Testing for fun and profit

Rich FitzJohn
MRC Centre for Global Disease Analysis
Imperial College London

richfitz
reside-ic
mrc-ide
vimc

The take-home:

If testing feels like a chore, change how you do it

Testing in <1 minute

```
add <- function(a, b) {
  a + b
}
expect_equal(add(1, 2), 3)
```

How to draw an Owl.

"A fun and creative guide for beginners"

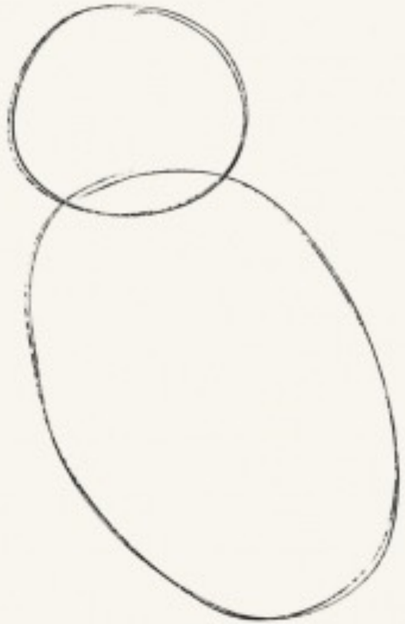


Fig 1. Draw two circles

Fig 2. Draw the rest of the damn Owl

```
   1 expect_equal_to_reference
   5 expect_gte
   6 expect_named
   6 expect_with_retry
   7 expect_missing_error
   9 expect_lte
  15 expect_s3_class
  15 expect_type
  15 expect_valid_json
  20 expect_lt
  51 expect_output
  53 expect_gt
  78 expect_equivalent
  93 expect_warning
  94 expect_setequal
 157 expect_length
 181 expect_message
 276 expect_match
 279 expect_silent
 549 expect_null
 773 expect_false
 941 expect_is
 982 expect_identical
1757 expect_true
2853 expect_error
8670 expect_equal
```

Only a few commands to learn

What sorts of test?

