Package ‘na.tools’

June 25, 2018

Title  Comprehensive Library for Working with Missing (NA) Values in Vectors

Version  0.3.1

Date  2018-06-25

Description  This comprehensive toolkit provide a consistent and extensible framework for working with missing values in vectors. The companion package ‘tidyimpute’ provides similar functionality for list-like and table-like structures. Functions exist for detection, removal, replacement, imputation, recollection, etc. of ‘NAs’.

URL  https://github.com/decisionpatterns/na.tools

BugReports  https://github.com/decisionpatterns/na.tools/issues

Depends  R (>= 3.1.0)

Imports  stats, methods

Suggests  testthat (>= 1.0.2)

License  GPL-3 | file LICENSE

LazyData  true

RoxygenNote  6.0.1.9000

Repository  CRAN

Encoding  UTF-8

NeedsCompilation  no

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

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

Date/Publication  2018-06-25 20:02:57 UTC
all_na Tests for missing values

Description
Test if all values are missing

Usage
all_na(x)

## Default S3 method:
all_na(x)

any_na(x)

is_na()

which_na(x)

Arguments
x object to test.

Details
These are S3 Generics that provide default methods.
all_na reports if all values are missing.
any_na reports if any values are missing. If always returns a logical scalar.
is_na is a wrapper around base::is.na() created to keep stylistic consistent with the other functions.
which_na is implemented as which(is.na(x)). It is a S3 generic function.

Value

logical scalar indicating if values are missing.

logical scalar; either TRUE or FALSE.

integer of indexes of x that correspond to elements of x that are missing (NA). Names of the result are set to the names of x.

See Also

- `base::anyNA()`
- `base::is.na()` - for the variant returning logical

Examples

```r
all_na(c(NA, NA, 1))  # FALSE
all_na(c(NA, NA, NA))  # TRUE

df <- data.frame(char = rep(NA_character_, 3), nums=1:3)
all_na(df)  # FALSE

df <- data.frame(char = rep(NA_character_, 3), nums=rep(NA_real_,3))
all_na(df)  # TRUE

any_na(1:10)  # FALSE
any_na(c(1, NA, 3))  # TRUE

x <- c(1, NA, NA, 4:6)
which_na(x)

names(x) <- letters[1:6]
which_na(x)
```

Description

Coerce values in a safe, non-destructive and consistent way.

Usage

```r
coerce_safe(object, class, alert = stop, ..., alert_irreversible = alert, alert_na = alert)
```
Arguments

object to be coerced
class character; class to which object should be coerced.
alert function to use to raise exceptions: (Default: base::stop())
... unused
alert_irreversible function to raise alert when coercion is not reversible. See Details.
alert_na function to raise when NAs are produced.

coerce_safe transform the object to class in a safe, consistent, non-destructive way.
Safe means that coercion:
1. is non-destructive (i.e information is not lost in the transformation)
2. is reversible:
   \[ f^{-1}(f(x)) = x \]
3. does not introduce (additional) missing values (NA)

By default, coerce_safe raises an alert (message|warning|error) when the attempted coercion violates these constraints. The alert argument (and alert_irreversible or alert_na) can be used to flexible change the response. Valid values for these are base::message(), base::warning() and base::stop among others.

Value

object coerced to class but ensured that there has been no loss in data and no additional Missing values introduced.

Note

There must be a as method to the reverse coercion for this function to work.

See Also

methods::as 'coercion::try_as()"

Examples

## Not run:
# Error
coerce_safe(1.01, "integer") # 1.01 != 1
coerce_safe( c("1","2","a"), "integer" )

## End(Not run)
Description

Imputation by Cummutative Functions

Impute using replacement values calculated from a univariate, commutative function.

\texttt{na.median} imputes with the median value of \( x \). The median is only valid for numeric or logical values.

