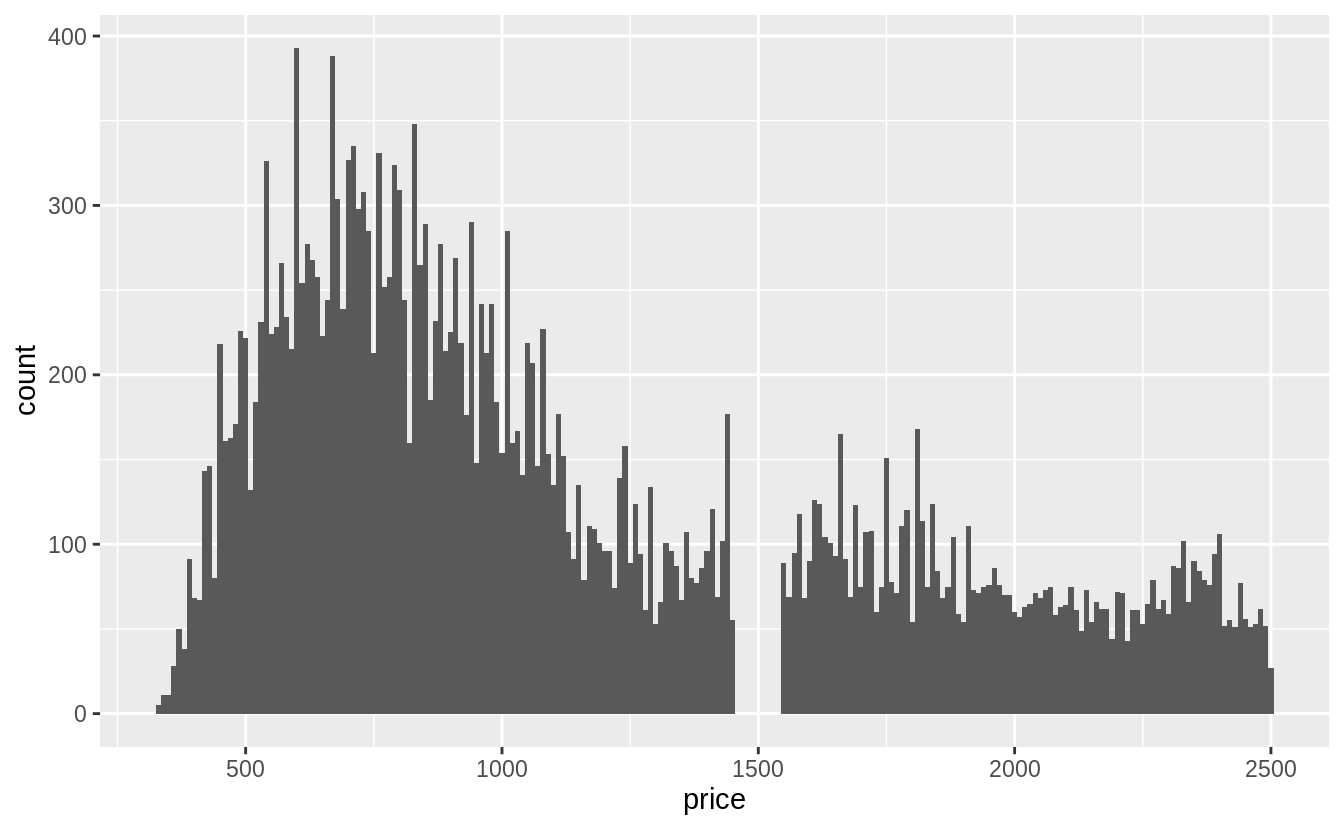
**שאלה 1**

Explore the distribution of price. Do you discover anything unusual or surprising? (Hint: Carefully think about the binwidth and make sure you try a wide range of values.)

* The price data has many spikes, but I can’t tell what each spike corresponds to. The following plots don’t show much difference in the distributions in the last one or two digits.
* There are no diamonds with a price of $1,500 (between $1,455 and $1,545, including).
* There’s a bulge in the distribution around $750.

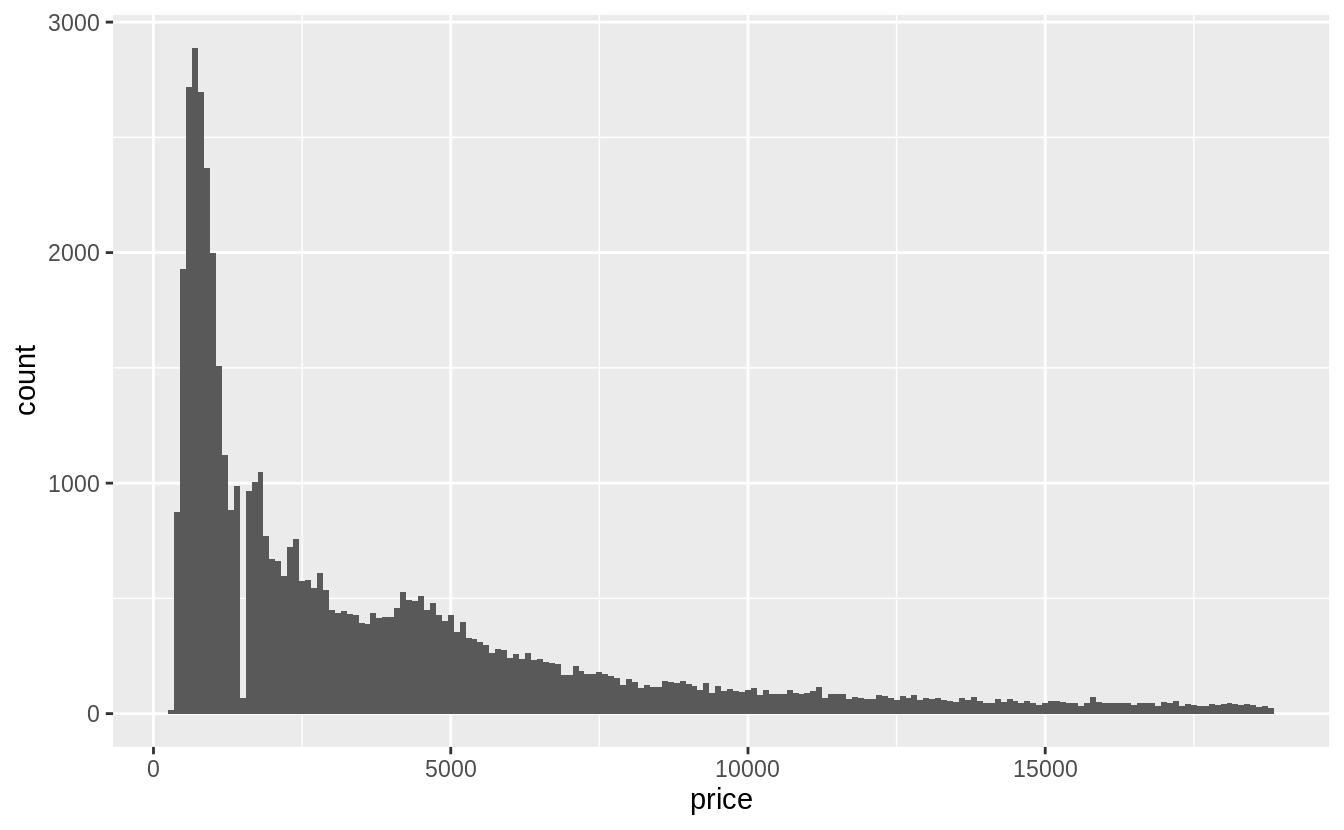
**ggplot**(**filter**(diamonds, price < 2500), **aes**(x = price)) +

**geom\_histogram**(binwidth = 10, center = 0)



**ggplot**(**filter**(diamonds), **aes**(x = price)) +

**geom\_histogram**(binwidth = 100, center = 0)



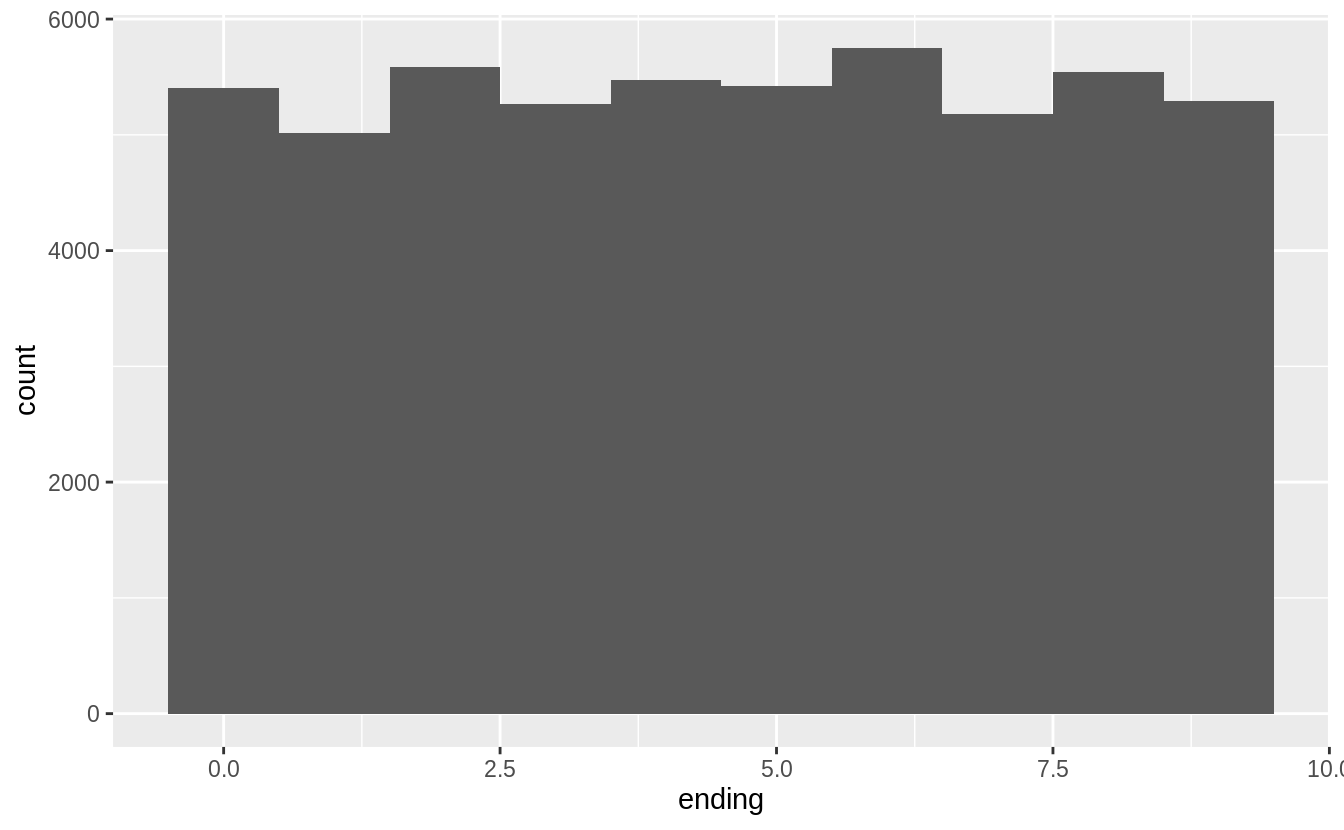
The last digits of prices are often not uniformly distributed. They are often round, ending in 0 or 5 (for one-half). Another common pattern is ending in 99, as in $1999. If we plot the distribution of the last one and two digits of prices do we observe patterns like that?

