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

Using Data

- **What are some things we might want (or need) to do with data in order to analyze it?**

Using Data

- **What are some things we might want (or need) to do with data in order to analyze it?**

- Select some (but not all) columns
- Filter to some (but not all) rows
- Mutate the data i.e. add or modify a column
- Arrange the rows in a specific order
- Summarize column with a single value(s)

The 5 Verbs: `dplyr`

- **What are some things we might want (or need) to do with data in order to analyze it?**

- `select()` some (but not all) columns
- `filter()` to some (but not all) rows
- `mutate()` the data i.e. add or modify a column
- `arrange()` the rows in a specific order
- `summarize()` column with a single value(s)

dplyr

- R package for data wrangling (cleaning, reshaping, and analyzing data)

- Big ideas:
  - Each "verb" (function) takes as input a `tbl_df` and returns a `tbl_df`
  - Verbs can be combined with "chaining" via the pipe operator (`%>%`)

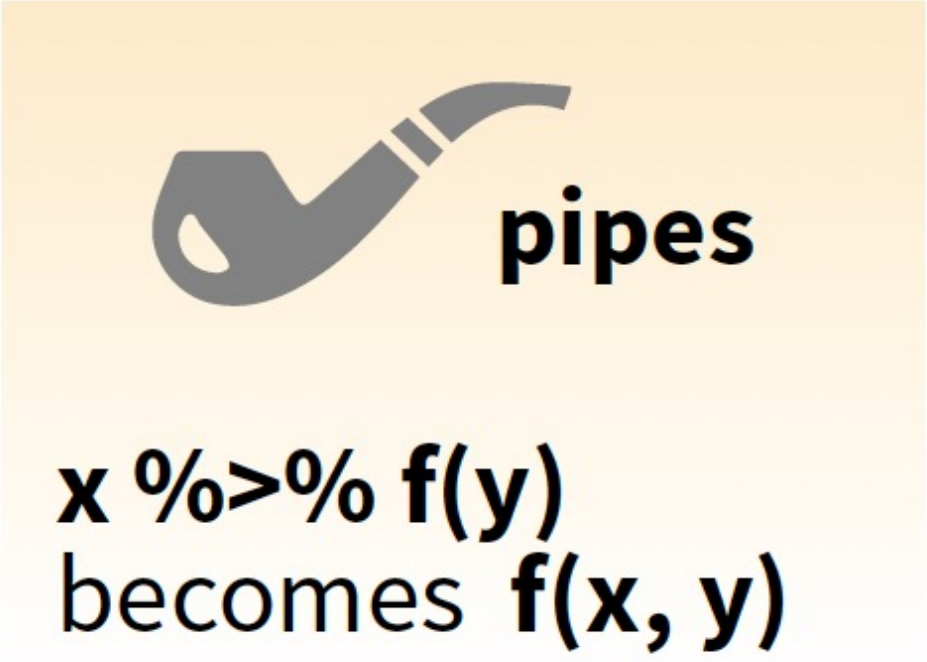
- Cheatsheet: https://www.rstudio.com/resources/cheatsheets/

tbl_df

- "tibble"
- object of class `tbl`
- re-imagining of `data.frame` (makes them easier to work with!)
- `tidyverse` (which includes `dplyr`) works with tibbles

Pipe Operator

Verbs are used with the **pipe (%>%) operator**



Pipe operator

%>% (pipe operator)

With the pipe operator the expression

```
verb(mydata, arguments)
```

becomes

```
mydata %>%
  verb(arguments)
```

Pipe operator

%>% (pipe operator)

More generally,

```
function(x, args)
```

becomes

```
x %>%
  function(args)
```

Pipe operator

%>% (pipe operator)

This helps A LOT with readability!

Work with the person next to you to rewrite this using pipes:

```
select(data, arg)
```

Pipe operator

%>% (pipe operator)

This helps A LOT with readability!

```
data %>%
   select(arg)
```

Pipe operator

%>% (pipe operator)

This helps A LOT with readability!

