Lab: Manipulating data with dplyr

Mark Huber

Summary

In this lab you will learn how to manipulate data using the `dplyr` package.

- Start by loading in the `dplyr` library (installing the package first if necessary.)

```
# install.packages("dplyr")
library(dplyr)
```

- The `dplyr` package contains tools for manipulating data contained in a data.frame or tibble. Let's look at the start of the `starwars` dataset.

```
starwars |> head(5)
```

name	height	m...	hair_color	skin_color	eye_color	birth_year	sex	gender	▸
<chr>	<int>	<dbl>	<chr>	<chr>	<chr>	<dbl>	<chr>	<chr>	
Luke Skywalker	172	77	blond	fair	blue	19.0	male	masculine	
C-3PO	167	75	NA	gold	yellow	112.0	none	masculine	
R2-D2	96	32	NA	white, blue	red	33.0	none	masculine	
Darth Vader	202	136	none	white	yellow	41.9	male	masculine	
Leia Organa	150	49	brown	light	brown	19.0	female	feminine	

5 rows | 1-9 of 14 columns

The data in this tibble consists of some of the characters that appear in the Star Wars movies. Since it is 87 by 13, there are 87 observations. Each observation consists of 13 measurements of *variables* (which are also called *factors*.)

Select for variables/factors

We might not be interested in all the variables, and the `select` function allows us to only look at the variables that are important. For instance, if we only wanted the name, mass, species, and homeworld, we could use

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

The result is a tibble that just contains the 4 variables listed. We can also use helper functions like `starts_with`, `ends_with`, and `contains`. Try

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

to see the variables that end with the string `"color"`, and

```
select(starwars, contains("a"))
```

to see those variables that have the string `"a"` somewhere in the name.

The first parameter we pass to `select` is the name of the variable, but it also possible to use **pipes** to accomplish the same task. The following command pipes the variable `starwars` into the `select` function:

```
starwars |> select(contains("a"))
```

Of course, the results of these select commands can be assigned to new variables using the assignment operator `<-` .

```
starwars_a <-
  starwars |>
  select(contains("a"))
```

Problem 1

Assign to `answer01` the observations in the `starwars` dataset with variables `name` , `gender` , and `homeworld` .

Problem 2

Assign to `answer02` the factors of `starwars` that contain the letter e.

slice picks out fixed observations

To pick out a particular set of rows, just name those rows. For intance, the following picks out rows 2, 4, and 7.

```
starwars |> slice(c(2, 4, 7))
```

name	height	...	hair_color	skin_color	eye_color	birth_year	sex	gender
<chr>	<int>	<dbl>	<chr>	<chr>	<chr>	<dbl>	<chr>	<chr>
C-3PO	167	75	NA	gold	yellow	112.0	none	masculine
Darth Vader	202	136	none	white	yellow	41.9	male	masculine
Beru Whitesun lars	165	75	brown	light	blue	47.0	female	feminine

3 rows | 1-9 of 14 columns

The seq function can also be used here. The following picks out rows 3, 6, 9, up to 30.

```
starwars |> slice(seq(3, 30, by = 3))
```

name	height	ma...	hair_color	skin_color	eye_color	birth_year	sex	gender
<chr>	<int>	<dbl>	<chr>	<chr>	<chr>	<dbl>	<chr>	<chr>
R2-D2	96	32.0	NA	white, blue	red	33.0	none	masculine

Owen Lars	178	120.0	brown, grey	light	blue	52.0	male	masculine
Biggs Darklighter	183	84.0	black	light	brown	24.0	male	masculine
Wilhuff Tarkin	180	*NA*	auburn, grey	fair	blue	64.0	male	masculine
Greedo	173	74.0	*NA*	green	black	44.0	male	masculine
Jek Tono Porkins	180	110.0	brown	fair	blue	*NA*	male	masculine
Boba Fett	183	78.2	black	fair	brown	31.5	male	masculine
Lando Calrissian	177	79.0	black	dark	brown	31.0	male	masculine
Mon Mothma	150	*NA*	auburn	fair	blue	48.0	female	feminine
Nien Nunb	160	68.0	none	grey	black	*NA*	male	masculine