Unit test

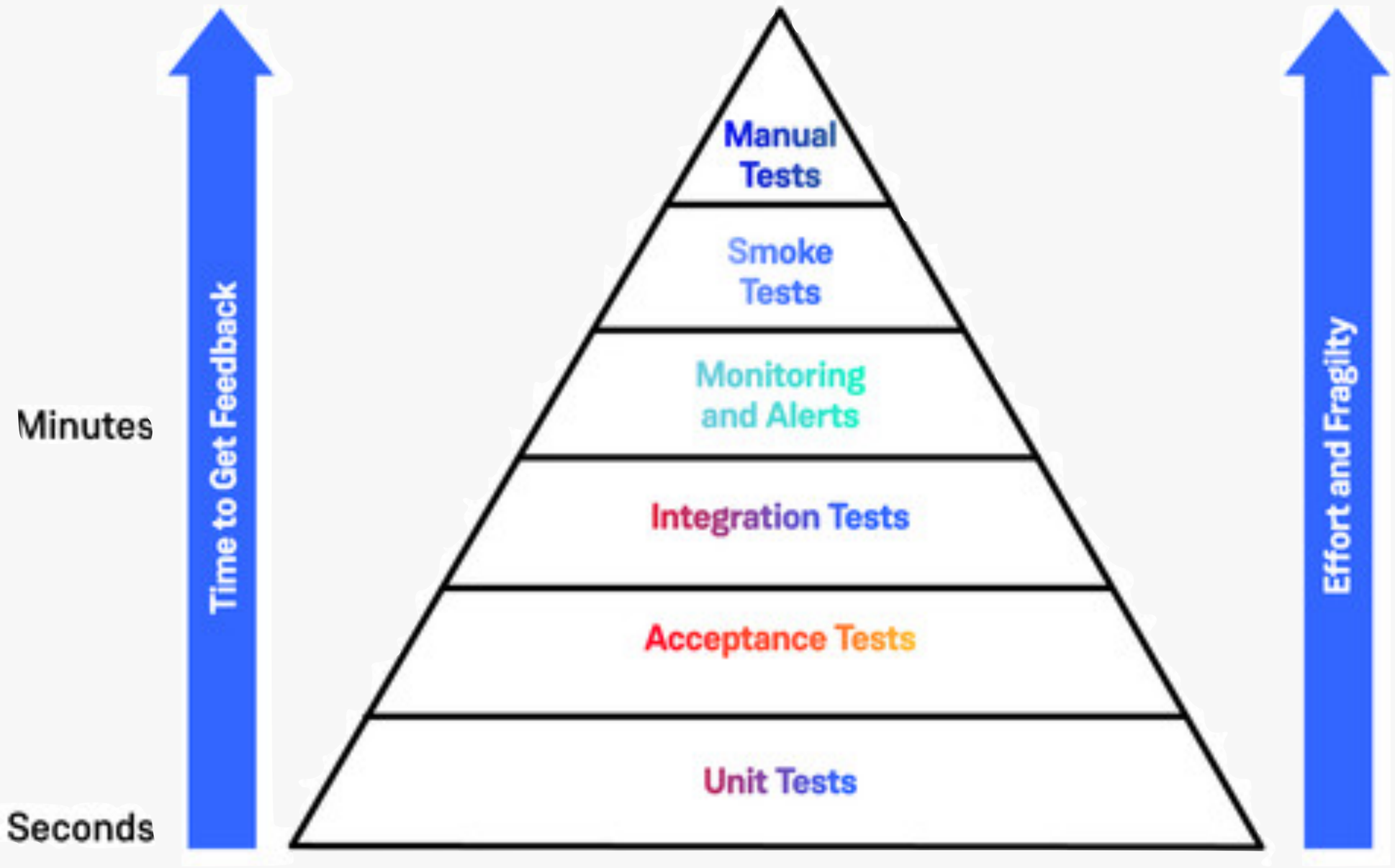
Acceptance test

Integration test

End-to-end test

Smoke test

Regression test



https://www.contino.io/insights/the-testing-pyramid

This still does not sound fun

Why test?
Better workflows
Mocking
Things that are hard

But why test?

code that is easy to test is
easy to understand
and easy to reuse
and easy to replace

```
data <- read.csv("input.csv")
mymodel <- function(a, b) {
  for (i in unique(data$group)) {
    fit <- long_running_fit(a, b, data)
    plot(y ~ x, data)
    if (fit$pvalue < 0.05) {
      lines(fit)
    }
  }
  saveRDS(fit, "c:/myfiles/fit.rds")
  lapply(fit, coef)
}
```

```r
data <- read.csv("input.csv")
mymodel <- function(a, b) {
  for (i in unique(data$group)) {
    fit <- long_running_fit(a, b, data)
    plot(y ~ x, data)
    if (fit$pvalue < 0.05) {
      lines(fit)
    }
  }
  saveRDS(fit, "c:/myfiles/fit.rds")
  lapply(fit, coef)
}
```

```
data <- read.csv("input.csv")
mymodel <- function(a, b) {
  for (i in unique(data$group)) {
    fit <- long_running_fit(a, b, data)
    plot(y ~ x, data)
    if (fit$pvalue < 0.05) {
      lines(fit)
    }
  }
  saveRDS(fit, "c:/myfiles/fit.rds")
  lapply(fit, coef)
}
```

```
data <- read.csv("input.csv")
mymodel <- function(a, b) {
  for (i in unique(data$group)) {
    fit <- long_running_fit(a, b, data)
    plot(y ~ x, data)
    if (fit$pvalue < 0.05) {
      lines(fit)
    }
  }
  saveRDS(fit, "c:/myfiles/fit.rds")
  lapply(fit, coef)
}
```

```
data <- read.csv("input.csv")
mymodel <- function(a, b) {
  for (i in unique(data$group)) {
    fit <- long_running_fit(a, b, data)
    plot(y ~ x, data)
    if (fit$pvalue < 0.05) {
      lines(fit)
    }
  }
  saveRDS(fit, "c:/myfiles/fit.rds")
  lapply(fit, coef)
}
```

```
function(...) {
  if (...) {
    if (...) {
      a1
    } else if (...) {
      a2
    } else {
      a3
    }
  } else if (...) {
    b
  } else {
    c
  }
}
```

