

# R Workflow Best Practices

bmRn CSM: managing code, data, and files

Martin Frigaard

2020-12-13

# R Workflow Best Practices

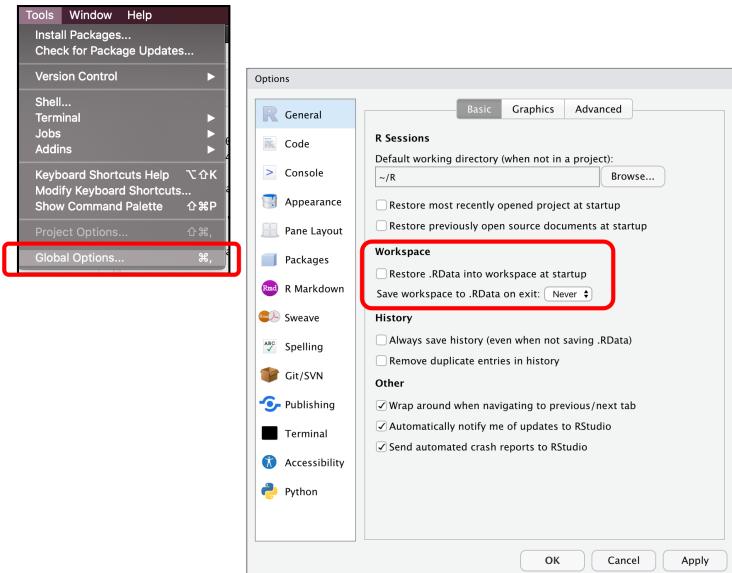
Managing your code, data, and files with RStudio

# Change RStudio's Default Settings

Click on **Tools > Global Options...**

- We want to uncheck "*Restore .RData into work space at start up*"

- We also want to make sure we change "*Save work space to .Rdata on exit*" to "Never"



# Customize RStudio

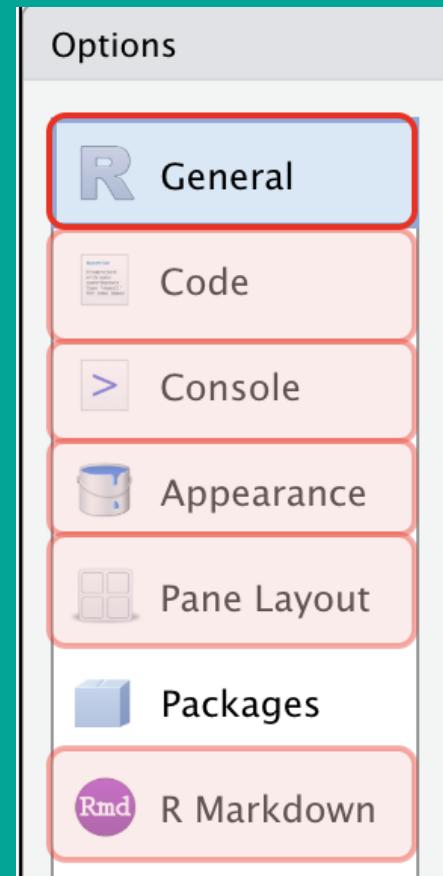
Code

Console

Appearance

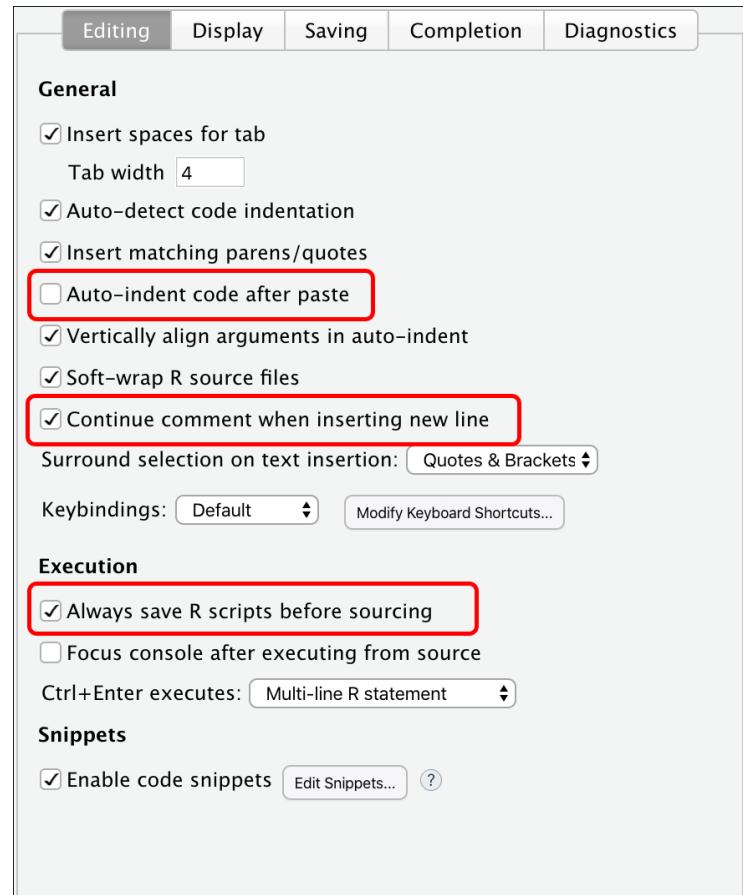
Pane Layout

R Markdown



# Code Editing

- Auto indent?
- Continue comment lines?
- Save R scripts before sourcing?

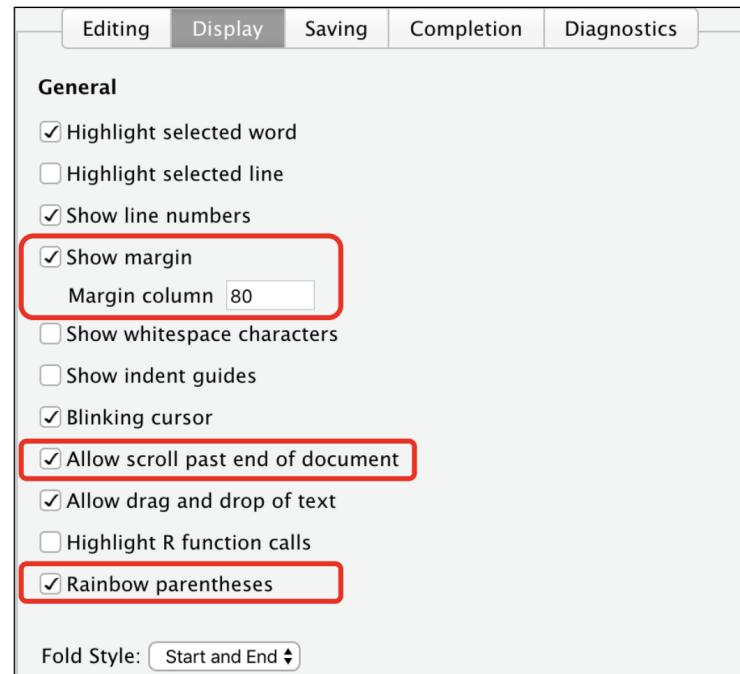


# Code Display

- Margins?

