

# Package ‘assertive.files’

October 12, 2022

**Type** Package

**Title** Assertions to Check Properties of Files

**Version** 0.0-2

**Date** 2016-05-10

**Author** Richard Cotton [aut, cre]

**Maintainer** Richard Cotton <richierocks@gmail.com>

**Description** A set of predicates and assertions for checking the properties of files and connections. This is mainly for use by other package developers who want to include run-time testing features in their own packages. End-users will usually want to use assertive directly.

**URL** <https://bitbucket.org/richierocks/assertive.files>

**BugReports** <https://bitbucket.org/richierocks/assertive.files/issues>

**Depends** R (>= 3.0.0)

**Imports** assertive.base (>= 0.0-2), assertive.numbers

**Suggests** testthat

**License** GPL (>= 3)

**LazyLoad** yes

**LazyData** yes

**Acknowledgments** Development of this package was partially funded by the Proteomics Core at Weill Cornell Medical College in Qatar <<http://qatar-weill.cornell.edu>>. The Core is supported by 'Biomedical Research Program' funds, a program funded by Qatar Foundation.

**Collate** 'imports.R' 'assert-is-connection.R' 'assert-is-file-size.R' 'assert-is-file.R' 'internal-connection.R' 'is-connection.R' 'is-file-size.R' 'is-file.R'

**RoxygenNote** 5.0.1

**ByteCompile** true

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2016-05-10 10:30:32

## R topics documented:

as.character.file . . . . .	2
assert_all_are_dirs . . . . .	3
assert_all_are_empty_files . . . . .	3
assert_all_are_executable_files . . . . .	5
assert_all_are_existing_files . . . . .	6
assert_all_are_libraries . . . . .	7
assert_is_bzfile_connection . . . . .	8

<b>Index</b>	<b>12</b>
--------------	-----------

---

as.character.file	<i>Convert file connections to strings</i>
-------------------	--

---

### Description

as.character method for file connections.

### Usage

```
## S3 method for class 'file'
as.character(x, ...)
```

### Arguments

x	A file connection.
...	Not currently used.

### Value

A string containing the target location of the file connection.

### See Also

[file](#), [summary.connection](#), [as.character](#)

### Examples

```
rprofile <- file.path(R.home("etc"), "Rprofile.site")
fcon <- file(rprofile)
assertive.base::assert_all_are_true(identical(as.character(fcon), rprofile))
close(fcon)
```

---

assert\_all\_are\_dirs *Is the path a directory? Checks to see if the input path is a directory.*

---

### Description

Is the path a directory? Checks to see if the input path is a directory.

### Usage

```
assert_all_are_dirs(x, severity = getOption("assertive.severity", "stop"))
```

```
assert_any_are_dirs(x, severity = getOption("assertive.severity", "stop"))
```

```
is_dir(x, .xname = get_name_in_parent(x))
```

### Arguments

x	File paths.
severity	How severe should the consequences of the assertion be? Either "stop", "warning", "message", or "none".
.xname	Not intended to be used directly.

### Value

is\_dir returns TRUE if and only if the input path is a directory that exists, as determined by file.info.

### Examples

```
assert_all_are_dirs(R.home())
```

---

assert\_all\_are\_empty\_files  
*Is a file too big or small?*

---

### Description

Checks to see if a file is within a given size range.

**Usage**

```

assert_all_are_empty_files(x, severity = getOption("assertive.severity",
  "stop"))

assert_any_are_empty_files(x, severity = getOption("assertive.severity",
  "stop"))

assert_all_are_non_empty_files(x, severity = getOption("assertive.severity",
  "stop"))

assert_any_are_non_empty_files(x, severity = getOption("assertive.severity",
  "stop"))

assert_all_file_sizes_are_in_range(x, lower = 0, upper = Inf,
  lower_is_strict = FALSE, upper_is_strict = FALSE, na_ignore = FALSE,
  severity = getOption("assertive.severity", "stop"))

assert_any_file_sizes_are_in_range(x, lower = 0, upper = Inf,
  lower_is_strict = FALSE, upper_is_strict = FALSE,
  severity = getOption("assertive.severity", "stop"))

is_empty_file(x, .xname = get_name_in_parent(x))

is_non_empty_file(x, .xname = get_name_in_parent(x))

is_file_size_in_range(x, lower = 0, upper = Inf, lower_is_strict = FALSE,
  upper_is_strict = FALSE, .xname = get_name_in_parent(x))

