Getting Started With R in RStudio

Martin Frigaard & Peter Spangler

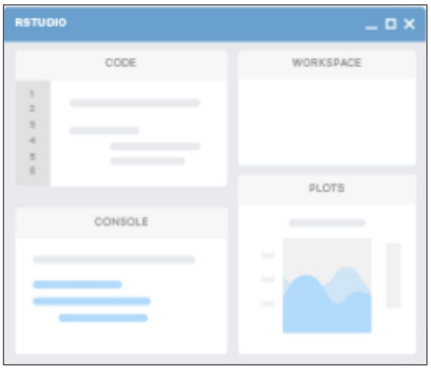
icons by https://www.freepik.com/

Objectives

- Why are you here?
- RStudio cloud
- RStudio IDE
- Learn basic markdown formatting
- Questions

What is RStudio?

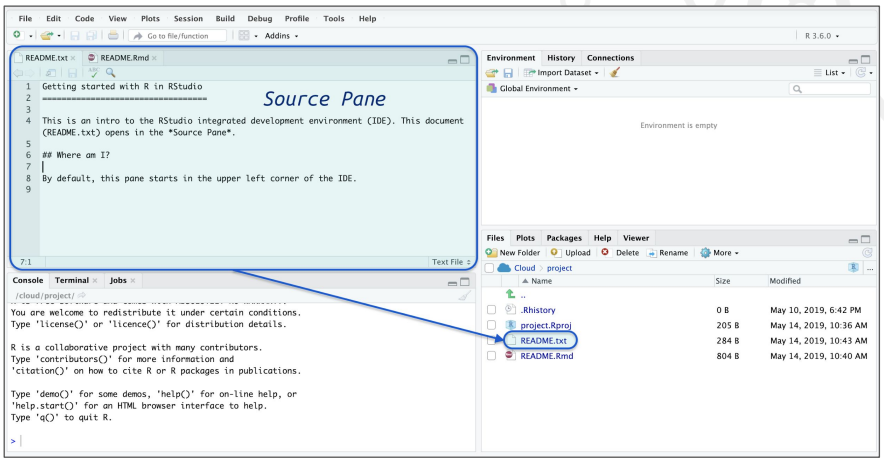
- Integrated development environment (IDE)
  - Write and test code
  - Display output and results
  - Build applications/websites
- RStudio is an IDE for R
  - Desktop application (free)
  - Open source server (free)
  - RStudio.cloud (free)
  - RStudio Pro Server (paid)



RStudio Source Pane

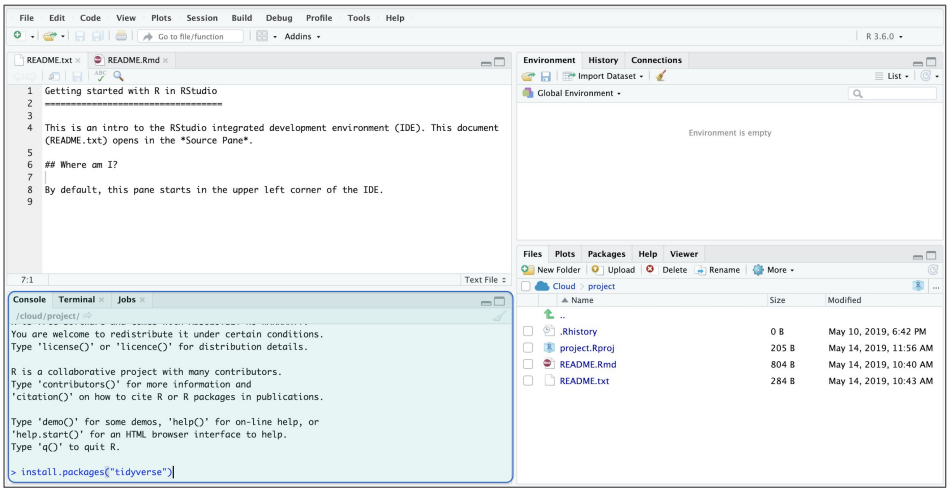
The **RStudio Source Pane** is where most of your work is done

Code and plain text files (.R, .txt, .Rmd, .md) open in the source pane.



In RStudio cloud, this will be in the upper left corner of the IDE.

