

RStudio

Working with R – RStudio

RStudio is an Integrated Development Environment (IDE) for R and it helps you:

- write code - makes suggestions
- view the output of your code, including plots
- find errors
- manage files
- view documentation

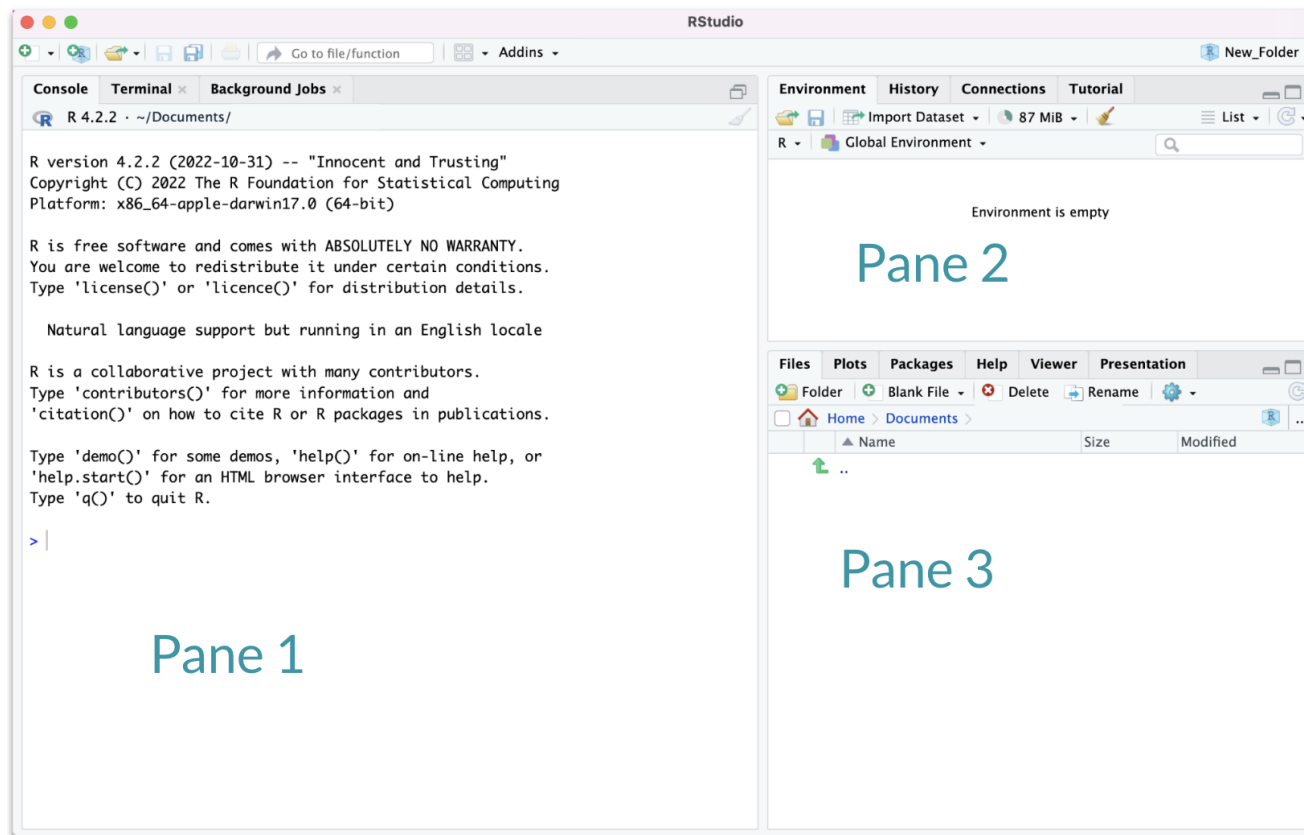


[\[source\]](#)

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

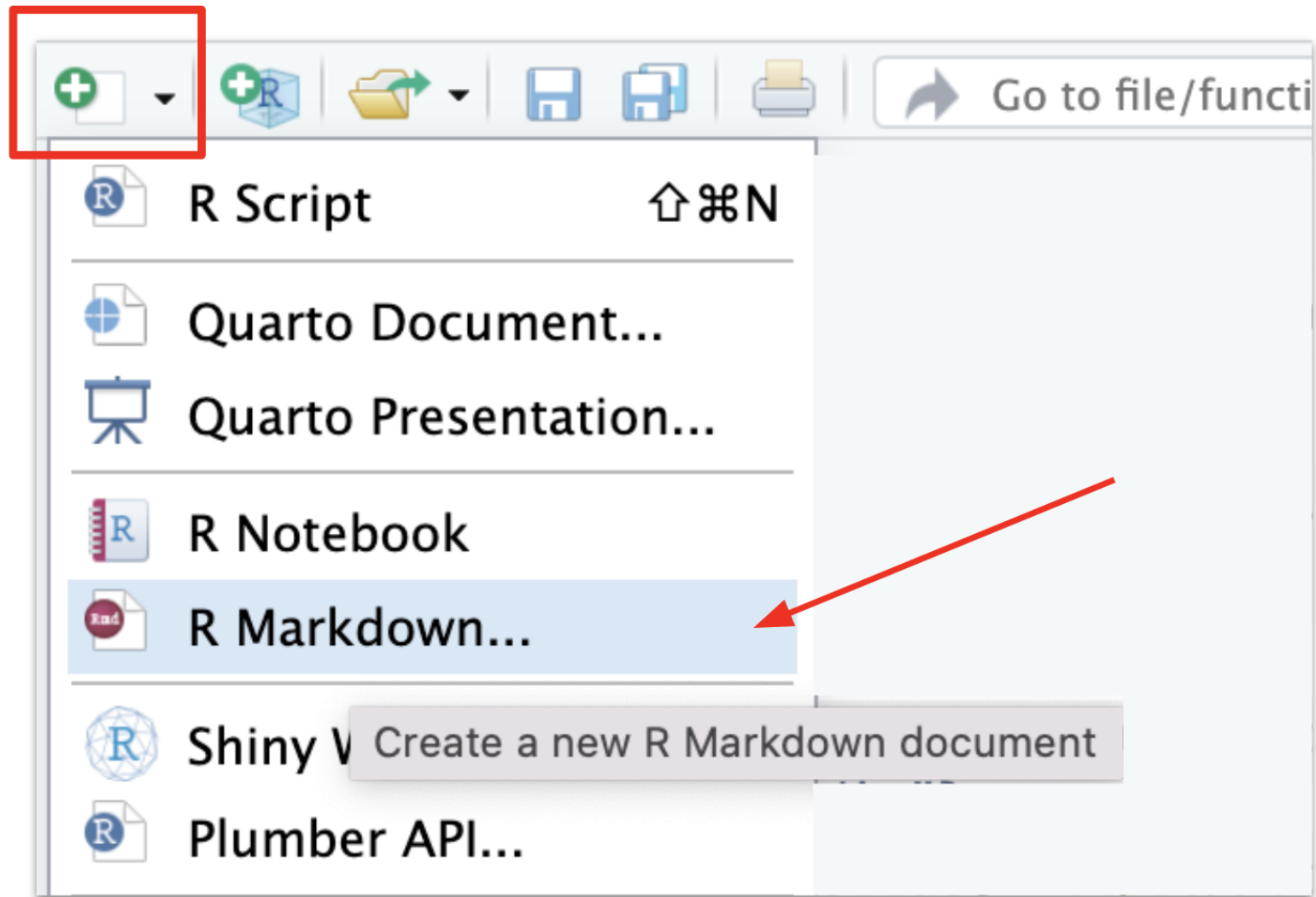
RStudio

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.

