Package 'framecleaner'

October 13, 2022

Type Package

Title Clean Data Frames

Version 0.2.0

Maintainer Harrison Tietze <Harrison4192@gmail.com>

Description Provides a friendly interface for modifying data frames with a sequence of piped commands built upon the 'tidyverse' Wickham et al., (2019) <doi:10.21105/joss.01686> . The majority of commands wrap 'dplyr' mutate statements in a convenient way to concisely solve common issues that arise when tidying small to medium data sets. Includes smart defaults and allows flexible selection of columns via 'tidyselect'.

License MIT + file LICENSE

Encoding UTF-8

Imports dplyr, stringr, tidyselect, purrr, janitor, rlang, lubridate, magrittr, tibble, rstudioapi, forcats, bit64, rio, readr, vroom, fs, rlist, fastDummies

RoxygenNote 7.1.2

Suggests knitr, rmarkdown, badger, readxl

VignetteBuilder knitr

URL https://harrison4192.github.io/framecleaner/, https://github.com/Harrison4192/framecleaner

BugReports https://github.com/Harrison4192/framecleaner/issues

NeedsCompilation no

Author Harrison Tietze [aut, cre]

Repository CRAN

Date/Publication 2021-11-17 05:40:02 UTC

R topics documented:

as_integer16_or_64 2
auto_setwd 3

clean_frame . . . 3
create_dummies . . . 4
create_flag . . . 5
fill_na . . . 6
filter_for . . . 7
filter_missing . . . 8
import_dir . . . 9
import_tibble . . . 9
make_na.data.frame . . . 10
pad_auto . . . 11
pad_col . . . 12
recode_chr . . . 12
relocate_all . . . 13
remove_whitespace . . . 14
select_otherwise . . . 15
set_chr . . . 16
set_date . . . 16
set_dbl . . . 17
set_fct . . . 18
set_int . . . 19
set_lgl.data.frame . . . 21

Index **22**

`as_integer16_or_64` *as_integer16_or_64*

Description

coerce to integer. if too large, coerces to 64-bit integer

Usage

```
as_integer16_or_64(x)
```

Arguments

x	integerish vec

Value

int or int64

auto_setwd *auto setwd*

Description

Call from a saved R script. Automatically sets your working directory to the directory that you saved the current R script in. Takes no arguments.

Usage

```
auto_setwd()
```

Value

No return value.

clean_frame *Clean Data Frame*

Description

Uses the functions of framecleaner and other operations to apply cleaning operations to a data frame

Usage

```
clean_frame(.data)
```

Arguments

.data	a data frame

Details

Functions applied in `clean_frame`

- `remove_empty`
- `rename_with .fn = enc2utf8`
- `clean_names case = "all_caps", ascii = FALSE)`
- `set_int`
- `set_date`
- `make_na`
- `as_tibble`

Value

data frame

Examples

```
iris %>%
clean_frame()
```

`create_dummies` *create dummies*

Description

adapted from the `dummy_cols` function Added the option to truncate the dummy column names, and to specify dummy cols using tidyselect.

Usage

```
create_dummies(
  .data,
  ...,
  append_col_name = TRUE,
  max_levels = 10L,
  remove_first_dummy = FALSE,
  remove_most_frequent_dummy = FALSE,
  ignore_na = FALSE,
  split = NULL,
  remove_selected_columns = TRUE
)
```

Arguments

`.data`	data frame
`...`	tidyselect columns. default selection is all character or factor variables
`append_col_name`	logical, default TRUE. Appends original column name to dummy col name
`max_levels`	uses `fct_lump_n` to limit the number of categories. Only the top n levels are preserved, and the rest being lumped into "other". Default is set to 10 levels, to prevent accidental overload. Set value to `Inf` to use all levels
`remove_first_dummy`	logical, default FALSE.
`remove_most_frequent_dummy`	logical, default FALSE
`ignore_na`	logical, default FALSE
`split`	NULL
`remove_selected_columns`	logical, default TRUE

Details

reference the fastDummies package for documentation on the original function.

