Package ‘testthat’

February 20, 2015

Version 0.9.1
Title Testthat code. Tools to make testing fun :)
Description A testing package specifically tailored for R that's fun, flexible and easy to set up.

URL https://github.com/hadley/testthat
BugReports https://github.com/hadley/testthat/issues
Depends R (>= 3.1.0), methods
Imports digest
Suggests devtools
License MIT + file LICENSE
Collate 'auto-test.r' 'colour-text.r' 'compare.r' 'context.r'
    'describe.r' 'evaluate-promise.r' 'expect-that.r'
    'expectation.r' 'expectations.r' 'library.r'
    'make-expectation.r' 'mock.r' 'reporter.r' 'reporter-list.r'
    'reporter-minimal.r' 'reporter-multi.r' 'reporter-silent.r'
    'reporter-stop.r' 'reporter-summary.r' 'reporter-tap.r'
    'reporter-teamcity.r' 'reporter-xxx.r' 'test-files.r'
    'test-package.r' 'test-that.r' 'traceback.r' 'utils.r'
    'watcher.r'

Author Hadley Wickham [aut, cre],
    RStudio [cph]
Maintainer Hadley Wickham <hadley@rstudio.com>

NeedsCompilation yes
Repository CRAN
Date/Publication 2014-10-01 19:16:27

R topics documented:

    auto_test ................................................................. 3
    auto_test_package ..................................................... 4
R topics documented:

colourise .................................................. 4
context ...................................................... 5
describe .................................................... 5
equals .......................................................... 6
equals_reference .......................................... 8
evaluate.promise ......................................... 9
expect-compare .......................................... 10
expect_that ............................................... 11
fail ............................................................. 12
gives.warning ........................................... 13
has_names .................................................. 14
is.a ............................................................ 15
is_equivalent_to ......................................... 16
is_identical_to ........................................... 17
is_null ....................................................... 18
is_true ....................................................... 18
library_if_available ..................................... 20
ListReporter-class ......................................... 20
make_expectation ......................................... 21
matches ...................................................... 21
MinimalReporter-class .................................... 22
MultiReporter-class ........................................ 23
not ............................................................. 23
prints_text ................................................. 24
Reporter-class .............................................. 25
shows_message ............................................ 25
SilentReporter-class ....................................... 26
skip ........................................................... 26
StopReporter-class ......................................... 27
SummaryReporter-class .................................... 27
takes_less_than ........................................... 28
TapReporter-class .......................................... 28
TeamcityReporter-class .................................... 29
testthat ...................................................... 29
test_dir ...................................................... 30
test_file ..................................................... 30
test_package ............................................... 31
test_that ..................................................... 32
throws_error .............................................. 33
watch ........................................................ 34
with_mock ................................................... 34

Index ................................................... 36
**auto_test**

Watches code and tests for changes, rerunning tests as appropriate.

---

**Description**

The idea behind `auto_test` is that you just leave it running while you develop your code. Everytime you save a file it will be automatically tested and you can easily see if your changes have caused any test failures.

**Usage**

```
auto_test(code_path, test_path, reporter = "summary", env = test_env())
```

**Arguments**

- `code_path` path to directory containing code
- `test_path` path to directory containing tests
- `reporter` test reporter to use
- `env` environment in which to execute test suite.

**Details**

The current strategy for rerunning tests is as follows:

- if any code has changed, then those files are reloaded and all tests rerun
- otherwise, each new or modified test is run

In the future, `auto_test` might implement one of the following more intelligent alternatives:

- Use codetools to build up dependency tree and then rerun tests only when a dependency changes.
- Mimic ruby’s autotest and rerun only failing tests until they pass, and then rerun all tests.

**See Also**

`auto_test_package`
auto_test_package  Watches a package for changes, rerunning tests as appropriate.

Description
Watches a package for changes, rerunning tests as appropriate.

Usage
auto_test_package(pkg = ".", reporter = "summary")

Arguments
pkg  path to package
reporter  test reporter to use

See Also
auto_test for details on how method works

colourise  Colourise text for display in the terminal.

Description
If R is not currently running in a system that supports terminal colours the text will be returned unchanged.

Usage
colourise(text, fg = "black", bg = NULL)

Arguments
text  character vector
fg  foreground colour, defaults to white
bg  background colour, defaults to transparent

Details
Allowed colours are: black, blue, brown, cyan, dark gray, green, light blue, light cyan, light gray,
lime green, light purple, light red, purple, red, white, yellow
Colourization of the output can be turned-off by setting option 'testthat.use_colours' to FALSE.
**context**

Describe the context of a set of tests.

**Description**

A context defines a set of tests that test related functionality. Usually you will have one context per file, but you may have multiple contexts in a single file if you so choose.

**Usage**

context(desc)

**Arguments**

desc
description of context. Should start with a capital letter.

**Examples**

context("String processing")
context("Remote procedure calls")

---

**describe**

describe: a BDD testing language

**Description**

A simple BDD DSL for writing tests. The language is similar to RSpec for Ruby or Mocha for JavaScript. BDD tests read like sentences and it should thus be easier to understand what the specification of a function/component is.

**Usage**

describe(description, code)

**Arguments**

description
description of the feature
code
test code containing the specs
Details

Tests using the describe syntax not only verify the tested code, but also document its intended behaviour. Each describe block specifies a larger component or function and contains a set of specifications. A specification is defined by an it block. Each it block functions as a test and is evaluated in its own environment. You can also have nested describe blocks.

This test syntax helps to test the intended behaviour of your code. For example: you want to write a new function for your package. Try to describe the specification first using describe, before you write any code. After that, you start to implement the tests for each specification (i.e. the it block).

Use describe to verify that you implement the right things and use test_that to ensure you do the things right.