The tool shapes the hand

How to incorporate testing into your workflows

1. Write a bunch of code
2. Play around with it interactively
3. Get your colleagues to try it out
4. Publish a number of papers
5. Realise that bugs have crept in
6. Start writing tests

First attempt at RUnit based unit tests

master v0.9-11 ··· v0.9-2

Browse files

Rich FitzJohn committed on 26 Oct 2010

1 parent 2df9ee5 commit 10e69a1e070a9b1cb50dd0f109b342dac483d412 patch diff

Showing 5 changed files with 237 additions and 0 deletions.

No Whitespace

27 diversitree-tests/runit-bisse.R

60 diversitree-tests/runit-geosse.R

```
@@ -0,0 +1,60 @@
## Test case code, based on Emma's tests.
test.geosse <- function() {
  tree <- read.tree(text="((((0:0.461876,(1:0.307717,(2:0.231825,3:0.231825):0.075892):0.154159):0.425922,((4:0.244819,5

  states <- c(0, 1, 0, 2, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0)
  names(states) <- c("0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14")

  starting.point.geosse(tree)
                                        #       sA       sB      sAB       xA       xB       dA       dB
                                        # 1.5788510 1.5788510 1.5788510 0.0000000 0.0000000 0.3157702 0.3157702

  pars <- c(0.9, 0.8, 0.1, 0.2, 0.3, 0.5, 0.6)
  lnL.7par <- make.geosse(tree, states)
  names(pars) <- argnames(lnL.7par)

  checkEquals(lnL.7par(pars), -23.71574, tolerance=1e-7)
  checkEquals(lnL.7par(pars, condition.surv=TRUE), -24.10259,
              tolerance=1.06e-7)

```

github.com/richfitz/diversitree/commit/10e69a

Install, once more, the correct version of g++
richfitz committed on 14 May 2015 ✓

Attempt to fix both C++11 and Fortran installation on travis ···
richfitz committed on 14 May 2015 ✗

Commits on May 13, 2015

Attempt to fix gfortan installation
richfitz committed on 13 May 2015 ✗

Use g++ 4.8 not 4.9
richfitz committed on 13 May 2015 ✗

Attempt to get fortran installed properly
richfitz committed on 13 May 2015 ✗

Install newer gcc on travis
richfitz committed on 13 May 2015 ✗

Remove util_lang_range.h in the hope that allows compilation
richfitz committed on 13 May 2015 ✗

Select C++11 compiler in the correct way
richfitz committed on 13 May 2015 ✗

Make early termination warning optional in grow_plant_to_size
richfitz committed on 13 May 2015

Install RcppR6 on travis
richfitz committed on 13 May 2015 ✗

Add travis
richfitz committed on 13 May 2015 ✗

github.com/traitecoevo/plant/commits/d5aebd

1. Write a bunch of code
2. Play around with it interactively
3. Get your colleagues to try it out
4. Publish a number of papers
5. Realise that bugs have crept in
6. Start writing tests

Turn your experimentation into tests

Turn your user requirements into tests

Turn your bug reports into tests

1. Create a trivial skeleton
2. Write a function and test it
3. Set up covr/codecov
4. Create a branch
5. Create a Minimal Viable Product (with tests)
6. Create a Pull Request
7. Justify all your coverage gaps
8. GOTO 4

The Beyoncé Rule

If you liked it, you should have put a test on it

increment.com/testing/testing-as-communication/

Tree: 42b130feda traduire / R / util.R Find file Copy path

richfitz Initial import 648724a 22 days ago

1 contributor

3 lines (3 sloc) | 56 Bytes Copy Raw Blame History

```
`%||%` <- function(a, b) {
  if (is.null(a)) b else a
}
```

