Introggplot

Ryan

September 28, 2015

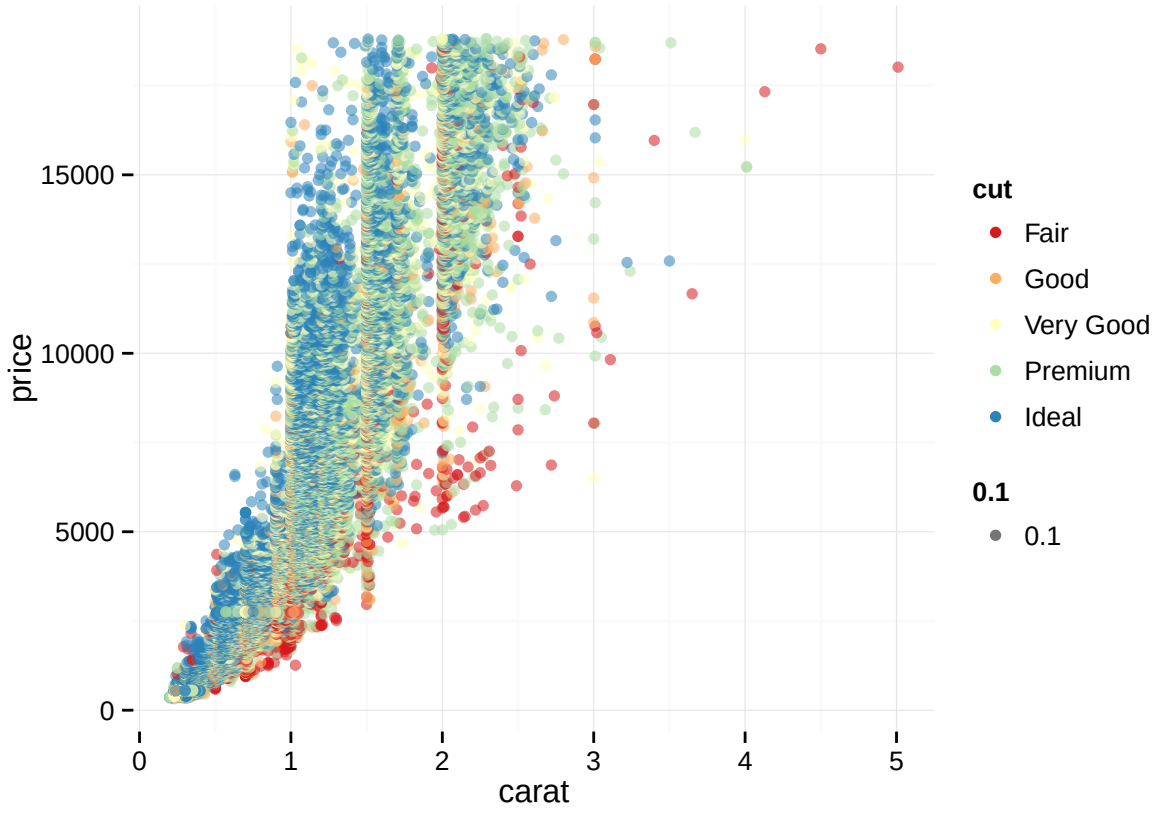
Install the packages first

```
#install.packages(c("ggplot2","RColorBrewer","ggthemes","gridExtra"))
library(ggplot2)
require(RColorBrewer)
library(ggthemes)
require(gridExtra)
```

A first plot

The diamonds dataset come with ggplot2 package and often used to show examples of ggplot2
Don't worry everything will be clear to you very soon

```
data(diamonds)
qplot(data = diamonds, x = carat, y = price, color = cut, alpha = 0.1) +
    scale_color_brewer(palette = "Spectral") +
    theme_minimal()
```



Variables in the diamonds dataset

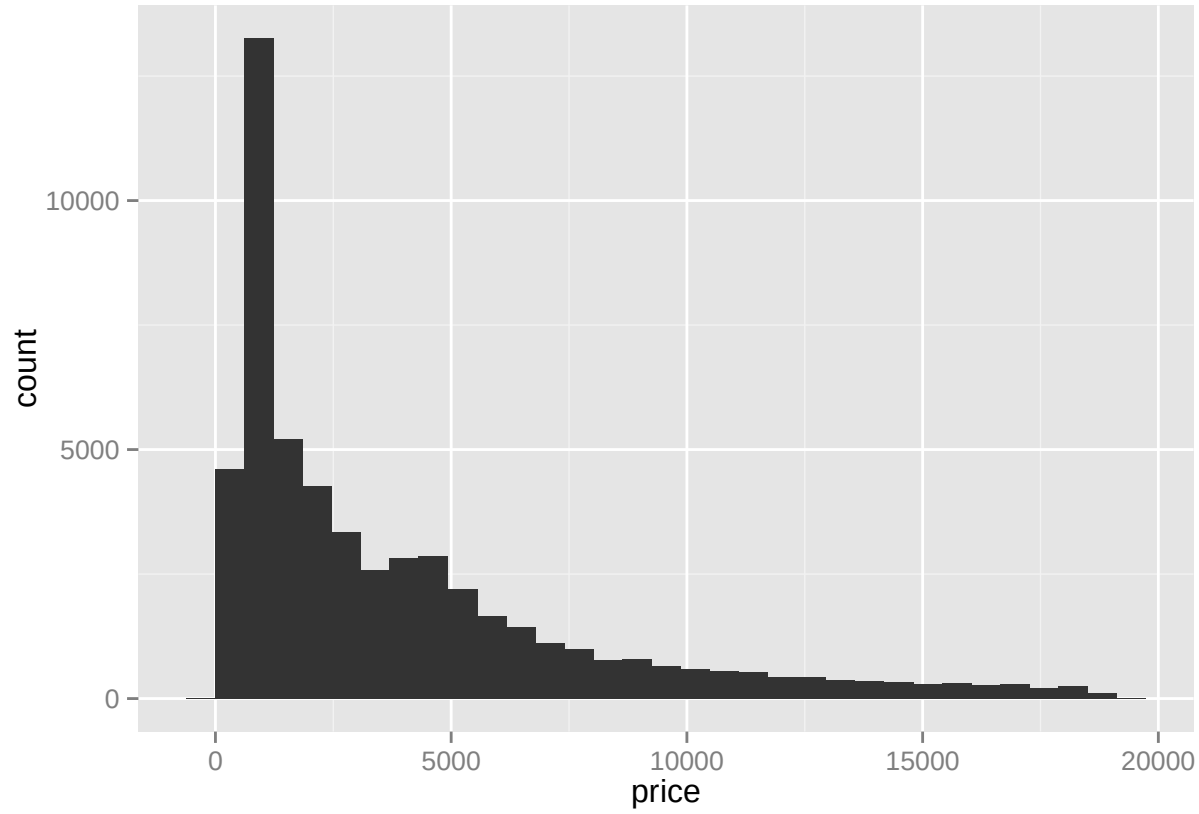
```
str(diamonds)
```

```
## 'data.frame':    53940 obs. of  10 variables:
##  $ carat  : num  0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
##  $ cut    : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 3 3 1 3 ...
##  $ color  : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6 5 2 5 ...
##  $ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2 6 7 3 4 5 ...
##  $ depth  : num  61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
##  $ table  : num  55 61 65 58 58 57 57 55 61 61 ...
##  $ price  : int  326 326 327 334 335 336 336 337 337 338 ...
##  $ x      : num  3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
##  $ y      : num  3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
##  $ z      : num  2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
```

qplot()

Histogram using `qplot()` function
`qplot()` is one plotting function in ggplot, the other one is `ggplot()`
`ggplot()` syntax is more difficult to understand
We will start with

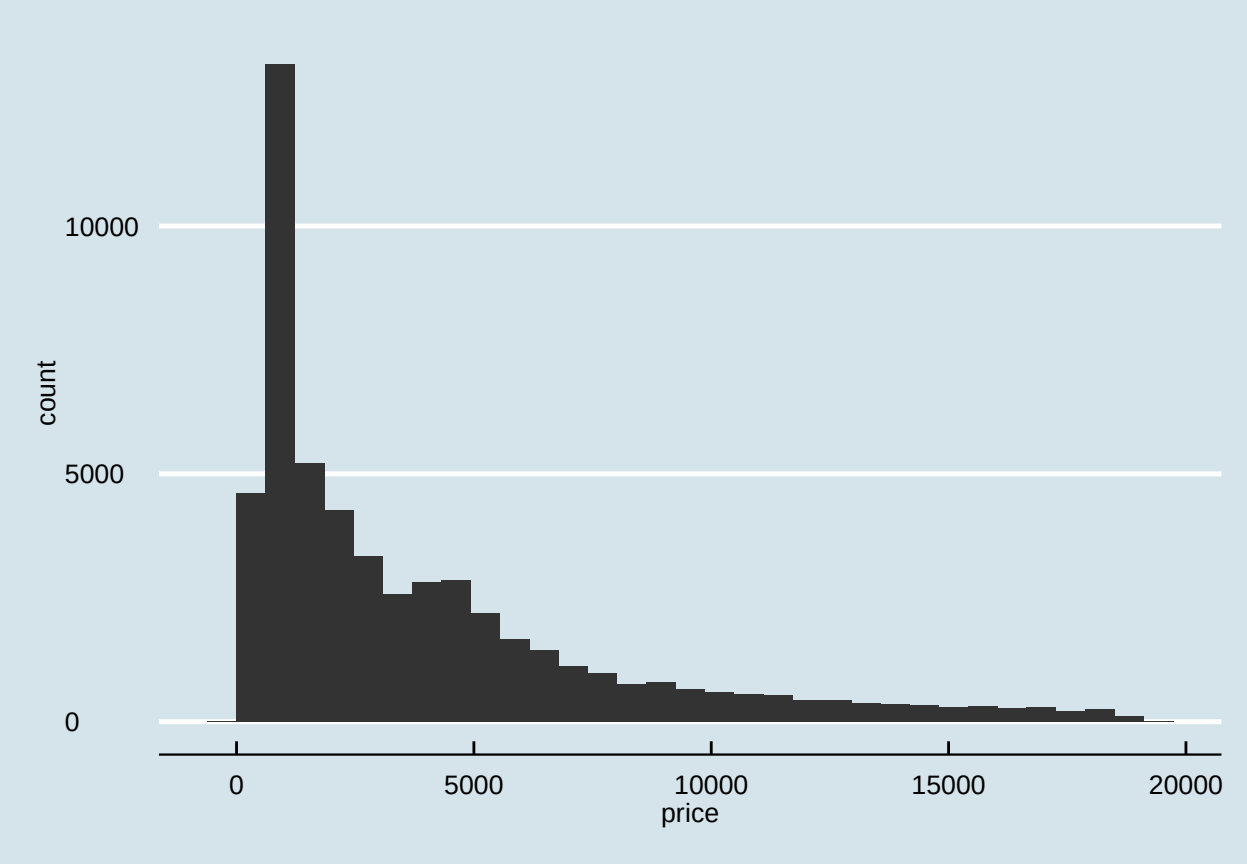
```
qplot(x = price, data = diamonds)
```



Alter the look using themes in ggthemes package

Try different themes

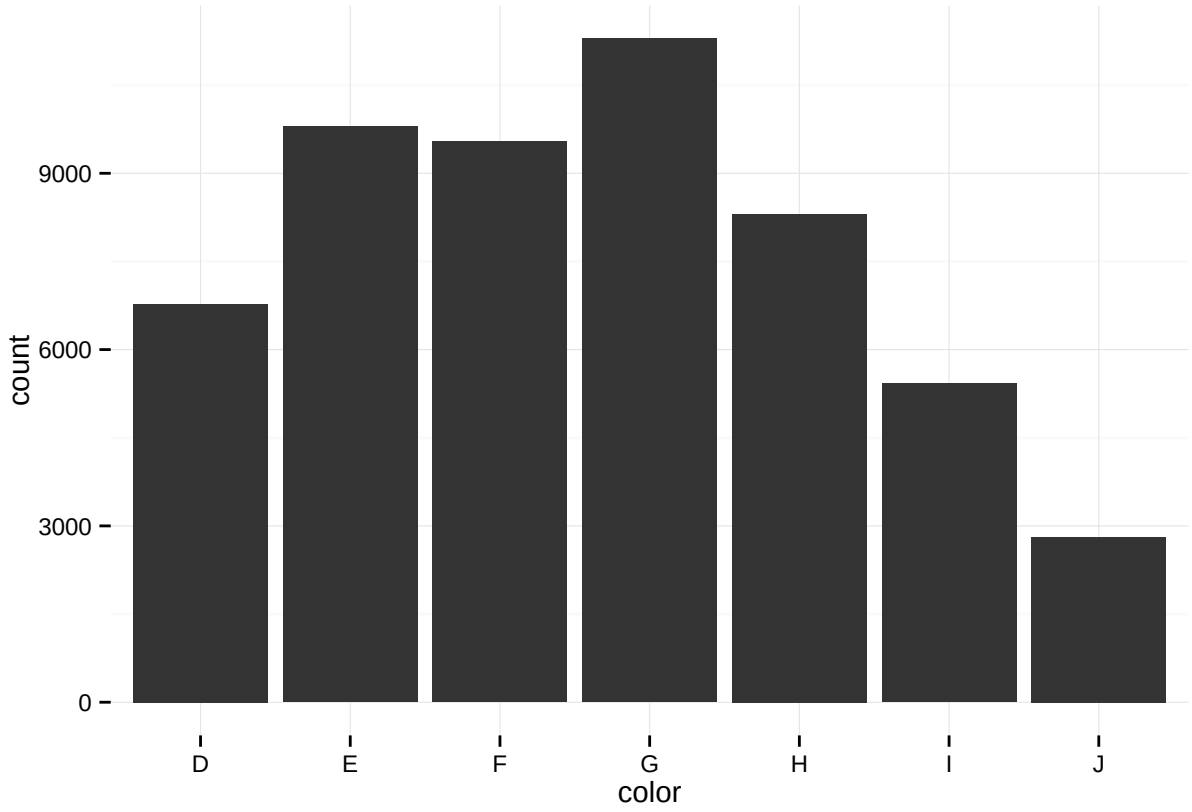
```
qplot(x = price, data = diamonds) + theme_economist()
```



