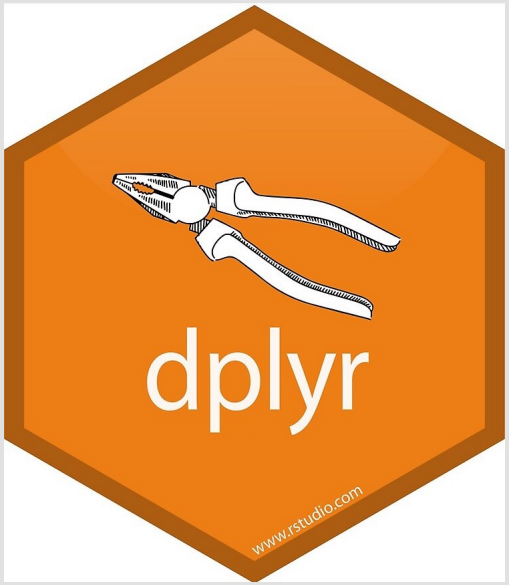
Introduction to `dplyr`

Data Science for Biologists

Spring 2021, Dr. Spielman


dplyr
www.rstudio.com


dplyr
dplyr.tidyverse.org

The great and powerful pipe

Examples of piping, and formatting goals

```
## Tired
log(5)
```

```
## [1] 1.609438
```

```
## Wired
5 %>% log()
```

```
## [1] 1.609438
```

```
## Inspired
5 %>%
  log()
```

```
## [1] 1.609438
```

Loading libraries

You must load your libraries for each R session OR NONE OF THIS WORKS!!

```
library(tidyverse) # convenient for "everything"
#or
library(dplyr) # if you just want to load dplyr (less common)
```

```
print(diamonds)
```

```
## # A tibble: 20 x 5
##    carat cut       color price     x
##    <dbl> <ord>     <ord> <int> <dbl>
##  1  0.34 Ideal     D      1272  4.49
##  2  0.55 Premium   F      1354  5.26
##  3  0.23 Very Good E       402  4.01
##  4  0.31 Premium   H       558  4.34
##  5  1.11 Ideal     F      7823  6.62
##  6  0.23 Very Good E       485  4.02
##  7  1.32 Ideal     H      9572  6.98
##  8  0.35 Ideal     E       767  4.52
##  9  0.34 Premium   I       765  4.49
## 10  1.35 Premium   I      5715  7.26
## 11  1.41 Ideal     G     11009  7.22
## 12  1.02 Premium   I      4113  6.48
## 13  1.51 Premium   H      9762  7.54
## 14  1    Good      G      5484  6.38
## 15  0.73 Ideal     H      3463  5.8
## 16  1.01 Very Good D      4338  6.43
## 17  0.72 Ideal     F      2879  5.76
## 18  2.02 Very Good D     15334  8.19
## 19  0.59 Ideal     E      1607  5.36
## 20  0.3  Ideal     G       684  4.31
```

`price`: Price in US dollars

`carat`: Weight of the diamond

`cut`: Quality of the diamond

`color`

`x`: width of diamond in mm

Your first `dplyr` function: `glimpse()`

```
str(diamonds)
```

```
## tibble [20 × 5] (S3: tbl_df/tbl/data.frame)
##  $ carat: num [1:20] 0.34 0.55 0.23 0.31 1.11 0.23 1.32 0.35 0.34 1.35 ...
##  $ cut  : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 3 4 5 3 5 5 4 4 ...
##  $ color: Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 1 3 2 5 3 2 5 2 6 6 ...
##  $ price: int [1:20] 1272 1354 402 558 7823 485 9572 767 765 5715 ...
##  $ x    : num [1:20] 4.49 5.26 4.01 4.34 6.62 4.02 6.98 4.52 4.49 7.26 ...
```

```
glimpse(diamonds)
```

```
## Rows: 20
## Columns: 5
## $ carat <dbl> 0.34, 0.55, 0.23, 0.31, 1.11, 0.23, 1.32, 0.35, 0.34, 1.35, 1.4…
## $ cut   <ord> Ideal, Premium, Very Good, Premium, Ideal, Very Good, Ideal, Id…
## $ color <ord> D, F, E, H, F, E, H, E, I, I, G, I, H, G, H, D, F, D, E, G
## $ price <int> 1272, 1354, 402, 558, 7823, 485, 9572, 767, 765, 5715, 11009, 4…
## $ x     <dbl> 4.49, 5.26, 4.01, 4.34, 6.62, 4.02, 6.98, 4.52, 4.49, 7.26, 7.2…
```

