Introduction to RStudio

What is RStudio?

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.

You can see the complete RStudio user-interface cheatsheet (PDF) here

Creating a new project directory in RStudio

Let's create a new project directory for the Research and Design course.

1. Open RStudio
2. Go to the `File` menu and select `New Project`.
3. In the `New Project` window, choose `New Directory`. Then, choose `New Project`. Name your new directory whatever you want and then "Create the project as subdirectory of:" the Desktop (or location of your choice).
4. Click on `Create Project`.
5. 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 `[your project name].Rproj`.
6. When RStudio opens, you will see three panels in the window.
7. Go to the `File` menu and select `New File`, and select `R Script`. The RStudio interface should now look like the screenshot below.

TIP: If your RStudio displays only one left pane it is because you have no scripts open yet.

What is a project in RStudio?

It is simply a directory that contains everything related your analyses for a specific project. RStudio projects are useful when you are working on context- specific analyses and you wish to keep them separate. When creating a project in RStudio you associate it with a working directory of your choice (either an existing one, or a new one). A `. RProj file` is created within that directory and that keeps track of your command history and variables in the environment. The `. RProj file` can be used to open the project in its current state but at a later date.

When a project is **(re) opened** within RStudio the following actions are taken:
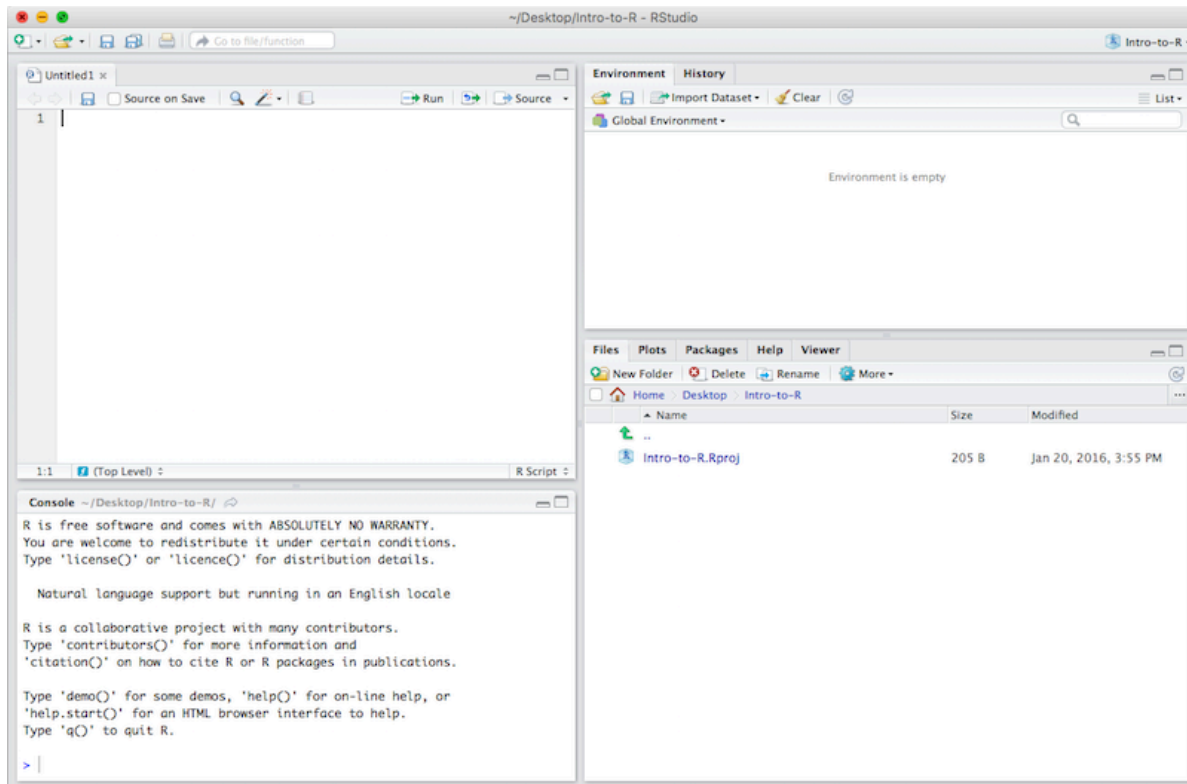
- A new R session (process) is started

Figure 1: RStudio interface

- The .RData file in the project's main directory is loaded, populating the environment with any objects that were present when the project was closed.
- The .Rhistory file in the project's main directory is loaded into the RStudio History pane (and used for Console Up/Down arrow command history).
- The current working directory is set to the project directory.
- Previously edited source documents are restored into editor tabs
- Other RStudio settings (e.g. active tabs, splitter positions, etc.) are restored to where they were the last time the project was closed.

