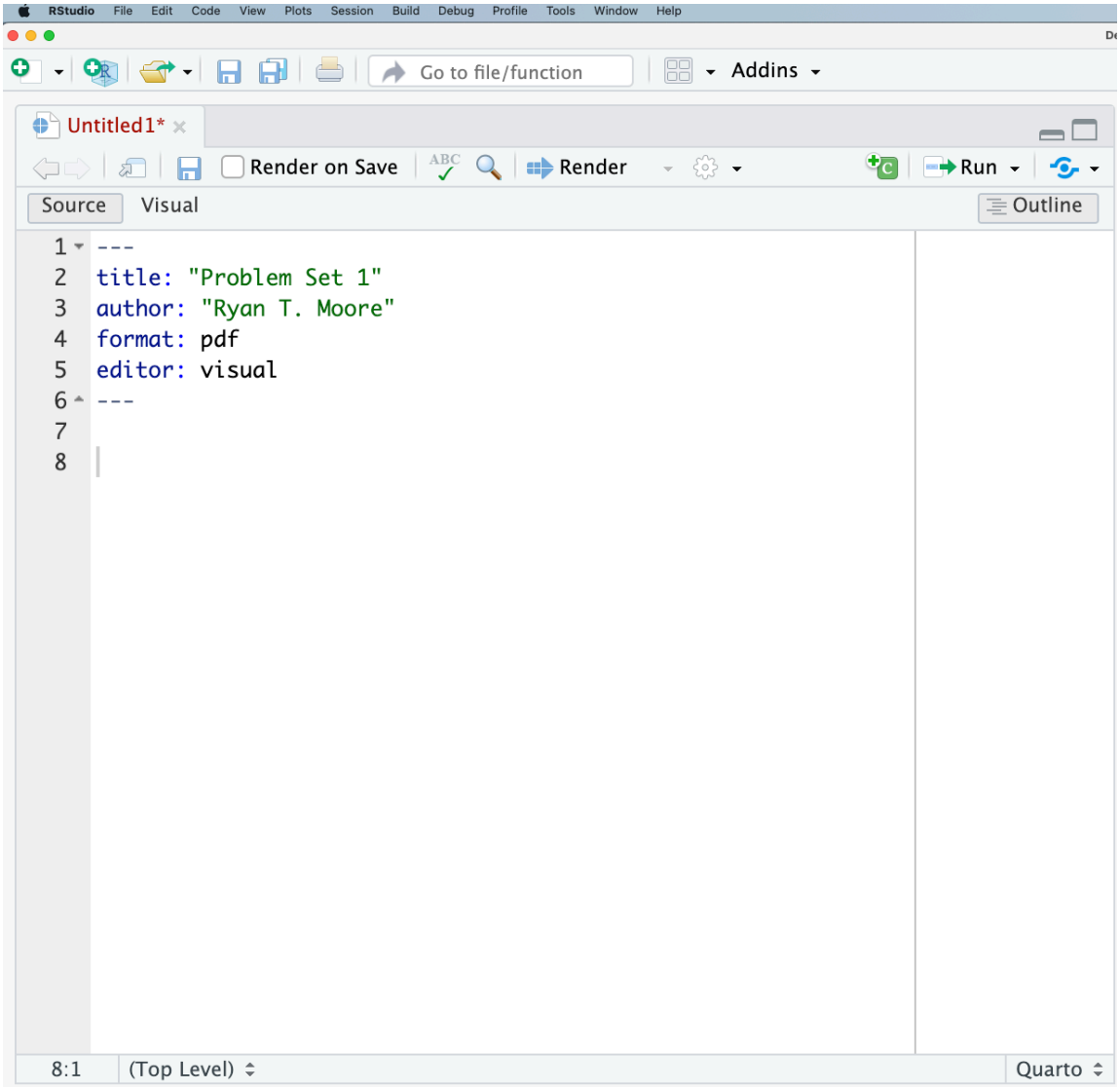
RStudio will ask you to give the document a title and select the default output. If you have `{tinytex}` installed, you should select PDF. Otherwise, choose HTML. Once you have filled in these fields, select Create.



