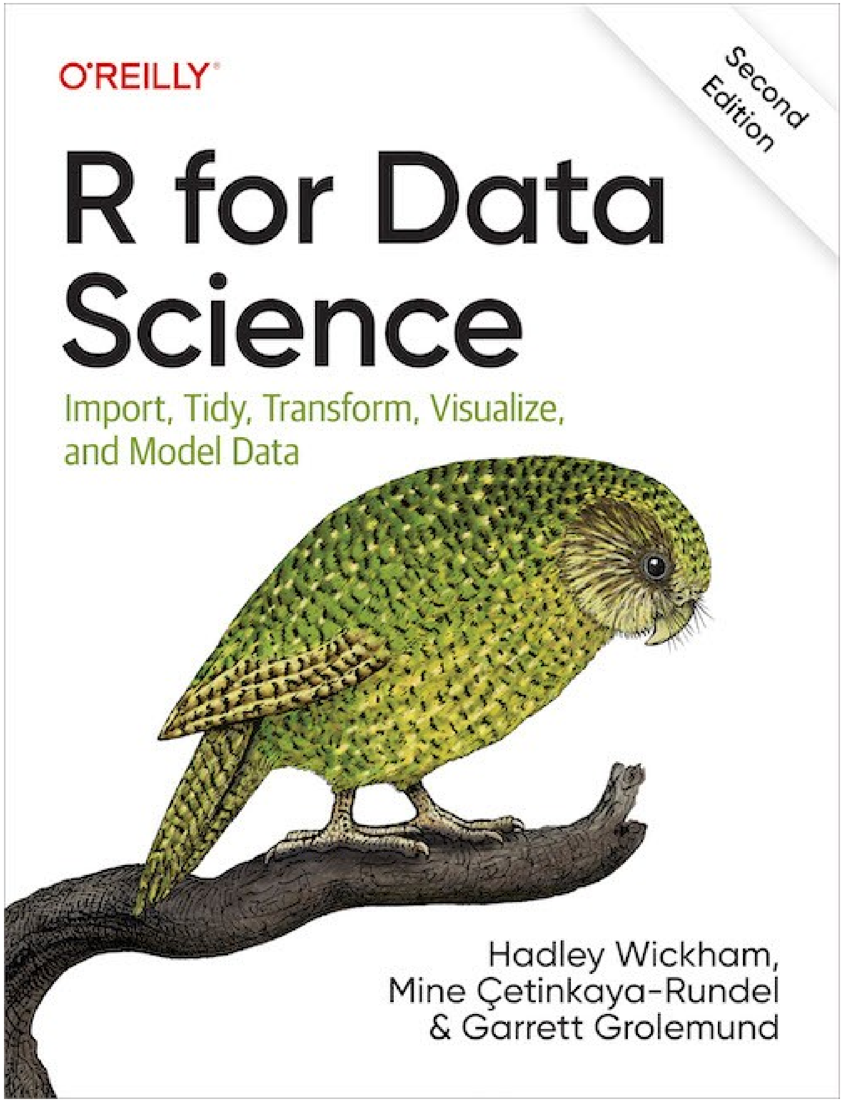
Data Science
CS301

Exploratory First Steps, Continued

Week 4
Fall 2024
Oliver BONHAM-CARTER

Where in the Web?



Web:

Chap 10: Exploratory Data Analysis

– https://r4ds.hadley.nz/eda

Missing Data Points?

Missing Data Entries

- Missing data in R appears as ***NA***.
- *NA* is not a string or a numeric value, but an indicator of missing data.
- Let’s create vectors with missing values to test

```
library(tidyverse)
library(tibble)
x1 <- c(1, 4, 3, NA, 7)
x2 <- c("a", "B", NA, "NA")
is.na(x1)
is.na(x2)
```

Spot missing data

Missing Data Entries

- What to do when elements of your data go missing?
- **Why not just DROP the ENTIRE ROW, as well as to drop all the value contained by its other variables as well??**

```
diamonds2 <- diamonds  %>%  filter(between(y, 3, 20))
```

This is a shortcut for y >= 3 & y <= 20

```
View(diamonds2)
# compare to the the size of original dataset
View(diamonds)
# Note: Good data may have been lost by dropping rows.
```

IfElse(): Condition Statement

```
y = ifelse(y < 3 | y > 20, NA, y)
```

- **Function**
- **Test Condition**
- **If True, then assign this**
- **If False, then assign this**

Data: *Diamond*

The book recommends to *mark* the data as bad or missing.

diamonds2 <- diamonds %>%

mutate(y = **ifelse(y < 3 | y > 20, NA, y)**)

syntax: `ifelse(test, yes, no)`

Inspect each value of *y*. If the *y* is not between 3 and 20, then *y* = NA, else *y* = *y*

We Plot All Non-NA Values

```
# Missing, outliers values marked as NA

ggplot(data = diamonds2, mapping = aes(x = x, y = y)) +  geom_point()

# compared to, no removed missing or outlier values

ggplot(data = diamonds, mapping = aes(x = x, y = y)) +  geom_point()
```

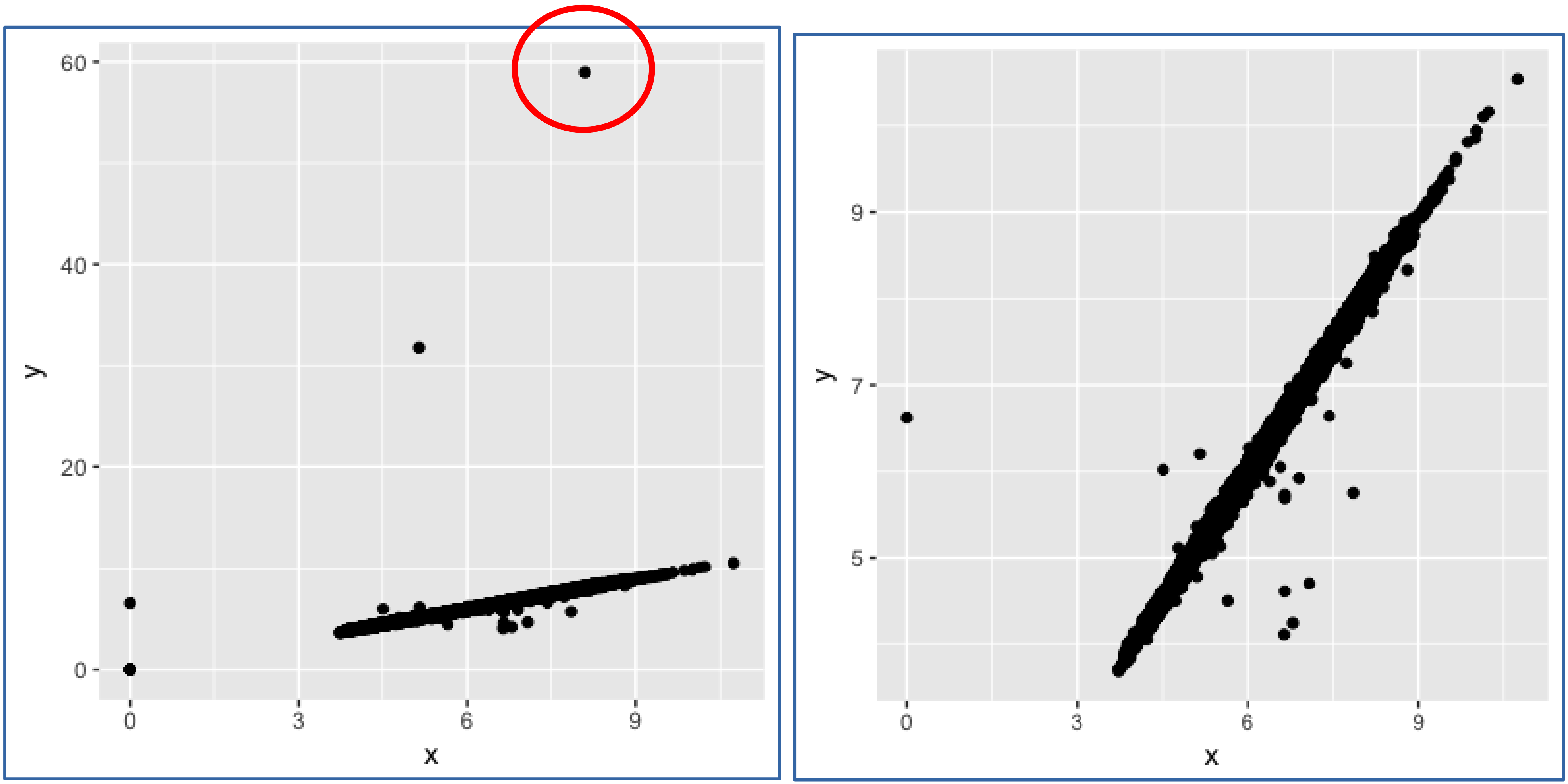
Missing Values, continued

ALLEGHENY COLLEGE

```
# remove the outliers for y (I.e., y<3 and y >20)
library(tidyverse)
?ifelse # get online help
diamonds2 <- diamonds %>%
    mutate(y = ifelse(y < 3 | y > 20, NA, y))
ggplot(data = diamonds2,
   mapping = aes(x = x, y = y)) +  geom_point()
```

Add the ifelse() code directly to your other code.