Examples

```r
describe("matrix()", {
  it("can be multiplied by a scalar", {
    m1 <- matrix(1:4, 2, 2)
    m2 <- m1 * 2
    expect_equivalent(matrix(1:4 * 2, 2, 2), m2)
  })
  it("can have not yet tested specs")
})

# Nested specs:
## code
addition <- function(a, b) a + b
division <- function(a, b) a / b

## specs
describe("math library", {
  describe("addition()", {
    it("can add two numbers", {
      expect_equivalent(1 + 1, addition(1, 1))
    })
  })
  describe("division()", {
    it("can divide two numbers", {
      expect_equivalent(10 / 2, division(10, 2))
    })
    it("can handle division by 0") # not yet implemented
  })
})
```

equals

Expectation: is the object equal (with numerical tolerance) to a value?

Description

Comparison performed using all.equal.
equals

Usage

equals(expected, label = NULL, ...)

expect_equal(object, expected, ..., info = NULL, label = NULL,
expected.label = NULL)

Arguments

expected Expected value
label For full form, label of expected object used in error messages. Useful to override
default (deparsed expected expression) when doing tests in a loop. For short cut
form, object label. When NULL, computed from deparsed object.

... other values passed to all.equal
object object to test
info extra information to be included in the message (useful when writing tests in
loops).

expected.label Equivalent of label for shortcut form.

See Also

Other expectations: equals_reference, expect_equal_to_reference; expect-compare, expect_less_than,
expect_more_than, is_less_than, is_more_than; expect_equivalent, is_equivalent_to;
expect_error, throws_error; expect_false, expect_true, is_false, is_true; expect_identical,
is_identical_to; expect_is, is_a; expect_match, matches; expect_message, shows_message;
expect_named, has_names; expect_null, is_null; expect_output, prints_text; expect_warning,
gives_warning; takes_less_than

Examples

a <- 10
expect_equal(a, 10)

# Use equals() when testing for numeric equality
sqrt(2) ^ 2 - 1
expect_equal(sqrt(2) ^ 2, 2)
# Neither of these forms take floating point representation errors into
# account
## Not run:
expect_true(sqrt(2) ^ 2 == 2)
expect_identical(sqrt(2) ^ 2, 2)

## End(Not run)

# You can pass on additional arguments to all.equal:
## Not run:
# Test the ABSOLUTE difference is within .002
expect_equal(object = 10.01, expected = 10, tolerance = .002, scale = 1)
equals_reference

# Test the RELATIVE difference is within .002
expectedValue <- 10
expect_equal(object = 10.01, expected = expectedValue, tolerance = 0.002,
  scale = expectedValue)

## End(Not run)

equals_reference  Expectation: is the object equal to a reference value stored in a file?

Description

This expectation is equivalent to `equals`, except that the expected value is stored in an RDS file instead of being specified literally. This can be helpful when the value is necessarily complex. If the file does not exist then it will be created using the value of the specified object, and subsequent tests will check for consistency against that generated value. The test can be reset by deleting the RDS file.

Usage

equals_reference(file, label = NULL, ...)

expect_equal_to_reference(object, file, ..., info = NULL, label = NULL,
  expected.label = NULL)

Arguments

file  The file name used to store the object. Should have an "rds" extension.
label For the full form, a label for the expected object, which is used in error messages. Useful to override the default (which is based on the file name), when doing tests in a loop. For the short-cut form, the object label, which is computed from the deparsed object by default.
... other values passed to `equals`
object object to test
info  extra information to be included in the message (useful when writing tests in loops).
expected.label Equivalent of label for shortcut form.

See Also

Other expectations: `equals`, `expect_equal`; `expect_compare`, `expect_less_than`, `expect_more_than`, `is_less_than`, `is_more_than`; `expect_equivalent`, `is_equivalent_to`; `expect_error`, `throws_error`; `expect_false`, `expect_true`, `is_false`, `is_true`; `expect_identical`, `is_identical_to`; `expect_is`, `is_a`; `expect_match`, `matches`; `expect_message`, `shows_message`; `expect_named`, `has_names`; `expect_null`, `is_null`; `expect_output`, `prints_text`; `expect_warning`, `gives_warning`; `takes_less_than`
evaluate.promise

Examples

```r
## Not run:
expect.equal_to_reference(1, "one.rds")

## End(Not run)
```

evaluate.promise  Evaluate a promise, capturing all types of output.

Description

This uses `evaluate` a promise, returning the result, test, messages and warnings that the code creates in a list. It is used to evaluate code for all test that tests, ensuring that (as much as possible) any spurious output is suppressed during the testing process.

Usage

```r
evaluate.promise(code, print = FALSE)
```

Arguments

- `code` Code to evaluate. This should be an unevaluated expression.
- `print` If TRUE and the result of evaluating `code` is visible this will print the result, ensuring that the output of printing the object is included in the overall output.

Value

A list containing

- `result` The result of the function
- `output` A string containing all the output from the function
- `warnings` A character vector containing the text from each warning
- `messages` A character vector containing the text from each message

Examples

```r
evaluate.promise(
  print("1"),
  message("2"),
  warning("3")
)
```
**expect-compare**

*Expectation: is returned value less or greater than specified value?*

**Description**

This is useful for ensuring returned value is below a ceiling or above a floor.

**Usage**

```r
is_less_than(expected, label = NULL, ...)
expect_less_than(object, expected, ..., info = NULL, label = NULL,
                 expected.label = NULL)

is_more_than(expected, label = NULL, ...)
expect_more_than(object, expected, ..., info = NULL, label = NULL,
                 expected.label = NULL)
```

**Arguments**

- `expected`: Expected value
- `label`: For full form, label of expected object used in error messages. Useful to override default (deparsed expected expression) when doing tests in a loop. For short cut form, object label. When `NULL`, computed from deparsed object.
- `...`: other values passed to `all.equal`
- `object`: object to test
- `info`: extra information to be included in the message (useful when writing tests in loops).
- `expected.label`: Equivalent of `label` for shortcut form.

