Package 'testit'

April 21, 2026

Type Package

Title A Simple Package for Testing R Packages

Version 0.18

Description Provides two convenience functions assert() and test_pkg() to facilitate testing R packages.

License MIT + file LICENSE

URL https://github.com/yihui/testit

BugReports https://github.com/yihui/testit/issues

RoxygenNote 7.3.3

Encoding UTF-8

NeedsCompilation no

Author Yihui Xie [aut, cre] (ORCID: <https://orcid.org/0000-0003-0645-5666>, URL: https://yihui.org), Tomas Kalibera [ctb], Steven Mortimer [ctb]

Maintainer Yihui Xie <xie@yihui.name>

Repository CRAN

Date/Publication 2026-04-21 05:10:02 UTC

Contents

`assert` *Assertions with an optional message*

Description

The function `assert()` was inspired by `stopifnot()`. It emits a message in case of errors, which can be a helpful hint for diagnosing the errors (`stopifnot()` only prints the possibly truncated source code of the expressions).

The infix operator `%==%` is simply an alias of the `identical()` function to make it slightly easier and intuitive to write test conditions. `x %==% y` is the same as `identical(x, y)`. When it is used inside `assert()`, a message will be printed if the returned value is not `TRUE`, to show the values of the LHS (`x`) and RHS (`y`) via `str()`, which can be helpful for you to check why the assertion failed.

Usage

```
assert(fact, ...)

x %==% y
```

Arguments

`fact`	A message for the assertions when any of them fails; treated the same way as expressions in `...` if it is not a character string, which means you are not required to provide a message to this function.
`...`	An R expression; see Details.
`x, y`	two R objects to be compared

Details

For the `...` argument, it should be a single R expression wrapped in `{}`. This expression may contain multiple sub-expressions. A sub-expression is treated as a test condition if it is wrapped in `()` (meaning its value will be checked to see if it is a logical vector containing any `FALSE` values) , otherwise it is evaluated in the normal way and its value will not be checked. If the value of the last sub-expression is logical, it will also be treated as a test condition.

Value

For `assert()`, invisible `NULL` if all expressions returned `TRUE`, otherwise an error is signaled and the user-provided message is emitted. For `%==%`, `TRUE` or `FALSE`.

Note

The internal implementation of `assert()` is different with the `stopifnot()` function in R **base**: (1) the custom message `fact` is emitted if an error occurs; (2) `assert()` requires the logical values to be non-empty (`logical(0)` will trigger an error); (3) if `...` contains a compound expression in `{}` that returns `FALSE` (e.g., `if (TRUE) {1+1; FALSE}`), the first and the last but one line of the source code from `deparse()` are printed in the error message, otherwise the first line is printed; (4) the

arguments in ... are evaluated sequentially, and `assert()` will signal an error upon the first failed assertion, and will ignore the rest of assertions.

Examples

```
library(testit)
assert("T is bad for TRUE, and so is F for FALSE", {
    T = FALSE
    F = TRUE
    (T != TRUE)  # note the parentheses
    (F != FALSE)
})

assert("A Poisson random number is non-negative", {
    x = rpois(1, 10)
    (x >= 0)
    (x > -1)  # () is optional because it's the last expression
})
```

`has_warning` *Check if an R expression produces warnings or errors*

Description

The two functions `has_warning()` and `has_error()` check if an expression produces warnings and errors, respectively.

Usage

```
has_warning(expr)

has_error(expr, silent = !interactive())
```

Arguments

`expr`	an R expression
`silent`	logical: should the report of error messages be suppressed?

Value

A logical value.

Examples

```
has_warning(1 + 1)
has_warning(1:2 + 1:3)

has_error(2 - 3)
has_error(1 + "a")
has_error(stop("err"), silent = TRUE)
```

`test_pkg` *Run the tests of a package in its namespace*

Description

The tests are executed in a clean environment with the namespace of the package to be tested as the parent environment, which means you can use non-exported objects in the package without having to resort to the triple colon `:::` trick.

Usage

```
test_pkg(package = pkg_name(), dir = c("testit", "tests/testit"), update = NA)
```

Arguments

`package`	The package name. By default, it is detected from the 'DESCRIPTION' file if exists.
`dir`	The directory of the test files; by default, it is the directory 'testit/' or 'tests/testit/' under the current working directory, whichever exists. You can also specify a custom directory.
`update`	If `TRUE`, update snapshot files with actual output instead of comparing. If `NA` (the default), update snapshot files only if they are tracked by GIT (so you can view the diffs in GIT and decide whether to accept or discard the changes). If `FALSE`, never update snapshot files and always compare. For `NA` and `FALSE`, if the snapshot test fails, it will throw an error with a message showing the location of the failed test. For `TRUE`, it will update the snapshot file and never throw an error.

Details

The tests are assumed to be under the 'testit/' or 'tests/testit/' directory by default (depending on your working directory is the package root directory or the 'tests/' directory). The test scripts must be named of the form 'test-*.R' (or 'test-*.md' for snapshot tests); other files will not be treated as test files (but may also be useful, e.g. you can `source()` other scripts in tests).

When a test is executed, the working directory is the same as the directory containing this test, and all existing objects in the test environment will be removed before the code is executed.

See https://pkg.yihui.org/testit/#snapshot-testing for more details about snapshot testing.

Value

`NULL`. All test files are executed, unless an error occurs.

Note

All test scripts must be encoded in UTF-8 if they contain any multibyte characters.

Examples

```
## Not run:
test_pkg("testit")

## End(Not run)
```

Index