Chapter 4: Writing your own functions

Contents

1 Rules to write a function: **1**

1.1 Make and check body 2

1.2 Generalization 2

1.3 Apply 2

2 Overview of some useful functions in R **3**

2.1 Functions to convert to integers 3

2.2 Integer operators 3

2.3 Some common functions 4

2.4 Functions on 1 vector 4

2.5 Functions on 2 vectors or more 4

2.6 Statistical functions 4

3 Exercises **5**

Syntax:

name_function <- function(arg1, arg2,. . . .){Expression}

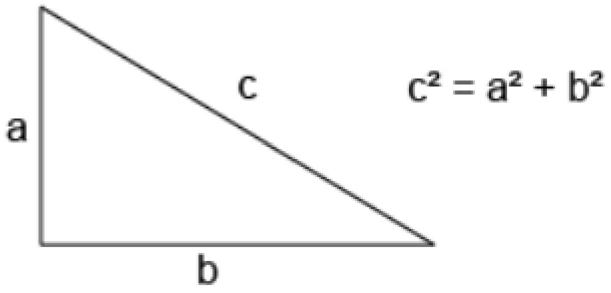
Expression is an R expression that uses the arguments (*arg1*, *arg2*,. . .) to calculate a value.

1 Rules to write a function:

1. Make and check the body of the function.
2. If the body of the function is OK, generalize it.
3. Apply the function.

Example

Write a function *Pythagoras* to calculate the length of hypotenuse from length of the legs of a right-angled triangle (Theorem of Pythagoras).



Apply your function when a = 1 and b = 1 and apply the function when a = 3 and b = 4.

1.1 Make and check body

Step 1: Make the body of the function for a specific case and check whether the body of the function is OK:

```r
# Step 1: Make and check the body of the function
a <- 3
b <- 4
c <- sqrt(a^2 + b^2)
c
```

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

1.2 Generalization

Step 2: If the body of the function is OK, generalize it:

```r
# Step 2: Write the function
Pythagoras <- function(a,b)
{
  c <- sqrt(a^2 + b^2)
  print(a)
  print(b)
  c
}
```

1.3 Apply

Step 3: Apply the function.
How to apply?

- Highlight and submit the function. Then R will recognize it as an R function.

- Apply the created function for other values of the argument(s).

```r
# Step 3: Apply the function
Pythagoras(a=1,b=1)
```

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

## [1] 1.414214
```

```r
Pythagoras(a=3,b=4)
```

```
## [1] 3
## [1] 4

## [1] 5
```

Remark:

1. The last command executed is the **return value** of the function. This can be forced by:
    - using `return` function;
    - using `print` function to force the printout.
2. If you want to obtain several components as result of your function, you have to make use of a list statement.

```r
# Use of the list function in your Pythagoras function
Pythagoras <- function(a,b)
{
  c <- sqrt(a^2 + b^2)
  list(a=a, b=b, hypothenusa=c)
}

# Apply your function
Pythagoras(a=1,b=1)
```

```
## $a
## [1] 1
##
## $b
## [1] 1
##
## $hypothenusa
## [1] 1.414214
```

```r
Pythagoras(a=3,b=4)
```

```
## $a
## [1] 3
##
## $b
## [1] 4
##
## $hypothenusa
## [1] 5
```

2 Overview of some useful functions in R

Some standard functions

2.1 Functions to convert to integers

```r
x <- -3.526
```

Function	Description	Result in R for x = -3.526
`round(x)`	rounds to nearest integer	-4
`trunc(x)`	leaves out the decimal part	-3
`floor(x)`	takes the nearest integer which is smaller than x	-4
`ceiling(x)`	takes the nearest integer which is larger than x	-3

2.2 Integer operators

```r
x1 <- 21
x2 <- 5
```

Function	Description	Alternative way	Result in R for x1 = 21 and x2 = 5
`%/%`	Integer divide	`floor(x1/x2)`	4
`%%`	Modulu reduction	`x1-floor(x1/x2)*x2`	1

2.3 Some common functions

`abs` (computes the absolute value), `log`, `sqrt` (computes the square root), `exp`, `sin`, `cos`, `tan`, `acos`, `asin`, `atan`, `cosh`, `sinh`, `tanh`...
`log(x, base)` has a second (optional) argument, i.e. the base number (default e)

2.4 Functions on 1 vector

Function	Description	Result in R for `vec <- 1:5`
`length(vec)`	Returns the length of an object	5
`sum(vec)`	Returns the sum of all the values present in `vec`	15
`prod(vec)`	Returns the product of all the values present in `vec`	120
`cumsum(vec)`	Returns a vector whose elements are the cumulative sums of `vec`	1, 3, 6, 10, 15
`cumprod(vec)`	Returns a vector whose elements are the cumulative products of `vec`	1, 2, 6, 24, 120
`max(vec)`		5
`min(vec)`		1
`cummax(vec)`	Returns a vector whose elements are the cumulative maxima of `vec`	1, 2, 3, 4, 5
`cummin(vec)`	Returns a vector whose elements are the cumulative minima of `vec`	1, 1, 1, 1, 1
`range(vec)`	Returns a vector containing the minimum and maximum	1, 5
`sort(vec)`		1, 2, 3, 4, 5
`rev(vec)`		5, 4, 3, 2, 1

2.5 Functions on 2 vectors or more

`pmax(vec1, vec2...)`, `pmin(vec1, vec2...)`, `max(vec1, vec2...)`, `min(vec1, vec2...)`, etc.

Function	Description	Result in R
`pmax(c(1,7,3), c(3,4,5))`	Returns a vector with the parallel maxima of the argument vectors	3, 7, 5
`max(c(1,7,3), c(3,4,5))`	Returns the maximum of all the values present in their arguments	7

2.6 Statistical functions

`mean(vec)`, `var(vec)`, `sd(vec)`

3 Exercises

1. Write a function which gives the most elementary statistics for a sample x: min, median, max, mean, sd and length. Apply your function on a vector x with values from 25 to 80.

2. Write a function fun1 which produces the text 'Non-negative number' if you apply fun1 to a positive number and 'negative number' if you apply fun1 to a negative number. You can make use of the `ifelse` function in R. Apply this function to the values 9 and -13:

Input	Desired output
`x <- 9; fun1(x)`	"Non-negative number"
`x <- -13; fun1(x)`	"Negative number"

3. Write a function to solve an equation of second degree ($ax^2 + bx + c = 0$). To solve this equation, first calculate $D = b^2 - 4ac$. In the case $D > 0$, there are two roots: $x_1 = \frac{-b+\sqrt{D}}{2a}$ and $x_2 = \frac{-b-\sqrt{D}}{2a}$.
   If possible, make also a plot of the function. Apply your function for the equation $-8x^2 + 6x + 4 = 0$.