diamonds %>%

**mutate**(ending = price %% 10) %>%

**ggplot**(**aes**(x = ending)) +

**geom\_histogram**(binwidth = 1, center = 0)

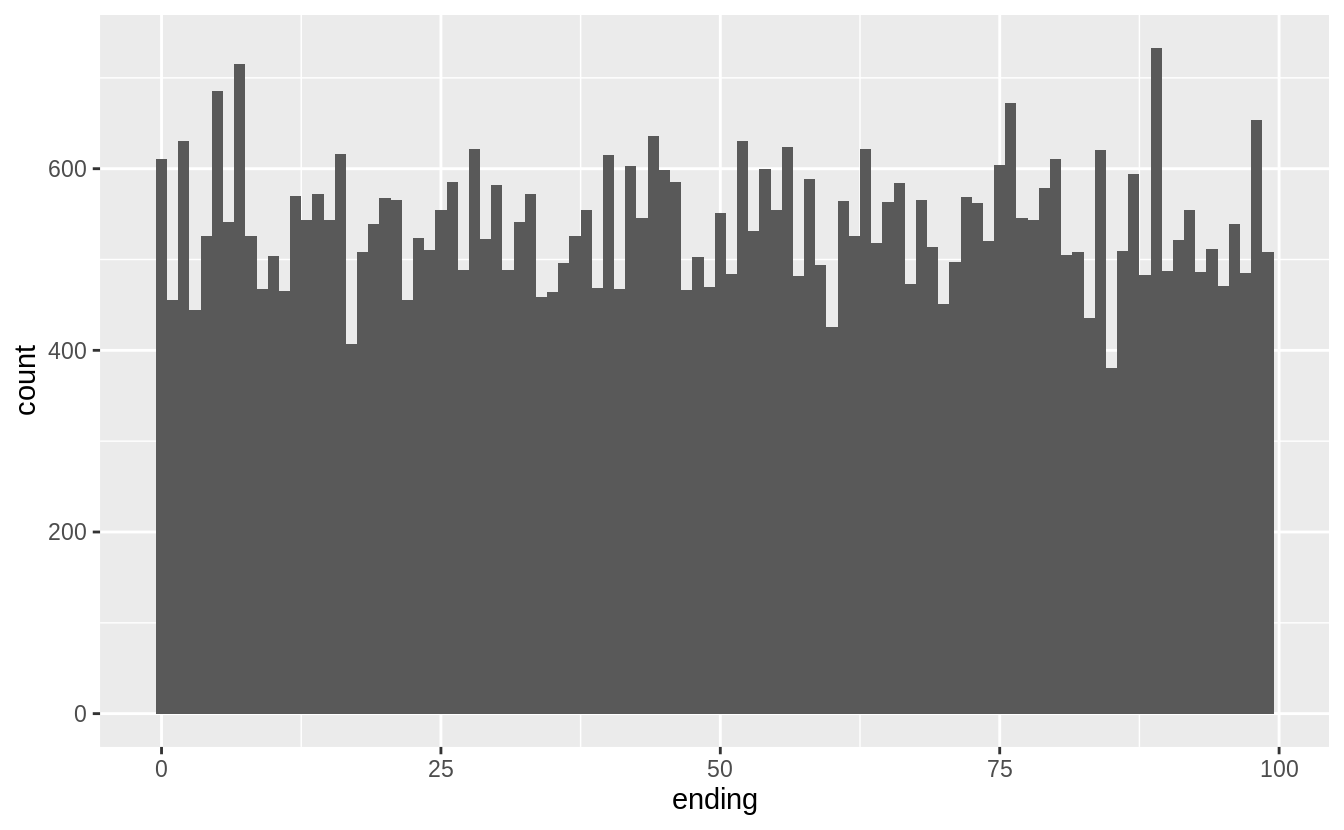


diamonds %>%

**mutate**(ending = price %% 100) %>%

**ggplot**(**aes**(x = ending)) +

**geom\_histogram**(binwidth = 1)



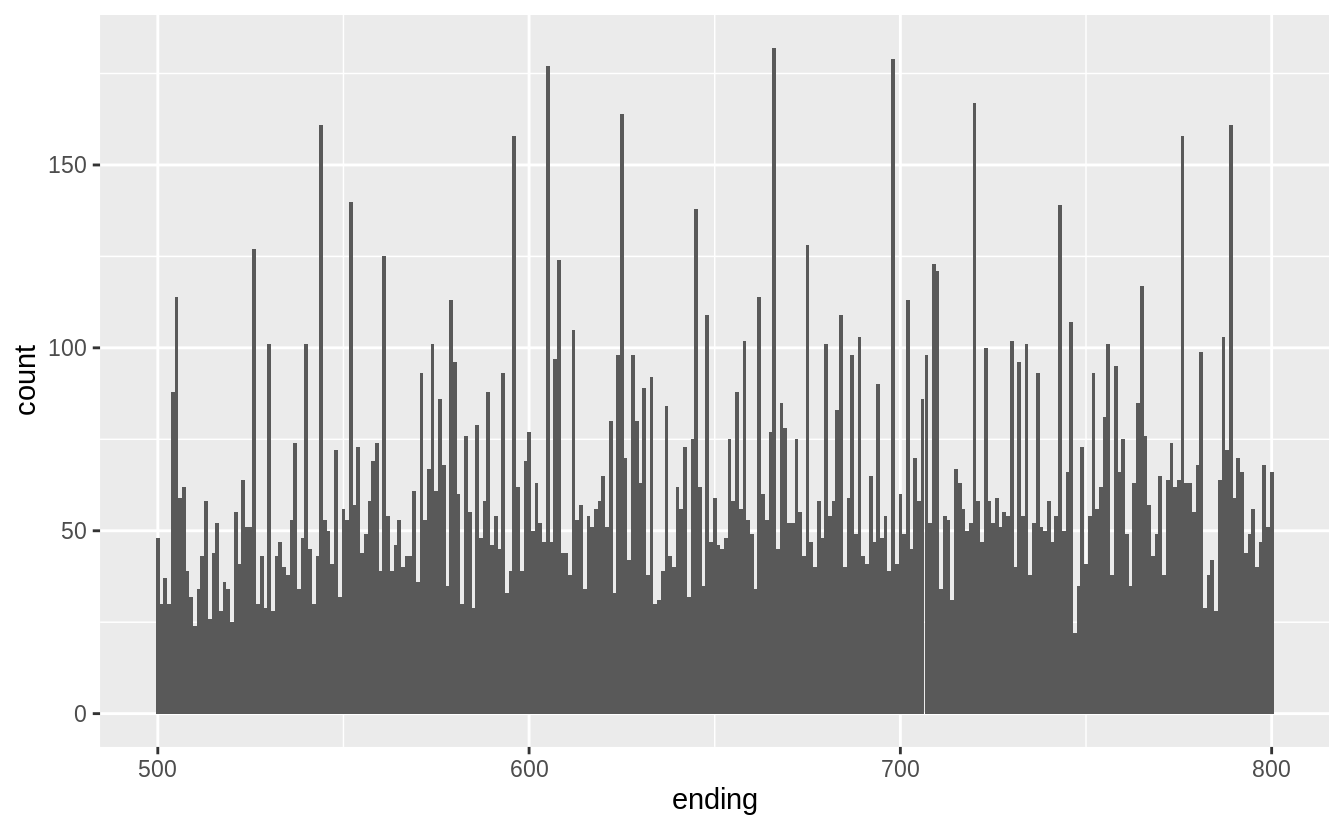
diamonds %>%

**mutate**(ending = price %% 1000) %>%

**filter**(ending >= 500, ending <= 800) %>%

**ggplot**(**aes**(x = ending)) +

**geom\_histogram**(binwidth = 1)



**שאלה 2**

How many diamonds are 0.99 carat? How many are 1 carat? What do you think is the cause of the difference?

There are more than 70 times as many 1 carat diamonds as 0.99 carat diamond.

diamonds %>%

**filter**(carat >= 0.99, carat <= 1) %>%

**count**(carat)

*#> # A tibble: 2 x 2*

*#> carat n*

*#> <dbl> <int>*

*#> 1 0.99 23*

*#> 2 1 1558*

I don’t know exactly the process behind how carats are measured, but some way or another some diamonds carat values are being “rounded up” Presumably there is a premium for a 1 carat diamond vs. a 0.99 carat diamond beyond the expected increase in price due to a 0.01 carat increase.[7](https://jrnold.github.io/r4ds-exercise-solutions/exploratory-data-analysis.html#fn7)

To check this intuition, we would want to look at the number of diamonds in each carat range to see if there is an unusually low number of 0.99 carat diamonds, and an abnormally large number of 1 carat diamonds.

diamonds %>%

**filter**(carat >= 0.9, carat <= 1.1) %>%

**count**(carat) %>%

**print**(n = Inf)

*#> # A tibble: 21 x 2*

*#> carat n*

*#> <dbl> <int>*

*#> 1 0.9 1485*

*#> 2 0.91 570*

*#> 3 0.92 226*

*#> 4 0.93 142*

*#> 5 0.94 59*

*#> 6 0.95 65*

*#> 7 0.96 103*

*#> 8 0.97 59*

*#> 9 0.98 31*

*#> 10 0.99 23*

*#> 11 1 1558*

*#> 12 1.01 2242*

*#> 13 1.02 883*

*#> 14 1.03 523*

*#> 15 1.04 475*

*#> 16 1.05 361*

*#> 17 1.06 373*

*#> 18 1.07 342*

*#> 19 1.08 246*

*#> 20 1.09 287*

*#> 21 1.1 278*

**שאלה 3**

Compare and contrast coord\_cartesian() vs xlim() or ylim() when zooming in on a histogram. What happens if you leave binwidth unset? What happens if you try and zoom so only half a bar shows?

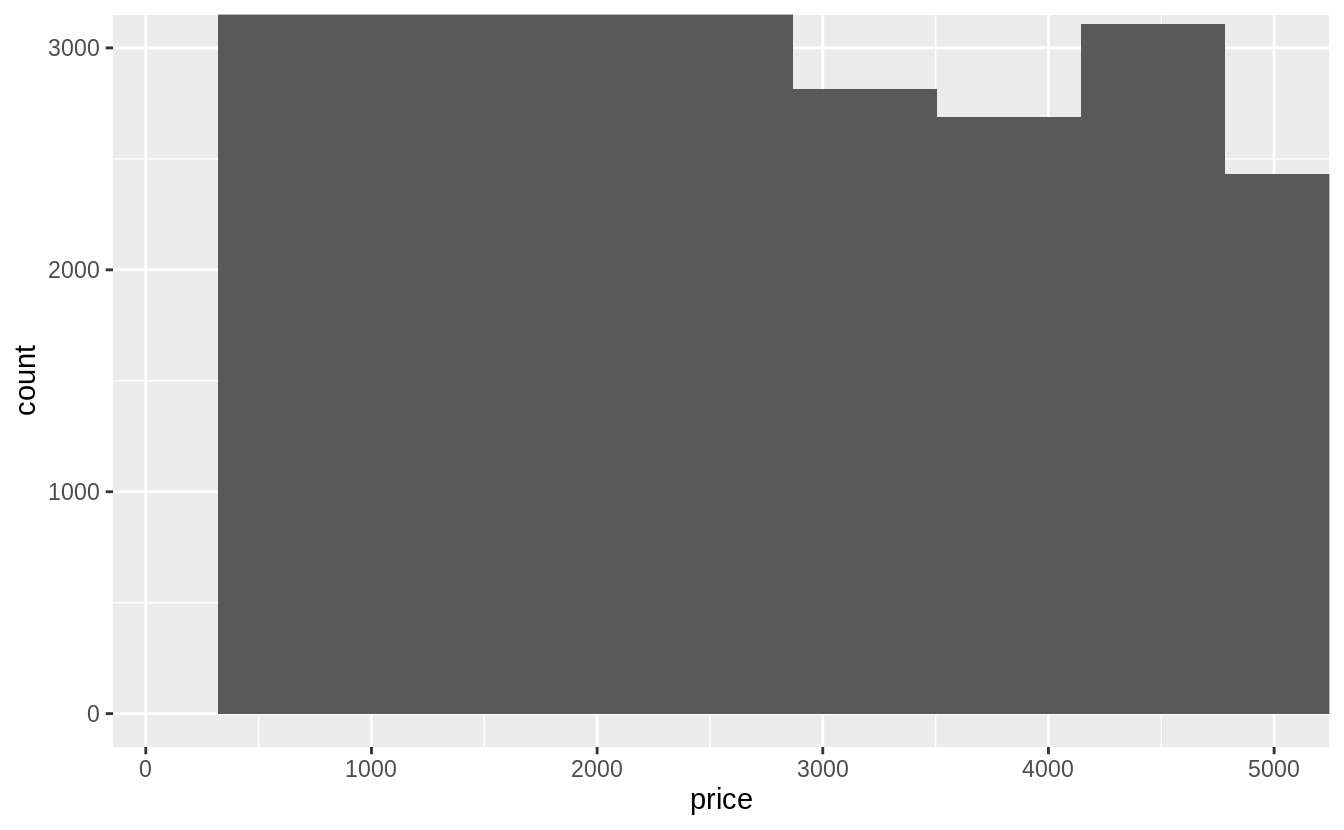
The coord\_cartesian() function zooms in on the area specified by the limits, after having calculated and drawn the geoms. Since the histogram bins have already been calculated, it is unaffected.

**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`.*



However, the xlim() and ylim() functions influence actions before the calculation of the stats related to the histogram. Thus, any values outside the x- and y-limits are dropped before calculating bin widths and counts. This can influence how the histogram looks.

