

Preliminaries in R

Function calls

Function calls

In general, function calls in R take the following structure:

```
## Generic code (this won't run)  
function_name(formal_argument_1 = named_argument_1,  
               formal_argument_2 = named_argument_2,  
               [etc.])
```

A function call forms a complete R expression, and the output will be the result of running `print` or `show` on the object that is output by the function call.

Function calls

Here is an example of this structure:

```
print(x = "Hello world")
```

```
## [1] "Hello world"
```

Function calls

A handwritten diagram illustrating the components of a function call. The code `print(x = "Hello world")` is written in black ink. The word `print` is underlined with a black line, and a black arrow points from the label "function name" to it. The `x =` is underlined with an orange line, and an orange arrow points from the label "formal argument" to it. The string `"Hello world"` is underlined with a red line, and a red arrow points from the label "actual argument" to it.

In this example, we're **calling** a function with the **name** `print`. It has one **argument**, with a **formal argument** of `x`, which in this call we've provided the **named argument** `"Hello world"`.

Function calls

The **arguments** are how you customize the call to an R function.

For example, you can use change the named argument value to print different messages with the `print` function:

```
print(x = "Hello world")
```

```
## [1] "Hello world"
```

```
print(x = "Hi Fort Collins")
```

```
## [1] "Hi Fort Collins"
```

Some functions do not require any arguments. For example, the `getRversion` function will print out the version of R you are using.

```
getRversion()
```

```
## [1] '4.4.1'
```

Function calls

Some functions will accept multiple arguments. For example, the `print` function allows you to specify whether the output should include quotation marks, using the `quote` formal argument:

```
print(x = "Hello world", quote = TRUE)
```

```
## [1] "Hello world"
```

```
print(x = "Hello world", quote = FALSE)
```

```
## [1] Hello world
```


Function calls

Arguments can be **required** or **optional**.

For a required argument, if you don't provide a value for the argument when you call the function, R will respond with an error. For example, `x` is a **required argument** for the `print` function, so if you try to call the function without it, you'll get an error:

```
print()
```

```
Error in print.default() : argument "x" is  
  missing, with no default
```

Function calls

For an **optional argument** on the other hand, R knows a **default value** for that argument, so if you don't give it a value for that argument, it will just use the default value for that argument.

For example, for the `print` function, the `quote` argument has the default value `TRUE`. So if you don't specify a value for that argument, R will assume it should use `quote = TRUE`. That's why the following two calls give the same result:

```
print(x = "Hello world", quote = TRUE)
```

```
## [1] "Hello world"
```

```
print(x = "Hello world")
```

```
## [1] "Hello world"
```

Often, you'll want to find out more about a function, including:

- Examples of how to use the function
- Which arguments you can include for the function
- Which arguments are required versus optional
- What the default values are for optional arguments.

You can find out all this information in the function's **helpfile**, which you can access using the function ?.

For example, the `mean` function will let you calculate the mean (average) of a group of numbers. To find out more about this function, at the console type:

```
?mean
```

This will open a helpfile in the “Help” pane in RStudio.

Function helpfiles

Helpfile for ~mean~

mean (base)

Arithmetic Mean

Description
Generic function for the (trimmed) arithmetic mean.

Usage
mean(x, ...)

Default S3 method:
mean(x, trim = 0, na.rm = FALSE, ...)

Arguments
x An R object. Currently there are methods for numeric, logical vectors and [date](#), [date-time](#) and [time interval](#) objects. Complex vectors are allowed for trim = 0, only.
trim the fraction (0 to 0.5) of observations to be trimmed from each end of x before the mean is computed. Values of trim outside that range are taken as the nearest endpoint.
na.rm a logical value indicating whether NA values should be stripped before the computation proceeds.
... further arguments passed to or from other methods.

Value
If trim is zero (the default), the arithmetic mean of the values in x is computed, as a numeric or complex vector of length one. If x is not logical (coerced to numeric), numeric (including integer) or complex, NA_real_ is returned, with a warning.
If trim is non-zero, a symmetrically trimmed mean is computed with a fraction of trim observations deleted from each end before the mean is computed.

References
Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.
See Also
[weighted.mean](#), [mean.POSIXct](#), [colMeans](#) for row and column means.

Examples
x <- c(8:10, 50)
xs <- mean(x)
c(m, mean(x, trim = 0.10))

Usage
Required and optional arguments

Arguments
Descriptions of arguments

Value
What the function outputs

Examples
Examples of using the function

The helpfile includes sections giving the function's **usage**, **arguments**, **value**, and **examples**.

Function helpfiles

Helpfile for 'mean'

mean (base)

Arithmetic Mean

Description
Generic function for the (trimmed) arithmetic mean.

Usage
mean(x, ...)

Default S3 method:
mean(x, trim = 0, na.rm = FALSE, ...)

Arguments
x As an object. Currently there are methods for numeric, logical vectors and `data.frame` and `data.table` objects. Complex vectors are allowed but `length(x) > 0` only.
trim The fraction (0 to 0.5) of observations to be trimmed from each end of a before the mean is computed. Values of trim outside that range are taken as the nearest endpoint.
na.rm a logical value indicating whether NA values should be stripped before the computation proceeds.
... further arguments passed to or from other methods.

Value
If trim is zero (the default), the arithmetic mean of the values in x is computed, as a numeric or complex vector of length one. If x is not logical (coerced to numeric), NA values (including integer) or complex, `NA_real_` is returned, with warning.
If trim is non-zero, a symmetrically trimmed mean is computed with a fraction of trim observations deleted from each end before the mean is computed.

References
Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.
See Also
`setMethod`, `trim`, `mean.POSIXct`, `colMeans` for row and column means.

Examples
x = c(0, 10, 10)
m = mean(x)
c(m, mean(x, trim = 0.10))

Usage
`mean(x, trim = 0, na.rm = FALSE)`

Value
What the function outputs

Arguments
Descriptions of arguments

Examples
Examples of using the function

Usage
`mean(x, trim = 0, na.rm = FALSE)`

Value
What the function outputs

Arguments
Descriptions of arguments

Examples
Examples of using the function

You can figure out which arguments are **required** and which are **optional** in the Usage section of the helpfile.

Operators

There's one class of functions that looks a bit different from others. These are the infix **operator** functions.

Instead using parentheses after the function name, they usually go *between* two arguments.

One common example is the + operator:

```
2 + 3
```

```
## [1] 5
```

Operators

There are operators for several mathematical functions: $+$, $-$, $*$, $/$.

There are also other operators, including **logical operators** and **assignment operators**, which we'll cover later.