Package 'tbltools'

February 8, 2019

Title Tools for Working with Tibbles

Version 0.1.0

Description Tools for working with tibbles, or data.frame-like objects generated by the 'tibble' package <https://tibble.tidyverse.org/>. Functions assist in converting objects to tibbles and creating frequency tables.

License GPL-3

Encoding UTF-8

LazyData true

Imports magrittr

Suggests testthat, covr

RoxygenNote 6.1.1

URL https://github.com/mkearney/tbltools

BugReports https://github.com/mkearney/tbltools/issues

NeedsCompilation no

Author Michael W. Kearney [aut, cre] (<https://orcid.org/0000-0002-0730-4694>)

Maintainer Michael W. Kearney <kearneymw@missouri.edu>

Repository CRAN

Date/Publication 2019-02-08 17:30:02 UTC

R topics documented:

`arrange_data` *Arrange rows*

Description

Arrange rows via descending or ascending column values

Usage

```
arrange_data(.data, ...)
```

Arguments

`.data`	data frame
`...`	One or more unquoted names of columns on which to arrange the rows. If none are supplied, the data are returned as is. By default, ordering is done in ascending order. To orer in descending order, use `decr` on desired variable(s).

Value

Rearranged data frame

Examples

```
## data frame to arrange
dat <- data.frame(
  a = c(rep("a", 3), rep("b", 3), rep("c", 4)),
  b = c( 3, 3, 2,     8, 8, 1,     5,  5, 5, 9),
  c = c(-1, 0, 0,    -5, 0, 2,    -2, -4, 1, 0),
  stringsAsFactors = FALSE
)

## arrange by one column
arrange_data(dat, a)

## arrange by multiple columns
arrange_data(dat, decr(a), b, c)
```

`as_tbl_data` *as_tbl_data*

Description

Converts data objects to tibbles.

Usage

```
as_tbl_data(x, row_names = FALSE)
```

Arguments

`x`	Data frame or data frame-like input.
`row_names`	Logical indicating whether to convert non-null row names into the first column.

Examples

```
## data with row names
d <- data.frame(x = rnorm(5), y = rnorm(5), row.names = letters[1:5])

## convert to tibble
as_tbl_data(d)

## convert to tibble and create row_names variable
as_tbl_data(d, row_names = TRUE)
```

bind_cols_data *Bind columns*

Description

Bind together the columns of two or more data frames.

Usage

```
bind_cols_data(...)
```

Arguments

...	Data frames

Value

Single data frame with combined columns

bind_rows_data *Bind rows*

Description

Convenient wrapper around `do.call("rbind", ...)` that (a) sets the 'quote' argument to TRUE and (b) fills data frames with missing columns with NAs of the appropriate class.

Usage

```
bind_rows_data(..., fill = FALSE)
```

Arguments

...	Input data frames or list of data frames
fill	Logical indicating whether to fill missing columns in data frames with missing values.

Value

The list collapsed into a single data frame

Examples

```
## list of data frames with inconsistent columns
x <- as_tbl_data(mtcars[1:3, ])
xx <- x
xx$y <- "a"
l <- list(x, xx, mtcars)

## bind rows and fill missing columns with NAs
bind_rows_data(l, fill = TRUE)
```

---

decr *Decreasing order*

Description

Arranges vector in descending order

Usage

```
decr(x)
```

Arguments

x	Input vector

Value

Reordered vector

Examples

```
## decreasing 1:10
decr(1:10)
```

---

env_tbls *Convert all data frames in environment into tibbles*

Description

Converts data frames found in a given environment into tibbles (tbl_df)

Usage

```
env_tbls(env = globalenv(), row_names = TRUE)
```

Arguments

env	Name of environment from which data frames should be converted to tibbles. Defaults to global environment.
row_names	Logical indicating whether to create a row_names variable if non-auto row names are found.

Value

The function will print messages when converting occurs and it will print a final completion message, but otherwise returns nothing.