Tree: 42b130feda traduire / tests / testthat / test-util.R Find file Copy path

richfitz Initial import 648724a 22 days ago

1 contributor

8 lines (7 sloc) | 182 Bytes Copy Raw Blame History

```
context("util")

test_that("null-or-value works", {
  expect_equal(1 %||% NULL, 1)
  expect_equal(1 %||% 2, 1)
  expect_equal(NULL %||% NULL, NULL)
  expect_equal(NULL %||% 2, 2)
})
```

github.com/reside-ic/traduire/tree/42b130

Conversation 2 | Commits 19 | Checks 2 | Files changed 21 | +4,144 −5

richfitz 22 days ago — Member

There's heaps more to the added here, though the upstream docs don't make it super easy. I am not 100% sure about the name (always feels a bit just going with the same name as the upstream package) but am coming up short on alternatives (my 2nd choice was "traduire", which at least has a good R sound to it).

The big drama with packaging this was getting v8 to see `Promise` - turns out that's only in recent V8 and debian ships with an ancient version.

Reviewers
r-ash

Assignees – assign yourself

Labels

Projects

github.com/reside-ic/traduire/pull/1

codecov bot commented 22 days ago • edited

Codecov Report

> ❗ No coverage uploaded for pull request base (`master@e5f578c`). Click here to learn what that means.
> The diff coverage is `100%`.

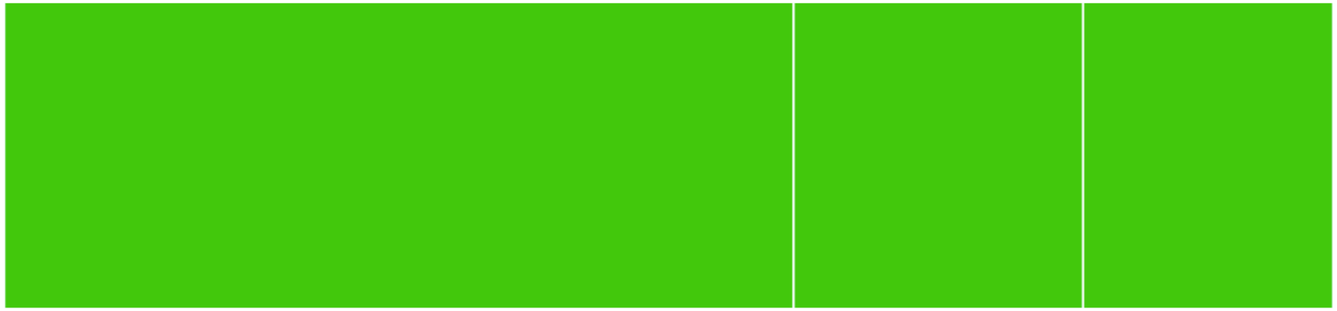
```
@@            Coverage Diff            @@
##              master     #1    +/-   ##
=========================================
  Coverage          ?   100%
=========================================
  Files             ?      2
  Lines             ?     29
  Branches          ?      0
=========================================
  Hits              ?     29
  Misses            ?      0
  Partials          ?      0
```

github.com/reside-ic/traduire/pull/1

codecov bot commented 6 days ago • edited

Codecov Report

> Merging #7 into master will **not change** coverage.
> The diff coverage is `100%`.

