

Week 1_Day 2

Introduction to R and RStudio

Module 1_extended_R project

Learning Objectives

- Describe what R and RStudio are.
- Interact with R using RStudio.
- Use the various components of RStudio.

What is R?

The common misconception is that R is a programming language but in fact it is much more than that. R is not a programming language like C, C++, Python and Java. It is a statistical software. Think of R as an environment for statistical computing and graphics, which brings together a number of features to provide powerful functionality.

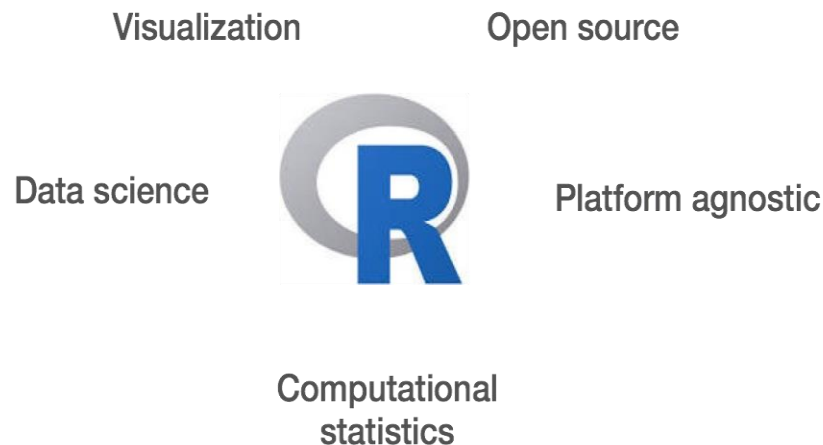
A Brief History of R

R has been around since 1995 and was created by **Ross Ihaka** and **Robert Gentleman** at the University of Auckland, New Zealand. R is based off the S programming language developed at Bell Labs and was developed to teach intro statistics.

The R environment combines:

- effective handling of big data
- collection of integrated tools
- graphical facilities
- simple and effective programming language

Why use R?



- R is a powerful, extensible environment. It has a wide range of statistics and general data analysis and visualization capabilities.

- Data handling, wrangling, and storage
- Wide array of statistical methods and graphical techniques available
- Easy to install on any platform and use (and it's free!)
- Open source with a large and growing community of peers

What is RStudio? (From October 2022, the name will be Posit)

RStudio is freely available open-source **Integrated Development Environment (IDE)**. RStudio provides an environment with many features to make using R easier and is a great alternative to working on R in the terminal.



- Graphical user interface (GUI), not just a command prompt
- Great learning tool
- Free for academic use
- Open source

The first look at R console: : R's Heart

Interactive data analysis usually occurs on the R console that executes commands as you type them. There are several ways to gain access to an R console. One way is to simply start R on your computer. The console looks something like this:

A screenshot of the R Console window within the RGui (32-bit) application. The window title is 'R Console'. The menu bar includes 'File', 'Edit', 'View', 'Misc', 'Packages', 'Windows', and 'Help'. The toolbar shows icons for file operations and execution. The console text displays the R version (3.4.4), copyright information (© 2018 The R Foundation for Statistical Computing), platform details (i386-w64-mingw32/i386 (32-bit)), and various help instructions. The prompt '> |' is visible at the bottom of the console.

```
R version 3.4.4 (2018-03-15) -- "Someone to Lean On"
Copyright (C) 2018 The R Foundation for Statistical Computing
Platform: i386-w64-mingw32/i386 (32-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it 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.

> |
```

As a quick example, try using the console to calculate a multiplication of 15 times 5:

```
15 * 5  
#> [1] 75
```