Examples

```
## data with row names
d <- data.frame(x = rnorm(5), y = rnorm(5), row.names = letters[1:5])

## convert data frames in global environment to tibbles
env_tbls()
```

---

filter_data *Filter rows*

Description

Filter rows via integer/numeric position or logical vector

Usage

```
filter_data(.data, ...)
```

Arguments

.data	Data frame or two dimensional array
...	Each argument/expression should should evaluate and reduce down to an integer (row number) or logical vector. The filter will keep all row numbers that appear in all evaluated expressions (commas are the equivalent to &. Row numbers higher than what exists in x will be ignored. Any numeric vector must be either all positive or all negative (excludes). This function uses non-standard evaluation–users can refer to column names without quotations.

Value

Sliced/filtered data frame

Examples

```
set.seed(12)
d <- data.frame(
  mpg = rnorm(100, 25, 3),
  gear = sample(3:6, 100, replace = TRUE),
  vs = sample(0:1, 100, replace = TRUE),
  stringsAsFactors = FALSE
)

filter_data(d, mpg > 30)
filter_data(d, !mpg < 30)
filter_data(d, mpg > 30, !mpg < 30)
filter_data(d, mpg > 30, gear == 4)
filter_data(d, mpg > 30 | gear == 4, vs == 1)
```

`full_join_data` *Joins*

Description

Full join: join two data frames preserving all possible information

Left join: Join two data frames by matching the second (right) data frame to the left (first) such that the structure of the first (left) data frame is preserved

Right join: Join two data frames by matching the first (left) data frame to the right (second) such that the structure of the second (right) data frame is preserved

Usage

```
full_join_data(x, y, by = NULL)

left_join_data(x, y, by = NULL)

right_join_data(x, y, by = NULL)
```

Arguments

x	Left or first data frame
y	Right or second data frame
by	Name (character) of variable(s) on which to join

Value

Joined (merged) data frame

Examples

```
d1 <- tbl_data(
  x = 1:10,
  y = rnorm(10),
  z = letters[1:10]
)
d2 <- tbl_data(
  x = sample(1:10, 20, replace = TRUE),
  y2 = rnorm(20)
)

## left join
left_join_data(d1, d2)

## right join
right_join_data(d1, d2)

## full join
full_join_data(d1, d2)
```

`group_by_data` *Group data*

Description

Indicate grouping variables in data frame

Usage

```
group_by_data(.data, ...)
```

Arguments

`.data`	Data frame
`...`	Unquoted (non-standard evaluation) name(s) of group variable(s).

Value

A data frame with groups attribute

Examples

```
d <- data.frame(a = c("a", "b", "c"), b = 1:3, stringsAsFactors = FALSE)
group_by_data(d, a)
```

group_by_data_data *Group row numbers in grouped data*

Description

Returns row numbers for each group in grouped data

Usage

```
group_by_data_data(x)
```

Arguments

x	Groupted data frame

Value

List of row numbers for each group

group_by_data_groups *Groups in grouped data*

Description

Returns grouping variable names of grouped data

Usage

```
group_by_data_groups(x)
```

Arguments

x	Grouped data frame

Value

Names of grouping variables

mutate_data *Mutate data*

Description

Wrangle data by adding or transforming columns

Usage

```
mutate_data(.data, ...)
```

Arguments

.data	Data frame
...	One or more expressions using non-standard evaluation designed to return a column of values. These should be named variables. Any names already found in the input data will override those input data frame columns

Value

Mutated data frame

See Also

Other mutate: mutate_if_data

mutate_if_data *Mutate certain columns*

Description

Wrangle only columns that pass a logical test

Usage

```
mutate_if_data(.data, .predicate, .f)
```

Arguments

.data	Input data frame
.predicate	Function applied to each column evaluating to a logical
.f	Function applied to each .predicate-passing column. Can be written in the formula ~ .x format.

Value

Data frame with .predicate-passing columns mutated.

