Package 'clean'

August 20, 2019

Title Fast and Easy Data Cleaning

Version 1.1.0

Date 2019-08-15

Description Data cleaning functions for classes 'logical', 'factor', 'numeric', 'character', 'currency' and 'Date' to make data cleaning fast and easy. Relying on very few dependencies, it provides smart guessing, but with user options to override anything if needed.

Depends R (>= 3.0.0)

Imports crayon, knitr, pillar, rlang (>= 0.3.1)

Suggests rmarkdown, testthat (>= 1.0.2)

URL https://github.com/msberends/clean

BugReports https://github.com/msberends/clean/issues

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

NeedsCompilation no

Author Matthijs S. Berends [aut, cre] (<https://orcid.org/0000-0001-7620-1800>)

Maintainer Matthijs S. Berends <m.s.berends@umcg.nl>

Repository CRAN

Date/Publication 2019-08-20 11:10:02 UTC

R topics documented:

clean 2
currency 5
format_datetime 6

freq 7
regex_true_false 10
unclean 11

Index **12**

`clean` *Clean column data to a class*

Description

Use any of these functions to quickly clean columns in your data set. Use `clean()` to pick the functions that return the least relative number of `NA`s. They **always** return the class from the function name (e.g. `clean_Date()` always returns class `Date`).

Usage

```
clean(x)

## S3 method for class 'data.frame'
clean(x)

clean_logical(x, true = regex_true(), false = regex_false(),
  na = NULL, fixed = FALSE, ignore.case = TRUE)

clean_factor(x, levels = unique(x), ordered = FALSE,
  droplevels = FALSE, fixed = FALSE, ignore.case = TRUE)

clean_numeric(x, remove = "[^0-9.,]", fixed = FALSE)

clean_character(x, remove = "[^a-z \t\r\n]", fixed = FALSE,
  ignore.case = TRUE, trim = TRUE)

clean_currency(x, currency_symbol = NULL, ...)

clean_Date(x, format = NULL, ...)

clean_POSIXct(x, remove = "[^.0-9 :/-]", fixed = FALSE, ...)
```

Arguments

`x`	data to clean
`true`	regex to interpret values as `TRUE` (which defaults to `regex_true`), see Details
`false`	regex to interpret values as `FALSE` (which defaults to `regex_false`), see Details
`na`	regex to force interpret values as `NA`, i.e. not as `TRUE` or `FALSE`
`fixed`	logical to indicate whether regular expressions should be turned off
`ignore.case`	logical to indicate whether matching should be case-insensitive

levels	new factor levels, may be named with regular expressions to match existing values, see Details
ordered	logical to indicate whether the factor levels should be ordered
droplevels	logical to indicate whether non-existing factor levels should be dropped
remove	regex to define the character(s) that should be removed, see Details
trim	logical to indicate whether the result should be trimmed with trimws
currency_symbol	the currency symbol to use, which will be guessed based on the input and otherwise defaults to the current system locale setting (see Sys.localeconv)
...	other parameters passed on to as.Date or as.POSIXct
format	a date format that will be passed on to format_datetime, see Details

Details

Using clean() on a vector will guess a cleaning function based on the potential number of NAs it returns. Using clean() on a data.frame to apply this guessed cleaning over all columns.

Info about the different functions:

- clean_logical():
  Use parameters true and false to match values using case-insensitive regular expressions (regex). Unmatched values are considered NA. At default, values are matched with regex_true and regex_false. This allows support for values "Yes" and "No" in the following languages: Arabic, Bengali, Chinese (Mandarin), Dutch, English, French, German, Hindi, Indonesian, Japanese, Malay, Portuguese, Russian, Spanish, Telugu, Turkish and Urdu. Use parameter na to override values as NA that would else be matched with true or false. See Examples.
- clean_factor():
  Use parameter levels to set new factor levels. They can be case-insensitive regular expressions to match existing values of x. For matching, new values for levels are internally temporary sorted descending on text length. See Examples.
- clean_numeric() and clean_character():
  Use parameter remove to match values that must be removed from the input, using regular expressions (regex). In case of clean_numeric(), comma's will be read as dots and only the last dot will be kept. Function clean_character() will keep middle spaces at default. See Examples.
- clean_currency():
  This new class works like clean_numeric(), but transforms it with as.currency. The currency symbol is guessed based on the most traded currencies by value (see Source): the United States dollar, Euro, Japanese yen, Pound sterling, Swiss franc, Renminbi, Swedish krona, Mexican peso, South Korean won, Turkish lira, Russian ruble, Indian rupee and the South African rand. See Examples.
- clean_Date():
  Use parameter format to define a date format, or leave it empty to have the format guessed. Use "Excel" to read values as Microsoft Excel dates. The format parameter will be evaluated with format_datetime, which means that a format like "d-mmm-yy" with be translated internally to "%e-%b-%y" for convenience. See Examples.

