# Data Wrangling

SSEP 2022 Afternoon Day 1

Dr. Ab Mosca (they/them)

Slides based on slides courtesy of Jordan Crouser: https://jcrouser.github.io/MassMutual-IntroR/, https://jcrouser.github.io/MassMutual-DataVis/, https://beanumber.github.io/sds192/

# Data

# Definition: Data

## Definition of *data*

**1** **:** factual information (such as measurements or statistics) used as a basis for reasoning, discussion, or calculation

*//* the *data* is plentiful and easily available

— H. A. Gleason, Jr.

*//* comprehensive *data* on economic growth have been published

— N. H. Jacoby

**2** **:** information in digital form that can be transmitted or processed

**3** **:** information output by a sensing device or organ that includes both useful and irrelevant or redundant information and must be processed to be meaningful

https://www.merriam-webster.com/dictionary/data

# Examples of Data

- Work with the person next to you to find some examples of data (on the internet, or from your own life)
- Add links to your examples to the data examples Jamboard: Data Examples Jamboard

# Using Data

- Data Examples Jamboard
- 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 (`%>%`)

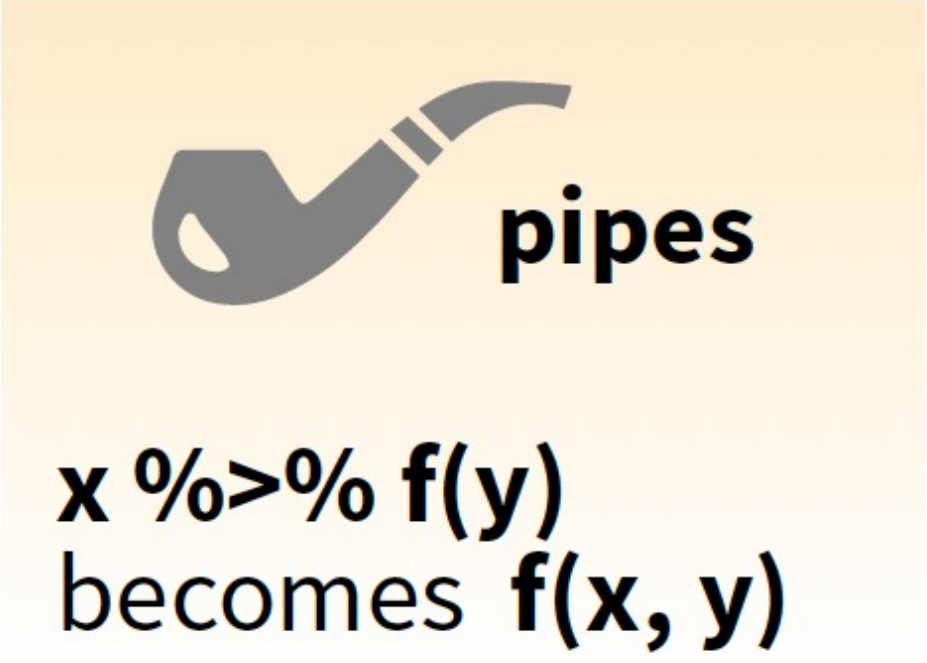
- Cheatsheet: https://www.rstudio.com/resources/cheatsheets/


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

Pipe operator

# %>% (pipe operator)

This helps A LOT with readability!

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

vs.

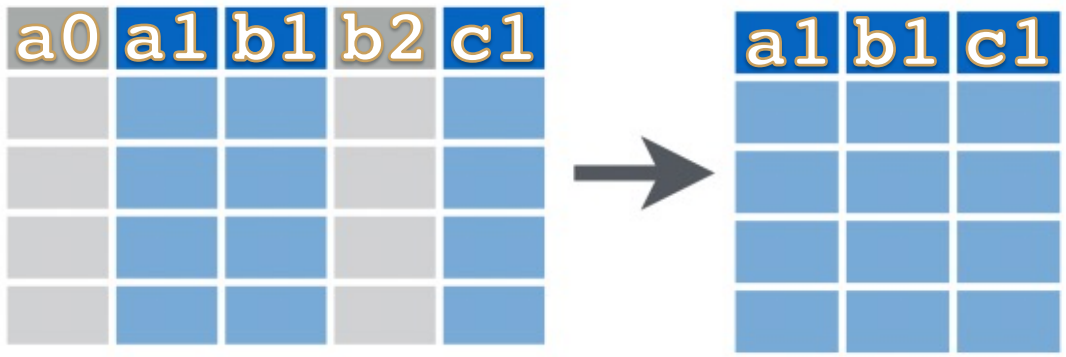
```
data %>%
   mutate(args1) %>%
   filter(args2) %>%
   select(args3)
```

