w7

Irlanda Ayon-Moreno

4/15/2018

R Expressions

Compound expressions consist of simple expressions separated by semicolons or newlines, and grouped within braces.

Every expression in R has a value: the value of the last evaluated statement.

It is possible to have assignments within compound expressions and the values of the variables which this produces can be used in later expressions.

Conditionals

If-then-else

The condition is an expression that when evaluated returns a logical value of length one. In other words, whatever you pass as the input of the if clause, it has to be something that becomes TRUE or FALSE. If it's a vector, you'll get a warning message; if it's an NA, you'll get an error.

With simple expressions, hat can be written in one line, there's actually no need to use braces

structure:

```
if (this) {
  # do that
} else if (that) {
  # do something else
} else {
  #
}
```

Switch

If you find yourself using many if-else statements with identical structure for slightly different cases, you may want to consider a **switch** statement instead

It allows you to evaluate selected code based on position or name.

```
function(x, y, op) {
   switch(op,
     plus = x + y,
     minus = x - y,
     times = x * y,
     divide = x / y,
     stop("Unknown op!")
   )
}
```

Functions

Naming functions

Generally, function names should be verbs (and arguments should be nouns).

The following list provides some examples with different naming styles:

- `squareroot()`
- `SquareRoot()`
- `squareRoot()`
- `square.root()`
- `square_root()`

It is also important that you know which names are invalid in R:

- `5squareroot()`: cannot begin with a number
- `_square()`: cannot begin with an underscore
- `square-root()`: cannot use hyphenated names

In addition, avoid using an already existing name, e.g. `sqrt()`.

Sometimes you will find functions with names starting with a dot: `.hidden()`; this type of functions are hidden functions, meaning that the function won't be visible by default in the list of objects in your working environment.

Creating a function

- To define a new function in R you use the function `function()`.
- You need to specify a name for the function, and then assign `function()` to the chosen name.
- You also need to define optional arguments (i.e. inputs).
- And of course, you must write the code (i.e. the body) so the function does something when you use it

```
# anatomy of a function
some_name <- function(arguments) {
  # body of the function
}
```

- Generally, you give a name to a function.
- A function takes one or more inputs (or none), known as *arguments*.
- The expressions forming the operations comprise the **body** of the function.
- Usually, you wrap the body of the functions with curly braces.
- A function returns a single value.

Function Output

The value of a function can be established in two ways:

- As the last evaluated simple expression (in the body of the function)
- An explicitly **returned** value via `return()`

```r
f <- function() {
  if (!x) {
    return(something_short)
  }

  # Do
  # something
  # that
  # takes
  # many
  # lines
  # to
  # express
}
```

Function error

use `stop()` to stop the execution of the function & display an error message.

```r
circle_area <- function(radius = 1) {
  if (radius < 0) {
    stop("radius must be positive")
  }
  pi * radius^2
}
```

`stopifnot()`: it checks that each argument is TRUE, and produces a generic error message if not.

Dealing with missing values

Many functions in R like `sum()`, `mean()`, and `median()` have the so-called `na.rm` argument to specify if missing values should be removed before any computation this feature. We can take advantage of `na.rm = TRUE`:

```r
standardize <- function(x, na_rm = FALSE) {
  z <- (x - mean(x, na.rm = na_rm)) / sd(x, na.rm = na_rm)
  return(z)
}
```

Documenting Functions

label	meaning	description
`@title`	title	name of your function
`@description`	description	what the function does
`@param input`	parameter	describe `input` parameter
`@return`	output	what is the returned value

```r
#' @title Standardize
#' @description Transforms values in standard units (i.e. standard scores)
#' @param x numeric vector
#' @param na_rm whether to remove missing values
#' @return standardized values
#' @examples
#'   standardize(rnorm(10))
standardize <- function(x, na_rm = FALSE) {
  z <- (x - mean(x, na.rm = na_rm)) / sd(x, na.rm = na_rm)
  return(z)
}
```