# Package: mockthat (via r-universe)

September 23, 2024

**Title** Function Mocking for Unit Testing

**Version** 0.2.8

**Description** With the deprecation of mocking capabilities shipped with
'testthat' as of 'edition 3' it is left to third-party packages
to replace this functionality, which in some test-scenarios is
essential in order to run unit tests in limited environments
(such as no Internet connection). Mocking in this setting means
temporarily substituting a function with a stub that acts in
some sense like the original function (for example by serving a
HTTP response that has been cached as a file). The only
exported function 'with_mock()' is modeled after the eponymous
'testthat' function with the intention of providing a drop-in
replacement.

**License** MIT + file LICENSE

**URL** `https://nbenn.github.io/mockthat/`

**BugReports** `https://github.com/nbenn/mockthat/issues`

**Depends** R (>= 3.3.0)

**Imports** utils, rlang

**Suggests** testthat, pkgload, curl, jsonlite, withr

**Encoding** UTF-8

**Language** en-US

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.2.1

**Repository** https://nbenn.r-universe.dev

**RemoteUrl** https://github.com/nbenn/mockthat

**RemoteRef** HEAD

**RemoteSha** 4a070bfd39fbc3f7a996ea9ee34b7f1409e8d4b2

# Contents

mock . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
with_mock . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3

**Index** **6**

---

`mock` *Mocking helper functions*

---

## Description

Calls to mock-objects either constructed using `mock()` or returned by `local_mock()` can keep track of how they were called and functions `mock_call()`, `mock_arg/s()` and `mock_n_called()` can be used to retrieve related information.

## Usage

```
mock(expr, env = parent.frame())

mock_call(x, call_no = mock_n_called(x))

mock_args(x, call_no = mock_n_called(x))

mock_arg(x, arg, call_no = mock_n_called(x))

mock_n_called(x)
```

## Arguments

| | |
|---|---|
| `expr` | Expression to be used as body of the function to be mocked. |
| `env` | Environment used as ancestor to the mock function environment. |
| `x` | Object of class `mock_fun` to be queried for call and argument information. |
| `call_no` | The call number of interest (in case the function was called multiple times). |
| `arg` | String-valued argument name to be retrieved. |

## Details

A mocking function can be created either from a single object to be used as return value or from an expression which is used as function body. In both cases, the function signature is inferred from the mock-target. Furthermore, closures constructed by `mock()` are able to keep track of call objects and arguments passed to their respective targets. The following utility functions are available to query this information:

- `mock_call()`: retrieves the call captured by `base::match.call()`
- `mock_arg()`: retrieves the value of the argument with name passed as string-valued argument `arg`

- `mock_args()`: retrieves a list of all arguments used for calling the mocked function
- `mock_n_called()`: counts the number of times the mocked function was called

Calls to mock objects are indexed chronologically and both `mock_call()` and `mock_args()` provide an argument `call_no` which can be used to specify which call is of interest, with the default being the most recent (or last) one.

## Value

- `mock()`: a `mock_fun` object
- `mock_call()`: a call (created by `base::match.call()`)
- `mock_arg()`: the object used as specified function argument
- `mock_args()`: a list of all function arguments used to create a call to the `mock_fun` object in question
- `mock_n_called()`: a scalar integer

## Examples

```
url <- "https://eu.httpbin.org/get?foo=123"

mk <- mock("mocked request")
dl <- function(x) curl::curl(x)

with_mock(`curl::curl` = mk, dl(url))

mock_call(mk)
mock_args(mk)
mock_n_called(mk)

mk <- mock({
  url
})

with_mock(`curl::curl` = mk, dl(url))

my_return_val <- "mocked request"
mk <- mock(my_return_val)

with_mock(`curl::curl` = mk, dl(url))
```

---

`with_mock` *Mock functions in a package.*

---

## Description

Mocking allows you to temporary replace the implementation of functions within a package, which useful for testing code that relies on functions that are slow, have unintended side effects or access resources that may not be available when testing.

