

Package ‘bigassertr’

June 27, 2025

Title Assertion and Message Functions

Version 0.1.7

Date 2025-06-27

Description Enhanced message functions (cat() / message() / warning() / error())
using wrappers around sprintf(). Also, multiple assertion functions
(e.g. to check class, length, values, files, arguments, etc.).

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

Suggests testthat, covr

URL <https://github.com/privefl/bigassertr>

BugReports <https://github.com/privefl/bigassertr/issues>

NeedsCompilation no

Author Florian Privé [aut, cre]

Maintainer Florian Privé <florian.prive.21@gmail.com>

Repository CRAN

Date/Publication 2025-06-27 14:30:02 UTC

Contents

assert	2
printf	4
Index	6

assert

*Assertions***Description**

- `assert_not_null()`: checks that it is not NULL.
- `assert_nona()`: checks that there is no missing value.
- `assert_args()`: checks that `f` is a function and that it has arguments called `args.name`.
- `assert_lengths()`: checks that objects have the same length.
- `assert_one_bool()`: checks whether either TRUE or FALSE.
- `assert_one_int()`: checks that object is integer-ish of length 1.
- `assert_int()`: checks that values are integer-ish (or NULL).
- `assert_pos()`: checks that values are (strictly) positive.
- `assert_01()`: checks that values are either 0 or 1.
- `assert_multiple()`: checks that there are multiple unique values. Errors if there are one unique values (or none). Warns if there are two unique values only.
- `assert_class()`: checks that object is of a particular class.
- `assert_class_or_null()`: checks that object is of a particular class (or NULL).
- `assert_all()`: checks that all values of an object are the same as `value`. Default checks that values are all TRUE. This function tests equality; beware precision errors (e.g. $(0.1 + 0.2) \neq 0.3$).
- `assert_dir()`: checks that directory exists, or tries to create it.
- `assert_exist()`: checks that file exists.
- `assert_noexist()`: checks that file does not exist.
- `assert_nodots()`: checks that `...` are not used in a function.
- `assert_ext()`: checks that file has a particular extension.
- `assert_type()`: checks that values are of a particular type.
- `assert_sorted()`: checks that values are sorted.
- `assert_package()`: checks that a package is installed.
- `assert_df_with_names()`: checks that it is a data frame with some required variables names. There can be additional variables as well.

Usage

```
assert_not_null(x)
```

```
assert_nona(x)
```

```
assert_args(f, args.name)
```

```
assert_lengths(...)  
assert_int(x)  
assert_one_int(x)  
assert_one_bool(x)  
assert_pos(x, strict = TRUE)  
assert_01(x)  
assert_multiple(x)  
assert_class(x, class)  
assert_class_or_null(x, class)  
assert_all(x, value = TRUE)  
assert_dir(dir.path)  
assert_exist(file)  
assert_noexist(file)  
assert_nodots()  
assert_ext(file, ext)  
assert_type(x, type)  
assert_sorted(x, strict = FALSE)  
assert_package(pkg)  
assert_df_with_names(df, names)
```

Arguments

x	Usually a vector.
f	A function.
args.name	Vector of (argument) names to check.
...	Objects to check.
strict	Whether to check for strict positivity? Default is TRUE.
class	Class to check.
value	Value to check.

<code>dir.path</code>	Directory to check.
<code>file</code>	File to check.
<code>ext</code>	Extension to check (without the dot at the beginning).
<code>type</code>	Type to check.
<code>pkg</code>	Name of a package.
<code>df</code>	A data frame.
<code>names</code>	Variable names to check.

Examples

```

assert_not_null(1)
assert_nona(1:3)
assert_args(assert_nona, "x")
assert_lengths(1:3, 4:6, as.list(1:3))
assert_int(c(1, 2, 3))
assert_01(c(0, 1, 0))
assert_multiple(1:3)
assert_class(assert_nona, "function")
assert_all(1:3 > 0)
assert_all(rep(0, 3), 0)
assert_dir(tempdir())
assert_noexist(tmp <- tempfile())
write("test", tmp)
assert_exist(tmp)
assert_ext("test.txt", "txt")
assert_type(1:3, "integer")
assert_sorted(1:3)
assert_package("stats")
assert_df_with_names(iris, c("Sepal.Length", "Sepal.Width"))

test <- function(...) {
  assert_nodots()
  NULL
}
test()
```

printf

Easy messages

Description

Easy messages

Usage

```
printf(...)
```

```
message2(...)
```

```
warning2(...)
```

```
stop2(...)
```

Arguments

... Arguments passed on to [base::sprintf](#).

See Also

[base::sprintf](#)

Examples

```
printf("My name is %s.", "Florian")
```

Index

`assert`, [2](#)
`assert_01 (assert)`, [2](#)
`assert_all (assert)`, [2](#)
`assert_args (assert)`, [2](#)
`assert_class (assert)`, [2](#)
`assert_class_or_null (assert)`, [2](#)
`assert_df_with_names (assert)`, [2](#)
`assert_dir (assert)`, [2](#)
`assert_exist (assert)`, [2](#)
`assert_ext (assert)`, [2](#)
`assert_int (assert)`, [2](#)
`assert_lengths (assert)`, [2](#)
`assert_multiple (assert)`, [2](#)
`assert_nodots (assert)`, [2](#)
`assert_noexist (assert)`, [2](#)
`assert_nona (assert)`, [2](#)
`assert_not_null (assert)`, [2](#)
`assert_one_bool (assert)`, [2](#)
`assert_one_int (assert)`, [2](#)
`assert_package (assert)`, [2](#)
`assert_pos (assert)`, [2](#)
`assert_sorted (assert)`, [2](#)
`assert_type (assert)`, [2](#)

`base::sprintf`, [5](#)

`message2 (printf)`, [4](#)

`printf`, [4](#)

`stop2 (printf)`, [4](#)

`warning2 (printf)`, [4](#)