Best Practices & advice for getting started:

**Use RProjects and directories**

For each analysis, create an R Project and store it in its own directory. Within that directory, use sub-folders to keep things organized. At minimum, have a “data” folder. Example:

MyProject1  
 -----data  
 -----scripts

**Differences between RScript, RNotebook, RMarkdown:**

RStudio gives you many options for where to write your code. Here are the differences between the three most common. RScript is really good for code that you need to run, but where you don’t need to display output such as tables or charts. For example, you could have a script that imports your data and does a bunch of data cleanup. RNotebook is like a script, but it allows you to display output, such as tables and charts, within the page. You don’t have much ability to make the output look really “pretty” but it’s good enough. RMarkdown allows you to make a pretty document, such as an HTML page, from your analysis. You can tell it to display the code you wrote or tell it to hide the code. This is especially useful for sharing your results (but not the code) with others on your team, or even with sources who might be commenting on your analysis. Also, you can link a script to a markdown page, so that when you run the code on the markdown page, it will first run the script (i.e. your importing and cleanup).

**Start with something you’re familiar with**

For your first analysis, either find a dataset you are already familiar with and/or re-create an analysis you’ve already done. Perhaps you have a dataset you analyzed using Excel or a SQL program. Try to rebuild it in R. This eliminates the headache of learning a new dataset at the same time as learning a new language.

**Annotate! Annotate! Annotate!**

Use commenting in your code to make notes about what each chunk of code is doing, plus other little things like the source of your data or why your code is doing what it’s doing (i.e. why you are adding a new variable to your data frame). This will be helpful to someone else reviewing your code, but it will also be useful to your future self for a variety of reasons – including when you need to reuse a piece of code that you don’t remember how to write.

**Keep a list of useful code snippets**

You will be gradually learning more and more code as you need it. I’ve found it useful to keep a text file on my computer with useful code snippets that I might need again. I keep adding to it each time I learn something new. For example, the syntax for joining fields, but only pulling certain fields from one of the files. Or the syntax for exporting data as JSON. You can find these things on the Internet, but they aren’t always well explained and it’s nice to have them all in one place (annotated with my own notes)

**Finding help on the Internet**

It’s important to know that there are numerous ways to do the same thing. Also important to know that anyone who has been using R for a long time is probably using Basic R for most of their work. Tidyverse, which is newer, offers a lot of improvements over Basic R functions and is becoming more widely used. You’ll find that this is what data journalists are using. So, when you go searching for help on the web, try to look for solutions that involve Tidyverse where possible. Limit your searches to the recent past. Include “dplyr” or “tidyverse” in your search string.