RStudio Basics

Gene Ho

rev. April 11, 2021

Contents

Introduction

Installation

R and RStudio Desktop

For more detailed instructions, go to this link.

1. Download R from CRAN for your specific operating system (e.g. Windows or macOS). The most recent release as of March 2021 is 4.0.4 (Lost Library Book). The version of R that you install does not really matter as long as it is above 4.0.
   Fun Fact: R versions are named after lines from Peanuts/Charlie Brown!
2. Download and install R Studio Desktop from RStudio. RStudio Desktop is the program that you will use to actually code in R.

RStudio Desktop Dependencies

When opening RStudio Desktop for the first time, you may be asked to install additional software. Depending on what programs you have, some, none or even all of this part may apply to you.

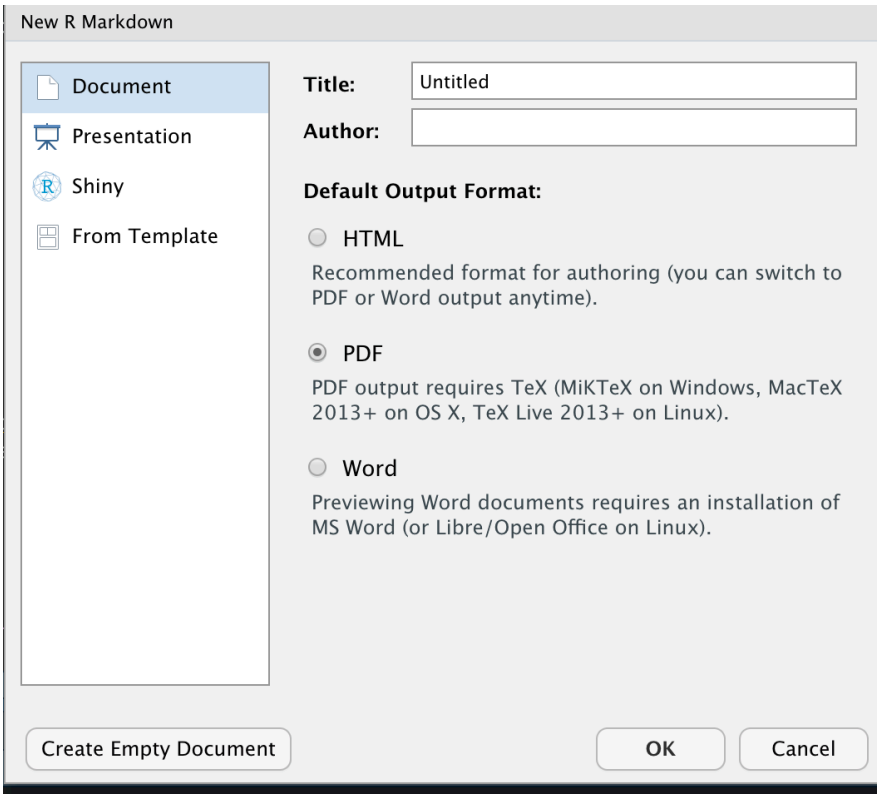
Between each step, allow the installation to complete before proceeding to the next step.

First, open RStudio Desktop.

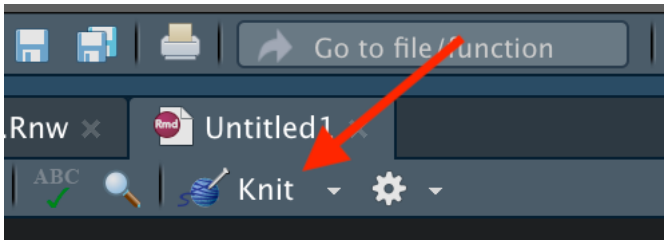
1. If a pop-up appears asking you to install `git`, click *Install*. It is not necessary to install all of Xcode (if asked).
2. Near the top of the Source (top left box), there should be a bar that says that the `knitr` package is not installed. Click *Install*.

RStudio Desktop allows you to "knit" or save your R Markdown files to other file formats. You can knit to HTML, DOCX (Microsoft Word), and PDF. In order to knit to PDF, you will need to make sure you have LaTeX installed. First test if you can already knit to PDF.

1. Create a new R Markdown file by going to File > New File > R Markdown...
2. Select the PDF button and click OK.



3. Click *Knit* in the inner toolbar, under all of the tabs with your open files.



4. If the PDF is created successfully, you're done with the technical installation of R!
   (a) If the creation of the PDF fails, proceed to the next section to install appropriate libraries.

Installing LaTeX to Export to PDF

You will need to have a LaTeX package installed in order to knit to PDF. LaTeX is a document preparation system similar to Microsoft Word and Apple Pages. It is what RStudio uses to generate text in PDF documents.

