

Data Management with dplyr

Anthony Chau

UCI Center for Statistical Consulting

7/24/2021 (updated: 2021-07-26)

What is data management?

- Data management is the practice of manipulating data into more useful, correct, and accurate forms
- Data cleaning is another synonymous term for data management
- Real data is never "clean" - most likely, the data needs heavy data cleaning before we can do any analysis and visualization

What is dplyr

- `dplyr` is a package for general data management
- Straightforward syntax and intuitive names
- Works well in conjunction with `ggplot2`
- Core functions: `select`, `filter`, `mutate`, `arrange`, `summarise`
- Additional functions: `***_join` functions

select()

- `select` selects specific columns from our data
- `select` is another way of expressing ``df[,c("col1", "col2")]`

Aside: %>% pipe operator

- `dplyr` code benefits from the use of the pipe operator `%>%`
- Think of the pipe as chaining together expressions
- The pipe allow us to write less redundant code and condense long data manipulation workflows
- Analogy: In-N-Out assembly line

select() example

- starwars dataset containing character information from dplyr

```
starwars ← dplyr::starwars
colnames(starwars)
#> [1] "name"      "height"    "mass"      "hair_color" "skin_color" "eye_color"
#> [7] "birth_year" "sex"       "gender"    "homeworld" "species"    "films"
#> [13] "vehicles"   "starships"
starwars %>%
  select(name, height, species) %>%
  head(5)
#> # A tibble: 5 x 3
#>   name          height species
#>   <chr>         <int> <chr>
#> 1 Luke Skywalker  172 Human
#> 2 C-3PO          167 Droid
#> 3 R2-D2           96 Droid
#> 4 Darth Vader    202 Human
#> 5 Leia Organa    150 Human
```

select() helpers

- There are many support functions and utilities to help you select columns
- ie: select variables that match a pattern, excluding specific variables, select all variables
- See this help page: [select\(\) helpers](#)

```
starwars %>%
  select(name, ends_with("color")) %>%
  head(5)
#> # A tibble: 5 x 4
#>   name          hair_color skin_color eye_color
#>   <chr>         <chr>      <chr>    <chr>
#> 1 Luke Skywalker blond      fair     blue
#> 2 C-3PO         <NA>      gold     yellow
#> 3 R2-D2         <NA>      white, blue red
#> 4 Darth Vader   none      white    yellow
#> 5 Leia Organa   brown     light    brown
```

filter()

- `filter` filters data given a condition
- Use `filter` to reduce the size of your dataset
- Create expressions with logical operators (`<`, `<=`, `==`, `!=`, `>=`, `>`) and boolean operators (`!`, `|`, `&`)
 - Show anyone with weight greater than 200 pounds: `weight > 200`
 - Show data from Florida and California: `state == "Florida" & state == "California"`
 - Exclude all United airlines flights: `carrier != "United"`

filter() example

```
starwars %>%
  filter(sex = "male") %>%
  select(name, sex, everything()) %>%
  head(3)
#> # A tibble: 3 x 14
#>   name      sex  height  mass hair_color skin_color eye_color birth_year
#>   <chr>    <chr>  <int> <dbl> <chr>      <chr>      <chr>      <dbl>
#> 1 Luke Sky~ male    172    77 blond      fair        blue        19
#> 2 Darth Va~ male    202   136 none       white       yellow      41.9
#> 3 Owen Lars male    178   120 brown, gr~ light      blue        52
#> # ... with 4 more variables: species <chr>, films <list>, vehicles <list>,
#> #   starships <list>

# we can combine dplyr functions
starwars %>%
  filter(sex = "female") %>%
  select(name, species) %>%
  head(3)
#> # A tibble: 3 x 2
#>   name      species
#>   <chr>    <chr>
#> 1 Leia Organa Human
#> 2 Beru Whitesun lars Human
#> 3 Mon Mothma Human
```