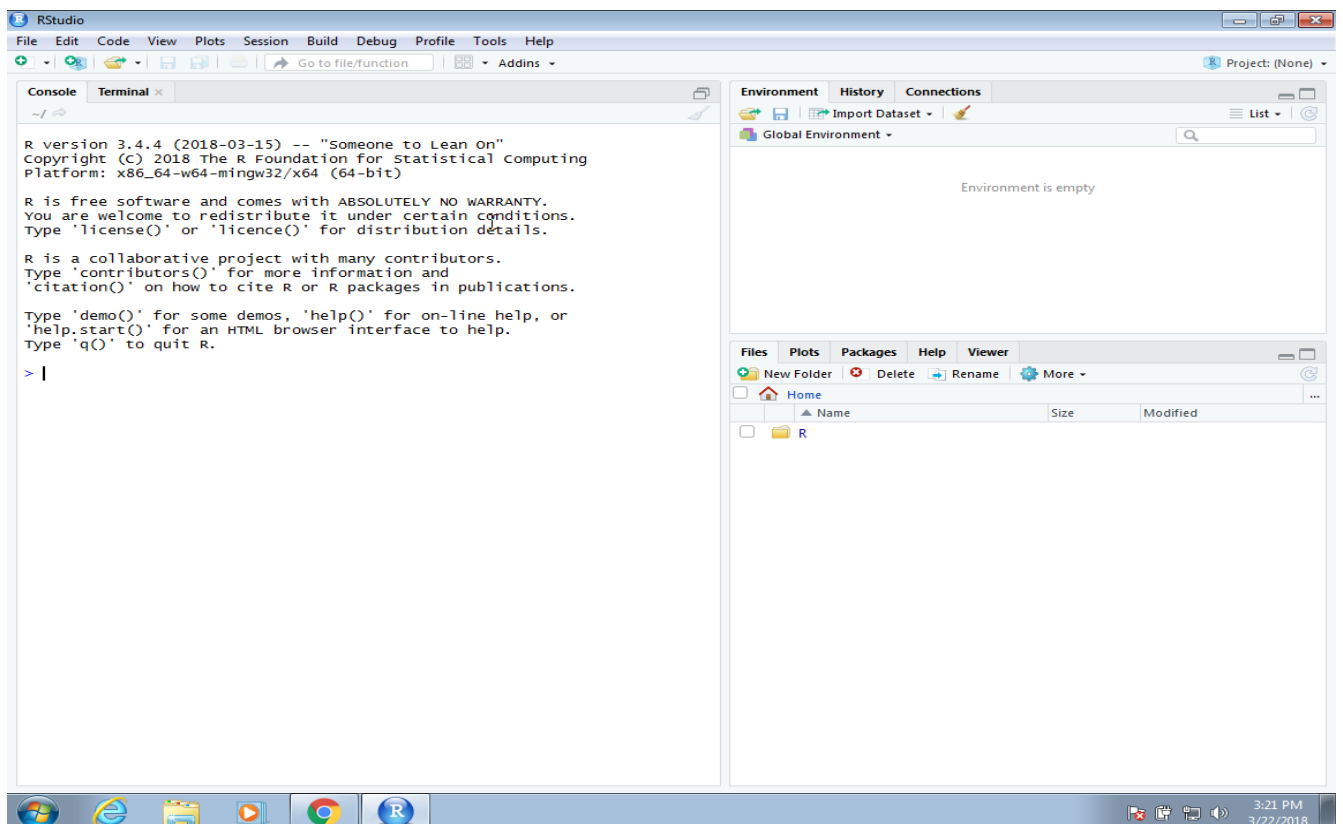
Note that in this note, purple color and boxes are used to show R code typed into the R console. The symbol #> is used to denote what the R console outputs.

Getting Started with RStudio

RStudio is an open-source tool for programming in R. RStudio is a flexible tool that helps you create readable analyses, and keeps your code, images, comments, and plots together in one place. It's worth knowing about the capabilities of RStudio for data analysis and programming in R.

First Look at RStudio

When we open R-Studio for the first time, we'll probably see a layout like this:



The panes

When you start RStudio for the first time, you will see three panes like the figure above. The left pane shows the R console. On the right, the top pane includes tabs such as Environment and History, while the bottom pane shows five tabs: File, Plots, Packages, Help, and Viewer (these tabs may change in new versions or your own preferred settings). You can click on each tab to move across the different features.

When we open RStudio, R is launched as well. A common mistake by new users is to open R instead of RStudio. To open RStudio, search for RStudio on the desktop, and pin the RStudio icon to the preferred location (e.g. Desktop or toolbar).

