Dplyr Make New Variables (Columns)

Victoire Migashane

9/6/2022

```
head(df)
```

```
## # A tibble: 6 x 8
##   species island    bill_length_mm bill_depth_mm flipper_l~1 body_~2 sex    year
##   <fct>   <fct>              <dbl>         <dbl>       <int>   <int> <fct> <int>
## 1 Adelie  Torgersen           39.1          18.7         181    3750 male   2007
## 2 Adelie  Torgersen           39.5          17.4         186    3800 fema~  2007
## 3 Adelie  Torgersen           40.3          18           195    3250 fema~  2007
## 4 Adelie  Torgersen           NA            NA            NA      NA <NA>   2007
## 5 Adelie  Torgersen           36.7          19.3         193    3450 fema~  2007
## 6 Adelie  Torgersen           39.3          20.6         190    3650 male   2007
## # ... with abbreviated variable names 1: flipper_length_mm, 2: body_mass_g
```

1. Rename function

The `rename()` function changes the names of individual columns using the `new_name = old_name`. A similar function is `rename_with` which renames the column using a function.

Arguments

```
.data: A data frame to apply changes to
...: rename selected volumns ‘new = old‘
.fn: transforms the selected columns to return a charcture vector of same length
.cols: columns to rename (deafult is ALL the columns)
```

```
df %>%
    rename(gender = sex, col = c("bill_length_mm", "bill_depth_mm", "flipper_length_mm")) %>%
    head(3)
```

Usage

```
## # A tibble: 3 x 8
##   species island     col1  col2  col3 body_mass_g gender  year
##   <fct>   <fct>     <dbl> <dbl> <int>       <int> <fct>  <int>
## 1 Adelie  Torgersen  39.1  18.7   181        3750 male    2007
## 2 Adelie  Torgersen  39.5  17.4   186        3800 female  2007
## 3 Adelie  Torgersen  40.3  18     195        3250 female  2007
```

2. Mutate and Transmute Functions

The `mutate()` function is used to acompute and append new columns to an existing data frame.

The `transmute()` function does the opposite of `mutate()`. This function add new variables and drops (remove) existing ones.

Arguments

```
.data: A data frame
...: The name given to the new column
.keep: control which columns to retain in the out put. ‘.keep = c("all", "used", "unused", "none")‘
.before/.after: control where new columns should appear
```

```
df %>%
    mutate(bill_and_flipper_len = bill_length_mm + flipper_length_mm, .keep = "used") %>%
    head(3)
```

Usage

```
## # A tibble: 3 x 3
##   bill_length_mm flipper_length_mm bill_and_flipper_len
##            <dbl>             <int>                <dbl>
## 1           39.1               181                 220.
## 2           39.5               186                 226.
## 3           40.3               195                 235.
```

```
df %>%
    transmute(species, island, bill_and_flipper_len = bill_length_mm + flipper_length_mm) %>%
    head(3)
```

```
## # A tibble: 3 x 3
##   species island    bill_and_flipper_len
##   <fct>   <fct>                    <dbl>
## 1 Adelie  Torgersen                 220.
## 2 Adelie  Torgersen                 226.
## 3 Adelie  Torgersen                 235.
```

3. Mutate window functions

The `muate()` function uses uses window functions. These are functions that take vector of values and return another vector of values with same lenght.

Offset

```
‘lag()‘: offset elements by 1
‘lead(): offset elements by -1
```

Cumulative aggregate

```
‘cumall()‘: cumulative all
‘cumany()‘: cumulative any
‘cummax()‘: cumulative max
‘cummin()‘: cumulative min
‘cummean()‘: cumulative mean
‘cumprod()‘: cumulative prod
‘cumsum()‘: cumulative sum
```

Ranking

```
‘cume_dist()‘: proportion of all values <=
‘dense_rank()‘: randk w ties  = min, no gapes
‘min_rank()‘: randk with ties = min
‘ntile()‘: bins into n bins
‘percent_rank()‘: min_rank scaled to [0,1]
‘row_number()‘: rank with ties =  "first
```

Math

```
arithmetic ops: ‘+, -, *, /, ^, %/%, %%‘
logs: ‘log(), log2(), log10()‘
logical comparizons: ‘<, >, <=, >=, ==, !=‘
x >= left <= right: ‘between()‘
sage == for floating point numbers: ‘near()‘
```

Miscellaneous

```
‘case_when()‘: multi_case if_else()
‘coalsce()‘: first non-NA values by element across a set of vector
‘if_else()‘: element-wise if() + else()
‘na_if‘: replace specific values with NA
‘pmax()‘: element wise max()
‘pmin()‘: element wise min()
```