Fully look at the data

```
glimpse(diamonds)
```

```
## Rows: 20
## Columns: 5
## $ carat <dbl> 0.34, 0.55, 0.23, 0.31, 1.11, 0.23, 1.32, 0.35, 0.34, 1.35, 1.4…
## $ cut   <ord> Ideal, Premium, Very Good, Premium, Ideal, Very Good, Ideal, Id…
## $ color <ord> D, F, E, H, F, E, H, E, I, I, G, I, H, G, H, D, F, D, E, G
## $ price <int> 1272, 1354, 402, 558, 7823, 485, 9572, 767, 765, 5715, 11009, 4…
## $ x     <dbl> 4.49, 5.26, 4.01, 4.34, 6.62, 4.02, 6.98, 4.52, 4.49, 7.26, 7.2…
```

```
summary(diamonds)
```

```
##      carat               cut        color        price             x         
##  Min.   :0.230   Fair      :0   D:3   Min.   :  402.0   Min.   :4.010  
##  1st Qu.:0.340   Good      :1   E:4   1st Qu.:  766.5   1st Qu.:4.490  
##  Median :0.725   Very Good:4   F:3   Median : 3171.0   Median :5.780  
##  Mean   :0.822   Premium   :6   G:3   Mean   : 4369.3   Mean   :5.773  
##  3rd Qu.:1.163   Ideal     :9   H:4   3rd Qu.: 6242.0   3rd Qu.:6.710  
##  Max.   :2.020                 I:3   Max.   :15334.0   Max.   :8.190  
##                                J:0
```

Common tasks we perform on datasets

- Subsetting rows.
  - Ex: Work with Premium diamonds.
  - Ex: Work with only diamonds above a certain carat.
- Removing duplicate rows

- Creating new columns
- Rearranging, removing, or keeping only certain columns
- Renaming columns

- Arrange the data based on a column
  - Ex: Arrange in order of price

- Summarizing data
  - Ex: Calculating the mean price
  - Ex: Calculating the mean price for *each* diamond quality category

Functions ("verbs") for wrangling datasets

- Subsetting rows. `filter()`
  - Ex: Work with Premium diamonds.
  - Ex: Work with only diamonds above a certain carat.
- Removing duplicate rows `distinct()`

- Creating new columns `mutate()`
- Rearranging, removing, or keeping only certain columns `select()`
- Renaming columns `rename()`

- Arrange the data based on a column `arrange()`
  - Ex: Arrange in order of price

- Summarizing data `summarize()`
  - Ex: Calculating the mean price
  - Ex: Calculating the mean price for *each* diamond quality category

Data frame in, data frame out.

The *first argument* for most `dplyr` verbs (functions) is a data frame (tibble).

Those verbs all *return* a data frame (tibble).

THIS IS REALLY IMPORTANT!

Diving in with `filter()`

```
# Reminder of the dataset:
diamonds
```

```
## # A tibble: 20 x 5
##    carat cut       color price     x
##    <dbl> <ord>     <ord> <int> <dbl>
##  1  0.34 Ideal     D      1272  4.49
##  2  0.55 Premium   F      1354  5.26
##  3  0.23 Very Good E       402  4.01
##  4  0.31 Premium   H       558  4.34
##  5  1.11 Ideal     F      7823  6.62
##  6  0.23 Very Good E       485  4.02
##  7  1.32 Ideal     H      9572  6.98
##  8  0.35 Ideal     E       767  4.52
##  9  0.34 Premium   I       765  4.49
## 10  1.35 Premium   I      5715  7.26
## 11  1.41 Ideal     G     11009  7.22
## 12  1.02 Premium   I      4113  6.48
## 13  1.51 Premium   H      9762  7.54
## 14  1    Good      G      5484  6.38
## 15  0.73 Ideal     H      3463  5.8
## 16  1.01 Very Good D      4338  6.43
## 17  0.72 Ideal     F      2879  5.76
## 18  2.02 Very Good D     15334  8.19
## 19  0.59 Ideal     E      1607  5.36
## 20  0.3  Ideal     G       684  4.31
```

Diving in with `filter()`

Goal: Subset the data to only Premium quality diamonds

- First argument: a data frame
- Additional argument(s): **logical statements** (code that gives `TRUE` OR `FALSE`) about which rows you want to keep
- *We can directly refer to columns within `dplyr` code*

```
# Keep only Premium quality diamonds:
filter(diamonds, cut == "Premium")
```

```
## # A tibble: 6 x 5
##   carat cut     color price     x
##   <dbl> <ord>   <ord> <int> <dbl>
## 1  0.55 Premium F      1354  5.26
## 2  0.31 Premium H       558  4.34
## 3  0.34 Premium I       765  4.49
## 4  1.35 Premium I      5715  7.26
## 5  1.02 Premium I      4113  6.48
## 6  1.51 Premium H      9762  7.54
```

Using pipe is the preferred style

```
# Not preferred:
# filter(diamonds, cut == "Premium")

# Also not preferred:
# diamonds %>% filter(cut == "Premium")

# THIS!!!!
# STARS STARS STARS! THIS WAY!
diamonds %>%
  filter(cut == "Premium")
```

```
## # A tibble: 6 x 5
##   carat cut     color price     x
##   <dbl> <ord>   <ord> <int> <dbl>
## 1  0.55 Premium F      1354  5.26
## 2  0.31 Premium H       558  4.34
## 3  0.34 Premium I       765  4.49
## 4  1.35 Premium I      5715  7.26
## 5  1.02 Premium I      4113  6.48
## 6  1.51 Premium H      9762  7.54
```