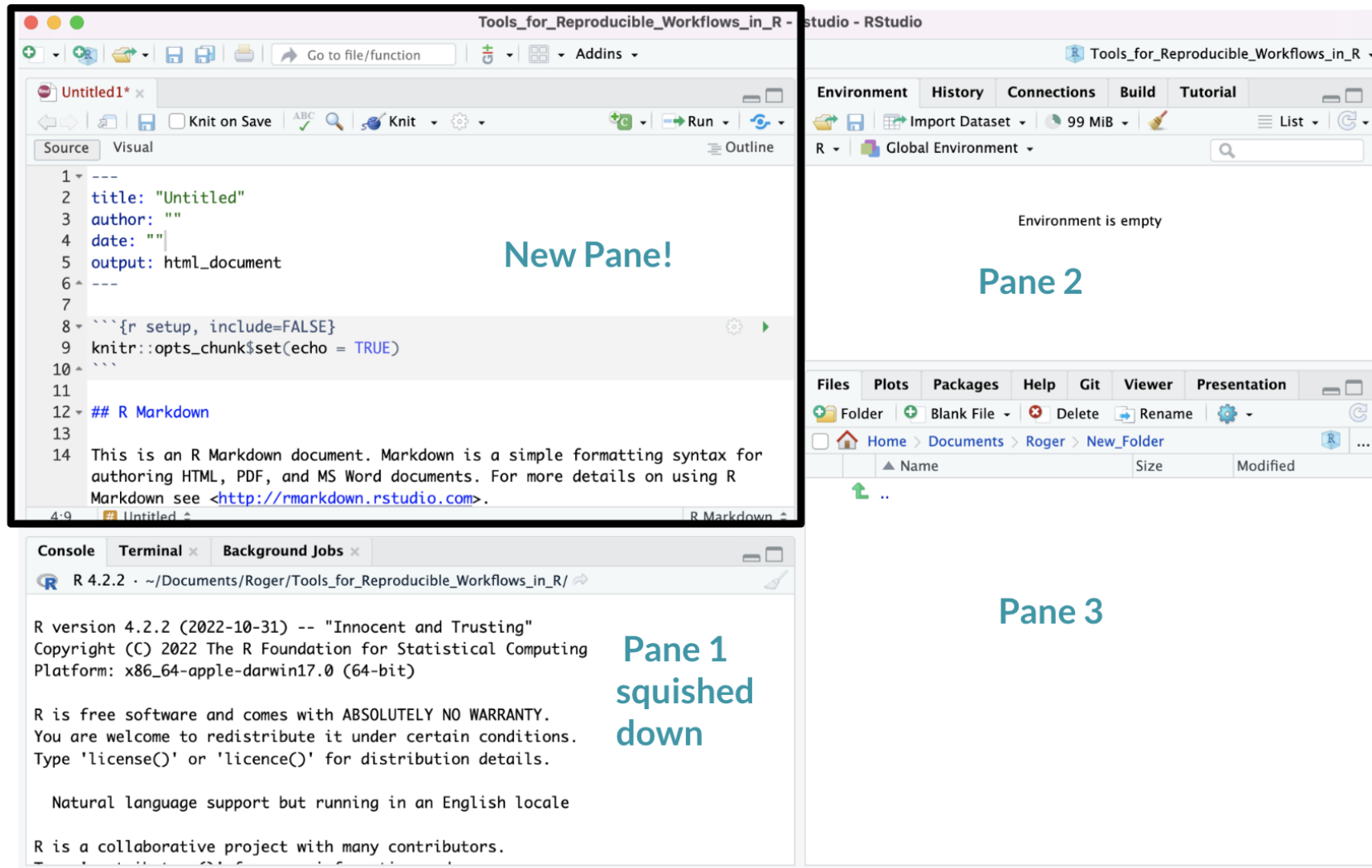


Hidden Pane

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**:

- Top by default
- **saves your code**

2. The **R Console**:

- Bottom by default
- Calculator
- Place to try things out, then add to your editor
- **doesn't save your code**

RStudio

Super useful “cheatsheet”: [LINK](#)

Write Code

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

R Support

Import data with wizard
History of past commands to run/copy
Display .RPres slideshows
File > New File > R Presentation

The screenshot shows the RStudio interface with several panels and annotations:

- Source Editor:** Contains R code with annotations for cursors, multiple cursors, code diagnostics, syntax highlighting, tab completion, and multi-language code snippets.
- Environment Panel:** Shows the Global Environment with objects like 'iris' and 'foo'. Annotations include 'Load workspace', 'Save workspace', 'Delete all saved objects', and 'Search inside environment'.
- Files Panel:** Shows the file browser with annotations for 'Create folder', 'Upload file', 'Delete file', 'Rename file', and 'Change directory'.
- Console:** Shows the output of the code execution with annotations for 'Working Directory' and 'Maximize, minimize panes'.

Annotations in the Source Editor include:

- 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

Annotations in the Environment Panel include:

- 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

Annotations in the Files Panel include:

- 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.

Annotations in the Console include:

- 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.

The screenshot shows the RStudio interface with a file named 'first_markdown' open. The file content is as follows:

```
1 ---
2 title: "first_markdown"
3 output: html_document
4 ---
5
6 ```{r setup, include=FALSE}
7 knitr::opts_chunk$set(echo = TRUE)
8 ```
9
10 ## R Markdown
11
12 This is an R Markdown document. Markdown is a simple formatting syntax for
13 authoring HTML, PDF, and MS Word documents. For more details on using R
14 Markdown see <http://rmarkdown.rstudio.com>.
15
16 When you click the Knit button a document will be generated that includes
17 both content as well as the output of any embedded R code chunks within the
18 document. You can embed an R code chunk like this:
19
20 ```{r cars}
21
```