Trimmed Data, Slightly Different Plot...



- Left: WITH outliers
- Above: NO outliers

Data: *Diamond*

Can you use the below code to further trim outliers or missing data?

Plot your new graphic after using *ifelse()*

```
diamonds3 <- diamonds %>%
  mutate(y = ifelse(y < ## | y > ##, NA, y))
```

Missing Values May Have Their Own Meanings

- Q: Does missing flight arrival-time data indicate canceled flights?

Missing Values May Have Their Own Meanings

```
# install the flights data, if necessary.
#install.packages("nycflights13")
library(tidyverse, nycflights13)
flights <- nycflights13::flights
View(flights)
# Where are the missing values
flights$dep_time
```

The Distribution of a **Continuous** Variable, Aggregated By a **Categorical** variable

```
# compare the scheduled departure times for cancelled and non-cancelled times
flights %>%
  mutate(
    cancelled = is.na(dep_time),
    #%/% is a whole number division
    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)
```

Erm, What Did That Previous Code Do?

First, the data frame `flights` is being piped using the %>% operator to allows to transform for the next step.

1. mutate() creates new columns in the data frame

   - cancelled is a logical column that gets a value of TRUE for cancelled flights (i.e., where dep_time is NA)

   - sched_hour is the hour component of the scheduled departure time, obtained by using whole number division (%/% 100) on the sched_dep_time Note: 5 %/% 2 == 2 (quotient is 2)

   - sched_min is the minute component of the scheduled departure time, obtained by using the `modulo` operator (%% 100) on sched_dep_time Note: 5 %% 2 == 0 (remainder is 0)

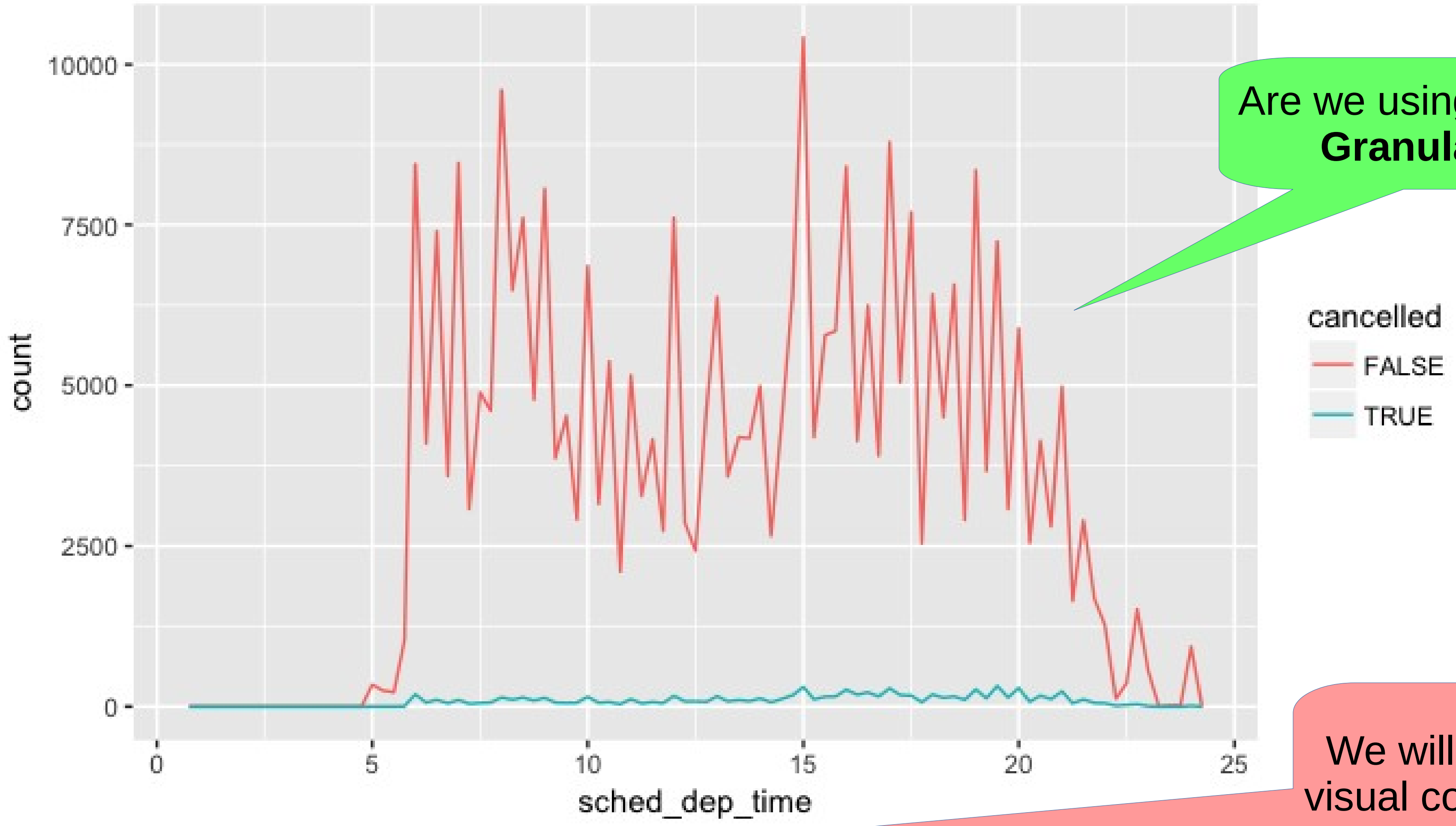
   - sched_dep_time is a new column that combines the hour and minute components into one value, with the hour as an integer and minutes as a decimal

2. The resulting data frame is then being passed to ggplot() for visualization

3. In ggplot(), the data is being mapped to the `x-axis` using sched_dep_time and colored based on the cancelled column using the aes() function

4. The geom_freqpoly() layer is used for density estimation, which generates a histogram-like plot with polynomial lines connecting the bins. The binwidth argument sets the width of each bin to 1/4 hour

Potential Pitfalls in Theory



Are we using the best **Granularity**?

We will return to visual comparisons

- We get an slight idea of when cancellations happen
- Many more non-cancelled flights than cancelled flights: *does the business side of flying introduce a bias for not-canceling flights?*

Covariation

covariance

co·var·i·ance

/ˌkōˈverēəns/

noun

1. MATHEMATICS
   the property of a function of retaining its form when the variables are linearly transformed.
2. STATISTICS
   the mean value of the product of the deviations of two variates from their respective means.