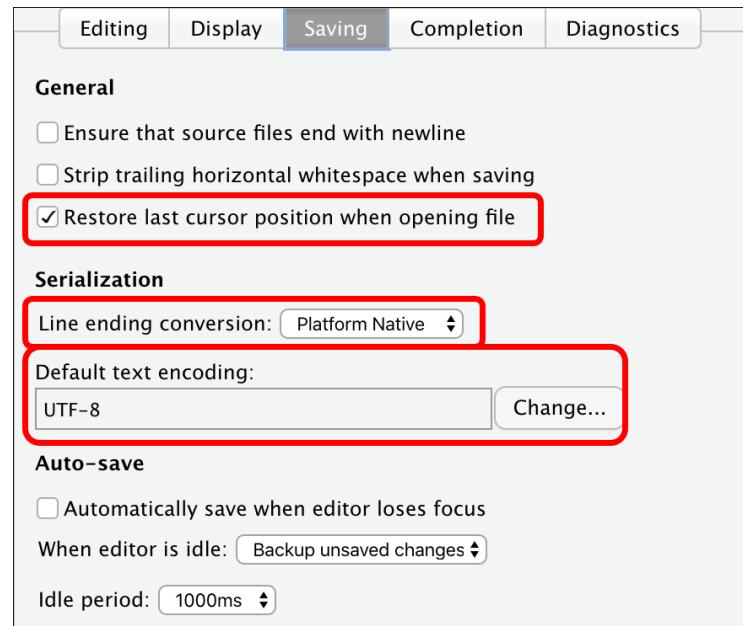
- Scrolling?

- Rainbow parentheses?



# Code Saving

- Cursor position?
- Line endings?
- Text encoding?

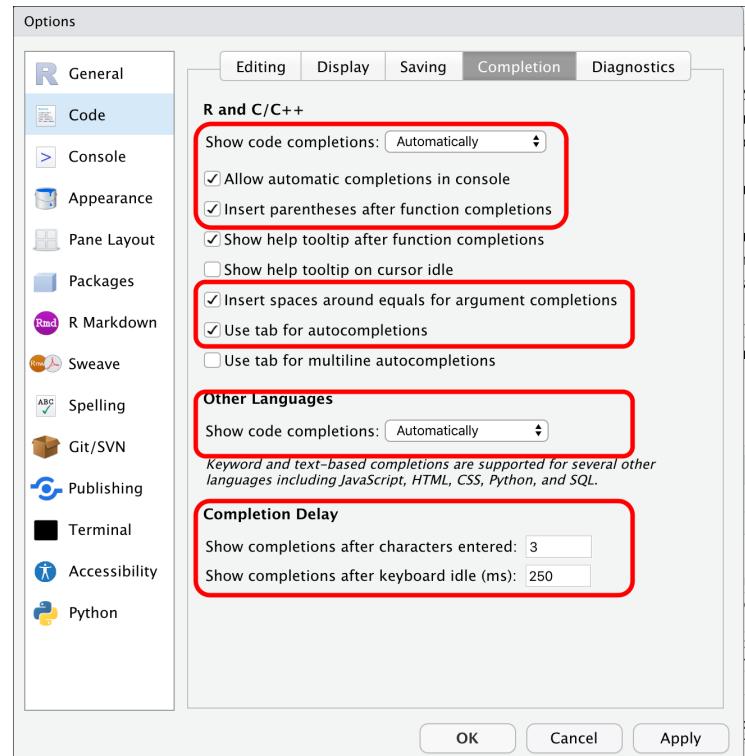


# Code Completion

- Insert parentheses?

- Insert spaces?

- Completion delay setting?

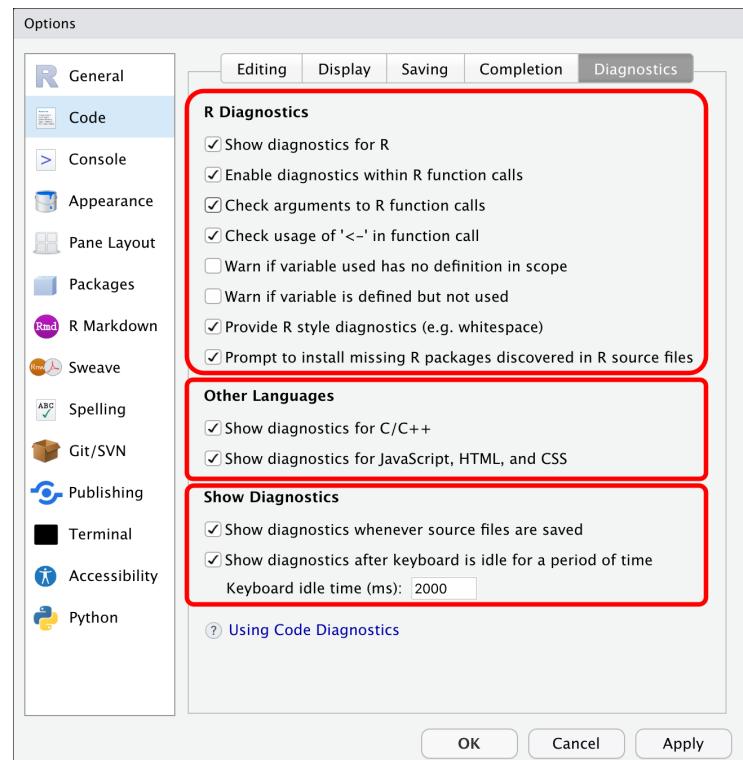


# Code Diagnostics

- Check your R Code?

- Check other languages?

- How long?

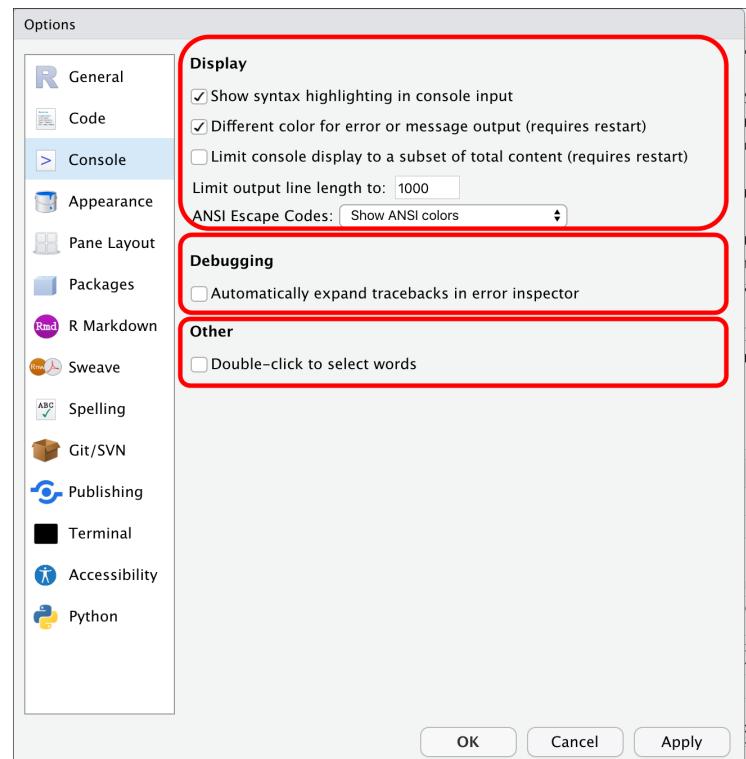


# Console

- Display?

