DATA 100,Week 4 (A)

Exploratory data analysis (1e: Chapter 7 and 2e: Chapter 11)

The 2e. Section 11.1 Introduction is worth a read. In particular, the process for **EDA** is

1. Generate questions about your data.

2. Search for answers by **visualizing, transforming**, and **modelling** your data.

3. Use what you learn to refine your questions and/or generate new questions.

Rinse and repeat. Quote from the book 2e. Section 11.2:

EDA is fundamentally a creative process. And like most creative processes, the key to asking quality questions is to generate a large quantity of questions.

DATA 100,Week 4 (A)

Visualizing Distributions

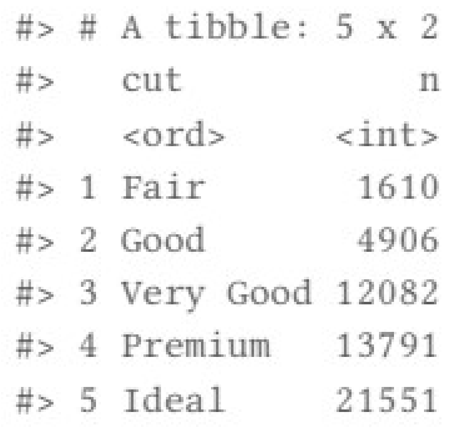
Describe categorical variables (ordinal, nominal)

Numerically

- categories
- frequency, relative frequency

DATA 100,Week 4 (A)

diamonds |> count(cut)

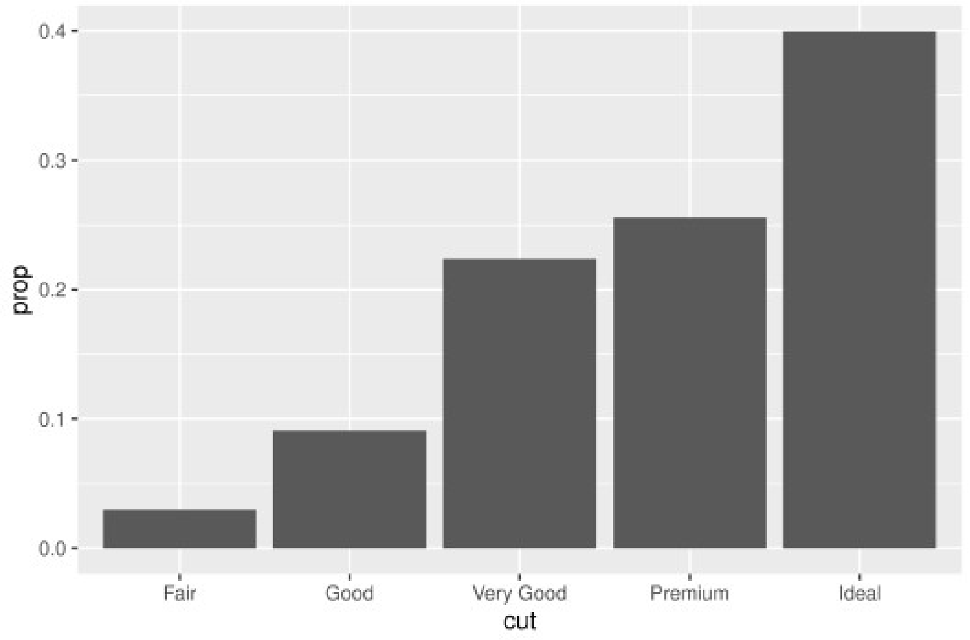


```
#> # A tibble: 5 x 2
#>   cut            n
#>   <ord>      <int>
#> 1 Fair        1610
#> 2 Good        4906
#> 3 Very Good 12082
#> 4 Premium    13791
#> 5 Ideal      21551
```

Graphical method
• Bar chart (geom_bar)
• The categories are represented by indvidual bars
• The height of each bar is either frequency or relative frequency
• after_stat(prop): asks the function to compute *proportion*, or relative frequency and plot it

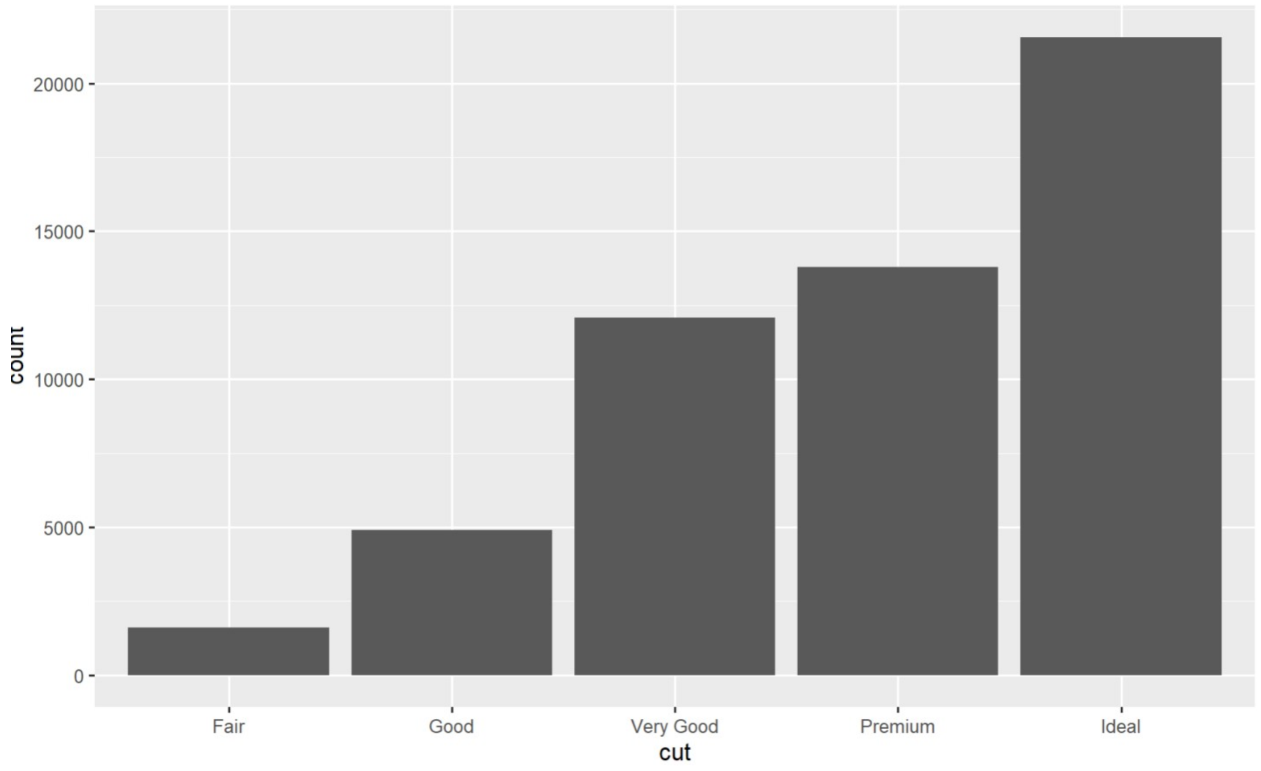
DATA 100,Week 4 (A)

```
ggplot(data = diamonds) +
geom_bar(mapping = aes(x = cut, y = after_stat(prop), group = 1))

 # geom_bar(mapping = aes(x = cut))
```



DATA 100,Week 4 (A)

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



DATA 100,Week 4 (A)

Describe continuous variables (numerical, date-time, etc)

Numerical methods

- central tendency measures (mean, median, mode, etc)
- spread (range, IQR, outliers, variance, standard deviation, etc)

DATA 100,Week 4 (A)

Graphical methods

Histogram (geom_histogram, geom_freqpoly)

• Summarize the data with a frequency table

• Divide the number line into intervals (bins)

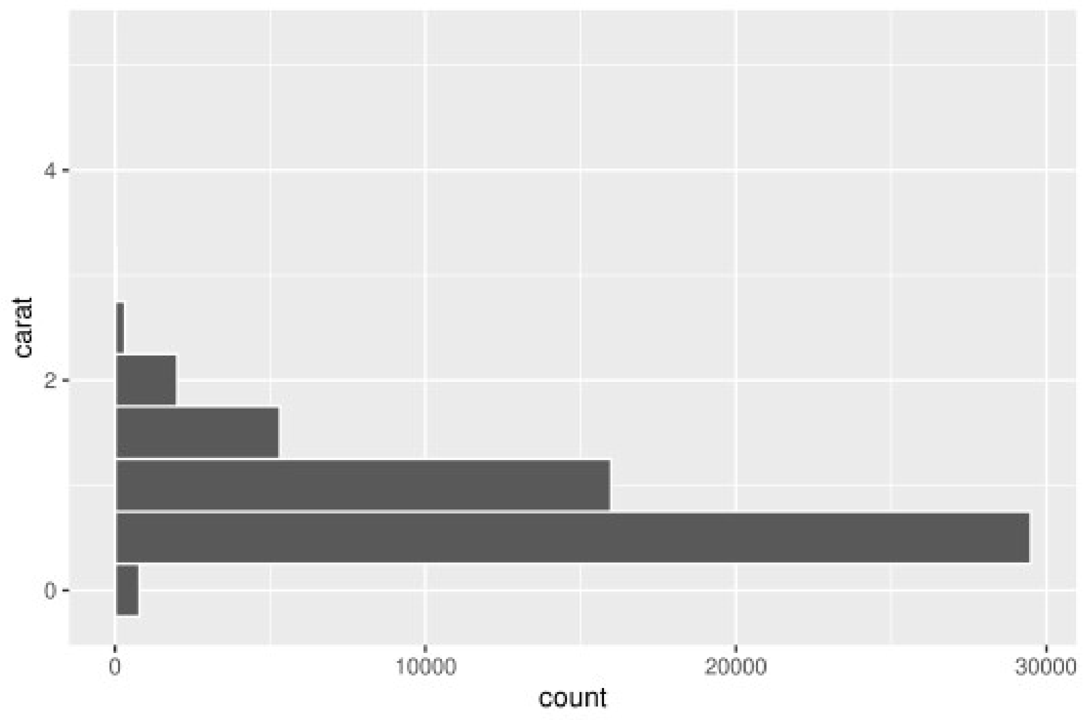
– binwidth below indicates the length of the small intervals