```

**Arguments**

<code>x</code>	Input to check.
<code>severity</code>	How severe should the consequences of the assertion be? Either "stop", "warning", "message", or "none".
<code>lower</code>	Smallest file size allowed, in bytes.
<code>upper</code>	Largest file size allowed, in bytes.
<code>lower_is_strict</code>	If TRUE, the lower bound is open (strict) otherwise it is closed.
<code>upper_is_strict</code>	If TRUE, the upper bound is open (strict) otherwise it is closed.
<code>na_ignore</code>	A logical value. If FALSE, NA values cause an error; otherwise they do not. Like <code>na.rm</code> in many stats package functions, except that the position of the failing values does not change.
<code>.xname</code>	Not intended to be used directly.

**Value**

is\_empty\_file wraps file.info, returning TRUE when the input is a file that exists with size zero. assert\_\*\_are\_empty\_files return nothing but throws an error if is\_empty\_file returns FALSE.

**See Also**

[file.info](#).

**Examples**

```
tf <- tempfile()
file.create(tf)
is_empty_file(tf)
cat("some stuff", file = tf)
is_non_empty_file(tf)
assertive.base::dont_stop(assert_all_file_sizes_are_in_range(tf, lower = 100))
unlink(tf)
```

---

assert\_all\_are\_executable\_files

*Is the file accessible?*

---

**Description**

Checks to see if the input files can be executed/read/written to.

**Usage**

```
assert_all_are_executable_files(x, warn_about_windows = TRUE,
  severity = getOption("assertive.severity", "stop"))
```

```
assert_any_are_executable_files(x, warn_about_windows = TRUE,
  severity = getOption("assertive.severity", "stop"))
```

```
assert_all_are_readable_files(x, warn_about_windows = TRUE,
  severity = getOption("assertive.severity", "stop"))
```

```
assert_any_are_readable_files(x, warn_about_windows = TRUE,
  severity = getOption("assertive.severity", "stop"))
```

```
assert_all_are_writable_files(x, warn_about_windows = TRUE,
  severity = getOption("assertive.severity", "stop"))
```

```
assert_any_are_writable_files(x, warn_about_windows = TRUE,
  severity = getOption("assertive.severity", "stop"))
```

```
is_executable_file(x, warn_about_windows = TRUE,
```

```

    .xname = get_name_in_parent(x))

is_ex_file(x)

is_readable_file(x, warn_about_windows = TRUE,
  .xname = get_name_in_parent(x))

is_writable_file(x, warn_about_windows = TRUE,
  .xname = get_name_in_parent(x))

```

### Arguments

x	Input to check.
warn_about_windows	Logical. If TRUE, then calling the function under Windows will throw a warning about the problems with <a href="#">file.access</a> .
severity	How severe should the consequences of the assertion be? Either "stop", "warning", "message", or "none".
.xname	Not intended to be used directly.

### Value

is\_executable\_file wraps file.access, showing the names of the inputs in the answer. assert\_is\_executable\_file returns nothing but throws an error if is\_executable\_file returns FALSE.

### See Also

[file.access](#).

### Examples

```

files <- dir()
is_readable_file(files)
is_writable_file(files, warn_about_windows = FALSE)
is_executable_file(files, warn_about_windows = FALSE)

```

---

assert\_all\_are\_existing\_files

*Does the file exist?*

---

### Description

Checks to see if the input files exist.

**Usage**

```
assert_all_are_existing_files(x, severity = getOption("assertive.severity",
  "stop"))

assert_any_are_existing_files(x, severity = getOption("assertive.severity",
  "stop"))

is_existing_file(x, .xname = get_name_in_parent(x))
```

**Arguments**

x	Input to check.
severity	How severe should the consequences of the assertion be? Either "stop", "warning", "message", or "none".
.xname	Not intended to be used directly.

**Value**

is\_existing\_file wraps file.exists, showing the names of the inputs in the answer. assert\_\*\_are\_existing\_files return nothing but throws an error if is\_existing\_file returns FALSE.