2.2 Editing the `.qmd`

By default, the new `.qmd` starts in "Visual" editor mode. We recommend changing this to "Source" by clicking the "Source" button in the upper left hand corner. Then, change the header YAML tag from `editor: visual` to `editor: source`.

The default `.qmd` also includes some sample text to show you the kinds of information that can be entered. To remove this, simply delete lines 8 to 27. After you've done that, your file should look like this:



2.3 Saving and naming the `.qmd` file

Save the file using the save icon, File - Save, Cmd-S, or however you prefer to save files.

Give the file a good filename like `ps1_Moore.qmd`. No spaces.

Save the file to an appropriate directory/folder.

2.4 Customizing the YAML header

You can edit the header that is between the two sets of `---` (lines 1-6) above, but be careful. Spaces matter in this header.

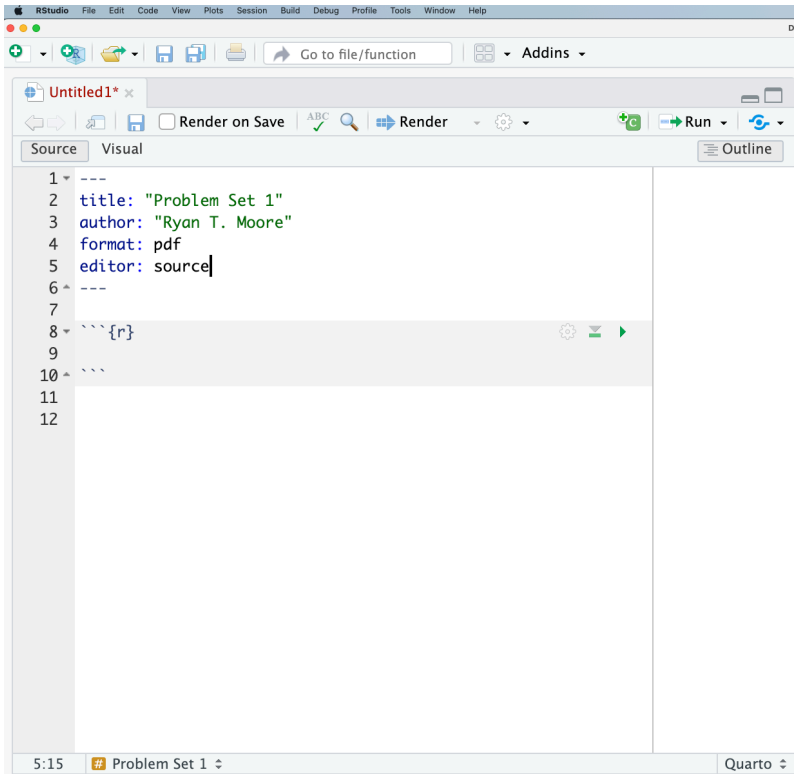
You might change the document class to `article` by adding `documentclass: article` to the header in a new line. This will produce a more standard U.S. article. You might also number your sections. To do these, add these lines to your YAML:

```
documentclass: article
format:
  pdf:
    number-sections: true
```

Note exactly 2 spaces in the successive lines above.