Work with the person next to you to rewrite this using pipes:

```
select(filter(mutate(data, args1), args2), args3)
```

Pipe operator

%>% (pipe operator)

This helps A LOT with readability!

```
select(filter(mutate(data, args1), args2), args3)
```

vs.

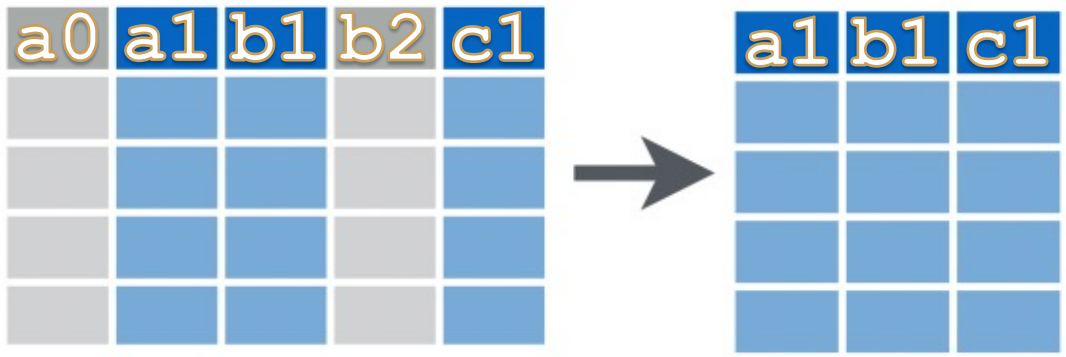
```
data %>%
   mutate(args1) %>%
   filter(args2) %>%
   select(args3)
```