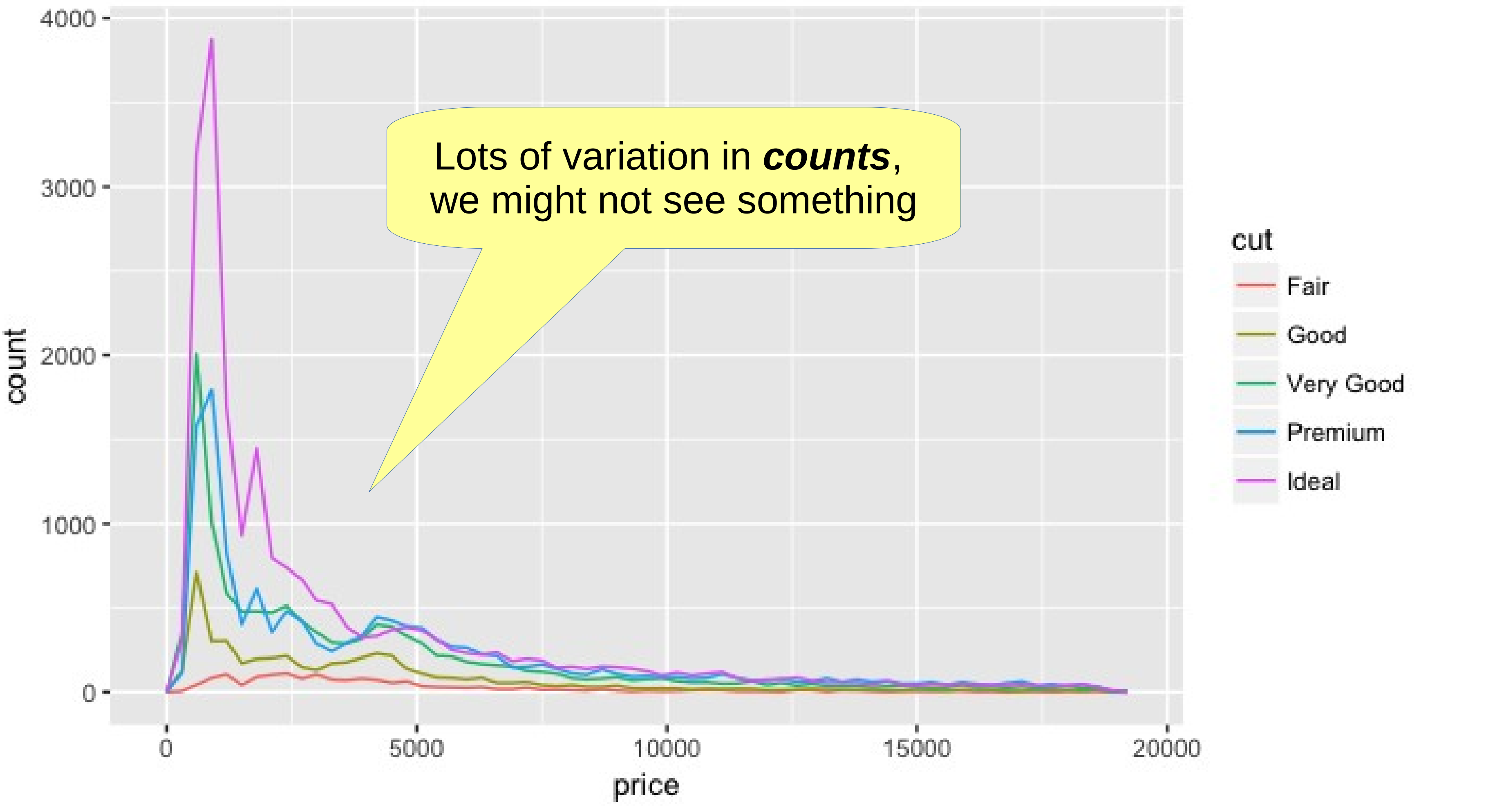
- **Covariation** is the tendency for the values of two or more variables to vary together in a related way.
- Study covariation by visualizing relationships between two or more variables.
- Pay attention to your variables to known how best to visualize these variables

Covariation

- Back to the Diamonds dataset
- How do the prices of diamonds vary with quality?

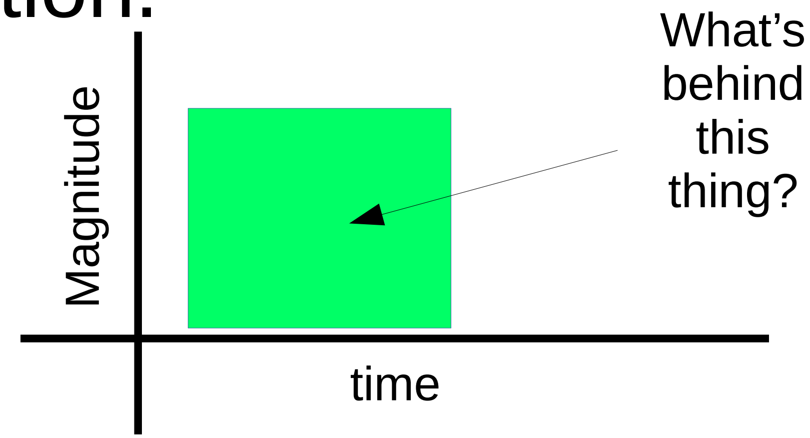
```
# Plot the count of each each cut quality according to price.

ggplot(data = diamonds, mapping = aes(x = price)) + geom_freqpoly(mapping = aes(colour = cut), binwidth = 500)
```

The Plot of the Diamond Counts



This Plot May Make It Hard To See The Phenomenon

- The counts variable seems to have values from all over the range.
- This is noise in our plot
- If one group is much smaller than the others, then it is hard to see the differences in its distribution.



Let's Change Our Plotting

Does a histogram help?

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

#Note: **Density**, the count is standardized so that the area under each frequency polygon is one unit

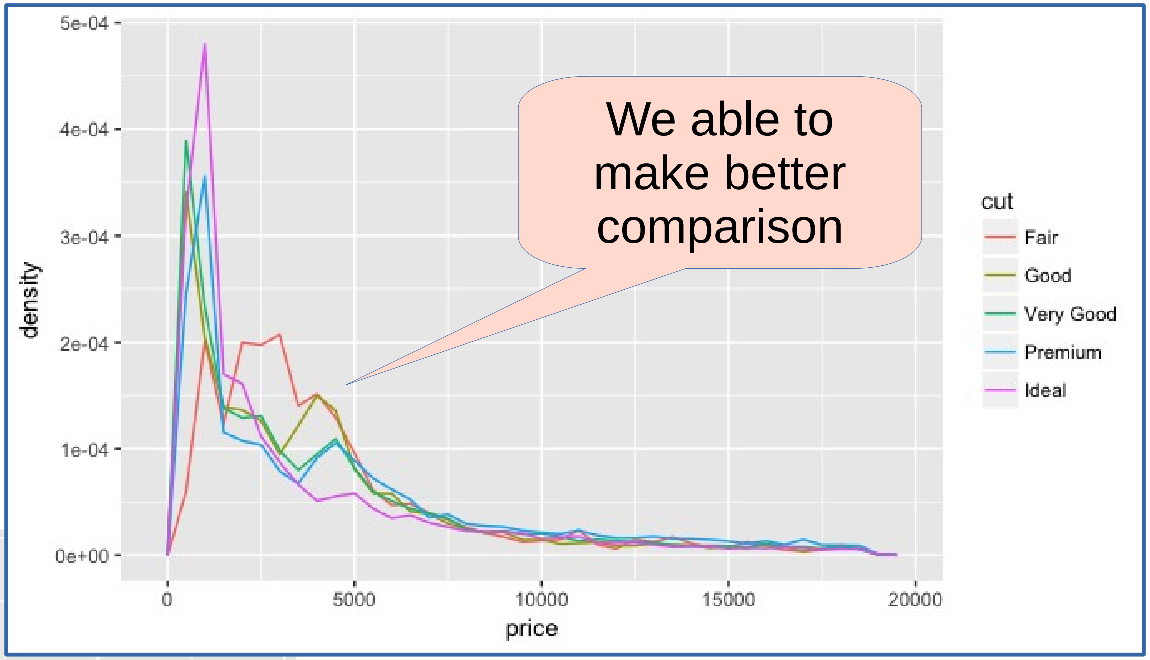
We change the axis "level" the view for all

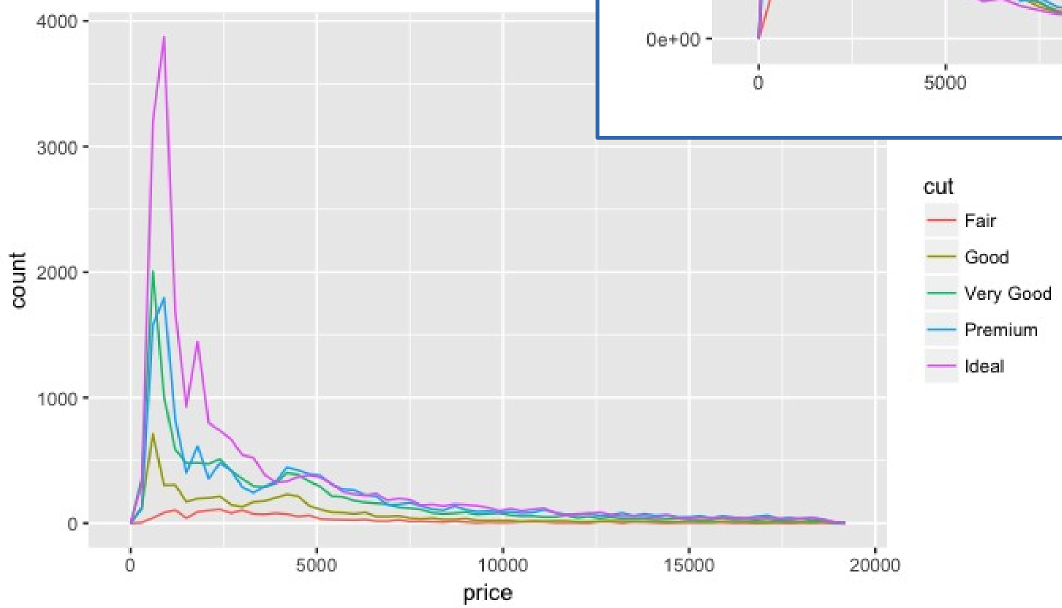
```
ggplot(data = diamonds, mapping = aes(x = price, y
= ..density..)) +  geom_freqpoly(mapping = aes(colour
= cut), binwidth = 500)
```

Normalize Your View!