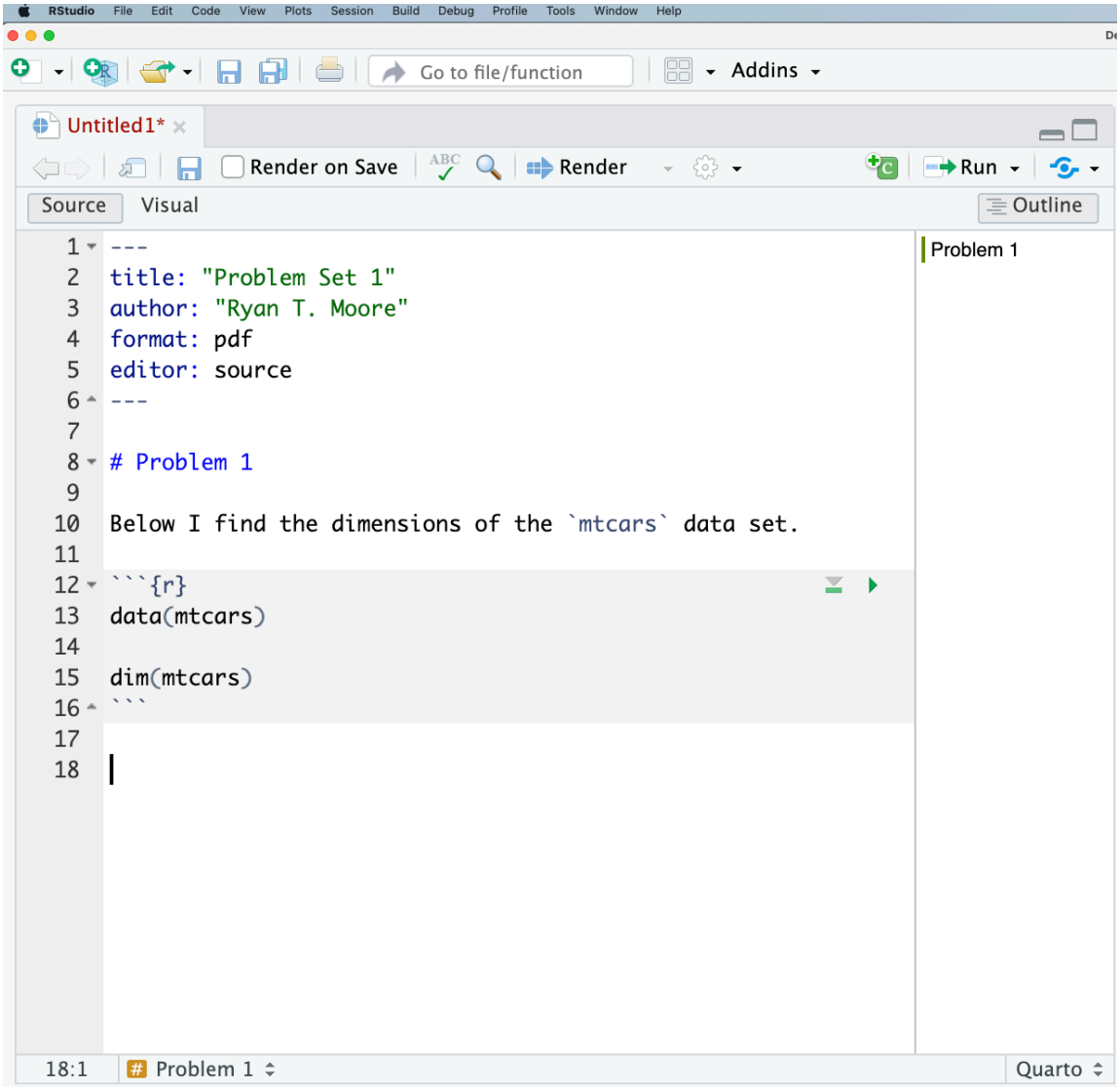
2.5 Adding content to an `.qmd` file

Your new `.qmd` file will accept both code and regular text. Anything you want R to evaluate (i.e. your code) should be placed inside a code *chunk*. To make a new chunk, select a blank line and type out three backticks followed by curly brackets that contain the letter `r`. Skip a few lines and type three more backticks to close the chunk. You can also add a chunk by selecting Code > Insert Chunk. Any lines inside a code chunk will appear gray.



Text can be added to your document normally by typing anywhere *outside* of a code chunk. There are numerous ways to customize your `.qmd` using LaTeX, a document preparation system commonly used in social science. To learn more, you can read the *Getting Started with LaTeX* guide via the URL at the top of this guide. For now, you can add a section header by using the # symbol.

Here is an example of how to format the answer to a problem set using these techniques:



There is another way to add code into an `.qmd`: inline code. This allows you to write sentences as you would normally, but also integrate the results of your code. This is done by putting your code inside of `` `r ` `` instead of a regular chunk. Here's an example:

Instead of typing ...

