

RStudio

Working with R – RStudio

RStudio is an Integrated Development Environment (IDE) for R

- Helps you write code - makes suggestions
- Helps you view the output of your code
- Helps you find errors
- Is NOT a dropdown statistical tool (such as Stata)
 - See [Rcmdr](#) or [Radiant](#)



[\[source\]](#)

RStudio used to be the name of a company that is now called [Posit](#).

RStudio

Easier working with R

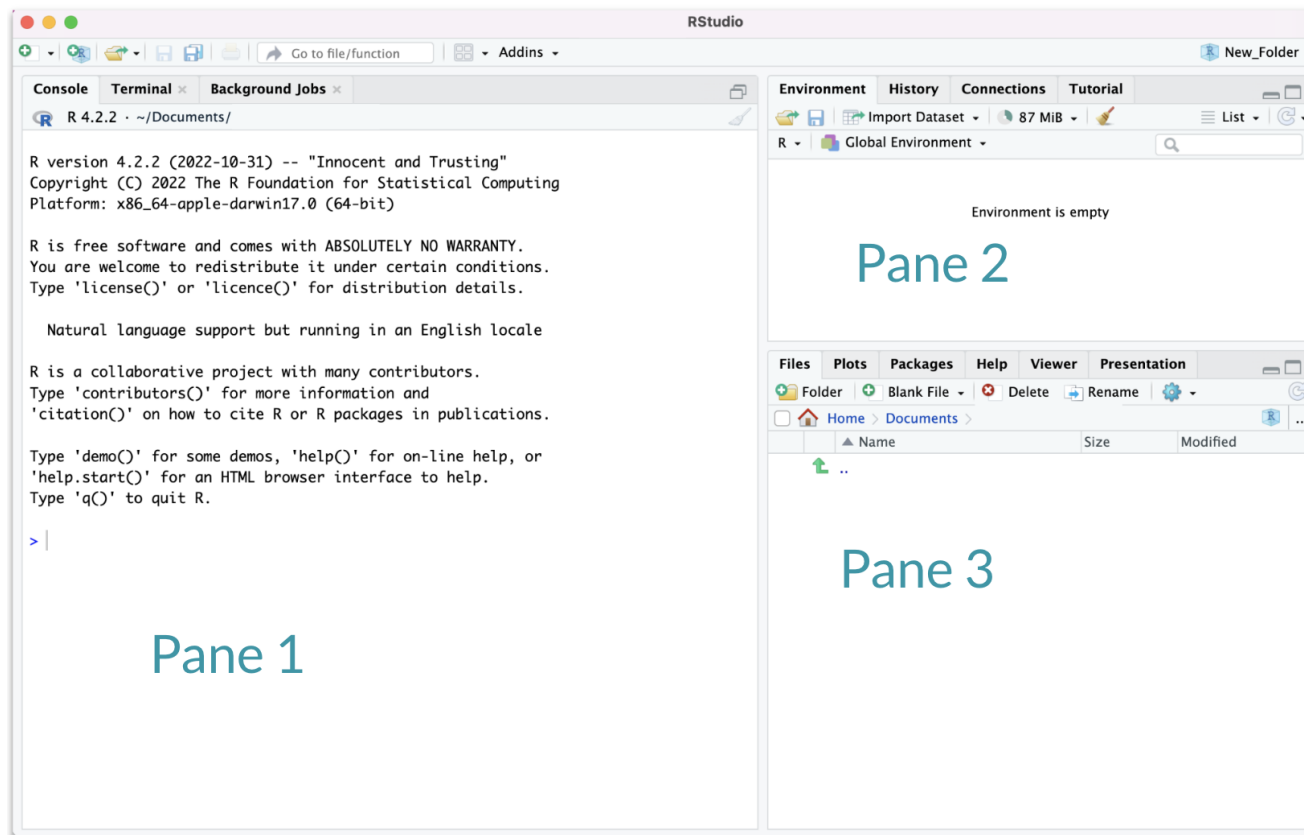
- Syntax highlighting, code completion, and smart indentation
- Easily manage multiple working directories and projects

More information

- Workspace browser and data viewer
- Plot history, zooming, and flexible image and file export
- Integrated R help and documentation

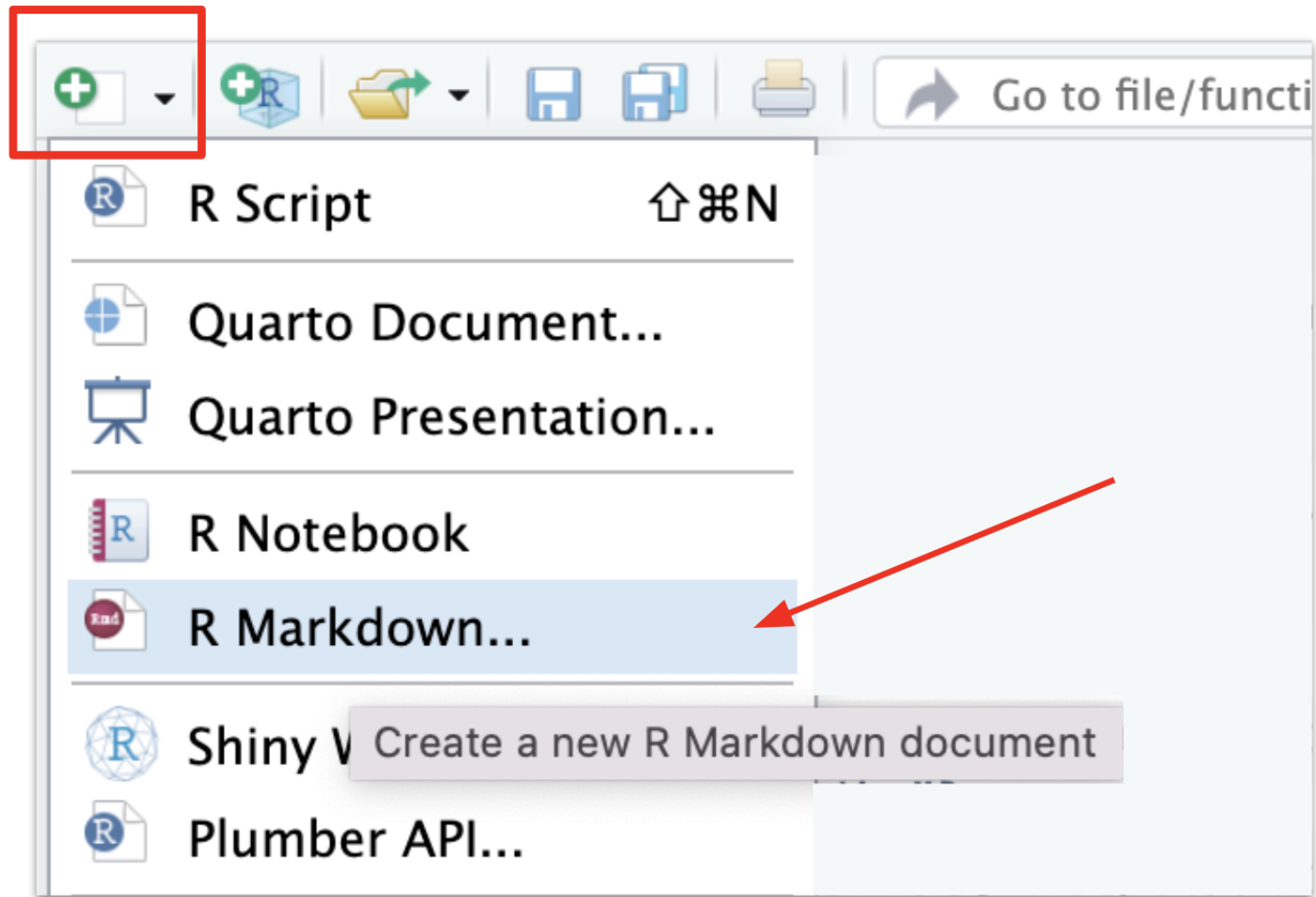
RStudio

First it is important to be familiar with the layout. When you first open RStudio, you will see 3 panes.



Hidden Pane

To save a copy of your code. You must open a file first - this will open a 4th pane. These files include Scripts or what are called R Markdown files.