Usage

\texttt{na.max(.x, ...)}
\texttt{na.min(.x, ...)}
\texttt{na.mean(.x, ...)}
\texttt{na.median(.x, ...)}
\texttt{na.quantile(.x, ...)}
\texttt{na.mode(.x, ...)}
\texttt{na.most_freq(.x, ...)}

Arguments

\verb|.x| vector in which NA values are to be replaced. The ordering of \( x \) does not matter.
\verb|...| additional arguments passed to lower-level summary functions.

Details

This collection of functions calculates a replacement value using an univariate function where the order of values in \( x \) do not matter, i.e. commutative.

\texttt{na.max} and \texttt{na.min} replace missing values (NA) with the maximum or minimum of non-missing values \( x \). (Internally: \texttt{base::max(..., na.rm=TRUE)} and \texttt{base::min(..., na.rm=TRUE)}. ... has no affect.

\texttt{na.mean} replaces NA values with the mean of \( x \). Internally, \texttt{mean(x, na.rm=TRUE, ...)} is used. If mean cannot be calculated (e.g. \( x \) isn’t numeric) then \( x \) is returned with a warning.

\texttt{na.quantile} imputes with a quantile. The quantile is specified by a \texttt{probs} argument that is passed to \texttt{stats::quantile()}. If \texttt{probs} can be a scalar value in which all values are replaced by that quantile or a vector of length(\( x \)) values which replaces the missing values of \( x \) with the \texttt{probs}. The ability to provide a vector may be deprecated in the future.
na.mode replaces all NA with the most frequently occurring value. In the event of ties, the value encountered first in .x is used.

na.most.freq is an alias for na.mode.

Value
A vector of class(x) and length(x) in which missing values (NA) have been replaced the result of a function call:

\[ \text{fun}(x, ...) \]

See Also
- `na.replace()` - used internally by these functions
- `na.constant()`
- `base::max()` and `base::min()`
- `median()`
- `quantile()`

Examples
na.median( c(1,2,NA_real_,3) )

na.quantile( c(1,2,NA_real_,3), prob=0.4 )

na.mode( c(1,1,NA,4) )
na.mode( c(1,1,4,NA) )

---

### impute-constant

**Impute by Constant Value Replaces NAs by a constant**

**Description**
Impute by Constant Value

Replaces NAs by a constant

**Usage**

na.constant(.x, .na)

na.inf(.x)

na.neginf(.x)

na.true(.x)
Arguments

\texttt{x} \quad \text{vector; of values to have the NA}

\texttt{na} \quad \text{scalar to use as replacement.}

Details

These functions replace \textbf{ALL} NA values in \texttt{x} with an scalar value specified by \texttt{na}.

\texttt{na.constant} replaces missing values with a scalar constant. It is a wrapper around \texttt{na.replace()} but permits \texttt{na} to only be a scalar.

\texttt{na.inf} and \texttt{na.neginf} replace all missing values with Inf and \texttt{-Inf} repectively. ‘.

\texttt{na.true} and \texttt{na.false} replace missing values with \texttt{TRUE} and \texttt{FALSE} respectively.

\texttt{na.zero} replaces missing values with 0 which gets coerced to the \texttt{class(x)} as needed.

Value

A vector with the type and length of \texttt{x} with all missing values replaces by \texttt{.na}.

See Also

- \texttt{na.replace()} the underlying function that performs the replacement.

Examples

\begin{verbatim}
na.constant( c(1,NA,2), -1 )

na.inf( c( 1, 2, NA, 4 ) )
na.neginf( c( 1, 2, NA, 4 ) )

na.true( c(TRUE, NA.logical, FALSE) )  # T T F
na.false( c(TRUE, NA.logical, FALSE) )  # T F F

na.zero( c(1,NA,3) )  # 1 0 3
\end{verbatim}
impute-noncommutative
non-commutative imputation
Impute missing values using non-commutative functions, i.e. where the order matters.

Description

non-commutative imputation

Impute missing values using non-commutative functions, i.e. where the order matters.