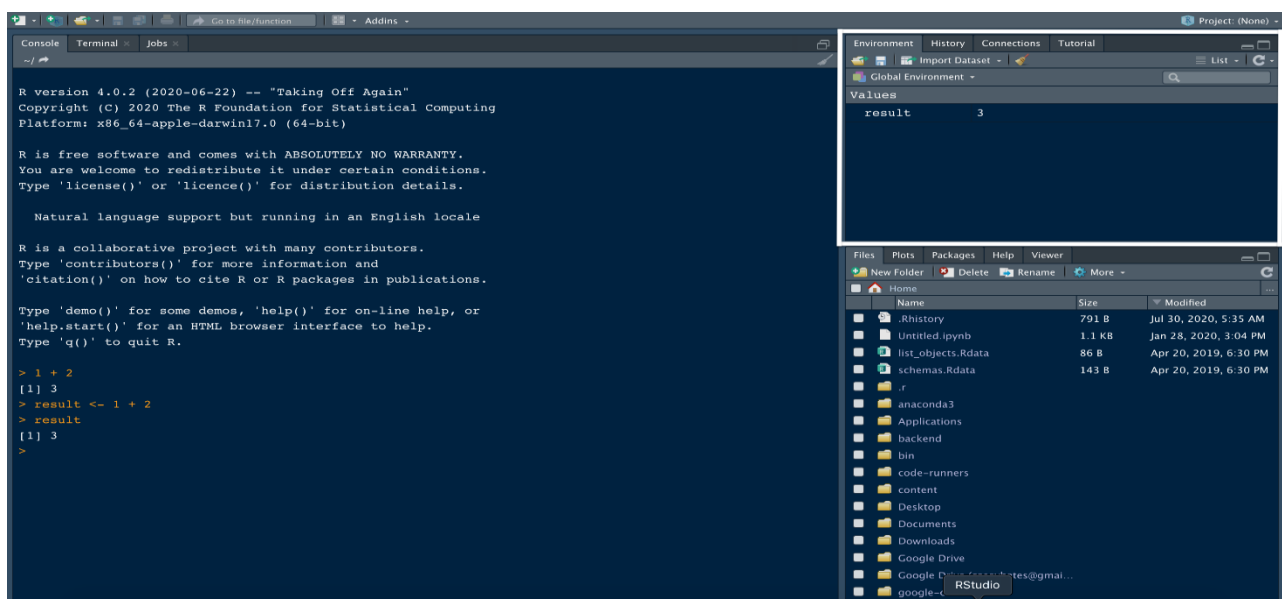
The Console

Let's start off by introducing some features of the Console. The Console is a tab in RStudio where we can run R code. Notice that the window pane where the console is located contains three tabs: Console, Terminal and Background Jobs (this may vary depending on the version of RStudio in use). We'll focus on the Console for now.

When we open RStudio, the console contains information about the version of R we're working with. You may clean this welcoming text by typing **ctrl+I**. Scroll down and try typing a few expressions like this one. Press the enter key to see the result. we can use the console to test code immediately. When we type an expression like **1 + 2**, we'll see the output below after hitting the enter key.

The Global Environment

We can think of the global environment as our workspace. During a programming session in R, any variables we define, or data we import and save in a dataframe, are stored in our global environment. In RStudio, we can see the objects in our global environment in the Environment tab at the top right of the interface:

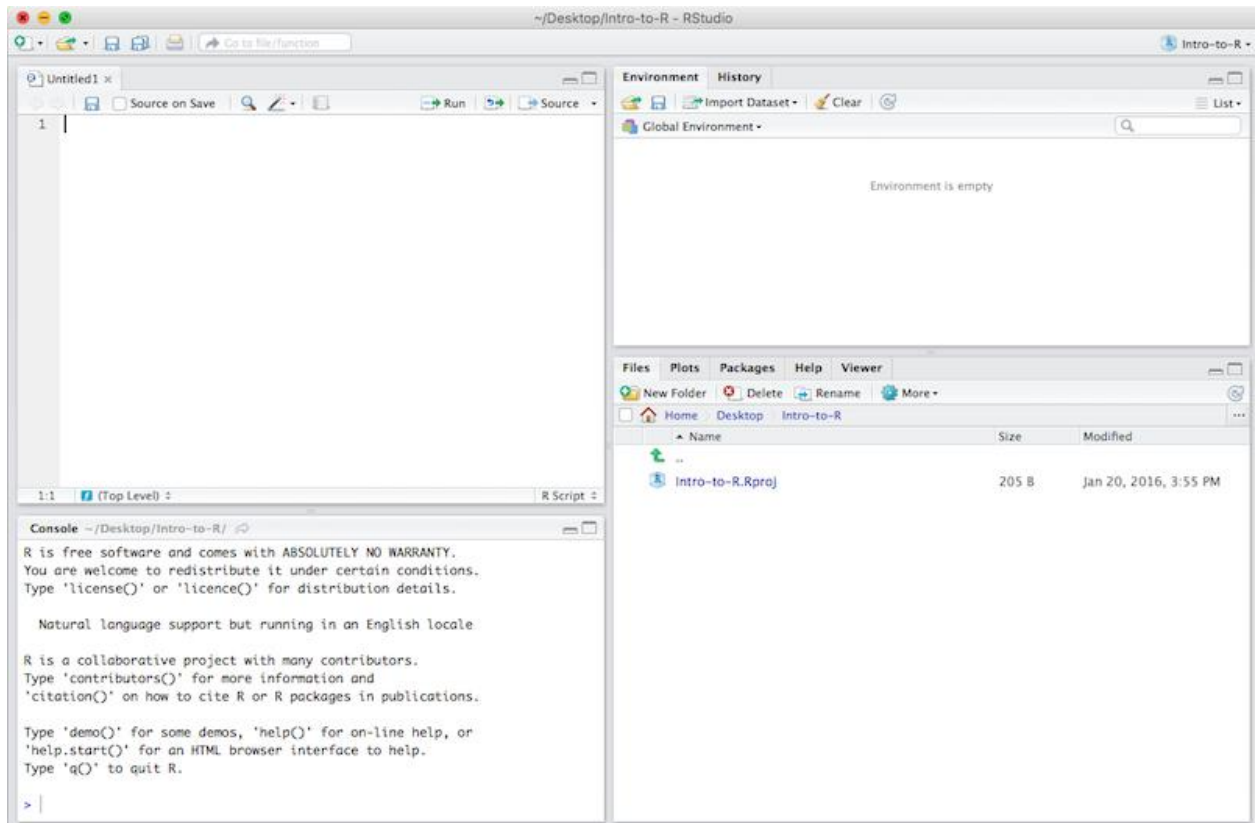


Sometimes, having too many named objects in the global environment creates confusion. Maybe we'd like to remove all or some of the objects. To remove all objects, click the broom icon at the top of the window.

Creating a new project directory in RStudio

Let's create a new project directory for our day_2 Lab lesson today.

- Open RStudio
- Go to the File menu and select New Project.
- In the New Project window, choose New Directory. Then, choose New Project. Name your new directory my_day2 and then "Create the project as subdirectory of:" the Desktop (or location of your choice).
- Click on Create Project.
- After your project is completed, if the project does not automatically open in RStudio, then go to the File menu, select Open Project, and choose my_day2.Rproj.
- When RStudio opens, you will see three panels in the window.
- Go to the File menu and select New File, and select R Script. The RStudio interface should now look like the screenshot below.



RStudio Interface

The RStudio interface has four main panels:

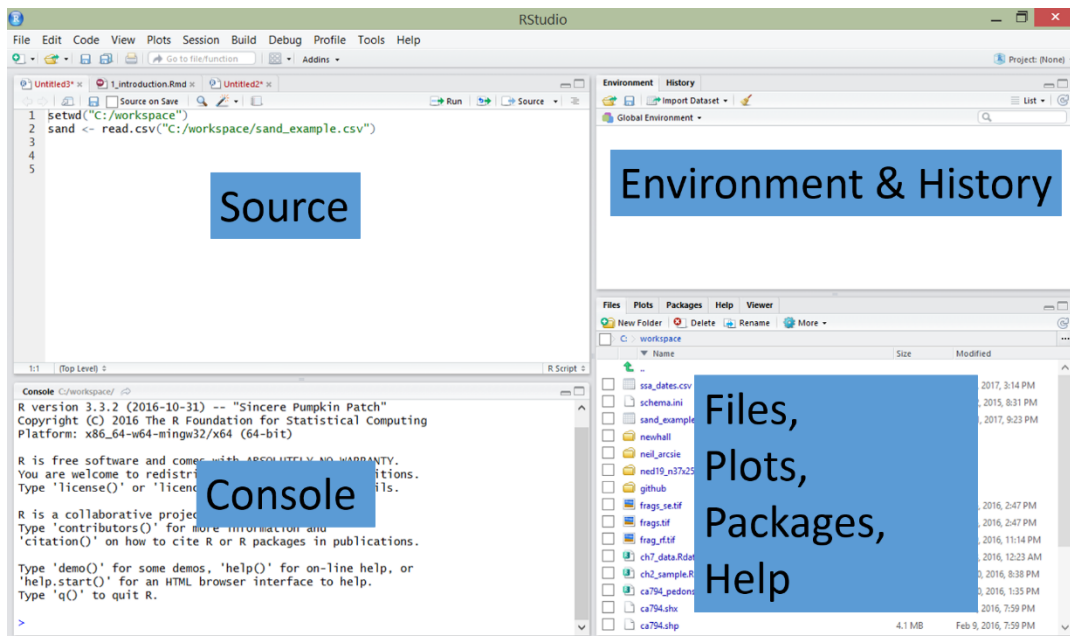
Console: where you can type commands and see output. The console is all you would see if you ran R in the command line without RStudio.

Script editor/Source: where you can type out commands and save to file. You can also submit the commands to run in the console.

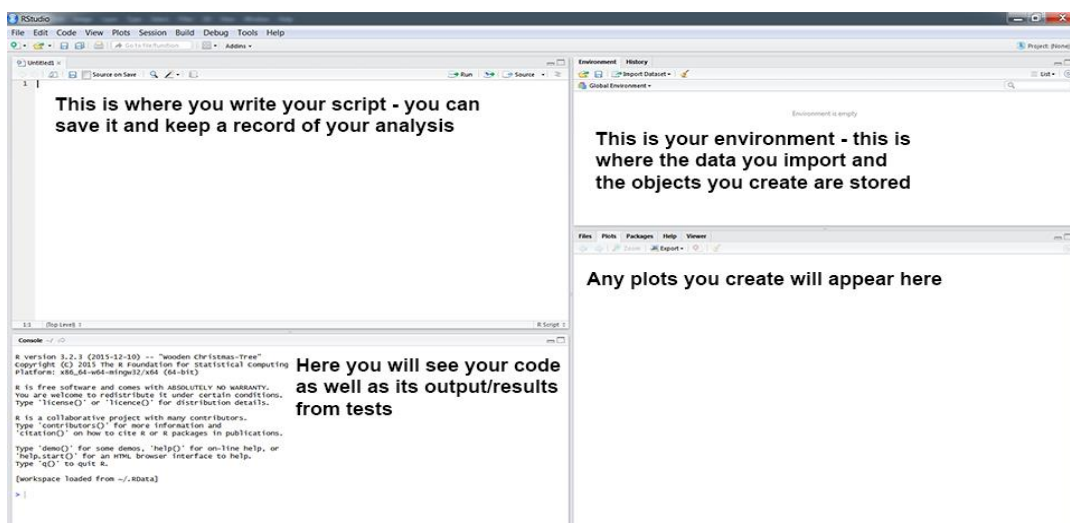
Environment/History: environment shows all active objects and history keeps track of all commands run in console

Files/Plots/Packages/Help

Once we create an R-script and that will bring us to the following 4 panes:



The Files / Plots / Packages / Help panel shows you lots of helpful information. Let's go through each tab in detail:



Files - The files panel gives you access to the file directory on your hard drive. One nice feature of the “Files” panel is that you can use it to set your working directory - once you navigate to a folder you want to read and save files to, click “More” and then “Set As Working Directory.” We’ll talk about working directories in more detail soon.

Plots - The Plots panel (no big surprise), shows all your plots. There are buttons for opening the plot in a separate window and exporting the plot as a pdf or jpeg (though you can also do this with code using the `pdf()` or `jpeg()` functions.)

Packages - Shows a list of all the R packages installed on your harddrive and indicates whether or not they are currently loaded. Packages that are loaded in the current session are checked while those that are installed but not yet loaded are unchecked. We’ll discuss packages in more detail in the next section.

