6 GGplot Plot

Sandeep Kumar

March 16, 2018

GGplot2

*Created by Hadley Wickham

*Based on the Grammer of Graphics

install.packages("ggplot2")

library(ggplot2)

Seven Layers:

1.Data

2.Aesthetics

3.Geometry

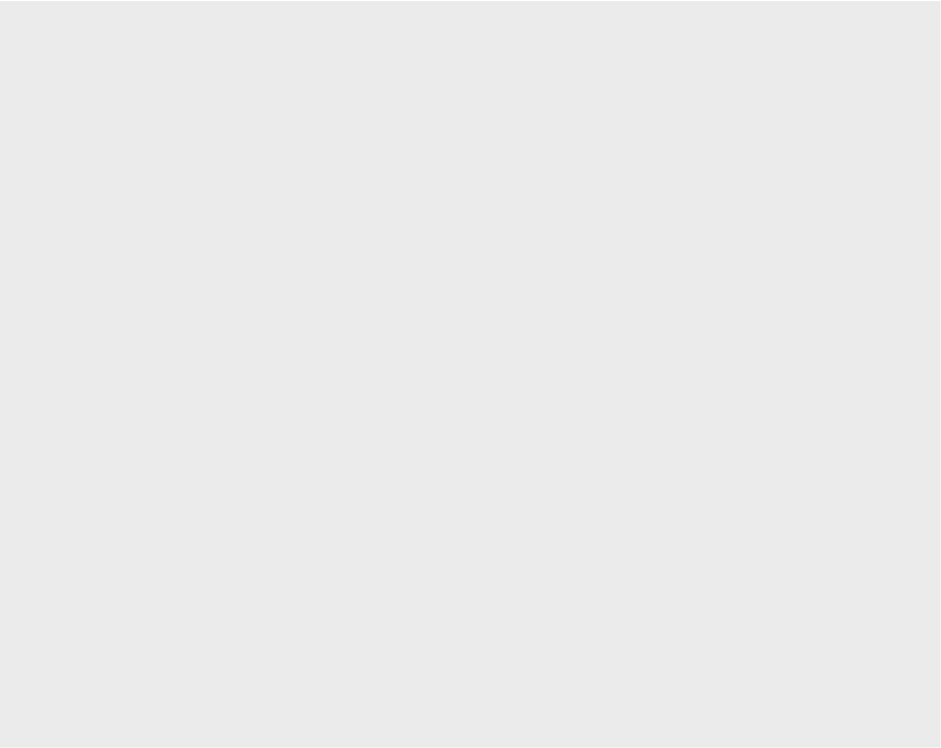
4.Facets

5.Statistics

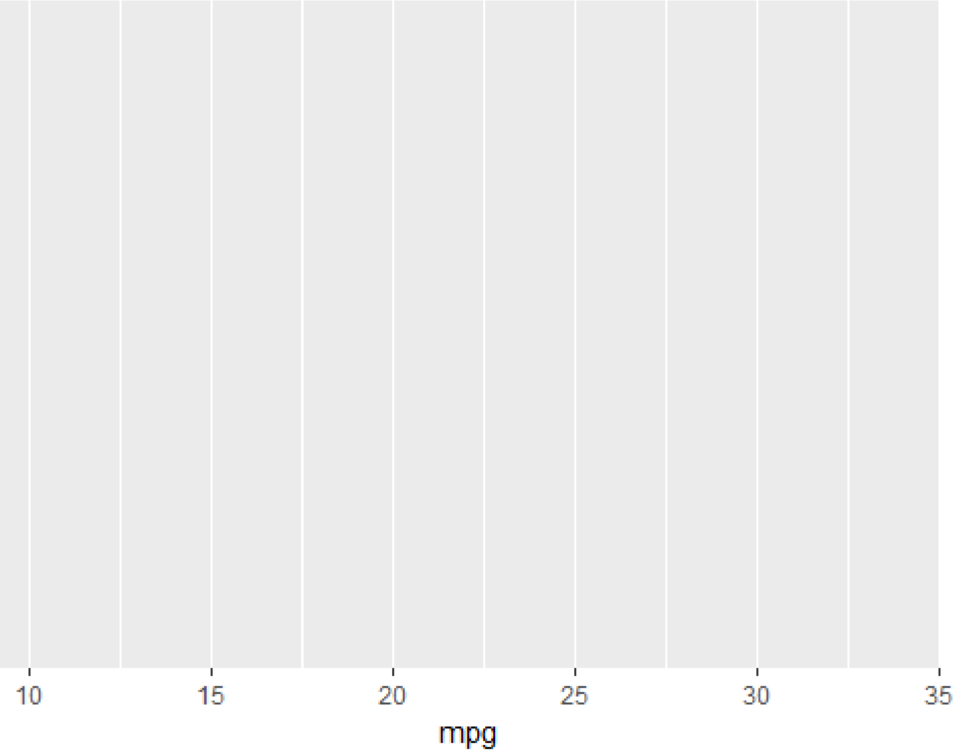
6.Coordinate system(coord)

7.Theme

```
ggplot(data = mtcars)
```

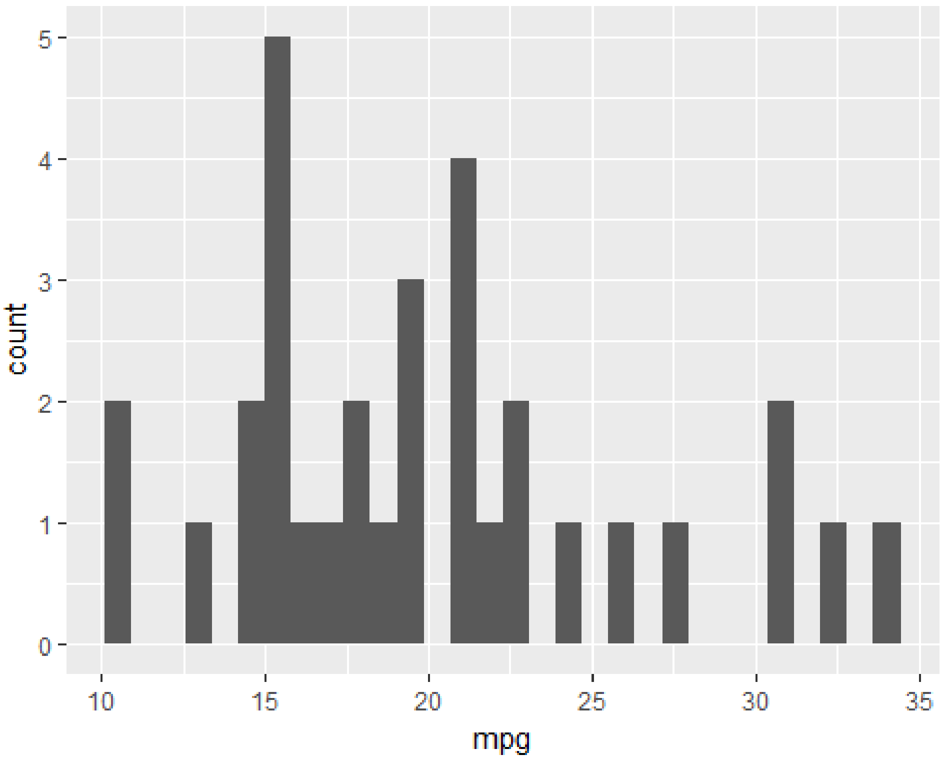


```r
ggplot(data = mtcars, aes(x = mpg))
```

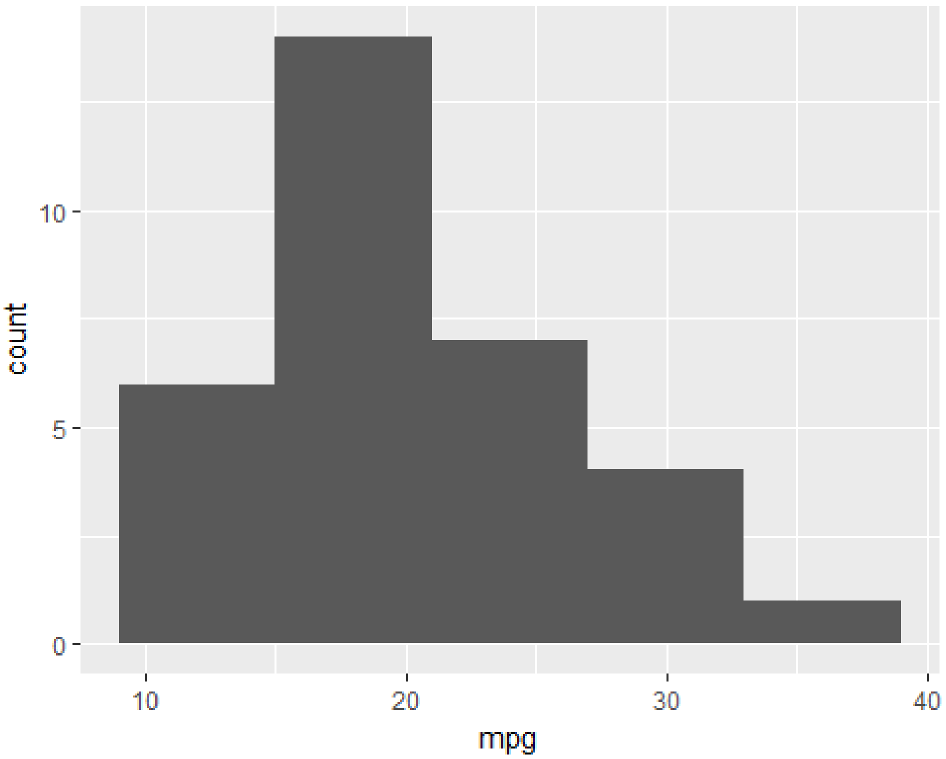


```r
ggplot(data = mtcars, aes(x = mpg)) + geom_histogram()
```

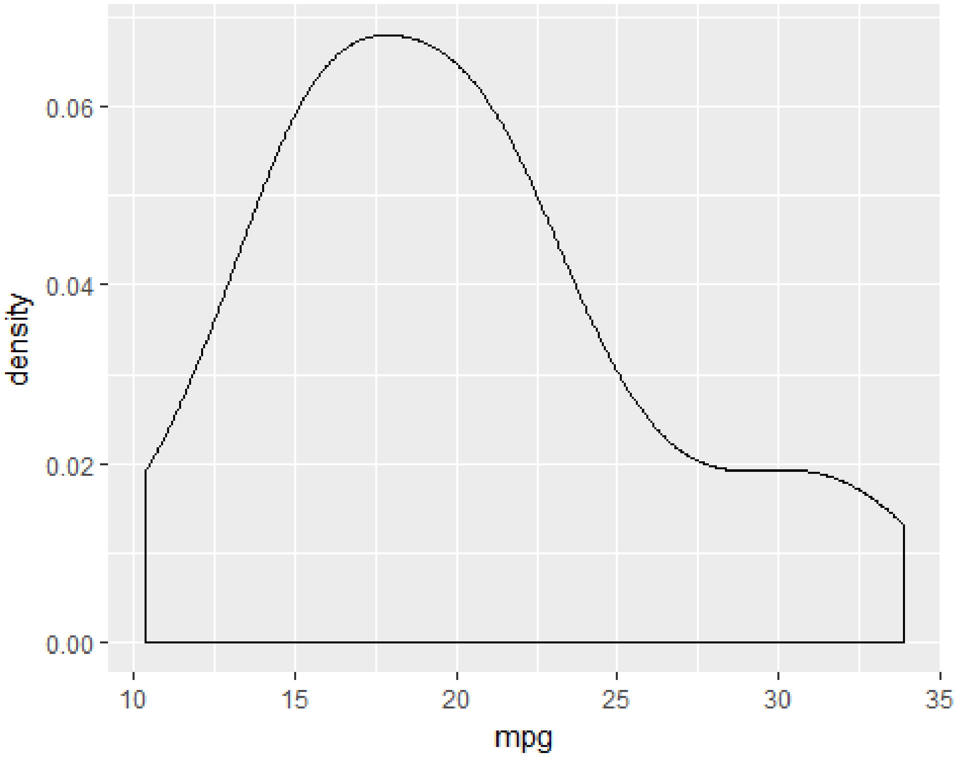
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



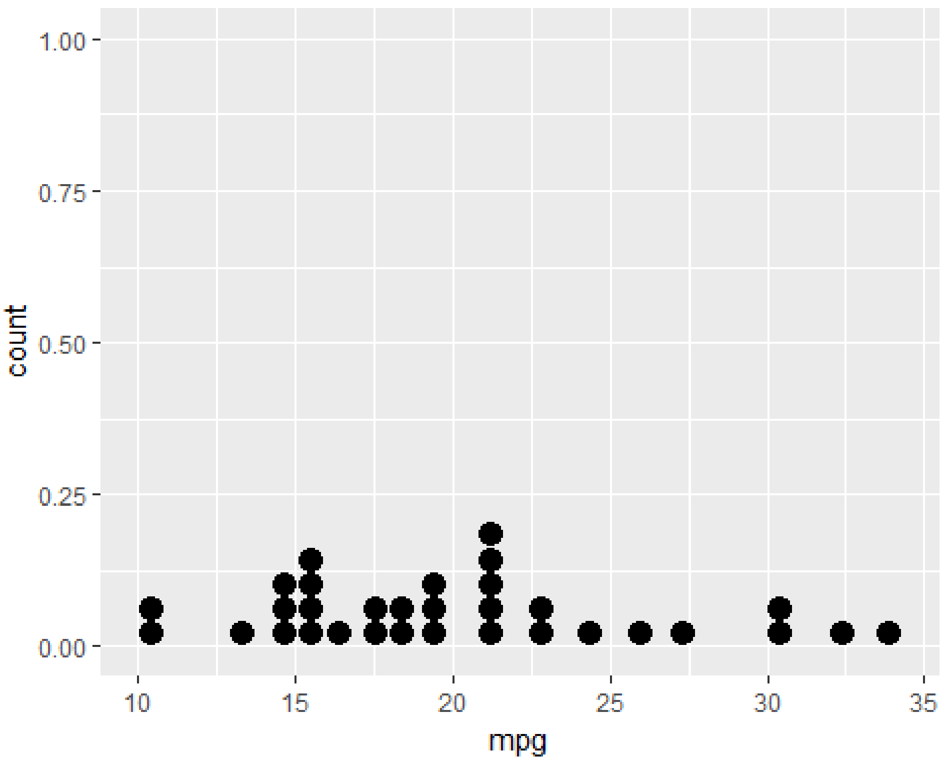
```r
ggplot(data = mtcars, aes(x = mpg)) + geom_histogram(binwidth = 6)
```



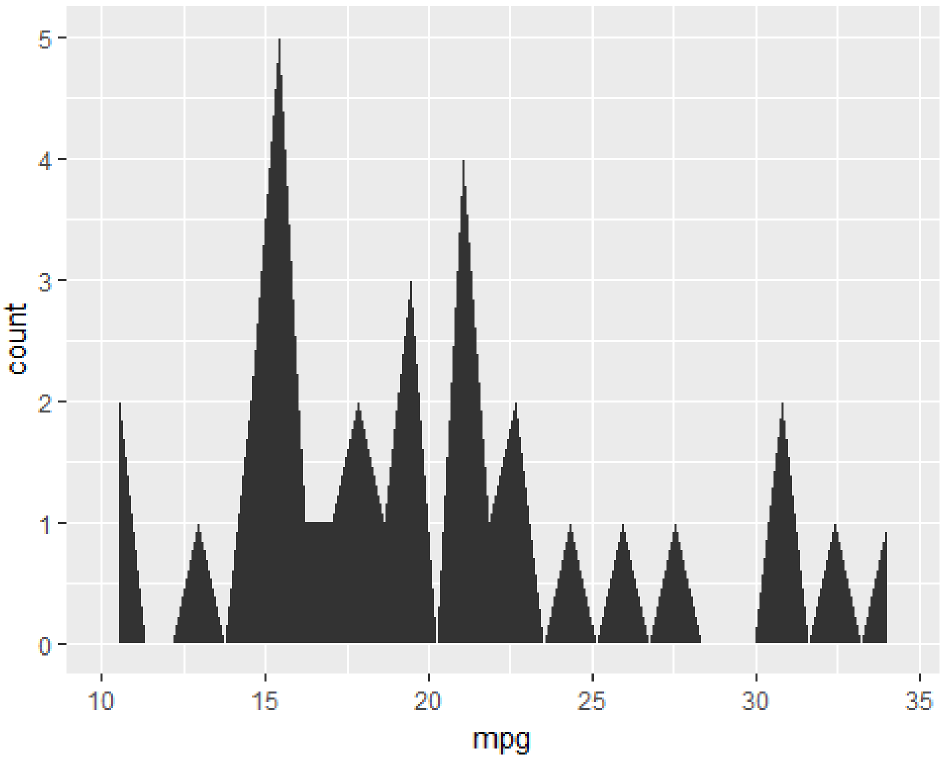
```
ggplot(data = mtcars, aes(x = mpg)) + geom_density()
```