Value

data frame

Examples

```
iris %>%
create_dummies(Species, append_col_name = FALSE) %>%
  tibble::as_tibble()
```

`create_flag` *create flag*

Description

create flag

Usage

```
create_flag(.data, col, flag, full_name = FALSE, drop = FALSE)
```

Arguments

`.data`	data frame
`col`	column
`flag`	column entry
`full_name`	Logical. default F. if T, new column name is original name + flag. other wise just flag
`drop`	logical. default F. If T, drop original column.

Value

data frame

Examples

```
iris %>%
  create_flag(
  col = Species,
  flag = "versicolor",
  drop = TRUE) %>%
  head()
```

`fill_na` *Fill NAs*

Description

use tidyselect to fill NA values Default behavior is to fill all integer or double columns cols with 0, preserving their types.

Usage

```
fill_na(.data, ..., fill = 0L, missing_type = c("all", "NA", "NaN", "Inf"))
```

Arguments

`.data`	data frame
`...`	tidyselect specification. Default selection: none
`fill`	value to fill missings
`missing_type`	character vector. Choose what type of missing to fill. Default is all types. choose from "all", "Na", "NaN", "Inf"

Value

data frame

Examples

```
tibble::tibble(x = c(NA, 1L, 2L, NA, NaN, 5L, Inf)) -> tbl

tbl %>%
  fill_na()

tbl %>%
 fill_na(fill = 1L, missing_type = "Inf")

tbl %>%
 fill_na(missing_type = "NaN")
```

filter_for *filter for*

Description

Filter for all instances of a column that meet a specific condition at least once.

Usage

```
filter_for(.data, what, where)
```

Arguments

.data	data frame
what	unquote col or vector of unquoted cols.
where	a logical condition used for filter

Value

data frame

Examples

```
# An example using some time series data
tibble::tibble( CLIENT_ID = c("A1001", "B1001", "C1001",
"A1001", "B1001", "C1001", "A1001", "B1001", "C1001"),
                YEAR = c(2019L, 2019L, 2019L, 2020L, 2020L, 2020L, 2021L, 2021L, 2021L),
                SALES = c(3124, 56424, 3214132, 65534, 2342, 6566, 87654, 2332, 6565)
) %>%
dplyr::arrange(CLIENT_ID, YEAR) -> sales_data

sales_data

# filter for Clients that had sales greater than 4000 in the year 2019.
# this way we can see how the same clients sales looked in subsequent years

sales_data %>%
  filter_for(what = CLIENT_ID, where = YEAR == 2019 & SALES > 4000L)


# filter for clients whose sales were less than 4000 in the year 2021
 sales_data %>%
  filter_for(what = CLIENT_ID, where = YEAR == 2021 & SALES < 4000L)
```

`filter_missing` *filter out missings*

Description

More complex wrapper around `dplyr::filter(!is.na())` to remove `NA` rows using tidyselect. If any specified column contains an `NA` the whole row is removed. Reports the amount of rows removed containing `NaN`, `NA`, `Inf`, in that order. For example if one row contains `Inf` in one column and in another, the removed row will be counted in the `NA` tally.

Usage

```
filter_missing(.data, ..., remove_inf = TRUE)

## S3 method for class 'data.frame'
filter_missing(.data, ..., remove_inf = TRUE, condition = c("any", "all"))
```

Arguments

`.data`	dataframe
`...`	tidyselect. default selection is all columns
`remove_inf`	logical. default is to also remove `Inf` values. set to `FALSE` otherwise.
`condition`	defaults to "any". in which case removes rows if `NA` is in any specified column. "all" will remove rows only if each specified column is missing