- Debugging?

- Other?



# Appearance

- RStudio theme?

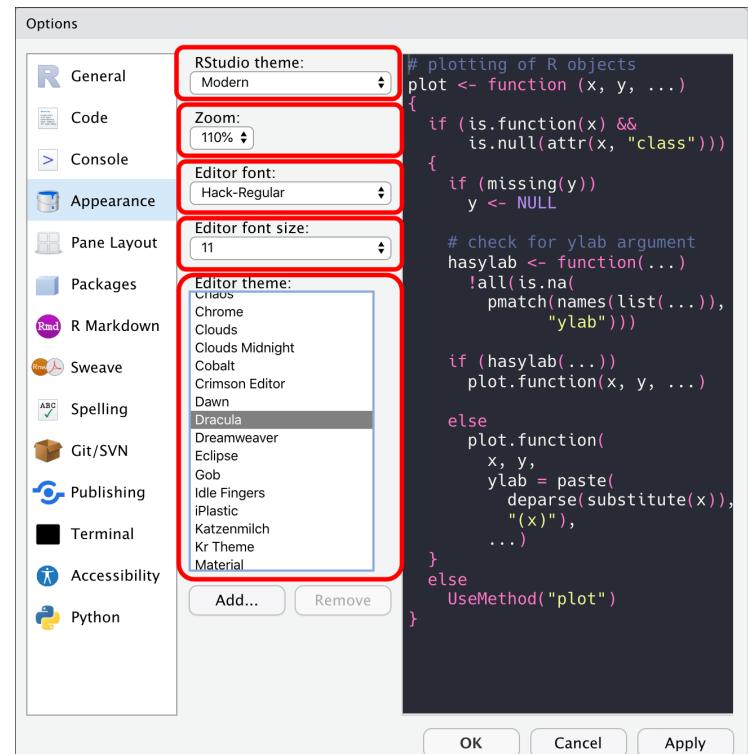
- Zoom?

- Also hold ⌘ and press + on macOS

- Also hold ctrl and press + on Windows

- Font?

- Editor theme?



# Pane layout

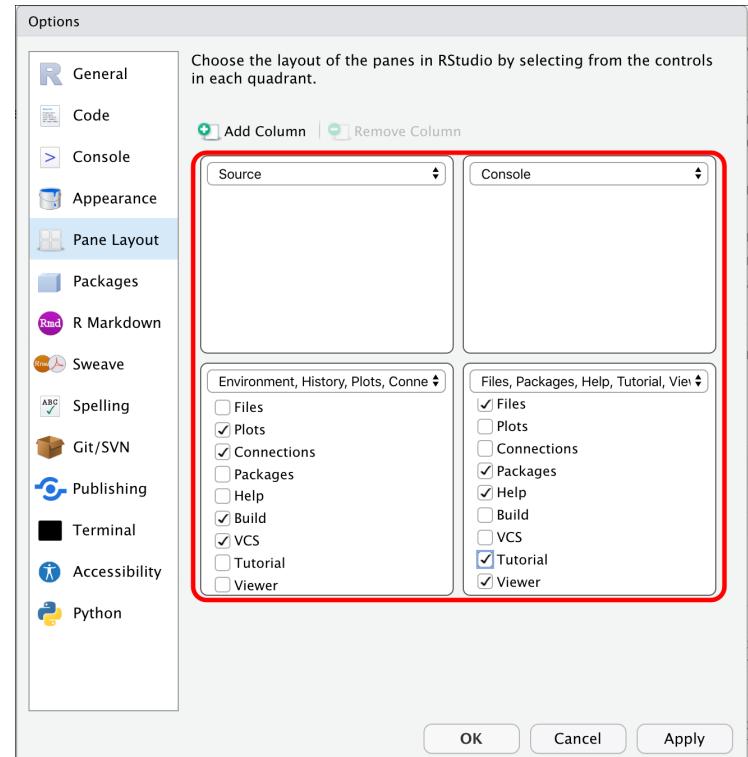
- Source?

- Console?

Combining pane elements?

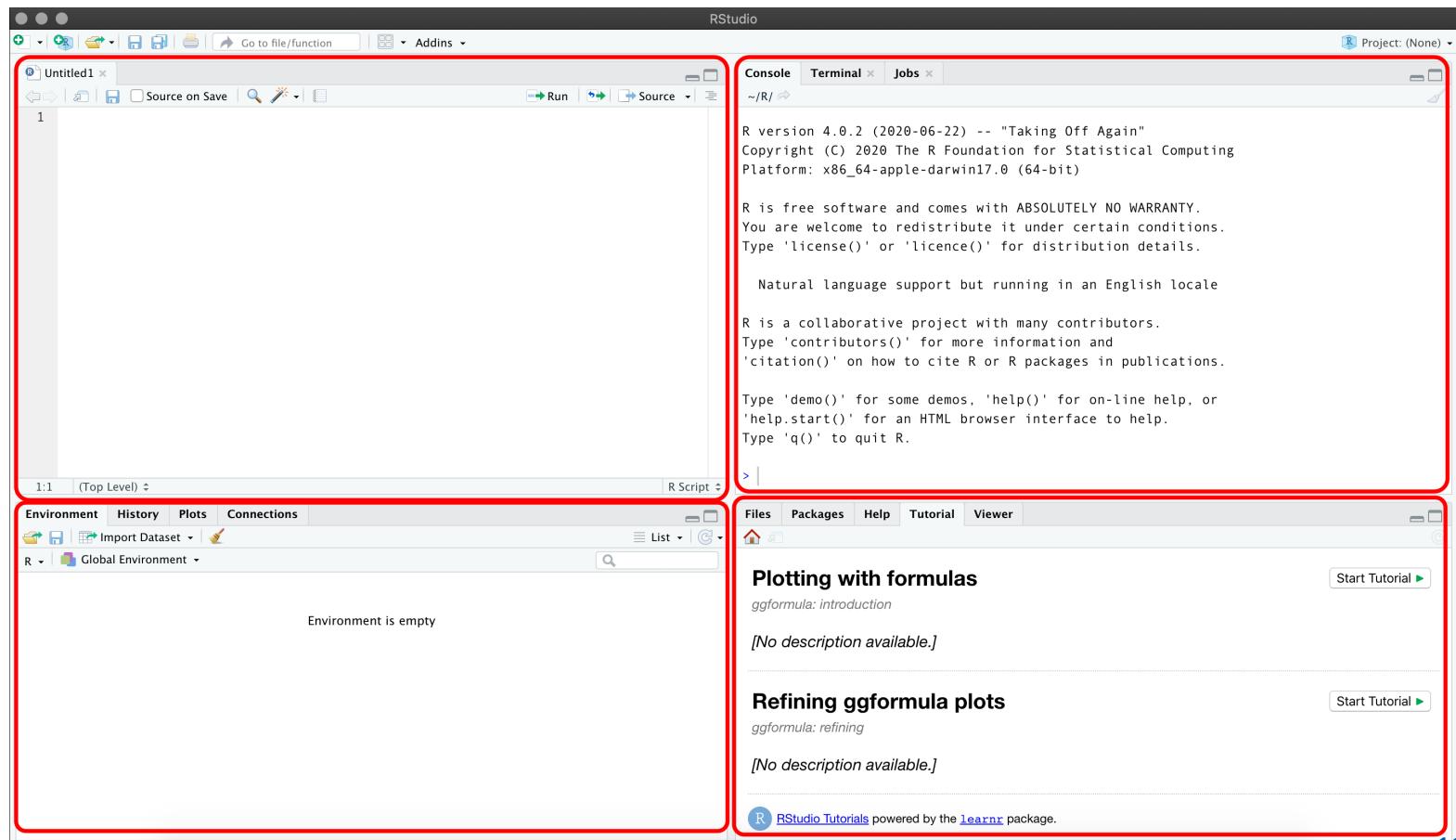
- Plots, Connections, Build, VCS, Presentation

