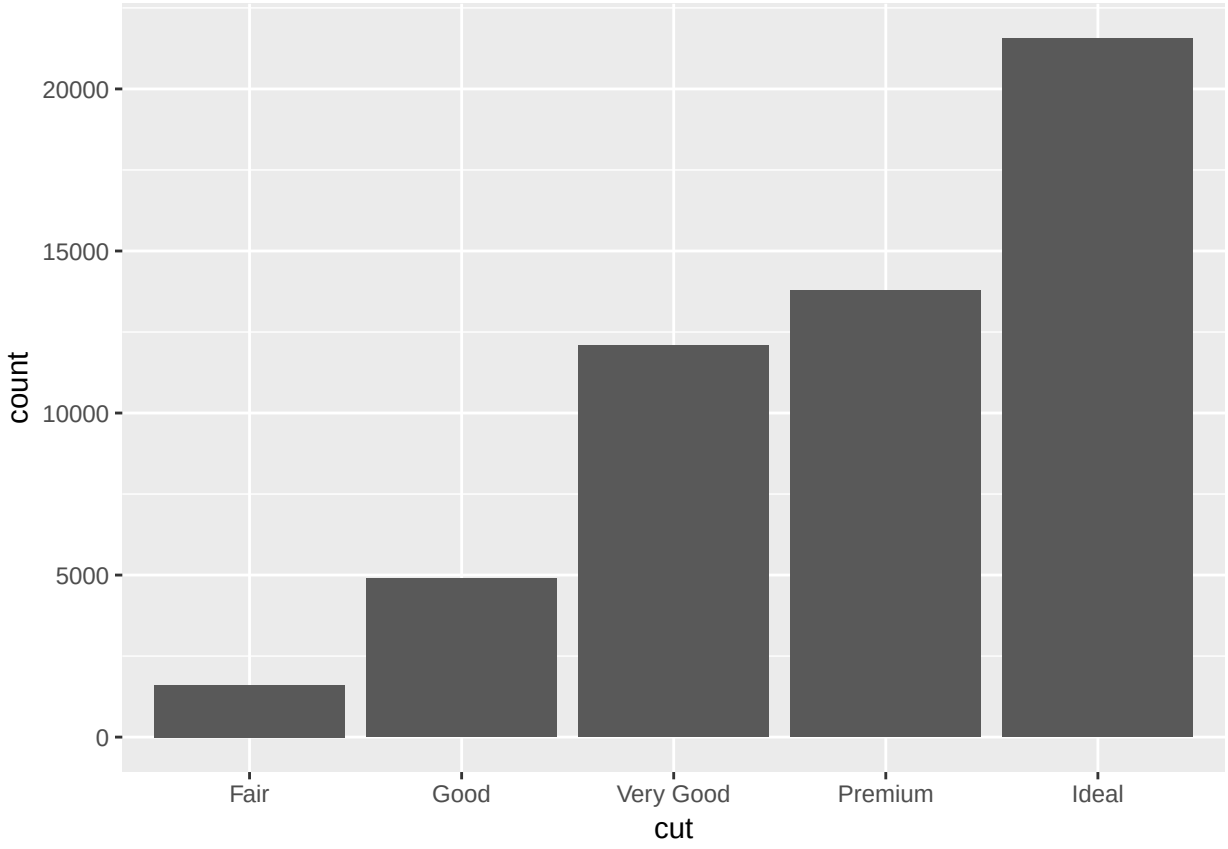
Chapter 5 - Exploratory Data Analysis

Michael Rose

1. What type of variation occurs within my variables?
2. What type of covariation occurs within my variables?

Visualizing Distributions

```
diamonds <- ggplot2::diamonds

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

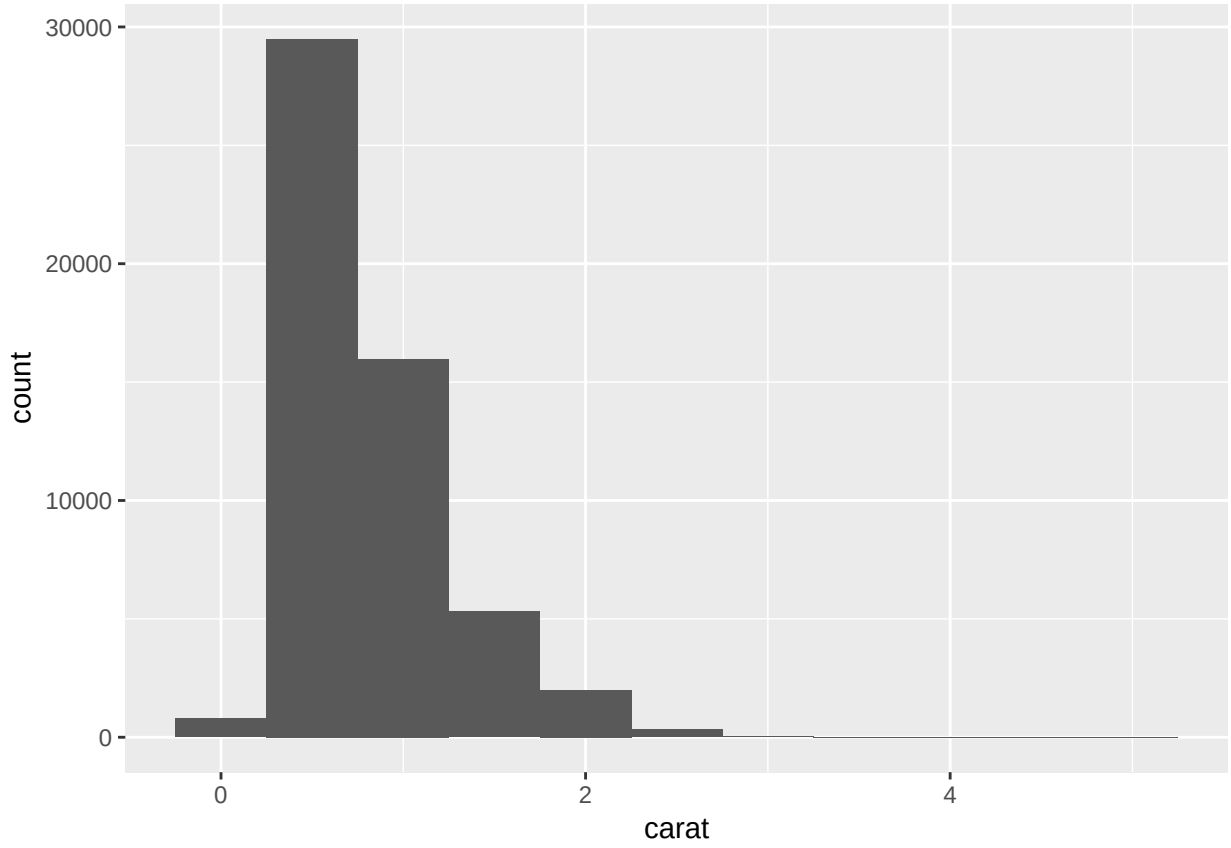



```
# manual counts
diamonds %>%
  count(cut)
```

```
## # 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
```

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

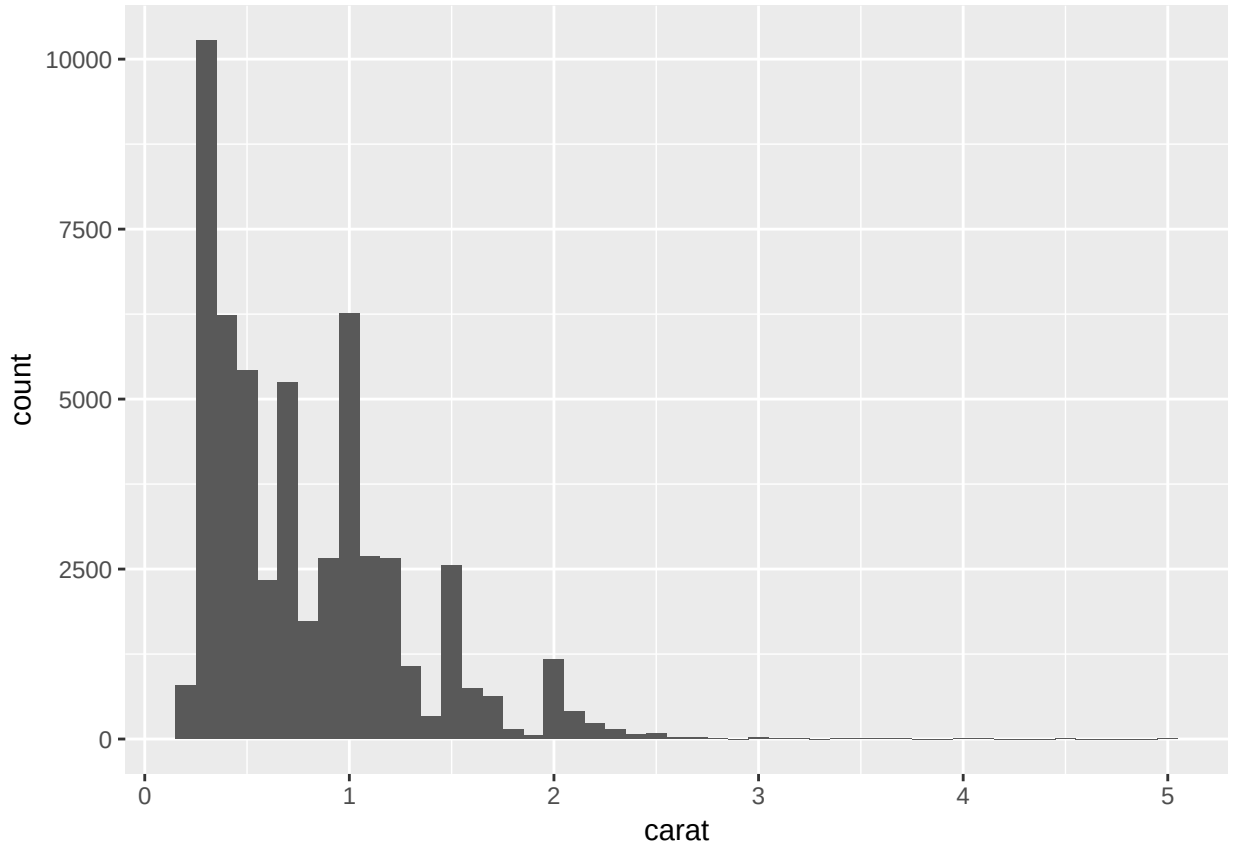



```
# manual
diamonds %>%
  count(cut_width(carat, 0.5))
```

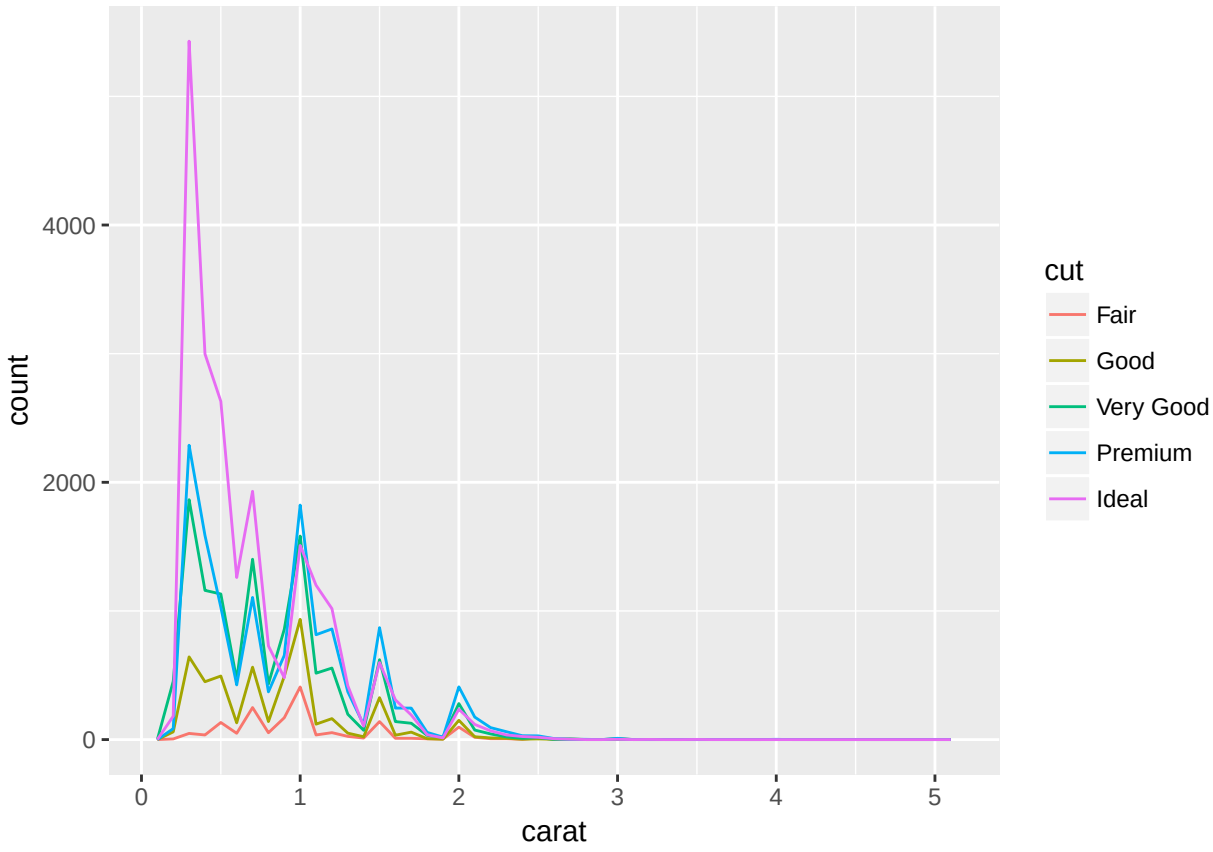
```
## # A tibble: 11 x 2
##    `cut_width(carat, 0.5)`     n
##    <fct>                   <int>
##  1 [-0.25,0.25]              785
##  2 (0.25,0.75]             29498
##  3 (0.75,1.25]             15977
##  4 (1.25,1.75]              5313
##  5 (1.75,2.25]              2002
##  6 (2.25,2.75]               322
##  7 (2.75,3.25]                32
##  8 (3.25,3.75]                 5
##  9 (3.75,4.25]                 4
## 10 (4.25,4.75]                 1
## 11 (4.75,5.25]                 1
```

```
# different binwidth sizes reveal different patterns
smaller <- diamonds %>%
  filter(carat < 3)
```

```r
ggplot(data = diamonds) +
  geom_histogram(mapping = aes(x = carat), binwidth = 0.1)
```




```r
# when dealing with multiple overlapping histograms, its better to use geom_freqpoly
ggplot(data = diamonds, mapping = aes(x = carat, color = cut)) +
  geom_freqpoly(binwidth = 0.1)
```



Typical Values

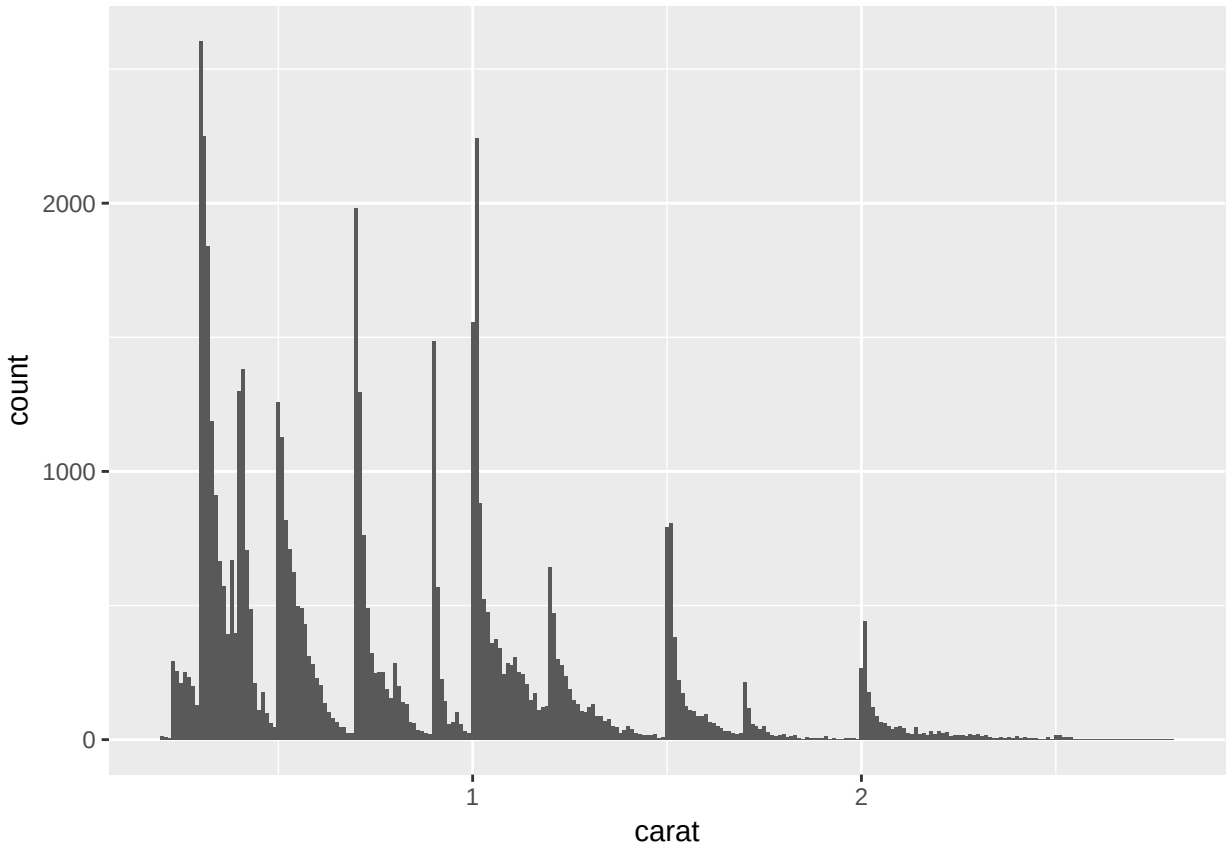
Heights of bars and lines show the abundance or lack thereof of particular values. To turn this into useful questions, consider:

- Which values are the most common and why?
- Which values are rare? Why? Does that match your expectations?
- Can you see any unusual patterns? What might explain them?

Examples for the Diamonds data set:

- Why are there more diamonds at whole carats and common fractions of carats?
- Why are there more diamonds slightly to the right of each peak than there are slightly to the left of each peak?
- Why are no diamonds bigger than 3 carats?

```
ggplot(data = smaller, mapping = aes(x = carat)) +
  geom_histogram(binwidth = 0.01)
```

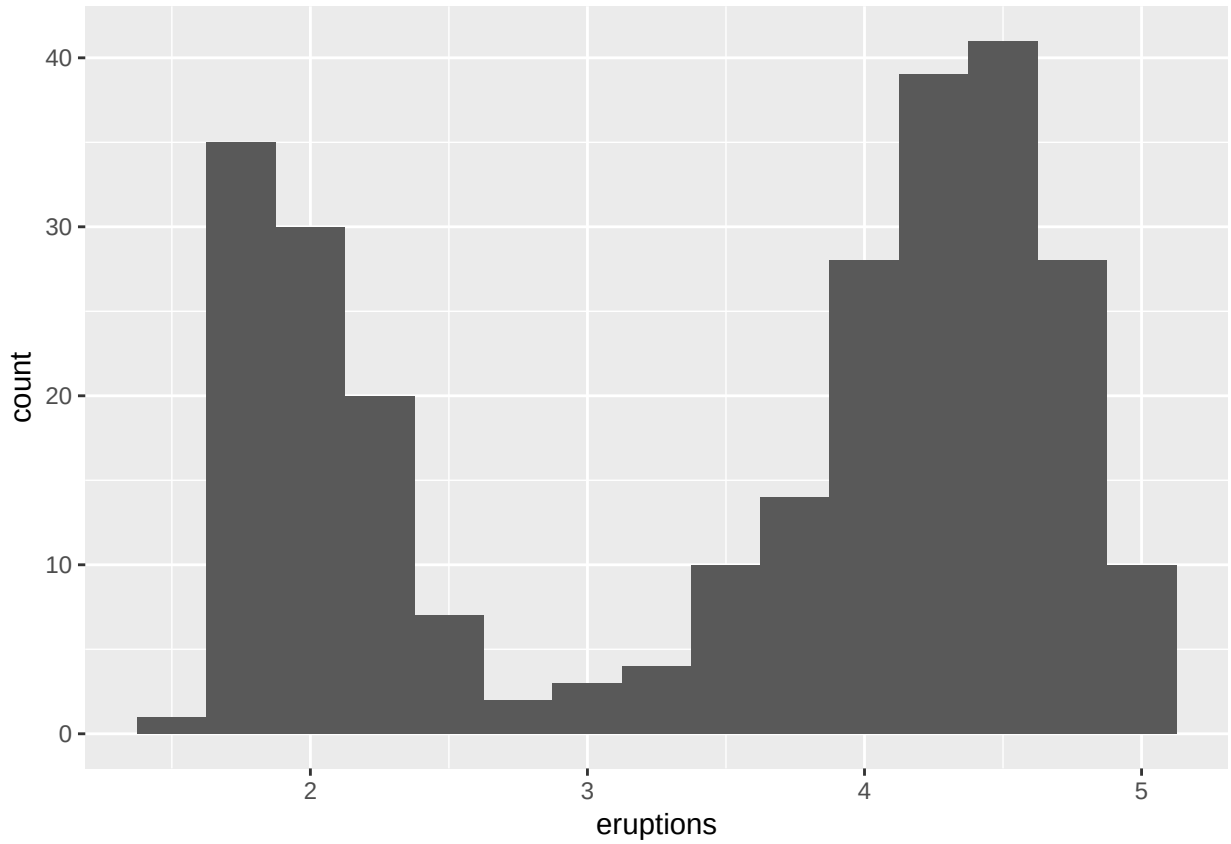


In general, clusters of similar values suggest that subgroups exist in the data. To understand the subgroups, ask:

- How are the observations within each cluster similar to each other?
- How are the observations in seperate clusters different from each other?
- How can you explain or describe the clusters?
- Why might the appearance of clusters be misleading?

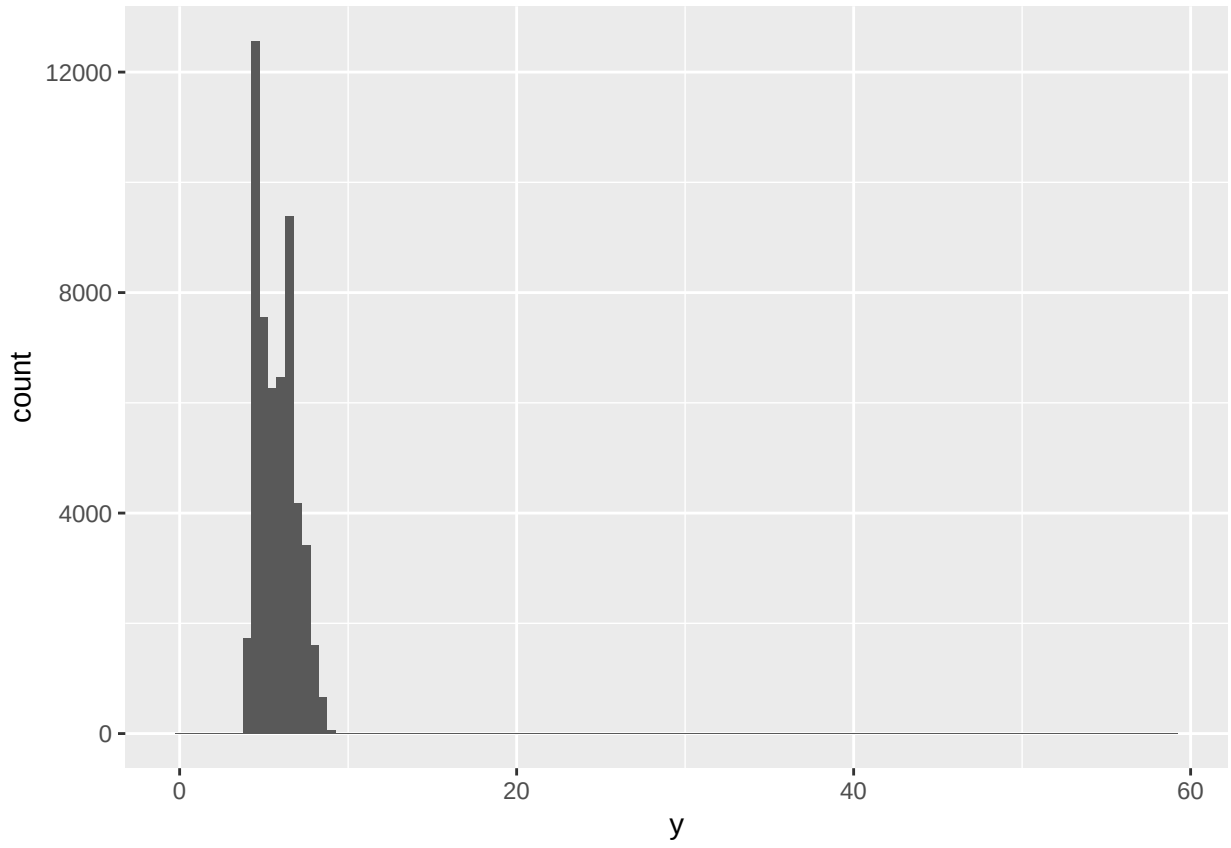
The following histogram shows the length (in minutes) of 272 eruptions of the old faithful geyser in Yellowstone National Park. There are 2 clusters, or groupings of eruptions: those around 2 minutes and those around 4-5 minutes.

```
ggplot(data = faithful, mapping = aes(x = eruptions)) +
  geom_histogram(binwidth = 0.25)
```

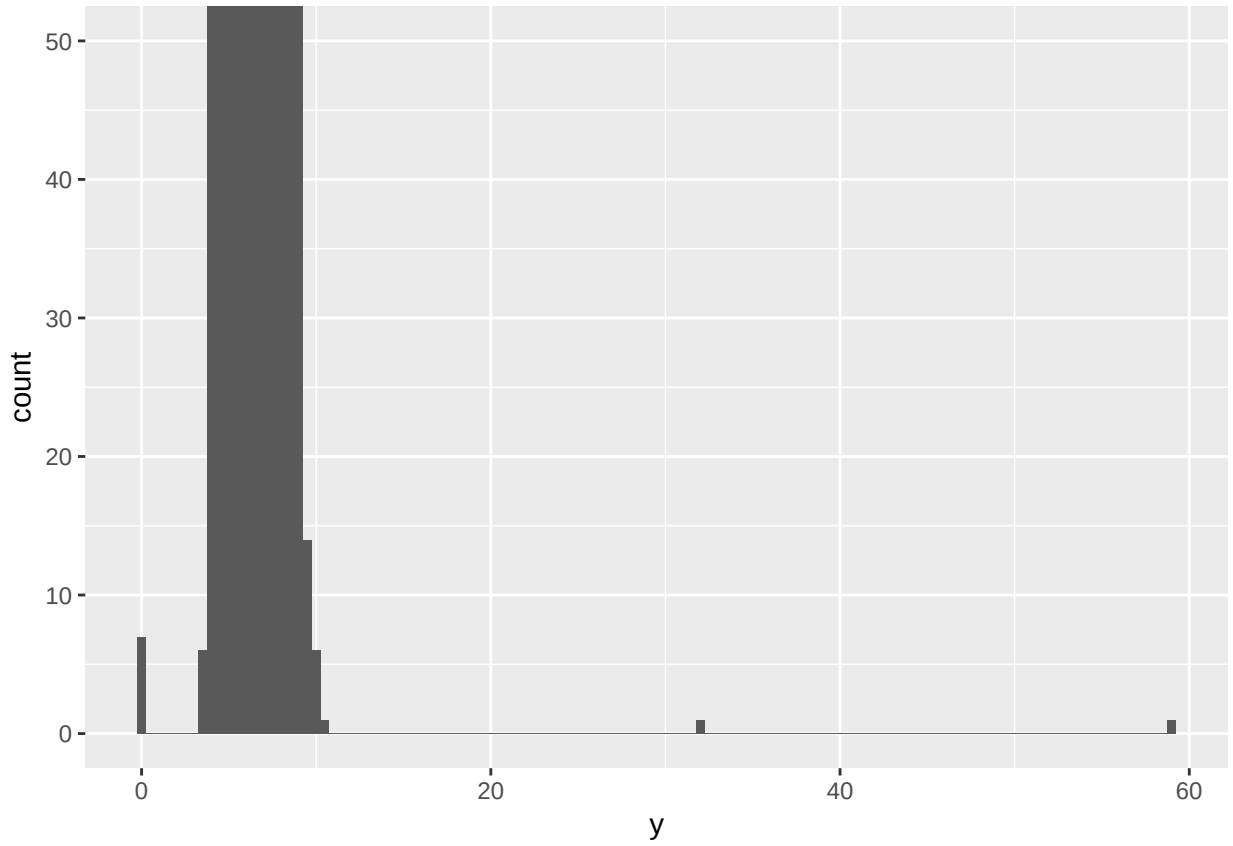


Unusual Values

```
# notice the long x axis. This is indicative of an outlier.
ggplot(diamonds) +
  geom_histogram(mapping = aes(x = y), binwidth = 0.5)
```




