Modes and Data Structures

What is a "mode"?

In `R`, individual elements (the smallest unit that we store) have a "mode", which describes the type of quantity they describe. Possible modes include:

- Integer
- Numeric (Floating Point, Double)
- Character (String)
- Logical (Boolean)
- Complex

We usually don't need to tell `R` what the mode should be when we define something. It guesses from what we provide.

We can use the `str` and `typeof` functions to learn what the mode of a variable we have defined is.

Numeric (Floating Point, Double)

```
x <- 1.1
str(x)
```

```
 num 1.1
```

```
typeof(x)
```

```
[1] "double"
```

```
mode(x)
```

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

Character (String)

```
x <- "a"
str(x)
```

```
 chr "a"
```

```
typeof(x)
```

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

```
mode(x)
```

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

Note: Characters can include more than one element, e.g. `x <- "abc"`.

Logical (Boolean)

```
x <- TRUE
str(x)
```

```
 logi TRUE
```

```
typeof(x)
```

```
[1] "logical"
```

```
mode(x)
```

```
[1] "logical"
```

The logical mode can take on values `TRUE` and `FALSE`, which can be abbreviated `T` and `F`.

Note: For this reason, naming variables `T` or `F` is discouraged.

Integer?

```
x <- 1
str(x)
```

```
 num 1
```

```
typeof(x)
```

```
[1] "double"
```

```
mode(x)
```

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

If R has to guess whether a number is an integer or a numeric, it will default to numeric.

Integer!

We actually do need to tell R the mode when we want to define an integer. A way to do that is to apply the function `as.integer` to the integer we provide.

```
x <- as.integer(1)
str(x)
```

```
 int 1
```

```
typeof(x)
```

```
[1] "integer"
```

```
mode(x)
```

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

Vectors

Vectors are collections of elements that share the same mode.

The length of a vector describes the number of elements in a vector.

In fact, everything we've seen so far was a vector of length 1!

```
x <- 1
str(x)
```

```
 num 1
```

Creating a Vector

We can construct vectors from multiple elements using the c function, where c stands for **concatenate**.

```
x <- c(1, 5, 2)
```

```
str(x)
```

```
 num [1:3] 1 5 2
```

```
x
```

```
[1] 1 5 2
```

Determining the Number of Elements in a Vector

The `length` function, when applied to a vector, returns the number of elements in a vector.

```
length(x)
```

```
[1] 3
```

Viewing an Element of a Vector

```
x[1]
```

```
[1] 1
```

```
x[2]
```

```
[1] 5
```

```
x[3]
```

```
[1] 2
```

Viewing Elements of a Vector

```
x[1:2]
```

```
[1] 1 5
```

```
x[c(1, 3)]
```

```
[1] 1 2
```

```
x[-2]
```

```
[1] 1 2
```

```
x[-c(1, 3)]
```

```
[1] 5
```

Growing a Vector

Unlike some other languages, R allows you to make a vector longer or make it shorter.

```
x <- c(x, 4)
```

```
x
```

```
[1] 1 5 2 4
```

Shortening a Vector

```
x <- x[1:3]
```

```
x
```

```
[1] 1 5 2
```

Replacing an Element of a Vector

```
x[2] <- 5.1
```

```
x
```

```
[1] 1.0 5.1 2.0
```

Looping Over Elements of a Vector

It is common that we may want to apply a function to one element of a vector at a time.

```
for (i in 1:length(x)) {
  x[i] <- i
}
```

```
x
```

```
[1] 1 2 3
```