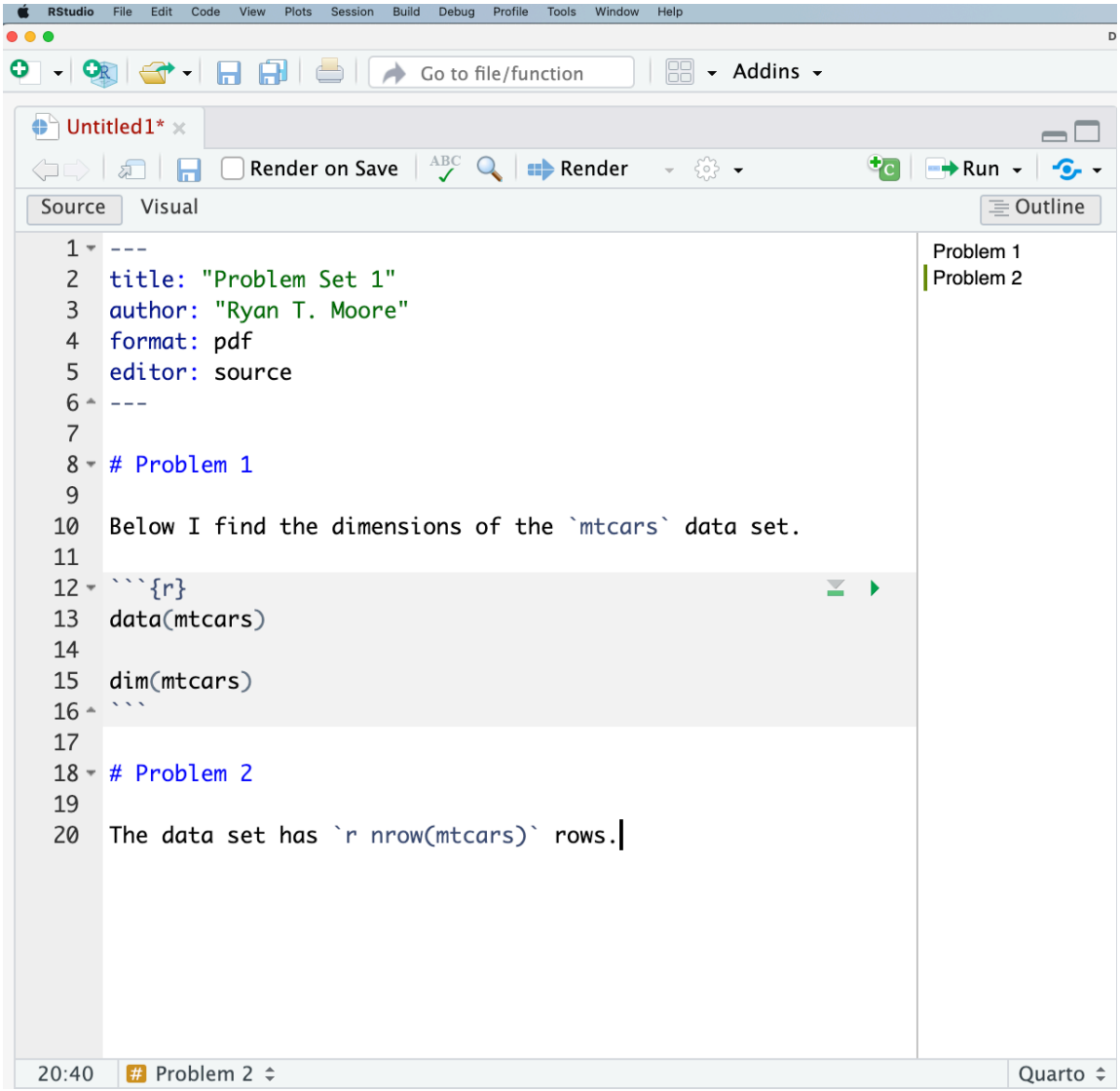
- `The sum of 12 and 45 is 57.`

You can type ...