**ggplot**(diamonds) +

**geom\_histogram**(mapping = **aes**(x = price)) +

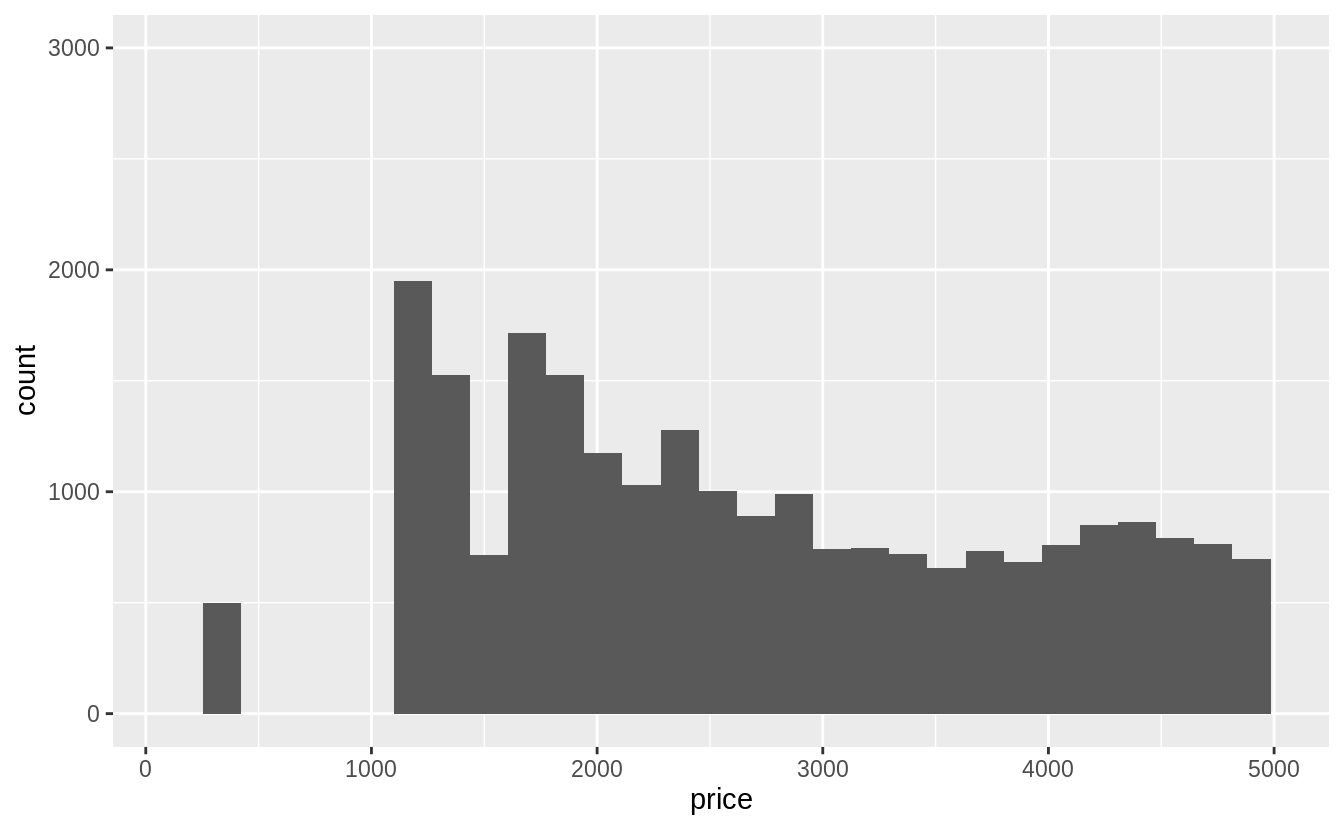
**xlim**(100, 5000) +

**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 6 rows containing missing values (geom\_bar).*



**שאלה 4**

What happens to missing values in a histogram? What happens to missing values in a bar chart? Why is there a difference?

Missing values are removed when the number of observations in each bin are calculated. See the warning message: Removed 9 rows containing non-finite values (stat\_bin)

diamonds2 <- diamonds %>%

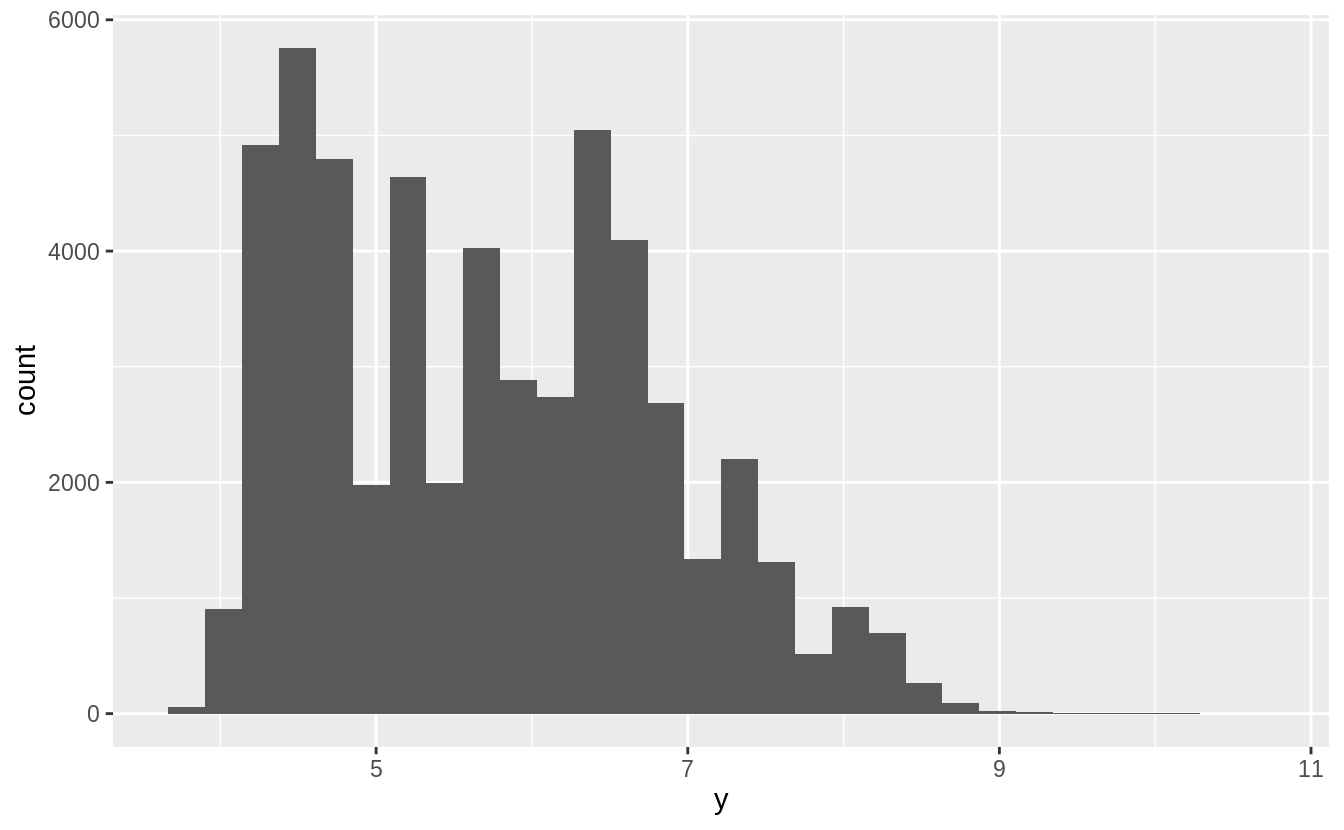
**mutate**(y = **ifelse**(y < 3 | y > 20, NA, y))

**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).*



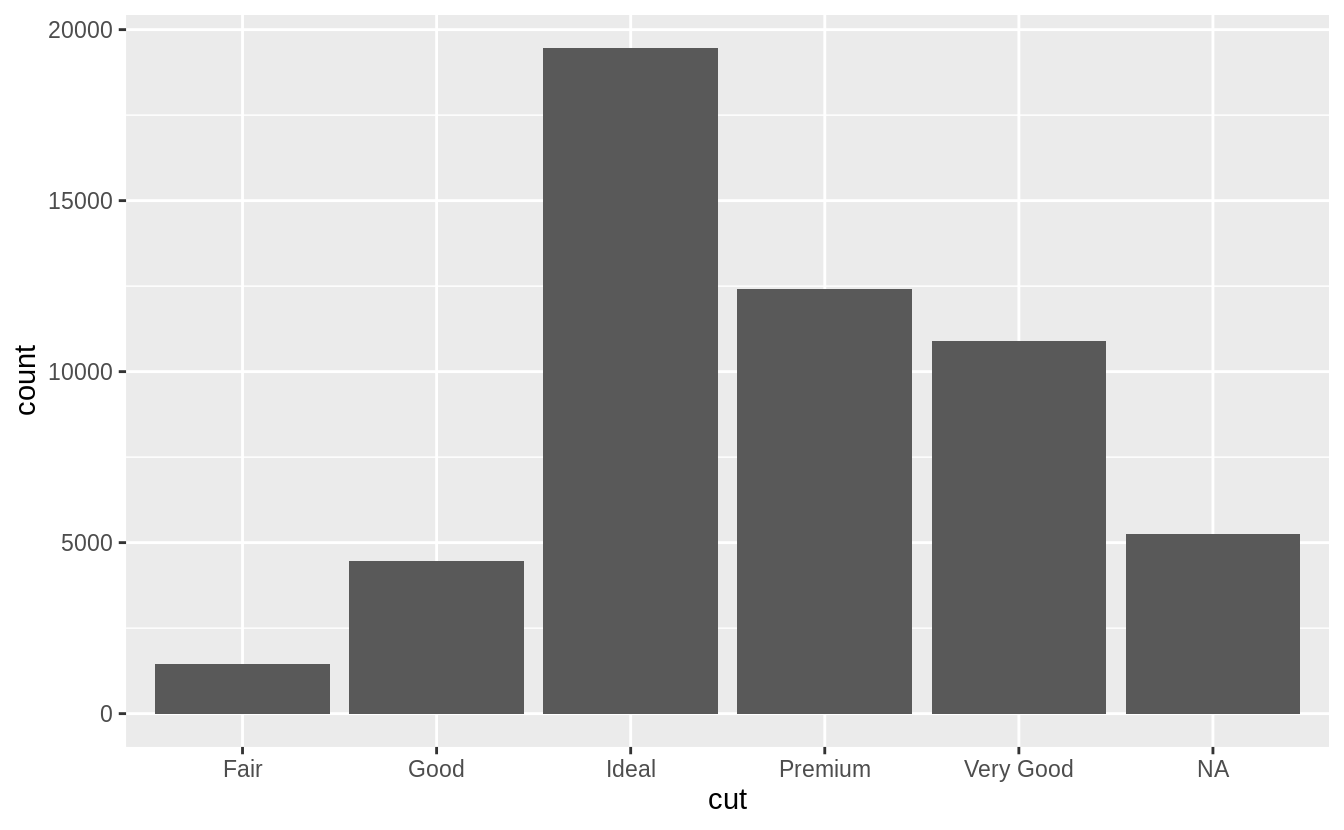
In the geom\_bar() function, NA is treated as another category. The x aesthetic in geom\_bar() requires a discrete (categorical) variable, and missing values act like another category.

diamonds %>%

**mutate**(cut = **if\_else**(**runif**(**n**()) < 0.1, NA\_character\_, **as.character**(cut))) %>%

**ggplot**() +

**geom\_bar**(mapping = **aes**(x = cut))



In a histogram, the x aesthetic variable needs to be numeric, and stat\_bin() groups the observations by ranges into bins. Since the numeric value of the NA observations is unknown, they cannot be placed in a particular bin, and are dropped.