Help - Help menu for R functions. You can either type the name of a function in the search window, or use the code to search for a function with the name

Organizing your working directory & setting up

Viewing your working directory

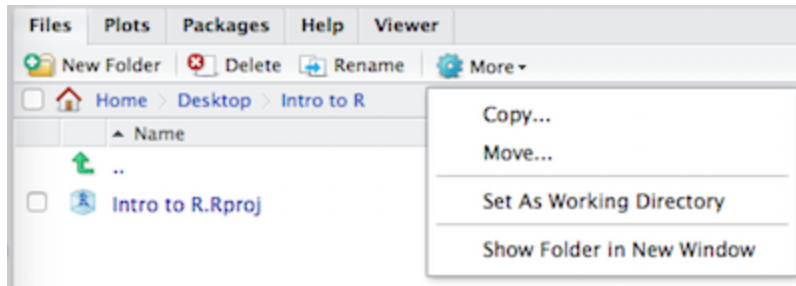
Before we organize our working directory, let’s check to see where our current working directory is located by typing into the console:

```
getwd()
```

Your working directory should be the **my_day2** folder constructed when you created the project. The working directory is where RStudio will automatically look for any files you bring in and where it will automatically save any files you create, unless otherwise specified.

You can visualize your working directory by selecting the Files tab from the **Files/Plots/Packages/Help window**.

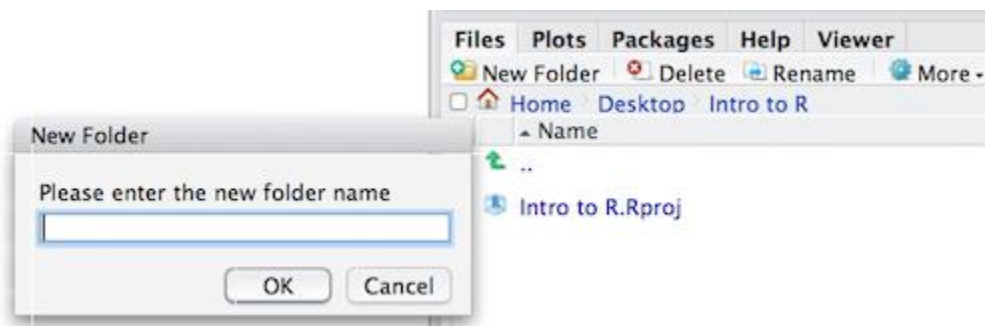
If you wanted to choose a different directory to be your working directory, you could navigate to a different folder in the Files tab, then, click on the More dropdown menu and select Set As Working Directory.



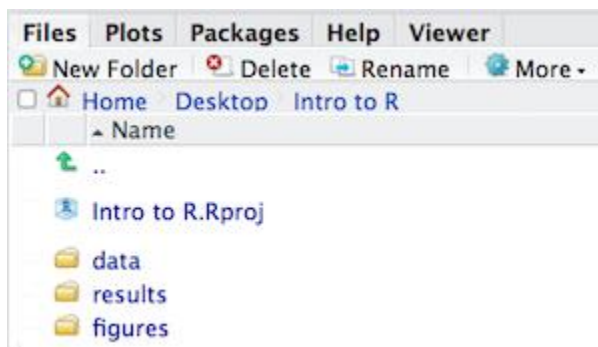
Structuring your working directory

To organize your working directory for a particular analysis, you should separate the original data (raw data) from intermediate datasets. For instance, you may want to create a data/ directory within your working directory that stores the raw data and have a results/ directory for intermediate datasets and a figures/ directory for the plots you will generate.

Let's create these three directories within your working directory by clicking on New Folder within the Files tab.