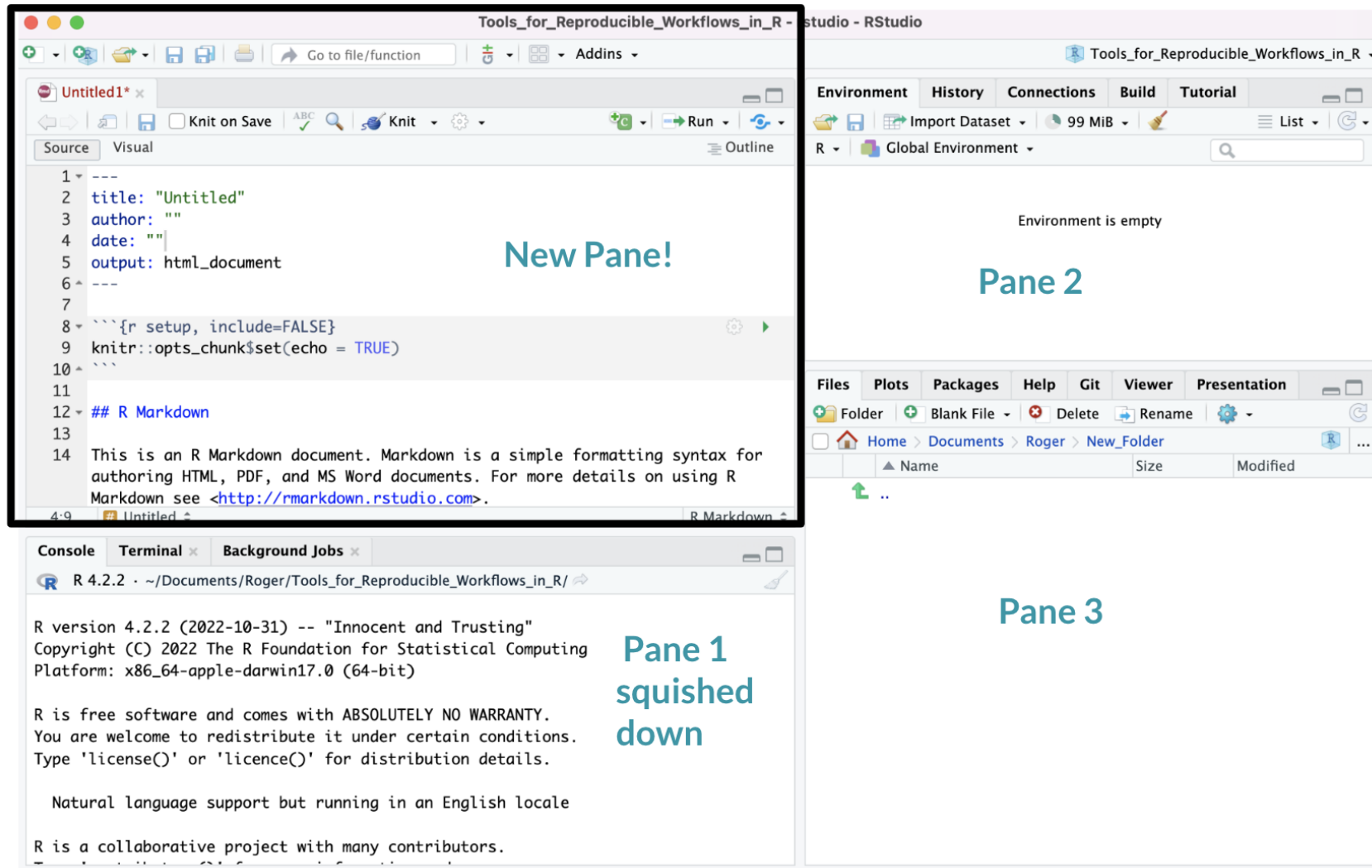


Hidden Pane

You will see a popup that you can just say “OK” to for now.

Hidden Pane

Nice! now we have a place to save code! This is where we will mostly be working.



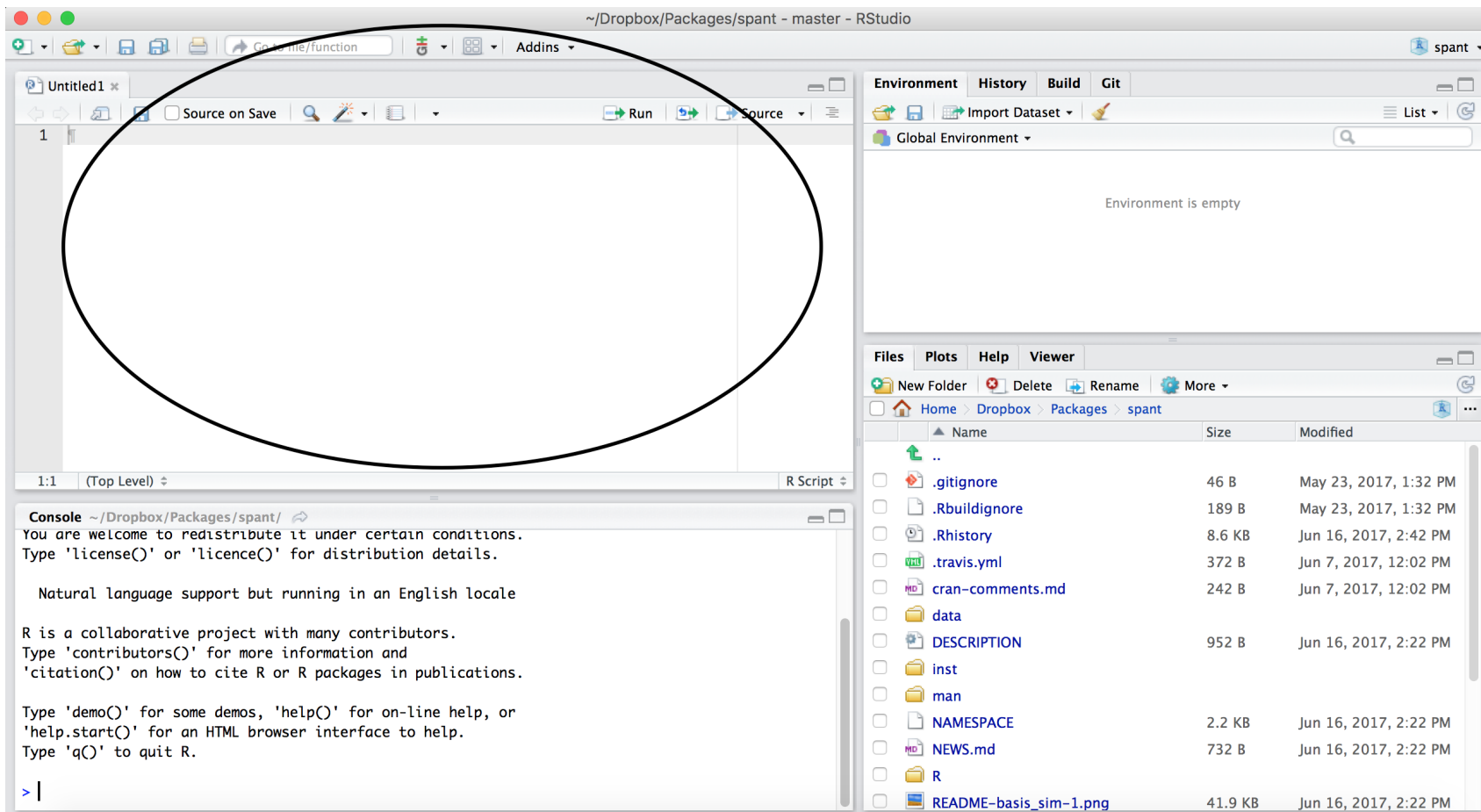
Working with R in R Studio - 2 major panes:

1. The **Source/Editor**: “Analysis” Script + Interactive Exploration
 - Static copy of what you did (reproducibility)
 - Top by default
2. The **R Console**: “interprets” whatever you type
 - Calculator
 - Try things out interactively, then add to your editor
 - Bottom by default

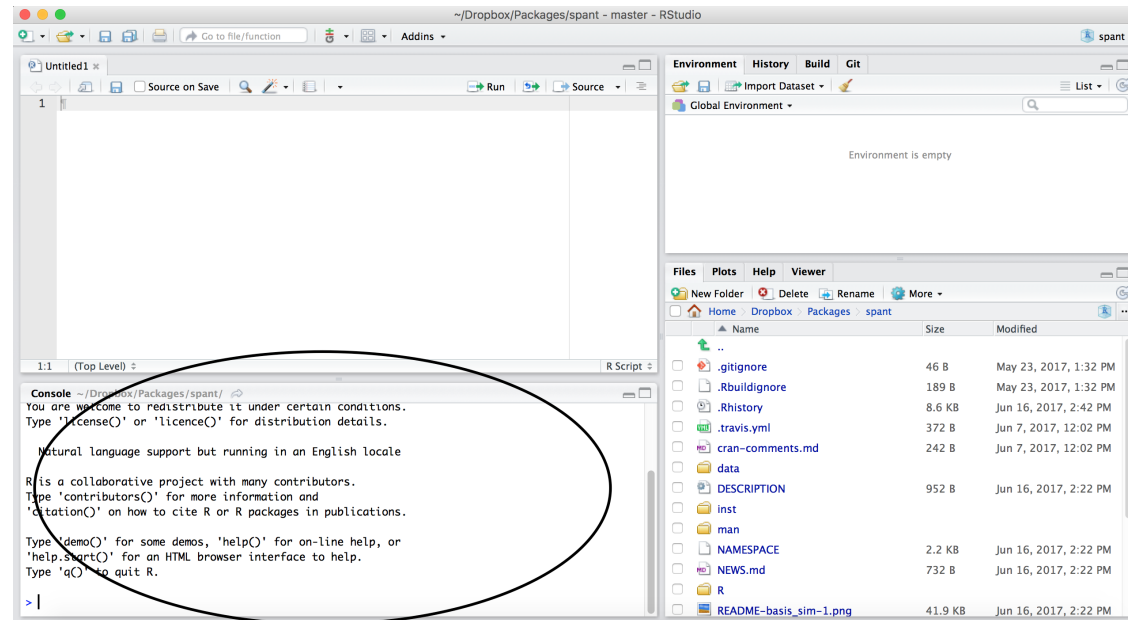
Source / Editor

- Where files open to
- Have R code and comments in them
- Can highlight and press (CMD+Enter (Mac) or Ctrl+Enter (Windows)) to run the code