**See Also**

Other expectations: `equals_reference`, `expect_equal_to_reference`; `equals`, `expect_equal`; `expect_equivalent`, `is_equivalent_to`; `expect_error`, `throws_error`; `expect_false`, `expect_true`, `is_false`, `is_true`; `expect_identical`, `is_identical_to`; `expect_is`, `is_a`; `expect_match`, `matches`; `expect_message`, `shows_message`; `expect_named`, `has_names`; `expect_null`, `is_null`; `expect_output`, `prints_text`; `expect_warning`, `gives_warning`; `takes_less_than`

**Examples**

```r
a <- 9
expect_less_than(a, 10)

# Not run:
expect_less_than(11, 10)
```
### Description

An expectation checks whether a single condition holds true. **testthat** currently provides the following expectations. See their documentation for more details.

### Usage

```r
expect_that(object, condition, info = NULL, label = NULL)
```

### Arguments

- `object`: object to test
- `condition`: a function that returns whether or not the condition is met, and if not, an error message to display.
- `info`: extra information to be included in the message (useful when writing tests in loops).
- `label`: object label. When NULL, computed from deparsed object.

### Details

- **is_true**: truth
- **is_false**: falsehood
- **is_a**: inheritance
- **equals**: equality with numerical tolerance
- **equals_reference**: equality relative to a reference
- **is_equivalent_to**: equality ignoring attributes
- **is_identical_to**: exact identity
- **matches**: string matching
- **prints_text**: output matching
- **throws_error**: error matching
- **gives_warning**: warning matching
Value

the (internal) expectation result as an invisible list

See Also

fail for an expectation that always fails.

Examples

```r
expect_that(5 * 2, equals(10))
expect_that(sqrt(2) ^ 2, equals(2))
## Not run:
expect_that(sqrt(2) ^ 2, is_identical_to(2))
## End(Not run)
```

---

**fail**  
A default expectation that always fails.

Description

The fail function forces a test to fail. This is useful if you want to test a pre-condition.

Usage

`fail(message = "Failure has been forced.")`

Arguments

- `message` a string to display.

Examples

```r
## Not run:
test_that("this test fails", fail())
## End(Not run)
```
gives_warning

**Expectation: does expression give a warning?**

**Description**

Needs to match at least one of the warnings produced by the expression.

**Usage**

```r
gives_warning(regexp = NULL, all = FALSE, ...)
expect_warning(object, regexp = NULL, ..., info = NULL, label = NULL)
```

**Arguments**

- `regexp` optional regular expression to match. If not specified, just asserts that expression gives some warning.
- `all` if TRUE, all warnings must match given regular expression; if FALSE (the default), then only only warning needs to match
- `...` other arguments passed to `matches`
- `object` object to test
- `info` extra information to be included in the message (useful when writing tests in loops).
- `label` object label. When NULL, computed from deparsed object.

**See Also**

Other expectations: `equals_reference`, `expect_equal_to_reference`; `equals`, `expect_equal`; `expect_compare`, `expect_less_than`, `expect_more_than`, `is_less_than`, `is_more_than`; `expect_equivalent`, `is_equivalent_to`; `expect_error`, `throws_error`; `expect_false`, `expect_true`, `is_false`, `is_true`; `expect_identical`, `is_identical_to`; `expect_is`, `is_a`; `expect_match`, `matches`; `expect_message`, `shows_message`; `expect_named`, `has_names`; `expect_null`, `is_null`; `expect_output`, `prints_text`; `takes_less_than`

**Examples**

```r
f <- function(x) {
  if (x < 0) warning("*x* is already negative")
  -x
}
expect_warning(f(-1))
expect_warning(f(-1), "already negative")
## Not run: expect_warning(f(1))

# You can use the arguments of grepl to control the matching
expect_warning(f(-1), "*xx*", fixed = TRUE)
expect_warning(f(-1), "NEGATIVE", ignore.case = TRUE)
```
has_names  

Expectation: does object have names?

Description

You can either check for the presence of names (leaving expected blank), specific names (by suppling a vector of names), or absence of names (with NULL).

Usage

has_names(expected, ignore.order = FALSE, ignore.case = FALSE)

expect_named(object, expected, ..., info = NULL, label = NULL)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>expected</td>
<td>Character vector of expected names. Leave missing to match any names. Use NULL to check for absence of names.</td>
</tr>
<tr>
<td>ignore.order</td>
<td>If TRUE, sorts names before comparing to ignore the effect of order.</td>
</tr>
<tr>
<td>ignore.case</td>
<td>If TRUE, lowercases all names to ignore the effect of case.</td>
</tr>
<tr>
<td>object</td>
<td>object to test</td>
</tr>
<tr>
<td>...</td>
<td>Other arguments passed onto has_names.</td>
</tr>
<tr>
<td>info</td>
<td>extra information to be included in the message (useful when writing tests in loops).</td>
</tr>
<tr>
<td>label</td>
<td>object label. When NULL, computed from deparsed object.</td>
</tr>
</tbody>
</table>

See Also

Other expectations: equals_reference, expect_equal_to_reference; equals, expect_equal; expect_less_than, expect_more_than, is_less_than, is_more_than; expect_equivalent, is_equivalent_to; expect_error, throws_error; expect_false, expect_true, is_false, is_true; expect_identical, is_identical_to; expect_is, is_a; expect_match, matches; expect_message, shows_message; expect_null, is_null; expect_output, prints_text; expect_warning, gives_warning; takes_less_than

Examples

```r
x <- c(a = 1, b = 2, c = 3)
extpect_named(x)
extpect_named(x, c("a", "b", "c"))

# Use options to control sensitivity
expect_named(x, c("B", "C", "A"), ignore.order = TRUE, ignore.case = TRUE)

# Can also check for the absence of names with NULL
z <- 1:4
expect_named(z, NULL)
```
is_a

Expectation: does the object inherit from a class?

