Lecture Assignment 8

Taiki Yamashita

2024-04-29

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.0     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
##
## Attaching package: 'ggstance'
##
##
## The following objects are masked from 'package:ggplot2':
##
##     geom_errorbarh, GeomErrorbarh
```

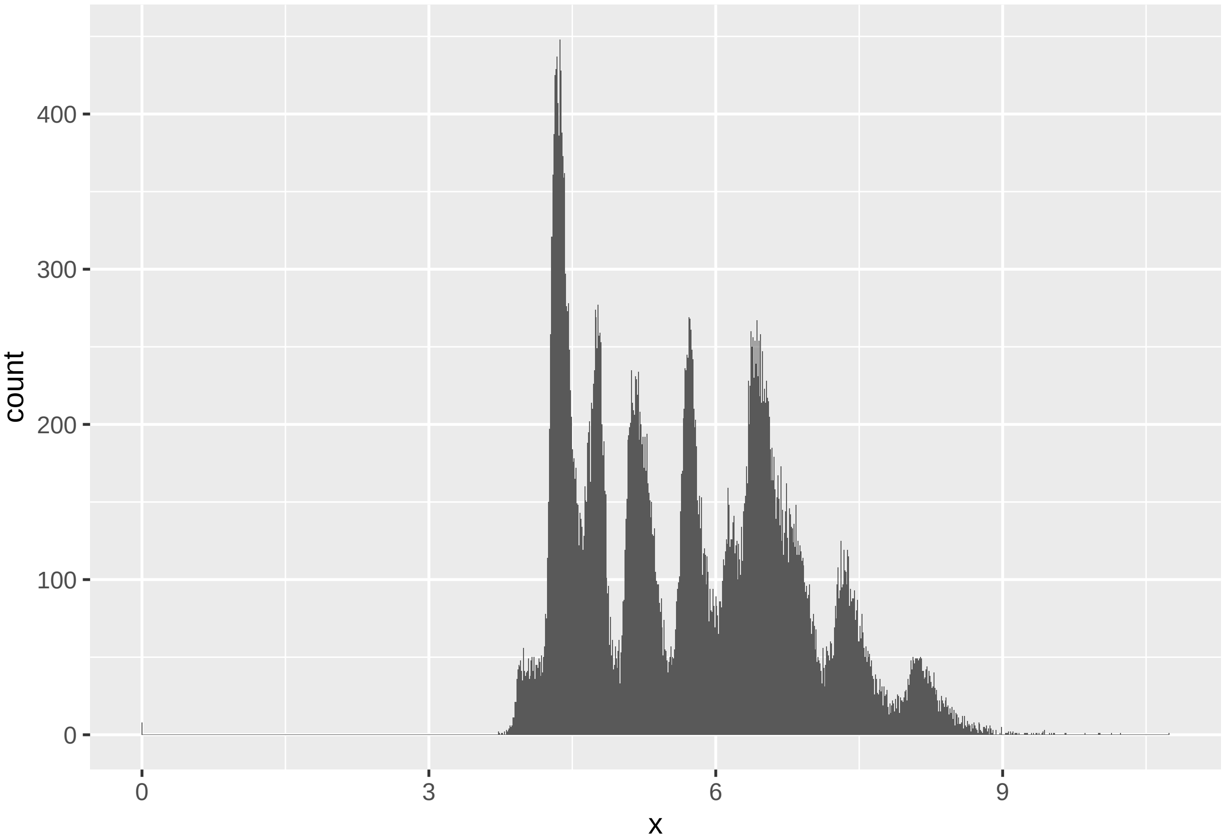
7.3.4 Question 1 ———

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.

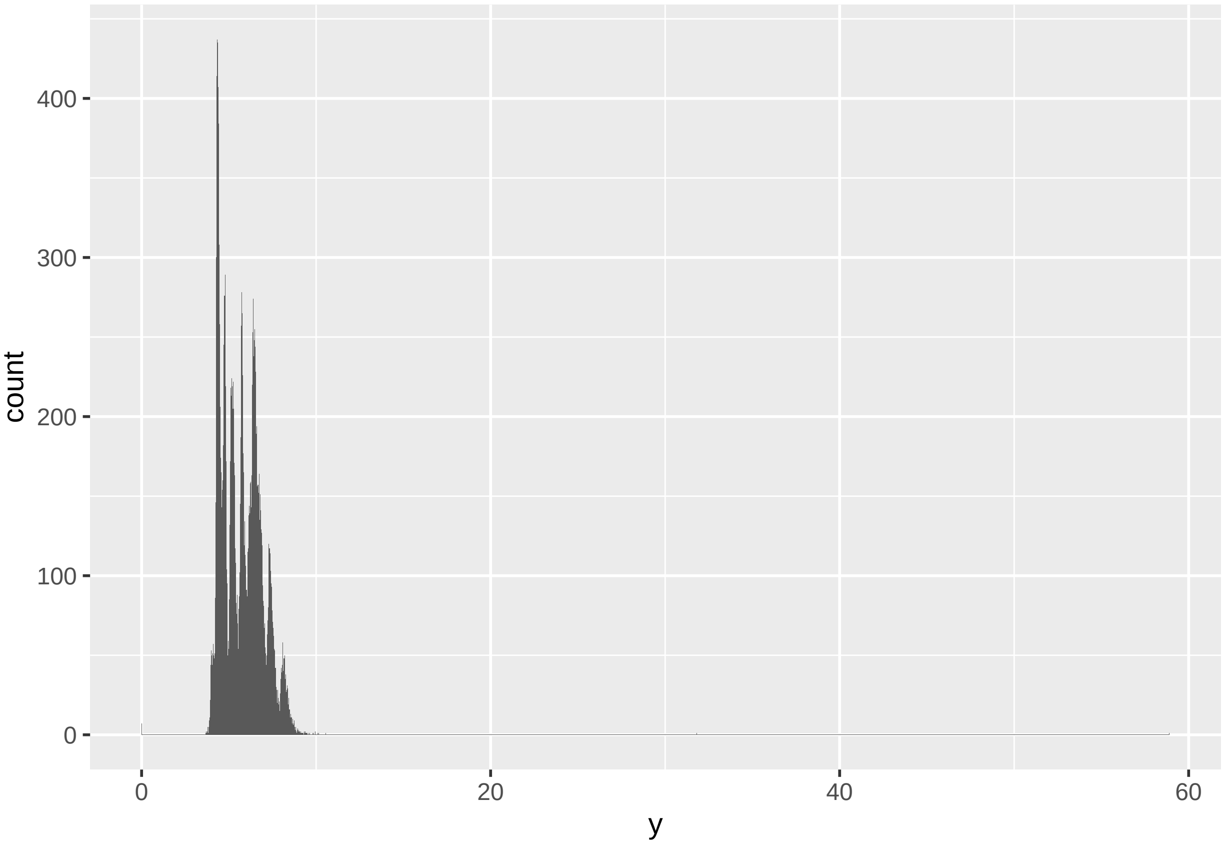
```
# first, we must calculate the summary statistics for these variable and plot their distributions.
summary(select(diamonds, x, y, z))
```