– represented by the width of the bars

• Count the number of objects within each interval

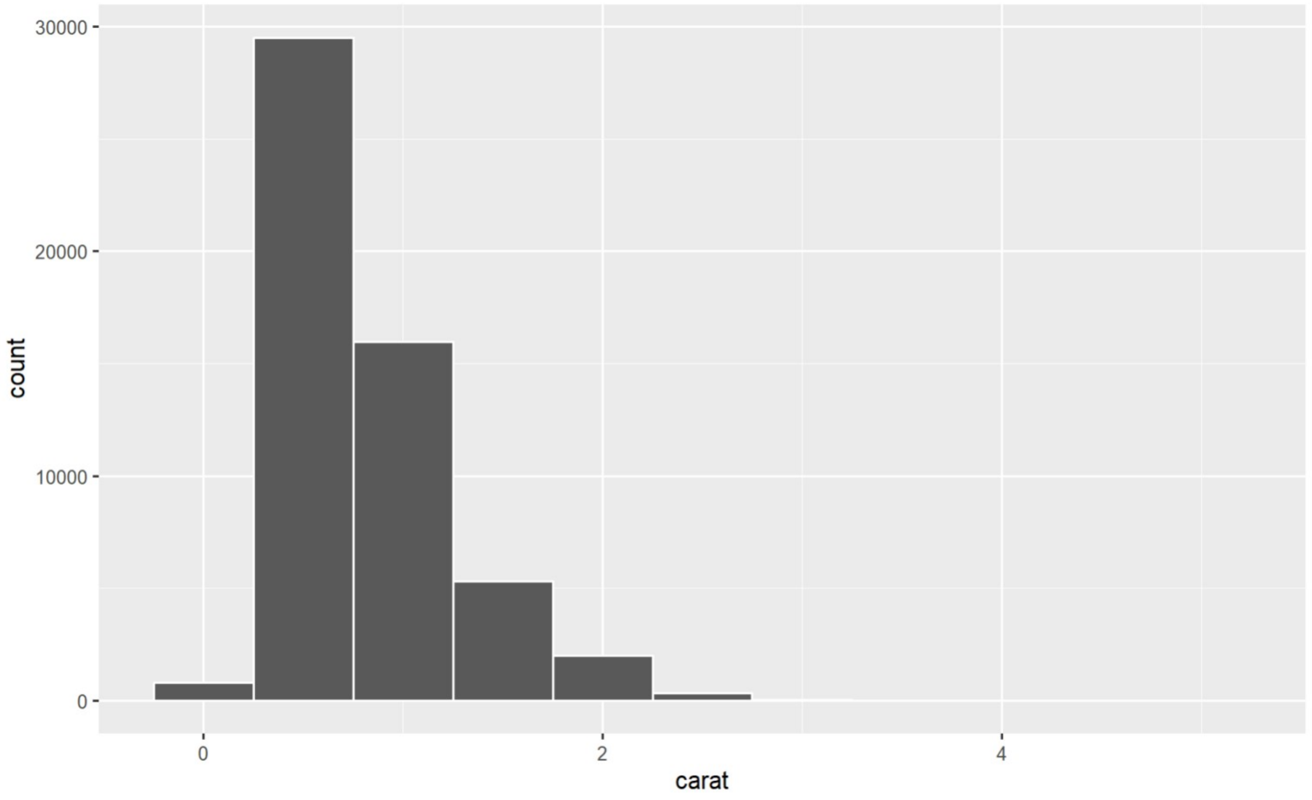
DATA 100,Week 4 (A)

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



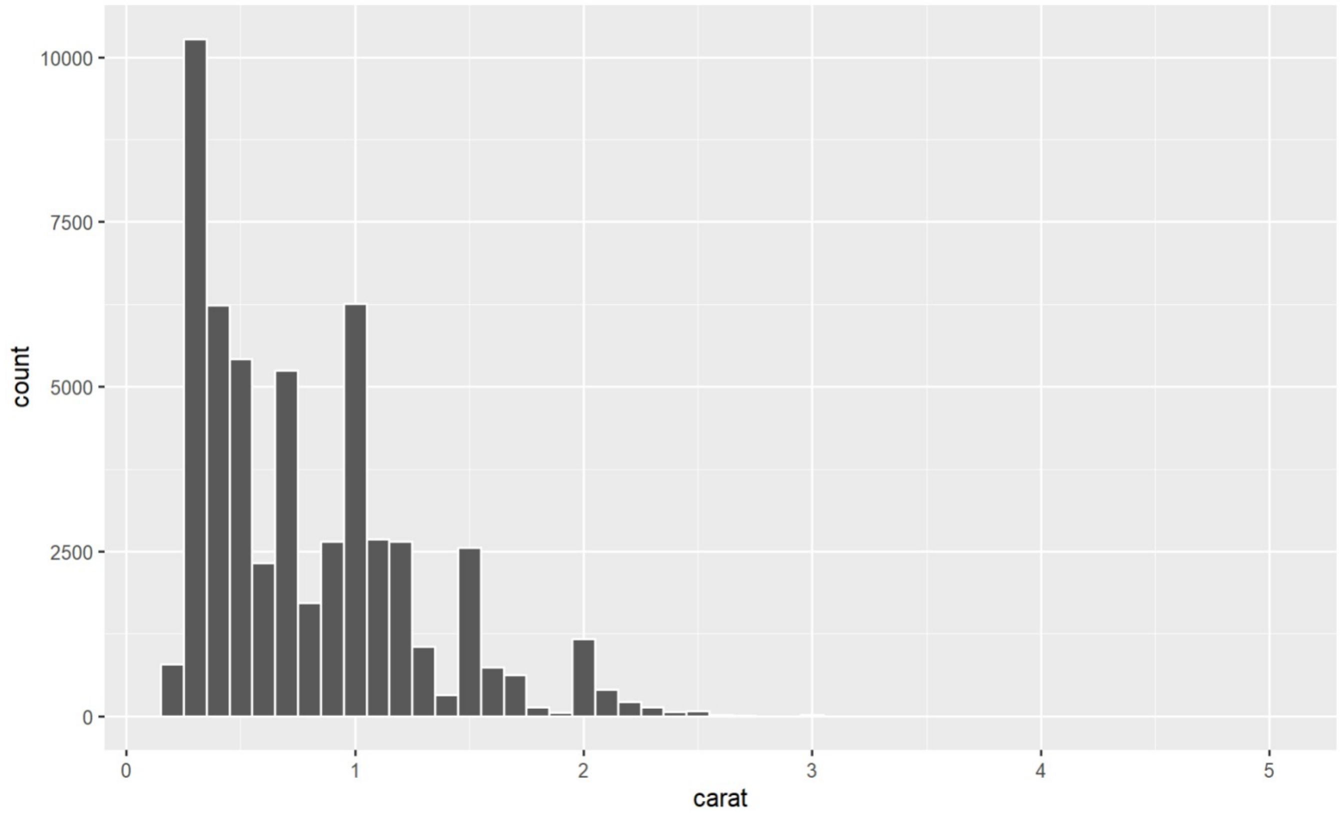
DATA 100,Week 4 (A)

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



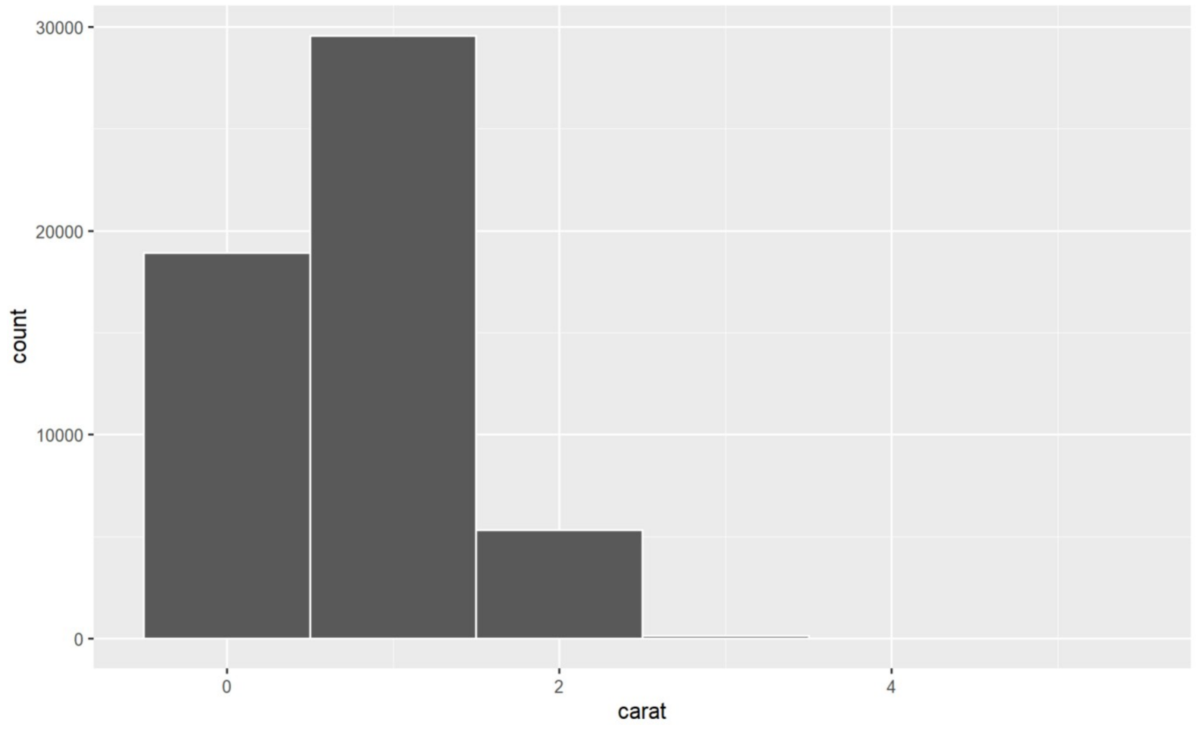
DATA 100,Week 4 (A)

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



DATA 100,Week 4 (A)

```
ggplot(data=diamonds)+
geom_histogram(mapping = aes(x = carat), binwidth = 1, color="white")
```



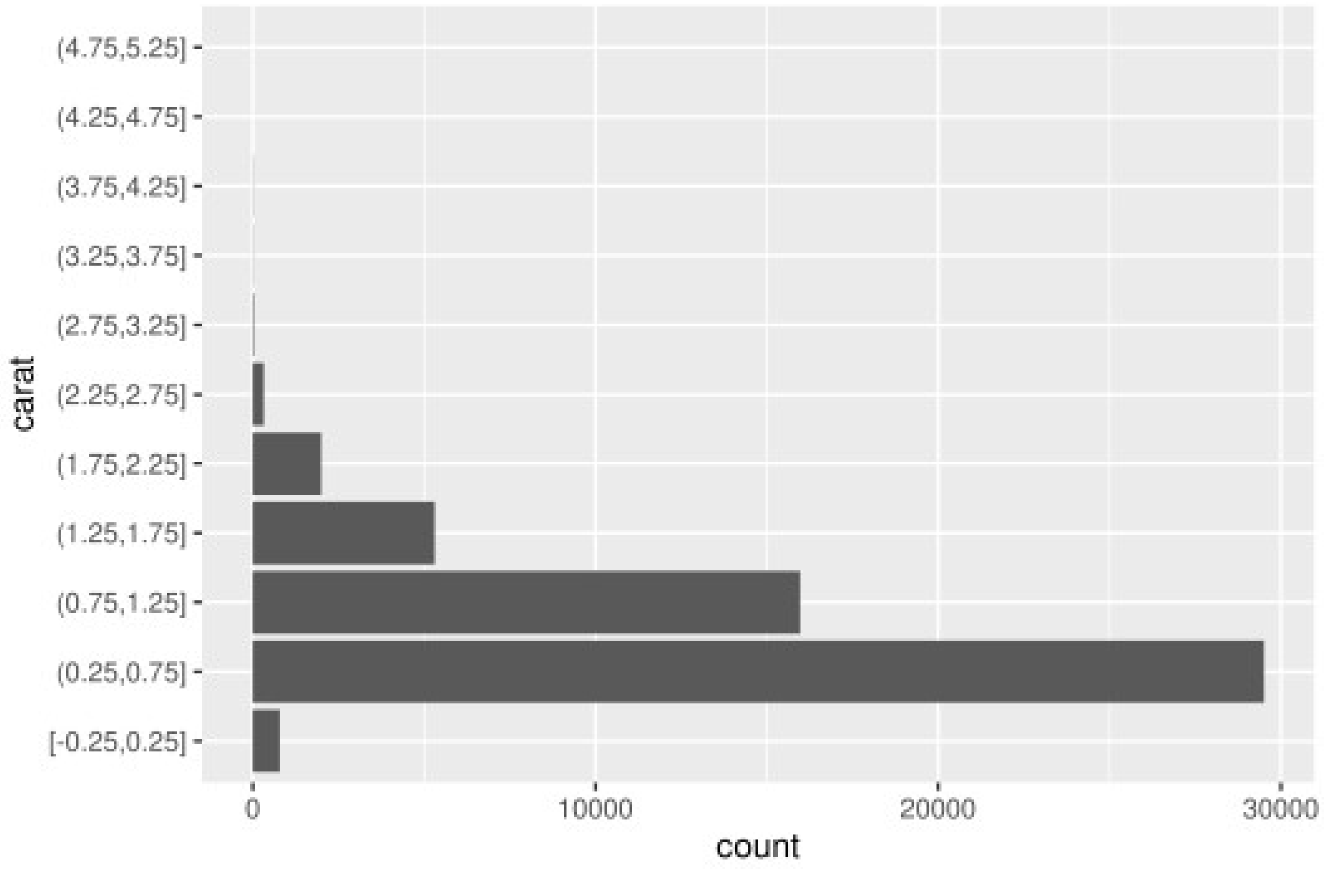
DATA 100,Week 4 (A)