Cautionary tales

```
head(diamonds)
```

```
## # A tibble: 6 x 5
##   carat cut       color price     x
##   <dbl> <ord>     <ord> <int> <dbl>
## 1  0.34 Ideal     D      1272  4.49
## 2  0.55 Premium   F      1354  5.26
## 3  0.23 Very Good E       402  4.01
## 4  0.31 Premium   H       558  4.34
## 5  1.11 Ideal     F      7823  6.62
## 6  0.23 Very Good E       485  4.02
```

```
diamonds %>%
  filter(cut == "premium")
```

```
## # A tibble: 0 x 5
## # … with 5 variables: carat <dbl>, cut <ord>, color <ord>, price <int>, x <dbl>
```

Cautionary tales

```
head(diamonds)
```

```
## # A tibble: 6 x 5
##   carat cut       color price     x
##   <dbl> <ord>     <ord> <int> <dbl>
## 1  0.34 Ideal     D      1272  4.49
## 2  0.55 Premium   F      1354  5.26
## 3  0.23 Very Good E       402  4.01
## 4  0.31 Premium   H       558  4.34
## 5  1.11 Ideal     F      7823  6.62
## 6  0.23 Very Good E       485  4.02
```

```
diamonds %>%
  filter(cut == "Premuim")
```

```
## # A tibble: 0 x 5
## # … with 5 variables: carat <dbl>, cut <ord>, color <ord>, price <int>, x <dbl>
```

Cautionary tales

```
head(diamonds)
```

```
## # A tibble: 6 x 5
##   carat cut       color price     x
##   <dbl> <ord>     <ord> <int> <dbl>
## 1  0.34 Ideal     D      1272  4.49
## 2  0.55 Premium   F      1354  5.26
## 3  0.23 Very Good E       402  4.01
## 4  0.31 Premium   H       558  4.34
## 5  1.11 Ideal     F      7823  6.62
## 6  0.23 Very Good E       485  4.02
```

```
diamonds %>%
  filter(quality == "Premium")
```

```
## Error: Problem with `filter()` input `..1`.
## ✖ object 'quality' not found
## ℹ Input `..1` is `quality == "Premium"`.
```

Cautionary tales

```
head(diamonds)
```

```
## # A tibble: 6 x 5
##   carat cut       color price     x
##   <dbl> <ord>     <ord> <int> <dbl>
## 1  0.34 Ideal     D      1272  4.49
## 2  0.55 Premium   F      1354  5.26
## 3  0.23 Very Good E       402  4.01
## 4  0.31 Premium   H       558  4.34
## 5  1.11 Ideal     F      7823  6.62
## 6  0.23 Very Good E       485  4.02
```

```
diamonds %>%
  filter(CUT == "Premium")
```

```
## Error: Problem with `filter()` input `..1`.
## ✖ object 'CUT' not found
## ℹ Input `..1` is `CUT == "Premium"`.
```

Cautionary tales

```
head(diamonds)
```

```
## # A tibble: 6 x 5
##   carat cut       color price     x
##   <dbl> <ord>     <ord> <int> <dbl>
## 1  0.34 Ideal     D      1272  4.49
## 2  0.55 Premium   F      1354  5.26
## 3  0.23 Very Good E       402  4.01
## 4  0.31 Premium   H       558  4.34
## 5  1.11 Ideal     F      7823  6.62
## 6  0.23 Very Good E       485  4.02
```

```
diamonds %>%
  filter(cut = "Premium")
```

```
## Error: Problem with `filter()` input `..1`.
## ✖ Input `..1` is named.
## ℹ This usually means that you've used `=` instead of `==`.
## ℹ Did you mean `cut == "Premium"`?
```

Cautionary tales

```
head(diamonds)
```

```
## # A tibble: 6 x 5
##   carat cut       color price     x
##   <dbl> <ord>     <ord> <int> <dbl>
## 1  0.34 Ideal     D      1272  4.49
## 2  0.55 Premium   F      1354  5.26
## 3  0.23 Very Good E       402  4.01
## 4  0.31 Premium   H       558  4.34
## 5  1.11 Ideal     F      7823  6.62
## 6  0.23 Very Good E       485  4.02
```

```
diamonds
  %>% filter(cut == "Premium")
```

```
## Error: <text>:2:3: unexpected SPECIAL
## 1: diamonds
## 2:   %>%
##      ^
```

We can have multiple conditions in `filter()`

- Using a comma in filter acts as "and" (`&`).
- Keep all rows where ALL conditions are `TRUE`

```
diamonds %>%
  filter(cut == "Premium", color == "I")
```

```
## # A tibble: 3 x 5
##   carat cut     color price     x
##   <dbl> <ord>   <ord> <int> <dbl>
## 1  0.34 Premium I       765  4.49
## 2  1.35 Premium I      5715  7.26
## 3  1.02 Premium I      4113  6.48
```

We can have multiple conditions in `filter()`

```
diamonds %>%
  filter(cut == "Premium", color == "I", carat > 1)
```

```
## # A tibble: 2 x 5
##   carat cut     color price     x
##   <dbl> <ord>   <ord> <int> <dbl>
## 1  1.35 Premium I      5715  7.26
## 2  1.02 Premium I      4113  6.48
```

Separate onto new lines when code gets too long

```
diamonds %>%
  filter(cut == "Premium",
         color == "I",
         carat > 1)
```

```
## # A tibble: 2 x 5
##   carat cut     color price     x
##   <dbl> <ord>   <ord> <int> <dbl>
## 1  1.35 Premium I      5715  7.26
## 2  1.02 Premium I      4113  6.48
```

Caution!! Make sure the commas are always on at the *end of a line*

Goal: Subset to only "Ideal and "Premium" diamonds

```
diamonds %>%
  filter(cut == "Ideal",
         cut == "Premium")
```

```
## # A tibble: 0 x 5
## # … with 5 variables: carat <dbl>, cut <ord>, color <ord>, price <int>, x <dbl>
```

That code DID WORK. It retained all rows whose `cut` equaled "Ideal" **AND** "Premium"! ....But there are no such rows!!

