SPEC REU R Resources: Intro to R - Calculations & Logical Operations

Alix Ziff, Gaea Morales, Zachary Johnson

Summer 2022

Walk Through Work: Arithmetic and Logical Operators

These guided exercises will walk us through the need-to-know R basics. We'll start with arithmetic and logical operations in R.

Arithmetic in `R`

You can use `R` as a calculator!

	Operator	Example
Addition	`+`	`2+4`
Subtraction	`-`	`2-4`
Multiplication	`*`	`2*4`
Division	`/`	`4/2`
Exponentiation	`^ or **`	`2^4`
Square Root	`sqrt()`	`sqrt(144)`
Absolute Value	`abs()`	`abs(-4)`
Finding the Remainder	`%%`	`2%%4`
Division w/ round down	`%/%`	`2%/%4`

Examples:

```
4*9
```

```
## [1] 36
```

```
sqrt(144)
```

```
## [1] 12
```

```
4^5
```

```
## [1] 1024
```

```
4 ^ 5
```

```
## [1] 1024
```

Note that spaces between operators do not matter. So `4 + 5` is the same as `4+5`. We usually add spaces to make the code more legible. Note, however, that spaces *do* matter when defining character values. We will talk about character values in the next `R` walkthrough.

Just like any regular calculator, you have to pay attention to the order of operations! Example:

```
6 * 8 - sqrt(7) + abs(-10) * (4/5)
```

```
## [1] 53.35425
```

```
6 * (8 - sqrt(7)) + abs(-10) * (4/5)
```

```
## [1] 40.12549
```

Basic R Operations

Beyond operations, R is also capable of calculating basic numerical descriptions.

Let's say we have the following vector:

```
r_fun <- rnorm(20)
# You can see now in your Global Environment, under 'Values', the r_fun vector.
```

Note: rnorm() is a command that generates a vector of n values that is normally distributed. We will learn more about it in later modules. For now, just know that there are many ways to generate numbers in R, and we could have also simply used concatenate (`c()`) to manually define the values in the vector.

	Operator	Example
Sum	`sum()`	`sum(r_fun)`
Mean	`mean()`	`mean(r_fun)`
Median	`median()`	`median(r_fun)`
Quantile	`quantile()`	`quantile(r_fun)`
Minimum	`min()`	`min(r_fun)`
Maximum	`max()`	`max(r_fun)`
Variance	`var()`	`var(r_fun)`
Standard Deviation	`sd()`	`sd(r_fun)`
Summary	`summary()`	`summary(r_fun)`
Sort	`sort()`	`sort(r_fun)` or `sort(r_fun, decreasing = TRUE)`

The `summary()` command will print out (all at once) the min, max, mean, median, and quantiles. The `sort()` command will reorder the values in the set (ascending by default, but you can also do descending by typing a comma after the vector name, followed by the text 'decreasing = TRUE' within the parentheses).

Logical operators

Arithmetic operations will yield a numerical output. (Correctly specified) Logical operations yield one of two results: `TRUE` (or `T`) or `FALSE` (`F`). Logical operators are incredibly helpful for subsetting data, any type of exploratory analysis, data cleaning and/or visualization task.

	Operator
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`
`x` and `y`	`x & y`
`%in%`	Testing whether a value is contained within a set.

Here are some simple examples:

```
4 > 2
```

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

```
4 <= 2
```

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

```
5 == 2^2
```

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

The examples above are very straightforward because we already know the result. The true value of logical operators lies in their ability to be applied across large amounts of data.

Let's take a set (or collection of numbers)–in `R`, the technical term is a vector–and use logical operators on them.

```
2 == c(2, 3, 4)
```

```
## [1]  TRUE FALSE FALSE
```

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

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

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

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

Question: What to you think is the output of the following operation?

```
2 = 4
```

And the following?

```
2 == 4
```

```
2 = 4
```

```
## Error in 2 = 4: invalid (do_set) left-hand side to assignment
```

We get an error, because `R` reads the single equal sign as an assignment operator (i.e., an alternative to <-, which we use when creating objects to be saved in the Global Environment)

```
2 == 4
```

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

We get "FALSE", because `R` interprets the two equal signs as a logical question, i.e., "is 2 exactly equal to 4?"