dplyr in R

Siyu Shen

2022-11-11

Introduction to dplyr package in R

The **dplyr** package is mainly used for cleaning and sorting data. By using **dplyr** package, we can efficiently process the data, which is very important for people to do data analysis. In this note, I am going to introduce the most common used functions in **dplyr**.

```
#load 'dplyr' package
library(dplyr)
```

```
#load data 'mtcars' as example
data(mtcars)

#show the structure of the data 'mtcars'
str(mtcars)
```

```
## 'data.frame':    32 obs. of  11 variables:
##  $ mpg : num  21 21 22.8 21.4 18.7 18.1 14.3 24.4 22.8 19.2 ...
##  $ cyl : num  6 6 4 6 8 6 8 4 4 6 ...
##  $ disp: num  160 160 108 258 360 ...
##  $ hp  : num  110 110 93 110 175 105 245 62 95 123 ...
##  $ drat: num  3.9 3.9 3.85 3.08 3.15 2.76 3.21 3.69 3.92 3.92 ...
##  $ wt  : num  2.62 2.88 2.32 3.21 3.44 ...
##  $ qsec: num  16.5 17 18.6 19.4 17 ...
##  $ vs  : num  0 0 1 1 0 1 0 1 1 1 ...
##  $ am  : num  1 1 1 0 0 0 0 0 0 0 ...
##  $ gear: num  4 4 4 3 3 3 3 4 4 4 ...
##  $ carb: num  4 4 1 1 2 1 4 2 2 4 ...
```

There are 32 observations on 11 variables. The meanings of each variables are:
mpg: Miles/(US) gallon,
cyl: Number of cylinders,
disp: Displacement (cu.in.),
hp: Gross horsepower,
drat: Rear axle ratio,
wt: Weight (1000 lbs),
qsec: 1/4 mile time,
vs: Engine (0 = V-shaped, 1 = straight),
am: Transmission (0 = automatic, 1 = manual),
gear: Number of forward gears,
carb: Number of carburetors.

1. Pipe operator '%>%'

It is important to know the pipe %>% operator before we start to learn the functions in **dplyr** package. With the use of %>% , multiple functions can be wrapper together. And it can be used with any function.

```
# usage in filter() function
filter(data_frame, variable == value)
data_frame %>% filter(variable == value)

# usage in mutate() function
mutate(data_frame, expression(s))
data_frame %>% mutate(expression(s))
```

2. Filter by row: filter() function

You can use **filter()** function to filter the subsets by the given logic, which is similar to the **subset()** function, for example:

```
#filter when mpg is greater than or equal to 22
filter(mtcars, mpg >= 22)
```

```
##                 mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Datsun 710     22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Merc 240D      24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230       22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Fiat 128       32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic    30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Fiat X1-9      27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
## Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
```

```
#filter when cyl=4 or gear=3
filter(mtcars, cyl == 4 | gear == 3)

#filter when cyl=4 and gear=3
filter(mtcars, cyl == 4 & gear == 3)

#note: when using AND operation, avoid using '&&' instead of '&'
```

3. Filter by column: select() function

select() function selects subdatasets with column names as arguments. The **dplyr** package provides some special functions to be used in conjunction with the **select()** function to filter variables, including **starts_with**, **ends_with**, **contains**, **matches**, **one_of**, **num_range**, and **everything**.

```
#choose data 'iris' as example
data(iris)

iris = tbl_df(iris)
```

```r
#show the 'iris' data
head(iris)
```

```
## # A tibble: 6 x 5
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
##          <dbl>       <dbl>        <dbl>       <dbl> <fct>
## 1          5.1         3.5          1.4         0.2 setosa
## 2          4.9         3            1.4         0.2 setosa
## 3          4.7         3.2          1.3         0.2 setosa
## 4          4.6         3.1          1.5         0.2 setosa
## 5          5           3.6          1.4         0.2 setosa
## 6          5.4         3.9          1.7         0.4 setosa
```

```r
#select the columns that start with 'Petal'
select(iris, starts_with("Petal"))
```

```
## # A tibble: 150 x 2
##    Petal.Length Petal.Width
##           <dbl>       <dbl>
##  1          1.4         0.2
##  2          1.4         0.2
##  3          1.3         0.2
##  4          1.5         0.2
##  5          1.4         0.2
##  6          1.7         0.4
##  7          1.4         0.3
##  8          1.5         0.2
##  9          1.4         0.2
## 10          1.5         0.1
## # ... with 140 more rows
```

```r
#select the columns that not start with 'Petal'
select(iris, -starts_with("Petal"))
select(iris, !starts_with("Petal"))

#select the columns that end with 'Petal'
select(iris, ends_with("Width"))

#select the columns that contain with 'etal'
select(iris, contains("etal"))

#select the columns that the variables name is with 't'
select(iris, matches(".t."))

#select the columns you want directly
select(iris, Petal.Length, Petal.Width)

#select multi-columns by using colon
select(iris, Sepal.Length:Petal.Width)
```

```r
#when we cannot use the character vector filter, we need to use 'one_of()' function
vars <- c("Petal.Length", "Petal.Width")
select(iris, one_of(vars))
```

```r
#return all columns, generally used when adjusting the order of variables in a dataset
select(iris, everything())

#return all columns, but reorder the columns and put 'Species' column in the front
select(iris, Species, everything())
```

