Test your tests with R metaprogramming

Tom Taverner, Chris Campbell

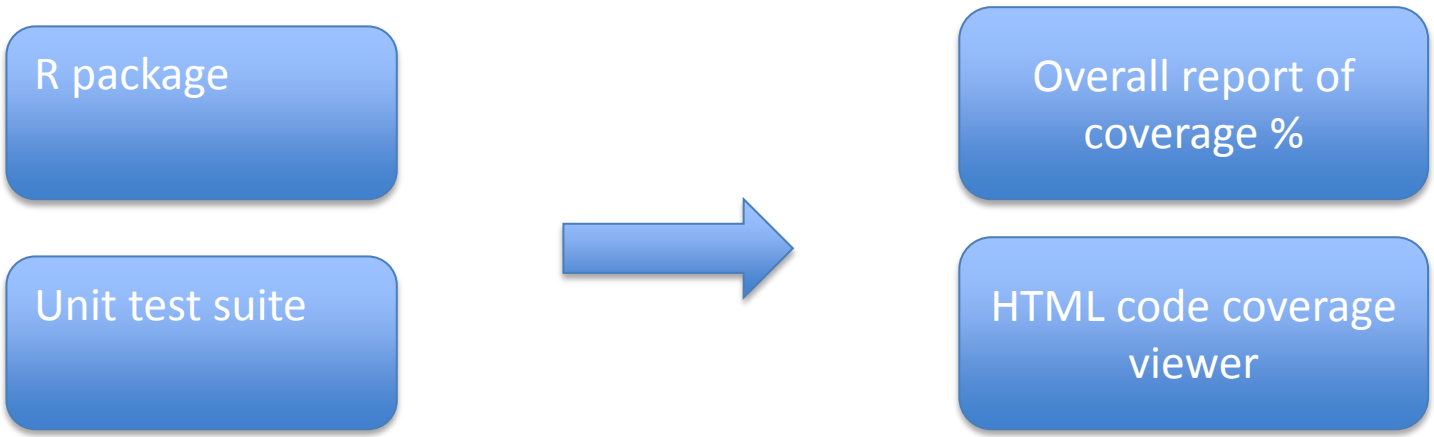
Mango Solutions

Two ideas for testing software

- Black box testing (RUnit and testthat) shows broken code
- Clear box testing shows faulty/dead/unreachable code

R based test coverage

- Code coverage is the degree to which code is tested by a particular test suite
- Want to find which code is not executed
- We developed a code coverage testing tool



Example

- Function ‘trapezium’ is read in but not called by the unit test suite

```
trapezium <- function(x, y)
{
    # xIndex : integer vector of indices
    xIndex <- 2:length(x)
    as.double(((x[xIndex] - x[xIndex - 1]) %*% (y[xIndex ] + y[xIndex -1])) / 2)
}
```

- In practise we found 90-100% coverage in production code
- Error handling code is especially hard to test

The instrumentation algorithm

- Rewrites code to put trace calls everywhere
- Uses R 3.0's alternate parser

```
y <- x + 10
```

1. Get parse table mapping symbols to unique ID

y	1
x	2

```
`_1` <- `_2` + 10
```

2. Replace symbols by UIDs

```
`_1` <- {trace(); `_2` + 10}
```

3. Insert trace calls

```
y <- {trace(2); x + 10}
```

4. Instrument trace calls and replace UIDs by symbols

Aims for the near future

- Put on Github
- Integrate with CI tools
- Take inspiration from Java tools (clover)
- Other types of coverage?