```
# make it easier to see
ggplot(diamonds) +
  geom_histogram(mapping = aes(x = y), binwidth = 0.5) +
  coord_cartesian(ylim = c(0, 50))
```

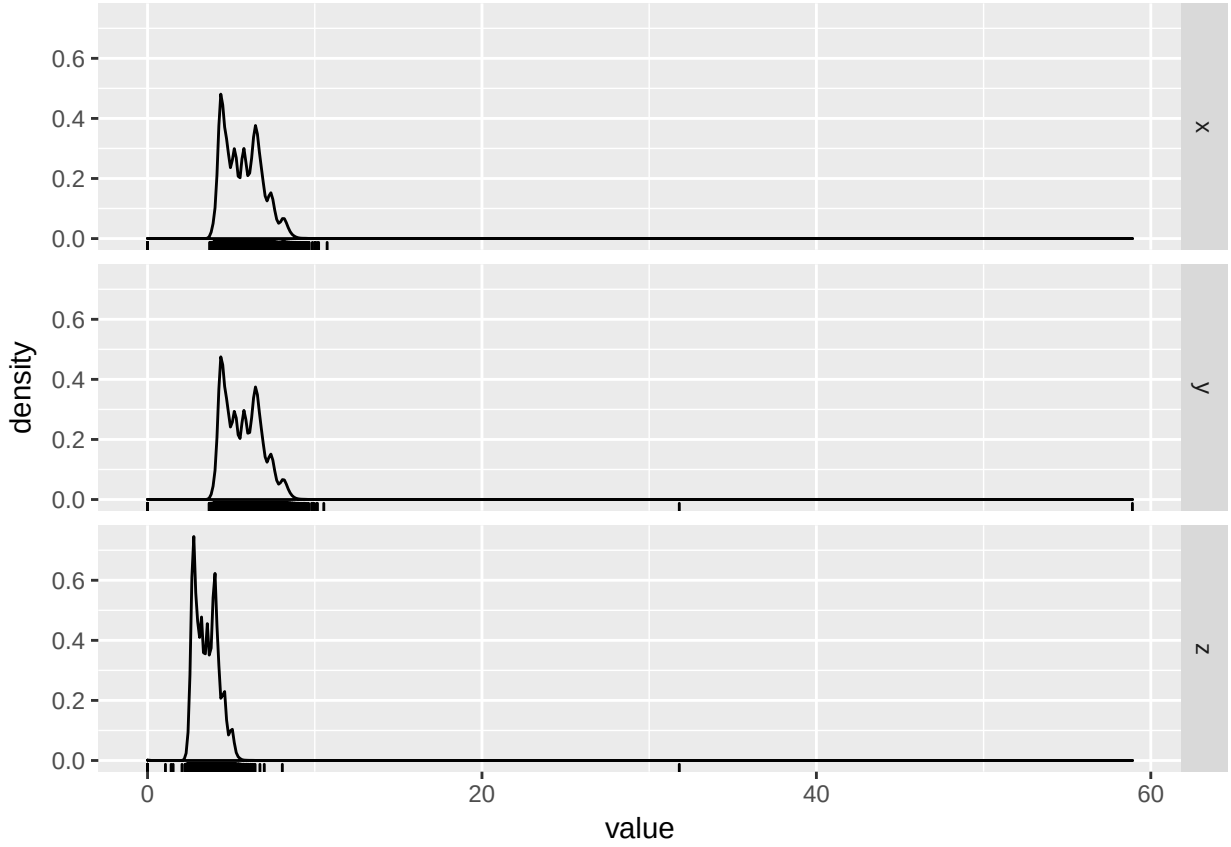



```
# pluck out the data points with dplyr
unusual <- diamonds %>%
  filter(y < 3 | y > 20) %>%
  arrange(y)
unusual
```

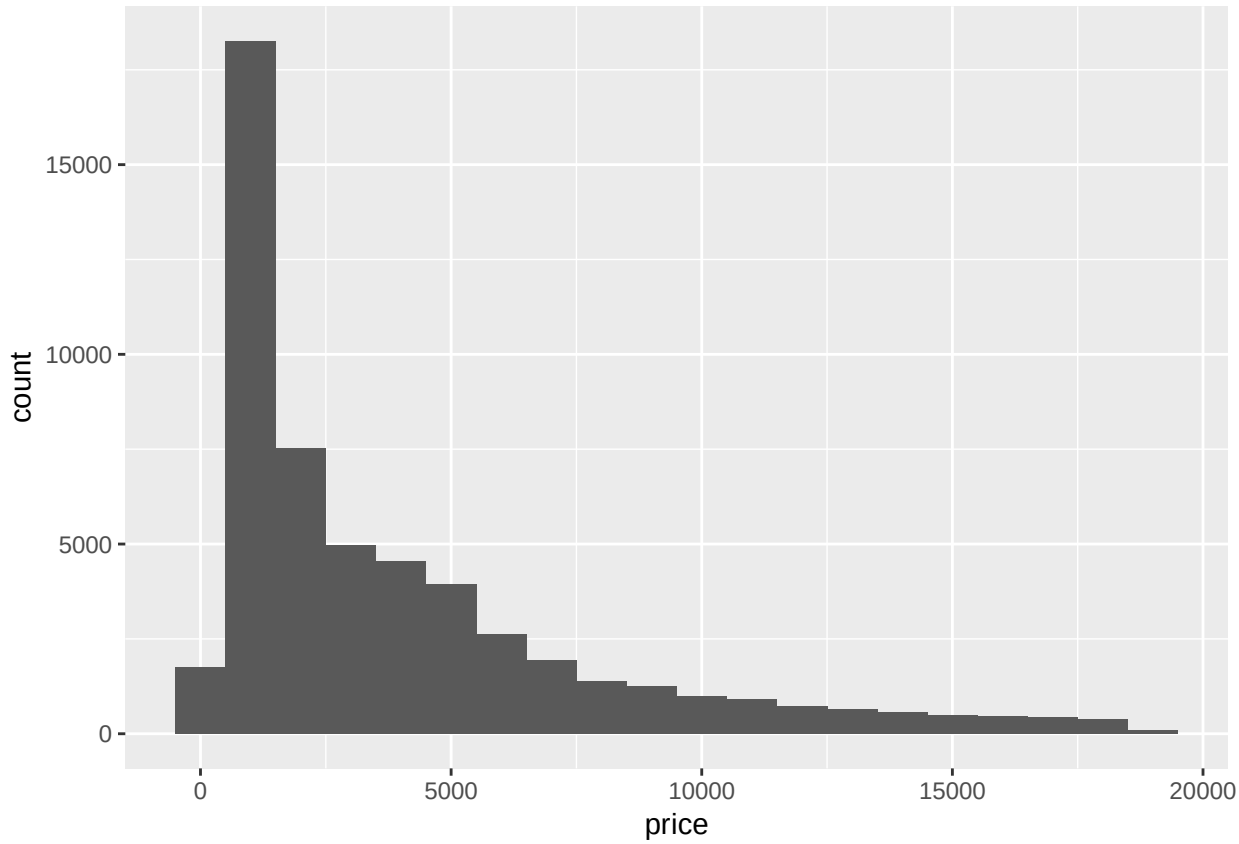
```
## # A tibble: 9 x 10
##   carat cut       color clarity depth table price     x     y     z
##   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1  1    Very Good H     VS2      63.3    53  5139  0      0    0
## 2  1.14 Fair      G     VS1      57.5    67  6381  0      0    0
## 3  1.56 Ideal     G     VS2      62.2    54 12800  0      0    0
## 4  1.2  Premium   D     VVS1     62.1    59 15686  0      0    0
## 5  2.25 Premium   H     SI2      62.8    59 18034  0      0    0
## 6  0.71 Good      F     SI2      64.1    60  2130  0      0    0
## 7  0.71 Good      F     SI2      64.1    60  2130  0      0    0
## 8  0.51 Ideal     E     VS1      61.8    55  2075  5.15  31.8  5.12
## 9  2    Premium   H     SI2      58.9    57 12210  8.09  58.9  8.06
```

```
# ctrl alt space %>%

# explore the distribution of x, y, and z

diamonds %>%
  mutate(id = row_number()) %>%
  select(x, y, z, id) %>%
  gather(variable, value, -id) %>%
  ggplot(aes(x = value)) +
```

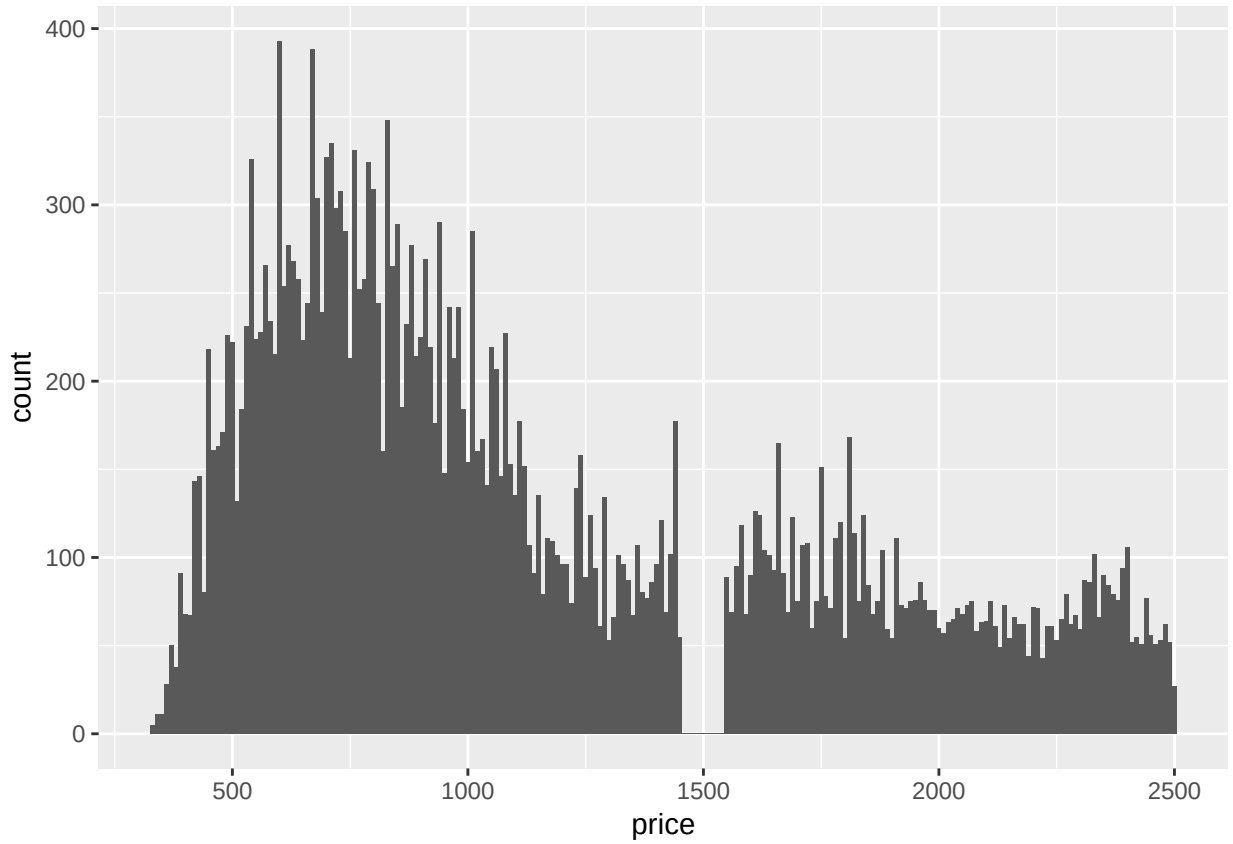
```
  geom_density() +
  geom_rug() +
  facet_grid(variable ~.)
```



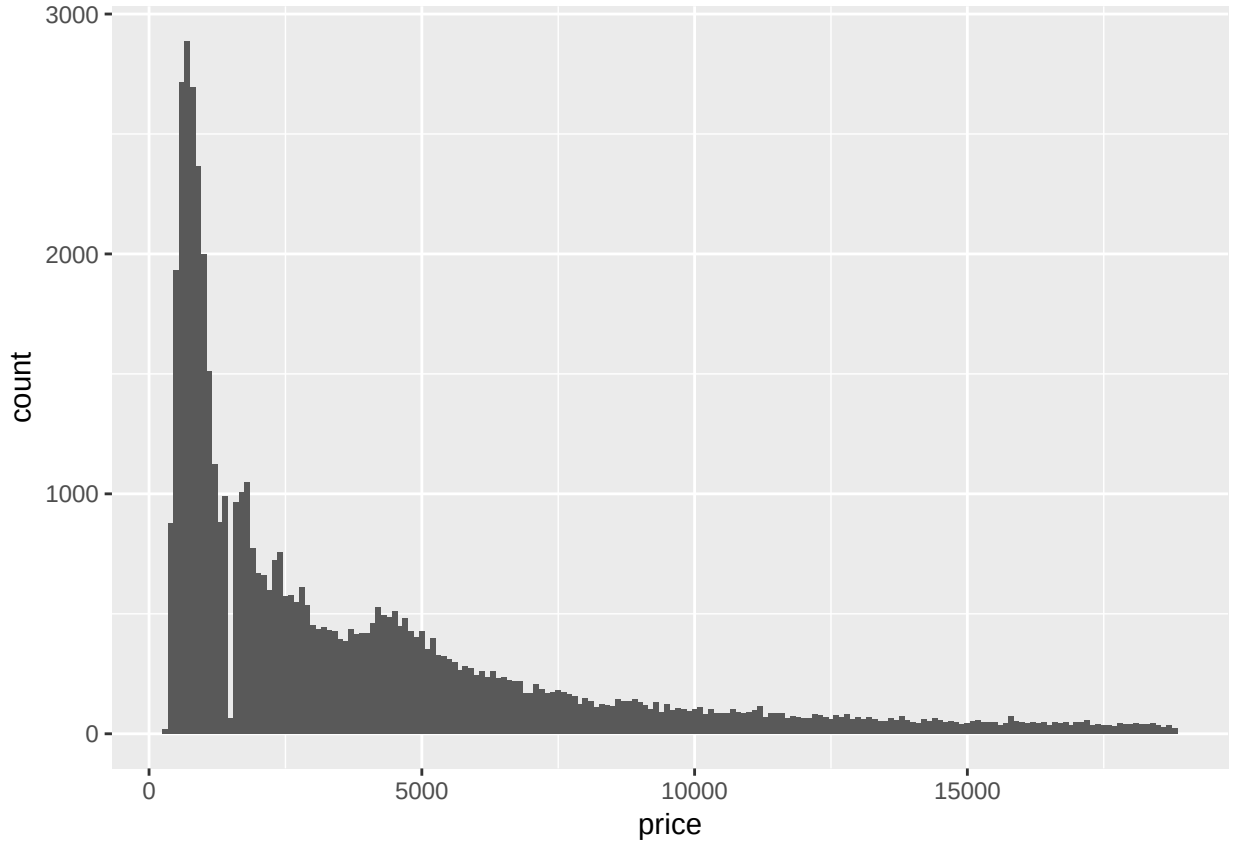

```
# explore the distribution of price

ggplot(diamonds) +
  geom_histogram(mapping = aes(x = price), binwidth = 1000)
```



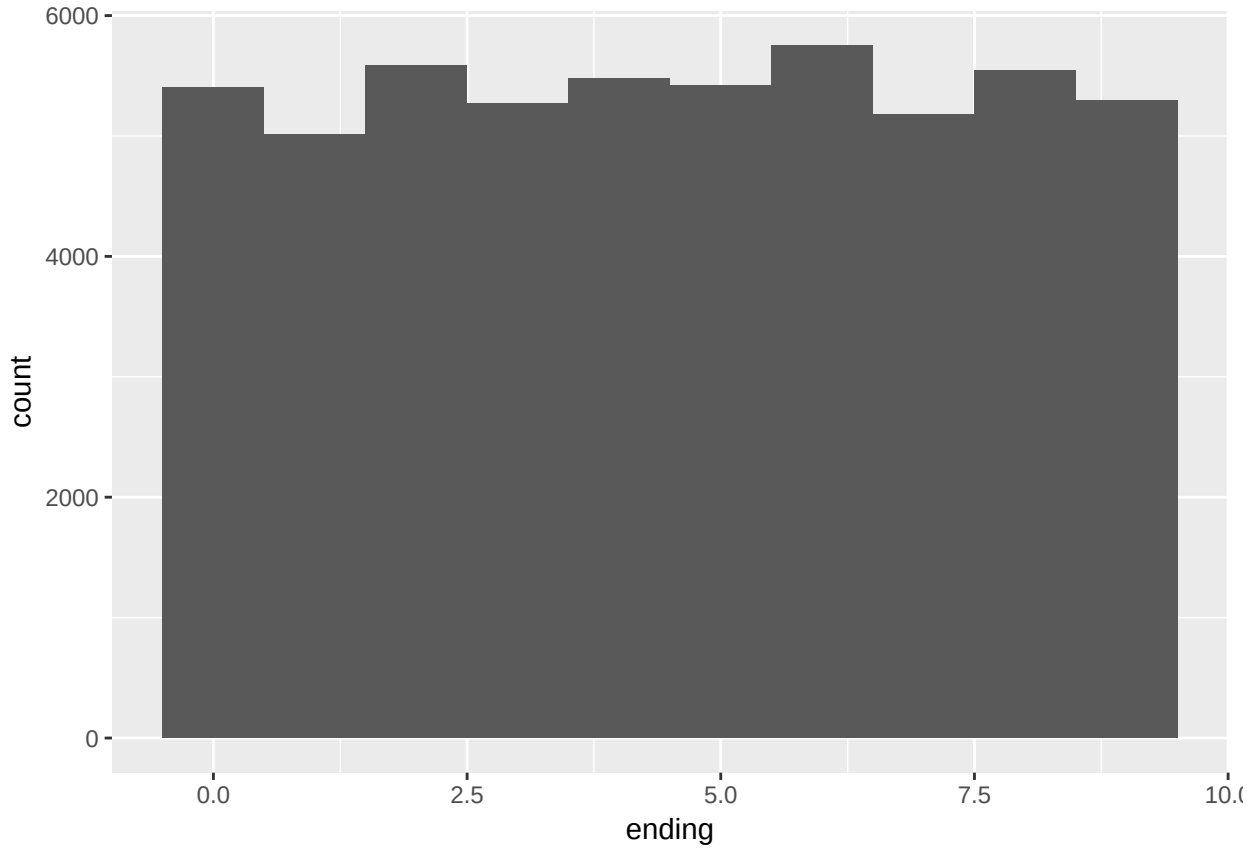

```
ggplot(filter(diamonds, price < 2500), aes(x = price)) +
  geom_histogram(binwidth = 10, center = 0)
```



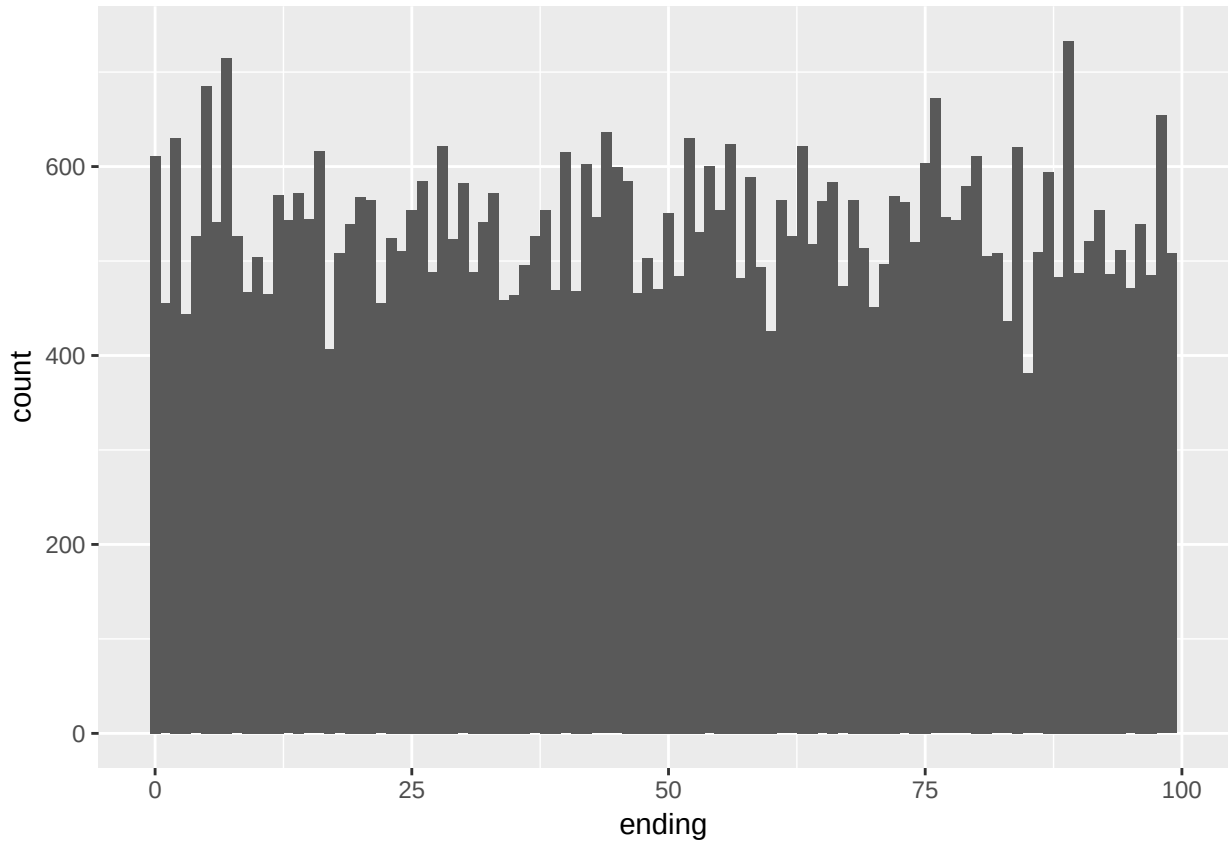

```
ggplot(filter(diamonds), aes(x = price)) +
  geom_histogram(binwidth = 100, center = 0)
```



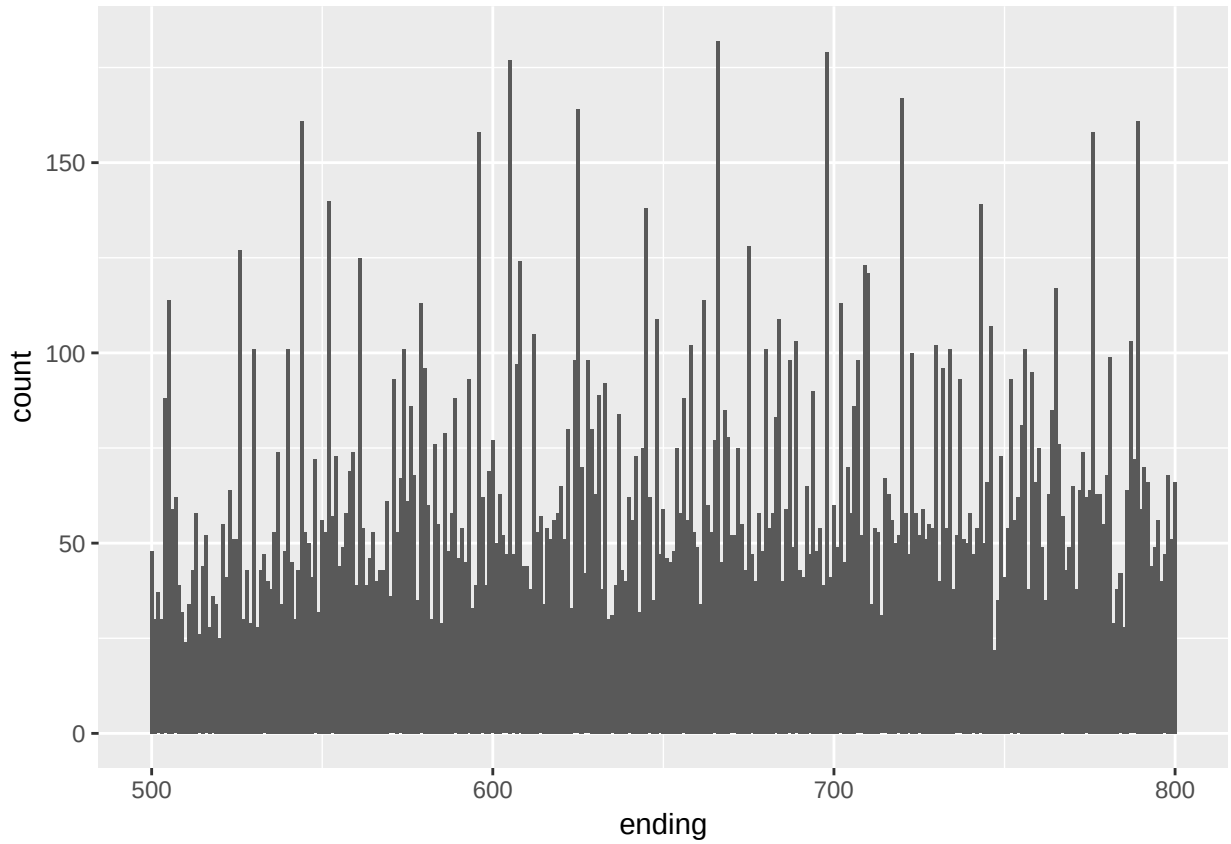

```r
# Distributionn of last digit
diamonds %>%
  mutate(ending = price %% 10) %>%
  ggplot(aes(x = ending)) +
  geom_histogram(binwidth = 1, center = 0) +
  geom_bar()
```



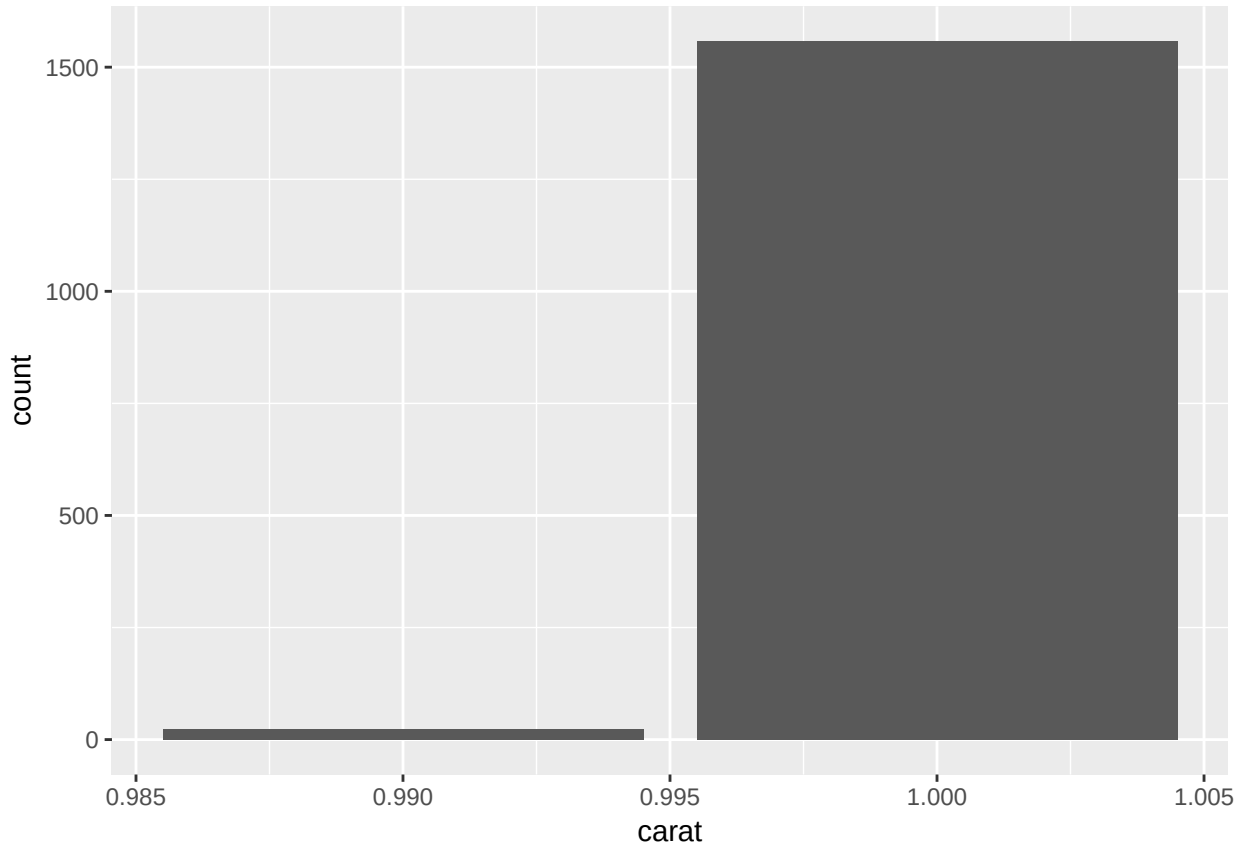

```
diamonds %>%
  mutate(ending = price %% 100) %>%
  ggplot(aes(x = ending)) +
  geom_histogram(binwidth = 1) +
  geom_bar()
```




```
diamonds %>%
  mutate(ending = price %% 1000) %>%
  filter(ending >= 500, ending <= 800) %>%
  ggplot(aes(x = ending)) +
  geom_histogram(binwidth = 1) +
  geom_bar()
```

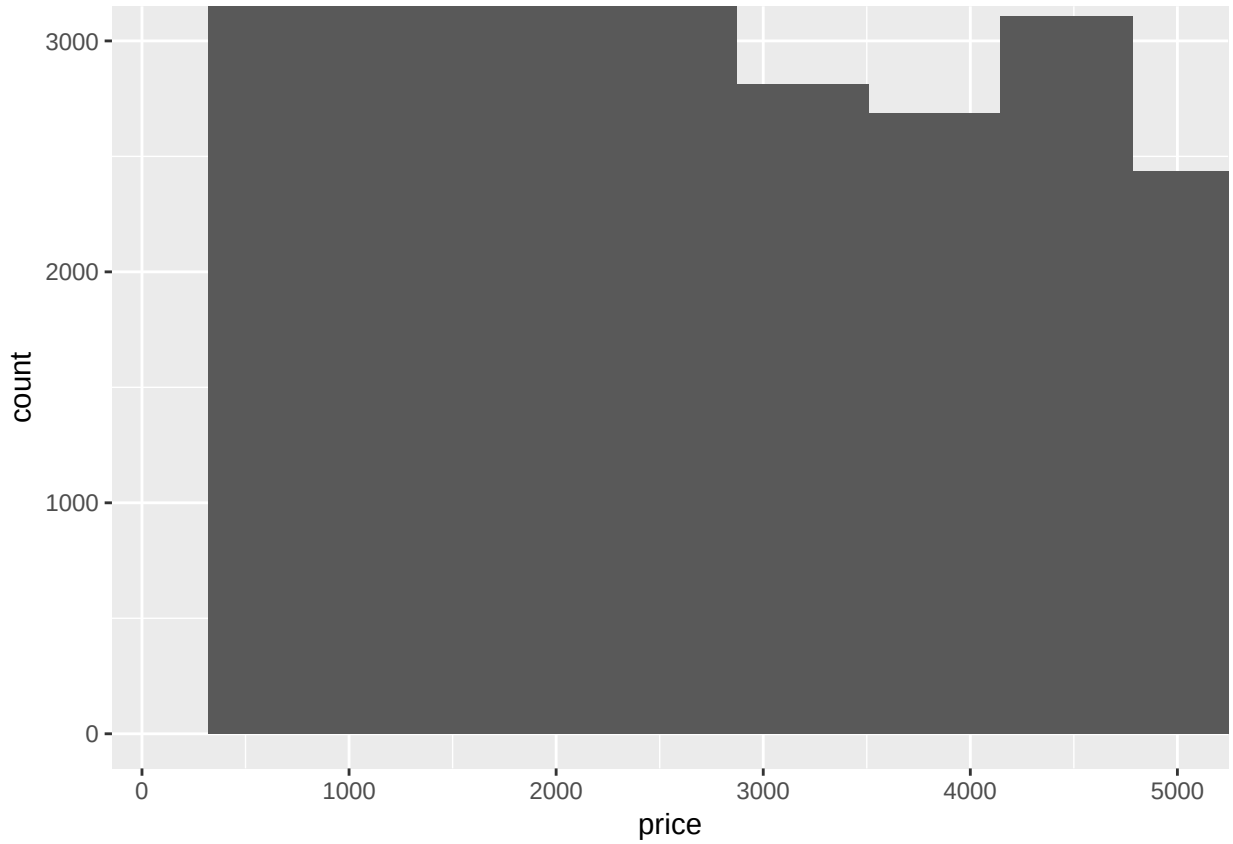