In a .R file (we call a script), code is saved on your disk



R Console



- Where code is executed (where things happen)
- You can type here for things interactively to test code
- Code is **not saved** on your disk

RStudio

Super useful “cheat sheet”: [LINK](#)

Write Code

Navigate tabs
Open in new window
Save
Find and replace
Compile as notebook
Run selected code

The screenshot shows the RStudio IDE interface with various panels and annotations. The main editor window displays R code with syntax highlighting and line numbers. The Environment panel on the right shows the current workspace with objects like 'iris' and 'foo'. The Files panel at the bottom shows the file browser. Annotations are placed over the interface to describe key features.

Write Code

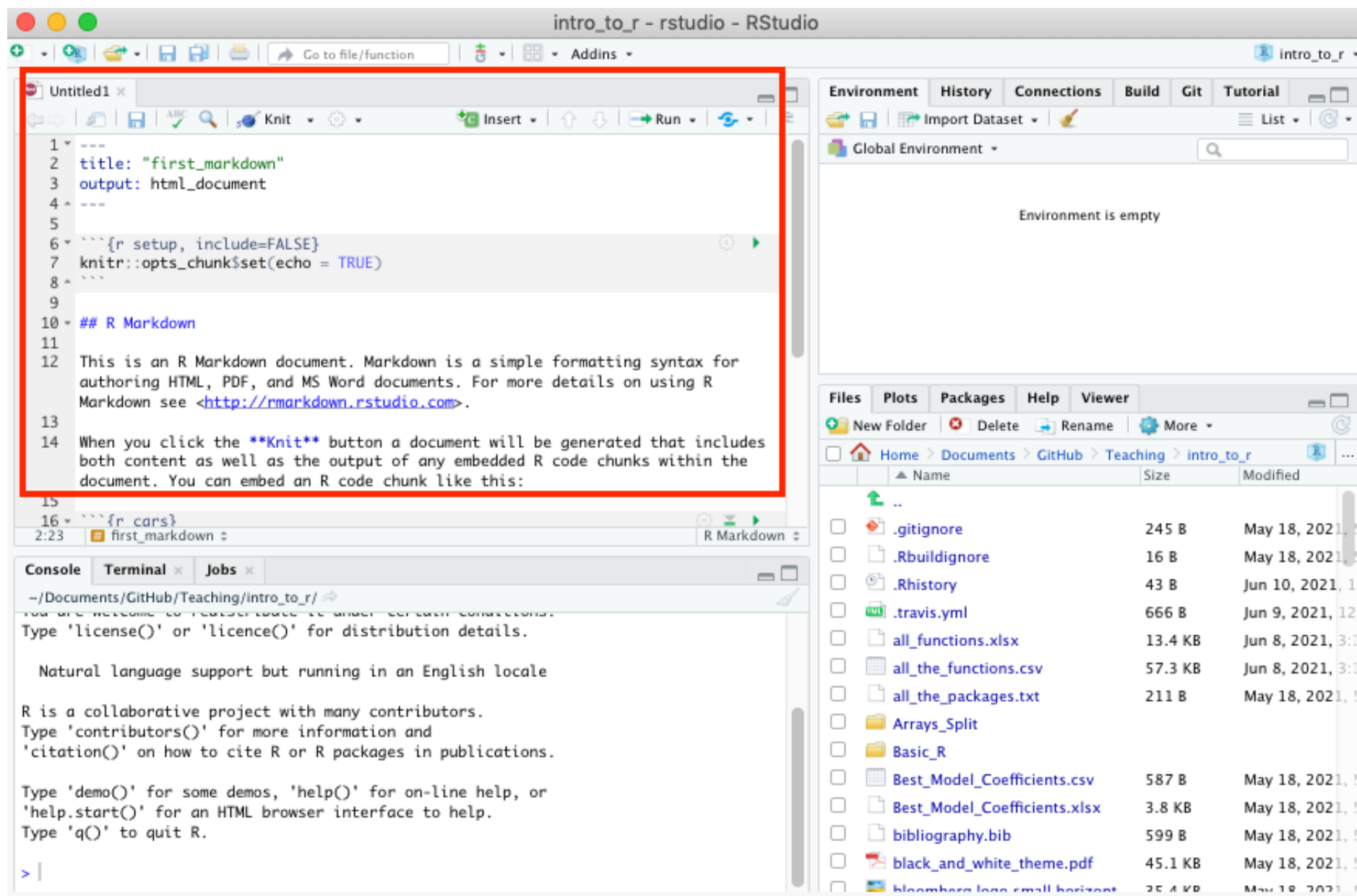
- Navigate tabs
- Open in new window
- Save
- Find and replace
- Compile as notebook
- Run selected code
- Cursors of shared users
- Re-run previous code
- Source with or without Echo
- Show file outline
- Multiple cursors/column selection with **Alt + mouse drag**.
- Code diagnostics that appear in the margin. Hover over diagnostic symbols for details.
- Syntax highlighting based on your file's extension
- Tab completion to finish function names, file paths, arguments, and more.
- Multi-language code snippets to quickly use common blocks of code.
- Jump to function in file
- Change file type

R Support

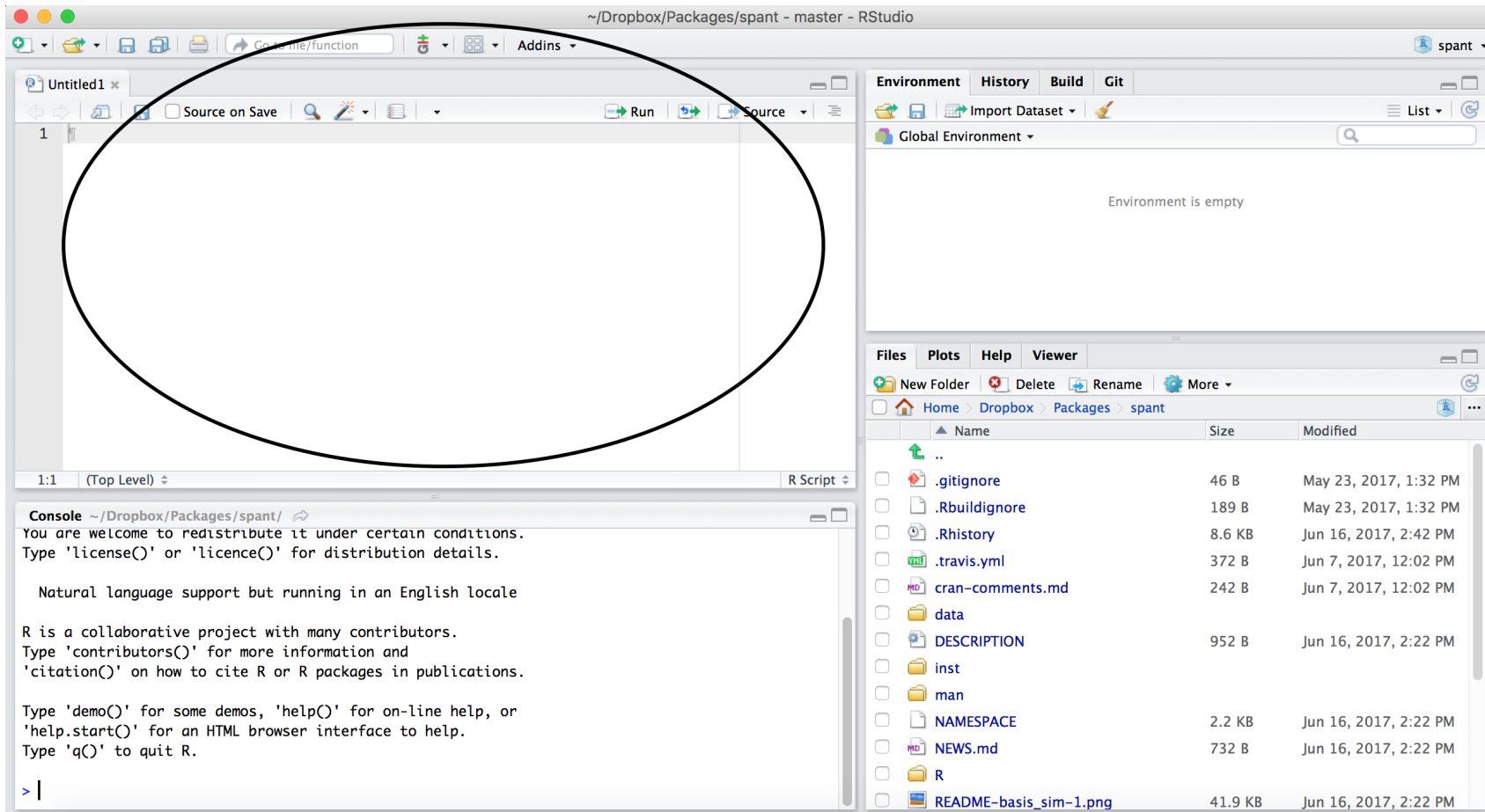
- Import data with wizard
- History of past commands to run/copy
- Display .RPres slideshows **File > New File > R Presentation**
- Load workspace
- Save workspace
- Delete all saved objects
- Search inside environment
- Choose environment to display from list of parent environments
- Display objects as list or grid
- Displays saved objects by type with short description
- View in data viewer
- View function source code
- Create folder
- Upload file
- Delete file
- Rename file
- Change directory
- Path to displayed directory
- A File browser keyed to your working directory. Click on file or directory name to open.
- Working Directory
- Maximize, minimize panes
- Press **↑** to see command history
- Drag pane boundaries

