Package 'hablar'

March 19, 2020

Type Package

Title Non-Astonishing Results in R

Version 0.3.0

Author David Sjoberg

Maintainer David Sjoberg <dav.sjob@gmail.com>

Description Simple tools for converting columns to new data types. Intuitive functions for columns with missing values.

License MIT + file LICENSE

URL https://davidsjoberg.github.io/

BugReports https://github.com/davidsjoberg/hablar/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

Imports dplyr (>= 0.8.0), purrr, lubridate

Suggests testthat, knitr, rmarkdown, webshot, gapminder, DiagrammeR, rstudioapi

VignetteBuilder knitr

NeedsCompilation no

Repository CRAN

Date/Publication 2020-03-19 22:40:02 UTC

R topics documented:

`as_reliable` *Reliable conversion to another data type*

Description

Support functions for the `convert` function. These functions coerces vectors to a new data type, e.g. `as.numeric` except that it converts factors to character first. See `convert` for more information.

Usage

```
as_reliable_num(.x, ...)

as_reliable_int(.x, ...)

as_reliable_lgl(.x, ...)

as_reliable_dte(.x, origin = "1970-01-01", ...)

as_reliable_dtm(.x, origin = "1970-01-01", tz = "Europe/London", ...)

as_reliable_int(.x, ...)

as_reliable_lgl(.x, ...)

as_reliable_dte(.x, origin = "1970-01-01", ...)

as_reliable_dtm(.x, origin = "1970-01-01", tz = "Europe/London", ...)
```

Arguments

.x	vector
...	additional arguments
origin	argument to set origin for date/date time.
tz	argument to set time zone for date/date time. Default is Europe/London.

Value

vector

See Also

vignette("convert"), vignette("hablar")

Examples

```
x <- as.factor(c("1", "3.5"))
as_reliable_num(x)

x <- as.factor(c("9", "7"))
as_reliable_int(x)

x <- as.factor(c("1", "0"))
as_reliable_lgl(x)
```

---

check_df *Special checks*

Description

Returns TRUE if data frame have the specified special cases. For example, find_duplicates() returns TRUE if any rows are duplicates. If variables are passed to the function then TRUE or FALSE is returned for those variables.

Usage

```
check_duplicates(.data, ...)

check_na(.data, ...)

check_irrational(.data, ...)

check_nan(.data, ...)

check_inf(.data, ...)

check_complete_set(.data, ...)
```

Arguments

.data	a data frame
...	variables that should be considered. If empty, all variables are used.

Details

irrational values are Inf and NaN. 'check_complete_set' tests if all combinations of elements exists in the data frame.

Value

TRUE or FALSE

See Also

find_in_df to return rows instead of TRUE or FALSE. vignette("s"), vignette("hablar")

Examples

```
## Not run:
df <- data.frame(a = c("A", NA, "B", "C", "C"),
                 b = c(7, 8, 2, 3, 3),
                 c = c(NA, 1, NaN, 3, 2),
                 stringsAsFactors = FALSE)

# Returns FALSE because there is no duplicates
df %>% check_duplicates()

# Returns TRUE because there is duplicates in column a through b
df %>% check_duplicates(a:b)

# Returns FALSE because there is no NA column b
df %>% check_na(b)

# Returns TRUE because there is no NaN column c
df %>% check_nan(c)

## End(Not run)
```

---

convert *Convert data type of columns*

Description

Convert data type of columns

Usage

```
num(..., .args = list())

chr(..., .args = list())

lgl(..., .args = list())

int(..., .args = list())

dbl(..., .args = list())

fct(..., .args = list())

dtm(..., .args = list())

dte(..., .args = list())

convert(.x, ...)
```

Arguments

...	Scoping functions, see details
.args	extra argument to be passed to support function.
.x	A data.frame

Value

a tbl data frame

See Also

vignette("convert"), vignette("hablar")

Examples

```
## Not run:

# Change one column to numeric and another to character
mtcars %>%
  convert(num(gear),
          chr(mpg))


# Changing multiple data types on multiple columns
mtcars %>%
  convert(int(hp,
              wt),
          fct(qsec,
              cyl,
```

```
              drat))

# Also works with tidyselect convenience functions
mtcars %>%
  convert(int(vs:carb),
          fct(last_col()))


## End(Not run)
```

`could_this_be_that` *Tests is a vector could be of another data type*

Description

Tests if vector could be a another data type without errors.