- Files, Packages, Help, Tutorial, Viewer



# Pane layout view

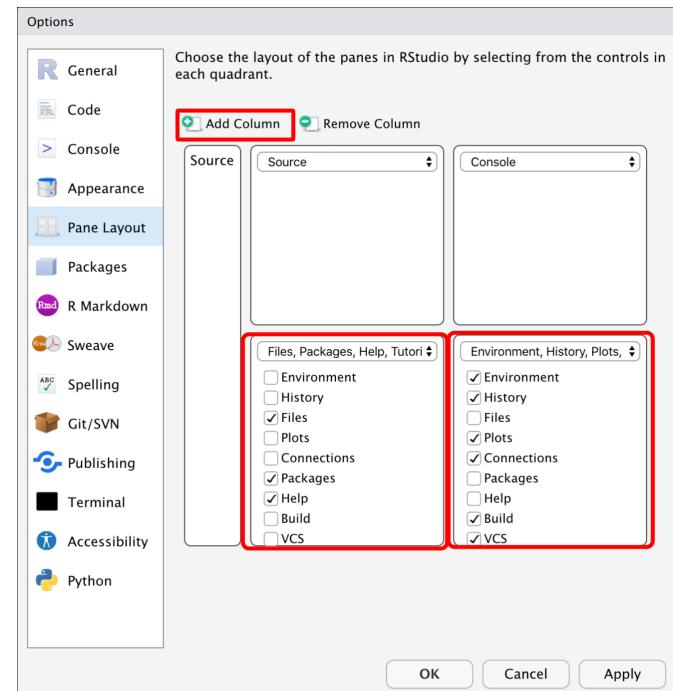
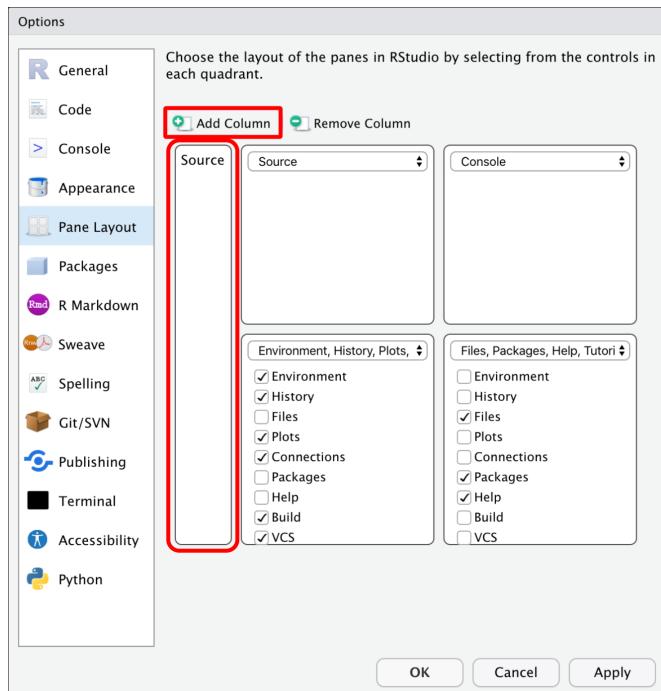
## Standard layout options



# Pane layout: add column

Two screens?

- add a Source column and rearrange the panes



# Pane layout: add column view

Now you see **Source**, **Tutorial**, and **Console** panes on a single screen!

The screenshot shows the RStudio interface with three main panes:

- Source pane:** Displays an R script titled "Untitled2" with code related to plotting birth data.
- Tutorial pane:** Shows a "Plotting with formulas" tutorial section about US Births in 1978, featuring a scatter plot of births over time and a "Continue" button.
- Console pane:** Displays the R environment information and a command-line interface with R's welcome message and help text.

# RStudio Projects

# Why RStudio Projects?

Keep track of all your files with RStudio project files (.Rproj).

## **Self contained**

Using R projects keeps track or your current working directory!

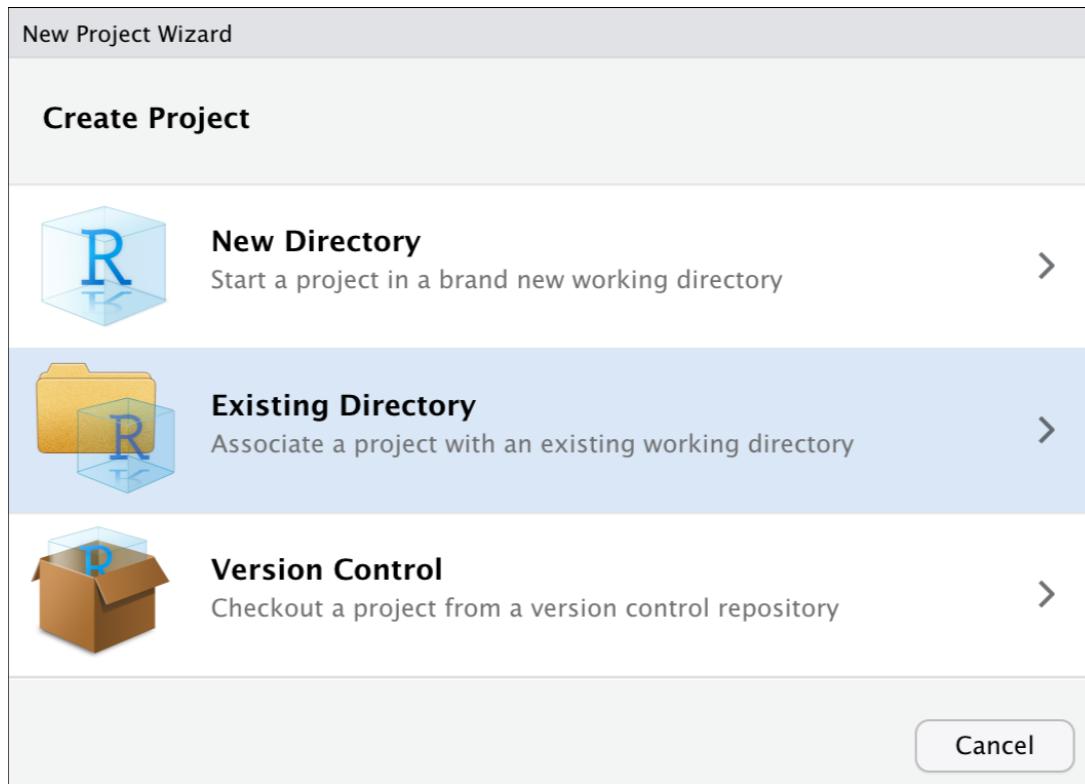
## **Project orientated**

.Rproj files make bundling and shipping files and folders easier!

