SPEC REU: Introduction to R

Alix Ziff, Gaea Morales, Zachary Johnson

Summer 2022

Ready or not

Today we will explore R and learn the very *very basics*. We will go over what R is, how to write a line of code, basic arithmetic, and getting help.

This section would not be possible without the previous efforts and brilliance of Miriam Barnum and Therese Anders. Except for the alliteration and puns. . . that's all Alix Ziff.

R'nt you going to explain what R is?

R is a programming language for statistical computing and data visualization. It's open source (free and frequently updated) unlike Stata or SPSS. R is maintained and developed by a vibrant community of programmers and statisticians (and Therese!) and offers packages for just about any statistical task imaginable.

For political science and international relations scholars, R has become the dominant programming language in the field and is used for everything from regression analysis to data management to making pretty pictures.

R we gonna do this or what?!

If you have not downloaded this: RStudio Desktop and this The Comprehensive R Archive Network . . . do it.

You need to download both 'R' and R studio. R is the actual statistical computing software. R studio is a warm and fuzzy user interface.

R Studio

Generally, students and scholars type in an **R script** (in the menu: `File > New File > R Script`) that they save to keep track of their code and operations.

The **Console** (by default, a window below the `R Script`) is where the computing is done.

The (Global) **Environment** (by default, a window to the right of the `R Script`) keeps track of the data you are adding or manipulating.

The **Files, Plots, Packages, Help and Viewer** (located below the `Global Environment`) is where you can view visualizations, get information on functions and packages, or load a file. **You will type into either the R script or the console**. *However*, it is best practice to type your code into the R script (and save that `.R` file).

I command you to. . .

To make a command (also called a function) you can either highlight your code and click 'Run' at the top of the R script or you can highlight and press the `ctrl/command` and `enter` keys simultaneously.

You can make comments that will not run as code using a `#` before any word or line.

I object!

In R, you can assign values to an **object**. This is how R stores information and any data you create, use, or manipulate. You can name objects almost whatever you want. **Except** you cannot start them with numbers, use special characters, or anything that doubles as an operator or function, i.e. "reserved words" (e.g., function, false, if, else, repeat, for, NA, etc.). For the list, you can type `help(reserved)` or `?reserved` and run the code in your R script.

For example: if I wanted to store 10102020 as my favorite number I would do the following:

```
luckyno <- 10102020
```

Then I can see that value using the function `print`.

```
print(luckyno) # or simply typing luckyno and running it
luckyno
```

Arithmetic in R

While it can complete a broad range of quite complex calculations, R can also serve as a basic calculator.

```
5*2
```

You can also use numbers stored as objects instead of writing out the value itself.

```
luckyno/2
```

Or make new objects from the results of operations.

```
haha <- 1601.6 * 5
```

Or change or rewrite the value of existing objects. This is important: you can rewrite objects by using the same object name, with a new value.

```
haha <- 8007393-742
```

Using Arithmetic Operators

There are several logical operators you can use that will result in TRUE or FALSE

	Operator
Add `x` and `y`	`x + y`
Subtract `x` and `y`	`x - y`
Multiply `x` times `y`	`x * y`
Divide `x` by `y`	`x / y`
Raise `x` to the power of `y`	`x ^ y` (or `x**y`)
Remainder of dividing `x` by `y`	`x %% y`
Divide `x` by `y` but round down to integer	`x % / % y`
Less than	`<`
Less than or equal to	`<=`
Greater than	`>`
Greater than or equal to	`>=`
Exactly equal to	`==`
Not equal to	`!=`
Not `x`	`!x`
`x` or `y`	`x | y`

	Operator
x and y	`x & y`
is x in y	`x %in% y`

This last operator asks whether a value is contained within a set (i.e., in a vector), and provides a logical output (either True or False)

For example: *Is 8 in the vector (a collection of ordered information) 2,3,4,5?*

```
8 %in% c(2,3,4,5)
```

Vectors & Variables

A vector is the simplest type of data you can work with in R. It is essentially a list of data of a single or same variable type. You should be familiar with **numerical** (integer, single, double); **character**; **logical** and **factor** variables.

- **Numerical variables**: Well, any number.
  - **Integer**: These are whole numbers without decimal points.
  - **Single** and **double** precision types just refer to how R stores and represents the numeric data.
- **Character variables**: This is what R (and other programming languages such as Python) calls a string. We typically store any alphanumeric data that is not ordered as a character vector.
- **Logical variables**: A collection of TRUE and FALSE values.
- **Factor variables**: Think about it as an ordinal variable, i.e. an ordered categorical variable.

To create a vector we use the function `c()` ("concatenate") to combine separate data points. The general format for creating a vector in R is as follows:

```
name_of_vector <- c("what you want to put into the vector")
```

The "Oh No!"s: Arrows, quotes, and typos

Let's avoid some errors:

- Make sure your operator is facing the right direction `<-`.
- Make sure you do not have a `#` before a command. That makes it a comment which won't run in R.
- Make sure you pay attention to capitalization; R is case sensitive.
- Make sure you have both sides of the quotation mark or parenthesis.
- CHECK FOR TYPOS: typos are often the cause of your 'Oh No!' moment.
- Make sure all your objects have unique names (you may accidentally overwrite existing objects).

Note: You can use the `ls()` command to print out all the vectors/objects you've defined. This can be helpful when you're unsure if you've used a word/name already. Using the same vector/object name as a previous object to store a new operation/new data will rewrite that vector/object.

How to get Help

Google: No seriously. If you are struggling and it isn't any of the above, then someone has likely had the same struggle. There are several websites like Stack Overflow that know everything about everything R.

R Studio Help Menu: It is the tab between `Packages` and `Viewer` and it can help with how to order things, what a package does, etc. You can also access the help menu directly in your script by running a command with a question mark at the start, e.g.: `?c()`.

SPEC Statistics Consulting Hours: There's at least an hour every single day with a brilliant SPEC student or doctoral affiliate. Take advantage.