```
# How many diamonds are 0.99 carat? How many are 1 carat?

diamonds %>%
  filter(carat >= 0.99, carat <= 1) %>%
  ggplot(aes(x = carat)) +
  geom_bar()
```



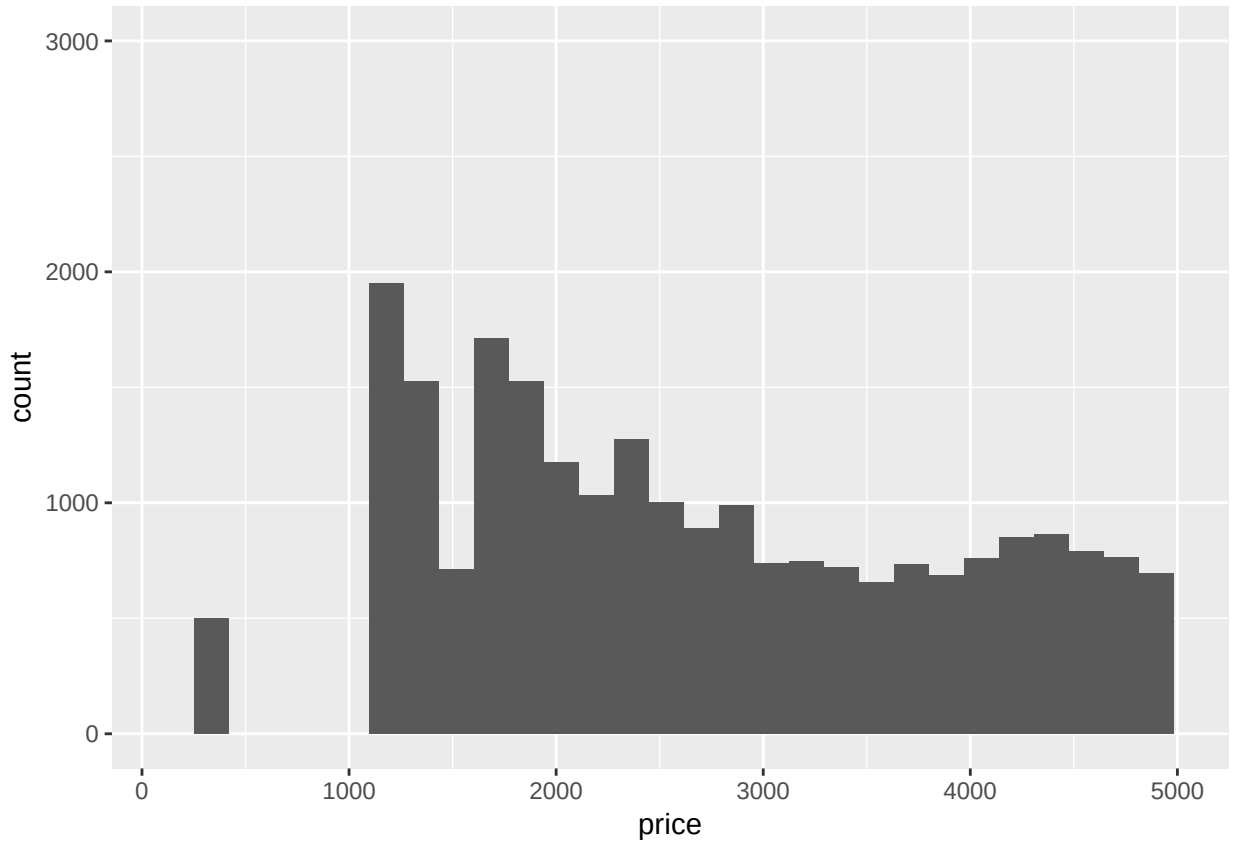

```
#coord_cartesian vs. xlim, ylim

ggplot(diamonds) +
  geom_histogram(mapping = aes(x = price)) +
  coord_cartesian(xlim = c(100, 5000), ylim = c(0, 3000))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



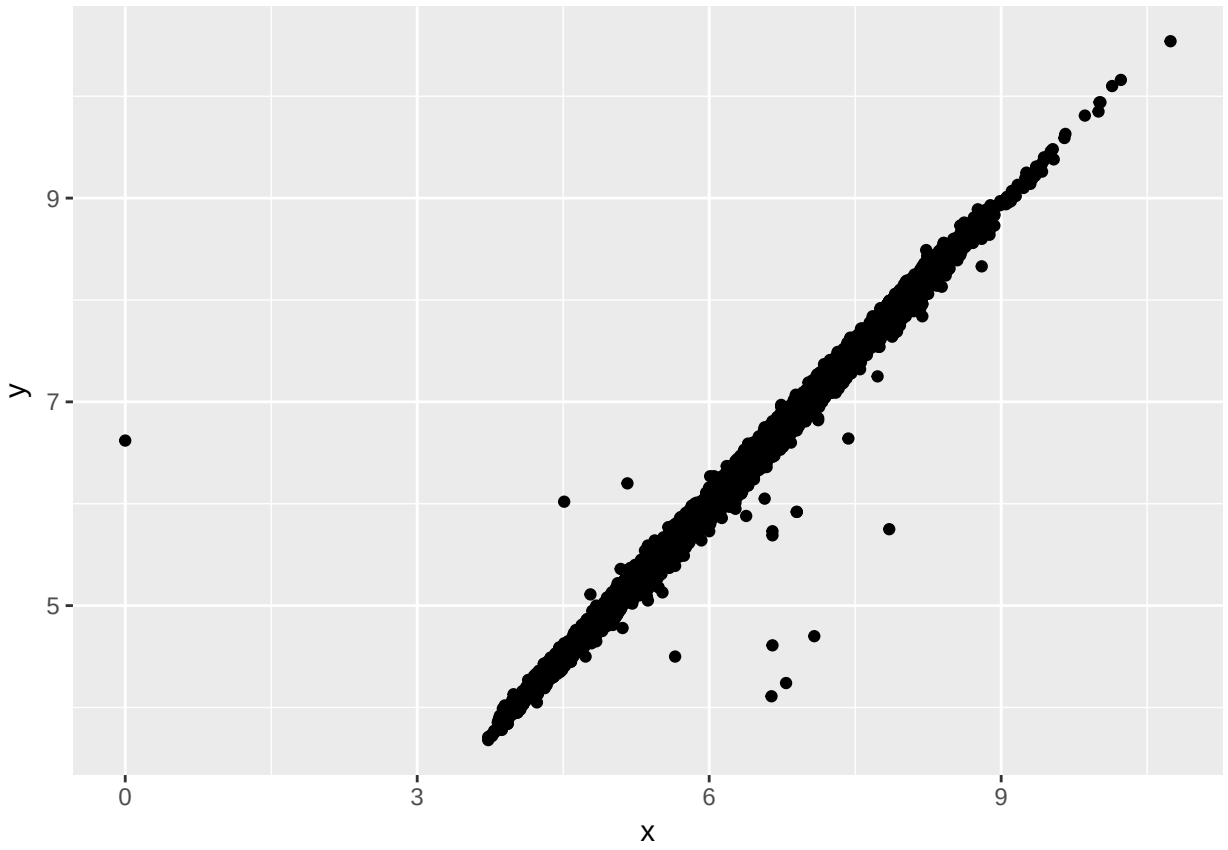

```
# coord_cartesian zooms in on the area specified by the limits. The calculation of the histogram is una

# xlim and ylim functions drop any values outside the limits, then calculate the histogram and then dra

ggplot(diamonds) +
  geom_histogram(mapping = aes(x = price)) +
  xlim(100, 5000) +
  ylim(0, 3000)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## Warning: Removed 14714 rows containing non-finite values (stat_bin).
## Warning: Removed 5 rows containing missing values (geom_bar).
```

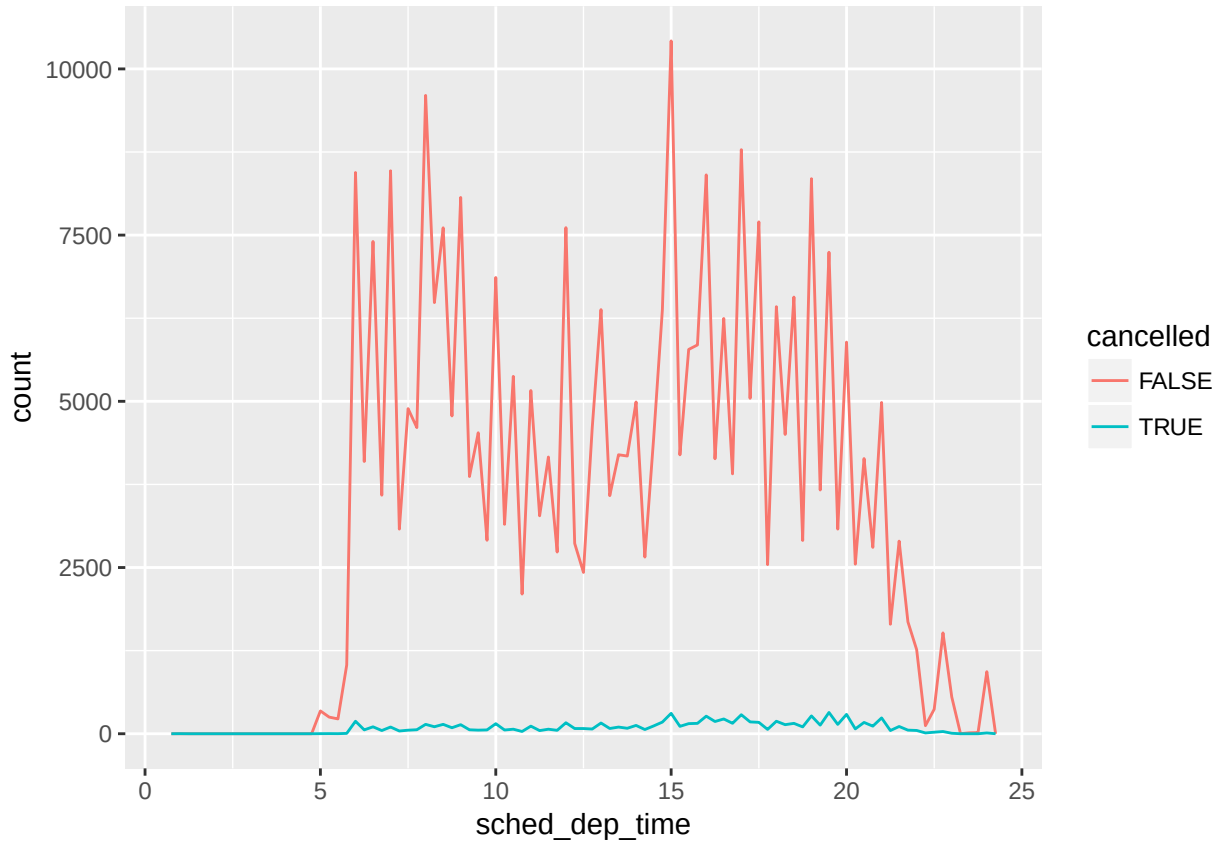


Missing Values

```
# if we encounter unusual values, it is better to make them into NA values than to delete them

# drop the entire row (don't do):
# diamonds2 <- diamonds %>% filter(between(y, 3, 20))

# replace with NA (recc'd):
diamonds2 <- diamonds %>%
  mutate(y = ifelse(y <3 | y > 20, NA, y))

ggplot(data = diamonds2, mapping = aes(x = x, y = y)) +
  geom_point(na.rm = TRUE)
```

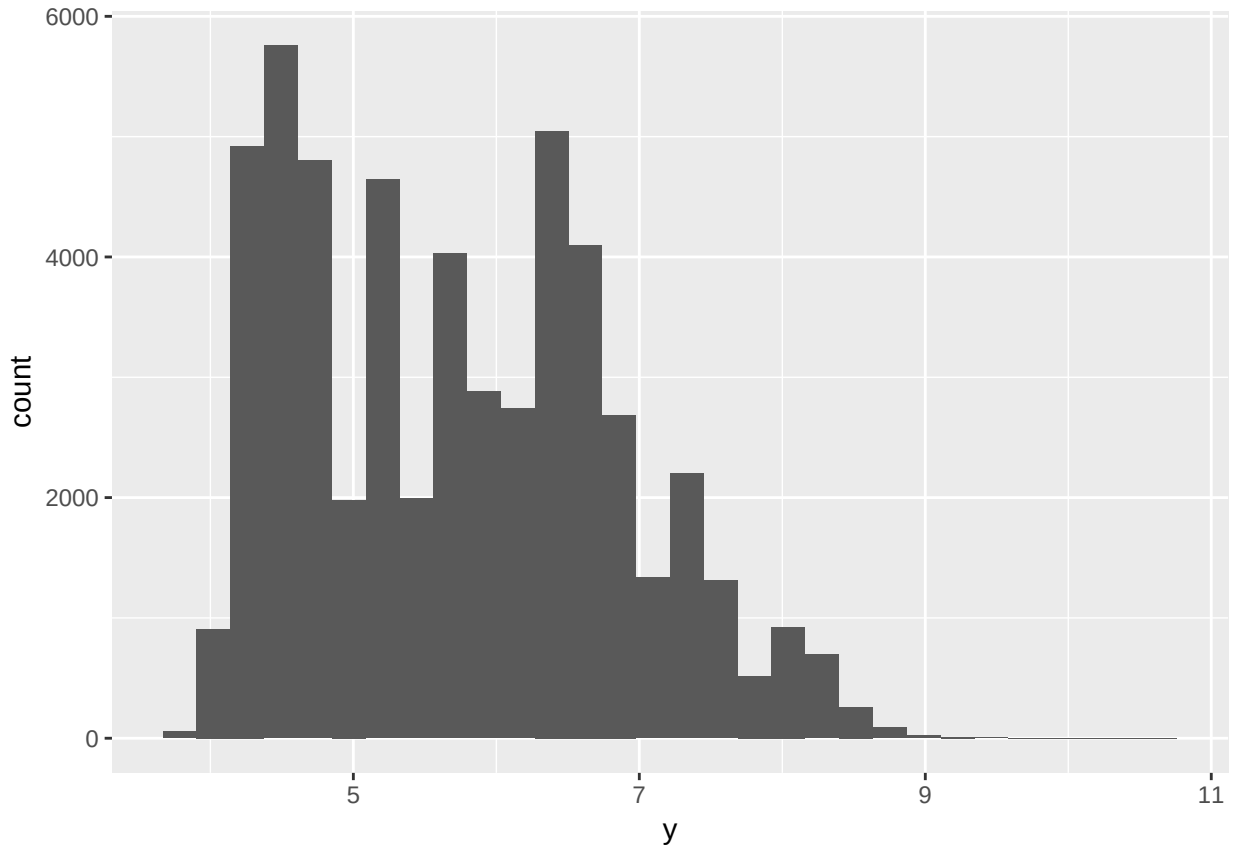



```
# to compare values that are na with values that are not na

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(color = cancelled),
      binwidth = 1/4
    )
```

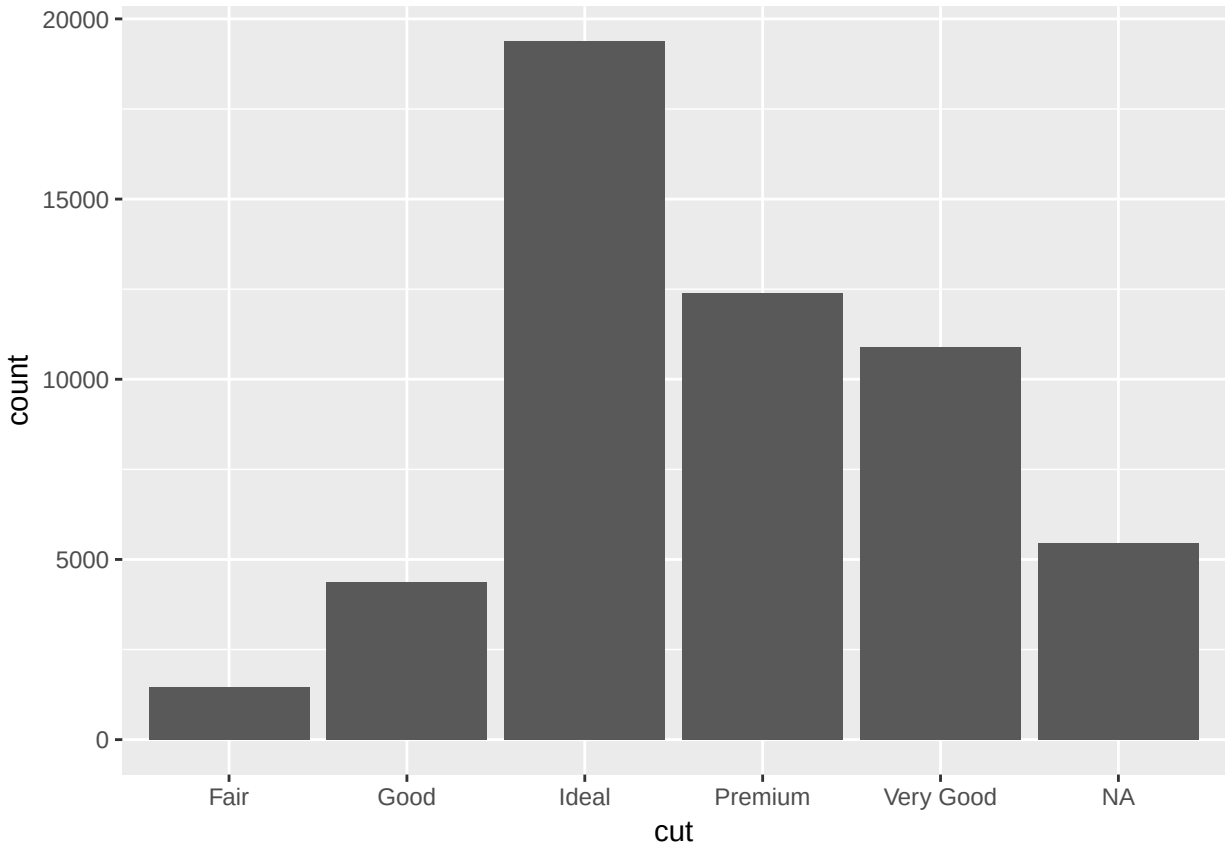



```
# Missing values are removed in a histogram.

ggplot(diamonds2, aes(x = y)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## Warning: Removed 9 rows containing non-finite values (stat_bin).
```



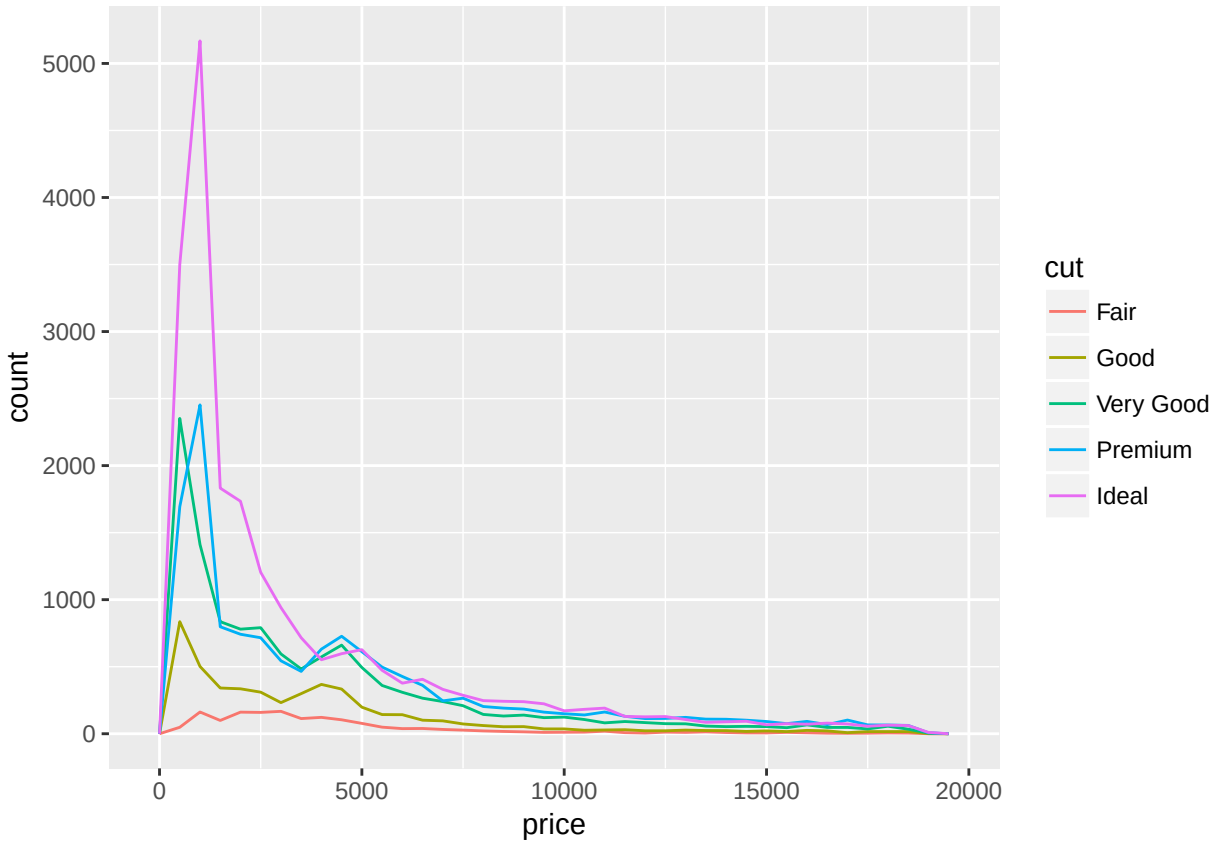

```
# in geom_bar, NA is considered a category

diamonds %>%
  mutate(cut = if_else(runif(n()) < 0.1, NA_character_, as.character(cut))) %>%
  ggplot() +
  geom_bar(mapping = aes(x = cut))
```



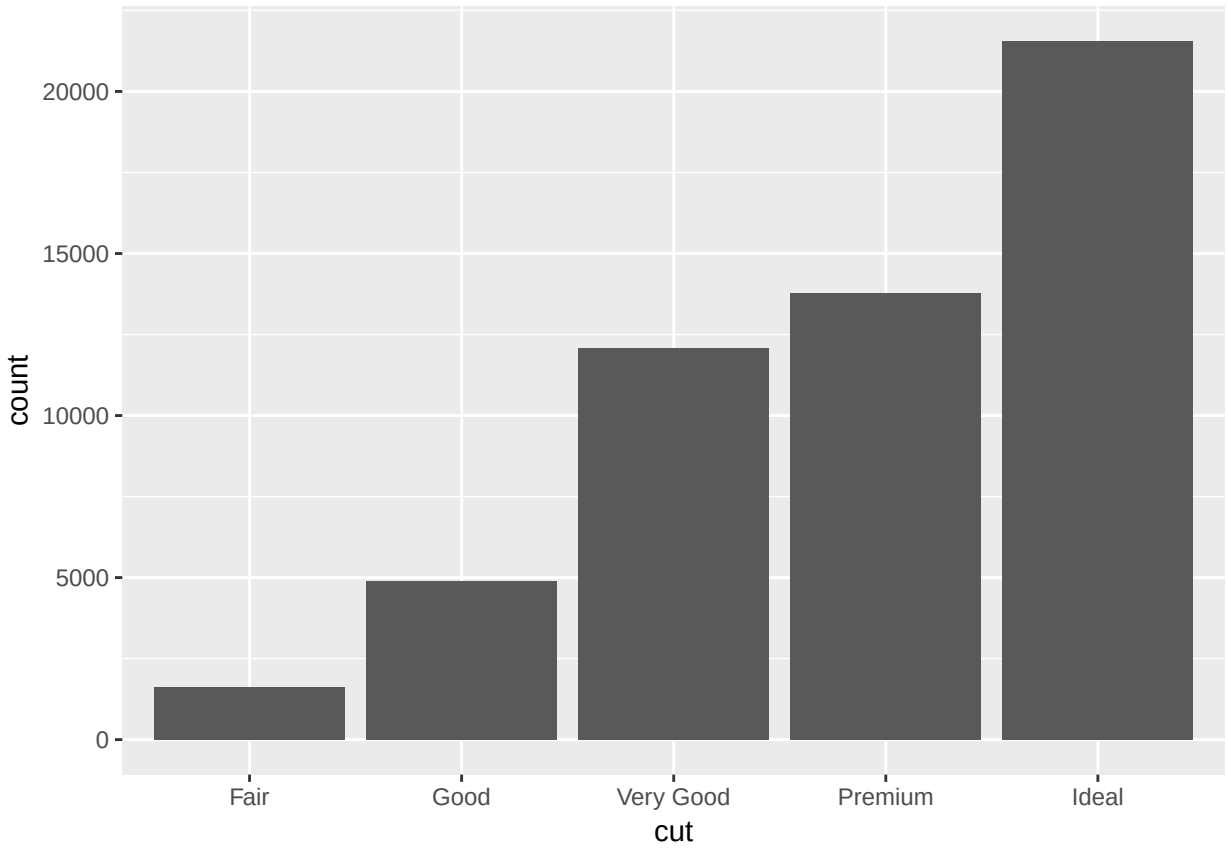
Covariation

Covariance describes the behavior between variables. Covariation is the tendency for the values or 2 or more variables to vary together in a related way.

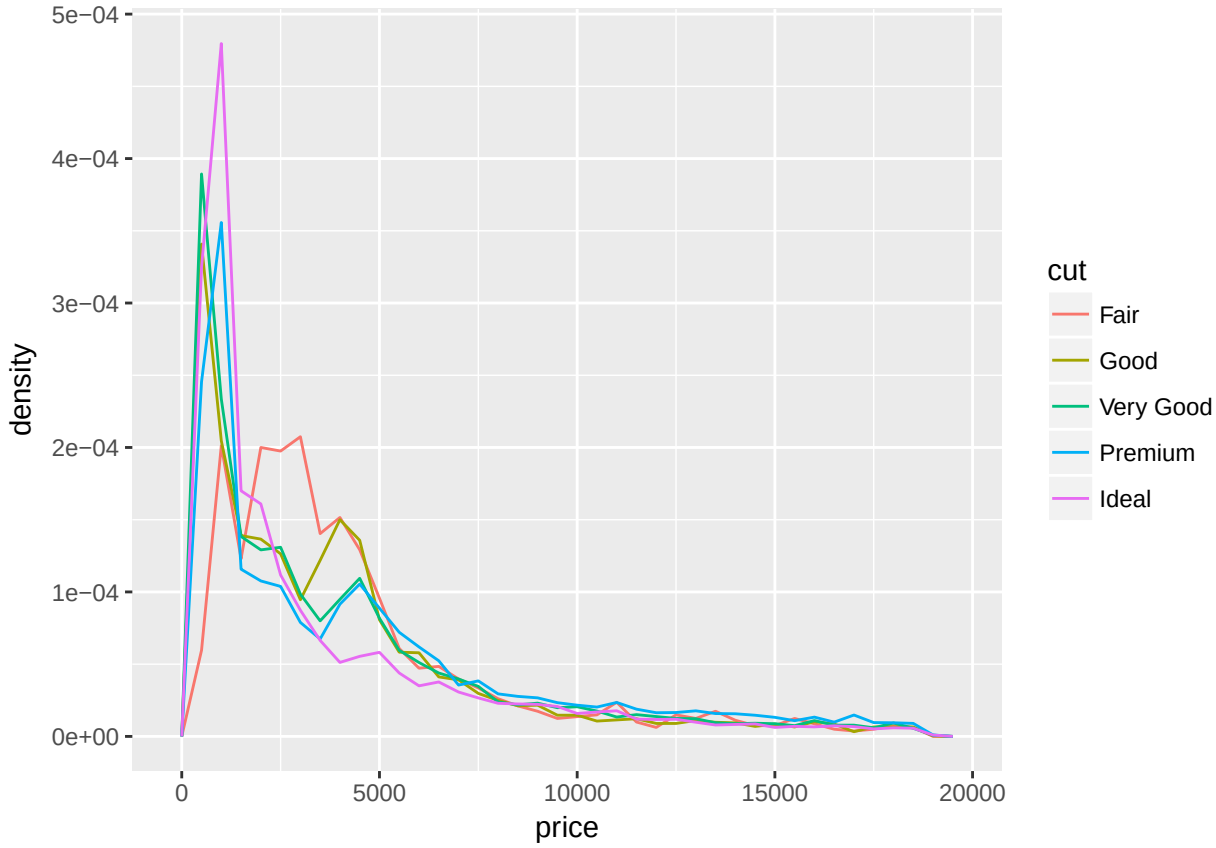
```
# check how the price of a diamond varies with its quality
ggplot(data = diamonds, mapping = aes(x = price)) +
  geom_freqpoly(mapping = aes(color = cut), binwidth = 500)
```



