Package 'assertthat'

October 12, 2022

Title Easy Pre and Post Assertions

Version 0.2.1

Description An extension to stopifnot() that makes it easy to declare the pre and post conditions that you code should satisfy, while also producing friendly error messages so that your users know what's gone wrong.

License GPL-3

Imports tools

Suggests testthat, covr

RoxygenNote 6.0.1

Collate 'assert-that.r' 'on-failure.r' 'assertions-file.r' 'assertions-scalar.R' 'assertions.r' 'base.r' 'base-comparison.r' 'base-is.r' 'base-logical.r' 'base-misc.r' 'utils.r' 'validate-that.R'

NeedsCompilation no

Author Hadley Wickham [aut, cre]

Maintainer Hadley Wickham <hadley@rstudio.com>

Repository CRAN

Date/Publication 2019-03-21 14:53:46 UTC

R topics documented:

are_equal 2
assert-is 2
assertions-file 3
assert_that 4
has_args 5
has_attr 6
noNA 6
not_empty 7
on_failure 8
scalar 8
validate_that 10

Index **11**

`are_equal` *Are two objects equal?*

Description

Are two objects equal?

Usage

```
are_equal(x, y, ...)
```

Arguments

`x, y`	objects to compare
`...`	additional arguments passed to `all.equal`

See Also

Other assertions: `is.error`, `is.scalar`, `noNA`, `not_empty`

Examples

```
x <- 2
see_if(are_equal(x, 1.9))
see_if(are_equal(x, 1.999, tol = 0.01))
see_if(are_equal(x, 2))
```

`assert-is` *Missing is functions.*

Description

Missing is functions.

Usage

```
is.error(x)

is.time(x)

is.date(x)
```

Arguments

`x`	object to test

See Also

Other assertions: are_equal, is.scalar, noNA, not_empty

Examples

```
a <- Sys.time()
is.time(a)
b <- Sys.Date()
is.date(b)
c <- try(stop("!!"))
is.error(c)
```

---

assertions-file *Useful test related to files*

Description

Useful test related to files

Usage

```
is.dir(path)

is.writeable(path)

is.readable(path)

has_extension(path, ext)
```

Arguments

path	a file path to examine
ext	extension to test for (has_extension only)

Examples

```
see_if(is.dir(1))

tmp <- tempfile()
see_if(file.exists(tmp))
see_if(is.dir(tmp))

writeLines("x", tmp)
see_if(file.exists(tmp))
see_if(is.dir(tmp))
see_if(is.writeable(tmp))
see_if(is.readable(tmp))
unlink(tmp)

see_if(is.readable(tmp))
```

assert_that *Assert that certain conditions are true.*

Description

assert_that is a drop-in replacement for stopifnot but is designed to give informative error messages.

Usage

```
assert_that(..., env = parent.frame(), msg = NULL)

see_if(..., env = parent.frame(), msg = NULL)
```

Arguments

...	unnamed expressions that describe the conditions to be tested. Rather than combining expressions with &&, separate them by commas so that better error messages can be generated.
env	(advanced use only) the environment in which to evaluate the assertions.
msg	a custom error message to be printed if one of the conditions is false.

Assertions

Assertion functions should return a single TRUE or FALSE: any other result is an error, and assert_that will complain about it. This will always be the case for the assertions provided by assertthat, but you may need be a more careful for base R functions.

To make your own assertions that work with assert_that, see the help for on_failure. Alternatively, a custom message can be specified for each call.

See Also

validate_that, which returns a message (not an error) if the condition is false.

Examples

```
x <- 1
# assert_that() generates errors, so can't be usefully run in
# examples
## Not run:
assert_that(is.character(x))
assert_that(length(x) == 3)
assert_that(is.dir("asdf"))
y <- tempfile()
writeLines("", y)
assert_that(is.dir(y))
assert_that(FALSE, msg = "Custom error message")
```

```
## End(Not run)

# But see_if just returns the values, so you'll see that a lot
# in the examples: but remember to use assert_that in your code.
see_if(is.character(x))
see_if(length(x) == 3)
see_if(is.dir(17))
see_if(is.dir("asdf"))
see_if(5 < 3, msg = "Five is not smaller than three")
```

---

has_args *Check a function has specified arguments*

Description

Check a function has specified arguments

Usage

```
has_args(f, args, exact = FALSE)

f %has_args% args
```

Arguments

f	a function
args	a character vector of argument names
exact	if TRUE, argument names must match args exactly (order and value); otherwise f just must have at least args in any order

Examples

```
has_args(mean, "x")
has_args(mean, "x", exact = TRUE)

see_if(mean %has_args% "x")
see_if(mean %has_args% "y")
```

has_attr *Has attribute or name?*

Description

Has attribute or name?

