# LECTURE 1: The R Language

## Basic actions:

**Initialize a vector:**

- `x <- 4`
- `y <- 1:10`

**Concatenate elements in the same vector (elements belonging to the same class) or different vectors together:**

- `x <- c(2,7,1)`
- `y <- c("cat", "dog", "hamster")`
- `z <- c(TRUE, FALSE, FALSE, TRUE)`
- `w <- c(x, y, z)`

```
x <- c(11, 3, 4, 9)
x[2]
```

**Call elements of a vector by their index:**

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

**Simple functions:**

- `class(x)` tells the class of the vector `x` (numeric, character, or logical)

```
x <- 3
class(x)
```

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

```
y <- "cat"
class(y)
```

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

```
z <- TRUE
class(z)
```

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

```
w <- c(x,y,z)
class(w)
```

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

- `length(x)` tells the length of the vector `x`

```r
x <- c(3, 8, 16)
length(x)
```

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

```r
x <- seq(from = 0, to = 20, by = 2)
x
```

**Sequences:**

```
##  [1]  0  2  4  6  8 10 12 14 16 18 20
```

The parameter `by` has 1 as default value, so that

```r
y <- seq(1, 10)
y
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

## Exercises

### Exercise 1

**Predict the output of this:**

```r
x <- 1:6
x - c(1,2)
```

If we print x, we get a sequence from 1 to 6: `1 2 3 4 5 6`

By subtracting c(1, 2) this is what happens: `1-1 2-2 3-1 4-2 5-1 6-2`

So that the output will be the following: `0 0 2 2 4 4`

Let's give a try:

```r
x <- 1:6
x-c(1,2)
```

```
## [1] 0 0 2 2 4 4
```

```r
y <- 2^x
log(y, 2)
```

**Predict the output of this:**

```
## [1] 1 2 3 4 5 6
```

```
log(y, y)
```

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

Since `x <- 1:6`, the output of **`y <- 2^x`** will result from this:

```
2^1   2^2   2^3   2^4   2^5   2^6
```

Which gives: `2 4 8 16 32 64`.

Let's check:

```
y <- 2^x
y
```

```
## [1]  2  4  8 16 32 64
```

While the **`log(y, 2)`** output is given by

```
log2(2) log2(4) log2(8) log2(16) log2(32) log2(64)
```

which corresponds to `x` as it comes out the following: `1 2 3 4 5 6`

Let's check:

```
log(y, 2)
```

```
## [1] 1 2 3 4 5 6
```

As regards **`log(y, y)`** instead, this is more trivial. Since y is both the first and seocnd argument, the two arguments have the same length, and so this is what happens:

```
log2(2) log4(4) log8(8) log16(16) log32(32) log64(64)
```

Which gives you back the following:

```
1 1 1 1 1 1
```

Let's check again:

```
log(y, y)
```

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

**Exercise 2**

**Write an instruction that produces a numerical vector containing the squares of the even numbers from 2 to 60**:

```r
x <- seq(from=2, to=60, by=2)^2
x
```

```
##  [1]    4   16   36   64  100  144  196  256  324  400  484  576  676  784  900
## [16] 1024 1156 1296 1444 1600 1764 1936 2116 2304 2500 2704 2916 3136 3364 3600
```

**Exercise 3**

```r
x <- 1:10
log(x^2)-2*log(x)
```

**Check numerically, on the integers between 1 and 10, that indeed $log(x^2) = 2log(x)$:**

```
##  [1] 0 0 0 0 0 0 0 0 0 0
```

**Exercise 4**

```r
x
```

**Check numerically, on integers between 1 and 10, that indeed $log(e^x) = x$:**

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```r
log(exp(1)^x, exp(1))
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

**Exercise 5**

```r
log(x, 2)-(log(x, 10)/log(2, 10))
```

**Check numerically, on the integers between 1 and 10, that indeed $log_2(x) = log_{10}(x)/log_{10}(2)$:**

```
##  [1]  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00 -4.440892e-16
##  [6]  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00  0.000000e+00
```