- `clean_POSIXct()`:
  Use parameter `remove` to match values that must be removed from the input, using regular expressions (regex). The resulting string will be coerced to a date/time element with class `POSIXct`, using `as.POSIXct()`. See Examples.

The use of invalid regular expressions in any of the above functions will not return an error (like in base R), but will instead interpret the expression as a fixed value and will throw a warning.

Value

The `clean` functions **always** return the class from the function name:

- `clean_logical()`: class `logical`
- `clean_factor()`: class `factor`
- `clean_numeric()`: class `numeric`
- `clean_character()`: class `character`
- `clean_currency()`: class `currency`
- `clean_Date()`: class `Date`
- `clean_POSIXct()`: classes `POSIXct/POSIXt`

Source

Triennial Central Bank Survey Foreign exchange turnover in April 2016 (PDF). Bank for International Settlements. 11 December 2016. p. 10.

Examples

```
clean_logical(c("Yes", "No"))   # English
clean_logical(c("Oui", "Non"))  # French
clean_logical(c("ya", "tidak")) # Indonesian
clean_logical(x = c("Positive", "Negative", "Unknown", "Some value"),
              true = "pos", false = "neg")

gender_age <- c("male 0-50", "male 50+", "female 0-50", "female 50+")
clean_factor(gender_age, c("M", "F"))
clean_factor(gender_age, c("Male", "Female"))
clean_factor(gender_age, c("0-50", "50+"), ordered = TRUE)

clean_Date("13jul18", "ddmmmyy")
clean_Date("12 august 2010")
clean_Date("12 06 2012")
clean_Date(36526) # Excel date
clean_Date("43658")
clean_Date("14526", "Excel") # "1939-10-08"

clean_POSIXct("Created log on 2019/04/11 11:23 by user Joe")

clean_numeric("qwerty123456")
clean_numeric("Positive (0.143)")
clean_numeric("0,143")
```

```
clean_character("qwerty123456")
clean_character("Positive (0.143)")

clean_currency(c("Received $ 25", "Received $ 31.40"))
clean_currency(c("Jack sent £ 25", "Bill sent £ 31.40"))

clean("12 06 2012")
clean(data.frame(dates = "2013-04-02",
                 logicals = c("yes", "no")))
```

---

currency *Transform to currency*

Description

Transform input to a currency. The actual values are numeric, but will be printed as formatted currency values.

Usage

```
as.currency(x, currency_symbol = Sys.localeconv()["int_curr_symbol"],
  ...)

is.currency(x)

## S3 method for class 'currency'
print(x, decimal.mark = getOption("OutDec"),
  big.mark = ifelse(decimal.mark == ",", ".", ","), ...)

## S3 method for class 'currency'
format(x,
  currency_symbol = attributes(x)$currency_symbol,
  decimal.mark = getOption("OutDec"), big.mark = ifelse(decimal.mark ==
  ",", ".", ","), ...)
```

Arguments

x	input
currency_symbol	the currency symbol to use, which defaults to the current system locale setting (see Sys.localeconv)
...	other parameters passed on to methods
decimal.mark	symbol to use as a decimal separator, defaults to getOption("OutDec")
big.mark	symbol to use as a thousands separator, defaults to a dot if decimal.mark is a comma, and a comma otherwise

Details

Printing currency will always have a currency symbol followed by a space, 2 decimal places and is never written in scientific format (like 2.5e+04).

Examples

```
money <- as.currency(c(0.25, 2.5, 25, 25000))
money
sum(money)
max(money)
mean(money)

format(money, currency_symbol = "$")
format(money, currency_symbol = "€", decimal.mark = ",")

as.currency(2.5e+04)
```

`format_datetime` *Readable date format to POSIX*

Description

Use this function to transform generic date/time info writing (dd-mm-yyyy) to POSIX standardised format (%d-%m-%Y), see Examples.

Usage

```
format_datetime(format)
```

Arguments

`format`	the format that needs to be transformed

Value

A `character` string (a POSIX standardised format)

Examples

```
format_datetime("yyyy/mm/dd")

# Very hard to remember all these characters:
format(Sys.time(), "%a %b %d %Y %X")

# Easy to remember and write the same as above:
format(Sys.time(), format_datetime("ddd mmm dd yyyy HH:MM:ss"))
```

`freq` *Frequency table*

Description

Create a frequency table of a `vector` or a `data.frame`. It supports tidyverse's quasiquotation and markdown for reports. Easiest practice is: `data %>% freq(var)` using the tidyverse.