Setting default theme

Set once, use through out the file

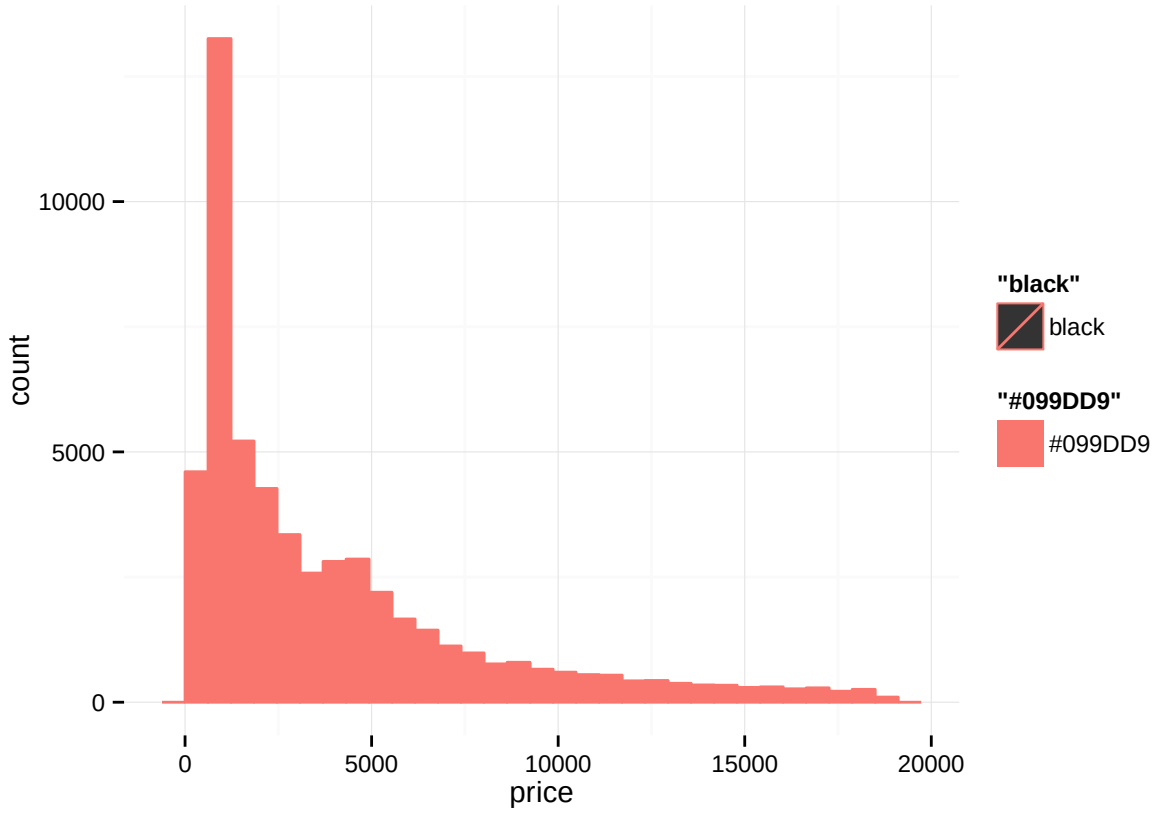
```
theme_set(theme_minimal(11))
qplot(x = color, data = diamonds)
```



Introducing color

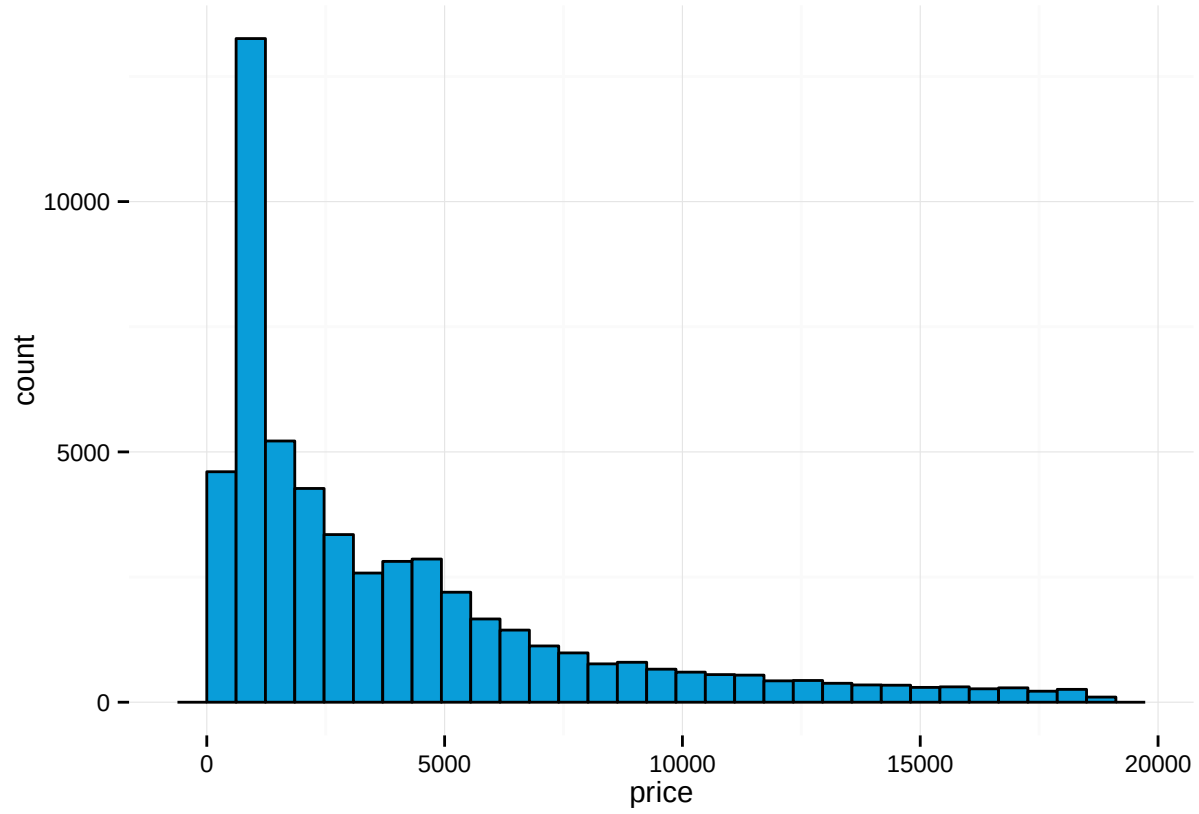
try #099DD9 here w3School color picker It is light blue
Let's use color name and the hex code

```
qplot(x = price, data = diamonds, color = 'black', fill = '#099DD9')
```



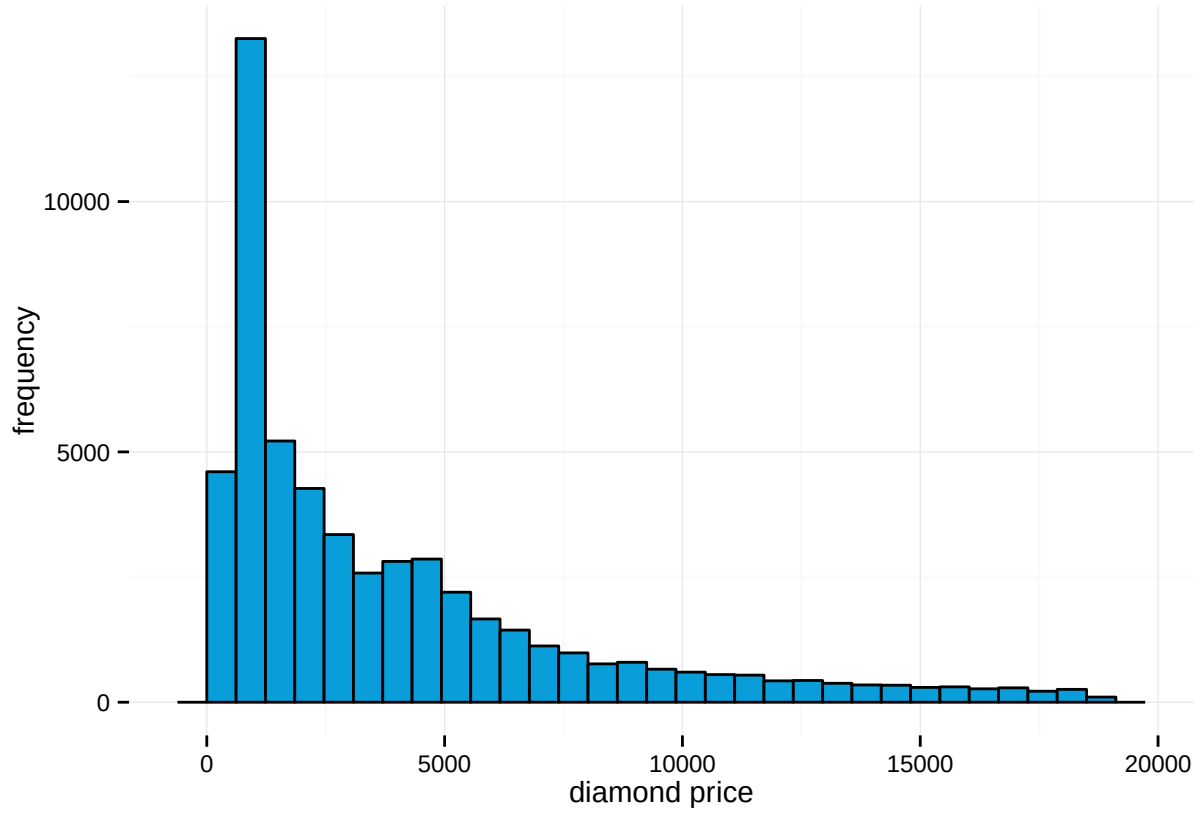
What's wrong...?
ggplot does not treat 'black' and '#099DD9' in default, to force ggplot to treat them as colors, use `I()` the 'as is' function

```
qplot(x = price, data = diamonds, color = I('black'), fill = I('#099DD9'))
```



Change the axis labels

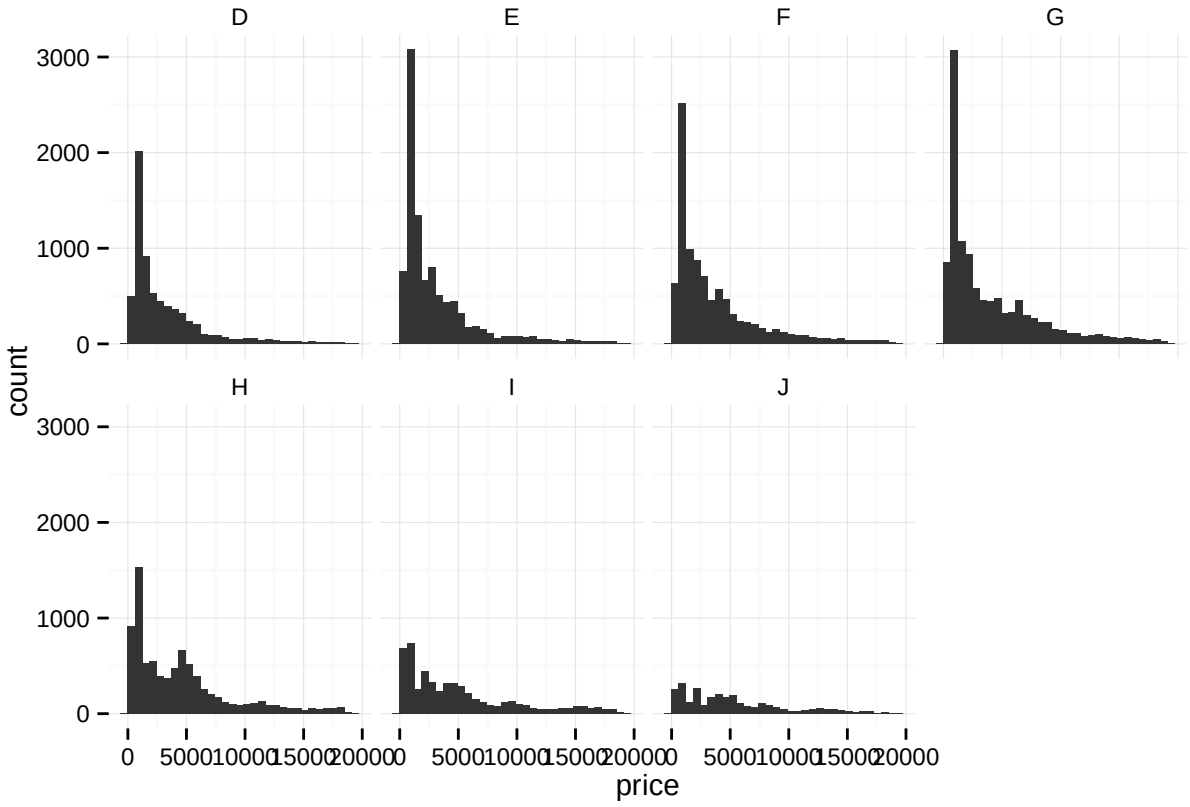
```
qplot(x = price, data = diamonds, color = I('black'), fill = I('#099DD9'),
      xlab = "diamond price", ylab = 'frequency')
```



Facet wrap

Use facet wrap to create a series of plot on subset of the data
This is equivlent to subset the dataframe using color, and plot histograms for each subset

