Package 'ordering'

May 9, 2026

Type Package

Title Test, Check, Verify, Investigate the Monotonic Properties of Vectors

Version 0.7.0

Date 2018-11-09

Maintainer Christopher Brown <chris.brown@decisionpatterns.com>

Description Functions to test/check/verify/investigate the ordering of vectors. The 'is_[strictly_]*' family of functions test vectors for 'sorted', 'monotonic', 'increasing', 'decreasing' order; 'is_constant' and 'is_incremental' test for the degree of ordering. `ordering` provides a numeric indication of ordering -2 (strictly decreasing) to 2 (strictly increasing).

Suggests testthat, na.tools(>= 0.3.0)

License GPL (>= 2)

URL https://github.com/decisionpatterns/ordering

BugReports https://github.com/decisionpatterns/ordering/issues

RoxygenNote 6.1.1

Repository CRAN

Encoding UTF-8

NeedsCompilation no

Author Christopher Brown [aut, cre], Decision Patterns [cph]

Date/Publication 2018-11-19 18:00:20 UTC

Contents

is_constant 2
ordering 4

Index **6**

is_constant *ordering Tests*

Description

Tests vectors for (strictly) increasing, decreasing, monotonic and constant properties

Usage

```
is_constant(x, na.rm = TRUE)

is_increasing(x, na.rm = TRUE)

is_strictly_increasing(x, na.rm = na.omit)

is_decreasing(x, na.rm = na.omit)

is_strictly_decreasing(x, na.rm = na.omit)

is_incremental(x, step = 1, na.rm = TRUE)

is_uniform(x, step = NULL, na.rm = TRUE)

is_monotonic(x, na.rm = TRUE)

is_strictly_monotonic(x, na.rm = TRUE)

is_sorted(x, na.rm = TRUE)

is_strictly_sorted(x, na.rm = TRUE)

is_unsorted(...)

is_strictly_unsorted(...)
```

Arguments

x	vector
na.rm	function or NULL; action to perform on input to handle the missing values
step	integer; step size for `is_incremental`. (Default: 1)‘
...	used for passing default arguments

Details

Tests to various monotone properties of vectors.

`is_incremental` determines if x is incremental, i.e. monotonic and equally spaced.

`is_uniform` is a wrapper around `is_incremental` with `step=1`

`is_[strictly_]monotonic` determine the sort properties of `x`.

This differes from `base::is.unsorted()` which should more properly be called `is.increasing` since `base::is.unsorted(3:1)` == `TRUE`; `3:1` is obviously sorted.

`is_sorted()` is a alias for `is_monotonic()` and `is_strictly_sorted()` is an alias for `is_strictly_monotonic()`.

Value

logical or NA. (NB: NA is returned because it is a logical vector and this is needed to put these results cleanly in tables.)

logical

Note

The behavior of this package is The functions `base::is.unsorted()` is perhaps misnamed and should properly be names `is_not_increasing` since `base::is.unsorted(3:1)` == `TRUE` after all vector 3,2,1 is sorted but not increasing.

See Also

- `base::diff()`
- `base::is.unsorted()`

Examples

```
is_constant( rep(3,5) )

is_increasing( 1:5 )                   # TRUE
is_increasing( c(1,2,1,3) )            # FALSE

is_increasing( c(1,NA,2,3) )           # NA
is_increasing( c(1,NA,2,3), na.omit )  # TRUE

is_monotonic( 1:5 )                    # TRUE
is_monotonic( -5:5 )                   # TRUE
is_monotonic( 5:-5 )                   # TRUE
is_monotonic( c(1,5,3))                # FALSE


is_incremental(1:5 )
is_incremental( c(1,2,5))

is_incremental(1:5, step=NULL)
is_uniform(1:5)


is_monotonic( 1:3 )
is_strictly_monotonic(1:3)
```

```
is_monotonic( c(1,3,2) )
is_strictly_monotonic( c(1,3,2) )

is_sorted(1:3)
is_sorted(c(1,3,2))

lets <- letters[1:3]
is_monotonic( lets )
is_monotonic( c('a', 'c', 'b') )

is_sorted(1:10)

is_sorted(-5:5)
is_sorted(5:-5)

is_sorted( letters )
is_sorted( rev(letters) )
```

`ordering` *ordering*

Description

Determining the ordering (montonicity) of a vector

Usage

```
ordering(x, na.rm = TRUE)

monotonicity(x, na.rm = TRUE)
```

Arguments

`x`	numeric vector
`na.rm`	logical; whether to omit `NA` values. (Default: TRUE)

Details

`monotonicty` determines the monotonicy (first derivative) of a numeric vector as one of:

- stictly increasing,
- increasing / montonically increasing / non-decreasing,
- decreasing / monotonically decreasing / non-increasing,
- strictly decreasing, or
- constant

ordering tests, e.g. is_increasing are more efficient at testing.

`monotonicity()` is an alias for `ordering`.

Value

integer;

- **2** : stictly increasing
- **1** : increasing / montonically increasing / non-decreasing
- **0** : non-ordered or constant
- **-1**: decreasing / monotonically decreasing / non-increasing
- **-2**: strictly decreasing
- NA:contains onlyNAs all na.rmdid not resolve allNA‘s

References

http://en.wikipedia.org/wiki/Monotonic_function http://stackoverflow.com/questions/13093912/how-to-check-if-a-sequence-of-numbers-is-monotonically-increasing-or-decreasing

Examples

```
ordering( 1:3 )      # 2
ordering( c(1,1,3) ) # 1
ordering( c(1,0,1) ) # 0  "No ordering, does not apply constant"
ordering( c(3,1,1) ) # -1
ordering( 3:1 )      # -2
```

Index

base::diff(), *3*
base::is.unsorted(), *3*

is_constant, 2
is_decreasing *(*is_constant*)*, 2
is_increasing, *4*
is_increasing *(*is_constant*)*, 2
is_incremental *(*is_constant*)*, 2
is_monotonic *(*is_constant*)*, 2
is_monotonic(), *3*
is_sorted *(*is_constant*)*, 2
is_sorted(), *3*
is_strictly_decreasing *(*is_constant*)*, 2
is_strictly_increasing *(*is_constant*)*, 2
is_strictly_monotonic *(*is_constant*)*, 2
is_strictly_monotonic(), *3*
is_strictly_sorted *(*is_constant*)*, 2
is_strictly_sorted(), *3*
is_strictly_unsorted *(*is_constant*)*, 2
is_uniform *(*is_constant*)*, 2
is_unsorted *(*is_constant*)*, 2

monotonicity *(*ordering*)*, 4

ordering, 4