```
##        x                y                z
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 4.710   1st Qu.: 4.720   1st Qu.: 2.910
##  Median : 5.700   Median : 5.710   Median : 3.530
##  Mean   : 5.731   Mean   : 5.735   Mean   : 3.539
##  3rd Qu.: 6.540   3rd Qu.: 6.540   3rd Qu.: 4.040
##  Max.   :10.740   Max.   :58.900   Max.   :31.800
```

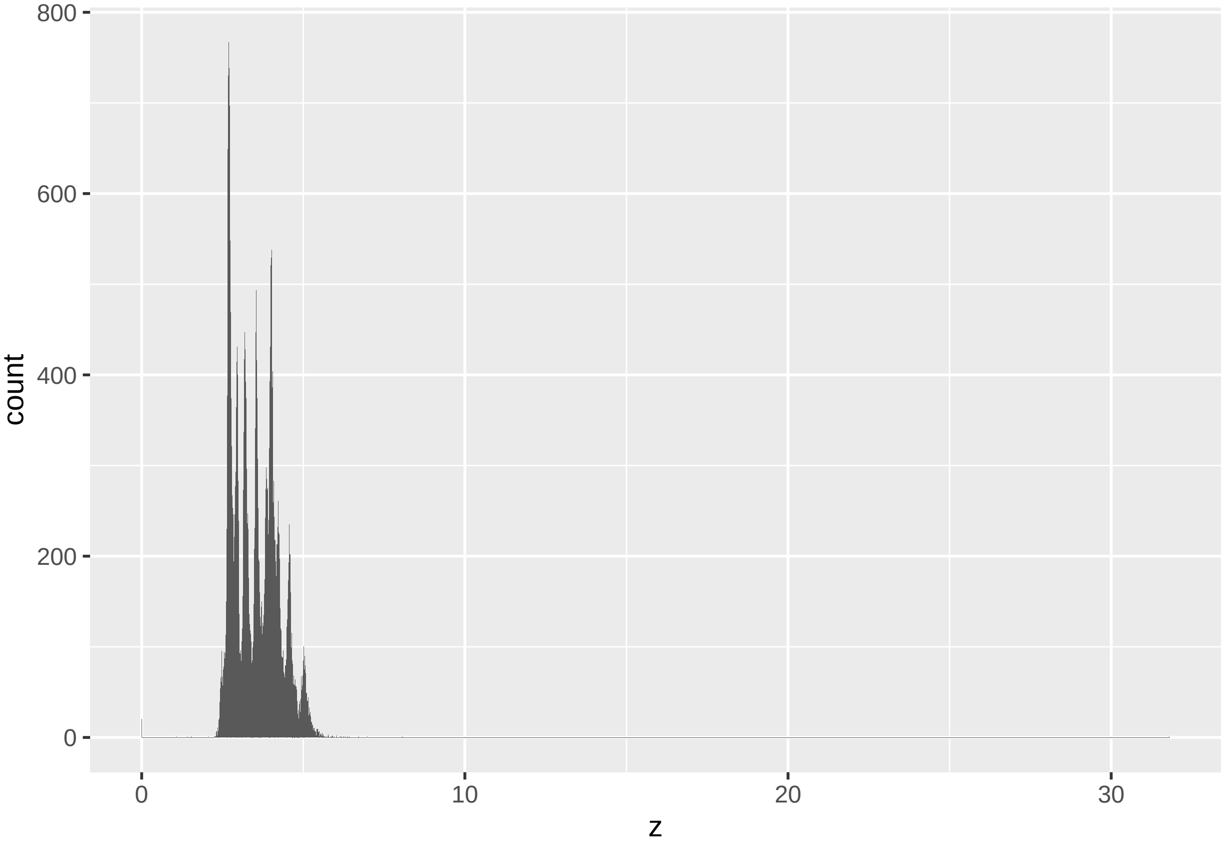
```
# for x
ggplot(diamonds) +
  geom_histogram(mapping = aes(x=x), binwidth = 0.01)
```



