Data transformation

Luis Francisco Gomez Lopez

2023-08-10

Contents

dplyr basics

- **Name origin**:
  - **d**: dataframe
  - **plyr**: plier (pinza)
    - Check the package hex sticker
    - plyr has the first package where it applies a split-apply-combine strategy (Wickham, 2011)

The d is for dataframes, the plyr is to evoke pliers. Pronounce however you like —- Hadley Alexander Wickham

- **Verbs based on what they operate on**
  - Rows
  - Columns
  - Groups
  - Data frames (Wickham et al., 2023, Chapter 20)

`dplyr` basics

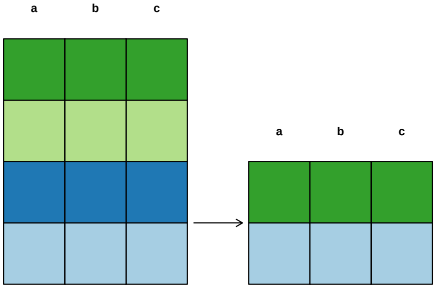
- **Common elements between verbs**
    - The first argument is always a data frame
    - The subsequent arguments typically describe which columns to operate on, using the variable names (without quotes)
    - The output is always a new data frame

`dplyr` basics

- **Rows**
  - `dplyr::filter`
  - `dplyr::arrange`
  - `dplyr::distinct`
- **Columns**
  - `dplyr::mutate`
  - `dplyr::select`
  - `dplyr::rename`
  - `dplyr::relocate`
- **Groups**
  - `dplyr::group_by`
  - `dplyr::summarise`
  - `dplyr::ungroup`

dplyr basics

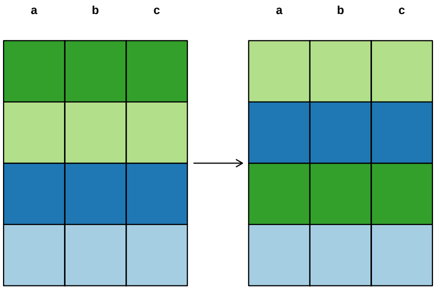
- **Rows**
  - `dplyr::filter`



```
filter(data = <DATA>, <EXPRESSION RETURNING A LOGICAL VALUE>)
```

dplyr basics

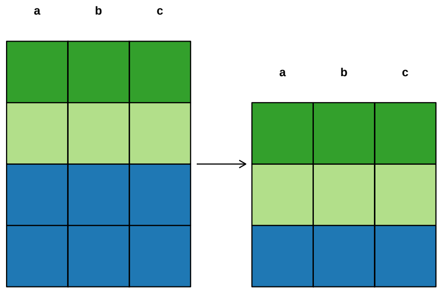
- **Rows**
  - `dplyr::arrange`



```
arrange(data = <DATA>, <VARIABLES OR FUNCTIONS APPLIED TO VARIABLES>)
```

dplyr basics

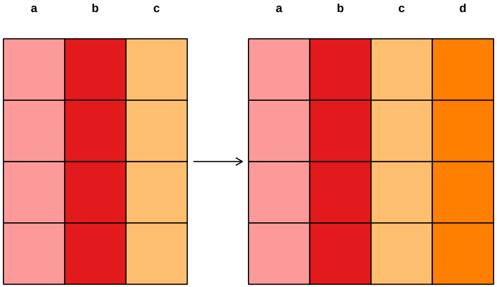
- **Rows**
  - `dplyr::distinct`



```
distinct(data = <DATA>, <VARIABLES>)
```

dplyr basics

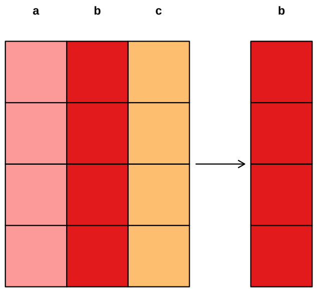
- **Columns**
  - `dplyr::mutate`



```
mutate(data = <DATA>, <ORDERED PAIR OF NAME AND VALUE>)
```

