Intro to R and Simulations

Becky Tang

02-19-2025

R is a calculator

```
1 + 2
```

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

```
3 *4
```

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

```
5^2
```

```
## [1] 25
```

```
exp(2)
```

```
## [1] 7.389056
```

```
factorial(4)
```

```
## [1] 24
```

```
choose(5, 3)
```

```
## [1] 10
```

R respects PEMDAS:

```
1+1/2
```

```
## [1] 1.5
```

```
(1+1)/2
```

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

Booleans

Booleans/logicals can be formed with (in)equality symbols:

```
1 > 2
```

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

```
2 >= -1
```

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

```
3 == 4
```

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

Variables

Storing/saving values into variables is achieved by using the syntax `<variable name> <- <value>`. Note that the variable name cannot begin with a number! The result of a stored variable is not displayed to the user. Type out the variable name after storing to explicitly show the values.

```
a <- 1
a
```

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

Vectors

Create vectors of values (i.e. a list of values) using the concatenate `c()` function:

```
v <- c(1, 4, 16, 9)
v
```

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

The syntax `a:b` creates a sequence of integers from `a` to `b`.

```
c <- 1:4
c
```

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

R is vectorized language, so it operates element-wise in vectors. Also, under the hood, booleans/logicals TRUE/FALSE are treated as 1/0:

```
v + c
```

```
## [1]  2  6 19 13
```

```
v >= 3
```

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

```
sum(v >= 3)
```

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

Functions

A lot of the commands we have been working with are functions. You can tell a command is a function if it has parentheses (e.g. sum() and c()). A function usually takes in some argument(s) as input and returns something back as output. We can write custom functions using the function `function()`. We need to specify the name of the function, what it expects as input, as well as what to output.

Birthday problem, generalized

Suppose a group of k people each choose a number randomly with replacement from a list of n distinct numbers. The probability of at least one match in the group is 1 minus the probability of no matches:

$$1 - \binom{n}{k} \frac{k!}{n^k}$$

We can write this as a function in R:

```
prob_match <- function(n, k){
  1 - choose(n, k) * factorial(k)/(n^k)
}
```

Note that the function name is prob_match, and the arguments/inputs are called `n` and `k`. The output is the probability of at least one match for the specified values of `n` and `k`. We can now use this function. For example, the probability of at least one match among 3 people choosing a number between 1 and 40 is:

```
prob_match(365, 25)
```

```
## [1] 0.5686997
```

Approximating probabilities

In the following, simulation flipping a fair coin once:

```
sides <- c("H", "T")
sample(sides, size = 1)
```

```
## [1] "T"
```

We can approximate probabilities by counting up repeatedly simulating an experiment, counting up how many times a favorable outcome occurred, and dividing by the total number of experiments we did. This is easily done using the `replicate()` function:

```r
flips <- replicate(1000, sample(sides, size = 1))
sum(flips == "H")/1000
```

```
## [1] 0.502
```