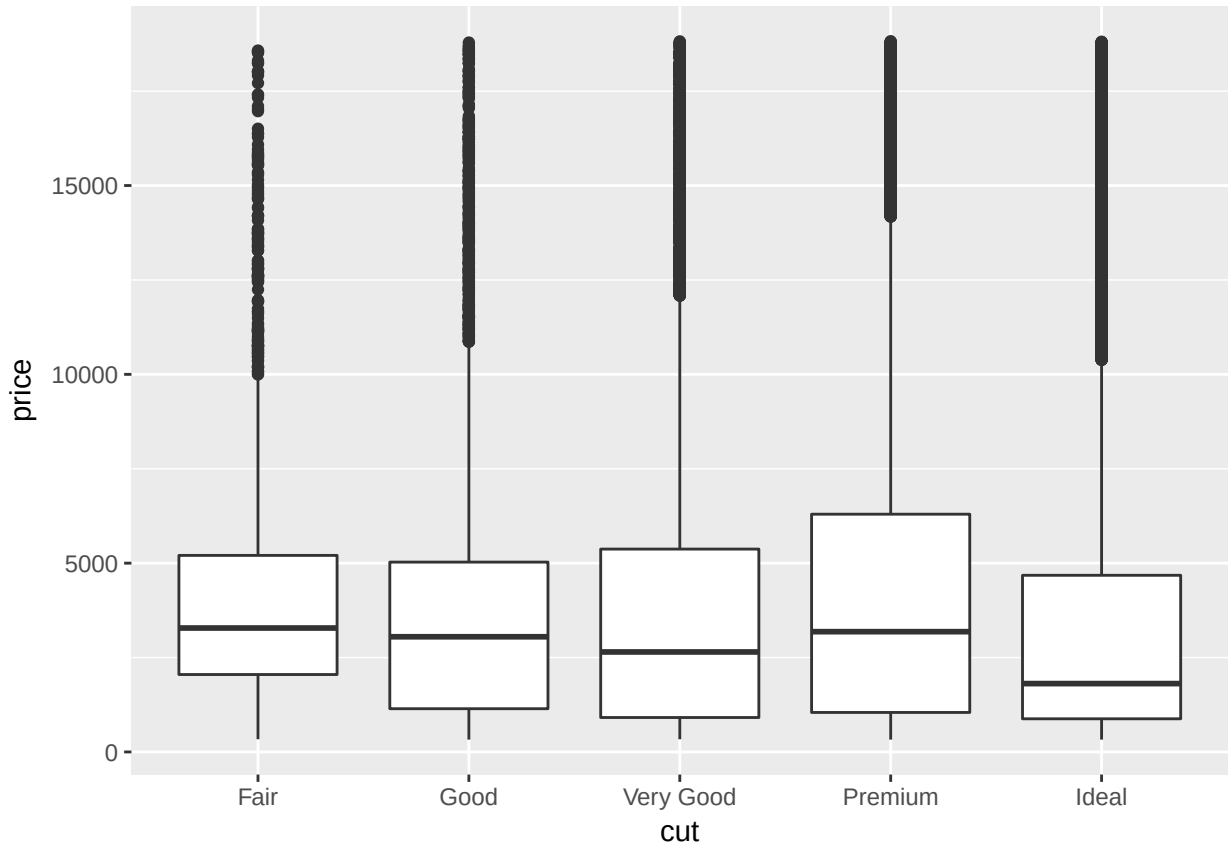

```
# see difference in counts
ggplot(diamonds) + geom_bar(mapping = aes(x = cut))
```



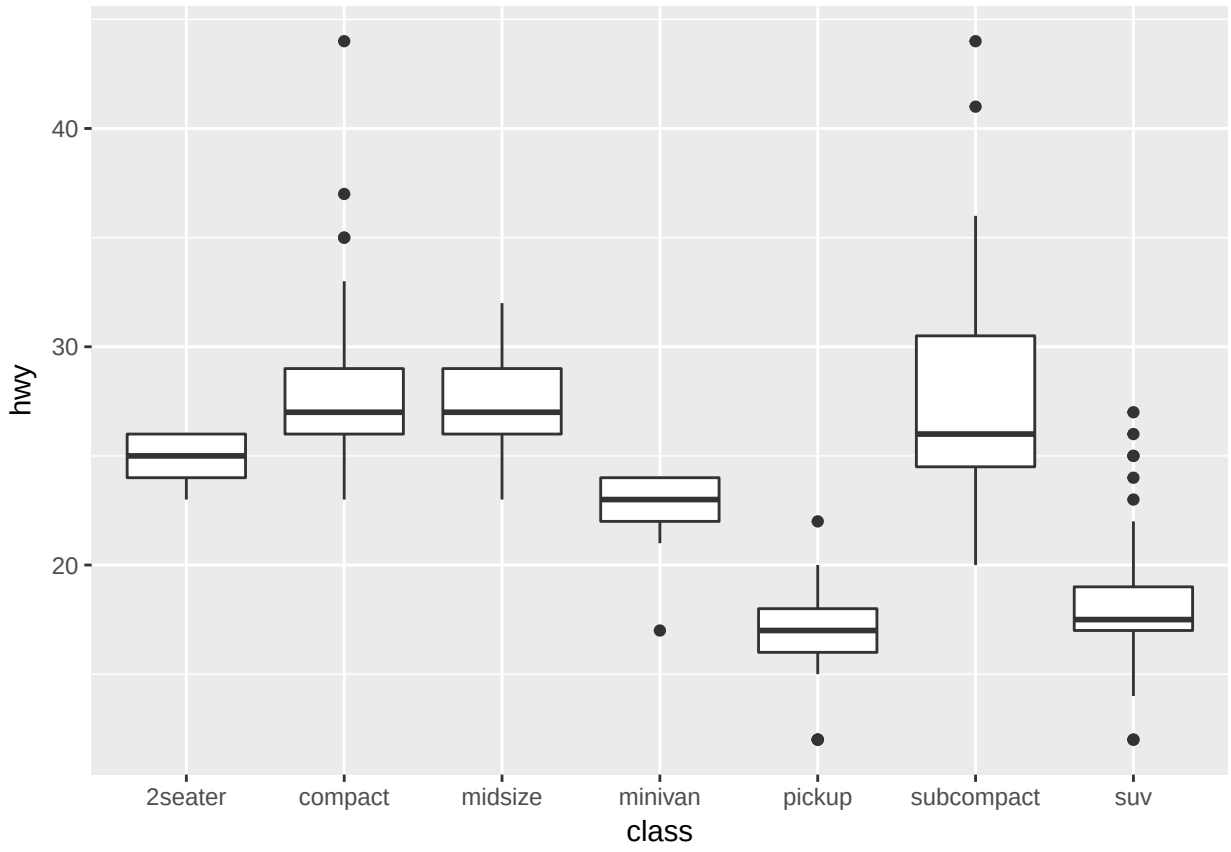

```
# swap axis. density is the count standardized so that the area under each frequency poly is 1.

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



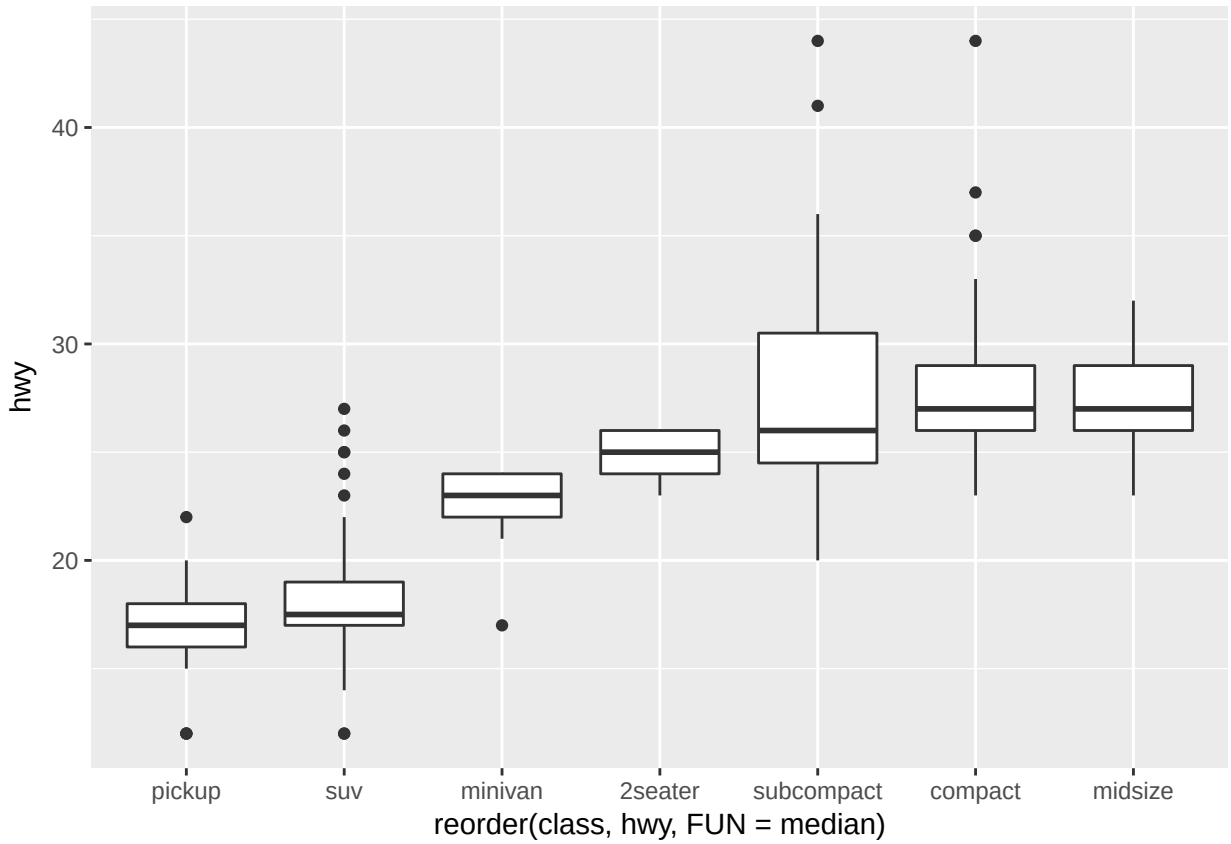

```
# box plot

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



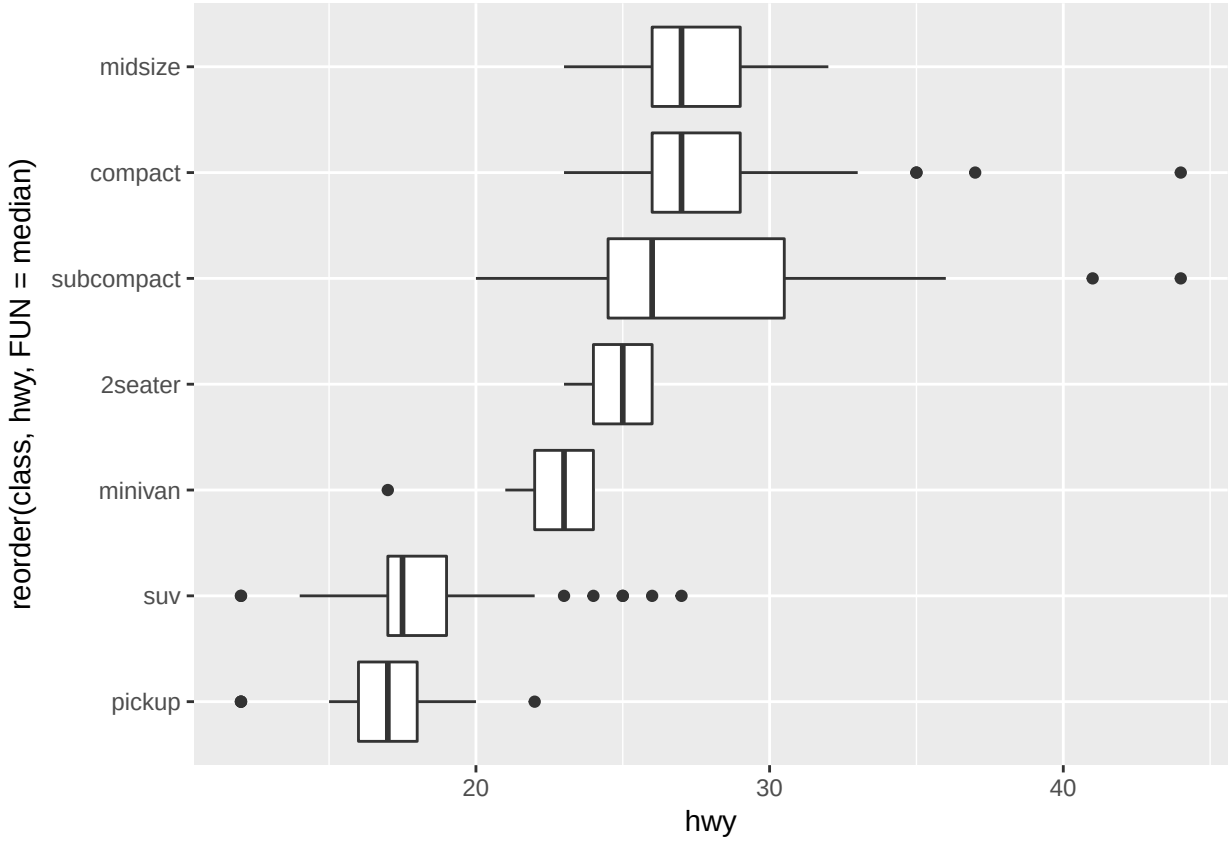

```
# cut is an ordered factor, but when the factors aren't ordered we can use reorder()

# original
ggplot(data = mpg, mapping = aes(x = class, y = hwy)) + geom_boxplot()
```



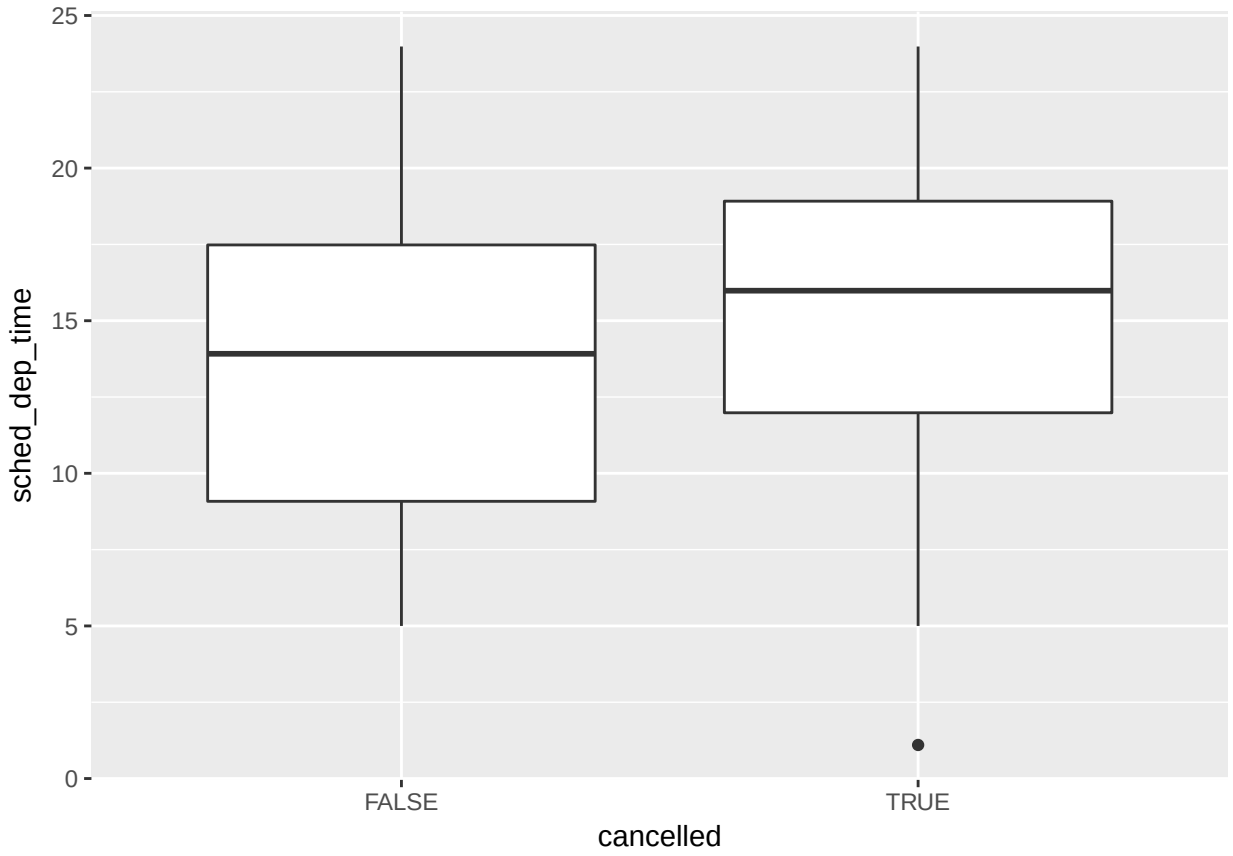

```
# reordered

ggplot(data = mpg) +
  geom_boxplot(mapping = aes(
    x = reorder(class, hwy, FUN = median),
    y = hwy
  ))
```



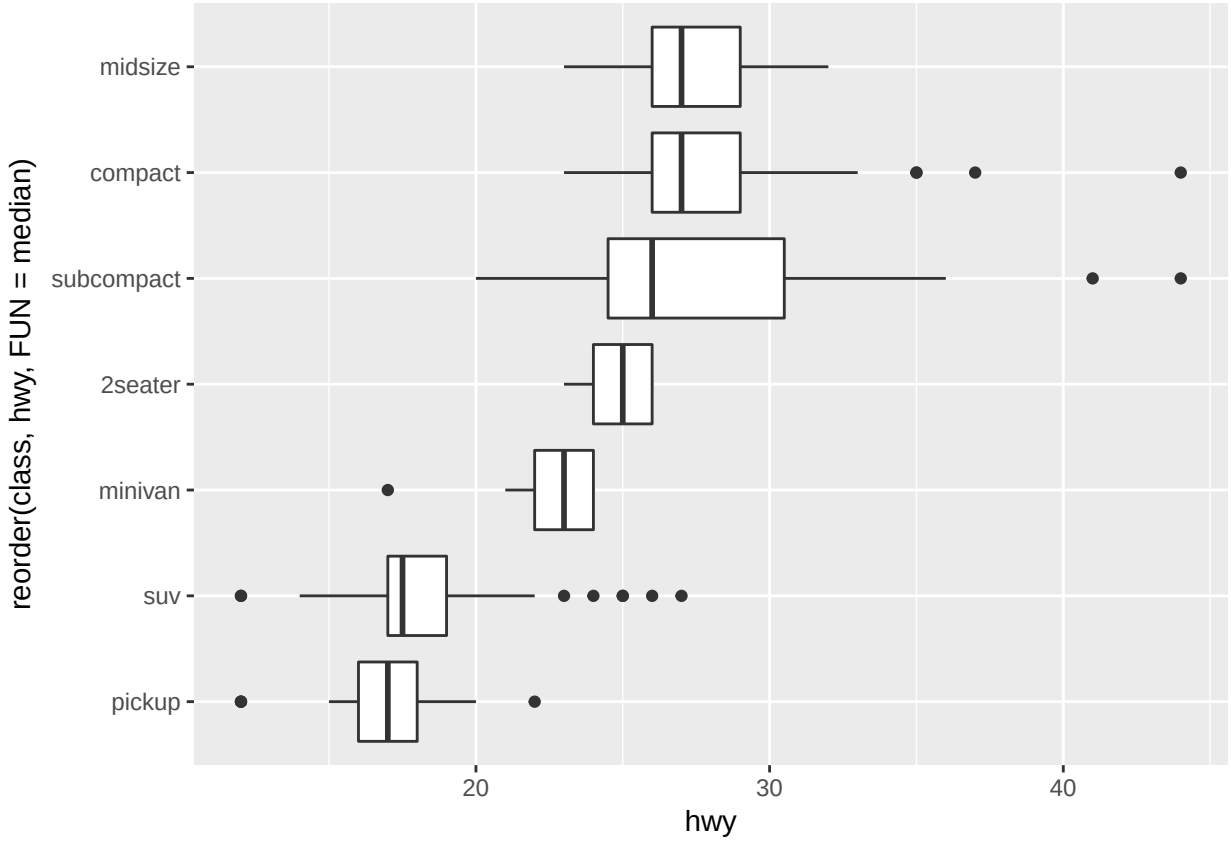

```
# with long var names, flip the boxplot

ggplot(data = mpg) +
  geom_boxplot(
    mapping = aes(
      x = reorder(class, hwy, FUN = median),
      y = hwy
    )
  ) +
  coord_flip()
```



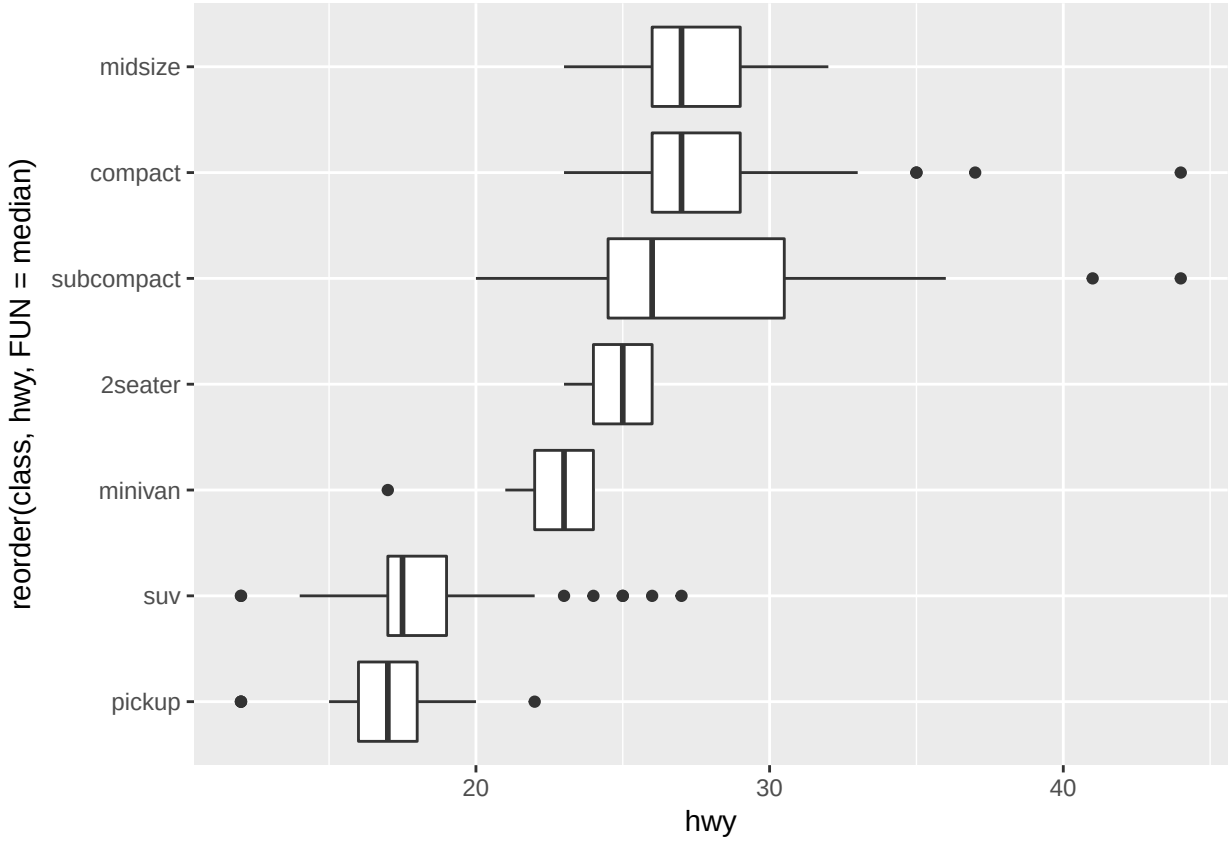

```r
# improving departure times of cancelled and non-cancelled flights

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() + geom_boxplot(mapping = aes(x = cancelled, y = sched_dep_time))
```



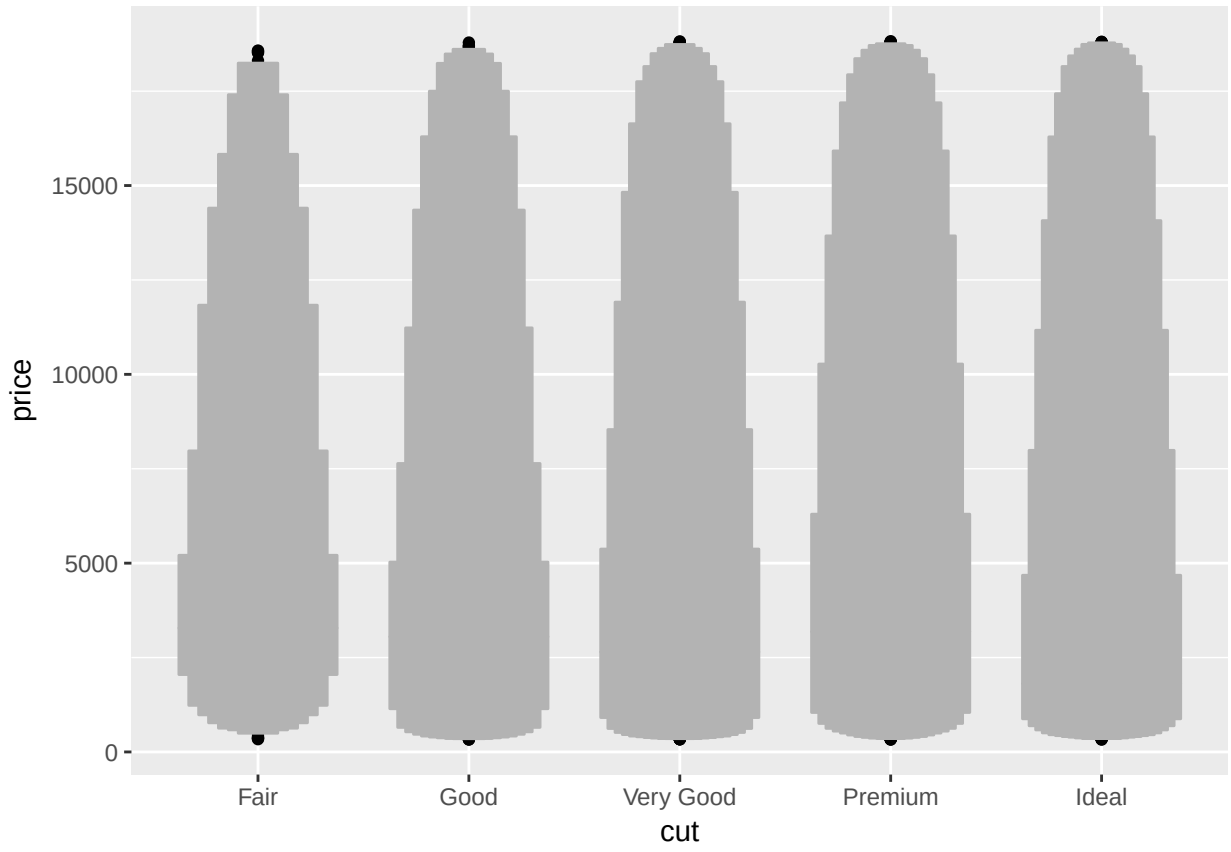

```
# What variables in the diamonds dataset is most important for predicting the price of a diamond? How i

# sounds like some regression is needed

# horizontal boxplot

# regular

ggplot(data = mpg) +
  geom_boxplot(mapping = aes(x = reorder(class, hwy, FUN = median), y = hwy)) + coord_flip()
```




```
# ggstance

ggplot(data = mpg) +
  geom_boxploth(mapping = aes(y = reorder(class, hwy, FUN = median), x = hwy))
```

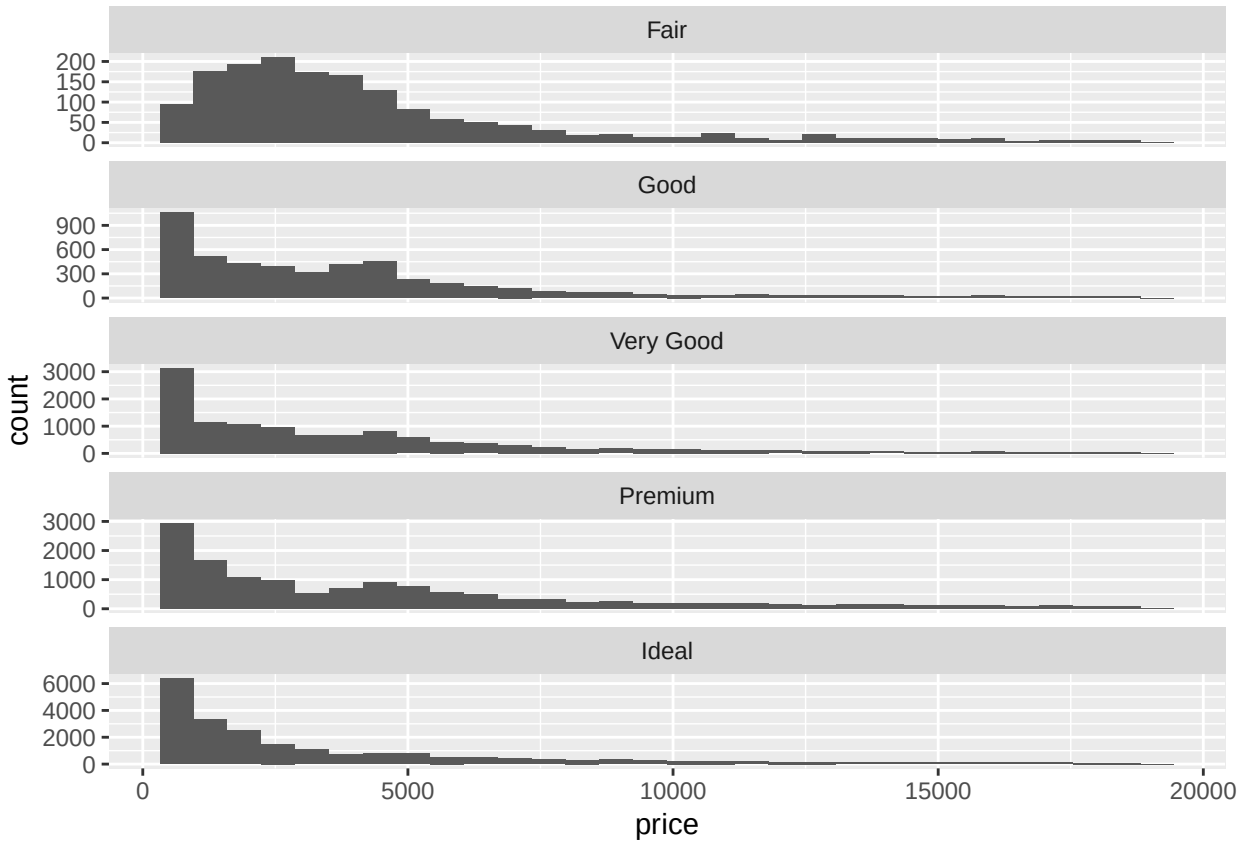



```
# lvplots show more quantiles and look quite nice. The more quantiles part makes them more useful for l

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

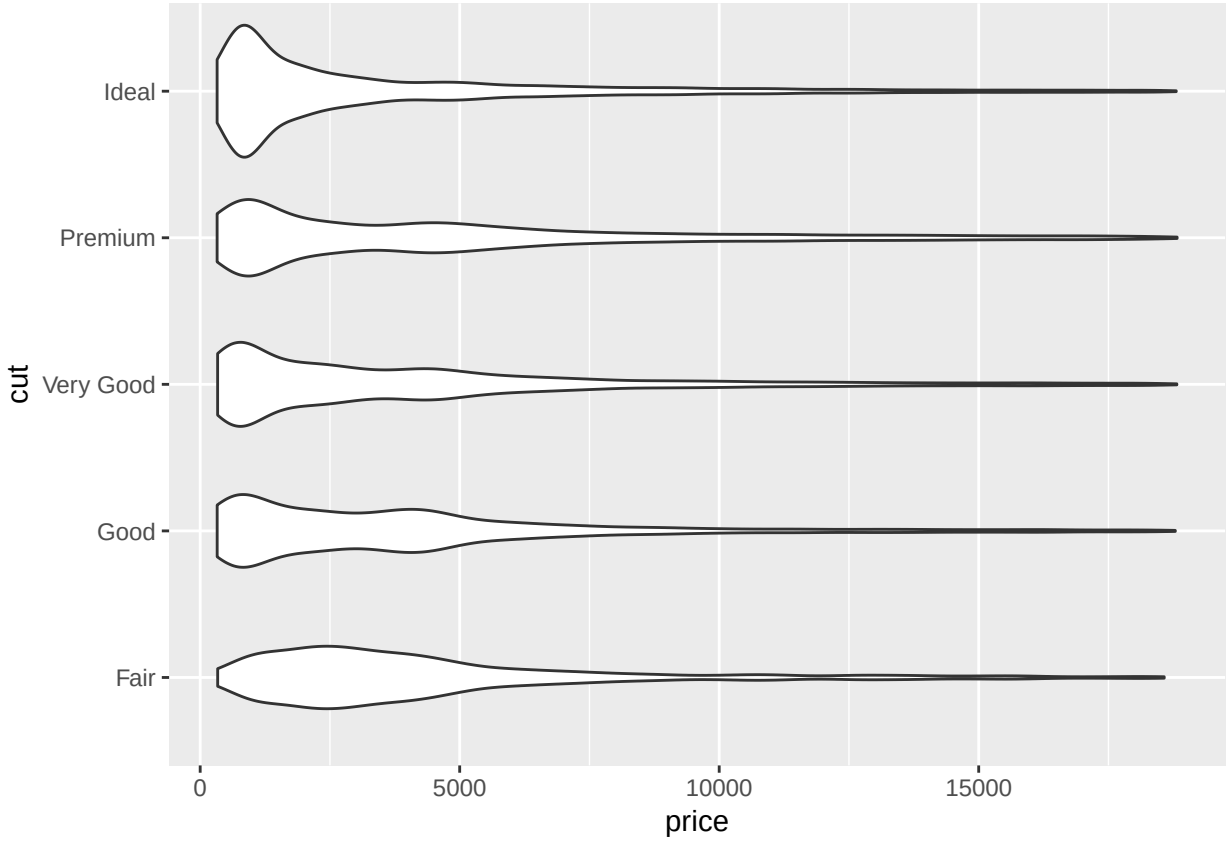