RStudio Console Pane



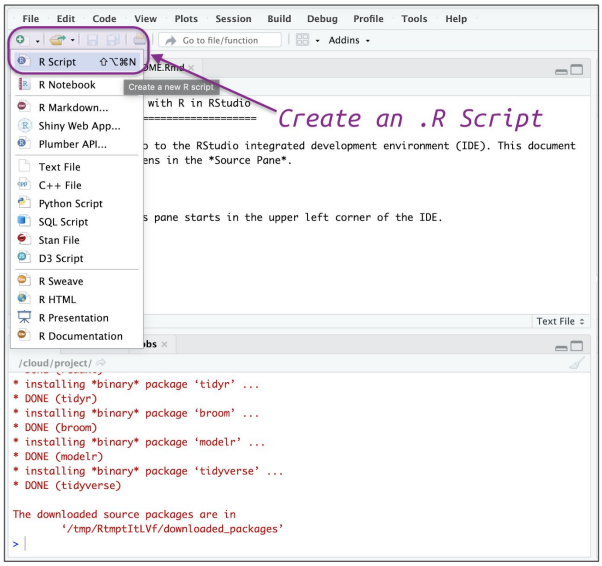
The **RStudio Console Pane** displays output and results.

Commands can also be entered directly into the Console.

This is the Console Pane. Enter the following command into the Console Pane, "install.packages("tidyverse")"

This will display the output from installing a package.

R Scripts in RStudio



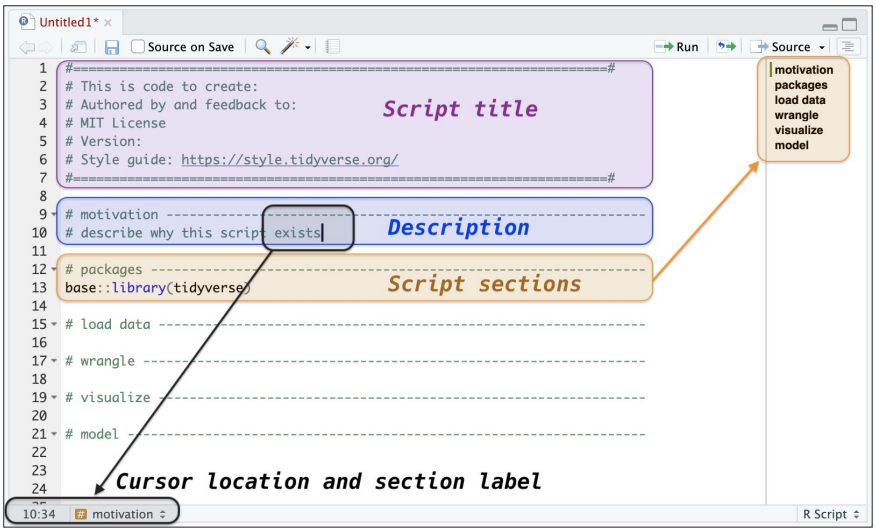
Open a new **.R** Script by clicking on the green + icon and Selecting "**.R Script**"

You can also use **File** > **New File** > **R Script**

Create a new R script.

These are used for running R commands. The file ending ".R" tells the computer what application should be used to run each script.

Code files in R



Use **cmd+r** or **ctrl+r** to create a new section heading in RStudio

Use **cmd+shift+c** to create comments on a particular line

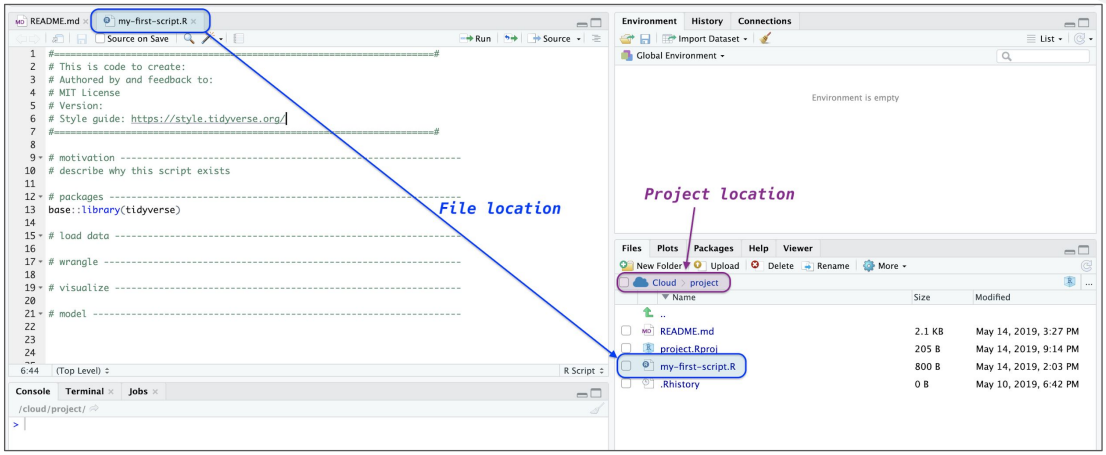
- Load all packages at the beginning of the script so there aren't hidden dependencies.