The R console output shows the following text:

```
~/Documents/GitHub/Teaching/intro_to_r/
You are welcome to redistribute this under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

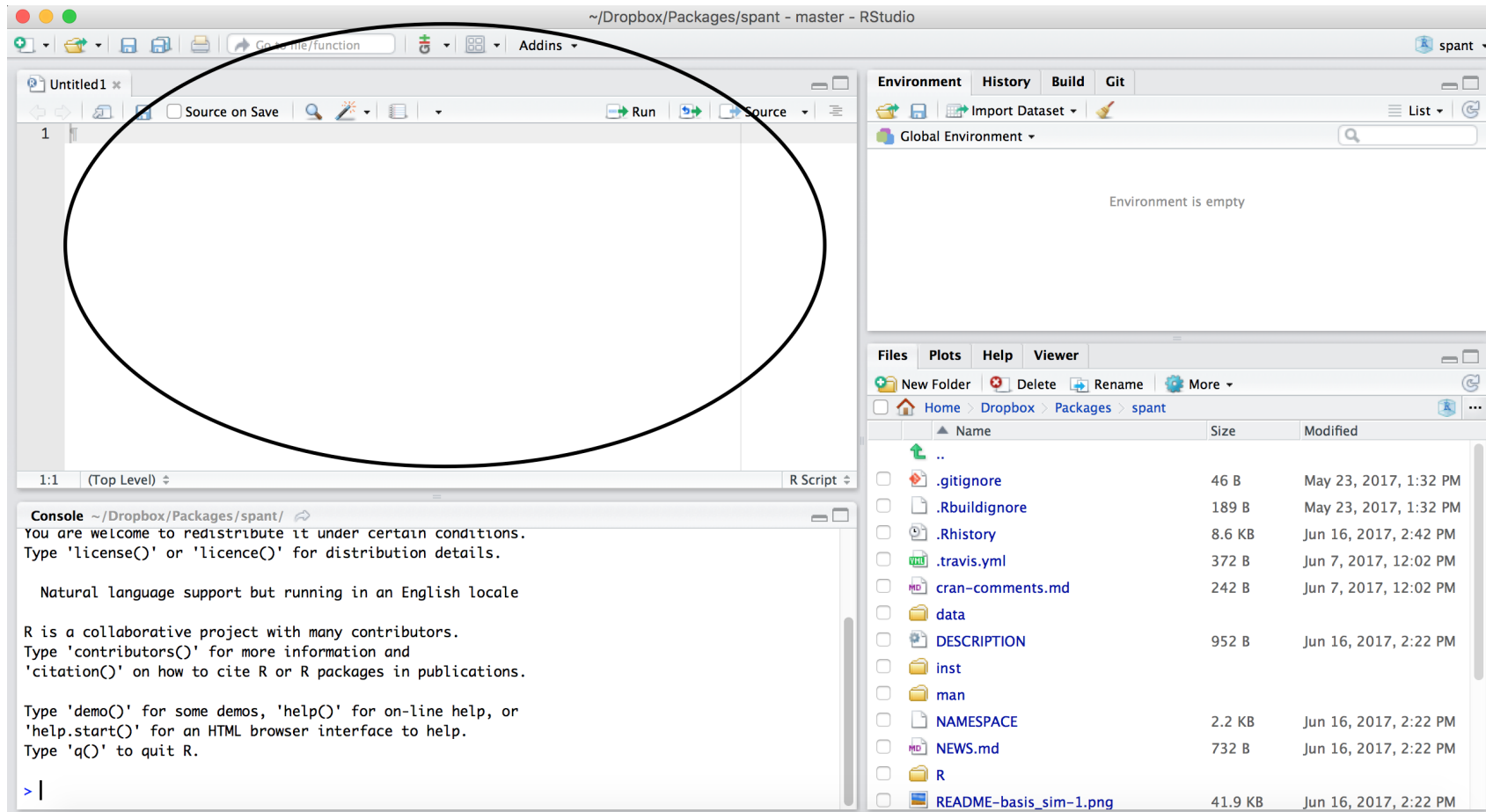
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

The file explorer on the right shows the following files and folders:

Name	Size	Modified
..		
.gitignore	245 B	May 18, 2021, 10:10
.Rbuildignore	16 B	May 18, 2021, 10:10
.Rhistory	43 B	Jun 10, 2021, 10:10
.travis.yml	666 B	Jun 9, 2021, 12:12
all_functions.xlsx	13.4 KB	Jun 8, 2021, 3:10
all_the_functions.csv	57.3 KB	Jun 8, 2021, 3:10
all_the_packages.txt	211 B	May 18, 2021, 10:10
Arrays_Split		
Basic_R		
Best_Model_Coefficients.csv	587 B	May 18, 2021, 10:10
Best_Model_Coefficients.xlsx	3.8 KB	May 18, 2021, 10:10
bibliography.bib	599 B	May 18, 2021, 10:10
