

# Package ‘testthat’

February 15, 2012

**Type** Package

**Title** Testthat code. Tools to make testing fun :)

**Version** 0.6

**Author** Hadley Wickham <h.wickham@gmail.com>

**Maintainer** Hadley Wickham <h.wickham@gmail.com>

**Description** A testing package specifically tailored for R that’s fun,flexible and easy to set up.

**URL** <http://had.co.nz/>

**Depends** R (>= 2.8)

**Imports** digest, stringr (>= 0.4), evaluate (>= 0.3), methods

**License** GPL

**LazyData** true

**Collate** ‘auto-test.r’ ‘colour-text.r’ ‘context.r’ ‘expect-that.r’ ‘expectation.r’ ‘expectations.r’ ‘library.r’ ‘reporter.r’ ‘reporter-minimal.r’ ‘reporter-stop.r’ ‘reporter-summary.r’ ‘reporter-zzz.r’ ‘test-files.r’ ‘test-package.r’ ‘test-that.r’ ‘utils.r’ ‘watcher.r’

**Repository** CRAN

**Date/Publication** 2011-12-30 11:23:11

## R topics documented:

auto_test . . . . .	2
auto_test_package . . . . .	3
colourise . . . . .	4
context . . . . .	4
equals . . . . .	5
expect_that . . . . .	6
gives_warning . . . . .	7

is_a . . . . .	8
is_equivalent_to . . . . .	9
is_false . . . . .	10
is_identical_to . . . . .	10
is_true . . . . .	11
library_if_available . . . . .	12
matches . . . . .	13
MinimalReporter . . . . .	14
prints_text . . . . .	14
Reporter . . . . .	15
shows_message . . . . .	15
StopReporter . . . . .	16
SummaryReporter . . . . .	16
takes_less_than . . . . .	16
test_dir . . . . .	17
test_file . . . . .	17
test_package . . . . .	18
test_that . . . . .	18
throws_error . . . . .	19
watch . . . . .	20

## Index 21

---

auto_test	<i>Watches code and tests for changes, rerunning tests as appropriate.</i>
-----------	--

---

### 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 = NULL)
```

### Arguments

<code>code_path</code>	path to directory containing code
<code>test_path</code>	path to directory containing tests
<code>reporter</code>	test reporter to use
<code>env</code>	environment in which to execute test suite. Defaults to new environment inheriting from the global environment.

## 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	<i>Watches a package for changes, rerunning tests as appropriate.</i>
-------------------	---

---

## Description

Watches a package for changes, rerunning tests as appropriate.

## Usage

```
auto_test_package(path, reporter = "summary")
```

## Arguments

path	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, light green, light purple, light red, purple, red, white, yellow

**Examples**

```
print(colourise("Red", "red"))
cat(colourise("Red", "red"), "\n")
cat(colourise("White on red", "white", "red"), "\n")
```

---

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")
```

---

equals	<i>Expectation: is the object equal (with numerical tolerance) to a value?</i>
--------	--

---

**Description**

Comparison performed using [all.equal](#).

**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.
expected.label	Equivalent of label for shortcut form.
...	other values passed to <a href="#">all.equal</a>
object	object to test
info	extra information to be included in the message (useful when writing tests in loops).

**See Also**

Other expectations: [expect\\_equivalent](#), [expect\\_error](#), [expect\\_false](#), [expect\\_identical](#), [expect\\_is](#), [expect\\_match](#), [expect\\_message](#), [expect\\_output](#), [expect\\_true](#), [expect\\_warning](#), [gives\\_warning](#), [is\\_a](#), [is\\_equivalent\\_to](#), [is\\_false](#), [is\\_identical\\_to](#), [is\\_true](#), [matches](#), [prints\\_text](#), [shows\\_message](#), [takes\\_less\\_than](#), [throws\\_error](#)

**Examples**

```
a <- 10
expect_that(a, equals(10))
expect_equal(a, 10)

# Use equals() when testing for numeric equality
sqrt(2) ^ 2 - 1
expect_that(sqrt(2) ^ 2, equals(2))
```

```

expect_equal(sqrt(2) ^ 2, 2)
# Neither of these forms take floating point representation errors into
# account
## Not run:
expect_that(sqrt(2) ^ 2 == 2, is_true())
expect_that(sqrt(2) ^ 2, is_identical_to(2))

## End(Not run)

