R4DS

Cohort 4

Wed 6:00 – 7:00 US Central

Twitter: @Rspjut

5-MINUTE ICE BREAKER

What is some good news you got recently?

AGENDA

- 5-Minute Ice breaker
- Quick Housekeeping Reminders
- Recap of Chapter 7
- Finish Chapter 7
- Next Week
- Getting Help

QUICK HOUSEKEEPING REMINDERS

- Video camera is optional, but encouraged.
- I purposely err on the side of going fast. Slowing me down does not hurt my feelings.
- Take time to learn the theory (Grammar of Graphics, Tidy Data whitepaper, Relational Database theory, Appropriate Visualization Types, etc.). data-to-viz.com
- Please do the chapter exercises. Second-best learning opportunity!
- Please plan on teaching one of the lessons. Best learning opportunity!

DATASET USED IN CHAPTER 7: DIAMONDS

Diamonds (load `tidyverse` then `?diamonds`)

Variable	Format
price	Price in US dollars
carat	Weight of the diamond (0.2 – 5.01)
cut	Quality of the cut (Fair, Good, Very Good, Premium, Ideal)
color	Diamond color from D (best) to J (worst)
clarity	How clear. Worst = I1, SI2, SI1, VS2, VS1, VVS2, VVS1, IF
x	Length in mm
y	Width in mm
z	Depth in mm
depth	Depth percentage
table	Width of top of diamond relative to widest point

`head(diamonds)`

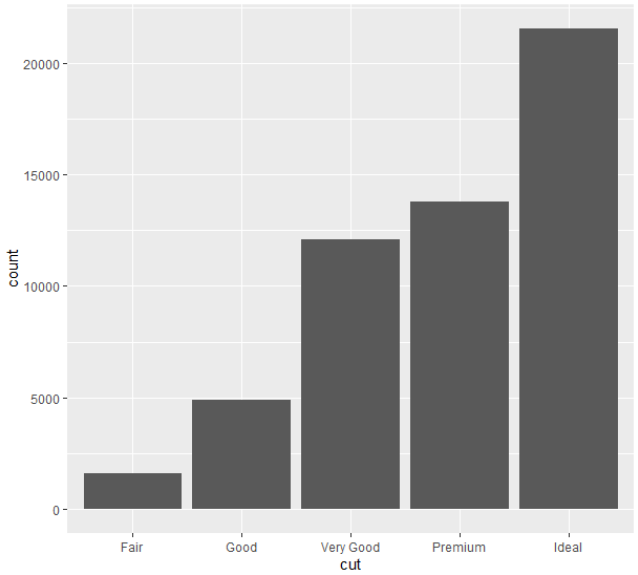
```
> head(diamonds)
# A tibble: 6 x 10
  carat cut       color clarity depth table price     x     y     z
  <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
1 0.23  Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
2 0.21  Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
3 0.23  Good      E     VS1      56.9    65   327  4.05  4.07  2.31
4 0.290 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
5 0.31  Good      J     SI2      63.3    58   335  4.34  4.35  2.75
6 0.24  Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
```

Total Records = 53,940

7.3.1 VISUALIZING DISTRIBUTIONS

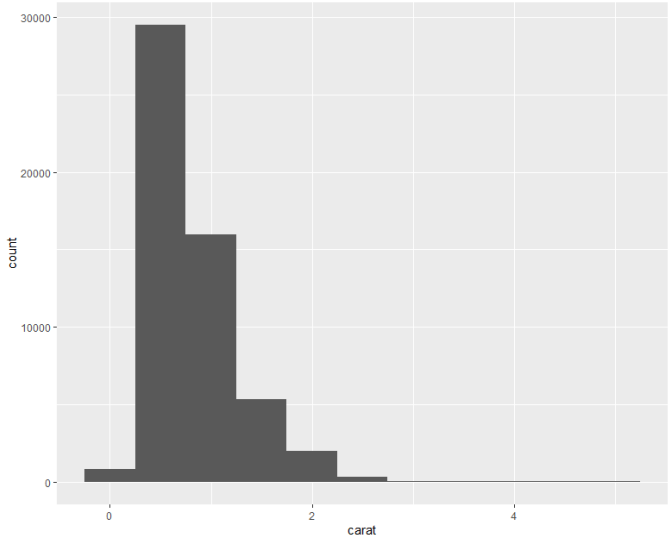
Using `diamonds`, let's visualize the number of diamonds that belong to each value of the `cut` variable.

```
ggplot(data = diamonds) + geom_bar(mapping = aes(x = cut))
```



Using `diamonds`, let's visualize the number of diamonds that belong to each value of the `carat` variable.

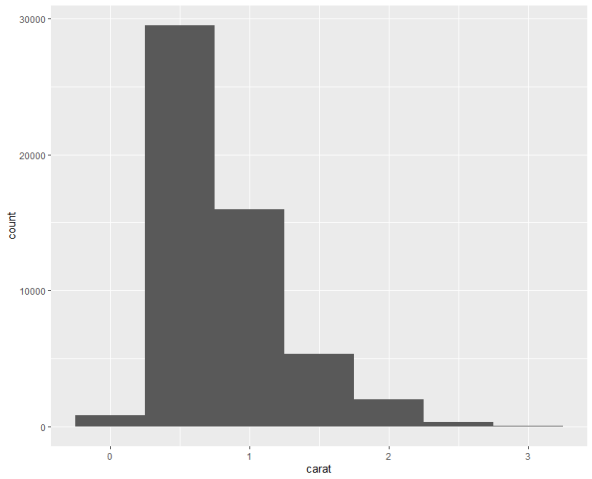
```
ggplot(data = diamonds) +
    geom_histogram(mapping = aes(x = carat), binwidth = 0.5)
```



7.3.1 VISUALIZING DISTRIBUTIONS

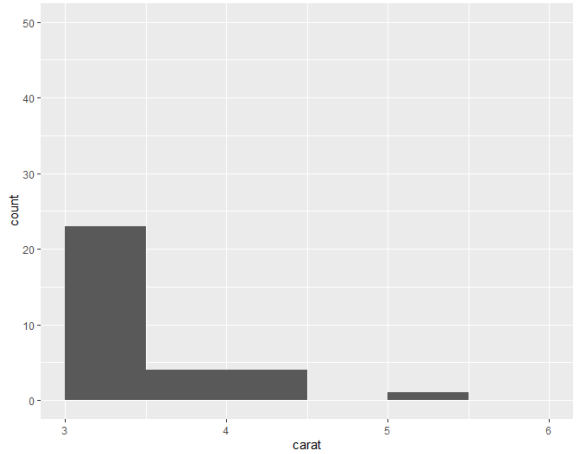
Let's filter our dataset to only include diamonds under 3.0 carats.

```
diamonds %>%
filter(carat < 3) %>%
ggplot() + geom_histogram(mapping = aes(x = carat), binwidth = 0.5)
```



Zoom in on just the 32 diamonds greater than 3.0 carats. Use xlim and ylim to cut the axes to the intervals you specify.

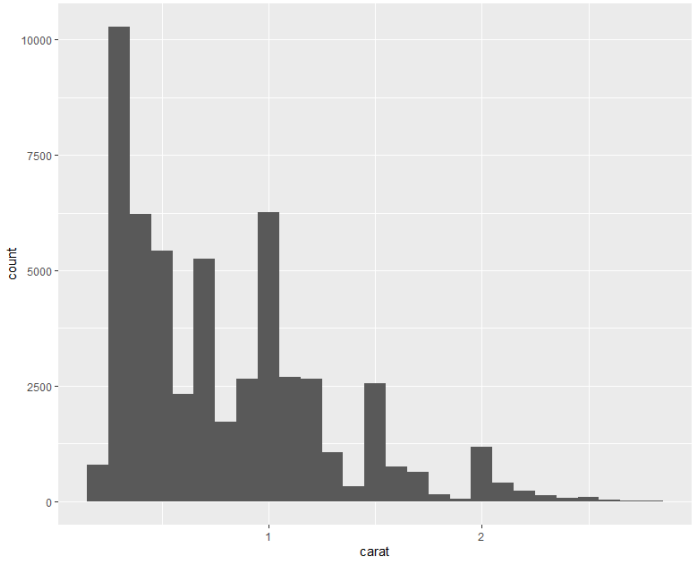
```
diamonds %>%
filter(carat > 3) %>%
ggplot(data = diamonds) +
    geom_histogram(mapping = aes(x = carat),
    binwidth = 0.5, boundary = 0)  +
    xlim(3, 6) + ylim(0, 50)
```



7.3.1 VISUALIZING DISTRIBUTIONS

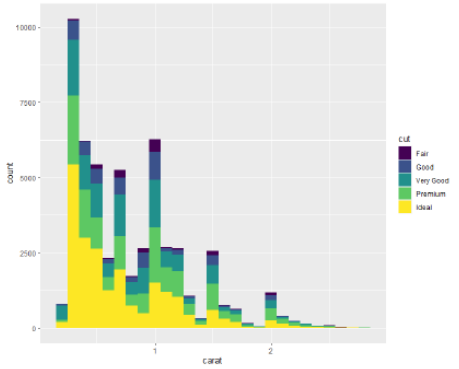
What if you reduce the `binwidth` from `0.5` to `0.1`?