To reiterate: The code worked perfectly fine, but it didn't do what you might have expected. This is a type of a bug.

Recall the `%in%` operator

```
colors <- c("red", "orange", "yellow")
"red" %in% colors
```

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

```
"blue" %in% colors
```

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

Get rows with *either* cut using `%in%`, NOT `==`

```
diamonds %>%
  filter(cut %in% c("Ideal", "Premium"))
```

```
## # A tibble: 15 x 5
##    carat cut     color price     x
##    <dbl> <ord>   <ord> <int> <dbl>
##  1  0.34 Ideal   D      1272  4.49
##  2  0.55 Premium F      1354  5.26
##  3  0.31 Premium H       558  4.34
##  4  1.11 Ideal   F      7823  6.62
##  5  1.32 Ideal   H      9572  6.98
##  6  0.35 Ideal   E       767  4.52
##  7  0.34 Premium I       765  4.49
##  8  1.35 Premium I      5715  7.26
##  9  1.41 Ideal   G     11009  7.22
## 10  1.02 Premium I      4113  6.48
## 11  1.51 Premium H      9762  7.54
## 12  0.73 Ideal   H      3463  5.8
## 13  0.72 Ideal   F      2879  5.76
## 14  0.59 Ideal   E      1607  5.36
## 15  0.3  Ideal   G       684  4.31
```

Helpful tips

```
# Avoid hardcoding by defining this separately
cuts <- c("Ideal", "Premium")
diamonds %>%
  filter(cut %in% cuts)
```

Below might seem like it works, but it DOES NOT!

```
diamonds %>%
  # Why is this NO NO NO NO NO NO NO????
  filter(cut == c("Ideal", "Premium"))
```

```
## # A tibble: 11 x 5
##    carat cut     color price     x
##    <dbl> <ord>   <ord> <int> <dbl>
##  1  0.34 Ideal   D      1272  4.49
##  2  0.55 Premium F      1354  5.26
##  3  0.31 Premium H       558  4.34
##  4  1.11 Ideal   F      7823  6.62
##  5  1.32 Ideal   H      9572  6.98
##  6  1.35 Premium I      5715  7.26
##  7  1.41 Ideal   G     11009  7.22
##  8  1.02 Premium I      4113  6.48
##  9  0.73 Ideal   H      3463  5.8
## 10  0.72 Ideal   F      2879  5.76
## 11  0.59 Ideal   E      1607  5.36
```

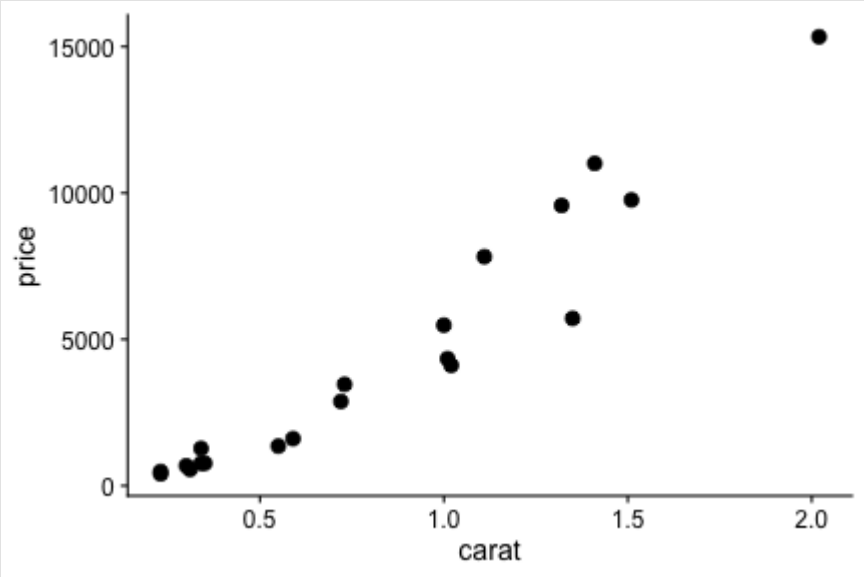
Have as many options as you want:

```
diamonds %>%
  filter(cut %in% c("Ideal", "Premium", "Good"))
```

```
## # A tibble: 16 x 5
##    carat cut     color price     x
##    <dbl> <ord>   <ord> <int> <dbl>
##  1  0.34 Ideal   D      1272  4.49
##  2  0.55 Premium F      1354  5.26
##  3  0.31 Premium H       558  4.34
##  4  1.11 Ideal   F      7823  6.62
##  5  1.32 Ideal   H      9572  6.98
##  6  0.35 Ideal   E       767  4.52
##  7  0.34 Premium I       765  4.49
##  8  1.35 Premium I      5715  7.26
##  9  1.41 Ideal   G     11009  7.22
## 10  1.02 Premium I      4113  6.48
## 11  1.51 Premium H      9762  7.54
## 12  1    Good    G      5484  6.38
## 13  0.73 Ideal   H      3463  5.8
## 14  0.72 Ideal   F      2879  5.76
## 15  0.59 Ideal   E      1607  5.36
## 16  0.3  Ideal   G       684  4.31
```