Different Plots

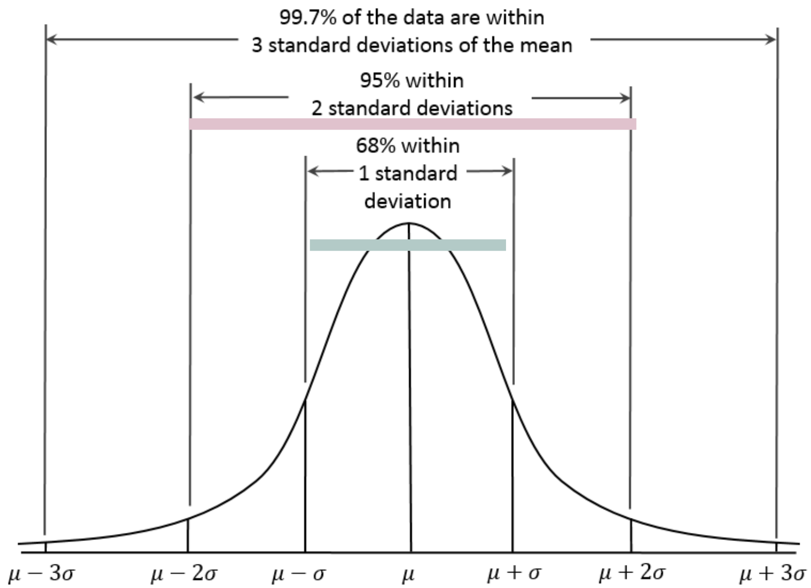
mapping = aes(
x = price,
y = **..density..**
)

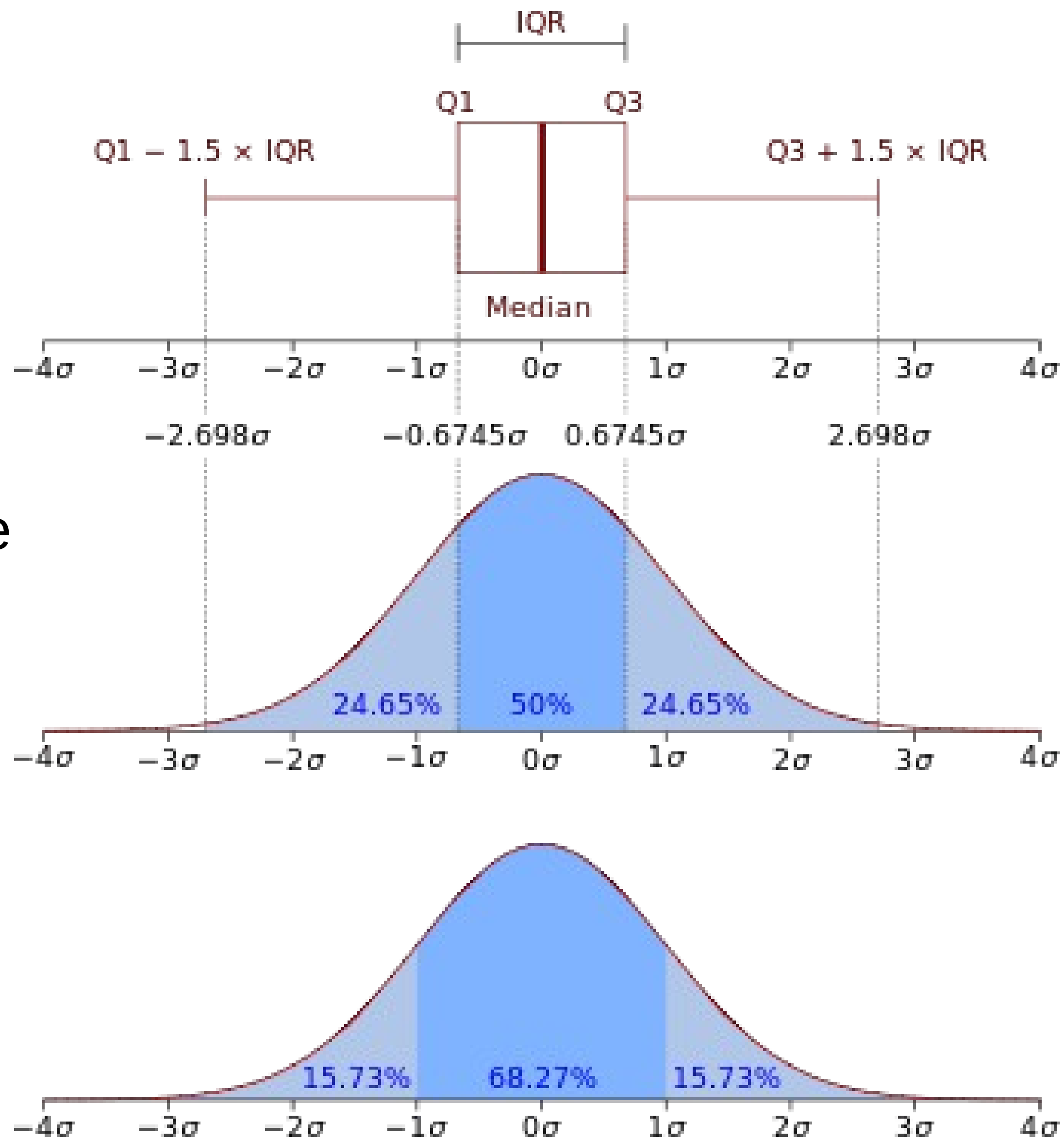




mapping = aes(
x = price,
)