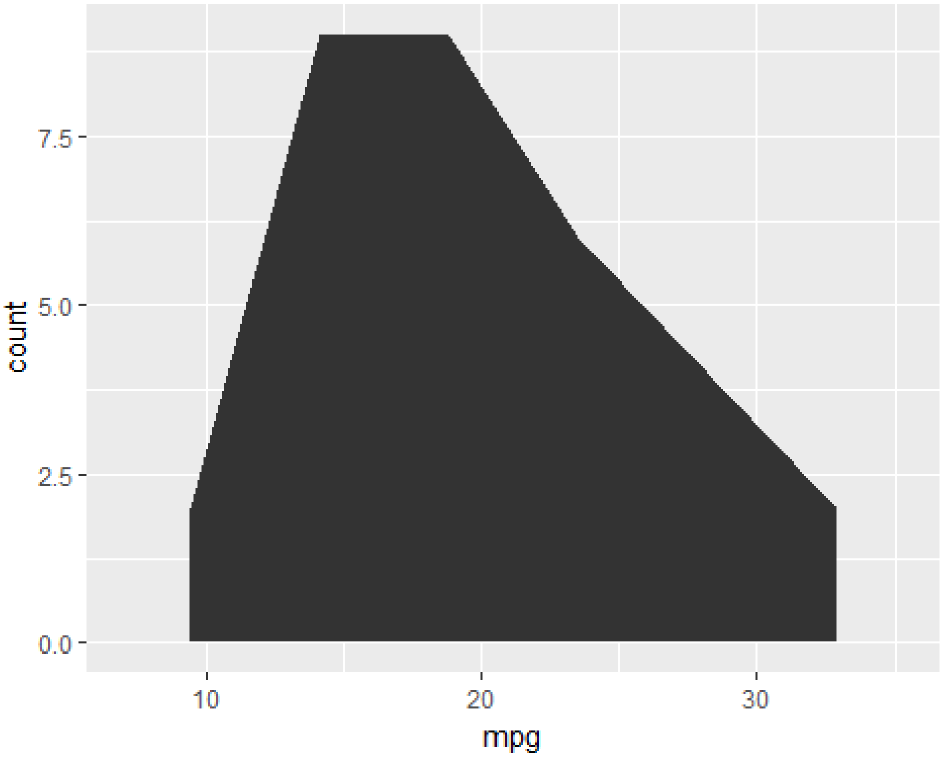
```
ggplot(data = mtcars, aes(x = mpg)) + geom_dotplot()

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



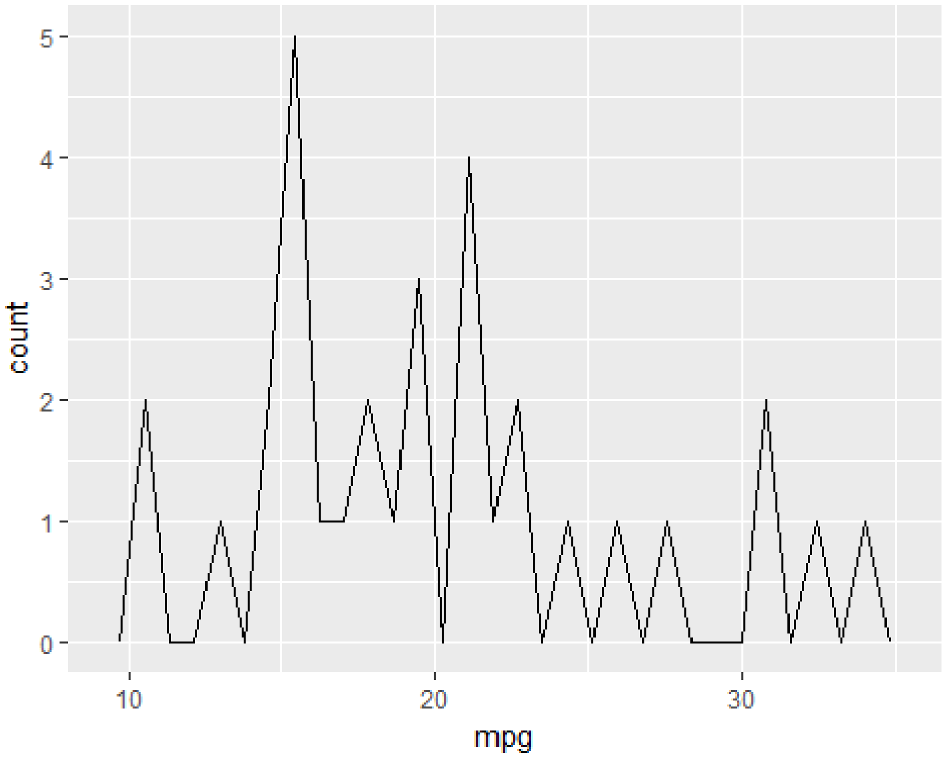
```
ggplot(data = mtcars, aes(x = mpg)) + geom_area(stat="bin")

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



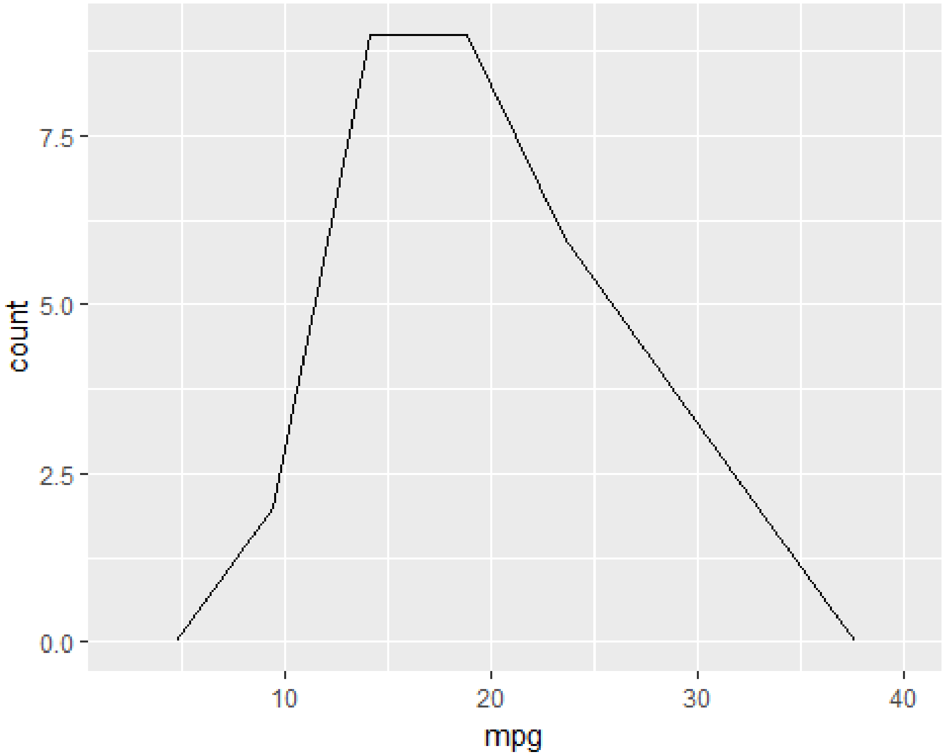
```
ggplot(data = mtcars, aes(x = mpg)) + geom_area(stat="bin", bins=6)
```



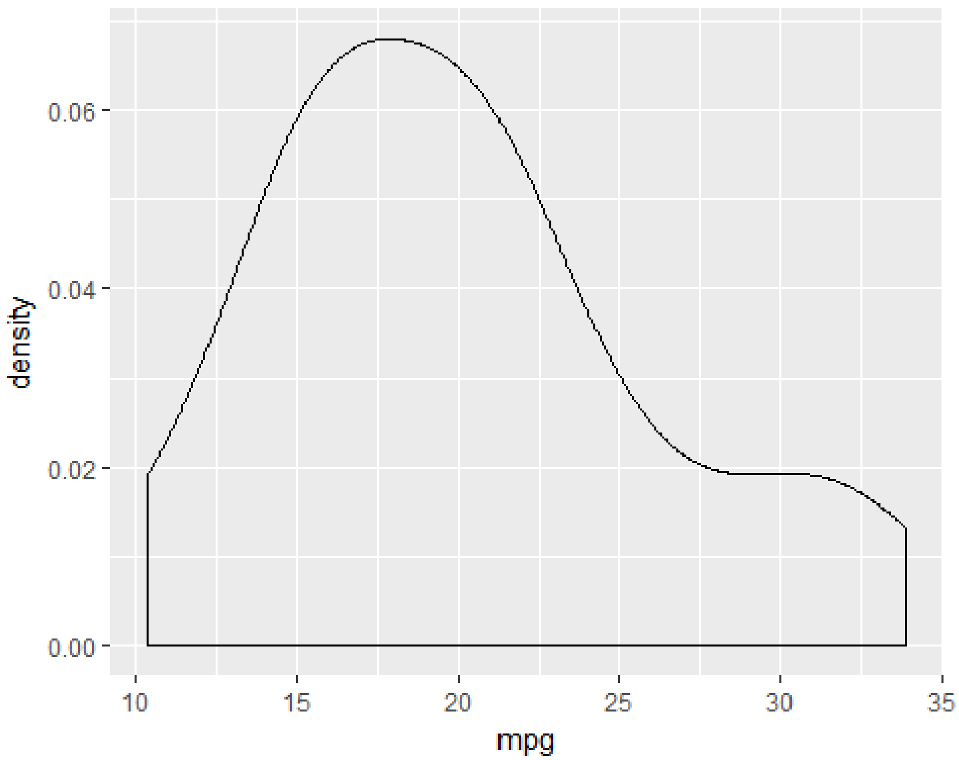
```
ggplot(data = mtcars, aes(x = mpg)) + geom_freqpoly()

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



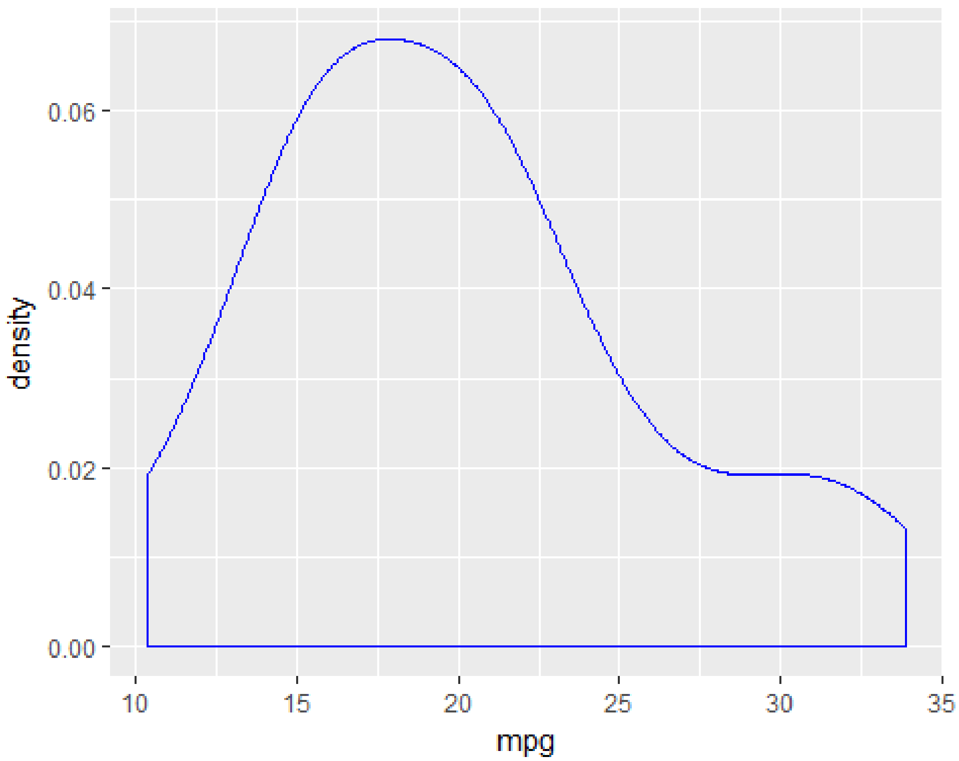
```
ggplot(data = mtcars, aes(x = mpg)) + geom_freqpoly(bins=6)
```



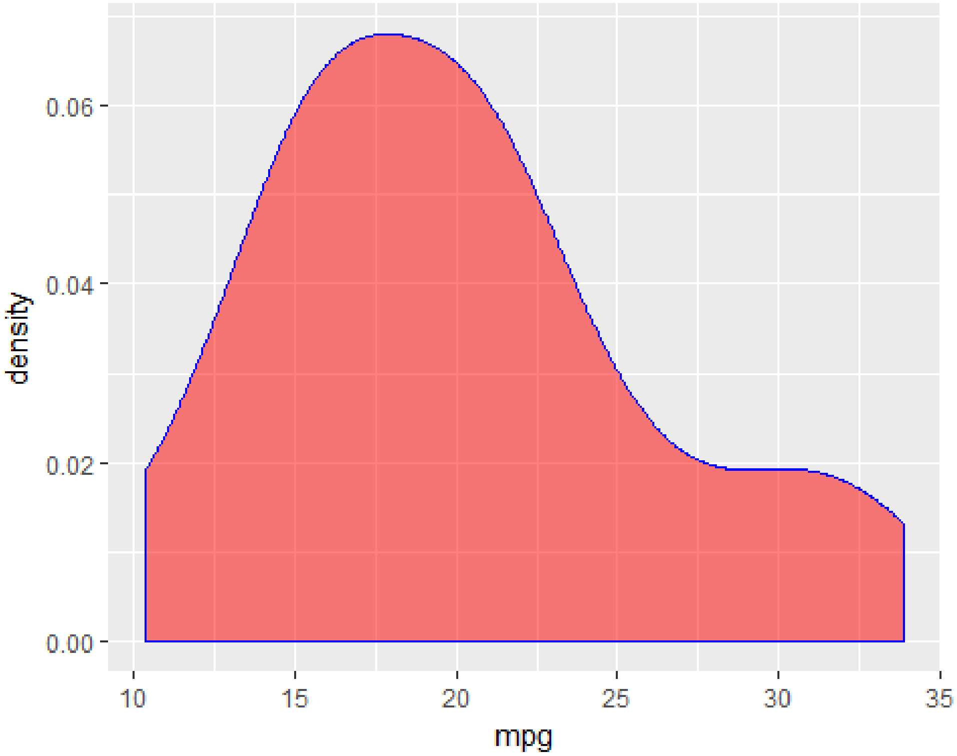
```
ggplot(data = mtcars, aes(x = mpg)) + geom_density()
```



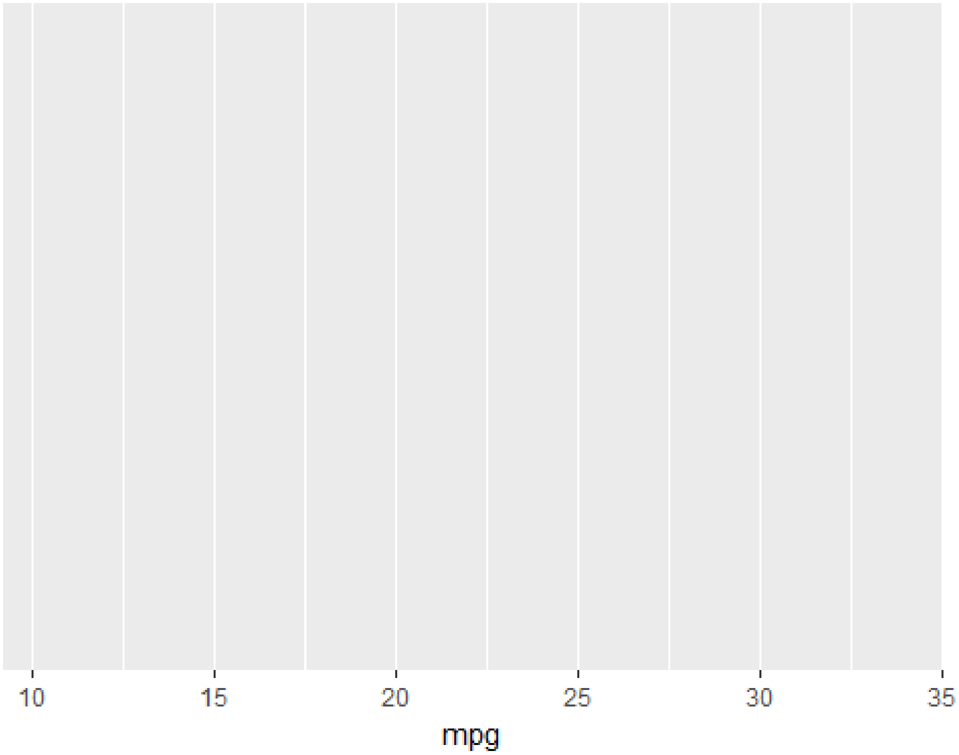
```r
ggplot(data = mtcars, aes(x = mpg)) + geom_density(colour = "blue")
```



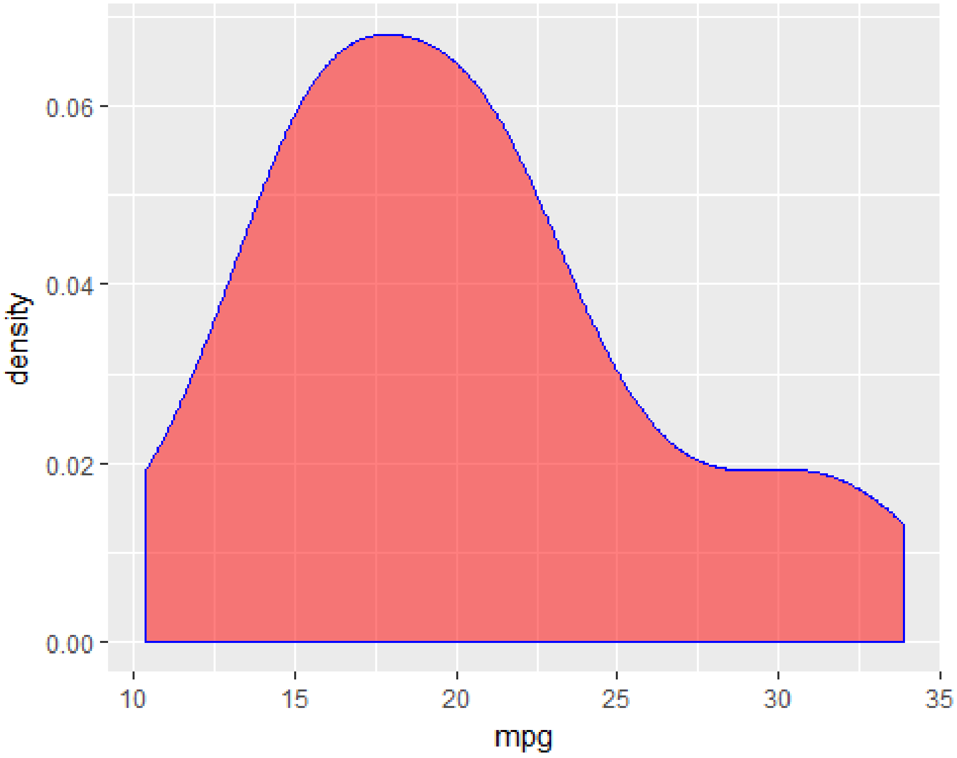
```
ggplot(data = mtcars, aes(x = mpg)) + geom_density(colour = "blue",
                                                    fill = "red", alpha = 0.5)
```



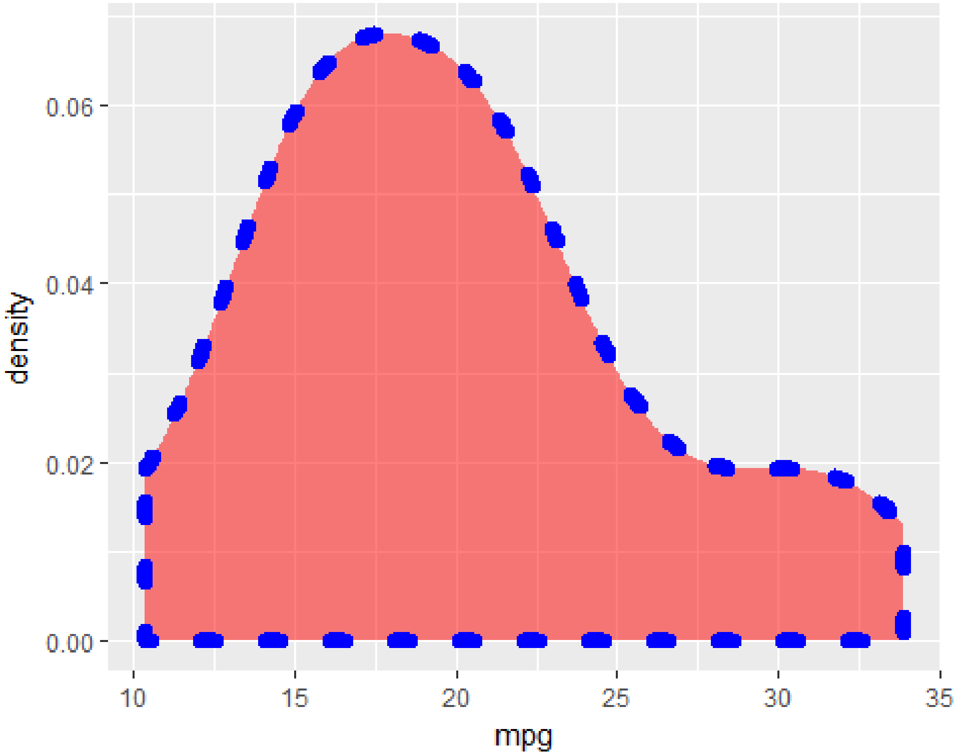
```
p <- ggplot(data = mtcars, aes(x = mpg))
p
```



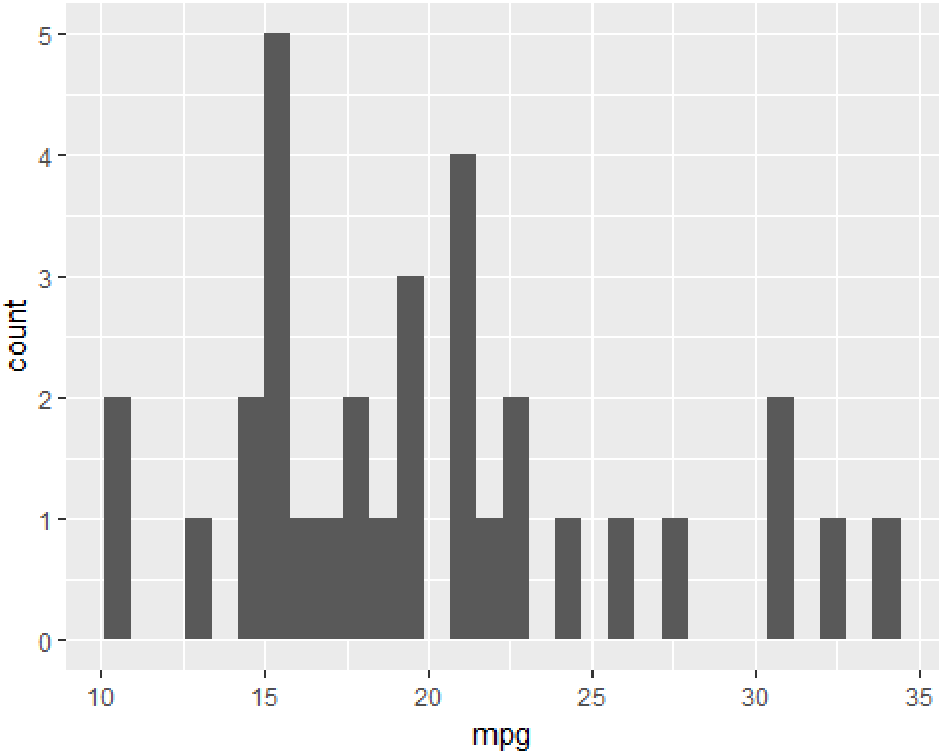
```
p + geom_density(colour = "blue",
                 fill = "red", alpha = 0.5)
```