Usage

```
could_chr_be_num(.x)

could_chr_be_int(.x)

could_num_be_int(.x)

could_chr_be_dtm(.x)

could_dtm_be_dte(.x)
```

Arguments

`.x`	vector of the data type that should be tested.

Details

The name logic of `could_chr_be_num` should be interpreted as: Could this character vector be a numeric vector? The same logic goes for all functions named could_this_be_that.

Value

TRUE or FALSE

See Also

`vignette("s")`, `vignette("hablar")`

Examples

```
x <- c("1", "3", "7")
could_chr_be_num(x)
could_chr_be_int(x)

x <- c("abc", "3", "Hello world")
could_chr_be_num(x)

x <- c(NA, "3.45", "5,98")
could_chr_be_num(x)
could_chr_be_int(x)

x <- as.numeric(c(3.45, 1.5))
could_num_be_int(x)

x <- as.numeric(c(7, 2))
could_num_be_int(x)
```

`create_dummy` *Create a simple dummy*

Description

Creates a vector of the integers 1 and 0. If condition is true it returns 1. If false 0. If condition returns NA it returns NA, if not explicitly not stated than NA should be replaced.

Usage

```
dummy(condition, missing = NA)
dummy_(condition, missing = 0L)

dummy_(condition, missing = 0L)
```

Arguments

`condition`	a predicament
`missing`	a replacement if condition is NA

Value

a vector of the integers 1, 0 and NA (if not dummy_ is used).

See Also

```
vignette("hablar")
```

Examples

```
v <- c(10, 5, 3, NA, 9)
dummy(v > 5)
dummy_(v > 5)
```

`cumulative_` *cumulative_*

Description

cumulative functions. 'cumsum_' is the cumulative sum ignoring missing values. 'cum_unique' counts the cumulative unique value including NA as ONE value. 'cum_unique_' ignores missing values

Usage

```
cumsum_(.v, ignore_na = TRUE)

cummean_(.v, ignore_na = TRUE)

cum_unique(.v, ignore_na = FALSE)

cum_unique_(.v, ignore_na = TRUE)
```

Arguments

`.v`	a vector
`ignore_na`	should missing values be ignores?

Value

a vector

`find_in_df` *Special filters*

Description

Filters a data frame for special cases. For example, find_duplicates() returns all rows that are duplicates. If variables are passed to the function then duplicates for those variables are returned.

Usage

```
find_duplicates(.data, ...)

find_na(.data, ...)

find_irrational(.data, ...)

find_nan(.data, ...)

find_inf(.data, ...)
```

Arguments

.data	a data frame
...	variables that should be considered. If empty, all variables are used.

Details

irrational values are Inf and NaN

Value

a filtered data frame

See Also

vignette("s"), vignette("hablar")

check_df to return TRUE or FALSE instead of rows.

Examples

```
## Not run:
df <- data.frame(a = c("A", NA, "B", "C", "C"),
                 b = c(NA, 1, 1, 3, 3),
                 c = c(7, 8, 2, 3, 3),
                 stringsAsFactors = FALSE)

# Returns duplicated rows
df %>% find_duplicates()

# Returns duplicates in specific variables
df %>% find_duplicates(b:c)

# Returns rows where NA in variable b
df %>% find_na(b)

## End(Not run)
```

given *given*

Description

Simple function that filters a vector while helping with missing values. Replacing expression like 'x[x > 3 & !is.null(x)]'

Usage

```
given(.x, .y, ignore_na = FALSE)

given_(.x, .y, ignore_na = TRUE)
```

Arguments

.x	the vector to filter
.y	a logical vector to filter with
ignore_na	should NA be removed?

