

# Package: fuzzr (via r-universe)

September 4, 2024

**Type** Package

**Title** Fuzz-Test R Functions

**Version** 0.2.2.9000

**Description** Test function arguments with a wide array of inputs, and produce reports summarizing messages, warnings, errors, and returned values.

**License** MIT + file LICENSE

**URL** <https://github.com/mdlincoln/fuzzr>

**BugReports** <https://github.com/mdlincoln/fuzzr/issues>

**Imports** assertthat, progress, purrr

**Suggests** knitr, rmarkdown, testthat

**VignetteBuilder** knitr

**LazyData** TRUE

**RoxygenNote** 6.0.1

**Repository** <https://mdlincoln.r-universe.dev>

**RemoteUrl** <https://github.com/mdlincoln/fuzzr>

**RemoteRef** HEAD

**RemoteSha** b772ca7164459f8724c2627d9cbf160ac12ee9d7

## Contents

as.data.frame.fuzz_results . . . . .	2
fuzzr . . . . .	2
fuzz_function . . . . .	3
fuzz_results . . . . .	4
test_all . . . . .	5

<b>Index</b>	<b>8</b>
--------------	----------

---

```
as.data.frame.fuzz_results
```

*Summarize fuzz test results as a data frame*

---

### Description

Summarize fuzz test results as a data frame

### Usage

```
## S3 method for class 'fuzz_results'
as.data.frame(x, ..., delim = "; ")
```

### Arguments

x	Object returned by <a href="#">fuzz_function</a> .
...	Additional arguments to be passed to or from methods.
delim	The delimiter to use for fields like messages or warnings in which there may be multiple results.

### Value

A data frame with the following columns:

fuzz_input	The name of the fuzz test performed.
output	Delimited outputs to the command line from the process, if applicable.
messages	Delimited messages, if applicable.
warnings	Delimited warnings, if applicable.
errors	Error returned, if applicable.
value_classes	Delimited classes of the object returned by the function, if applicable
results_index	Index of x from which the summary was produced.

---

```
fuzzr
```

*Fuzz-Test R Functions*

---

### Description

Test function arguments with a wide array of inputs, and produce reports summarizing messages, warnings, errors, and returned values.

---

fuzz_function	<i>Fuzz-test a function</i>
---------------	-----------------------------

---

**Description**

Evaluate how a function responds to unexpected or non-standard inputs.

**Usage**

```
fuzz_function(fun, arg_name, ..., tests = test_all(), check_args = TRUE,
              progress = interactive())
```

```
p_fuzz_function(fun, .l, check_args = TRUE, progress = interactive())
```

**Arguments**

fun	A function.
arg_name	Quoted name of the argument to fuzz test.
...	Other non-dynamic arguments to pass to fun. These will be repeated for every one of the tests.
tests	Which fuzz tests to run. Accepts a named list of inputs, defaulting to <code>test_all</code> .
check_args	Check if <code>arg_name</code> and any arguments passed as ... are accepted by fun. Set to FALSE if you need to pass arguments to a function that accepts arguments via ...
progress	Show a progress bar while running tests?
.l	A named list of tests.

**Details**

`fuzz_function` provides a simple interface to fuzz test a single argument of a function by passing the function, name of the argument, static values of other required arguments, and a named list of test values.

`p_fuzz_function` takes a nested list of arguments paired with lists of tests to run on each argument, and will evaluate every combination of argument and provided test.

**Value**

A `fuzz_results` object.

**Note**

The user will be asked to confirm before proceeding if the combinations of potential tests exceeds 500,000.

**See Also**

[fuzz\\_results](#) and [as.data.frame.fuzz\\_results](#) to access fuzz test results.

**Examples**

```
# Evaluate the 'formula' argument of lm, passing additional required variables
fr <- fuzz_function(lm, "formula", data = iris)

# When evaluating a function that takes ..., set check_args to FALSE
fr <- fuzz_function(paste, "x", check_args = FALSE)

# Pass tests to multiple arguments via a named list
test_args <- list(
  data = test_df(),
  subset = test_all(),
  # Specify custom tests with a new named list
  formula = list(all_vars = Sepal.Length ~ ., one_var = mpg ~ .))
fr <- p_fuzz_function(lm, test_args)
```

---

fuzz_results	<i>Access individual fuzz test results</i>
--------------	--

---

**Description**

Access individual fuzz test results

**Usage**

```
fuzz_value(fr, index = NULL, ...)
```

```
fuzz_call(fr, index = NULL, ...)
```

**Arguments**

fr	fuzz_results object
index	The test index (by position) to access. Same as the results_index in the data frame returned by <a href="#">as.data.frame.fuzz_results</a> .
...	Additional arguments must be named regex patterns that will be used to match against test names. The names of the patterns must match the function argument name(s) whose test names you wish to match.