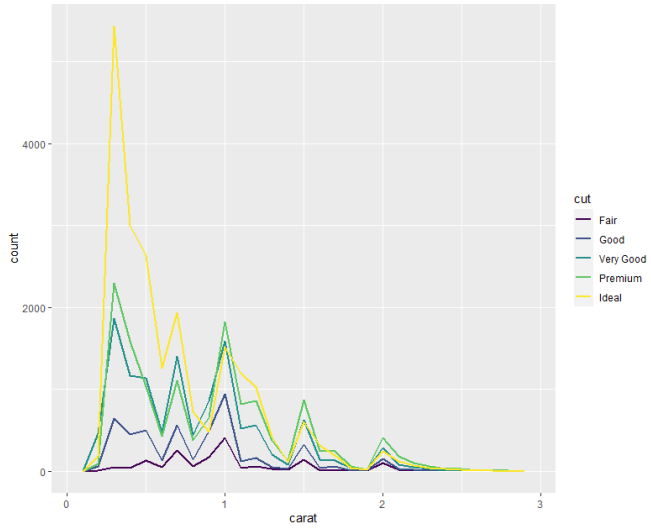
```
diamonds %>%
filter(carat < 3) %>%
ggplot() + geom_histogram(mapping = aes(x = carat), binwidth = 0.1)
```



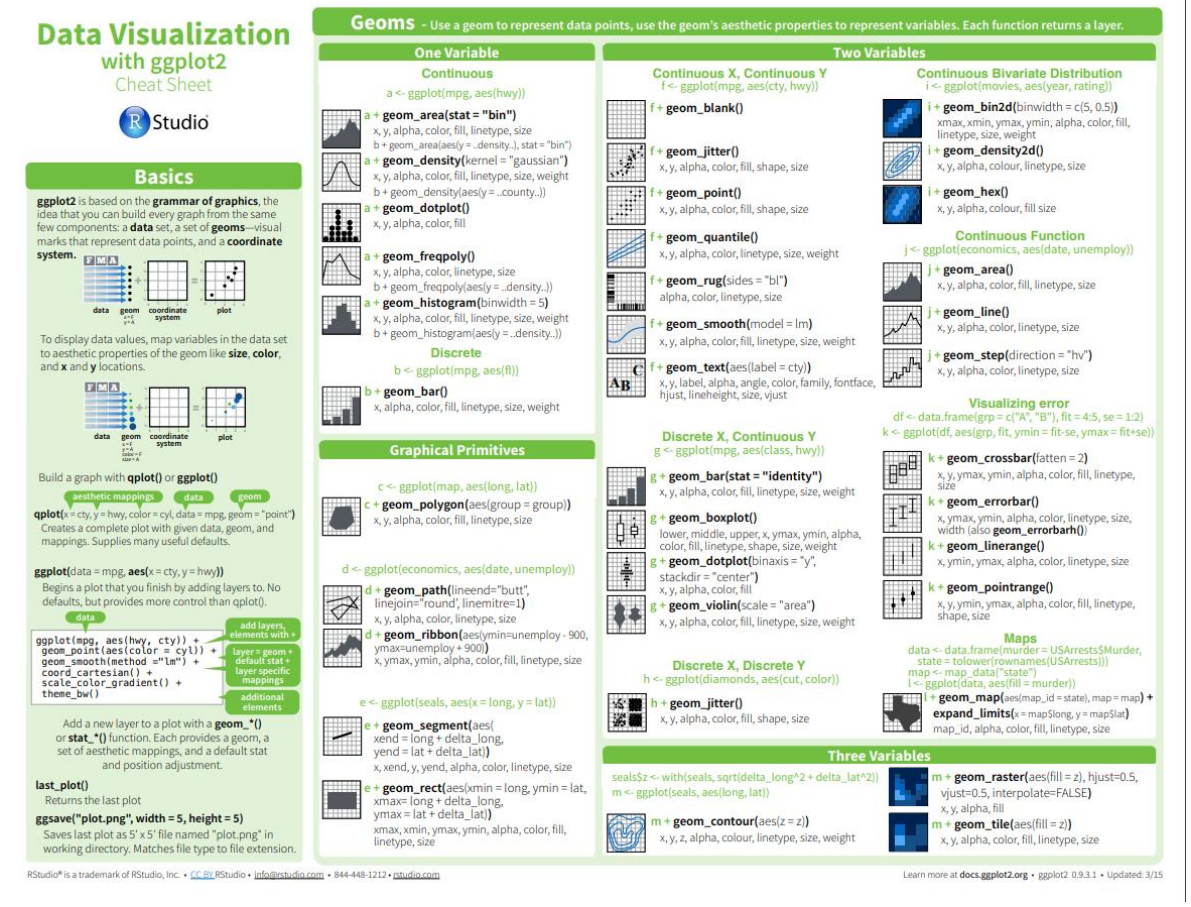
The `freqpoly` visualization is better.
The aesthetic name is `color` instead of `fill`.

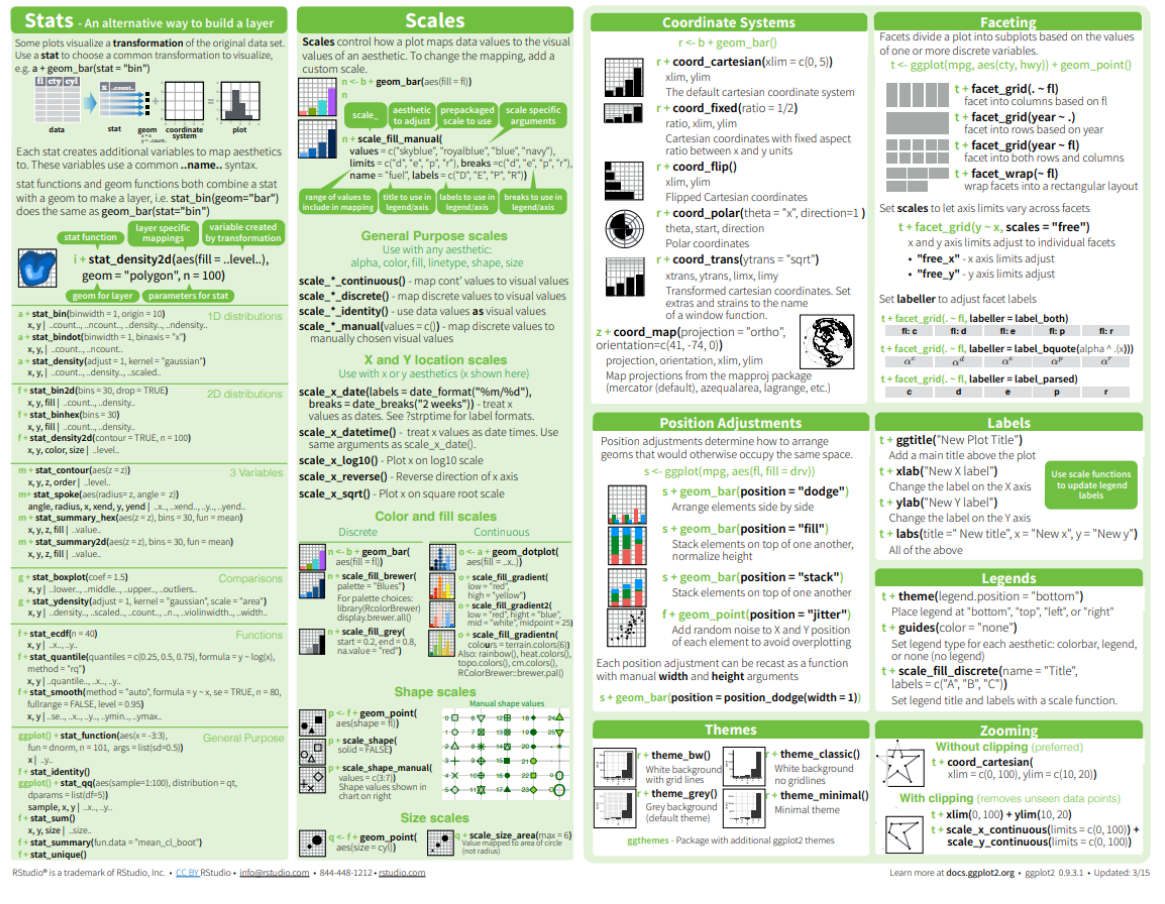
```
diamonds %>%
filter(carat < 3) %>%
ggplot() + geom_freqpoly(mapping = aes(x = carat, color = cut), binwidth = 0.1)
```