```
# faceted geom_histogram

ggplot(data = diamonds, mapping = aes(x = price)) +
  geom_histogram() + facet_wrap(~ cut, ncol = 1, scales = "free_y")
```

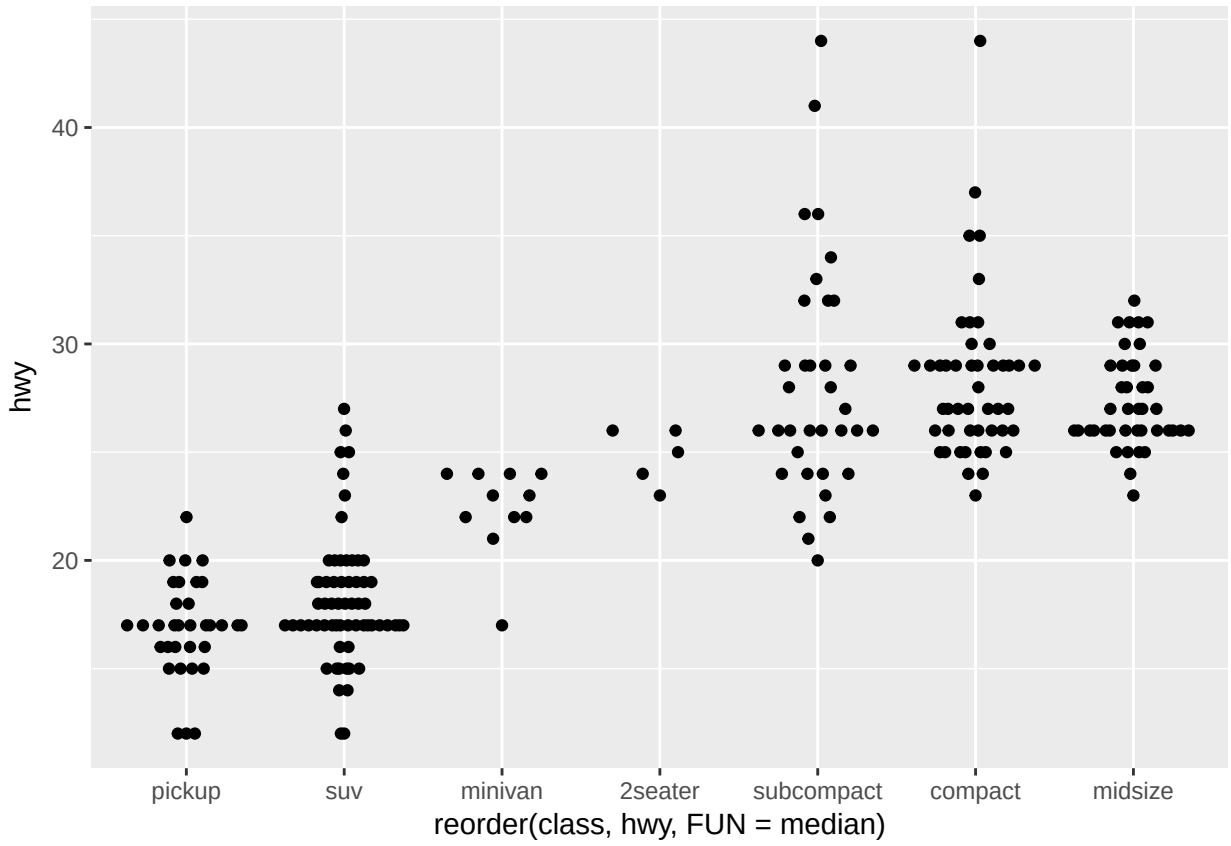
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



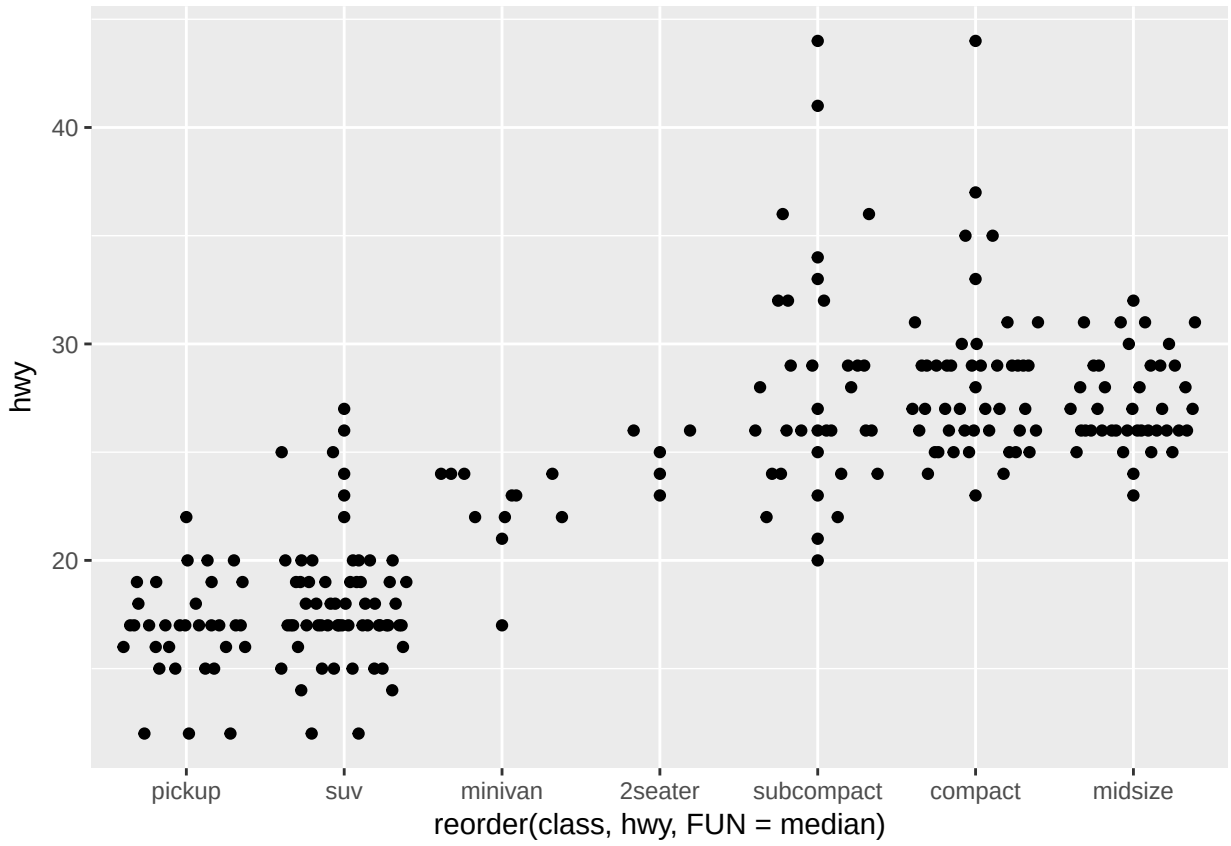

```
# violin

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



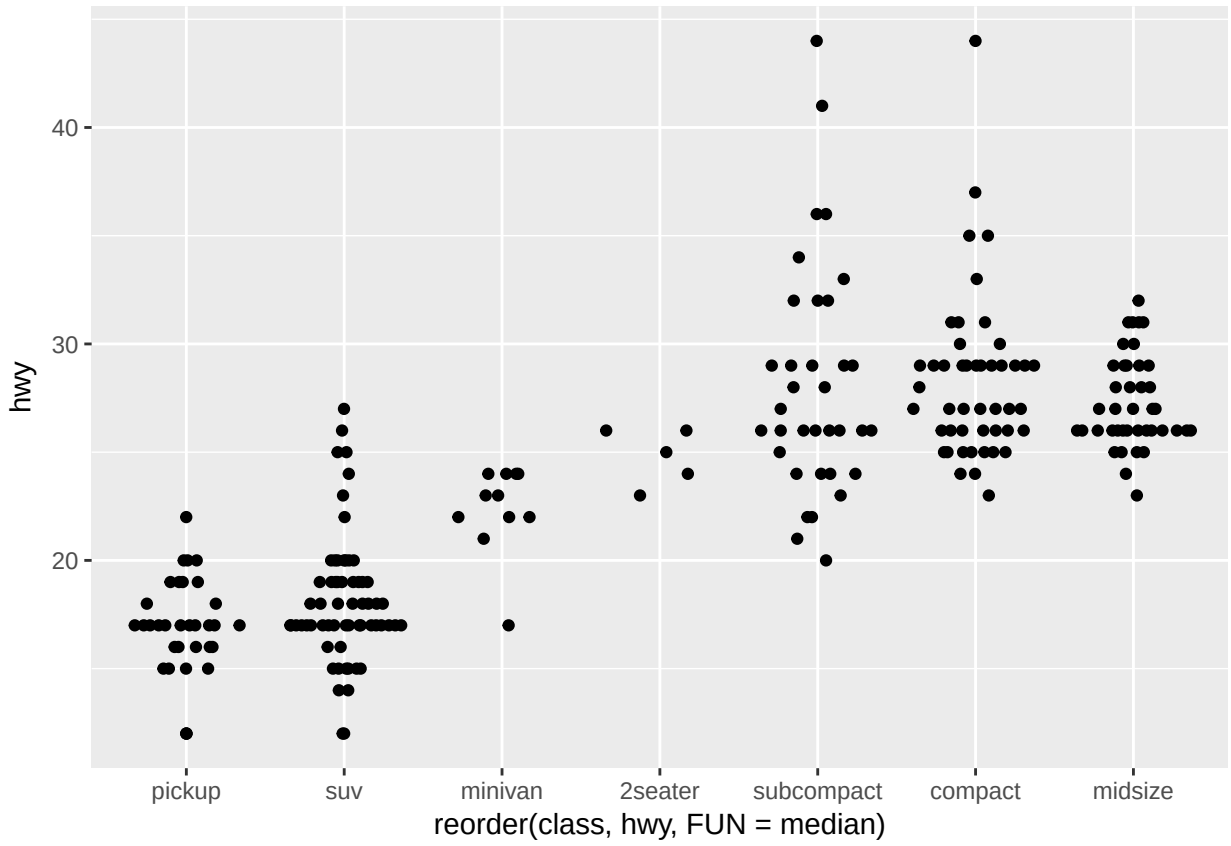

```
# ggbeeswarm

# geom_quasirandom

ggplot(data = mpg) +
  geom_quasirandom(mapping = aes(x = reorder(class, hwy, FUN = median), y = hwy))
```



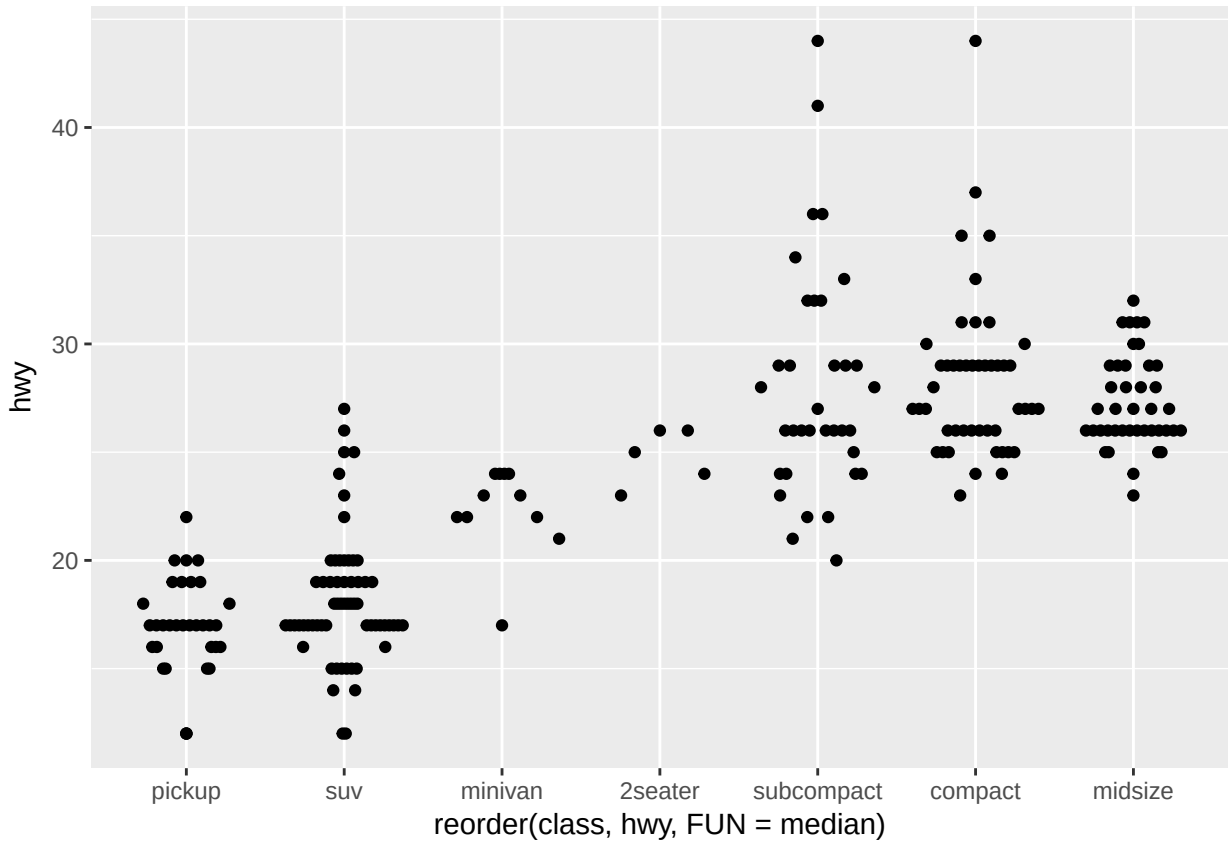

```
ggplot(data = mpg) +
  geom_quasirandom(mapping = aes(x = reorder(class, hwy, FUN = median),
                                 y = hwy),
                   method = "tukey")
```



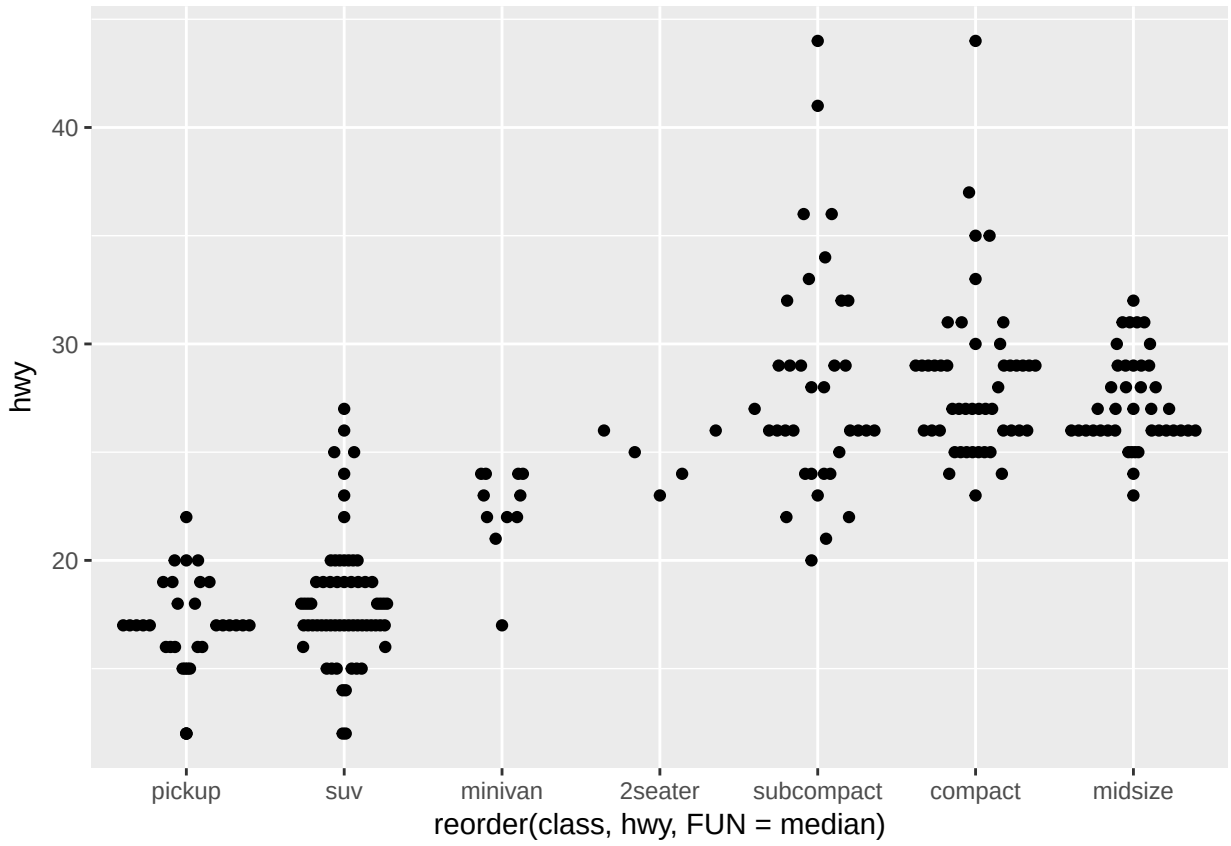

```
ggplot(data = mpg) +
  geom_quasirandom(mapping = aes(x = reorder(class, hwy, FUN = median), y = hwy),
                   method = "tukeyDense")
```



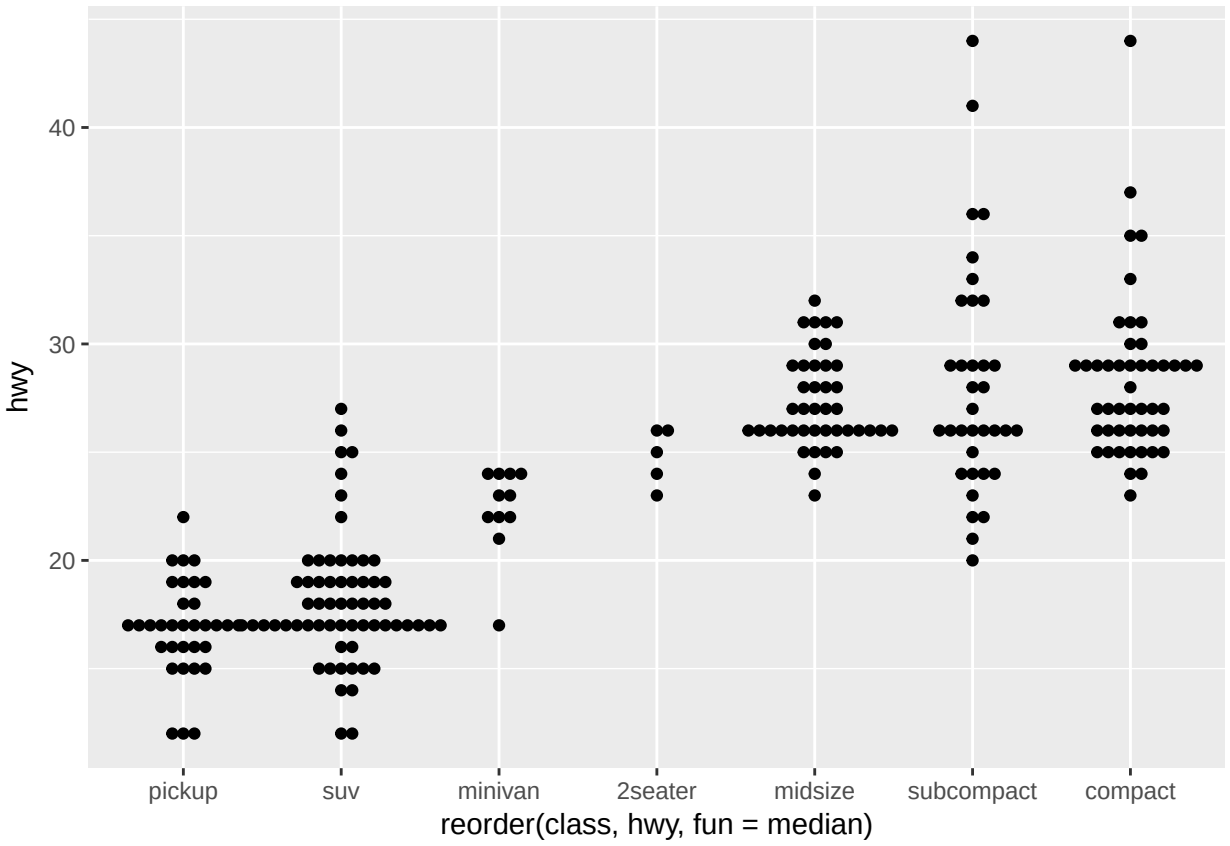

```
ggplot(data = mpg) +
  geom_quasirandom(mapping = aes(x = reorder(class, hwy, FUN = median), y = hwy), method = "frowney")
```




```
ggplot(data = mpg) +
  geom_quasirandom(mapping = aes(x = reorder(class, hwy, FUN = median), y = hwy), method = "smiley")
```

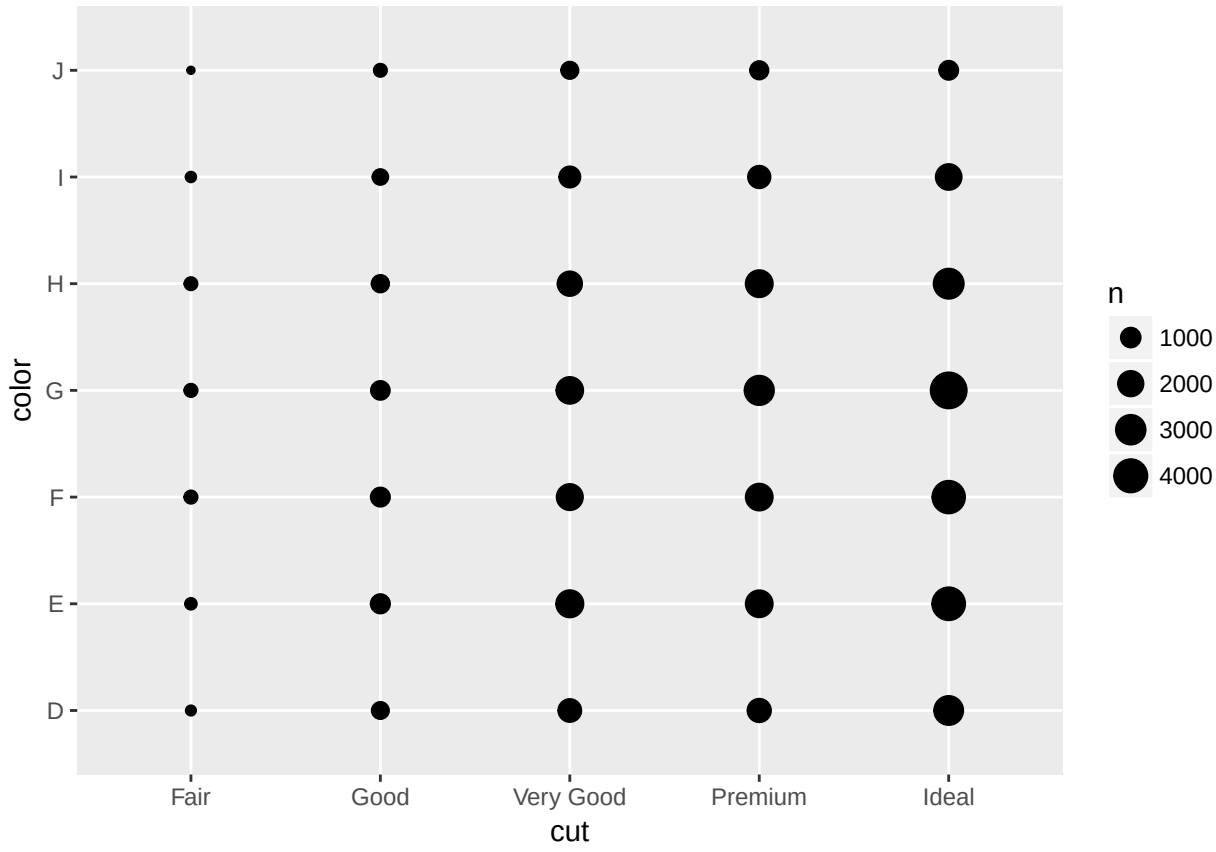



```
# geom_beeswarm

ggplot(data = mpg) +
  geom_beeswarm(mapping = aes(x = reorder(class, hwy, fun = median), y = hwy))
```