Description
Tests whether or not an object inherits from any of a list of classes.

Usage

is_a(class)

expect_is(object, class, info = NULL, label = NULL)

Arguments

class character vector of class names
object object to test
info extra information to be included in the message (useful when writing tests in loops).
label object label. When NULL, computed from deparsed object.

See Also

inherits

Other expectations: equals_reference, expect_equal_to_reference; equals, expect_equal; expect_compare, expect_less_than, expect_more_than, is_less_than, is_more_than; expect_equivalent, is_equivalent_to; expect_error, throws_error; expect_false, expect_true, is_false, is_true; expect_identical, is_identical_to; expect_match, matches; expect_message, shows_message; expect_named, has_names; expect_null, is_null; expect_output, prints_text; expect_warning, gives_warning; takes_less_than

Examples

expect_is(1, "numeric")
a <- matrix(1:10, nrow = 5)
expect_is(a, "matrix")

expect_is(mtcars, "data.frame")
# alternatively for classes that have an is method
expect_true(is.data.frame(mtcars))
is_equivalent_to  

**Description**

Expectation: is the object equivalent to a value? This expectation tests for equivalency: are two objects equal once their attributes have been removed.

**Usage**

```r
is_equivalent_to(expected, label = NULL)
expect_equivalent(object, expected, info = NULL, label = NULL,
                  expected.label = NULL)
```

**Arguments**

- `expected` Expected value
- `label` For full form, label of expected object used in error messages. Useful to override default (deparsed expected expression) when doing tests in a loop. For short cut form, object label. When NULL, computed from deparsed object.
- `object` object to test
- `info` extra information to be included in the message (useful when writing tests in loops).
- `expected.label` Equivalent of label for shortcut form.

**See Also**

Other expectations: `equals_reference`, `expect_equal_to_reference`; `equals`, `expect_equal`;
`expect_compare`, `expect_less_than`, `expect_more_than`, `is_less_than`, `is_more_than`; `expect_error`,
`throws_error`; `expect_false`, `expect_true`, `is_false`, `is_true`; `expect_identical`, `is_identical_to`;
`expect_is`, `is_a`; `expect_match`, `matches`; `expect_message`, `shows_message`; `expect_named`,
`has_names`; `expect_null`, `is_null`; `expect_output`, `prints_text`; `expect_warning`, `gives_warning`;
`takes_less_than`

**Examples**

```r
a <- b <- 1:3
names(b) <- letters[1:3]
expect_equivalent(a, b)
```
is_identical_to

Expectation: is the object identical to another?

Description
Comparison performed using identical.

Usage
is_identical_to(expected, label = NULL)
expect_identical(object, expected, info = NULL, label = NULL,
expected.label = NULL)

Arguments
expected Expected value
label For full form, label of expected object used in error messages. Useful to override
default (deparsed expected expression) when doing tests in a loop. For short cut
form, object label. When NULL, computed from deparsed object.
object object to test
info extra information to be included in the message (useful when writing tests in
loops).
expected.label Equivalent of label for shortcut form.

See Also
Other expectations: equals_reference, expect_equal_to_reference; equals, expect_equal;
expect_compare, expect_less_than, expect_more_than, is_less_than, is_more_than; expect_equivalent,
is_equivalent_to; expect_error, throws_error; expect_false, expect_true, is_false,
is_true; expect_is, is_a; expect_match, matches; expect_message, shows_message; expect_named,
has_names; expect_null, is_null; expect_output, prints_text; expect_warning, gives_warning;
takes_less_than

Examples
a <- letters[1:3]
expect_identical(a, c("a", "b", "c"))

# Identical does not take into account numeric tolerance
## Not run:
expect_identical(sqrt(2) ^ 2, 2)

## End(Not run)
is_null  

**Description**

Expectation: is the object NULL?

**Usage**

```r
is_null()
```

```r
eexpect_null(object, info = NULL, label = NULL)
```

**Arguments**

- **object**: object to test
- **info**: extra information to be included in the message (useful when writing tests in loops).
- **label**: object label. When NULL, computed from deparsed object.

**See Also**

Other expectations: `equals_reference`, `expect_equal_to_reference`; `equals`, `expect_equal`; `expect_compare`, `expect_less_than`, `expect_more_than`, `is_less_than`, `is_more_than`; `expect_equivalent`, `is_equivalent_to`; `expect_error`, `throws_error`; `expect_false`, `expect_true`, `is_false`, `is_true`; `expect_identical`, `is_identical_to`; `expect_is`, `is_a`; `expect_match`, `matches`; `expect_message`, `shows_message`; `expect_named`, `has_names`; `expect_output`, `prints_text`; `expect_warning`, `gives_warning`; `takes_less_than`

**Examples**

```r
expect_null(NULL)
```

------

is_true  

**Description**

These are fall-back expectations that you can use when none of the other more specific expectations apply. The disadvantage is that you may get a less informative error message.
**is_true**

Usage

```r
is_true()
expect_true(object, info = NULL, label = NULL)
```

```r
is_false()
expect_false(object, info = NULL, label = NULL)
```

Arguments

- **object** object to test
- **info** extra information to be included in the message (useful when writing tests in loops).
- **label** object label. When NULL, computed from deparsed object.

Details

Attributes are ignored.