Choosing binwidth = 0.5 corresponds to

- generating a table using count() and cut_width()
- plot a bar chart using the resulting table

```
diamonds |>
count(cut_width(carat, 0.5)) |>
rename(carat = `cut_width(carat, 0.5)`, count = n) |>
ggplot() +
geom_bar(mapping = aes(x = carat, y = count), stat="identity") + coord_flip()
```

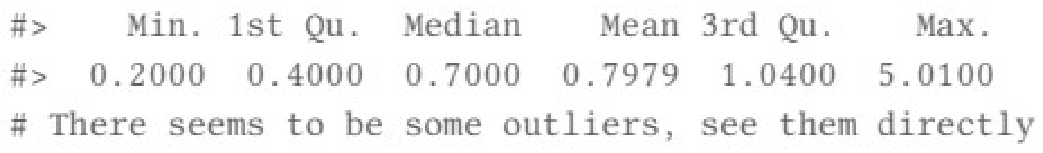
DATA 100,Week 4 (A)



DATA 100,Week 4 (A)

Selecting binwidth can reveal finer details

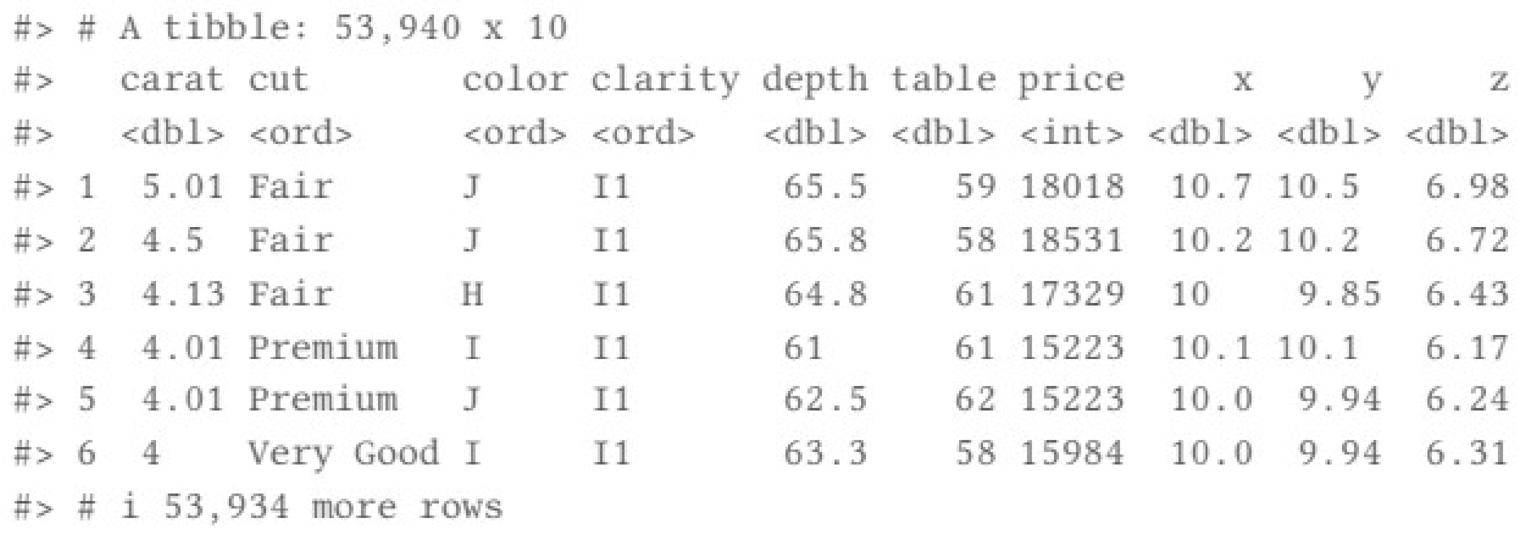
```
# Get a rough idea of the distribution
summary(diamonds$carat)
```



```
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  0.2000  0.4000  0.7000  0.7979  1.0400  5.0100
# There seems to be some outliers, see them directly
```

DATA 100,Week 4 (A)

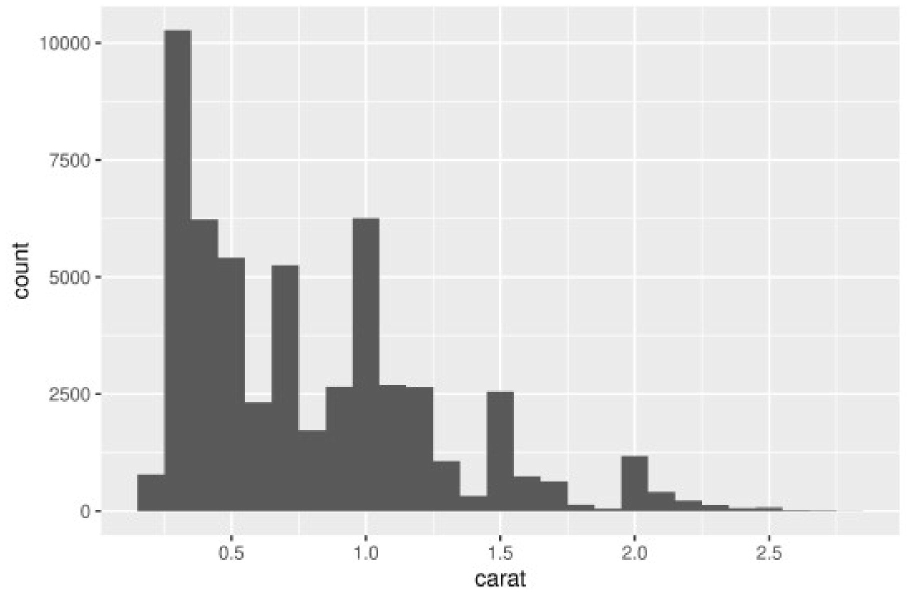
```
diamonds |>
arrange(desc(carat))
```



```
#> # A tibble: 53,940 x 10
#>   carat cut       color clarity depth table price     x     y     z
#>   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
#> 1  5.01 Fair      J     I1       65.5    59 18018  10.7 10.5   6.98
#> 2  4.5  Fair      J     I1       65.8    58 18531  10.2 10.2   6.72
#> 3  4.13 Fair      H     I1       64.8    61 17329  10     9.85  6.43
#> 4  4.01 Premium   I     I1       61      61 15223  10.1 10.1   6.17
#> 5  4.01 Premium   J     I1       62.5    62 15223  10.0  9.94  6.24
#> 6  4    Very Good I     I1       63.3    58 15984  10.0  9.94  6.31
#> # i 53,934 more rows
```

DATA 100,Week 4 (A)

```
# Now plot the majority of diamonds in finer details

smaller <- diamonds |> filter(carat < 3)

ggplot(data = smaller, mapping = aes(x = carat)) +
geom_histogram(binwidth=0.1) + scale_x_continuous(breaks = seq(0,3,0.5))
```



DATA 100,Week 4 (A)

• scale_x_continuous puts the scale labels onto the axis

– breaks defines where the ticks should be

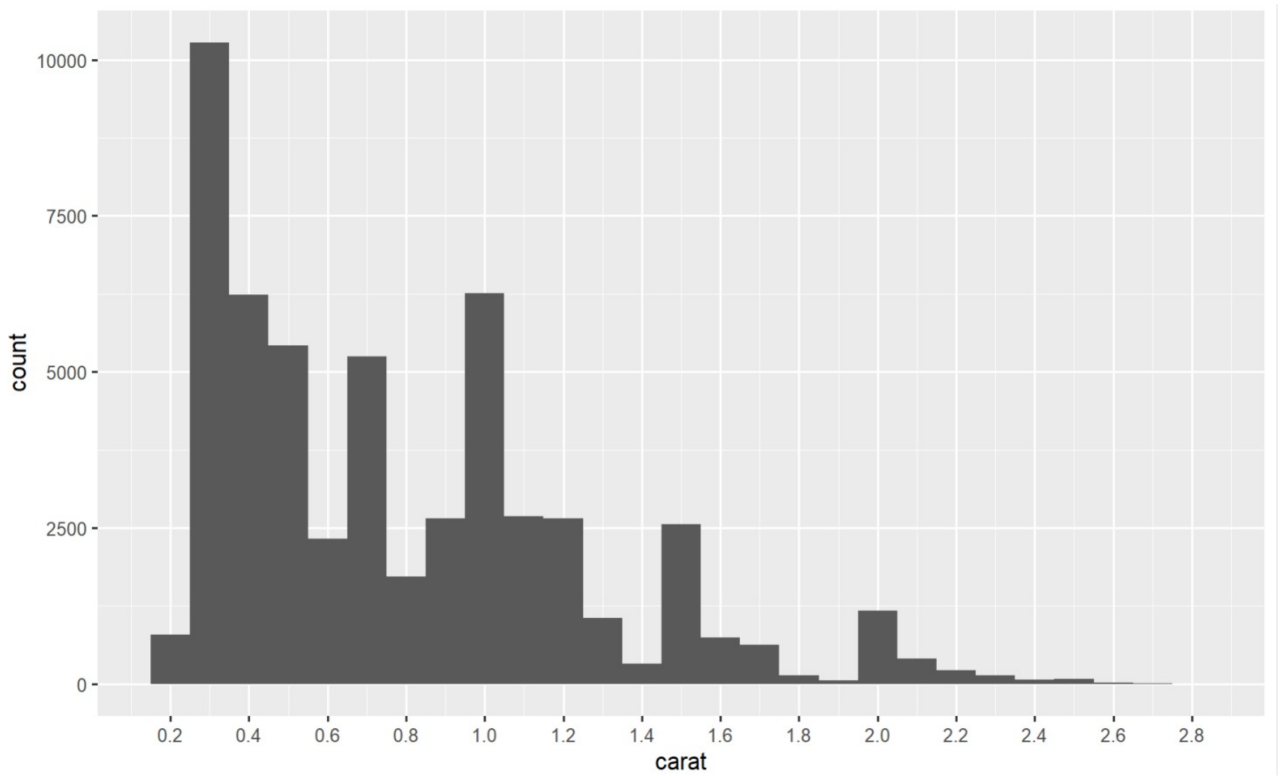
– seq(0,3,0.1) generates a numeric vector, from 0 to 3 in steps of 0.1.