```
# for y
ggplot(diamonds) +
  geom_histogram(mapping = aes(x=y), binwidth = 0.01)
```



```r
# for z
ggplot(diamonds) +
  geom_histogram(mapping = aes(x=z), binwidth = 0.01)
```

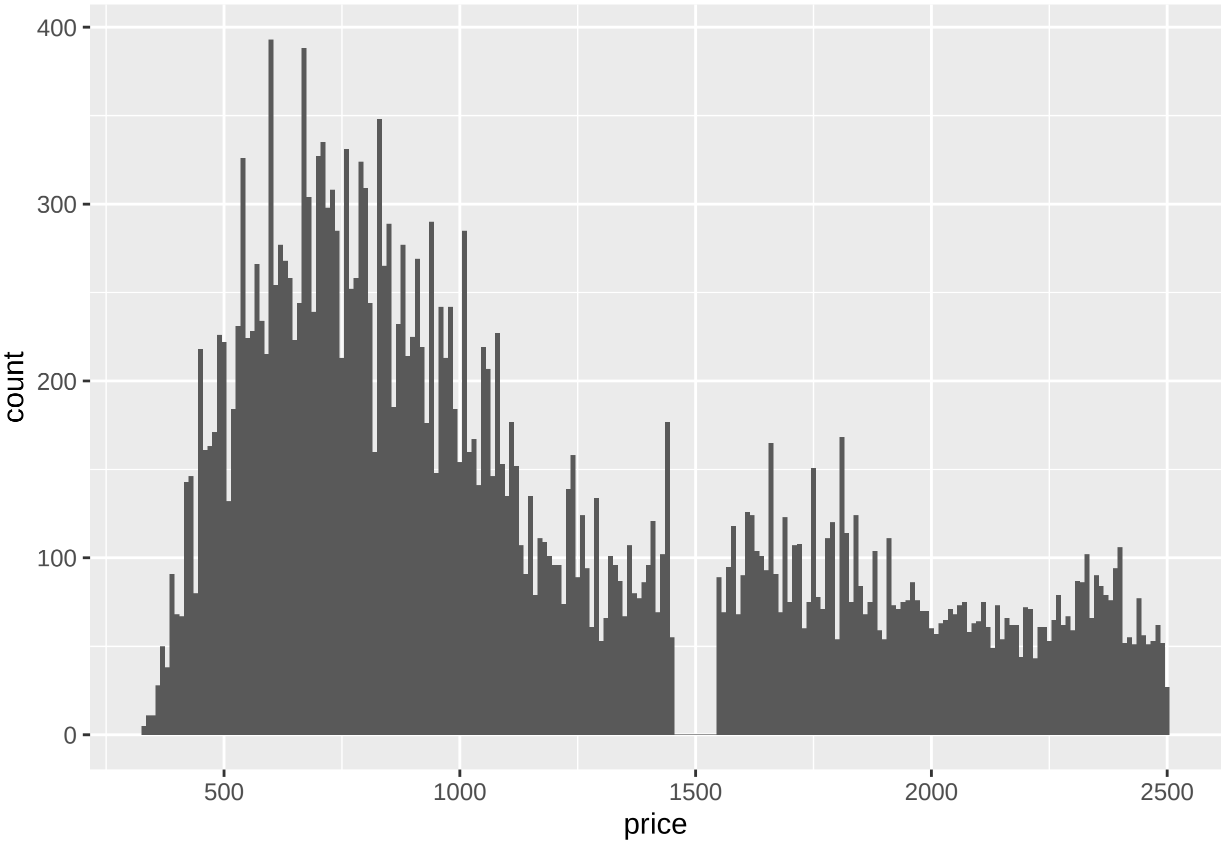


What we can see after exploring each of the distributions with different variables is that x and y are larger