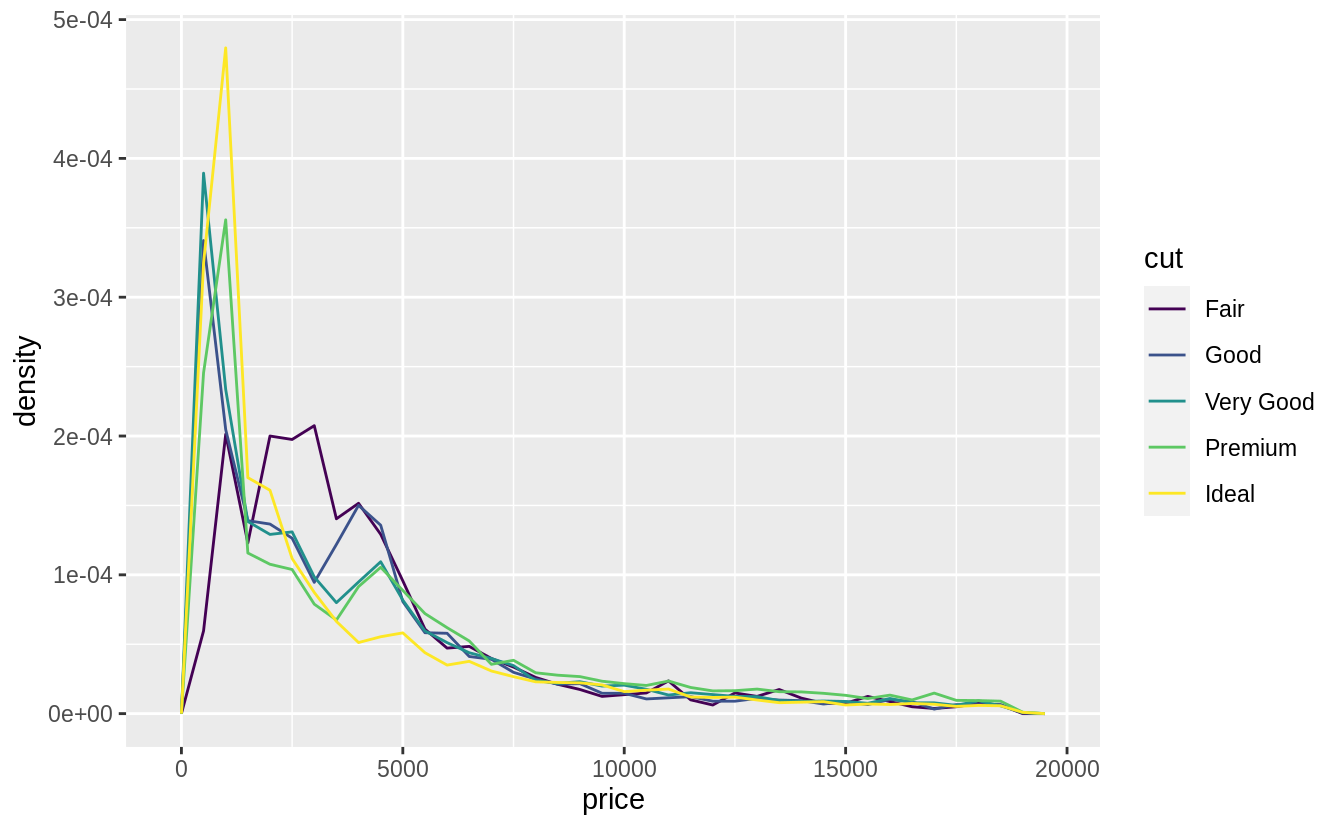
**שאלה 5**

Compare and contrast geom\_violin() with a faceted geom\_histogram(), or a colored geom\_freqpoly(). What are the pros and cons of each method?

I produce plots for these three methods below. The geom\_freqpoly() is better for look-up: meaning that given a price, it is easy to tell which cut has the highest density. However, the overlapping lines makes it difficult to distinguish how the overall distributions relate to each other. The geom\_violin() and faceted geom\_histogram() have similar strengths and weaknesses. It is easy to visually distinguish differences in the overall shape of the distributions (skewness, central values, variance, etc). However, since we can’t easily compare the vertical values of the distribution, it is difficult to look up which category has the highest density for a given price. All of these methods depend on tuning parameters to determine the level of smoothness of the distribution.

**ggplot**(data = diamonds, mapping = **aes**(x = price, y = ..density..)) +

**geom\_freqpoly**(mapping = **aes**(color = cut), binwidth = 500)

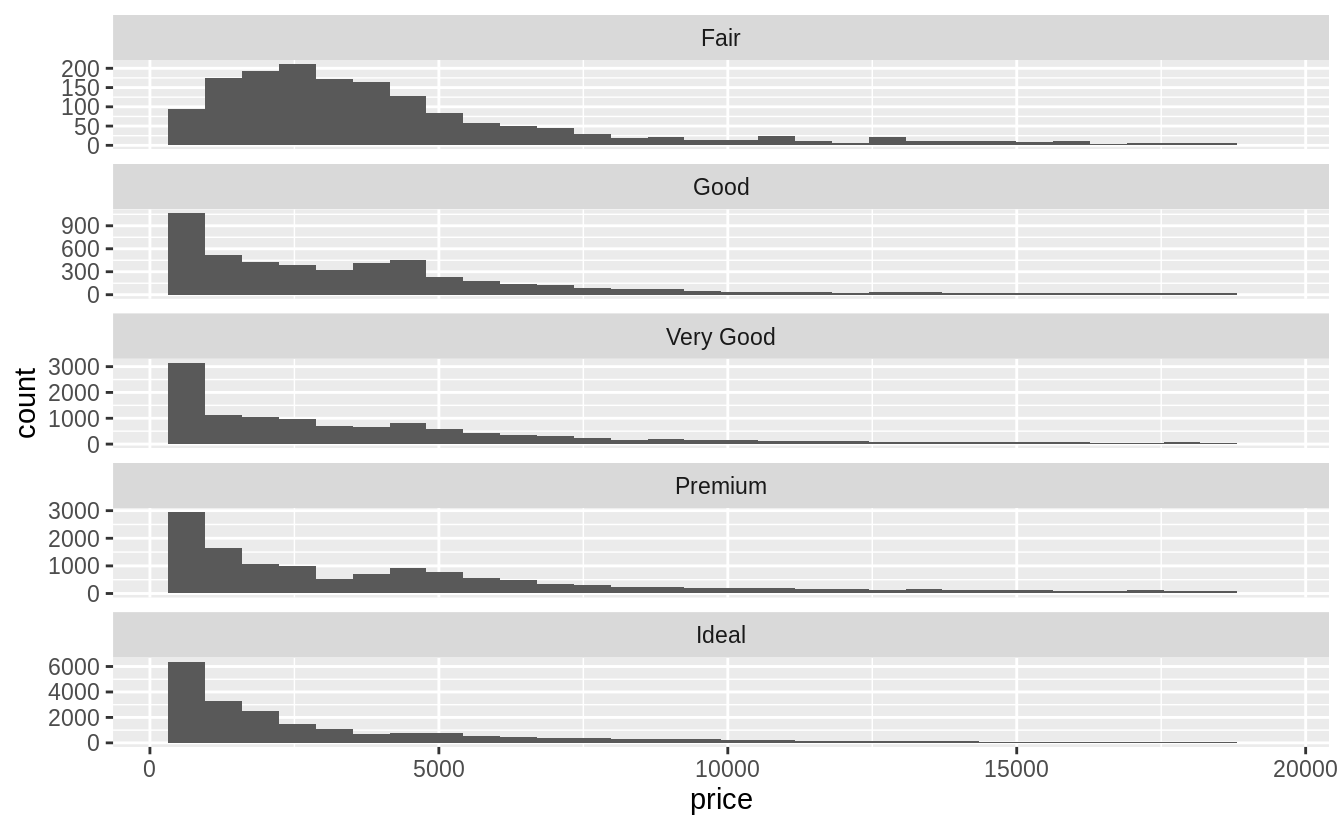


**ggplot**(data = diamonds, mapping = **aes**(x = price)) +

**geom\_histogram**() +

**facet\_wrap**(~cut, ncol = 1, scales = "free\_y")

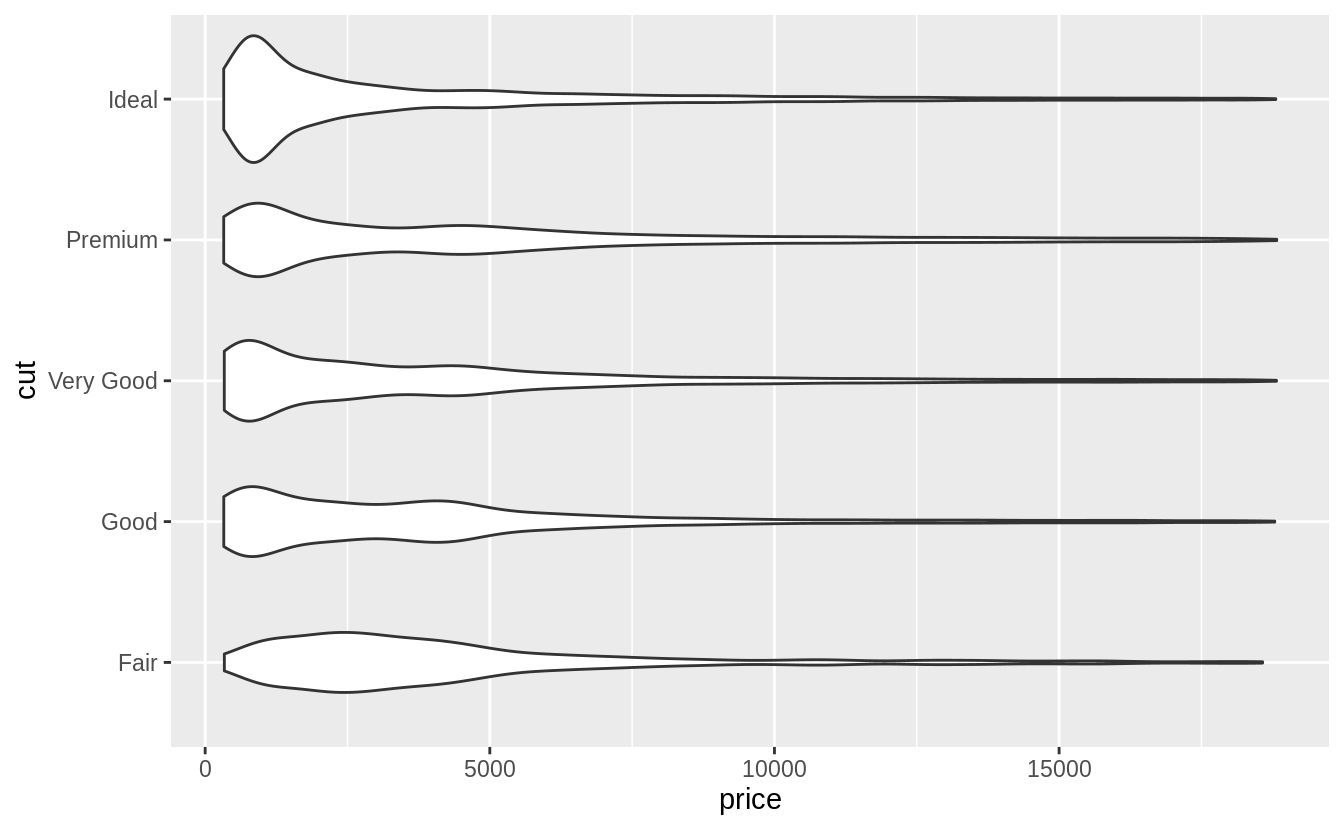
*#> `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.*



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

**geom\_violin**() +

**coord\_flip**()



The violin plot was first described in Hintze and Nelson ([1998](https://jrnold.github.io/r4ds-exercise-solutions/exploratory-data-analysis.html#ref-HintzeNelson1998)).

**שאלה 6 (לשנות בקובץ ה Html)**

6. הצג את ההתפלגות של קראט בפילוח לפי מחיר היהלומים.

**6. הצג את ההתפלגות של המחיר היהלומים בפילוח לפי קראט.**

Visualize the distribution of carat, partitioned by price.