```

---

expect_that	<i>Expect that a condition holds.</i>
-------------	---------------------------------------

---

## Description

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

## Usage

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

## Details

- [is\\_true](#): truth
- [is\\_false](#): falsehood
- [is\\_a](#): inheritance
- [equals](#): equality with numerical tolerance
- [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
- [shows\\_message](#): message matching
- [takes\\_less\\_than](#): performance

Expectations are arranged into tests with [test\\_that](#) and tests are arranged into contexts with [context](#).

## Examples

```
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)
```

---

gives_warning	<i>Expectation: does expression give a warning?</i>
---------------	---

---

## Description

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

## Usage

```
gives_warning(regex = NULL)

expect_warning(object, regex = NULL, info = NULL,
  label = NULL)
```

## Arguments

regex	optional regular expression to match. If not specified, just asserts that expression gives some warning.
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](#), [expect\\_equal](#), [expect\\_equivalent](#), [expect\\_error](#), [expect\\_false](#), [expect\\_identical](#), [expect\\_is](#), [expect\\_match](#), [expect\\_message](#), [expect\\_output](#), [expect\\_true](#), [is\\_a](#), [is\\_equivalent\\_to](#), [is\\_false](#), [is\\_identical\\_to](#), [is\\_true](#), [matches](#), [prints\\_text](#), [shows\\_message](#), [takes\\_less\\_than](#), [throws\\_error](#)

## Examples

```
expect_that(warning("a"), gives_warning())
expect_that(warning("a"), gives_warning("a"))
```

---

`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

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

## See Also

[inherits](#)

Other expectations: [equals](#), [expect\\_equal](#), [expect\\_equivalent](#), [expect\\_error](#), [expect\\_false](#), [expect\\_identical](#), [expect\\_match](#), [expect\\_message](#), [expect\\_output](#), [expect\\_true](#), [expect\\_warning](#), [gives\\_warning](#), [is\\_equivalent\\_to](#), [is\\_false](#), [is\\_identical\\_to](#), [is\\_true](#), [matches](#), [prints\\_text](#), [shows\\_message](#), [takes\\_less\\_than](#), [throws\\_error](#)

## Examples

```
expect_that(1, is_a("numeric"))
a <- matrix(1:10, nrow = 5)
expect_that(a, is_a("matrix"))

expect_that(mtcars, is_a("data.frame"))
expect_is(mtcars, "data.frame")
# alternatively for classes that have an is method
expect_that(is.data.frame(mtcars), is_true())
# doesn't read quite as nicely
```

---

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

---

## 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

```
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](#), [expect\\_equal](#), [expect\\_error](#), [expect\\_false](#), [expect\\_identical](#), [expect\\_is](#), [expect\\_match](#), [expect\\_message](#), [expect\\_output](#), [expect\\_true](#), [expect\\_warning](#), [gives\\_warning](#), [is\\_a](#), [is\\_false](#), [is\\_identical\\_to](#), [is\\_true](#), [matches](#), [prints\\_text](#), [shows\\_message](#), [takes\\_less\\_than](#), [throws\\_error](#)

## Examples

```
a <- b <- 1:3  
names(b) <- letters[1:3]  
expect_that(a, is_equivalent_to(b, label = b))  
expect_equivalent(a, b)
```

---

is_false	<i>Expectation: is the object false?</i>
----------	--

---

### Description

A useful fall-back expectation like [is\\_true](#)

### Usage

```
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.