Up until recently, such capability was offered via `testthat::with_mock()`, but with release of version 3.0.0 and introduction of edition 3, this was deprecated from 'testthat', leaving it to third party packages to replace this feature. Powered by `utils::assignInNamespace()`, this mocking implementation can be used to stub out both exported and non-exported functions from a package, as well as functions explicitly imported from other packages using either `importFrom` directives or namespaced function calls using `::`.

## Usage

```
with_mock(..., mock_env = pkg_env(), eval_env = parent.frame())

local_mock(
  ...,
  mock_env = pkg_env(),
  eval_env = parent.frame(),
  local_env = eval_env
)
```

## Arguments

| | |
|---|---|
| `...` | Named parameters redefine mocked functions, unnamed parameters will be evaluated after mocking the functions. |
| `mock_env` | The environment in which to patch the functions, defaults to either the package namespace when the environment variable `TESTTHAT_PKG` is set pr the calling environment. A string is interpreted as package name. |
| `eval_env` | Environment in which expressions passed as `...` are evaluated, defaults to `base::parent.frame()`. |
| `local_env` | Passed to `withr::defer()` as `envir` argument (defaults to the values passed as `eval_env`) |

## Details

Borrowing the API from the now-deprecated `testthat::with_mock()`, named arguments passed as `...` are used to define functions to be mocked, where names specify the target functions and the arguments themselves are used as replacement functions. Unnamed arguments passed as `...` will be evaluated in the environment specified as `eval_env` using the mocked functions. Functions to be stubbed should be specified as they would be used in package core. This means that when a function from a third party package is imported, prefixing the function name with `pkg_name::` will not give the desired result. Conversely, if the function is not imported, the package prefix is of course required. On exit of `with_mock()`, the mocked functions are reverted to their original state.

Replacement functions can either be specified as complete functions, or as either quoted expressions, subsequently used as function body or objects used as return values. If functions are created from return values or complete function bodies, they inherit the signatures from the respective functions they are used to mock, alongside the ability to keep track of how they are subsequently called.

A constructor for such mock-objects is available as `mock()`, which quotes the expression passed as `expr`.

If mocking is desirable for multiple separate calls to the function being tested, `local_mock()` is available, which holds onto the mocked state for the lifetime of the environment passed as `local_env` using `withr::defer()`. Unlike `with_mock()`, which returns the result of evaluating the last unnamed argument passed as `...`, `local_mock()` (invisibly) returns the functions used for mocking, which if not fully specified as functions, will be mock-objects described in the previous paragraph.

## Value

The result of the last unnamed argument passed as `...` (evaluated in the environment passed as `eval_env`) in the case of `local_mock()` and a list of functions or `mock_fun` objects (invisibly) for calls to `local_mock()`.

## Examples

```
url <- "https://eu.httpbin.org/get?foo=123"
mok <- function(...) "mocked request"

with_mock(
  `curl::curl_fetch_memory` = mok,
  curl::curl_fetch_memory(url)
)

dl_fun <- function(x) curl::curl_fetch_memory(x)

with_mock(
  `curl::curl_fetch_memory` = mok,
  dl_fun(url)
)

with_mock(
  `curl::curl_fetch_memory` = "mocked request",
  dl_fun(url)
)

dl <- function(x) curl::curl(x)

local({
  mk <- local_mock(`curl::curl` = "mocked request")
  list(dl(url), mock_arg(mk, "url"))
})
```

# Index

base::match.call(), *2*, *3*
base::parent.frame(), *4*

local_mock *(*with_mock*)*, 3

mock, 2
mock_arg *(*mock*)*, 2
mock_args *(*mock*)*, 2
mock_call *(*mock*)*, 2
mock_n_called *(*mock*)*, 2

testthat::with_mock(), *4*

utils::assignInNamespace(), *4*

with_mock, 3
withr::defer(), *4*, *5*