1-10 of 10 rows | 1-9 of 14 columns

The colon notation `:` can also be used. The following picks out the first ten rows.

```
starwars |> slice(1:10)
```

name	**height**	**...**	**hair_color**	**skin_color**	**eye_color**	**birth_year**	**sex**	**gender**
<chr>	<int>	<dbl>	<chr>	<chr>	<chr>	<dbl>	<chr>	<chr>
Luke Skywalker	172	77	blond	fair	blue	19.0	male	masculine
C-3PO	167	75	*NA*	gold	yellow	112.0	none	masculine
R2-D2	96	32	*NA*	white, blue	red	33.0	none	masculine
Darth Vader	202	136	none	white	yellow	41.9	male	masculine
Leia Organa	150	49	brown	light	brown	19.0	female	feminine
Owen Lars	178	120	brown, grey	light	blue	52.0	male	masculine
Beru Whitesun lars	165	75	brown	light	blue	47.0	female	feminine
R5-D4	97	32	*NA*	white, red	red	*NA*	none	masculine
Biggs Darklighter	183	84	black	light	brown	24.0	male	masculine
Obi-Wan Kenobi	182	77	auburn, white	fair	blue-gray	57.0	male	masculine

1-10 of 10 rows | 1-9 of 14 columns

Problem 3

Assign to `answer03` the characters in rows 5 through 10 inclusive.

filter picks out observations

In the `starwars` data set, each row/data point/observation is a particular character in the Star Wars universe.

Now let's search for the droid characters. To find the droids that we are looking for, try

```
starwars |>
  filter(species == "Droid")
```

name	**height**	**m...**	**hair_color**	**skin_color**	**eye_color**	**birth_year**	**sex**	**gender**	**homeworld** ▶
<chr>	<int>	<dbl>	<chr>	<chr>	<chr>	<dbl>	<chr>	<chr>	<chr>
C-3PO	167	75	*NA*	gold	yellow	112	none	masculine	Tatooine
R2-D2	96	32	*NA*	white, blue	red	33	none	masculine	Naboo
R5-D4	97	32	*NA*	white, red	red	*NA*	none	masculine	Tatooine
IG-88	200	140	none	metal	red	15	none	masculine	*NA*
R4-P17	96	*NA*	none	silver, red	red, blue	*NA*	none	feminine	*NA*
BB8	*NA*	*NA*	none	none	black	*NA*	none	masculine	*NA*

6 rows | 1-10 of 14 columns

Of course this search was practically instantaneous because there are so few rows of data. In practice, there are often more data rows than variables. So it can be helpful to insert a `select` function before the `filter` function. We then connect the `select` function to the `filter` functin with a pipe.

```
starwars |>
  select(name, mass, species, gender) |>
  filter(species == "Droid")
```

name	**mass**	**species**	**gender**
<chr>	<dbl>	<chr>	<chr>
C-3PO	75	Droid	masculine
R2-D2	32	Droid	masculine
R5-D4	32	Droid	masculine
IG-88	140	Droid	masculine
R4-P17	*NA*	Droid	feminine
BB8	*NA*	Droid	masculine

6 rows

Problem 4

Assign to `answer04` the characters whose gender is `masculine`.

Logical operators

We can also use filters to search for more than one characteristic with the `&` logical operator. This represents logical and, which is true only if both of the expressions are true. So `TRUE & TRUE` equals `TRUE`, `FALSE & TRUE` is `FALSE`, `TRUE & FALSE` is `FALSE`, and `FALSE & FALSE` is `FALSE`.

Try

```
starwars |>
  select(name, mass, species, gender) |>
  filter(species == "Droid" & mass > 50)
```

name <chr>	mass <dbl>	species <chr>	gender <chr>
C-3PO	75	Droid	masculine
IG-88	140	Droid	masculine

2 rows

to find the droids that have mass greater than 50 kilograms.

