Working with R

Marcin Krzystanek
PhD student at Cancer Systems Biology group, CBS
Technical University of Denmark
R is prepared to handle objects with large amounts of data.

Simple numerical objects in R (building blocks) are vectors.

In fact, even the simple: \( x \leftarrow 1 \), can be thought of like a vector with one element.

\[
\begin{align*}
> & \quad x \leftarrow 1 \\
> & \quad \text{is.vector}(x) \\
> & \quad [1] \; \text{TRUE}
\end{align*}
\]
R is prepared to handle objects with large amounts of data.

One could get an idea about size of an object without looking at it.

```r
> x <- 1:20
> length(x)
[1] 20
```

The functions `length()`, `dim()` and `str()` return information on the size or structure and object.

```r
> str(x)
int [1:20] 1 2 3 4 5 6 7 8 9 10 ...
```
More on Vectors

Elements in a vector can be accessed individually.

> x[1]  #Prints first element
> x[1:10]  #Prints first 10 elements
> x[c(1,3)]  #Prints element 1 and 3

Most functions expect an object as argument, rather than individual numbers.

> mean(1,2,3)  #Replies '1'
> y <- c(1,2,3)
> mean(y)  #Replies '2'
You can use conditionals as indexes!

```R
> x <- 1:10  # Data to play with
> x > 5      # Prints Boolean vector
> x[x > 5]   # Elements greater than 5
```

You can use the `which()` function for even greater control, and the `grep()` function to search for complete or partial strings.
The Recycling Rule

The *recycling rule* is an important concept for vector algebra in R.

When a vector is too short for a given operation, the elements are recycled and used again.

Examples of vectors that are too short:

```r
> x <- c(1,2,3,4)
> y <- c(1,2)     #y is too short
> x + y           #Returns '2,4,4,6'
```
Important object types in R

**vector** – "A series of numbers"

**matrix** – "Table of numbers"

**data.frame** – "More 'powerful' matrix (can store different classes at the same time)"

**list** – "Collections of other objects"

**string** – “A series of consecutive characters"
Data Matrices

Matrices are created with the `matrix()` function.

```r
> m <- matrix(1:12, nrow=3)
```

This produces something like this:

```r
> m
[1,]   1   4   7  10
[2,]   2   5   8  11
[3,]   3   6   9  12
```
Matrices also recycle

The *recycling rule* still applies:

```r
> m <- matrix(c(2,5), nrow=3, ncol=3)
```

Gives the following matrix:

```r
> m
```

```
[,1] [,2] [,3]
[1,]  2  5  2
[2,]  5  2  5
[3,]  2  5  2
```
Indexing Matrices

For vectors we could specify one index vector like this:

```r
> x <- c(2, 0, 1, 5)
> x[c(1, 3)]    # Returns '2' and '1'
```

For matrices we have to specify two vectors:

```r
> m <- matrix(1:3, nrow=3, ncol=3)
> m[c(1, 3), c(1, 3)]    # Ret. 2x2 matrix
> m[1,]    # First row as vector
```
The **apply()** function 'applies' another function along a specified dimension.

```r
> apply(m,1,sum)       #Sum of rows
> apply(m,2,sum)       #Sum of columns
> apply(m,1,mean)      #Mean of rows
```

Apply has cousins like **lapply()**:

```r
>f <- as.list(as.data.frame(m)) #Converting
to a list

> lapply(f, sum) #Works for a list of
#objects - returns a list
```
List Objects

A list is like a vector, but it can store anything, even other objects!

Lists are created with the `list()` function:

```r
>L1 <- list(1:10, c("d", "f"))
>L1[[2]]  #Returns "a" "b"
```

```r
>L2 <- list(a=1:10, b=c("d", "f"))
>L2$b     #Returns "d" "f"
```
Data frames work like a list of vectors of identical length.

Data frames could be created like this:

```r
> g <- rep(2, 5)  # Creates vector of 2s (length 5)
> j <- 1:5        # Another vector of length 5
> f <- data.frame(g, j)
> f               # Vectors g and j became columns
```

The demonstration dataset `USArrests` is a data.frame.

```r
> data(USArrests)  # Activate dataset
> str(USArrests)   # View the structure
```
Data.frame has some functionalities of both a Matrix and a List

Standard algebra, add to each element:
>USArrests + 100

Indexing data frames (various ways):
>USArrests[2,] #Possible in Matrix
>USArrests["Alaska",] #Possible in Matrix
>USArrests[,1] #Possible in Matrix
>USArrests[, "Murder"] #Possible in Matrix
>USArrests$Murder #Possible in List
>USArrests[["Murder"]]] #Possible in List

Average arrests across the States:
>apply(USArrests,2,mean)