One reason filtering is important:

```
ggplot(diamonds, aes(x = carat,
                     y = price)) +
  geom_point(size = 3)
```

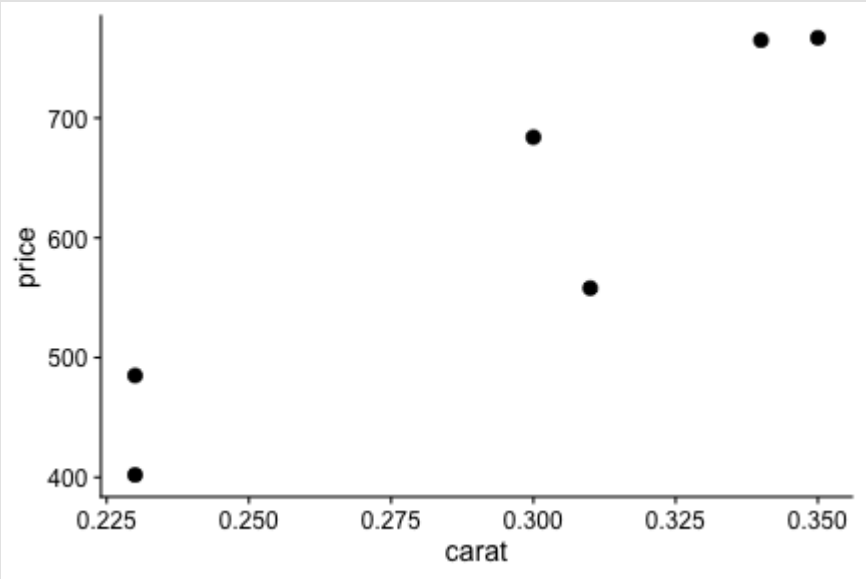


But what if I *only* want to plot diamonds that cost less than $1000?

Plotting filtered data

```
diamonds %>%
  filter(price < 1000) -> diamonds_cheaper

ggplot(diamonds_cheaper, aes(x = carat, y = price)) +
  geom_point(size = 3)
```

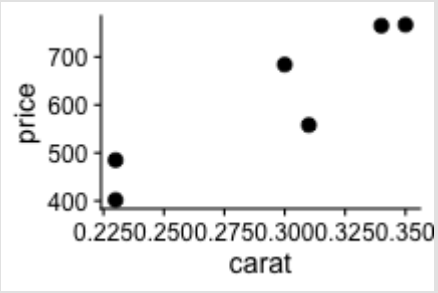


Plotting filtered data

```
diamonds %>%
  filter(price < 1000) -> diamonds_cheaper

ggplot(diamonds_cheaper, aes(x = carat, y = price)) +
  geom_point(size = 3)
```

```
diamonds %>%
  filter(price < 1000) %>%
  ggplot(aes(x = carat, y = pric
    geom_point(size = 3)
```



Caution: Be careful to use `%>%` and `+` in the right place. Using the wrong one in the wrong place is a **common bug**!

Working with columns with `select()`

```
diamonds %>%
  select(price, cut)
```

```
## # A tibble: 20 x 2
##    price cut
##    <int> <ord>
##  1  1272 Ideal
##  2  1354 Premium
##  3   402 Very Good
##  4   558 Premium
##  5  7823 Ideal
##  6   485 Very Good
##  7  9572 Ideal
##  8   767 Ideal
##  9   765 Premium
## 10  5715 Premium
## 11 11009 Ideal
## 12  4113 Premium
## 13  9762 Premium
## 14  5484 Good
## 15  3463 Ideal
## 16  4338 Very Good
## 17  2879 Ideal
## 18 15334 Very Good
## 19  1607 Ideal
## 20   684 Ideal
```

```
diamonds %>%
  select(-price, -cut)
```

```
## # A tibble: 20 x 3
##    carat color     x
##    <dbl> <ord> <dbl>
##  1  0.34 D      4.49
##  2  0.55 F      5.26
##  3  0.23 E      4.01
##  4  0.31 H      4.34
##  5  1.11 F      6.62
##  6  0.23 E      4.02
##  7  1.32 H      6.98
##  8  0.35 E      4.52
##  9  0.34 I      4.49
## 10  1.35 I      7.26
## 11  1.41 G      7.22
## 12  1.02 I      6.48
## 13  1.51 H      7.54
## 14  1    G      6.38
## 15  0.73 H      5.8
## 16  1.01 D      6.43
## 17  0.72 F      5.76
## 18  2.02 D      8.19
## 19  0.59 E      5.36
## 20  0.3  G      4.31
```