Usage

na.cummax(.x, ...)
na.cummin(.x, ...)
na.cumsum(.x, ...)
na.cumprod(.x, ...)

Arguments

.x atomic-vector with 0 or more missing values
... additional arguments

Details

Non-commutative imputations functions assume that .x is in the proper order since the values depend on order. Usually, this is relevant then .x is part of a table.

These functions replaces NA values with the cumulative max of .x. Internally, fun(.x, na.rm=TRUE, ... ) is used. If the function cannot be calculated (e.g. .x isn’t numeric) then x is returned unchanged with a warning.

Use of na.cumsum and na.cumprod are dangerous since they omit missing values that may contribute to

See Also

• base::cummax()
• impute-commutative
na.bootstrap

Description

Replace missing values with value randomly drawn from x

Usage

na.bootstrap(x, ...)  
na.resample(x, ...)  

Arguments

.x vector with 
... additional arguments passed to base::sample()

Details

na.random replaces missing values by sampling the non-missing values. By default sampling occurs with replacement since more valuables may be needed than are available. This function is based on base::sample().

The default is to replace by sampling a population defined by the non-missing values of .x with replacement

na.random is an alias for na.bootstrap.

Note

na.bootstrap is non-deterministic. Use base::set.seed() to make it deterministic

See Also

- base::sample()

Examples

x <- c(1,NA,3)  
na.bootstrap(x)
na.replace  Replace Missing Values

Description
Replaces NA values with explicit values.

Usage
na.replace(x, .na, ...)
na.explicit(x)

Arguments
x vector in which NA values are to be replaced.
.Na scalar, length(x)-vector or function used to replace NA. See #Details.
... additional arguments passed to .na when it is a function.

Details
na.replace replaces missing values in x by .na if possible.
In R, replacement of values can cause a change in the class/type of an object. This is not often desired. na.replace is class/type-safe and length-safe. It replaces missing values without changing the x's class or length regardless of the value provided by .na.

Param: x
If x is categorical (e.g. character or factor), .na is optional. The default is "(NA)" and can be set with options( NA_explicit_ = new_value ). It can also be referenced directly with NA_explicit_.
If x is a factor, unique values of .na not in already present in levels(x) will be added. They are appended silently unless getOption('verbose') == TRUE in which a message reports the added levels.

Param: .na
.Na can be either a scalar, vector or function.
If a scalar, each missing value of x is replaced by .na.
If a vector, .na must have length(x). Missing values of x are replaced by corresponding elements of .na. Recycling is not allowed.
If a function, x is transformed by .na' with:
  .na(x, ...)
then preceding with normal operations.

na.explicit is an alias for na.replace that uses NA_explicit_ for '.na"; it returns x unchanged if it cannot change the value.
**Description**

Removes NA values from objects

**Usage**

`na.rm(object, ...)`
**Arguments**

- `object` to remove NAs from
- `...` further arguments special methods could require.

**Details**

For vectors this is the same as `stats::na.omit()` or `stats::na.exclude()`. It will also work on recursive objects.

This is predominantly maintained for syntactic convenience since a number of functions have na.omir.

**Value**

An object of the same class with all NA values removed. For data.frame and data.table objects entire columns are removed if they contain solely NA values.

**See Also**

- `stats::na.omit()`, `stats::na.exclude()`
- `all_na()`

---

**Description**

Change values to NAs, ie make explicit NAs back to NA.

**Usage**

```r
da.unreplace(x, values)
```

```r
## Default S3 method:
da.unreplace(x, values = NULL)
```

```r
## S3 method for class 'character'
da.unreplace(x, values = c("NA", NA_explicit_))
```

```r
## S3 method for class 'factor'
da.unreplace(x, values = c("NA", NA_explicit_))
```

`na.implicit(x, values)`

**Arguments**

- `x` object
- `values` values that are (or can be coerced to) `class(x)` that are to be set to NA.
**Details**

`na.unreplace` replaces values by `NA`. It is meant to be nearly inverse operation to `na.replace` (and `na.explicit`). It can be used on both atomic and recursive objects. Unlike `na.replace` however, values express the values that if matched are set to `NA`. It is basically:

```r
x[x
```

`na.unreplace` is a S3 method that can be used to define additional methods for other objects.

