Unit testing and the `testthat` package

Dave Harris
5/29/12

(with content from Hadley Wickham's *R Journal* piece on `testthat`)

Why test?

- Confirm that the program does what it should (output matches expectations)
- Confirm that changes don't break anything
- Confidence in your code

Unit testing

- Test small *units* of code (e.g. functions)
- Specify the expected output of each unit under the important use cases
- De facto functional specification

Why test *formally*?

- Keep good tests around, in case things change
- Ensure tests *cover* all the important code
- “Code that’s easy to test is usually better designed”

Why not just do all your tests manually with a script?

- Need to remember to re-run tests periodically
- Possible weirdness if multiple tests fail simultaneously
- Environment would get cluttered--tests could conflict with one another

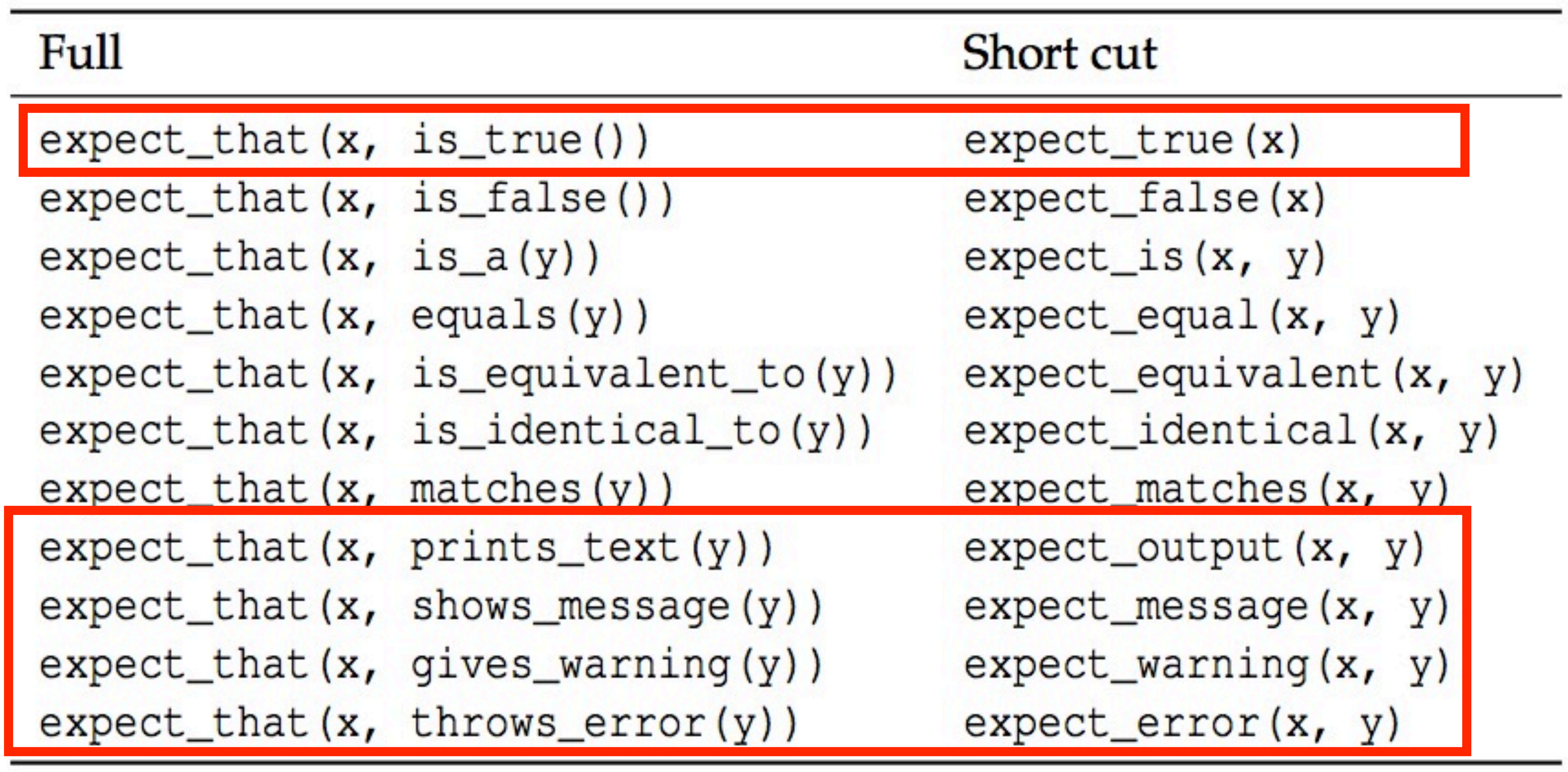
The `testthat` package

- Tools for *automatically* running tests & describing what broke
- Scope handling
- Better exception handling than stop() and related functions