Box Plots

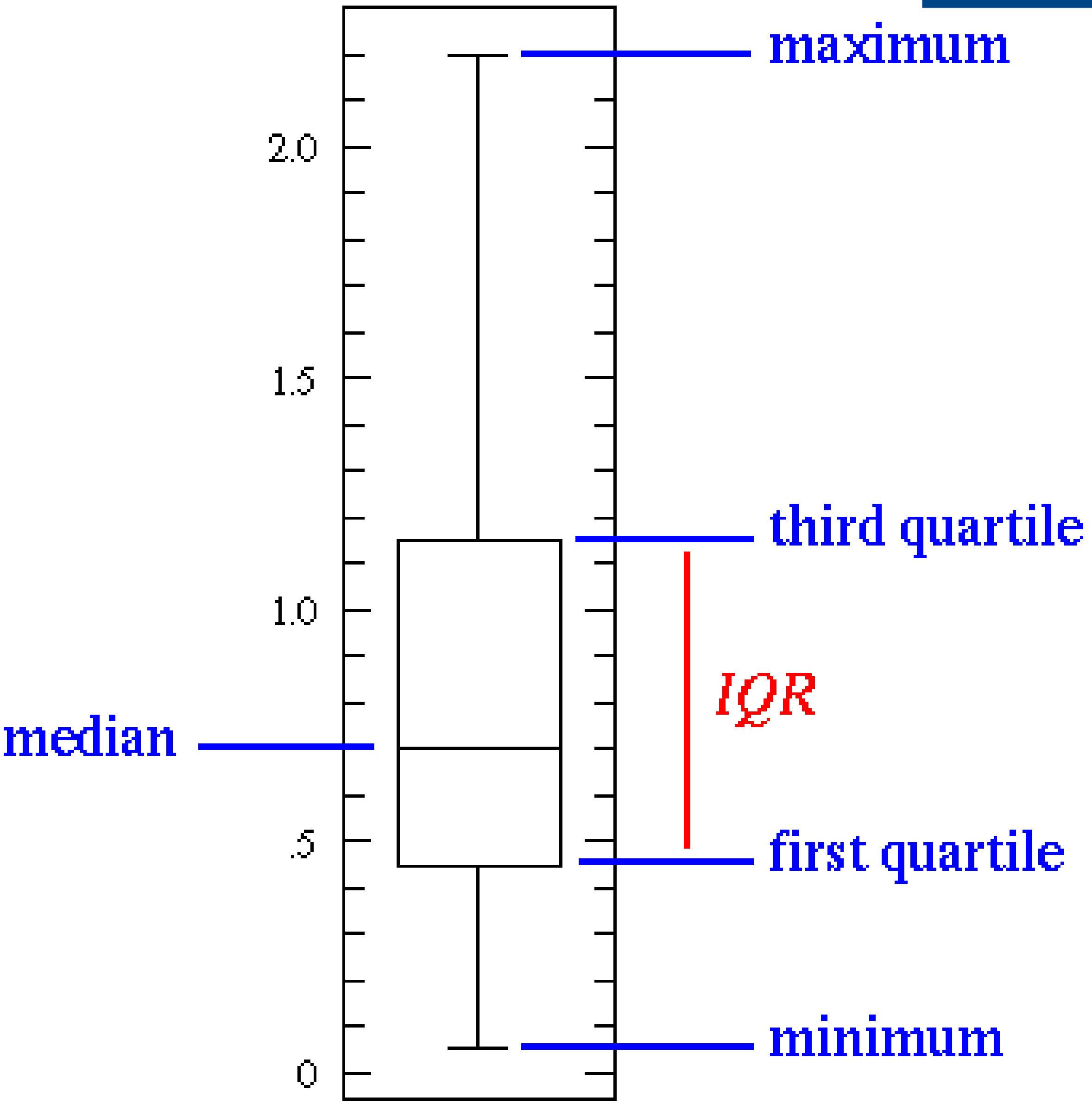




- For the Normal Distribution, the values less than one standard deviation away from the mean account for 68.27% of the set; while two standard deviations from the mean account for 95.45%; and three standard deviations account for 99.73%.

Explore Data Using Box Plots

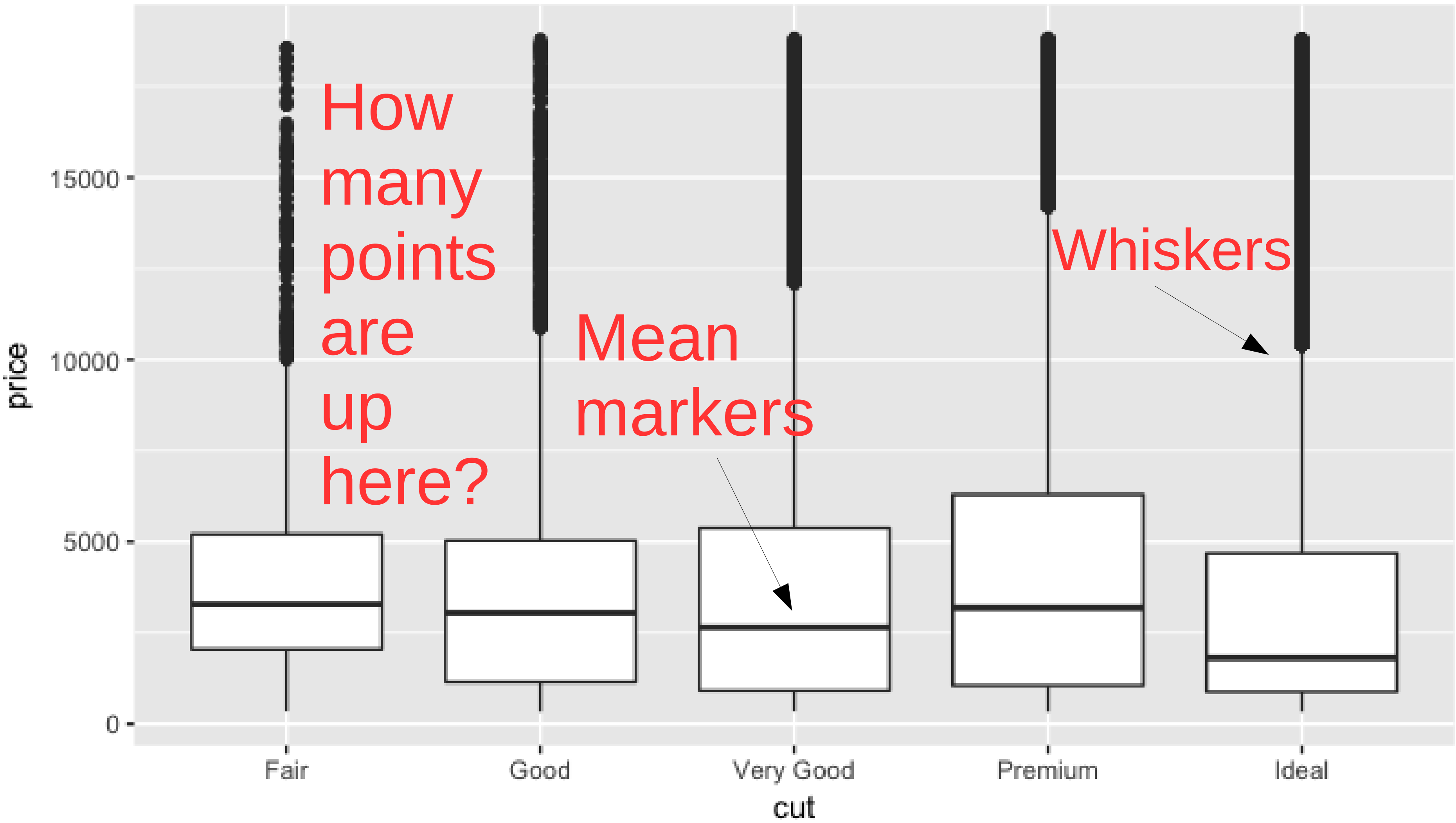
Standardized way of displaying the distribution of data based on the five number summary: minimum, first quartile, median, third quartile, and maximum



Explore Data Using Box Plots

```
# Make a box plot to describe covariance between cut and price.

ggplot(data = diamonds, mapping = aes(x = cut, y = price)) + geom_boxplot()
```

Explore Data Using Box Plots



Box Plots: Pros and Cons

- Pro
  - Box plots are more compact for convenient comparison
- Cons
  - Much less information about the *cut* distribution
  - Be careful, we could incorrectly conclude that better quality diamonds are cheaper on average!

Two Categorical Variables

Visualize the covariation between categorical variables with a “Plot of Dots” to determine observations.

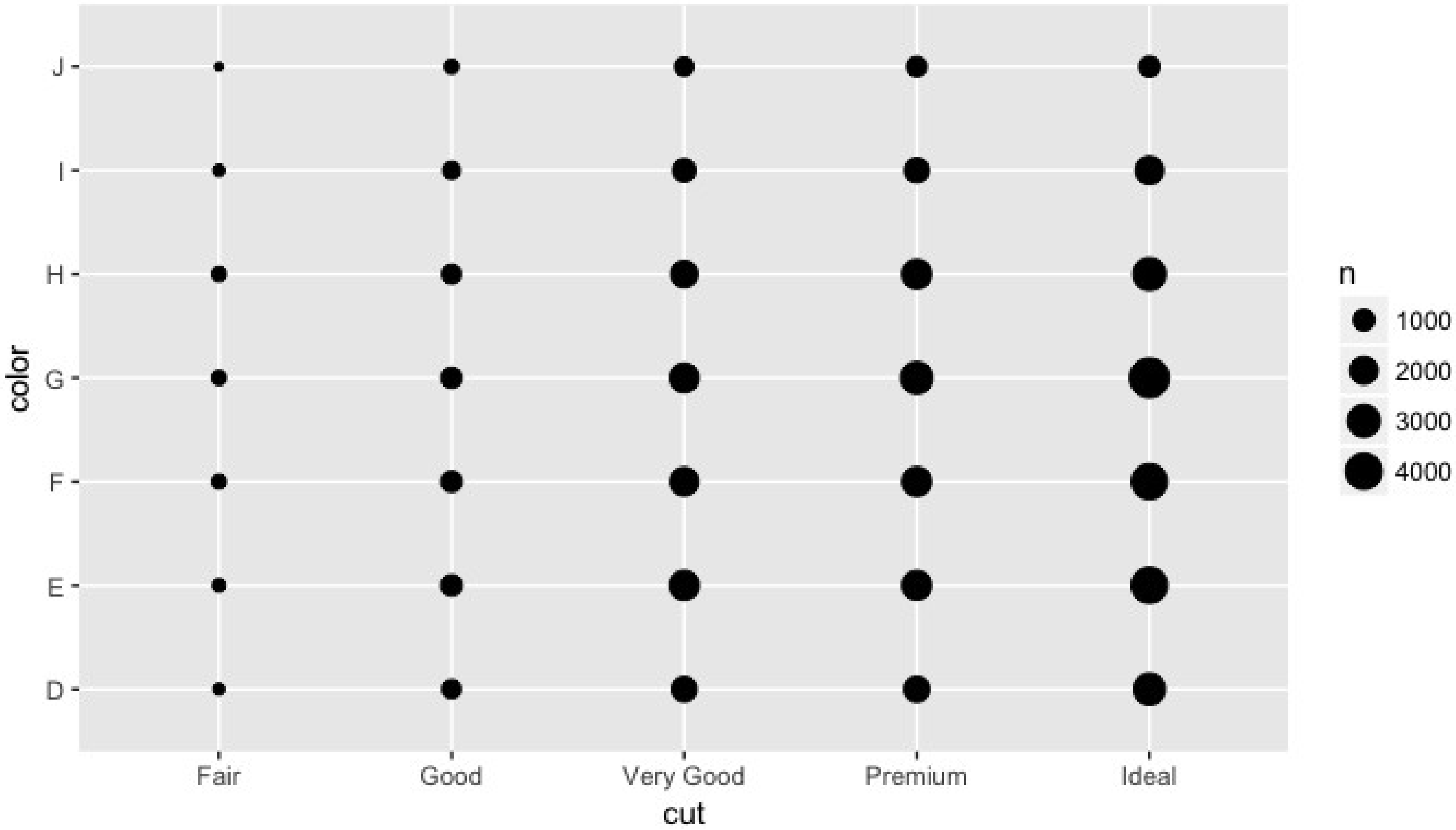
```
ggplot(data = diamonds) +
geom_count(mapping = aes(x = cut, y = color))
```