### See Also

Other expectations: [equals](#), [expect\\_equal](#), [expect\\_equivalent](#), [expect\\_error](#), [expect\\_identical](#), [expect\\_is](#), [expect\\_match](#), [expect\\_message](#), [expect\\_output](#), [expect\\_true](#), [expect\\_warning](#), [gives\\_warning](#), [is\\_a](#), [is\\_equivalent\\_to](#), [is\\_identical\\_to](#), [is\\_true](#), [matches](#), [prints\\_text](#), [shows\\_message](#), [takes\\_less\\_than](#), [throws\\_error](#)

### Examples

```
expect_that(3 == 2, is_false())
expect_false(3 == 2)

a <- 1:3
expect_that(length(a) == 4, is_false())
```

---

is_identical_to	<i>Expectation: is the object identical to another?</i>
-----------------	---

---

### 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](#), [expect\\_equal](#), [expect\\_equivalent](#), [expect\\_error](#), [expect\\_false](#), [expect\\_is](#), [expect\\_match](#), [expect\\_message](#), [expect\\_output](#), [expect\\_true](#), [expect\\_warning](#), [gives\\_warning](#), [is\\_a](#), [is\\_equivalent\\_to](#), [is\\_false](#), [is\\_true](#), [matches](#), [prints\\_text](#), [shows\\_message](#), [takes\\_less\\_than](#), [throws\\_error](#)

**Examples**

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

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

## End(Not run)
```

---

is\_true

*Expectation: is the object true?*


---

**Description**

This is a fall-back expectation 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.

**Usage**

```
is_true()

expect_true(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**

[is\\_false](#) for complement

Other expectations: [equals](#), [expect\\_equal](#), [expect\\_equivalent](#), [expect\\_error](#), [expect\\_false](#), [expect\\_identical](#), [expect\\_is](#), [expect\\_match](#), [expect\\_message](#), [expect\\_output](#), [expect\\_warning](#), [gives\\_warning](#), [is\\_a](#), [is\\_equivalent\\_to](#), [is\\_false](#), [is\\_identical\\_to](#), [matches](#), [prints\\_text](#), [shows\\_message](#), [takes\\_less\\_than](#), [throws\\_error](#)

**Examples**

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

## End(Not run)
expect_that(!(2 != 2), is_true())
# or better:
expect_that(2 != 2, is_false())

a <- 1:3
expect_that(length(a) == 3, is_true())
# but better to use more specific expectation, if available
expect_that(length(a), equals(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**

```
library_if_available(testthat)
library_if_available(packagethatdoesntexist)
```

---

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(regex, all = TRUE)

expect_match(object, regex, all = TRUE, info = NULL,
             label = NULL)
```

**Arguments**

regex            regular expression to test against

all              should all elements of actual value match regex (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**

[str\\_detect](#) for the function that powers the string matching

Other expectations: [equals](#), [expect\\_equal](#), [expect\\_equivalent](#), [expect\\_error](#), [expect\\_false](#), [expect\\_identical](#), [expect\\_is](#), [expect\\_message](#), [expect\\_output](#), [expect\\_true](#), [expect\\_warning](#), [gives\\_warning](#), [is\\_a](#), [is\\_equivalent\\_to](#), [is\\_false](#), [is\\_identical\\_to](#), [is\\_true](#), [prints\\_text](#), [shows\\_message](#), [takes\\_less\\_than](#), [throws\\_error](#)

**Examples**

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

---

MinimalReporter	<i>Test reporter: minimal.</i>
-----------------	--------------------------------

---

**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.

---

prints_text	<i>Expectation: does printed output match a regular expression?</i>
-------------	---

---

**Description**

Expectation: does printed output match a regular expression?

**Usage**

```
prints_text(regex, ...)

expect_output(object, regex, ..., info = NULL,
              label = NULL)
```

**Arguments**

regex	regular expression to test against
...	other arguments passed to <a href="#">grepl</a>
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](#), [expect\\_equal](#), [expect\\_equivalent](#), [expect\\_error](#), [expect\\_false](#), [expect\\_identical](#), [expect\\_is](#), [expect\\_match](#), [expect\\_message](#), [expect\\_true](#), [expect\\_warning](#), [gives\\_warning](#), [is\\_a](#), [is\\_equivalent\\_to](#), [is\\_false](#), [is\\_identical\\_to](#), [is\\_true](#), [matches](#), [shows\\_message](#), [takes\\_less\\_than](#), [throws\\_error](#)

**Examples**