There is a lot of magic in `select()`

```
head(diamonds)
```

```
## # A tibble: 6 x 5
##   carat cut       color price     x
##   <dbl> <ord>     <ord> <int> <dbl>
## 1  0.34 Ideal     D      1272  4.49
## 2  0.55 Premium   F      1354  5.26
## 3  0.23 Very Good E       402  4.01
## 4  0.31 Premium   H       558  4.34
## 5  1.11 Ideal     F      7823  6.62
## 6  0.23 Very Good E       485  4.02
```

```
diamonds %>%
  select(price, everything())
```

```
## # A tibble: 20 x 5
##    price carat cut       color     x
##    <int> <dbl> <ord>     <ord> <dbl>
##  1  1272  0.34 Ideal     D      4.49
##  2  1354  0.55 Premium   F      5.26
##  3   402  0.23 Very Good E      4.01
##  4   558  0.31 Premium   H      4.34
##  5  7823  1.11 Ideal     F      6.62
##  6   485  0.23 Very Good E      4.02
##  7  9572  1.32 Ideal     H      6.98
##  8   767  0.35 Ideal     E      4.52
```

Creating new columns with `mutate()`

```
diamonds %>%
  mutate(the_number_5 = 5)
```

```
## # A tibble: 20 x 6
##    carat cut       color price     x the_number_5
##    <dbl> <ord>     <ord> <int> <dbl>        <dbl>
##  1  0.34 Ideal     D      1272  4.49            5
##  2  0.55 Premium   F      1354  5.26            5
##  3  0.23 Very Good E       402  4.01            5
##  4  0.31 Premium   H       558  4.34            5
##  5  1.11 Ideal     F      7823  6.62            5
##  6  0.23 Very Good E       485  4.02            5
##  7  1.32 Ideal     H      9572  6.98            5
##  8  0.35 Ideal     E       767  4.52            5
##  9  0.34 Premium   I       765  4.49            5
## 10  1.35 Premium   I      5715  7.26            5
## 11  1.41 Ideal     G     11009  7.22            5
## 12  1.02 Premium   I      4113  6.48            5
## 13  1.51 Premium   H      9762  7.54            5
## 14  1    Good      G      5484  6.38            5
## 15  0.73 Ideal     H      3463  5.8             5
## 16  1.01 Very Good D      4338  6.43            5
## 17  0.72 Ideal     F      2879  5.76            5
## 18  2.02 Very Good D     15334  8.19            5
## 19  0.59 Ideal     E      1607  5.36            5
## 20  0.3  Ideal     G       684  4.31            5
```

Creating new columns with `mutate()`

Again, we can directly reference existing columns:

```
# 'x' is measured in mm
diamonds %>%
  mutate(x_in_cm = x/10)
```

```
## # A tibble: 20 x 6
##    carat cut       color price     x x_in_cm
##    <dbl> <ord>     <ord> <int> <dbl>   <dbl>
##  1  0.34 Ideal     D      1272  4.49   0.449
##  2  0.55 Premium   F      1354  5.26   0.526
##  3  0.23 Very Good E       402  4.01   0.401
##  4  0.31 Premium   H       558  4.34   0.434
##  5  1.11 Ideal     F      7823  6.62   0.662
##  6  0.23 Very Good E       485  4.02   0.402
##  7  1.32 Ideal     H      9572  6.98   0.698
##  8  0.35 Ideal     E       767  4.52   0.452
##  9  0.34 Premium   I       765  4.49   0.449
## 10  1.35 Premium   I      5715  7.26   0.726
## 11  1.41 Ideal     G     11009  7.22   0.722
## 12  1.02 Premium   I      4113  6.48   0.648
## 13  1.51 Premium   H      9762  7.54   0.754
## 14  1    Good      G      5484  6.38   0.638
## 15  0.73 Ideal     H      3463  5.8    0.580
## 16  1.01 Very Good D      4338  6.43   0.643
## 17  0.72 Ideal     F      2879  5.76   0.576
```

What if I want my new variable to be first?

```
# 'x' is measured in mm
diamonds %>%
  mutate(x_in_cm = x/10) %>%
  select(x_in_cm, everything())
```

```
## # A tibble: 20 x 6
##    x_in_cm carat cut       color price     x
##      <dbl> <dbl> <ord>     <ord> <int> <dbl>
##  1   0.449  0.34 Ideal     D      1272  4.49
##  2   0.526  0.55 Premium   F      1354  5.26
##  3   0.401  0.23 Very Good E       402  4.01
##  4   0.434  0.31 Premium   H       558  4.34
##  5   0.662  1.11 Ideal     F      7823  6.62
##  6   0.402  0.23 Very Good E       485  4.02
##  7   0.698  1.32 Ideal     H      9572  6.98
##  8   0.452  0.35 Ideal     E       767  4.52
##  9   0.449  0.34 Premium   I       765  4.49
## 10   0.726  1.35 Premium   I      5715  7.26
## 11   0.722  1.41 Ideal     G     11009  7.22
## 12   0.648  1.02 Premium   I      4113  6.48
## 13   0.754  1.51 Premium   H      9762  7.54
## 14   0.638  1    Good      G      5484  6.38
## 15   0.580  0.73 Ideal     H      3463  5.8
## 16   0.643  1.01 Very Good D      4338  6.43
## 17   0.576  0.72 Ideal     F      2879  5.76
## 18   0.819  2.02 Very Good D     15334  8.19
## 19   0.536  0.59 Ideal     E      1607  5.36
```