The logical operator `|` is true if either one (or both) of the expressions it connects is true. So `TRUE | TRUE` equals `TRUE`, `FALSE | TRUE` is `TRUE`, `TRUE | FALSE` is `TRUE`, and `FALSE | FALSE` is `FALSE`. Try

```
starwars |>
  select(name, mass, species, gender) |>
  filter(species == "Droid" | mass == 136)
```

name <chr>	mass <dbl>	species <chr>	gender <chr>
C-3PO	75	Droid	masculine
R2-D2	32	Droid	masculine
Darth Vader	136	Human	masculine
R5-D4	32	Droid	masculine
IG-88	140	Droid	masculine
R4-P17	*NA*	Droid	feminine
Tarfful	136	Wookiee	masculine
BB8	*NA*	Droid	masculine

8 rows

This should pick up the well known Darth Vader and the less well-known Tarfful, who was a Wookie general during the Clone Wars.

You will notice that some of the droids are missing values for factors. For instance, BB8 does not have a height, mass, birth_year, or homeworld value. These entries are listed as NA (not available). To locate these values, we can use the `is.na` function. Try

```
starwars |>
  select(name, mass, species, gender) |>
  filter(is.na(mass) & species == "Droid")
```

name <chr>	mass <dbl>	species <chr>	gender <chr>
R4-P17	*NA*	Droid	feminine
BB8	*NA*	Droid	masculine

2 rows

to find all the data where the mass is not available.

Another useful logical operator in this context is `!`, which means **not**. So the following will tell us the droids where the mass does not equal NA.

```
starwars |>
  select(name, mass, species, gender) |>
  filter(!is.na(mass) & species == "Droid")
```

name <chr>	mass <dbl>	species <chr>	gender <chr>
C-3PO	75	Droid	masculine
R2-D2	32	Droid	masculine
R5-D4	32	Droid	masculine
IG-88	140	Droid	masculine

4 rows

Logical operators are evaluated from left to right. So for instance,

```
starwars |>
  select(name, mass, species, gender) |>
  filter(species == "Droid" & mass > 100 | mass < 40)
```

name <chr>	mass <dbl>	species <chr>	gender <chr>
R2-D2	32	Droid	masculine
R5-D4	32	Droid	masculine
Yoda	17	Yoda's species	masculine

IG-88	140	Droid	masculine
Wicket Systri Warrick	20	Ewok	masculine
Ratts Tyerell	15	Aleena	masculine

6 rows

For Wicket, it was false that his species is a droid, and false that his mass is greater than 100. So the first two clauses become false. But the final mass value is less than 40, and `FALSE | TRUE` evaluates to `TRUE`.

If instead we are interested in only those droids who have mass greater than 100 or mass less than 40, then

```
starwars |>
  select(name, mass, species, gender) |>
  filter(species == "Droid" & (mass > 100 | mass < 40))
```

does the job.

Problem 5

Assign to `answer05` the characters who are female with a mass of at most 50 kilograms.

Problem 6

Assign to `answer06` the characters who are female with a mass of at most 50 kilograms and at least 40 kilograms.

Problem 7

Assign to `answer07` the data from `starwars` that has the factors `name`, `gender`, `hair_color`, and `homeworld`, and only characters with blond hair from Tatooine.

Mutate

Mutate alters a tibble by adding an extra variable that can be some function of other variables. For instance, suppose we are interested in how the mass varies with height. We could compute the ratio as follows.

```
starwars |>
  select(name, mass, height) |>
  mutate(massweightratio = mass / height)
```

name <chr>	mass <dbl>	height <int>	massweightratio <dbl>
Luke Skywalker	77.0	172	0.4476744
C-3PO	75.0	167	0.4491018
R2-D2	32.0	96	0.3333333

Darth Vader	136.0	202	0.6732673
Leia Organa	49.0	150	0.3266667
Owen Lars	120.0	178	0.6741573
Beru Whitesun lars	75.0	165	0.4545455
R5-D4	32.0	97	0.3298969
Biggs Darklighter	84.0	183	0.4590164
Obi-Wan Kenobi	77.0	182	0.4230769