Value

a vector

if_else_ *if_this_else_that_*

Description

A vectorised if or else function. It checks that the true or false (or the optional missing) arguments have the same type. However it accepts a generic NA. Built upon dplyr's [if_else()] function. The only difference is that the user do not have to specify the type of NA. if_else_ is faster than base [ifelse()] and a tad slower than dplyr's [if_else()]. Attributes are taken from either true or false because one generic NA.

Usage

```
if_else_(condition, true, false, missing = NULL)
```

Arguments

condition	logical vector
true	value to replace if condition is true. Must be same length as condition or 1.
false	value to replace if condition is false. Must be same length as condition or 1.
missing	optional. a replacement if condition returns NA. Must be same length as condition or 1.

Details

If the returning vector have attributes (e.g. for factors) it returns the attributes for the first non-generic NA in the order true, false and then missing.

Value

a vector

See Also

`vignette("s")`, `vignette("hablar")`

Examples

```
v <- c(TRUE, FALSE, TRUE, FALSE)
if_else_(v, "true", "false")

v <- c(TRUE, FALSE, NA, FALSE)
if_else_(v, 1, NA, 999)
```

`math ignore NA in math funs`

Ignore NA in math

Description

Simplifying math functions are simple wrappers of math function (- +). If any of the left-hand side or right-hand side is NA, Inf or NaN it returns any rational value, if there is any.

However, if the both values are irrational it returns NA. The result is then passed to the corresponding math function.

Usage

```
.x %minus_% .y

.x %plus_% .y
```

Arguments

`.x`	numeric or integer element
`.y`	numeric or integer element

Value

a single value

See Also

vignette("s"), vignette("hablar")

Examples

```
## Not run: # The simplest case
3 %minus_% 2

# But with NA it returns 3 as if the NA were zero
3 %minus_% NA

# It doesnt matter if the irrational number is on left- or right-hand.
NA %plus_% 5

## End(Not run)
```

---

n_unique count unique elements

n_unique

Description

Simple wrapper for length(unique(.x)). If you use n_unique_(.x) then NA is ignored when counting.

Usage

```
n_unique(.x, ignore_na = FALSE)

n_unique_(.x, ignore_na = TRUE)

n_unique_(.x, ignore_na = TRUE)
```

Arguments

.x	a vector
ignore_na	a logical indicating whether missing values should be removed

Value

a single numeric vector of the same length as the data frame it is applied to.

See Also

vignette("s"), vignette("hablar")

Examples

```
# Simple
n_unique(c(1, 2, 2, 3))

# Same result as above eventhough vector includes NA
n_unique_(c(1, 2, 2, 3, NA))
```

`rationalize` *Only allow rational values in numeric vectors* `rationalize` *transforms all numeric elements to be rational values or NA, thus removes all* `NaN`, `Inf` *and replaces them with* `NA`.

Description

Only allow rational values in numeric vectors

`rationalize` transforms all numeric elements to be rational values or NA, thus removes all `NaN`, `Inf` and replaces them with `NA`.

Usage

```
rationalize(.x, ...)

## Default S3 method:
rationalize(.x, ...)

## S3 method for class 'numeric'
rationalize(.x, ...)

## S3 method for class 'data.frame'
rationalize(.x, ...)
```

Arguments

`.x`	vector or data.frame
`...`	columns to be evaluated. Only applicable if .x is a data frame.

Details

#' If a non-numeric vector is passed, it is unchanged. If a data.frame is passed, it evaluates all columns separately.

Value

For vectors: same data type/class as .x.

For data.frame: a tbl data frame.

NULL

NULL

NULL

See Also

s, rationalize, vignette("s"), vignette("hablar")

Examples

```
x <- c(3, -Inf, 6.56, 9.3, NaN, 5, -Inf)
rationalize(x)

df <- data.frame(num_col = c(Inf, 3, NaN),
                 chr_col = c("a", "b", "c"),
                 stringsAsFactors = FALSE)
df
rationalize(df)
```

---

repeat_df *repeat_df*

Description

Repeats a data frame n times. Useful for testing on large data frames.