than z, all of the distributions are right skewed, there are outliers for each of the graphs, and they are multimodal.

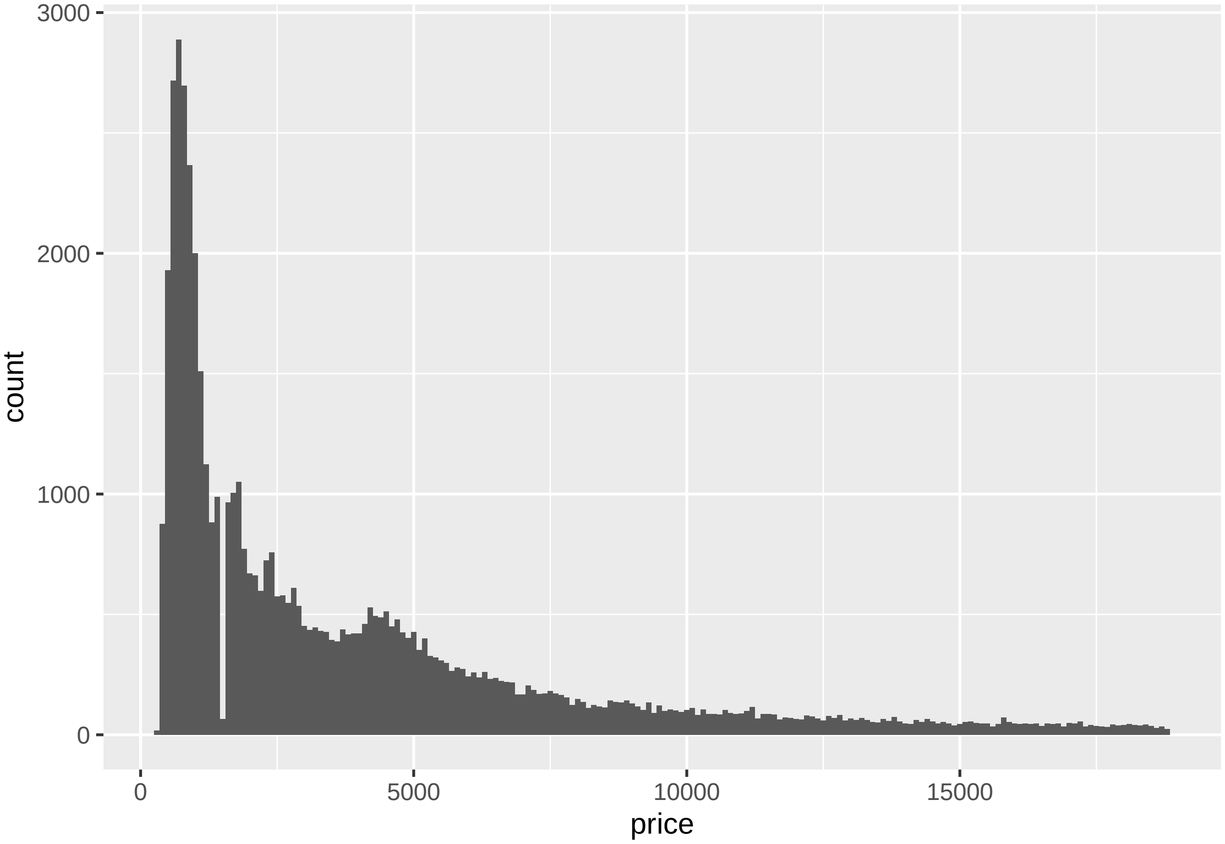
7.3.4 Question 2 ———

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.)

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

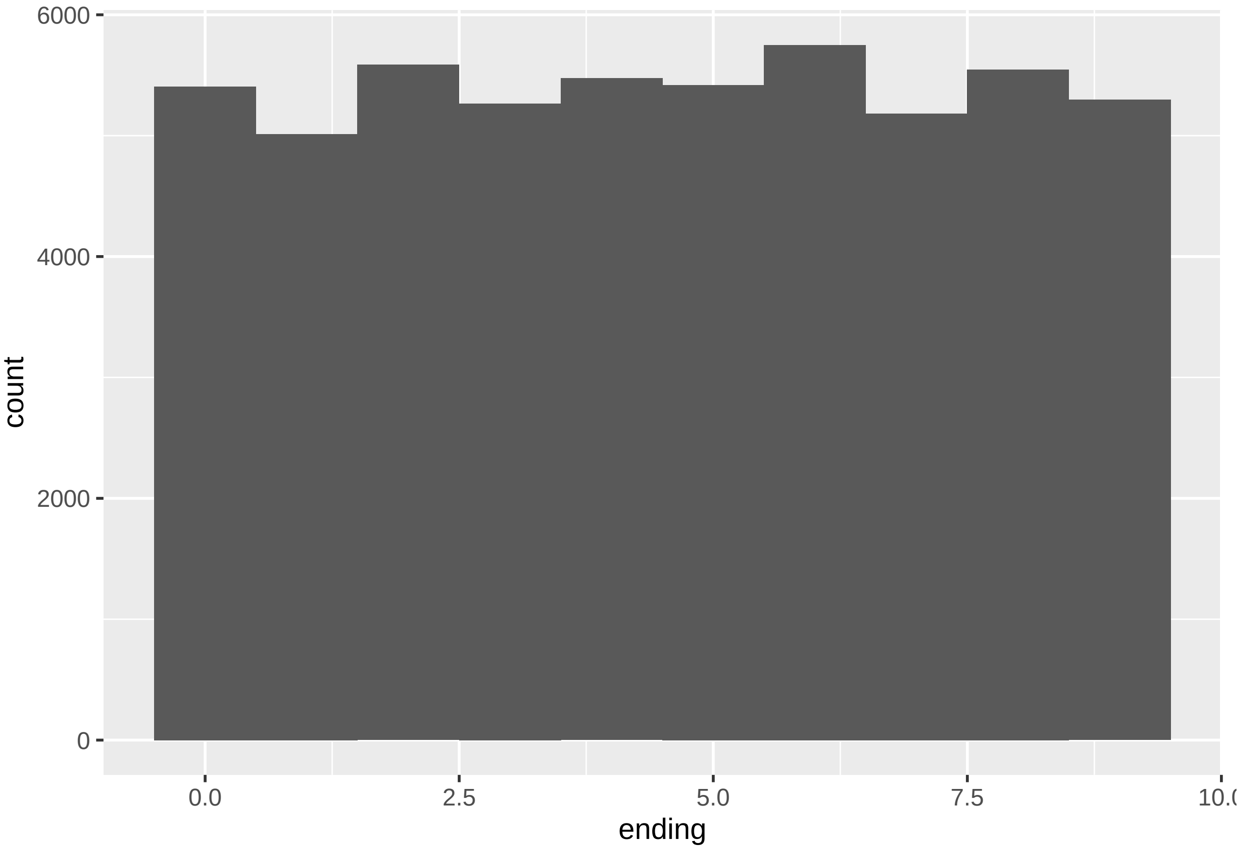


