

Introduction to `tester`

Gaston Sanchez

www.gastonsanchez.com

1 Introduction and Motivation

`tester` provides human readable functions to test characteristics of some common R objects. The main purpose behind `tester` is to help you validate objects, especially for programming and developing purposes (e.g. creating R packages)

Testing objects When we write a function, more often than not, we need to validate its arguments. In order to do so, we can use some of the already available functions in R that allow us to test whether objects have certain features. For instance, we can use `is.matrix(M)` to test if `M` is a matrix. Likewise, if you want to test if an object is a list, we can use the `is.list()` function.

The interesting part comes when we want to test for more specific characteristics, like testing if `M` is a numeric matrix, or test if a number is a positive integer, or maybe if it is a decimal number. Let's take the case in which we want to test whether an object is a character matrix. One way to do that would be to write something like this:

```
# test if object is a character matrix
object = matrix(letters[1:6], 2, 3)

if (is.matrix(object) & is.character(object)) TRUE else FALSE

## [1] TRUE
```

Now let's say we want to test if a given number is a positive integer:

```
# test if number is a positive integer
number = 1

if (number > 0 & is.integer(number)) TRUE else FALSE

## [1] FALSE
```

In this case, we know that `number = 1` but the test returned `FALSE`. The reason is that the number 1 is not an strict integer in R. Instead, we need to declare `number = 1L`. Now, if we test again we will get `TRUE`:

```
# test if number is a positive integer
number = 1L

if (number > 0 & is.integer(number)) TRUE else FALSE

## [1] TRUE
```

Easier tests If we just have a couple of functions, testing its arguments may not be a big deal. But when we have dozens or hundreds of functions, even if they are not in the form of a package, testing their arguments can be more complicated. Instead of writing expressions like the following one:

```
if (number > 0 & is.integer(number)) TRUE else FALSE
```

it would also be desirable to simply write something like this:

```
is_positive_integer(number)
```

This is precisely what **tester** allows us to do by providing a set of functions to test objects in a friendly way, following the so-called *literate programming* paradigm. Under this paradigm, instead of writing programs instructing the computer what to do, we write programs explaining humans what we want the computer to do. The advantage is that *when we read code, we should be able to do so as if we were reading a text*. In this sense, the goal of **tester** is twofold: 1) help you test objects, and 2) help you write more human readable code.

Here is another example. Suppose we want to check if a vector has missing values. One option to answer that question is to use the function `is.na()`:

```
# test for missing values
is.na(c(1, 2, 3, 4, NA))

## [1] FALSE FALSE FALSE FALSE TRUE
```

Depending on your goals, `is.na()` might be enough. But what if we just want to simply test if a vector has missing values? With **tester** now we can do that using the function `has_missing()`:

```
# test for missing values
has_missing(c(1, 2, 3, 4, NA))

## [1] TRUE

# or equivalently
has_NA(c(1, 2, 3, 4, NA))

## [1] TRUE
```

2 About tester

To use `tester` (once you have installed it), load it with the function `library()`:

```
# load package tester
library(tester)
```

2.1 Numbers

To test if we have number, as well as different types of numbers, we can use one of the following functions:

Testing Numbers	
Function	Description
<code>is_positive()</code>	tests if a number is positive
<code>is_negative()</code>	tests if a number is negative
<code>is_integer()</code>	tests if a number is an integer
<code>is_natural()</code>	tests if a number is a natural number
<code>is_odd()</code>	tests if a number is an odd number
<code>is_even()</code>	tests if a number is an even number
<code>is_positive_integer()</code>	tests if a number is a positive integer
<code>is_negative_integer()</code>	tests if a number is a negative integer
<code>is_decimal()</code>	tests if a number is decimal
<code>is_positive_decimal()</code>	tests if a number is a positive decimal
<code>is_negative_decimal()</code>	tests if a number is a negative decimal

2.2 Logical

To test if an object (or a condition) is `TRUE` or `FALSE`, we can use the following functions:

Testing Logicals	
Function	Description
<code>is_TRUE()</code>	tests if an object is <code>TRUE</code>
<code>is_FALSE()</code>	tests if an object is <code>FALSE</code>
<code>true_or_false()</code>	tests if is <code>TRUE</code> or <code>FALSE</code>

