Package 'extdplyr'

February 27, 2017

Type Package

Title Data Manipulation Extensions of 'Dplyr' and 'Tidyr'

Version 0.1.4

Description If 'dplyr' is a grammar for data manipulation, 'extdplyr' is like a short paragraph written in 'dplyr'. 'extdplyr' extends 'dplyr' and 'tidyr' verbs to some common ``routines'' that manipulate data sets. It uses the same interface and preserves all the features from 'dplyr', has good performance, and supports various data sources.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports dplyr, tidyr, lazyeval

RoxygenNote 6.0.1

Suggests testthat, data.table

NeedsCompilation no

Author Yuchen Wang [aut, cre]

Maintainer Yuchen Wang <ycwang0712@gmail.com>

Repository CRAN

Date/Publication 2017-02-27 08:15:28

R topics documented:

check_missing *Check missing rate in variables.*

Description

Check missing (NA) proportion or counts of variables. This function works like summarize_at where the missing rate or count for the selected columns are returned.

Usage

```
check_missing(data, ..., ret_prop = TRUE)

check_missing_(data, ..., .dots, ret_prop = TRUE)
```

Arguments

data	A data.frame or tbl.
...	Comma separated list of unquoted expressions. You can treat variable names like they are positions. Use positive values to select variables; use negative values to drop variables.
ret_prop	Whether to return the rate of missing (default) or counts.
.dots	Used in conjunction with ... to support both explicit and implicit arguments.

Functions

- check_missing_: SE version of check_missing.

Author(s)

Min Ma

grp_routine *Mutate a character/factor based on conditions.*

Description

grp_routine functions like a serious of nested ifelse where a series of conditions are evaluated and different values are assigned based on those conditions.

Usage

```
grp_routine(data, col, ..., ret_factor = FALSE)

grp_routine_(data, col, ..., .dots, ret_factor = FALSE)
```

Arguments

data	A data.frame or tbl.
col	Name of the generated column. Use a bare name when using NSE functions and a character (quoted) name when using SE functions (functions that end with underscores).
...	Specification of group assignment. Use named conditions, like top2 = x > 5.
ret_factor	Whether to convert the column into factor.
.dots	Used in conjunction with ... to support both explicit and implicit arguments.

Functions

- grp_routine_: SE version of grp_routine.

Examples

```
df <- data.frame(v1 = letters[1:5], v2 = 1:5)
df

# By default, it creates new groups
grp_routine(df, "group",
               first = v1 %in% c("a", "b"),
               second = v2 == 3,
               third = v2 >= 4)

# Gives a warning when the groups are not collectively exhaustive
grp_routine(df, "group",
               first = v1 %in% c("a", "b"),
               second = v2 == 3,
               third = v2 > 4)


# SE version
grp_routine_(df, "group",
               "first" = ~ v1 %in% c("a", "b"),
               "second" = ~ v2 == 3,
               .dots = setNames(list(~ v2 > 4), "third"))
```

---

ind_to_char *Convert indicator data.frame to character/factor.*

Description

This is the reverse operation of using model.matrix a factor. ind_to_char works like dplyr::unite, it combines multiple indicator columns into one character/factor column and add it to the data.

Usage

```
ind_to_char(data, col, ..., ret_factor = FALSE, remove = TRUE,
  mutually_exclusive = TRUE, collectively_exhaustive = TRUE)

ind_to_char_(data, col, from, ret_factor = FALSE, remove = TRUE,
  mutually_exclusive = TRUE, collectively_exhaustive = TRUE)
```

Arguments

data	A data.frame or tbl.
col	Name of the generated column. Use a bare name when using NSE functions and a character (quoted) name when using SE functions (functions that end with underscores).
...	Specification of indicator columns. Use bare variable names. Select all variables between x and z with x:z. For more options, see the select documentation.
ret_factor	Whether to convert the column into factor.
remove	If TRUE, remove input columns from output data frame.
mutually_exclusive	Check if the indicators are mutually exclusive.
collectively_exhaustive	Check if the indicators are collectively exhaustive.
from	Names of existing columns as character vector

Functions

- ind_to_char_: SE version of ind_to_char.