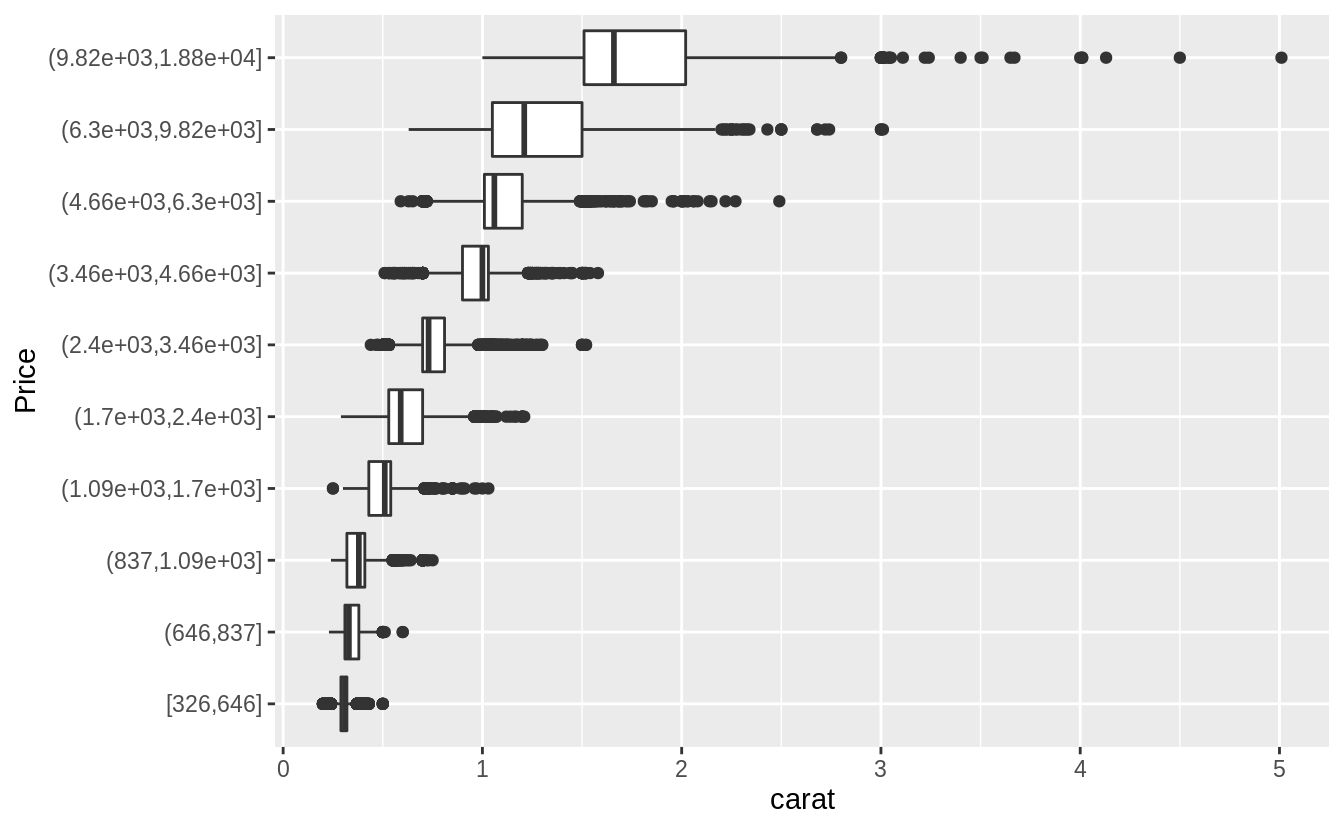
Plotted with a box plot with 10 bins with an equal number of observations, and the width determined by the number of observations.

**ggplot**(diamonds, **aes**(x = **cut\_number**(price, 10), y = carat)) +

**geom\_boxplot**() +

**coord\_flip**() +

**xlab**("Price")

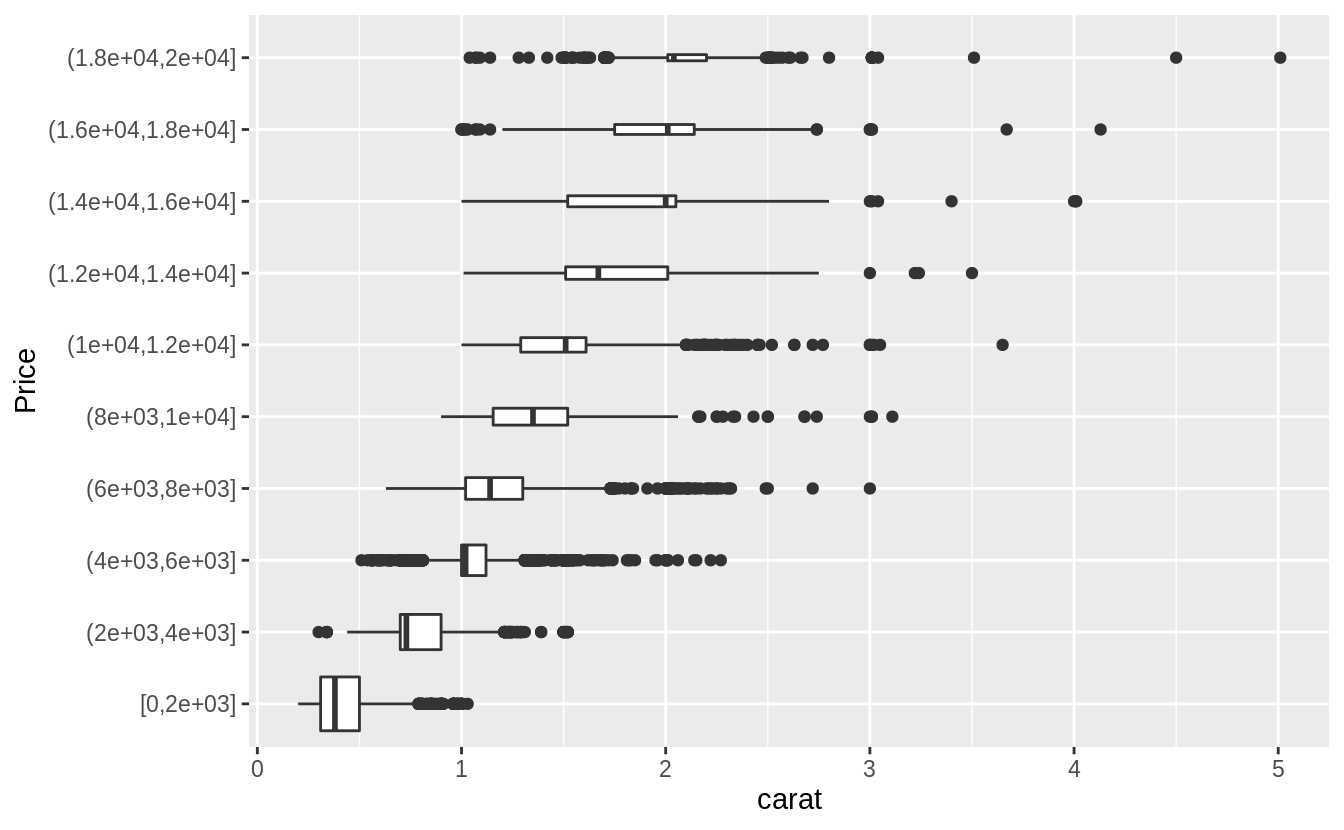
Plotted with a box plot with 10 equal-width bins of $2,000. The argument boundary = 0 ensures that first bin is $0–$2,000.

**ggplot**(diamonds, **aes**(x = **cut\_width**(price, 2000, boundary = 0), y = carat)) +

**geom\_boxplot**(varwidth = TRUE) +

**coord\_flip**() +

**xlab**("Price")



**שאלה 7**

If you have a small dataset, it’s sometimes useful to use geom\_jitter() to see the relationship between a continuous and categorical variable. The ggbeeswarm package provides a number of methods similar to geom\_jitter(). List them and briefly describe what each one does.

There are two methods:

* geom\_quasirandom() produces plots that are a mix of jitter and violin plots. There are several different methods that determine exactly how the random location of the points is generated.
* geom\_beeswarm() produces a plot similar to a violin plot, but by offsetting the points.

I’ll use the mpg box plot example since these methods display individual points, they are better suited for smaller datasets.

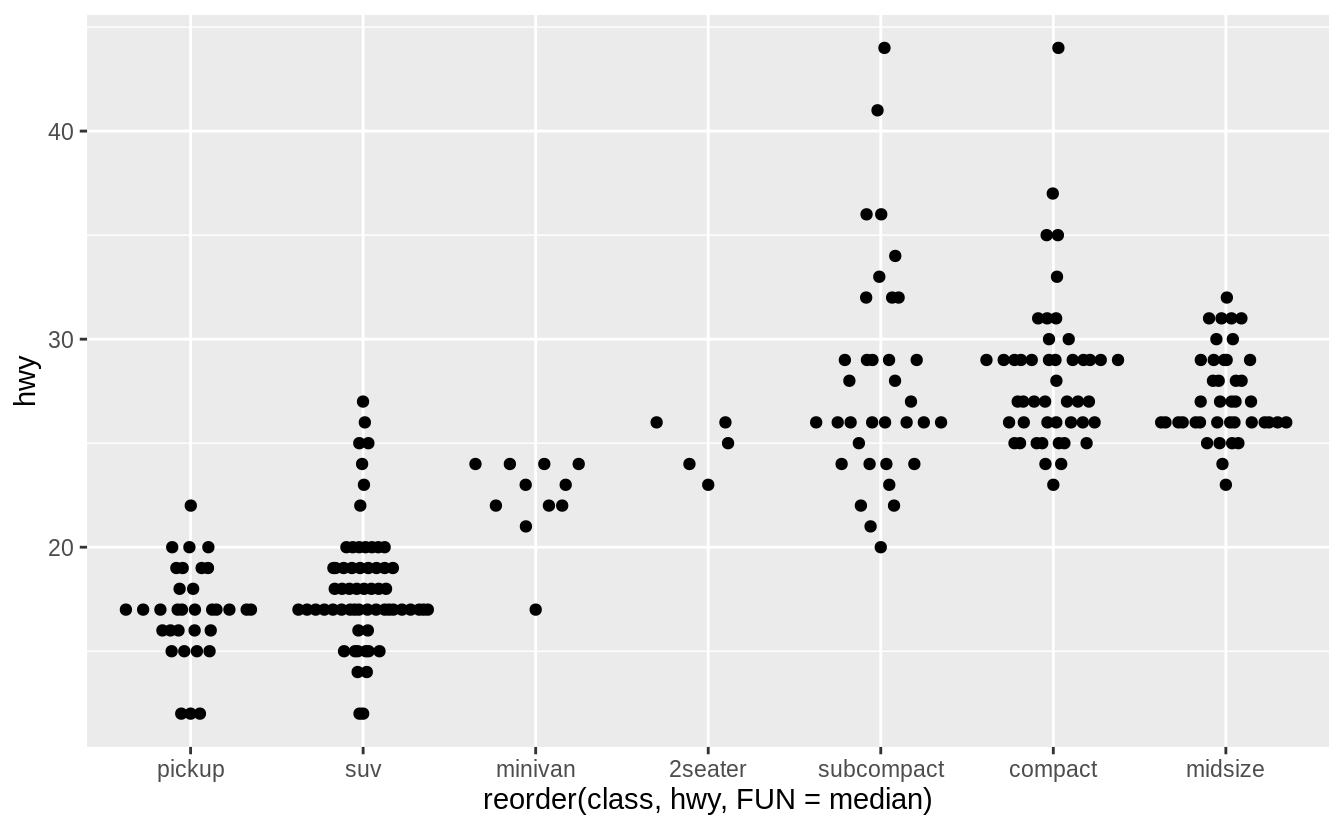
**ggplot**(data = mpg) +

**geom\_quasirandom**(mapping = **aes**(

x = **reorder**(class, hwy, FUN = median),

y = hwy

))