Expectations and tests

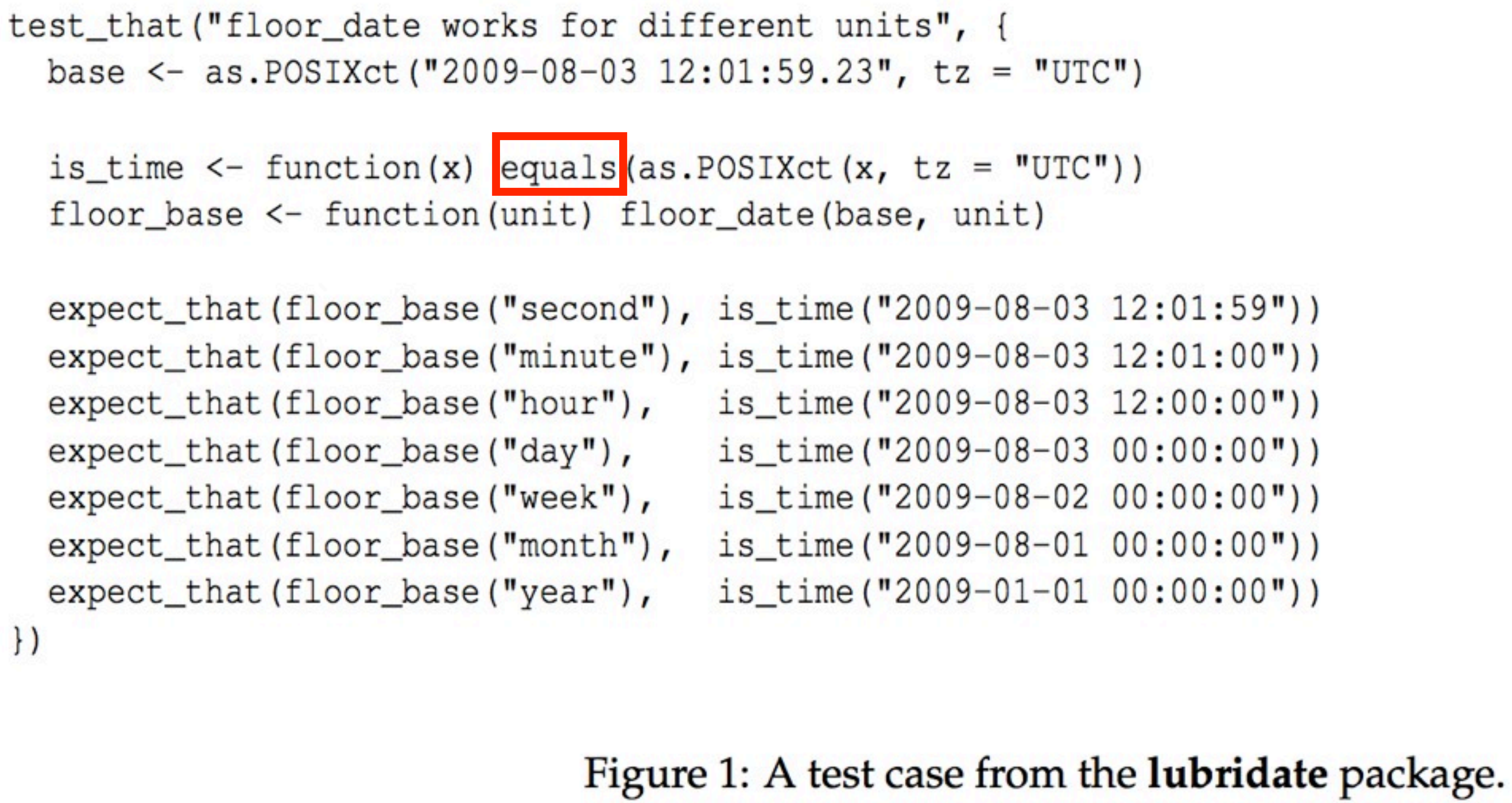
- An *expectation* describes what the result of a computation should be.
  - Does it have the right value and right class? Does it produce error messages when you expect it to?
- A *test* groups together multiple expectations to test one function, or tightly related functionality across multiple functions.