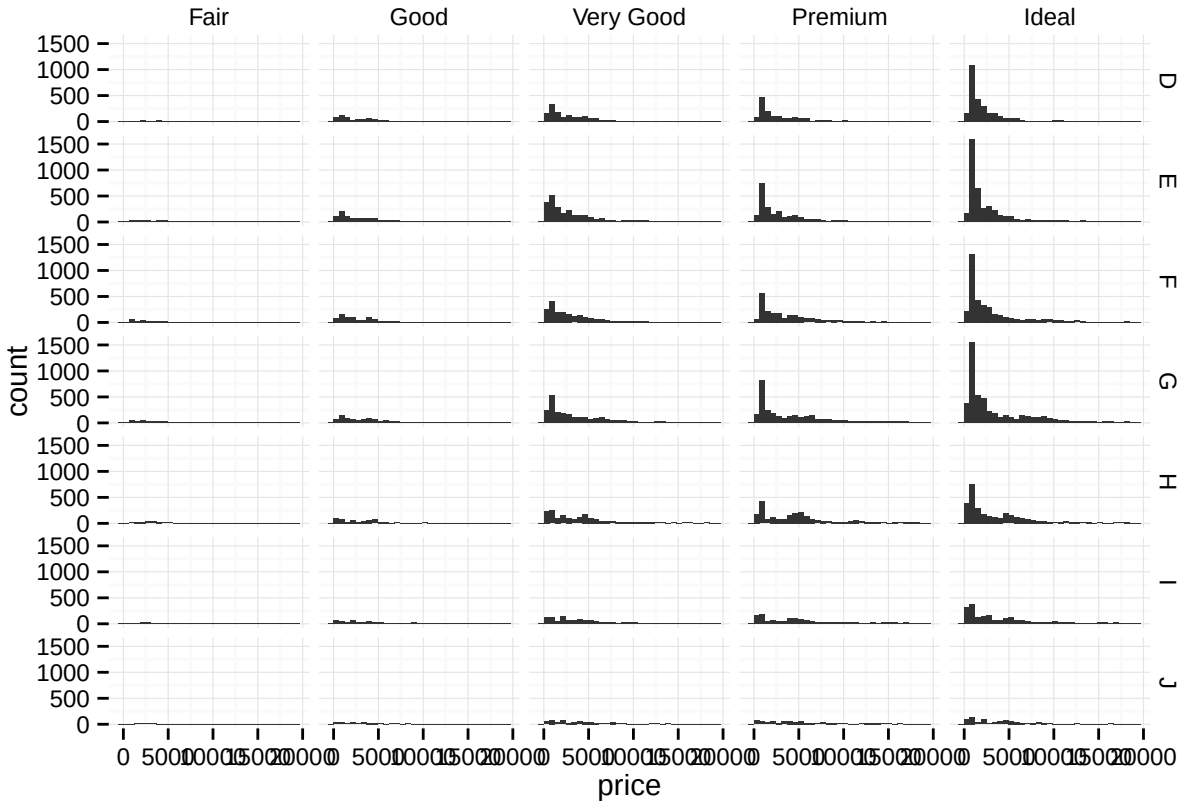
```
qplot(x = price, data = diamonds) +
    facet_wrap(~color, ncol = 4)
```



Facet grid

Use facet grid to explore subset according to combinations of two variables

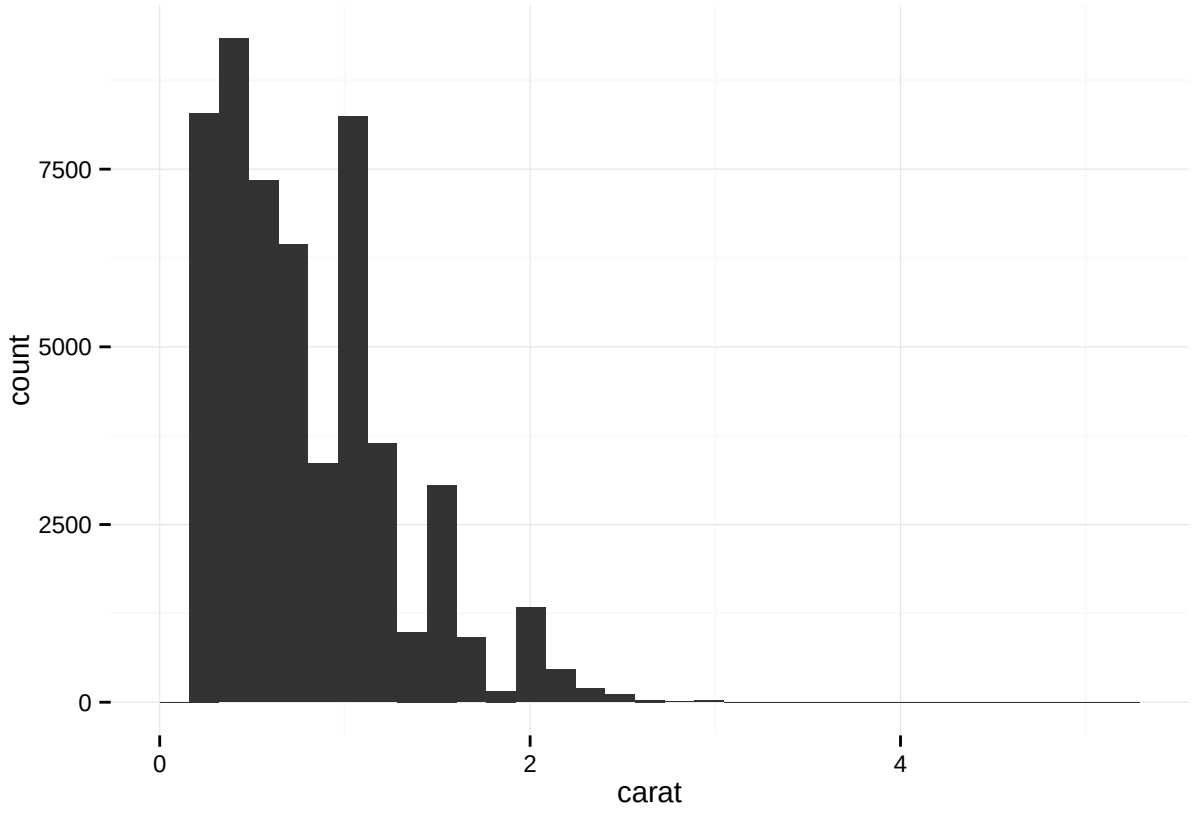
```
qplot(x = price, data = diamonds) +
    facet_grid(color~cut)
```



Adjust axes

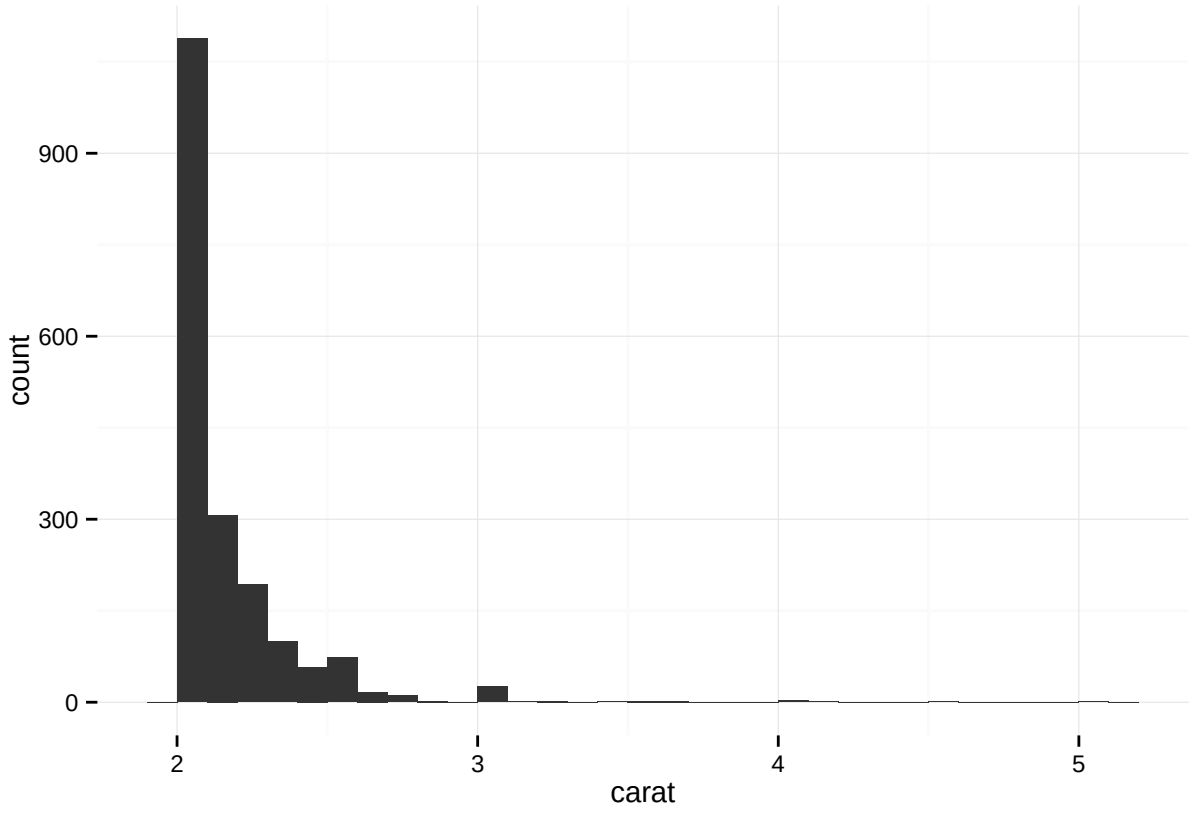
Let's look at the measure that really matters. . . .

```
qplot(x = carat, data = diamonds)
```



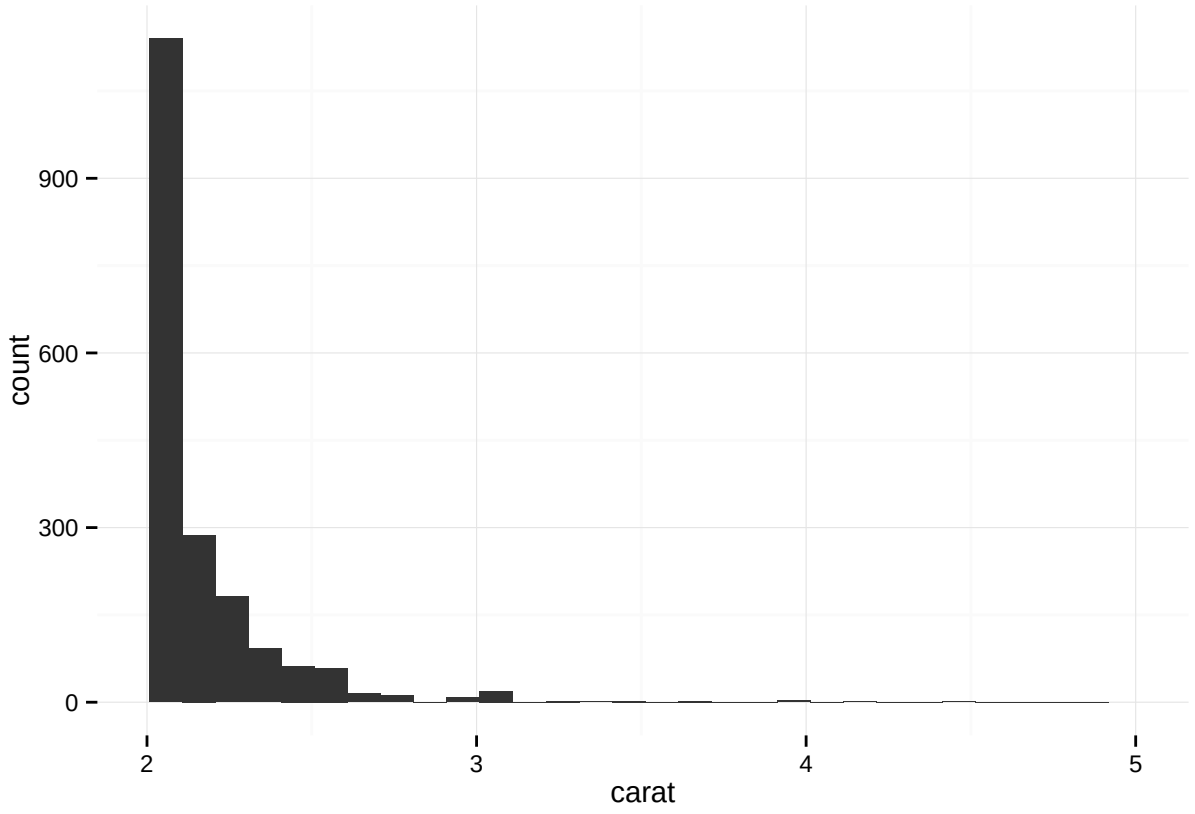
What if... I am only interested in the BIG diamonds?
Subsetting is ofcourse a way to do it

```
qplot(x = carat, data = diamonds[diamonds$carat > 2,])
```



Or do it using a more ggplot way

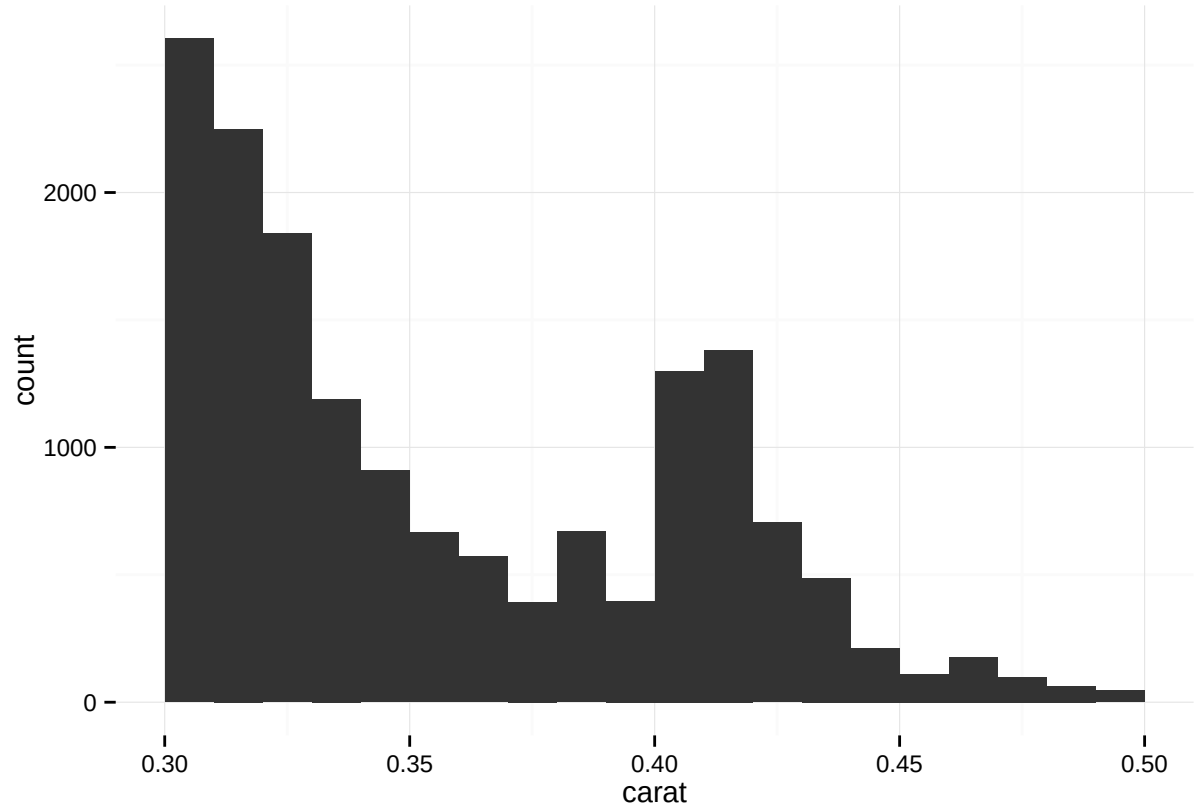
```
qplot(x = carat, data = diamonds) +
    scale_x_continuous(limits = c(2,max(diamonds$carat)))
```