black_and_white_theme.pdf	45.1 KB	May 18, 2021, 10:10
bloemhage_logo_small_horizontal	25.4 KB	May 18, 2021, 10:10

Scripts will just be empty

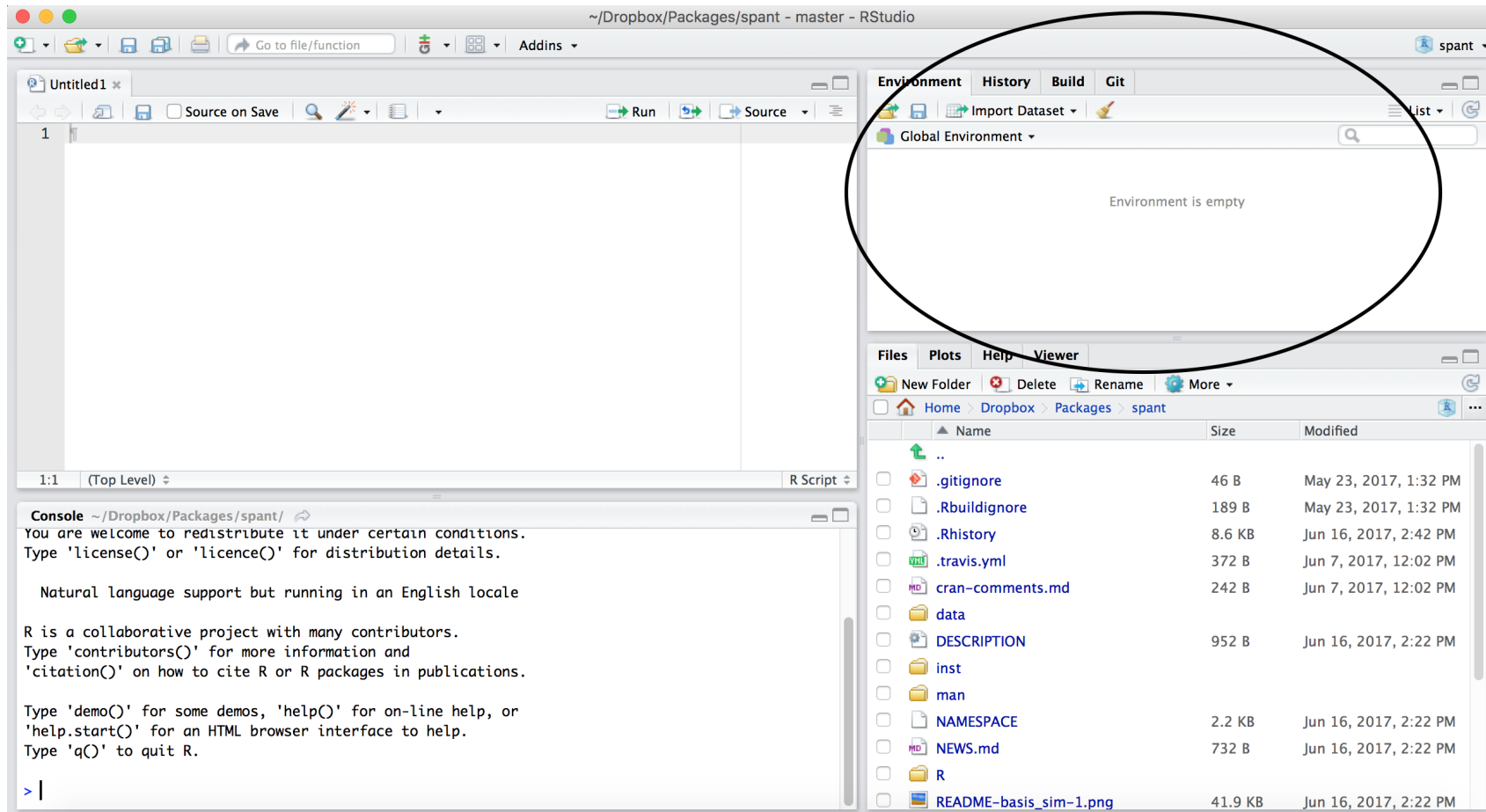


Scripts and R Markdown

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

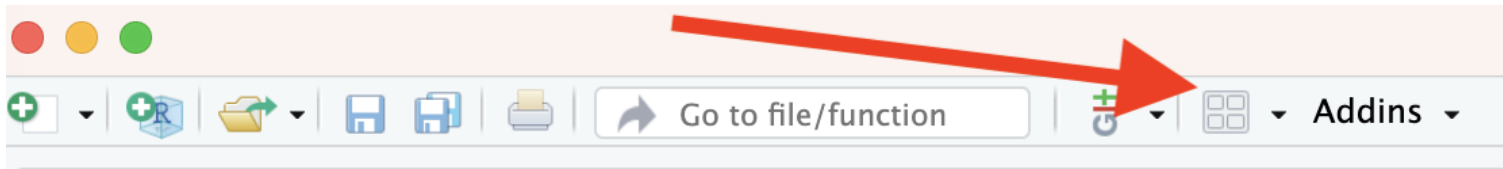
Lower Right Pane

- **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

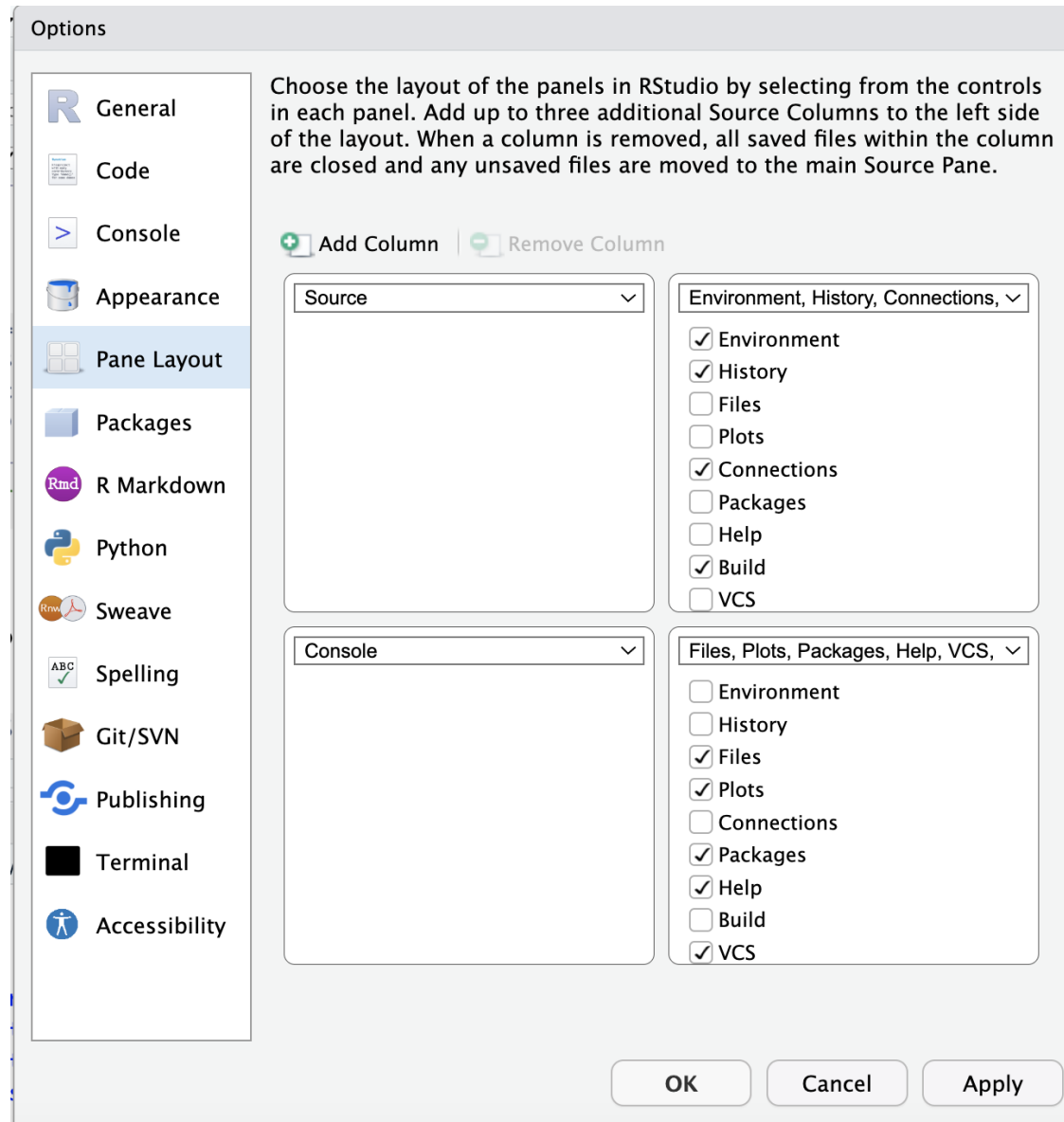
RStudio Layout

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!

R Markdown file

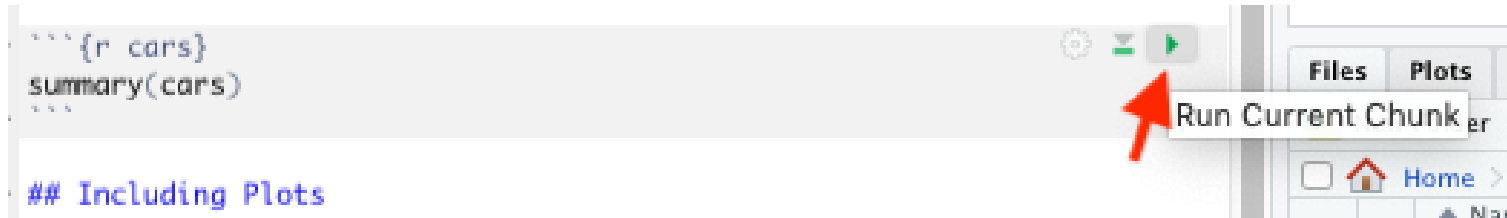
R Markdown files (.Rmd) help generate reports that include your code and output.

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

Code chunks

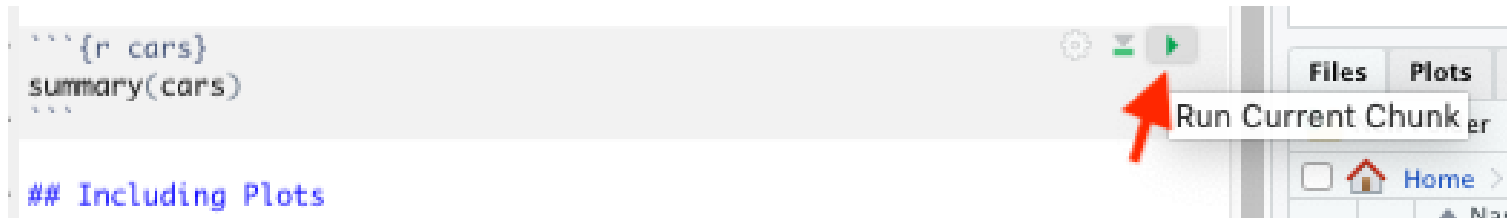
Within R Markdown files are code “chunks”.

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