# The 5 Verbs

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

# The 5 Verbs: dplyr

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

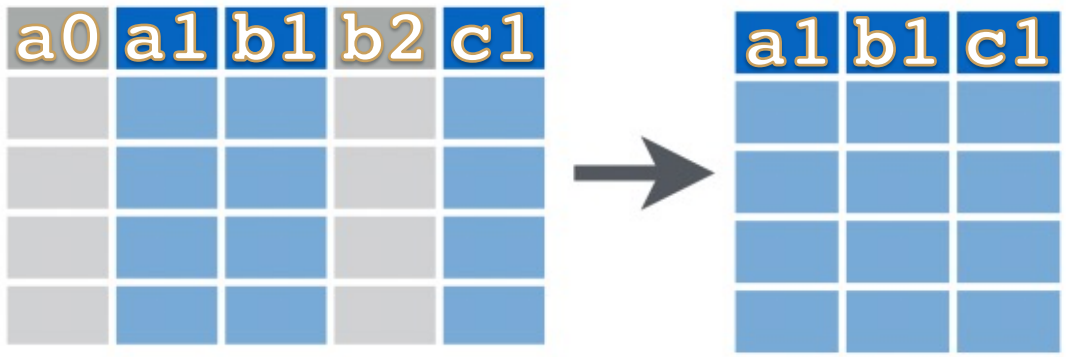



- Select column(s) by name. Ex:

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

# The 5 Verbs: `dplyr`

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

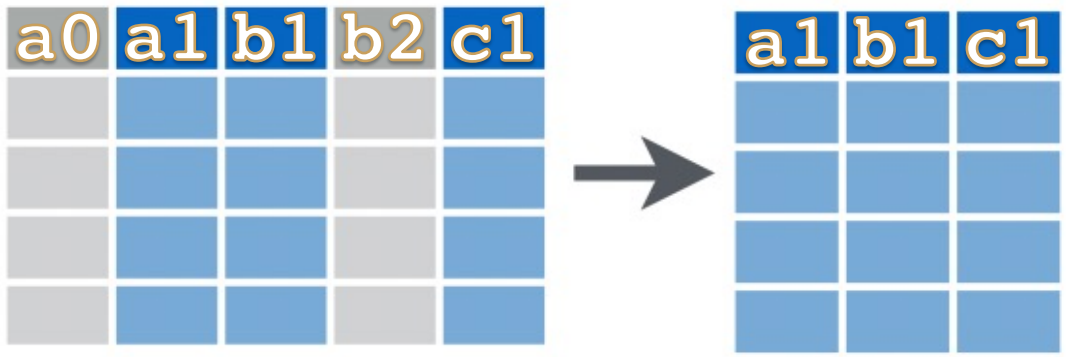



- Select column(s) by name or use other helper functions. Ex.
  - contains(match), ends_with(match), matches(match), starts_with(match)

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

# The 5 Verbs: dplyr

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

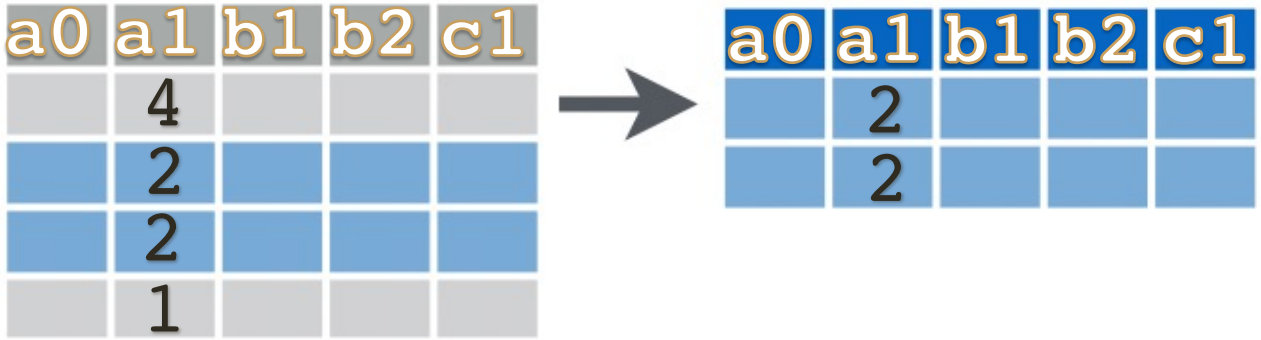



- Select column(s) by specifying exclusions. Ex.