## **Avoid removing all the files**

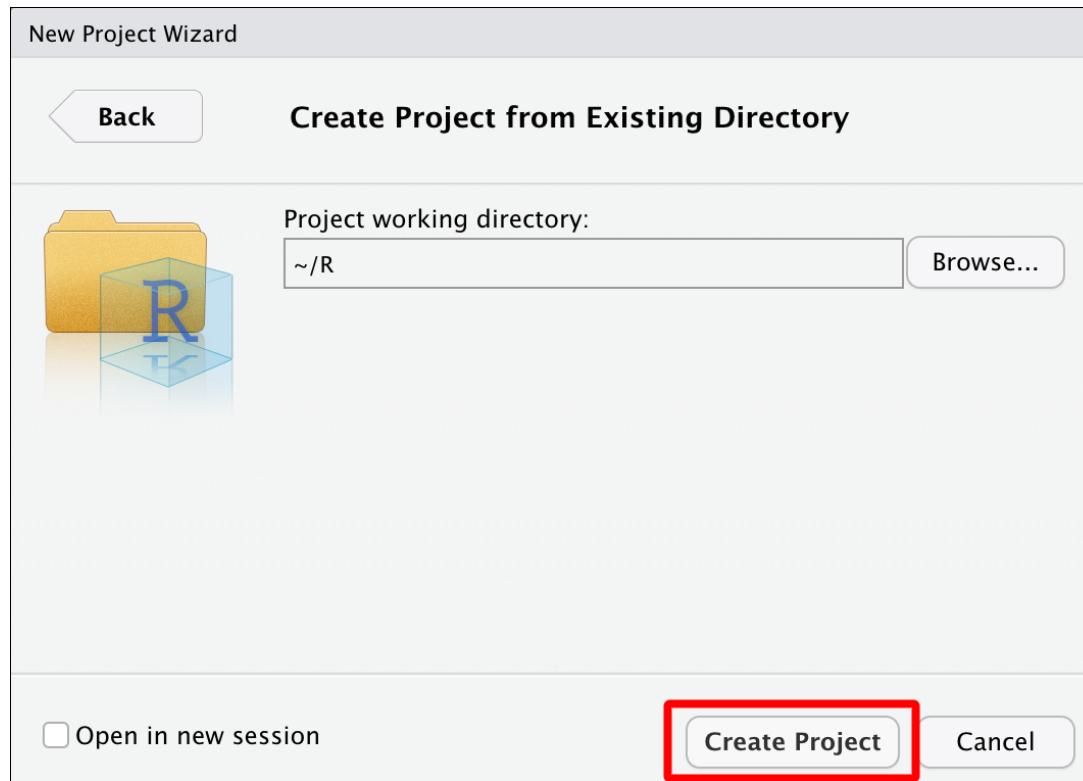
# Creating RStudio project in existing folder

Click on '*Project: (None)*' > '*New Project*'



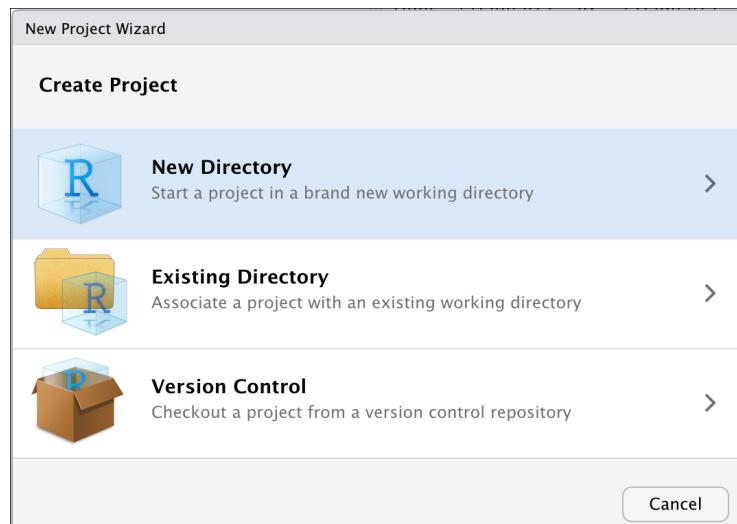
# Creating RStudio project in existing folder

Click on '*Browse* > '*Create Project*'

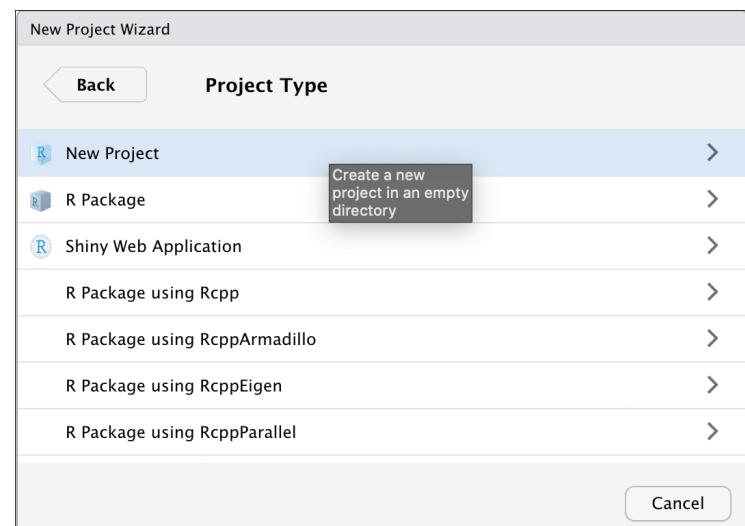


# Creating RStudio projects in new folder

Click on '*Project: (None)*' > '*New Project*'

