Lecture 8: Data Wrangling with `dplyr`

STAT 450, Fall 2021

The R package `dplyr` provides a set of functions for data manipulation, or data wrangling. `dplyr` is one of the core packages in the tidyverse.

For this lecture, we will focus on the following commonly used `dplyr` functions:

- `select()` take a subset of the columns (variables)
- `filter()` take a subset of the rows (observations)
- `arrange()` reorder the rows
- `mutate()` creates new variables that are functions of existing variables

The names of these functions are *verbs* that provide a grammar for data wrangling.

To load `dplyr` into your R session run the following command:

```
library(tidyverse)
```

This loads the core tidyverse packages, which includes `dplyr`. Often `dplyr` is used in combination with the other packages such as `ggplot2`.

Or you can just load `dplyr`, without the other `tidyverse` packages:

```
library(dplyr)
```

`dplyr` references:

https://dplyr.tidyverse.org/index.html

https://dplyr.tidyverse.org/reference/index.html

Star Wars Data

We will use the `starwars` data set, which comes with `dplyr`.

```
starwars
```

```
## # A tibble: 87 x 14
##    name    height  mass hair_color  skin_color eye_color birth_year sex   gender
##    <chr>    <int> <dbl> <chr>       <chr>      <chr>           <dbl> <chr> <chr>
##  1 Luke S~    172    77 blond       fair       blue             19   male  mascu~
##  2 C-3PO      167    75 <NA>        gold       yellow          112   none  mascu~
##  3 R2-D2       96    32 <NA>        white, bl~ red              33   none  mascu~
##  4 Darth ~    202   136 none        white      yellow           41.9 male  mascu~
##  5 Leia O~    150    49 brown       light      brown            19   fema~ femin~
##  6 Owen L~    178   120 brown, grey light      blue             52   male  mascu~
##  7 Beru W~    165    75 brown       light      blue             47   fema~ femin~
##  8 R5-D4       97    32 <NA>        white, red red              NA   none  mascu~
##  9 Biggs ~    183    84 black       light      brown            24   male  mascu~
## 10 Obi-Wa~    182    77 auburn, wh~ fair       blue-gray        57   male  mascu~
## # ... with 77 more rows, and 5 more variables: homeworld <chr>, species <chr>,
## #   films <list>, vehicles <list>, starships <list>
```

Type `help(starwars)` to read about this data set in the help menu.

select()

Use `select()` to subset the columns (variables) of a data frame.

```
select(starwars, name, homeworld, species)
```

```
## # A tibble: 87 x 3
##    name               homeworld species
##    <chr>              <chr>     <chr>
##  1 Luke Skywalker     Tatooine  Human
##  2 C-3PO              Tatooine  Droid
##  3 R2-D2              Naboo     Droid
##  4 Darth Vader        Tatooine  Human
##  5 Leia Organa        Alderaan  Human
##  6 Owen Lars          Tatooine  Human
##  7 Beru Whitesun lars Tatooine  Human
##  8 R5-D4              Tatooine  Droid
##  9 Biggs Darklighter  Tatooine  Human
## 10 Obi-Wan Kenobi     Stewjon   Human
## # ... with 77 more rows
```

Use : to select a range of consecutive variables.

```
select(starwars, name:hair_color)
```

```
## # A tibble: 87 x 4
##    name               height  mass hair_color
##    <chr>               <int> <dbl> <chr>
##  1 Luke Skywalker        172    77 blond
##  2 C-3PO                 167    75 <NA>
##  3 R2-D2                  96    32 <NA>
##  4 Darth Vader           202   136 none
##  5 Leia Organa           150    49 brown
##  6 Owen Lars             178   120 brown, grey
##  7 Beru Whitesun lars    165    75 brown
##  8 R5-D4                  97    32 <NA>
##  9 Biggs Darklighter     183    84 black
## 10 Obi-Wan Kenobi        182    77 auburn, white
## # ... with 77 more rows
```

Select all columns ending with "color":

```
select(starwars, ends_with("color"))
```

```
## # A tibble: 87 x 3
##    hair_color    skin_color  eye_color
##    <chr>         <chr>       <chr>
##  1 blond         fair        blue
##  2 <NA>          gold        yellow
##  3 <NA>          white, blue red
##  4 none          white       yellow
##  5 brown         light       brown
##  6 brown, grey   light       blue
##  7 brown         light       blue
##  8 <NA>          white, red  red
##  9 black         light       brown
## 10 auburn, white fair        blue-gray
## # ... with 77 more rows
```

filter()

Use `filter()` to subset the rows of a data frame. The first argument is the name of the data frame. The second argument is a **logical expression** that specifies the rows to subset.