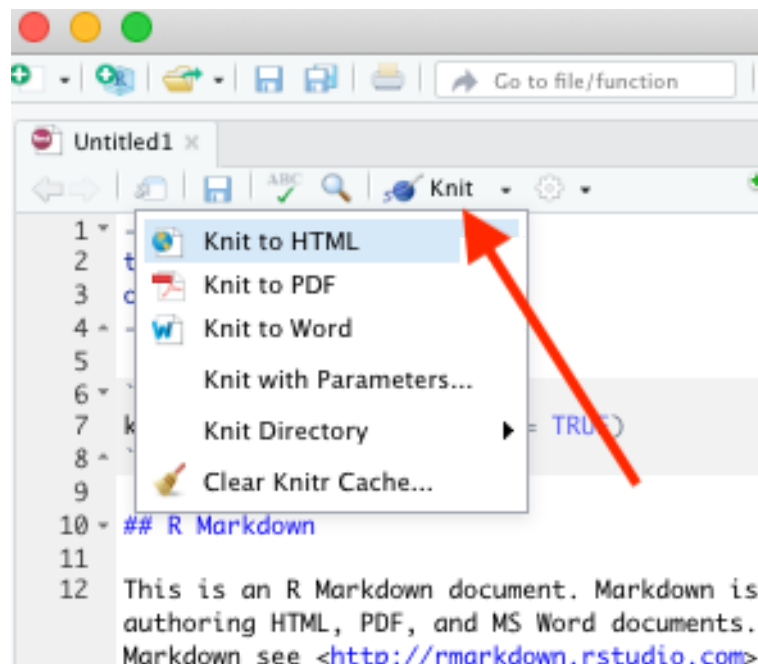
If you get annoyed by code previews in Markdown files...

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

Tools > Global Options > Appearance

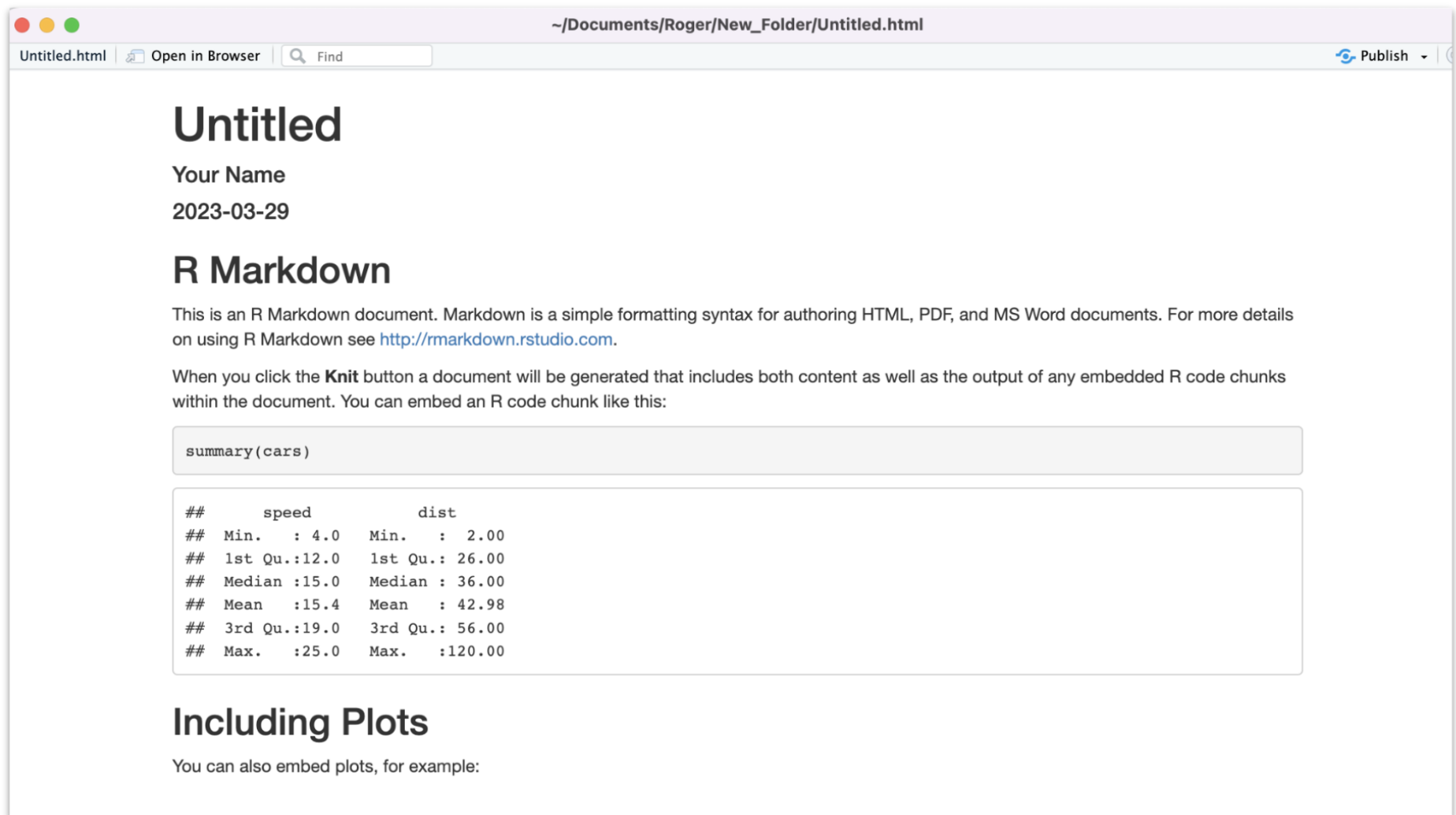
Knit file to html

Running all chunks - this will create a report from the R Markdown document!
Note that it can't use anything not included in the file, it can't use objects in your environment that you were modifying interactively.



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", followed by 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, a code block contains the command `summary(cars)`, and a preformatted text block shows the resulting summary statistics for the 'cars' dataset. The final heading is "Including Plots", with a note that plots can also 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
```

