Column-wise and Row-wise Operations in `dplyr`

Yingqi Jing

July 22, 2025

Contents

List of Figures

List of Tables

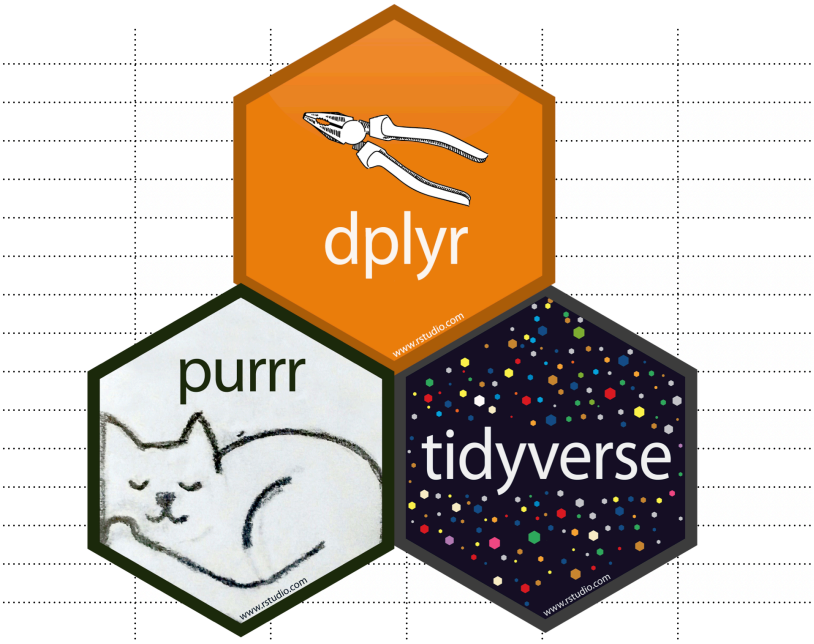

Figure 1: row-wise and column-wise operations

With the development of **dplyr** or its umbrella package **tidyverse**, it becomes quite easy to perform operations over columns or rows in R. These column- or row-wise methods can also be directly integrated with other dplyr verbs like `select`, `mutate`, `filter` and `summarise`, making them more comparable with other functions in `apply` or `map` families. In this blog, I will briefly cover some useful column- or row-wise operations.

1 Column-wise operation

Example 1: select those string columns with less than 5 levels in the dataset of **starwars**.

```
starwars %>%
  select_if(~ any(is.character(.x) & length(unique(.x)) <= 5)) %>%
  head()
```

```
# A tibble: 6 x 2
  sex    gender
  <chr>  <chr>
1 male   masculine
2 none   masculine
```

```
3 none   masculine
4 male   masculine
5 female feminine
6 male   masculine
```

We can combine `select_if` and `any` to identify certain columns by certain criterion. **Note:** we are using tilde (~) to define an anonymous function, and thus we should use `.x` to refer to the selected columns. See this link for detailed illustration of tilde (~), dot (.), and dot x (.x) in dplyr.

If you want to calculate the levels of those selected columns, you can try `across` function and `summarise` the number of levels by column.

```
starwars %>%
  summarise(across(where(is.character), ~ length(unique(.x))))
```

```
# A tibble: 1 x 8
   name hair_color skin_color eye_color   sex gender homeworld species
  <int>      <int>      <int>     <int> <int>  <int>     <int>   <int>
1    87         12         31        15     5      3        49      38
```

Alternatively, you can make use of the `map` or `map_dbl` function in **purrr** by the following command. Note that when a `map` function is applied to a data.frame, it will operate over columns by default.

```
# map_dbl returns a double vector, while map returns a list
starwars %>%
  select_if(~ is.character(.x)) %>%
  map_dbl(~length(unique(.x))) %>%
  head()
```

```
      name hair_color skin_color  eye_color        sex     gender
        87         12         31         15          5          3
```

Example 2: select those numeric columns and calculate the means and sds across columns in the dataset of **starwars**.

```
starwars %>%
  summarise(across(where(~ is.numeric(.x)),
                   list(Mean = ~ mean(.x, na.rm = TRUE),
                        Sd = ~ sd(.x, na.rm = TRUE))))
```

```
# A tibble: 1 x 6
  height_Mean height_Sd mass_Mean mass_Sd birth_year_Mean birth_year_Sd
        <dbl>     <dbl>     <dbl>   <dbl>           <dbl>         <dbl>
1        175.      34.8      97.3    169.            87.6          155.
```

This example provides us a good illustration of the use of `.x` in **dplyr** style syntax, since we have some missing values (NAs) in certain columns. Thus, we need to specify the parameter with `na.rm = TRUE` inside the functions.