Setting binwidth

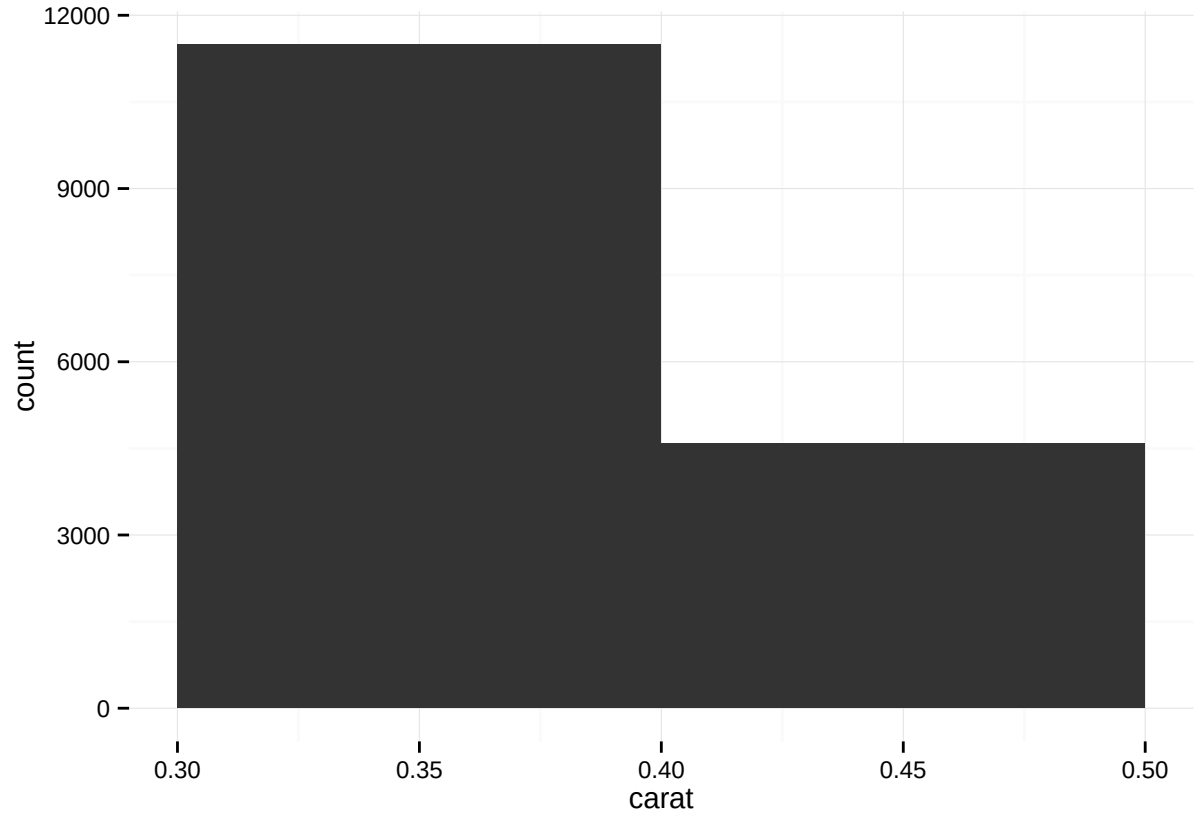
to set binwidth, use breaks()

```
qplot(x = carat, data = diamonds, binwidth = 0.01) +
    scale_x_continuous(limits = c(0.3,0.5))
```



What is a good binwidht?

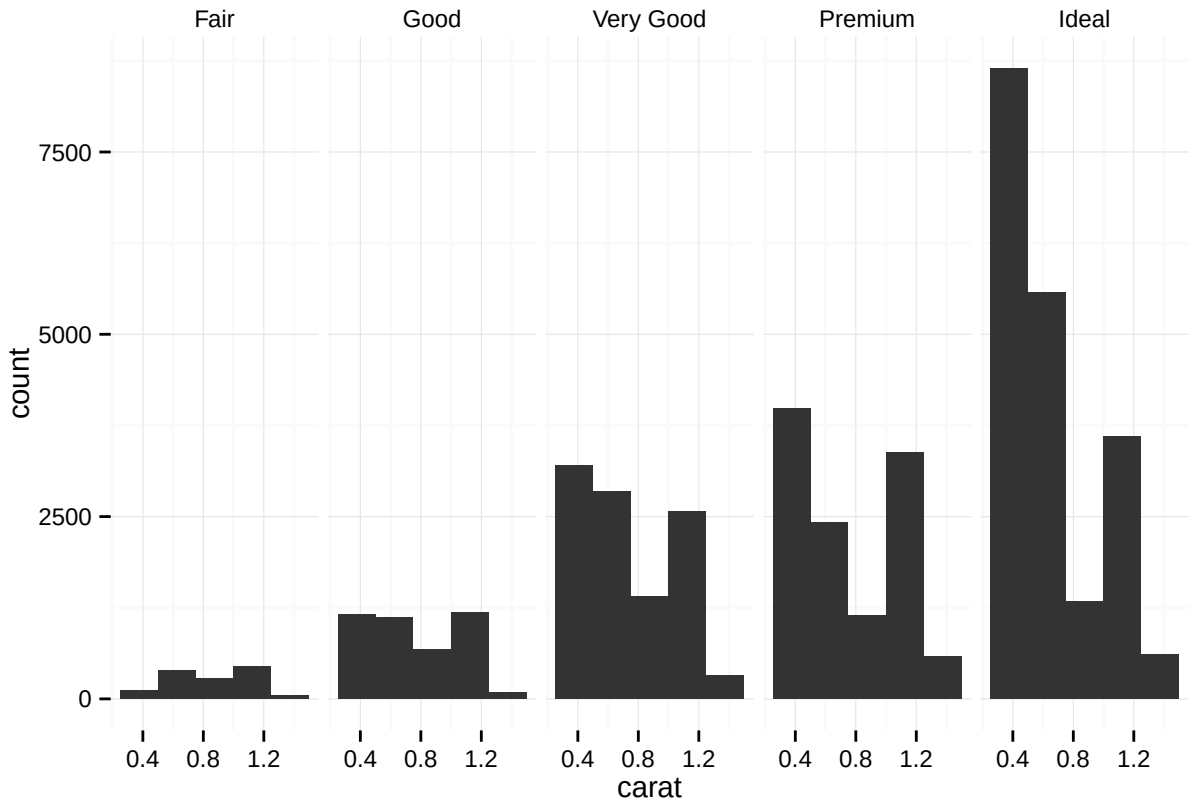
```
qplot(x = carat, data = diamonds, binwidth = 0.1) +
    scale_x_continuous(limits = c(0.3,0.5))
```



Layer on layer

The natural way to think about ggplot is adding layers to the existing plots to modify behavior

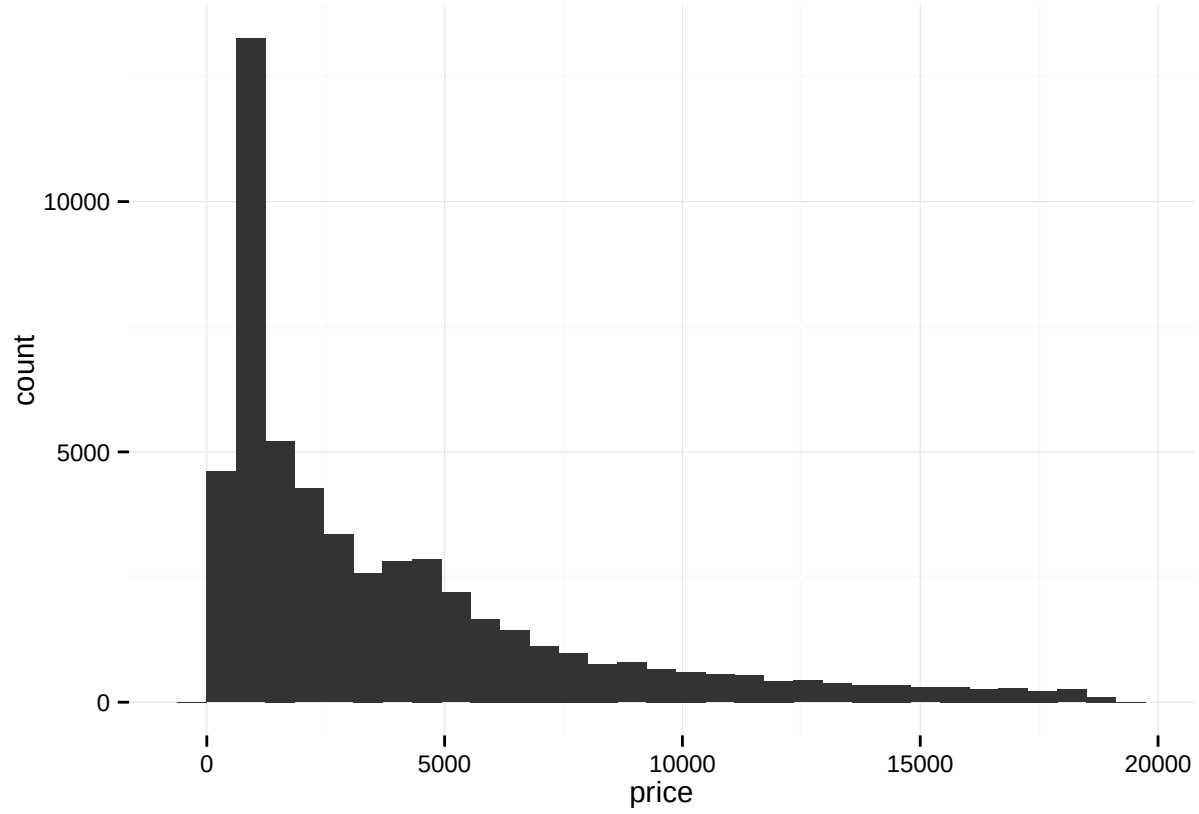
```
qplot(x = carat, data = diamonds, binwidth = 0.25) +
    scale_x_continuous(limits = c(0.25,1.5)) +
    facet_wrap(~cut, ncol = length(levels(diamonds$cut)))
```



Transformation

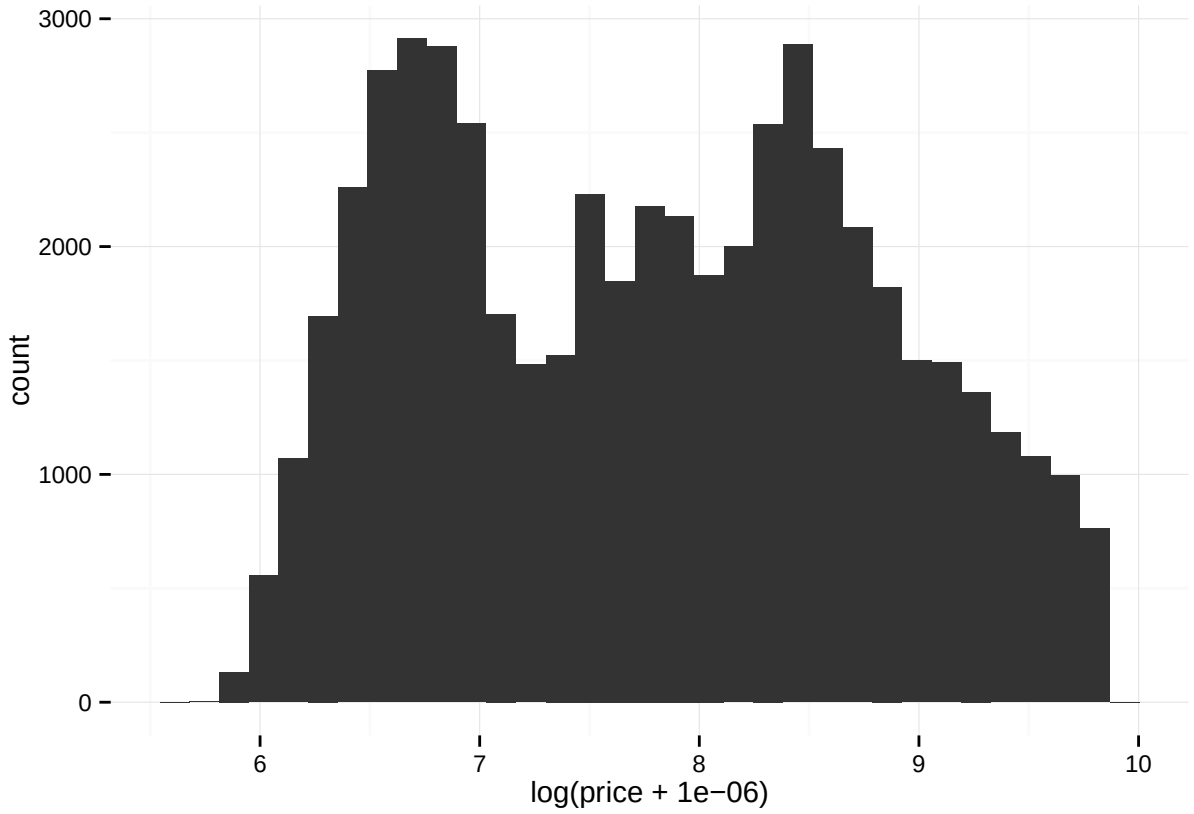
Originally low tail distributed data

```
qplot(x = price, data = diamonds)
```