Also bring `x` forward for easy comparison

> This is part of checking that your `mutate()` command worked. Use your eyes to confirm that `x_in_cm` really is 10*x

```
# 'x' is measured in mm
diamonds %>%
  mutate(x_in_cm = x/10) %>%
  select(x_in_cm, x, everything())
```

```
## # A tibble: 20 x 6
##    x_in_cm     x carat cut       color price
##      <dbl> <dbl> <dbl> <ord>     <ord> <int>
##  1   0.449  4.49  0.34 Ideal     D      1272
##  2   0.526  5.26  0.55 Premium   F      1354
##  3   0.401  4.01  0.23 Very Good E       402
##  4   0.434  4.34  0.31 Premium   H       558
##  5   0.662  6.62  1.11 Ideal     F      7823
##  6   0.402  4.02  0.23 Very Good E       485
##  7   0.698  6.98  1.32 Ideal     H      9572
##  8   0.452  4.52  0.35 Ideal     E       767
##  9   0.449  4.49  0.34 Premium   I       765
## 10   0.726  7.26  1.35 Premium   I      5715
## 11   0.722  7.22  1.41 Ideal     G     11009
## 12   0.648  6.48  1.02 Premium   I      4113
## 13   0.754  7.54  1.51 Premium   H      9762
## 14   0.638  6.38  1    Good      G      5484
## 15   0.580  5.8   0.73 Ideal     H      3463
```

This the magic of the pipe

```
diamonds %>%
  filter(carat < 0.5) %>%
  select(carat, price)
```

```
## # A tibble: 7 x 2
##   carat price
##   <dbl> <int>
## 1  0.34  1272
## 2  0.23   402
## 3  0.31   558
## 4  0.23   485
## 5  0.35   767
## 6  0.34   765
## 7  0.3    684
```

ONE LINE AT A TIME

IF YOU DO NOT GO ONE LINE AT A TIME, YOU WILL DO VERY VERY POORLY.

I REALLY CANNOT EMPHASIZE THIS ENOUGH.

ONE. LINE. AT. A TIME.

Make sure this works FIRST

```
diamonds %>%
  filter(carat < 0.5)
```

```
## # A tibble: 7 x 5
##   carat cut       color price     x
##   <dbl> <ord>     <ord> <int> <dbl>
## 1  0.34 Ideal     D      1272  4.49
## 2  0.23 Very Good E       402  4.01
## 3  0.31 Premium   H       558  4.34
## 4  0.23 Very Good E       485  4.02
## 5  0.35 Ideal     E       767  4.52
## 6  0.34 Premium   I       765  4.49
## 7  0.3  Ideal     G       684  4.31
```

Then, add the next line

```
diamonds %>%
  filter(carat < 0.5) %>%
  select(carat, price)
```

```
## # A tibble: 7 x 2
##   carat price
##   <dbl> <int>
## 1  0.34  1272
## 2  0.23   402
## 3  0.31   558
## 4  0.23   485
## 5  0.35   767
## 6  0.34   765
## 7  0.3    684
```

An smallish example of real-word piping

> This code should not make sense to you, don't worry!

```
models %>%
  separate(name, into=c("id", "dataset", "trash"), sep = "\\.") %>%
  replace_na(list(dataset = "PANDIT"))  %>%
  select(-trash) %>%
  group_by(id, datatype) %>%
  mutate(num = 1:n()) %>%
  ungroup() %>%
  pivot_longer(AIC:BIC,
               names_to = "ic_type",
               values_to = "best_model") %>%
  mutate(best_matrix = str_replace(best_model, "\\+.+", "")) -> proce
```

So far...

- `glimpse()` for data frame overviews
- `filter()` for subsetting rows (observations)
- `select()` for keeping/removing columns
- `mutate()` for creating columns

- Other `dplyr` functions you will see in exercises this week (and will be on the homework)
  - `arrange()`
  - `rename()`
  - `distinct()`
  - `tally()` (we will learn more about this function next week, too)

Arranging rows by a variable

```
diamonds %>%
  arrange(carat)
```

```
## # A tibble: 20 x 5
##    carat cut       color price     x
##    <dbl> <ord>     <ord> <int> <dbl>
##  1  0.23 Very Good E       402  4.01
##  2  0.23 Very Good E       485  4.02
##  3  0.3  Ideal     G       684  4.31
##  4  0.31 Premium   H       558  4.34
##  5  0.34 Ideal     D      1272  4.49
##  6  0.34 Premium   I       765  4.49
##  7  0.35 Ideal     E       767  4.52
##  8  0.55 Premium   F      1354  5.26
##  9  0.59 Ideal     E      1607  5.36
## 10  0.72 Ideal     F      2879  5.76
## 11  0.73 Ideal     H      3463  5.8
## 12  1    Good      G      5484  6.38
## 13  1.01 Very Good D      4338  6.43
## 14  1.02 Premium   I      4113  6.48
## 15  1.11 Ideal     F      7823  6.62
## 16  1.32 Ideal     H      9572  6.98
## 17  1.35 Premium   I      5715  7.26
## 18  1.41 Ideal     G     11009  7.22
## 19  1.51 Premium   H      9762  7.54
## 20  2.02 Very Good D     15334  8.19
```