**Note**

Trailing slashes are removed from paths to avoid a lot of false negatives by the underlying function file.exists.

**See Also**

[file.exists](#).

**Examples**

```
assert_all_are_existing_files(dir())
# These examples should fail.
assertive.base::dont_stop(
  assert_all_are_existing_files("not an existing file (probably)")
)
```

---

assert\_all\_are\_libraries

*Is the directory a known R library?*

---

**Description**

Checks to see if the input directories are known R libraries.

**Usage**

```

assert_all_are_libraries(x, severity = getOption("assertive.severity",
  "stop"))

assert_any_are_libraries(x, severity = getOption("assertive.severity",
  "stop"))

is_library(x, .xname = get_name_in_parent(x))

```

**Arguments**

x	Directory paths
severity	How severe should the consequences of the assertion be? Either "stop", "warning", "message", or "none".
.xname	Not intended to be used directly.

**Value**

is\_library returns TRUE if and only if the input paths are known R package libraries. That is, they must be paths returned by .libPaths.

**Note**

Input paths are converted to character, and then normalized using normalizePaths.

**Examples**

```
is_library(c(.libPaths(), R.home()))
```

---

```

assert_is_bzfile_connection
  Is the input a connection?

```

---

**Description**

Various checks to see if the input is a (particular type of/open/incomplete) connection.

**Usage**

```

assert_is_bzfile_connection(x, severity = getOption("assertive.severity",
  "stop"))

assert_is_connection(x, severity = getOption("assertive.severity", "stop"))

assert_is_fifo_connection(x, severity = getOption("assertive.severity",
  "stop"))

```



```
assert_is_file_connection(x, severity = getOption("assertive.severity",
"stop"))

assert_is_gzfile_connection(x, severity = getOption("assertive.severity",
"stop"))

assert_is_incomplete_connection(x, severity = getOption("assertive.severity",
"stop"))

assert_is_open_connection(x, rw = "",
severity = getOption("assertive.severity", "stop"))

assert_is_pipe_connection(x, severity = getOption("assertive.severity",
"stop"))

assert_is_readable_connection(x, severity = getOption("assertive.severity",
"stop"))

assert_is_socket_connection(x, severity = getOption("assertive.severity",
"stop"))

assert_is_stderr(x, severity = getOption("assertive.severity", "stop"))

assert_is_stdin(x, severity = getOption("assertive.severity", "stop"))

assert_is_stdout(x, severity = getOption("assertive.severity", "stop"))

assert_is_terminal_connection(x, severity = getOption("assertive.severity",
"stop"))

assert_is_text_connection(x, severity = getOption("assertive.severity",
"stop"))

assert_is_unz_connection(x, severity = getOption("assertive.severity",
"stop"))

assert_is_url_connection(x, severity = getOption("assertive.severity",
"stop"))

assert_is_writable_connection(x, severity = getOption("assertive.severity",
"stop"))

assert_is_xzfile_connection(x, severity = getOption("assertive.severity",
"stop"))

is_bzfile_connection(x, .xname = get_name_in_parent(x))

is_connection(x, .xname = get_name_in_parent(x))
```

```
is_fifo_connection(x, .xname = get_name_in_parent(x))
is_file_connection(x, .xname = get_name_in_parent(x))
is_gzfile_connection(x, .xname = get_name_in_parent(x))
is_incomplete_connection(x, .xname = get_name_in_parent(x))
is_open_connection(x, rw = "", .xname = get_name_in_parent(x))
is_pipe_connection(x, .xname = get_name_in_parent(x))
is_readable_connection(x, .xname = get_name_in_parent(x))
is_socket_connection(x, .xname = get_name_in_parent(x))
is_stderr(x, .xname = get_name_in_parent(x))
is_stdin(x, .xname = get_name_in_parent(x))
is_stdout(x, .xname = get_name_in_parent(x))
is_terminal_connection(x, .xname = get_name_in_parent(x))
is_text_connection(x, .xname = get_name_in_parent(x))
is_unz_connection(x, .xname = get_name_in_parent(x))
is_url_connection(x, .xname = get_name_in_parent(x))
is_writable_connection(x, .xname = get_name_in_parent(x))
is_xzfile_connection(x, .xname = get_name_in_parent(x))
```

### Arguments

x	Input to check.
severity	How severe should the consequences of the assertion be? Either "stop", "warning", "message", or "none".
rw	Read-write status of connection. Passed to isOpen.
.xname	Not intended to be used directly.