```
p + geom_density(colour = "blue", fill = "red", alpha = 0.5,
                 linetype = "dotted", size = 3)
```



```
p + geom_histogram()

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



Diamonds dataset

diamonds dataset has following columns:

*carat - weight of the diamond (0.2--5.01)

*cut - quality of the cut (Fair, Good, Very Good, Premium, Ideal)

*color - diamond colour, from J (worst) to D (best)

*clarity - a measurement of how clear the diamond is (I1 (worst), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (best))

*x - length in mm (0--10.74)

*y - width in mm (0--58.9)

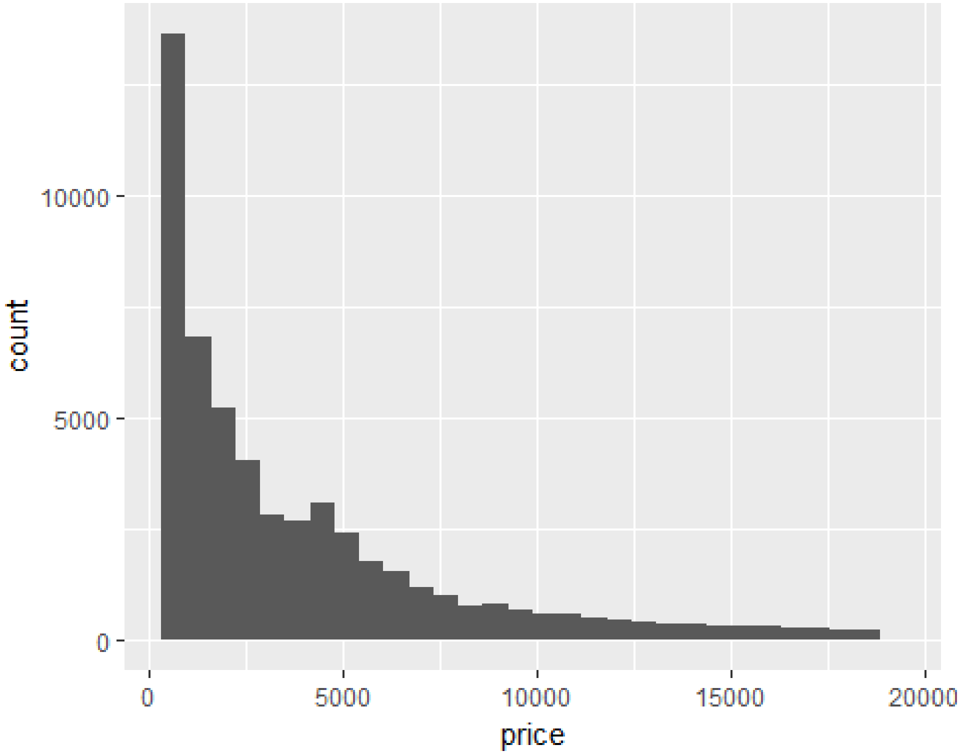
*z - depth in mm (0--31.8)

*depth - total depth percentage = z / mean(x, y) = 2 * z / (x + y) (43--79)

*table - width of top of diamond relative to widest point (43--95)

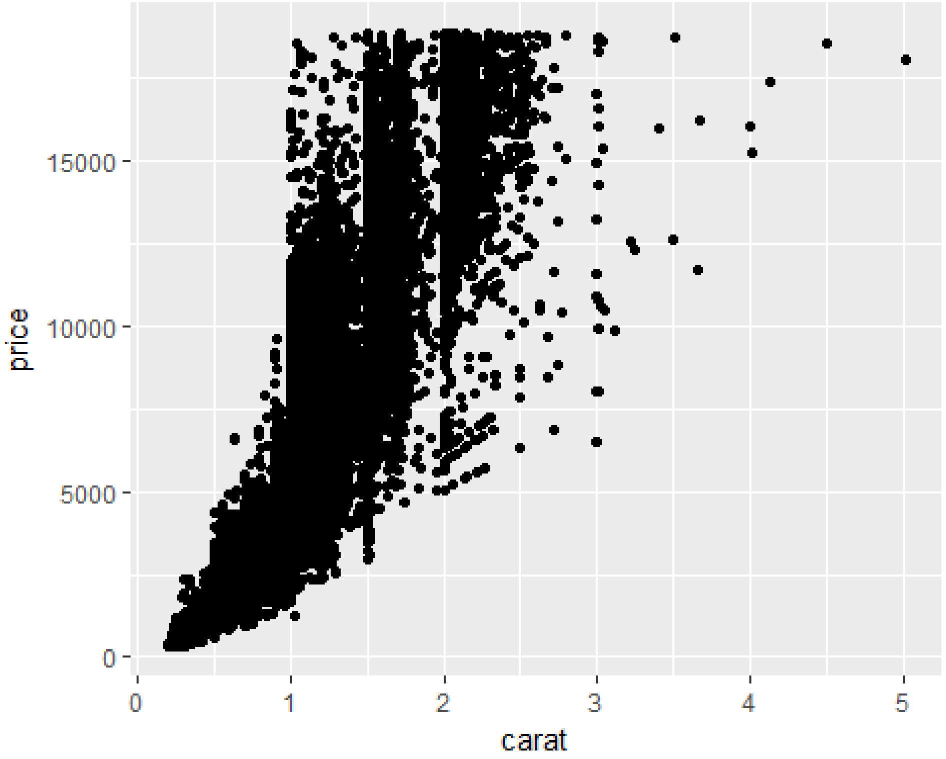
Histogram of price:

```
ggplot(data = diamonds, aes(x = price)) + geom_histogram()

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

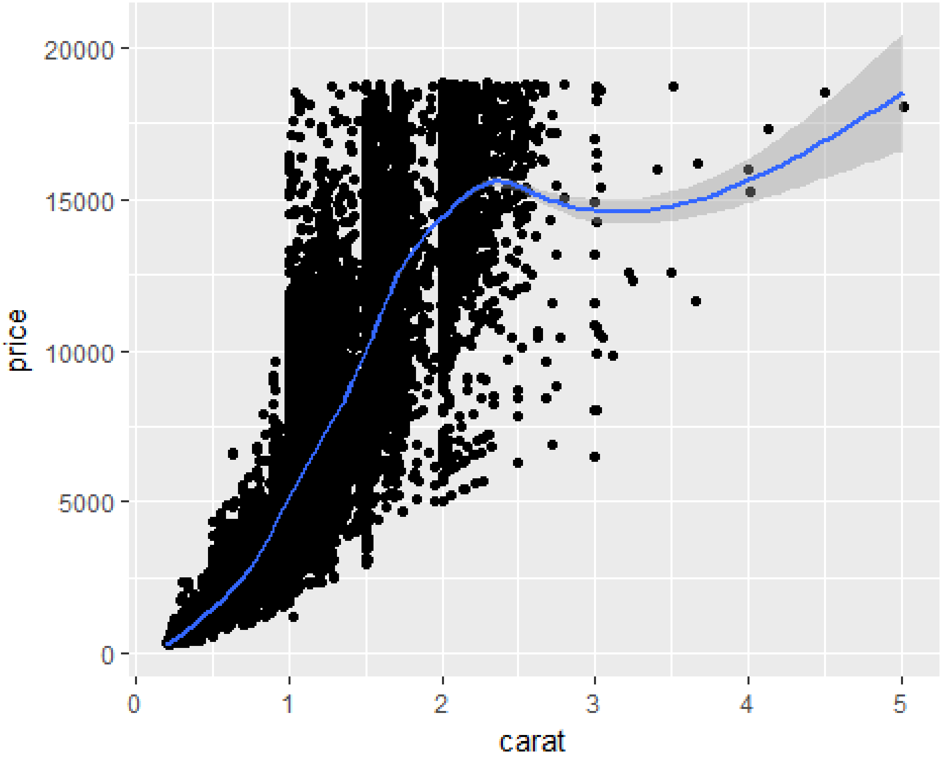


Carat vs Price

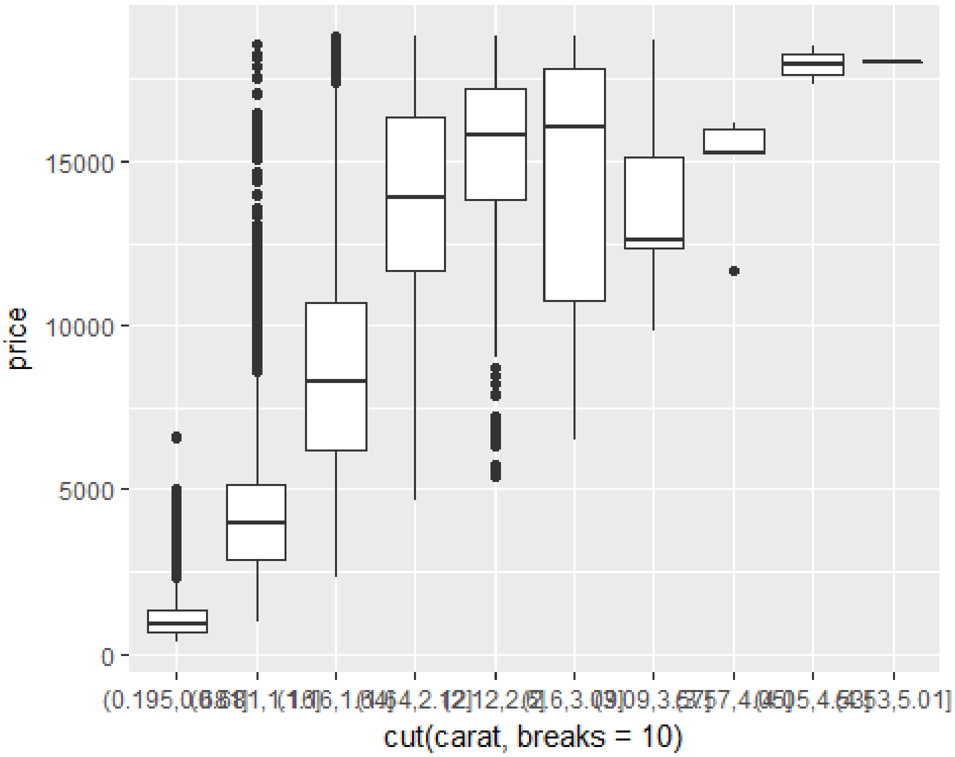
```r
ggplot(data = diamonds, aes(x = carat, y = price)) + geom_point()
```



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

## `geom_smooth()` using method = 'gam'
```

