Package 'CBTF'

August 21, 2025

Type Package

Title Caught by the Fuzz! - A Minimalistic Fuzz-Test Runner

Version 0.5.0

Date 2025-08-21

Maintainer Marco Colombo <mar.colombo13@gmail.com>

Description A simple runner for fuzz-testing functions in an R package's public interface. Fuzz testing helps identify functions lacking sufficient argument validation, and uncovers problematic inputs that, while valid by function signature, may cause issues within the function body.

URL https://mcol.github.io/caught-by-the-fuzz/

BugReports https://github.com/mcol/caught-by-the-fuzz/issues

Imports cli (>= 3.6.5)

Suggests testthat (>= 3.2.3), withr (>= 3.0.2)

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

Config/testthat/edition 3

NeedsCompilation no

Author Marco Colombo [aut, cre] (ORCID: <https://orcid.org/0000-0001-6672-0623>)

Repository CRAN

Date/Publication 2025-08-21 08:30:27 UTC

Contents

`CBTF-package` *CBTF: Caught by the Fuzz! A minimalistic fuzz-test runner*

Description

This package implements a very simple mechanism for fuzz-testing functions in the public interface of an R package.

Details

Fuzz testing helps identify functions lacking sufficient argument validation, and uncovers sets of inputs that, while valid by function signature, may cause issues within the function body.

The core functionality of the package is in fuzz, which calls each provided function with a certain input and records the output produced. If an error or a warning is generated, this is captured and reported to the user, unless it matches a pattern of whitelisted messages. The objects returned by fuzz can be inspected with summary.cbtf and print.cbtf.

Whitelisting can also be done after a fuzz run has been completed via the whitelist function, so that only messages that need to be acted upon are actually shown. Using whitelist has the advantage of not requiring the completion of a fuzz run of all functions over all inputs again.

The helper function get_exported_functions identifies the functions in the public interface of a given package, facilitating the generation of the list of functions to be fuzzed.

The helper function test_inputs is invoked by fuzz if the user doesn't specify the set of inputs to be tested. By default generates a large set of potentially problematic inputs, but these can be limited just to the desired classes of inputs.

The helper function namify can be used to generate automatically pretty names in the list of input object, which can improve the output, especially when structures such as data frames, matrices, and more complex objects are involved. These names are based on the deparsed representation of the unevaluated inputs.

Author(s)

Maintainer: Marco Colombo <mar.colombo13@gmail.com> (ORCID)

See Also

Useful links:

- https://mcol.github.io/caught-by-the-fuzz/
- Report bugs at https://github.com/mcol/caught-by-the-fuzz/issues

fuzz *Fuzz-test the specified functions*

Description

This function calls each of the functions in `funs` with each of the objects specified in `what`, recording if any errors or warnings are thrown in the process.

Usage

```
fuzz(
  funs,
  what = test_inputs(),
  package = NULL,
  listify_what = FALSE,
  ignore_patterns = "",
  ignore_warnings = FALSE
)
```

Arguments

`funs`	A character vector of function names to test. If a `"package"` attribute is set and is no `package` argument is provided, functions are loaded from the namespace specified in the attribute.
`what`	A list of objects to be passed, one at a time, as the first argument to each function in `funs`. Ideally, the list should be named, so that each input can be pretty-printed with its corresponding name; function namify provides an automatic way to create a named list. For unnamed lists, a deparsed representation of the inputs will be used, which may appear unwieldy in some cases. If no inputs are provided, a default set of inputs generated by test_inputs will be used.
`package`	A character string specifying the name of the package to search for functions. If `NULL` (default), the function will first check the `"package"` attribute of `funs`, and if that is not set, names will be searched in the global namespace.
`listify_what`	Whether each input in `what` should also be tested in its listified version (`FALSE` by default). When set to `TRUE`, if `what` is `list(x = x)`, the function will operate as if it were `list(x = x, "list(x)" = list(x))`, for any input object `x`.
`ignore_patterns`	One or more strings containing regular expressions to match the errors to ignore. The string "is missing, with no default" is always ignored.
`ignore_warnings`	Whether warnings should be ignored (`FALSE` by default).

Details

In order to reduce the number of false positive results produced, this function applies the following set rules, to establish if an error or warning condition should ignored (whitelisting):

- If the name of the function appears in the error or warning message, as it is considered that the condition has been handled by the developer.
- If the error or warning message contains the text "is missing, with no default", which is produced when a missing argument is used without a value being assigned to it.
- If the error or warning message contains any of the patterns specified in `ignore_patterns`.
- If a warning is thrown but `ignore_warnings = TRUE` is set.

In all whitelisted cases, the result is "OK", and the message that was received is stored in the `$msg` field (see the *Value* section).

Value

An object of class `cbtf` that stores the results obtained for each of the functions tested. This contains the following fields:

`runs` a list of data frames, each containing the results of fuzzing all the functions in `funs` with one of the inputs in `what`. The data frame contains the following columns and attributes:
- `res`: The result of the fuzz test, see below for the possible values.
- `msg`: The error or warning message returned by the function, if any.
- `attr(*, "what")`: The character representation of the input tested.

`funs` a vector of names of the functions tested.

`package` a character string specifying the package name where function names were searched, or `NA` if none was provided.

`ignore_patterns` The value of the `ignore_patterns` argument.

`ignore_warnings` The value of the `ignore_warnings` argument.

The `res` column in each of the data frames in the `$runs` field can contain the following values:

- **OK**: either no error or warning was produced (in which case, the `msg` entry is left blank), or it was whitelisted (in which case, the message received is stored in `msg`).
- **SKIP**: no test was run, either because the given name cannot be found, or it doesn't correspond to a function, or the function accepts no arguments, or the function contains a call to readline; the exact reason is given in `msg`.
- **WARN**: a warning was thrown for which no whitelisting occurred and `ignore_warnings = FALSE`; its message is stored in `msg`.
- **FAIL**: an error was thrown for which no whitelisting occurred; its message is stored in `msg`.