Usage

```
has_attr(x, which)

x %has_attr% which

has_name(x, which)

x %has_name% which
```

Arguments

x	object to test
which	name or attribute

Examples

```
has_attr(has_attr, "fail")
x <- 10
x %has_attr% "a"

y <- list(a = 1, b = 2)
see_if(y %has_name% "c")
```

noNA *Does object contain any missing values?*

Description

Does object contain any missing values?

Usage

```
noNA(x)
```

Arguments

x	object to test

See Also

Other assertions: are_equal, is.error, is.scalar, not_empty

Examples

```
see_if(noNA("a"))
see_if(noNA(c(TRUE, NA)))
x <- sample(c(1:10, NA), 100, rep = TRUE)
see_if(noNA(x))
```

---

not_empty *Check an object doesn't have any empty dimensions*

Description

Check an object doesn't have any empty dimensions

Usage

```
not_empty(x)
```

Arguments

x	object to test

See Also

Other assertions: are_equal, is.error, is.scalar, noNA

Examples

```
not_empty(numeric())
not_empty(mtcars[0, ])
not_empty(mtcars[, 0])
```

on_failure *Custom failure messages for assertions.*

Description

Custom failure messages for assertions.

Usage

```
on_failure(x)

on_failure(x) <- value
```

Arguments

x	a assertion function that returns TRUE if the assertion is met, FALSE otherwise.
value	a function with parameters call and env that returns a custom error message as a string.

Examples

```
is_odd <- function(x) {
  assert_that(is.numeric(x), length(x) == 1)
  x %% 2 == 1
}
see_if(is_odd(2))

on_failure(is_odd) <- function(call, env) {
  paste0(deparse(call$x), " is even")
}
see_if(is_odd(2))
```

scalar *Assert input is a scalar.*

Description

is.scalar provides a generic method for checking input is a scalar. is.string, is.flag, is.number and is.count provide tests for specific types.

Usage

```
is.scalar(x)

is.string(x)

is.number(x)

is.flag(x)

is.count(x)
```

Arguments

x	object to test

See Also

Other assertions: are_equal, is.error, noNA, not_empty

Examples

```
# Generic check for scalars
see_if(is.scalar("a"))
see_if(is.scalar(1:10))

# string = scalar character vector
see_if(is.string(1:3))
see_if(is.string(c("a", "b")))
see_if(is.string("x"))

# number = scalar numeric/integer vector
see_if(is.number(1:3))
see_if(is.number(1.5))

# flag = scalar logical vector
see_if(is.flag(1:3))
see_if(is.flag("a"))
see_if(is.flag(c(FALSE, FALSE, TRUE)))
see_if(is.flag(FALSE))

# count = scalar positive integer
see_if(is.count("a"))
see_if(is.count(-1))
see_if(is.count(1:5))
see_if(is.count(1.5))
see_if(is.count(1))
```

validate_that *Validate that certain conditions are true.*

Description

validate_that is an alternative to the function assert_that, that returns a character vector. This makes them easier to use within S4 "validate" methods.

Usage

```
validate_that(..., env = parent.frame(), msg = NULL)
```

Arguments

...	unnamed expressions that describe the conditions to be tested. Rather than combining expressions with &&, separate them by commas so that better error messages can be generated.
env	(advanced use only) the environment in which to evaluate the assertions.
msg	a custom error message to be printed if one of the conditions is false.

Value

A character vector if the assertion is false, or TRUE if the assertion is true.

See Also

assert_that, which returns an error if the condition is false.

Examples

```
x <- 1
# assert_that() generates errors, so can't be usefully run in
# examples
validate_that(is.numeric(x))
validate_that(is.character(x))
validate_that(length(x) == 3)
validate_that(is.dir("asdf"))
```

Index

%has_args% *(has_args)*, 5
%has_attr% *(has_attr)*, 6
%has_name% *(has_attr)*, 6

all.equal, *2*
are_equal, 2, *3*, *7*, *9*
assert-is, 2
assert_that, 4, *10*
assertions-file, 3

has_args, 5
has_attr, 6
has_extension *(assertions-file)*, 3
has_name *(has_attr)*, 6

is.count *(scalar)*, 8
is.date *(assert-is)*, 2
is.dir *(assertions-file)*, 3
is.error, *2*, *7*, *9*
is.error *(assert-is)*, 2
is.flag *(scalar)*, 8
is.number *(scalar)*, 8
is.readable *(assertions-file)*, 3
is.scalar, *2*, *3*, *7*
is.scalar *(scalar)*, 8
is.string *(scalar)*, 8
is.time *(assert-is)*, 2
is.writeable *(assertions-file)*, 3

noNA, *2*, *3*, 6, *7*, *9*
not_empty, *2*, *3*, *7*, 7, *9*

on_failure, *4*, 8
on_failure<- *(on_failure)*, 8

scalar, 8
see_if *(assert_that)*, 4
stopifnot, *4*

validate_that, *4*, 10