Details

S3 method, can also be used on vectors

Value

data frame

Examples

```
tibble::tibble(x = c(NA, 1L, 2L, NA, NaN, 5L, Inf),
y = c(1L, NA, 2L, NA, Inf, 5L, Inf)) -> tbl1

tbl1

# remove any row with a missing or Inf
tbl1 %>%
filter_missing()

# remove any row with Na or NaN in the x column
tbl1 %>%
filter_missing(x, remove_inf = FALSE)
```

```
# only remove rows where every entry is Na, NaN, or Inf
tbl1 %>%
filter_missing(condition = "all")
```

`import_dir` *import directory*

Description

import directory

Usage

```
import_dir(
  dir,
  ...,
  method = c("rio", "vroom", "vroom_jp", "read_csv"),
  return_type = c("df", "list")
)
```

Arguments

`dir`	dir path
`...`	arguments passed to import method
`method`	import method chosen from import tibble
`return_type`	default is to bind dataframes together and remove duplicates. only recommended for a folder of files with the same data format. otherwise specify return as list of data frames

Value

data frame

`import_tibble` *import tibble*

Description

wrapper around multiple file readers. The default being [vroom] set to return a tibble, with [set_int] to encode integers. Also available is rio and vroom_jp for japanese characters.

Usage

```
import_tibble(
  path,
  ...,
  method = c("vroom", "rio", "vroom_jp", "read_csv", "read_excel")
)
```

Arguments

path	filepath
...	other arguments
method	method of import. default is rio

Details

Supports multiple types of importing through [method]

Value

a tibble

---

make_na.data.frame *Make NAs*

Description

Set elements to NA values using tidyselect specification. Don't use this function on columns of different modes at once. Defaults to choosing all character columns.

Usage

```
## S3 method for class 'data.frame'
make_na(.data, ..., vec = c("-", "", " ", "null", "NA", "NA_"))

make_na(.data, ..., vec = c("-", "", " ", "null", "NA", "NA_"))
```

Arguments

.data	data frame
...	tidyselect. Default selection: all chr cols
vec	vector of possible elements to replace with NA

Value

data frame

Examples

```
# easily set NA values. blank space and empty space are default options

tibble::tibble(x = c("a", "b", "", "d", " ", "", "e")) %>%
   make_na()
```

---

pad_auto *pad auto*

Description

Automatically pads elements of a column to the largest sized element. Useful when an integer code with leading zeros is read in as an integer and needs to be fixed.

Usage

```
pad_auto(mdb, ..., side = "left", pad = "0")
```

Arguments

mdb	data frame
...	tidyselect specification
side	str_pad side
pad	str_pad pad

Value

data frame

Examples

```
# good for putting leading 0's

tibble::tibble(x = 1:10) %>%
  pad_auto(x)
```

pad_col *pad column*

Description

wrapper around mutate and str_pad

Usage

```
pad_col(mdb, ..., width, pad = "0", side = "left")
```

Arguments

mdb	data frame
...	tidyselect
width	str_pad width
pad	str_pad pad
side	str_pad side

Value

data frame

Examples

```
# manually pad with 0's (or other value)
# use case over [pad_auto()]: the desired width is greater than the widest element

tibble::tibble(
ID = c(2, 13, 86, 302)
) %>%
pad_col(ID, width = 4)
```

recode_chr *recode_chr*

Description

recode_chr

Usage

```
recode_chr(df, col, old_names, new_name, regex = FALSE, negate = FALSE)
```

Arguments

df	data frame
col	unquoted col
old_names	character vector or regular expression
new_name	atomic chr string
regex	Logical, default F. Specify elements for old_names using a regex?
negate	logical, defailt F. If negating the regex, set to T

Value

df

Examples

```
# Use a negative regex to rename all species other than "virginica" to "none"

iris %>%
   recode_chr(
 col = Species,
 old_names = "vir",
 new_name = "none",
 regex = TRUE,
 negate = TRUE) %>%
 dplyr::count(Species)

# Specify old names using a regex

iris %>%
   recode_chr(
   col = Species,
   old_names = "set|vir",
   new_name = "other",
   regex = TRUE) %>%
   dplyr::count(Species)
```