Namely, it is the same as the numerical vector

(0, 0.1, 0.2, 0.3, ..., 2.8, 2.9, 3)

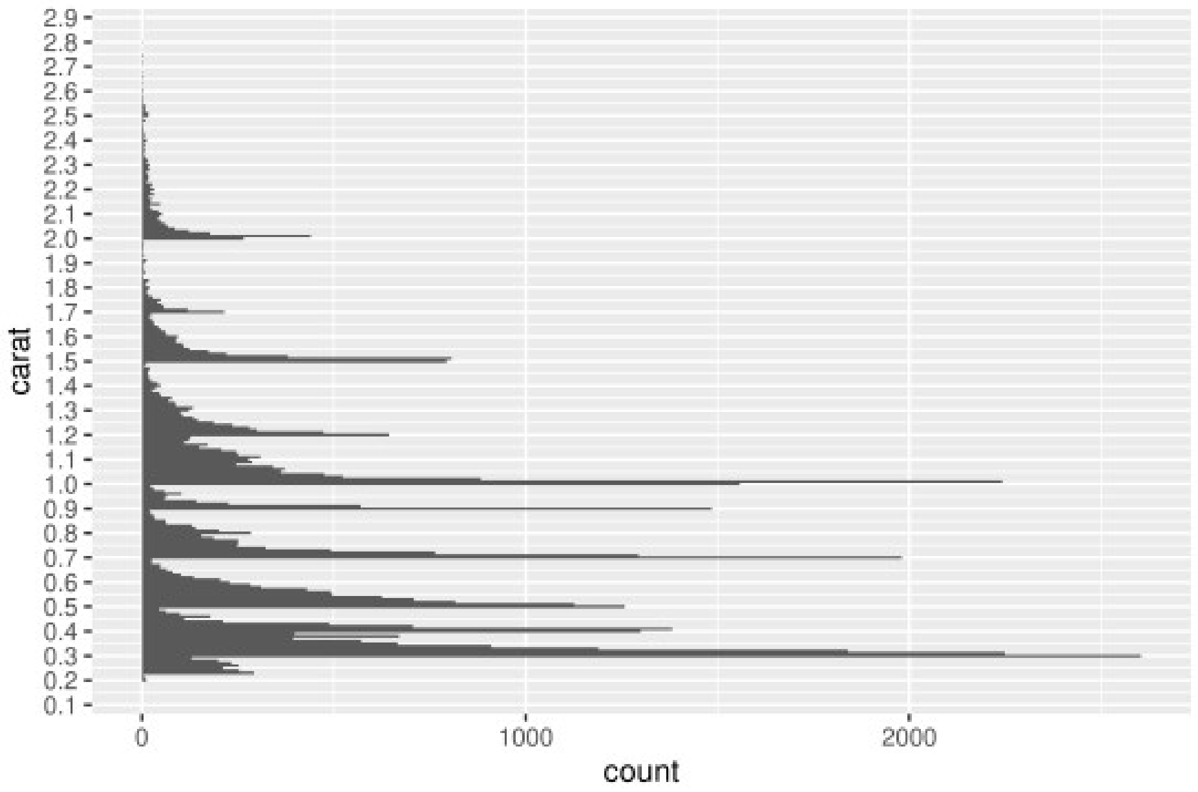
DATA 100,Week 4 (A)

```
ggplot(data = smaller, mapping = aes(x = carat)) +
geom_histogram(binwidth=0.1) + scale_x_continuous(breaks = seq(0,3,0.2))
```



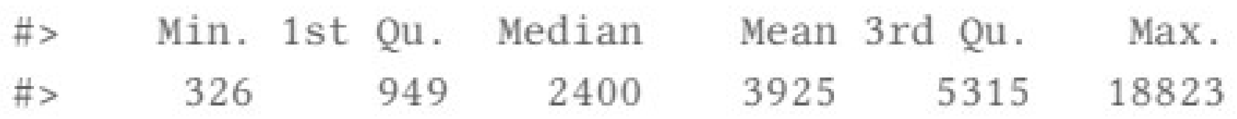
DATA 100,Week 4 (A)

```
ggplot(data = smaller, mapping = aes(x=carat)) + geom_histogram(binwidth =
0.01) + scale_x_continuous(breaks = seq(0,3,0.1)) + coord_flip()
```



DATA 100,Week 4 (A)

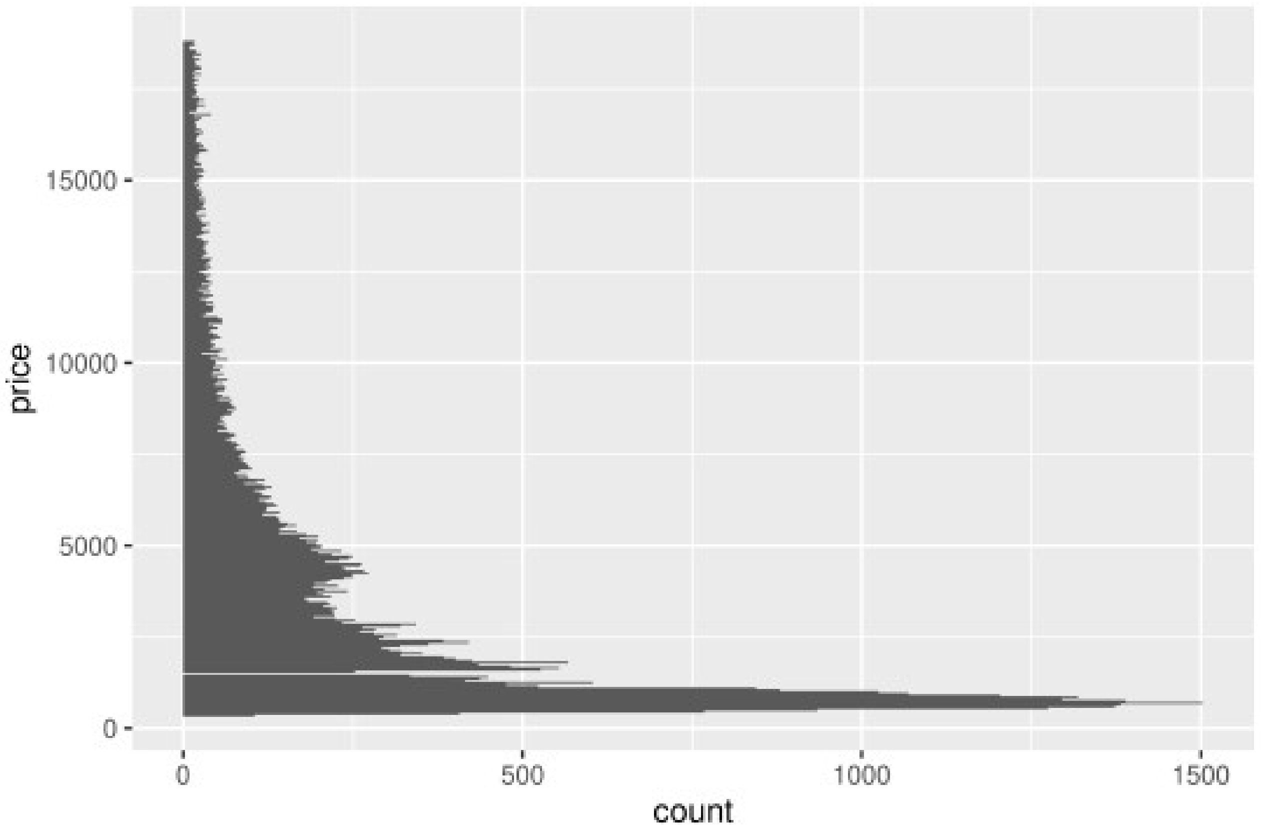
```
# use summary to get some rough idea on prices

summary(smaller$price)
```



```
#>     Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>      326     949    2400    3925    5315   18823
```

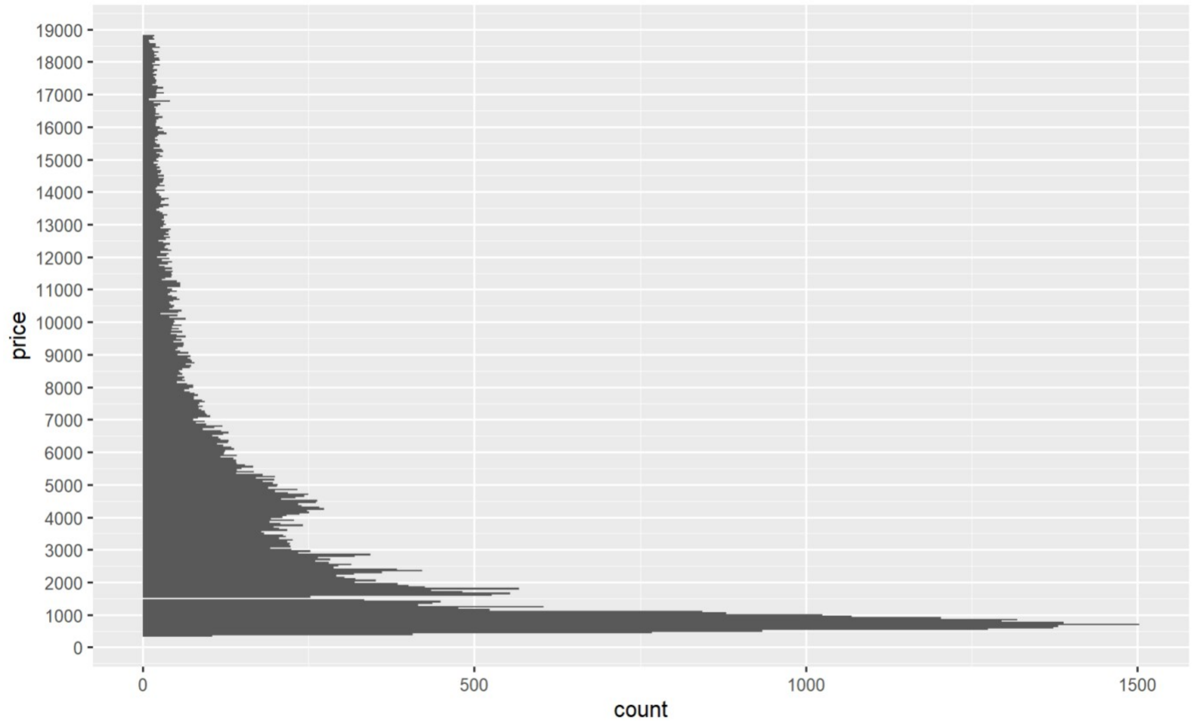
```
# then check distribution of prices

ggplot(data = smaller, mapping = aes(x=price)) +
geom_histogram(binwidth = 50) +
#scale_x_continuous(breaks = seq(0,20000,1000)) +
coord_flip()
```

DATA 100,Week 4 (A)



DATA 100,Week 4 (A)

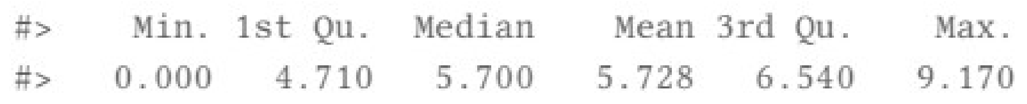
```
ggplot(data = smaller, mapping = aes(x=price)) +
geom_histogram(binwidth = 50) +
scale_x_continuous(breaks = seq(0,20000,1000)) +
coord_flip()
```



DATA 100,Week 4 (A)