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

# The 5 Verbs: dplyr

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

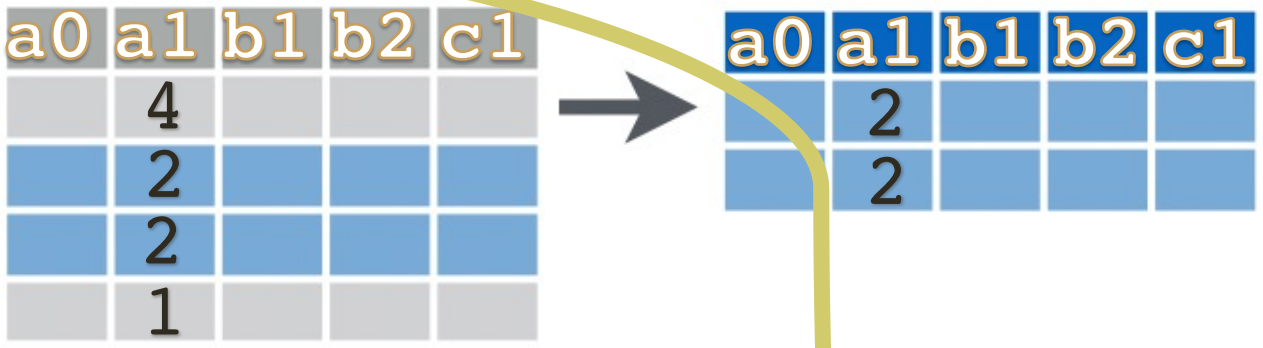



- Select rows that meet logical criteria. Ex:

```
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 |

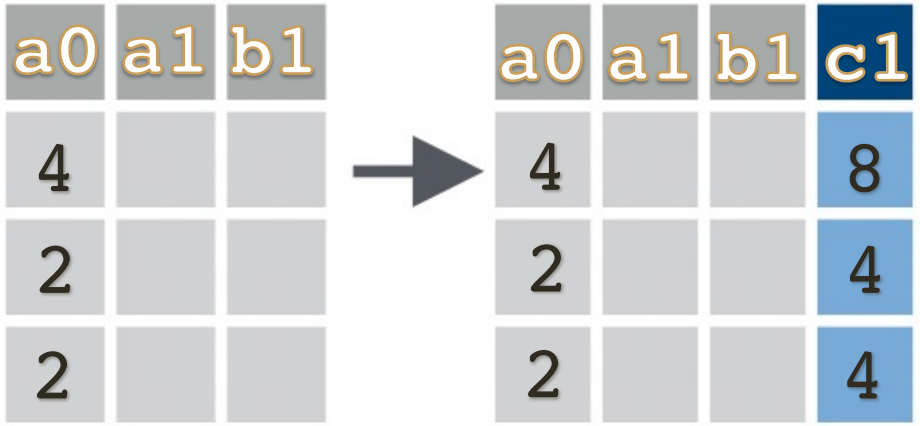



- Select rows that meet logical criteria. Ex:

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

# The 5 Verbs: dplyr

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

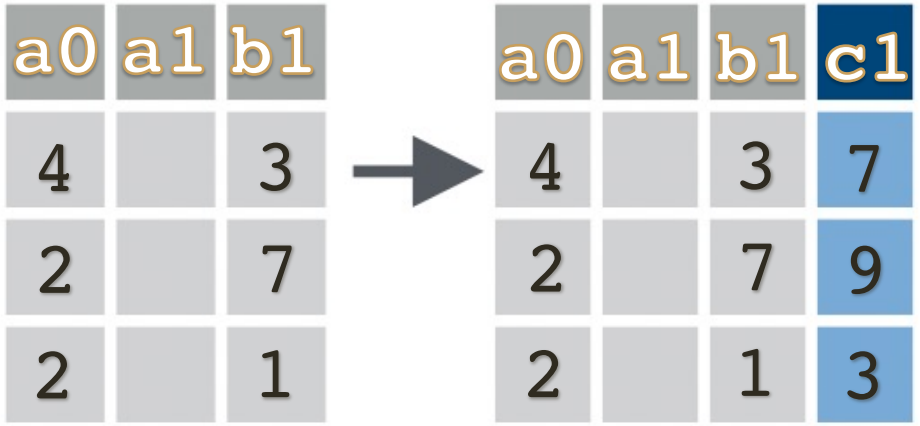



- Add a column to the dataset as a product of existing column(s). Ex.

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

# The 5 Verbs: dplyr

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

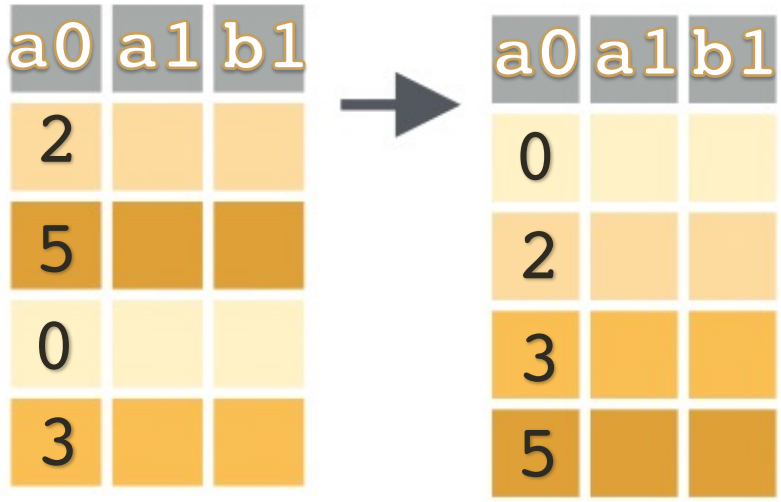