R Markdown files look different from scripts

It will look like this with text in it, unlike a script.



Recall that a script was just empty

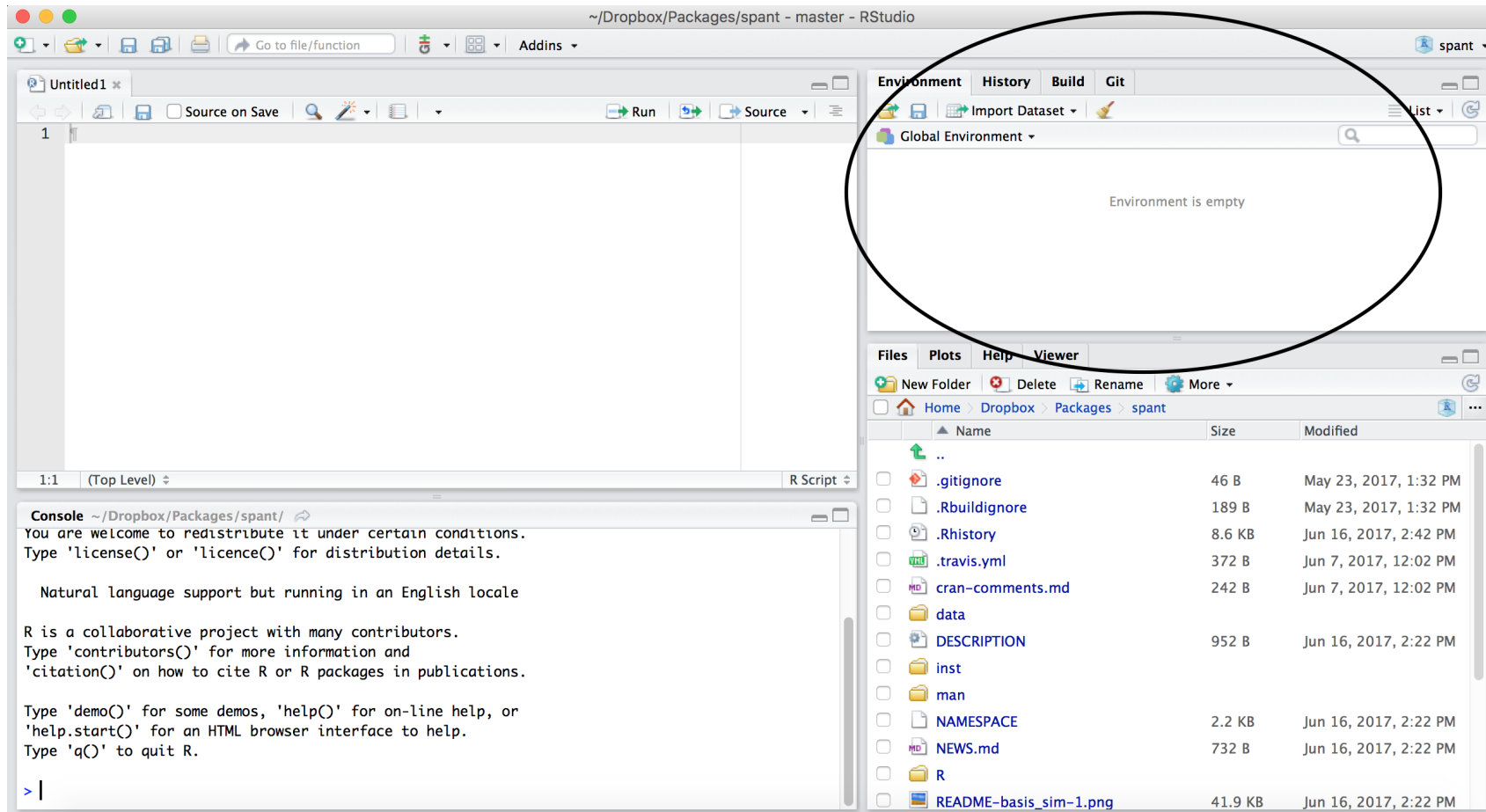


Scripts and R Markdown

Although people will use scripts often, and they are good for more programmatic purposes, we generally don't recommend them for Public Health Researchers.

For data analyses, R Markdown files are generally superior because they allow you to check your code and write more info about your code.

Workspace/Environment



Workspace/Environment

- Tells you what **objects** are in R
- What exists in memory/what is loaded?/what did I read in?

History

- Shows previous commands. Good to look at for debugging, but **don't rely** on it.
Instead use RMarkdown!
- Also type the “up” key in the Console to scroll through previous commands

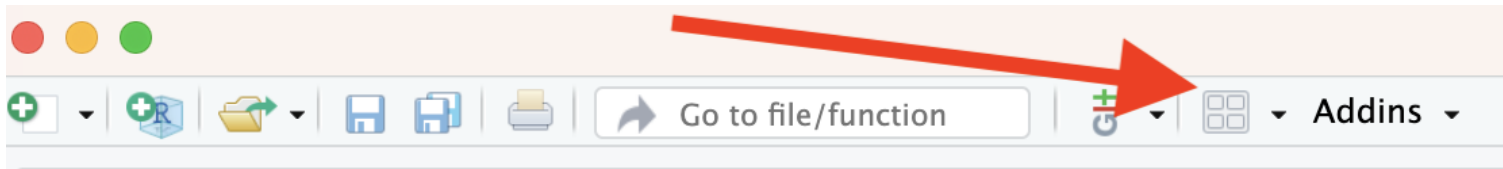
Other Panes

- **Files** - shows the files on your computer of the directory you are working in
- **Viewer** - can view data or R objects
- **Help** - shows help of R commands
- **Plots** - pictures and figures
- **Packages** - list of R packages that are loaded in memory