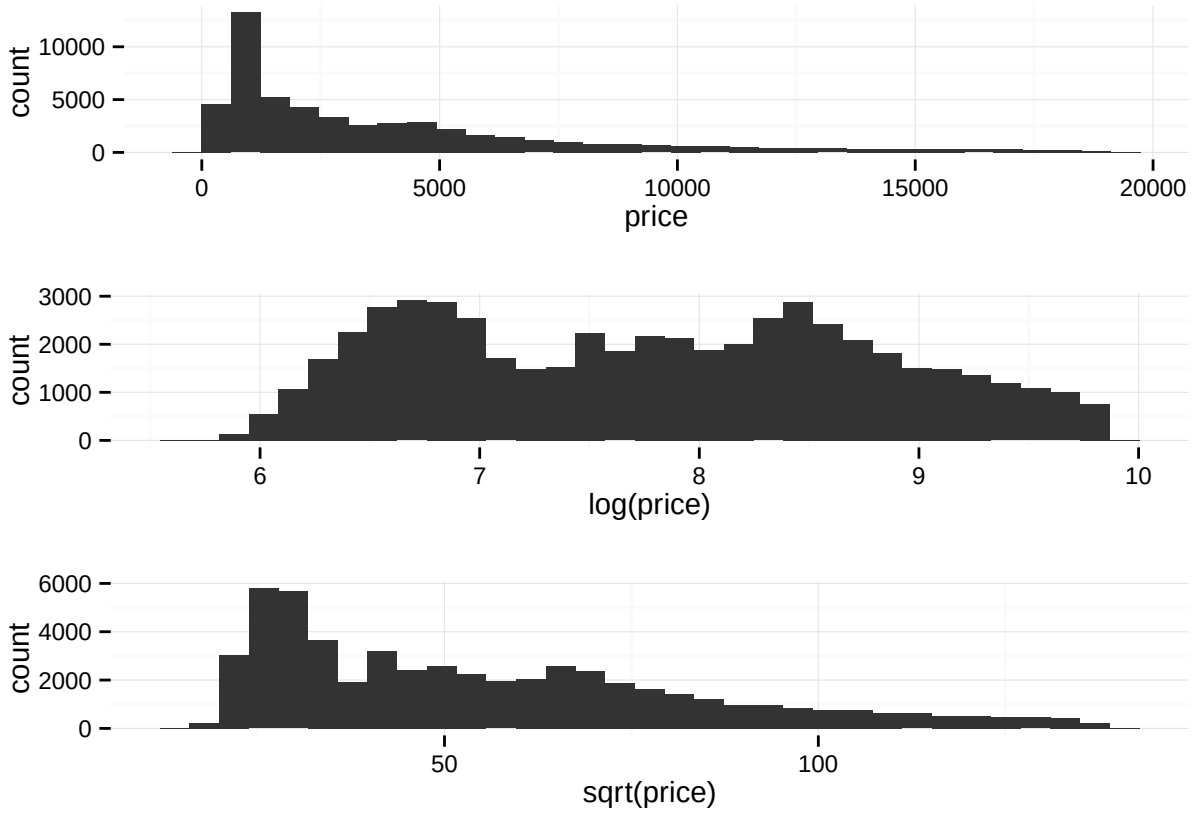
Log transformation

```
qplot(x = log(price + 0.000001), data  = diamonds)
```



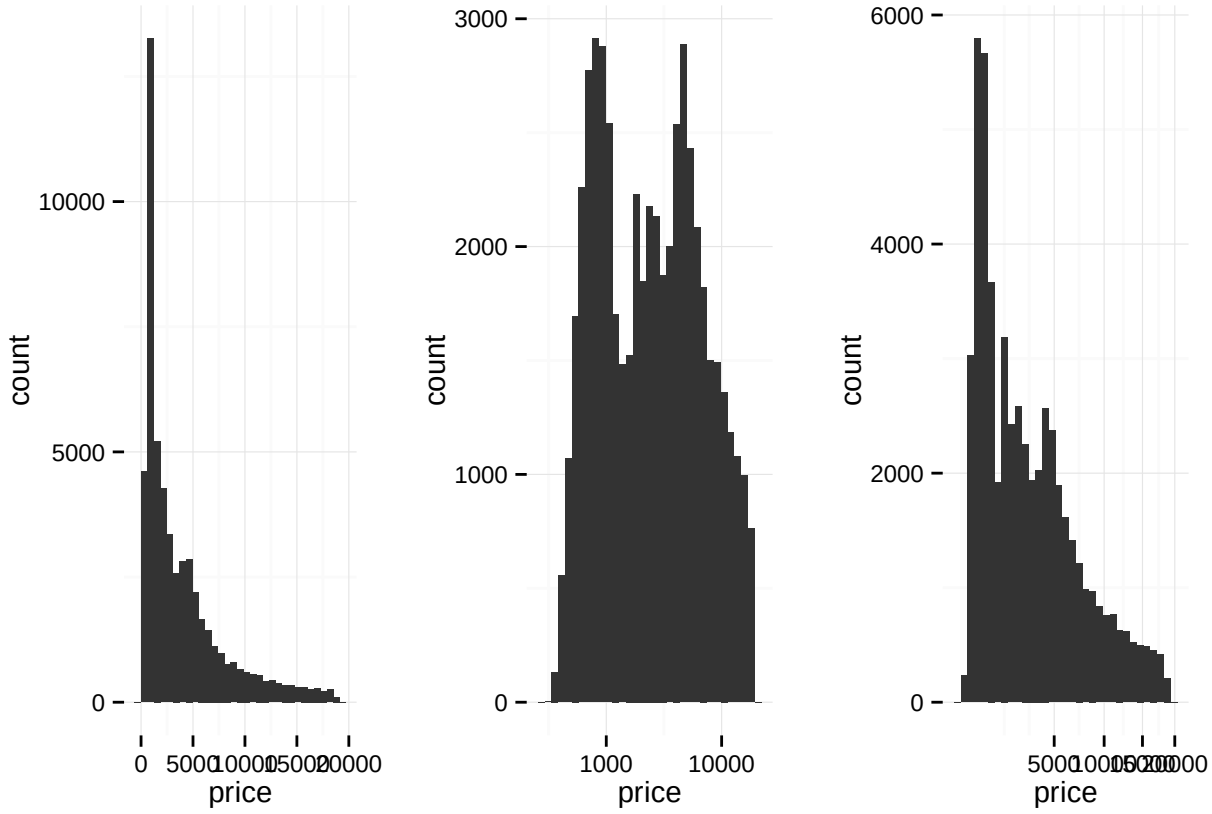
Save plot into a variable and then use `gridExtra` to show multiple plots

```
p1 <- qplot(x = price, data = diamonds)
p2 <- qplot(x = log(price), data = diamonds)
p3 <- qplot(x = sqrt(price), data = diamonds)
grid.arrange(p1,p2,p3,ncol = 1, nrow = 3)
```



Actually... more ggplot way of doing this is.....
Add a layer!

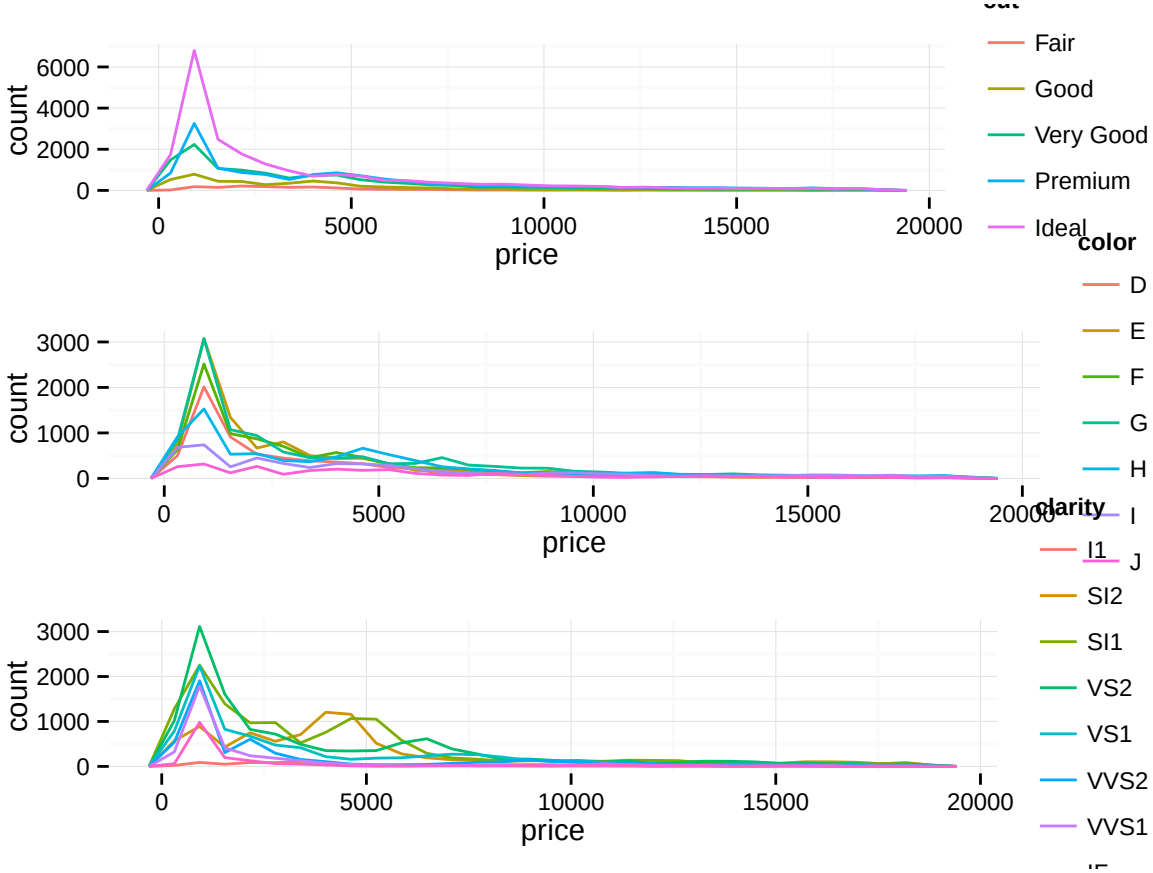
```
p2 <- qplot(x = price, data = diamonds) + scale_x_log10()
p3 <- qplot(x = price, data = diamonds) + scale_x_sqrt()
grid.arrange(p1,p2,p3,ncol = 3, nrow = 1)
```



Introducing the goem

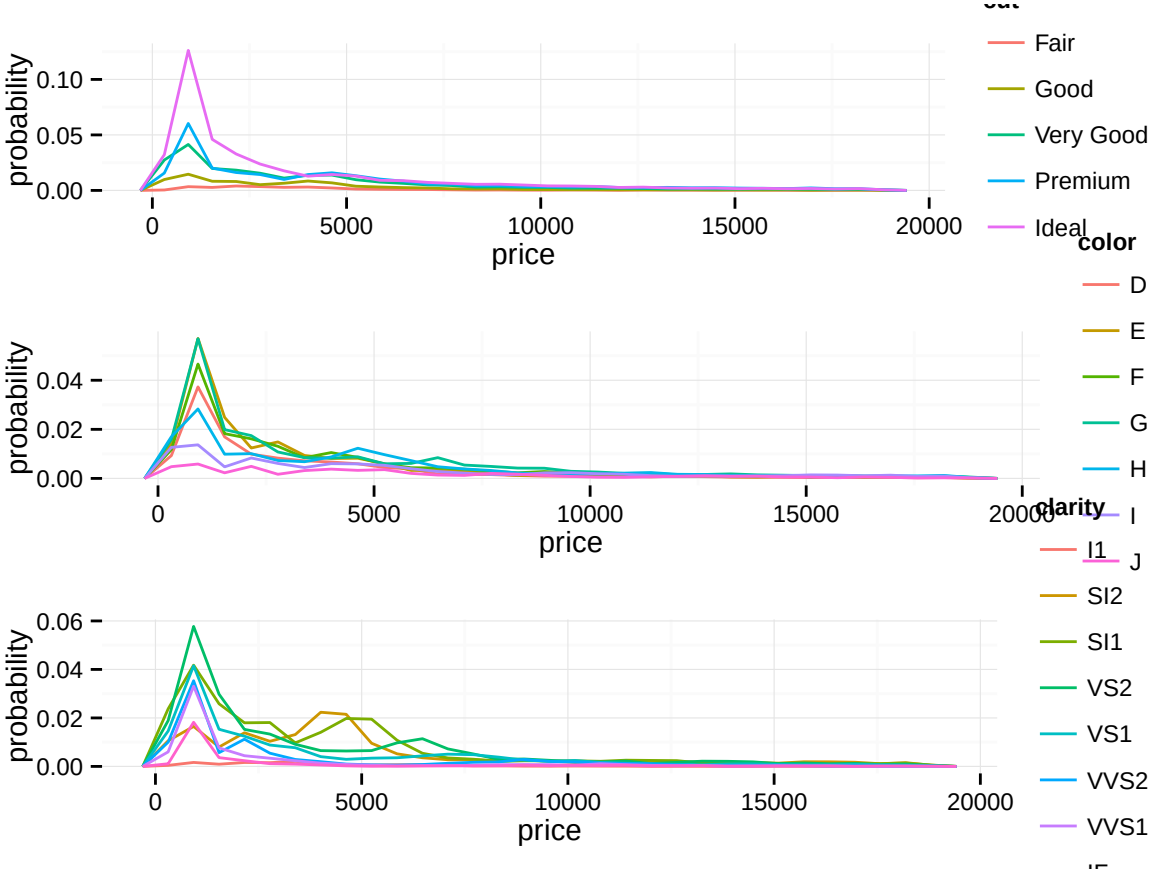
Tired of histograms. . . .? Me too
Let's look at different geoms

```
pcut <- qplot(x = price, data = diamonds, geom = 'freqpoly', color = cut)
pcolor <- qplot(x = price, data = diamonds, geom = 'freqpoly', color = color)
pclarity <- qplot(x = price, data = diamonds, geom = 'freqpoly', color = clarity)
grid.arrange(pcut, pcolor, pclarity, nrow =3 )
```



almost look like CDFs, except the y axis...
warning!!! strange looking code ahead...