See Also

Other mutate: mutate_data

---

n_obs *Number of observersations*

Description

Returns number of observations in group (if no group then number of obs in data frame)

Usage

```
n_obs()
```

Value

Number of observations in group

---

print.tbl_data *Print tbl_data*

Description

Method for printing pseduo tibble

Usage

```
## S3 method for class 'tbl_data'
print(x, n = NULL, ...)
```

Arguments

x	Input object
n	Maximum number of rows to print, if NULL (default) defaults to getOption("tbltools.print_n", 10)
...	Other args passed to tibble or data.frame print.

pull_data *Pull data*

Description

Pull (extract) data from a data frame

Usage

```
pull_data(.data, ...)
```

Arguments

.data	Input data frame
...	Unquoted name of column to pull from data frame

Value

A column pulled from its data frame (inheriting whatever class the column is)

Examples

```
## pull the 'y' variable from the data frame
tbl_data(x = rnorm(5), y = letters[1:5]) %>% pull_data(y)
```

repos_back *move vars to back*

Description

move vars to back

Usage

```
repos_back(data, ...)
```

Arguments

data	data frame
...	columns to move to back

Value

Reordered data frame.

Examples

```
## data with row names
d <- data.frame(x = rnorm(5), y = rnorm(5), row.names = letters[1:5])

## move x to back
repos_back(d, x)
```

`repos_front` *move vars to front*

Description

move vars to front

Usage

```
repos_front(data, ...)
```

Arguments

`data`	data frame
`...`	columns to move to front

Value

Reordered data frame.

Examples

```
## data with row names
d <- data.frame(x = rnorm(5), y = rnorm(5), row.names = letters[1:5])

## move y to front
repos_front(d, y)
```

select_data *Select columns*

Description

Select columns with non-standard evaluation

Usage

```
select_data(.data, ...)
```

Arguments

.data	Input data frame
...	Unquoted names of columns to select

Value

Data frame with select columns

slice_data *Slice data*

Description

Slice data by row number

Usage

```
slice_data(.data, ...)
```

Arguments

.data	Input data frame
...	Expression to evaluate to integer row positions

Value

.data of evaluated row positions

Examples

```
## data set
d <- tbl_data(x = rnorm(10), y = rnorm(10))

## slice first 4 rows
slice_data(d, 1:4)
```

summarise_data *Summarise data*

Description

Returns a summary-level data frame

Usage

```
summarise_data(.data, ...)
```

Arguments

.data	Data frame
...	One or more namecd expressions evaluating with non-standard evaluation to a single summary value.

Value

A summary-level data frame

tabsort *tabsort*

Description

Returns a sorted (descending) frequency tbl

Usage

```
tabsort(.data, ..., prop = TRUE, na_omit = TRUE, sort = TRUE)

ntbl(.data, ...)
```

Arguments

.data	Data
...	Unquoted column names of variables to include in table. Default is to use all columns.
prop	Logical indicating whether to include a proportion of total obs column.
na_omit	Logical indicating whether to exclude missing. If all responses are missing, a missing value is used as the single category.
sort	Logical indicating whether to sort the returned object.

Value

Frequency tbl

Examples

```
## generate example data
x <- sample(letters[1:4], 200, replace = TRUE)
y <- sample(letters[5:8], 200, replace = TRUE)

## count and sort frequencies for each vector
tabsort(x)
tabsort(y)

## combine x and y into data frame
dat <- data.frame(x, y)

## select columns and create freq table
tabsort(dat, x)
tabsort(dat, x, y)
```

---

tbl_data *tbl data*

Description

Create a tibble data frame

Usage

```
tbl_data(...)
```

Arguments

...	A data frame, vector, or list of values of equal or single-value length–similar to data.frame.

Value

An object of class `c("tbl_data", "tbl_df", "tbl", "data.frame")`

tbl_data_frame *Create tbl_data_frame*

Description

Creates a data frame using non-standard evaluation

Usage

```
tbl_data_frame(...)
```

Arguments

...	Column vectors or expressions that reduce down to desired column vectors of data frame

Value

A tbl_data data frame

Examples

```
## create data frame with two random variables
tbl_data_frame(
  a = rnorm(10),
  b = rnorm(10)
)

## create variables calculated using previous variables
tbl_data_frame(
  a = rnorm(10),
  b = rnorm(10),
  c = (a + b) / 2,
  d = a + b + c
)
```

ungroup_data *Ungroup data*

Description

Ungroups grouped data

Usage

```
ungroup_data(.data)
```

Arguments

`.data`	Grouped data

Value

Data grame without groups attribute

Index