### Value

is\_connection checks for objects of class "connection". is\_open\_connection and is\_incomplete\_connection wrap isOpen and isIncomplete respectively, providing more information on failure. is\_readable\_connection and is\_writable\_connection tell you whether the connection is readable from or writable to.

is\_bzfile\_connection, is\_fifo\_connection, is\_file\_connection, is\_pipe\_connection, is\_socket\_connection, is\_stderr, is\_stdin, is\_stdout, is\_text\_connection, is\_unz\_connection, is\_url\_connection and is\_xzfile\_connection give more specific tests on the type of connection. The assert\_\* functions return nothing but throw an error if the corresponding is\_\* function returns FALSE.

### Note

is\_incomplete\_connection will return false for closed connections, regardless of whether or not the connection ends with a newline character. (isIncomplete throws an error for closed connections.)

### See Also

[isOpen](#).

### Examples

```
assert_is_terminal_connection(stdin())
assert_is_readable_connection(stdin())
assert_is_open_connection(stdin())
assert_is_stdin(stdin())
# Next line is usually true but, e.g., devtools::run_examples overrides it
assertive.base::dont_stop(assert_is_terminal_connection(stdout()))
assert_is_writable_connection(stdout())
assert_is_open_connection(stdout())
assert_is_stdout(stdout())
assert_is_terminal_connection(stderr())
assert_is_writable_connection(stderr())
assert_is_open_connection(stderr())
assert_is_stderr(stderr())
tcon <- textConnection("txt", "w", local = TRUE)
assert_is_text_connection(tcon)
assert_is_open_connection(tcon)
cat("this has no final newline character", file = tcon)
assert_is_incomplete_connection(tcon)
close(tcon)
# These examples should fail.
assertive.base::dont_stop({
  assert_is_connection("not a connection")
  assert_is_readable_connection(stdout())
  assert_is_writable_connection(stdin())
})
## Not run:
fcon <- file()
close(fcon)
assert_is_open_connection(fcon)

## End(Not run)
```

# Index

as.character, 2  
as.character.file, 2  
assert\_all\_are\_dirs, 3  
assert\_all\_are\_empty\_files, 3  
assert\_all\_are\_executable\_files, 5  
assert\_all\_are\_existing\_files, 6  
assert\_all\_are\_libraries, 7  
assert\_all\_are\_non\_empty\_files  
    (assert\_all\_are\_empty\_files), 3  
assert\_all\_are\_readable\_files  
    (assert\_all\_are\_executable\_files),  
    5  
assert\_all\_are\_writable\_files  
    (assert\_all\_are\_executable\_files),  
    5  
assert\_all\_file\_sizes\_are\_in\_range  
    (assert\_all\_are\_empty\_files), 3  
assert\_any\_are\_dirs  
    (assert\_all\_are\_dirs), 3  
assert\_any\_are\_empty\_files  
    (assert\_all\_are\_empty\_files), 3  
assert\_any\_are\_executable\_files  
    (assert\_all\_are\_executable\_files),  
    5  
assert\_any\_are\_existing\_files  
    (assert\_all\_are\_existing\_files),  
    6  
assert\_any\_are\_libraries  
    (assert\_all\_are\_libraries), 7  
assert\_any\_are\_non\_empty\_files  
    (assert\_all\_are\_empty\_files), 3  
assert\_any\_are\_readable\_files  
    (assert\_all\_are\_executable\_files),  
    5  
assert\_any\_are\_writable\_files  
    (assert\_all\_are\_executable\_files),  
    5  
assert\_any\_file\_sizes\_are\_in\_range  
    (assert\_all\_are\_empty\_files), 3  
assert\_is\_bzfile\_connection, 8  
assert\_is\_connection  
    (assert\_is\_bzfile\_connection),  
    8  
assert\_is\_fifo\_connection  
    (assert\_is\_bzfile\_connection),  
    8  
assert\_is\_file\_connection  
    (assert\_is\_bzfile\_connection),  
    8  
assert\_is\_gzfile\_connection  
    (assert\_is\_bzfile\_connection),  
    8  
assert\_is\_incomplete\_connection  
    (assert\_is\_bzfile\_connection),  
    8  
assert\_is\_open\_connection  
    (assert\_is\_bzfile\_connection),  
    8  
assert\_is\_pipe\_connection  
    (assert\_is\_bzfile\_connection),  
    8  
assert\_is\_readable\_connection  
    (assert\_is\_bzfile\_connection),  
    8  
assert\_is\_socket\_connection  
    (assert\_is\_bzfile\_connection),  
    8  
assert\_is\_stderr  
    (assert\_is\_bzfile\_connection),  
    8  
assert\_is\_stdin  
    (assert\_is\_bzfile\_connection),  
    8  
assert\_is\_stdout  
    (assert\_is\_bzfile\_connection),  
    8  
assert\_is\_terminal\_connection  
    (assert\_is\_bzfile\_connection),