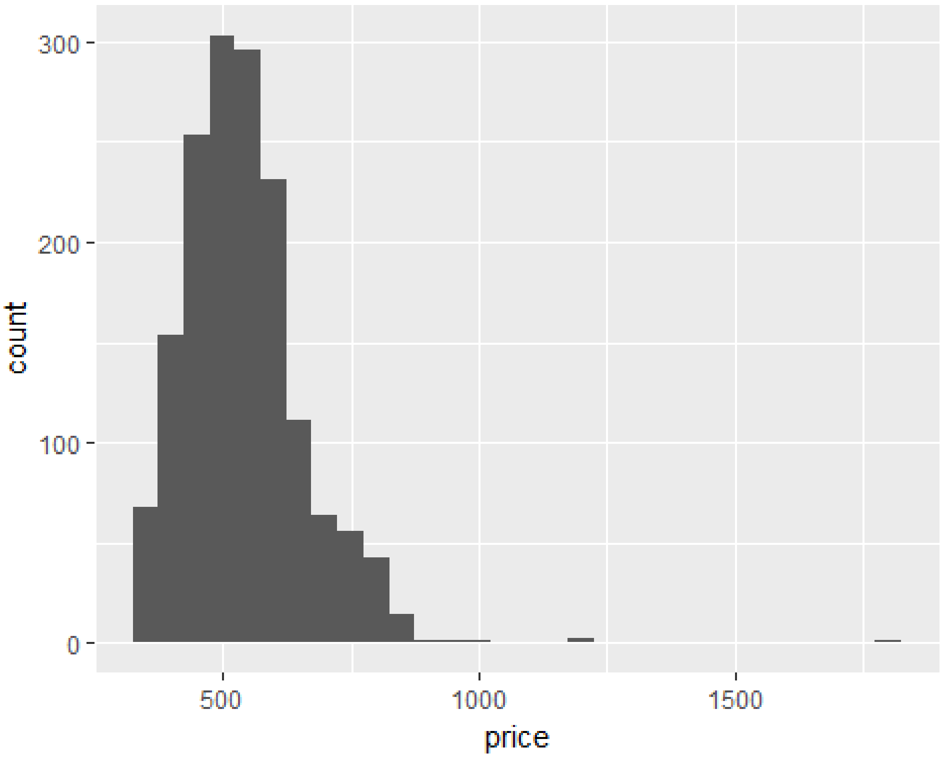


```
ggplot(data = diamonds, aes(x = cut(carat, breaks = 10), y = price)) +
geom_boxplot()
```



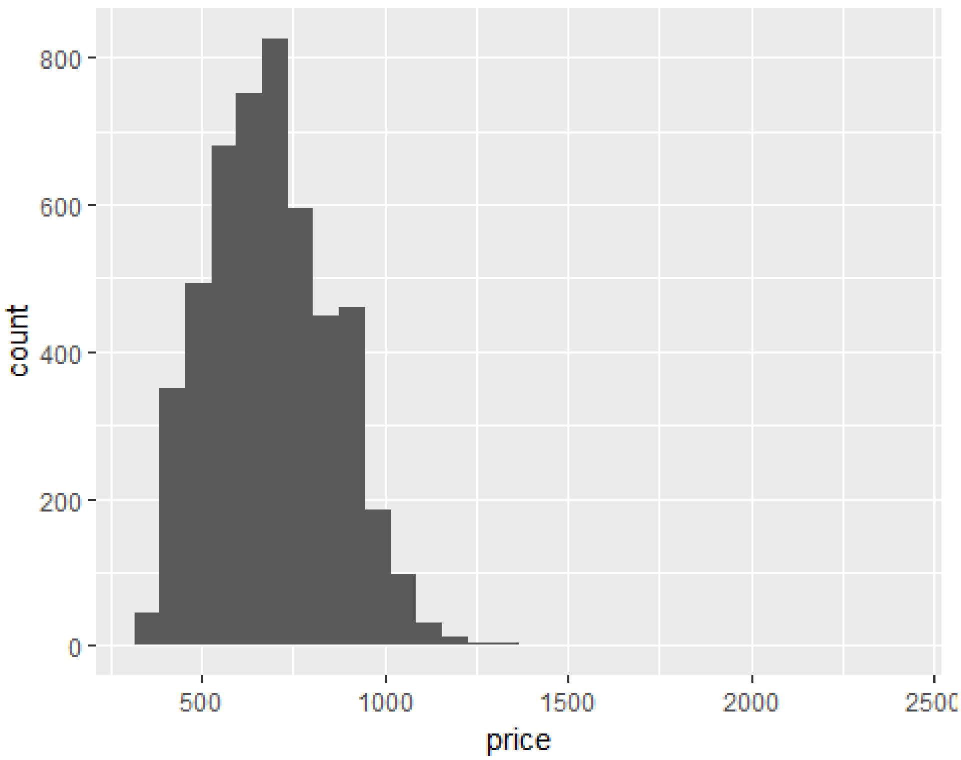
Price of diamonds less than 0.3 carats - Histogram

```
d03 <- diamonds[diamonds$carat < 0.3,]
ggplot(data = d03, aes(x = price)) +geom_histogram()

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



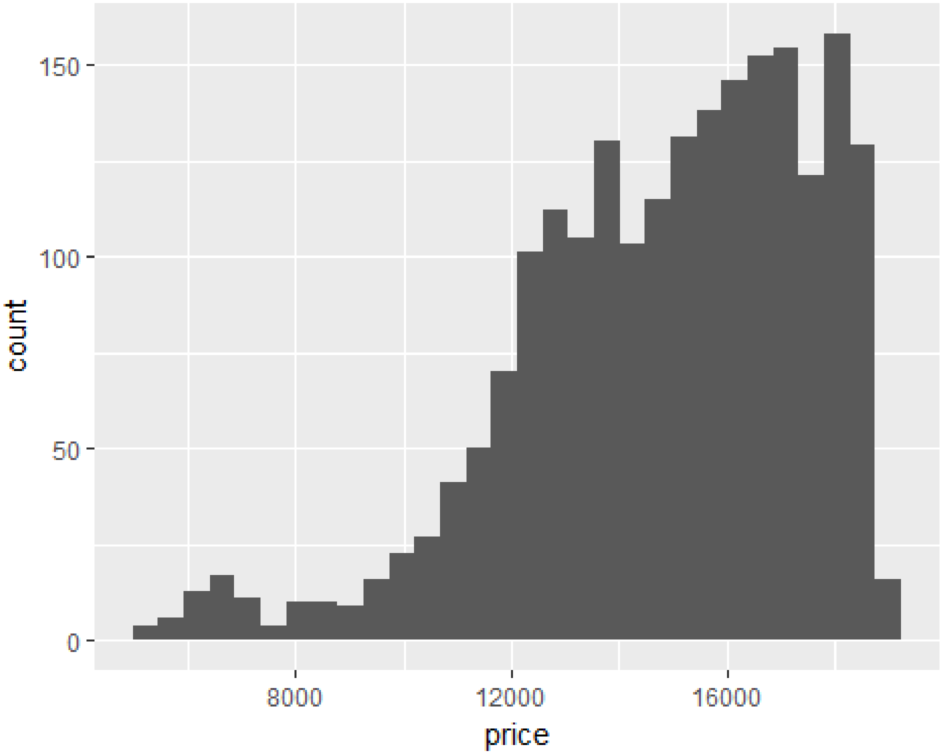
Price of diamonds between 0.29 and 0.31 carats - Histogram

```
d2931 <- diamonds[diamonds$carat <= 0.31 & diamonds$carat >= 0.29,]
ggplot(data = d2931, aes(x = price)) +geom_histogram()