Before demonstrating the `filter()` function, let's use `select()` to narrow down the columns of the `starwars` data frame:

```
starwars2 <- select(starwars, name, height, mass, homeworld, species)
```

Here, the assignment operator `<-` stores the selected columns in a new data frame called `starwars2`. We will use this `starwars2` data frame throughout the remainder of this lecture.

Subset all the rows corresponding to Star Wars characters that are from Tatooine:

```
filter(starwars2, homeworld == "Tatooine")
```

```
## # A tibble: 10 x 5
##    name               height  mass homeworld species
##    <chr>               <int> <dbl> <chr>     <chr>
##  1 Luke Skywalker        172    77 Tatooine  Human
##  2 C-3PO                 167    75 Tatooine  Droid
##  3 Darth Vader           202   136 Tatooine  Human
##  4 Owen Lars             178   120 Tatooine  Human
##  5 Beru Whitesun lars    165    75 Tatooine  Human
##  6 R5-D4                  97    32 Tatooine  Droid
##  7 Biggs Darklighter     183    84 Tatooine  Human
##  8 Anakin Skywalker      188    84 Tatooine  Human
##  9 Shmi Skywalker        163    NA Tatooine  Human
## 10 Cliegg Lars           183    NA Tatooine  Human
```

Subset all the rows corresponding to characters that are from Tatooine **and** are Human:

```
filter(starwars2, homeworld == "Tatooine" & species == "Human")
```

```
## # A tibble: 8 x 5
##   name               height  mass homeworld species
##   <chr>               <int> <dbl> <chr>     <chr>
## 1 Luke Skywalker        172    77 Tatooine  Human
## 2 Darth Vader           202   136 Tatooine  Human
## 3 Owen Lars             178   120 Tatooine  Human
## 4 Beru Whitesun lars    165    75 Tatooine  Human
## 5 Biggs Darklighter     183    84 Tatooine  Human
## 6 Anakin Skywalker      188    84 Tatooine  Human
## 7 Shmi Skywalker        163    NA Tatooine  Human
## 8 Cliegg Lars           183    NA Tatooine  Human
```

Subset all the rows corresponding to characters that are Droids **or** Wookiees:

```
filter(starwars2, species == "Droid" | species == "Wookiee")
```

```
## # A tibble: 8 x 5
##   name      height  mass homeworld species
##   <chr>      <int> <dbl> <chr>     <chr>
## 1 C-3PO        167    75 Tatooine  Droid
## 2 R2-D2         96    32 Naboo     Droid
## 3 R5-D4         97    32 Tatooine  Droid
## 4 Chewbacca    228   112 Kashyyyk  Wookiee
## 5 IG-88        200   140 <NA>      Droid
## 6 R4-P17        96    NA <NA>      Droid
## 7 Tarfful      234   136 Kashyyyk  Wookiee
## 8 BB8           NA    NA <NA>      Droid
```

Subset all the rows corresponding to characters that are over 200 cm (about 6.5 ft) tall.

```
filter(starwars2, height > 200)
```

```
## # A tibble: 10 x 5
##    name         height  mass homeworld species
##    <chr>         <int> <dbl> <chr>     <chr>
##  1 Darth Vader     202   136 Tatooine  Human
##  2 Chewbacca       228   112 Kashyyyk  Wookiee
##  3 Roos Tarpals    224    82 Naboo     Gungan
##  4 Rugor Nass      206    NA Naboo     Gungan
##  5 Yarael Poof     264    NA Quermia   Quermian
##  6 Lama Su         229    88 Kamino    Kaminoan
##  7 Taun We         213    NA Kamino    Kaminoan
##  8 Grievous        216   159 Kalee     Kaleesh
##  9 Tarfful         234   136 Kashyyyk  Wookiee
## 10 Tion Medon      206    80 Utapau    Pau'an
```

Logical operators

The second argument of `filter()` is a logical expression that returns `TRUE`, `FALSE`, or `NA` (for missing values).

For example, type the following code:

```
starwars$homeworld
starwars$homeworld == "Tatooine"
```

You'll see that the logical expression returns `TRUE` if the `homeworld` is Tatooine, and `FALSE` or `NA` otherwise.

The == symbol is a logical operator (or comparison operator) that tests for equality. A common mistake is to use = instead of ==. The = symbol is used for assignment, while == is the logical operator for equality. For example:

```
x = 1
x
```

```
## [1] 1
```

```
x == 1
```

```
## [1] TRUE
```

```
x == 2
```

```
## [1] FALSE
```

```
x != 2 # example of not operator
```

```
## [1] TRUE
```

The following table summarizes the different logical operators in R that you can use with `filter()`:

Operator	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to
x | y	x OR y
x & y	x AND y

Exercises:

1. What's wrong with the following code? Why does it return an error message?

```
filter(starwars2, species = "Droid")
```

2. Use `filter()` to subset all the rows of the `starwars2` data frame corresponding to characters that are **not** Human.

