Package 'fauxnaif'

August 2, 2020

Title Convert Values to NA

Version 0.6.0

Description Provides a replacement for dplyr::na_if(). Allows you to specify multiple values to be replaced with NA using a single function.

License MIT + file LICENSE

URL `https://github.com/rossellhayes/fauxnaif`

BugReports `https://github.com/rossellhayes/fauxnaif/issues`

Depends R (>= 3.5)

Imports dplyr,
glue,
lifecycle,
magrittr,
rlang

Suggests covr,
intrval,
knitr,
rmarkdown,
roxygen2,
testthat,
tibble,
tidyr,
vctrs

VignetteBuilder knitr

RdMacros lifecycle

Encoding UTF-8

Language en-US

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.1.1

R topics documented:

`faux_census` *A small sample of a fabricated census-like dataset*

Description

A dataset containing fake demographic data, used in the `fauxnaif` vignette.

Usage

```
faux_census
```

Format

A tibble with 20 rows and 6 variables.

Source

Fabricated

Examples

```
faux_census
```

`na_if_in` *Convert values to NA*

Description

This is a replacement for `dplyr::na_if()`. It is useful if you want to convert annoying values to `NA`. Unlike `dplyr::na_if()`, this function allows you to specify multiple values to be replaced with `NA` at the same time.

- `na_if_in()` replaces values that match its arguments with `NA`.
- `na_if_not()` replaces values that *do not* match its arguments with `NA`.

Usage

```
na_if_in(input, ...)

na_if(input, ...)

na_if_not(input, ...)
```

Arguments

`input`	Vector to modify
`...`	Values to replace with `NA`, specified as either: • An object, vector of objects, or list of objects • A one-sided formula (see section "Formulas")

Value

A modified version of `input` with selected values replaced with `NA`.

Formulas

These functions accept one-sided formulas that can evaluate to logical vectors. The input is represented in these conditional statements as "`.`". Valid formulas take the form `~ . < 0`. Additional examples are included in section "Examples".

See Also

Scoped variants can be used in pipelines and modify multiple variables at once

`dplyr::na_if()` to replace a single value with `NA`.

`dplyr::coalesce()` to replace missing values with a specified value.

`tidyr::replace_na()` to replace `NA` with a value.

`dplyr::recode()` and `dplyr::case_when()` to more generally replace values.

Examples

```
-1:10
# We can replace -1...
# ... explicitly
na_if_in(-1:10, -1)
# ... by specifying values to keep
na_if_not(-1:10, 0:10)
# ... using a formula
na_if_in(-1:10, ~ . < 0)
# ... or using a function
na_if_in(-1:10, min)

messy_string <- c("abc", "", "def", "NA", "ghi", 42, "jkl", "NULL", "mno")
# We can replace unwanted values...
# ... one at a time
na_if_in(messy_string, "")
# ... or all at once
na_if_in(messy_string, "", "NA", "NULL", 1:100)
na_if_in(messy_string, c("", "NA", "NULL", 1:100))
na_if_in(messy_string, list("", "NA", "NULL", 1:100))
# ... or using a clever formula
grepl("[a-z]{3,}", messy_string)
na_if_not(messy_string, ~ grepl("[a-z]{3,}", .))

# na_if_in() is particularly useful inside dplyr::mutate
faux_census %>%
  dplyr::mutate(
    state = na_if_in(state, "Canada"),
    age   = na_if_in(age, ~ . < 18, ~ . > 120)
  )

# We get a message if our values to replace don't exist
na_if_in(-1:10, 11)
# And a warning if we use an invalid input...
# ... like a two-sided formula
na_if_in(-1:10, x ~ . < 0)
```

```
# ... NULL
na_if_in(-1:10, NULL)
# ... or nothing at all
na_if_in(-1:10)

# This function handles vector values differently than dplyr,
# and returns a different result with vector replacement values:
na_if_in(1:5, 5:1)
dplyr::na_if(1:5, 5:1)
```

`scoped_na_if` *Convert values to NA in multiple columns*

Description

Deprecated

Usage

```
na_if_all(.tbl, ...)

na_if_not_all(.tbl, ...)

na_if_at(.tbl, .vars, ...)

na_if_not_at(.tbl, .vars, ...)

na_if_if(.tbl, .predicate, ...)

na_if_not_if(.tbl, .predicate, ...)
```

Arguments

`.tbl`	A `tbl` object
`...`	Values to replace with `NA`, specified as either: • An object, vector of objects, or list of objects • A one-sided formula (see section "Formulas" in `na_if()`)
`.vars`	A list of columns generated by `dplyr::vars()`, a character vector of column names, a numeric vector of column positions, or NULL.
`.predicate`	A predicate function to be applied to the columns or a logical vector. The variables for which .predicate is or returns TRUE are selected. This argument is passed to `rlang::as_function()` and thus supports quosure-style lambda functions and strings representing function names.

Details

The dplyr::scoped variants of `na_if()` and `na_if_not()` can be used directly within pipelines and can modify multiple variables at once.

- *_all() affects every variable
- *_at() affects variables selected with a character vector or `dplyr::vars()`
- *_if() affects variables selected with a predicate function

Value

A modified data frame. Matched values in selected columns are replaced with `NA`.

See Also

`na_if_in()` and `na_if_not()` operate directly on vectors

`dplyr::mutate_all()`, `dplyr::mutate_at()` and `dplyr::mutate_if()` can apply any function to variables selected in the same way

Examples

```
## Not run:
df <- data.frame(a = 0:5, b = 5:0, c = as.numeric(0:5), d = letters[1:6])

na_if_all(df, 0)
na_if_not_all(df, 0:3, "c")

na_if_at(df, c("a", "c"), 0)
na_if_not_at(df, c("a", "c"), 0:3)

na_if_if(df, is.integer, 0)
na_if_not_if(df, is.integer, 0:3)

## End(Not run)
```

Index