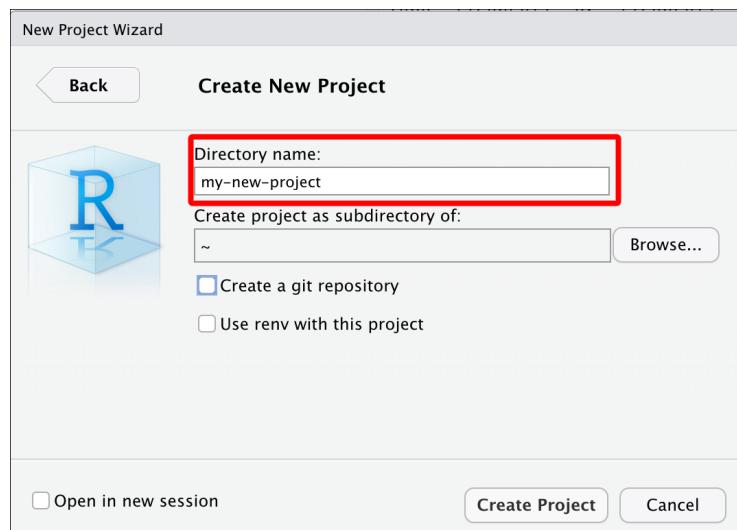


Select project type

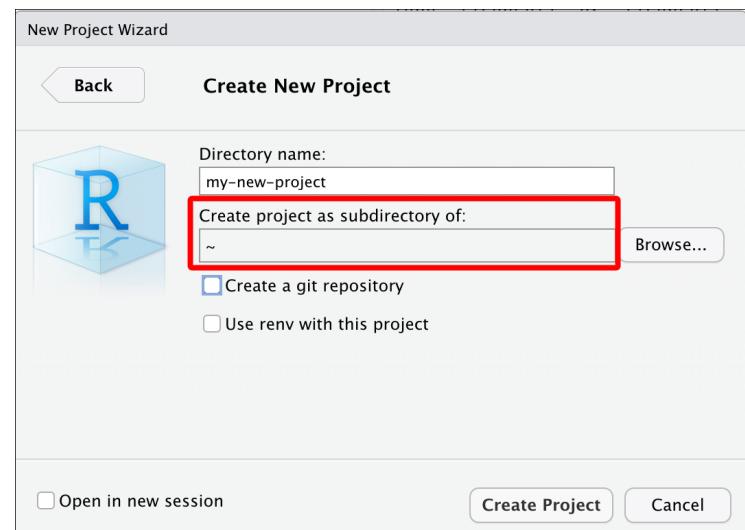


# Creating RStudio projects in new folder

Create new folder name

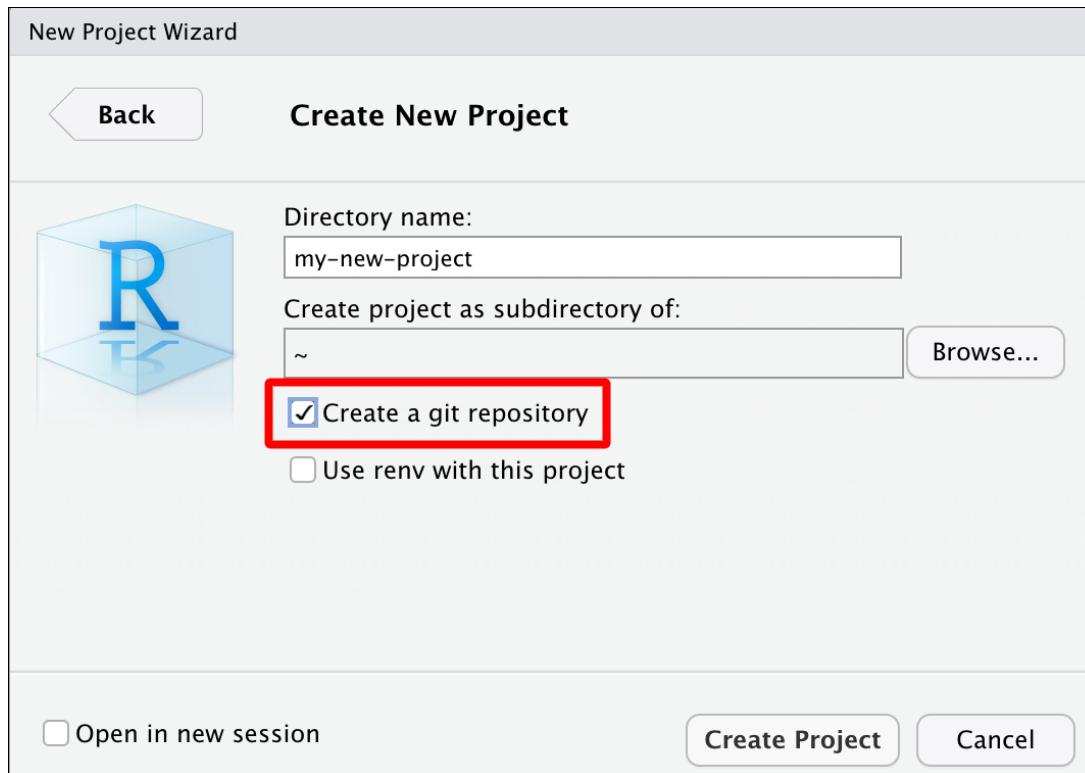


Choose parent folder



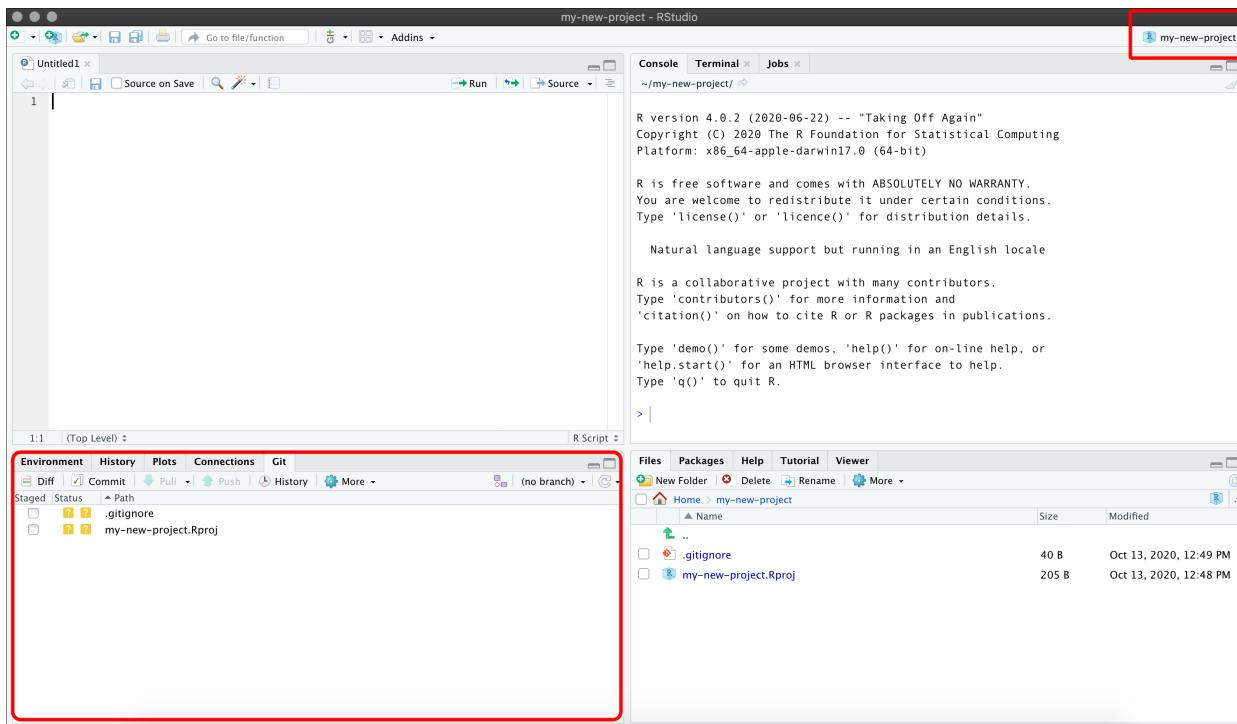
# Creating RStudio projects in new folder