For *descending* order:

```
diamonds %>%
  arrange(desc(carat))
```

```
## # A tibble: 20 x 5
##    carat cut       color price     x
##    <dbl> <ord>     <ord> <int> <dbl>
##  1  2.02 Very Good D     15334  8.19
##  2  1.51 Premium   H      9762  7.54
##  3  1.41 Ideal     G     11009  7.22
##  4  1.35 Premium   I      5715  7.26
##  5  1.32 Ideal     H      9572  6.98
##  6  1.11 Ideal     F      7823  6.62
##  7  1.02 Premium   I      4113  6.48
##  8  1.01 Very Good D      4338  6.43
##  9  1    Good      G      5484  6.38
## 10  0.73 Ideal     H      3463  5.8
## 11  0.72 Ideal     F      2879  5.76
## 12  0.59 Ideal     E      1607  5.36
## 13  0.55 Premium   F      1354  5.26
## 14  0.35 Ideal     E       767  4.52
## 15  0.34 Ideal     D      1272  4.49
## 16  0.34 Premium   I       765  4.49
## 17  0.31 Premium   H       558  4.34
## 18  0.3  Ideal     G       684  4.31
## 19  0.23 Very Good E       402  4.01
## 20  0.23 Very Good E       485  4.02
```

Renaming columns

```
# rename(newname = oldname) !!!
diamonds %>%
  rename(width = x)
```

```
## # A tibble: 20 x 5
##    carat cut       color price width
##    <dbl> <ord>     <ord> <int> <dbl>
##  1  0.34 Ideal     D      1272  4.49
##  2  0.55 Premium   F      1354  5.26
##  3  0.23 Very Good E       402  4.01
##  4  0.31 Premium   H       558  4.34
##  5  1.11 Ideal     F      7823  6.62
##  6  0.23 Very Good E       485  4.02
##  7  1.32 Ideal     H      9572  6.98
##  8  0.35 Ideal     E       767  4.52
##  9  0.34 Premium   I       765  4.49
## 10  1.35 Premium   I      5715  7.26
## 11  1.41 Ideal     G     11009  7.22
## 12  1.02 Premium   I      4113  6.48
## 13  1.51 Premium   H      9762  7.54
## 14  1    Good      G      5484  6.38
## 15  0.73 Ideal     H      3463  5.8
## 16  1.01 Very Good D      4338  6.43
## 17  0.72 Ideal     F      2879  5.76
## 18  2.02 Very Good D     15334  8.19
## 19  0.59 Ideal     E      1607  5.36
## 20  0.3  Ideal     G       684  4.31
```

Removing duplicate rows

```
# boring, since all rows are unique!
diamonds %>%
  distinct()
```

```
## # A tibble: 20 x 5
##    carat cut       color price     x
##    <dbl> <ord>     <ord> <int> <dbl>
##  1  0.34 Ideal     D      1272  4.49
##  2  0.55 Premium   F      1354  5.26
##  3  0.23 Very Good E       402  4.01
##  4  0.31 Premium   H       558  4.34
##  5  1.11 Ideal     F      7823  6.62
##  6  0.23 Very Good E       485  4.02
##  7  1.32 Ideal     H      9572  6.98
##  8  0.35 Ideal     E       767  4.52
##  9  0.34 Premium   I       765  4.49
## 10  1.35 Premium   I      5715  7.26
## 11  1.41 Ideal     G     11009  7.22
## 12  1.02 Premium   I      4113  6.48
## 13  1.51 Premium   H      9762  7.54
## 14  1    Good      G      5484  6.38
## 15  0.73 Ideal     H      3463  5.8
## 16  1.01 Very Good D      4338  6.43
## 17  0.72 Ideal     F      2879  5.76
## 18  2.02 Very Good D     15334  8.19
## 19  0.59 Ideal     E      1607  5.36
## 20  0.3  Ideal     G       684  4.31
```

Removing duplicate rows

```
diamonds %>%
  select(color)
```

```
## # A tibble: 20 x 1
##    color
##    <ord>
##  1 D
##  2 F
##  3 E
##  4 H
##  5 F
##  6 E
##  7 H
##  8 E
##  9 I
## 10 I
## 11 G
## 12 I
## 13 H
## 14 G
## 15 H
## 16 D
## 17 F
## 18 D
## 19 E
## 20 G
```

```
diamonds %>%
  select(color) %>%
  distinct()
```

```
## # A tibble: 6 x 1
##   color
##   <ord>
## 1 D
## 2 F
## 3 E
## 4 H
## 5 I
## 6 G
```

Counting the number of rows with code

```
# Returns a number
nrow(diamonds)
```

```
## [1] 20
```

```
# Returns a number
diamonds %>%
  nrow()
```

```
## [1] 20
```

```
# Returns a TIBBLE
diamonds %>%
  tally()
```

```
## # A tibble: 1 x 1
##       n
##   <int>
## 1    20
```