Package: tinyexpect (via r-universe)

August 24, 2024

Title What the Package Does (One Line, Title Case)

Version 0.0.0.9002

Description What the package does (one paragraph).

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.1

Depends R (>= 4.2), tinytest (>= 1.4.1)

Imports methods, rlang (>= 1.1.3), utils

Suggests Matrix, testthat, tools, withr

Repository https://mojaveazure.r-universe.dev

RemoteUrl https://github.com/mojaveazure/tinyexpect

RemoteRef HEAD

RemoteSha 94cd1ff6fad05d57d3328ba0380cce49abea9dc6

Contents

`tinyexpect-package` *tinyexpect: What the Package Does (One Line, Title Case)*

Description

What the package does (one paragraph).

Author(s)

Maintainer: Paul Hoffman `<hoff0792@alumni.umn.edu>` (ORCID)

`expect_invisible` *Expect Invisible*

Description

Expect Invisible

Usage

```
expect_invisible(current, ..., info = NA_character_)
```

Arguments

`current`	`[R object or expression]` Object or expression to be tested
`...`	Ignored
`info`	`[character(1L)]` Optional user-defined message; must be a single character string

Value

A `tinytest` object. A tinytest object is a `logical` with attributes holding information about the test that was run

See Also

testthat equivalent: `testthat::expect_invisible()`

`withVisible()`, `invisible()`

Other visibility: `expect_visible()`

Examples

```
f <- \() invisible()
expect_invisible(f())

g <- \() NULL
expect_invisible(g())
```

`expect_s3_class` *Expect S3 Object*

Description

Check that `current` is an S3 object that inherits from `class`

Usage

```
expect_s3_class(current, class, ..., info = NA_character_, exact = FALSE)
```

Arguments

`current`	`[R object or expression]` Object or expression to be tested
`class`	`[character]` Expected class of `current`; may optionally pass `NA` to assert that `current` is **not** an object
`...`	Ignored
`info`	`[character(1L)]` Optional user-defined message; must be a single character string
`exact`	`[logical(1L)]` If `TRUE`, check that `class(current)` is identical to `class`; otherwise, check that `current` inherits `class`

Value

A `tinytest` object. A tinytest object is a `logical` with attributes holding information about the test that was run

See Also

testthat equivalent: `testthat::expect_s3_class()`

Other inheritance: `expect_s4_class()`, `expect_type()`

Examples

```
expect_s3_class(data.frame(), "data.frame")
expect_s3_class(1L, "integer")
expect_s3_class(1L, NA)

if (requireNamespace("Matrix", quietly = TRUE)) {
  expect_s3_class(Matrix::Matrix(), "Matrix")
}

if (requireNamespace("Matrix", quietly = TRUE)) {
  expect_s3_class(Matrix::Matrix(), NA)
}
```

---

expect_s4_class *Expect S4 Object*

Description

Check that `current` is an S4 object that is of class `class`

Usage

```
expect_s4_class(current, class, ..., info = NA_character_)
```

Arguments

`current`	`[R object or expression]` Object or expression to be tested
`class`	`[character]` Expected class of `current`; may optionally pass `NA` to assert that `current` is **not** an object
`...`	Ignored
`info`	`[character(1L)]` Optional user-defined message; must be a single character string

Value

A `tinytest` object. A tinytest object is a `logical` with attributes holding information about the test that was run

See Also

testthat equivalent: `testthat::expect_s4_class()`

Other inheritance: `expect_s3_class()`, `expect_type()`

Examples

```
if (requireNamespace("Matrix", quietly = TRUE)) {
  expect_s4_class(Matrix::Matrix(), "Matrix")
}

expect_s4_class(data.frame(), "data.frame")
expect_s4_class(data.frame(), NA)
```

`expect_type` *Expect Object Type*

Description

Check that `typeof(current)` is `type`

Usage

```
expect_type(current, type, ..., info = NA_character_)
```

Arguments

`current`	`[R object or expression]` Object or expression to be tested
`type`	`[character(1L)]` Expected base type (as given by `typeof()`)
`...`	Ignored
`info`	`[character(1L)]` Optional user-defined message; must be a single character string