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



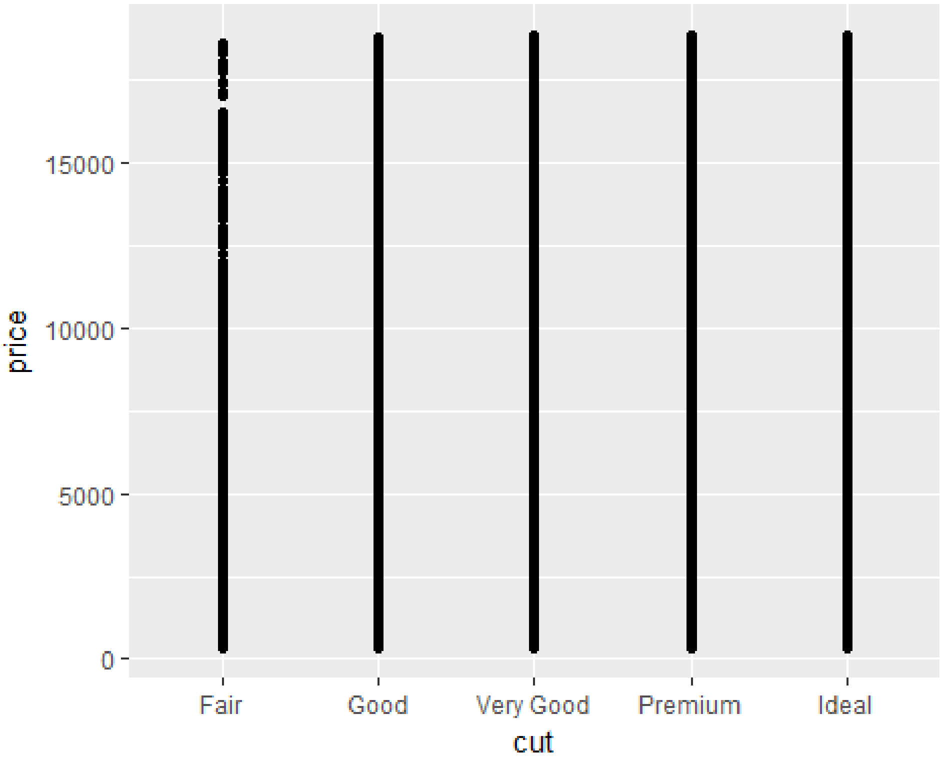
What about big diamonds (between 2 to 3 carats)

```
d23 <- diamonds[diamonds$carat <= 3 & diamonds$carat >= 2,]
ggplot(data = d23, aes(x = price)) +geom_histogram()

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



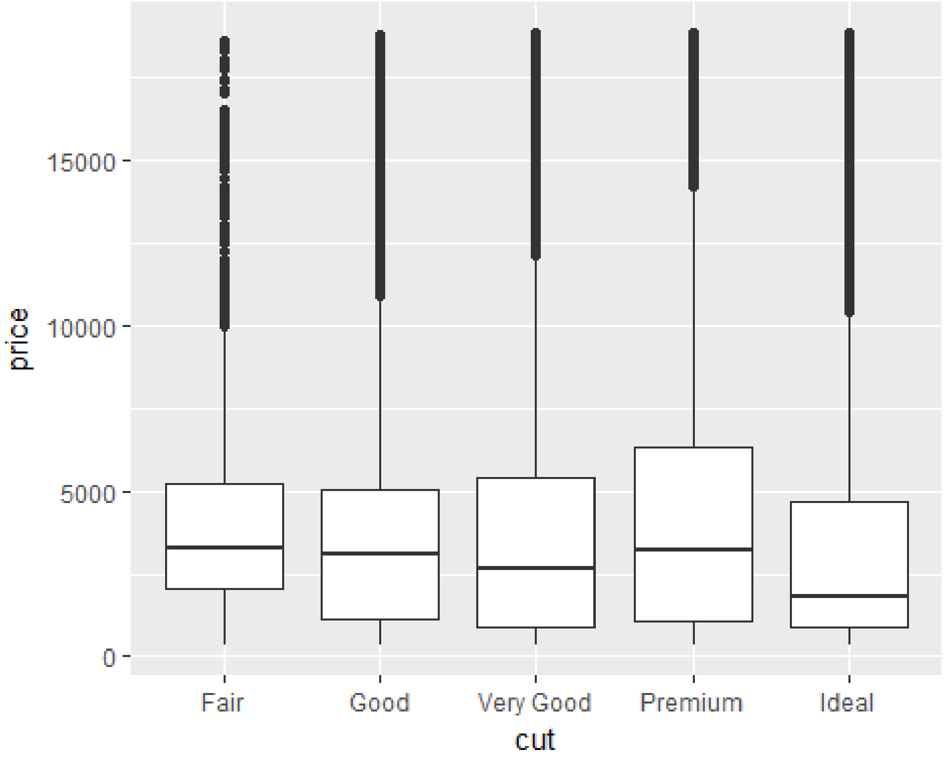
Cut big picture - Lets look at all diamonds by Cut *

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



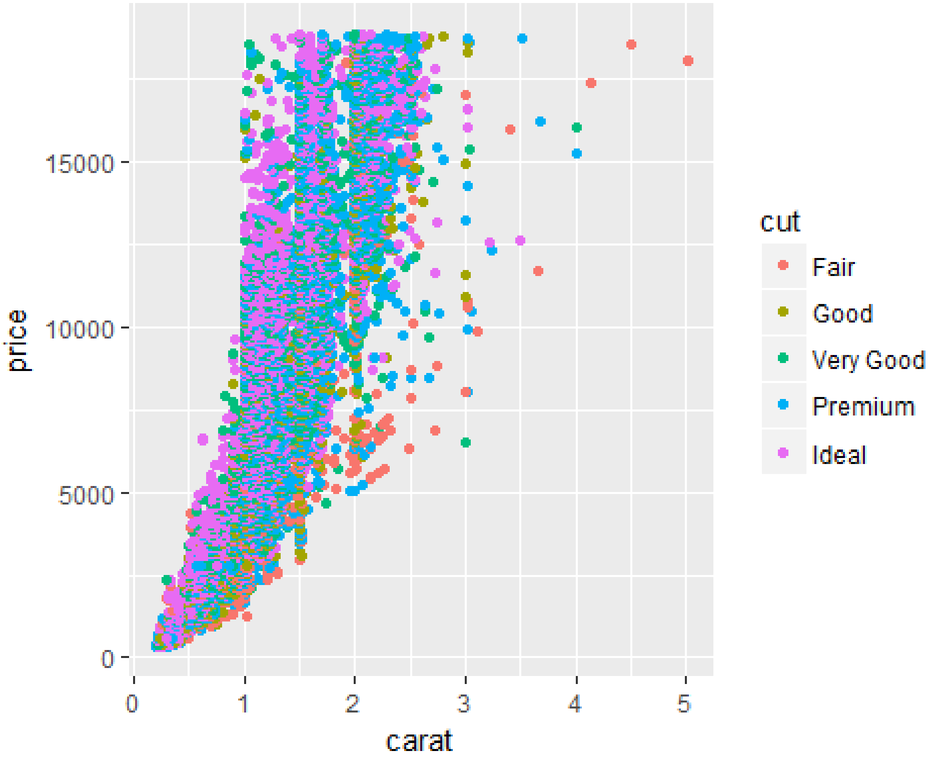
Unfortunatelty the above plot does not give much information

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



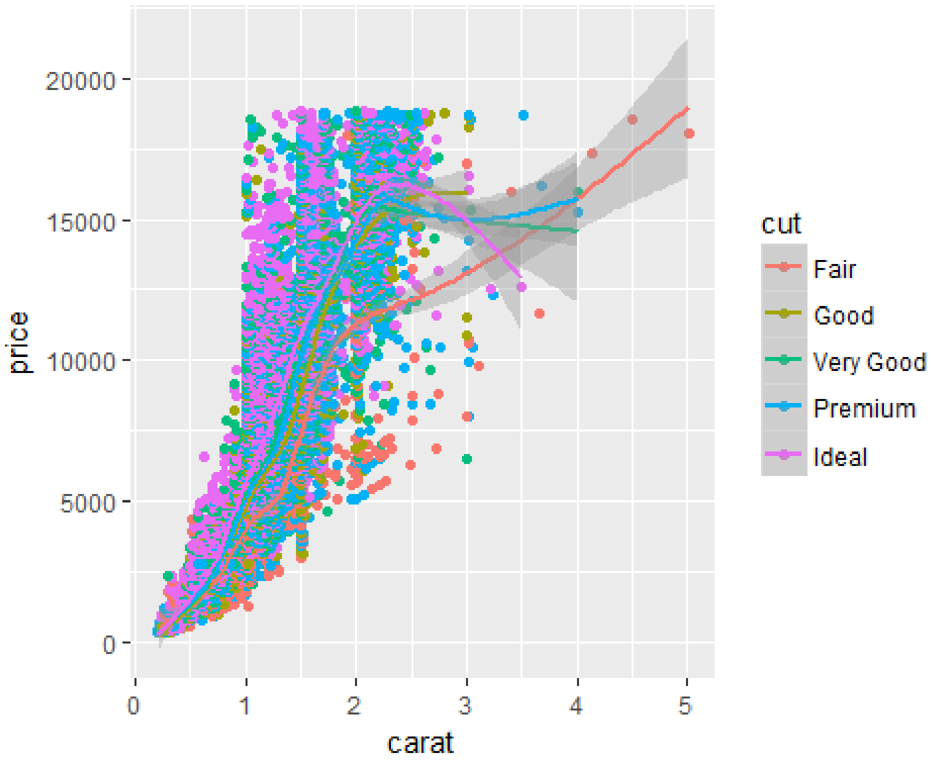
Neither does this help

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

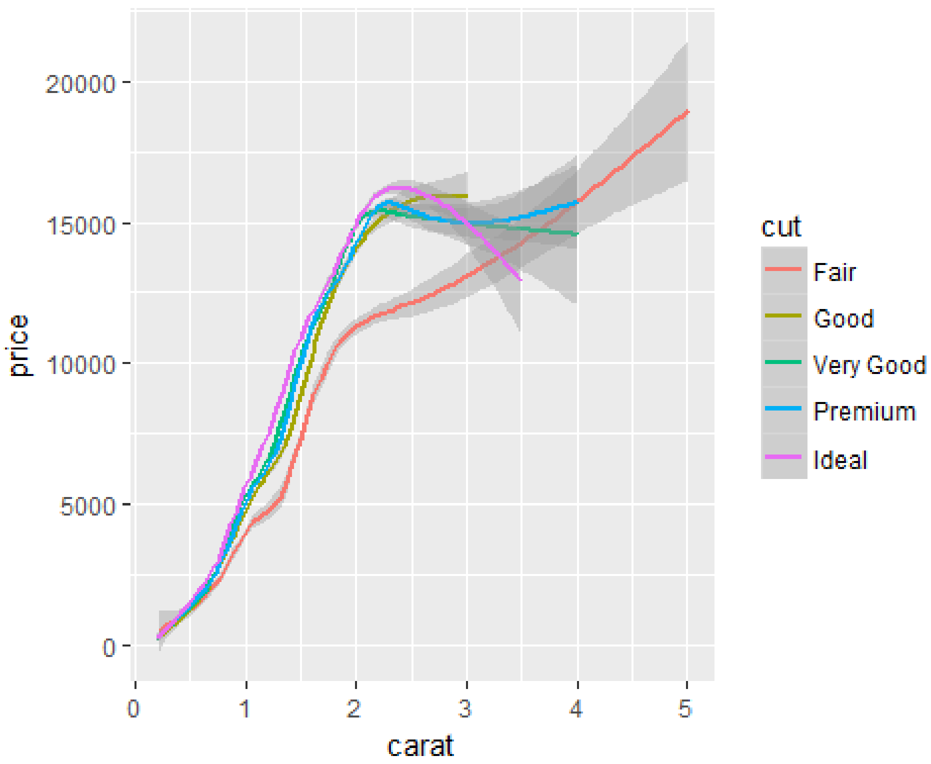


Here we see that Ideal cuts are higher priced than Fair cut

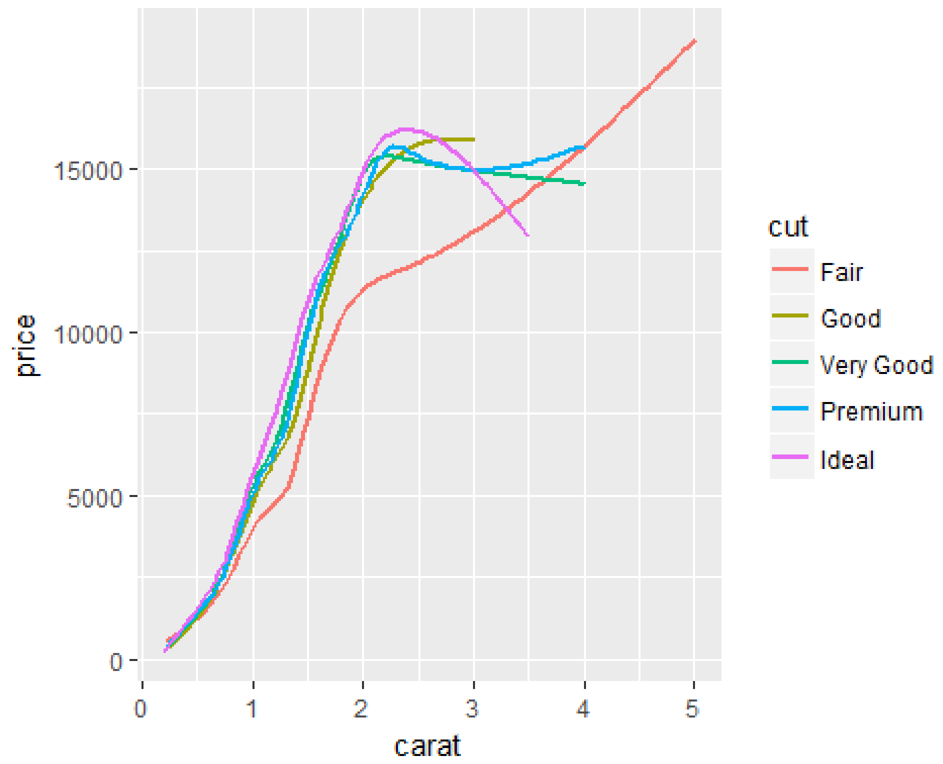
```
ggplot(data = diamonds, aes(x = carat, y = price, color = cut)) +
geom_point() +geom_smooth()

## `geom_smooth()` using method = 'gam'
```



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

## `geom_smooth()` using method = 'gam'
```

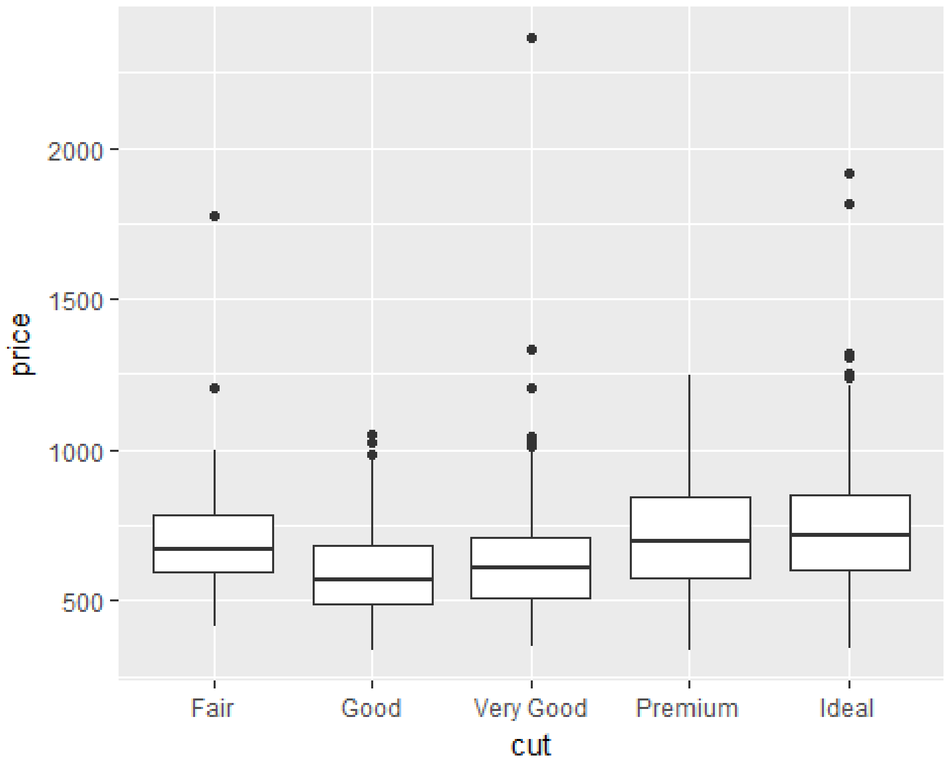


```
ggplot(data = diamonds, aes(x = carat, y = price, color = cut)) +
geom_smooth(se = F)

## `geom_smooth()` using method = 'gam'
```

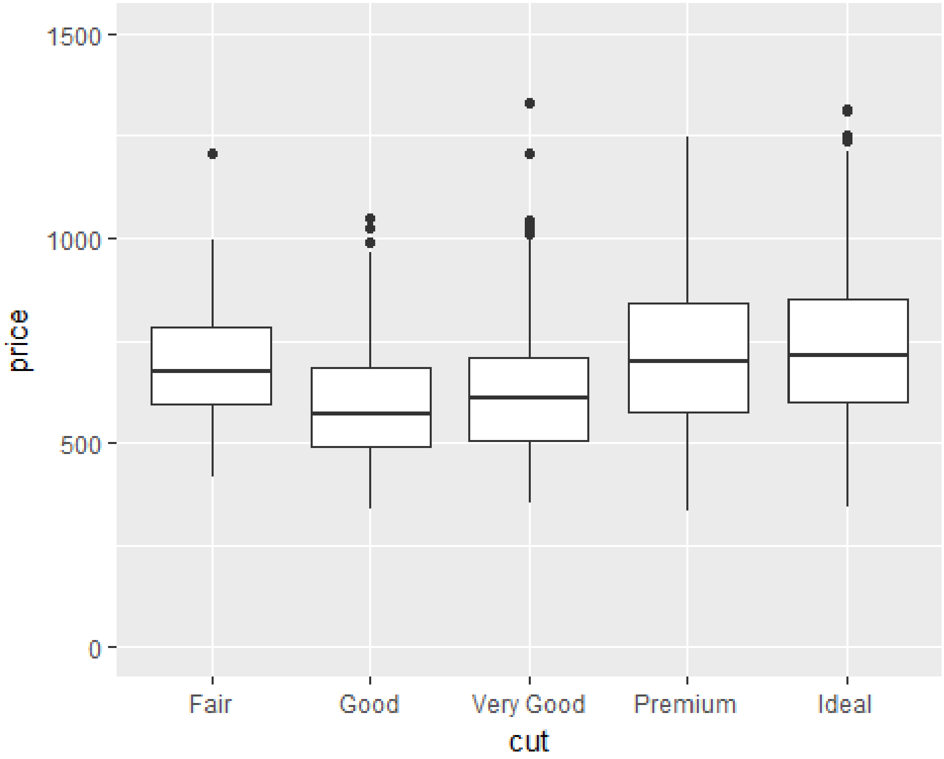


Cut for 0.3 carat diamonds (0.29 to 0.31)

```
ggplot(data= d2931, aes(x = cut, y = price)) + geom_boxplot()
```

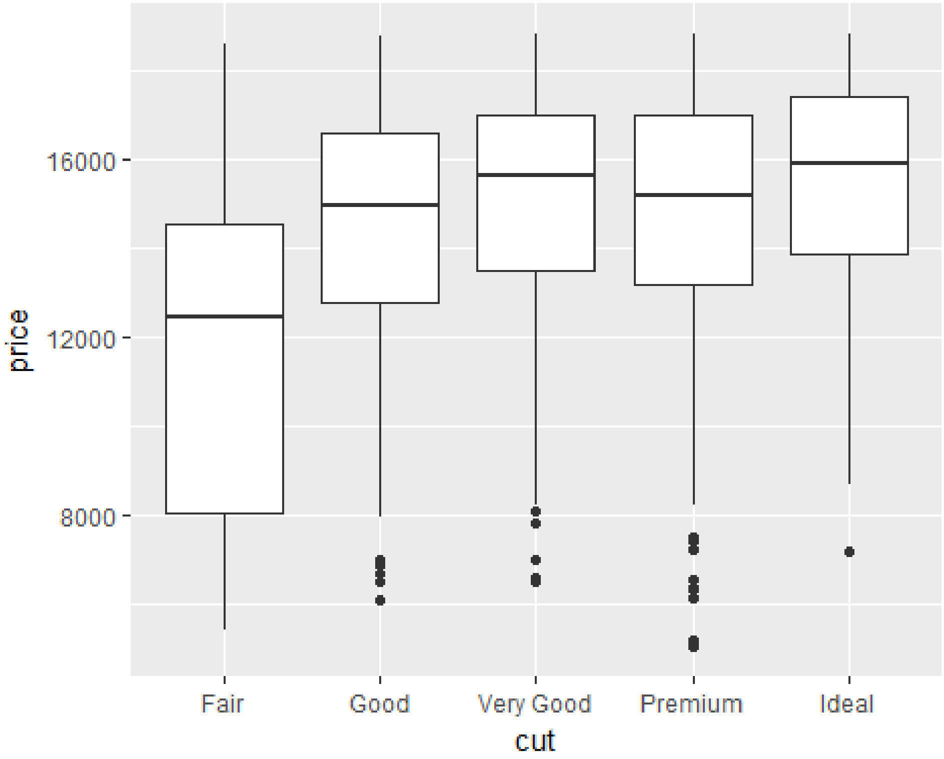


```
ggplot(data= d2931, aes(x = cut, y = price)) + geom_boxplot()+
coord_cartesian(ylim = c(0, 1500))
```



Cut really does not make a big difference for small diamonds. What about big diamonds?

```
ggplot(data= d23, aes(x = cut, y = price)) + geom_boxplot()
```

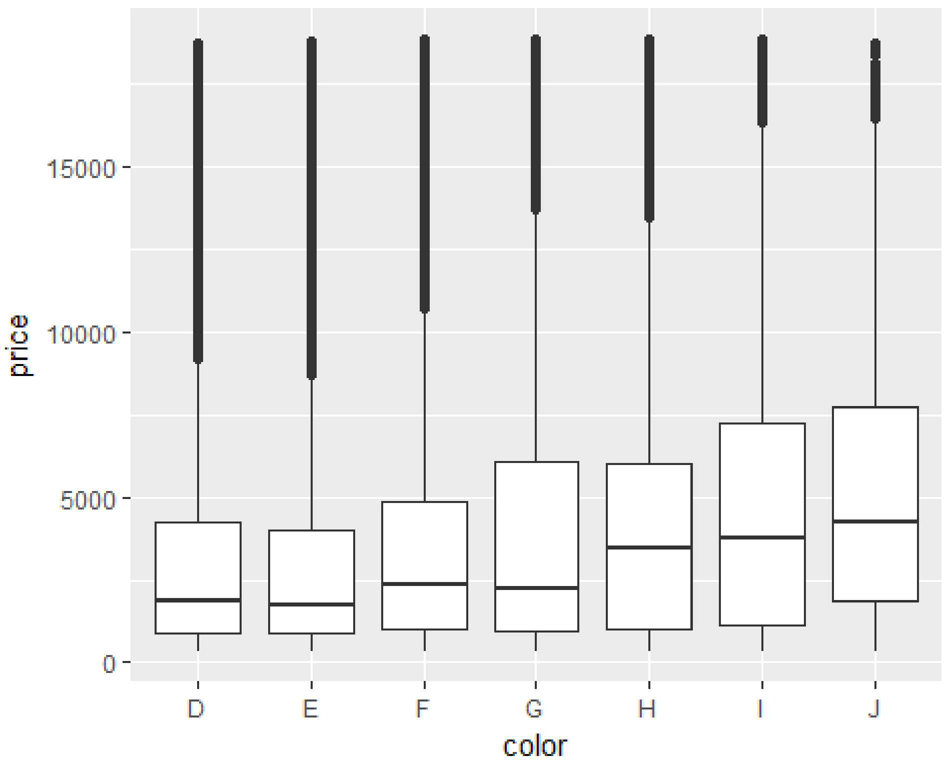


Based on this I decided to go for 0.29 to 0.31 carat + Ideal cut. Lets filter the 0.29 to 0.31 carat diamonds to just select the Ideal Cut diamonds**

```
d2931I <- d2931[d2931$cut == "Ideal",]
```