See Also

get_exported_functions, test_inputs, namify, whitelist, summary.cbtf, print.cbtf

Examples

```
## this should produce no errors
res <- fuzz(funs = c("list", "matrix", "mean"),
            what = test_inputs(c("numeric", "raw")))
summary(res)

## display all results even for successful tests
print(res, show_all = TRUE)

## this will catch an error (false positive)
fuzz(funs = "matrix",  what = test_inputs("scalar"))
```

`get_exported_functions` *Get the names of the exported functions of a package*

Description

This function extracts the exports from the namespace of the given package via getNamespaceExports and discards non-fuzzable objects (non-functions and functions with no arguments). The set of names returned can be further restricted via the `ignore_names` argument.

Usage

```
get_exported_functions(package, ignore_names = "")
```

Arguments

`package`	Name of the package to fuzz-test.
`ignore_names`	Names of functions to ignore: these are removed from the names returned. This can be helpful, for example, to discard function aliases.

Value

A character vector of the names of the fuzzable functions exported from the given package, with the `"package"` attribute set. This can be used directly as the `funs` argument of fuzz without need to specify the `package` argument.

See Also

fuzz

Examples

```
## get the fuzzable functions in the public interface of this package
funs <- get_exported_functions("CBTF")
```

length.cbtf *Compute the number of tests performed*

Description

Compute the number of tests performed

Usage

```
## S3 method for class 'cbtf'
length(x)
```

Arguments

x	An object of class cbtf.

Value

An integer corresponding to the number of tests performed in a run.

Examples

```
res <- fuzz(funs = c("list", "matrix", "mean"),
            what = test_inputs(c("numeric", "raw")))
length(res)
```

namify *Add names to a list of inputs*

Description

This function can be used to generate automatically pretty names in a list of custom input object. This can improve the output, especially when structures such as data frames, matrices, and more complex objects are involved.

Usage

```
namify(...)
```

Arguments

...	Objects, possibly named.

Value

A named list containing the evaluated arguments. For unnamed arguments, names are generated by deparsing the unevaluated inputs.

See Also

fuzz

Examples

```
namify(data.frame(a = 1, b = 2))
```

---

print.cbtf *Print the results from a fuzz run*

Description

This formats with colours the results from a fuzz run and prints them to the terminal.

Usage

```
## S3 method for class 'cbtf'
print(x, show_all = FALSE, ...)
```

Arguments

x	An object of class cbtf.
show_all	Whether all results should be printed. By default (FALSE), only the functions that reported an error or a warning are printed. If TRUE, all functions tested are printed, including those that were successful or were skipped.
...	Further arguments passed to or from other methods. These are currently ignored.

Value

No return value, called for side effects.

See Also

summary.cbtf

Examples

```
res <- fuzz(funs = c("list", "matrix", "mean"),
            what = test_inputs(c("numeric", "raw")))
print(res, show.all = TRUE)
```

summary.cbtf *Results summary from a fuzz run*

Description

Reports some summary statistics from the results of a run of fuzz.

Usage

```
## S3 method for class 'cbtf'
summary(object, tabulate = TRUE, ...)
```

Arguments

object	An object of class cbtf.
tabulate	Whether a tabulation of results should be printed out (TRUE by default). The tabulation can always be retrieved from the "summary_table" attribute of the returned object also when tabulate = FALSE.
...	Further arguments passed to or from other methods. These are currently ignored.

Value

A data frame containing the following columns and attributes is returned invisibly:

fun	The names of the function tested.
what	The inputs tested.
res	One of "OK", "FAIL", "WARN" or "SKIP" for each combination of function and input tested (see the *Value* section in fuzz).
msg	The message received in case of error, warning or skip, or an empty string if no failure occurred.
attr(*, "summary_table")	The tabulation of results that was printed out.

See Also

print.cbtf

Examples

```
res <- fuzz(funs = c("list", "matrix", "mean"),
            what = test_inputs(c("numeric", "raw")))
summary(res)
```

test_inputs *Default input tests*

Description

This function provides a selection of potentially problematic inputs by class. List inputs are very limited by design, as they can be automatically generated by setting `listify_what = TRUE` in fuzz.

Usage

```
test_inputs(use = "all", skip = "")
```

Arguments

use	Names of input classes to use. Valid names are "all" (default), "scalar", "numeric", "integer", "logical", "character", "factor", "data.frame", "matrix", "array", "date", "raw" and "list". A vector of valid classes can be retrieved programmatically by setting this argument to "help".
skip	Names of input classes to skip.

Value

A named list of inputs corresponding to the input classes selected, or a character vector of valid input classes if `use = "help"`.

See Also

fuzz

Examples

```
## only the scalar and numeric tests
inputs1 <- test_inputs(use = c("scalar", "numeric"))

## everything but the data, raw and list tests
inputs2 <- test_inputs(skip = c("date", "raw", "list"))

## print the valid input classes
test_inputs("help")
```

`whitelist` *Apply additional whitelist patterns to the results of a fuzz run*

Description

This allows for post-hoc whitelisting of results according to the patterns specified.

Usage

```
whitelist(object, patterns)
```

Arguments

`object`	An object of class `cbtf`.
`patterns`	One or more strings containing regular expressions to match the errors to whitelist.

Value

An object of class `cbtf` with the additional whitelist patterns applied.

See Also

fuzz

Examples

```
## this reports a false positive result
(res <- fuzz(funs = "matrix", what = test_inputs("scalar")))

## with whitelisting, we can remove that
whitelist(res, "must be of a vector type")
```

Index