Value

A `tinytest` object. A tinytest object is a `logical` with attributes holding information about the test that was run

See Also

testthat equivalent: `testthat::expect_type()`

Other inheritance: `expect_s3_class()`, `expect_s4_class()`

Examples

```
expect_type(1L, "integer")
expect_type(1.0, "integer")
```

expect_visible *Expect Visible*

Description

Expect Visible

Usage

```
expect_visible(current, ..., info = NA_character_)
```

Arguments

`current`	`[R object or expression]` Object or expression to be tested
`...`	Ignored
`info`	`[character(1L)]` Optional user-defined message; must be a single character string

Value

A `tinytest` object. A tinytest object is a `logical` with attributes holding information about the test that was run

See Also

testthat equivalent: `testthat::expect_visible()`

`withVisible()`

Other visibility: `expect_invisible()`

Examples

```
f <- \() invisible()
expect_visible(f())

g <- \() NULL
expect_visible(g())
```

skip *Stop Testing*

Description

Unconditionally stop testing a **tinytest** test file, preventing additional tests in the file from running without triggering a failure. This is the low-level exit function for other `skip_()` functions in **tinyexpect**

Usage

```
skip(message = "Skipping")
```

Arguments

`message`	A message describing why the test file is being skipped

Value

If called within a **tinytest** test, triggers an exit condition; otherwise, returns `message`

See Also

testthat equivalent: `testthat::skip()`

`tinytest::exit_file()`

Other "stop testing" functions: `skip_if_not_installed()`, `skip_on_bioc()`, `skip_on_ci()`, `skip_on_covr()`, `skip_on_cran()`, `skip_on_os()`

Examples

```
skip()
```

skip_if_not_installed *Stop Testing if a Required Package is Not Installed*

Description

Conditionally stop testing a **tinytest** test file if a required package is not available or not of a minimum required version

Usage

```
skip_if_not_installed(pkg, minimum_version = NULL, quietly = TRUE)

exit_if_not_installed(pkg, minimum_version = NULL, quietly = TRUE)
```

Arguments

`pkg`	Name of package to check for
`minimum_version`	Optional minimum required version of `pkg`
`quietly`	Attempt to find the package quietly; passed to `requireNamespace()`

Value

If called within a **tinytest** test and `pkg` is either not installed or not at least `minimum_version`, triggers an exit condition; otherwise, returns one of

- A string saying that `pkg` is not installed
- A string saying that `pkg` is installed, but not at least `minimum_version`
- `NULL` invisibly

"Skip" vs "Exit"

tinyexpect provides both "`skip_`" and "`exit_`" versions of "stop testing" functions due to the different philosophies of **tinytest** and **testthat**; in **testthat**, tests are encapsulated by `test_that()` to create smaller testing units within a single test file. As such, if a series of tests need to be passed over for some reason, it makes sense to "skip" a `test_that()` block and move on to the next block

tinytest, however, treats each test file as a testing unit. Each file in `inst/tinytest` is equivalent to a **testthat** `test_that()` block; as such, if a series of tests need to be passed over for some reason, it makes sense to "exit" a test file and mnove on to the next file in `inst/tinytest`

In order to provide compatibility with users transitioning from **testthat** to **tinytest**, and to provide continuity with the **tinytest** philosophy, **tinyexpect** provides both `skip_`- and `exit_`- prefixed "stop testing" functions that work identically to one another

See Also

testthat equivalent: `testthat::skip_if_not_installed()`

Other "stop testing" functions: `skip()`, `skip_on_bioc()`, `skip_on_ci()`, `skip_on_covr()`, `skip_on_cran()`, `skip_on_os()`

Examples

```
pkg <- paste(sample(letters, size = 7L, replace = TRUE), collapse = "")
skip_if_not_installed(pkg)

skip_if_not_installed("tinyexpect", minimum_version = "99.0.1")
```