In RStudio,

1. In the console, enter `tinytex::install_tinytex()`. The `tinytex` package should automatically install and configure your computer to knit to PDF.
2. Wait for the installation to complete. It might take a while.
3. If the installation fails, install an external program with all of these libraries: MikTex for Windows or MacTex for macOS.

- MacTex is a fairly large program to install at roughly 4GB as it contains "all files most users ever need." Thus, you won't have to learn how to search for and install LaTeX packages. However, if space is a limitation, BasicTeX is only 80MB and only contains the bare minimum for LaTeX to run.
- If your installation still fails **and** you're using a mac, go to Appendix , to troubleshoot this error.

Libraries and Packages

Packages (e.g. `ggplot2`, `dplyr`, `stats`) are collections of functions that you can run in R. Functions are the codes that you can call and run in R that do "stuff." People can contribute their own code to further improve the functionality of R. As for the distinction between packages and libraries:

> @ijlyttle a package is a like a book, a library is like a library; you use `library()` to check a package out of the library #rsats
>
> (Hadley Wickham (@hadleywickham) December 8, 2014)

Using Libraries

At the beginning of every R session, you will need to reload all of the appropriate packages for your script. Every time you close R, it will remove all of your previously loaded packages to free up memory. This is why you typically see multiple calls to the `library()` function at the top of every script.

This is how you load packages into your R session.

```
# notice package name is NOT quoted
library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.3.0
--
## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.1.0     v dplyr   1.0.5
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts()
--
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

As seen above, sometimes you will see mentions of "conflicts" in the functions. This just means that the loaded packages have functions with the same name, so when you don't name a package when you're

calling a function, it will default to the function that masked the other. Generally, you don't have to worry about this warning. However, if you find that a function is getting called from the wrong package, it might be because it's masked by another one. This is a case where order of package loading matters.

In the example above, `tidyverse` masks `stats::filter()` and `stats::lag()`. In order to call a function from a specific package, use the syntax `package::function()` (e.g. `dplyr::filter()`).

Installing Packages

If you run into one of these errors, it likely means you don't have the package installed.

```
library(functions)

## Error in library(functions): there is no package called 'functions'

# and you're sure you spelled the function correctly
# double check by running ??function_here
function_here()

## Error in function_here(): could not find function "function_here"
```

To install the package, run `install.packages('quoted_name')` in your console. You should only have to do this one time per package.

```
# quotations are required!
# it will give you an error if you don't!
install.packages('tidyverse')
```

Managing Your Files

Downloading

When you download files from bCourses, they will usually be downloaded into your Downloads folder. Usually they will have a path of:

- macOS: /Users/SomeUser/Downloads/
- Windows: C:\Users\SomeUser\Downloads\

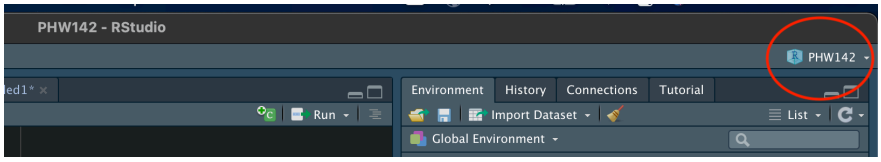
Notice the difference in direction of the slashes in each of the slashes in the folder paths. Regardless of what operating system you use, you must use forward slashes–the one near the right shift key and not the one under the backspace key. This means in Windows, if you copy the file path, you will need to replace the backslashes with forward slashes. I usually do this Ctrl + F, finding "\", replacing it with "/", and clicking the right "All" button to replace all.

R Project Files

R Project files (which end in '.RProj) are a convenient way to manage and compartmentalize your files. When you create an R Project, you can avoid many headaches around file management and paths, your environment which contains all of your variables and functions, etc.

I would suggest creating an R Project file for each class or project that you're working on. This will ensure you don't mix up your work between various projects.

1. File > New Project
2. Click *'New Directory'*, to create a folder to house this R Project file.
    - If you've already created a folder to put your R Project file in, click *'Existing Directory'*
3. Click *'New Project'*
4. Name your project (e.g. "PHW142") and choose where you want to create your R Project and associated files.



File Paths

Assume my folder structure looks like this:

```
./ph142 (main folder)
├── phw142.rproj
├── sf-housing-prices.csv
├── code
│   ├── hw01.rmd
│   └── us_births_2019-2020.csv
└── data
    └── toxic-ink.xlsx
```

Being intentional about organizing your files for this course can make your coding experience smoother and less headache inducing. We recommend creating a folder on your computer specifically for this class. Within the course folder, you are free to organize your folders/files how you wish. I would suggest either the schema above with code and data folders, or one with a folder for each assignment.