`top_freq` can be used to get the top/bottom *n* items of a frequency table, with counts as names. It respects ties.

Usage

```
freq(x, ...)

## Default S3 method:
freq(x, sort.count = TRUE,
  nmax = getOption("max.print.freq"), na.rm = TRUE, row.names = TRUE,
  markdown = !interactive(), digits = 2, quote = NULL,
  header = TRUE, title = NULL, na = "<NA>", sep = " ",
  decimal.mark = getOption("OutDec"), big.mark = "", ...)

## S3 method for class 'factor'
freq(x, ..., droplevels = FALSE)

## S3 method for class 'matrix'
freq(x, ..., quote = FALSE)

## S3 method for class 'table'
freq(x, ..., sep = " ")

## S3 method for class 'numeric'
freq(x, ..., digits = 2)

## S3 method for class 'Date'
freq(x, ..., format = "yyyy-mm-dd")

## S3 method for class 'hms'
freq(x, ..., format = "HH:MM:SS")

is.freq(f)

top_freq(f, n)

header(f, property = NULL)

## S3 method for class 'freq'
```

```
print(x, nmax = getOption("max.print.freq", default = 10),
  markdown = !interactive(), header = TRUE,
  decimal.mark = getOption("OutDec"), big.mark = ifelse(decimal.mark !=
  ",", ",", "."), ...)
```

Arguments

x	vector of any class or a data.frame or table
...	up to nine different columns of x when x is a data.frame or tibble, to calculate frequencies from - see Examples. Also supports quasiquotion.
sort.count	sort on count, i.e. frequencies. This will be TRUE at default for everything except when using grouping variables.
nmax	number of row to print. The default, 10, uses getOption("max.print.freq"). Use nmax = 0, nmax = Inf, nmax = NULL or nmax = NA to print all rows.
na.rm	a logical value indicating whether NA values should be removed from the frequency table. The header (if set) will always print the amount of NAs.
row.names	a logical value indicating whether row indices should be printed as 1:nrow(x)
markdown	a logical value indicating whether the frequency table should be printed in markdown format. This will print all rows (except when nmax is defined) and is default behaviour in non-interactive R sessions (like when knitting RMarkdown files).
digits	how many significant digits are to be used for numeric values in the header (not for the items themselves, that depends on getOption("digits"))
quote	a logical value indicating whether or not strings should be printed with surrounding quotes. Default is to print them only around characters that are actually numeric values.
header	a logical value indicating whether an informative header should be printed
title	text to show above frequency table, at default to tries to coerce from the variables passed to x
na	a character string that should be used to show empty (NA) values (only useful when na.rm = FALSE)
sep	a character string to separate the terms when selecting multiple columns
decimal.mark	used for prettying (longish) numerical and complex sequences. Passed to prettyNum: that help page explains the details.
big.mark	used for prettying (longish) numerical and complex sequences. Passed to prettyNum: that help page explains the details.
droplevels	a logical value indicating whether in factors empty levels should be dropped
format	a character to define the printing format (it supports format_datetime to transform e.g. "d mmmm yyyy" to "%e %B %Y")
f	a frequency table
n	number of top *n* items to return, use -n for the bottom *n* items. It will include more than n rows if there are ties.
property	property in header to return this value directly

Details

Frequency tables (or frequency distributions) are summaries of the distribution of values in a sample. With the 'freq' function, you can create univariate frequency tables. Multiple variables will be pasted into one variable, so it forces a univariate distribution. This package also has a vignette available to explain the use of this function further, run `browseVignettes("clean")` to read it.

For numeric values of any class, these additional values will all be calculated with `na.rm = TRUE` and shown into the header:

- Mean, using `mean`
- Standard Deviation, using `sd`
- Coefficient of Variation (CV), the standard deviation divided by the mean
- Mean Absolute Deviation (MAD), using `mad`
- Tukey Five-Number Summaries (minimum, Q1, median, Q3, maximum), see *NOTE* below
- Interquartile Range (IQR) calculated as `Q3 - Q1`, see *NOTE* below
- Coefficient of Quartile Variation (CQV, sometimes called coefficient of dispersion) calculated as `(Q3 - Q1) / (Q3 + Q1)`, see *NOTE* below
- Outliers (total count and percentage), using `boxplot.stats`

NOTE: These values are calculated using the same algorithm as used by Minitab and SPSS: *p[k] = E[F(x[k])]*. See Type 6 on the `quantile` page.