Information adapted from RStudio Support Site

RStudio Interface

The RStudio interface has four main panels:

1. **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.*
2. **Script editor**: where you can type out commands and save to file. You can also submit the commands to run in the console.
3. **Environment/History**: environment shows all active objects and history keeps track of all commands run in console
4. **Files/Plots/Packages/Help**

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 `Intro-to-R` 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`.
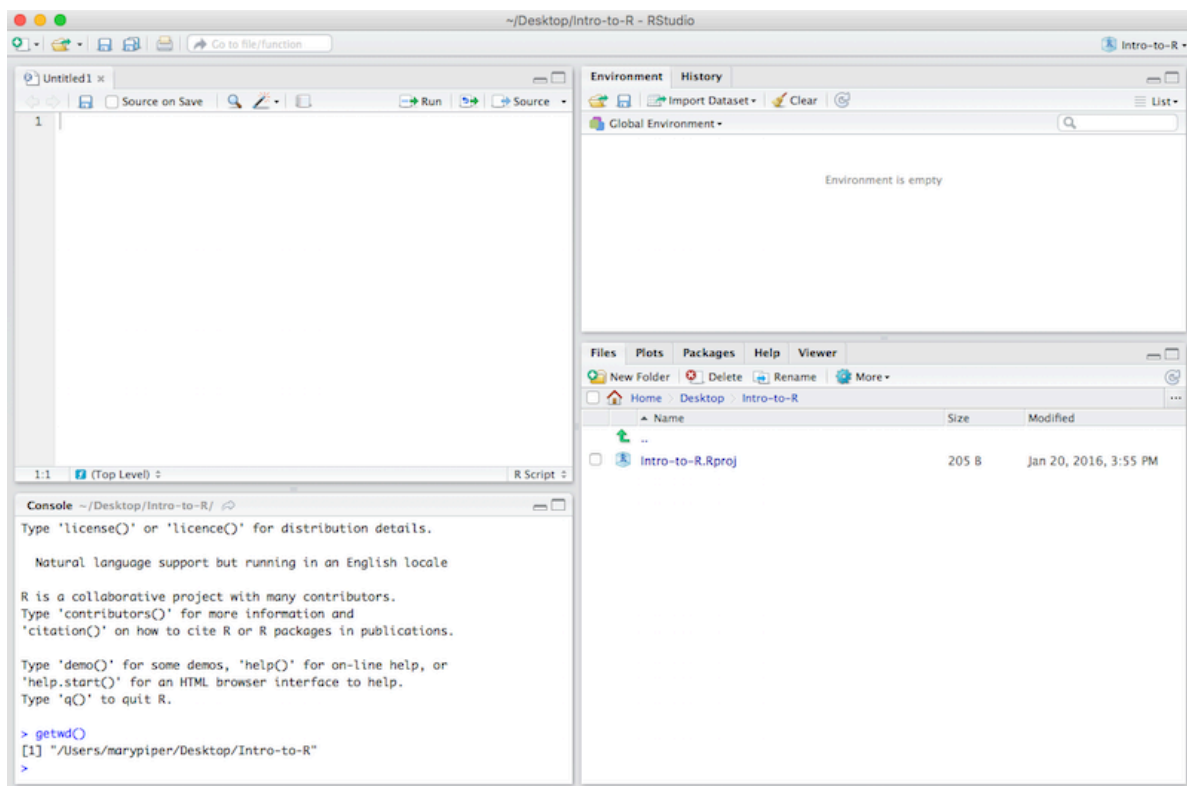
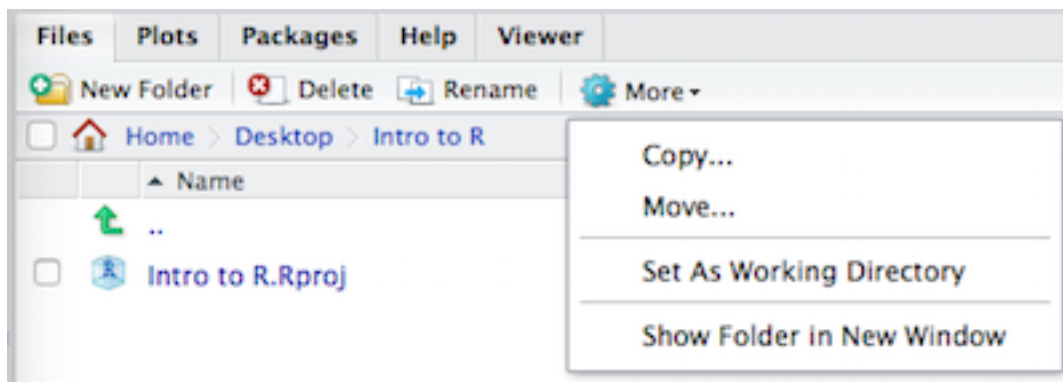Figure 2: Viewing your working directory

