# COMP226: Slides 02

## Variables, vectors, and lists in R

**Rahul Savani**

rahul.savani@liverpool.ac.uk

## Overview

- **Variable assignment**
- **Some R basics** - data types, `is` and `as`
- **Vectors**
- **Lists**
- Getting **help within R**

## Variable assignments

- Variables can be assigned with either `<-` or with `=`
- I mainly use `<-` due to historical reasons

`variables.R`:

```
a <- 100
b <- 80
print(a+b)
```

```
> source('variables.R')
[1] 180
```

## Basic data types in R

- **character** (strings)
- **logical** (booleans - TRUE/FALSE)
- **numeric** (numbers)
- **factors** (categories - used a lot in statistics; we won't use them much)

## Checking the data type with "is"

```
> is.character(213)
[1] FALSE
> is.numeric(213)
[1] TRUE
> is.character("Hello")
[1] TRUE
```

```
> x <- 5; y <- 10; x > 3
[1] TRUE

> bool <- x > 3 && y < 9; bool
[1] FALSE
> is.logical(bool)
[1] TRUE
```

## Converting types with "as"

```
> as.logical(1)
[1] TRUE
> as.logical(0)
[1] FALSE

> as.character(0)
[1] "0"
> as.numeric("0")
[1] 0
```

## Vectors

```
> a <- c(1,7,6,8)
> is.vector(a); length(a)
[1] TRUE
[1] 4
```

Actually **single numbers are vectors**:

```
> 3
[1] 3

> is.vector(3); length(3)
[1] TRUE
[1] 1
```

So the function `length` itself returns a vector of length 1!

## Creating vectors: combine

`c` stands for **combine** and creates vectors from shorter vectors:

```
> a <- c(1,7,6,8)
> a
[1] 1 7 6 8

> a <- c(a, c(1,2))
> a
[1] 1 7 6 8 1 2
```

## Creating vectors: seq

`seq` is used to **generate regular sequences**

```
> seq(1,10)
[1]  1  2  3  4  5  6  7  8  9 10
```

A shorthand for this very common type of sequence uses `":"`

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

`seq` is more powerful though:

```
> seq(from=23,to=32,by=3.2)
```

## Accessing elements of vectors

```
> a <- c(1,7,6,8)

> a[1]
[1] 1

> a[5]
[1] NA

> a[2:4]
[1] 7 6 8

> a[4:2]
[1] 8 6 7
```

```
> source('vectors.R',echo=TRUE)

> a <- c(1,7,6,8)

> b <- seq(from=2,length.out=4,by=2)

> print(b)
[1] 2 4 6 8

> print(a+b) # operation is done component-wise
[1]  3 11 12 16

> print(2*a + a/b) # again component-wise
[1]  2.50 15.75 13.00 17.00

> print(a[1:3] + b[4:2])
[1]  9 13 10
```

## Lists

- A list is an important and simple construct in R
- Vector elements must be the **same** type; list element types may differ

```
> c(1,"hello") # 1 will be converted to the string "1"
[1] "1"     "hello"

> list(1,"hello")
[[1]]
[1] 1

[[2]]
[1] "hello"
```

## Accessing elements of a lists

Elements can be **accessed by name**, but are also ordered and can **accessed by index**

```
> l <- list(a=17,b=20); l
$a
[1] 17

$b
[1] 20

> names(l); l$a; l[[1]] # note the double square brackets
[1] "a" "b"
[1] 17
[1] 17
```

## Searching for information in R

Within R, use a **question mark** before a method name to find out what something does, e.g.

```
> ?seq
```

The question mark is actually shorthand for the `help` function. So we could also write

```
> help(seq)
```

Sometimes you need to use quotes, e.g.

```
> ?"for"
```

## Help with a package

We will use **packages of functions** a lot, e.g. the `TTR` package

```
> help(package=TTR)
```

**To use a package, it must be installed**

# The example function

Try the following:

```
> example(seq)
```

Now try

```
> example(persp)
```

## Help when you don't know what your looking for

If you don't know what you are looking for you can use `help.search()` to do a Google-style search through R's documentation. E.g.,

```
> help.search("moving average")
```

A shorthand for this is

```
> ??"moving average"
```

## Advanced searching

For advanced regular-expression based searching, see

```
> ?apropos
```

For example:

```
> apropos("^as\\.[a-c]")

 [1] "as.array"                      "as.array.default"
 [3] "as.call"                       "as.character"
 [5] "as.character.condition"        "as.character.Date"
 [7] "as.character.default"          "as.character.error"
 [9] "as.character.factor"           "as.character.hexmode"
[11] "as.character.numeric_version"  "as.character.octmode"
[13] "as.character.POSIXt"           "as.character.srcref"
[15] "as.complex"
```