**ggplot**(data = mpg) +

**geom\_quasirandom**(

mapping = **aes**(

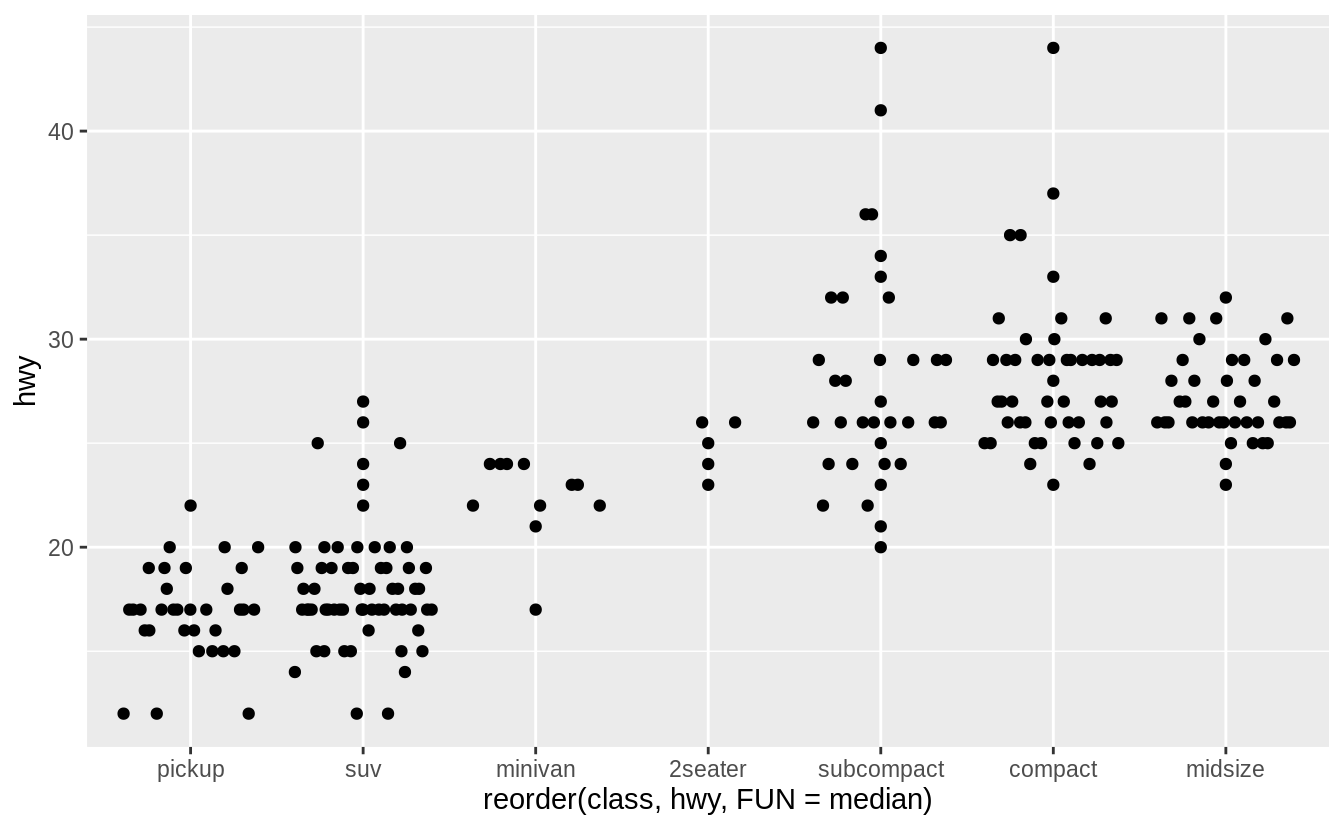
x = **reorder**(class, hwy, FUN = median),

y = hwy

),

method = "tukey"

)



**ggplot**(data = mpg) +

**geom\_quasirandom**(

mapping = **aes**(

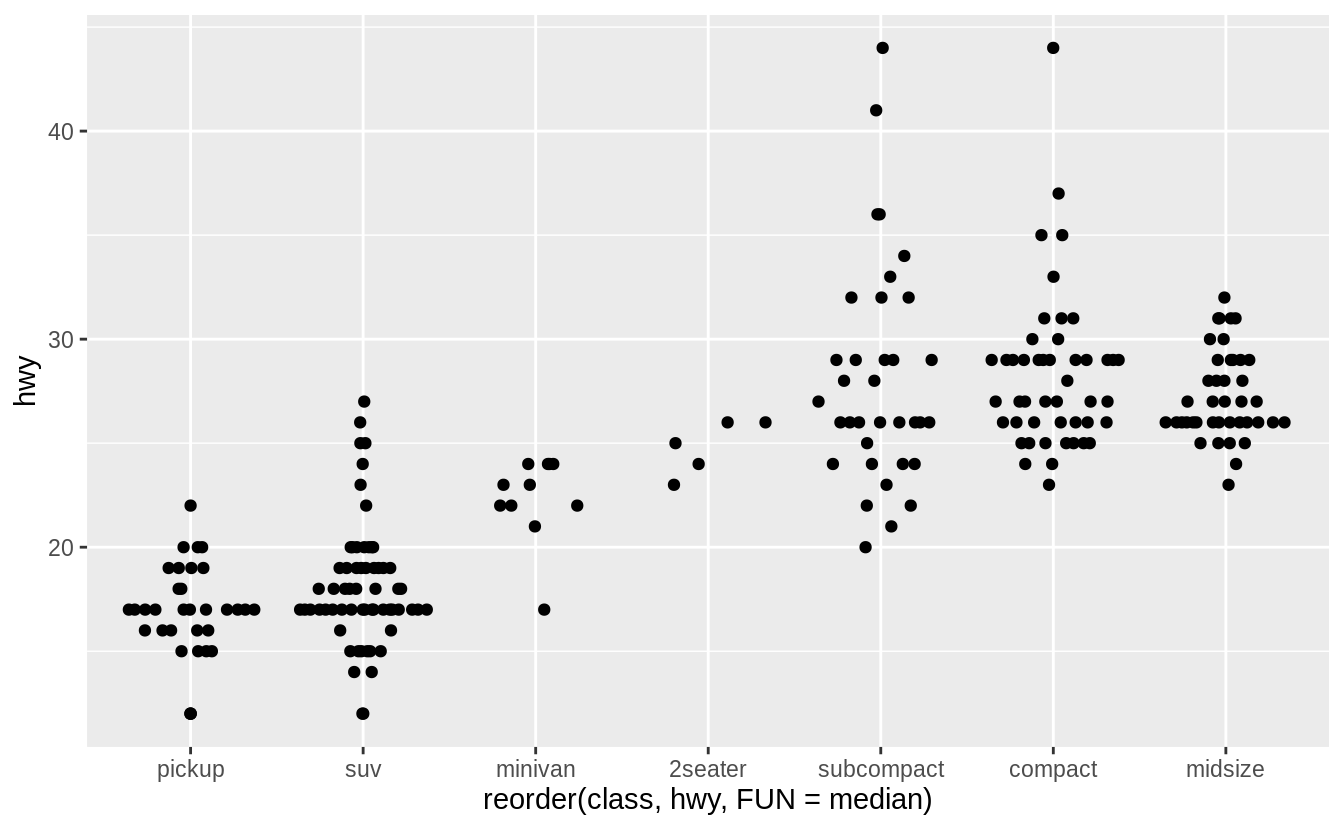
x = **reorder**(class, hwy, FUN = median),

y = hwy

),

method = "tukeyDense"

)



**ggplot**(data = mpg) +

**geom\_quasirandom**(

mapping = **aes**(

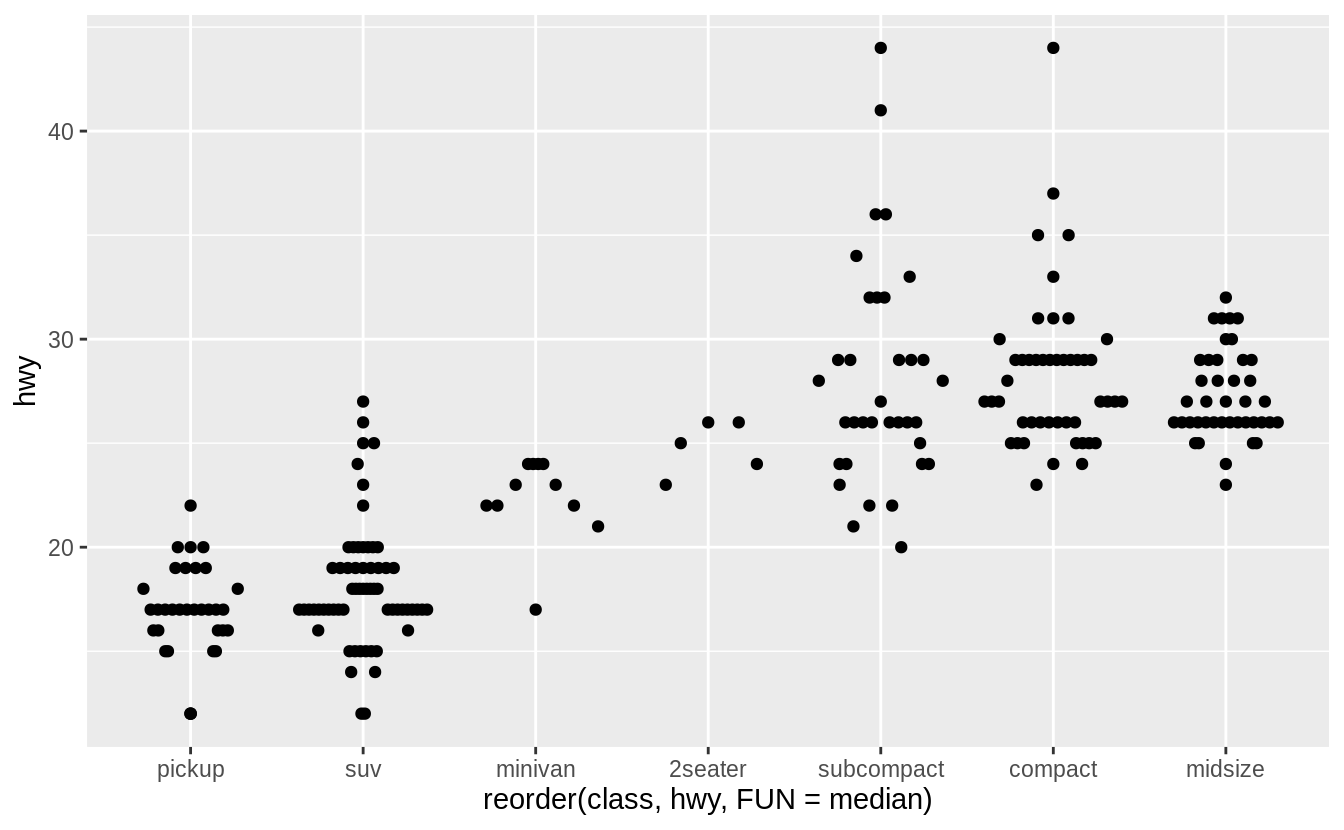
x = **reorder**(class, hwy, FUN = median),

y = hwy

),

method = "frowney"

)



**ggplot**(data = mpg) +

**geom\_quasirandom**(

mapping = **aes**(

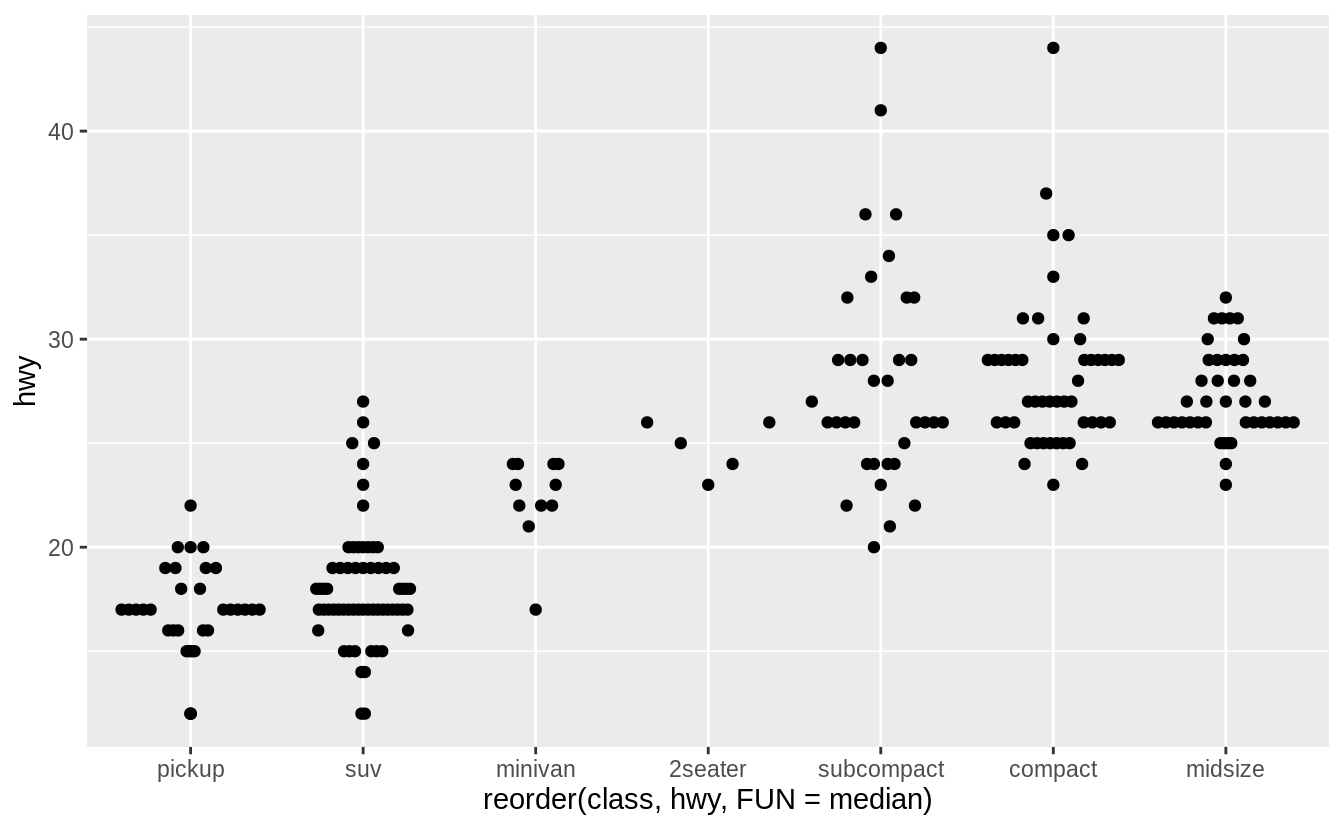
x = **reorder**(class, hwy, FUN = median),

y = hwy

),

method = "smiley"