Wickham 2011



Full	Short cut
expect_that(x, is_true())	expect_true(x)
expect_that(x, is_false())	expect_false(x)
expect_that(x, is_a(y))	expect_is(x, y)
expect_that(x, equals(y))	expect_equal(x, y)
expect_that(x, is_equivalent_to(y))	expect_equivalent(x, y)
expect_that(x, is_identical_to(y))	expect_identical(x, y)
expect_that(x, matches(y))	expect_matches(x, y)
expect_that(x, prints_text(y))	expect_output(x, y)
expect_that(x, shows_message(y))	expect_message(x, y)
expect_that(x, gives_warning(y))	expect_warning(x, y)
expect_that(x, throws_error(y))	expect_error(x, y)

Table 1: Expectation shortcuts

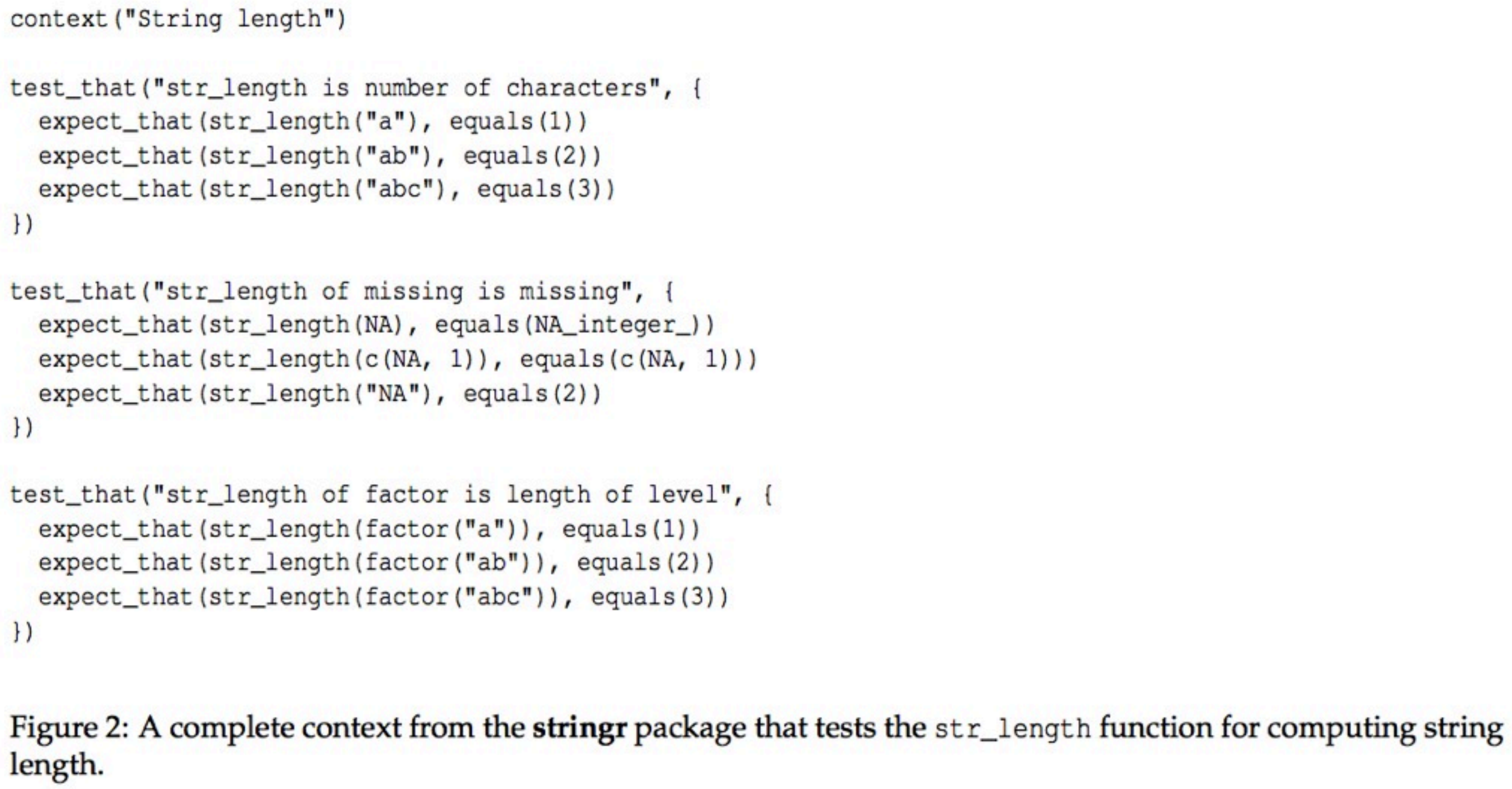


```r
test_that("floor_date works for different units", {
  base <- as.POSIXct("2009-08-03 12:01:59.23", tz = "UTC")

  is_time <- function(x) equals(as.POSIXct(x, tz = "UTC"))
  floor_base <- function(unit) floor_date(base, unit)

  expect_that(floor_base("second"), is_time("2009-08-03 12:01:59"))
  expect_that(floor_base("minute"), is_time("2009-08-03 12:01:00"))
  expect_that(floor_base("hour"),   is_time("2009-08-03 12:00:00"))
  expect_that(floor_base("day"),    is_time("2009-08-03 00:00:00"))
  expect_that(floor_base("week"),   is_time("2009-08-02 00:00:00"))
  expect_that(floor_base("month"),  is_time("2009-08-01 00:00:00"))
  expect_that(floor_base("year"),   is_time("2009-01-01 00:00:00"))
})
```

