

# testR – R language test driven specification

## Motivation

Every computer language that seeks widespread adoption and large user base requires a formal language specification. Language references, such as the ECMAScript reference for Javascript<sup>1</sup> are used to this end as these documents are relatively easy for programmers and virtual machine (VM) implementers to understand. But while such a reference can be easily created for a standardized language like JavaScript, creating such document for an evolving and community driven language like R is extremely time consuming and bound to lag behind the latest language development.

## Introducing testR

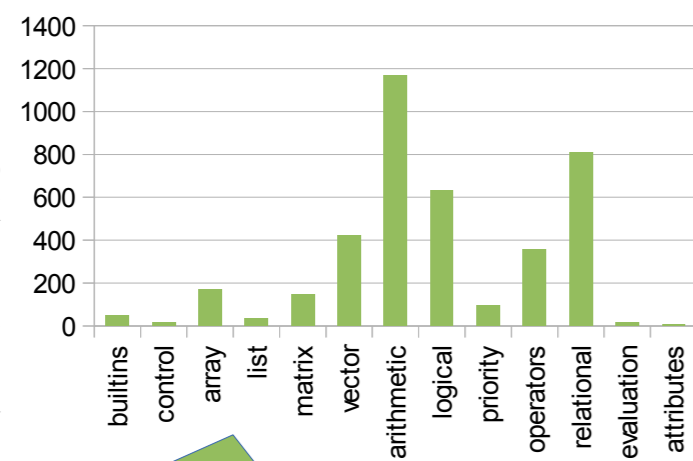
TestR provides a different approach to the specification problem. It consists of a growing number of relatively simple tests that aim to cover the behavior of the R language with all its features and corner cases exposed.

Tests are structured according to the language features and are written in such way they read like a manual, complete with code examples. Maintaining the suite in sync with latest language development is as easy as running the suite and attending to the failures.

TestR has also the potential to help VM developers. The suite can be used to measure GNU-R compatibility, or validate new commits in regression mode. It is simple and extensible enough to either be integrated to, or make a test suite for nearly any R VM greatly simplifying the development efforts.

## Current State

The test suite consists of 4K tests covering Vector, array, matrix and list data types, arithmetic, relational and logic operators, operator priorities, few builtins and some other language features, such as attributes. We support GNU-R, FastR and Renjin Vms for tests and benchmarking.



number of tests in different areas

## Implementation

A single TestR test consists of a header defining the test name and its properties and the test code, which is R code. A simple preprocessor allows automating repetitive tasks by simple generics.

TestR commands begin with #!. They can only appear in test headers.

Each test is given a name that describes its function, This allows the tests to easily form human readable description

When a generic is specified, a new test is created for each of its values. If multiple generics are present, all possible combinations of their values are created.

This is not true for dependent generics. The dependent generic changes each time its original generic does. In the example, RESULT is dependent generic and changes with the OP generic always providing the correct value for the result of the operation.

Generic usage, this will be replaced by the current value of the generic. Generics can also be used in the header commands as shown in the #!t check above.

```

#! testName
#!g OP = ( + # - # * # / )
#!g RESULT(OP) = ( 8 # 4 # 12 # 3 )
#!t @RESULT
a <- 6
b <- 2
a @OP b
    
```

test header

test body

Example header commands:

- g generic argument
- t expected result
- w expected warning
- e expected error
- o expected output
- t test on/off for different targets

New commands can be added by creating simple python functions.

While the generics can be used to create powerful super tests, to increase readability, these are not used and the bulk of the suite consists of simple tests aimed at a single feature. As an example the test below documents the inconsistent propagation of NAs in arithmetic operations:

```

#! NA propagation in arithmetic operators
#!g OP = (+ # - # * # /)
#!t NA NA
NA @OP NaN
NaN @OP NA
    
```

	Win32	Win64	Linux32	Linux64
NA NA	NA NA	NA NaN	NA NA	NA NaN

Another interesting failure found is the wrong default value for the exact argument for [[]] matching. Two tests and their outputs are shown below displaying the difference.

```

#! partial matching exact default value is NA
#!o TRUE
#!w partial match of 'b' to 'bar'
a <- list(foo=1, bar=2, foobar=3)
a[["b"]] == 2
    
```

```

a = list(foo=1, bar=2, foobar=3)
a[["b", exact=NA]] == 2
    
```

logical(0)

TRUE  
warning message:  
In a[["b", exact=NA]] : partial match of 'b' to 'bar'

## Framework

TestR itself is written in Python and great emphasis has been placed on its modularity. New targets (R VMs) can be added with a few lines as well as new modules to gather and analyze the results. TestR can be used in almost any test oriented task from regressions to profiling and benchmarking R on multiple VMs.

```

#! timer module example
#!g SIZE = (10 # 100 # 1000 # 10000 # 100000)
f <- function(a,b) {
  a + b * a / (b-a)
}
c <- 0
for (i in 1:@SIZE) {
  c <- f(c,c+i)
}
    
```

