Package: essentials (via r-universe)

October 10, 2024

Type Package

Title Essential Functions not Included in Base R

Version 0.1.0

Author Andrew Simmons

Maintainer Andrew Simmons <akwsimmo@gmail.com>

Description Functions for converting objects to scalars (vectors of length 1) and a more inclusive definition of data that can be interpreted as numbers (numeric and complex alike).

License MIT + file LICENSE

Depends methods

Encoding UTF-8

LazyData true

NeedsCompilation yes

Repository CRAN

Date/Publication 2021-01-29 08:50:07 UTC

Contents

essentials-package *Essential Functions not Included in Base R*

Description

Functions for converting objects to scalars (vectors of length 1) and a more inclusive definition of data that can be interpreted as numbers (numeric and complex alike).

Details

The four most important functions from this package are `as.numbers`, `is.numbers`, `as.scalar` and `aslength1`.

`as.numbers` coerces its argument to type double or complex. `is.numbers` tests if its argument is interpretable as numbers.

`as.scalar` coerces its argument to an scalar (an atomic vector of length 1). It strips attributes including names.

`aslength1` coerces its argument to a vector of length 1 (not necessarily atomic). It strips attributes from arguments that are not vectors, but preserves names for arguments that are vectors.

Author(s)

Andrew Simmons

Maintainer: Andrew Simmons <akwsimmo@gmail.com>

Examples

```
as.numbers("4")
as.numbers("4+0i")  # imaginary component is removed
as.numbers("4+1i")

is.numbers(4L)
is.numbers(4)
is.numbers(4+1i)

as.scalar(1:100)
as.scalar(as.list(1:100))  # coerced to NA_character_ since argument isn't atomic

aslength1(1:100)  # identical to as.scalar(1:100)
aslength1(as.list(1:100))  # returns a list of length 1
```

`as.scalar` *Scalars*

Description

Coerce objects to scalars (vectors of length 1).

Usage

```
as.scalar(x)

as.scalar.logical(x)

as.scalar.integer(x)

as.scalar.real(x)
as.scalar.double(x)
as.scalar.numeric(x)

as.scalar.complex(x)

as.scalar.number(x, strict = TRUE)

as.scalar.string(x)
as.scalar.character(x)
```

Arguments

`x`	object to be coerced.
`strict`	`TRUE` or `FALSE`, should a complex number that is strictly real (real component is NA or NaN or imaginary component is NA or NaN or 0) be converted to a real number?

Details

`as.scalar.logical` coerces an object to a vector of type "logical" of length 1.

`as.scalar.integer` coerces an object to a vector of type "integer" of length 1.

`as.scalar.real`, `as.scalar.double` and `as.scalar.numeric` coerces an object to a vector of type "numeric" of length 1.

`as.scalar.complex` coerces an object to a vector of type "complex" of length 1.

`as.scalar.number` coerces an object to a vector of type "numeric" or "complex" of length 1.

`as.scalar.string` and `as.scalar.character` coerces an object to a vector of type "character" of length 1.

`as.scalar` coerces an object to a vector of length 1.

Value

a vector of length 1

Examples

```
## if the type converting from and converting to are identical, as.scalar is a
## much shorter way of writing what you intend.
as.scalar(c(TRUE, FALSE, NA))
as.scalar(1:100)
as.scalar(1:10 + 0.5)
as.scalar(exp((0+1i) * 6 * (-4:4)))
as.scalar(letters)


## if the type converting from and converting to are not identical, it is better
## to specify the type converting to.
as.scalar.logical(c(TRUE, FALSE, NA))
as.scalar.integer(c(TRUE, FALSE, NA))
as.scalar.numeric(c(TRUE, FALSE, NA))
as.scalar.complex(c(TRUE, FALSE, NA))
as.scalar.character(c(TRUE, FALSE, NA))
```

---

aslength1 *Subset the First Element of a Vector*

Description

Subset the first element of a vector.

