Basic syntax

Arithmetic operators

```
x + y   # Addition
x - y   # Subtraction
x * y   # Multiplication
x / y   # Division
x ^ y   # Raise to a power
x ** y  # Raise to a power (old style /still not deprecated/ translated to ^ in the parser.)
x %% y  # Modulus - remainder after the division
x %/% y # Integer Division

x, y: numeric or complex vectors or objects which can be coerced to such, or other objects fo
```

For documentation write ?Arithmetic in the R console:

```
> ?Arithmetic
> help("Arithmetic")
```

Using example we can run the examples in the end of the documentation:

```
> example("Arithmetic")
```

Using apropos we can look for similar function names:

```
> apropos("sq")
[1] "chisq.test" "dchisq"     "pchisq"     "qchisq"     "rchisq"
[6] "sqrt"       "sQuote"
```

Using find we can find from which package is the function.

```
> find("sqrt")
[1] "package:base
```

We can only search functions from the already loaded packages.

```
> find("simple.lm")
character(0)
> library(UsingR)
Warning: package 'UsingR' was built under R version 4.0.3
Loading required package: MASS
Loading required package: HistData
Loading required package: Hmisc
Loading required package: lattice
Loading required package: survival
Loading required package: Formula
Loading required package: ggplot2

Attaching package: 'Hmisc'
The following objects are masked from 'package:base':

    format.pval, units

Attaching package: 'UsingR'
The following object is masked from 'package:survival':

    cancer
> find("simple.lm")
[1] "package:UsingR"
```

Asignement operators

For documentation:

```
> ?assignOps
```

There are 3 different assignment operators:

```
> x = 5
> y <- 5  # Recommended
> 5 -> z
> x; y; z # Prints
```

Differences between them:

They have different operator precedence (look at ?Syntax).

= has two meanings:

```
<ul>
        <li>operator: assignment operator</li>
        <li>syntax token: named argument passing in a function call</li>
    </ul>
```

```
> x = 20
> mean(x = 3); x
[1] 3
[1] 20
> mean (x <- 3); x # Not a good idea
[1] 3
[1] 3
```

Syntax

- R is case sensitive

```
> A <- 5
> a
Error in eval(expr, envir, enclos): object 'a' not found
```

- R is (dynamic language) not typified

```
> a <- 5
> a <- 5.4
> a <- "string"
```

- Valid names consist of letters, numbers and the dot or underline character.

Functions

- R can have default arguments and the arguments can be matched by position or by name.

Printing values

You can print an object just by typing its name, because R is wrapping that object name within the **print** command, so the following lines of code are identical:

```
> a
[1] "string"
> print(a)
[1] "string"
```

R uses the idea of generic functions, so **print** function looks for the attribute class of the object and the class type shows **print** how to generate the output.

print gives some options for formatting the output

- removing the quotes from the output

```
> print(a)
[1] "string"
> print(a, quote = FALSE)
[1] string
```

- determine how many digits from the output to be shown

```
> a <- 3145.429357453; a
[1] 3145.429
> print(a, digits = 10)
[1] 3145.429357
```

Also we can redirect the output to a file using **sink** and then return it back to the console

```
> sink("myoutput.txt")
```

Working directory

getwd returns the absolute path to the current working directory.

```
> getwd()
[1] "/home/daniel/git/r/Basic syntax"
```

The working directory tells R where to look for files and where to create files. So the file that you have just created in the previous example **myoutput.txt** will be created in this directory.

If you want to change the working directory you can use **setwd** function

```
> setwd("<path-to-directory>")
```

Objects

Everything in R is an object.

ls prints all names of the objects in the global environment

```
> ls()
[1] "a" "A" "x" "y" "z"
```

rm removes objects from the current environment

```
> rm(x, y, z)
> ls()
[1] "a" "A"
```

You can also remove all the objects from the current environment using

```
> rm(list = ls())
> ls()
character(0)
```

Additional notes

- Comments start with #
- For execution on Linux (in terminal): `Rscript main.R`