- include relevant information in the script header about the project, the reasoning this script exists, and the author

- Save this file as "my-first-script.R"

The RStudio Files Pane

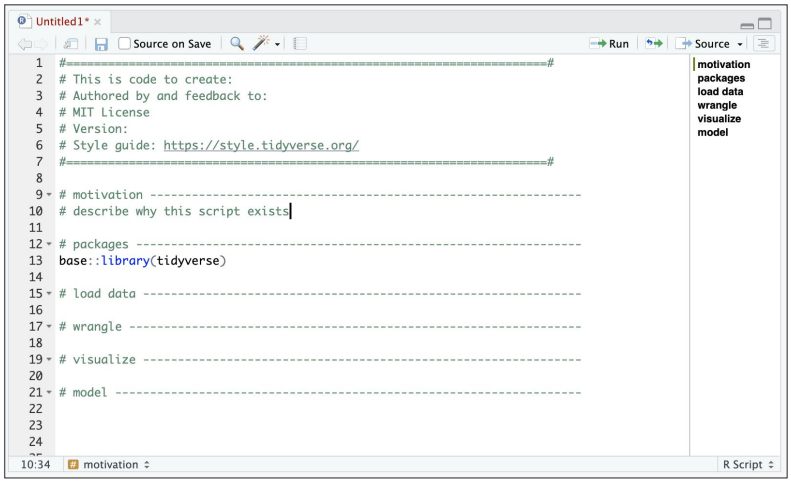
The **Files Pane** displays the current location of the RStudio project session, and associated files and folders.



- This project lives on the cloud, so the location is **Cloud/project**

- Create new folders for project files (**Data**, **Code**, **Figs**, **Meta**, **Text**, etc.)

Writing code in R



Writing clean, clear code makes it easier for everyone to read (especially you, six months from now).

Stick to some basic rules to make your file contents more consistent and organized.

- In code, use comments to explain the "**why**" not the "**what**" or "**how**" - https://style.tidyverse.org/

- Use comments to structure your .R files.

- Stick to < 80 columns for easier reading

The R Syntax

"R is a scripting language for statistical data manipulation and analysis." - The Art of R Programming

Generally speaking, the R language can be divided into **packages**, **functions** and **objects**

- **Packages** contain functions and object and get loaded into R
  *you just did this with **install.packages()** and **library()***

- **Functions** are like verbs

- **Objects** are like nouns

- ***Functions do things to objects***

```
package::function(object)
```

- You'll often hear the terms 'object oriented programming' or 'functional programming'. Both of these apply to R.

- if you call the **lm()** function in R, the **function** returns an **object** containing all the results—the estimated coefficients, their standard errors, residuals, and so on. You can then run additional **functions** on this **object** to extract what you need.

Packages, functions, and objects

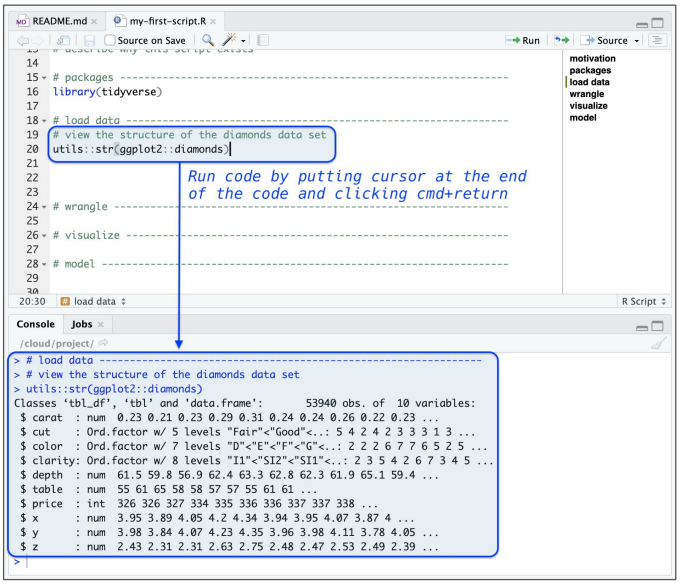
Below I use the **str()** function from the **utils** package to view the **diamonds** object

from the **ggplot2** package using the **package::function(package::object)** format

```
# load data ----------------------------------------------------
# view the structure of the diamonds data set
utils::str(ggplot2::diamonds)
```

- In the

Packages, functions, and objects



The results of the function are displayed in the **Console Pane**.