- Add a column to the dataset as a product of existing column(s). Ex.

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

# The 5 Verbs: dplyr

**`arrange()`** the rows in a specific order

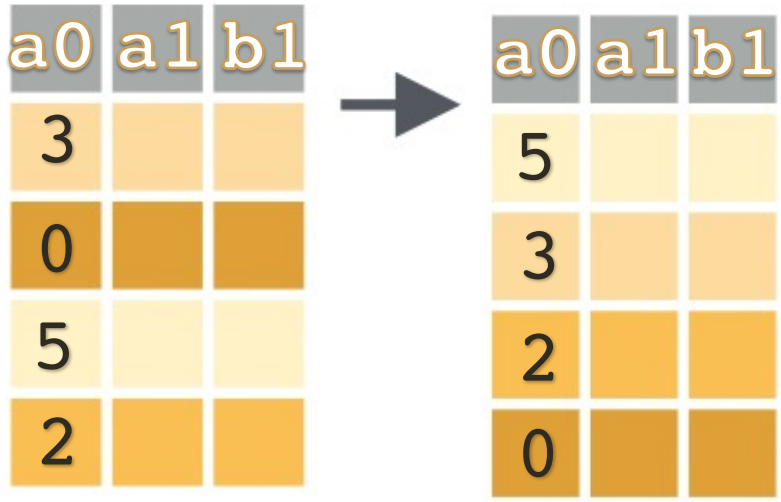



- Order rows by value of a column(s) from low to high. Ex.

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

# The 5 Verbs: dplyr

**`arrange()`** the rows in a specific order

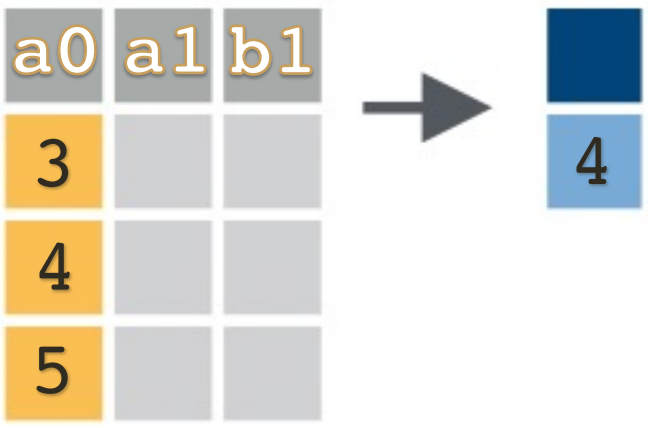



- Order rows by value of a column(s) from low to high. Use `desc()` to go from high to low. Ex.

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

# The 5 Verbs: dplyr

**`summarize()`** column with a single value(s)




- Apply a summary function to a column. Ex.

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