**Functions**

- `fuzz_value`: Access the object returned by the fuzz test
- `fuzz_call`: Access the call used for the fuzz test

---

test_all	<i>Fuzz test inputs</i>
----------	-------------------------

---

### Description

Each `test_all` returns a named list that concatenates all the available tests specified below.

### Usage

```
test_all()
test_char()
test_int()
test_dbl()
test_lgl()
test_fctr()
test_date()
test_raw()
test_df()
test_null()
```

### Functions

- `test_char`: Character vectors
  - `char_empty`: `character(0)`
  - `char_single`: `"a"`
  - `char_single_blank`: `""`
  - `char_multiple`: `c("a", "b", "c")`
  - `char_multiple_blank`: `c("a", "b", "c", "")`
  - `char_with_na`: `c("a", "b", NA)`
  - `char_single_na`: `NA_character_`
  - `char_all_na`: `c(NA_character_, NA_character_, NA_character_)`
- `test_int`: Integer vectors
  - `int_empty`: `integer(0)`
  - `int_single`: `1L`
  - `int_multiple`: `1:3`
  - `int_with_na`: `c(1L, 2L, NA)`

- int\_single\_na: NA\_integer\_
- int\_all\_na: c(NA\_integer\_, NA\_integer\_, NA\_integer\_)
- test\_dbl: Double vectors
  - dbl\_empty: numeric(0)
  - dbl\_single: 1.5
  - dbl\_multiple: c(1.5, 2.5, 3.5)
  - dbl\_with\_na: c(1.5, 2.5, NA)
  - dbl\_single\_na: NA\_real\_
  - dbl\_all\_na: c(NA\_real\_, NA\_real\_, NA\_real\_)
- test\_lgl: Logical vectors
  - lgl\_empty: logical(0)
  - lgl\_single: TRUE
  - lgl\_multiple: c(TRUE, FALSE, FALSE)
  - lgl\_with\_na: c(TRUE, NA, FALSE)
  - lgl\_single\_na: NA
  - lgl\_all\_na: c(NA, NA, NA)
- test\_fctr: Factor vectors
  - fctr\_empty: structure(integer(0), .Label = character(0), class = "factor")
  - fctr\_single: structure(1L, .Label = "a", class = "factor")
  - fctr\_multiple: structure(1:3, .Label = c("a", "b", "c"), class = "factor")
  - fctr\_with\_na: structure(c(1L, 2L, NA), .Label = c("a", "b"), class = "factor")
  - fctr\_missing\_levels: structure(1:3, .Label = c("a", "b", "c", "d"), class = "factor")
  - fctr\_single\_na: structure(NA\_integer\_, .Label = character(0), class = "factor")
  - fctr\_all\_na: structure(c(NA\_integer\_, NA\_integer\_, NA\_integer\_), .Label = character(0), class = "factor")
- test\_date: Date vectors
  - date\_single: as.Date("2001-01-01")
  - date\_multiple: as.Date(c("2001-01-01", "1950-05-05"))
  - date\_with\_na: as.Date(c("2001-01-01", NA, "1950-05-05"))
  - date\_single\_na: as.Date(NA\_integer\_, origin = "1971-01-01")
  - date\_all\_na: as.Date(rep(NA\_integer\_, 3), origin = "1971-01-01")
- test\_raw: Raw vectors
  - raw\_empty: raw(0)
  - raw\_char: as.raw(0x62),
  - raw\_na: charToRaw(NA\_character\_)
- test\_df: Data frames
  - df\_complete: datasets::iris
  - df\_empty: data.frame(NULL)
  - df\_one\_row: datasets::iris[1, ]
  - df\_one\_col: datasets::iris[, 1]

- df\_with\_na: iris with several NAs added to each column.
- test\_null: Null value
  - null\_value: NULL

# Index

`as.data.frame.fuzz_results`, 2, 4

`fuzz_call` (`fuzz_results`), 4

`fuzz_function`, 2, 3

`fuzz_results`, 4, 4

`fuzz_value` (`fuzz_results`), 4

`fuzzr`, 2

`fuzzr-package` (`fuzzr`), 2

`p_fuzz_function` (`fuzz_function`), 3

`test_all`, 3, 5

`test_char` (`test_all`), 5

`test_date` (`test_all`), 5

`test_dbl` (`test_all`), 5

`test_df` (`test_all`), 5

`test_fctr` (`test_all`), 5

`test_int` (`test_all`), 5

`test_lgl` (`test_all`), 5

`test_null` (`test_all`), 5

`test_raw` (`test_all`), 5