For dates and times of any class, these additional values will be calculated with `na.rm = TRUE` and shown into the header:

- Oldest, using `min`
- Newest, using `max`, with difference between newest and oldest

In factors, all factor levels that are not existing in the input data will be dropped at default.

The function `top_freq` will include more than `n` rows if there are ties. Use a negative number for *n* (like `n = -3`) to select the bottom *n* values.

Value

A `data.frame` (with an additional class `"freq"`) with five columns: `item`, `count`, `percent`, `cum_count` and `cum_percent`.

Extending the `freq()` function

Interested in extending the `freq()` function with your own class? Add a method like below to your package, and optionally define some header info by passing a `list` to the `.add_header` parameter, like below example for class `difftime`. This example assumes that you use the `roxygen2` package for package development.

```
#' @exportMethod freq.difftime
#' @importFrom clean freq.default
#' @export
#' @noRd
```

```
freq.difftime <- function(x, ...) {
  freq.default(x = x, ...,
               .add_header = list(units = attributes(x)$units))
}
```

Be sure to call `freq.default` in your function and not just `freq`. Also, add `clean` to the `Imports:` field of your `DESCRIPTION` file, to make sure that it will be installed with your package, e.g.:

```
Imports: clean
```

Examples

```
## Not run:

# this all gives the same results:
freq(df$variable)
freq(df[, "variable"])
df$variable %>% freq()
df[, "variable"] %>% freq()
df %>% freq("variable")
df %>% freq(variable) # <- tidyverse way

## End(Not run)

clean_gender <- clean_factor(unclean$gender,
                             levels = c("^m" = "Male",
                                        "^f" = "Female"))
freq(unclean$gender)
freq(clean_gender)
```

---

regex_true_false *Regular expressions for* TRUE *and* FALSE

Description

These functions just return a regular expression to define values `TRUE` and `FALSE` in the most spoken languages in the world. They are the default input for the function `clean_logical`.

Usage

```
regex_true()

regex_false()
```

Details

Both functions support values "Yes" and "No" in the following languages: Arabic, Bengali, Chinese (Mandarin), Dutch, English, French, German, Hindi, Indonesian, Japanese, Malay, Portuguese, Russian, Spanish, Telugu, Turkish and Urdu.

Note: all these translations are in Latin characters only (e.g. "da" for Russian, "haan" for Hindi and "hai" for Japanese).

Source

Wolfram Alpha, query: https://www.wolframalpha.com/input/?i=20+most+spoken+languages

---

`unclean` *Example data that is not clean*

Description

This typical data example can be used for checking and cleaning.

Usage

```
unclean
```

Format

A `data.frame` with 500 observations and the following variables:

`date` Dates imported from Excel, they are integers ranging from ~30,000 to ~43,000.

`gender` Characters with mixed values observed in original data about patients gender.

See Also

`freq` to check values and `clean` to clean them.

Index

∗Topic **datasets**
unclean, 11
∗Topic **frequency**
freq, 7
∗Topic **freq**
freq, 7
∗Topic **summarise**
freq, 7
∗Topic **summary**
freq, 7

as.currency, *3*
as.currency *(*currency*)*, 5
as.Date, *3*
as.POSIXct, *3, 4*

boxplot.stats, *9*

clean, 2, *11*
clean_character *(*clean*)*, 2
clean_currency *(*clean*)*, 2
clean_Date *(*clean*)*, 2
clean_factor *(*clean*)*, 2
clean_logical, *10*
clean_logical *(*clean*)*, 2
clean_numeric *(*clean*)*, 2
clean_POSIXct *(*clean*)*, 2
currency, 5

data.frame, *8*, *11*

format.currency *(*currency*)*, 5
format_datetime, *3*, 6, *8*
freq, 7, *11*

getOption, *5*, *8*

header *(*freq*)*, 7

is.currency *(*currency*)*, 5
is.freq *(*freq*)*, 7

list, *9*

mad, *9*
max, *9*
mean, *9*
min, *9*

prettyNum, *8*
print.currency *(*currency*)*, 5
print.freq *(*freq*)*, 7

quantile, *9*

regex, *2–4*
regex_false, *2, 3*
regex_false *(*regex_true_false*)*, 10
regex_true, *2, 3*
regex_true *(*regex_true_false*)*, 10
regex_true_false, 10

sd, *9*
Sys.localeconv, *3*, *5*

table, *8*
top_freq *(*freq*)*, 7
trimws, *3*

unclean, 11