Week 2: Intro to R

Ellen Bledsoe

2024-01-23

Introduction to R/RStudio

R vs. RStudio vs. Posit Cloud

R

- Programming language
- Started as a statistics and data analysis environment, but it can also build websites, run simulations, and many other things
- R is what runs all of the code we will write this semester
- Separate from RStudio

RStudio

- IDE - Integrated Development Environment
- Makes developing code in R easier by including a number of tools in one place

Posit Cloud

- An online version of RStudio that runs in your browser
- We're using it because it:
    1. Avoids installation difficulties
    2. Makes sharing code with instructors for debugging easier
    3. Let's us leave some of the complexities of working with R until after we've learned the basics
- We will switch to using RStudio on our own computers in the second half of the semester.

Rstudio Mini-tour

1. What do the different panels do?
2. Console versus a script/Rmarkdown file

Intro to Coding in R

Basic expressions

(AKA using R as a calculator)

Let's start by typing an expression in the console (below).

Try multiplying 5 and 3 in the Console (hint: * means multiply). Hit `Enter` to run that line of code.

Rmarkdown and Code Chunks

Rmarkdown (.Rmd) is a file format that lets us incorporate text and code into one document seamlessly. In fact, it is the file format for this document!

- For writing text, you can type as you normally would.
- Code chunks are a bit different:
  - Near the top right of your screen you can toggle between viewing this document as under Source or Visual.
  - If you view this document under `Source` mode, you will see that all code chunks are sandwiched between " `{r} " and " `".
  - But under `Visual` mode, you can type R code in the lines under `{r}`.
  - To include text in chunks, you will need to put a `#` in front. R will not read anything in the line after `#` as code.
  - Commenting your code is considered best practice. Not only does it help other people understand what your code is doing, but it will help *you* remember what your code is doing and why!

Code chunks look like this:

```
# This is a code chunk!
```

To run a chunk of code, click the green arrow on the top right corner of the chunk.

You can also run one or a few lines of code at a time by having your cursor on the line or highlighting multiple lines and hitting `Ctrl` + `Enter` (or `Cmd` + `Enter` on a Mac).

A quick shortcut for adding a code chunk is `Ctrl + Alt + i` (`Cmd + Opt + i` on a Mac). Alternatively, you can go to Code > Insert Chunk.

Some Code Chunk Practice

Let's work with an example code chunk.

```
# convert grams to kg
50 / 1000 * 2.2
```

```
## [1] 0.11
```

Assigning Objects

(AKA Creating Variables)

Assignments are really key to almost everything we do in R. This is how we create permanence in R. Anything can be saved to an object, and we do this with the assignment operator, `<-`.

The short-cut for `<-` is `Alt + -` (or `Option + -` on a Mac)

Note: You can technically use either `<-` or `=`. I recommend getting used to `<-`, as this is standard practice (now) in R and recommended in style guides. The reasons why are complicated and don't often arise in most situations, but it is still good practice.

```
weight_g <- 50
```

We can see that this object has been created by looking in the Environment tab.

Objects works just like the value itself. For example, we can use them in mathematical expression.

```r
weight_g / 1000
```

```
## [1] 0.05
```

```r
weight_g / 1000 * 2.2
```

```
## [1] 0.11
```

```r
weight_lb <- weight_g / 1000 * 2.2
```

The object won't change unless you assign a new value to it directly using the assignment operator (`<-`).

```r
weight_g
```

```
## [1] 50
```

```r
weight_g * 2
```

```
## [1] 100
```

```r
weight_g
```

```
## [1] 50
```

```r
weight_g <- 26
weight_g
```

```
## [1] 26
```

Some RStudio Tips

Use the`tab` key to autocomplete objects (and other things)

- Let the computer do repetitious work.
- It's easier and causes fewer mistakes.

Also, you can see the commands you've run under `History` in case you forgot to write something down in your Source code.

Let's Practice

Head over to this week's assignment. It is already in this Posit Cloud project, so you can find it in the Files tab. I recommend clicking over into Visual mode.

Start working on Exercises 1 and 2 in the Assignment. Remember to comment your code as you go!

Functions

Functions are pre-written bits of codes that perform specific tasks for us. Effectively, they are complex expressions rather than "basic" ones.

A function call consists of two main parts.

- First, the name of the function, followed by parentheses.
- Next, the arguments that the function requires to perform its task. Anything you type into the parentheses are arguments.

For example:

```r
sqrt(49)
```

```
## [1] 7
```

Here,`sqrt()` is the name of the function, and `49` is the argument.

We can also pass objects as the argument in functions.

```r
weight_lb <- 0.11
sqrt(weight_lb)
```

```
## [1] 0.3316625
```

Data Types

Another useful function is `str()`, which is short for "structure."

It will tell us information about the dimensions and data type of an object.

```r
# numeric
str(weight_lb)
```

```
##  num 0.11
```

Another data type is for text data. We write text inside of quotation marks.

```r
"hello"
```

```
## [1] "hello"
```

If we look at the structure of some text we see that it is type "character."

```r
str("hello world")
```

```
##  chr "hello world"
```

Many functions can take multiple arguments. For example, let's use a function to round `weight_lb` to one decimal place.

```r
round(weight_lb, 1)
```

```
## [1] 0.1
```

Functions return values, so as with other values and expressions, if we don't save the output of a function then there is no way to access it later.

It is common to forget this when dealing with functions and expect the function to have changed the value of the variable. However, if we look at `weight_lb`, we see that it hasn't been rounded.

```r
weight_lb
```

```
## [1] 0.11
```

To save the output of a function we assign it to a variable.

```r
weight_rounded <- round(weight_lb, 1)
weight_rounded
```

```
## [1] 0.1
```

Let's Practice

Start working on Exercises 4 and 5.