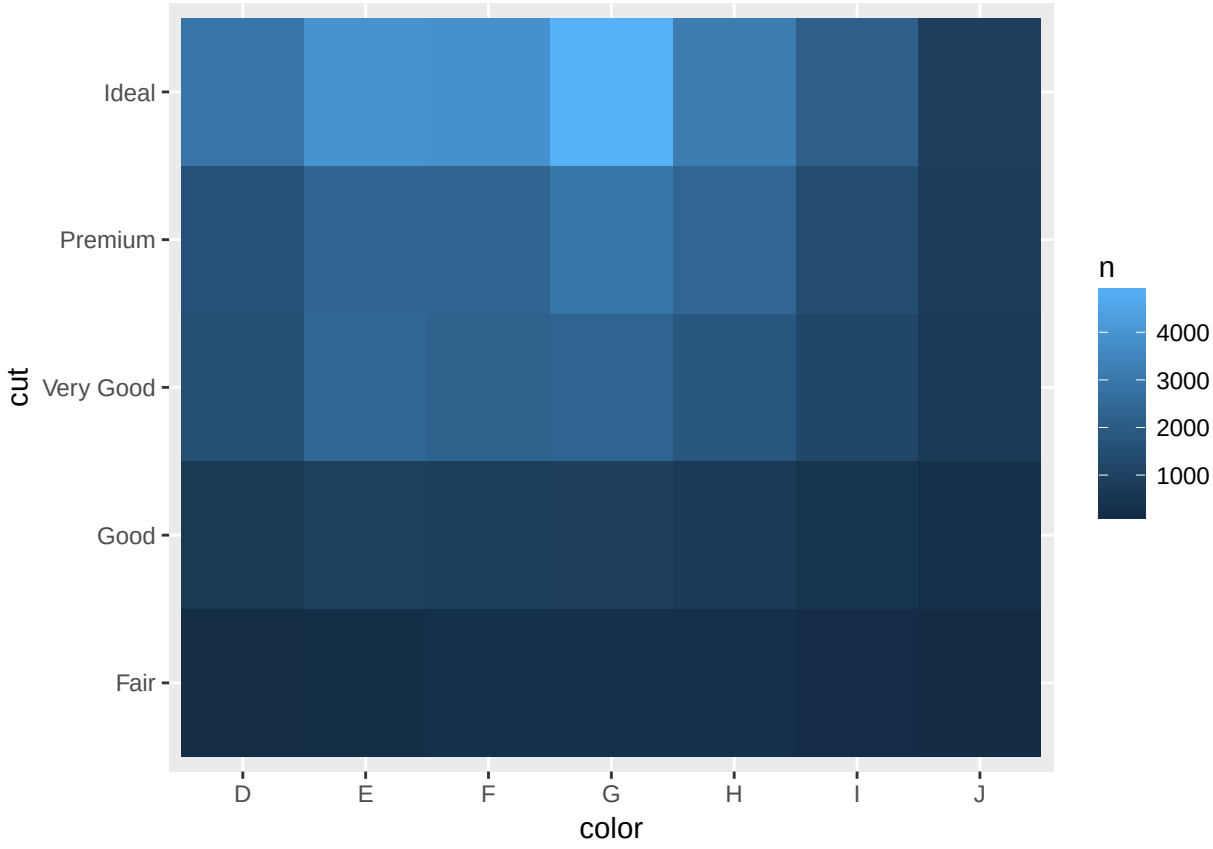


Two Categorical Variables

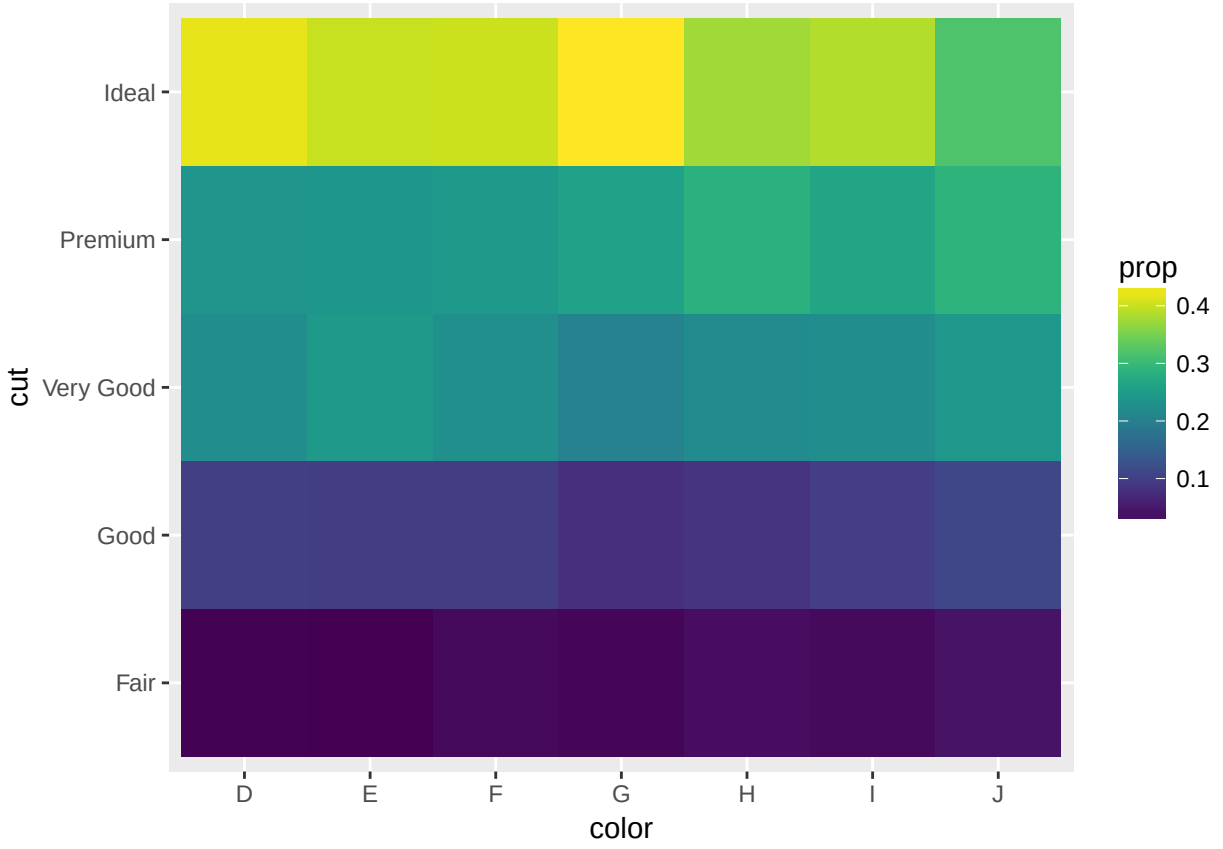
```
# count the number of observations for each combination of cut and color

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



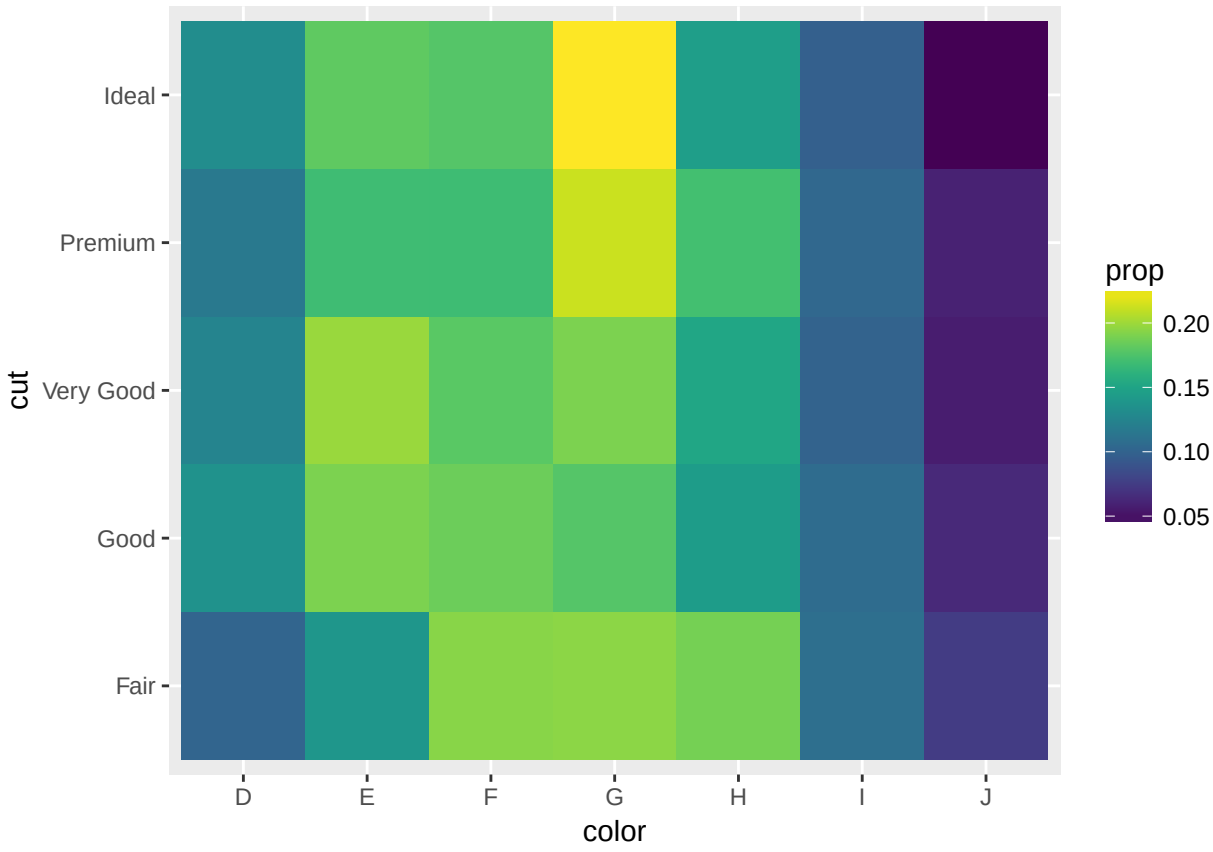

```
# count with dplyr

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



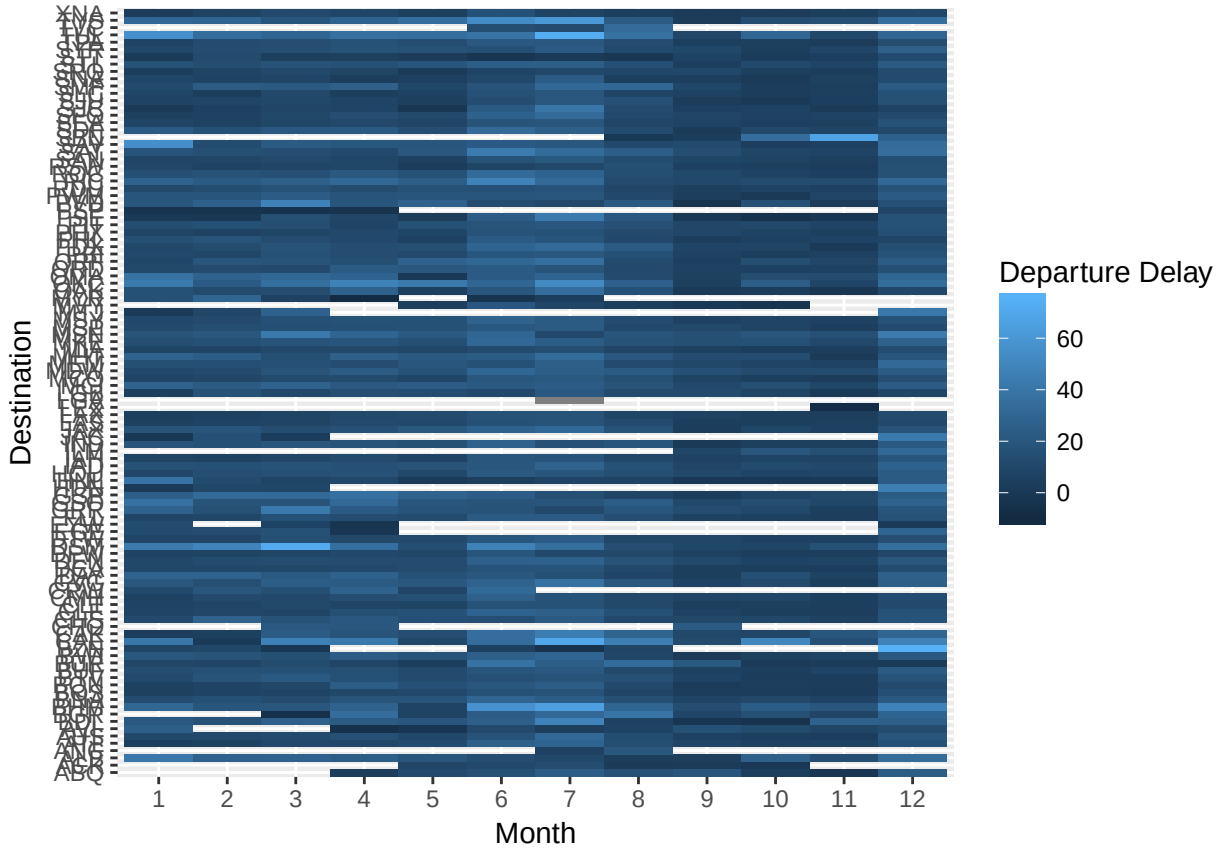

```
# How could you rescale the count dataset to more clearly show the distribution of cut within color, or

# calc a new var, prop which shows proportion of each cut within a color

diamonds %>%
  count(color, cut) %>% # make counts of colors
  group_by(color) %>% # create color groups
  mutate(prop = n / sum(n)) %>%  # create prop var
  ggplot(mapping = aes(x = color, y = cut)) + # ggplot
  geom_tile(mapping = aes(fill = prop)) +
  scale_fill_viridis() # color
```



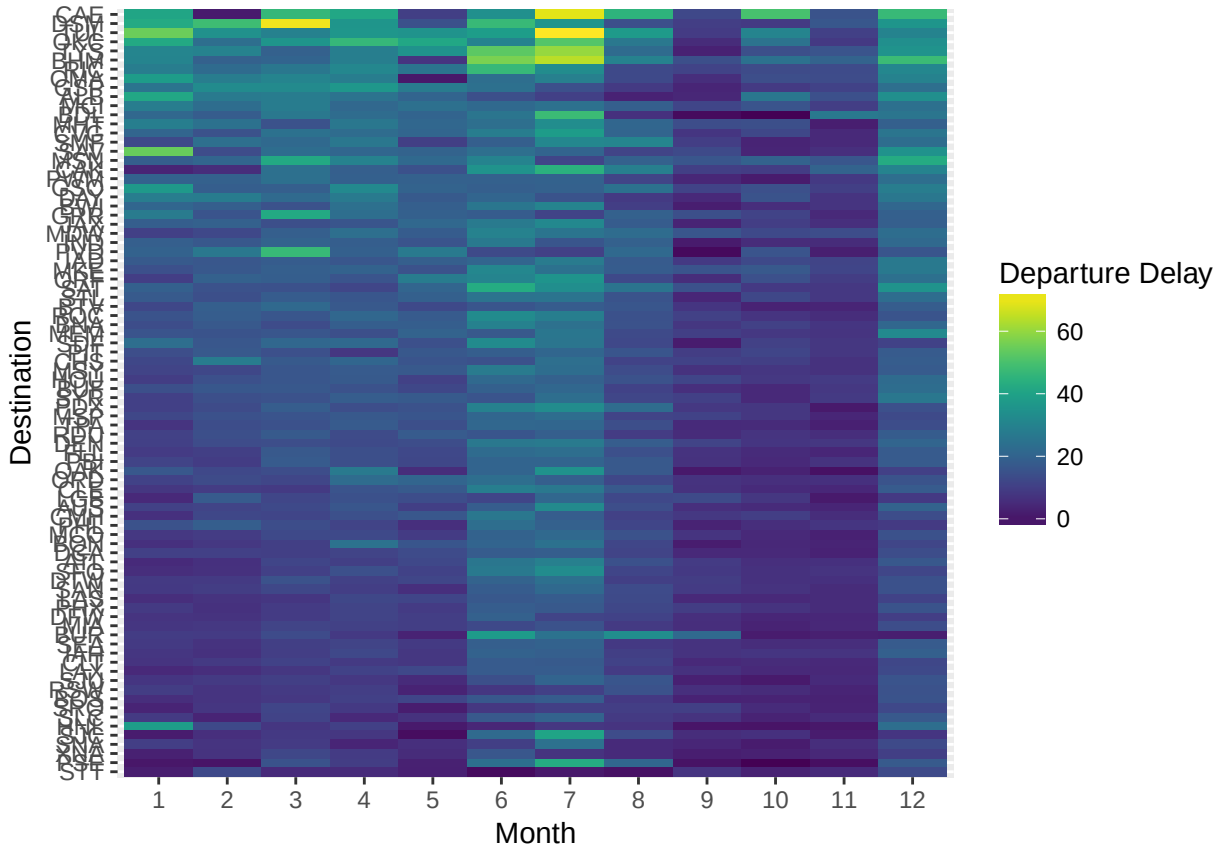

```
# scale by color within a cut

diamonds %>%
  count(color, cut) %>% # make counts of colors
  group_by(cut) %>% # create color groups
  mutate(prop = n / sum(n)) %>%  # create prop var
  ggplot(mapping = aes(x = color, y = cut)) + # ggplot
  geom_tile(mapping = aes(fill = prop)) +
  scale_fill_viridis() # color
```



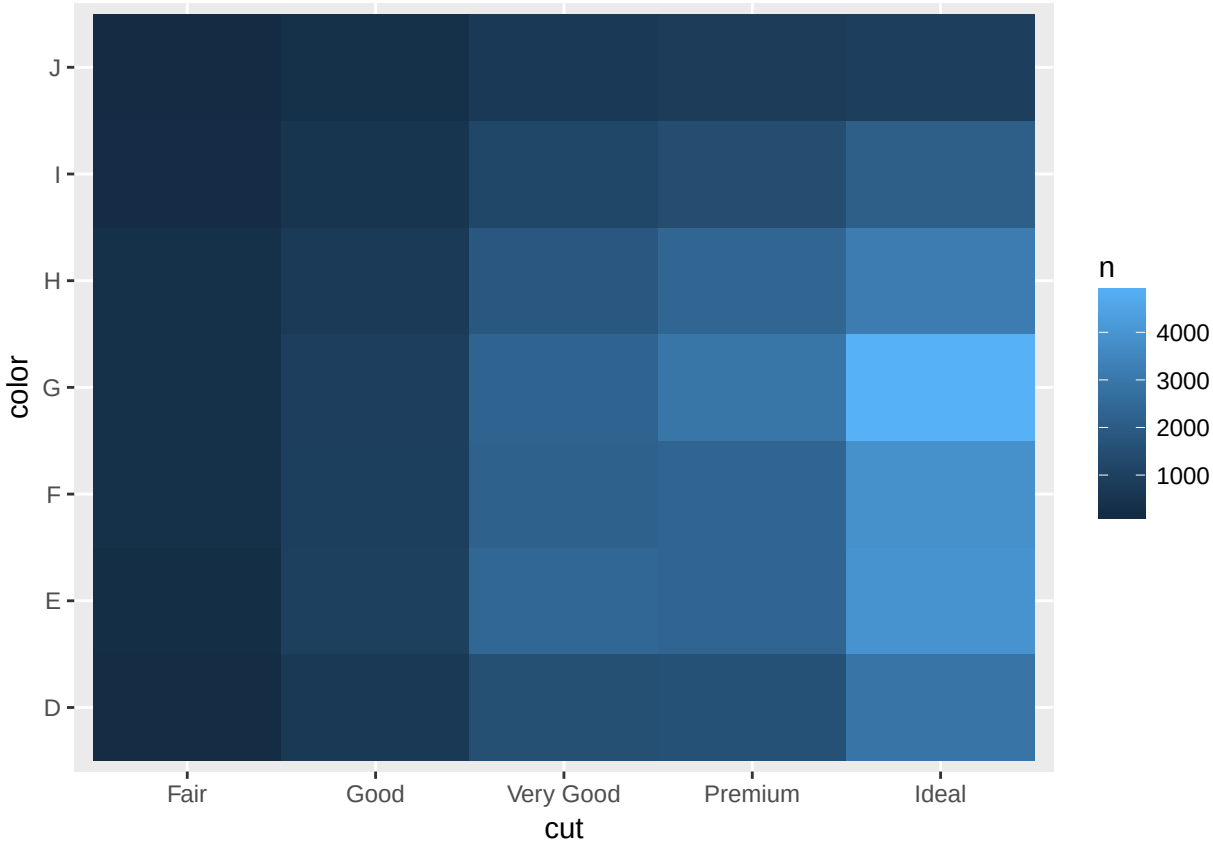

```
# How do average flight delays vary by destination and month of year

flights <- nycflights13::flights

# looks like shit

flights %>%
  group_by(month, dest) %>%
  summarize(dep_delay = mean(dep_delay, na.rm = TRUE)) %>%
  ggplot(aes(x = factor(month), y = dest, fill = dep_delay)) +
  geom_tile() +
  labs(x = "Month", y = "Destination", fill = "Departure Delay")
```




```
# less so

flights %>%
  group_by(month, dest) %>%
  summarize(dep_delay = mean(dep_delay, na.rm = TRUE)) %>%
  group_by(dest) %>%
  filter(n() == 12) %>%
  ungroup() %>%
  mutate(dest = fct_reorder(dest, dep_delay)) %>%
  ggplot(aes(x = factor(month), y = dest, fill = dep_delay)) +
  geom_tile() +
  scale_fill_viridis() +
  labs(x = "Month", y = "Destination", fill = "Departure Delay")
```

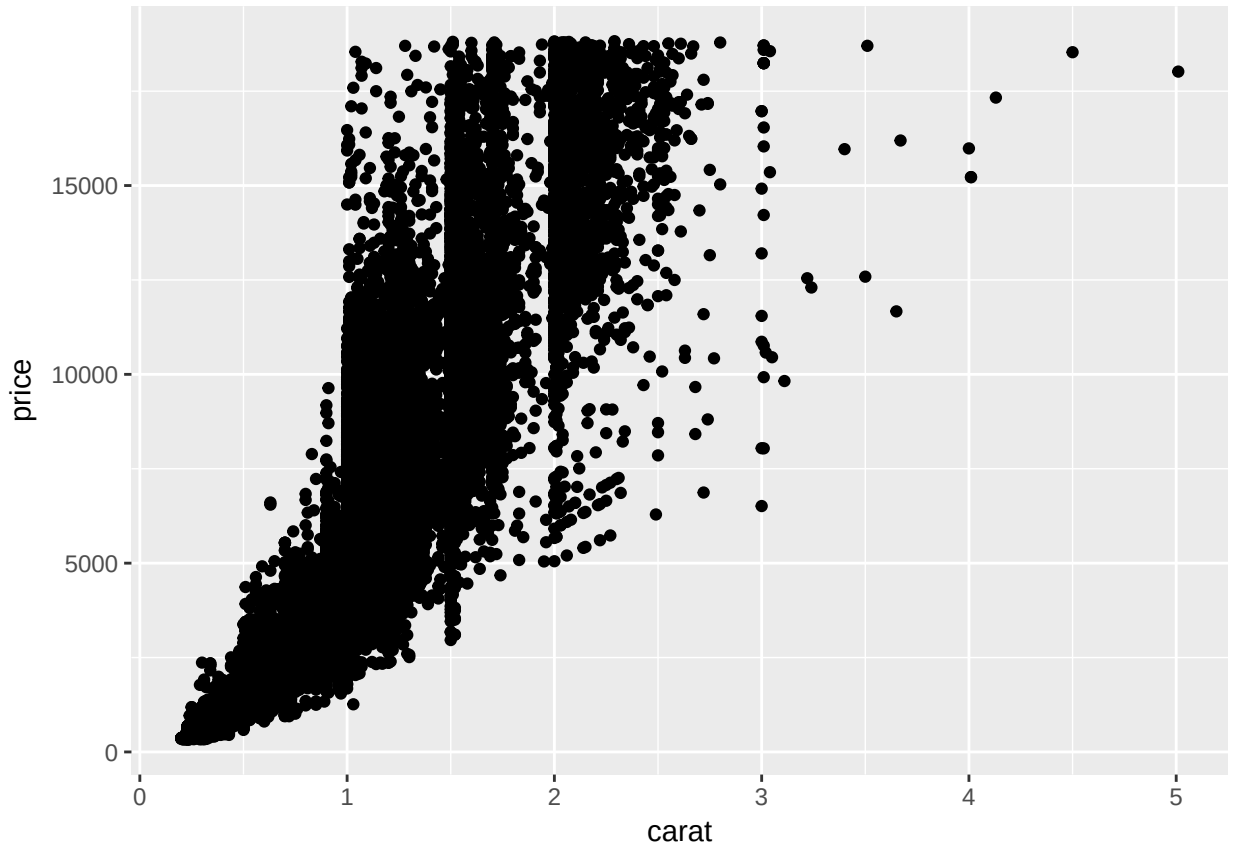



```
# Why is it better to use aes(x = color, y = cut) rather than aes(x = cut, y = color)

# Interpretability

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

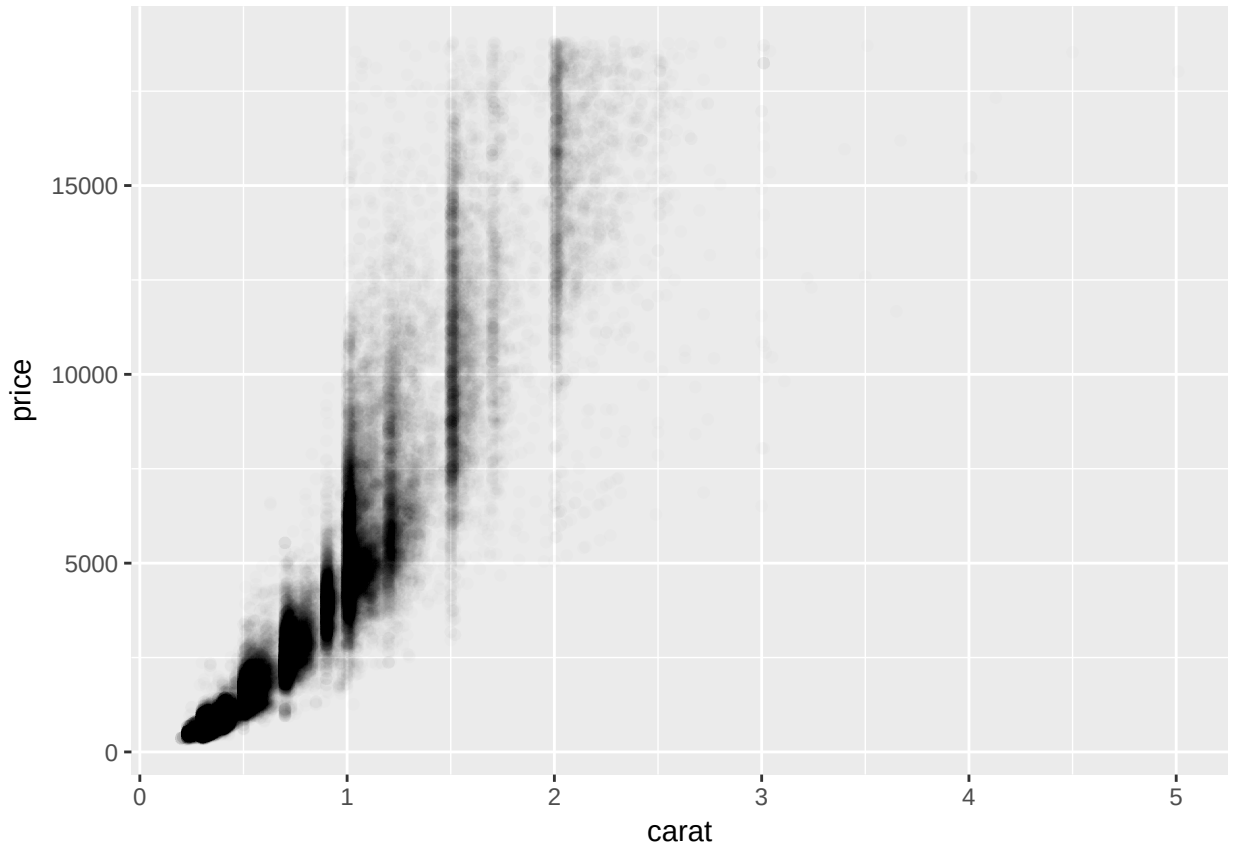


Two Continuous Variables

```
# relationship between carat size and price

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

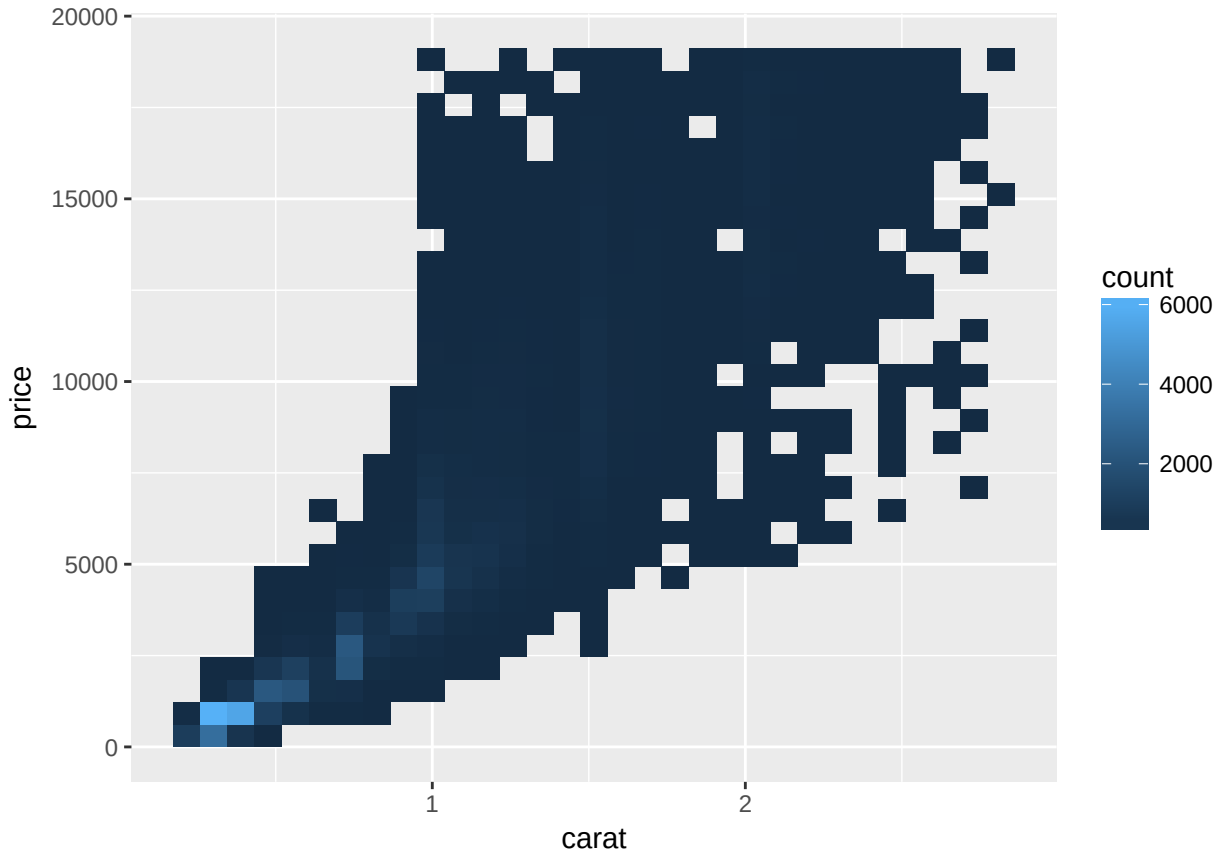



```
# to deal with the blackout effect of large datasets, we can use alpha

ggplot(data = diamonds) +
  geom_point(mapping = aes(x = carat, y = price),
             alpha = 1/100)
```