Note: The size of each circle in the plot displays how many observations occurred at each combination of values

Get exact text details of the plot

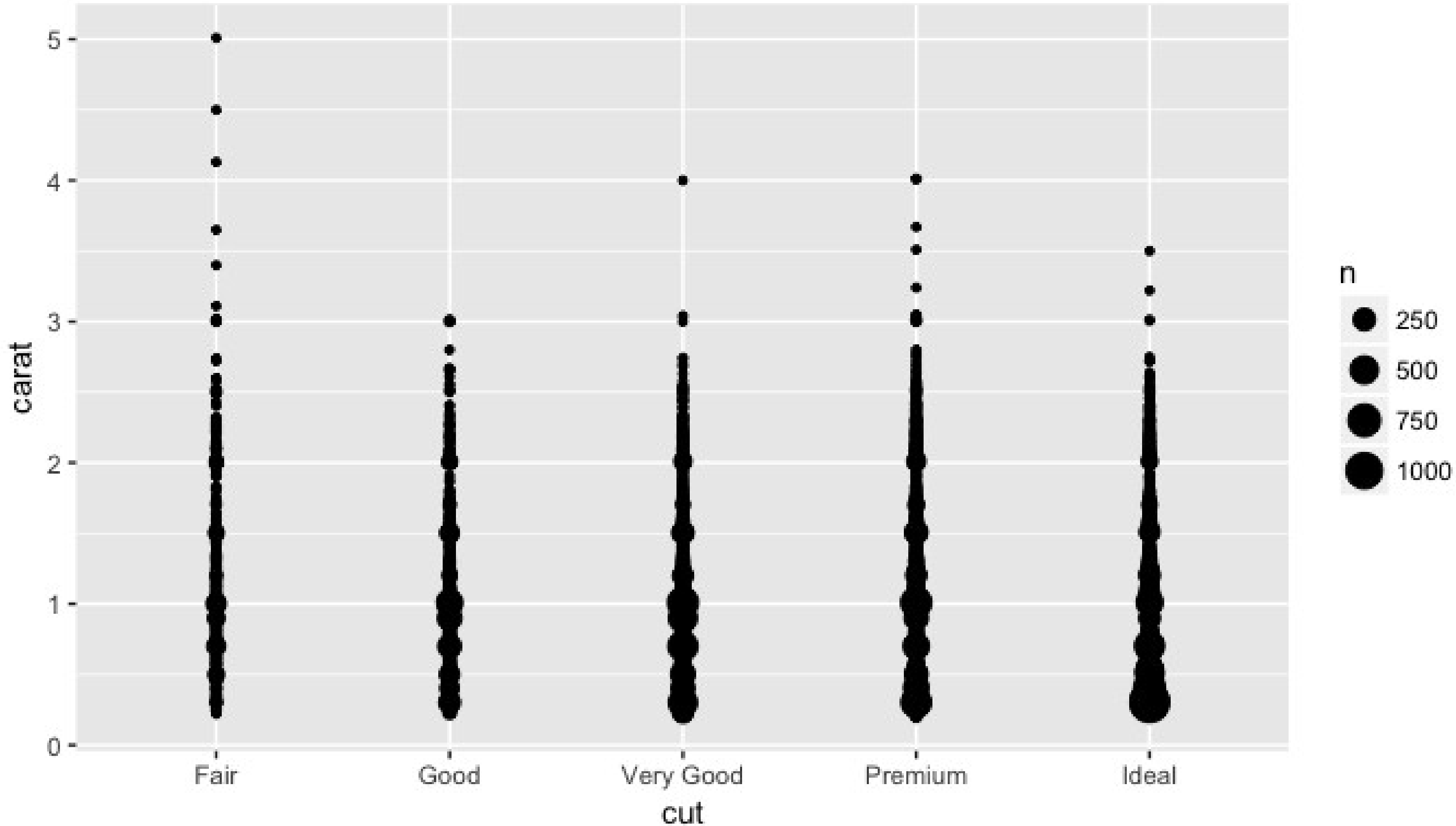
```
diamonds %>% count(color, cut)
```

Mini Distributions: Cut vs Color



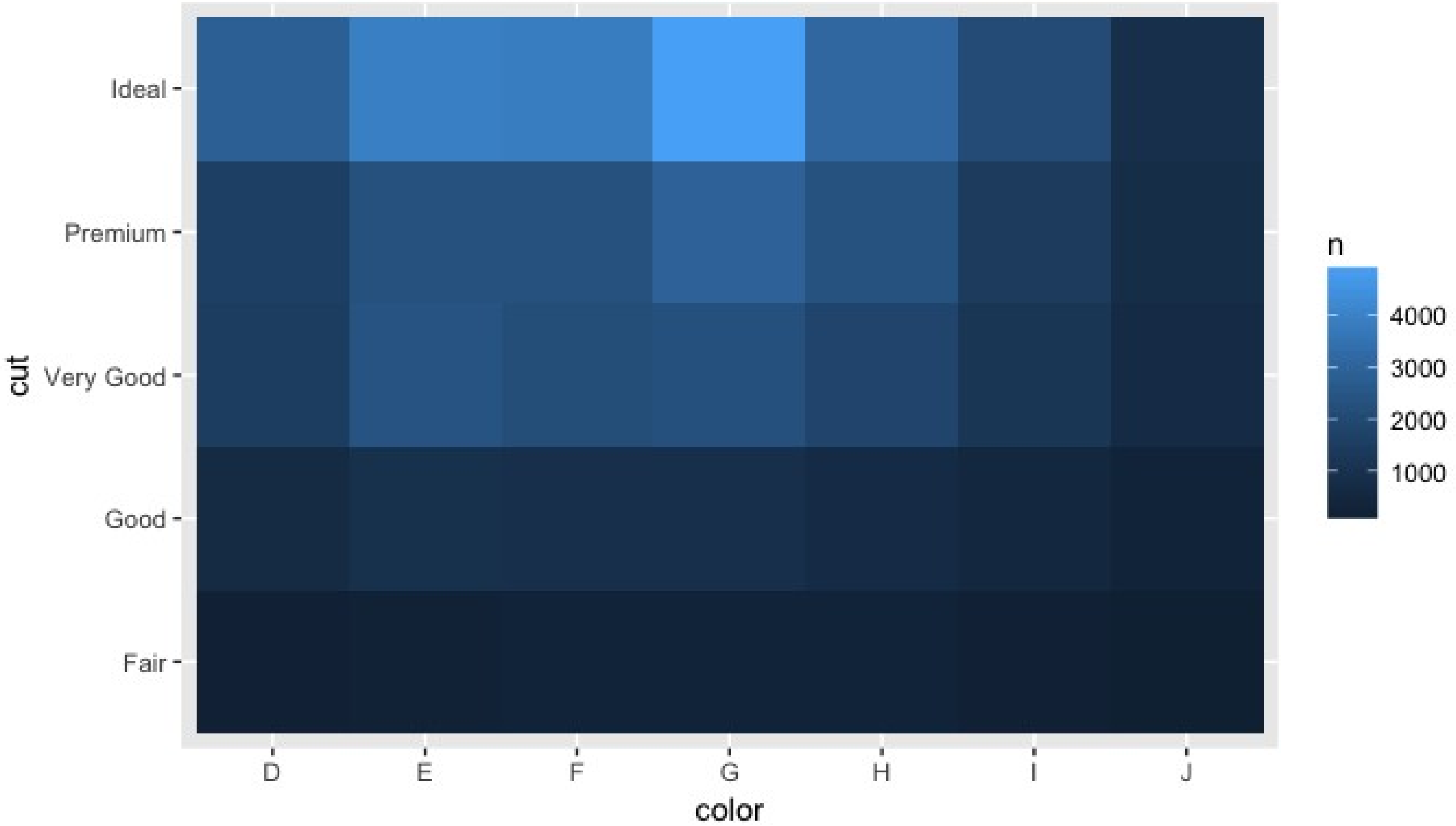

```
ggplot(data = diamonds) +
    geom_count(mapping = aes(x = cut, y = color))
```