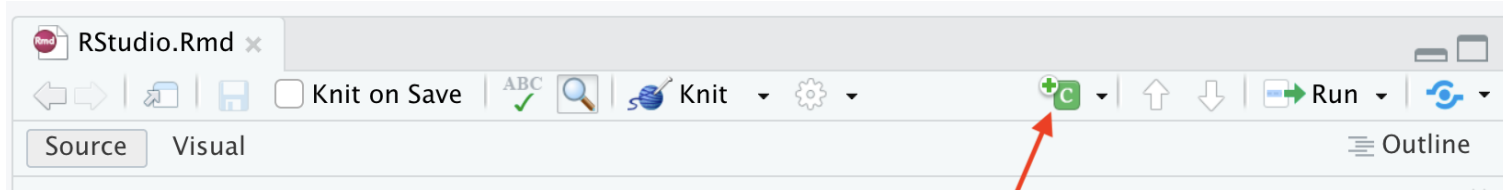
Including Plots

You can also embed plots, for example:

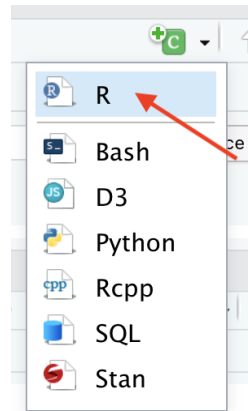
Create Chunks

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.

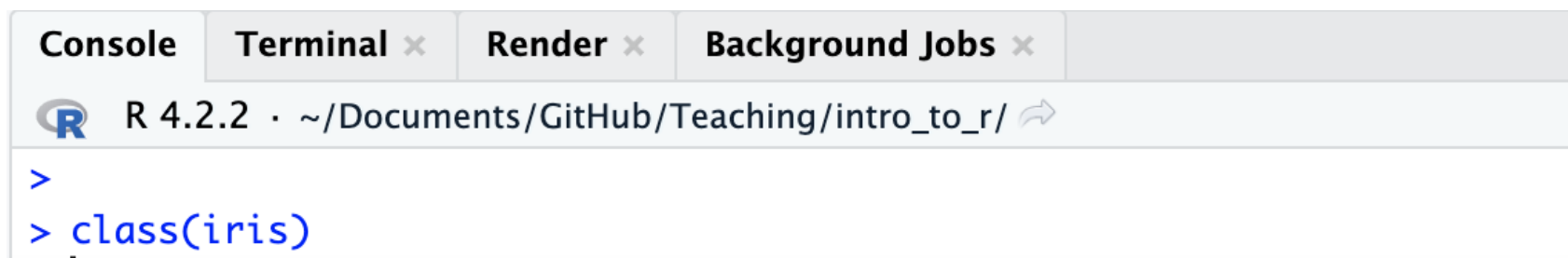


The screenshot shows a snippet of R code in the RStudio editor. Line 305 contains `print(x, ...)`, which is highlighted with a yellow background. Line 306 contains `{r}`. Line 307 contains `print(x))|`, with a red 'x' icon to its left indicating an error. A tooltip box is open over the closing parenthesis of line 307, 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 dropdown menu (a triangle), and a run button (a green play icon).

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

Recap of where code goes

- you can test code in the console



The screenshot shows the RStudio console interface. At the top, there are four tabs: 'Console', 'Terminal', 'Render', and 'Background Jobs'. The 'Console' tab is active. Below the tabs, the R logo and version 'R 4.2.2' are displayed, followed by the file path '~/Documents/GitHub/Teaching/intro_to_r/'. The console prompt is '>', and the code 'class(iris)' has been entered, with a cursor at the end of the line.

- 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)
```
```

Gut Check

Why are R Markdown files so useful?

1. They let you test your code
2. They let you view the output of your code
3. They let you generate cool reports
4. All of the above

Gut Check

Where does code go typically in an Rmd file?

A

```
```{r}
```

**B**

```
```
```

C

Gut Check

Which button do you click to run the code in a current chunk?

```
```{r}  
library(tidyverse)
```
```



A



B

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). To the right, a yellow pop-up window displays the documentation for `class(x)` under the heading "Object Classes". The text explains that R has a generic function mechanism for object-oriented programming, where method dispatch is based on the class of the first argument. A footer in the pop-up 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). To the right, a yellow pop-up window displays the documentation for `read_csv(file, col_names = TRUE, col_types = NULL, col_select = NULL, id = NULL, locale = default_locale(), na = c("", "NA"), quoted_na = TRUE, quote = "\"", comment = "", trim_ws = TRUE, skip = 0, n_max = Inf, guess_max = min(1000, n_max), name_repair = "unique", num_threads = readr_threads(), progress = show_progress(), show_col_types = should_show_types(), skip_empty_lines = TRUE, show_col_types = should_show_col_types())` under the heading "The 'tidyverse'". The text explains that this function reads a CSV file into a tibble. A footer in the pop-up 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, col_select = NULL, id = NULL, locale = default_locale(), na = c("", "NA"), quoted_na = TRUE, quote = "\"", comment = "", trim_ws = TRUE, skip = 0, n_max = Inf, guess_max = min(1000, n_max), name_repair = "unique", num_threads = readr_threads(), progress = show_progress(), show_col_types = should_show_types(), skip_empty_lines = TRUE, show_col_types = should_show_col_types())

The 'tidyverse'