- `The sum of 12 and 45 is` `r 12 + 45`.

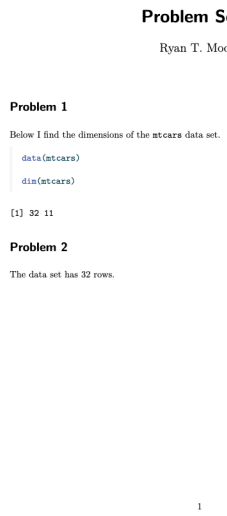
Both of which will output …

- The sum of 12 and 45 is 57.



2.6 Compiling a `.qmd` file ("rendering"/"knitting")

The last step in preparing your document is to compile the `.qmd` into a human-readable PDF; this is also called *rendering* or *knitting*. To do this, simply click on the **Render** button when you are ready. Recall that unlike a `.R` file, a `.qmd` will be read and evaluated all at once. If a single line is incorrect or has a missing character, RStudio will produce an error and the document will not be rendered. See Figure 1 for a finished PDF document after knitting:



Problem Set 1

Ryan T. Moore

Problem 1

Below I find the dimensions of the `mtcars` data set.

```
data(mtcars)
dim(mtcars)
```

```
[1] 32 11
```

Problem 2

The data set has 32 rows.

1

Figure 1: A rendered `.qmd` file (using the default document class)

With these techniques, you should be able to add both code and narrative response to your `.qmd` file and produce a well-formatted PDF. Remember that your narrative responses should be *outside* of code chunks. For problem sets and labs, remember to always show the code used to reach your answer.[1] It's also helpful to use section headers and separate code chunks for each problem.

3 Optimizing Your Workflow

Quarto is great at creating a professional-looking work product that integrates all of your content. It may be difficult, however, to use a `.qmd` when solving problems and analyzing data. To solve this, we sometimes use both a `.R` and `.qmd` file when doing analysis. This achieves several goals:

- **Solve problems faster**: You can run your code line-by-line without having to wait for your whole file to be rendered.
- **Stay organized**: You can add hundreds of lines to your `.R` without worrying about cluttering your `.qmd`.
- **Archive your process**: You have a record of the work you did, the errors you made, and how you came to the solution.

You can think of R Script as your scrap paper and Quarto as the final product. Use comments to break up your `.R` file and then copy and paste your work in the `.qmd` after you've solved the problem. This is an iterative process; you should go back to your `.R` file any time you need to test the code in your `.qmd`.

Here are the steps:

[1] In papers, you should not show your code.

1. Open RStudio (by opening the `.Rproj` file)
2. Create a `.R` file and name it according to your project
3. Create a `.qmd` file with a similar name
4. In your `.R` file, write, test, and edit code for each problem or section of analysis
5. Copy and paste the best code from your `.R` file into your `.qmd`. Use a separate code chunk for each problem part. (Even better: `source()` your `.R` file.)
6. Add text, graphs, and other formatting to your `.qmd`
7. Render your `.qmd`
8. Save your work
9. Repeat steps 4-8 until you've solved every problem!

4 Quarto vs. the Console: Beware the Green Arrow

When you knit a `.qmd` file, R runs the code from the top of the `.qmd` file to the bottom. It does not run other code, and it does **not** look in the Console's workspace. This means your `.qmd` file needs to be entirely self-contained. You need to set the working directory, read the data, create intermediate objects, etc. *within* the `.qmd` file itself.

RStudio provides a small green arrow to the right of your code chunks. When you click the green arrow, R looks in the current workspace for objects, prints the results to the console or plotter, and then reprints them in the confines of your `.qmd`. This behavior is "notebook-like". However, the objects are not really in your `.qmd` file, in the sense that they are not part of the compiled/knit output. When you compile/knit to create an output file, the only code that is run is the code in your `.qmd` file, from top to bottom.

This notebook-like feature is an aspect of RStudio that is not *inherent* in `.qmd` (which could be compiled from a command line outside of RStudio, e.g.). To avoid confusion about the state, we recommend examining your output by knitting your `.qmd` file, rather than by using the green arrow.