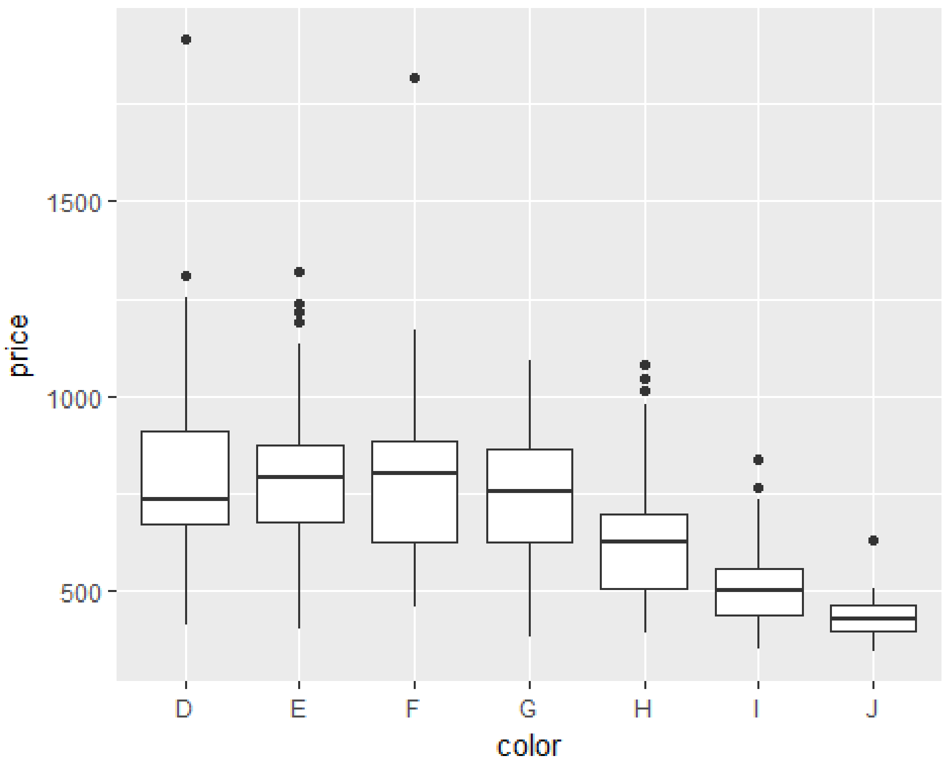
Price by Color for all diamonds

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



Different sizes of diamonds are grouped here. Lets focus on 0.29 to 0.31 with Ideal cut

```
ggplot(data = d2931I, aes(x = color, y = price)) + geom_boxplot()
```



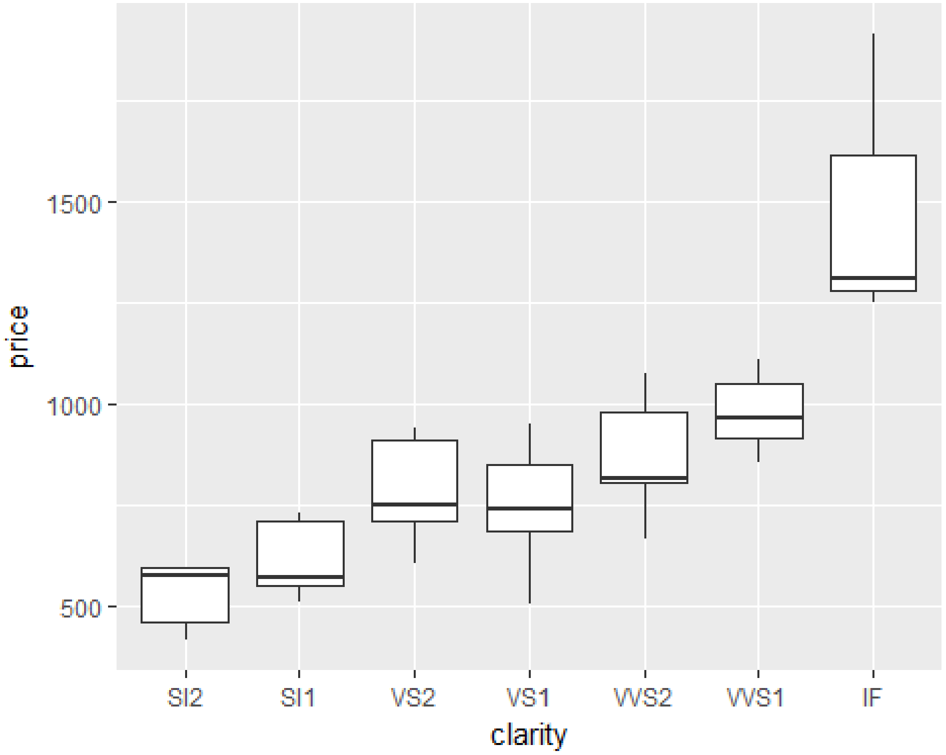
Based on this I will go with D (colorless).

Lets filter the 0.29 to 0.31 Carat Ideal cut diamonds to include just the D color.

```
d2931ID <- d2931I[d2931I$color == "D",]
```

Price by Clarity for our selected set of diamonds

```
ggplot(data = d2931ID, aes(x=clarity, y = price)) +geom_boxplot()
```

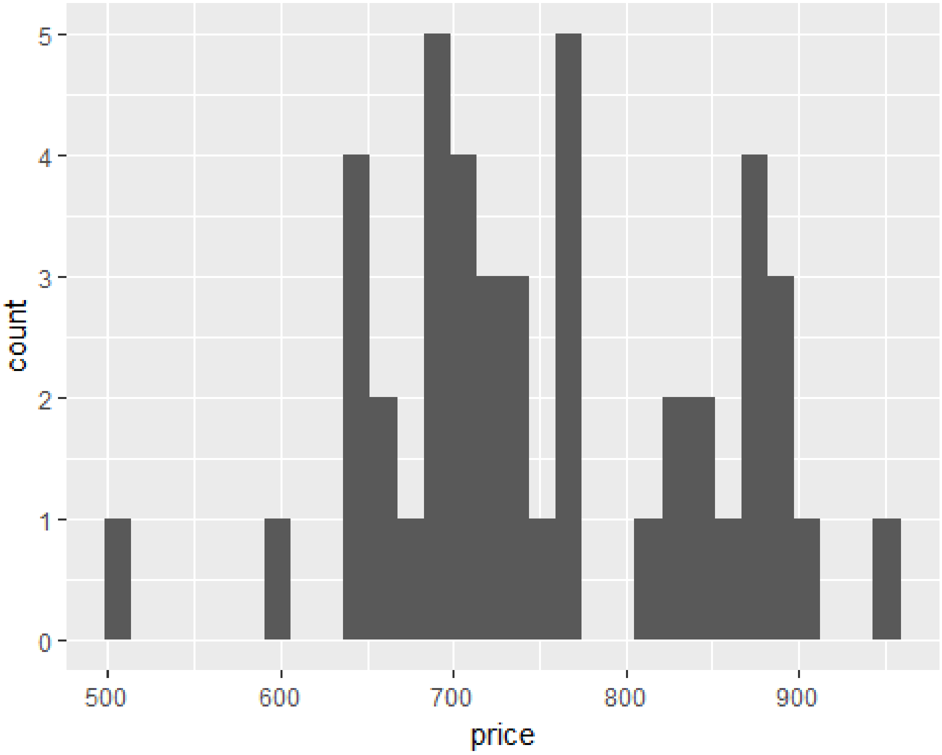


Based on this I will go with VS1 clarity. Lets add that too to the selected dataset.

```
d2931IDVS1 <- d2931ID[d2931ID$clarity == "VS1",]
```

Now we are left with 45 diamonds. Lets see the price distribution

```
ggplot(data = d2931IDVS1, aes(x=price)) + geom_histogram()

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



I got a good bargain at around $500

```
d2931IDVS1[d2931IDVS1$price < 550, ]
```

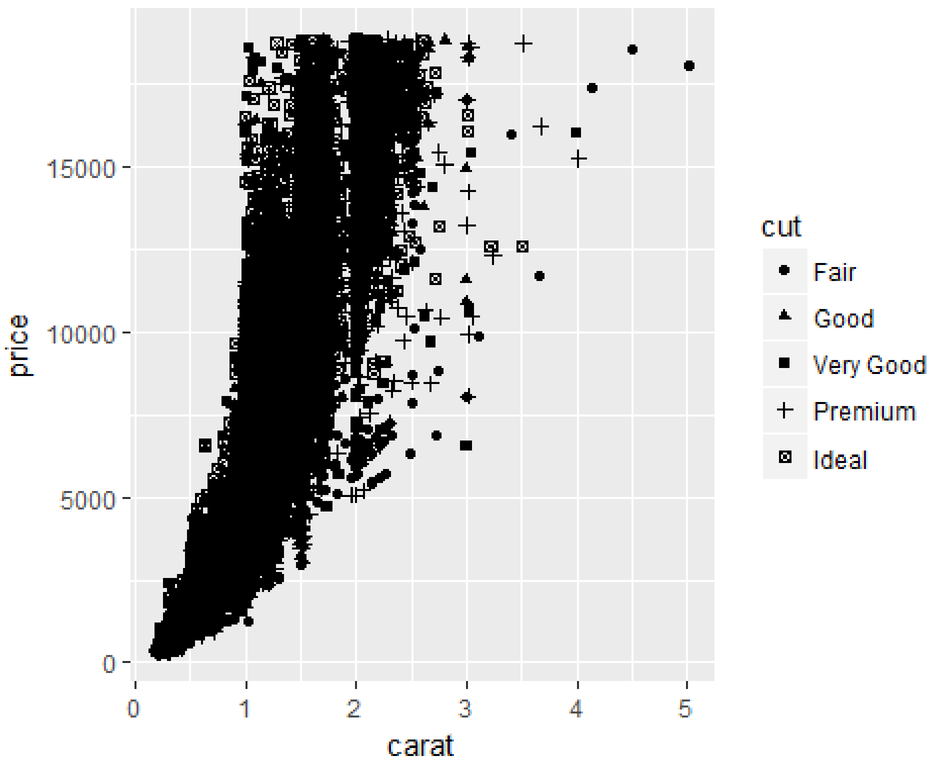
```
## # A tibble: 1 x 10
##   carat   cut color clarity depth table price     x     y     z
##   <dbl> <ord> <ord>   <ord> <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1  0.29 Ideal     D     VS1  61.7    57   504  4.23  4.26  2.62
```

2. AES

Mapping go in aes(), attributes go in geom layer. (typically)

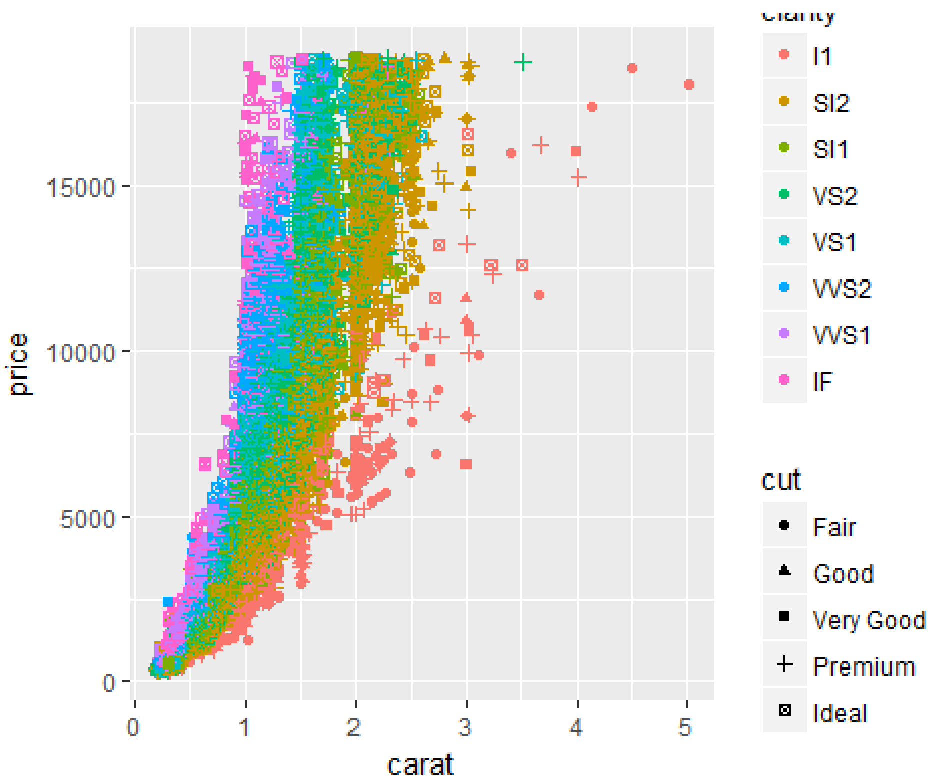
Maping points shape to the Cut

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



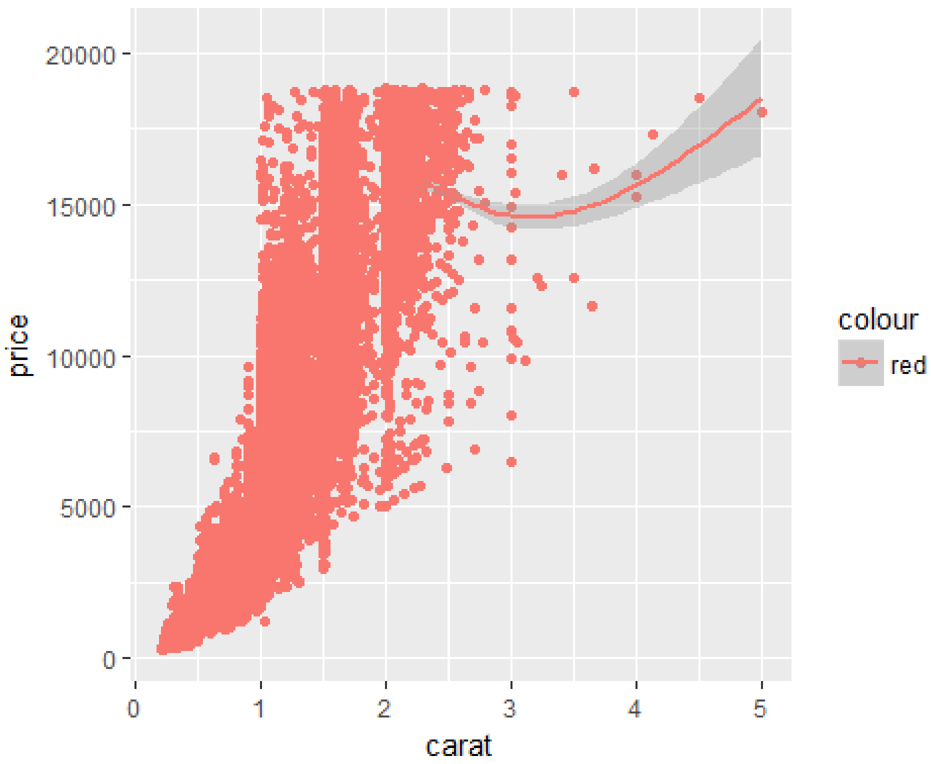
Maping points shape to the Cut and color to Clairity

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

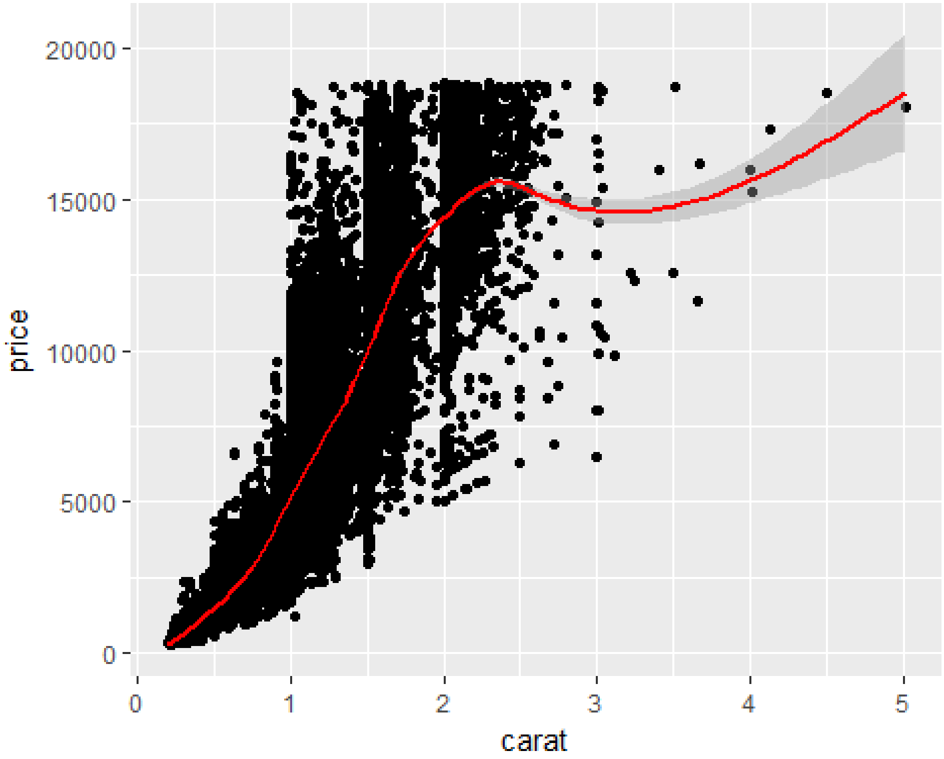


Adding the point attribute as red color