7.3.1 GGPLOT CHEATSHEET





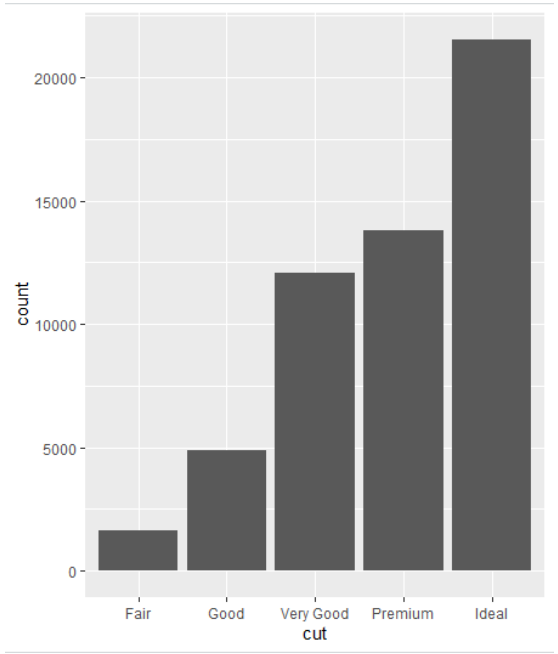
GGPLOT + GEOM(STAT)

`geom_bar` default stat is count. So bars represent the number of observations in each value of `cut` (i.e., the count)

```
diamonds %>%
  count(cut)
```

```
# A tibble: 5 x 2
  cut           n
  <ord>     <int>
1 Fair       1610
2 Good       4906
3 Very Good 12082
4 Premium   13791
5 Ideal     21551
```

```
ggplot(diamonds) +
  geom_bar(aes(x = cut))
```



Say, instead, you want to add up the number of carats in each value of cut (i.e., how many carats are `cut = Fair`, how many carats are `cut = Good`, etc.)

```
diamonds %>%
  group_by(cut) %>%
  summarize(total_carats = sum(carat))
```

```
# A tibble: 5 x 2
  cut       total_carats
  <ord>            <dbl>
1 Fair             1684.
2 Good             4166.
3 Very Good        9743.
4 Premium         12301.
5 Ideal           15147.
```

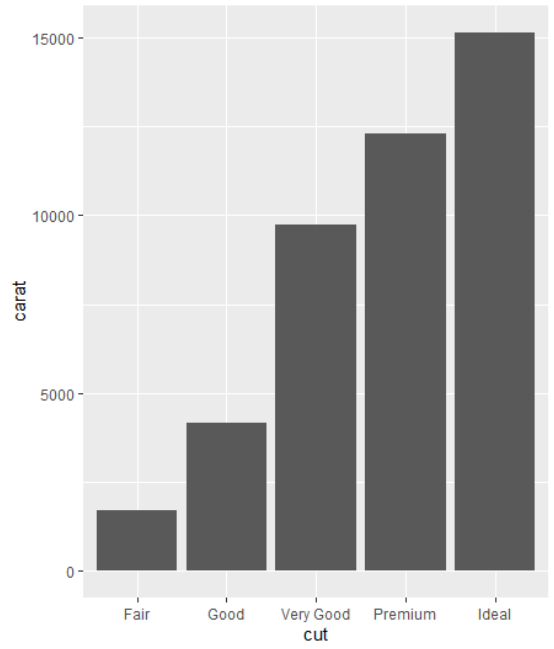
Discrete X, Continuous Y

g <- ggplot(mpg, aes(class, hwy))

g + **geom_bar(stat = "identity")**

x, y, alpha, color, fill, linetype, size, weight

```
ggplot(diamonds) +
  geom_bar(
      aes(x = cut, y = carat),
      stat = "identity")
```

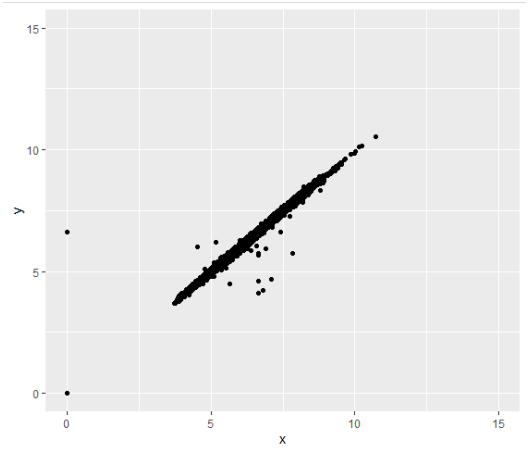


7.3 EXERCISES

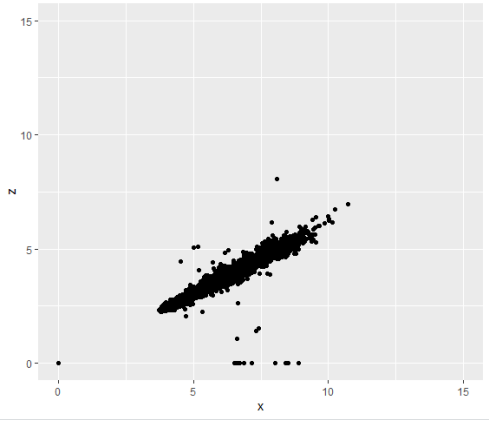
7.3.4 Exercises

1. Explore the distribution of each of the `x`, `y`, and `z` variables in `diamonds`. What do you learn? Think about a diamond and how you might decide which dimension is the length, width, and depth.

```
diamonds %>%
  ggplot(aes(x,y)) + geom_point() + xlim(0,15) + ylim(0,15)
```



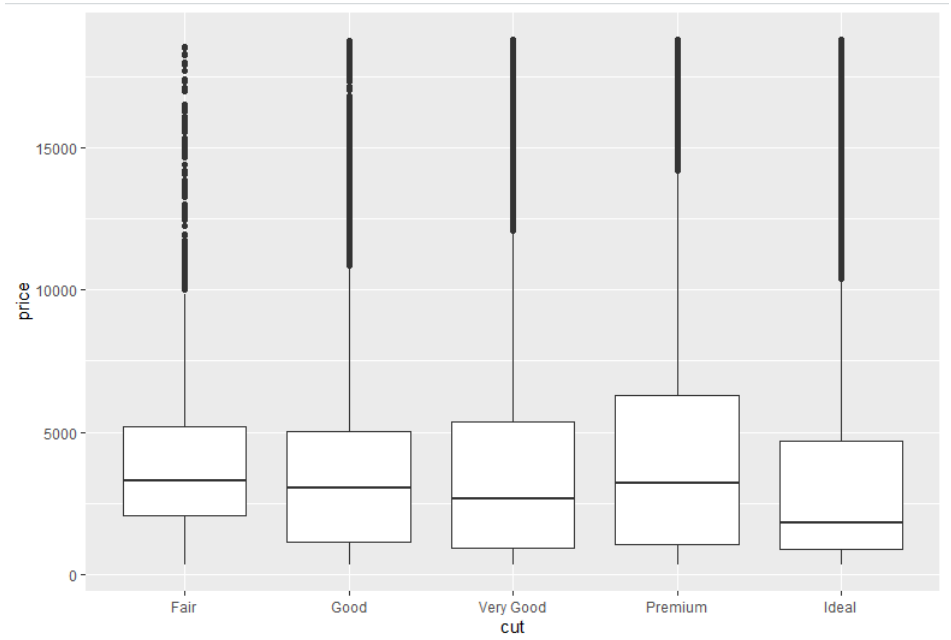

```
diamonds %>%
  ggplot(aes(x,z)) + geom_point() + xlim(0,15) + ylim(0,15)
```