filter() example

```
# hair_color = "blond" & eye_color = "blue"
starwars %>%
  filter(hair_color = "blond", eye_color = "blue") %>%
  select(name, hair_color, eye_color, everything())
#> # A tibble: 3 x 14
#>   name      hair_color eye_color height  mass skin_color birth_year sex
#>   <chr>      <chr>      <chr>    <int> <dbl> <chr>          <dbl> <chr>
#> 1 Luke Sky~ blond      blue      172    77 fair          19    male
#> 2 Anakin S~ blond      blue      188    84 fair          41.9 male
#> 3 Finis Va~ blond      blue      170    NA fair          91    male
#> # ... with 4 more variables: species <chr>, films <list>, vehicles <list>,
#> #   starships <list>
```

filter() example

```
starwars %>%  
  filter(species = "Human" | species = "Droid") %>%  
  select(name, species) %>%  
  head(5)  
#> # A tibble: 5 x 2  
#>   name          species  
#>   <chr>         <chr>  
#> 1 Luke Skywalker Human  
#> 2 C-3P0          Droid  
#> 3 R2-D2          Droid  
#> 4 Darth Vader   Human  
#> 5 Leia Organa   Human
```

mutate()

- `mutate` creates new variables in the data
- Use `mutate` to create variables not in the original data

mutate() example

```
starwars %>%  
  mutate(height_ft = height / 30.48) %>%  
  select(name, height, height_ft) %>%  
  head(5)  
#> # A tibble: 5 x 3  
#>   name          height height_ft  
#>   <chr>         <int>     <dbl>  
#> 1 Luke Skywalker    172      5.64  
#> 2 C-3P0             167      5.48  
#> 3 R2-D2              96      3.15  
#> 4 Darth Vader       202      6.63  
#> 5 Leia Organa       150      4.92
```

mutate() example

```
# save data with new variables
starwars2 <-
  starwars %>%
    mutate(height_m = height / 100,
           bmi = mass / (height_m)^2)

starwars2 %>%
  select(name, height, height_m, mass, bmi)
#> # A tibble: 87 x 5
#>   name                height height_m  mass  bmi
#>   <chr>              <int>    <dbl> <dbl> <dbl>
#> 1 Luke Skywalker      172      1.72    77  26.0
#> 2 C-3PO                167      1.67    75  26.9
#> 3 R2-D2                 96      0.96    32  34.7
#> 4 Darth Vader         202      2.02   136  33.3
#> 5 Leia Organa         150      1.5     49  21.8
#> 6 Owen Lars           178      1.78   120  37.9
#> 7 Beru Whitesun lars  165      1.65    75  27.5
#> 8 R5-D4                97      0.97    32  34.0
#> 9 Biggs Darklighter   183      1.83    84  25.1
#> 10 Obi-Wan Kenobi     182      1.82    77  23.2
#> # ... with 77 more rows
```

arrange()

- `arrange` orders rows by the values of variables in the data
- Use `arrange` to present a sorted version of your data

arrange() example

```
starwars %>%  
  # ascending order by height  
  arrange(height) %>%  
  select(name, height) %>%  
  head(8)  
#> # A tibble: 8 x 2  
#>   name                height  
#>   <chr>              <int>  
#> 1 Yoda                66  
#> 2 Ratts Tyerell       79  
#> 3 Wicket Systri Warrick 88  
#> 4 Dud Bolt           94  
#> 5 R2-D2              96  
#> 6 R4-P17             96  
#> 7 R5-D4              97  
#> 8 Sebulba           112
```


arrange() example

```
starwars %>%  
  # descending order by height  
  arrange(desc(height)) %>%  
  select(name, height) %>%  
  head(8)  
#> # A tibble: 8 x 2  
#>   name          height  
#>   <chr>         <int>  
#> 1 Yarael Poof      264  
#> 2 Tarfful         234  
#> 3 Lama Su         229  
#> 4 Chewbacca       228  
#> 5 Roos Tarpals    224  
#> 6 Grievous        216  
#> 7 Taun We         213  
#> 8 Rugor Nass      206
```

arrange() example

```
mtcars %>%  
  # descending order by number of cylinders and then ascending order by mass  
  arrange(desc(cyl), mpg) %>%  
  select(cyl, mpg) %>%  
  head(8)  
#>           cyl  mpg  
#> Cadillac Fleetwood    8 10.4  
#> Lincoln Continental    8 10.4  
#> Camaro Z28             8 13.3  
#> Duster 360             8 14.3  
#> Chrysler Imperial     8 14.7  
#> Maserati Bora          8 15.0  
#> Merc 450SLC            8 15.2  
#> AMC Javelin            8 15.2
```

summarise()