Press F1 for additional help

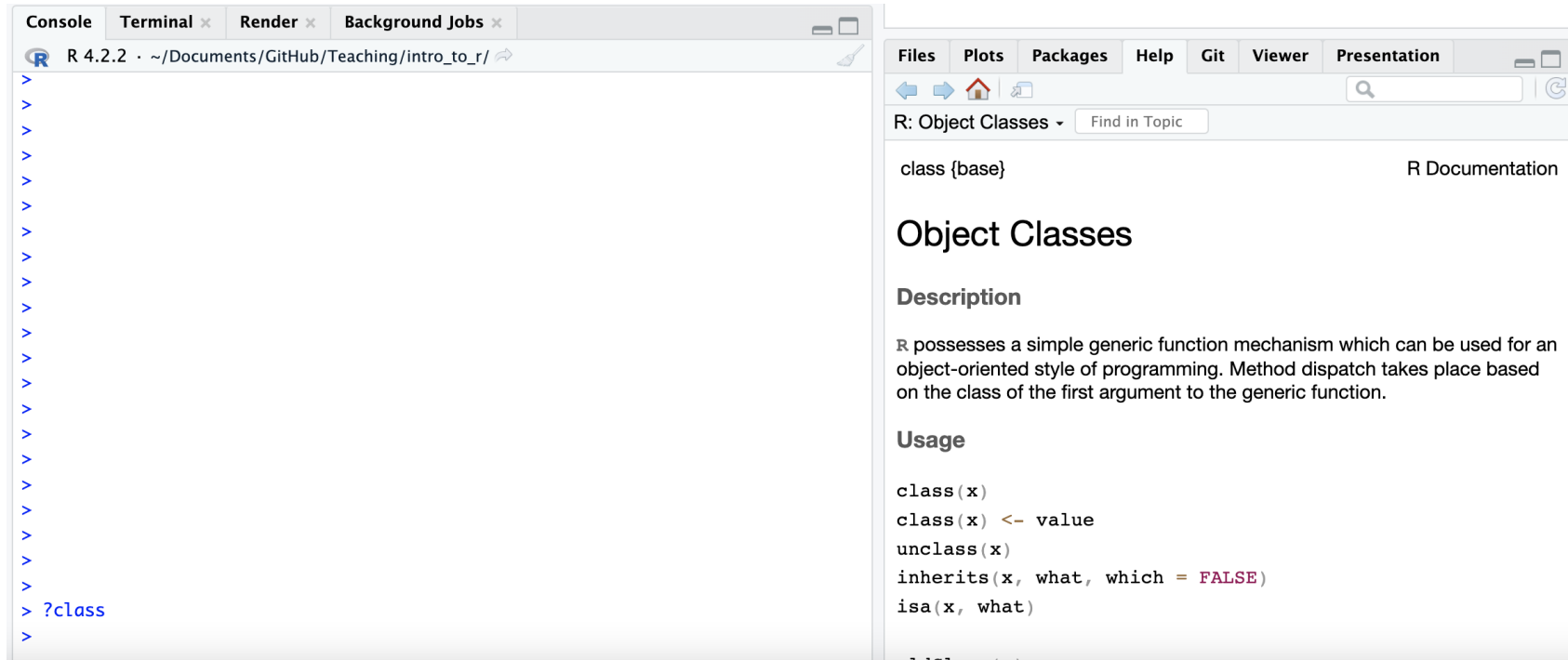
Get help with the help pane

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.

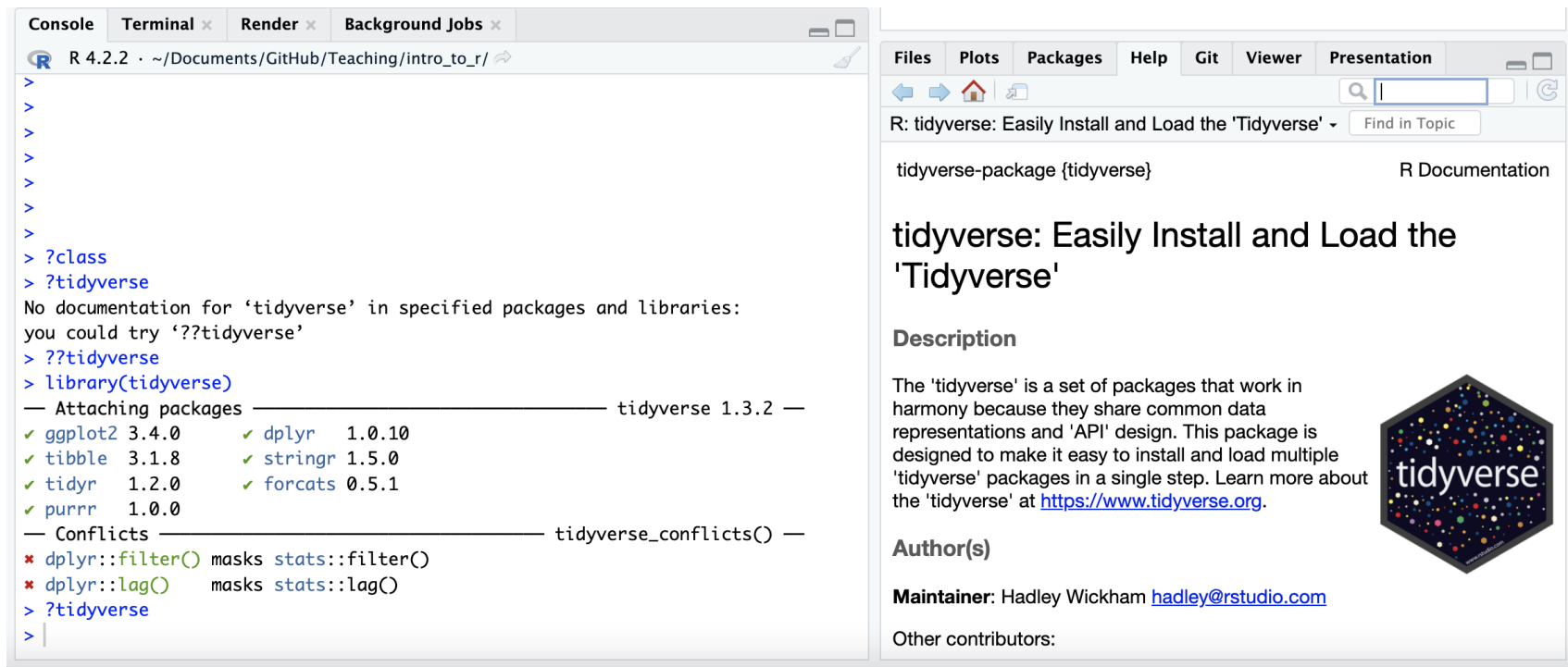
For example: `?readr` or `?read_csv`.



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, with the email hadley@rstudio.com. The tidyverse logo is displayed on the right side of the help window.

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)

