Welcome to R!

Nicholas A. Del Grosso

July 22, 2018

Running R from the Terminal (Mac, Linux) or Command Window (Windows)

Open a terminal window and start the R program:

If R is installed and available, you'll see some copywrite information on the screen and a prompt to start typing R code into, an environment called the **R Console**. You're now working in R!

R as a Calculator

Operators

R is a fully-functional calculator; it recognizes all of the math operators for doing simple arithmetic, for example (+, -, *, /, ^). Typing any arithmetic problem and pressing Enter will return the answer. Try it out!

```
3 + 4
```

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

```
5 * -20
```

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

```
2 ^ 5
```

```
## [1] 32
```

It comes with a lot of math **functions**, as well, including trigonometry. These functions are "called" using parentheses:

```
sin(2)
```

```
## [1] 0.9092974
```

```
cos(5)
```

```
## [1] 0.2836622
```

```
atan(pi)
```

```
## [1] 1.262627
```

```
sqrt(9)
```

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

You can combine multiple operations on the same line:

```
tan(10) * 100 + 3
```

```
## [1] 67.83608
```

You can also saving the output of a line to a variable name with the left arrow symbol:

```
x <- 3 + 5
data <- sin(pi) * 10
x
```

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

These variable names can be used in the future:

```
x * data
```

```
## [1] 9.797174e-15
```

Vectors

A full sequence of values can also be saved to a single variable name as a **vector**. These values are combined together using the **c()** function:

```
aa <- c(1, 2, 3, 4)
aa
```

```
## [1] 1 2 3 4
```

Math operations on vectors work on each value individually. This saves a lot of typing and ensures consistency across data:

```
aa ^ 2
```

```
## [1]  1  4  9 16
```

```
sin(aa)
```

```
## [1]  0.8414710  0.9092974  0.1411200 -0.7568025
```

Statistics Functions

As R is made to be a statistics environment, it comes with a wealth of statistics functions. From the start, it makes several available: **mean()**, **sd()**, **var()**, **median()**, etc. These functions accept vectors as inputs:

```
mean(c(1, 5, 7))
```

```
## [1] 4.333333
```

```
sd(c(1, 2, 3))
```

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

Exercises

Just to get our feet wet, practice converting these word problems into R code:

Math Exercise

Situation: Jack has 8 apples, Jill has 3 apples, and Ben has 20 apples.

1. Assign the number of apples of each person to a variable named after the person.
2. Altogether, how many apples do the three people have?
3. Does R care if Jack's name is capitalized or not?
4. What is the square root of Ben's number of apples?

Statistics Exercise

Situation: In a reaction time experiment with 5 trials, a subject had the following 5 reaction times (in milliseconds): 200, 250, 225, 230, 260.

1. Assign the sequence of reaction times to a variable as a vector.
2. What is subject's mean reaction time?
3. What is the standard deviation of their reaction times?
4. Convert the reaction times to seconds from milliseconds and save these new times in a new variable.
5. What is the sum of the reaction times, in seconds?
6. How many trials were done by the subject? (Hint: it is the same as the *length* of the vector.)
7. *Sort* the reaction times in ascending order.

Function Discovery

Understanding What a Function Does

When you have questions about what a function does, you give the function's name to the **help()** function:

```
help(mean)
```

This brings up a screen with lots of information about the function, including what it does, how to use it, and what additional arguments the function can accept. To exit the screen, press the **q** key and you will return to R console.

Because we often need help, R also let the question mark symbol, to save on the number of key strokes:

```
?mean
```

Finding a Function you Need

The double-question mark is used for *finding* functions. It doesn't always find useful results, and we will discuss more robust strategies for finding functions in the rest of the workshop, but it is good for a quick look if you can guess the function name but aren't exactly sure how it is spelled:

```
??median
```

The functions are listed next to the package they are stored in. If the package has been already imported into your function library (for example, the *stats* and *base* packages are loaded by default), you can simply type the function name. If not, you will need to import the package to your library first by using the **library()** function:

```
library("psych")
```

Notice that the package name is placed inside quotation marks–this indicates that the word "psych" is data, not a variable name. (Note: The quotation marks are not required by the library() function, but is a good practice and habit that will be helpful for later sections–many other functions will require it.)

Once the psych package has been loaded, it is available as a variable and all of its functions can be called directly.

Exercises

1. What does the **length()** function do?
2. What function reverses a vector?

3. What package is the **describe()** function in?
4. Use the describe() function on the reaction time data from the Statistics Exercise
5. What does the **psych** library do?
6. Vignettes are documents that have more descrptive help. Try seeing psych's intro vignette with vignette("intro", "psych") and learn more about the range of possibilities that R can do with a single package!