)



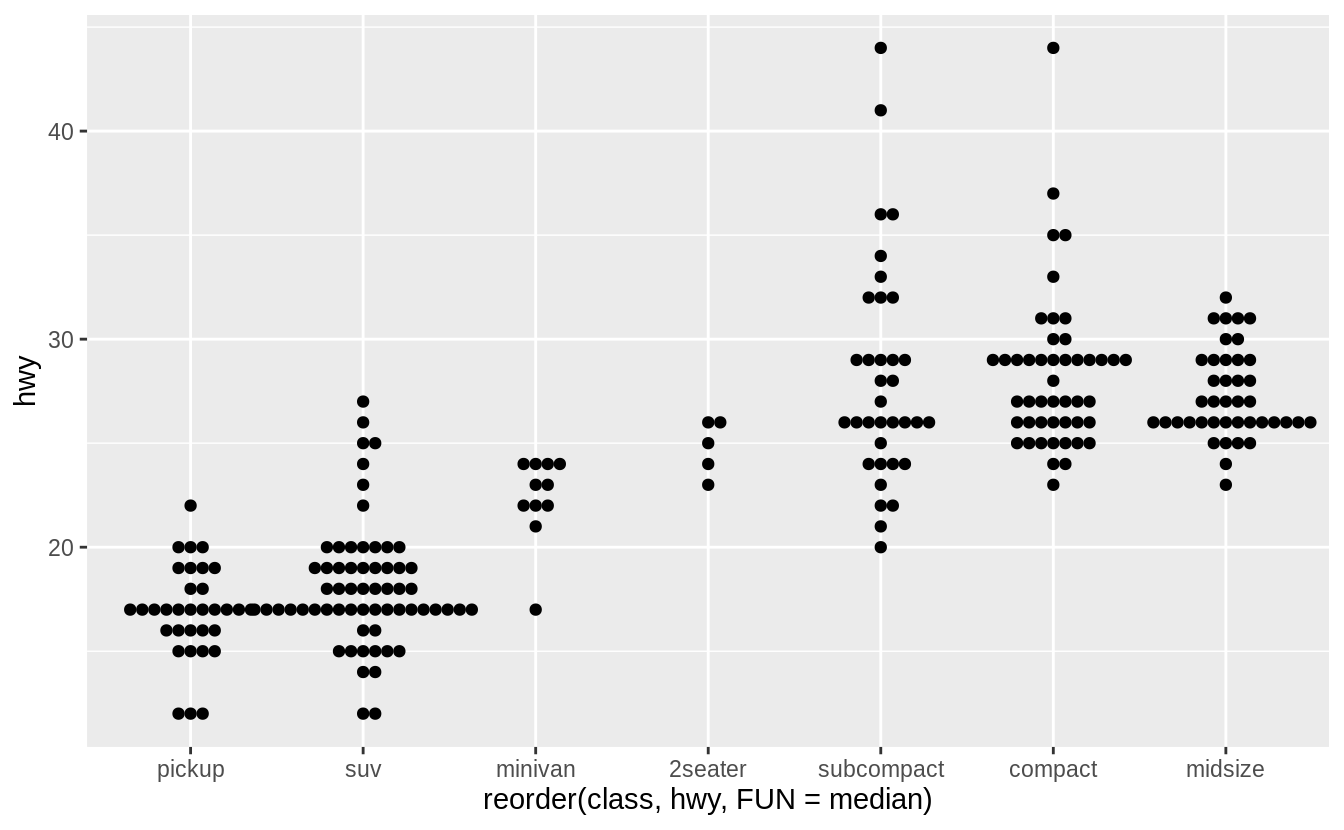
**ggplot**(data = mpg) +

**geom\_beeswarm**(mapping = **aes**(

x = **reorder**(class, hwy, FUN = median),

y = hwy

))

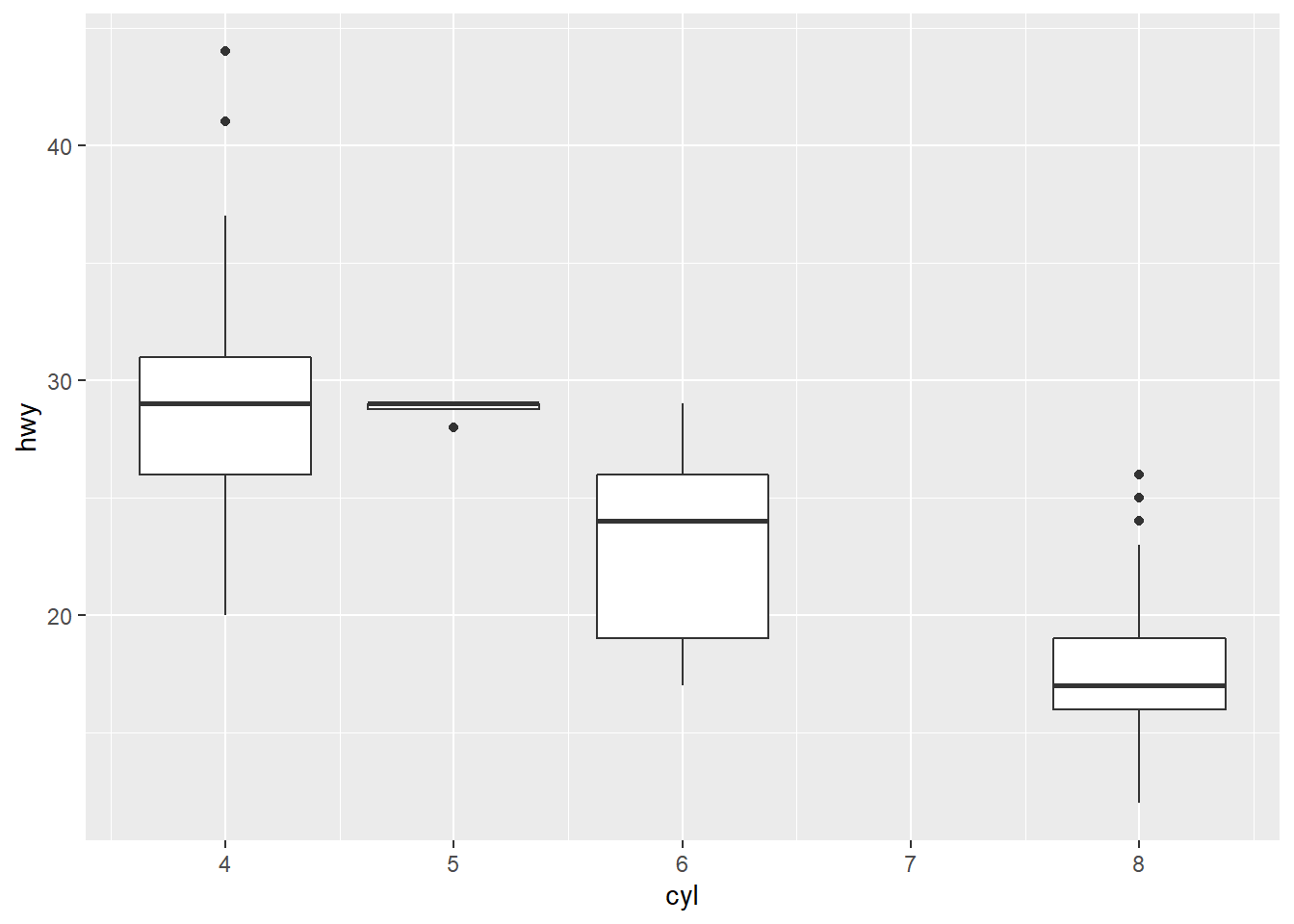


**שאלה 8**

Draw a boxplot of hwy for each value of cyl, without turning cyl into a factor. What extra aesthetic do you need to set?

ggplot(mpg, aes(cyl, hwy, group = cyl)) +

geom\_boxplot()



You simply add a “group = cyl” argument within the overall aes in ggplot()

**שאלה 9**

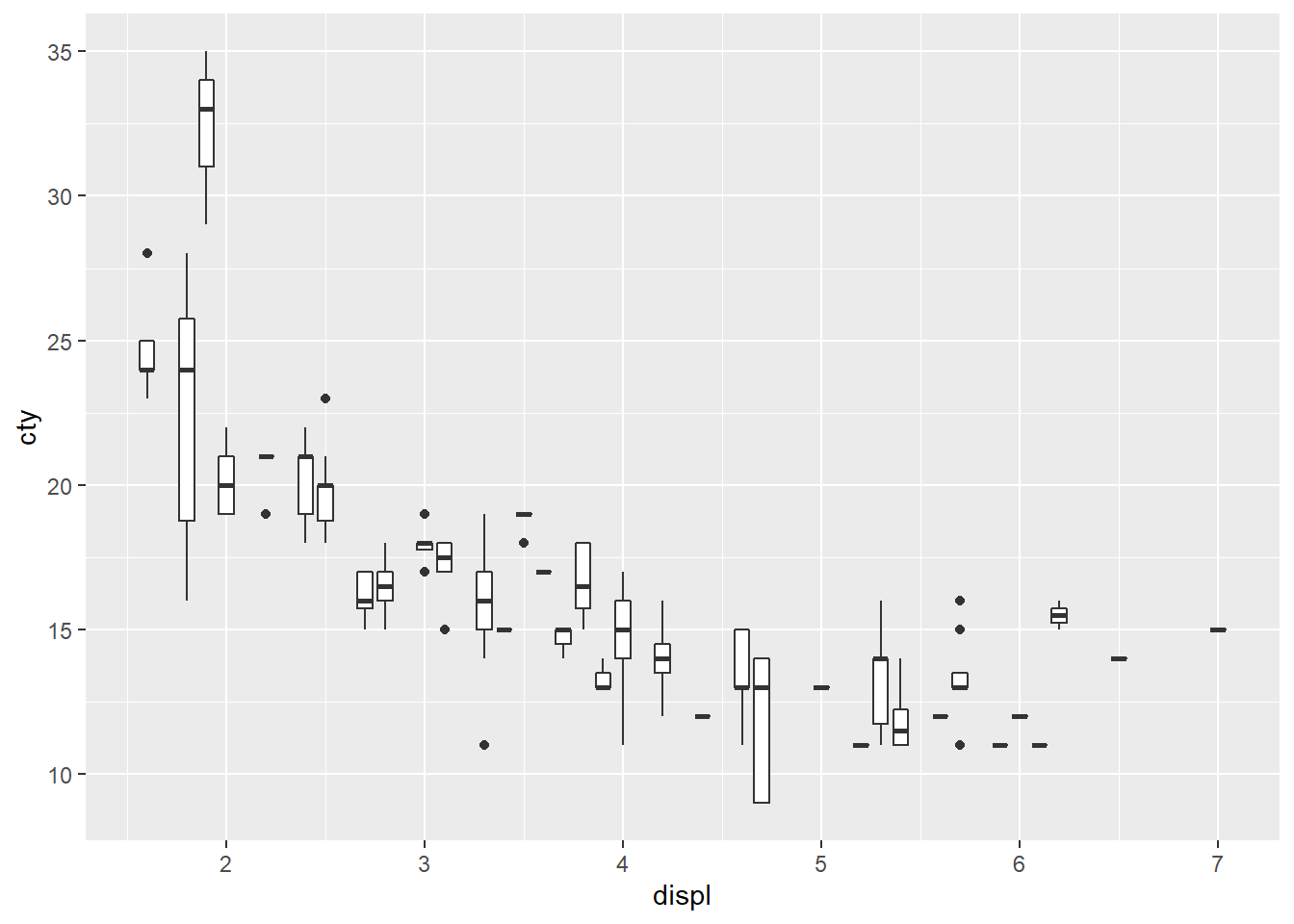
 Modify the following plot so that you get one boxplot per integer value of displ.