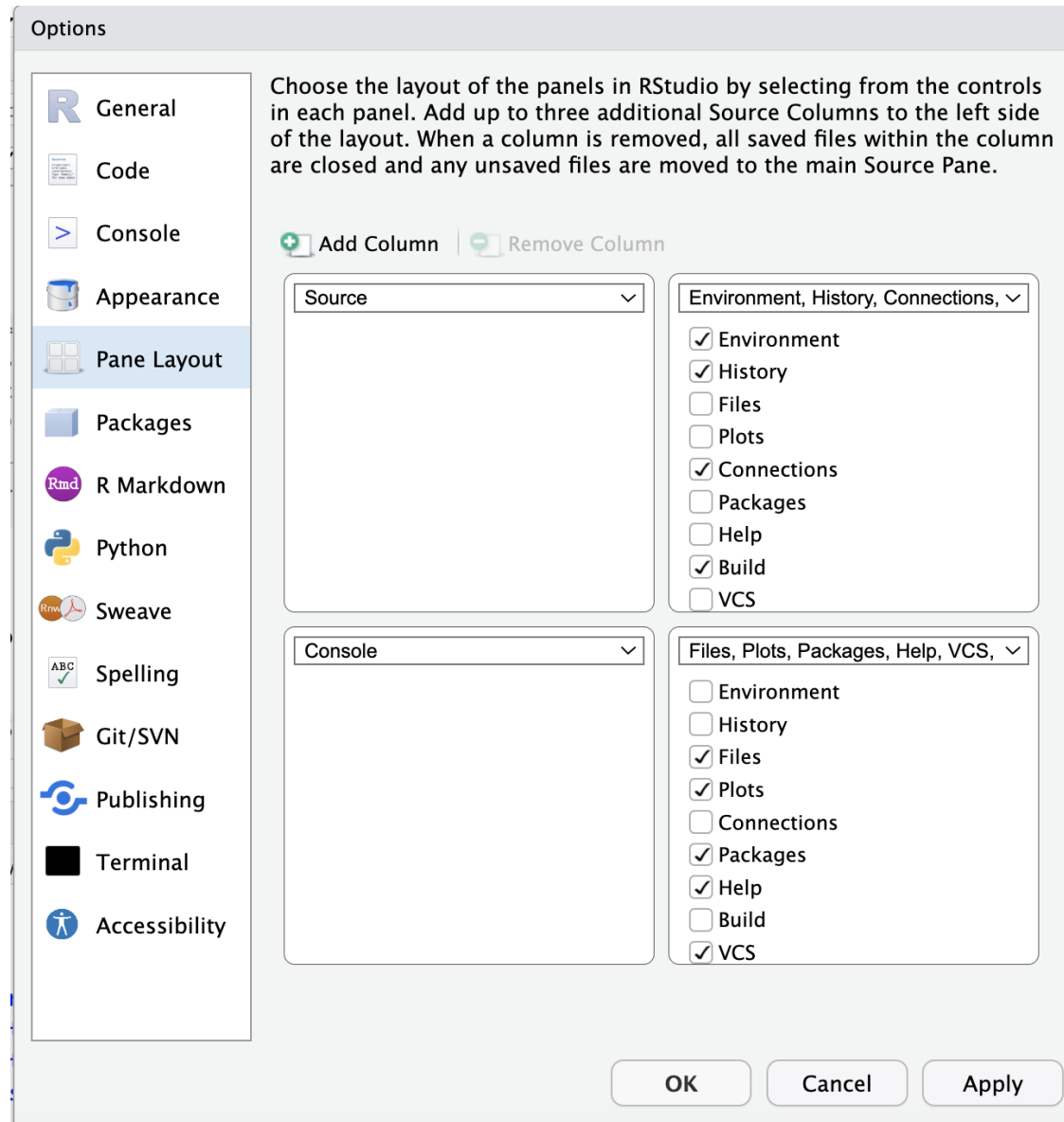
RStudio Layout

If RStudio doesn't look the way you want (or like our RStudio), then:

Click on the pane button, which looks like a waffle with 4 indentations. Scroll down to "Pane Layout".



Default Layout



Let's take a look at R Studio
ourselves!

Lab: Starting with R and RMarkdown

▮ [RStudio Lab](#)

To do this lab we need to:

- Download the file at the link above by clicking on it or go to the [website](#) schedule page
- Find the downloaded file on your computer
- Open the file in RStudio (double clicking the file name typically works)

These videos can help if you aren't sure where your downloads are:

If you have a PC: <https://youtu.be/we6vwB7DsNU>

If you have a Mac: <https://www.youtube.com/watch?v=Ao9e0cDzMrE>

You can find these on the resource page of the class website.

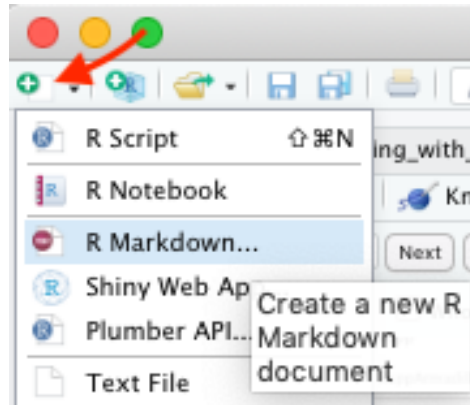
R Markdown file

R Markdown files (.Rmd) help generate reports that include your code and output. Think of them as fancier scripts.

1. Helps you describe your code
2. Allows you to check the output
3. Can create many different file types

Create an R Markdown file

Go to File → New File → R Markdown or click the green add file button.



Code chunks

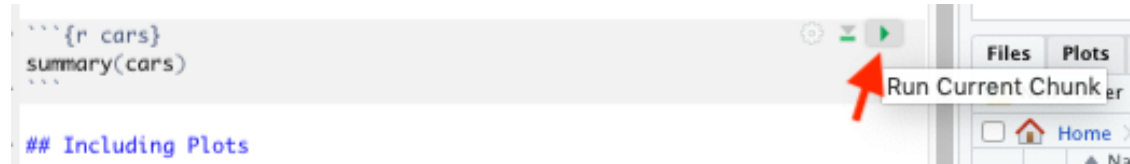
Within R Markdown files are code “chunks”.

This is where you can type R code and run it!



Run code in a chunk

Clicking the run (play) button runs the code in the chunk.



Ctrl + Enter on Windows or Command + Enter on Mac in your script evaluates that line of code

Running a chunk executes the code

- generally see a preview of the output of the code just below the chunk
- see the code in the console

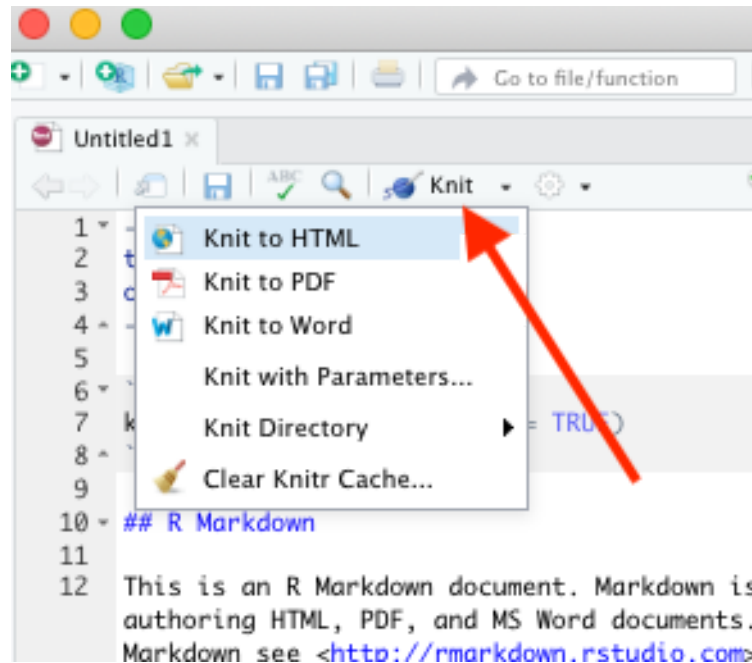
If you get annoyed by code previews in Markdown files...

See the [Help page](#) of the website. You can adjust this and change your RStudio settings:

