Package 'incase'

September 15, 2020

Type Package

Title Pipe-Friendly Vector Replacement with Case Statements

Version 0.1.0

Description Offers a pipe-friendly alternative to the 'dplyr' functions case_when() and if_else(). These functions accept a vector as an optional first argument, allowing conditional statements to be built using the 'magrittr' dot operator. The functions also coerce all possible outputs to the same type, meaning you no longer have to worry about using specific typed variants of NA or explicitly declaring integer outputs.

License MIT + file LICENSE

URL https://incase.rossellhayes.com, https://github.com/rossellhayes/incase

BugReports https://github.com/rossellhayes/incase/issues

Imports magrittr, plu, rlang

Suggests cli, covr, crayon, dplyr, testthat

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

NeedsCompilation no

Author Alexander Rossell Hayes [aut, cre, cph] (<https://orcid.org/0000-0001-9412-0457>)

Maintainer Alexander Rossell Hayes <alexander@rossellhayes.com>

Repository CRAN

Date/Publication 2020-09-15 10:00:06 UTC

R topics documented:

`if_case` *Pipe-friendly vectorized if*

Description

Compared to `dplyr::if_else()`, this function is easier to use with a pipe. A vector piped into this function will be quietly ignored. This allows magrittr dots to be used in arguments without requiring workarounds like wrapping the function in braces.

Usage

```
if_case(condition, true, false, missing = NA, ...)
```

Arguments

`condition`	Logical vector
`true, false, missing`	Values to use for `TRUE`, `FALSE`, and `NA` values of `condition`. They must be either the same length as `condition`, or length 1.
`...`	Values passed to `...` produce an error. This facilitates the quiet ignoring of a piped vector.

Details

This function is also less strict than `dplyr::if_else()`. If `true`, `false`, and `missing` are different types, they are silently coerced to a common type.

Value

Where `condition` is `TRUE`, the matching value from `true`; where it's `FALSE`, the matching value from `false`; and where it's `NA`, the matching value from `missing`.

See Also

`in_case()`, a pipeable alternative to `dplyr::case_when()`

`switch_case()`, a reimplementation of `switch()`

`dplyr::if_else()`, from which this function is derived

Examples

```
x <- c(1, 2, 5, NA)

# if_case() produces the same output as dplyr::if_else()
if_case(x > 3, "high", "low", "missing")
dplyr::if_else(x > 3, "high", "low", "missing")
```

```
# if_case() does not throw an error if arguments are not of the same type
if_case(x > 3, "high", "low", NA)
try(dplyr::if_else(x > 3, "high", "low", NA))

# if_case() can accept a piped input without an error or requiring braces
x %>% if_case(. > 3, "high", "low", "missing")
try(x %>% dplyr::if_else(. > 3, "high", "low", "missing"))
x %>% {dplyr::if_else(. > 3, "high", "low", "missing")}

# You can also pipe a conditional test like dplyr::if_else()
{x > 3} %>% if_case("high", "low", "missing")
{x > 3} %>% dplyr::if_else("high", "low", "missing")
```

`in_case` *A pipe-friendly general vectorized if*

Description

This function allows you to vectorize multiple if_else() statements. If no cases match, NA is returned. This function derived from `dplyr::case_when()`. Unlike `dplyr::case_when()`, `in_case()` supports piping elegantly and attempts to handle inconsistent types (see examples).

Usage

```
in_case(..., preserve = FALSE, default = NA)
```

Arguments

`...`	<dynamic-dots> A sequence of two-sided formulas. The left hand side (LHS) determines which values match this case. The right hand side (RHS) provides the replacement value. The LHS must evaluate to a logical vector. Both LHS and RHS may have the same length of either 1 or `n`. The value of `n` must be consistent across all cases. The case of `n == 0` is treated as a variant of `n != 1`. `NULL` inputs are ignored.
`preserve`	If `TRUE`, unmatched elements of the input will be returned unmodified. (The elements may have their type coerced to be compatible with replacement values.) If `FALSE`, unmatched elements of the input will be replaced with `default`. Defaults to `FALSE`.
`default`	If `preserve` is `FALSE`, a value to replace unmatched elements of the input. Defaults to `NA`.