```
ggplot(data = diamonds, aes(x = carat, y = price, color = "red")) +
    geom_point() +geom_smooth()

## `geom_smooth()` using method = 'gam'
```

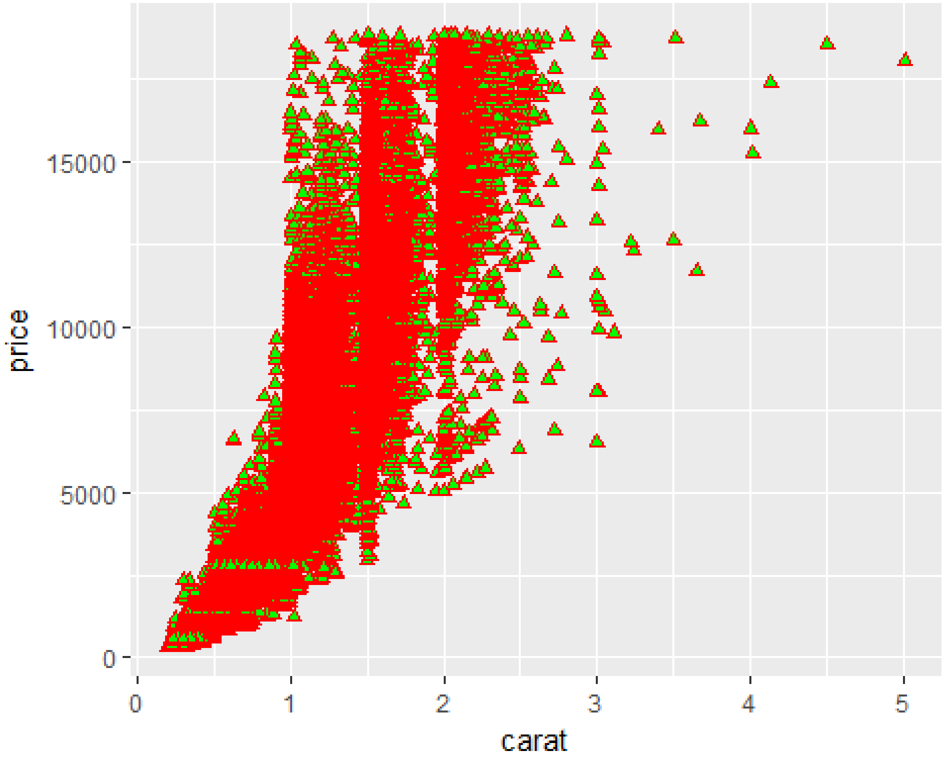


```
ggplot(data = diamonds, aes(x = carat, y = price)) +
  geom_point() +geom_smooth(color = "red")

## `geom_smooth()` using method = 'gam'
```



Changing the point attribute to a shape, color and fill

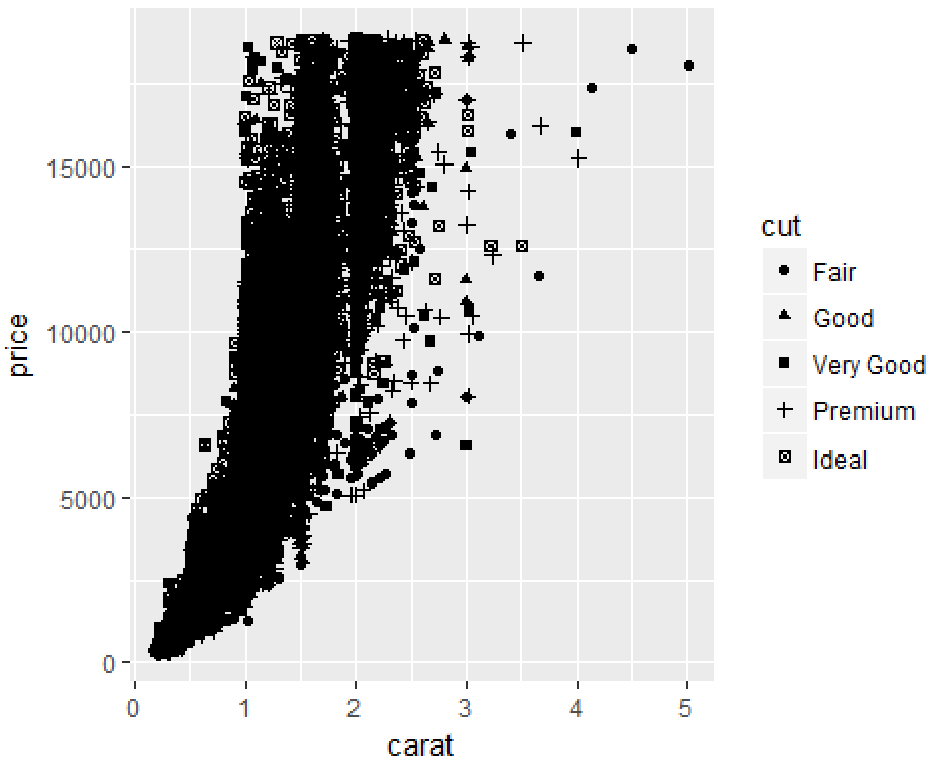
```
ggplot(data = diamonds, aes(x = carat, y = price)) + geom_point(shape = 24,
color = "red", fill = "green")
```



See shapes on cheat sheet.

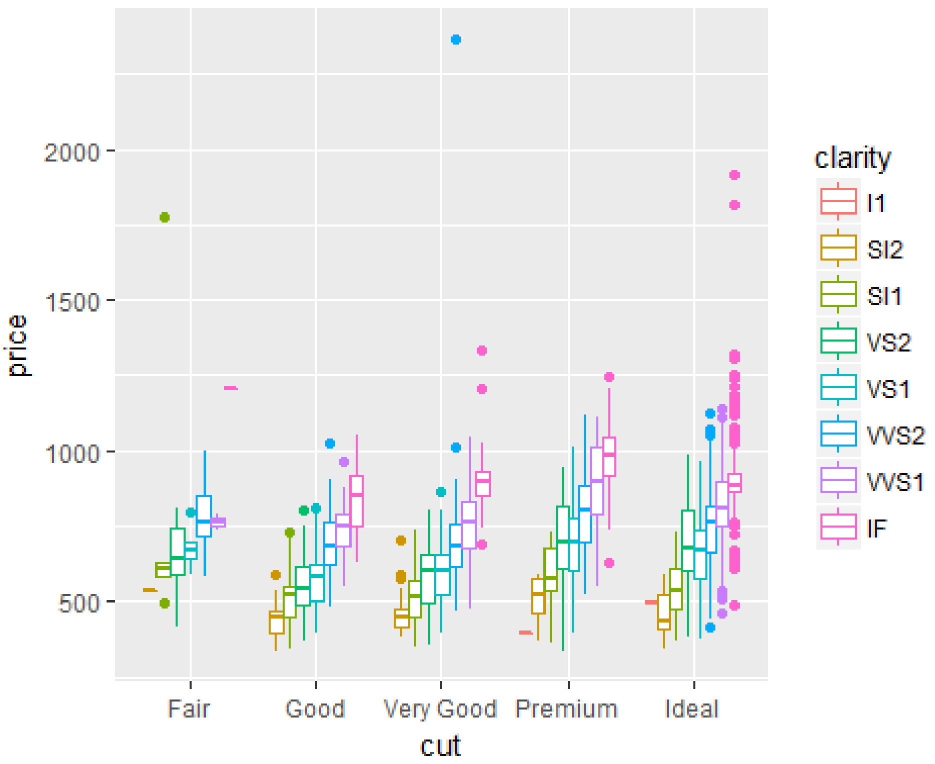
AES includes: x, y, color (outline except in case of dots), fill, size, alpha, linetypes, shape

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

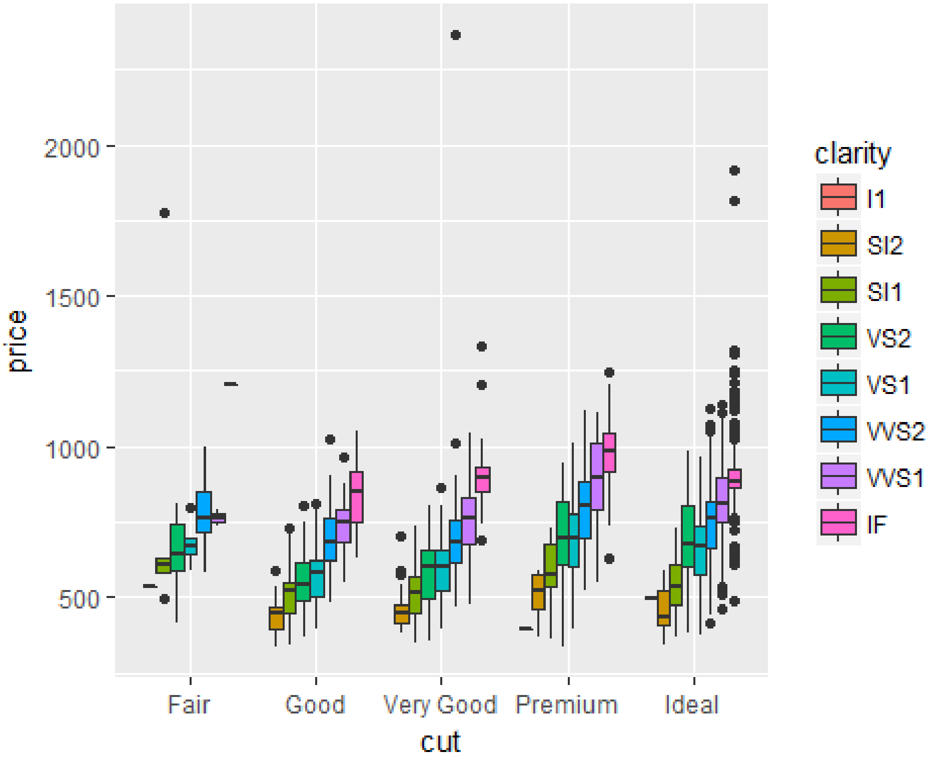


Box plots with color or fill mapped to clarity

```
ggplot(data = d2931, aes(x = cut, y = price, color = clarity)) +
geom_boxplot()
```

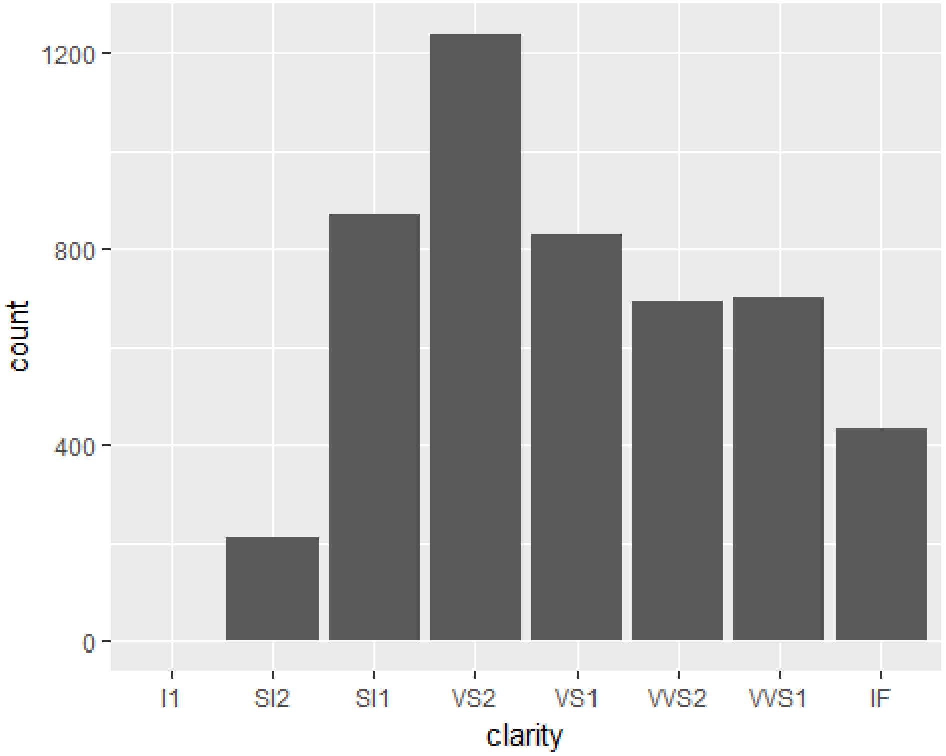


```
ggplot(data = d2931, aes(x = cut, y = price, fill = clarity)) +
geom_boxplot()
```



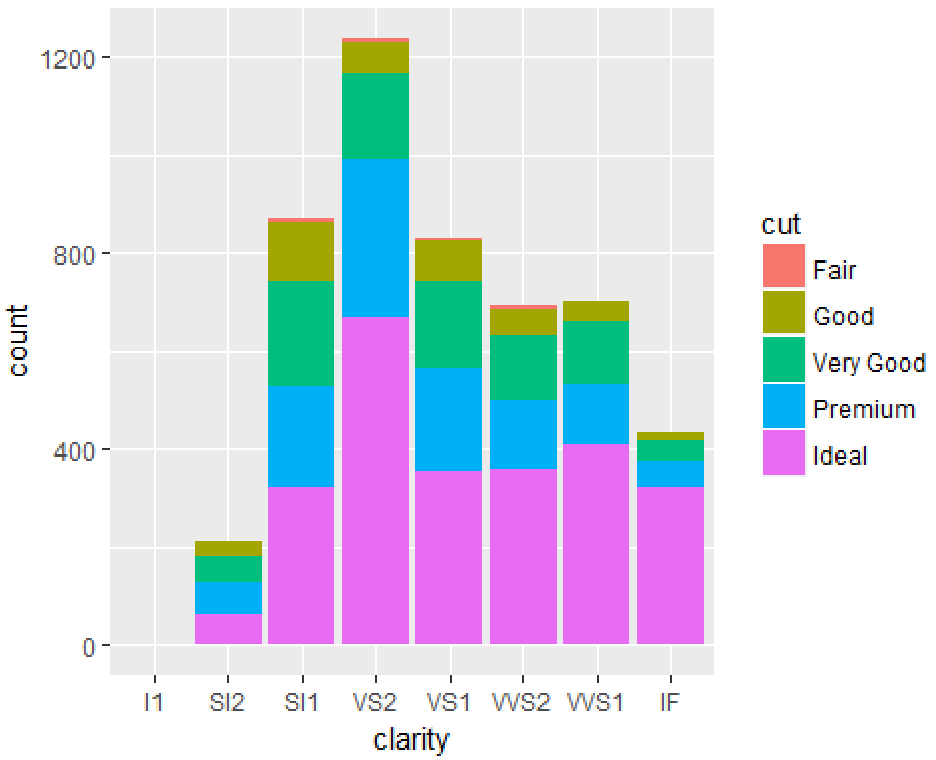
Lets draw a bar chart for clarity