Figure 1: A test case from the **lubridate** package.



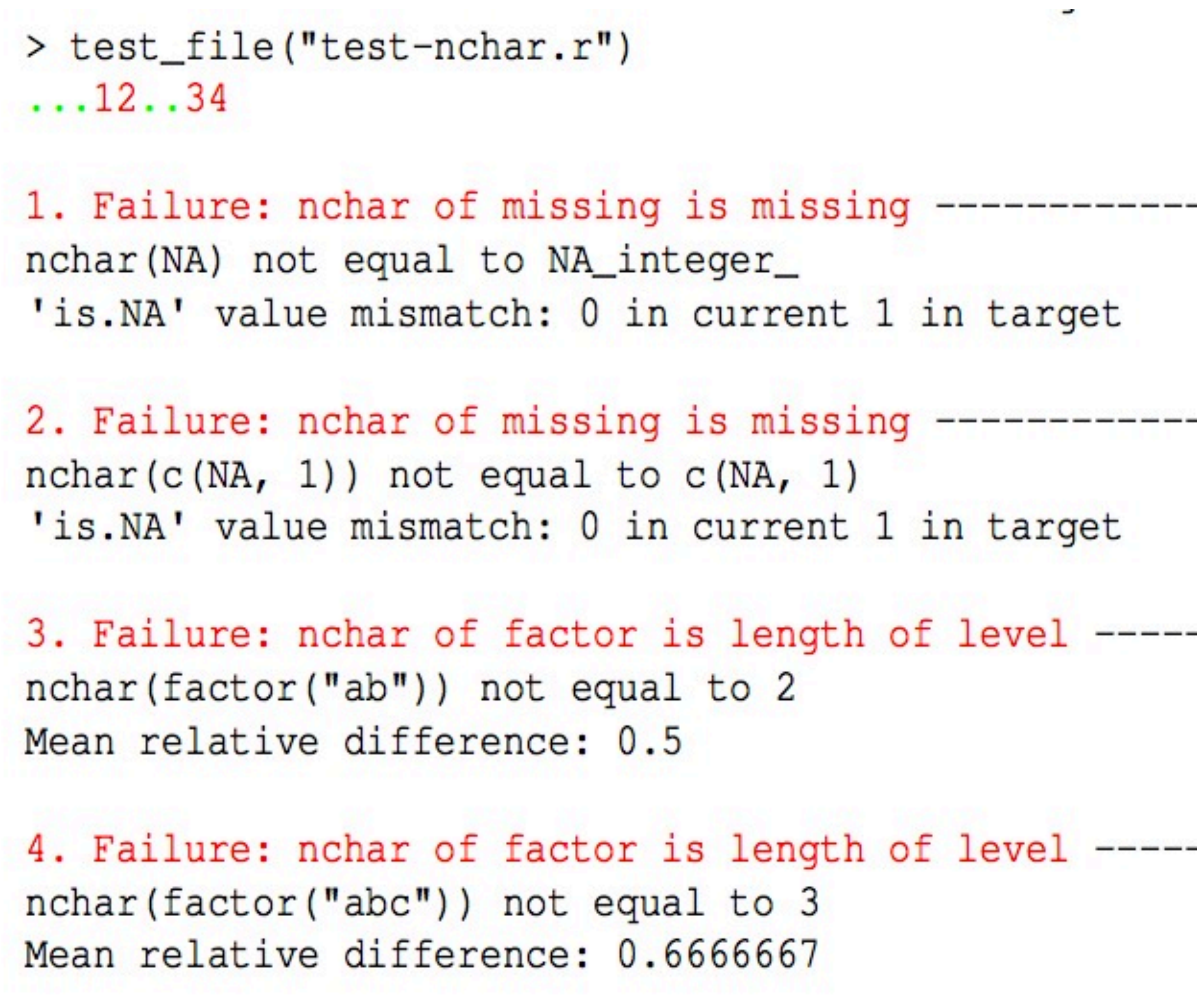
```
context("String length")

test_that("str_length is number of characters", {
  expect_that(str_length("a"), equals(1))
  expect_that(str_length("ab"), equals(2))
  expect_that(str_length("abc"), equals(3))
})

test_that("str_length of missing is missing", {
  expect_that(str_length(NA), equals(NA_integer_))
  expect_that(str_length(c(NA, 1)), equals(c(NA, 1)))
  expect_that(str_length("NA"), equals(2))
})

test_that("str_length of factor is length of level", {
  expect_that(str_length(factor("a")), equals(1))
  expect_that(str_length(factor("ab")), equals(2))
  expect_that(str_length(factor("abc")), equals(3))
})
```