# use summary to get some rough idea on prices

```
summary(smaller$x)
```

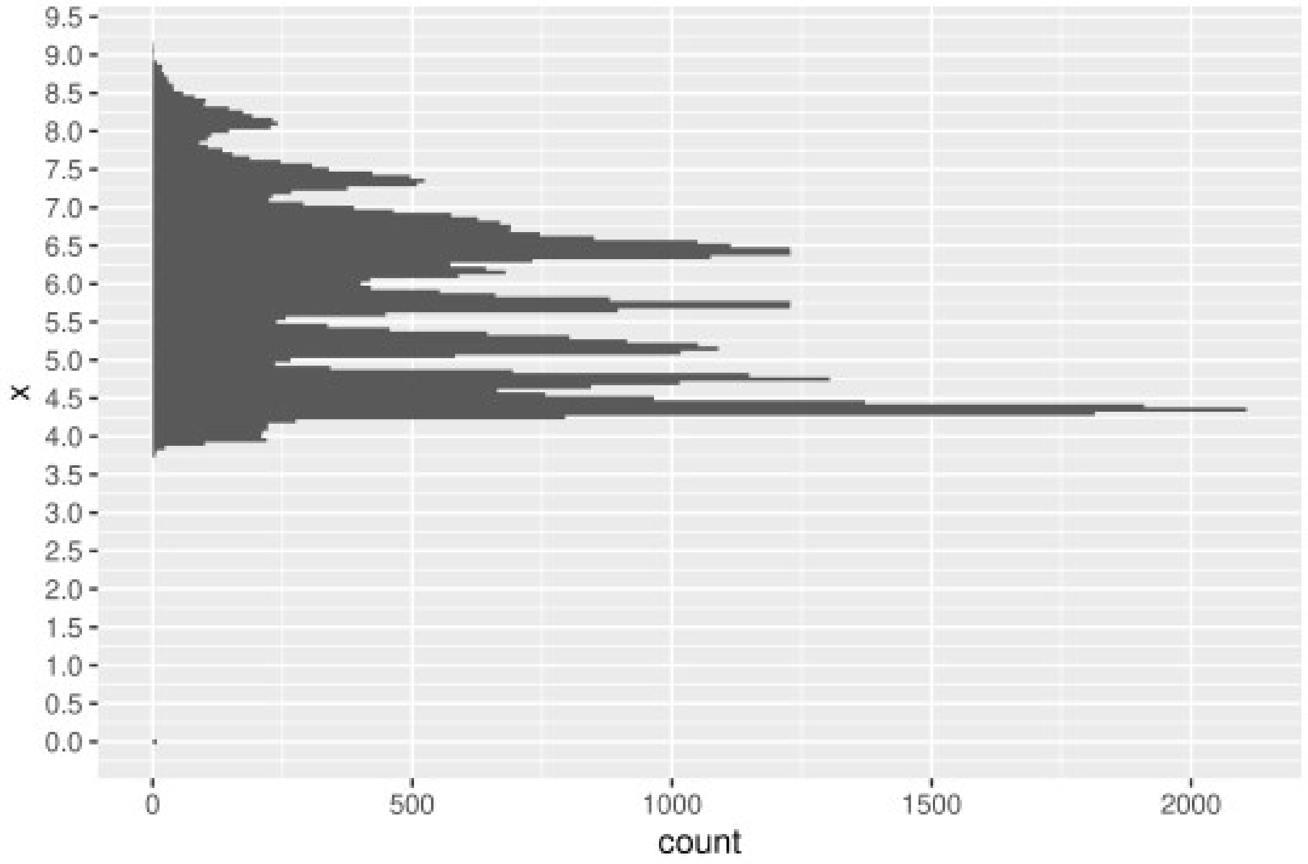


```
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>   0.000   4.710   5.700   5.728   6.540   9.170
```

# then check distribution of prices

```
ggplot(data = smaller, mapping = aes(x=x)) + geom_histogram(binwidth=0.05) +
scale_x_continuous(breaks = seq(0,9.5,0.5)) + coord_flip()
```

DATA 100,Week 4 (A)



DATA 100,Week 4 (A)

• data points that do not seem to fit the pattern

• observations that are unusual

They may come from data entry errors, or point toward important new knowledge.

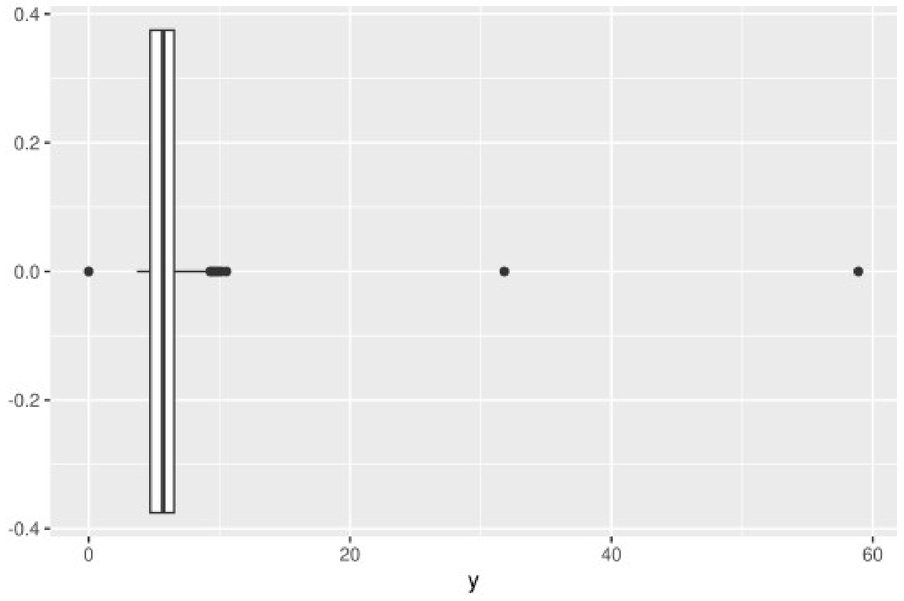
Where are the outliers?

• Below Q1 - 1.5 * IQR or above Q3 + 1.5 * IQR (Descriptive statistics II)
Can see it from boxplot.

DATA 100,Week 4 (A)

Note y below is one of the dimensions of the diamond

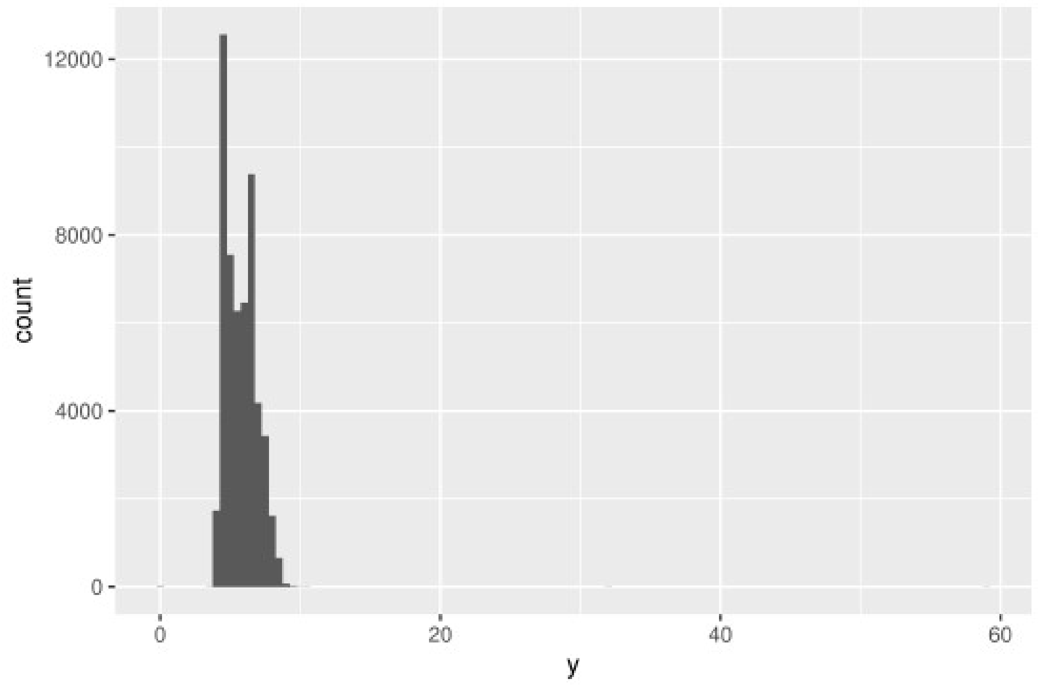
```
ggplot(diamonds) + geom_boxplot(mapping = aes(x = y))
```



DATA 100,Week 4 (A)

Bar chart or histogram can also help revealing **outliers**

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



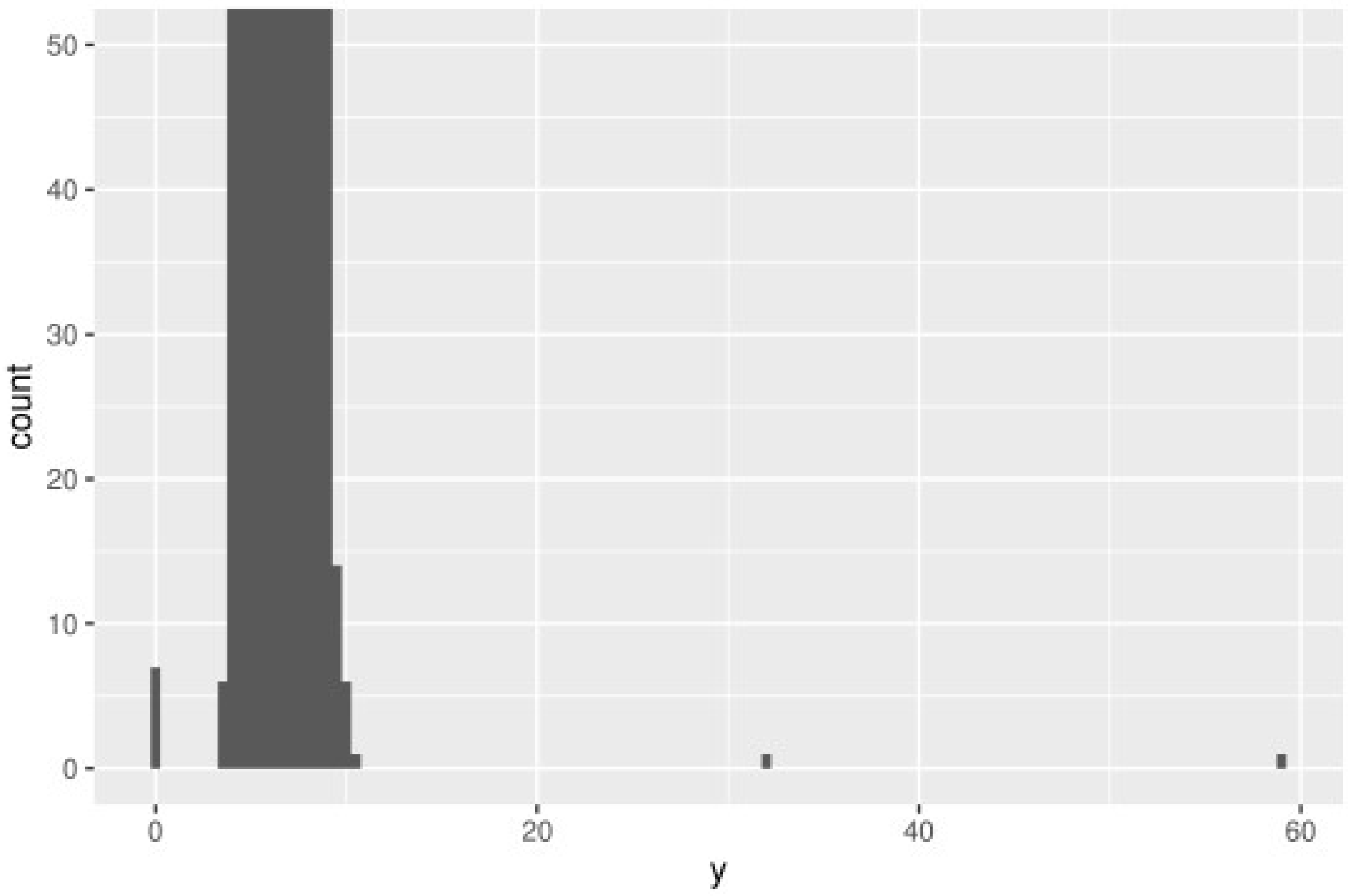
DATA 100,Week 4 (A)

Sometimes zooming in on the plot is needed:

- Using coord_cartesian() with ylim is the proper method for our purpose
- Using ylim() only achieves something completely different