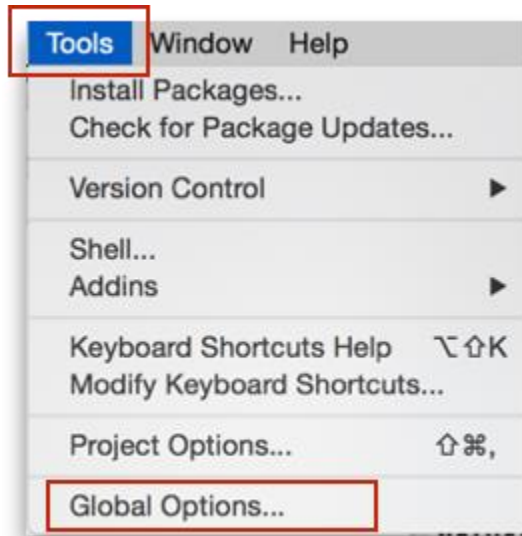


When finished, your working directory should look like:

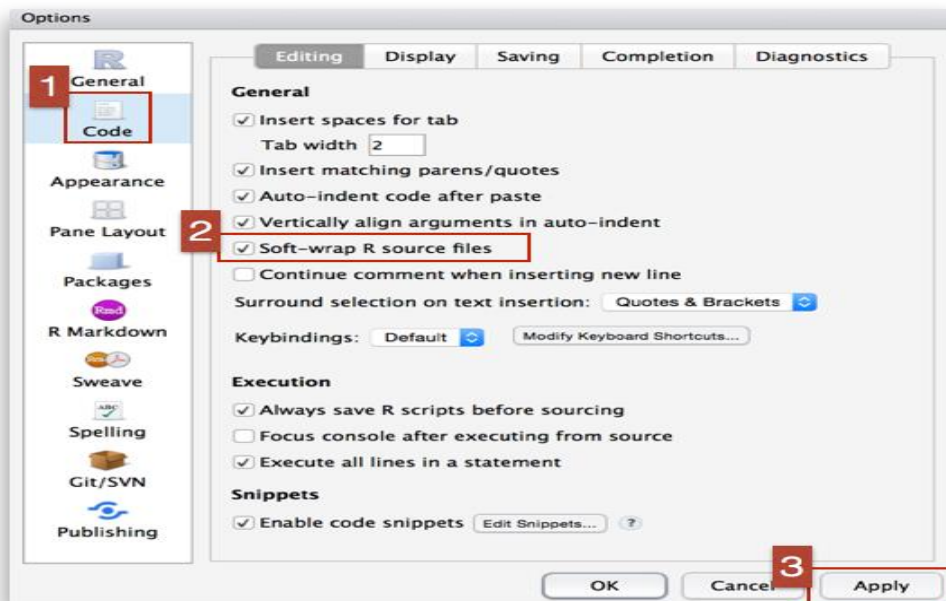


Setting up R-Studio (to change color, font, pane's location)

This is more of a housekeeping task. Just click on the global options and feel free to change color, font, pane's location of your choice. Also, we will be writing long lines of code in our script editor and want to make sure that the lines “wrap” and you don’t have to scroll back and forth to look at your long line of code. Click on “Tools” at the top of your RStudio screen and click on “Global Options” in the pull down menu.



On the left, select “Code” and put a check against “Soft-wrap R source files”. Make sure you click the “Apply” button at the bottom of the Window before saying “OK”.



Interacting with R


Now that we have our interface and directory structure set up, let's start playing with R! There are two main ways of interacting with R in RStudio: using the console or by using script editor (plain text files that contain your code).

Console window

The console window (in RStudio, the bottom left panel) is the place where R is waiting for you to tell it what to do, and where it will show the results of a command. You can type commands directly into the console, but they will be forgotten when you close the session.

Let's test it out:

3 + 5

```
Console ~/Desktop/Intro to R/   
R is free software and comes with ABSOLUTELY NO WARRANTY.  
You are welcome to redistribute it 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.  
  
> 3+5  
[1] 8  
> |
```

Script editor (Source)

