# 1. First Steps in R

Self-Paced Workshop #1

Prof. Jack Reilly

Fall 2026

## Coding in R

Many parts of working with R can be tricky the first time you do it. The purpose of this exercise is to get you familiar with the basic code and syntax of R without having to worry about your local R installation yet.

### Step 1. Objects

In the R language, most data is stored in an object. Objects can take any name we give them. Here, for instance, is a line of code that stores the number "7" under the object "horcrux".

```
horcrux<-7
horcrux
```

Click "Run Code" on the snippet above and see what happens![1]

Let's pull apart what is happening here. There are two lines of code.

- Line One: `horcrux<-7` does two things simultaneously. First, it *creates* an object named `horcrux`. Second, it stores something - in this case, the number `7` - in the object that it just created.
- Line Two: `horcrux` *recalls* the previously stored object from memory. When an object is recalled, R prints the contents of that object to the screen. (In this case, `7`).

This basic structure is the most fundamental to R. *"Do Math"* and *"Store Result in Object"* is, to a high level of approximation, all we ever do in R code.

Try it yourself! Change the code below so that the number "8" is stored under the object "horcrux". Click "Run" when you're ready to see the results.

```
horcrux<-7
horcrux
```

[1] You should see a `[1] 7` appear just below the code box.

### Step 2. Simple Math

We can use math to manipulate objects together. Look at the code below and try to figure out what the answer (output) is going to be.

```
horcrux<-7
harry<-1
harry+horcrux
```

Note, also, that we can store the result of any math in a new object, like this:

```
horcrux <- 7
harry <- 1
fragments <- harry + horcrux
fragments
```

Let's walk through precisely what is happening, line by line:

1. `horcrux <- 7` - *Create an object "horcrux" and store the number "7" in it.*
2. `harry <- 1` - *Create an object "harry" and store the number "1" in it*
3. `fragments <- harry + horcrux` - *Create an object "fragments" and store the result of "harry+horcrux" in it*
4. `fragments` - *Show me (print to screen) what is stored in the object "fragments"*

In the box below, try editing the code so that:

1. You create three objects, each with different numbers stored:
   - horcrux - 6
   - harry - 1
   - voldemort - 1
2. Add all three objects together, storing the result in `fragments`
3. Print (return) whatever is stored in fragments to the screen.

```
horcrux <- 7
harry <- 1
fragments <- harry + horcrux
fragments
```

Note that you can use all standard notation for basic math: addition `+`, subtraction `-`, multiplication `*`, division `/`, exponents `^`, and parentheses `( )`.

### Step 3. Comments

It was useful to see those comments right by the line of code, wasn't it? Fortunately, R syntax has a built in way that you can do this, with the pound `#` sign. Any text after the pound sign on a line will be ignored when you run your code. Thus, the code below runs exactly the same as the code above.

Click "Run Code" and try it!

```
horcrux <- 7 # Create an object "horcrux" and store the number "7" in it.
harry <- 1 # Create an object "harry" and store the number "1" in it
fragments <- harry + horcrux # Create an object "fragments" and store the result of "harry+ho
fragments # Show me (print to screen) what is stored in the object "fragments"
```

### Step 4. Functions

We can do more than perform simple math on objects. We can also use *functions* built into R to perform more complex math on objects.

A *function* is a command in R that does a particular mathematical operation on an object. It has a name, followed by parentheses, inside of which is the data on which the function should be performed. The `sqrt()` function, for instance, finds the square root of a number. Try it!

```
sqrt(9)
```

You can also perform functions on objects. For instance, if we wanted to find the square root of the number in object `horcrux`, we easily can:

```
horcrux <- 7 # Create an object "horcrux" and store the number "7" in it.
sqrt(horcrux) # Find the square root of "horcrux"
```

We could also store the result of this calculation in another object:

```
horcrux <- 7 # Create an object "horcrux" and store the number "7" in it.
roothorcrux <- sqrt(horcrux) # Find the square root of "horcrux", store it in a new object
roothorcrux #return (print) the contents of the object "roothorcrux"
```

Try it yourself. Manipulate the following code so that the number "3" is stored in the object "hallows", store the square root of hallows in another object, and then print that object to the screen.

```
hallows <- 4

```

### Step 5. Vectors

When we store single numbers in objects, we call that single number a "scalar". Most things we want to store, however, will be more complex than just simple scalars. You can also store groups of numbers in a vector - a list of numbers - using the `c()` function. Run the code and see what gets returned!

```
owls <- c(7, 10, 7)
owls
```

We can also create two vectors and add them together. Try it!

```
owls <- c(7, 10, 7)
newts <- c(3, 7, 3)
```

```
owls+newts
```

### Step 6. Functions on Vectors

Finally, we can also perform *functions* on vectors. The function `mean()` finds the mean of all numbers in a vector.

```
owls <- c(7, 10, 7)
mean(owls)
```

Try it yourself. Write additional code in the box below to calculate the mean for `newts`.

```
newts <- c(3, 7, 3)
```

Congratulations! You now know the elementary control functions built into the R language.

Work on Your Own

Return to Problem Set 1 and complete the rest of the assignment.