`skip_on_bioc` *Stop Testing on the Bioconductor Build System*

Description

Stop Testing on the Bioconductor Build System

Usage

```
skip_on_bioc()

exit_on_bioc()
```

Value

If called within a **tinytest** test running on the Bioconductor Build System, triggers an exit condition; otherwise, either a string saying "`On Bioconductor`" or `NULL` invisibly

See Also

testthat equivalent: `testthat::skip_on_bioc()`

Other "stop testing" functions: `skip()`, `skip_if_not_installed()`, `skip_on_ci()`, `skip_on_covr()`, `skip_on_cran()`, `skip_on_os()`

Examples

```
if (requireNamespace("withr", quietly = TRUE)) {
  withr::with_envvar(c(IS_BIOC_BUILD_MACHINE = 'true'), skip_on_bioc())
}
```

`skip_on_ci` *Stop Testing on CI*

Description

Stop Testing on CI

Usage

```
skip_on_ci()

exit_on_ci()
```

Value

If called within a **tinytest** test running on CI, triggers an exit condition; otherwise, either a string saying "`On CI`" or `NULL` invisibly

"Skip" vs "Exit"

tinyexpect provides both "`skip_`" and "`exit_`" versions of "stop testing" functions due to the different philosophies of **tinytest** and **testthat**; in **testthat**, tests are encapsulated by `test_that()` to create smaller testing units within a single test file. As such, if a series of tests need to be passed over for some reason, it makes sense to "skip" a `test_that()` block and move on to the next block

tinytest, however, treats each test file as a testing unit. Each file in `inst/tinytest` is equivalent to a **testthat** `test_that()` block; as such, if a series of tests need to be passed over for some reason, it makes sense to "exit" a test file and mnove on to the next file in `inst/tinytest`

In order to provide compatibility with users transitioning from **testthat** to **tinytest**, and to provide continuity with the **tinytest** philosophy, **tinyexpect** provides both `skip_`- and `exit_`- prefixed "stop testing" functions that work identically to one another

See Also

testthat equivalent: `testthat::skip_on_ci()`

Other "stop testing" functions: `skip()`, `skip_if_not_installed()`, `skip_on_bioc()`, `skip_on_covr()`, `skip_on_cran()`, `skip_on_os()`

Examples

```
if (requireNamespace("withr", quietly = TRUE)) {
  withr::with_envvar(c(CI = 'true'), skip_on_ci())
}
```

`skip_on_covr` *Stop Testing Under* R*hrefhttps://CRAN.R-project.org/package=covr***covr**

Description

Stop Testing Under **covr**

Usage

```
skip_on_covr()

exit_on_covr()
```

Value

If called within a **tinytest** test running under **covr**, triggers an exit condition; otherwise, either a string saying "`On covr`" or `NULL` invisibly

"Skip" vs "Exit"

tinyexpect provides both "`skip_`" and "`exit_`" versions of "stop testing" functions due to the different philosophies of **tinytest** and **testthat**; in **testthat**, tests are encapsulated by `test_that()` to create smaller testing units within a single test file. As such, if a series of tests need to be passed over for some reason, it makes sense to "skip" a `test_that()` block and move on to the next block

tinytest, however, treats each test file as a testing unit. Each file in `inst/tinytest` is equivalent to a **testthat** `test_that()` block; as such, if a series of tests need to be passed over for some reason, it makes sense to "exit" a test file and mnove on to the next file in `inst/tinytest`

In order to provide compatibility with users transitioning from **testthat** to **tinytest**, and to provide continuity with the **tinytest** philosophy, **tinyexpect** provides both `skip_`- and `exit_`- prefixed "stop testing" functions that work identically to one another

See Also

testthat equivalent: `testthat::skip_on_covr()`

Other "stop testing" functions: `skip()`, `skip_if_not_installed()`, `skip_on_bioc()`, `skip_on_ci()`, `skip_on_cran()`, `skip_on_os()`

Examples

```
if (requireNamespace("withr", quietly = TRUE)) {
  withr::with_envvar(c(R_COVR = 'true'), skip_on_covr())
}
```

`skip_on_cran` *Stop Testing on CRAN*

Description

Stop Testing on CRAN

Usage

```
skip_on_cran()

exit_on_cran()
```

Value

If called within a **tinytest** test running on CRAN in a non-interactive session, triggers an exit condition; otherwise, either a string saying "`On CRAN`" or `NULL` invisibly