The 5 Verbs

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

The 5 Verbs: `dplyr`

`select()` some (but not all) columns

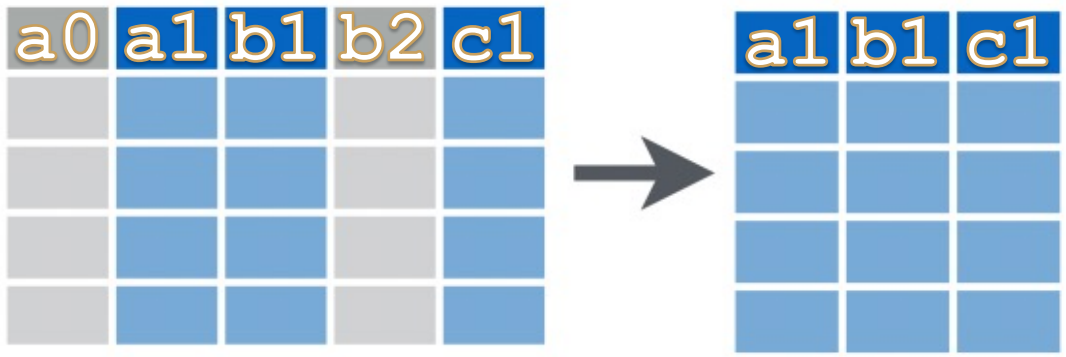


- Select column(s) by name. Ex:

```
data %>%
   select("a1", "b1", "c1")
```

The 5 Verbs: dplyr

`select()` some (but not all) columns

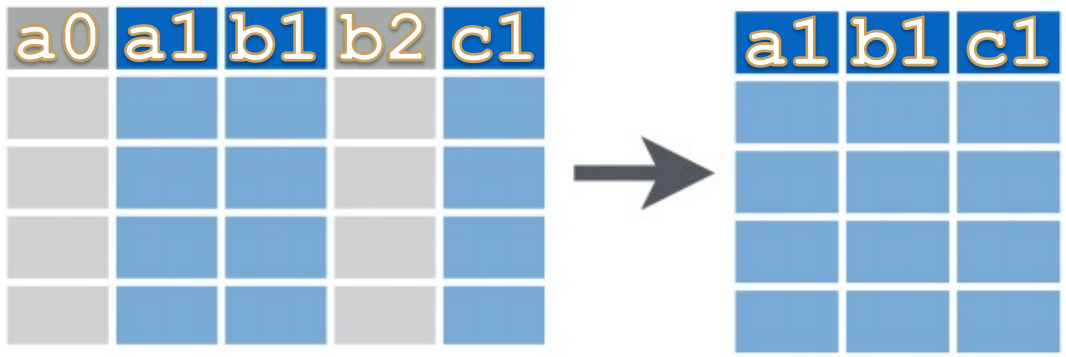


- Select column(s) by name or use other helper functions. Ex.
  - contains(match), ends_with(match), matches(match), starts_with(match)

```
data %>%
   select(contains("1"))
```

The 5 Verbs: `dplyr`

`select()` some (but not all) columns



- Select column(s) by specifying exclusions. Ex.

```
data %>%
   select(-a0, -b2)
```

`select()` some (but not all) columns

```{r, message=FALSE}
library(tidyverse)
library(babynames)
head(babynames)
```

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

`select()` some (but not all) columns

The 5
dpl

```{r}
babynames %>%
  select(year, name, n) %>%
  head()
```

A tibble: 6 x 3

year <dbl>	name <chr>	n <int>
1880	Mary	7065
1880	Anna	2604
1880	Emma	2003
1880	Elizabeth	1939
1880	Minnie	1746
1880	Margaret	1578

6 rows

The 5 Verbs: `dplyr`

`select()` some (but not all) columns

- Work with whoever is near you to select all columns of babynames except sex and n

The 5 Verbs: `dplyr`

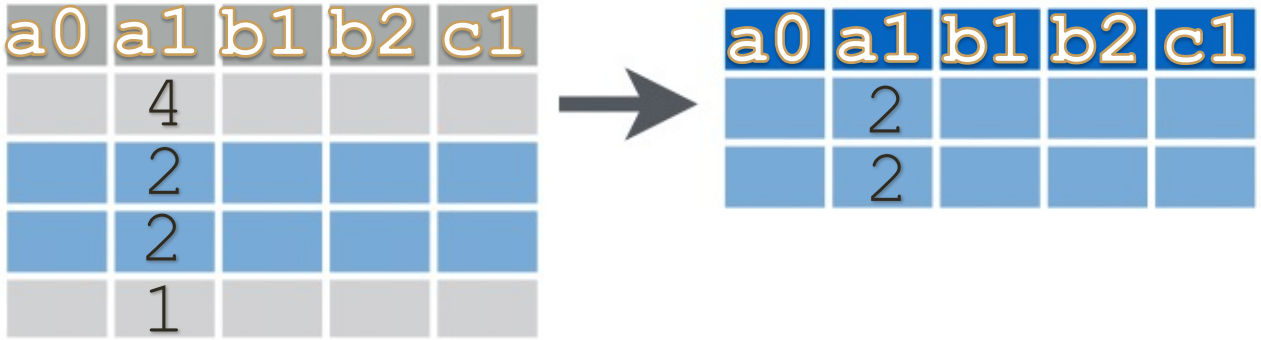
`select()` some (but not all) columns

- Work with whoever is near you to select all columns of babynames except sex and n

```
babynames %>%
  select(-sex, -n)
```

