Package 'unittest'

November 1, 2017

Encoding UTF-8

Type Package

Title TAP-Compliant Unit Testing

Version 1.3-0

Date 2017-11-01

Description Concise TAP <http://testanything.org/> compliant unit testing package. Authored tests can be run using CMD check with minimal implementation overhead.

License GPL (>= 3)

Depends R (>= 3.0.0)

Imports

Suggests

BugReports https://github.com/ravingmantis/unittest/issues

LazyData yes

NeedsCompilation no

Author Jamie Lentin [aut, cre],
Anthony Hennessey [aut]

Maintainer Jamie Lentin <jm@ravingmantis.com>

Repository CRAN

Date/Publication 2017-11-01 08:44:30 UTC

R topics documented:

unittest-package *TAP-compliant Unit Testing*

Description

Concise TAP-compliant unit testing package. Authored unit tests can be run using `R CMD check` with minimal implementation overhead. If you want more features there are other unit testing packages (see 'See Also').

Details

The `unittest` package provides two functions, `ok` and `ok_group`. The `ok` function prints `ok` when the expression provided evaluates to `TRUE` and prints `not ok` if the expression evaluates to anything else or results in a runtime error; this is the `TAP` format (`http://testanything.org/`) for reporting test results. The `ok_group` function is a convenience function for grouping related unit tests and produces `TAP` compliant comments in the output to separate the unit test groups.

A unit test summary is produced at the end of a session when a set of unit tests are run in non-interactive mode, for example when the unit tests are run using `Rscript` or by `R CMD check`. For using with `R CMD check`, see 'I'm writing a package, how do I put tests in it?'.

For a list of all documentation use `library(help="unittest")`. Good places to start are the 'Getting Started' and 'FAQ'vignettes. You can see these by typing `vignette('getting_started', package='unittest')` and `vignette('faq', package='unittest')` respectively.

Author(s)

Maintainer: Jamie Lentin <jm@ravingmantis.com>, Anthony Hennessey <ah@ravingmantis.com>.

References

Inspired by Perl's Test::Simple (`http://search.cpan.org/perldoc?Test::Simple`).

See Also

testthat, RUnit, svUnit.

ok *The unittest package's workhorse function*

Description

Report the test of an expression in TAP format.

Usage

```
ok(test, description)
```

Arguments

test	Expression to be tested. Evaluating to TRUE is treated as success, anything else as failure.
description	Character string describing the test. If a description is not given a character representation of the test expression will be used.

Details

See unittest package documentation.

Value

ok() returns whatever was returned when test is evaluated. More importantly it has the side effect of printing the result of the test in TAP format.

Examples

```
## Not run:
ok(1==1, "1 equals 1")
# ok - 1 equals 1

ok(1==1)
# ok - 1 == 1

ok(1==2, "1 equals 2")
# not ok - 1 equals 2
# # Test returned non-TRUE value:
# # [1] FALSE

ok(all.equal(c(1,2),c(1,2)), "compare vectors")
# ok - compare vectors

fn <- function () stop("oops")
ok(fn(), "something with a coding error")
# not ok - something with a coding error
# # Test resulted in error:
# #  oops
# # Whilst evaluating:
# #  fn()

ok(c("Some diagnostic", "messages"), "A failure with diagnostic messages")
# not ok - A failure with diagnostic messages
# # Test returned non-TRUE value:
# # Some diagnostic
# # messages

## End(Not run)
```

ok_group *Group associated unit tests*

Description

Group associated unit tests with TAP compliant comments separating the output.

Usage

```
ok_group(message, tests)
```

Arguments

message	Character vector describing this group. Will be printed as a comment before the tests are ran.
tests	A code block full of tests.

Details

Used to group a selection of tests together, for instance you may group the tests relating to a function together.

Value

Returns NULL.

Examples

```
## Not run:
ok_group("Test addition", {
    ok(1 + 1 == 2, "Can add 1")
    ok(1 + 3 == 4, "Can add 3")
})
ok_group("Test subtraction", {
    ok(1 - 1 == 0, "Can subtract 1")
    ok(1 - 3 == -2, "Can subtract 3")
})
# # Test addition
# ok - Can add 1
# ok - Can add 3
# # Test subtraction
# ok - Can subtract 1
# ok - Can subtract 3

# Multiline group message
ok_group(c("Test multiplication", "but not division"),{
    ok(1 * 1 == 1, "Can multiply by 1")
    ok(2 * 3 == 6, "Can multiply by 3")
})
```

```
  # # Test multiplication
  # # but not division
  # ok - Can multiply by 1
  # ok - Can multiply by 3

## End(Not run)
```

Index