Mini Distributions: Cut vs Carat



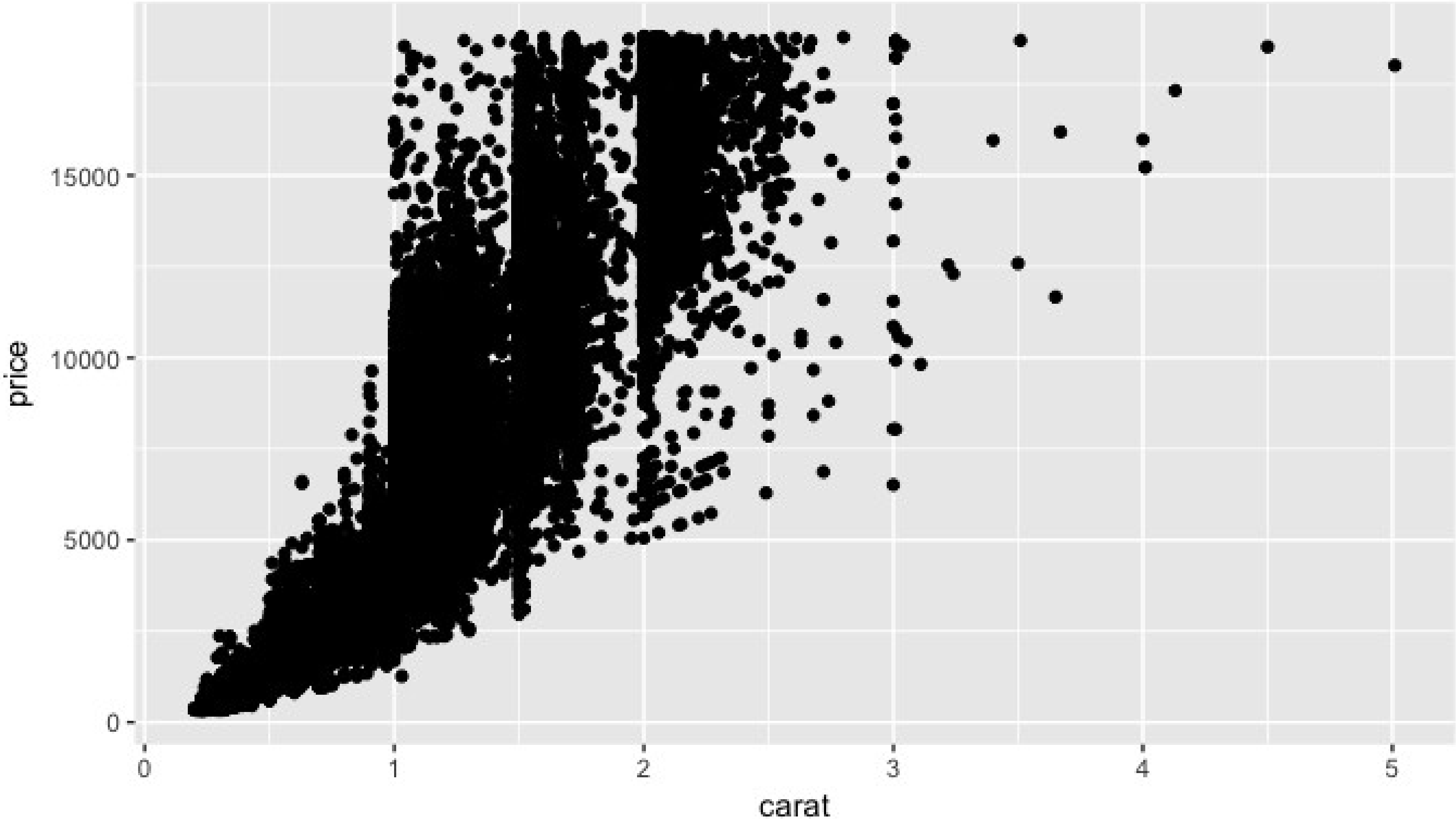

```
ggplot(data = diamonds) +
   geom_count(mapping = aes(x = cut, y = carat))
```

Mini Distributions: Cut vs Color



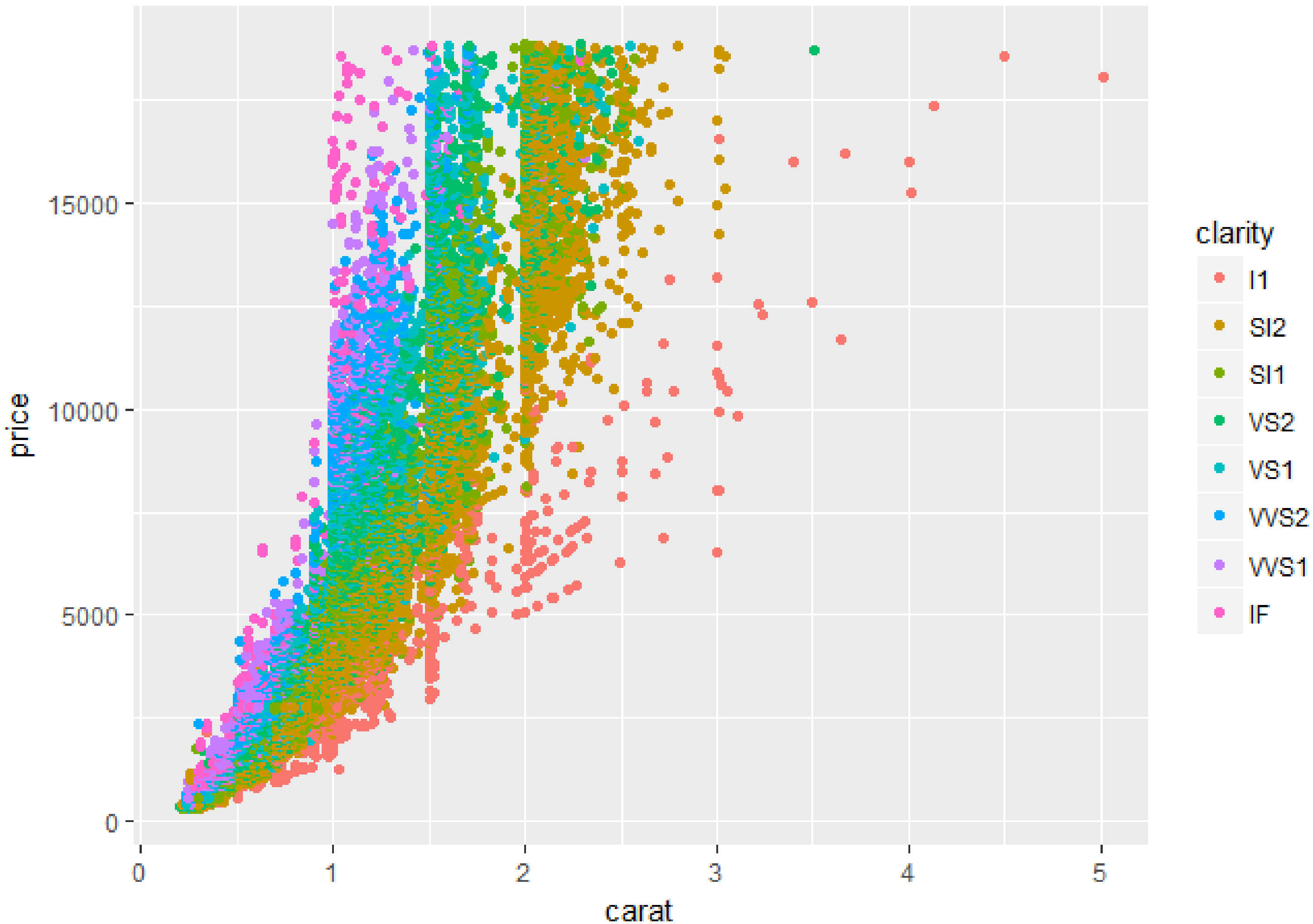

```
diamonds %>%
  count(color, cut) %>%
  ggplot(mapping = aes(x = color, y = cut)) +
    geom_tile(mapping = aes(fill = n))
```