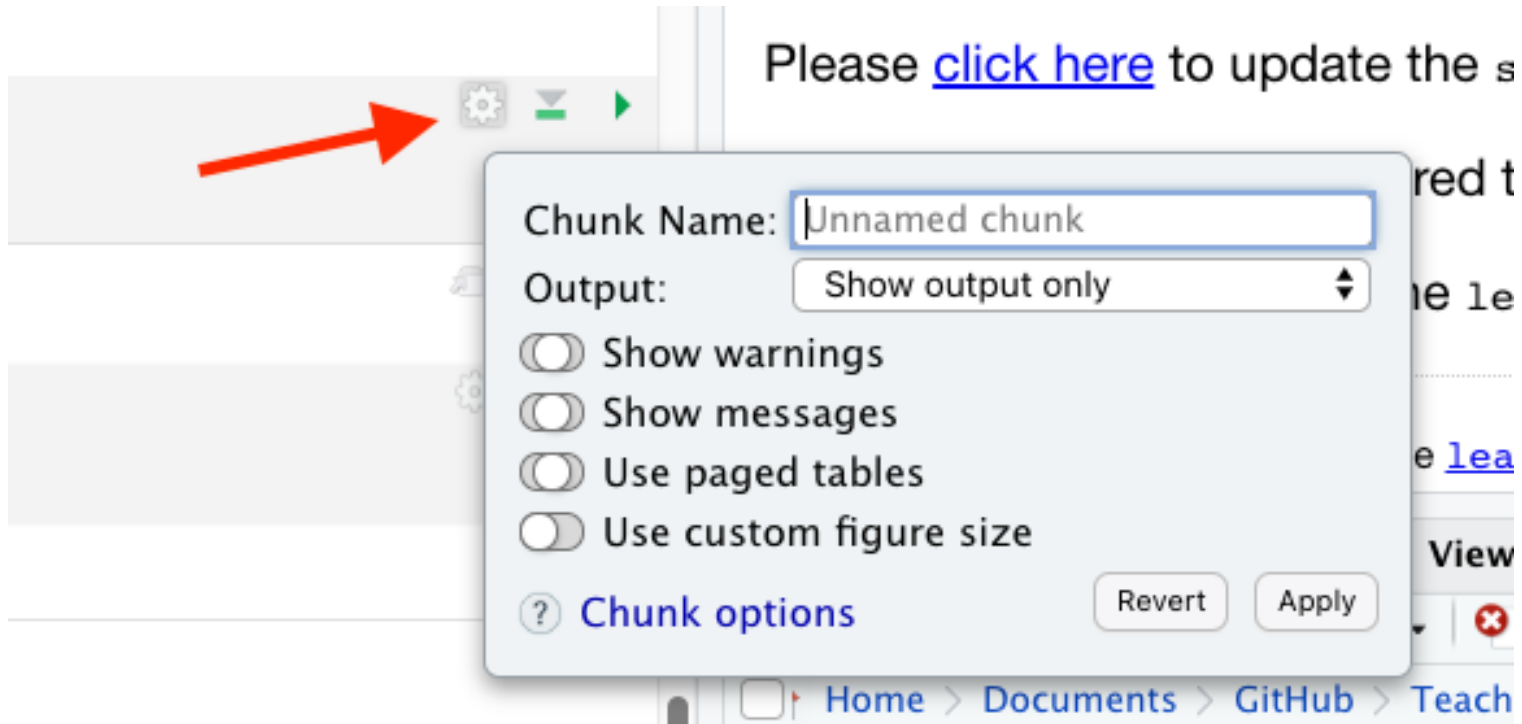
▢ [Class Website](#) ▢ [Lab](#) ▢ [Posit Cheatsheet](#) ▢ [Day 1 Cheatsheet](#)



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

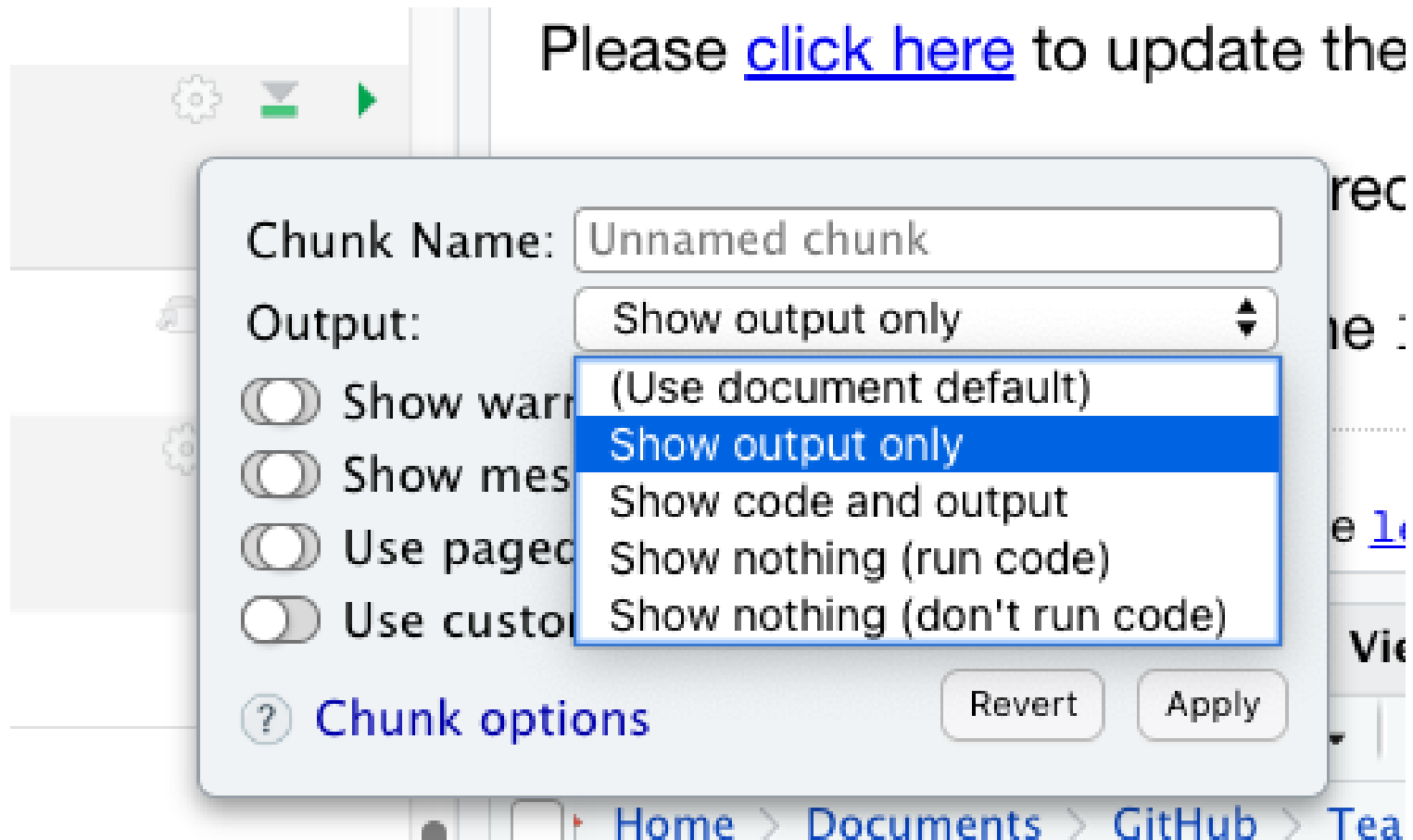
Extra Slides

Chunk settings



Chunk settings

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

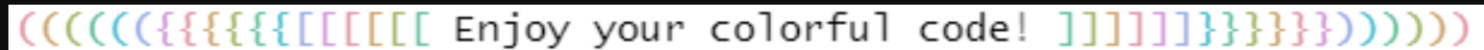


Rainbow Parentheses

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.

A screenshot of a code editor with a black background. A single line of text is displayed in a light gray font. The text consists of a series of rainbow-colored parentheses: (((((({{{[[[[[[Enjoy your colorful code!]]]]]}}}})))). The colors of the parentheses transition through a rainbow spectrum from left to right. The text "Enjoy your colorful code!" is in the center of the line, flanked by the parentheses. The entire line is highlighted with a light gray background.

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