Usage

```
repeat_df(.df, n, id = NULL)
```

Arguments

.df	a data frame
n	times the data frame should be repeated
id	a character element that creates a column with a number for each repetition

Value

a vector of the integers 1, 0 and NA (if not dummy_ is used).

See Also

vignette("hablar")

Examples

```
repeat_df(mtcars, 2)
```

`replacers` *replacemnt and specials*

Description

If-this-type-then replace with x. And the other way around; replace with x if this.

Usage

```
if_na(.x, replacement, missing = NULL)

if_nan(.x, replacement, missing = NULL)

if_inf(.x, replacement, missing = NULL)

if_zero(.x, replacement, missing = NULL)

na_if(.x, condition, replace_na = FALSE)

nan_if(.x, condition, replace_na = FALSE)

inf_if(.x, condition, replace_na = FALSE)

zero_if(.x, condition, replace_na = FALSE)

if_not_na(.x, replacement, missing = NULL)

if_inf(.x, replacement, missing = NULL)

if_nan(.x, replacement, missing = NULL)

if_zero(.x, replacement, missing = NULL)

na_if(.x, condition, replace_na = FALSE)

inf_if(.x, condition, replace_na = FALSE)

nan_if(.x, condition, replace_na = FALSE)

zero_if(.x, condition, replace_na = FALSE)
```

Arguments

.x	a vector
replacement	a replacement if condition is TRUE
missing	a value that replace missing values in condition.
condition	a predicament
replace_na	if TRUE, missing values in condition will be replaced as well

Value

a vector

See Also

vignette("s"), vignette("hablar")

Examples

```
v <- c(1, NA, 2)
if_na(v, 100)

v <- c(999, NA, 2)
zero_if(v, v == 999)
```

---

retype *Return simple data types*

Description

retype transforms all elements into simple classes. The simple classes are date, numeric and character. By transforming all elements to these classes no information is lost, while simplifying the object. See details below for more information or type vignette("retype") in the console.

Usage

```
retype(.x, ...)

## Default S3 method:
retype(.x, ...)

## S3 method for class 'logical'
retype(.x, ...)

## S3 method for class 'integer'
retype(.x, ...)
```

```
## S3 method for class 'Date'
retype(.x, ...)

## S3 method for class 'POSIXct'
retype(.x, ...)

## S3 method for class 'numeric'
retype(.x, ...)

## S3 method for class 'list'
retype(.x, ...)

## S3 method for class 'data.frame'
retype(.x, ...)
```

Arguments

.x	vector or data.frame
...	column names to be evaluated. Only if .x is a data frame.

Details

Each vector past to retype is reclassified into the highest position in a simplification hierarchy without loosing any information. This means that: Factors are converted to characters. However, character vectors (or vectors changed to character initially) are checked to see if they could be a numeric vector without error. If so, it is transformed into a numeric vector which is higher in the hierarchy. Vectors of class logical, integer are changed to numerical. Dates and date time (POSIXct) goes through the same procedure. Lists and complex vectors are left unchanged because the are neither simple nor complicated.

Value

For vectors: simple class of .x.

For data.frame: a tbl data frame with simple classes.

NULL

NULL

NULL

NULL

NULL

NULL

NULL

NULL

See Also

s, rationalize #' vignette("retype"), vignette("s"), vignette("hablar")

Examples

```
# Dates
dte <- as.Date(c("2018-01-01", "2016-03-21", "1970-01-05"))
retype(dte)
retype(dte)

# Factors
fct <- as.factor(c("good", "bad", "average"))
retype(dte)

# Character that only contains numeric elements
num_chr <- c("3","4.0", "3,5")
retype(num_chr)

# Logical
lgl <- c(TRUE, FALSE, TRUE)
retype(lgl)

# Data frame with all the above vectors
df <- data.frame(dte = dte,
                 fct = fct,
                 num_chr = num_chr,
                 lgl = lgl,
                 stringsAsFactos = FALSE)
df
retype(df)
```

---

s *Make vector shorter and simpler*

Description

s means simple and short. It removes all non-values, i.e. NA,Inf,NaN from a vector. However, if the length is 0 it returns NA. It is useful in combination with summary functions, e.g. mean, sum or min, when an answer is desired, if there is one in the data. In any other case NA is returned. Type vignette("s") in the console for more information.