Mini Distributions: Carat vs Price



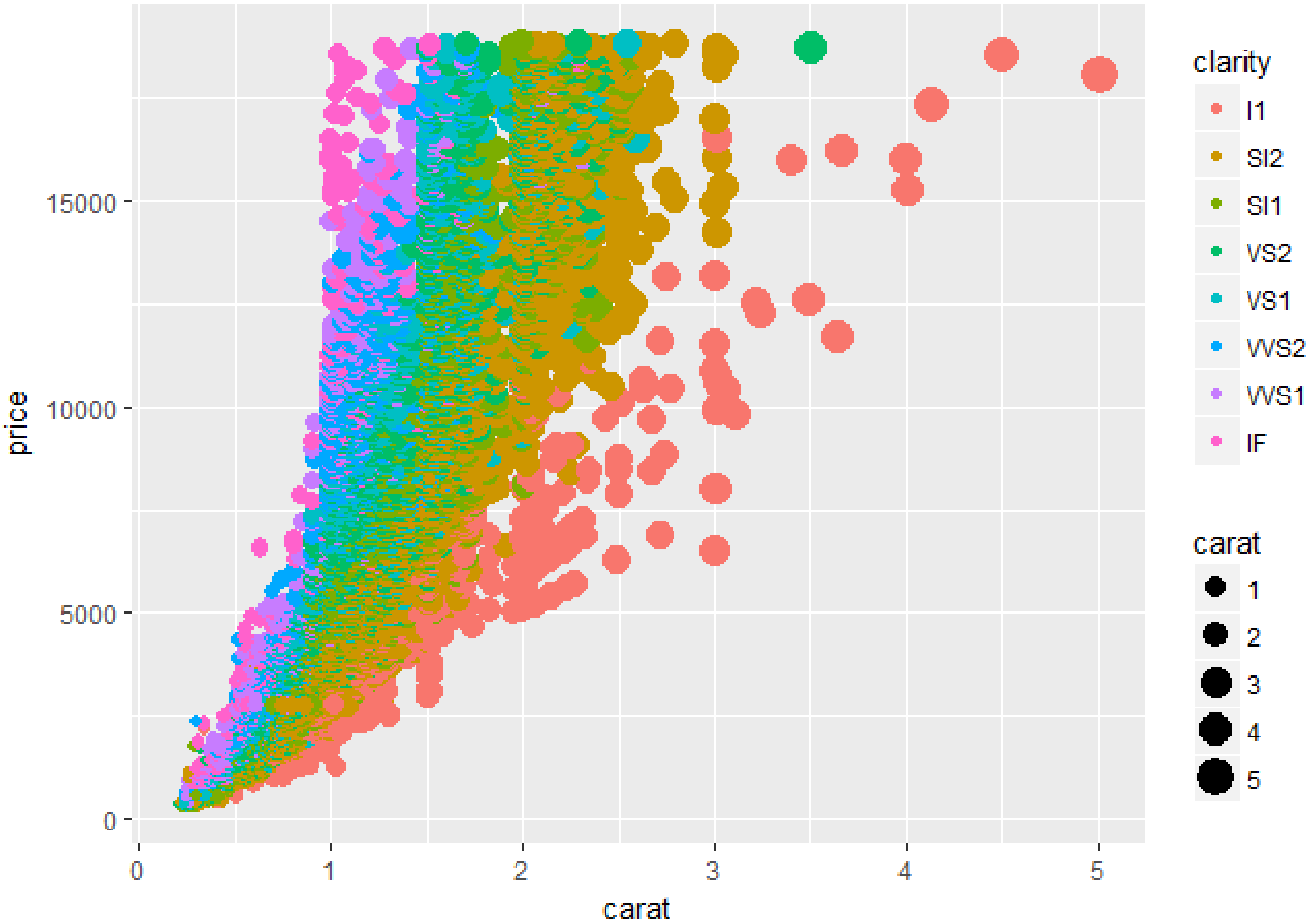

```
ggplot(data = diamonds) +
  geom_point(mapping = aes(x = carat, y = price))
```

Mini Distributions: Carat vs Price



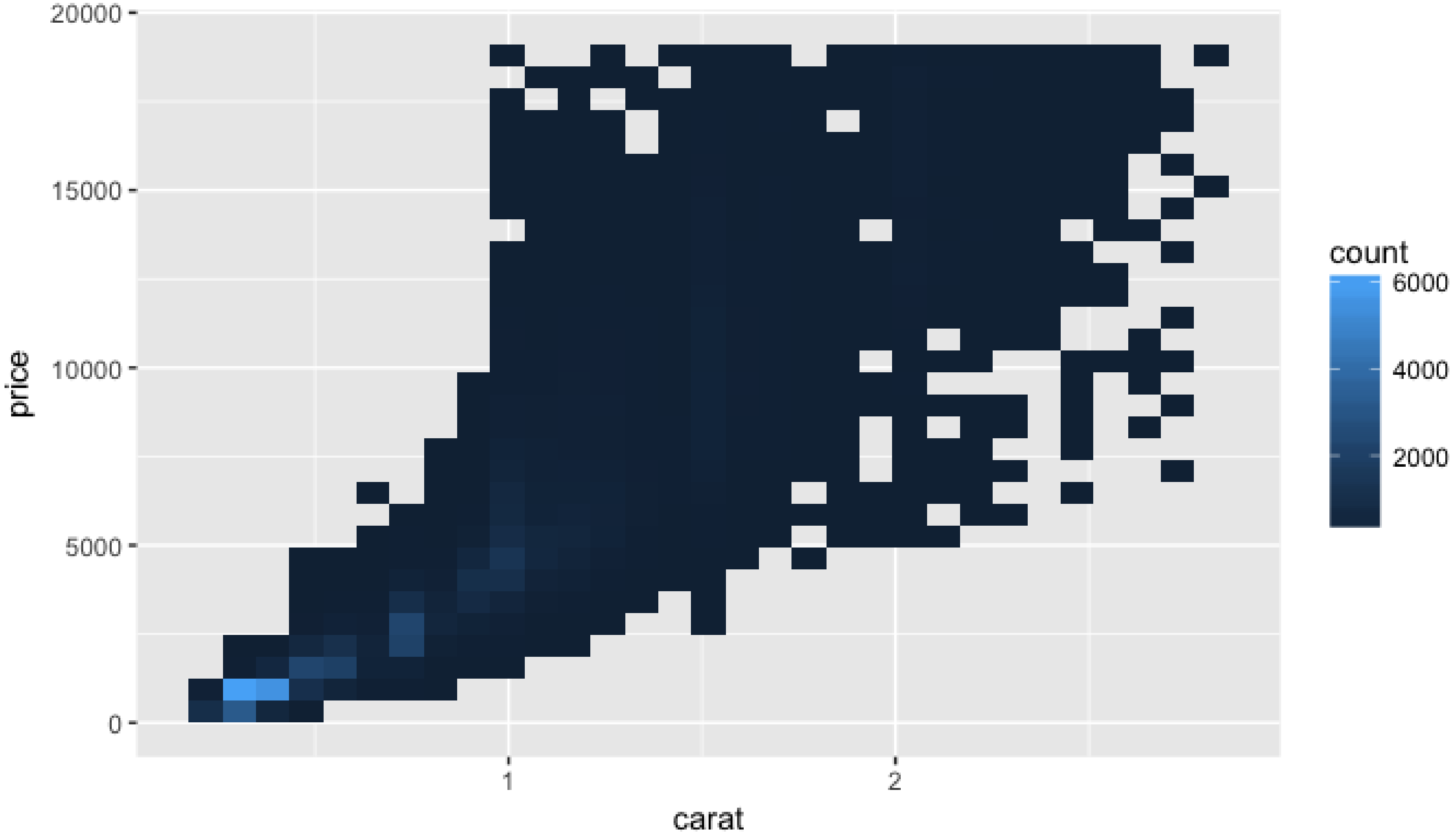

```
ggplot(data = diamonds) +  geom_point(mapping =
aes(x = carat, y = price, color= clarity))
```

Mini Distributions: Carat vs Price




```
ggplot(data = diamonds) +  geom_point(mapping = aes(x = carat, y = price,color= clarity, size = carat))
```

Mini Distributions: Carat vs Price




```
ggplot(data = smaller) +
  geom_bin2d(mapping = aes(x = carat, y = price))
```

Consider This: *plots*

- Can you plot your diamond dataset with different subsets of data? Compare your plots!
- Play with the below code to see what you can isolate to plot
- Time permitting, can you find another dataset apply to plots?

```
diamonds3 <- diamonds %>%
  mutate(y = ifelse(y < ## | y > ##, NA, y))
```