Package ‘extdplyr’

April 20, 2020

Type Package

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

Version 0.1.5

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 7.1.0

Suggests testthat, data.table

NeedsCompilation no

Author Yuchen Wang [aut, cre]

Maintainer Yuchen Wang <ycwang0712@gmail.com>

Repository CRAN

Date/Publication 2020-04-20 05:20:02 UTC

R topics documented:

check_missing .................................................. 2
grp_routine ....................................................... 2
ind_to_char ....................................................... 3
pct_routine ....................................................... 5

Index 7
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**

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

**Arguments**

- `data`: A `data.frame` or `tbl`.
- `...`: Pass to `tidyselect`. See [dplyr::dplyr_tidy_select] for details.
- `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**

```r
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**

```r
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 `tidyr::unite`, it combines multiple indicator columns into one character/factor column and add it to the data.
Usage

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

```r
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 column 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.)
pct_routine

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.
...
vars 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.
ungs 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

check_missing, 2
check_missing_(check_missing), 2
count, 5

data.frame, 2–4, 6

group_by, 6
grp_routine, 2
grp_routine_(grp_routine), 2

ind_to_char, 3
ind_to_char_(ind_to_char), 3

model.matrix, 3

pct_routine, 5
pct_routine_(pct_routine), 5

select, 4
summarize_at, 2

tally_pct(pct_routine), 5
tally_pct_(pct_routine), 5
tbl, 2–4, 6