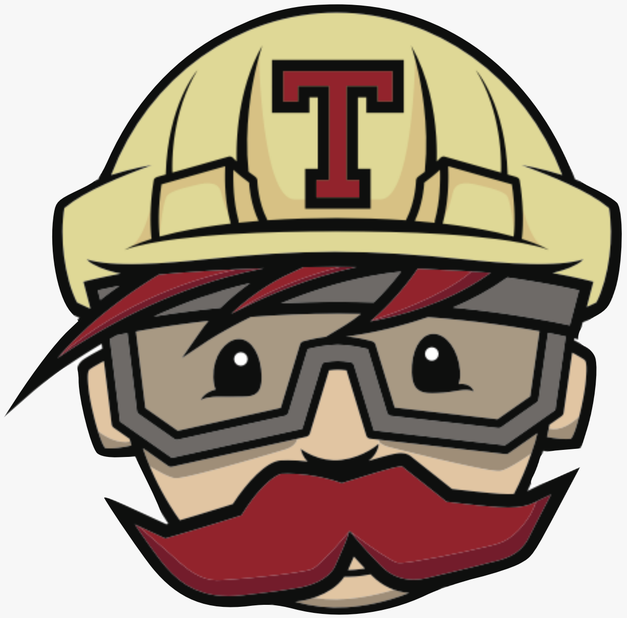


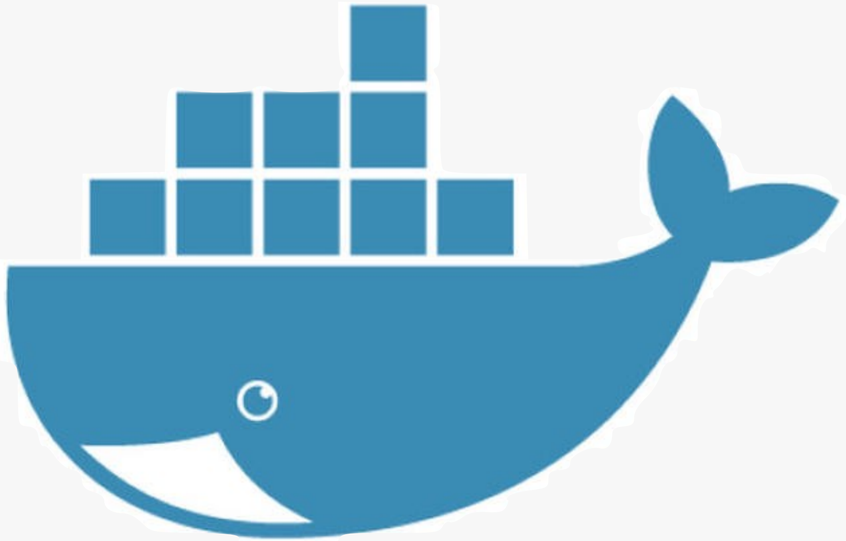
```
@@           Coverage Diff           @@
##            master     #7    +/-   ##
=====================================
  Coverage     100%   100%
=====================================
  Files           3      3
  Lines         100    101    +1
=====================================
+ Hits          100    101    +1
```

Impacted Files	Coverage Δ	
R/translator.R	`100% <100%> (ø)`	⬆️

github.com/reside-ic/traduire/pull/7


AUTOMATE
ALL THE THINGS!
imgflip.com


T



100% coverage is only the beginning

Mocking

blog.r-hub.io/2019/10/29/mocking/

```r
relink <- function(from, to) {
  backup <- paste0(from, ".bak")
  fs::file_move(from, backup)
  withCallingHandlers(
    fs::link_create(to, from, FALSE),
    error = function(e) fs::file_move(backup, from))
  fs::file_delete(backup)
}


test_that("relink error handling", {
  ...

  mockery::stub(relink, "fs::link_create",
                function(...) stop("Some error linking"))
  info <- fs::file_info(c(from, to))$inode
  expect_error(relink(from, to), "Some error linking")
  expect_true(all(fs::file_info(c(from, to))$inode == info))
})
```

System-specific behaviour

Long running processes

Sensitive data

Interactive user input

Awkward global state

blog.r-hub.io/2019/10/29/mocking/

The hard basket randomness

The hard basket

shiny

shiny.rstudio.com/articles/shinytest.html
shiny.rstudio.com/articles/integration-testing.html
github.com/reside-ic/shiny-selenium

The hard basket

data & analyses

The hard basket

long running tests

github.com/vimc/montagu-ci
github.com/features/actions

If testing feels like a chore

change how you do it

change why you do it