ggplot(mpg, aes(displ, cty)) +

geom\_boxplot()

ggplot(mpg, aes(displ, cty)) +

geom\_boxplot(aes(group = displ))



**שאלה 10**

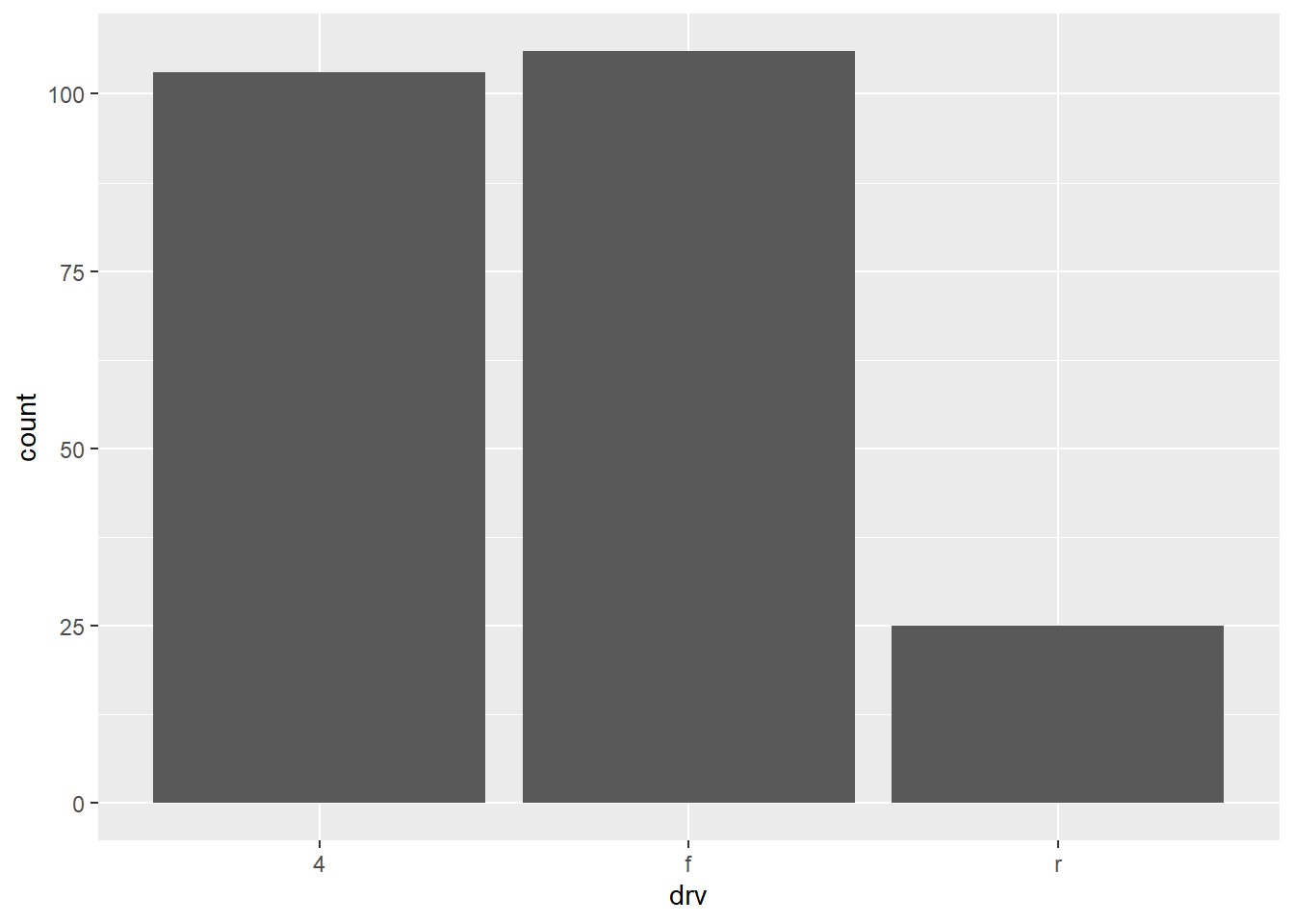
When illustrating the difference between mapping continuous and discrete colours to a line, the discrete example needed aes(group = 1). Why?What happens if that is omitted? What’s the difference between aes(group = 1) and aes(group = 2)? Why?

The important thing (for a line graph with a factor on the horizontal axis) is to manually specify the grouping. By default ggplot2 uses the combination of all categorical variables in the plot to group geoms - that doesn’t work for this plot because you get an individual line for each point. Manually specify group = 1 indicates you want a single line connecting all the points.

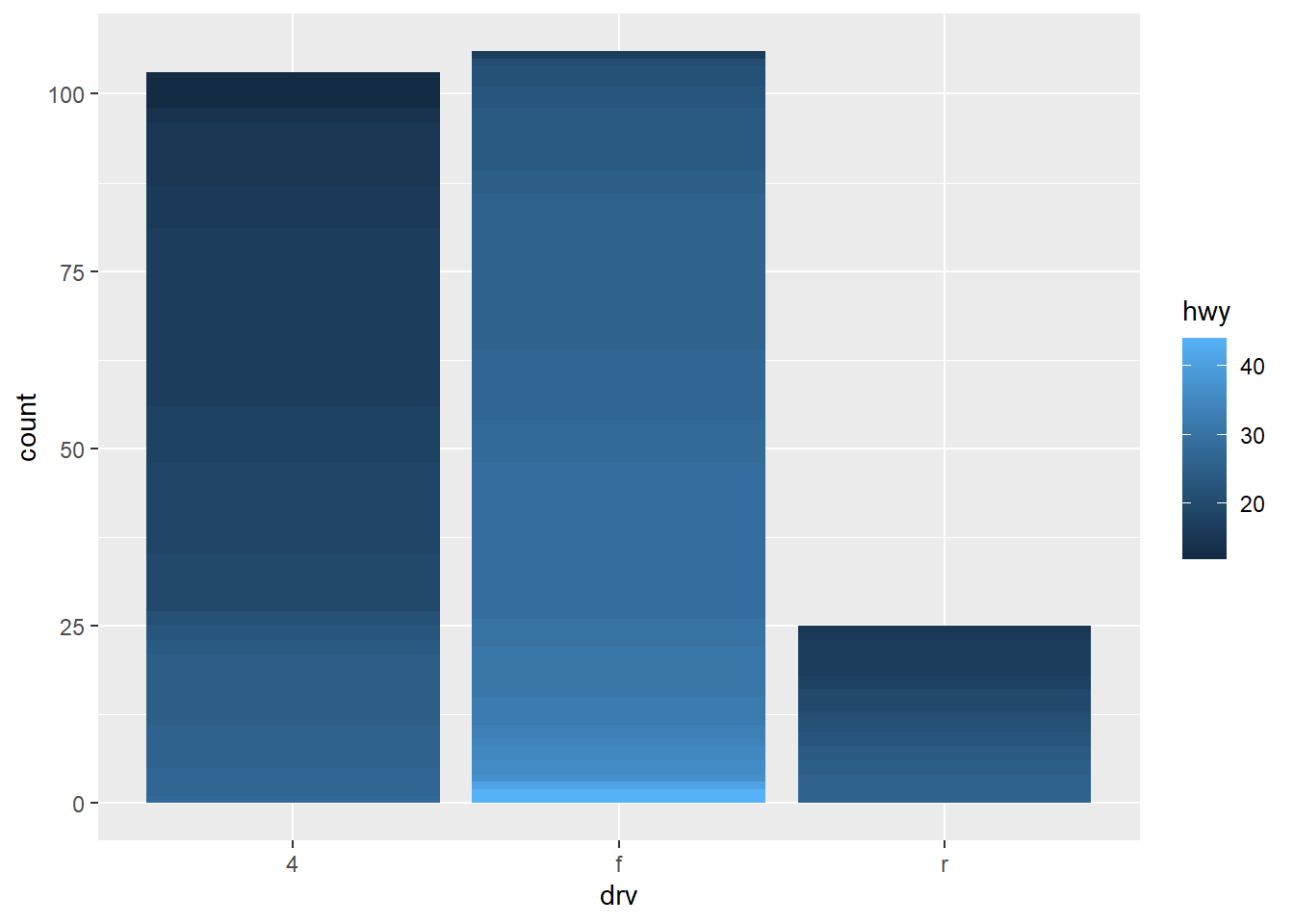
**שאלה 11**

How many bars are in each of the following plots?

ggplot(mpg, aes(drv)) + geom\_bar()



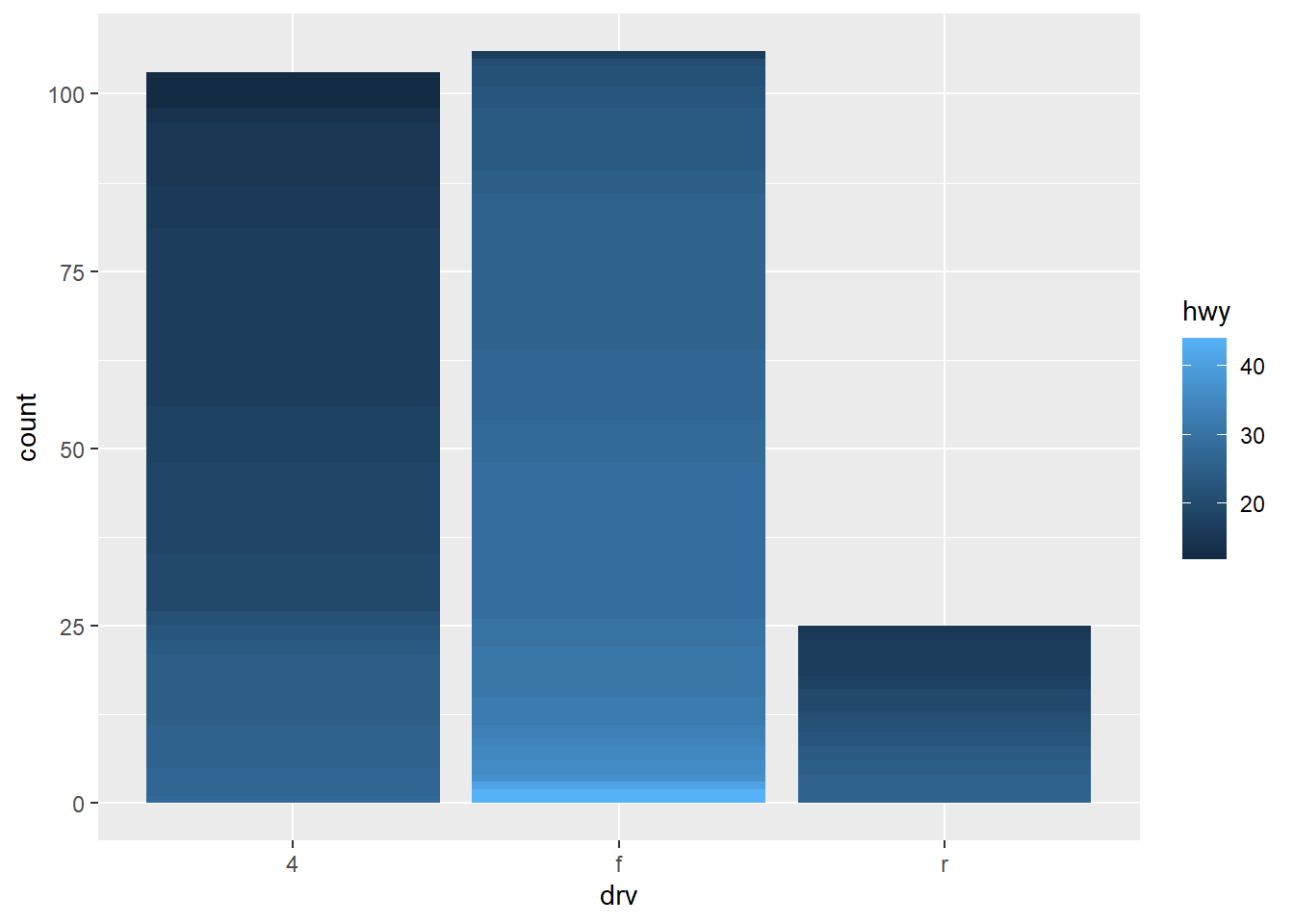
ggplot(mpg, aes(drv, fill = hwy, group = hwy)) + geom\_bar()



**library**(dplyr)

mpg2 <- mpg %>% arrange(hwy) %>% mutate(id = seq\_along(hwy))

ggplot(mpg2, aes(drv, fill = hwy, group = id)) + geom\_bar()



All have 3 bars.