Figure 2: A complete context from the **stringr** package that tests the `str_length` function for computing string length.



```
> test_file("test-nchar.r")
...12..34

1. Failure: nchar of missing is missing -----------
nchar(NA) not equal to NA_integer_
'is.NA' value mismatch: 0 in current 1 in target

2. Failure: nchar of missing is missing -----------
nchar(c(NA, 1)) not equal to c(NA, 1)
'is.NA' value mismatch: 0 in current 1 in target

3. Failure: nchar of factor is length of level -----
nchar(factor("ab")) not equal to 2
Mean relative difference: 0.5

4. Failure: nchar of factor is length of level -----
nchar(factor("abc")) not equal to 3
Mean relative difference: 0.6666667
```

Autotest

- `autotest()` has two arguments, `code_path` and `test_path`, which point to a directory of source code and tests respectively.
- Once run, `autotest()` will continuously scan both directories for changes.

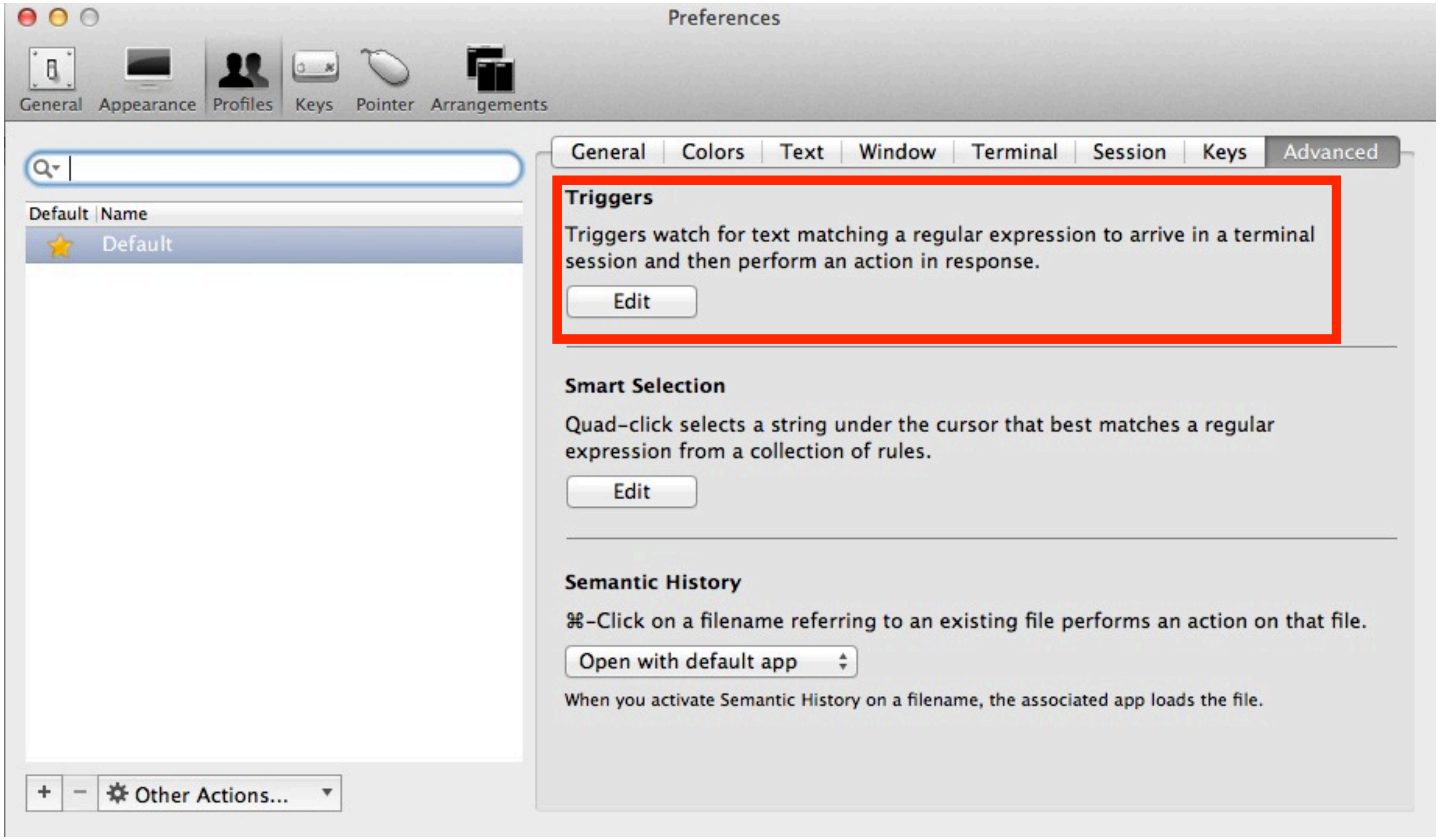