If you have Git installed, select '*Create a git repository*'



# Creating RStudio projects in new folder

Check for new project name & *Git* pane



# Folder Structure

- separate raw and cleaned data
- keep documents and code separate
- keep figures separate
- name files appropriately (preferably 2 digit prefix)
- structure is reusable and easy to understand

## Adapted from from 'Good enough practices in scientific computing'

```
project-name/
    |-- CITATION
    |-- project-name.Rproj
    |-- README.md
    |-- LICENSE
    |-- requirements.txt
    |--data/
        |--raw/
            |--raw-birds-data.csv
        |--processed/
            |--processed-birds-data.csv
    |--doc/
        |-- notebook.Rmd
        |-- manuscript.Rmd
        |-- changelog.txt
    |-- results/
        |-- summarized-results.csv
    |-- code/
        |-- 01-sightings-import.R
        |-- 02-sightings-wrangle.R
        |-- 03-sightings-model.R
        |-- runall.R
```

# Naming things

# Naming files\*

## File names should be:

- |                                       |   |
|---------------------------------------|---|
| 1. human readable -> (makes sense)    | 2020-10-12-270-301-central-lab-metrics.csv          |
| 2. machine readable -> (regex)        | 2020-10-12- <b>270-301</b> -central-lab-metrics.csv |
| 3. sort/order well -> (ISO 8601 date) | <b>2020-10-12-270-301-central-lab-metrics.csv</b>   |

We can perform regular expression searches for files like this:

*Find 270-301 files*

```
grepl(pattern = "270-301",  
       x = "2020-10-12-270-301-central-lab-metrics.csv")
```

```
## [1] TRUE
```

| \*Adapted from Jenny Byran's slides

# Naming files\*

## Also acceptable:

Logical order and underscores \_

```
files
```

```
## [1] "01.0-import_270-301_central-lab-metrics.R"  
## [2] "02.0-wrangle_270-301_central-lab-metrics.R"  
## [3] "03.0-eda_270-301_central-lab-metrics.R"  
## [4] "04.0-model_270-301_central-lab-metrics.R"
```

```
stringr::str_split_fixed(string = files, pattern = "_", 3)
```

```
##      [,1]      [,2]      [,3]  
## [1,] "01.0-import"  "270-301"  "central-lab-metrics.R"  
## [2,] "02.0-wrangle"  "270-301"  "central-lab-metrics.R"  
## [3,] "03.0-eda"     "270-301"  "central-lab-metrics.R"  
## [4,] "04.0-model"   "270-301"  "central-lab-metrics.R"
```

\*Adapted from [Jenny Byran's slides](#)

# File paths

# Use relative rather than absolute file paths

Absolute paths are specific to a system

/project-name/data -> absolute path in macOS

\\\project-name\\data -> absolute path in Windows

Relative paths are specific to a folder

project-name/data -> relative path in macOS

project-name\\data -> relative path in Windows

# Or use the `here` package

The `here::set_here()` function solves a lot of file path problems (*especially if you're not using R projects*)

```
library(here)
here::set_here(".")
list.files(all.files = TRUE, pattern = "here")

## [1] ".here"
```

This creates a `.here` file (similar to `.Rproj` files)

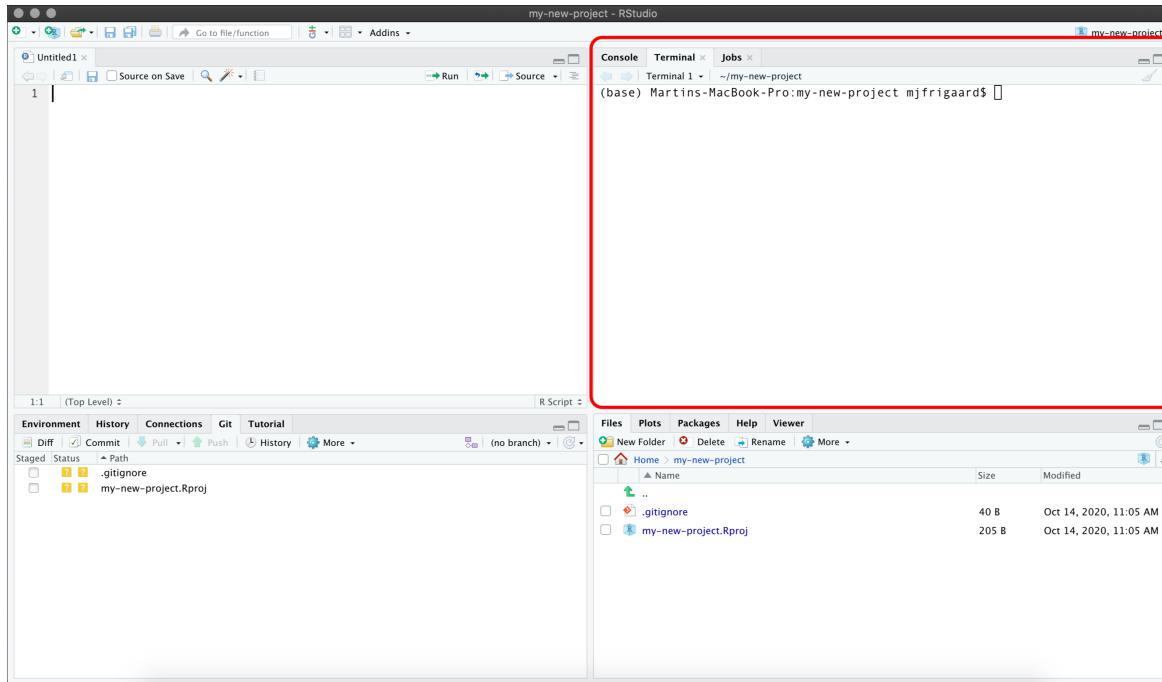
As long as the `.here` file stays in the referenced folder, you can include simply include `here::here()` in the top of your code files.

# Terminal pane

Learn a handful of command-line tools to make life easier

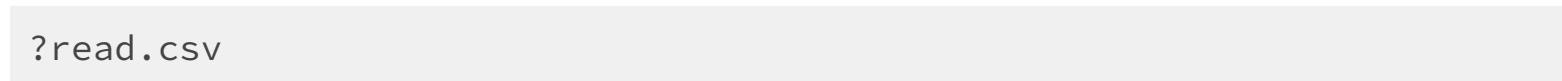
`cd`, `pwd`, `mkdir`, `rm`, `ls`, etc.

