

Data Science for Everyone – Data Wrangling

Dr. Ab Mosca (they/them)

Slides based off slides courtesy of Jordan Crouser (<https://jcrouser.github.io/>)

Plan for Today

- Wrangling Data in one Table

The 5 Verbs

- `select()`
- `filter()`
- `mutate()`
- `arrange()`
- `summarize()`

The 5 Verbs: dplyr

mutate() the data i.e. add or modify a column



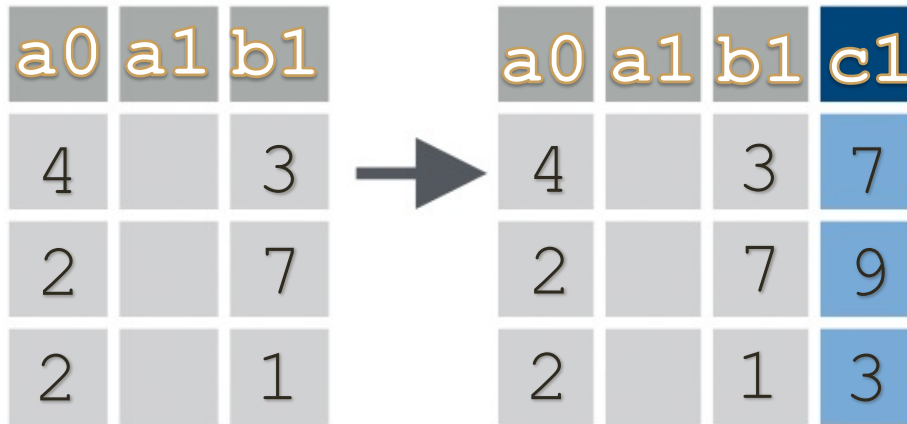
a0	a1	b1	c1
4			8
2			4
2			4

- Add a column to the dataset as a product of existing column. Ex.

```
data %>%  
  mutate(c1 = a0 * 2)
```

The 5 Verbs: dplyr

mutate() the data i.e. add or modify a column



a0	a1	b1	c1
4		3	7
2		7	9
2		1	3

- Add a column to the dataset as a product of existing column(s). Ex.

```
data %>%  
  mutate(c1 = a0 + b1)
```

mutate() the data i.e. add or modify a column

```
34 ```{r, message=FALSE}  
35 library(tidyverse)|  
36 library(babynames)  
37 head(babynames)  
38 ```
```

A tibble: 6 x 5

year <dbl>	sex <chr>	name <chr>	n <int>	prop <dbl>
1880	F	Mary	7065	0.07238359
1880	F	Anna	2604	0.02667896
1880	F	Emma	2003	0.02052149
1880	F	Elizabeth	1939	0.01986579
1880	F	Minnie	1746	0.01788843
1880	F	Margaret	1578	0.01616720

6 rows

The 5 Verbs: dplyr

mutate() the data i.e. add or modify a column

- Work with whoever is near you to add a column to babynames called popular that is TRUE if a name was assigned to more than 1% of all babies in a given year

The 5 Verbs: dplyr

mutate() the data i.e. add or modify a column

- Work with whoever is near you to add a column to babynames called popular that is TRUE if a name was assigned to more than 1% of all babies in a given year

```
babynames <- babynames %>%  
  mutate(popular = prop > 0.01)
```


rename

rename() change the name of a variable

```
babynames <- babynames %>%  
  rename(is_popular = popular)
```

Review

Use `is_popular` and `filter()` to create a subset of `babynames` that contains only popular names

Review

Use `is_popular` and `filter()` to create a subset of `babynames` that contains only popular names

```
popular <- babynames %>%  
  filter(is_popular)
```

Review

Use `is_popular` and `filter()` to create a subset of `babynames` that contains only popular names

```
popular <- babynames %>%  
  filter(is_popular)
```

How many names are popular?

Review

Use `is_popular` and `filter()` to create a subset of `babynames` that contains only popular names

```
popular <- babynames %>%  
  filter(is_popular)
```