Wickham 2011

```
library(testthat)

setwd("~/github/multispecies/")

auto_test("R", "inst/tests/")
```

iTerm



Triggers

Triggers watch for text matching a regular expression to arrive in a terminal session and then perform an action in response.

Edit

Smart Selection

Quad-click selects a string under the cursor that best matches a regular expression from a collection of rules.

Edit

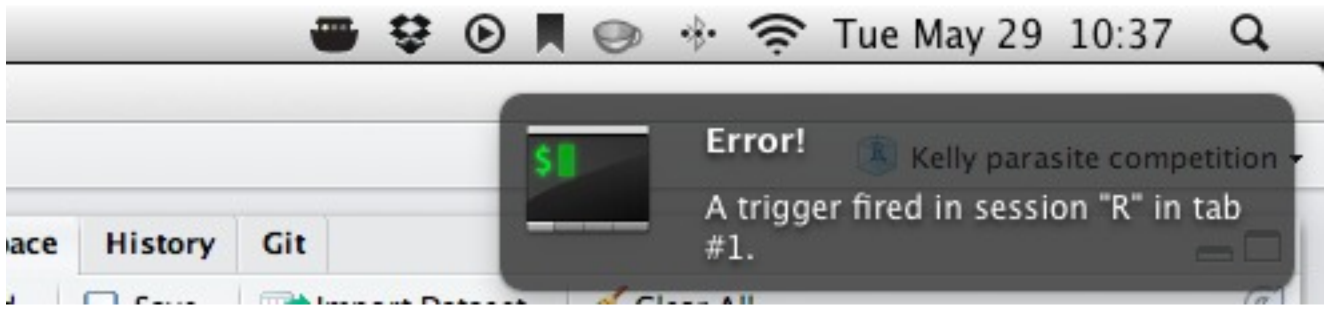
Semantic History

⌘-Click on a filename referring to an existing file performs an action on that file.

Open with default app

When you activate Semantic History on a filename, the associated app loads the file.

iTerm + Growl



Tue May 29 10:37

Error!

A trigger fired in session "R" in tab #1.

R CMD check and test_package()

- `test_package()` evaluates tests in the package namespace and throws an error if any tests fail.
- `R CMD check` won't pass unless all your tests pass