2.3 Vectors

To test if we have different types of vectors we can use the following functions:

Testing Vectors	
Function	Description
<code>is_vector()</code>	tests if an object is a vector
<code>is_numeric_vector()</code>	tests if an object is a numeric vector
<code>is_string_vector()</code>	tests if an object is a string vector
<code>is_logical_vector()</code>	tests if an object is a logical vector
<code>is_not_vector()</code>	tests if an object is not a vector

2.4 Matrices

Likewise, to test if we have different types of matrices we can use the following functions:

Testing Matrices	
Function	Description
<code>is_matrix()</code>	tests if an object is a matrix
<code>is_not_matrix()</code>	tests if an object is not a matrix
<code>is_numeric_matrix()</code>	tests if an object is a numeric matrix
<code>is_string_matrix()</code>	tests if an object is a string matrix
<code>is_logical_matrix()</code>	tests if an object is a logical matrix
<code>is_square_matrix()</code>	tests if an object is a square matrix
<code>is_rectangular_matrix()</code>	tests if an object is a rectangular matrix
<code>is_tall_matrix()</code>	tests if an object is a tall matrix
<code>is_wide_matrix()</code>	tests if an object is a wide matrix
<code>is_diagonal()</code>	tests if an object is a diagonal matrix
<code>is_triangular()</code>	tests if an object is a triangular matrix
<code>is_lower_triangular()</code>	tests if a matrix is lower triangular
<code>is_upper_triangular()</code>	tests if a matrix is upper triangular

2.5 Data Frame

To test if we have different types of data frames we can use the following functions:

Testing Data Frames	
Function	Description
<code>is_dataframe()</code>	tests if an object is a data frame
<code>is_numeric_dataframe()</code>	tests if an object is a numeric data frame
<code>is_string_dataframe()</code>	tests if an object is a string data frame
<code>is_factor_dataframe()</code>	tests if an object is a data frame of factors
<code>is_not_dataframe()</code>	tests if an object is not a data frame

2.6 Matrices and data frames attributes

Other functions related to matrices and data frames allows us to ask whether or not some properties are present:

Matrices and Data Frames attributes	
Function	Description
<code>has_dimension()</code>	tests if an object has dimension
<code>is_one_dim()</code>	tests if an object has one-dimension
<code>has_rownames()</code>	tests if an object row names
<code>has_colnames()</code>	tests if an object column names

2.7 Missing Values

For testing missing values, infinite values, not numbers, `tester` provides the following functions:

Testing Missing Values

Function	Description
<code>has_missing()</code>	tests if an object has missing values
<code>has_infinite()</code>	tests if an object has infinite values
<code>has_not_a_number()</code>	tests if an object has 'Not a Number'
<code>has_nas()</code>	tests if an object has NA, Inf, -Inf, NaN

2.8 Comparisons

Another interesting set of functions that come in **tester** are those for comparing purposes:

Comparison

Function	Description
<code>same_class()</code>	tests if two objects have the same class
<code>same_mode()</code>	tests if two objects have the same mode
<code>same_type()</code>	tests if two objects have the same type of
<code>same_length()</code>	tests if two objects have the same length
<code>same_dim()</code>	tests if two matrices (or data frames) have the same dimension
<code>same_nrow()</code>	tests if two matrices (or data frames) have the same number of rows
<code>same_ncol()</code>	tests if two matrices (or data frames) have the same number of columns

2.9 Other

tester comes with many more functions that will allow you to check—in a friendly way—whether some common R objects have certain characteristics. Some of the extra available functions are:

Other Tests

Function	Description
<code>is_tabular()</code>	tests if an object is a matrix or data frame
<code>is_multiple()</code>	tests if a number is multiple of a given number
<code>has_names()</code>	tests if an object has names
<code>list_of_vectors()</code>	tests if an object is a list of vectors
<code>list_of_numeric_vectors()</code>	tests if an object is a list of numeric vectors
<code>list_of_string_vectors()</code>	tests if an object is a list of string vectors
<code>list_of_logical_vectors()</code>	tests if an object is a list of logical vectors