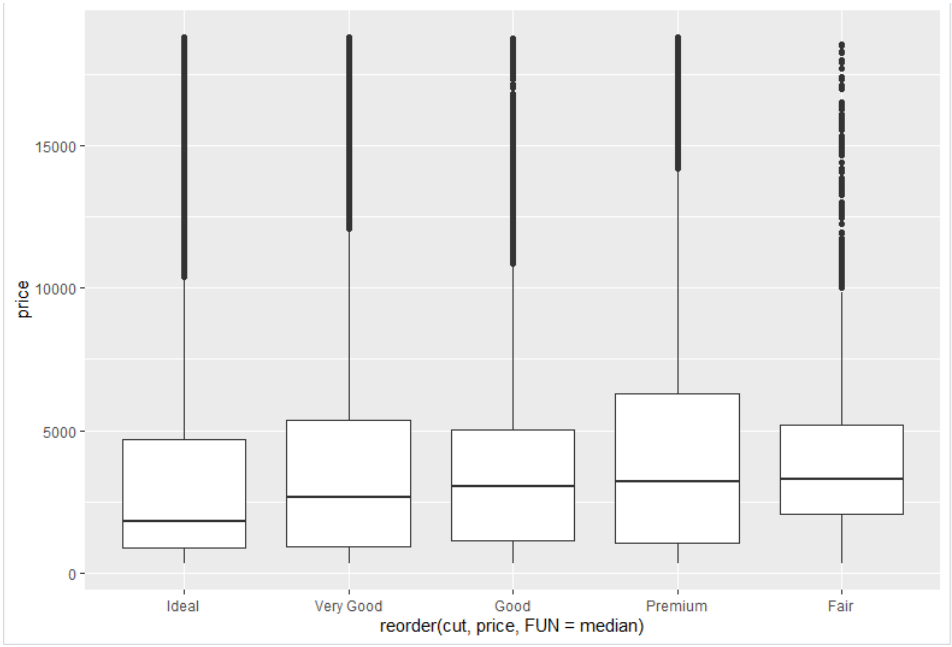


7.5 BOXPLOTS WITH DIAMONDS DATASET

Note that the cut variables are ordered based on increasing quality of the cut. What if we want to put them in order of increasing median price.

```
diamonds %>%
  ggplot() +
      geom_boxplot(aes(x = reorder(cut, price, FUN = median), y = price))
```





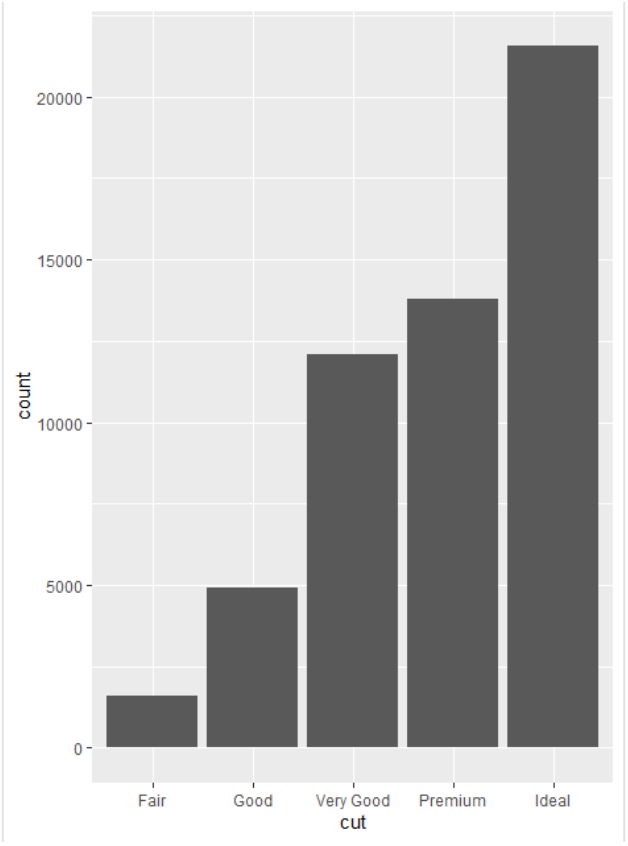
QUIZ

Count the number of records for each value in a categorical variable.

A) `ggplot(diamonds) + geom_cut(aes(count))`

B) `ggplot(diamonds) + geom_histogram(aes(cut))`

C) `ggplot(diamonds) + geom_bar(aes(cut))`



Answer:

C) `ggplot(diamonds) + geom_bar(aes(cut))`

QUIZ

Show a bar chart with the values of a categorical variable but instead of counting the number of records for each value, add up the values in a continuous variable

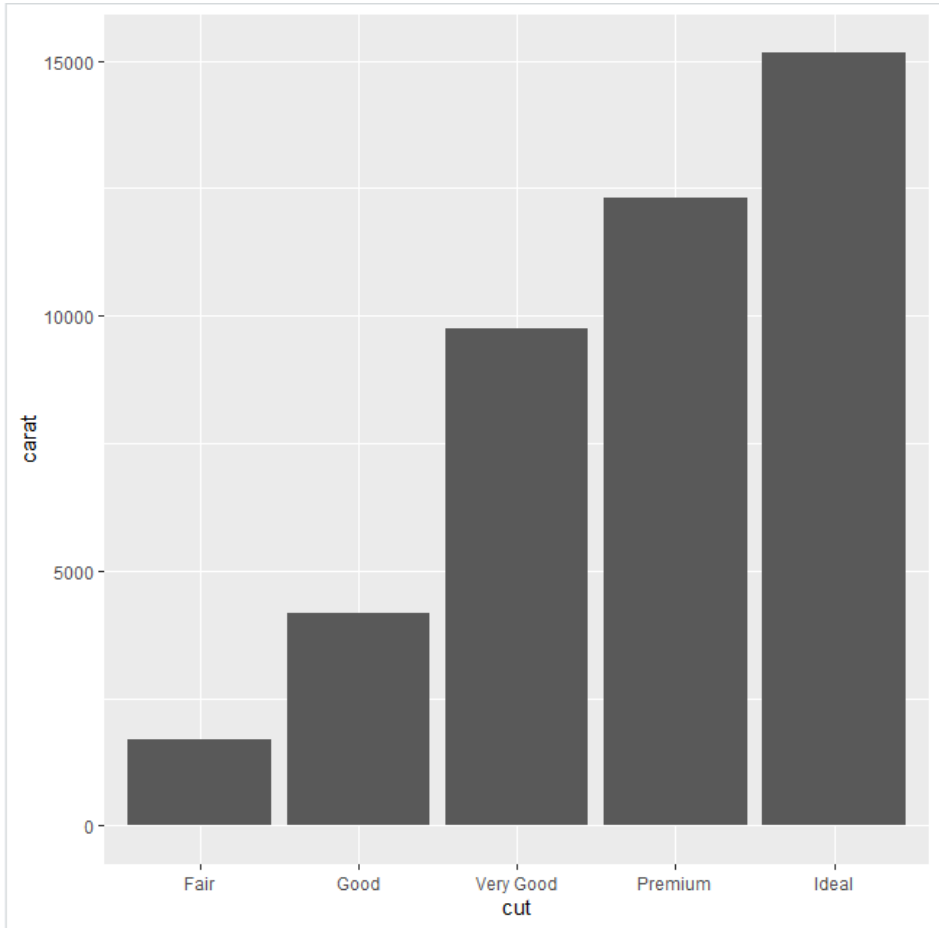
A) `ggplot(diamonds) + geom_cut(aes(carat))`

B)
```
ggplot(diamonds) + geom_histogram(aes(cut)) +
  geom_histogram(aes(carat))
```

C) `ggplot(diamonds) + geom_bar(aes(cut, carat), stat = "identity")`

Answer:

C) `ggplot(diamonds) + geom_bar(aes(cut, carat), stat = "identity")`



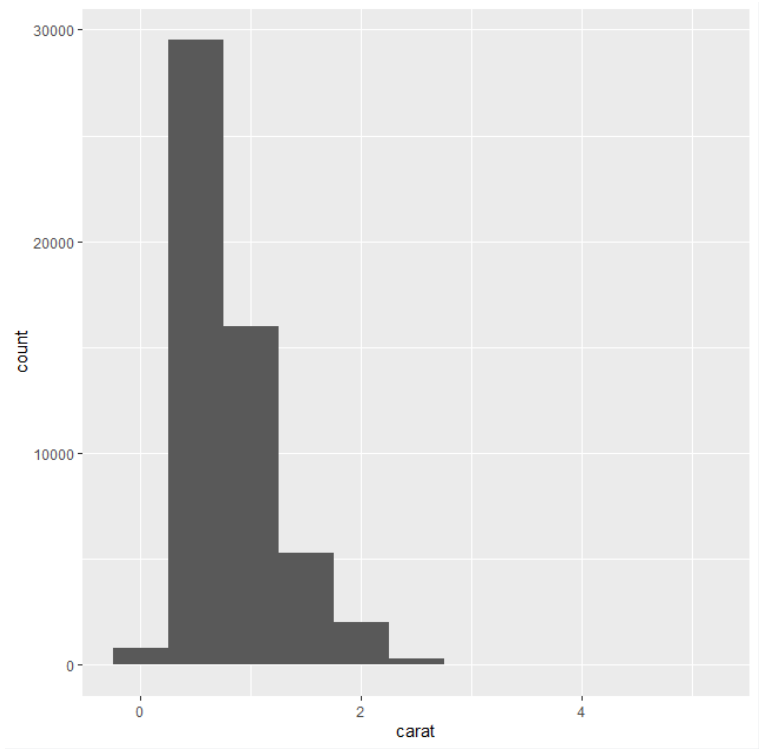
QUIZ

Count the number of records for each group in a continuous variable.