---

relocate_all *Relocate All*

Description

Arranges columns alphabetically and then by type The user can supply a tidyselect argument to specify columns that should come first

Usage

```
relocate_all(.data, ..., regex = NULL)
```

Arguments

.data	data frame
...	a tidyselect specification
regex	a regular expression to match columns that will be put at the front of the df

Value

data frame

Examples

```
iris %>%
head %>%
relocate_all(matches("Petal"))
```

---

remove_whitespace *Remove Whitespace*

Description

Remove whitespace from columns using a tidyselect specification.

Usage

```
remove_whitespace(.data, ...)
```

Arguments

.data	data frame
...	tidyselect specification (default selection: all character columns)

Value

data frame

Examples

```
tibble::tibble(a = c(" a ", "b ", " c")) -> t1

t1

t1 %>%
remove_whitespace()
```

`select_otherwise` *select_otherwise*

Description

flexible select operator that powers the tidy consultant universe. Used to set sensible defaults and flexibly return the chosen columns. A developer focused function, but may be useful in interactive programming due to the ability to return different types.

Usage

```
select_otherwise(
  .data,
  ...,
  otherwise = NULL,
  col = NULL,
  return_type = c("names", "index", "df")
)
```

Arguments

`.data`	dataframe
`...`	tidyselect. columns to choose
`otherwise`	tidyselect. default columns to choose if ... is not specified
`col`	tidyselect. column to choose regardless of ... or otherwise specifications
`return_type`	choose to return column index, names, or df. defaults to index

Value

integer vector by default. possibly data frame or character vector

Examples

```
iris %>%
select_otherwise(where(is.double), return_type = "index")
```

set_chr *set character*

Description

set character

Usage

```
set_chr(.data, ...)
```

Arguments

.data	dataframe
...	tidyselect. Default selection: none

Value

dataframe

Examples

```
iris %>%
tibble::as_tibble() %>%
set_chr(tidyselect::everything())
```

set_date *set date*

Description

set dates manually or automatically

Usage

```
set_date(.data, ..., date_fn = lubridate::ymd)
```

Arguments

.data	dataframe
...	tidyselect
date_fn	a function to convert to a date object

Details

note: can be called without any ... arguments and instead automatically determines which character columns are actually dates, then proceeds to set them. It checks for the date specified in `date_fn` and also `ymd_hms`. On auto detect mode, it sets `ymd_hms` output to ymd dates instead of datetimes with hms. This is because of the common occurrence of trying to extract a `ymd` date from an excel workbook, and having it come with extra 00:00:00. If you need a datetime, manually supply the appropriate lubridate function.

Auto mode is experimental. Commonly detected error is a long character string of integers being interpreted as a date.

Value

tibble

Examples

```
tibble::tibble(date_col1 = c("20190101", "20170205"),
date_col2 = c("20201015", "20180909"),
not_date_col = c("a345", "b040")) -> t1

t1

t1 %>%
set_date()

t1 %>%
set_date(date_col1)
```

---

set_dbl *set double*

Description

set double

Usage

```
set_dbl(.data, ...)

## S3 method for class 'character'
set_dbl(.data, ...)

## S3 method for class 'factor'
set_dbl(.data, ...)

## S3 method for class 'Date'
set_dbl(.data, ...)
```

```
## S3 method for class 'numeric'
set_dbl(.data, ...)

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

Arguments

.data	dataframe
...	tidyselect. Default selection: none

Value

tibble

Examples

```
date_col <- c(lubridate::ymd(20180101), lubridate::ymd(20210420))


tibble::tibble(int = c(1L, 2L),
fct = factor(c(10, 11)),
date = date_col,
chr = c("a2.1", "rtg50.5")) -> t1

t1

t1 %>%
set_dbl(tidyselect::everything())