```
str(mtcars)
expect_that(str(mtcars), prints_text("32 obs"))
expect_that(str(mtcars), prints_text("11 variables"))
expect_output(str(mtcars), "11 variables")
```

---

Reporter	<i>Stub object for managing a reporter of tests.</i>
----------	--

---

**Description**

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

---

shows_message	<i>Expectation: does expression show a message?</i>
---------------	---

---

**Description**

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

**Usage**

```
shows_message(regex = NULL)

expect_message(object, regex = NULL, info = NULL,
               label = NULL)
```

**Arguments**

regex	optional regular expression to match. If not specified, just asserts that expression shows some message.
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](#), [expect\\_equal](#), [expect\\_equivalent](#), [expect\\_error](#), [expect\\_false](#), [expect\\_identical](#), [expect\\_is](#), [expect\\_match](#), [expect\\_output](#), [expect\\_true](#), [expect\\_warning](#), [gives\\_warning](#), [is\\_a](#), [is\\_equivalent\\_to](#), [is\\_false](#), [is\\_identical\\_to](#), [is\\_true](#), [matches](#), [prints\\_text](#), [takes\\_less\\_than](#), [throws\\_error](#)

**Examples**

```
expect_that(message("a"), shows_message())
expect_that(message("a"), shows_message("a"))
```

---

StopReporter	<i>Test reporter: stop on error.</i>
--------------	--------------------------------------

---

### 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.

### 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	<i>Test reporter: summary of errors.</i>
-----------------	--

---

### 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`.

### Details

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

---

takes_less_than	<i>Expectation: does expression take less than a fixed amount of time to run?</i>
-----------------	---

---

### Description

This is useful for performance regression testing.

### Usage

```
takes_less_than(amount)
```

### Arguments

amount	maximum duration in seconds
--------	-----------------------------

**See Also**

Other expectations: [equals](#), [expect\\_equal](#), [expect\\_equivalent](#), [expect\\_error](#), [expect\\_false](#), [expect\\_identical](#), [expect\\_is](#), [expect\\_match](#), [expect\\_message](#), [expect\\_output](#), [expect\\_true](#), [expect\\_warning](#), [gives\\_warning](#), [is\\_a](#), [is\\_equivalent\\_to](#), [is\\_false](#), [is\\_identical\\_to](#), [is\\_true](#), [matches](#), [prints\\_text](#), [shows\\_message](#), [throws\\_error](#)

---

test_dir	<i>Run all of the tests in a directory.</i>
----------	---

---

**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 = NULL)
```

**Arguments**

path	path to tests
reporter	reporter to use
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".
env	environment in which to execute test suite. Defaults to new

---

test_file	<i>Run all tests in specified file.</i>
-----------	---

---

**Description**

Run all tests in specified file.

**Usage**

```
test_file(path, reporter = "summary")
```

**Arguments**

path	path to file
reporter	reporter to use

---

test_package	<i>Run all tests in an installed package</i>
--------------	--

---

**Description**

Tests are run in an environment that inherits from the package environment so that tests can access non-exported functions and variables.

**Usage**

```
test_package(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

**Examples**

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

---

test_that	<i>Create a test.</i>
-----------	-----------------------

---

**Description**

A test encapsulates a series of expectations about a small, self-contained set of functionality. Each test is contained in a [context](#) and contains multiple expectations 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

*Expectation: does expression throw an error?*


---

**Description**

Expectation: does expression throw an error?

**Usage**

```
throws_error(regex = NULL)
```

```
expect_error(object, regex = NULL, info = NULL,
             label = NULL)
```

**Arguments**

regex	optional regular expression to match. If not specified, just asserts that expression throws some error.
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](#), [expect\\_equal](#), [expect\\_equivalent](#), [expect\\_false](#), [expect\\_identical](#), [expect\\_is](#), [expect\\_match](#), [expect\\_message](#), [expect\\_output](#), [expect\\_true](#), [expect\\_warning](#), [gives\\_warning](#), [is\\_a](#), [is\\_equivalent\\_to](#), [is\\_false](#), [is\\_identical\\_to](#), [is\\_true](#), [matches](#), [prints\\_text](#), [shows\\_message](#), [takes\\_less\\_than](#)

**Examples**