1-10 of 87 rows Previous **1** 2 3 4 5 6 ... 9 Next

Note that if either the mass or the height variable is NA, then their ratio will also be NA

```
starwars |>
  select(name, mass, height) |>
  mutate(massweightratio = mass/height) |>
  filter(is.na(massweightratio))
```

name	**mass**	**height**	**massweightratio**
<chr>	<dbl>	<int>	<dbl>
Wilhuff Tarkin	*NA*	180	*NA*
Mon Mothma	*NA*	150	*NA*
Arvel Crynyd	*NA*	*NA*	*NA*
Finis Valorum	*NA*	170	*NA*
Rugor Nass	*NA*	206	*NA*
Ric Olié	*NA*	183	*NA*
Watto	*NA*	137	*NA*
Quarsh Panaka	*NA*	183	*NA*
Shmi Skywalker	*NA*	163	*NA*
Bib Fortuna	*NA*	180	*NA*

1-10 of 28 rows Previous **1** 2 3 Next

Problem 8

Currently the mass is in kilograms. Create a new tibble `answer08` that has a variable where the mass is measured in pounds by multiplying by 2.20462. Call this new variable `mass_lbs` .

Note that if we try to add a categorical variable like `hair_color` to a numerical variable like `height` , an error is thrown.

```
starwars |>
  mutate(test = hair_color + height)
```

```
## Error in `mutate()`:
## ! Problem while computing `test = hair_color + height`.
## Caused by error in `hair_color + height`:
## ! non-numeric argument to binary operator
```

arrange to sort observations

Another way to transform the data is through the `arrange` function. This sorts the data by a particular variable so we can learn about the highest or lowest values. The following sorts the variable by mass.

```
starwars |>
  select(name, mass, height) |>
  arrange(mass)
```

As you can see, this arranges the data from low mass to high mass.

When you arrange based on a string variable like `hair_color`, it sorts things alphabetically.

```
starwars |>
  select(name, hair_color, mass, height) |>
  arrange(hair_color)
```

If we want to reverse the sort, we use the helper function `desc`. By putting this around the variable name, we reverse the order of the sorting.

```
starwars |>
  select(name, hair_color, mass, height) |>
  mutate(massweightratio = mass/height) |>
  arrange(desc(hair_color))
```

To break ties, we can add another variable to the `arrange` function.

```
starwars |>
  select(name, hair_color, mass, height) |>
  mutate(massweightratio = mass / height) |>
  arrange(desc(hair_color), mass)
```

name <chr>	hair_color <chr>	mass <dbl>	height <int>	massweightratio <dbl>
Yoda	white	17.0	66	0.2575758
Dooku	white	80.0	193	0.4145078
Ki-Adi-Mundi	white	82.0	198	0.4141414
Jocasta Nu	white	NA	167	NA

Captain Phasma	unknown	NA	NA	NA
Ratts Tyerell	none	15.0	79	0.1898734
Sebulba	none	40.0	112	0.3571429
Dud Bolt	none	45.0	94	0.4787234
Wat Tambor	none	48.0	193	0.2487047
Sly Moore	none	48.0	178	0.2696629

1-10 of 87 rows Previous **1** 2 3 4 5 6 ... 9 Next

Problem 9

Arrange the `starwars` tibble in descending order of `mass` divided by `height`, and then pick out the first five rows. Assign the result to `answer09`.

summarize

The summarize (or summarise for those who prefer UK English) command allows the finding of functions of a variable.

For instance, to pick out the largest height (ignoring any NA values) in the tibble, use the following code.

```
starwars |>
  summarize(max(height, na.rm = TRUE))
```

max(height, na.rm = TRUE) <int>
264

1 row

This calculation could be given a better name using a parameter assignement.

```
starwars |>
  summarize(max_height = max(height, na.rm = TRUE))
```

max_height <int>
264

1 row

> Note that inside a function like summarize, it is necessary to assign a value to `max_height` using `=`, and not the regular assignement operator `<-`.

Other useful functions for summarize include mean for finding the sample average, and sd for estimating the standard deviation.

```
starwars |>
  summarize(avg_height = mean(height, na.rm = TRUE))
```

```
avg_height
     <dbl>
   174.358
1 row
```

Problem 10

Use summarize to find the average weight (in kg) of the characters in the `starwars` dataset, ignoring any NA values. Assign the result to `answer10` .