Tools > Global Options > R Markdown

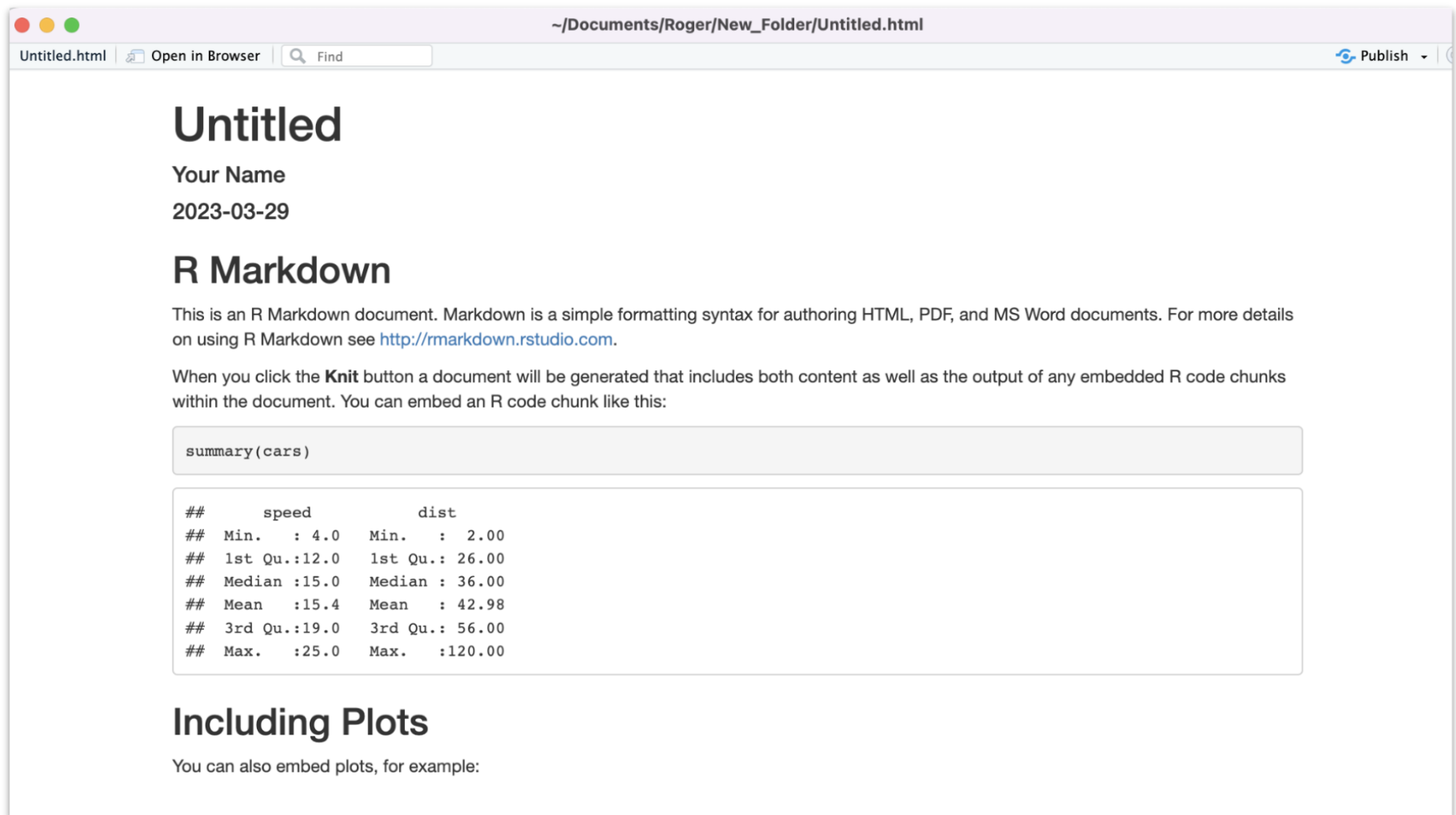
Knit file to html

Running all chunks - this will create a report from the R Markdown document!



Nice report!

This generates a nice report that you can share with others who can open in any browser.



The screenshot shows a web browser window displaying an R Markdown report. The browser's address bar shows the file path: `~/Documents/Roger/New_Folder/Untitled.html`. The report itself has a title "Untitled" and a subtitle "Your Name". The date "2023-03-29" is displayed below the subtitle. The main heading is "R Markdown". Below this, there is a paragraph explaining that this is an R Markdown document and providing a link to <http://rmarkdown.rstudio.com>. Another paragraph explains that clicking the "Knit" button generates a document including content and R code output. Below this, there is a code block containing the command `summary(cars)`. The output of this command is displayed in a table format. Finally, there is a section heading "Including Plots" with a note that plots can be embedded.

Untitled.html | Open in Browser | Find | Publish

Untitled

Your Name
2023-03-29

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see <http://rmarkdown.rstudio.com>.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

##	speed	dist
## Min.	: 4.0	Min. : 2.00
## 1st Qu.:	12.0	1st Qu.: 26.00
## Median :	15.0	Median : 36.00
## Mean :	15.4	Mean : 42.98
## 3rd Qu.:	19.0	3rd Qu.: 56.00
## Max.	:25.0	Max. :120.00

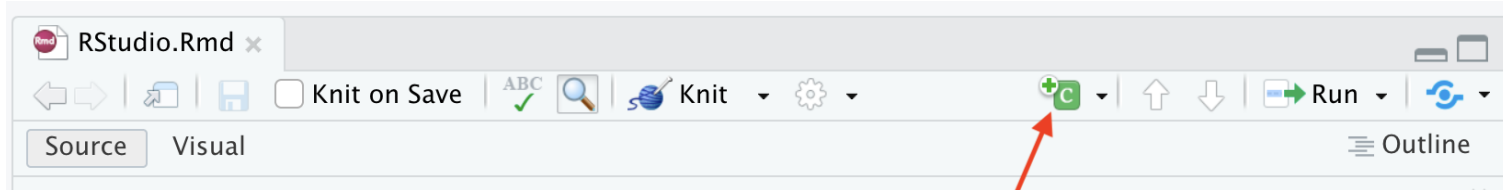
Including Plots

You can also embed plots, for example:

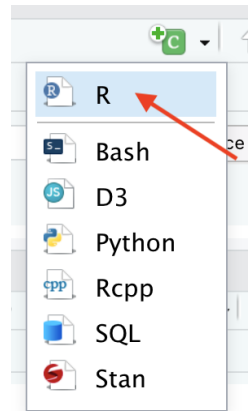
Create Chunks

To create a new R code chunk:

- Use the insert code chunk button at the top of RStudio.



- Select R (default) as the language:



Create Chunks

If you like keyboard shortcuts:

- Windows & Linux use Ctrl+Alt+I
- Mac use Command+Option+I

I is for insert.

Run previous chunks button

You can run all chunks above a specific chunk using this button:

```
```{r, out.width = "80%", echo = FALSE, fig.align='center'}  
knitr::include_graphics("images/chunk.png")
```
```



Errors

R studio can help you find issues in your code. Note that sometimes the error occurs earlier than RStudio thinks.



The screenshot shows a snippet of R code in the RStudio editor. Line 305 contains `print(x, ...)`, which is highlighted with a yellow background. Line 306 contains `{r}`. Line 307 contains `print(x))|` and is marked with a red 'x' icon, indicating an error. A tooltip box is open over the error, displaying the messages: `unexpected token ')'` and `unexpected end of document`. To the right of the code editor, there are icons for settings (a gear), a console window (a downward arrow), and a run button (a green play button).

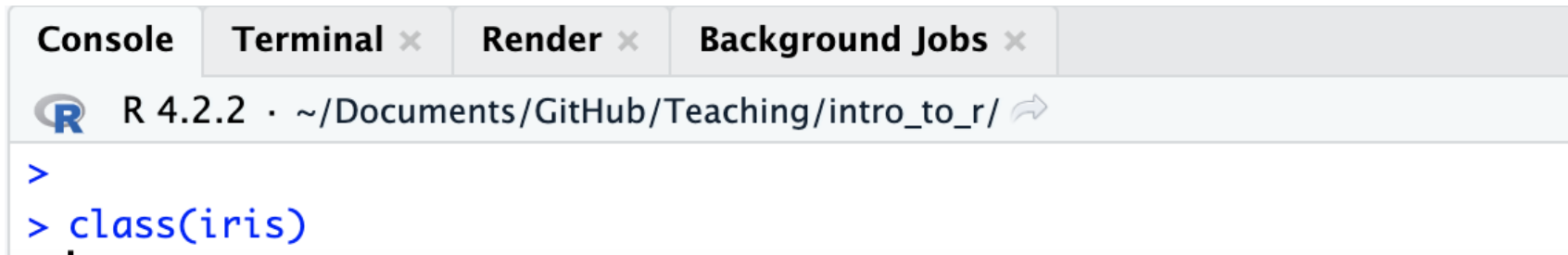
```
305 print(x, ...)  
306 {r}  
307 print(x))|  
308 ``````  
3 unexpected token ')'  
3 unexpected end of document
```

Useful R Studio Shortcuts

- `Ctrl + Enter` on Windows or `Command + Enter` on Mac in your script evaluates that line of code
 - It's like copying and pasting the code into the console for it to run.
- `Ctrl+1` on Windows or `Command + 1` on Mac takes you to the script page
- `Ctrl+2` on Windows or `Command + 2` on Mac takes you to the console
- http://www.rstudio.com/ide/docs/using/keyboard_shortcuts

Recap of where code goes

- you can test code in the console



The screenshot shows the RStudio interface with the 'Console' tab selected. The title bar indicates 'R 4.2.2 · ~/Documents/GitHub/Teaching/intro_to_r/'. The console contains the following text:

```
>  
> class(iris)
```

- you can save code in a chunk in the editor (Markdown file)

R Markdown

Code does not go here and instead goes within the grey chunks like this:

```
```${r}  
summary(cars)
```
```

Getting help from the preview

When you type in a function name, a pop up will preview documentation to help you. It also helps you remember the name of the function if you don't remember all of it!