8  
 assert\_is\_text\_connection  
     (assert\_is\_bzfile\_connection),  
     8  
 assert\_is\_unz\_connection  
     (assert\_is\_bzfile\_connection),  
     8  
 assert\_is\_url\_connection  
     (assert\_is\_bzfile\_connection),  
     8  
 assert\_is\_writable\_connection  
     (assert\_is\_bzfile\_connection),  
     8  
 assert\_is\_xzfile\_connection  
     (assert\_is\_bzfile\_connection),  
     8  
  
 file, 2  
 file.access, 6  
 file.exists, 7  
 file.info, 5  
  
 is\_bzfile\_connection  
     (assert\_is\_bzfile\_connection),  
     8  
 is\_connection  
     (assert\_is\_bzfile\_connection),  
     8  
 is\_dir(assert\_all\_are\_dirs), 3  
 is\_empty\_file  
     (assert\_all\_are\_empty\_files), 3  
 is\_ex\_file  
     (assert\_all\_are\_executable\_files),  
     5  
 is\_executable\_file  
     (assert\_all\_are\_executable\_files),  
     5  
 is\_existing\_file  
     (assert\_all\_are\_existing\_files),  
     6  
 is\_fifo\_connection  
     (assert\_is\_bzfile\_connection),  
     8  
 is\_file\_connection  
     (assert\_is\_bzfile\_connection),  
     8  
 is\_file\_size\_in\_range  
     (assert\_all\_are\_empty\_files), 3  
  
 is\_gzfile\_connection  
     (assert\_is\_bzfile\_connection),  
     8  
 is\_incomplete\_connection  
     (assert\_is\_bzfile\_connection),  
     8  
 is\_library(assert\_all\_are\_libraries), 7  
 is\_non\_empty\_file  
     (assert\_all\_are\_empty\_files), 3  
 is\_open\_connection  
     (assert\_is\_bzfile\_connection),  
     8  
 is\_pipe\_connection  
     (assert\_is\_bzfile\_connection),  
     8  
 is\_readable\_connection  
     (assert\_is\_bzfile\_connection),  
     8  
 is\_readable\_file  
     (assert\_all\_are\_executable\_files),  
     5  
 is\_socket\_connection  
     (assert\_is\_bzfile\_connection),  
     8  
 is\_stderr  
     (assert\_is\_bzfile\_connection),  
     8  
 is\_stdin(assert\_is\_bzfile\_connection),  
     8  
 is\_stdout  
     (assert\_is\_bzfile\_connection),  
     8  
 is\_terminal\_connection  
     (assert\_is\_bzfile\_connection),  
     8  
 is\_text\_connection  
     (assert\_is\_bzfile\_connection),  
     8  
 is\_unz\_connection  
     (assert\_is\_bzfile\_connection),  
     8  
 is\_url\_connection  
     (assert\_is\_bzfile\_connection),  
     8  
 is\_writable\_connection  
     (assert\_is\_bzfile\_connection),  
     8  
 is\_writable\_file

(assert\_all\_are\_executable\_files),  
5  
is\_xzfile\_connection  
(assert\_is\_bzfile\_connection),  
8  
isOpen, 11  
summary.connection, 2