See Also

`is_false` for complement

Other expectations: `equals_reference`, `expect_equal_to_reference`; `equals`, `expect_equal`; `expect_compare`, `expect_less_than`, `expect_more_than`, `is_less_than`, `is_more_than`; `expect_equivalent`, `is_equivalent_to`; `expect_error`, `throws_error`; `expect_identical`, `is_identical_to`; `expect_is`, `is_a`; `expect_match`, `matches`; `expect_message`, `shows_message`; `expect_named`, `has_names`; `expect_null`, `is_null`; `expect_output`, `prints_text`; `expect_warning`, `gives_warning`; `takes_less_than`

Examples

```r
expect_true(2 == 2)
# Failed expectations will throw an error
## Not run:
expect_true(2 != 2)

## End(Not run)
expect_true(! (2 != 2))
# or better:
expect_false(2 != 2)

a <- 1:3
expect_true(length(a) == 3)
# but better to use more specific expectation, if available
expect_equal(length(a), 3)
```
library_if_available  

Load package, if available.

Description

Quietly load a package if it is installed, otherwise do nothing. This is useful for testing files so that you can run them while you are developing your package, before it is installed for the first time; then continue to have the same code work when the tests are run automatically by R CMD CHECK.

Usage

library_if_available(package)

Arguments

package package name (without quotes)

Examples

## Not run:
library_if_available(testthat)
library_if_available(package_that_doesnt_exist)

## End(Not run)

ListReporter-class  

List reporter: gather all test results along with elapsed time and file information.

Description

This reporter gathers all results, adding additional information such as test elapsed time, and test filename if available. Very useful for reporting.

Arguments

... Arguments used to initialise class
make_expectation  

Description

This a convenience function to make a expectation that checks that input stays the same.

Usage

make_expectation(x, expectation = "equals")

Arguments

x a vector of values
expectation the type of equality you want to test for (equals, is_equivalent_to, is_identical_to)

Examples

x <- 1:10
make_expectation(x)

make_expectation(mtcars$mpg)

df <- data.frame(x = 2)
make_expectation(df)

matches  

Expectation: does string match regular expression?

Description

If the object to be tested has length greater than one, all elements of the vector must match the pattern in order to pass.

Usage

matches(regexp, all = TRUE, ...)

expect_match(object, regexp, ..., info = NULL, label = NULL)
**MinimalReporter-class**

**Arguments**

- `regexp`: regular expression to test against
- `all`: should all elements of actual value match `regexp` (TRUE), or does only one need to match (FALSE)
- `object`: object to test
- `info`: extra information to be included in the message (useful when writing tests in loops)
- `label`: object label. When NULL, computed from deparsed object.

**See Also**

Other expectations: `equals_reference, expect_equal_to_reference; equals, expect_equal; expect_compare, expect_less_than, expect_more_than, is_less_than, is_more_than; expect_equivalent, is_equivalent_to; expect_error, throws_error; expect_false, expect_true, is_false, is_true; expect_identical, is_identical_to; expect_is, is_a; expect_message, shows_message; expect_named, has_names; expect_null, is_null; expect_output, prints_text; expect_warning, gives_warning; takes_less_than`

**Examples**

```r
expect_match("Testing is fun", "fun")
expect_match("Testing is fun", "f.n")
```

---

**MinimalReporter-class**  
*Test reporter: minimal.*

**Description**

The minimal test reporter provides the absolutely minimum amount of information: whether each expectation has succeeded, failed or experienced an error. If you want to find out what the failures and errors actually were, you’ll need to run a more informative test reporter.

**Arguments**

```
... Arguments used to initialise class
```
MultiReporter-class  Multi reporter: combine several reporters in one.

Description

This reporter is useful to use several reporters at the same time, e.g. adding a custom reporter without removing the current one.

Arguments

...  Arguments used to initialise class

not  Negate an expectation

Description

This negates an expectation, making it possible to express that you want the opposite of a standard expectation.

Usage

not(f)

Arguments

f  an existing expectation function

Examples

```r
x <- 1
expect_that(x, equals(1))
expect_that(x, not(equals(2)))
## Not run:
expect_that(x, equals(2))
expect_that(x, not(equals(1)))
## End(Not run)
```
prints_text  

Expectation: does printed output match a regular expression?

Description

Expectation: does printed output match a regular expression?

Usage

prints_text(regexp, ...)

expect_output(object, regexp, ..., info = NULL, label = NULL)

Arguments

regexp  

regular expression to test against

...  

other arguments passed to matches

object  

object to test

info  

extra information to be included in the message (useful when writing tests in loops).

label  

object label. When NULL, computed from deparsed object.

See Also

Other expectations: equals_reference, expect_equal_to_reference; equals, expect_equal;
expect_compare, expect_less_than, expect_more_than, is_less_than, is_more_than; expect_equivalent,
is_equivalent_to; expect_error, throws_error; expect_false, expect_true, is_false,
is_true; expect_identical, is_identical_to; expect_is, is_a; expect_match, matches;
expect_message, shows_message; expect_named, has_names; expect_null, is_null; expect_warning,
gives_warning; takes_less_than

Examples

str(mtcars)
expect_output(str(mtcars), "32 obs")
expect_output(str(mtcars), "11 variables")

# You can use the arguments of grepl to control the matching
expect_output(str(mtcars), "11 VARIABLES", ignore.case = TRUE)
expect_output(str(mtcars), "$ mpg", fixed = TRUE)
Reporter-class

\[Stub \text{ object for managing a reporter of tests.}\]

Description

Do not clone directly from this object - children should implement all methods.

Arguments

\[
\ldots \quad \text{Arguments used to initialise class}
\]

shows_message

\[\text{Expectation: does expression show a message?}\]

Description

Needs to match at least one of the messages produced by the expression.