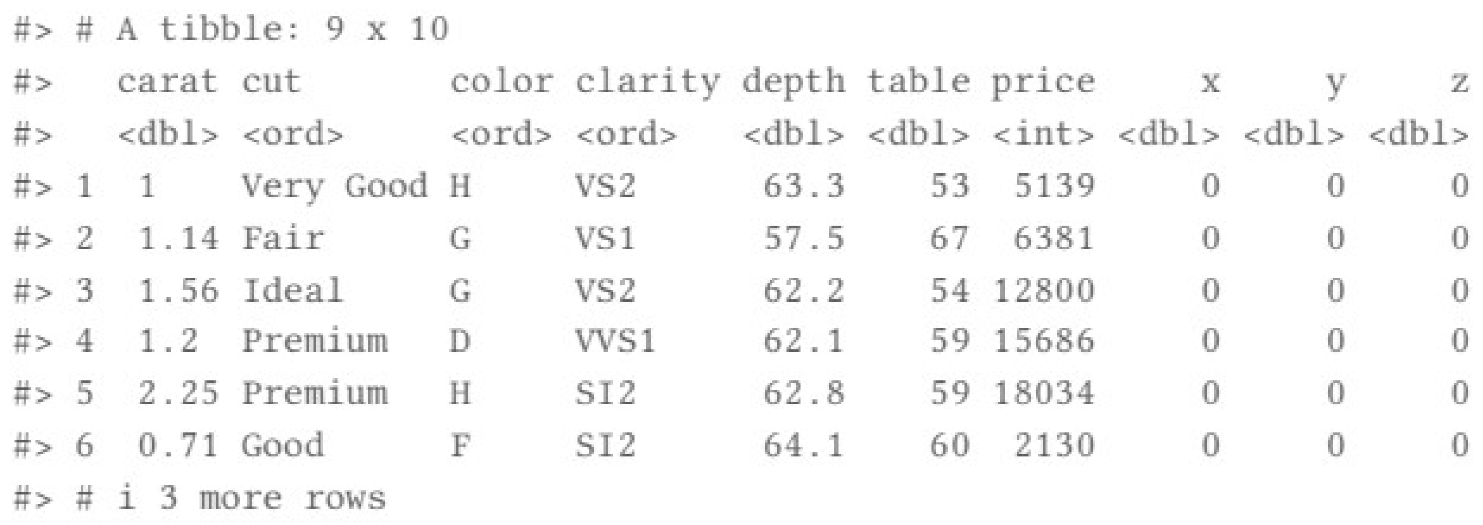
```
ggplot(diamonds) +
geom_histogram(mapping = aes(x = y), binwidth= 0.5) +
# setting limits on coord_cartesian has the effect of zoom-in
coord_cartesian(ylim = c(0, 50))
# ylim(0,50)
```

DATA 100,Week 4 (A)



DATA 100,Week 4 (A)

```
unusual <- diamonds |>
filter (y < 3 | y > 20) |>
arrange(y)

unusual
```

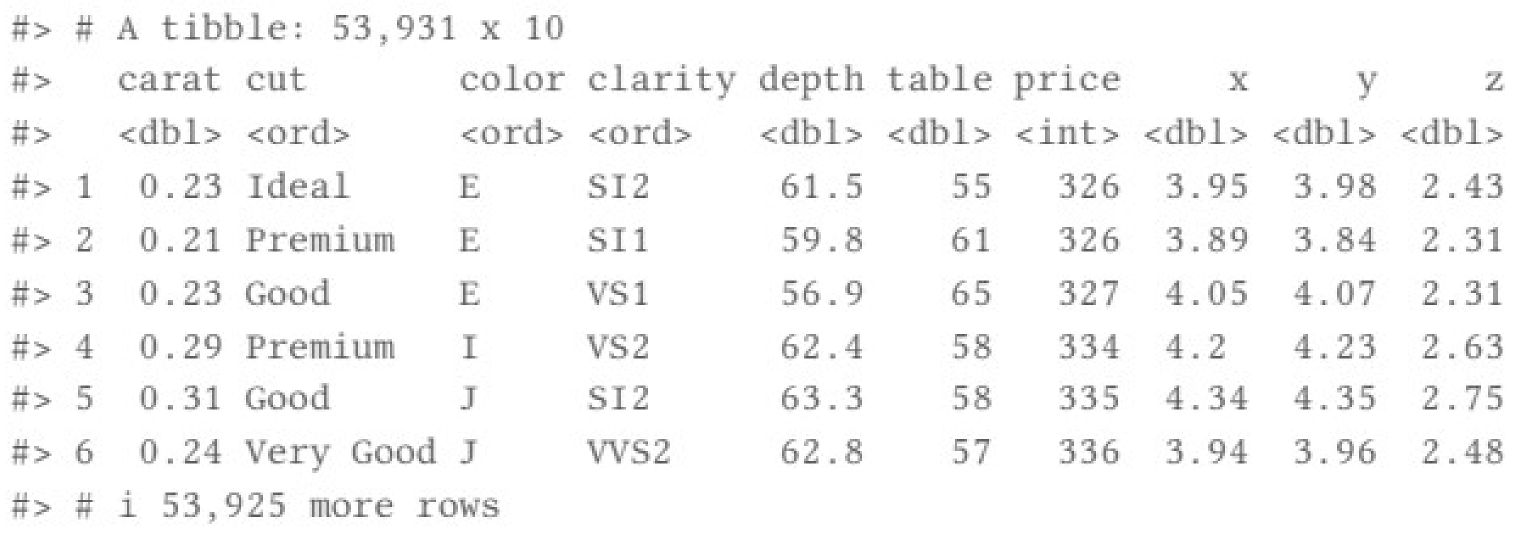


```
#> # A tibble: 9 x 10
#>   carat cut       color clarity depth table price     x     y     z
#>   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
#> 1  1    Very Good H     VS2      63.3    53  5139     0     0     0
#> 2  1.14 Fair      G     VS1      57.5    67  6381     0     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
#> # i 3 more rows
```

DATA 100,Week 4 (A)

Filter to drop the unusual values

```
diamonds |>
filter(between(y, 3, 20))
```



```
#> # A tibble: 53,931 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  0.29 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
#> # i 53,925 more rows
```

DATA 100,Week 4 (A)

Impossible values indicate various errors

• sometimes need a bit domain knowledge

Cast them aside from subsequent analysis would not affect much.

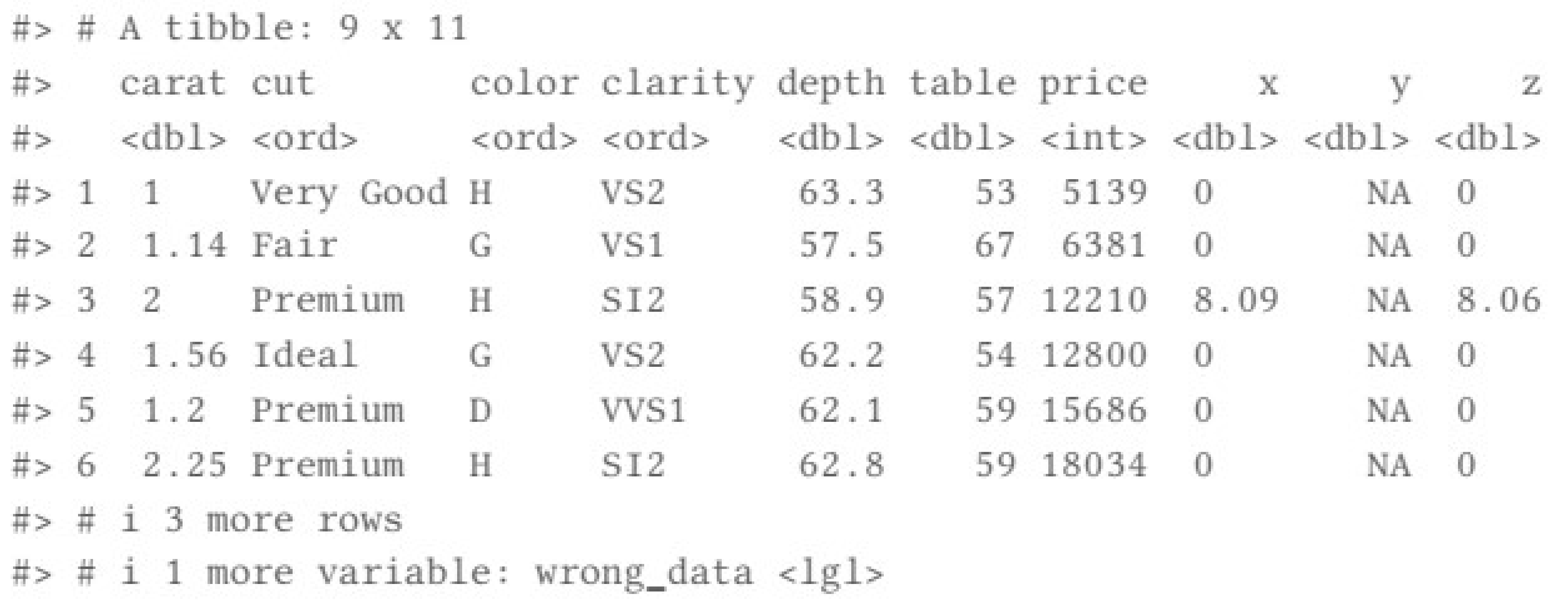
Two things can be done

• Set them as NA keeps the option of coming back for them later

```
diamonds2 <- diamonds |>
mutate(y = if_else( y < 3 | y > 20, NA, y))
```

DATA 100,Week 4 (A)

```
diamonds2 |>
mutate(wrong_data = is.na(y)) |>
filter(wrong_data)
```



```
#> # A tibble: 9 x 11
#>   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       NA  0
#> 2  1.14 Fair      G     VS1      57.5    67  6381  0       NA  0
#> 3  2    Premium   H     SI2      58.9    57 12210  8.09    NA  8.06
#> 4  1.56 Ideal     G     VS2      62.2    54 12800  0       NA  0
#> 5  1.2  Premium   D     VVS1     62.1    59 15686  0       NA  0
#> 6  2.25 Premium   H     SI2      62.8    59 18034  0       NA  0
#> # i 3 more rows
#> # i 1 more variable: wrong_data <lgl>
```

DATA 100,Week 4 (A)

Descriptive statistics, IV

Discussed here are **variance**, **standard deviation** and **standard error** **of a sample**

```
# Use the same collection of 20 integers

sample <- c(29, 35, 33, 32, 34, 30, 28, 33, 34, 35, 10, 14, 14, 12, 12, 12, 10, 14,
16, 10)
```

DATA 100,Week 4 (A)

Recall

• mean (for sample): the sum of entries divide by the number of entries

• median: at *most* 50% of the entries are below it, AND at *most* 50% of the entries are above it.

• quartile: three of these, Q1, Q2, Q3

– Q1: the 25th percentile, also called first quartile
– Q2: the 50th percentile, i.e. the median, also called second quartile
– Q3: the 75th percentile, also called the third quartile
– Q1 and Q3 may be thought of as the medians of the lower and upper half of the data respectively

DATA 100,Week 4 (A)

variance

- Use $x_1, x_2, \dots, x_n$ to denote the collection of entries (called samples)
- The variance discussed here is the sample variance

Sample variance usually denoted by s^2

$$s^2 = \frac{(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + \dots + (x_n - \bar{x})^2}{n - 1}$$

First compute "by hand" (with help from R)

DATA 100,Week 4 (A)

```r
avg <- mean(sample)
variance <- sum((sample - avg)^2) / (20-1)
variance
```

```
#> [1] 109.1868
```

Note `(sample - avg)^ 2` is definitely the wrong thing to write as a mathematical formula

- This is pecular in R
- arithmetic operations on a vector is vectorized, i.e. entry-wise!

The function var() computes the sample variance.