4. mutate() and transmute()

mutate() adds new variables and preserves existing ones; **transmute()** adds new variables and drops existing ones.

4.1. mutate()

4.1.1. New columns

```r
#create two more columns named 'cyl2' and 'cyl4'
head(mtcars %>% mutate(cyl2 = cyl * 2, cyl4 = cyl2 * 2))
```

```
##                   mpg cyl disp  hp drat    wt  qsec vs am gear carb cyl2 cyl4
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4   12   24
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4   12   24
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1    8   16
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1   12   24
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2   16   32
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1   12   24
```

4.1.2. Delete columns

```r
#delete the column 'mpg' and update the column 'disp'
mtcars %>% mutate(mpg = NULL, disp = disp * 0.0163871)

#delete the column 'cyl'
mtcars %>% mutate(cyl = NULL)
```

4.1.3. Application of Window function

```r
#create a new column 'rank' using min_rank() function by group 'cyl'
head(mtcars %>% group_by(cyl) %>% mutate(rank = min_rank(desc(mpg))))
```

```
## # A tibble: 6 x 12
## # Groups:   cyl [3]
##     mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb  rank
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <int>
## 1  21       6   160   110  3.9   2.62  16.5     0     1     4     4     2
## 2  21       6   160   110  3.9   2.88  17.0     0     1     4     4     2
## 3  22.8     4   108    93  3.85  2.32  18.6     1     1     4     1     8
## 4  21.4     6   258   110  3.08  3.22  19.4     1     0     3     1     1
## 5  18.7     8   360   175  3.15  3.44  17.0     0     0     3     2     2
## 6  18.1     6   225   105  2.76  3.46  20.2     1     0     3     1     6
```

```
#create a new column 'mpg_max' using max() function by group 'cyl'
head(mtcars %>% group_by(cyl) %>% mutate(mpg_max = max(mpg)))
```

```
## # A tibble: 6 x 12
## # Groups:   cyl [3]
##     mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb mpg_max
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1  21       6   160   110  3.9   2.62  16.5     0     1     4     4    21.4
## 2  21       6   160   110  3.9   2.88  17.0     0     1     4     4    21.4
## 3  22.8     4   108    93  3.85  2.32  18.6     1     1     4     1    33.9
## 4  21.4     6   258   110  3.08  3.22  19.4     1     0     3     1    21.4
## 5  18.7     8   360   175  3.15  3.44  17.0     0     0     3     2    19.2
## 6  18.1     6   225   105  2.76  3.46  20.2     1     0     3     1    21.4
```

4.2. transmute()

The return value does not contain the original dataset variables, only the variables after calculation and transformation are retained.

```
head(mtcars %>% mutate(wt_log=log(wt)))
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb    wt_log
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4 0.9631743
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4 1.0560527
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1 0.8415672
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1 1.1678274
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2 1.2354715
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1 1.2412686
```

```
head(mtcars %>% transmute(wt_log=log(wt)))
```

```
##                     wt_log
## Mazda RX4         0.9631743
## Mazda RX4 Wag     1.0560527
## Datsun 710        0.8415672
## Hornet 4 Drive    1.1678274
## Hornet Sportabout 1.2354715
## Valiant           1.2412686
```

```
head(mtcars %>% mutate(displ_l = disp / 61.0237))
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb  displ_l
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4 2.621932
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4 2.621932
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1 1.769804
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1 4.227866
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2 5.899347
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1 3.687092
```

```
head(mtcars %>% transmute(displ_l = disp / 61.0237))
```

```
##                    displ_l
## Mazda RX4         2.621932
## Mazda RX4 Wag     2.621932
## Datsun 710        1.769804
## Hornet 4 Drive    4.227866
## Hornet Sportabout 5.899347
## Valiant           3.687092
```