“Skip” vs “Exit”

tinyexpect provides both “`skip_`” and “`exit_`” versions of “stop testing” functions due to the different philosophies of **tinytest** and **testthat**; in **testthat**, tests are encapsulated by `test_that()` to create smaller testing units within a single test file. As such, if a series of tests need to be passed over for some reason, it makes sense to “skip” a `test_that()` block and move on to the next block

tinytest, however, treats each test file as a testing unit. Each file in `inst/tinytest` is equivalent to a **testthat** `test_that()` block; as such, if a series of tests need to be passed over for some reason, it makes sense to “exit” a test file and mnove on to the next file in `inst/tinytest`

In order to provide compatibility with users transitioning from **testthat** to **tinytest**, and to provide continuity with the **tinytest** philosophy, **tinyexpect** provides both `skip_`- and `exit_`- prefixed “stop testing” functions that work identically to one another

See Also

testthat equivalent: `testthat::skip_on_cran()`

Other “stop testing” functions: `skip()`, `skip_if_not_installed()`, `skip_on_bioc()`, `skip_on_ci()`, `skip_on_covr()`, `skip_on_os()`

Examples

```
if (requireNamespace("withr", quietly = TRUE)) {
  withr::with_envvar(c(NOT_CRAN = 'false'), skip_on_cran())
}
```

`skip_on_os` *Stop Testing on Specific Operating Systems and Architectures*

Description

Stop Testing on Specific Operating Systems and Architectures

Usage

```
skip_on_os(os, arch = NULL)

exit_on_os(os, arch = NULL)
```

Arguments

`os`	Operating system to not test on; choose one or more from: • “`windows`” • “`mac`” • “`linux`” • “`solaris`”

	The following OS designations are accepted as synonyms: • "`darwin`": accepted in place of "`mac`" • "`sunos`": accepted in place of "`solaris`" Pass `TRUE` to set `os` to all of the above; this is useful for limiting tests by architecture rather than OS/architecture combos
`arch`	Optional system architectures to not test on; note that this only applies to operating systems present in `os`

Value

If called within a **tinytest** test running under `os` and potentially on an `arch` system, triggers an exit condition; otherwise, returns one of

- A string saying that the code is running under `os`
- A string saying that the code is running under `os` on an `arch` system
- `NULL` invisibly

"Skip" vs "Exit"

tinyexpect provides both "`skip_`" and "`exit_`" versions of "stop testing" functions due to the different philosophies of **tinytest** and **testthat**; in **testthat**, tests are encapsulated by `test_that()` to create smaller testing units within a single test file. As such, if a series of tests need to be passed over for some reason, it makes sense to "skip" a `test_that()` block and move on to the next block

tinytest, however, treats each test file as a testing unit. Each file in `inst/tinytest` is equivalent to a **testthat** `test_that()` block; as such, if a series of tests need to be passed over for some reason, it makes sense to "exit" a test file and mnove on to the next file in `inst/tinytest`

In order to provide compatibility with users transitioning from **testthat** to **tinytest**, and to provide continuity with the **tinytest** philosophy, **tinyexpect** provides both `skip_`- and `exit_`- prefixed "stop testing" functions that work identically to one another

See Also

testthat equivalent: `testthat::skip_on_os()`

Tools for querying system OS and architecture: `Sys.info()`, `R.version[["arch"]]`

Other "stop testing" functions: `skip()`, `skip_if_not_installed()`, `skip_on_bioc()`, `skip_on_ci()`, `skip_on_covr()`, `skip_on_cran()`

Examples

```
(system <- tolower(Sys.info()[["sysname"]]))
skip_on_os(system)

# Nothing happens if on a different OS
(other <- sample(setdiff(c("windows", "mac", "linux", "solaris"), system), size = 1L))
skip_on_os(other)

# System architectures can be used to fine-tune skips
```

```
(sysarch <- R.version$arch)
skip_on_os(system, arch = sysarch)
```

Index