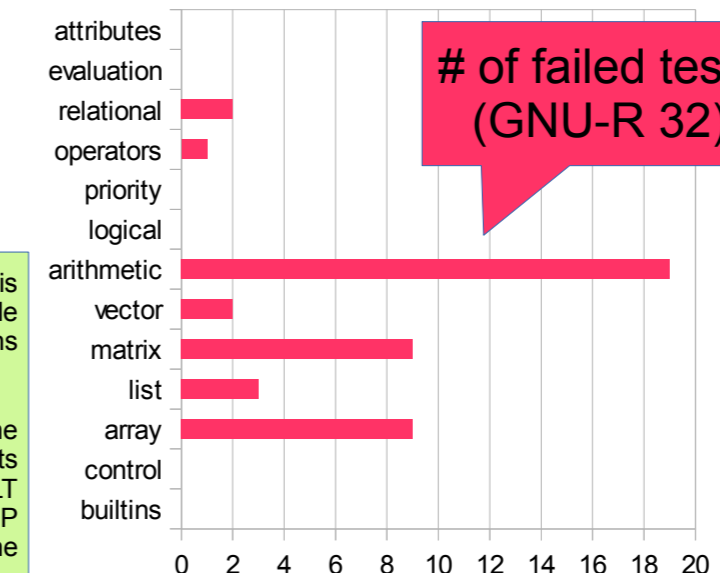
output of module timer

test name	gnur_linux	fastr	fastr_graal	renjin
timer module example [SIZE=10]	0.183993	1.242291	0.682417	3.248864
timer module example [SIZE=100]	0.186549	1.253723	0.701230	3.320019
timer module example [SIZE=1000]	0.189415	1.258490	0.711636	3.325598
timer module example [SIZE=10000]	0.209359	1.260419	0.725398	3.617592
timer module example [SIZE=100000]	0.452572	1.272221	0.733940	3.809238

## Findings

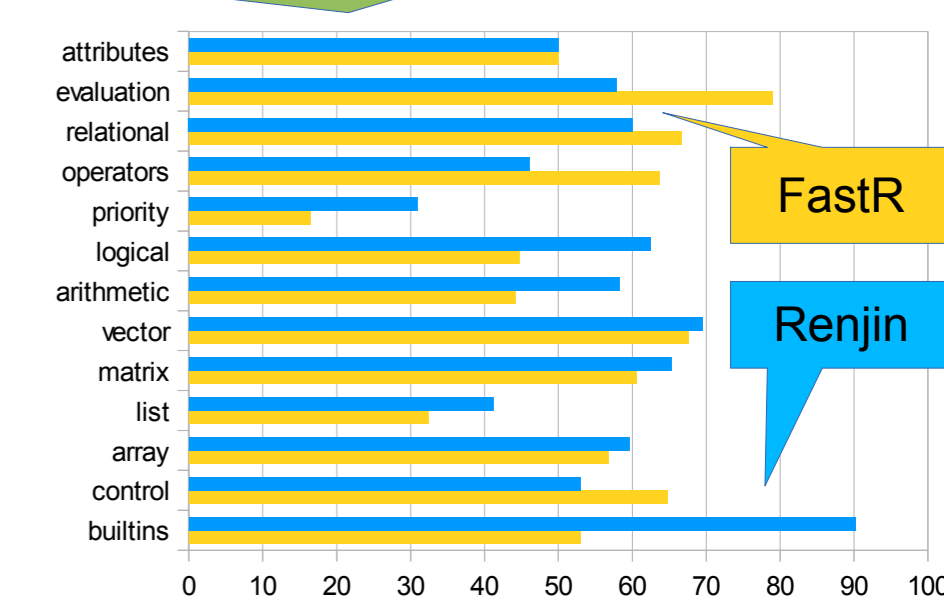
Since the tests are created to conform to the current R manual<sup>2</sup>, we present two different graphs: The first one shows the absolute number of tests failed (examples of these tests are displayed on this poster) by GNU-R (Linux64bit, version 2.15.2 (2012-10-26)) per test groups.

The second graph is percents of tests passed by our implementation, FastR (Linux) and Renjin (0.7.0) showing the degree of their implementation.



# of failed tests (GNU-R 32)

## % of passed tests for FastR & Renjin



## Conclusions

TestR is an approach to formal language specification using tests to map the features. It also provides compatibility and regression suite for the VM developers further helping with the adoption of the language.

Future work will include increasing the language coverage and possibly tests of non-core modules.

TestR and FastR can be obtained from:

<http://github.com/allr/testr.git>

<http://github.com/allr/fastr.git>

## References

1) ECMAScript reference can be obtained from <http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf>

2) R manual and reference can be obtained from: <http://cran.r-project.org/doc/manuals/r-release/R-lang.html>