How many names are popular?

```
nrow(popular)
```

3878

The 5 Verbs: dplyr

arrange() the rows in a specific order



a0	a1	b1
2		
5		
0		
3		

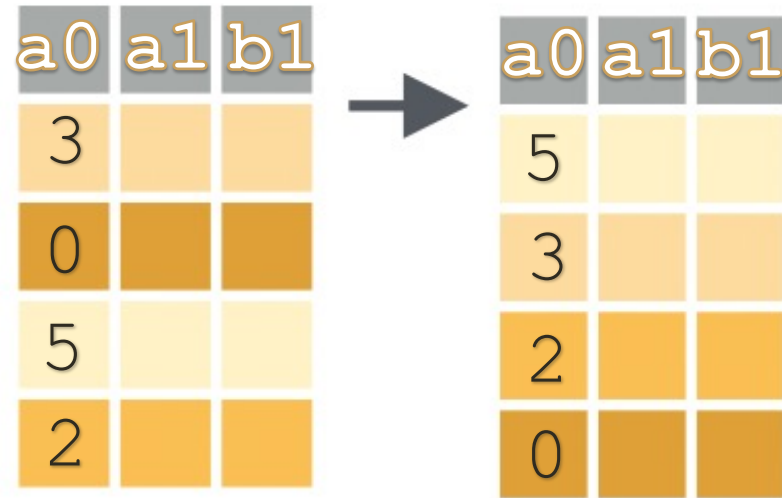
a0	a1	b1
0		
2		
3		
5		

- Order rows by value of a column(s) from low to high. Ex.

```
data %>%  
  arrange(a0)
```

The 5 Verbs: dplyr

arrange() the rows in a specific order



The diagram illustrates the `arrange()` function by showing a transformation of a data frame. On the left, the initial data frame has columns `a0`, `a1`, and `b1` with rows containing values (3, 0, 5, 2). On the right, after applying `arrange(a0)`, the rows are reordered based on the values in column `a0` in ascending order, resulting in rows with values (5, 3, 2, 0). The color intensity of the cells represents the magnitude of the values, with darker shades indicating higher values.

a0	a1	b1
3		
0		
5		
2		

→

a0	a1	b1
5		
3		
2		
0		

- Order rows by value of a column(s) from low to high. Use `desc()` to go from high to low. Ex.

```
data %>%  
  arrange(desc(a0))
```

The 5 Verbs: dplyr

arrange() the rows in a specific order

- Work with whoever is near you to find the most popular names of all time

The 5 Verbs: dplyr

arrange() the rows in a specific order

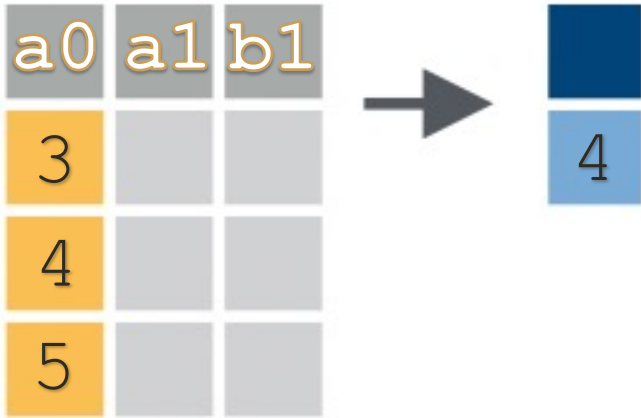
- Work with whoever is near you to find the most popular names of all time

```
popular %>%  
  arrange(desc(prop))
```

year <dbl>	sex <chr>	name <chr>	n <int>	prop <dbl>	is_popular <lgl>
1880	M	John	9655	0.08154561	TRUE
1881	M	John	8769	0.08098299	TRUE
1880	M	William	9532	0.08050676	TRUE
1883	M	John	8894	0.07907394	TRUE
1881	M	William	8524	0.07872038	TRUE
1882	M	John	9557	0.07831617	TRUE
1884	M	John	9288	0.07648812	TRUE

The 5 Verbs: dplyr

summarize() column with a single value(s)



- Apply a summary function to a column. Ex.

```
data %>%  
  summarize(mean(a0))
```

The 5 Verbs: dplyr

summarize() column with a single value(s)

```
171 `{{r}}`  
172 ab %>%  
173   summarize(max_abs = max(n))  
174 `{{r}}`
```

A tibble: 1 x 1

max_abs
<int>

41

1 row

The 5 Verbs: dplyr

summarize() column with a single value(s)



- Tip: use the function `n()` inside of `summarize` to keep track of how many rows were summarized

The 5 Verbs: dplyr

summarize() column with a single value(s)

```
191  
192 ▾ ```{r}  
193 ab %>%  
194   summarize(num_rows = n(), max_abs = max(n))  
195 ▸ ```
```

A tibble: 1 x 2

num_rows <int>	max_abs <int>
29	41

1 row

Your Turn!

- Work with classmates:
 - Choose a name and find the year it was used most frequently
 - What was the most popular name that year?
 - In which year was the name you picked given to M and F babies most equally? i.e. closest to a 50/50 split