dplyr basics

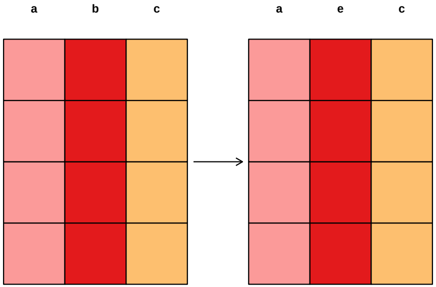
- **Columns**
  - `dplyr::select`



```
select(data = <DATA>, <VARIABLES OR EXPRESSIONS (WITHOUT QUOTATION MARKS)>)
```

dplyr basics

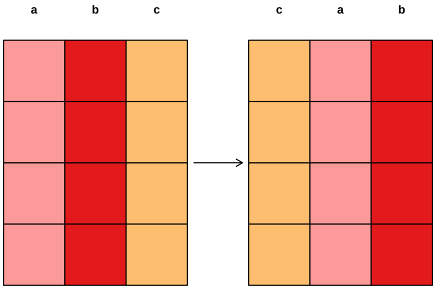
- **Columns**
  - `dplyr::rename`



```
rename(data = <DATA>, <ORDERED PAIR OF NEW NAME AND OLD NAME>)
```

dplyr basics

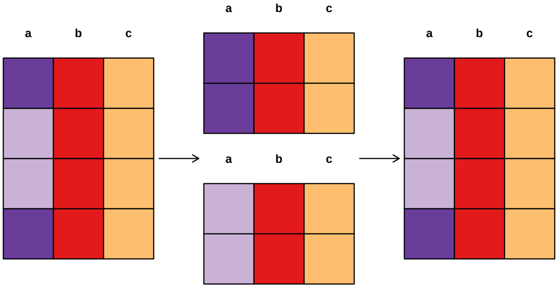
- **Columns**
  - `dplyr::relocate`



```
relocate(data = <DATA>, <VARIABLES OR FUNCTIONS APPLIED TO VARIABLES>)
```

dplyr basics

- **Groups**
  - dplyr::group_by and dplyr::ungroup_by

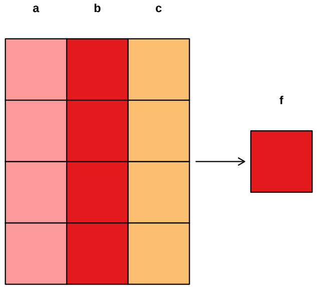


```r
group_by(data = <DATA>, <VARIABLES>)
```

```r
ungroup(data = <DATA>, <VARIABLES>)
```

dplyr basics

- **Groups**
  - `dplyr::summarise`



```
summarise(data = <DATA>, <ORDERED PAIR OF NAME AND FUNCTION APPLIED TO A VARIABLE>)
```

The pipe

- $|>$ is an operator to combine multiple verbs
  - $a \ |> f(x)$ is interpreted as $a \ |> f(x = a)$
  - $b \ |> f(x, y)$ is interpreted as $b \ |> f(x = b, y)$
  - $c \ |> f(x) \ |> g(y) \ |> h(z)$ is interpreted as $h(g(f(x = c)))$
  - $d \ |> f(x, y = \_)$ is interpreted as $f(x, y = d)$
- $|>$ make your code more readable
  - Structure sequences of data operations from left to right
  - Avoid nested function calls
  - Minimize the need for local variables and function definitions
  - Make it easy to add steps anywhere in the sequence of operations

References I

Wickham, H. (2011). The Split-Apply-Combine Strategy for Data Analysis. *Journal of Statistical Software*, *40*(1). https://doi.org/10.18637/jss.v040.i01

Wickham, H., Çetinkaya-Rundel, M., & Grolemund, G. (2023). *R for data science: Import, tidy, transform, visualize, and model data* (2nd edition). O'Reilly Media, Inc. https://r4ds.hadley.nz/