A) `ggplot(diamonds) + geom_histogram(aes(carat), binwidth = 0.5)`

B) `ggplot(diamonds) + geom_histogram(aes(carat, binwidth = 0.5))`

C) `ggplot(diamonds) + geom_histogram(aes(cut), binwidth = 0.5)`



Answer:

A) `ggplot(diamonds) + geom_histogram(aes(carat), binwidth = 0.5)`

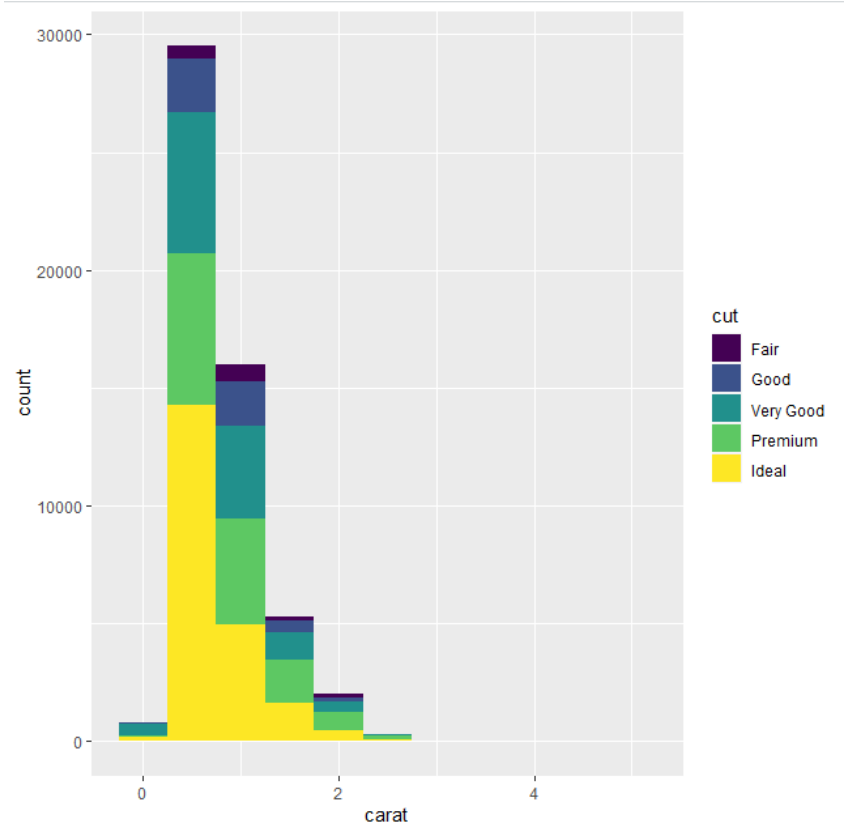
QUIZ

Count the number of records for each group in a continuous variable, but add color based on a categorical variable.

A) `ggplot(diamonds) + geom_histogram(aes(carat, color = cut), binwidth = 0.5)`

B) `ggplot(diamonds) + geom_histogram(aes(carat, fill = cut), binwidth = 0.5)`

C) `ggplot(diamonds) + geom_histogram(aes(cut, carat), binwidth = 0.5)`



Answer:

B) `ggplot(diamonds) + geom_histogram(aes(carat, fill = cut), binwidth = 0.5)`

QUIZ

Count the number of records for each group in a continuous variable, make it a line chart with each line a different color based on a categorical variable.

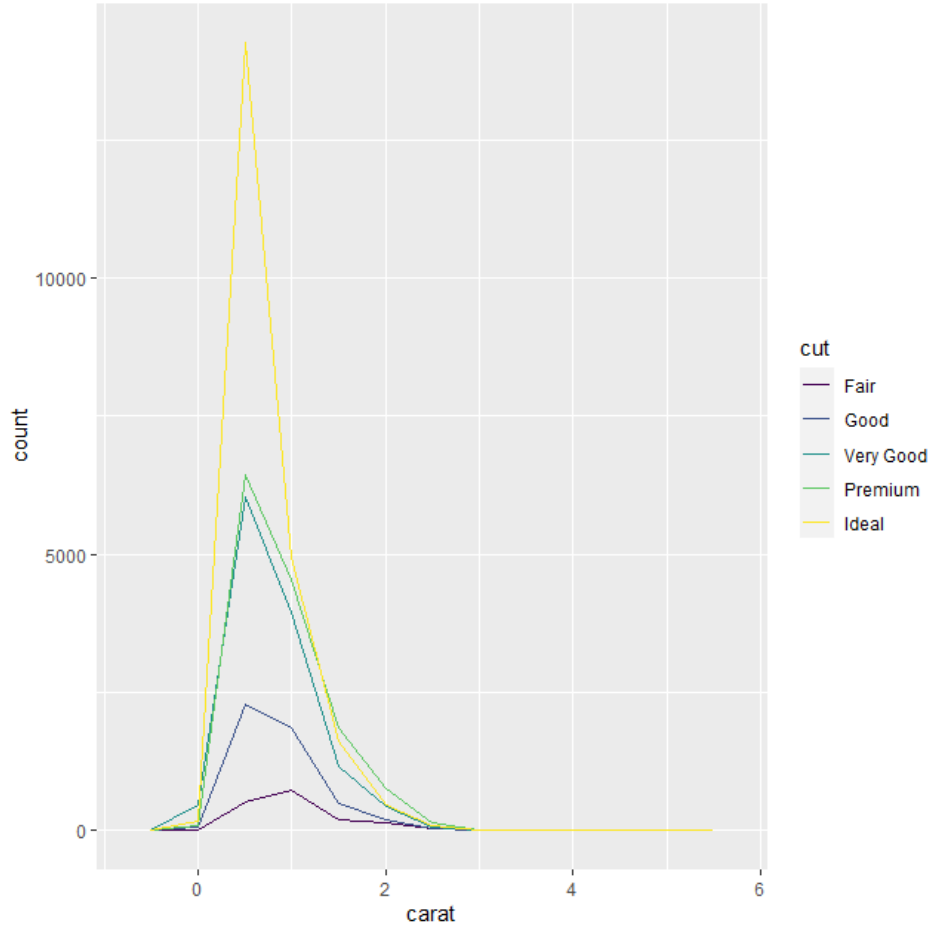
A) `ggplot(diamonds) + geom_freqpoly(aes(carat, color = cut), binwidth = 0.5)`

B) `ggplot(diamonds) + geom_freqpoly(aes(carat, fill = cut), binwidth = 0.5)`

C) `ggplot(diamonds) + geom_histogram(aes(carat), binwidth = 0.5) + color(cut)`

Answer:

A) `ggplot(diamonds) + geom_freqpoly(aes(carat, color = cut), binwidth = 0.5)`



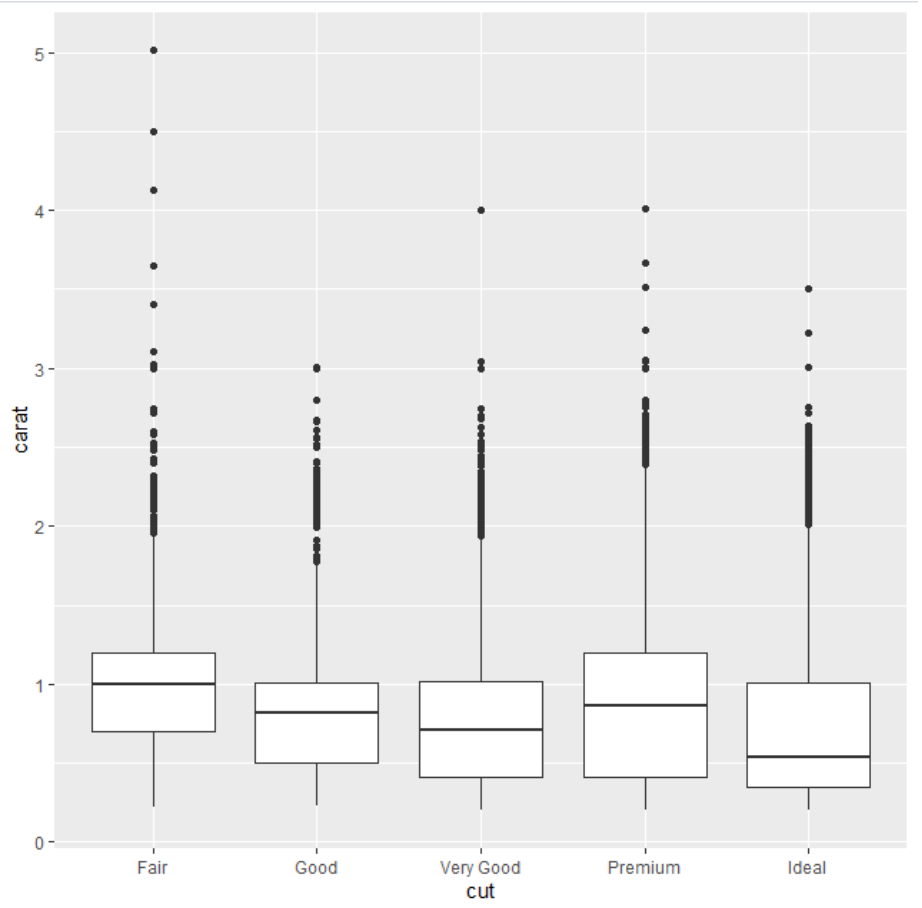
QUIZ

Show the distribution of a continuous variable, broken down by a categorical variable.

A) `ggplot(diamonds) + geom_freqpoly(aes(cut, carat))`

B) `ggplot(diamonds) + geom_bar(aes(cut, carat))`

C) `ggplot(diamonds) + geom_boxplot(aes(cut, carat))`



Answer:

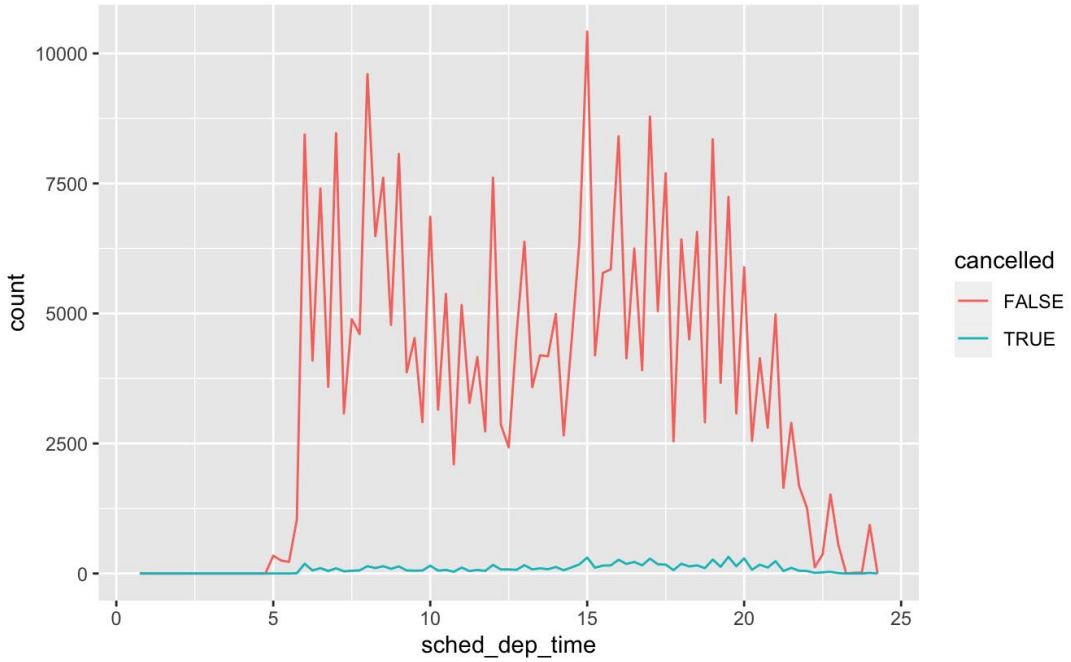
C) `ggplot(diamonds) + geom_boxplot(aes(cut, carat))`

7.4 MISSING VALUES

Other times you want to understand what makes observations with missing values different to observations with recorded values. For example, in `nycflights13::flights`, missing values in the `dep_time` variable indicate that the flight was cancelled. So you might want to compare the scheduled departure times for cancelled and non-cancelled times. You can do this by making a new variable with `is.na()`.

```
nycflights13::flights %>%
  mutate(
    cancelled = is.na(dep_time),
    sched_hour = sched_dep_time %/% 100,
    sched_min = sched_dep_time %% 100,
    sched_dep_time = sched_hour + sched_min / 60
  ) %>%
  ggplot(mapping = aes(sched_dep_time)) +
    geom_freqpoly(mapping = aes(colour = cancelled), binwidth = 1/4)
```

Copy



7.4 MISSING VALUES EXERCISE

7.5.1.1 **Exercises**

1. Use what you've learned to improve the visualisation of the departure times of cancelled vs. non-cancelled flights.

- Jeff Arnold Solution (boxplot)
- Bryan Shalloway Solution (density curve)
- Your Solution

DATASET: FLIGHTS

Flights (load `nycflights13` then `?flights`)

Variable	Format
year, month, day	Of departure
Dep_time, arr_time	Actual dep / arr times
Sched_dep_time, sched_arr_time	Scheduled dep / arr times
Dep_delay, arr_delay	Dep / arr delay in minutes.
Carrier	Two letter carrier abbreviation
Flight	Flight number
Tailnum	Plane tail number
Origin, dest	Origin and destination airport codes
Air_time	Flight time
Distance	Airport distance
Hour, minute	Scheduled departure in hour and minutes
Time_hour	Scheduled date and hour of flight

`head(flights)`

```
> head(flights)
# A tibble: 6 x 19
   year month   day dep_time sched_dep_time dep_delay arr_time
  <int> <int> <int>    <int>          <int>     <dbl>    <int>
1  2013     1     1      517            515         2      830
2  2013     1     1      533            529         4      850
3  2013     1     1      542            540         2      923
4  2013     1     1      544            545        -1     1004
5  2013     1     1      554            600        -6      812
6  2013     1     1      554            558        -4      740
>
```

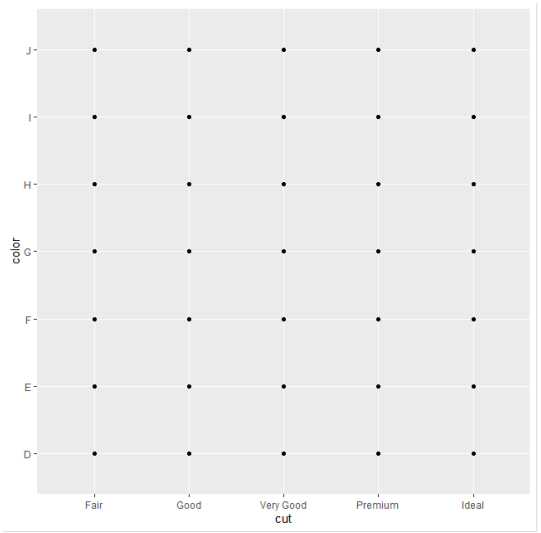
```
sched_arr_time arr_delay carrier flight tailnum origin dest
         <int>     <dbl> <chr>    <int> <chr>   <chr>  <chr>
           819        11 UA        1545 N14228  EWR    IAH
           830        20 UA        1714 N24211  LGA    IAH
           850        33 AA        1141 N619AA  JFK    MIA
          1022       -18 B6         725 N804JB  JFK    BQN
           837       -25 DL         461 N668DN  LGA    ATL
           728        12 UA        1696 N39463  EWR    ORD

air_time distance  hour minute time_hour
   <dbl>    <dbl> <dbl>  <dbl> <dttm>
     227     1400     5     15 2013-01-01 05:00:00
     227     1416     5     29 2013-01-01 05:00:00
     160     1089     5     40 2013-01-01 05:00:00
     183     1576     5     45 2013-01-01 05:00:00
     116      762     6      0 2013-01-01 06:00:00
     150      719     5     58 2013-01-01 05:00:00
```