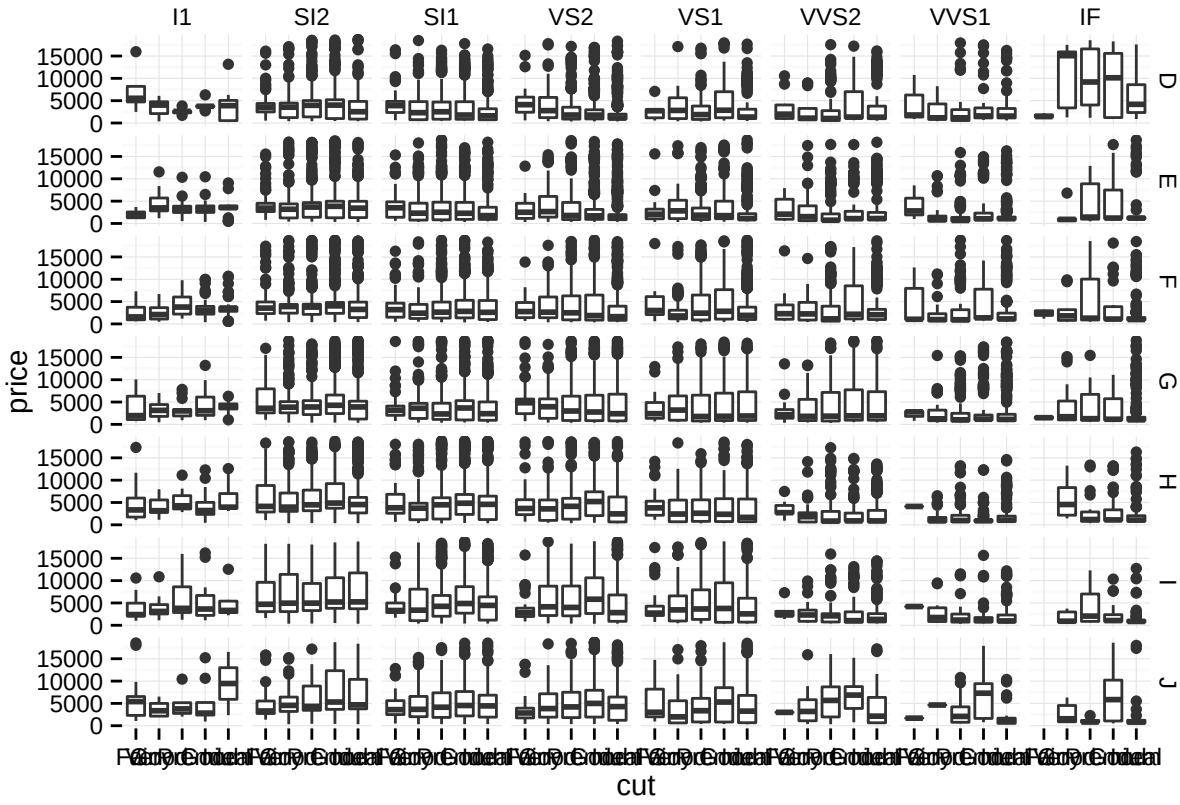
```
pcut <- qplot(x = price, y = ..count../sum(..count..), data = diamonds, geom = 'freqpoly', color = cut,
pcolor <- qplot(x = price, y = ..count../sum(..count..), data = diamonds, geom = 'freqpoly', color = co
pclarity <- qplot(x = price, y = ..count../sum(..count..), data = diamonds, geom = 'freqpoly', color =
grid.arrange(pcut, pcolor, pclarity, nrow =3 )
```



Box plots

Or this is to test if you need a new laptop or not...

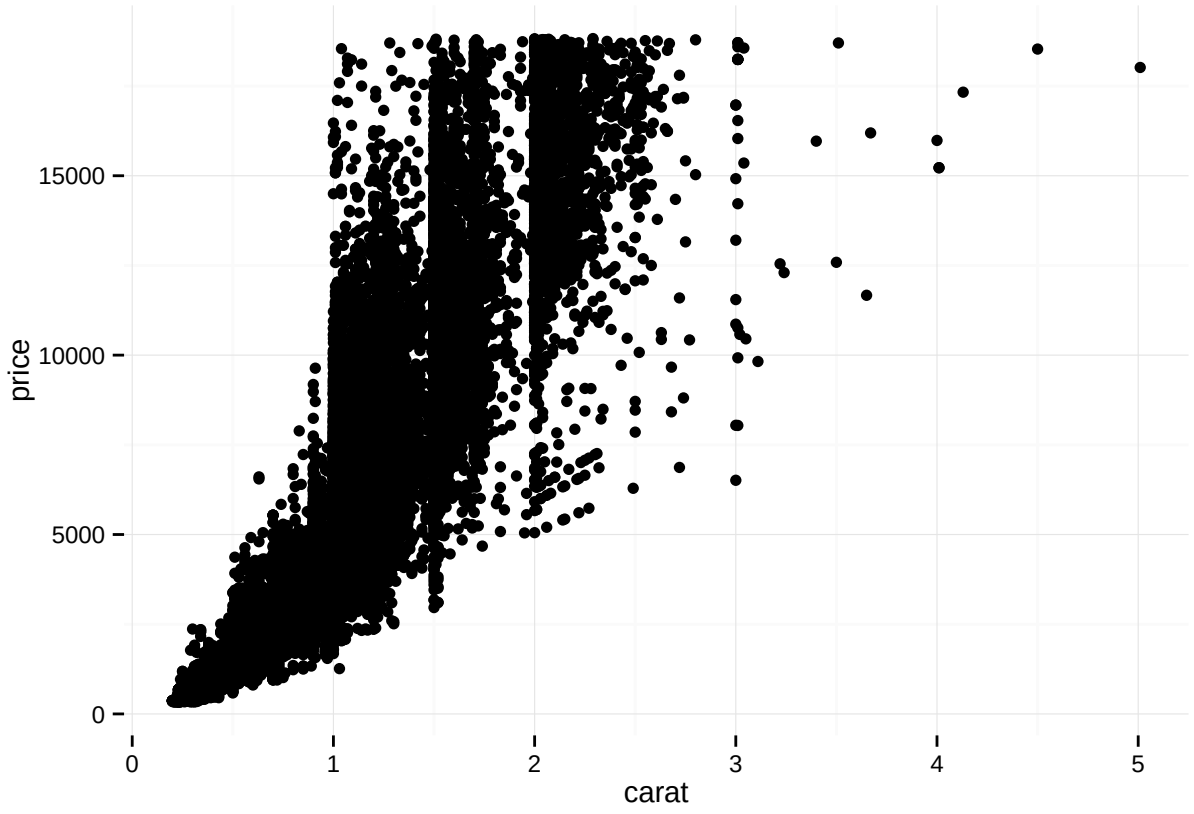
```
qplot(x = cut, y = price, data = diamonds, geom = 'boxplot') +
    facet_grid(color~clarity)
```



Scatter plot

Regression works?

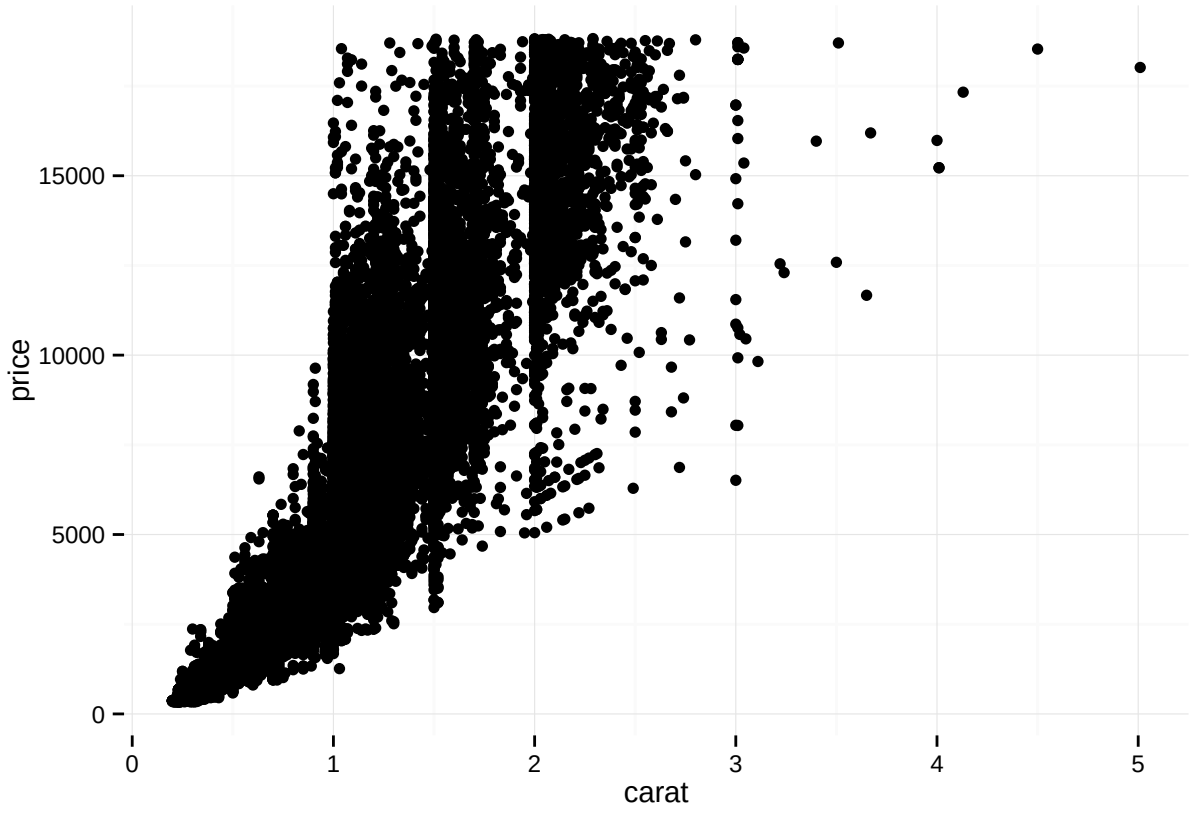
```
qplot(x = carat, y = price, data = diamonds)
```



We should start using `ggplot()` function now

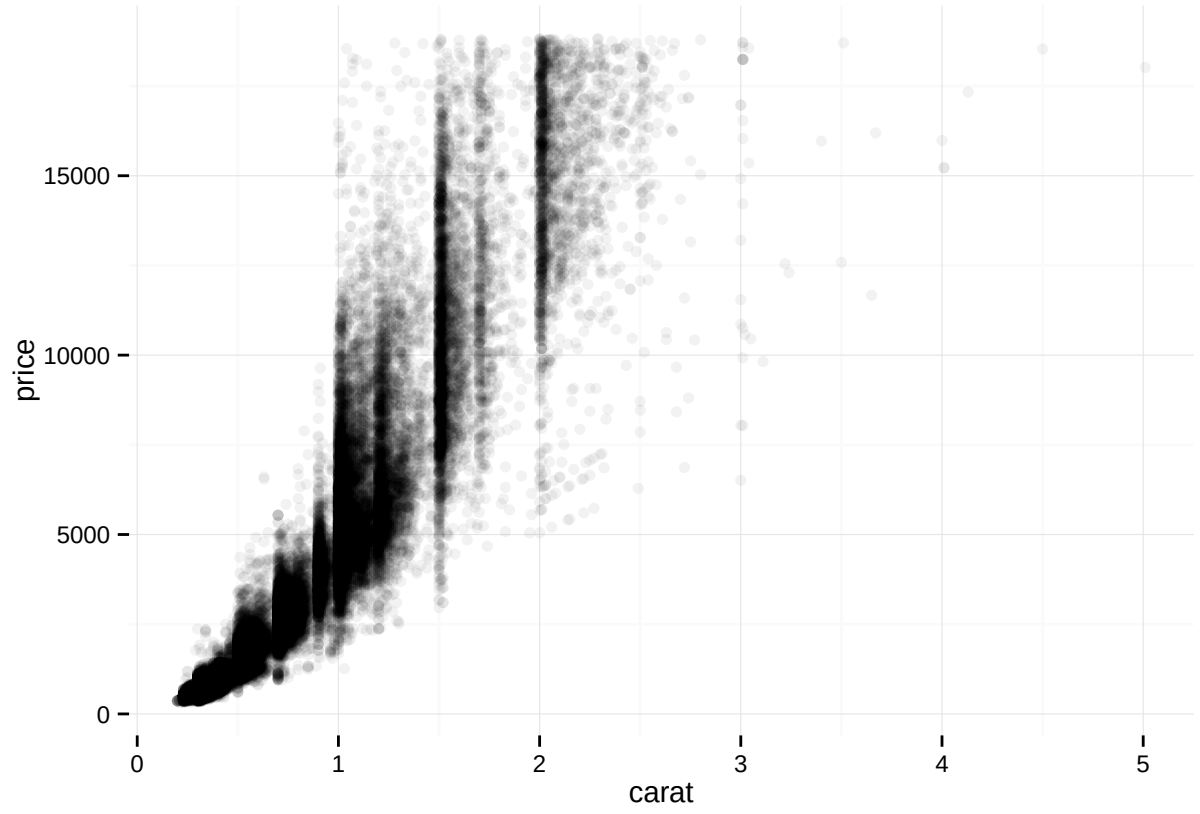
This is equvalent to the `qplot()` function we been using until now
But this syntax show the layer on top of layer concept in ggplot more clearly