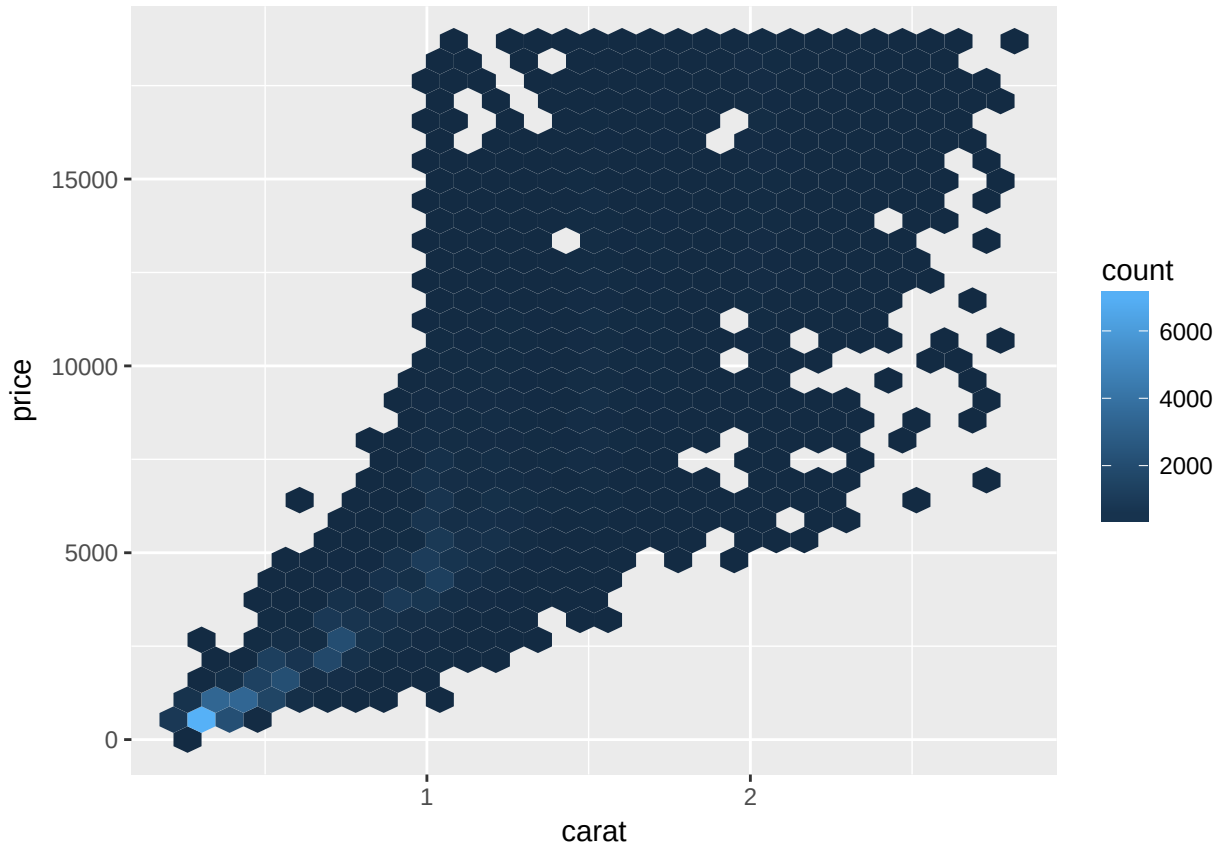


bins

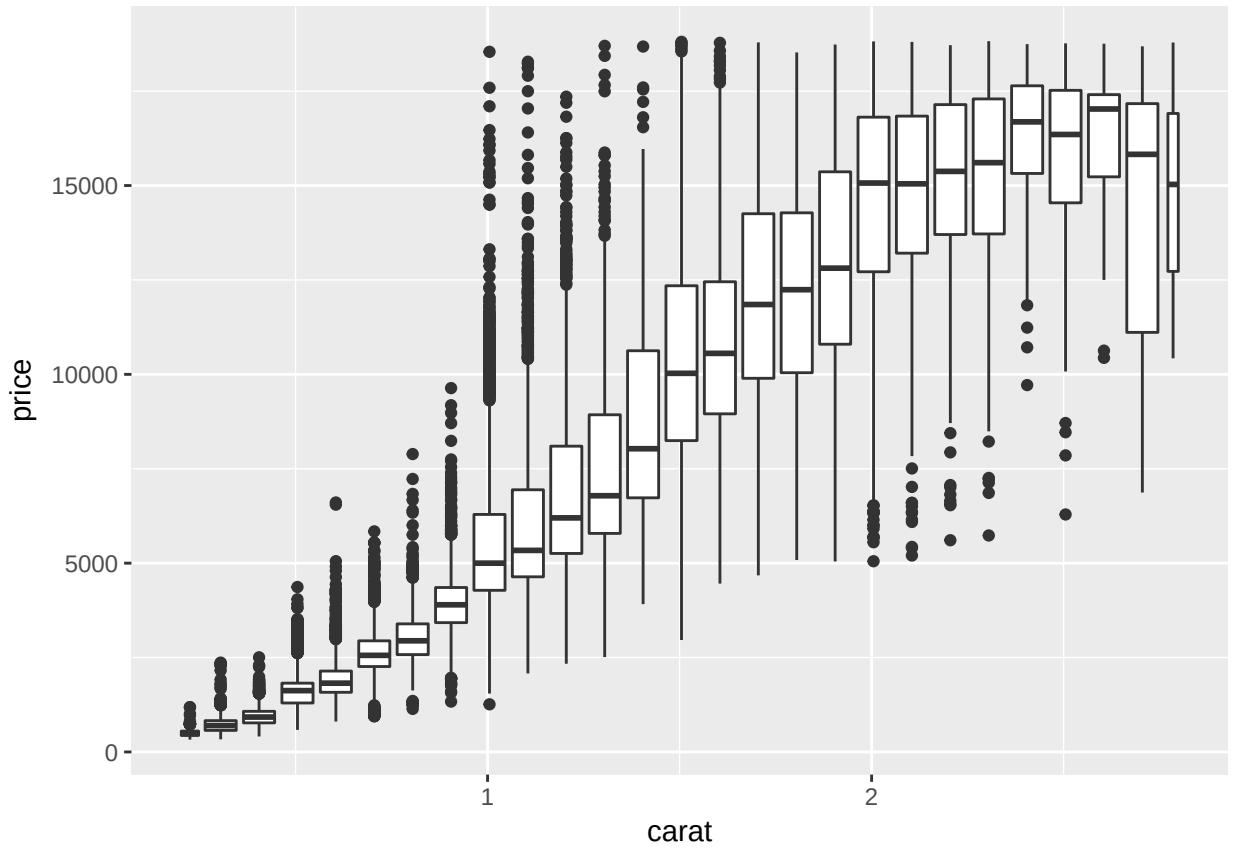
```
#geom_bin2d

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



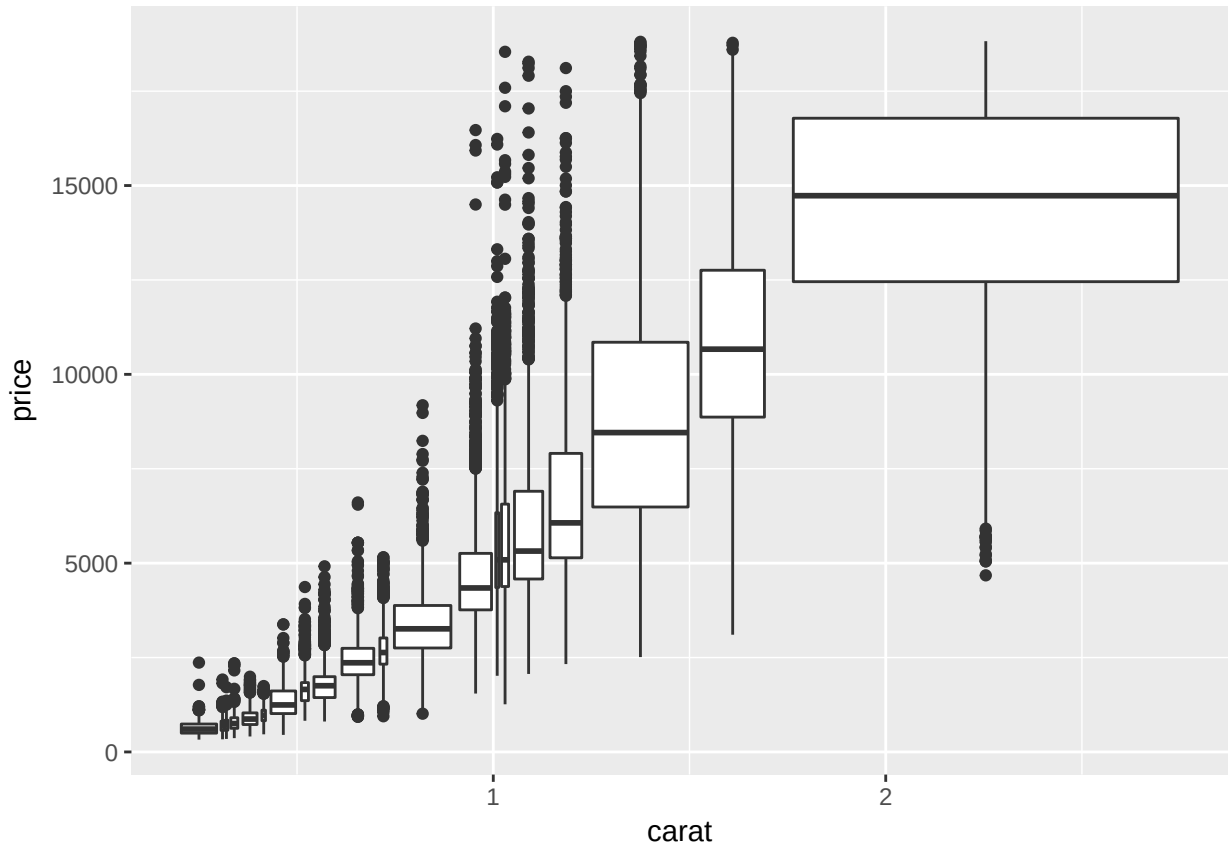

```
# geom_hex

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




```
# bin one continuous variable to make it act like a categorical variable

ggplot(data = smaller, mapping = aes(x = carat, y = price)) +
  geom_boxplot(mapping = aes(group = cut_width(carat, 0.1)))
```

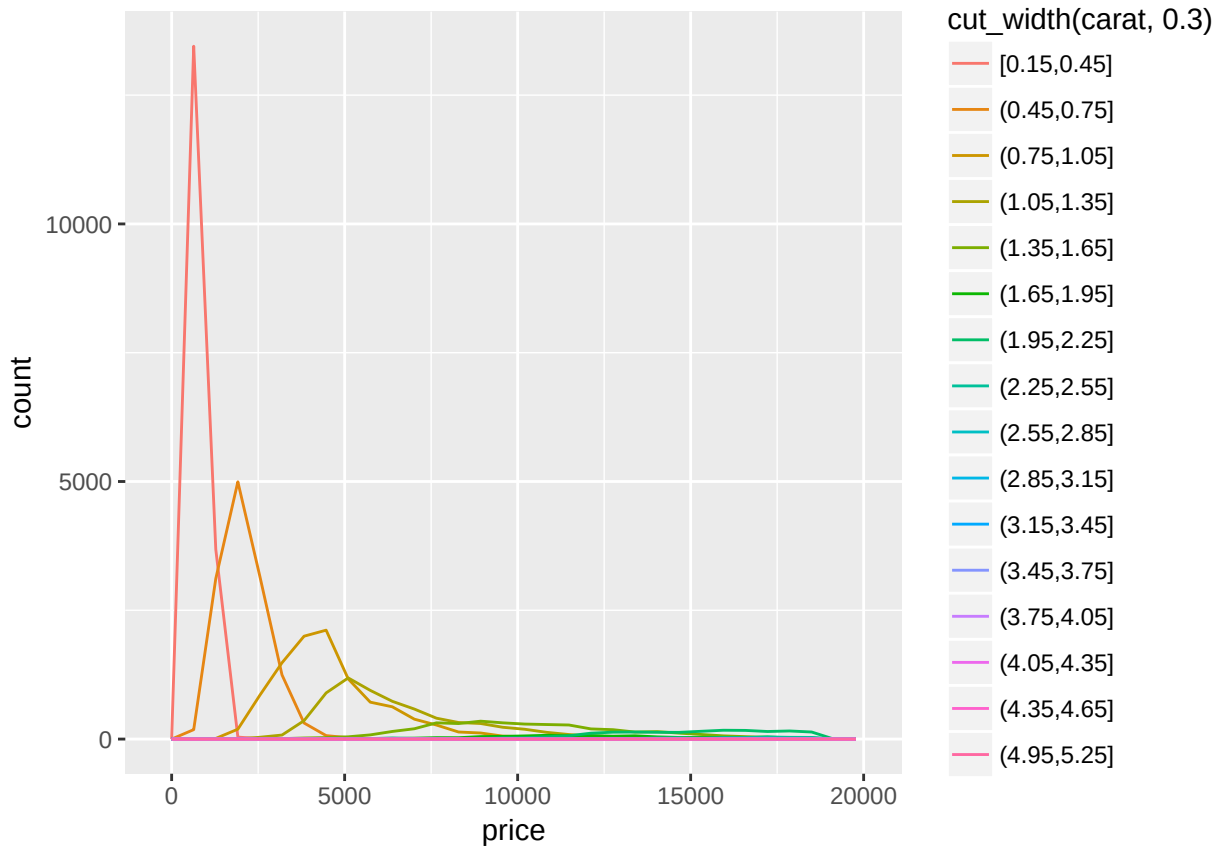



```
# boxplots don't show the proportion of points within well. We can display approximately the same numbe

ggplot(data = smaller, mapping = aes(x = carat, y = price)) +
  geom_boxplot(mapping = aes(group = cut_number(carat, 20)))
```



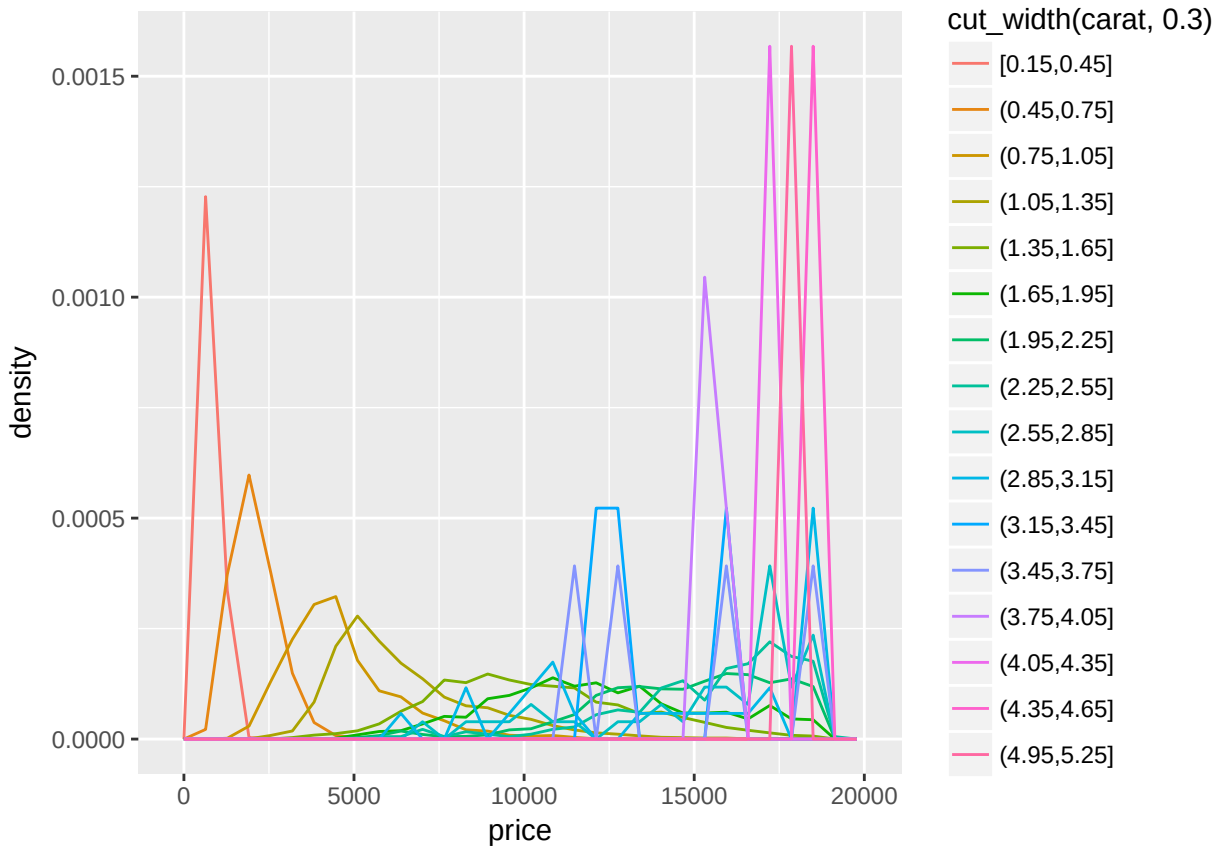

```
# Instead of summarizing the conditional distribution with a boxplot, you could use a frequency polygon

# with cut_width, the number of points in the bin may not be equal

ggplot(data = diamonds, mapping = aes(x = price, color = cut_width(carat, 0.3))) + geom_freqpoly()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



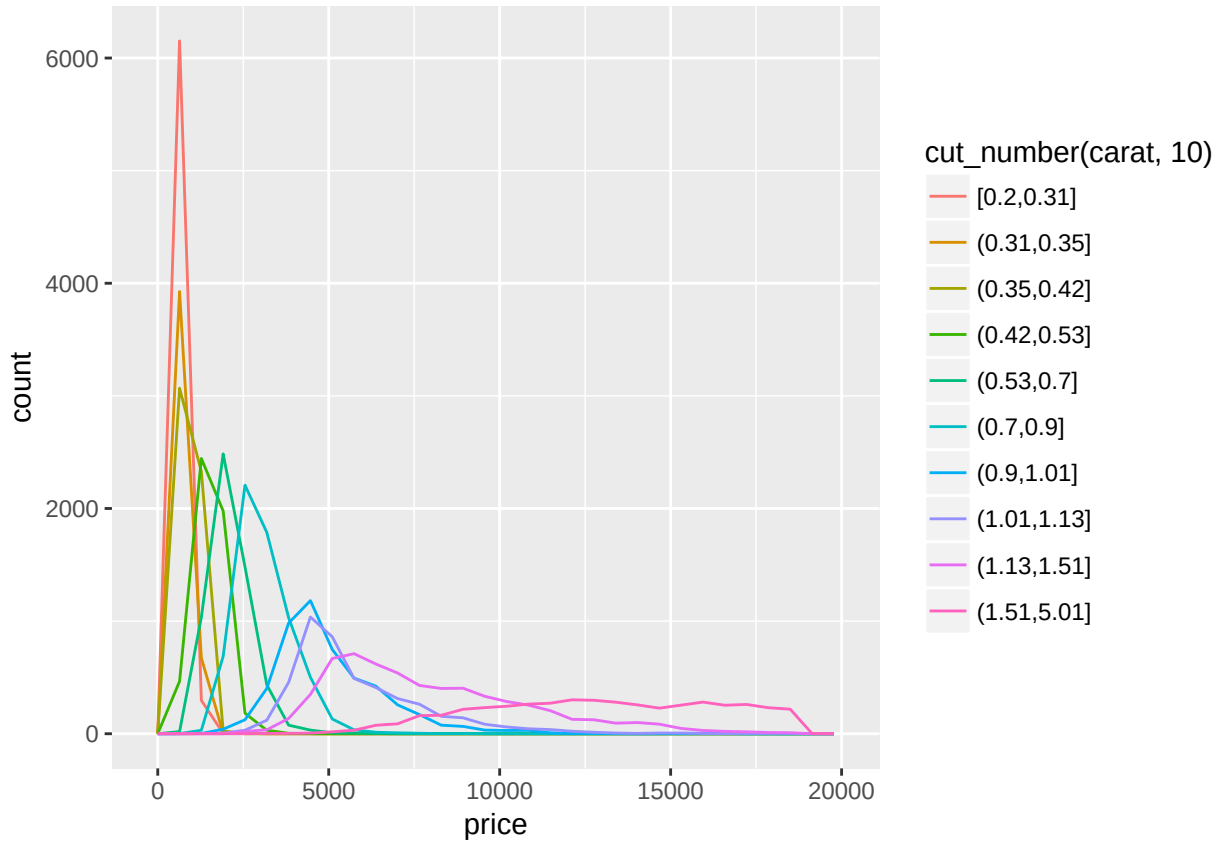

```
# plotting the density instead of counts makes distributions comparable, but bins with few observations

ggplot(data = diamonds, mapping = aes(x = price, y = ..density..,
                                      color = cut_width(carat, 0.3))) +
  geom_freqpoly()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



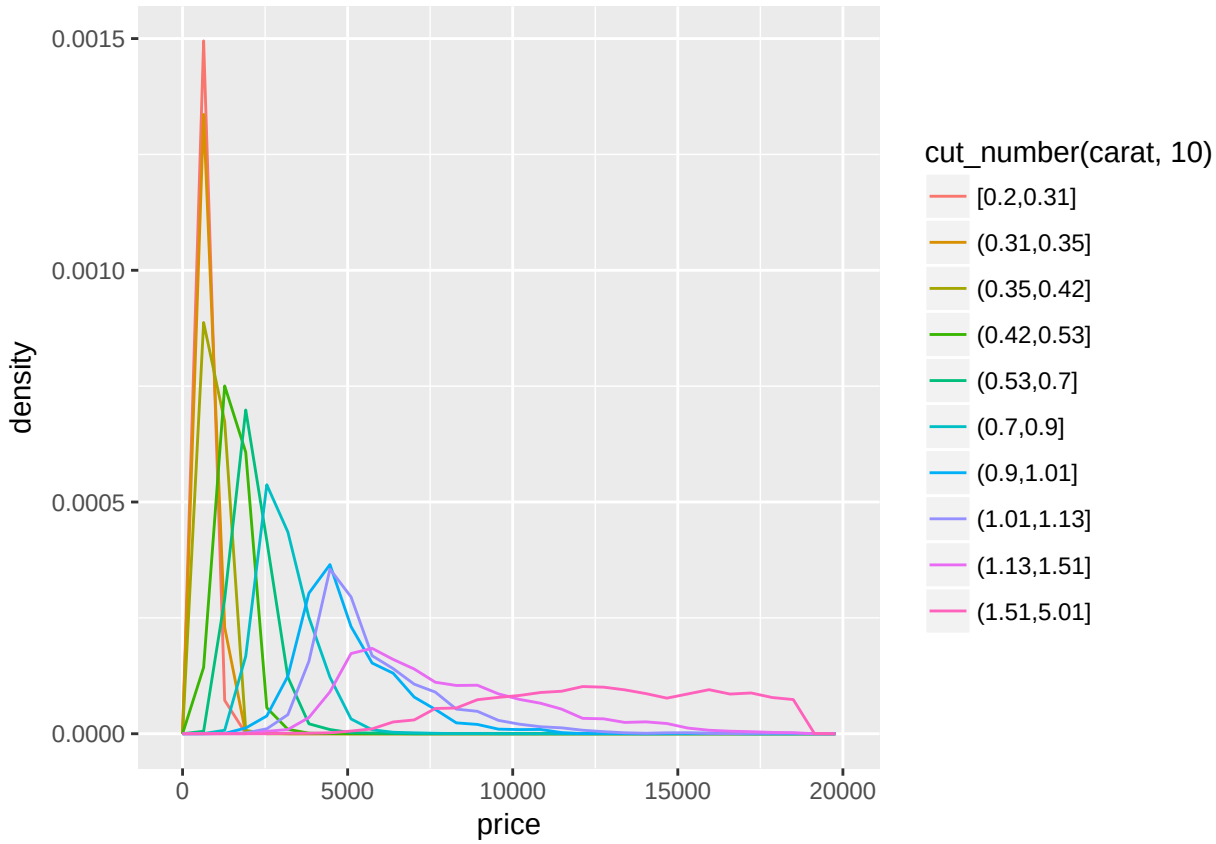

```
# adjusting the bin size

ggplot(data = diamonds,
       mapping = aes(x = price,
                     colour = cut_number(carat, 10))) +
  geom_freqpoly()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



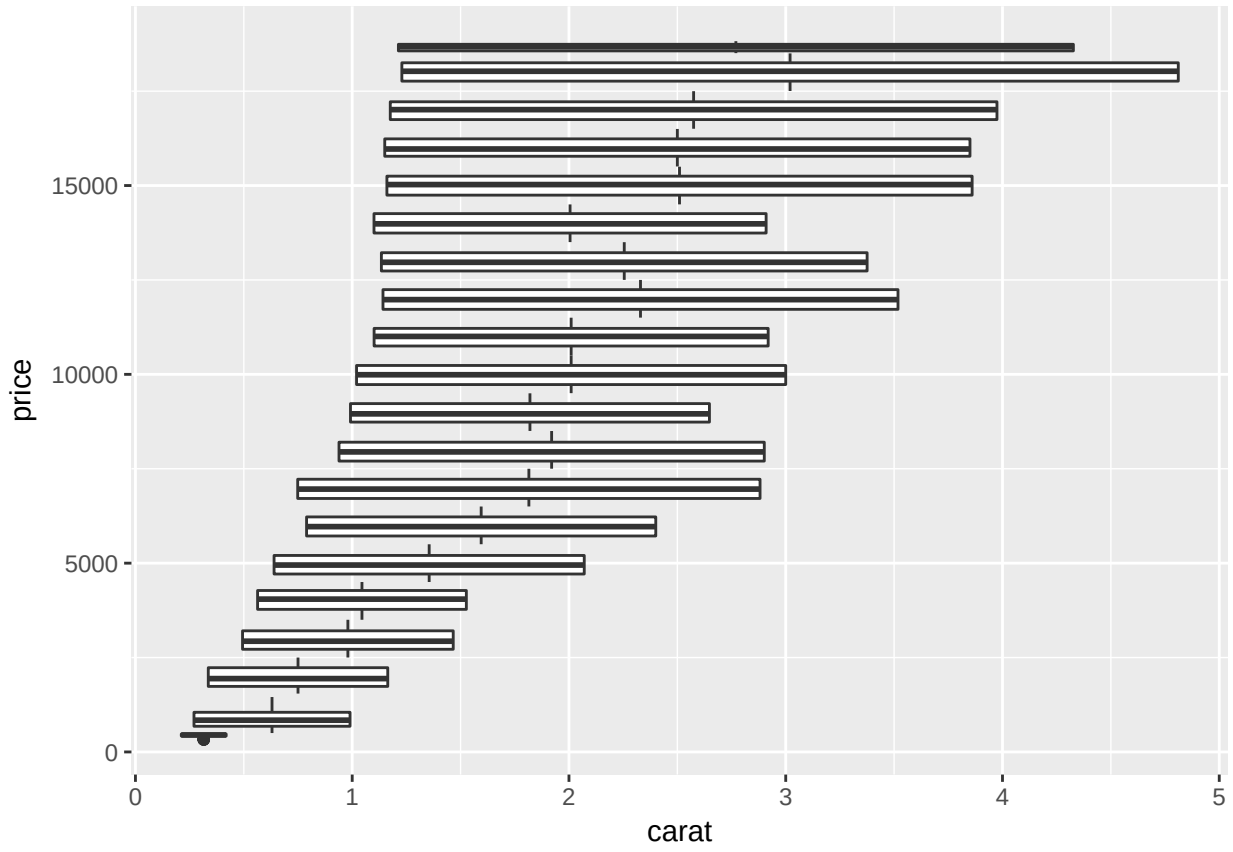

```
# with density

ggplot(data = diamonds,
       mapping = aes(x = price, y = ..density..,
                     color = cut_number(carat, 10))) +
  geom_freqpoly()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

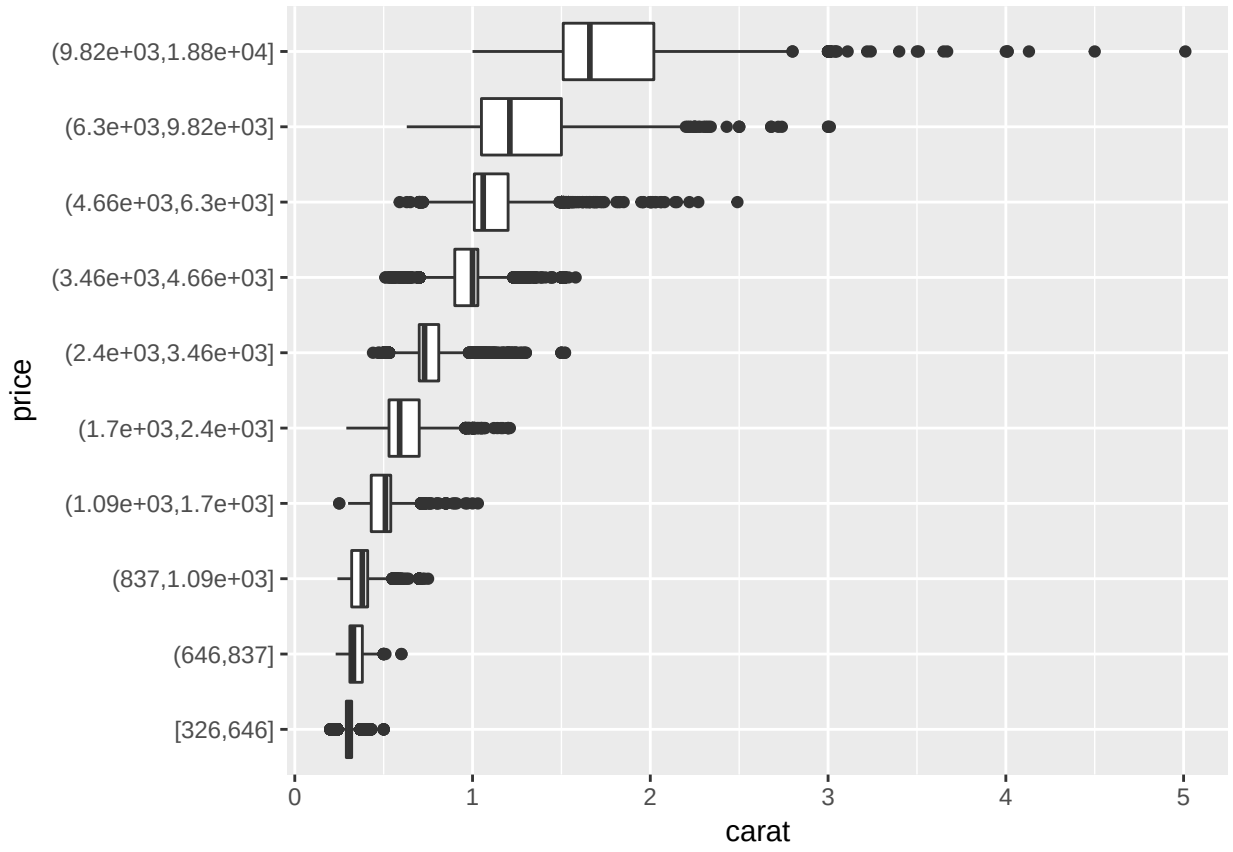



```
# visualize the distribution of carat, partitioned by price

ggplot(data = diamonds,
       mapping = aes(x = carat, y = price)) +
  geom_boxplot(mapping = aes(group = cut_width(price, 1000)))
```