```
ggplot(filter(diamonds), aes(x=price)) +
  geom_histogram(binwidth=100, center=0)
```



The price data has many spikes, but it is hard to tell what each spike corresponds to. The plots do not show much difference in the distribution in the last one or two digits. There are no diamonds with a price of $1500!

```
diamonds %>%
  mutate(ending = price %% 10) %>%
  ggplot(aes(x=ending)) +
  geom_histogram(binwidth=1, center=0)
```



This way it is easier to visualize the difference in the distribution by looking at the last one or two digits specifically.

7.3.4 Question 3

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 diamonds.

```
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
```

Some diamond carat values are being rounded up. I'm assuming there is a premium for a 1carat diamond vs a 0.99 carat diamond beyond the expected increase in price due to a 0.01 carat increase.

If we want to check this, we would have to look at the number of diamonds in each carat range to see if there is a very low number of 0.99 carat diamonds, and a very 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
```

As we can see in the data table, there is a very low amount of 0.99 carat diamonds, but an extremely high amount of 1 carat diamonds.

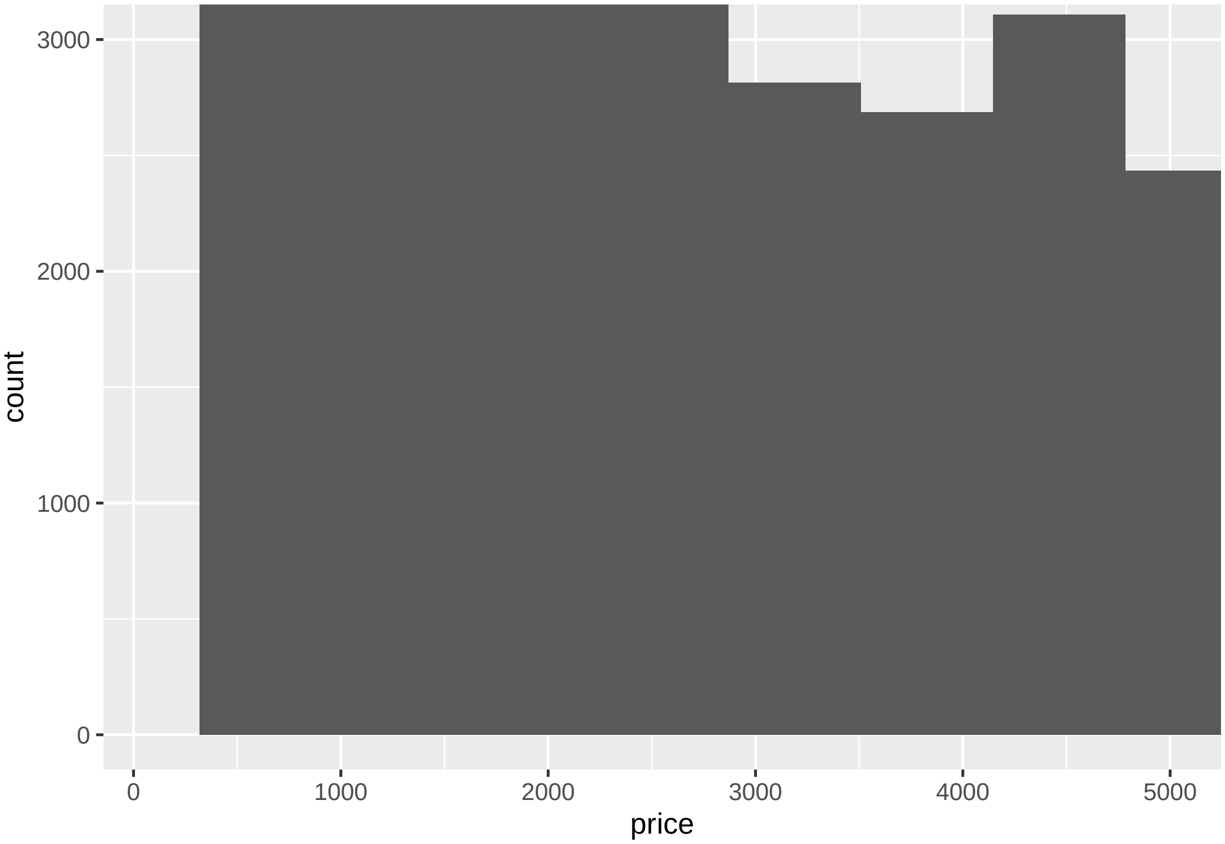
7.3.4 Question 4 ———

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 calculating and drawing the geoms. Since the histogram bins have already been calculated, it is not affected.

```
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‘.
```



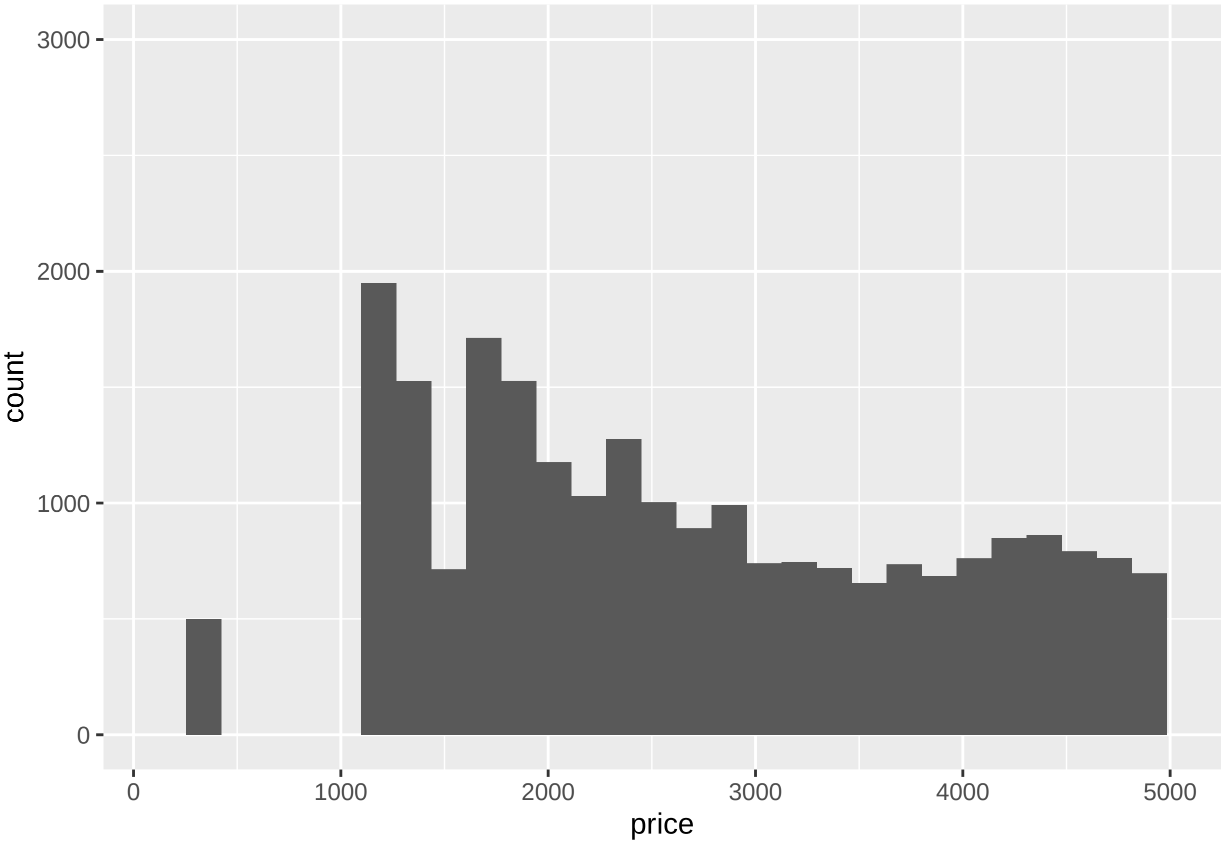
But, since the xlim() and ylim() functions influence actions before the calculation of the stats related to the histogram, any values outside the x- and y- limits are dropped before calculating bin widths and counts. This can influence how the histogram looks.

```r
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 outside the scale range
## (‘stat_bin()‘).
```

