Introduction to R

Overview

In this lab, you are going to get acquainted with R, the RStudio interface, and basic data manipulation. By the end of this lab, you should be comfortable using R as a very complex calcuator. This may not be inspiring now, but there are many labs to come. Start here.

Basics

R Studio is an Interpreter for R. R is a programming language. We will write R, but we will use RStudio as our environment. RStudio has a lot of features. Let's try them out.

There are 2 places to write code in RStudio: both on the left, you can write on the top and the bottom. The bottom is the Console. The console is interactive: you type a line, hit 'Enter', and R gives a response. You can't really save this output very easily, but it's good for experimentation and quick results. On top, you'll create a new R Script, where you can save your code and run it in batches. You can run code here by putting your cursor at the end of the sentence you want to run and clicking 'Run' in the upper Right corner or using any of the keyboard shortcuts in the 'Run' list. Try it out. Type the following code in the Console.

```
2+3
```

Notice that the sentence starts with a >. If you ever see + in the console, the sentence is incomplete.

Let's keep working in the console and set a variable. You'll use variables a lot. The biggest visual difference between R and other programming languages may be the use of <- to set variables rather than =.

```
x <- 4
x+2
```

Notice that your variable appears in the upper right corner - the Environment. All the variables you specify in your R session will appear here. If you are having

major problems and something just isn't running even though your code is right, clean up this environment and re-run your code. Let's try it now. Click on the broom (last icon in the Environment tab). Your variable disappeared. Now run this:

```
x+2
```

You should have gotten an error. x no longer exists in our environment.

As you may have guessed, R functions like a big calcuator. We won't use this much, but if you remember things like order of operations, you'll be ok. See the textbook for the operators.

We'll deal more with character types later in a later lab, but for now, just know that we have numbers that come as Doubles (numbers), Characters (strings), Booleans (True/False). Let's try these out.

```
my_num <- 2/2
my_char <- "hi!"
my_bool <- TRUE


typeof(my_num)
typeof(my_char)
typeof(my_bool)
```

There's one more special object, null values. Any time data is missing, you will get a null value. Let's look

```
my_nan <- 0/0
my_nan
```

You'll learn much more about how to deal with these.

In general, you'll want to use R as more than a calculator, you'll want to draw on pre-made packages, particularly the tidyverse. There are literally thousands of packages available for different functions and datasets. All of Jane Eyre's works, anyone?

If you think about human languages, we have the structure and vocabulary. If you recently learned German, you may know the structure very well, be able to conjugate verbs, make agreement, etc. Let's say you went to a dinner and you knew everyone there would want to talk about politics. You'd probably want to learn as much politics vocabulary as you could. You don't always need it, but you only need to import it into your brain once and then you can call on it any time. Your German is like R. R is a skeleton, it's the structure and some basic vocabulary. You need to import packages into it to make it the most useful. Let's load the tidyverse.

First install it (you need to be connected to the internet for this). You only need to do this once.

```
install.packages("tidyverse")
```

Then load it into your environment so you can use it.

```
library(tidyverse)
```

If you don't get an error, you are all good!

If you ever want to write notes to yourself (and you should write many notes: clear and coherent comments are not only good code, but also kind to your future self who has to figure out what your today self was doing with that line of code – trust me!), use a # in front of the line.

```
#creates a null variable
my_nan <- 0/0
```

The last thing you need to get started is the help pages. This is what makes RStudio an invaluable resource. if you forget the exact name of the command you are looking for, type help(KEYWORD) and hit enter. RStudio will display the help pages.

True story - when I was making this lab, I forgot how you find the type of an object in R - a common problem when switching between similar programming languages such as Python and R, or when you are brand new to a language. I typed the code below and was quickly reminded. You can do the same thing if you are trying to do a particular statistical test or find a function by its purpose.

```
help(type)
```

Congratulations! Armed with the difference between the console and a Notebook, Running your code, Sweeping your variables, and getting help, you are ready to dive in and start using R as more than a calculator!