Basic Syntax

2020

Comments are written after `#` sign.

```
# Comment
```

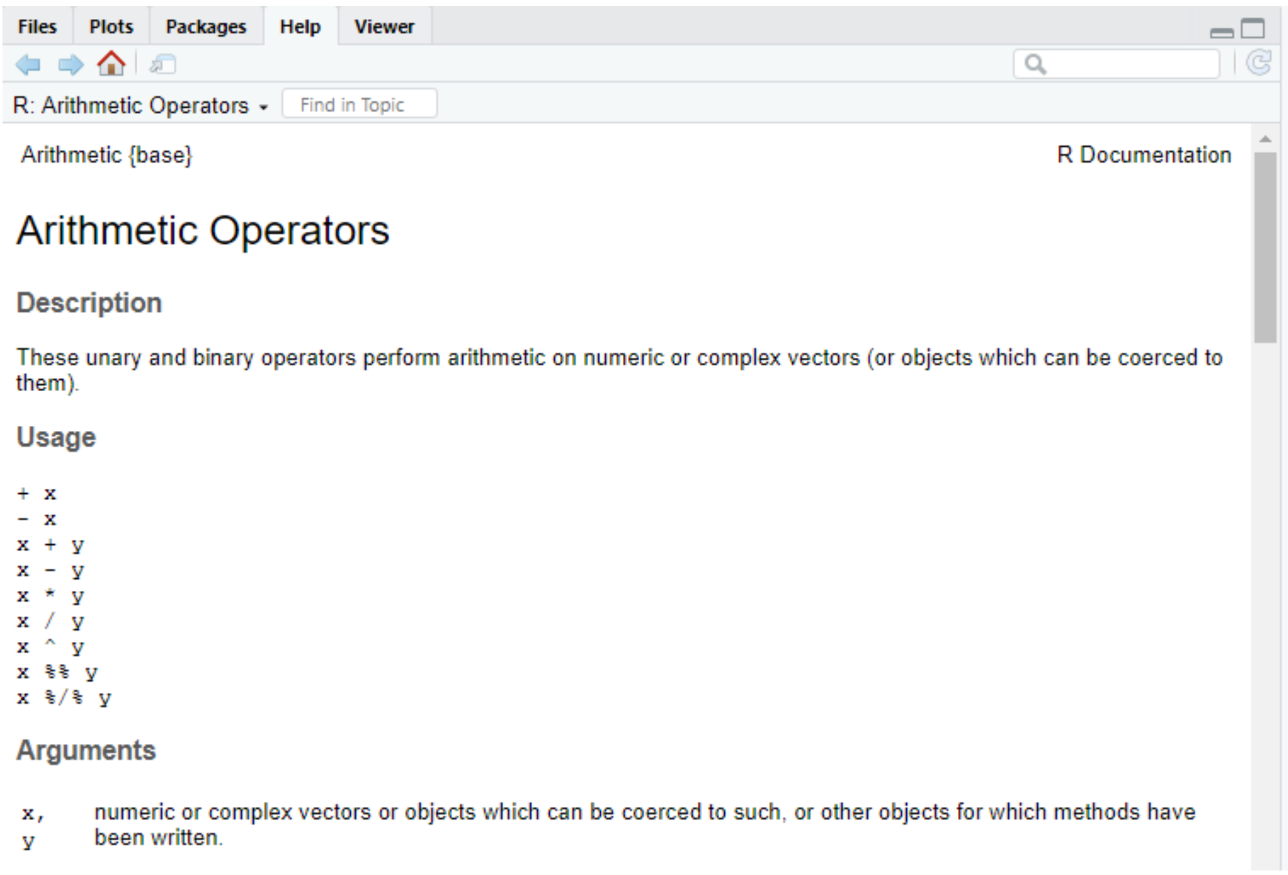
Arithmetic operators

```
> 2 + 3    # Addition
Output: [1] 5
> 2 + 3 + 5
Output: [1] 10
> 2 - 4    # Subtraction
Output: [1] -2
> 2 * 3    # Multiplication
Output: [1] 6
> (2 + 3) * 4
Output: [1] 20
> 11 / 3    # Division
Output: [1] 3.666667
> 11 %/% 3  # Integer Division
Output: [1] 3
> 11 %% 3   # Modulus - remainder after the division
Output: [1] 2
> 2^3      # Raise to a power
Output: [1] 8
> 2**3     # Raise to a power
Output: [1] 8
```

Where can we find some documentation?
What is the difference between `^` and `**`?

Getting help

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



Arithmetic {base} R Documentation

Arithmetic Operators

Description

These unary and binary operators perform arithmetic on numeric or complex vectors (or objects which can be coerced to them).

Usage

```
+ x
- x
x + y
x - y
x * y
x / y
x ^ y
x %% y
x %/% y
```

Arguments

`x, y`	numeric or complex vectors or objects which can be coerced to such, or other objects for which methods have been written.

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"
```

Assignment operators

Where can we find some documentation?

```
> ?assignOps
```

There are 3 different assignment operators

```
> x = 5
> y <- 5   # Recommended
> 5 -> z
> x; y; z # Prints
[1] 5
[1] 5
[1] 5
```

What are the differences between them?

1. They have different operator precedence `?Syntax`
2. `=` has two meanings
   - operator: assignment operator
   - syntax token: named argument passing in a function call

```
> x = 20
> mean(x = 3); x
[1] 3
[1] 20
> mean (x <- 3); x
[1] 3
[1] 3
```

Syntax

R is case sensitive

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

R is not typified

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

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"
```

`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 <- 3145.429357; a
[1] 3145.429
> print(a, digits = 10)
[1] 3145.429357
> print(a, digits = 5)
[1] 3145.4
> print(a, digits = 2)
[1] 3145
```

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

```
> sink("myoutput.R")
> a <- "string"; a
[1] "string"
> print(a, quote = FALSE)
[1] string
> a <- 3145.429357; a
[1] 3145.429
> print(a, digits = 10)
[1] 3145.429357
> print(a, digits = 5)
[1] 3145.4
> print(a, digits = 2)
[1] 3145
> sink()
```

Workng directory

`getwd` returns the absolute path to the current working directory.

```
> getwd()
```

```
[1] "E:/Coursses/FMI/SEM/R/01.IntroductionToR"
```

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.R` will be created in this directory.

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

```
> setwd("E:/Coursses/FMI/SEM/R/00.Introduction")
```

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

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