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



Add some transparency

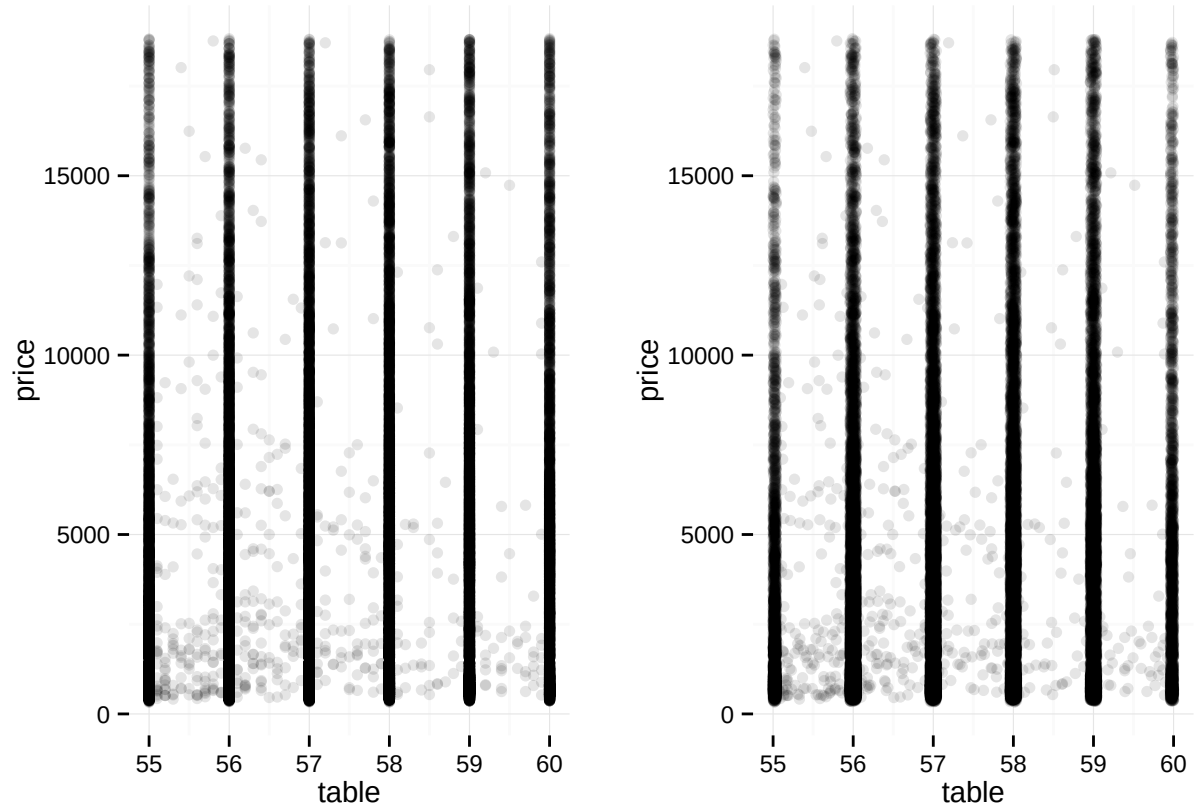
```
ggplot(aes(x= carat , y = price), data = diamonds) +
    geom_point(alpha = 0.05)
```



Add jitter

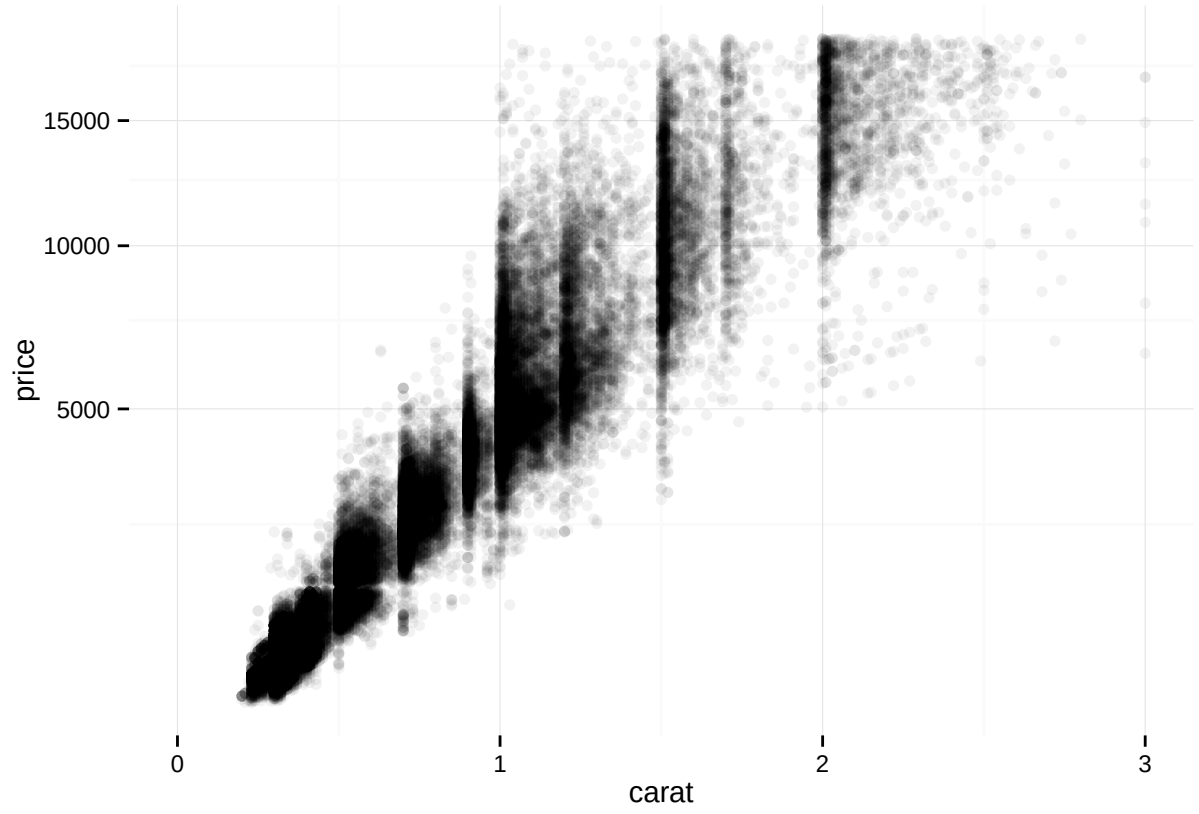
Trying to find a case to show the need for this....
But maybe not so good with this dataset

```
p1 <- ggplot(aes(x= table , y = price), data = diamonds) +
    geom_point(alpha = 0.1) +
    xlim(55,60)
p2 <- ggplot(aes(x= table , y = price), data = diamonds) +
    geom_jitter(alpha = 0.1) +
    xlim(55,60)
grid.arrange(p1,p2, ncol = 2)
```



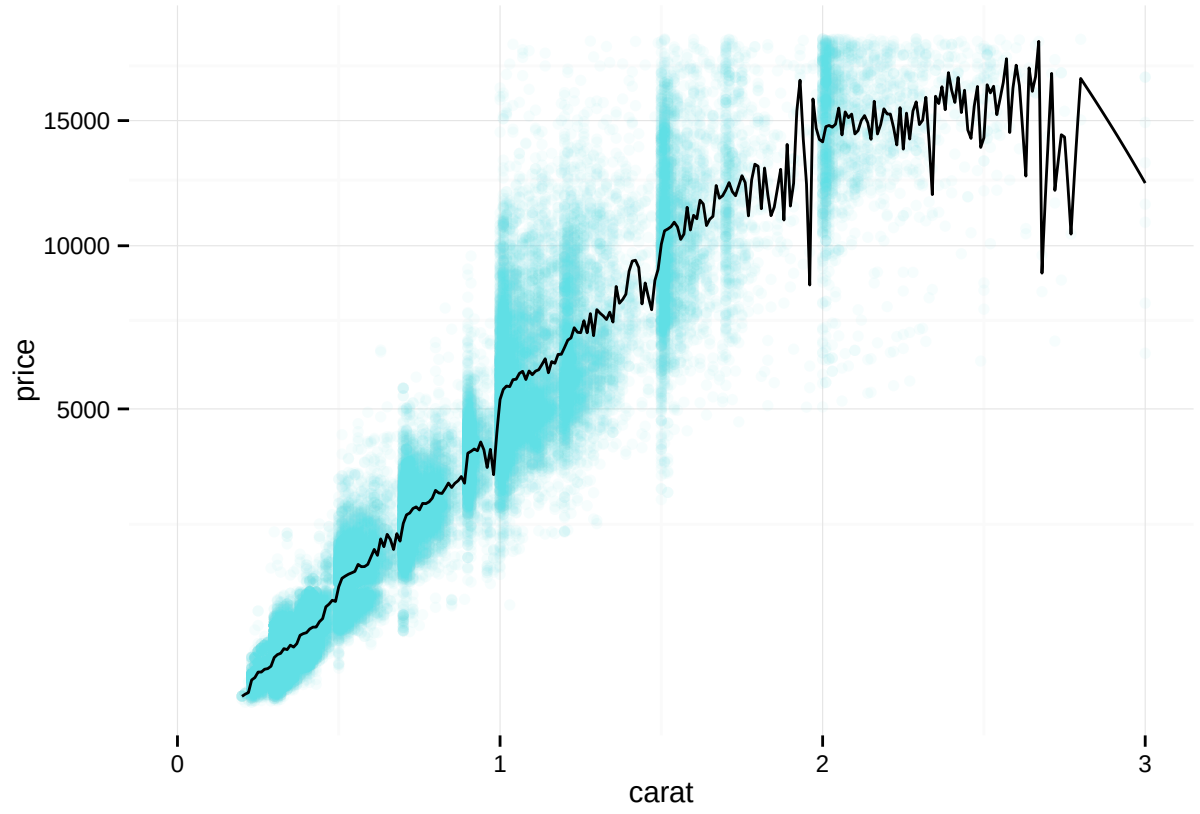
Transform using layer

```
ggplot(aes(x= carat , y = price), data = diamonds) +
    geom_point(alpha = 0.05) +
    coord_trans( y = 'sqrt' ) +
    xlim(0,3)
```



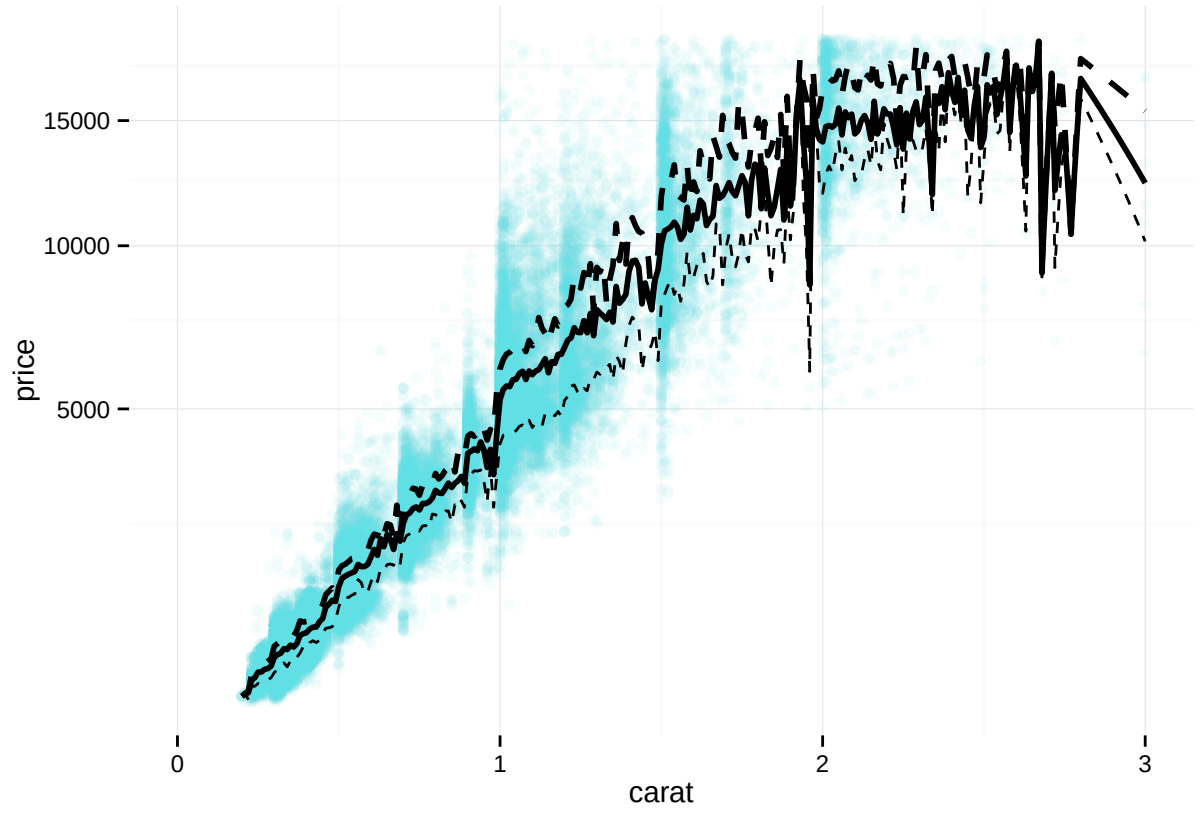
Conditional Mean

```
ggplot(aes(x= carat , y = price), data = diamonds) +
    geom_point(alpha = 0.05, color = I('#60DFE5')) +
    coord_trans( y = 'sqrt' ) +
    xlim(0,3) +
    geom_line(stat = 'summary', fun.y = mean)
```