Usage

```
aslength1(x)
```

Arguments

x	vector (or an object which can be coerced) with at least one element.

Details

Vectors of length one return themselves. Vectors of length greater than one return the first element with a warning. Vectors of length zero raise an error. If x is a vector (determined by `is.vector`), names will be preserved.

Value

A vector of length 1.

Examples

```
aslength1(1)
aslength1(1:10)
try(aslength1(integer(0)))
```

---

hypot *Hypotenuse*

Description

Compute the length of the hypotenuse.

Usage

```
hypot(x, y)
```

Arguments

x, y	numeric vectors; the lengths of non-hypotenuse sides, the sides adjacent to the right angle.

Details

The hypotenuse is the longest side of a right-angled triangle, the side opposite the right angle. The length of the hypotenuse is defined as:

$$\sqrt{(x^2 + y^2)}$$

If x[i] or y[i] is infinite, the result in the i-th position will always be Inf. Otherwise, if x[i] or y[i] is NA or NaN, the result in the i-th position will be NaN. Otherwise, if the absolute value of x[i] is considerably larger than the absolute value of y[i], the result in the i-th position will be the absolute value of x[i] (and vice versa). Otherwise, the value will be calculated using the above definition.

Value

A numeric vector. If x or y is a zero-length vector the result has length zero. Otherwise, the result has length of the maximum of the lengths of x and y.

Examples

```
hypot(Inf, NaN)     # still positive infinity
hypot(NaN, 0)       # NaN
hypot(NA_real_, 0)  # NaN


## numbers whose squares would overflow normally are handled well
hypot(.Machine$double.xmax, 5     )
```

```
hypot(1e+300              , 1e+300)


hypot(3, 4)         # 5
```

`numbers` *Number Vectors*

Description

Creates or coerces objects of type "numeric" or "complex". `is.numbers` is a more general test of an object being interpretable as numbers.

Usage

```
numbers(length = 0)
as.numbers(x, strict = TRUE, ...)
is.numbers(x)
```

Arguments

`length`	A non-negative integer specifying the desired length. Double values will be coerced to integer: supplying an argument of length other than one is an error.
`x`	object to be coerced or tested.
`strict`	`TRUE` or `FALSE`, should a vector of complex numbers where each number is strictly real (real component is NA or NaN or imaginary component is NA or NaN or 0) be converted to a vector of real numbers?
`...`	further arguments passed to or from other methods. `as.numbers` is not a generic function, but it can be made into one if desired. See `setGeneric`.

Details

`numbers` is identical to `numeric` and `double` (and `real`). It creates a double-precision vector of the specified length with each element equal to 0.

`as.numbers` attempts to coerce its argument to be of double or complex type: like `as.vector` it strips attributes including names.

`is.numbers` is a more general test of an object being considered numbers, meaning the base type of the class is `double` or `integer` or `complex` *and* values can reasonably be regarded as numbers (e.g., arithmetic on them makes sense, and comparison should be done via the base type).

Value

for `numbers` see `double`.

`as.numbers` returns either a `double` or `complex` vector.

`is.numbers(x)` is defined as `is.numeric(x) || is.complex(x)`.

Examples

```
x <- 1:5
names(x) <- c("a", "b", "c", "d", "e")
as.numbers(x)  # vector converted from integer to double, names removed


x <- x + 0i  # x is now a complex vector
as.numbers(x)  # vector of type double since all numbers were purely real


## vector of type complex, despite being purely real
as.numbers(x, strict = FALSE)


x <- x + 1i
## vector remains of type complex since numbers are not purely real
as.numbers(x)
```

---

numbers-class *Class* "numbers"

Description

An umbrella formal class encompassing all objects interpretable as numbers. This includes `integer`, `double` and `complex`.

Extends

This class extends both "numeric" and "complex", directly.

Methods

coerce A method is defined to coerce an arbitrary object to a numbers vector by calling `as.numbers`. The object is returned as is if it already extends class "numeric" or "complex".

Index