RStudio comes with a Terminal pane for quick access to the command-line



# Getting help

R comes with a *ton* of accessible help files



The screenshot shows the R Help Viewer interface. The title bar says "?read.csv". The menu bar includes Files, Plots, Packages, Help (which is selected), and Viewer. Below the menu is a toolbar with icons for back, forward, search, and refresh. The main content area shows the documentation for the `read.table` function. The title is "Data Input". The "Description" section states: "Reads a file in table format and creates a data frame from it, with cases corresponding to lines and variables to fields in the file." The "Usage" section contains two code examples:

```
read.table(file, header = FALSE, sep = "", quote = "\"\"",  
          dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),  
          row.names, col.names, as.is = !stringsAsFactors,  
          na.strings = "NA", colClasses = NA, nrow = -1,  
          skip = 0, check.names = TRUE, fill = !blank.lines.skip,  
          strip.white = FALSE, blank.lines.skip = TRUE,  
          comment.char = "#",  
          allowEscapes = FALSE, flush = FALSE,  
          stringsAsFactors = default.stringsAsFactors(),  
          fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)  
  
read.csv(file, header = TRUE, sep = ",", quote = "\"\"",  
        dec = ".", fill = TRUE, comment.char = "", ...)
```

# Getting help online

R also has an incredible community! Click on the links below to see some of the common places for Q & A.

- 1) Dedicated forum on RStudio Community**
- 2) Questions tagged R on StackOverflow**
- 3) Twitter topics with #rstats hashtag**

# Asking good questions (reproducible examples)

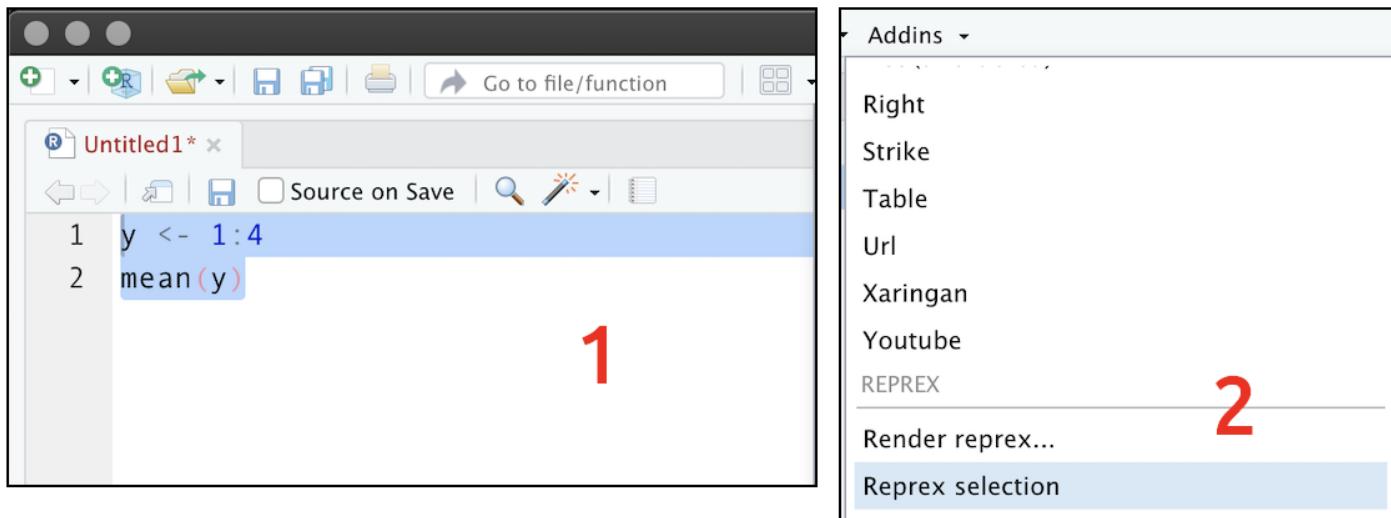
You'll get better results if you ask a question with a reproducible example. The `reprex` package was designed to help you create one!

```
install.packages("reprex")
library(reprex)
```

*Use the RStudio Addin to create a reproducible example from code you've copied onto your clipboard!*

# Reprex Addin 1

1. Copy code
2. Select Addin > *Render selection*



# Reprex Addin 2

1. Copy code
2. Select *Addin > Render selection*
3. Wait for console
4. Paste reprex

The screenshot shows the RStudio interface with the reprex addin installed. At the top, there are three tabs: 'Console', 'Terminal x', and 'Jobs x'. The 'Console' tab is active, showing the command '> |' and the output 'Rendering reprex...'. Below this, it says 'Rendered reprex is on the clipboard.' A large red number '3' is overlaid on the right side of the console area. At the bottom of the console window, it says 'Created on 2020-10-14 by the reprex package (v0.3.0)'. In the main workspace below, there is a code editor containing the following R code: 

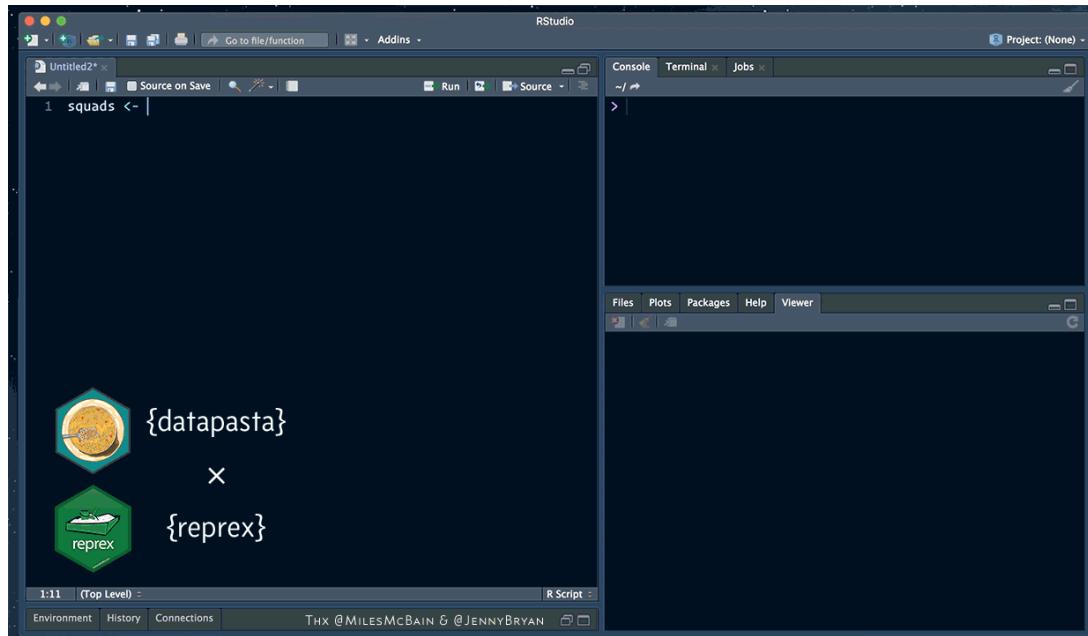
```
y <- 1:4  
mean(y)  
#> [1] 2.5
```

. A large red number '4' is overlaid on the right side of the code editor area. The RStudio menu bar at the top includes 'File', 'Plots', 'Packages', 'Help', 'Viewer', 'Zoom', 'Export', 'Publish', and 'C'.

# Reprex + datapasta

To copy + paste actual data in a reproducible example, try **datapasta**!

<https://reprex.tidyverse.org/articles/articles/datapasta-reprex.html>



Learn more about R best practices:

1. [R for Data Science](#)
2. [Tidyverse](#)
3. [RViews Community Blog](#)

# THANK YOU!

## Feedback

@mjfrigaard on Twitter and Github

[mjfrigaard@gmail.com](mailto:mjfrigaard@gmail.com)