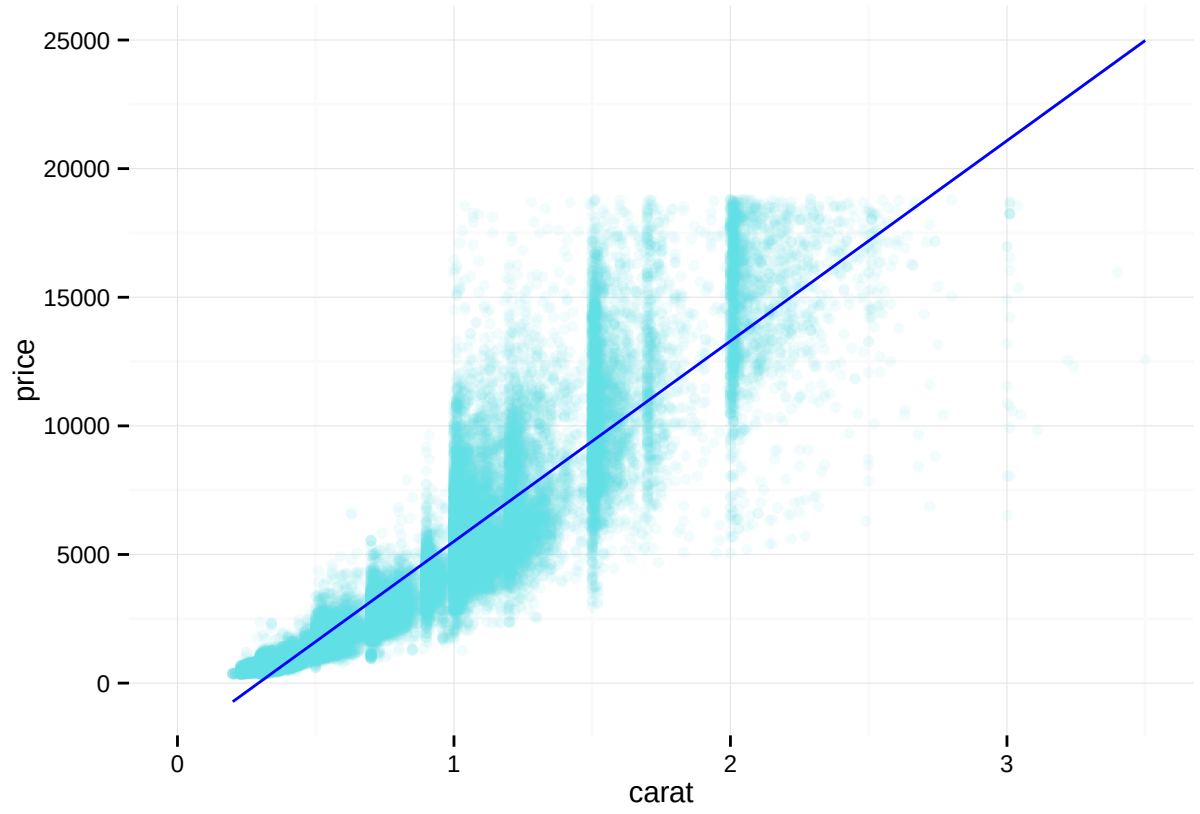
Quantiles as well

```
ggplot(aes(x= carat , y = price), data = diamonds) +
    geom_point(alpha = 0.05, color = I('#60DFE5')) +
    coord_trans( y = 'sqrt' ) +
    xlim(0,3) +
    geom_line(size = 1, stat = 'summary', fun.y = mean) +
    geom_line(siz3 = 1, linetype = 2, stat = 'summary', fun.y = quantile, probs = 0.25) +
    geom_line(size = 1, linetype = 2, stat = 'summary', fun.y = quantile, probs = 0.75)
```



Smoothed Conditional Mean

```
ggplot(aes(x= carat , y = price), data = diamonds) +
    geom_point(alpha = 0.08, color = I('#60DFE5')) +
    xlim(0,3.5) +
    stat_smooth(method = "gam" , color = I('blue'))
```

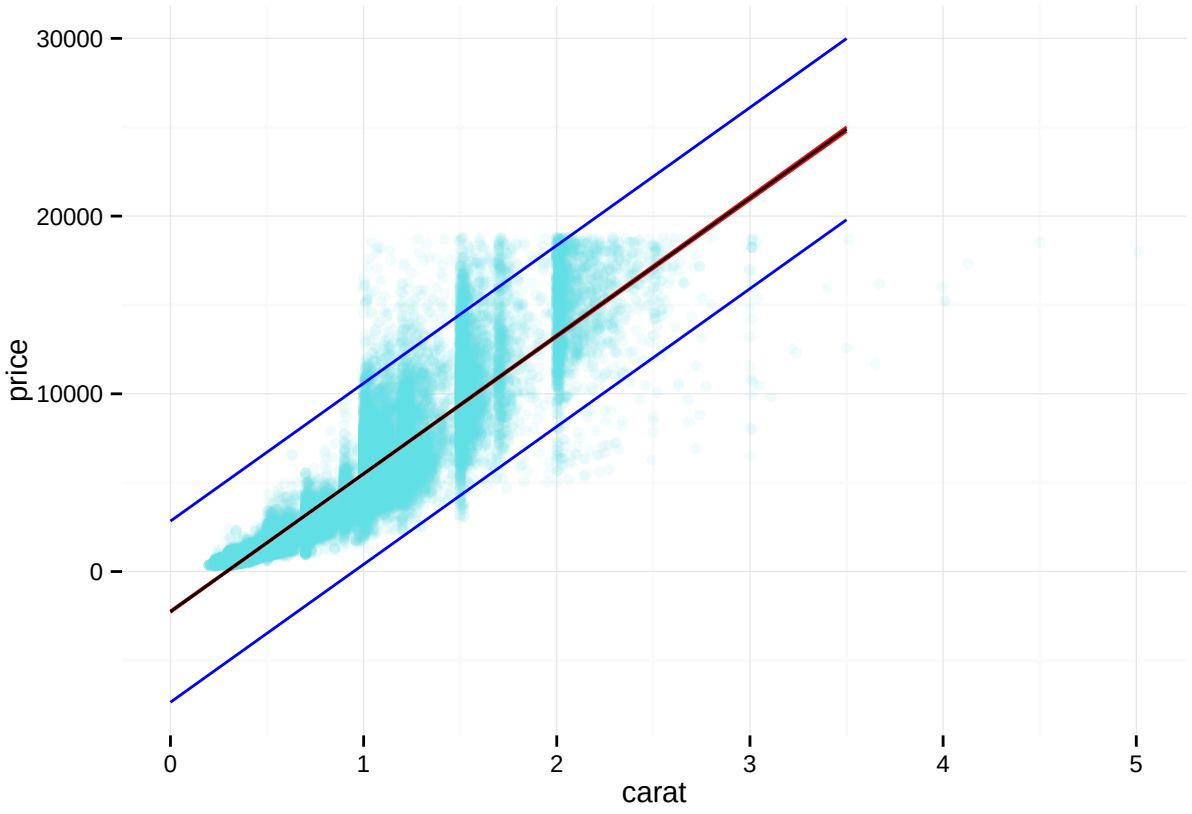


Some stats by hand

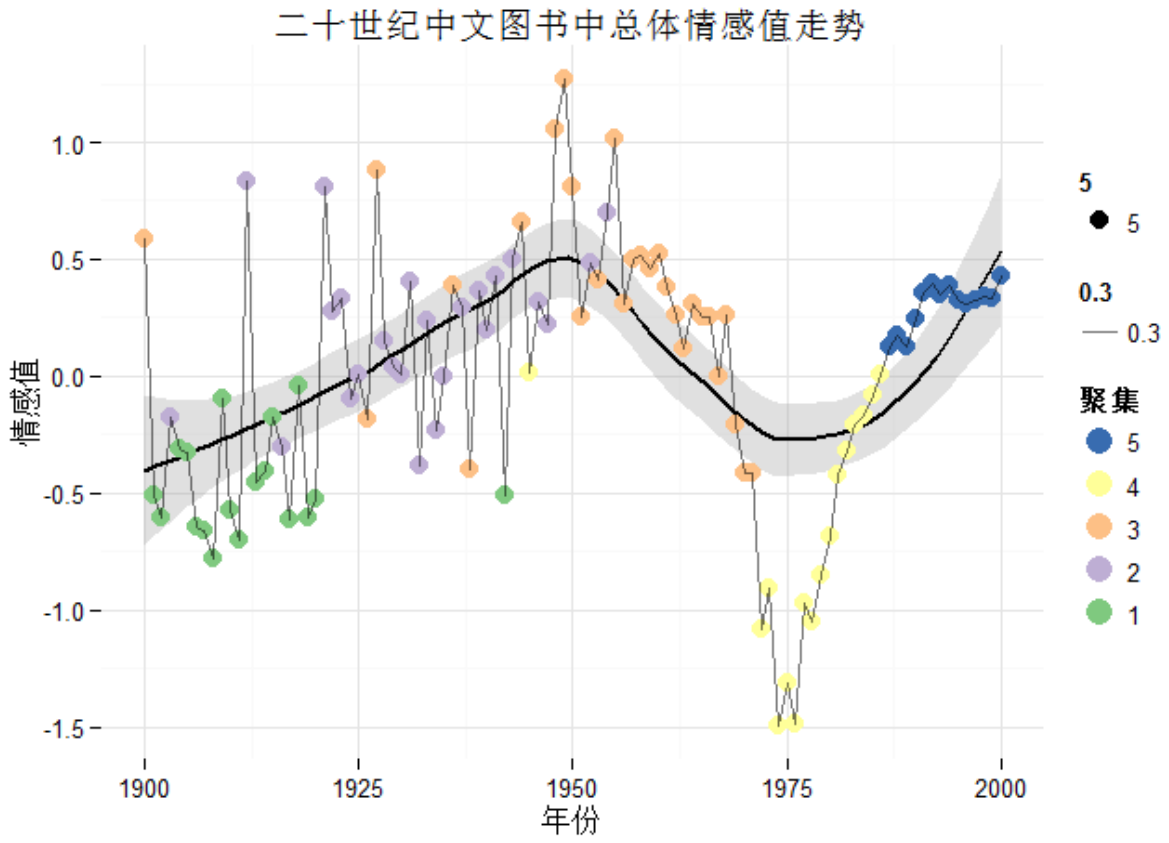
```
model <- lm(price~carat, data = diamonds)
linspace <-  seq(0, 3.5, 0.01)
ci <- predict(model, data.frame(carat = linspace), interval = "confidence", level = 0.999)
pi <- predict(model, data.frame(carat = linspace), interval = "prediction", level = 0.999)
mu_hat <- ci[,1]
```

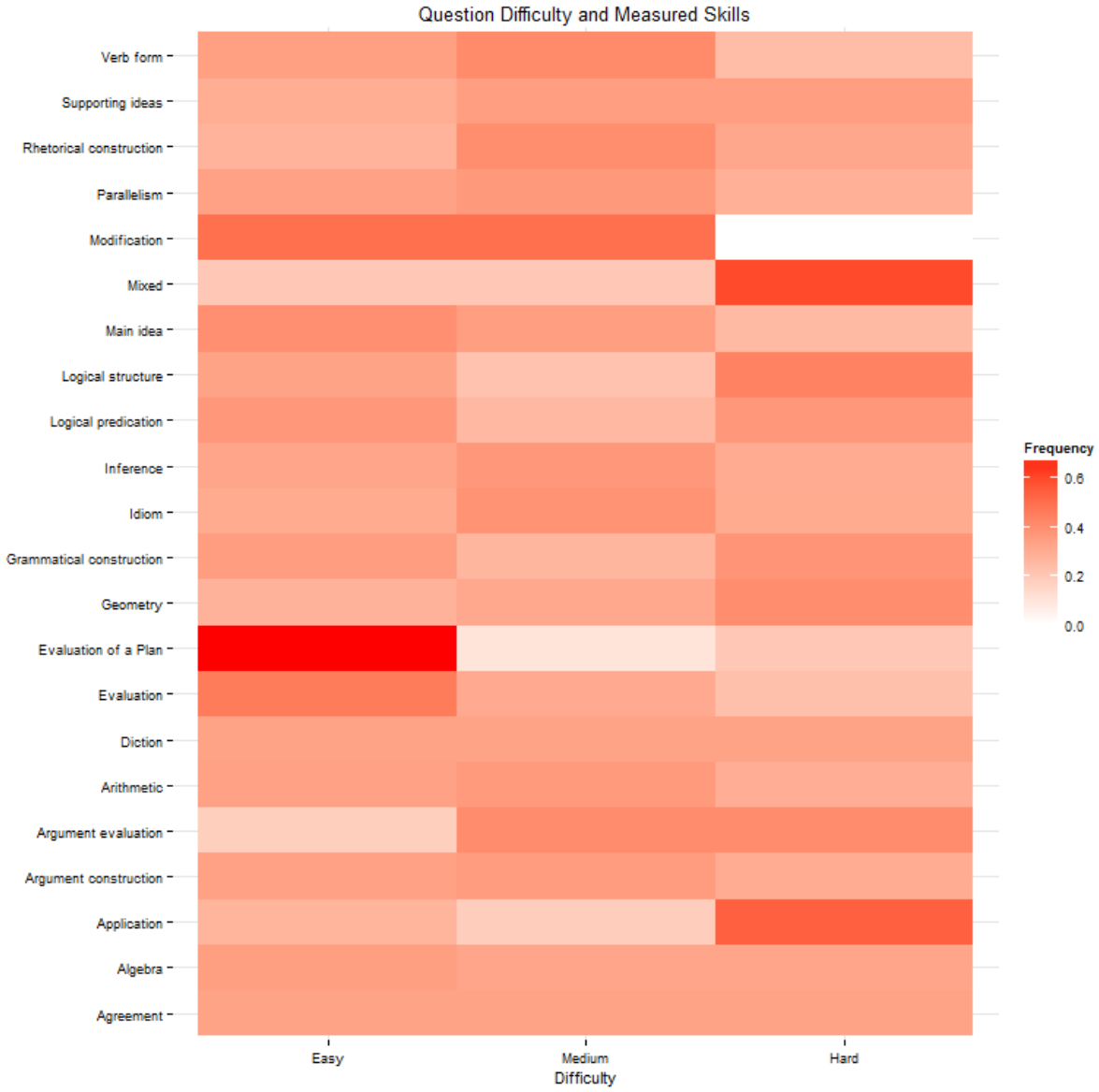
Explain?....

```
ggplot(aes(x= carat , y = price), data = diamonds) +
    geom_point(alpha = 0.08, color = I('#60DFE5')) +
    geom_line(data = data.frame(carat = linspace, price = ci[,2]), color = 'red') +
    geom_line(data = data.frame(carat = linspace, price = ci[,3]), color = 'red') +
    geom_line(data = data.frame(carat = linspace, price = pi[,2]), color = 'blue') +
    geom_line(data = data.frame(carat = linspace, price = pi[,3]), color = 'blue') +
    geom_line(data = data.frame(carat = linspace, price = mu_hat), color = 'black')
```


30000
20000
10000
0
price
0
1
2
3
4
5
carat

Some Plots I did in the past





You can use ggplot in python too

Yhat

Where to find examples

- Books:
  1.R Graphics Cookbook
  2.ggplot2: Elegant Graphics for Data Analysis

- Documentation: http://docs.ggplot2.org/current/
- CheatSheet: In our google drive