There is indeed a more convenient and elegant way of solving this by using the function `summarise_if`. It allows us to select certain columns and operate by columns like this:

```
starwars %>%
  summarise_if(is.numeric,
               list(Sum = sum, Mean = mean, Sd = sd),
               na.rm = TRUE)
```

```
# A tibble: 1 x 9
  height_Sum mass_Sum birth_year_Sum height_Mean mass_Mean birth_year_Mean height_Sd mass_Sd
       <int>    <dbl>          <dbl>       <dbl>     <dbl>           <dbl>     <dbl>   <dbl>
1      14143    5741.          3765.        175.      97.3            87.6      34.8    169.
```

2 Row-wise operation

Example 3: calculate the **sums**, **means** and **sds** for each row for the dataset of **iris**.

```
iris %>%
  rowwise() %>%
  mutate(
    Rowsum = sum(c_across(Sepal.Length:Petal.Width)),
    Rowsd = sd(c_across(Sepal.Length:Petal.Width)),
    Rowmean = mean(c_across(Sepal.Length:Petal.Width))
  ) %>%
  ungroup() %>%
  head()
```

```
# A tibble: 6 x 8
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species Rowsum Rowsd Rowmean
         <dbl>       <dbl>        <dbl>       <dbl> <fct>    <dbl> <dbl>   <dbl>
1          5.1         3.5          1.4         0.2 setosa    10.2  2.18    2.55
```

```
2          4.9         3           1.4         0.2 setosa     9.5  2.04    2.38
3          4.7         3.2         1.3         0.2 setosa     9.4  2.00    2.35
4          4.6         3.1         1.5         0.2 setosa     9.4  1.91    2.35
5          5           3.6         1.4         0.2 setosa    10.2  2.16    2.55
6          5.4         3.9         1.7         0.4 setosa    11.4  2.23    2.85
```

Here the function `c_across` is specifically designed to work with `rowwise` operations. **Note:** `rowwise` groups your data by row (class: *rowwise_df*), and it is best to `ungroup` immediately. Of course, if you are more comfortable with the `apply` function, you can also use the following command:

```
iris %>%
  select(Sepal.Length:Petal.Width) %>%
  apply(., 1, function(x) c(sum(x), sd(x), mean(x))) %>%
  as_tibble() %>%
  t() %>%
  head()
```

```
   [,1]     [,2]  [,3]
V1 10.2 2.179449 2.550
V2  9.5 2.036950 2.375
V3  9.4 1.997498 2.350
V4  9.4 1.912241 2.350
V5 10.2 2.156386 2.550
V6 11.4 2.230844 2.850
```

```
iris %>%
  rowwise() %>%
  dplyr::mutate(
    Rowsum = sum(c_across(Sepal.Length:Petal.Width)),
    Rowmean = mean(c_across(Sepal.Length:Petal.Width)),
    Rowsd = sd(c_across(Sepal.Length:Petal.Width)),
    .before = "Species"
  ) %>%
  ungroup() %>%
  head()
```

```
# A tibble: 6 x 8
  Sepal.Length Sepal.Width Petal.Length Petal.Width Rowsum Rowmean Rowsd Species
         <dbl>       <dbl>        <dbl>       <dbl>  <dbl>   <dbl> <dbl> <fct>
```

```
1          5.1         3.5          1.4         0.2   10.2    2.55  2.18 setosa
2          4.9         3            1.4         0.2    9.5    2.38  2.04 setosa
3          4.7         3.2          1.3         0.2    9.4    2.35  2.00 setosa
4          4.6         3.1          1.5         0.2    9.4    2.35  1.91 setosa
5          5           3.6          1.4         0.2   10.2    2.55  2.16 setosa
6          5.4         3.9          1.7         0.4   11.4    2.85  2.23 setosa
```

```
iris %>%
  as_tibble() %>%
  dplyr::mutate(
    row = pmap(across(1:4), ~ {
      list(rsum = sum, rmean = mean) %>%
        map_dfc(function(f) f(c(...)))
    }),
    .before = "Sepal.Length"
  ) %>%
  unnest(row) %>%
  head()
```

```
# A tibble: 6 x 7
   rsum rmean Sepal.Length Sepal.Width Petal.Length Petal.Width Species
  <dbl> <dbl>        <dbl>       <dbl>        <dbl>       <dbl> <fct>
1  10.2  2.55          5.1         3.5          1.4         0.2 setosa
2   9.5  2.38          4.9         3            1.4         0.2 setosa
3   9.4  2.35          4.7         3.2          1.3         0.2 setosa
4   9.4  2.35          4.6         3.1          1.5         0.2 setosa
5  10.2  2.55          5           3.6          1.4         0.2 setosa
6  11.4  2.85          5.4         3.9          1.7         0.4 setosa
```

```
iris %>%
  as_tibble() %>%
  dplyr::mutate(
    row = pmap(
      across(Sepal.Length:Petal.Width),
      ~ bind_cols(
        rsum = sum(c(...)),
        rmean = mean(c(...)),
        rsd = sd(c(...))
      )
    ),
    .before = "Sepal.Length"
```

```
  ) %>%
  unnest(row) %>%
  head()
```

```
# A tibble: 6 x 8
   rsum rmean   rsd Sepal.Length Sepal.Width Petal.Length Petal.Width Species
  <dbl> <dbl> <dbl>        <dbl>       <dbl>        <dbl>       <dbl> <fct>
1  10.2  2.55  2.18          5.1         3.5          1.4         0.2 setosa
2   9.5  2.38  2.04          4.9         3            1.4         0.2 setosa
3   9.4  2.35  2.00          4.7         3.2          1.3         0.2 setosa
4   9.4  2.35  1.91          4.6         3.1          1.5         0.2 setosa
5  10.2  2.55  2.16          5           3.6          1.4         0.2 setosa
6  11.4  2.85  2.23          5.4         3.9          1.7         0.4 setosa
```

3 Related links:

- https://dplyr.tidyverse.org/articles/rowwise.html
- https://purrr.tidyverse.org/reference/map.html