```
expect_that(log("a"), throws_error())
expect_error(log("a"))
expect_that(log("a"), throws_error("Non-numeric argument"))
expect_error(log("a"), "Non-numeric argument")
```

---

watch	<i>Watch a directory for changes (additions, deletions &amp; modifications).</i>
-------	--

---

**Description**

This is used to power the [auto\\_test](#) and `link{auto_test_package}` functions which are used to rerun tests whenever source code changes.

**Usage**

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

**Arguments**

path	character vector of paths to watch. Omit trailing backslash.
pattern	file pattern passed to <a href="#">dir</a>
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.
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.

# Index

## \*Topic **debugging**

- auto\_test, [2](#)
- auto\_test\_package, [3](#)
- MinimalReporter, [14](#)
- StopReporter, [16](#)
- SummaryReporter, [16](#)

## \*Topic **internal**

- Reporter, [15](#)

all.equal, [5](#)

auto\_test, [2](#), [3](#), [20](#)

auto\_test\_package, [3](#), [3](#)

colourise, [4](#)

context, [4](#), [6](#), [18](#)

dir, [20](#)

equals, [5](#), [6–15](#), [17](#), [20](#)

expect\_equal, [7–15](#), [17](#), [20](#)

expect\_equal (equals), [5](#)

expect\_equivalent, [5](#), [7](#), [8](#), [10–15](#), [17](#), [20](#)

expect\_equivalent (is\_equivalent\_to), [9](#)

expect\_error, [5](#), [7–15](#), [17](#)

expect\_error (throws\_error), [19](#)

expect\_false, [5](#), [7–9](#), [11–15](#), [17](#), [20](#)

expect\_false (is\_false), [10](#)

expect\_identical, [5](#), [7–10](#), [12–15](#), [17](#), [20](#)

expect\_identical (is\_identical\_to), [10](#)

expect\_is, [5](#), [7](#), [9–15](#), [17](#), [20](#)

expect\_is (is\_a), [8](#)

expect\_match, [5](#), [7–12](#), [14](#), [15](#), [17](#), [20](#)

expect\_match (matches), [13](#)

expect\_message, [5](#), [7–14](#), [17](#), [20](#)

expect\_message (shows\_message), [15](#)

expect\_output, [5](#), [7–13](#), [15](#), [17](#), [20](#)

expect\_output (prints\_text), [14](#)

expect\_that, [6](#), [18](#)

expect\_true, [5](#), [7–11](#), [13–15](#), [17](#), [20](#)

expect\_true (is\_true), [11](#)

expect\_warning, [5](#), [8–15](#), [17](#), [20](#)

expect\_warning (gives\_warning), [7](#)

gives\_warning, [5](#), [6](#), [7](#), [8–15](#), [17](#), [20](#)

grepl, [14](#)

identical, [10](#)

inherits, [8](#)

is\_a, [5–7](#), [8](#), [9–15](#), [17](#), [20](#)

is\_equivalent\_to, [5–8](#), [9](#), [10–15](#), [17](#), [20](#)

is\_false, [5–9](#), [10](#), [11–15](#), [17](#), [20](#)

is\_identical\_to, [5–9](#), [10](#), [10](#), [12–15](#), [17](#), [20](#)

is\_true, [5–10](#), [11](#), [11](#), [13–15](#), [17](#), [20](#)

library\_if\_available, [12](#)

matches, [5–12](#), [13](#), [14](#), [15](#), [17](#), [20](#)

MinimalReporter, [14](#)

prints\_text, [5–13](#), [14](#), [15](#), [17](#), [20](#)

Reporter, [15](#)

shows\_message, [5–14](#), [15](#), [17](#), [20](#)

stop, [16](#)

StopReporter, [16](#)

str\_detect, [13](#)

SummaryReporter, [16](#)

takes\_less\_than, [5–15](#), [16](#), [20](#)

test\_dir, [16](#), [17](#)

test\_file, [16](#), [17](#)

test\_package, [18](#)

test\_that, [6](#), [18](#)

throws\_error, [5–15](#), [17](#), [19](#)

watch, [20](#)