Usage

\[
\text{shows_message}(\text{regexp }= \text{NULL}, \text{all }= \text{FALSE}, \ldots)
\]

\[
\text{expect_message}(\text{object}, \text{regexp }= \text{NULL}, \ldots, \text{info }= \text{NULL}, \text{label }= \text{NULL})
\]

Arguments

\[
\text{regexp} \quad \text{optional regular expression to match. If not specified, just asserts that expression shows some message.}
\]

\[
\text{all} \quad \text{if TRUE, all messages must match given regular expression; if FALSE (the default), then only only message needs to match}
\]

\[
\ldots \quad \text{other arguments passed to matches}
\]

\[
\text{object} \quad \text{object to test}
\]

\[
\text{info} \quad \text{extra information to be included in the message (useful when writing tests in loops).}
\]

\[
\text{label} \quad \text{object label. When NULL, computed from deparsed object.}
\]

See Also

Other expectations: equals\_reference, expect\_equal\_to\_reference; equals, expect\_equal; expect\_compare, expect\_less\_than, expect\_more\_than, is\_less\_than, is\_more\_than; expect\_equivalent, is\_equivalent\_to; expect\_error, throws\_error; expect\_false, expect\_true, is\_false, is\_true; expect\_identical, is\_identical\_to; expect\_is, is\_a; expect\_match, matches; expect\_named, has\_names; expect\_null, is\_null; expect\_output, prints\_text; expect\_warning, gives\_warning; takes\_less\_than
Examples

```r
f <- function(x) {
  if (x < 0) message("x is already negative")
  -x
}
expect_message(f(-1))
expect_message(f(-1), "already negative")
## Not run: expect_message(f(1))

# You can use the arguments of grepl to control the matching
expect_message(f(-1), "*xx*", fixed = TRUE)
expect_message(f(-1), "NEGATIVE", ignore.case = TRUE)
```

SilentReporter-class  Test reporter: gather all errors silently.

Description

This reporter quietly runs all tests, simply gathering the results for later use. This is helpful for programmatically inspecting errors after a test run.

Arguments

...  Arguments used to initialise class

skip  

Skip a test.

Description

This function allows you to skip a test if it's not currently available. This will produce an informative message, but will not cause the test suite to fail. `skip_on_cran()` uses `skip()`, together with the NOT_CRAN environment variable (set by devtools), to automatically skip tests that should not be run on CRAN.

Usage

```r
skip(message)
```

Arguments

message  An message describing why the test was skipped.

Examples

```r
if (FALSE) skip("No internet connection")
```
**StopReporter-class**

*Test reporter: stop on error.*

**Description**

The default reporter, executed when `expect_that` is run interactively, or when the test files are executed by R CMD check. It responds by `stop()`ing on failures and doing nothing otherwise. This will ensure that a failing test will raise an error.

**Arguments**

... Arguments used to initialise class

**Details**

This should be used when doing a quick and dirty test, or during the final automated testing of R CMD check. Otherwise, use a reporter that runs all tests and gives you more context about the problem.

**SummaryReporter-class**

*Test reporter: summary of errors.*

**Description**

This is the most useful reporting reporter as it lets you know both which tests have run successfully, as well as fully reporting information about failures and errors. It is the default reporting reporter used by `test_dir` and `test_file`.

**Arguments**

... Arguments used to initialise class

**Details**

You can use the `max_reports` field to control the maximum number of detailed reports produced by this reporter. This is useful when running with `auto_test`.

As an additional benefit, this reporter will praise you from time-to-time if all your tests pass.
takes_less_than  

*Expectation: does expression take less than a fixed amount of time to run?*

**Description**

This is useful for performance regression testing.

**Usage**

```
takes_less_than(amount)
```

**Arguments**

- `amount` maximum duration in seconds

**See Also**

Other expectations: `equals_reference`, `expect_equal_to_reference`; `equals`, `expect_equal`; `expect_compare`, `expect_less_than`, `expect_more_than`, `is_less_than`, `is_more_than`; `expect_equivalent`, `is_equivalent_to`; `expect_error`, `throws_error`; `expect_false`, `expect_true`, `is_false`, `is_true`; `expect_identical`, `is_identical_to`; `expect_is`, `is_a`; `expect_match`, `matches`; `expect_message`, `shows_message`; `expect_named`, `has_names`; `expect_null`, `is_null`; `expect_output`, `prints_text`; `expect_warning`, `gives_warning`
TeamcityReporter-class

Test reporter: Teamcity format.

Description

This reporter will output results in the Teamcity message format. For more information about Teamcity messages, see http://confluence.jetbrains.com/display/TCD7/Build+Script+Interaction+with+TeamCity

Arguments

... Arguments used to initialise class

testthat R package to make testing fun!

Description

Try the example below. Have a look at the references and learn more from function documentation such as expect_that.

Details

Software testing is important, but, in part because it is frustrating and boring, many of us avoid it. testthat is a new testing framework for R that is easy learn and use, and integrates with your existing workflow.

References

https://github.com/hadley/testthat
http://adv-r.had.co.nz/Testing.html

Examples

library(testthat)
a <- 9
expect_that(a, is_less_than(10))
expect_less_than(a, 10)
test_dir

Run all of the tests in a directory.

Description

Test files start with test and are executed in alphabetical order (but they shouldn’t have dependencies). Helper files start with helper and loaded before any tests are run.

Usage

test_dir(path, filter = NULL, reporter = "summary", env = test_env())

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>path to tests</td>
</tr>
<tr>
<td>filter</td>
<td>If not NULL, only tests with file names matching this regular expression will be executed. Matching will take on the file name after it has been stripped of &quot;test-&quot; and &quot;.r&quot;.</td>
</tr>
<tr>
<td>reporter</td>
<td>reporter to use</td>
</tr>
<tr>
<td>env</td>
<td>environment in which to execute test suite.</td>
</tr>
</tbody>
</table>

Value

a data frame of the summary of test results

test_file

Run all tests in specified file.