DATA 100,Week 4 (A)

```
var(sample)

#> [1] 109.1868
```

DATA 100,Week 4 (A)

standard deviation

The square root of the sample variance:

$$s = \sqrt{\frac{(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + \ldots + (x_n - \bar{x})^2}{n - 1}}$$

Measures, in some sense, the average distance of the values from the mean

- gives an idea how far all the observations are from the mean.
- e.g. when $s = 0$ then all the observations have exactly the same value.

DATA 100,Week 4 (A)

In general, the smaller s is, the more chance that the mean $\bar{x}$ is a good representation of the measurements.

Compute by hand for our sample again:

```
sdeviat <- sqrt(variance)

sdeviat

#> [1] 10.44925
```

It roughly says that the entries in sample is spread about 10.5 away from the mean

DATA 100,Week 4 (A)

standard error **of the mean (a subtle concept)**

standard deviation devided by $\sqrt{n}$, denoted se $(\bar{x})$

- related to se in geom_smooth()

$$se(\bar{x}) = \frac{1}{\sqrt{n}} s = \sqrt{\frac{(x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + \dots + (x_n - \bar{x})^2}{n(n-1)}}$$

It gives an idea on how close $\bar{x}$ (the sample mean obtained from the n observations) is from the population mean

- in particular, we can see that the more we sample (larger n), we have a better chance of getting close to the population mean

- $se(\bar{x}) = 0$ DOES NOT imply that the sample mean is the population mean

DATA 100,Week 4 (A)

Compute by hand for our sample:

```
sdeviat / sqrt(20)
```

```
#> [1] 2.336524
```

Use the function sd() and length() to compute the standard error

- length() produces the number of entries in a vector

```
sd(sample) / sqrt(length(sample))
```

```
#> [1] 2.336524
```

DATA 100,Week 4 (A)

95% confidence interval of the mean is about 1.96se $(\bar{x})$ around the sample mean (if the distribution is normal)

- meaning that the true mean has 95% chance to be within $1.96se(\bar{x})$ distance from the sample mean
- in geom_smooth (), setting se = TRUE, shows $1.96se(\bar{x})$ as the shade around the curve

In our sample, it roughly means, if the distribution is normal, then there is a 95% chance that the true mean is within $1.96 * 2.336524$ from the mean of the sample:

```
1.96*2.336524
#> [1] 4.579587
mean(sample)
#> [1] 22.35
```

DATA 100,Week 4 (A)

Population mean and population standard deviation

Population mean: the hypothetical mean value if one could average over ALL observations

- usually denoted μ

One of the goals for doing statistics is to make predictions, which is to say that we gain knowledge about all possible occurances of similar phenomena by looking at a subset of them that we can access.

The subset we can access is called **sample**, and ALL occurance is called **population**.

DATA 100,Week 4 (A)

Thus we must make connection from the sample to the population.

Sample mean can be used to estimate population mean. Details in statistics courses.

population standard deviation: is usually denoted σ_x :

$$\sigma_x = \sqrt{\frac{(x_1 - \mu)^2 + (x_2 - \mu)^2 + \dots + (x_n - \mu)^2}{n}}$$

- Notice the different denominator n here instead of $\mathrm{n} - 1$ for sample standard deviation

DATA 100,Week 4 (A)

Continue on EDA (exploratory data analysis) (1e: Chapter 7 and 2e: Chapter 11)

Covariation: involving categorical variables

Describes how the values of one variable affects / are related to those of another one across observations

To see the tendency for the values of two or more variables to vary together in a related way

- Start by visualizing the relationship between two or more variables.

DATA 100,Week 4 (A)

Choice of visualization depends on type (categorical or continuous) of variables involved.

- Categorical and continuous

- Categorical and categorical

- Continuous and continuous

DATA 100,Week 4 (A)

A Categorical and Continuous Variable

The facet_ functions (Lecture 02) plot the continuous variables on separate figures, grouped by the values of the categorical variable.

For comparison, it is more useful to plot the different groups on the same figure.

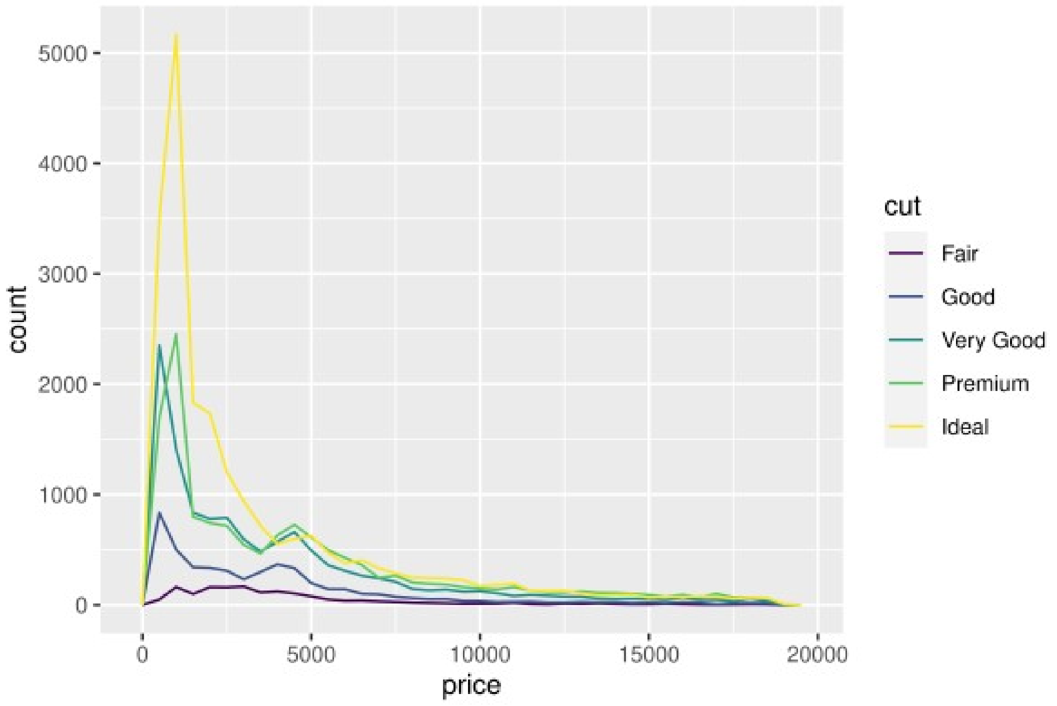
Expectation: Better quality may indicate higher proportion in higher prices.

Use geom_freqpoly to compare distributions of prices of diamonds across different qualities.

Plot the actual counts directly does not lend to easy comparison.

DATA 100,Week 4 (A)

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



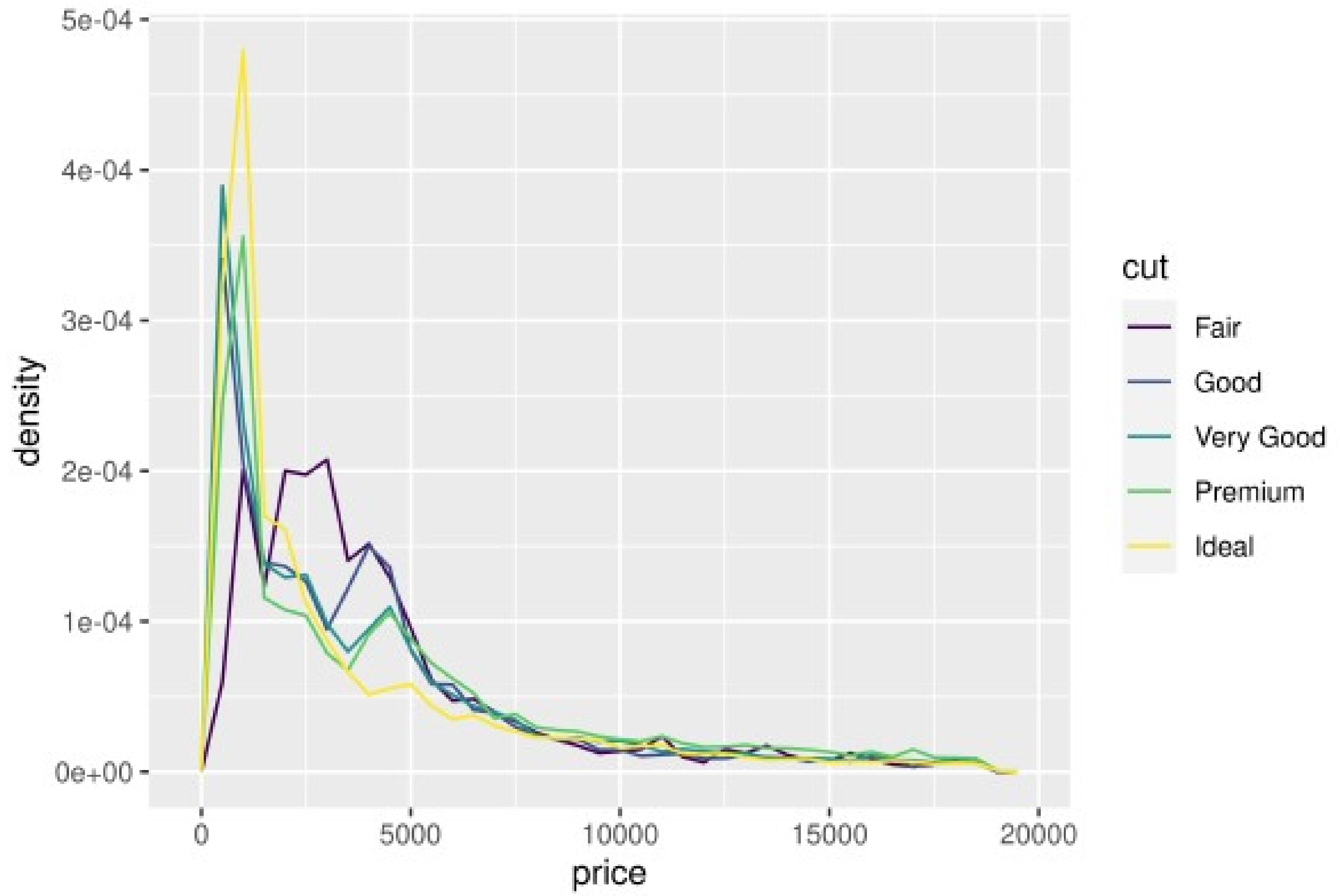
DATA 100,Week 4 (A)

Standardize the plots so that area below each curve now is 1, to illustrate the *proportions*.

- Use y = after_stat(density)

```
diamonds |>
ggplot(mapping = aes(x = price, y = after_stat(density)))+
geom_freqpoly(mapping = aes(colour=cut), binwidth = 500)
```

DATA 100,Week 4 (A)



DATA 100,Week 4 (A)

Observation:

Different quality cuts in fact have roughly the same proportion in high prices!

Even stranger: Ideal – the best cut – somehow has slightly more in the lower price range.

DATA 100,Week 4 (A)

boxplots gives a more compact display of (somewhat less) information.

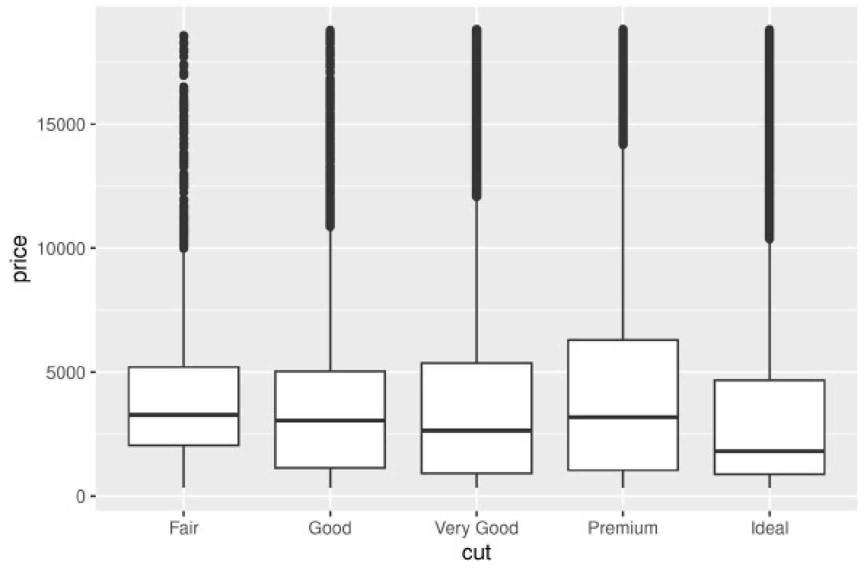
It still illustrates the gist of the observation.