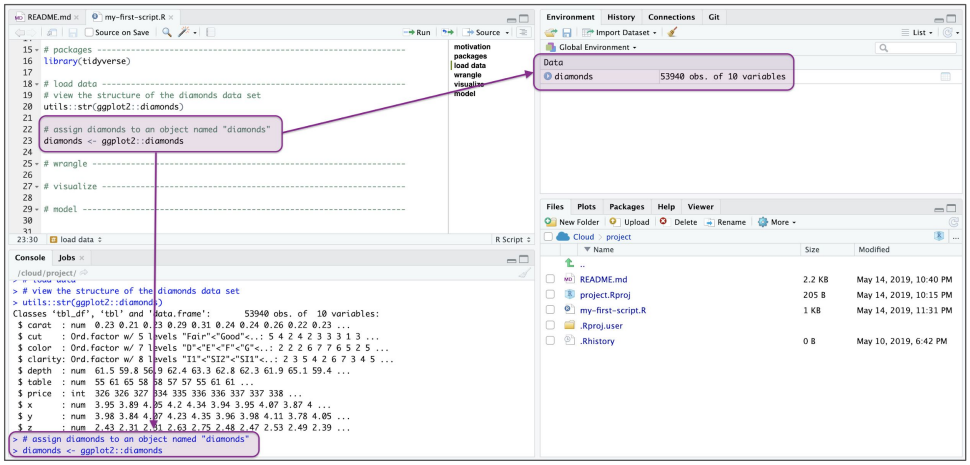
We can see the **diamonds** object is a **tbl_df**, **tbl**, and **data.frame** with **53940 observations** and **10 variables**

- In the my-first-script.R file, enter the following:

```
# view the structure of the diamonds data set
utils::str(ggplot2::diamonds)
```

Store and explore part 1

A common practice and workflow in R is to 1) use a **function** on an **object**, 2) store the results in a new **object**, and 3) use **functions** to explore this new results **object**

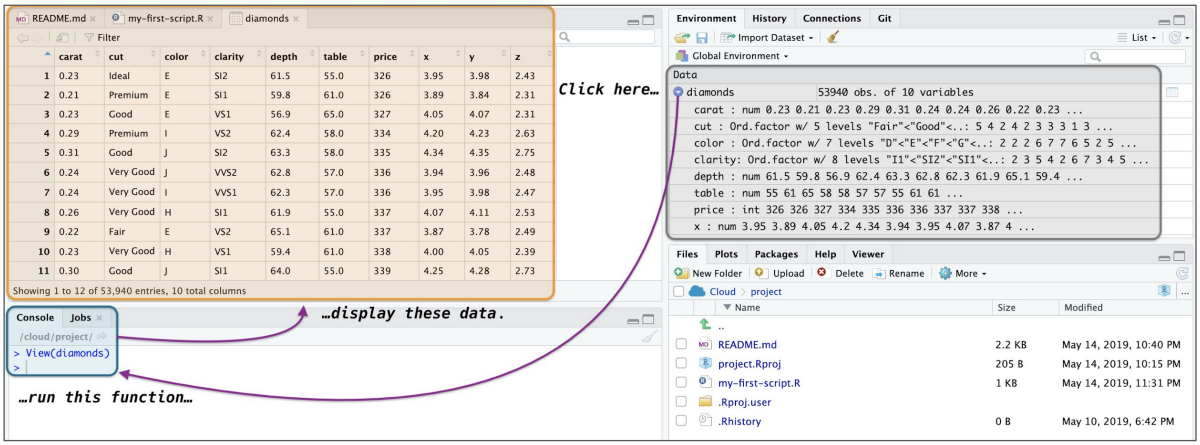


Store the **diamonds** data.frame into an object named **diamonds** using the assignment operator (**<-**)

After running this code, you will see the **diamonds** data.frame populate in the **Environment Pane**.