Description

Run all tests in specified file.

Usage

test_file(path, reporter = "summary", env = test_env())

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>path</td>
<td>path to file</td>
</tr>
<tr>
<td>reporter</td>
<td>reporter to use</td>
</tr>
<tr>
<td>env</td>
<td>environment in which to execute the tests</td>
</tr>
</tbody>
</table>

Value

a data frame of the summary of test results
test_package

Run all tests in an installed package

Description

Tests are run in an environment that inherits from the package’s namespace environment, so that tests can access non-exported functions and variables. Tests should be placed in either `inst/tests`, or (better) `tests/testthat`.

Usage

test_package(package, filter = NULL, reporter = "summary")

test_check(package, filter = NULL, reporter = "summary")

Arguments

- package: package name
- filter: If not `NULL`, only tests with file names matching this regular expression will be executed. Matching will take on the file name after it has been stripped of "test-" and ".r".
- reporter: reporter to use

Value

a data frame of the summary of test results

R CMD check

Use `test_package` to test an installed package, or in `tests/test-all.R` if you’re using the older `inst/tests` convention.

If your tests live in `tests/testthat` (preferred) use `test_check` in `tests/testthat.R`. You still use `test_package` when testing the installed package.

Examples

```r
## Not run: test_package("testthat")
```
test_that

Create a test.

Description

A test encapsulates a series of expectations about small, self-contained set of functionality. Each test is contained in a context and contains multiple expectation generated by expect_that.

Usage

test_that(desc, code)

Arguments

desc  test name. Names should be kept as brief as possible, as they are often used as line prefixes.

code  test code containing expectations

Details

Tests are evaluated in their own environments, and should not affect global state.

When run from the command line, tests return NULL if all expectations are met, otherwise it raises an error.

Examples

test_that("trigonometric functions match identities", {  
  expect_that(sin(pi / 4), equals(1 / sqrt(2)))  
  expect_that(cos(pi / 4), equals(1 / sqrt(2)))  
  expect_that(tan(pi / 4), equals(1))
})
# Failing test:
## Not run:
test_that("trigonometric functions match identities", {  
  expect_that(sin(pi / 4), equals(1))
})

## END(Not run)
throws_error

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expectation: does expression throw an error?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>throws_error(regexp = NULL, ...)</code></td>
</tr>
</tbody>
</table>

| `expect_error(object, regexp = NULL, ..., info = NULL, label = NULL)` |

<table>
<thead>
<tr>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>regexp</code></td>
</tr>
<tr>
<td><code>...</code></td>
</tr>
<tr>
<td><code>object</code></td>
</tr>
<tr>
<td><code>info</code></td>
</tr>
<tr>
<td><code>label</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>See Also</th>
</tr>
</thead>
<tbody>
<tr>
<td>Other expectations: <code>equals_reference, expect_equal_to_reference; equals, expect_equal; expect_compare, expect_less_than, expect_more_than, is_less_than, is_more_than; expect_equivalent, is_equivalent_to; expect_false, expect_true, is_false, is_true; expect_identical, is_identical_to; expect_is, is_a; expect_match, matches; expect_message, shows_message; expect_named, has_names; expect_null, is_null; expect_output, prints_text; expect_warning, gives_warning; takes_less_than</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>f &lt;- function() stop(&quot;My error!&quot;)</code></td>
</tr>
</tbody>
</table>

| `expect_error(f())` |

| `expect_error(f(), "My error!")` |

# You can use the arguments of grepl to control the matching
| `expect_error(f(), "my error!", ignore.case = TRUE)` |
### watch

**Watch a directory for changes (additions, deletions & modifications).**

**Description**

This is used to power the `auto_test` and `auto_test_package` functions which are used to rerun tests whenever source code changes.

**Usage**

```r
watch(path, callback, pattern = NULL, hash = TRUE)
```

**Arguments**

- **path**
  
  character vector of paths to watch. Omit trailing backslash.

- **callback**
  
  function called everytime a change occurs. It should have three parameters: added, deleted, modified, and should return TRUE to keep watching, or FALSE to stop.

- **pattern**
  
  file pattern passed to `dir`

- **hash**
  
  hashes are more accurate at detecting changes, but are slower for large files. When FALSE, uses modification time stamps

**Details**

Use Ctrl + break (windows), Esc (mac gui) or Ctrl + C (command line) to stop the watcher.

---

### with_mock

**Mock functions in a package.**

**Description**

Executes code after temporarily substituting implementations of package functions. This is useful for testing code that relies on functions that are slow, have unintended side effects or access resources that may not be available when testing.

**Usage**

```r
with_mock(..., .env = topoenv())
```

**Arguments**

- **...**
  
  named parameters redefine mocked functions, unnamed parameters will be evaluated after mocking the functions

- **.env**
  
  the environment in which to patch the functions, defaults to the top-level environment. A character is interpreted as package name.
Details

This works by using some C code to temporarily modify the mocked function in place. On exit (regular or error), all functions are restored to their previous state. This is somewhat abusive of R’s internals, and is still experimental, so use with care.