Usage

```
s(.x, ignore_na = TRUE)
```

Arguments

.x	one vector. Does not work for factors.
ignore_na	if TRUE then NA omitted from results, as long as any non-NA element is left.

Value

a shortened and simplified vector

See Also

retype, rationalize, vignette("s"), vignette("hablar")

Examples

```
## Not run:
library(dplyr)

## s on a weird numeric vector
vector <- c(7, NaN, 6, -Inf, 5, 4, NA)
s(vector)

## Sum vector with non-rational values
vector <- c(7, NaN, -Inf, 4)
# Base R
sum(vector)
# With s
sum(s(vector))

## Max of vector with only NA
# Base R
max(vector, na.rm = TRUE)
# With s
max(s(vector))

## First of vector when NA is first element
vector <- c(NA, "X", "Y")
# dplyr R
first(vector)
# With s
first(s(vector))

## Use of s when NA should not be removes
vector <- c(7, Inf, NA, 4)
# Base R
sum(vector)
# With s
sum(s(vector, ignore_na = FALSE))

## s when summarizing a weird data.frame
df_test <- data.frame(a = c(NaN, 1, -Inf, 3),
                      b = c(NA, "Q", "P", "P"),
                      c = c(NA, NA, NA, NA),
                      stringsAsFactors = FALSE)
df_test

# Base R aggregation with dplyr's summarize
summarise(df_test, mean_a = mean(a),
                   min_c = min(c, na.rm = TRUE))
# With s
summarise(df_test, mean_a = mean(s(a)),
                   min_c = min(s(c)))
```

```
## End(Not run)
```

`set_wd_to_script_path` *Set wd to script path*

Description

Sets working directory to the path where the R-script is located. Only works inside [Rstudio] and in a script (i.e. not in the console). Additionally, the R-script needs to be saved in a path to work.

Usage

```
set_wd_to_script_path()
```

Value

NULL. In the background the working directory has changed if not any errors occurred.

`this_date` *this_date*

Description

Returns the current day, month or year. Day and month returns dates and year a 4 digit number.

Usage

```
this_day()

this_month()

this_year()
```

Value

a date or number

Examples

```
this_day()
this_month()
this_year()
```

`wrapper - s and summary funs`

Combine aggregate functions and s

Description

`[summary function_*]` functions are simple wrappers of aggregate function and the `s` function. `s` removes all non-values, i.e. `NA,Inf,NaN` from a vector. However, if the length is 0 it returns NA. The result is then passed to the corresponding aggregation function. For example, `min_(x)` is identical to `min(s(x))`. Please read `vignette("s")` for more information.

Usage

```
max_(.x, ignore_na = TRUE)

min_(.x, ignore_na = TRUE)

sum_(.x, ignore_na = TRUE)

mean_(.x, ignore_na = TRUE)

median_(.x, ignore_na = TRUE)

sd_(.x, ignore_na = TRUE)

var_(.x, ignore_na = TRUE)

first_(.x, ignore_na = TRUE)

last_(.x, ignore_na = TRUE)

first_non_na(.x)

squeeze(.x, ignore_na = FALSE)

squeeze_(.x, ignore_na = TRUE)
```

Arguments

`.x`	a single vector
`ignore_na`	if false missing values are not omitted.

Details

'first_non_na' is a faster version of 'first' since it only search for a non NA value until it finds one. 'squeeze' on the other hand checks if all elements are equal and then returns only that value.

Value

a single aggregated value

See Also

`vignette("convert")`, `vignette("hablar")`

Examples

```
## sum_ on non-rational numeric vector
vector <- c(7, NaN, -Inf, 4)
sum_(vector)

## Min of vector with length 0
vector <- c()
# With a wrapped s
min_(vector)

## Max of vector with only NA
# With a wrapped s
max_(vector)

## Use of s when NA should not be removed
vector <- c(7, Inf, NA, 4)
# With a wrapped s
sum_(vector, ignore_na = FALSE)
```

Index