5. Ranking function

row_number: the results of the parallel rankings are in different order, and the elements appearing first are ranked first.

min_rank: the results of the tie ranks are the same, and the next rank will be occupied.

dense_rank: tied ranking does not occupy the ranking, for example: no matter how many tied for the second place, the subsequent ranking should still be the third place

percent_rank: rank by percentage

cume_dist: rank of cumulative distribution interval

ntile: roughly ranks vectors by dividing into n buckets. Larger buckets have lower rank.

```
x = c(5, 1, 3, 2, 2, NA)

row_number(x)
```

```
## [1]  5  1  4  2  3 NA
```

```
min_rank(x)
```

```
## [1]  5  1  4  2  2 NA
```

```
dense_rank(x)
```

```
## [1]  4  1  3  2  2 NA
```

```
percent_rank(x)
```

```
## [1] 1.00 0.00 0.75 0.25 0.25   NA
```

```
cume_dist(x)
```

```
## [1] 1.0 0.2 0.8 0.6 0.6  NA
```

```
ntile(x, 2)
```

```
## [1]  2  1  2  1  1 NA
```

```
head(mtcars%>%mutate(dense_rank=cume_dist(cyl)))
```

```
##                    mpg cyl disp  hp drat    wt  qsec vs am gear carb dense_rank
## Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4    0.56250
## Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4    0.56250
## Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1    0.34375
## Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1    0.56250
## Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2    1.00000
## Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1    0.56250
```

6. Sort function: arrange()

Note the difference between sorting and ranking.

arrange() sorts the rows sequentially by the given column names.

```
#sorted by column 'mpg'
head(arrange(mtcars, mpg))
```

```
##                     mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Cadillac Fleetwood  10.4   8  472 205 2.93 5.250 17.98  0  0    3    4
## Lincoln Continental 10.4   8  460 215 3.00 5.424 17.82  0  0    3    4
## Camaro Z28          13.3   8  350 245 3.73 3.840 15.41  0  0    3    4
## Duster 360          14.3   8  360 245 3.21 3.570 15.84  0  0    3    4
## Chrysler Imperial   14.7   8  440 230 3.23 5.345 17.42  0  0    3    4
## Maserati Bora       15.0   8  301 335 3.54 3.570 14.60  0  1    5    8
```

```
#sorted by column 'mpg' and 'disp'
head(arrange(mtcars, mpg, disp))
```

```
##                     mpg cyl disp  hp drat    wt  qsec vs am gear carb
## Lincoln Continental 10.4   8  460 215 3.00 5.424 17.82  0  0    3    4
## Cadillac Fleetwood  10.4   8  472 205 2.93 5.250 17.98  0  0    3    4
## Camaro Z28          13.3   8  350 245 3.73 3.840 15.41  0  0    3    4
## Duster 360          14.3   8  360 245 3.21 3.570 15.84  0  0    3    4
## Chrysler Imperial   14.7   8  440 230 3.23 5.345 17.42  0  0    3    4
## Maserati Bora       15.0   8  301 335 3.54 3.570 14.60  0  1    5    8
```

```
#reverse order by using desc()
head(arrange(mtcars, desc(mpg)))
```

```
##                mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Fiat 128       32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic    30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Fiat X1-9      27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
```

```r
#reverse order by using '-'
head(arrange(mtcars, -mpg))
```

```
##                 mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Toyota Corolla 33.9   4  71.1  65 4.22 1.835 19.90  1  1    4    1
## Fiat 128       32.4   4  78.7  66 4.08 2.200 19.47  1  1    4    1
## Honda Civic    30.4   4  75.7  52 4.93 1.615 18.52  1  1    4    2
## Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.90  1  1    5    2
## Fiat X1-9      27.3   4  79.0  66 4.08 1.935 18.90  1  1    4    1
## Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.70  0  1    5    2
```