FINISHING CHAPTER 7

To compare two categorical variables, a scatterplot (geom_point) can be a good place to start.

```
ggplot(diamonds) + geom_point(aes(cut, color))
```



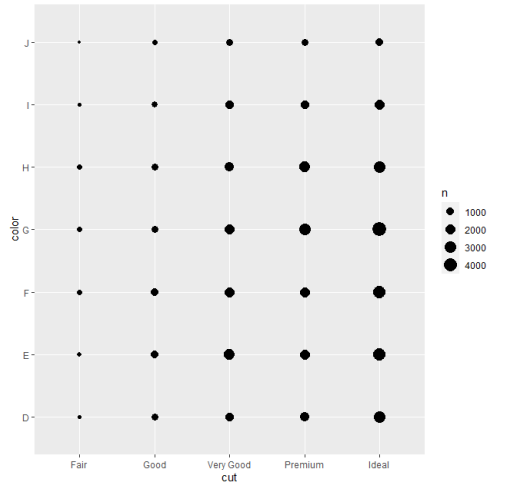
But this is pretty meaningless unless there is a quantitative way to compare them.

If you only want to count observations, use geom_count.

```
diamonds %>%
    count(cut, color)
```

```
# A tibble: 35 x 3
   cut   color     n
   <ord> <ord> <int>
 1 Fair  D       163
 2 Fair  E       224
 3 Fair  F       312
 4 Fair  G       314
 5 Fair  H       303
 6 Fair  I       175
 7 Fair  J       119
 8 Good  D       662
 9 Good  E       933
10 Good  F       909
# ... with 25 more rows
```

```
ggplot(diamonds) +
    geom_count(aes(cut, color))
```



FINISHING CHAPTER 7

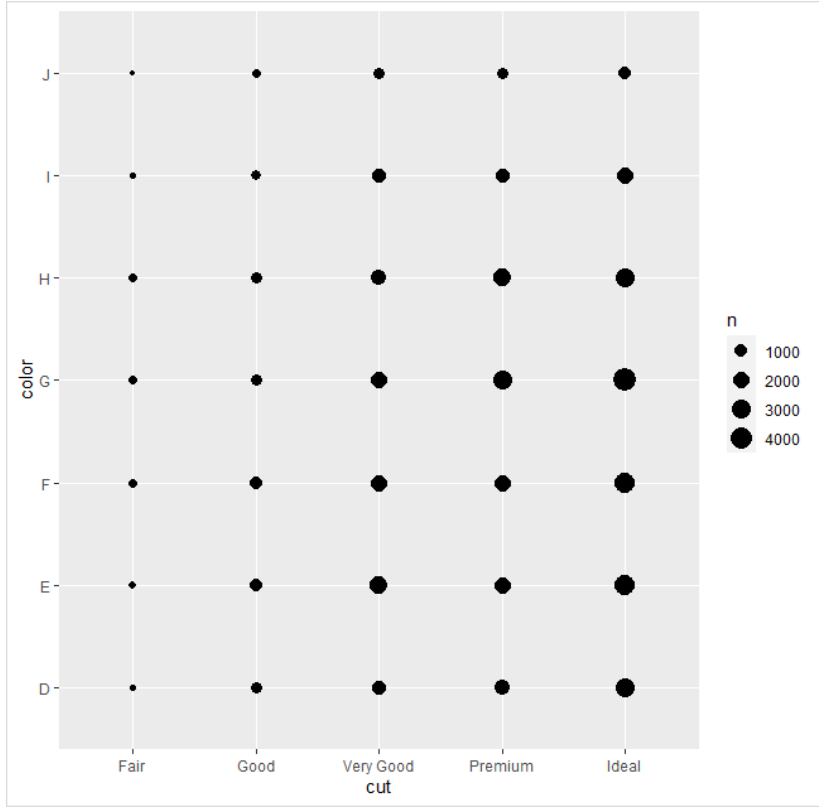
Another approach to compare two categorical variables is to calculate the counts using dplyr.

geom_count = great when only comparing counts

```
ggplot(diamonds) + geom_count(aes(cut, color))
```

dplyr = greater flexibility (`group_by` and `summarize`)

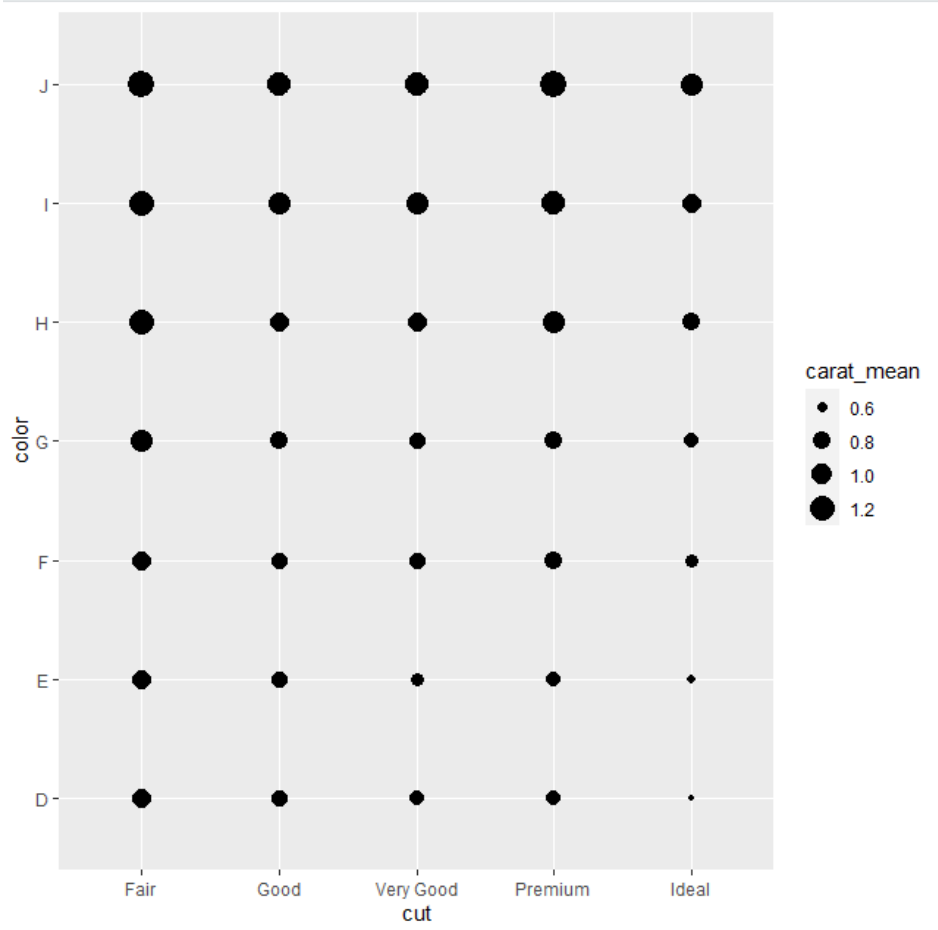
```
diamonds %>%
  group_by(cut, color) %>%
  summarize(total_n = n()) %>%
  ggplot() + geom_point(aes(cut, color, size = total_n))
```



FINISHING CHAPTER 7

The dplyr approach lets you substitute another aggregation method instead of count. Here we can still use a scatterplot (geom_point) but display the average carat size

```
diamonds %>%
  group_by(cut, color) %>%
  summarise(carat_mean = mean(carat)) %>%
  ggplot() + geom_point(aes(cut, color, size = carat_mean))
```