Figure 3: Setting your working directory

Structuring your working directory

To organize your working directory for a particular analysis, you typically want to 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.
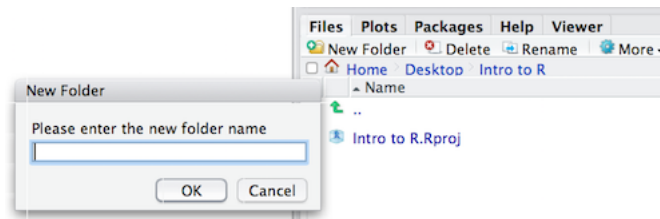

Figure 4: Structuring your working directory

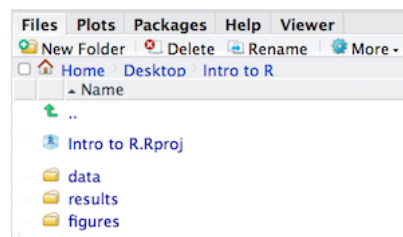When finished, your working directory should look like:

Figure 5: Your organized working directory

Setting up

This is more of a housekeeping task. 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".
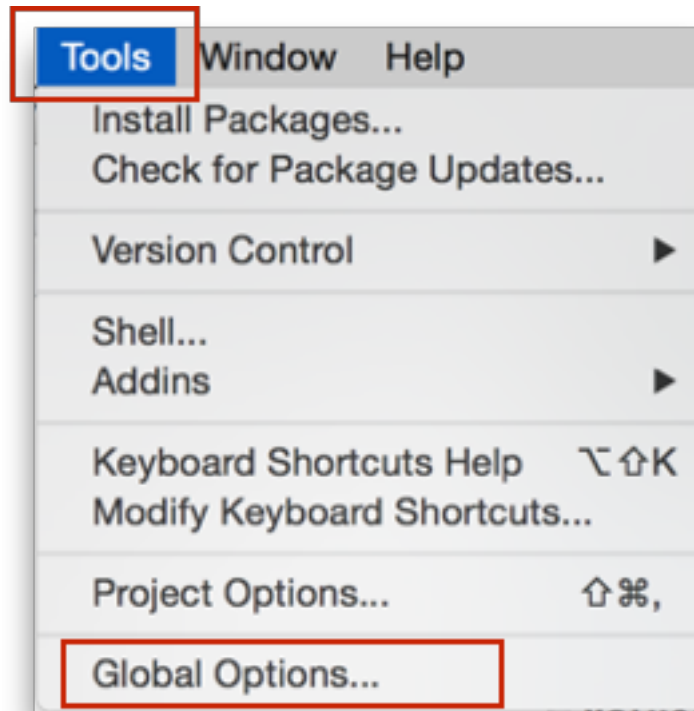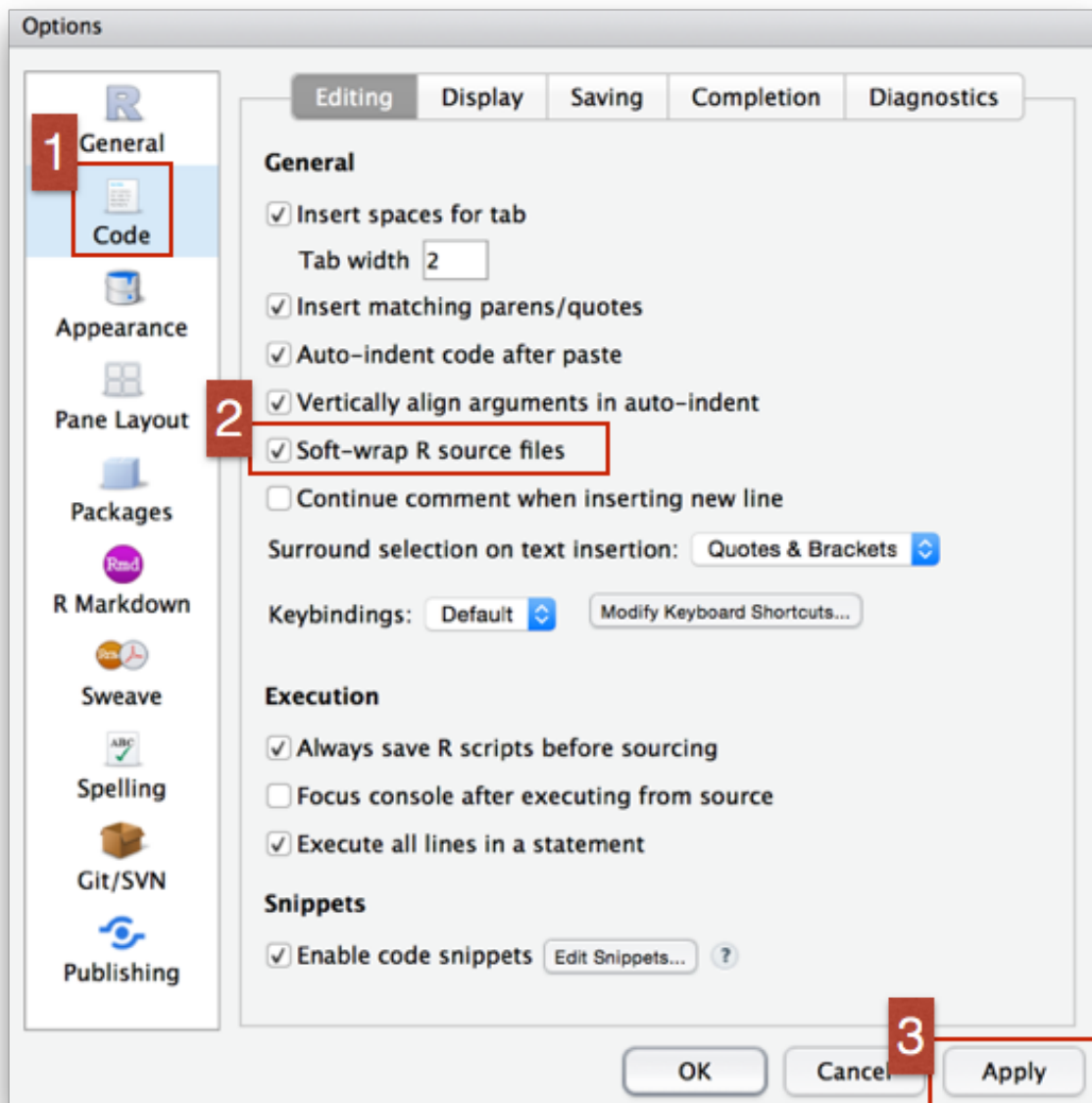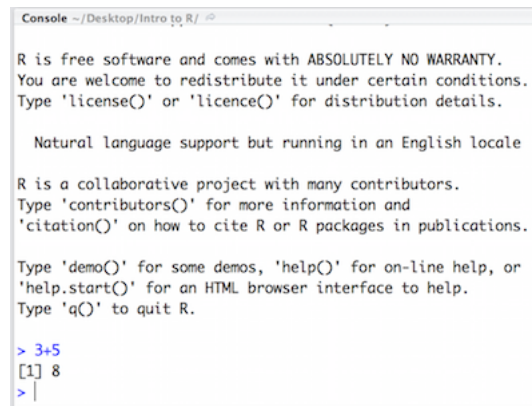
Figure 6: options

Figure 7: wrap_options

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.

```
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
>
```

Figure 8: Running in the console

Script editor

Best practice is to enter the commands in the **script editor**, and save the script. You are encouraged to comment liberally 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:

```
# Session 1
# Feb 3, 2023

# Interacting with R

# I am adding 3 and 5.
3+5
```
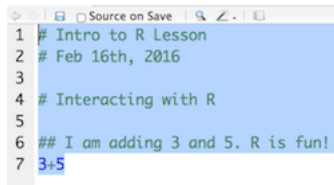

Figure 9: Running in the script editor

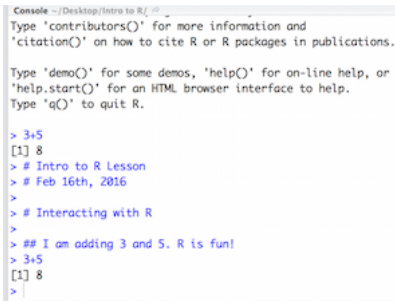You should see the command run in the console and output the result.