The screenshot shows the RStudio interface. In the console, the user has typed `> class`. A dropdown menu is visible, listing several functions: `class` (base), `class::`, `class<-` (base), `classesToAM` (methods), `classInt::`, `classLabel` (methods), and `classMetaName` (methods). A pop-up window titled `class(x)` is open, displaying the documentation for the `class` function. The documentation includes the title **Object Classes** and a description: "R possesses a simple generic function mechanism which can be used for an object-oriented style of programming. Method dispatch takes place based on the class of the first argument to the generic function." Below the description, it says "Press F1 for additional help".

| Function | Package |
|----------------------------|-----------|
| <code>class</code> | {base} |
| <code>class::</code> | |
| <code>class<-</code> | {base} |
| <code>classesToAM</code> | {methods} |
| <code>classInt::</code> | |
| <code>classLabel</code> | {methods} |
| <code>classMetaName</code> | {methods} |

class(x)
Object Classes
R possesses a simple generic function mechanism which can be used for an object-oriented style of programming. Method dispatch takes place based on the class of the first argument to the generic function.
Press F1 for additional help

The screenshot shows the RStudio interface. In the console, the user has typed `> read_`. A dropdown menu is visible, listing several functions: `read_builtin` (readr), `read_chunk` (knitr), `read_csv` (readr), `read_csv2` (readr), `read_csv2_chunked` (readr), `read_csv_chunked` (readr), and `read_delim` (readr). A pop-up window titled `read_csv(file, col_names = TRUE, col_types = NULL, ...)` is open, displaying the documentation for the `read_csv` function. The documentation includes the title **read_csv()** and a description: "read_csv(file, col_names = TRUE, col_types = NULL, ...). This function reads a CSV file into a data frame. It uses the readr package for reading CSV files." Below the description, it says "Press F1 for additional help".

| Function | Package |
|--------------------------------|---------|
| <code>read_builtin</code> | {readr} |
| <code>read_chunk</code> | {knitr} |
| <code>read_csv</code> | {readr} |
| <code>read_csv2</code> | {readr} |
| <code>read_csv2_chunked</code> | {readr} |
| <code>read_csv_chunked</code> | {readr} |
| <code>read_delim</code> | {readr} |

read_csv(file, col_names = TRUE, col_types = NULL, ...)
This function reads a CSV file into a data frame. It uses the readr package for reading CSV files.
Press F1 for additional help

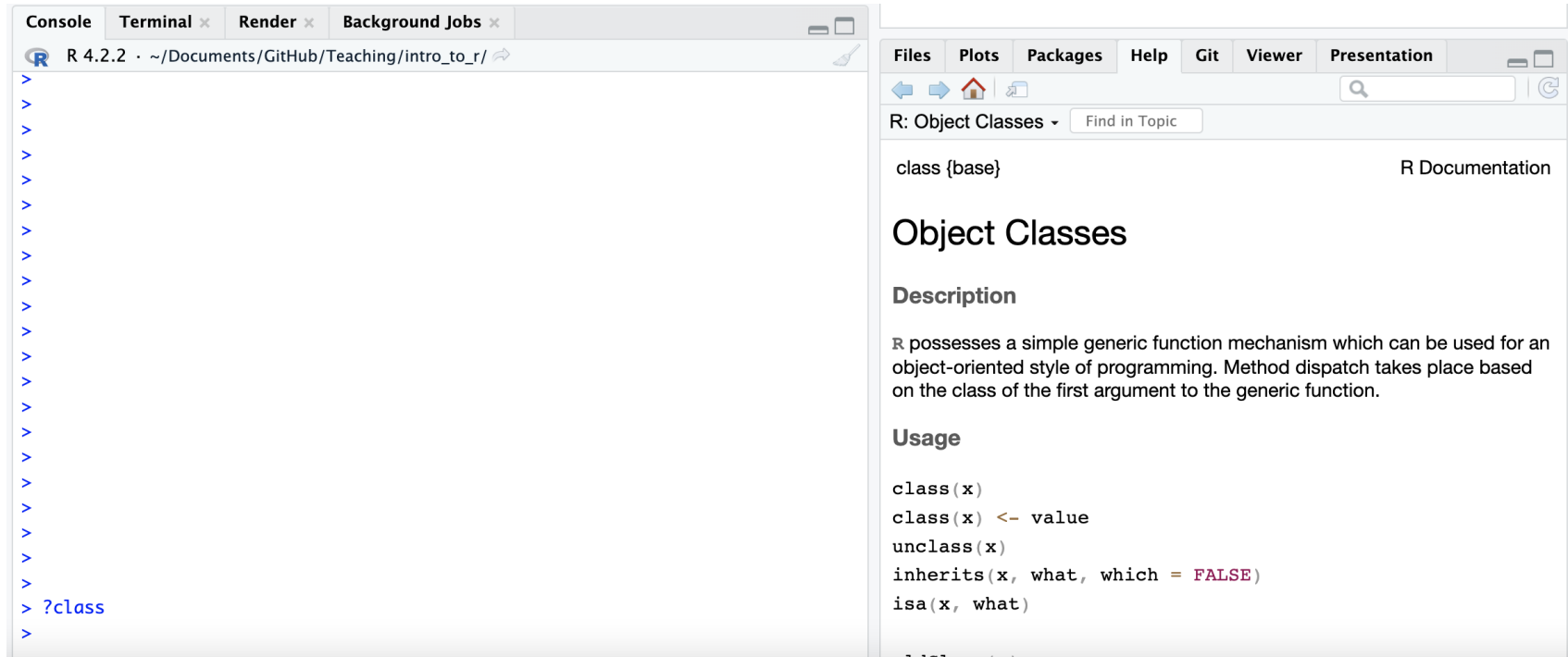
Get help with the help pane

Getting Help with ?

If you know the name of a package or function:

Type `?package_name` or `?function_name` in the console to get information about packages and functions.

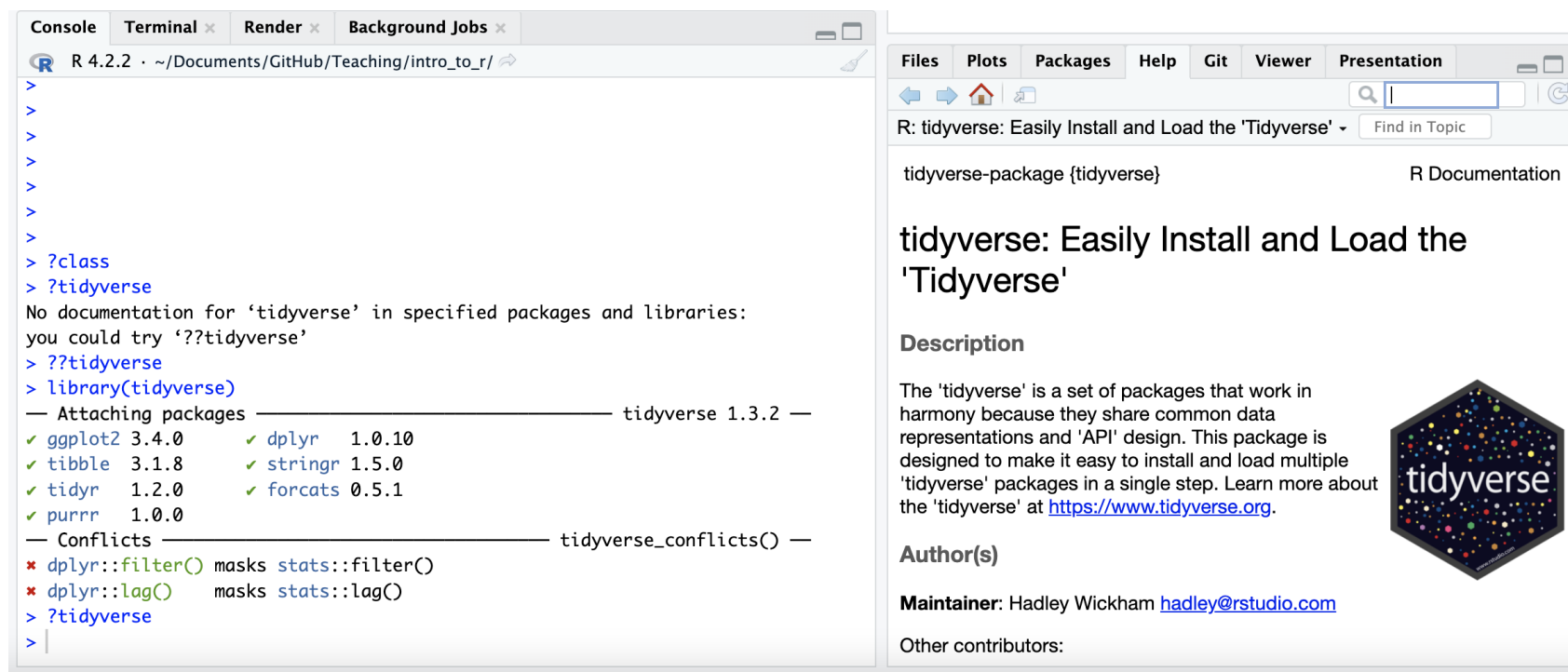
For example: `?readr` or `?read_csv`.



Double Question Mark

If you haven't loaded a package yet into R than you may get a response that there is no documentation.

Typing in `??package_name` can show you packages that you haven't loaded yet.



The screenshot shows the R Studio interface. The console on the left displays the following commands and output:

```
>
>
>
>
>
>
> ?class
> ?tidyverse
No documentation for 'tidyverse' in specified packages and libraries:
you could try '??tidyverse'
> ??tidyverse
> library(tidyverse)
— Attaching packages — tidyverse 1.3.2 —
✓ ggplot2 3.4.0      ✓ dplyr  1.0.10
✓ tibble  3.1.8      ✓ stringr 1.5.0
✓ tidyr   1.2.0      ✓ forcats 0.5.1
✓ purrr   1.0.0
— Conflicts — tidyverse_conflicts() —
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag() masks stats::lag()
> ?tidyverse
>
```

The help window on the right shows the documentation for the tidyverse package. The title is "tidyverse: Easily Install and Load the 'Tidyverse'". The description states: "The 'tidyverse' is a set of packages that work in harmony because they share common data representations and 'API' design. This package is designed to make it easy to install and load multiple 'tidyverse' packages in a single step. Learn more about the 'tidyverse' at <https://www.tidyverse.org>." The author is listed as Hadley Wickham, with the email hadley@rstudio.com. The maintainer is also Hadley Wickham. The window also shows the tidyverse logo, which is a hexagon with a starry pattern and the word "tidyverse" inside.