**See Also**

- `na.replace()`

**Examples**

```r
na.unreplace( c(1L,2,3,4), 3 )
na.unreplace( c("A", "(NA)", "B", "C") )
na.unreplace( c("A", NA_explicit_, "B", "C") )

df <- data.frame( char=c('A', 'NA', 'C', NA_explicit_), num=1:4 )
na.unreplace(df)
```

---

**Description**

Default replacement for missing values in categorical vectors.

**Usage**

`NA_explicit_`

**Format**

An object of class character of length 1.

**Details**

`NA_explicit_` is used as a default replacement for categorical vectors. It is an active binding to `getOptions('NA_explicit_')` and is exported to the caller's namespace. To change the value of `NA_explicit` use:

```r
options( NA_explicit = new_value )
```

`NA_explicit_` cannot be directly set.
See Also

na.replace()

---

| NA_logical | NA_logical |

Description

NA_logical

Usage

NA_logical

Format

An object of class logical of length 1.

Details

This simply creates a NA_logical variable. This is the same as NA

---

| n_na | Counts how many values are NA |

Description

Returns the number of values that are NA

Usage

n_na(x)

na.howmany(x)

na.n(x)

pct_na(x)

na.pct(x)

Arguments

x object to count how many values are NA
**Details**

*n_na* counts the number of missing values. *na.n* is an alias in the dplyr style.

*pct_na* gives the percentage of values that are *NA*.

**Value**

*n_na* returns an integer. *pct_na* returns a numeric value 0-1.

**Examples**

```r
x <- c( 1, NA, NA, 4:5 )
n_na(x)
pct_na(x)
```
Index

*Topic datasets
   NA_explicit_, 13
   NA_logical, 14

all_na, 2
all_na(), 12
any_na(all_na), 2

base::anyNA(), 3
base::cummax(), 8
base::ifelse(), 11
base::is.na(), 2, 3
base::max(), 6
base::message(), 4
base::min(), 6
base::replace(), 11
base::sample(), 9
base::set.seed(), 9
base::stop, 4
base::stop(), 4
base::warning(), 4

c coerce_safe, 3

impute-commutative, 5, 8
impute-constant, 6
impute-noncommutative, 8
is.na(all_na), 2

median(), 6
methods::as, 4

n_na, 14
na.bootstrap, 9
na.constant (impute-constant), 6
na.constant(), 6
na.cummax (impute-noncommutative), 8
na.cummin (impute-noncommutative), 8
na.cumprod (impute-noncommutative), 8
na.cumsum (impute-noncommutative), 8
na.explicit (na.replace), 10

na.false (impute-constant), 6
na.howmany (n_na), 14
na.implicit (na.unreplace), 12
na.inf (impute-constant), 6
na.max (impute-commutative), 5
na.mean (impute-commutative), 5
na.median (impute-commutative), 5
na.min (impute-commutative), 5
na.mode (impute-commutative), 5
na.most_freq (impute-commutative), 5
n.n (n_na), 14
na.neginf (impute-constant), 6
na.pct (n_na), 14
na.quantile (impute-commutative), 5
na.replace, 10
na.replace(), 6, 7, 13, 14
na.resample (na.bootstrap), 9
na.rm, 11
na.true (impute-constant), 6
na.unreplace, 12
na.zero (impute-constant), 6
NA_explicit_, 10, 13
NA_logical, 14

pct_na (n_na), 14
quantile(), 6

stats::na.exclude(), 12
stats::na.omit(), 12
stats::quantile(), 5

which_na(all_na), 2