Figure 10: Script editor output

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.

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 `>`.

Keyboard shortcuts in RStudio

In addition to some of the shortcuts described earlier in this lesson, we have listed a few more that can be helpful as you work in RStudio.

> **ℹ Shortcuts Table**
>
> Some very useful keyboard shortcuts are below. See all the keyboard shortcuts for Windows, Max, and Linux in the second page of this RStudio [user interface cheatsheet](#).
>
> | Windows/Linux | Mac | Action |
> |---|---|---|
> | Esc | Esc | Interrupt current command (useful if you accidentally ran an incomplete command and cannot escape seeing "+" in the R console) |
> | Ctrl+s | Cmd+s | Save (script) |
> | Tab | Tab | Auto-complete |
> | Ctrl + Enter | Cmd + Enter | Run current line(s)/selection of code |
> | Ctrl + Shift + C | Cmd + Shift + c | comment/uncomment the highlighted lines |
> | Alt + - | Option + - | Insert `<-` |
> | Ctrl + Shift + m | Cmd + Shift + m | Insert `%>%` |

Ctrl + l	Cmd + l	Clear the R console
Ctrl + Alt + b	Cmd + Option + b	Run from start to current line
Ctrl + Alt + t	Cmd + Option + t	Run the current code section (R Markdown)
Ctrl + Alt + i	Cmd + Shift + r	Insert code chunk (into R Markdown)
Ctrl + Alt + c	Cmd + Option + c	Run current code chunk (R Markdown)
up/down arrows in R console	Same	Toggle through recently run commands
Shift + up/down arrows in script	Same	Select multiple code lines
Ctrl + f	Cmd + f	Find and replace in current script
Ctrl + Shift + f	Cmd + Shift + f	Find in files (search/replace across many scripts)
Alt + l	Cmd + Option + l	Fold selected code
Shift + Alt + l	Cmd + Shift + Option+l	Unfold selected code

💡 Take advantage of auto-complete

Use your Tab key when typing to engage RStudio's auto-complete functionality. This can prevent spelling errors. Press Tab while typing to produce a drop-down menu of likely functions and objects, based on what you have typed so far.

R syntax

Now that we know how to talk with R via the script editor or the console, we want to use R for something more than adding numbers. To do this, we need to know more about the R syntax.

The main "parts of speech" in R (syntax) include:

- the **comments #** and how they are used to document function and its content
- **variables** and **functions**
- the **assignment operator <-**
- the **=** for **arguments** in functions

We will go through each of these "parts of speech" in more detail, starting with the assignment operator.

Assignment operator

To do useful and interesting things in R, we need to assign *values* to *variables* using the assignment operator, `<-`. For example, we can use the assignment operator to assign the value of `3` to `x` by executing:

```
x <- 3
```

The assignment operator (`<-`) assigns **values on the right** to **variables on the left**.

In RStudio, typing `Alt + -` (push `Alt` at the same time as the `-` key, on Mac type `option + -`) will write `<-` in a single keystroke.

Variables

A variable is a symbolic name for (or reference to) information. Variables in computer programming are analogous to "buckets", where information can be maintained and referenced. On the outside of the bucket is a name. When referring to the bucket, we use the name of the bucket, not the data stored in the bucket.

In the example above, we created a variable or a 'bucket' called `x`. Inside we put a value, `3`.

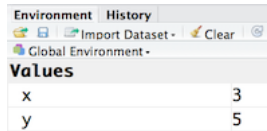Let's create another variable called `y` and give it a value of 5.

```
y <- 5
```

When assigning a value to an variable, R does not print anything to the console. You can force to print the value by using parentheses or by typing the variable name.

```
y
```

You can also view information on the variable by looking in your `Environment` window in the upper right-hand corner of the RStudio interface.

Now we can reference these buckets by name to perform mathematical operations on the values contained within. What do you get in the console for the following operation:

Figure 11: Viewing your environment

```
x + y
```

Try assigning the results of this operation to another variable called `number`.

```
number <- x + y
```