Epigenomics for Social Scientists 2021

00 Introduction to R and R Markdown

Kelly Bakulski

Last compiled on July 28, 2021

Software used

Programming language: R

The programming language used here is R, which is a primarily used for statistical computing. More information can be found here https://www.r-project.org/.

IDE: Rstudio

IDE (Integrated Development Environment) is fancy language for a piece of software that consolidates different elements of coding. We can see this in the 4 paneled window we are looking at.

Authoring Software: RMarkdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

Common new R users frustrations

1. Different versions of software
    - RStudio Cloud solves this
2. Working directory problems: trying to read files that R "can't find"
    - RStudio Cloud solves this and so does RStudio Projects
3. Data type problems (is that a string or a number?)
    - discussed throughout
4. Typos (R is **case sensitive**, `x` and `X` are different)
    - RStudio helps with "tab completion"
    - discussed throughout

Explaining commands/output

An R command (we'll also call it code or a code chunk) will be grey and look like this

```r
print("I'm code")
```

```
## [1] "I'm code"
```

And then directly after it, will be the output of the code.
So `print("I'm code")` is the code chunk and [1] "I'm code" is the output.

R as a calculator

The R console is a full calculator Try to play around with it: +, -, /, * are add, subtract, divide and multiply ˆ or ** is power parentheses – (and) – work with order of operations

```
2 + 2
```

```
## [1] 4
```

```
2 * 4
```

```
## [1] 8
```

```
2 ^ 3
```

```
## [1] 8
```

```
2 + (2 * 3)^2
```

```
## [1] 38
```

```
(1 + 3) / 2 + 45
```

```
## [1] 47
```

Note, when you type your command, R inherently thinks you want to print the result.

Try evaluating the following:

`2 + 2 * 3 / 4 -3` `2 * 3 / 4 * 2` `2^4 - 1`

Commenting in Scripts

# is the comment symbol in R

```
# Comments in R follow the hashtag symbol
# Nothing to its right is evaluated. Note the color of your code changes after you use a hashtag

# This # is still a comment
### You can use many #'s as you want
1 + 2 # Can be the right of code
```

```
## [1] 3
```

```
# Best practice is to take a ton of notes to help your future self and anyone who comes later to re-run
```

R variables

- You can create variables from within the R environment and from files on your computer
- R uses "<-"or "=" to assign values to a variable name
- Variable names are case-sensitive, i.e. X and x are different

```
x <- 2 # Same as: x = 2
x
```

```
## [1] 2
```

```
x * 4
```

```
## [1] 8
```

```
x + 2
```

```
## [1] 4
```

R variable classes

- The most comfortable and familiar class/data type for many of you will be `data.frame`
- You can think of these as essentially Excel spreadsheets with rows (usually subjects or observations) and columns (usually variables) `data.frame`s are somewhat advanced objects in R; we will start with simpler objects;
- Here we introduce "1 dimensional" classes; often referred to as 'vectors'
- Vectors can have multiple sets of observations, but each observation has to be the same class.

```
class(x)
```

```
## [1] "numeric"
```

```
y <- "hello world!"
print(y)
```

```
## [1] "hello world!"
```

```
class(y)
```

```
## [1] "character"
```

R variable practice

Try assigning your full name to an R variable called `name`

```
name <- "Kelly Bakulski"
name
```

```
## [1] "Kelly Bakulski"
```

The 'combine' function

The function `c()` collects/combines/joins single R objects into a vector of R objects. It is mostly used for creating vectors of numbers, character strings, and other data types.

```
x <- c(1, 4, 6, 8)
x
```

```
## [1] 1 4 6 8
```

```
class(x)
```

```
## [1] "numeric"
```

The 'combine' function practice

Try assigning your first and last name as 2 separate character strings into a length-2 vector called `name2`

```
name2 <- c("Kelly","Bakulski")
name2
```

```
## [1] "Kelly"    "Bakulski"
```

The 'length' function

`length()`: Get or set the length of vectors (including lists) and factors, and of any other R object for which a method has been defined.

```
length(x)
```

```
## [1] 4
```

```r
y
```

```
## [1] "hello world!"
```

```r
length(y)
```

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

The ‘length’ function practice

What do you expect for the length of the `name` variable? What about the `name2` variable?

What are the lengths of each?

```r
length(name)
```

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

```r
length(name2)
```

```
## [1] 2
```

R functions on vectors

You can perform functions to entire vectors of numbers very easily.

```r
x + 2
```

```
## [1]  3  6  8 10
```

```r
x * 3
```

```
## [1]  3 12 18 24
```

```r
x + c(1, 2, 3, 4)
```

```
## [1]  2  6  9 12
```

R functions on vectors depend on class

But things like algebra can only be performed on numbers.

```r
name2 + 4
```

```
## Error in name2 + 4: non-numeric argument to binary operator
```

R assign new vectors

And save these modified vectors as a new vector.

```r
y <- x + c(1, 2, 3, 4)
y
```

```
## [1]  2  6  9 12
```

Note that the R object `y` is no longer “Hello World!” - It has effectively been overwritten by assigning new data to the variable

The structure function

You can get more attributes than just class. The function `str` gives you the structure of the object.

```r
str(x)
```

```
##  num [1:4] 1 4 6 8
```

```r
str(y)
```

```
##  num [1:4] 2 6 9 12
```

This tells you that `x` is a numeric vector and tells you the length.

Review

- Creating a new script
- Using R as a calculator
- Assigning values to variables
- Performing algebra on numeric variables

Introduction to R code adapted from: http://johnmuschelli.com/intro_to_r/index.html