

# Package ‘assertive.sets’

October 12, 2022

**Type** Package

**Title** Assertions to Check Properties of Sets

**Version** 0.0-3

**Date** 2016-12-30

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

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

**Description** A set of predicates and assertions for checking the properties of sets. 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.sets>

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

**Depends** R (>= 3.0.0)

**Imports** assertive.base (>= 0.0-7)

**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.

**RoxygenNote** 5.0.1

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2016-12-30 19:35:49

## R topics documented:

assert\_are\_disjoint\_sets . . . . . 2

**Index** . . . . . 5

assert\_are\_disjoint\_sets  
*Set comparisons*

### Description

Checks on the contents of two vectors (ignoring the order of the elements).

### Usage

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

assert_are_intersecting_sets(x, y, severity = getOption("assertive.severity",
  "stop"))

assert_are_set_equal(x, y, severity = getOption("assertive.severity", "stop"))

assert_is_set_equal(x, y, severity = getOption("assertive.severity", "stop"))

assert_is_subset(x, y, strictly = FALSE,
  severity = getOption("assertive.severity", "stop"))

assert_is_superset(x, y, strictly = FALSE,
  severity = getOption("assertive.severity", "stop"))

are_disjoint_sets(x, y, .xname = get_name_in_parent(x),
  .yname = get_name_in_parent(y))

are_intersecting_sets(x, y, .xname = get_name_in_parent(x),
  .yname = get_name_in_parent(y))

are_set_equal(x, y, .xname = get_name_in_parent(x),
  .yname = get_name_in_parent(y))

is_set_equal(x, y, .xname = get_name_in_parent(x),
  .yname = get_name_in_parent(y))

is_subset(x, y, strictly = FALSE, .xname = get_name_in_parent(x),
  .yname = get_name_in_parent(y))

is_superset(x, y, strictly = FALSE, .xname = get_name_in_parent(x),
  .yname = get_name_in_parent(y))
```

**Arguments**

x	A vector.
y	Another vector.
severity	How severe should the consequences of the assertion be? Either "stop", "warning", "message", or "none".
strictly	Logical. If TRUE, x and y should not be set equal.
.xname	Not intended to be used directly.
.yname	Not intended to be used directly.

**Value**

The `is_*` functions return TRUE or FALSE. The `assert_*` functions throw an error in the event of failure.

**See Also**

[sets](#), [set\\_is\\_equal](#)

**Examples**

```
# Same contents, different order, returns TRUE
are_set_equal(1:5, 5:1)

# Different lengths
are_set_equal(1:5, 1:6)

# First vector contains values not in second vector
are_set_equal(1:5, c(1:4, 4))

# Second vector contains values not in first vector
are_set_equal(c(1:4, 4), 1:5)

# Is x a subset of y?
is_subset(1:4, 1:5)
is_subset(1:5, 1:4)

# Is x a superset of y?
is_superset(1:5, 1:4)
is_superset(1:4, 1:5)

# The strictly argument checks for a strict sub/superset
is_subset(1:5, 1:5, strictly = TRUE)
is_superset(1:5, 1:5, strictly = TRUE)

# Do x and y have common elements?
are_intersecting_sets(1:4, 3:6)
are_disjoint_sets(1:4, 3:6)

# Types are coerced to be the same, as per base::setdiff
are_set_equal(1:4, c("4", "3", "2", "1"))
```

```
# Errors are thrown in the event of failure
assert_are_set_equal(1:5, 5:1)
assertive.base::dont_stop(assert_are_set_equal(1:5, 1:6))

assert_is_subset(1:4, 1:5)
assertive.base::dont_stop(assert_is_subset(1:5, 1:4))

assert_is_superset(1:5, 1:4)
assertive.base::dont_stop(assert_is_superset(1:4, 1:5))

# A common use case: checking that data contains required columns
required_cols <- c("Time", "weight", "Diet")
assert_is_superset(colnames(ChickWeight), required_cols)
```

# Index

are\_disjoint\_sets  
    (assert\_are\_disjoint\_sets), 2  
are\_intersecting\_sets  
    (assert\_are\_disjoint\_sets), 2  
are\_set\_equal  
    (assert\_are\_disjoint\_sets), 2  
assert\_are\_disjoint\_sets, 2  
assert\_are\_intersecting\_sets  
    (assert\_are\_disjoint\_sets), 2  
assert\_are\_set\_equal  
    (assert\_are\_disjoint\_sets), 2  
assert\_is\_set\_equal  
    (assert\_are\_disjoint\_sets), 2  
assert\_is\_subset  
    (assert\_are\_disjoint\_sets), 2  
assert\_is\_superset  
    (assert\_are\_disjoint\_sets), 2  
  
is\_set\_equal  
    (assert\_are\_disjoint\_sets), 2  
is\_subset (assert\_are\_disjoint\_sets), 2  
is\_superset (assert\_are\_disjoint\_sets),  
    2  
  
set\_is\_equal, 3  
sets, 3