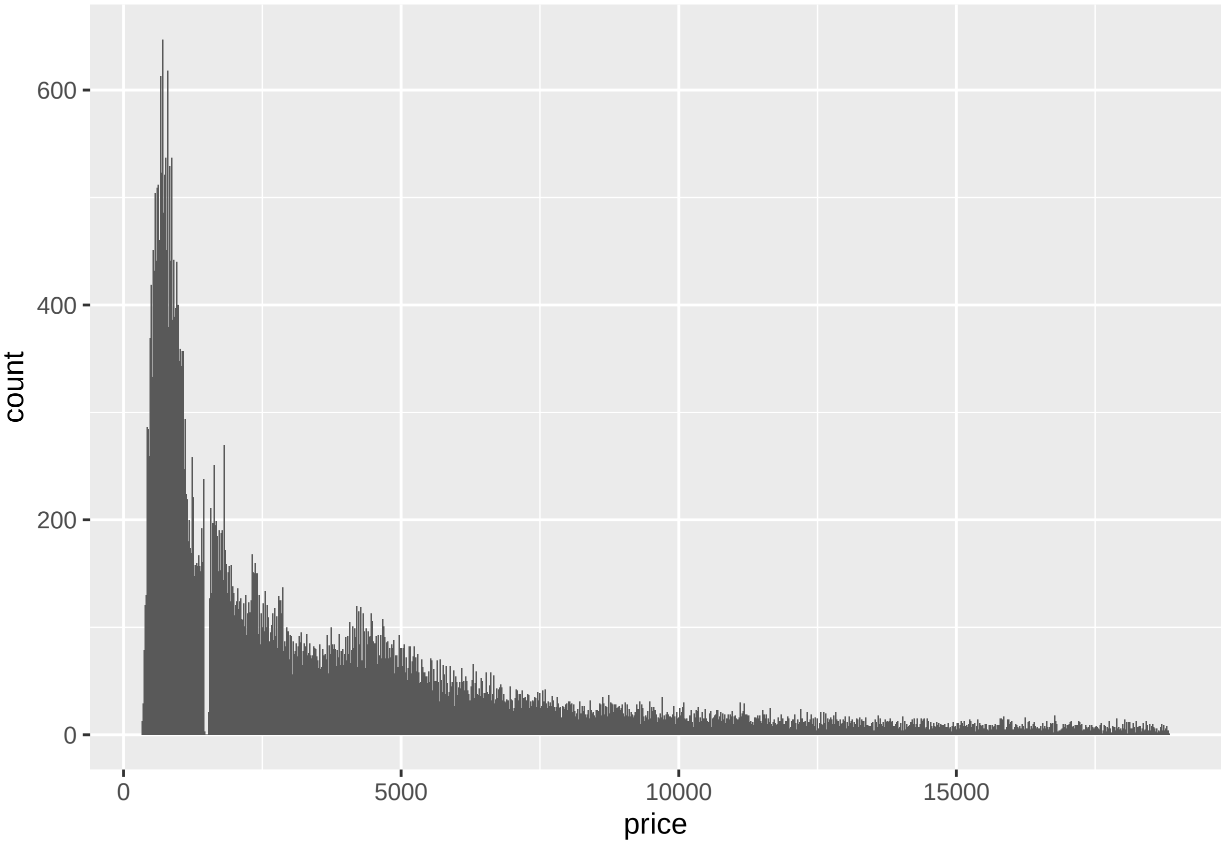
```
## Warning: Removed 6 rows containing missing values or values outside the scale range
## (‘geom_bar()‘).
```