Examples

```
# Supports converting the following atomic types to indicator

df <- data.frame(integer_ind = c(2L, 0L, 0L, 0L, 0L, 0L),
                 # non-zero integer is 1, otherwise 0.
                 logcal_ind = c(FALSE, TRUE, FALSE, FALSE, FALSE, FALSE),
                 # TRUE is 1.
                 double_ind = c(0, 0, 2.0, 0, 0, 0),
                 # non-zero double is 1.
                 char_ind = c("FALSE", "FALSE", "F", "TRUE", "T", "FALSE"),
                 # "T" and "TRUE" converts to 1.
                 factor_ind = factor(c(1, 1, 1, 1, 1, 0), levels = c(0, 1),
                                     labels = c(TRUE, FALSE)),
                 # Factors are converted based on levels.
                 stringsAsFactors = FALSE)

ind_to_char_(df, col = "new_y", from = names(df), remove = FALSE)


# ind_to_char as complement to use model.matrix on a factor
```

```
df <- data.frame(x = 1:6, y = factor(c(letters[1:5], NA)))
ind_df <- as.data.frame(model.matrix(~ x + y - 1,
                                     model.frame(df, na.action = na.pass)))
ind_df  # an indicator matrix with NAs

# New character column is generated with non-selected columns kept as is.
ind_to_char(ind_df, new_y, ya:ye)
ind_to_char(ind_df, new_y, -x)
ind_to_char(ind_df, col = new_y, ya:ye, remove = FALSE)
# Returns a factor column
ind_to_char(ind_df, col = new_y, ya:ye, ret_factor = TRUE)

# Using SE
ind_to_char_(ind_df, col = "new_y", from = c("ya", "yb", "yc", "yd", "ye"))
```

`pct_routine` *Calculate percentage by group.*

Description

`pct_routine` works like `count` except that it returns group percentages instead of counts. `tally_pct` is a underlying utility function that corresponds to `tally`. As the name implies, it also returns percentage.

Usage

```
pct_routine(data, ..., wt = NULL, ret_name = "pct", rebase = FALSE,
  ungroup = FALSE)

pct_routine_(data, vars, wt = NULL, ret_name = "pct", rebase = FALSE,
  ungroup = FALSE)

tally_pct(data, wt = NULL, ret_name = "pct", rebase = FALSE)

tally_pct_(data, wt = NULL, ret_name = "pct", rebase = FALSE)
```

Arguments

`data`	A `data.frame` or `tbl`.
`...`	Variables to group by, see `group_by`.
`wt`	Column name of weights.
`ret_name`	Character of the variable name returned.
`rebase`	Whether to remove the missing values in the percentage, e.g. rebase the percentage so that NAs in the last group are excluded.
`ungroup`	Whether to ungroup the returned table.
`vars`	A character vector of variable names to group by.

Functions

- pct_routine_: SE version of pct_routine.
- tally_pct: NSE version of tally_pct_.
- tally_pct_: Underlying SE function of pct_routine_ without options for groups.

Examples

```
data(esoph)
esoph
pct_routine(esoph, agegp, alcgp)
pct_routine(esoph, agegp, alcgp, wt = ncases)
# Crate new grouping variables
pct_routine(esoph, agegp, low_alcgp = alcgp %in% c("0-39g/day", "40-79"))


# This examples shows how rebase works
if (require(dplyr)) {
  iris %>%
    mutate(random_missing = ifelse(rnorm(n()) > 0, NA, round(Sepal.Length))) %>%
    group_by(Species, random_missing) %>%
    tally_pct(wt = Sepal.Width, rebase = TRUE)
}
```

Index