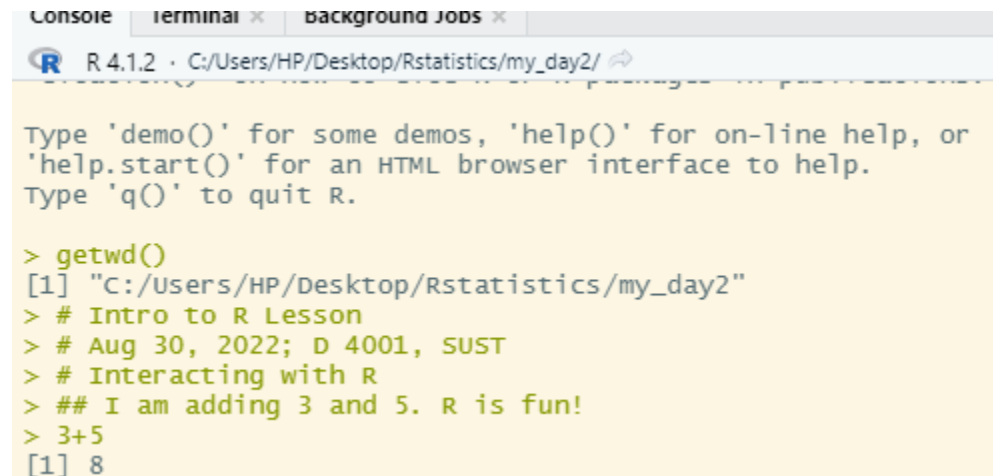
Best practice is to enter the commands in the script editor and save the script. You are encouraged to **comment** generously to describe the commands you are running using #. This way, you have a complete record of what you did, you can easily show others how you did it and you can do it again later on if needed.

The Rstudio script editor allows you to 'send' the current line or the currently highlighted text to the R console by clicking on the Run button in the upper-right hand corner of the script editor. Alternatively, you can run by simply pressing the Ctrl and Enter keys at the same time as a shortcut.

Now let's try entering commands to the script editor and using the comments character # to add descriptions and highlighting the text to run:

```
# Intro to R Lesson  
# Aug 30, 2022; D 4001, SUST  
# Interacting with R  
## I am adding 3 and 5. R is fun!  
3+5
```

You should see the command run in the console and output the result.



```
R 4.1.2 · C:/Users/HP/Desktop/Rstatistics/my_day2/

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.

> getwd()
[1] "C:/Users/HP/Desktop/Rstatistics/my_day2"
> # Intro to R Lesson
> # Aug 30, 2022; D 4001, SUST
> # Interacting with R
> ## I am adding 3 and 5. R is fun!
> 3+5
[1] 8
```

What happens if we do that same command without the comment symbol #? Re-run the command after removing the # sign in the front:

I am adding 3 and 5. R is fun!
3+5

Now R is trying to run that sentence as a command, and it doesn't work. We get an error in the console *"Error: unexpected symbol in 'I am'"* means that the R interpreter did not know what to do with that command."

Console command prompt

Interpreting the command prompt can help understand when R is ready to accept commands. Below lists the different states of the command prompt and how you can exit a command:

Console is ready to accept commands: >

If R is ready to accept commands, the R console shows a > prompt.

When the console receives a command (by directly typing into the console or running from the script editor (Ctrl-Enter), R will try to execute it.

After running, the console will show the results and come back with a new > prompt to wait for new commands.

Console is waiting for you to enter more data: +

If R is still waiting for you to enter more data because it isn't complete yet, the console will show a + prompt. It means that you haven't finished entering a complete command. Often this can be due to you having not 'closed' a parenthesis or quotation.

Exercise

Try highlighting only 3 + from your script editor and running it. Find a way to bring back the command prompt > in the console.

Escaping a command and getting a new prompt: `esc`

If you're in Rstudio and you can't figure out why your command isn't running, you can click inside the console window and press `esc` to escape the command and bring back a new prompt >.

Best practices

Before we move on to more complex concepts and getting familiar with the language, we want to point out a few things about best practices when working with R which will help you stay organized in the long run:

Code and workflow are more reproducible if we can document everything that we do. Our end goal is not just to "do stuff", but to do it in a way that anyone can easily and exactly replicate our workflow and results. All code should be written in the script editor and saved to file, rather than working in the console.

The R console should be mainly used to inspect objects, test a function or get help.

Use `#` signs to comment. Comment liberally in your R scripts. This will help future you and other collaborators know what each line of code (or code block) was meant to do. Anything to the right of a `#` is ignored by R. A shortcut for this is `Ctrl + Shift + C` if you want to comment an entire chunk of text.

Acknowledgement:

1. The materials used in this lesson are adapted from work that is Copyright © Data Carpentry (<http://datacarpentry.org/>), Harvard Chan Bioinformatics Core (HBC) under the open access terms of the Creative Commons Attribution license (CC BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.