```
./ph142 (main folder)
├── phw142.rproj
├── hw01
│   ├── hw01.rmd
│   └── us_births_2019-2020.csv
└── hw02
    └── hw02.r
```

```
└── toxic-ink.xlsx
```

Tip: Using leading 0's with sequentially numbered folders/files will ensure they remain in order when you get to >9 files. If you don't use leading 0's your files may get sorted: file1.r, file10.r, file11.r, file2.r, as periods have higher sort priority compared to letters and numbers.

Relative File Paths

Navigate to the course folder you created in the previous section and open the R Project file. As you continue to use R Projects, you may notice the following things:

Once you open R, you can tell what project you're in by looking in the top right corner. If you are in a project, it will say "Project: [project name]." If you are not in a project, then it will say "Project: (None)"

You can specify your files relative to the `Rproj` file. That is why it is a good idea to keep related files, like data sets, scripts in the same folder or subfolders of the R Project folder so you won't have to type out the file path every time. You can always specify a full or absolute file path instead of a relative one.

To find what your working directory is:

```
getwd()

## [1] "/Users/geneho/OneDrive/Berkeley/Course Materials/PHW142/rstudio-basics"
```

So in my case, I can specify all of my file paths relative to /Users/geneho/OneDrive/Berkeley/Course Materials/PHW142/rstudio-basics, which is my working directory.

Assume I have the first folder structure above and I want to import sf-housing-prices.csv. The relative version only requires me to include the parts of the path after my working directory.

```
# long version
read.csv('/Users/geneho/OneDrive/Berkeley/Course Materials/PHW142/rstudio-basics/sf-hous

##                STDADDRESS                    PRJ_NAME BLOCK LOT UNITS
## 1         1190 MISSION ST              Trinity Place  3702  52   418
## 2         333 HARRISON ST               Rincon Green  3766   9   326
## 3          1407 MARKET ST                       Nema  3507  41   317
## 4         1880 MISSION ST                       Vara  3547  29   202
...

# short version
read.csv('sf-housing-prices.csv')

##                STDADDRESS                    PRJ_NAME BLOCK LOT UNITS
```

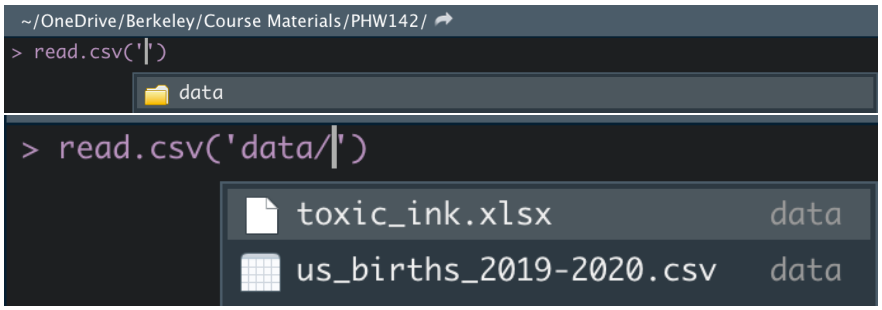
```
## 1          1190 MISSION ST                 Trinity Place  3702  52   418
## 2          333 HARRISON ST                  Rincon Green  3766   9   326
## 3           1407 MARKET ST                         Nema  3507  41   317
## 4          1880 MISSION ST                         Vara  3547  29   202
...
```

To find a file in a subfolder, type the subfolder's name proceded with a forward slash before the file name.

```
read.csv('data/us_births_2019-2020.csv')
```

```
##                    State Year     Month          Period
## 1          UNITED STATES 2019   January         Monthly
## 2          UNITED STATES 2019  February         Monthly
## 3          UNITED STATES 2019     March         Monthly
## 4          UNITED STATES 2019     April         Monthly
...
```

You can also pull up a menu of folders/files in your working directory by pressing tab when you're between two quotes. There you can select the file you want and R Studio will automatically place the relative file path. If your file is in a sub-subfolder, when you're in the tabbed menu, select the folder your file is in and click enter. The folder name should populate into the code. Press tab again to pull up another menu with files in that subfolder. Repeat as necessary.



Common File Types

To read in different file types, you can either type it into the console/source, or use the "Import Dataset" button in the environment pane. Using the import button will generate the code for you, but is likely slower than just writing out the code yourself.

Comma Separated Value (CSV) Files

This is typically the most common data format I've encountered. This is simply a text file that separates data in columns with a comma and row with a new line. Similarly there are tab-separated value files (TSV), which separate each column with a tab.

To read in csv files,

```
# base r
read.csv('sf-housing-prices.csv')