Value

A vector of length 1 or n, matching the length of the logical input or output vectors. Inconsistent lengths will generate an error.

See Also

`if_case()`, a pipeable alternative to `dplyr::if_else()`

`switch_case()`, a reimplementation of `switch()`

`dplyr::case_when()`, from which this function is derived

Examples

```
# Non-piped statements are handled the same as dplyr::case_when()
x <- 1:30
in_case(
  x %% 15 == 0 ~ "fizz buzz",
  x %%  3 == 0 ~ "fizz",
  x %%  5 == 0 ~ "buzz",
  TRUE         ~ x
)

# A vector can be directly piped into in_case() without error
1:30 %>%
  in_case(
    . %% 15 == 0 ~ "fizz buzz",
    . %%  3 == 0 ~ "fizz",
    . %%  5 == 0 ~ "buzz",
    TRUE         ~ .
  )

# in_case() silently converts types
1:30 %>%
  in_case(
    . %% 15 == 0 ~ 35,
    . %%  3 == 0 ~ 5,
    . %%  5 == 0 ~ 7,
    TRUE         ~ NA
  )

x <- 1:30
try(
  dplyr::case_when(
    x %% 15 == 0 ~ 35,
    x %%  3 == 0 ~ 5,
    x %%  5 == 0 ~ 7,
    TRUE         ~ NA
  )
)

# default and preserve make it easier to handle unmatched values
1:30 %>%
  in_case(
    . %% 15 == 0 ~ "fizz buzz",
    . %%  3 == 0 ~ "fizz",
    . %%  5 == 0 ~ "buzz",
    default      = "pass"
```

```
  )

1:30 %>%
  in_case(
    . %% 15 == 0 ~ "fizz buzz",
    . %%  3 == 0 ~ "fizz",
    . %%  5 == 0 ~ "buzz",
    preserve     = TRUE
  )
```

`switch_case` *Switch-style recoding of values*

Description

Switch-style recoding of values

Usage

```
switch_case(x, ..., preserve = FALSE, default = NA)
```

Arguments

`x`	A vector
`...`	<dynamic-dots> A sequence of two-sided formulas. Elements of `x` that match the left hand side (LHS) will be replaced with the value in the right hand side (RHS). The LHS must evaluate to an atomic vector. The RHS must be of length one. `NULL` inputs are ignored.
`preserve`	If `TRUE`, unmatched elements of `x` will be returned unmodified. (The elements may have their type coerced to be compatible with replacement values.) If `FALSE`, unmatched elements of `x` will be replaced with `default`. Defaults to `FALSE`.
`default`	If `preserve` is `FALSE`, a value to replace unmatched elements of `x`. Defaults to `NA`.

Value

A vector of the same length as `x`.

See Also

`in_case()`, a pipeable alternative to `dplyr::case_when()`

`if_case()`, a pipeable alternative to `dplyr::if_else()`

`switch()`, which inspired this function

Examples

```
parties <- sample(c("d", "r", "i", "g", "l"), 20, replace = TRUE)

switch_case(
  parties,
  "d" ~ "Democrat",
  "r" ~ "Republican",
  "i" ~ "Independent",
  "g" ~ "Green",
  "l" ~ "Libertarian"
)

parties %>%
  switch_case(
    "d" ~ "Democrat",
    "r" ~ "Republican",
    "i" ~ "Independent",
    "g" ~ "Green",
    "l" ~ "Libertarian"
  )

parties %>%
  switch_case(
    "d" ~ "Democrat",
    "r" ~ "Republican",
    c("i", "g", "l") ~ "Other"
  )

parties %>%
  switch_case(
    "d" ~ "Democrat",
    "r" ~ "Republican",
    default = "Other"
  )

parties %>%
  switch_case(
    "d" ~ "Democrat",
    "r" ~ "Republican",
    preserve = FALSE
  )

parties %>%
  switch_case(
    "d" ~ "Democrat",
    "r" ~ "Republican",
    preserve = TRUE
  )
```

Index