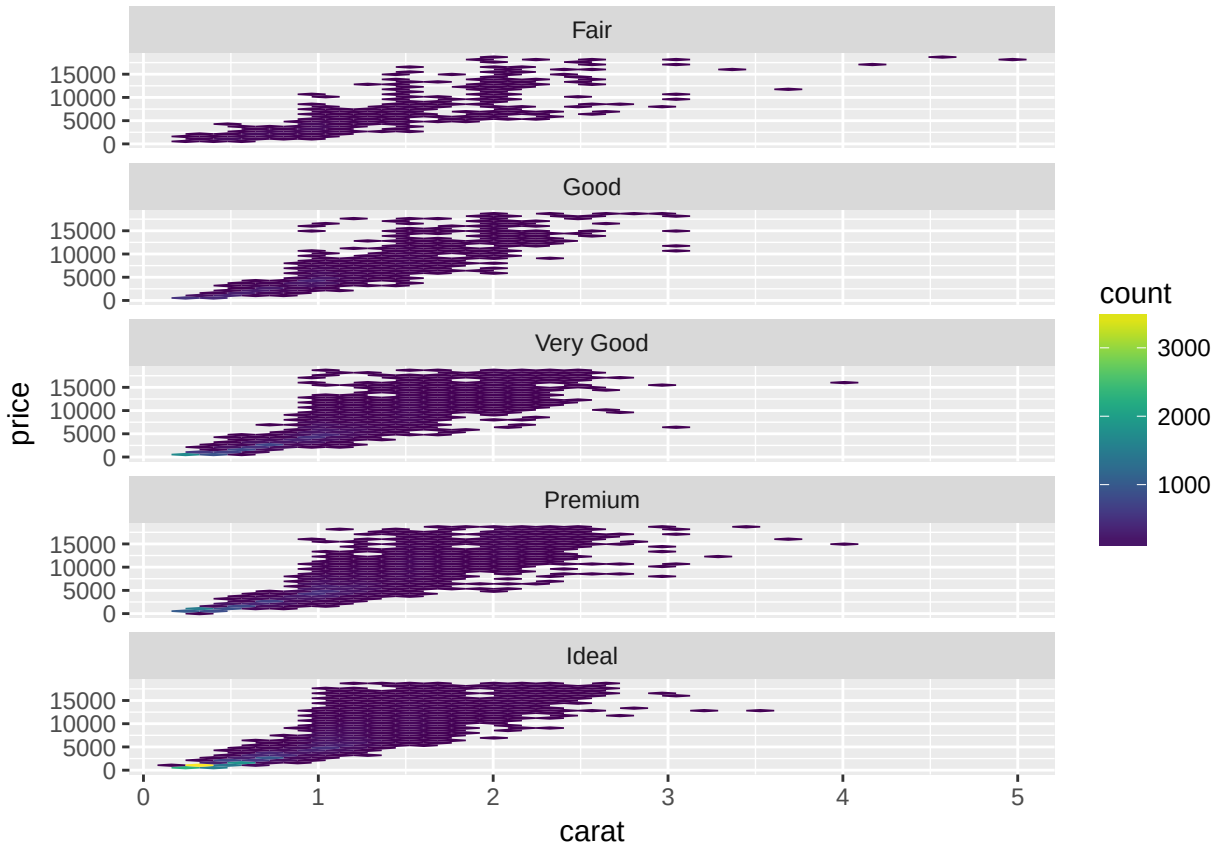
```
## Warning: position_dodge requires non-overlapping x intervals
```



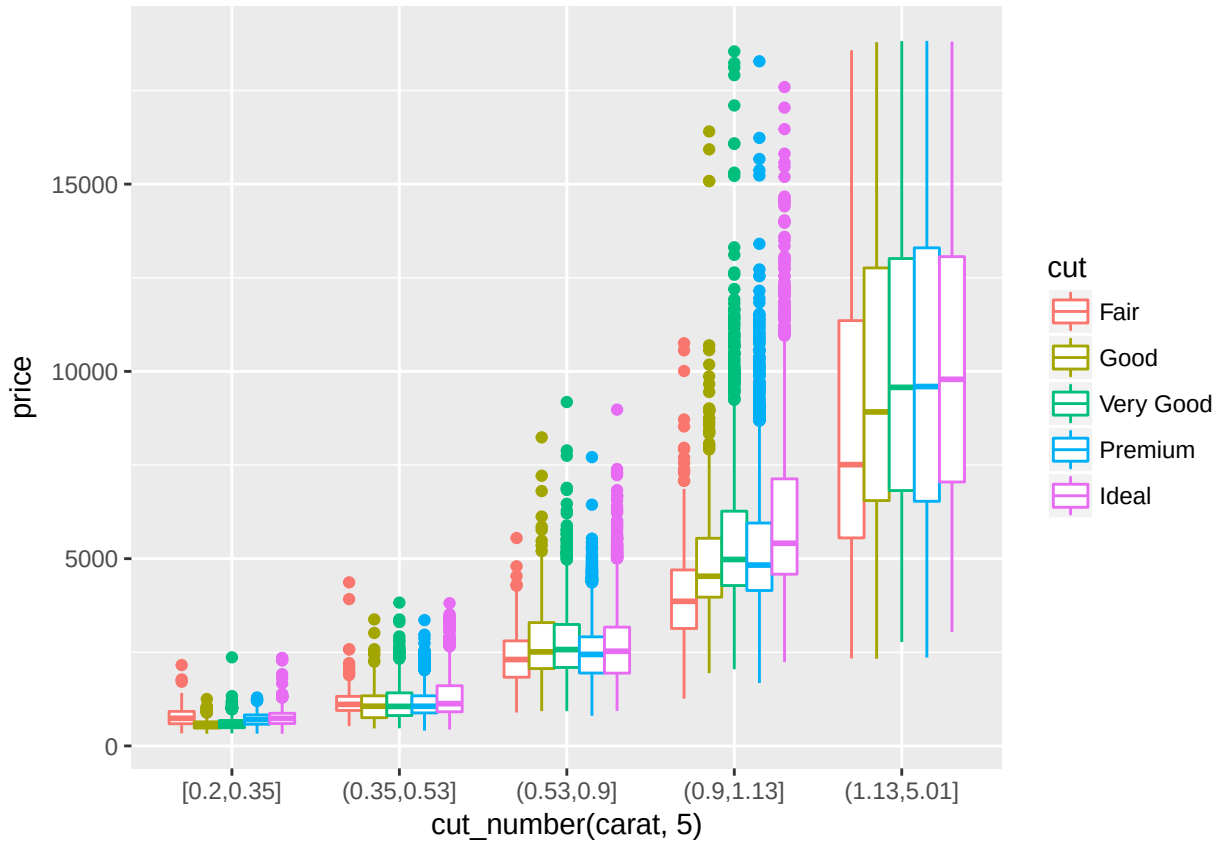

```
ggplot(data = diamonds, aes(x = cut_number(price, 10), y = carat)) +
  geom_boxplot() + coord_flip() + xlab("price")
```



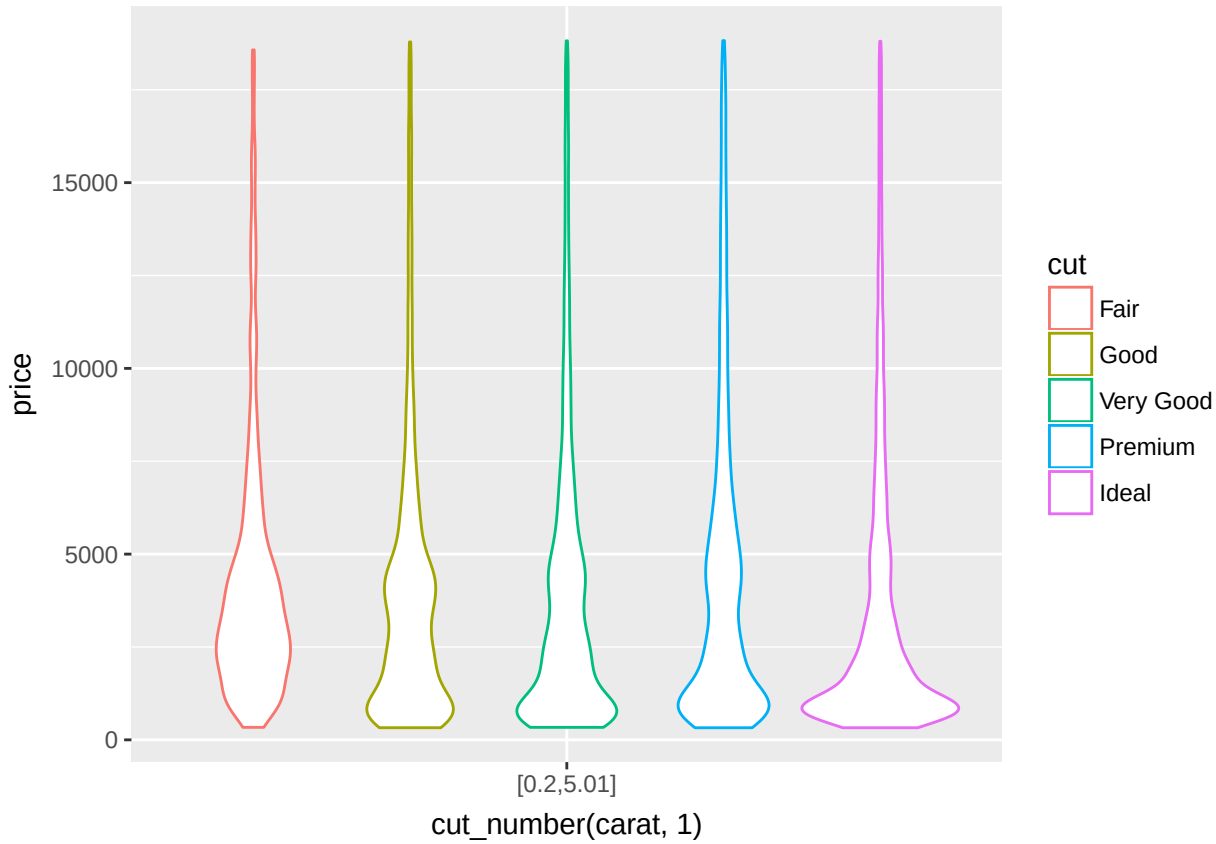

```
# Combine two of the techniques you've learned to visualize the combined distribution of cut, carat, an

# 1

ggplot(data = diamonds, aes(x = carat, y = price)) +
  geom_hex() +
  facet_wrap(~ cut, ncol = 1) +
  scale_fill_viridis()
```



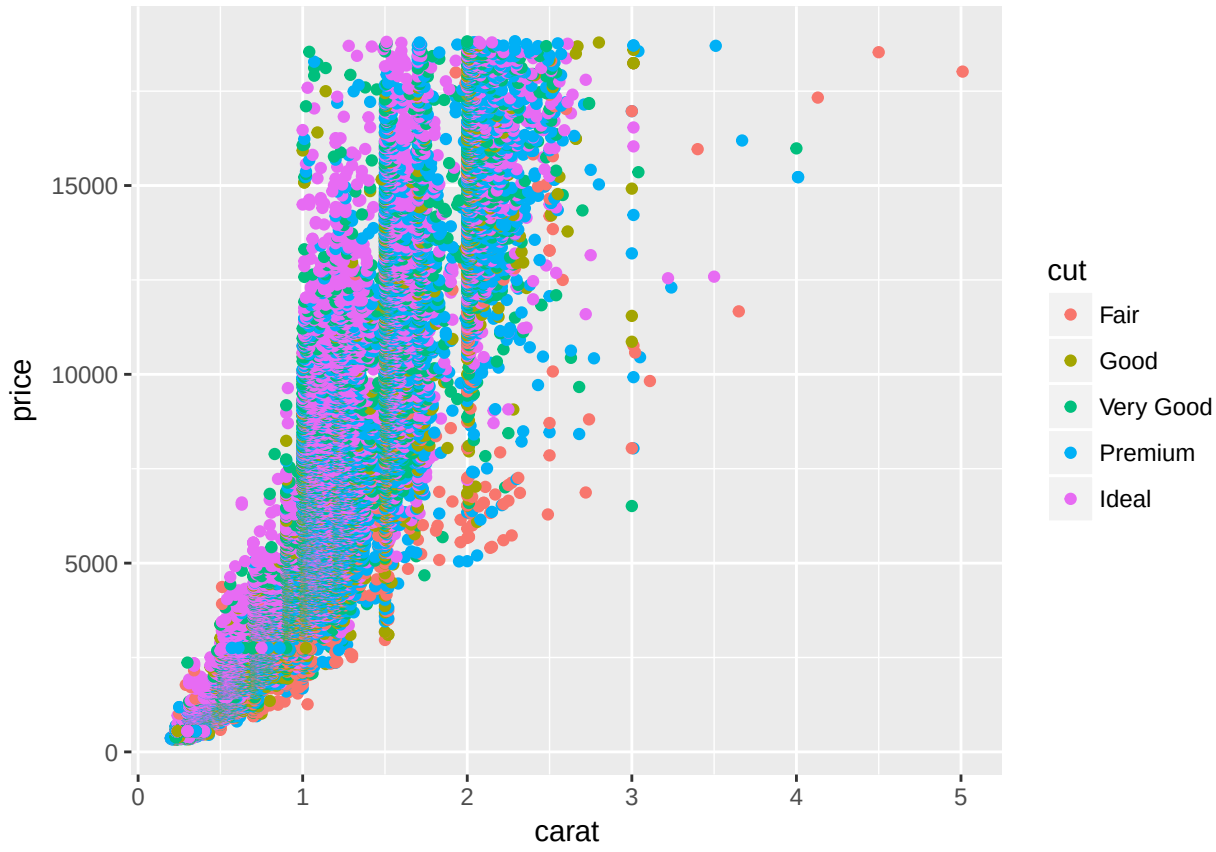

```
# 2

ggplot(data = diamonds, aes(x = cut_number(carat, 5), y = price, color = cut)) +
  geom_boxplot()
```




```
# 3

ggplot(data = diamonds, mapping = aes(x = cut_number(carat, 1), y = price, color = cut)) + geom_violin()
```




```
# 4

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

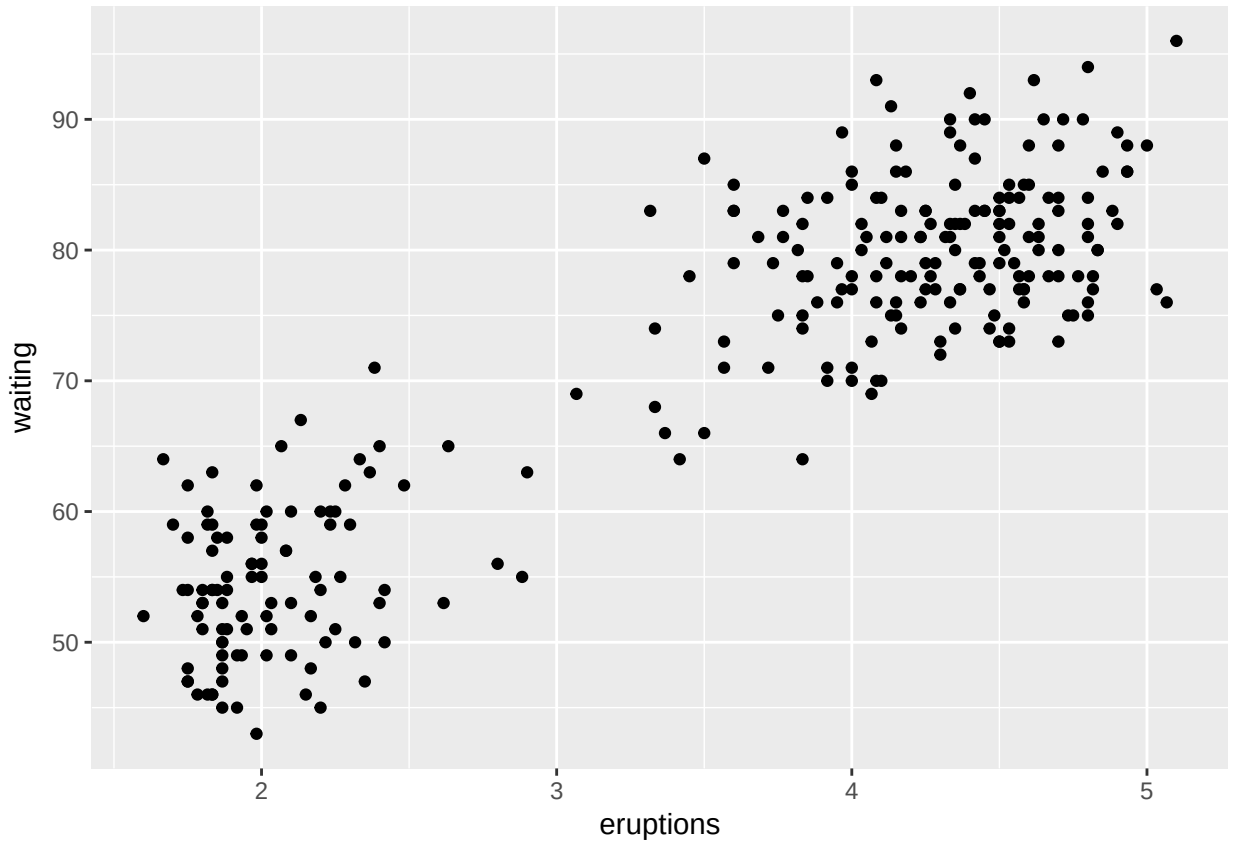


Patterns and Models

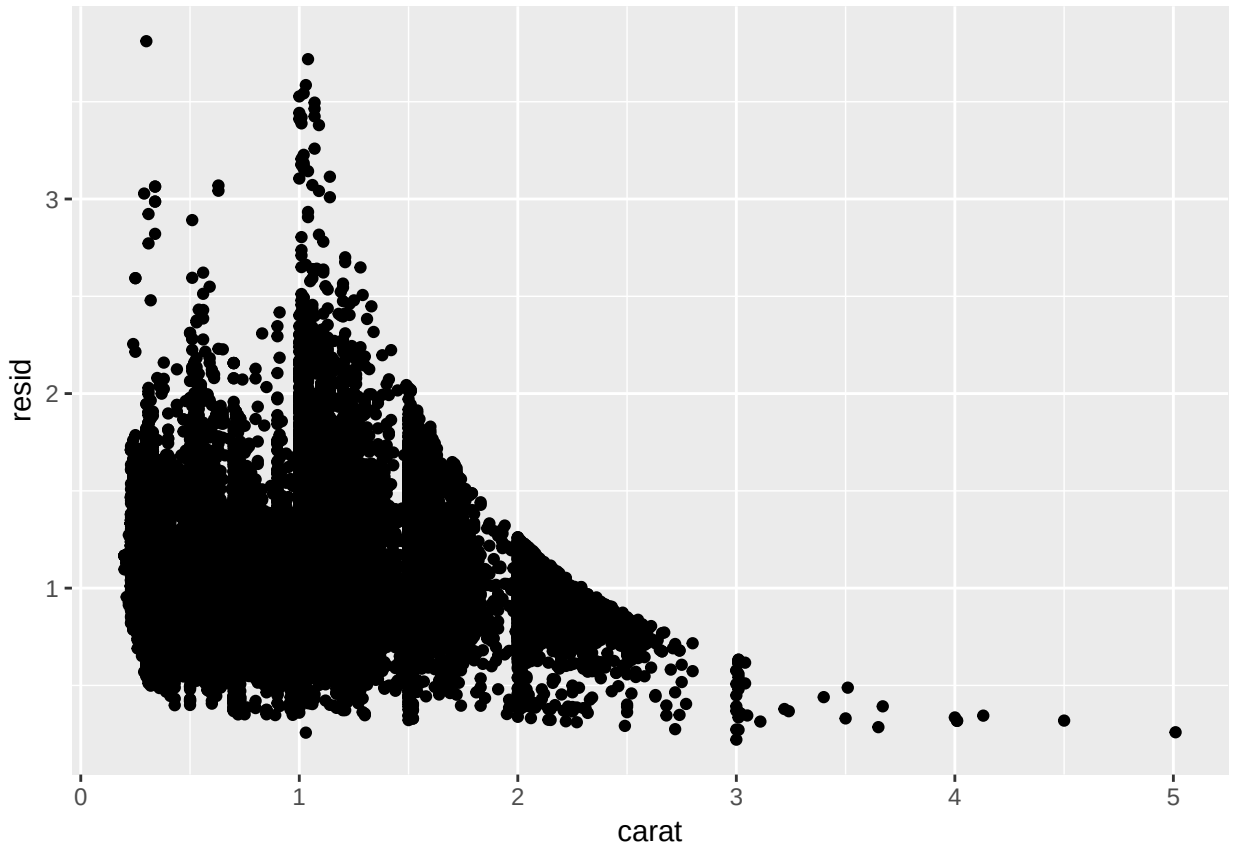
When spotting a pattern, ask:

* Could this be due to coincedence?
* How can you describe the relationship?
* How strong is the relationship?
* What other variables might affect the relationship?
* Does the relationship change if you look at individual subgroups of the data?

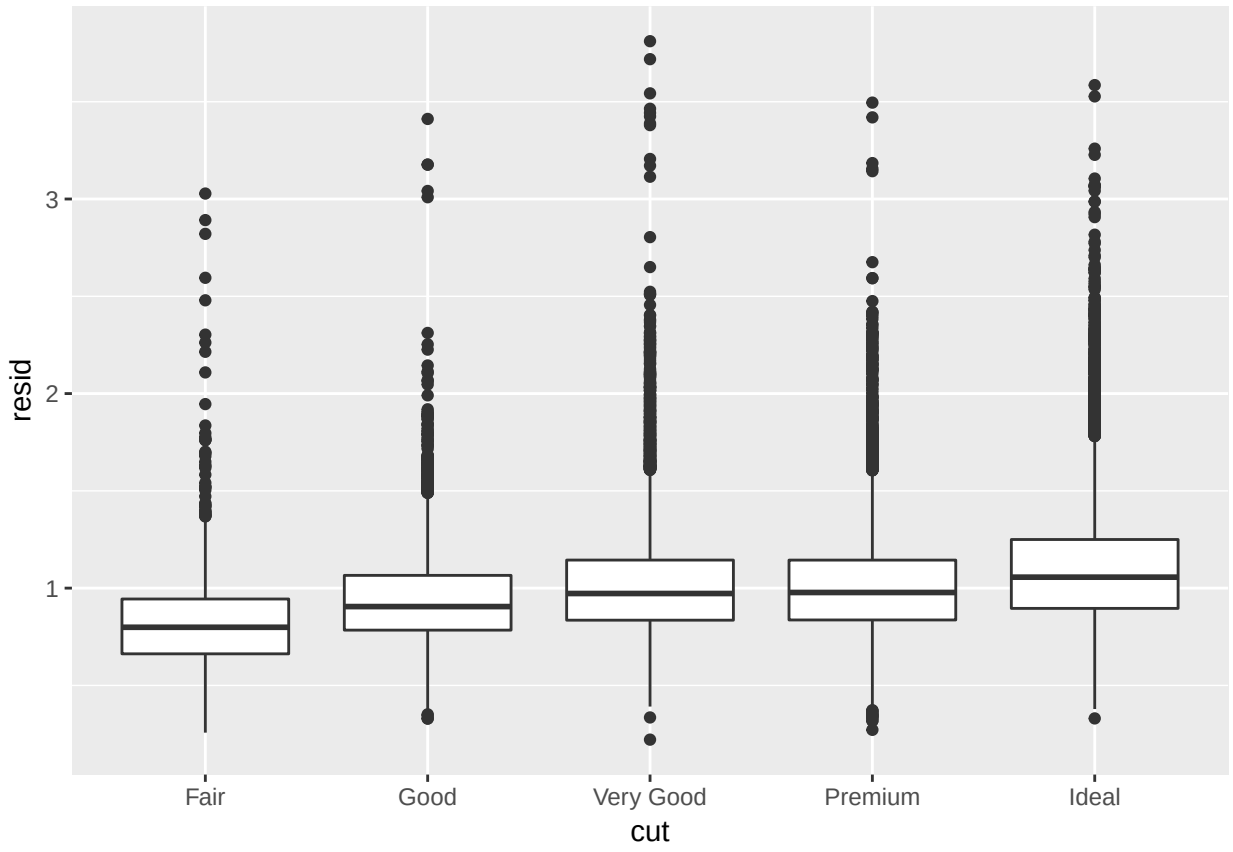
```
# plot old faithful. Notice the grouping. Clustering likely works best
ggplot(data = faithful) +
  geom_point(mapping = aes(x = eruptions, y = waiting))
```




```
# make log price vs log carat model
mod <- lm(log(price) ~ log(carat), data = diamonds)

# add residuals to data set
diamonds2 <- diamonds %>%
  add_residuals(mod) %>%
  mutate(resid = exp(resid))

# plot residuals

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

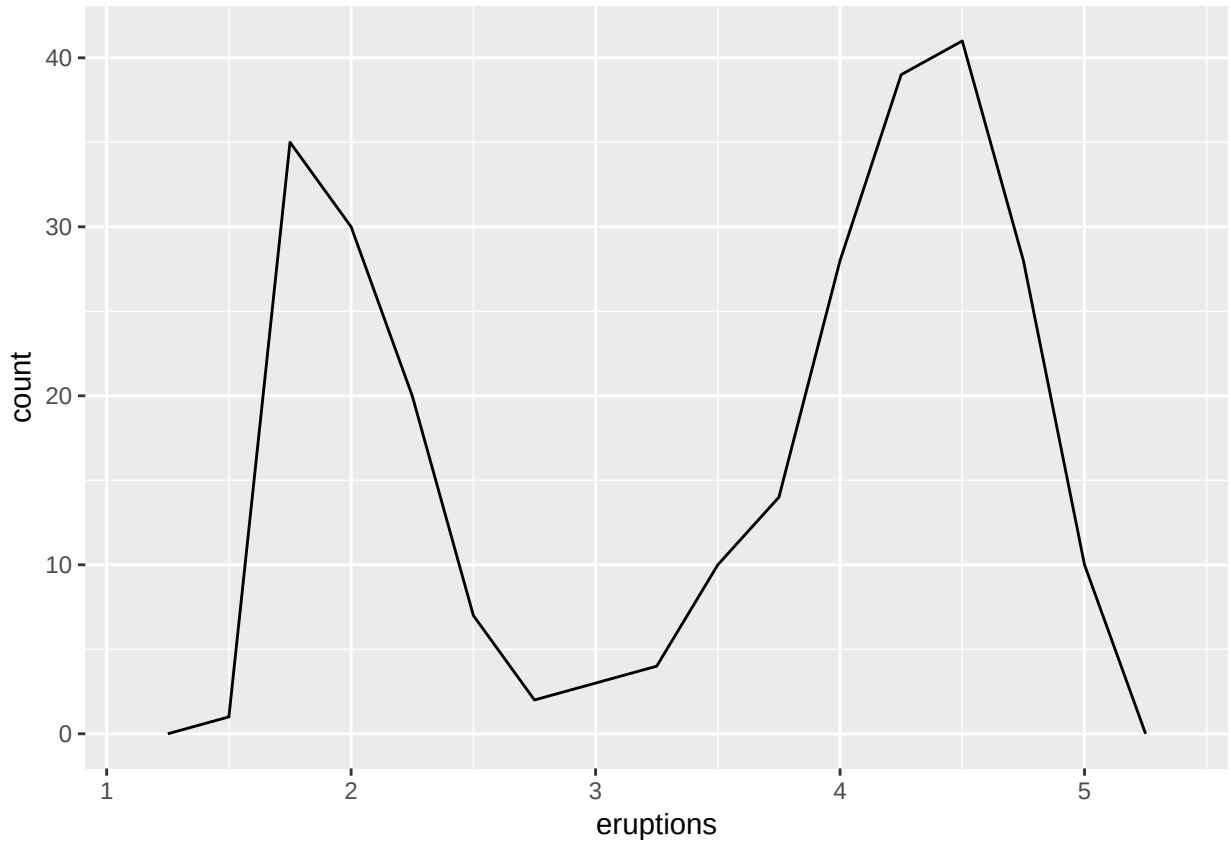



```
# plot boxplot
ggplot(data = diamonds2) +
  geom_boxplot(mapping = aes(x = cut, y = resid))
```

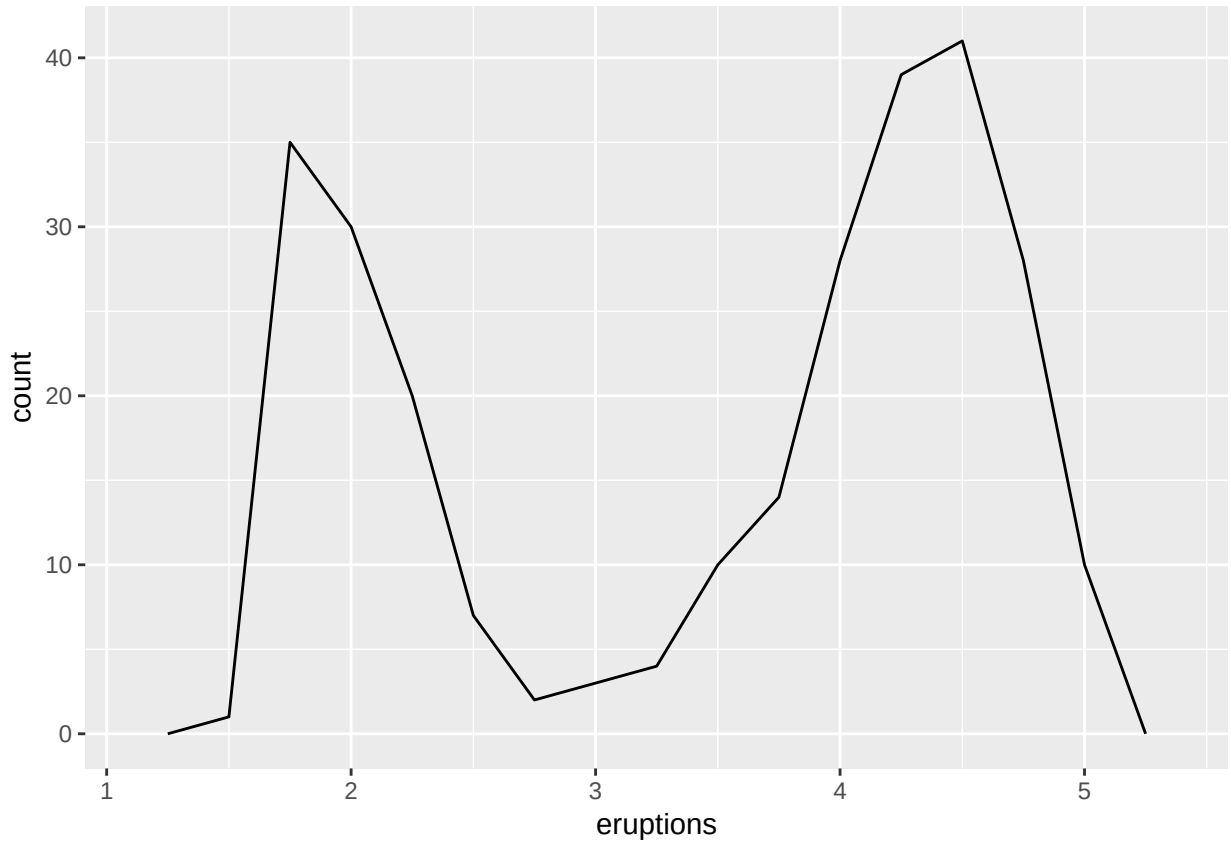


ggplot2 calls

```
# current
ggplot(data = faithful, mapping = aes(x = eruptions)) +
  geom_freqpoly(binwidth = 0.25)
```




```
# more concisely

ggplot(faithful, aes(eruptions)) +
  geom_freqpoly(binwidth = 0.25)
```


40
30
20
10
0
count
1
2
3
4
5
eruptions