Store and Explore: the Environment Pane

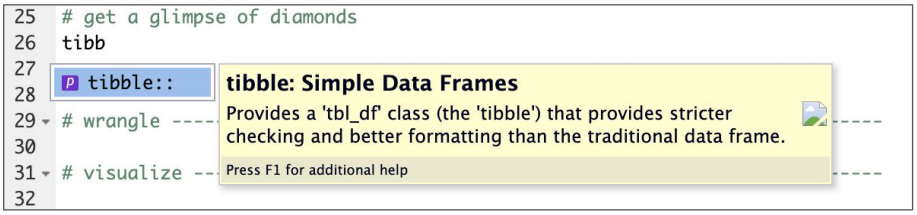
The **Environment Pane** is where objects are displayed. When the diamonds data.frame is assigned to the **diamonds** object, it shows up in the **Environment Pane**. Click on this object displays it in the **data viewer**.



We can see the new **diamonds data.frame** in the data viewer.

Store and explore: tab completion

Now that we have an object in the **Environment Pane**, we can explore it with functions from some of the package functions in the **tidyverse**.



Typing the first few letters of the package (i.e. "**tibb**") will bring up the help information.

- This is the **package** loaded in the **library** call.

Store and explore: function help

Hitting return will fill in **tibble::**

Tab completion saves time! (and keystrokes)

```
25  # get a glimpse of diamonds
26  tibble::g
27
28
29  # wrangle ----------------------
30
31  # visualize --------------------
32
33  # model ------------------------
34
35
36
```

glimpse {tibble}

```
glimpse(x, width = NULL, ...)
```

Maturing lifecycle

This is like a transposed version of `print()`: columns run down the page, and data runs across. This makes it possible to see every column in a data frame. It's a little like `str()` applied to a data frame but it tries to show you as much data as possible. (And it always shows the underlying data, even when applied to a remote data source.)

Press F1 for additional help

We will use the **glimpse()** function from the **tibble** package.

Now we can see all the functions in the **tibble** package. The additional help information is provided for each function (in yellow). We learn the function is similar to **utils::str()**

You can use the arrow keys to scroll through each function and object in the package.

Store and explore: function arguments

Hitting return will now give us **tibble::glimpse(|)** with "|" being our cursor position

Tab completion: hitting tab will now bring up the arguments for this function

```
25  # get a glimpse of diamonds
26  tibble::glimpse()
27
28
29  # wrangle -----
30
31  # visualize ---
32
33  # model ----------------------------------------------------------------
```

```
x =
width =
... =
diamonds
```

```
x
An object to glimpse at.
Press F1 for additional help
```

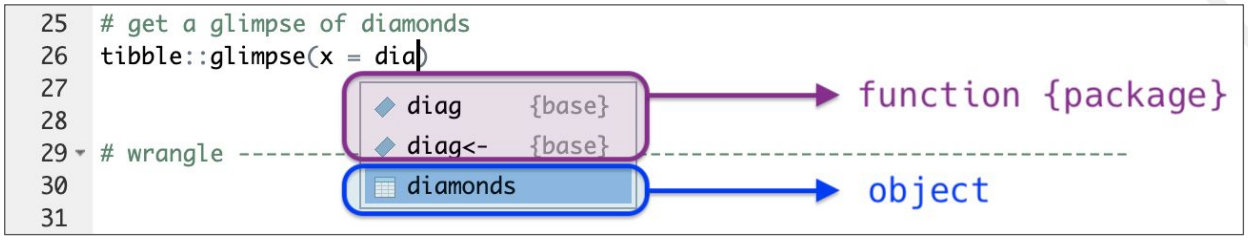
Functions contain a set of instructions, take inputs (objects and arguments), and return a result.

- You can use the arrow keys to scroll through each function and object in the package.

- "A function is a group of instructions that takes inputs, uses them to compute other values, and returns a result." - Art of R Programming

Store and explore: environment objects

The objects in the environment are also visible with tab completion.



Objects in the Environment do not have a package listed next to them.

- These include any local data frames, functions, or vectors

- "A function is a group of instructions that takes inputs, uses them to compute other values, and returns a result." - Art of R Programming

Recap

- RStudio is an IDE with for working and programming with R
- Panes we covered:
  - the **Source** editor (for writing code)
  - the **Console** (where commands are run and results are shown)
  - **Files** in our project folder
  - the **Environment** that displays objects
- Other panes include project management (**Git**), previous commands (**History**), documentation and help files (**Help**), database access (**Connections**), background processes (**Jobs**), and build reviewing windows (**Viewer** and **Plots**).

Questions?

Additional Resources

1. DataSciCom
2. Storybench
3. RStudio
4. Tidyverse