# if you have readr/tidyverse installed, this can actually be faster!
read_csv('sf-housing-prices.csv')
```

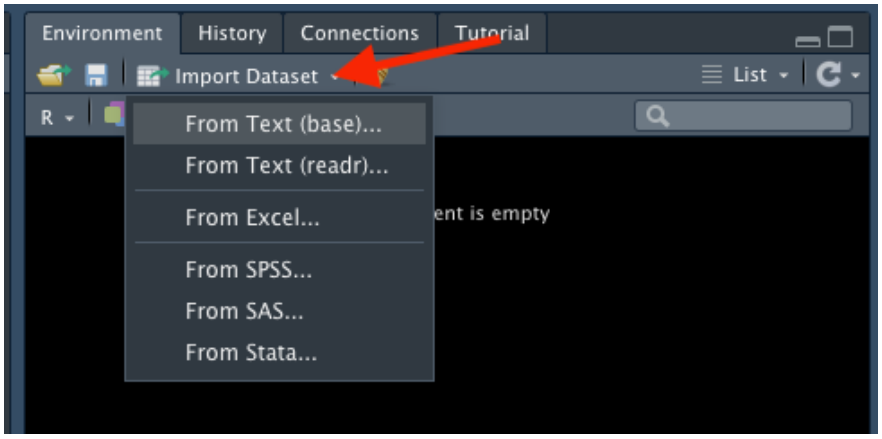
Excel Files

Excel files usually end with XLS or XLSX and can't be read in by base R or tidyverse. You have to use the `readxl` package.

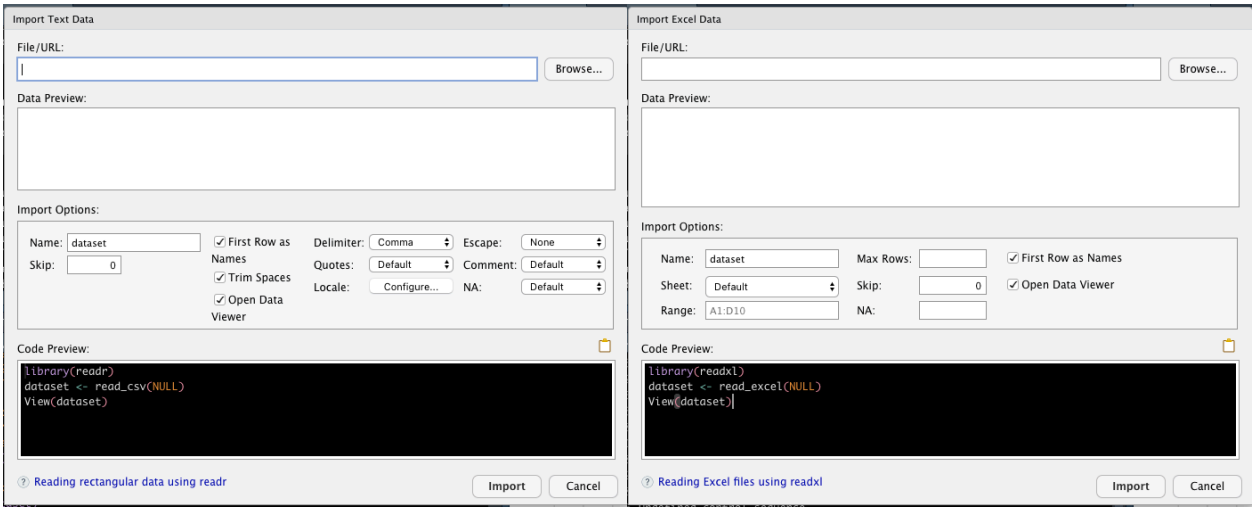
```
library(readxl)

read_xlsx('data/toxic_ink.xlsx')
```

The Import Utility

Near the top of the Environment pane, there is a "Import Dataset" button, which you can use to import various file types. What files appear in this menu depends on the packages you have installed (see above; e.g. `readr` and `readxl`). The import utility provides an interface for you to generate code and import data into your R session.



If you decide to use the import utility, over writing the import code yourself, remember to copy the import code at the bottom of the window that appears and paste it into your script or Rmd. If you forget to do this, you will have to use the import utility every time your environment is cleared or R session is restarted. By putting this code in your script, it will be easier for you to run (and not have to worry about debugging and finding the file) or, if you decide to "Run All," it will read that code and import it for you.



(a) (b)

Figure 1: (a) Import Utility for a CSV using `readr` (b) Import Utility for an Excel file using `readxl`

In the two figures above, the first box corresponds to the file location and has a button for you to browse for it. The import options varies depends on what file type you're trying to read in. If it's Excel, you need to specify which sheet and cells you want to import. Whereas for CSV, you specify delimiters and how quotes are handled (among other things).

The bottom box looks kind of like a code chunk and contains the import code. The code will automatically update based on the location and settings you set above. Before clicking the Import button on the bottom left, make sure to copy the code and paste it into the script.

Appendices

macOS: Installation Error

If you run into an error, reach out to Gene!