The 5 Verbs: `dplyr`

`filter()` to some (but not all) rows

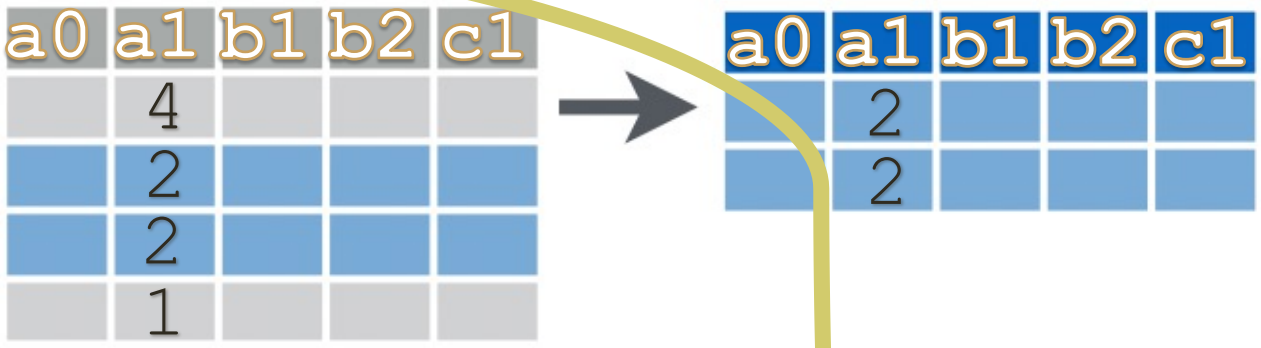


- Select rows that meet logical criteria. Ex:

```
data %>%
   filter(a1 == 2)
```

`filter()` to some (but not all) rows

Operator	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to
!x	Not x
x | y	x OR y
x & y	x AND y
isTRUE(x)	test if X is TRUE



- Select rows that meet logical criteria. Ex:

```
data %>%
   filter((a1 < 3) & (a1 > 1))
```

`filter()` to some (but not all) rows

```{r}
# Replace 'Ab' with your own name if you like!
babynames %>%
  filter(name == "Ab") %>%
  head()
```

A tibble: 6 x 5

year <dbl>	sex <chr>	name <chr>	n <int>	prop <dbl>
1880	M	Ab	5	4.223e-05
1882	M	Ab	5	4.097e-05
1885	M	Ab	6	5.175e-05
1887	M	Ab	5	4.574e-05
1916	M	Ab	8	8.660e-06
1917	M	Ab	6	6.250e-06

6 rows

The 5 Verbs: `dplyr`

`filter()` to some (but not all) rows

- Work with whoever is near you to filter babynames to only years after 1920

- Work with whoever is near you to filter babynames to only popular names (let popular be names that more than 15% of babies were named in a given year)

The 5 Verbs: `dplyr`

`filter()` to some (but not all) rows

- Work with whoever is near you to filter babynames to only years after 1920

```
babynames %>%
  filter(year > 1920)
```

- Work with whoever is near you to filter babynames to only popular names (let popular be names that at least 15% of babies were named in a given year)

```
babynames %>%
  filter(prop >= 0.15)
```

The 5 Verbs: `dplyr`

- I want to plot counts of the name Ab over the years by sex
- How do I get the dataset I need to make this plot?

- Talk it out with whoever is near you!

The 5 Verbs: dplyr

Combining verbs

```{r}
ab <- babynames %>%
            filter(name == "Ab") %>%
            select(year, name, sex, n)
```

The 5 Verbs: dplyr

Combining verbs

```{r}
ab <- babynames %>%
                filter(name == "Ab") %>%
                select(year, name, sex, n)
```

babynames is filtered to rows where name == “Ab”

The 5 Verbs: dplyr

Combining verbs

```{r}
ab <- babynames %>%
                filter(name == "Ab") %>%
                select(year, name, sex, n)
```

babynames is filtered to rows where name == “Ab”

we select year, name, sex, and n columns from filtered babynames

The 5 Verbs: `dplyr`

- I want to plot counts of the name Ab over the years by sex ☑
- Now, make the plot!
- Talk it out with whoever is near you!

The 5 Verbs: dplyr

```{r}
# Load ggplot
library(ggplot2)
#plot
ggplot(data = ab, aes(x = year, y = n)) +
  geom_line(aes(color = sex))
```