3. Use `filter()` to subset all the rows of of the `starwars2` data frame corresponding to characters that are **not** Human **and** are from the homeworld Tatooine.

arrange()

Use `arrange()` to order the rows of a data frame by the values of a column.

```
arrange(starwars2, height)
```

```
## # A tibble: 87 x 5
##    name                  height  mass homeworld   species
##    <chr>                  <int> <dbl> <chr>       <chr>
##  1 Yoda                      66    17 <NA>        Yoda's species
##  2 Ratts Tyerell             79    15 Aleen Minor Aleena
##  3 Wicket Systri Warrick     88    20 Endor       Ewok
##  4 Dud Bolt                  94    45 Vulpter     Vulptereen
##  5 R2-D2                     96    32 Naboo       Droid
##  6 R4-P17                    96    NA <NA>        Droid
##  7 R5-D4                     97    32 Tatooine    Droid
##  8 Sebulba                  112    40 Malastare   Dug
##  9 Gasgano                  122    NA Troiken     Xexto
## 10 Watto                    137    NA Toydaria    Toydarian
## # ... with 77 more rows
```

Use `desc()` to reorder in descending order:

```
arrange(starwars2, desc(height))
```

```
## # A tibble: 87 x 5
##    name         height  mass homeworld species
##    <chr>         <int> <dbl> <chr>     <chr>
##  1 Yarael Poof     264    NA Quermia   Quermian
##  2 Tarfful         234   136 Kashyyyk  Wookiee
##  3 Lama Su         229    88 Kamino    Kaminoan
##  4 Chewbacca       228   112 Kashyyyk  Wookiee
##  5 Roos Tarpals    224    82 Naboo     Gungan
##  6 Grievous        216   159 Kalee     Kaleesh
##  7 Taun We         213    NA Kamino    Kaminoan
##  8 Rugor Nass      206    NA Naboo     Gungan
##  9 Tion Medon      206    80 Utapau    Pau'an
## 10 Darth Vader     202   136 Tatooine  Human
## # ... with 77 more rows
```

Exercises:

4. Use `arrange()` to order the rows of `starwars2` according to the values of `mass`.
5. What does the following code do?

```
arrange(starwars2, species, height)
```

mutate()

Use `mutate()` to add a new variable to a data frame.

For example, in the `starwars2` data frame `height` is in centimeters. Let's add a new variable to this data frame called `height_ft`, which is height in feet.

```
starwars2 <- mutate(starwars2, height_ft = height / 30.48)
starwars2
```

```
## # A tibble: 87 x 6
##    name               height  mass homeworld species height_ft
##    <chr>               <int> <dbl> <chr>     <chr>       <dbl>
##  1 Luke Skywalker        172    77 Tatooine  Human        5.64
##  2 C-3PO                 167    75 Tatooine  Droid        5.48
##  3 R2-D2                  96    32 Naboo     Droid        3.15
##  4 Darth Vader           202   136 Tatooine  Human        6.63
##  5 Leia Organa           150    49 Alderaan  Human        4.92
##  6 Owen Lars             178   120 Tatooine  Human        5.84
##  7 Beru Whitesun lars    165    75 Tatooine  Human        5.41
##  8 R5-D4                  97    32 Tatooine  Droid        3.18
##  9 Biggs Darklighter     183    84 Tatooine  Human        6.00
## 10 Obi-Wan Kenobi        182    77 Stewjon   Human        5.97
## # ... with 77 more rows
```

The function `rename()` can be used change the name of a variable. For example, we now might want to change the name of the variable `height` to `height_cm`:

```
starwars2 <- rename(starwars2, height_cm = height)
starwars2
```

```
## # A tibble: 87 x 6
##    name               height_cm  mass homeworld species height_ft
##    <chr>                  <int> <dbl> <chr>     <chr>       <dbl>
##  1 Luke Skywalker           172    77 Tatooine  Human        5.64
##  2 C-3PO                    167    75 Tatooine  Droid        5.48
##  3 R2-D2                     96    32 Naboo     Droid        3.15
##  4 Darth Vader              202   136 Tatooine  Human        6.63
##  5 Leia Organa              150    49 Alderaan  Human        4.92
##  6 Owen Lars                178   120 Tatooine  Human        5.84
##  7 Beru Whitesun lars       165    75 Tatooine  Human        5.41
##  8 R5-D4                     97    32 Tatooine  Droid        3.18
##  9 Biggs Darklighter        183    84 Tatooine  Human        6.00
## 10 Obi-Wan Kenobi           182    77 Stewjon   Human        5.97
## # ... with 77 more rows
```