Namely

• Ideal (best) cut diamonds do not seem to cost more, in fact seems to be less, than the others over all.

DATA 100,Week 4 (A)

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



The medians for premium and fair seems to be close, which one is higher? They are too far away in the plot to be compared effectively

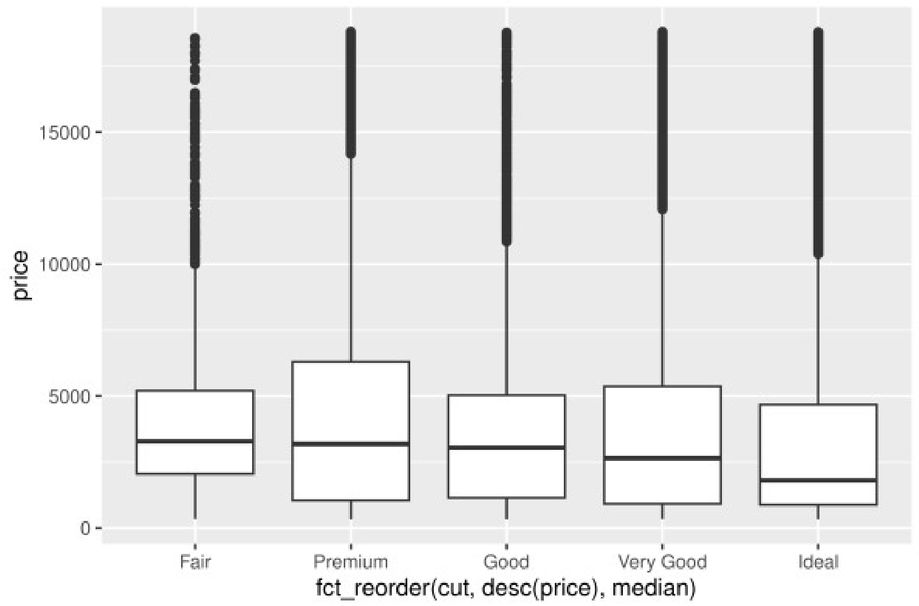
DATA 100,Week 4 (A)

- fct_reorder() one variable using the value of another (computed) variable

  Do ?fct_reorder in console to see more information

```
ggplot(data = diamonds) + geom_boxplot(
mapping = aes(
x = fct_reorder(cut, desc(price), median),
y = price )
) #+
#coord_flip()
```

DATA 100,Week 4 (A)



Potential observation: *Cut may not significantly influence the price*

DATA 100,Week 4 (A)

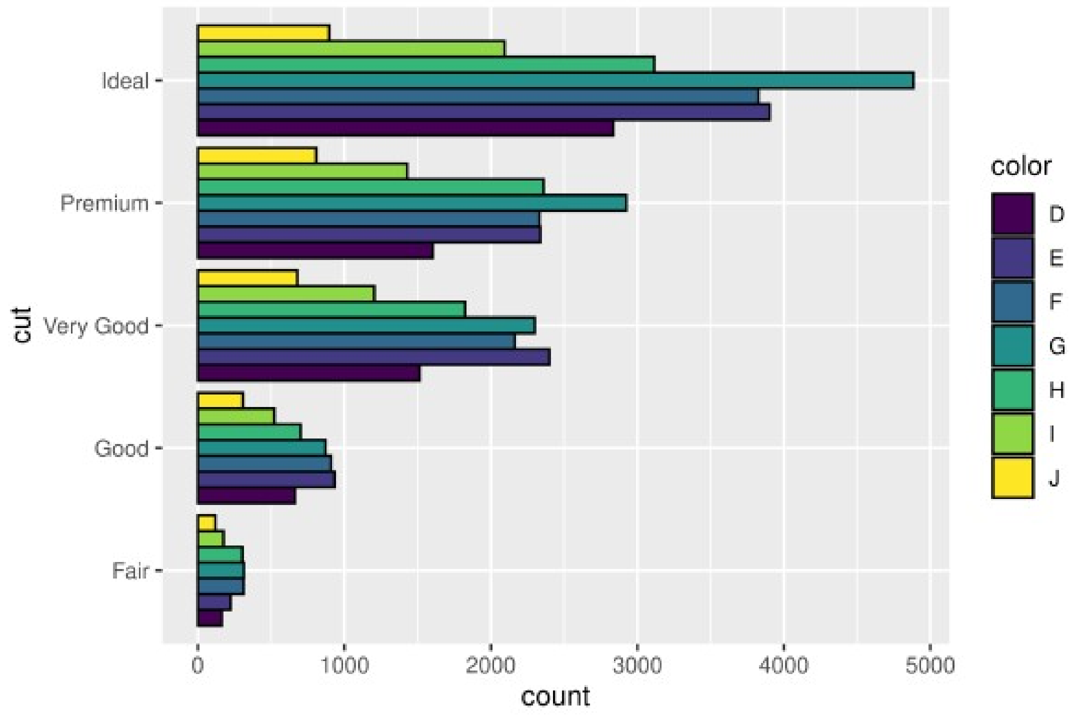
Two Categorical Variables

There are also various ways to illustrate how the distributions of one categorical variable is related to those of the other.

We have seen the following type of bar diagrams involving two categorical variables.

```
diamonds |>
ggplot(mapping = aes(x = cut)) +
geom_bar(mapping = aes(fill = color), color = "black", position = "dodge") +
coord_flip()
```

DATA 100,Week 4 (A)

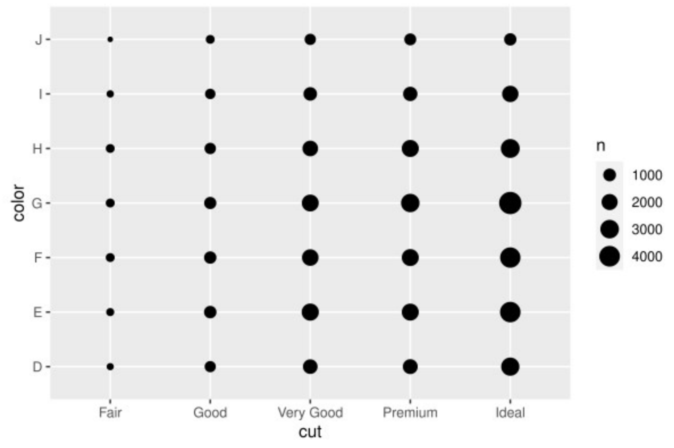


DATA 100,Week 4 (A)

Two dimensional plots such as geom_count() or geom_tile() may be more intuitive in this case.

• Thinking of the count as a function of cut and color

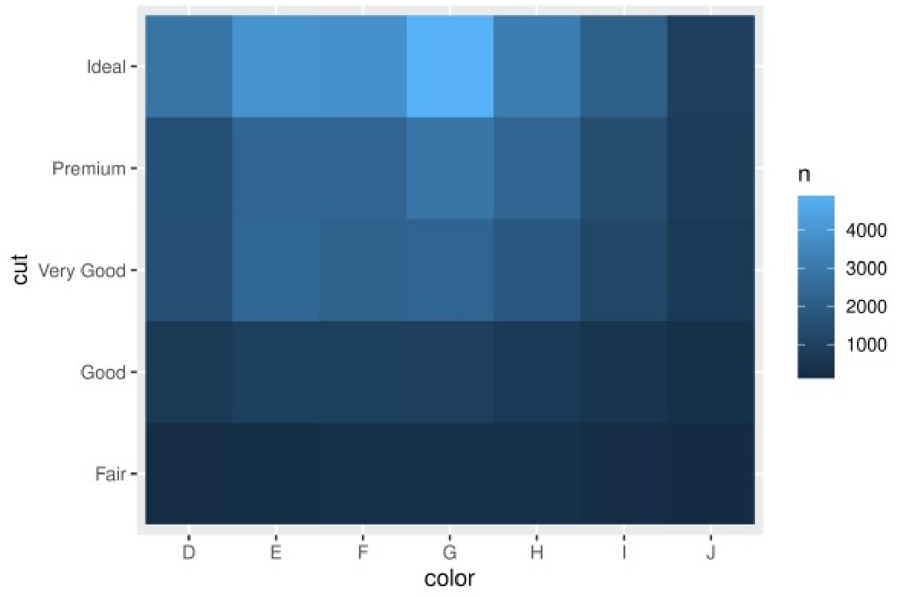
```
diamonds |>
ggplot() +
geom_count(mapping = aes(x = cut, y = color))
```



DATA 100,Week 4 (A)

After getting the count() use geom_tile to create a *heat map*

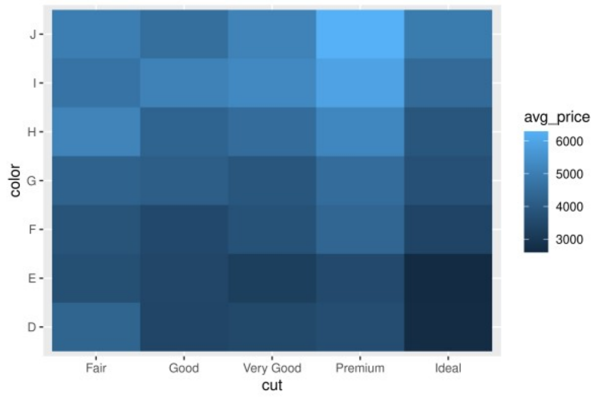
```
diamonds |>
count(color, cut) |>
ggplot(mapping = aes(x = color, y = cut)) + geom_tile(mapping = aes(fill = n))
```



DATA 100,Week 4 (A)

The same process can be adapted to illustrate how another variable depends on the combination of **two categorical variables**, putting more information into one plot.

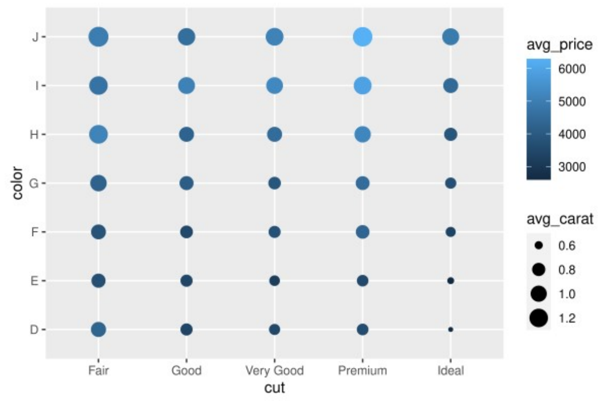
```
diamonds |>
summarize(
avg_price = mean(price), .by = c(cut, color)
) |>
ggplot(mapping = aes(x = cut, y = color)) + geom_tile(mapping = aes(fill =
avg_price))
```



DATA 100,Week 4 (A)

Even more information can be piled on.

```
diamonds |>
summarise(
avg_price = mean(price), avg_carat = mean(carat), .by = c(cut, color)
) |>
ggplot(mapping = aes(x = cut, y = color)) + geom_point(mapping = aes(color =
avg_price, size=avg_carat))
```


color
J
I
H
G
F
E
D
Fair
Good
Very Good
Premium
Ideal
cut
avg_price
avg_carat