7.4.1 Question 1 ———

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

```
diamonds %>%
  ggplot(aes(price)) +
  geom_histogram(bins = 1000)
```

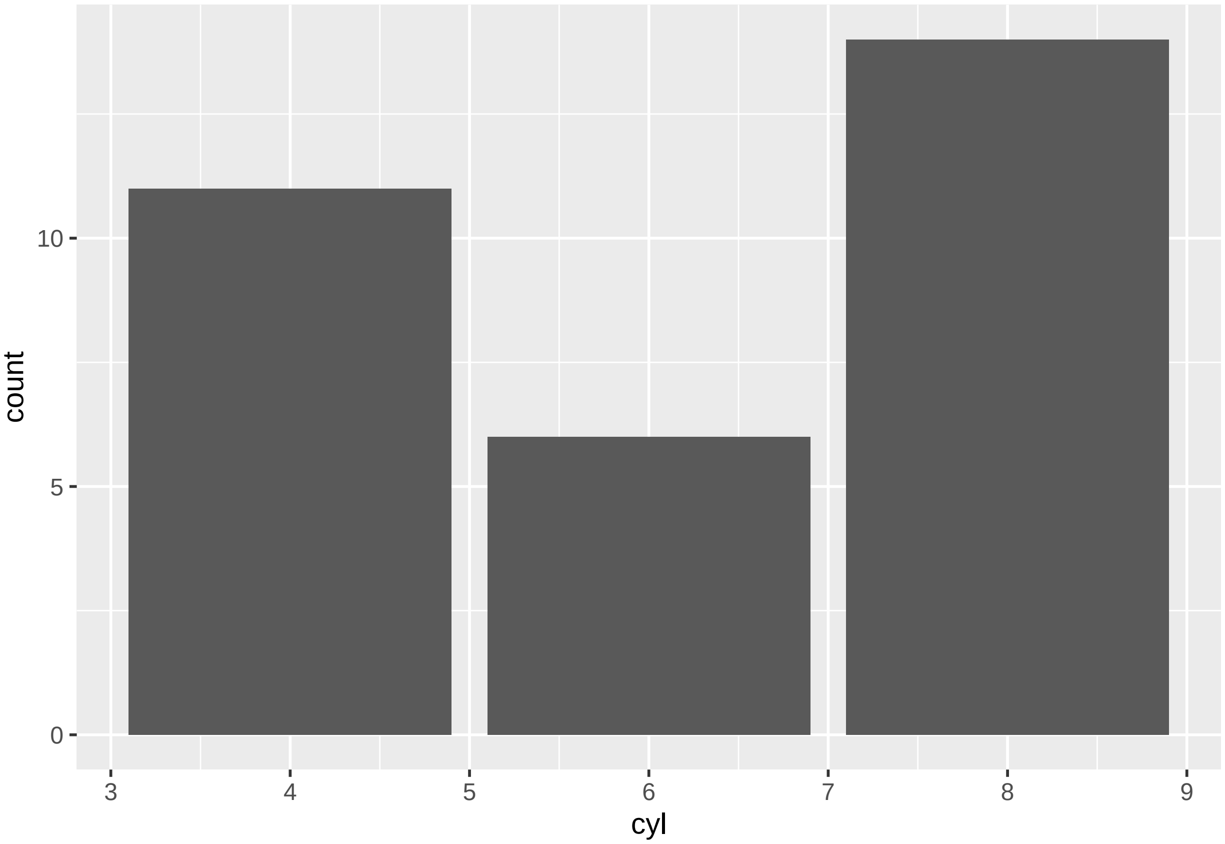


Missing values from a histogram are removed when the number of observations in each bin are calculated. They simply leave a gap in the distribution.

```
mtcars[1, 2] <- NA

mtcars %>%
  ggplot(aes(cyl)) +
  geom_bar()
```

```
## Warning: Removed 1 row containing non-finite outside the scale range
## (‘stat_count()‘).
```



In the geom_bar() function, NA is treated as another category. It removes the 'NA' from the calculation because the numeric value of the NA observations is unknown.

7.4.1 Question 2 ———

What does na.rm = TRUE do in mean() and sum()?

```
mean(c(0, 1, 2, NA), na.rm = TRUE)
```

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

```
sum(c(0, 1, 2, NA), na.rm = TRUE)
```

```
## [1] 3
```

It is the option to remove NA values from the vector prior to calculating the mean and sum.