# s3 method works for vectors individually
# custom date coercion to represent date as a number. For lubridate's coercion method, use set_int
date_col %>%
set_dbl
```

---

set_fct *set factor*

Description

allows option to manually set the first level of the factor, for consistency with yardstick which automatically considers the first level as the "positive class" when evaluating classification.

Usage

```
set_fct(.data, ..., first_level = NULL, order_fct = FALSE, max_levels = Inf)

## S3 method for class 'data.frame'
set_fct(.data, ..., first_level = NULL, order_fct = FALSE, max_levels = Inf)

## Default S3 method:
set_fct(.data, ...)
```

Arguments

.data	dataframe
...	tidyselect (default selection: all character columns)
first_level	character string to set the first level of the factor
order_fct	logical. ordered factor?
max_levels	uses fct_lump_n to limit the number of categories. Only the top n levels are preserved, and the rest being lumped into "other"

Value

tibble

Examples

```
## simply set the first level of a factor

iris$Species %>% levels

iris %>%
  set_fct(Species, first_level = "virginica") %>%
  dplyr::pull(Species) %>%
  levels()
```

---

set_int *set integer*

Description

set integer

Usage

```
set_int(.data, ...)

## S3 method for class 'data.frame'
set_int(.data, ...)

## S3 method for class 'grouped_df'
set_int(.data, ...)
```

Arguments

.data	dataframe
...	tidyselect. Default Selecton: integerish doubles or integerish characters

Value

tibble

Examples

```
int_vec <- c("1", "2", "10")

tibble::tibble(
  chr_int = int_vec,
  dbl_int = c(1.0, 5.0, 20.0),
  chr_int64 = c("1033493932", "4432500065", "30303022192"),
  string_int = c("SALES2020", "SALES2021", "SALES2022")) -> tbl

# automatically coerce integerish cols in a tibble
tbl

# integerish doubles or chars will be detected for coercion automatically
tbl %>%
  set_int()

# string_int requires parsing, so it must be specified directly for coercion
tbl %>%
  set_int(matches("str|chr"))

# s3 method works for vectors as well

int_vec

int_vec %>%
set_int()
```

`set_lgl.data.frame` *set logical*

Description

note: for non-binary data, all values other than the true_level will be set to false

Usage

```
## S3 method for class 'data.frame'
set_lgl(.data, ..., true_level = 1L)

set_lgl(.data, ..., true_level = 1L)

## Default S3 method:
set_lgl(.data, ...)

## S3 method for class 'numeric'
set_lgl(.data, ..., true_level = 1L)

## S3 method for class 'character'
set_lgl(.data, ..., true_level = c("T", "TRUE"))
```

Arguments

`.data`	dataframe
`...`	tidyselect. Default selection: none
`true_level`	specify the value to set as TRUE. Default value is 1 for seamless conversion between logicals and integers. Can be given as a vector of values.

Value

dataframe

Examples

```
# convert a 1/0 vector back into T/F

tibble::tibble(x = c(1, 0, 0, 1, 0, 1)) %>%
set_lgl(x)
```

Index

as_integer16_or_64, 2
as_tibble, *3*
auto_setwd, 3

clean_frame, 3
clean_names, *3*
create_dummies, 4
create_flag, 5

dummy_cols, *4*

enc2utf8, *3*

fct_lump_n, *4*, *19*
fill_na, 6
filter_for, 7
filter_missing, 8

import_dir, 9
import_tibble, 9

make_na, *3*
make_na *(*make_na.data.frame*)*, 10
make_na.data.frame, 10

pad_auto, 11
pad_col, 12

recode_chr, 12
relocate_all, 13
remove_empty, *3*
remove_whitespace, 14
rename_with, *3*

select_otherwise, 15
set_chr, 16
set_date, *3*, 16
set_dbl, 17
set_fct, 18
set_int, *3*, 19
set_lgl *(*set_lgl.data.frame*)*, 21
set_lgl.data.frame, 21

ymd, *17*
ymd_hms, *17*