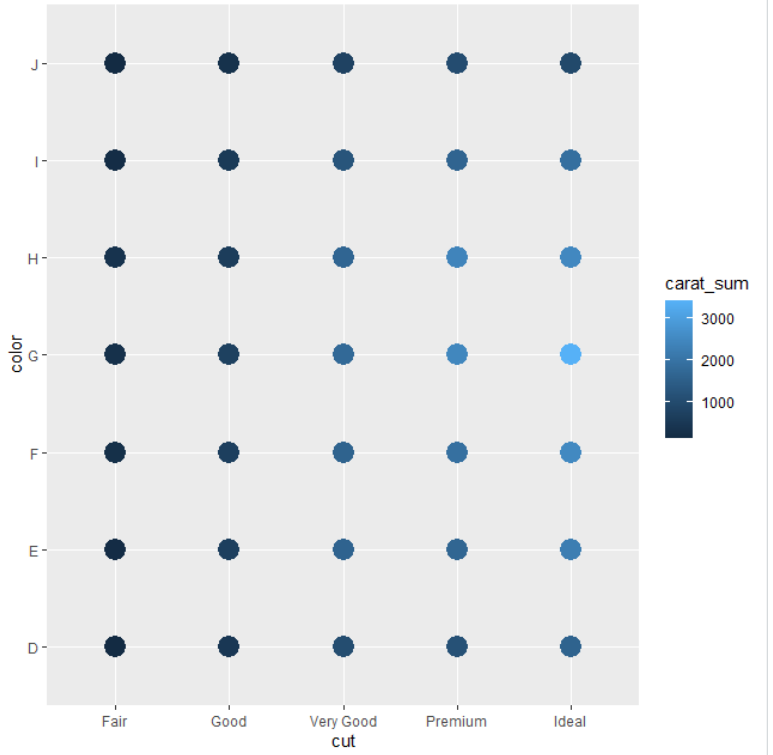
FINISHING CHAPTER 7

Or map the quantitative variable to a color instead of a size.

```
diamonds %>%
  group_by(cut, color) %>%
  summarize(total_n = n()) %>%
  ggplot() + geom_point(aes(cut, color, size = total_n))
```

```
diamonds %>%
  group_by(cut, color) %>%
  summarise(carat_sum = sum(carat)) %>%
  ggplot() + geom_point(
        aes(cut, color, color = carat_sum), size = 6)
```

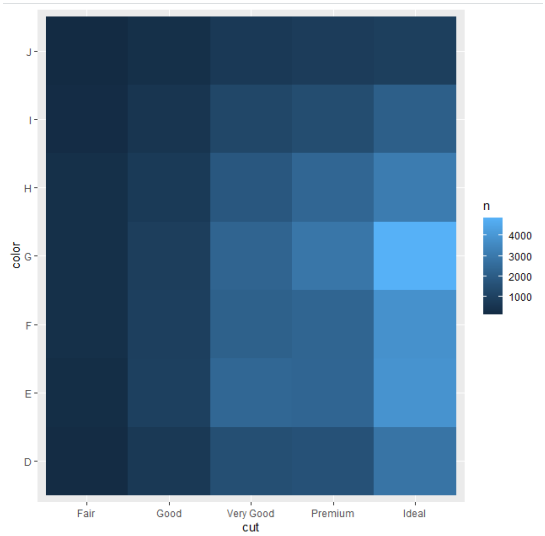
```
# A tibble: 35 x 3
# Groups:   cut [5]
   cut   color carat_sum
   <ord> <ord>     <dbl>
 1 Fair  D          150.
 2 Fair  E          192.
 3 Fair  F          282.
 4 Fair  G          321.
 5 Fair  H          369.
 6 Fair  I          210.
 7 Fair  J          160.
 8 Good  D          493.
 9 Good  E          695.
10 Good  F          705.
# ... with 25 more rows
```



FINISHING CHAPTER 7

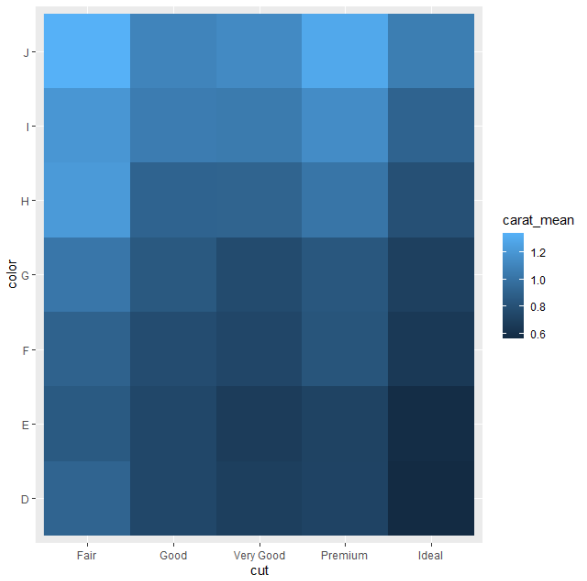
There's a better geom for comparing two categorical variables and mapping color to a different quantitative variable: `geom_tile`

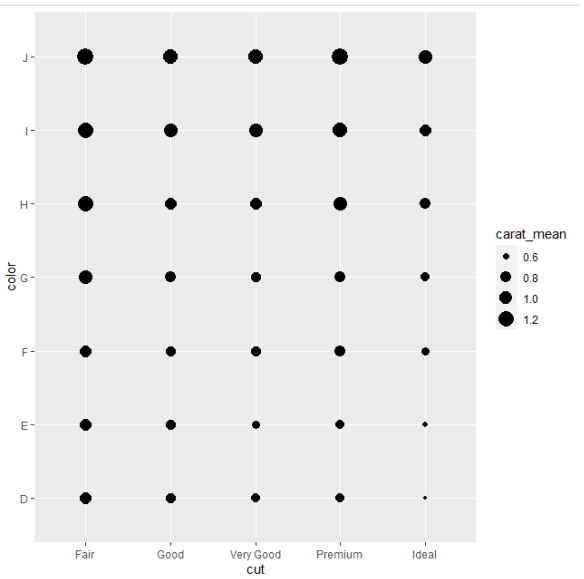
```
diamonds %>%
  count(cut, color) %>%
  ggplot() + geom_tile(aes(cut, color, fill = n))
```



Use dplyr for a little more flexibility beyond counts

```
diamonds %>%
  group_by(cut, color) %>%
  summarise(carat_mean = mean(carat)) %>%
  ggplot() + geom_tile(aes(cut, color, fill = carat_mean))
```



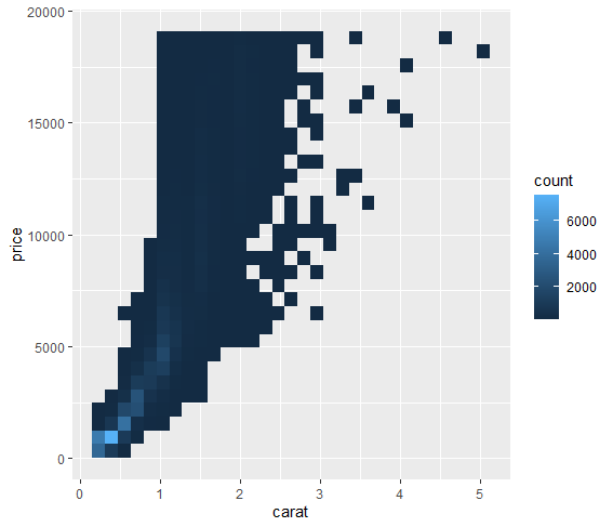


FINISHING CHAPTER 7

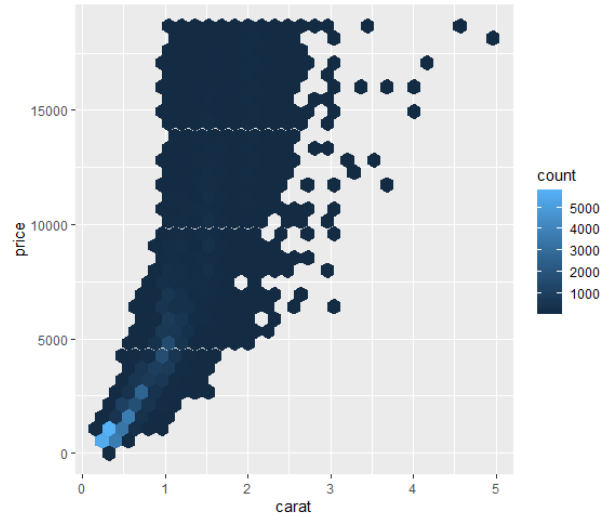
For comparing two continuous variables:

```
geom_point
geom_bin2d
geom_hex
```

```
diamonds %>%
  ggplot() + geom_bin2d(aes(carat, price))
```




```
library(hexbin)
diamonds %>%
  ggplot() + geom_hex(aes(carat, price))
```



NEXT WEEK…

- Chapter 8 – 10

GETTING HELP

- Ask questions during our call
- Google
- Stack Overflow
- Slack
- Office Hours r4ds.io/calendar
- Twitter #rstats
- r4ds answer keys: Jeff Arnold (preferred) or Bryan Shalloway (also good)
- Cheatsheets


RStudio