- `summarise` creates a new data frame with computed "summary" functions
- Use `summarise` to compute descriptive statistics and summary information about the data

summarise() example

```
starwars %>%  
  # remove missing values from height before computing height  
  # notice the naming of the summary variable on the left hand side  
  summarise(mean_height = mean(height, na.rm=TRUE))  
#> # A tibble: 1 x 1  
#>   mean_height  
#>       <dbl>  
#> 1       174.
```

```
starwars %>%  
  # n() computes the number of observations  
  summarise(max_mass = max(height, na.rm=TRUE),  
            n = n())  
#> # A tibble: 1 x 2  
#>   max_mass      n  
#>   <int> <int>  
#> 1     264     87
```

Aside: `group_by()`

- Often, we would like to compute summary information for different groups
- Groups are like categories in a categorical variable.
- `group_by()` is useful when used with `mutate` or `summarise` to create group level variables or summaries
- `mutate` example: Add a new variable that contains the mean weight for males only and the mean weight for females only
- `summarise` example: Compute a new data frame of the mean weight for males only and the mean weight for females only

Group by summarise()

```
starwars %>%  
  group_by(sex) %>%  
  summarise(mean_height = mean(height, na.rm=TRUE))  
#> # A tibble: 5 x 2  
#>   sex          mean_height  
#>   <chr>          <dbl>  
#> 1 female          169.  
#> 2 hermaphroditic  175  
#> 3 male            179.  
#> 4 none            131.  
#> 5 <NA>            181.
```

Group by mutate()

```
starwars %>%  
  group_by(sex) %>%  
  mutate(mean_height = mean(height, na.rm=TRUE)) %>%  
  select(name, sex, height, mean_height) %>%  
  head(6)  
#> # A tibble: 6 x 4  
#> # Groups:   sex [3]  
#>   name          sex    height mean_height  
#>   <chr>         <chr>    <int>      <dbl>  
#> 1 Luke Skywalker male      172      179.  
#> 2 C-3PO         none     167      131.  
#> 3 R2-D2         none      96      131.  
#> 4 Darth Vader   male     202      179.  
#> 5 Leia Organa   female   150      169.  
#> 6 Owen Lars     male     178      179.
```

rename()

- `rename()` renames variables
- `rename_with()` can rename multiple variables with a function

rename() example

```
starwars2 <-  
  starwars %>%  
    # new_name = old_name  
    rename(height_cm = height)
```

```
colnames(starwars)
```

```
#> [1] "name"      "height"    "mass"      "hair_color" "skin_color" "eye_color"  
#> [7] "birth_year" "sex"       "gender"     "homeworld"  "species"     "films"  
#> [13] "vehicles"   "starships"
```

```
# notice height is now height_cm
```

```
colnames(starwars2)
```

```
#> [1] "name"      "height_cm" "mass"      "hair_color" "skin_color" "eye_color"  
#> [7] "birth_year" "sex"       "gender"     "homeworld"  "species"     "films"  
#> [13] "vehicles"   "starships"
```

rename() example

```
starwars2 <-  
  starwars %>%  
    # rename_with looks at every variable name by default  
    # Use ~ to signal that we are supplying a function  
    # gsub replaces all underscores with dot  
    # .x indicates a single variable name  
    rename_with(~gsub(pattern = "_",  
                       replacement = ".",  
                       x = .x))  
  
colnames(starwars)  
#> [1] "name"      "height"    "mass"      "hair_color" "skin_color" "eye_color"  
#> [7] "birth_year" "sex"       "gender"    "homeworld" "species"    "films"  
#> [13] "vehicles"   "starships"  
# notice the _ replaced by .  
colnames(starwars2)  
#> [1] "name"      "height"    "mass"      "hair.color" "skin.color" "eye_color"  
#> [7] "birth.year" "sex"       "gender"    "homeworld" "species"    "films"  
#> [13] "vehicles"   "starships"
```

Recap

- `dplyr` facilitates quick and useful data manipulation
- Select specific variables with `select`
- Filter data with `filter`
- Create new variables with `mutate`
- Order data with `arrange`
- Summarize data with `summarise`
- Use `group_by` to perform grouped operations