Summary

- RStudio makes working in R easier
- the Editor (top) is for static code like scripts or R Markdown documents
- The console is for testing code (bottom) - best to save your code though!
- R markdown documents are really helpful for lots of reasons!
- R code goes within what is called a chunk (the gray box with a green play button)
- Code chunks can be modified so that they show differently in reports

▮ [Class Website](#)

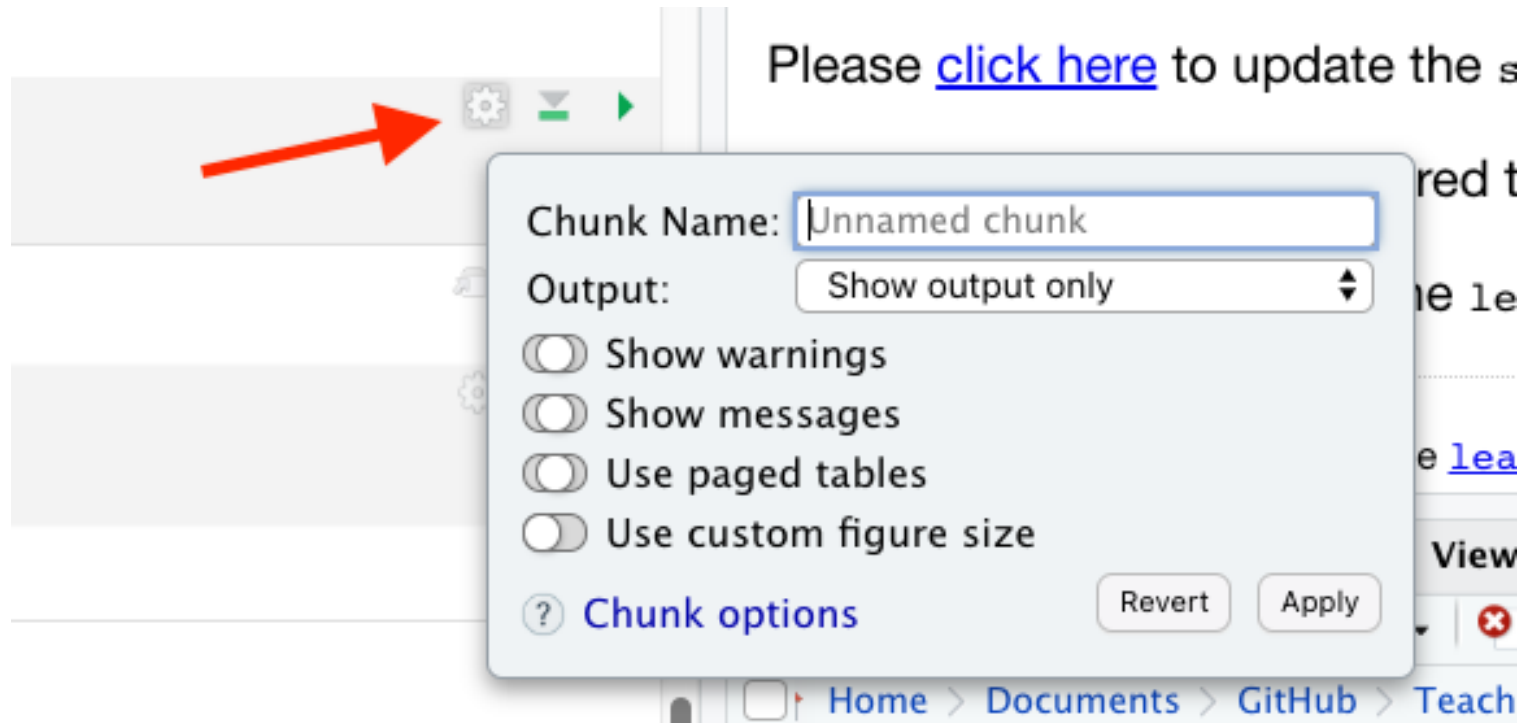
▮ [Lab](#)



Image by [Gerd Altmann](#) from [Pixabay](#)

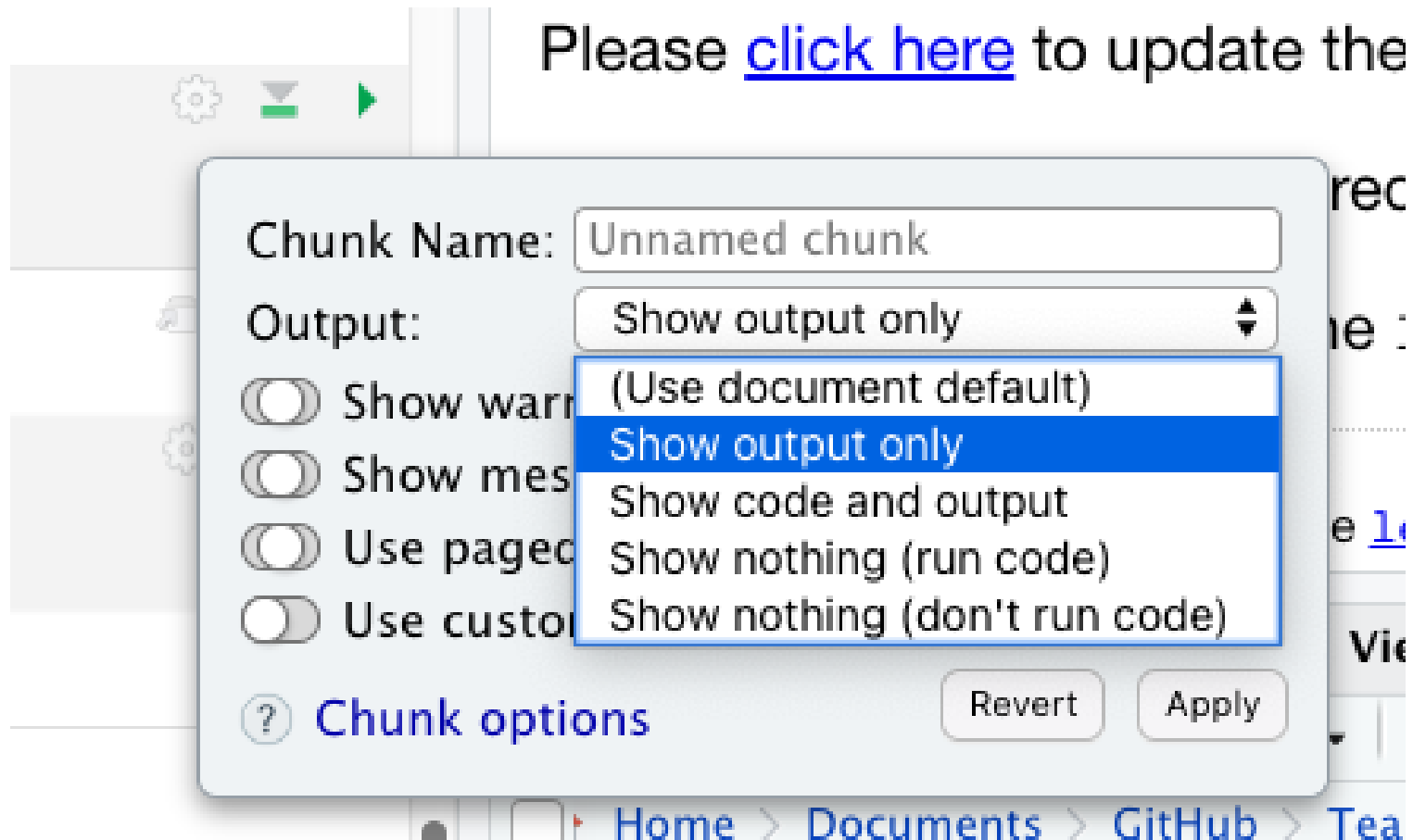
Extra Slides

Chunk settings



Chunk settings

You can specify if a chunk will be seen in the report or not.



Sometimes you want to hide your code

If you want to keep your code so people can see it if they want to there is a nice option called code folding - check it out here:

<https://stackoverflow.com/questions/69326576/show-output-but-hide-code-when-sending-rmd-to-other-people>

Rainbow Parentheses

Tools -> Global Options -> Code -> Display -> Use rainbow parentheses

This can help you see your code more easily.

Press enter to save this setting and get out of this menu.

(((((({{{{[[[[[Enjoy your colorful code!]]]]}}}})))))