Value

The result of the last unnamed parameter

References

Suraj Gupta (2012): How R Searches And Finds Stuff

Examples

```r
with_mock(
  all.equal = function(x, y, ...) TRUE,
  expect_equal(2 * 3, 4),
  .env = "base"
)
with_mock(
  ‘base::identical’ = function(x, y, ...) TRUE,
  ‘base::all.equal’ = function(x, y, ...) TRUE,
  expect_equal(x <= 3 * 3, 6),
  expect_identical(x + 4, 9)
)
throws_error(expect_equal(3, 5))
throws_error(expect_identical(3, 5))
```
Index

*Topic **debugging**
  auto_test, 3
  auto_test_package, 4
  ListReporter-class, 20
  MinimalReporter-class, 22
  MultiReporter-class, 23
  SilentReporter-class, 26
  StopReporter-class, 27
  SummaryReporter-class, 27
  TapReporter-class, 28
  TeamcityReporter-class, 29

*Topic **internal.**
  Reporter-class, 25

all.equal, 6, 7, 10
auto_test, 3, 4, 27, 34
auto_test_package, 3, 4, 34
colourise, 4
context, 5, 12, 32
describe, 5
dir, 34
equals, 6, 8, 10, 11, 13–19, 22, 24, 25, 28, 33
equals_reference, 7, 8, 10, 11, 13–19, 22, 24, 25, 28, 33
evaluate, 9
evaluate_promise, 9
expect-compare, 10
expect_equal, 8, 10, 13–19, 22, 24, 25, 28, 33
expect_equal(equals), 6
expect_equal_to_reference, 7, 10, 13–19, 22, 24, 25, 28, 33
expect_equal_to_reference (equals_reference), 8
expect_equivalent, 7, 8, 10, 13–15, 17–19, 22, 24, 25, 28, 33
expect_equivalent(is_equivalent_to), 16
expect_error, 7, 8, 10, 13–19, 22, 24, 25, 28
expect_error (throws_error), 33
expect_false, 7, 8, 10, 13–18, 22, 24, 25, 28, 33
expect_false (is_true), 18
expect_identical, 7, 8, 10, 13–16, 18, 19, 22, 24, 25, 28, 33
expect_identical(is_identical_to), 17
expect_is, 7, 8, 10, 13, 14, 16–19, 22, 24, 25, 28, 33
expect_is (is_a), 15
expect_less_than, 7, 8, 13–19, 22, 24, 25, 28, 33
expect_less_than (expect-compare), 10
expect_match, 7, 8, 10, 13–19, 24, 25, 28, 33
expect_match (matches), 21
expect_message, 7, 8, 10, 13–19, 22, 24, 28, 33
expect_message (shows_message), 25
expect_more_than, 7, 8, 13–19, 22, 24, 25, 28, 33
expect_more_than (expect-compare), 10
expect_named, 7, 8, 10, 13, 15–19, 22, 24, 25, 28, 33
expect_named (has_names), 14
expect_null, 7, 8, 10, 13–17, 19, 22, 24, 25, 28, 33
expect_null (is_null), 18
expect_output, 7, 8, 10, 13–19, 22, 25, 28, 33
expect_output (prints_text), 24
expect_that, 11, 29, 32
expect_true, 7, 8, 10, 13–18, 22, 24, 25, 28, 33
expect_true (is_true), 18
expect_warning, 7, 8, 10, 14–19, 22, 24, 25, 28, 33
expect_warning (gives_warning), 13
fail, 12, 12
gives_warning, 7, 8, 10, 11, 13, 14–19, 22,
INDEX

24, 25, 28, 33
grepl, 22

has_names, 7, 8, 10, 13, 14, 15–19, 22, 24, 25, 28, 33

identical, 17
inherits, 15
is_a, 7, 8, 10, 11, 13, 14, 15, 16–19, 22, 24, 25, 28, 33
is_equivalent_to, 7, 8, 10, 11, 13–15, 16, 17–19, 22, 24, 25, 28, 33
is_false, 7, 8, 10, 11, 13–19, 22, 24, 25, 28, 33
is_false (is_true), 18
is_identical_to, 7, 8, 10, 11, 13–16, 17, 18, 19, 22, 24, 25, 28, 33
is_less_than, 7, 8, 13–19, 22, 24, 25, 28, 33
is_less_than (expect_compare), 10
is_more_than, 7, 8, 13–19, 22, 24, 25, 28, 33
is_more_than (expect_compare), 10
is_null, 7, 8, 10, 13–17, 18, 19, 22, 24, 25, 28, 33
is_true, 7, 8, 10, 11, 13–18, 18, 22, 24, 25, 28, 33

library_if_available, 20
ListReporter (ListReporter-class), 20
ListReporter-class, 20

make_expectation, 21
matches, 7, 8, 10, 11, 13–19, 21, 24, 25, 28, 33
MinimalReporter
  (MinimalReporter-class), 22
MinimalReporter-class, 22
MultiReporter (MultiReporter-class), 23
MultiReporter-class, 23

not, 23

prints_text, 7, 8, 10, 11, 13–19, 22, 24, 25, 28, 33

Reporter (Reporter-class), 25
Reporter-class, 25

shows_message, 7, 8, 10, 12–19, 22, 24, 25, 28, 33
SilentReporter (SilentReporter-class), 26

SilentReporter-class, 26
skip, 26
skip_on_cran (skip), 26
StopReporter (StopReporter-class), 27
StopReporter-class, 27
SummaryReporter
  (SummaryReporter-class), 27
SummaryReporter-class, 27
takes_less_than, 7, 8, 10, 12–19, 22, 24, 25, 28, 33
TapReporter (TapReporter-class), 28
TapReporter-class, 28
TeamcityReporter
  (TeamcityReporter-class), 29
TeamcityReporter-class, 29
test_check (test_package), 31
test_dir, 27, 30
test_file, 27, 30
test_package, 31
test_that, 6, 12, 32
testthat, 29
testthat-package (testthat), 29
throws_error, 7, 8, 10, 11, 13–19, 22, 24, 25, 28, 33

watch, 34
with_mock, 34