TestR Overview

This document presents the infrastructure of the R version of testR, which is greatly different from the now outdated Python version (testr-py repo).

TestR can be obtained from a github repo https://github.com/allr/testr.

This document can be found also in the following repo: https://github.com/allr/documents.

Infrastructure

The general architecture of the testR system is presented on the figure below:

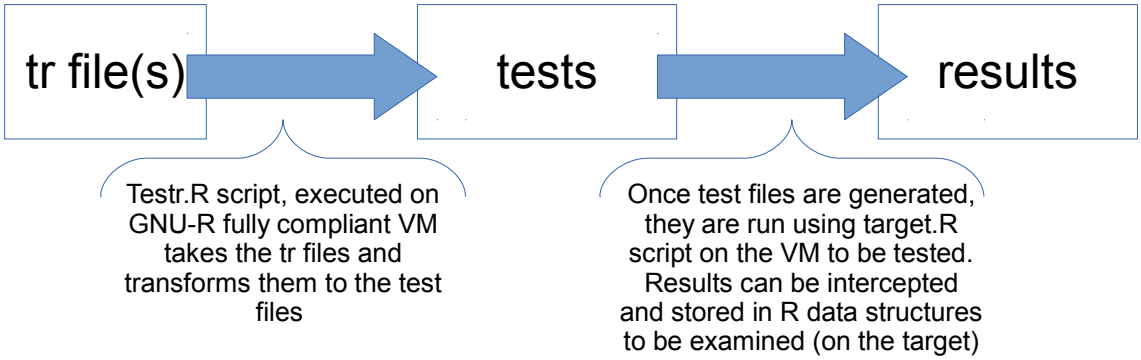

Illustration 1: Overview of testR system

A test is a call to a test function, defined in the target.R script. This function takes several arguments, notably the code of the test and expected output. The code of the test is expected to be plain R code as it would be directly executed and its result compared to the expected output. If the execution produces a warning, or an error, these too as specified as arguments.

An example of a final test follows:

```
test(id=17,
  {
      a = list(foo = 1, foolish = 2, bar = 3)
      a[["b", exact = NA]]
  },
  name = "partial matching with exact NA",
  o = 3,
  w =  "partial match"
)
```

The user should never write these tests directly. Instead a test template should be written and placed into a tr file. The test templates are rather similar. They too take the code of the test as an argument, the warnings produced and the expected outputs, however they differ notably in the following:

- test templates need not specify the output. If unspecified, during their translation to the tests the output is captured
- test templates need not specify the id, this is also automatically added by the testR generator

- and finally, the templates may use generics to create multiple final tests from a single test with very similar code.

Generics

A generic can only be specified in a test template. The generic specifies multiple values for which the test should be executed. Multiple generics will produce as many tests as there is permutations of their values. If a generic name appears in the code of the test, the name is replaced by the actual generic value. So for example the following test template:

```
test(name = "haha",
  g(a, 1, 2, 3, 4),
  g(b, c(1,2), c(2,3), c(3,4)),
  g(c, "+","-"),
  a %c% b
)
```

Creates 24 tests in total with a, c, and b in the code replaced with the scalar values, vectors and operators respectively. Note the %c% notation for replacement of operator generics and also note the absence of the output and test id. An example of the generated tests can be found below.

```
test(id=18,
  1 + c(1, 2),
  name = "haha [a = 1, b = c(1, 2), c = \"+\"]",
  o = c(2, 3)
)

test(id=19,
  1 - c(1, 2),
  name = "haha [a = 1, b = c(1, 2), c = \"-\"]",
  o = c(0, -1)
)
```

The generics can serve also more complex purposes, which are described in greater detail in the testR help itself.

Running the tests

When running the tests on the target vm (function runTests() in target.R) a directory containing the test files is supplied. The function runs all the tests in these files and returns TRUE if all pass, FALSE otherwise. If the data needs to be processed further, a callback function can be supplied that will be called upon execution of each single test, which can then build the preferred representation of the test execution.