```
ggplot(data = d2931, aes(x = clarity)) + geom_bar()
```



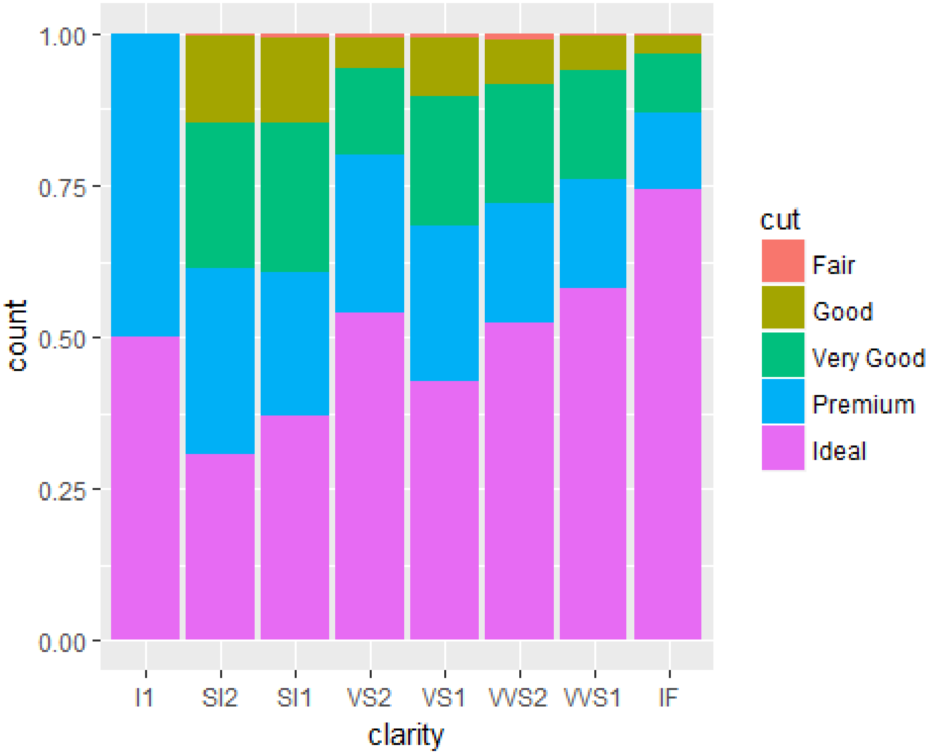
Fill the above by Cut

```
ggplot(data = d2931, aes(x = clarity, fill = cut)) + geom_bar()
```

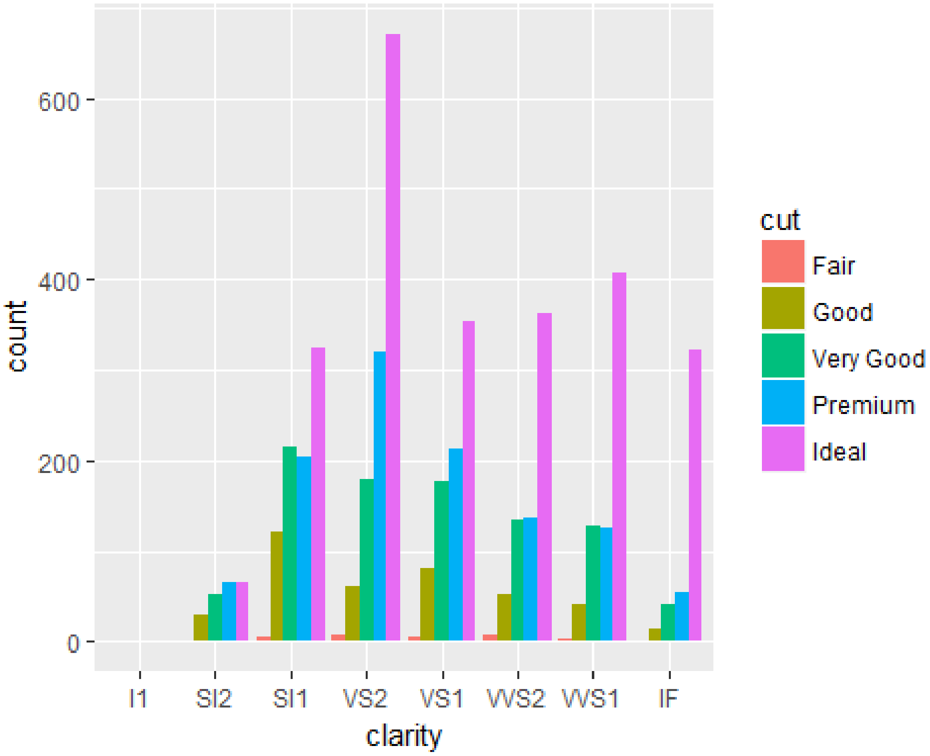


Changing the attributes of the bar chart

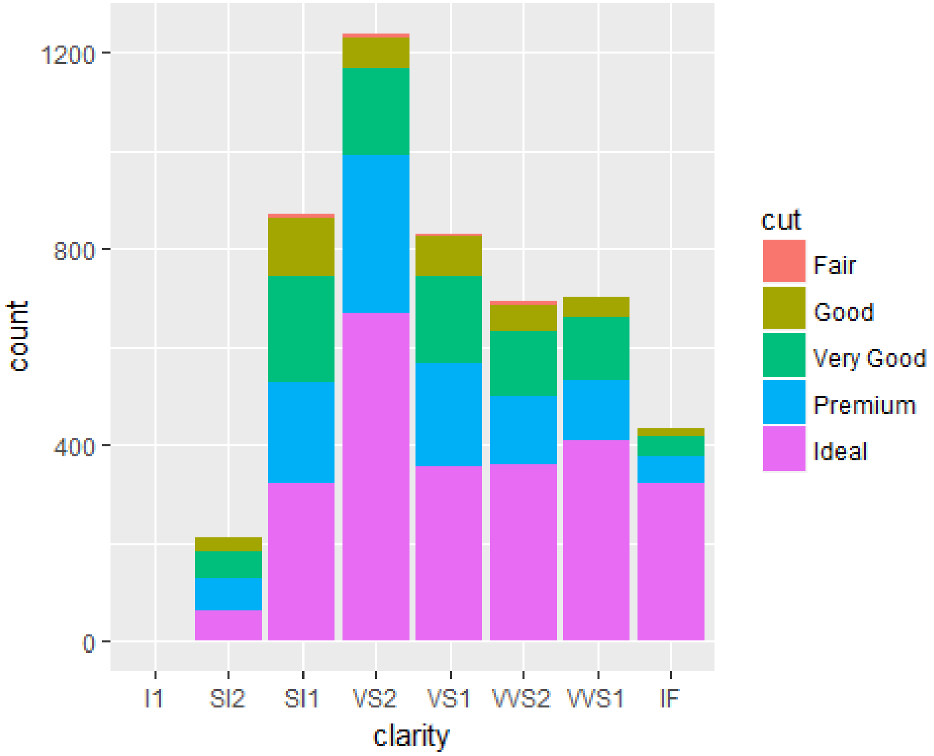
```
ggplot(data = d2931, aes(x = clarity, fill = cut)) + geom_bar(position =
"fill")
```



```
ggplot(data = d2931, aes(x = clarity, fill = cut)) + geom_bar(position =
"dodge")
```



```
ggplot(data = d2931, aes(x = clarity, fill = cut)) + geom_bar(position =
"stack")
```

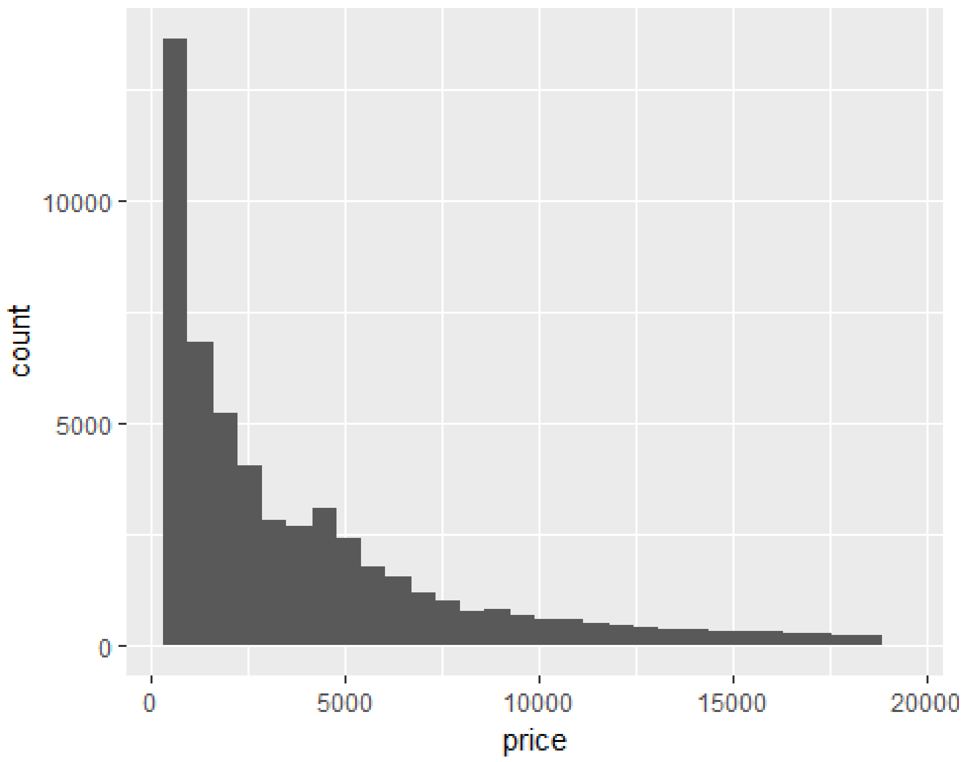


3. GEOM

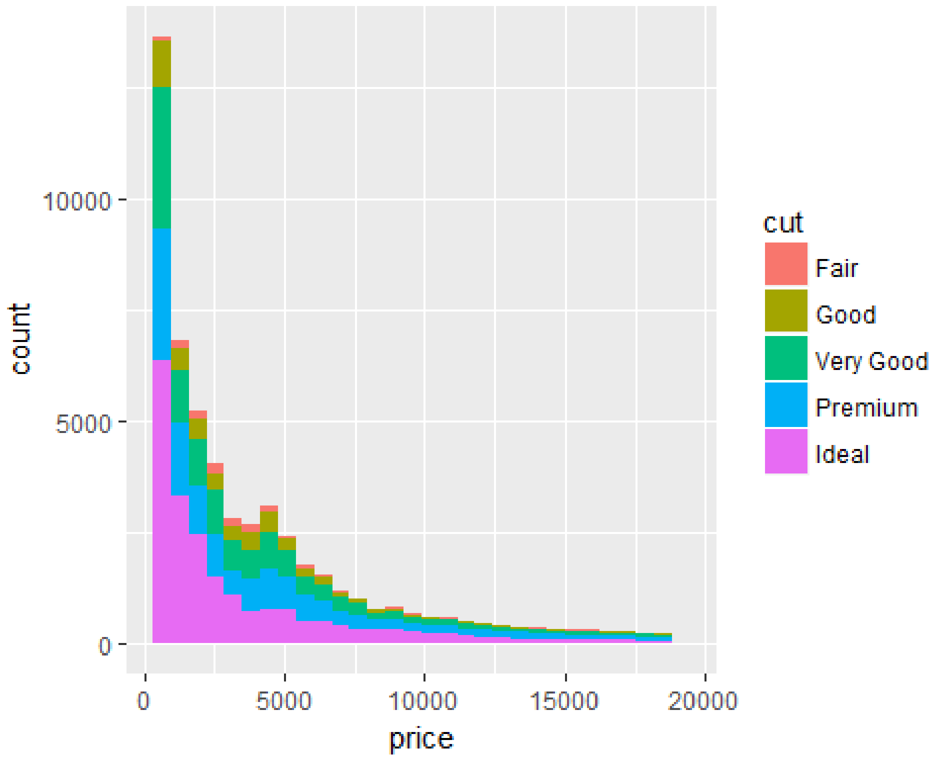
37 types of geometries are available in GGPlot2 - see the cheatsheet. Out of these we have already talked about scatter plot, bar chart, histogram, density plot

Lets relook at histogram:

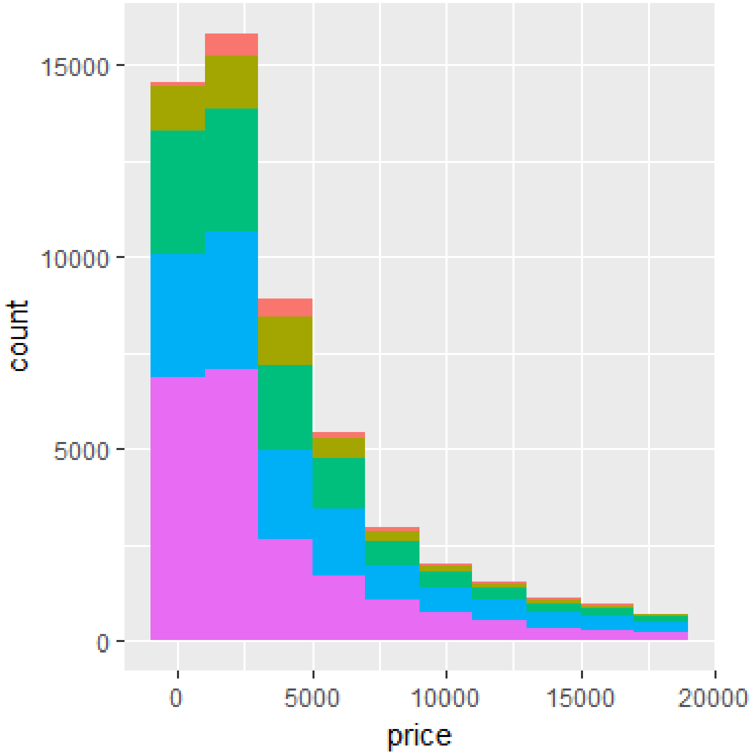
```
ggplot(data = diamonds, aes(x = price)) + geom_histogram()

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



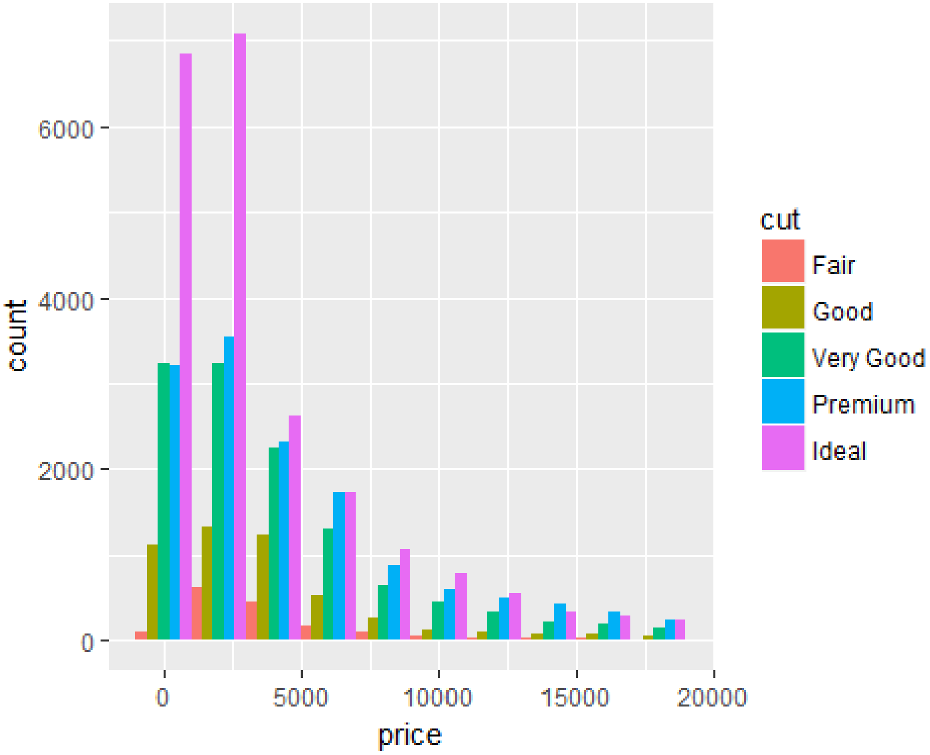
```
ggplot(data = diamonds, aes(x = price, fill = cut)) +geom_histogram()

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



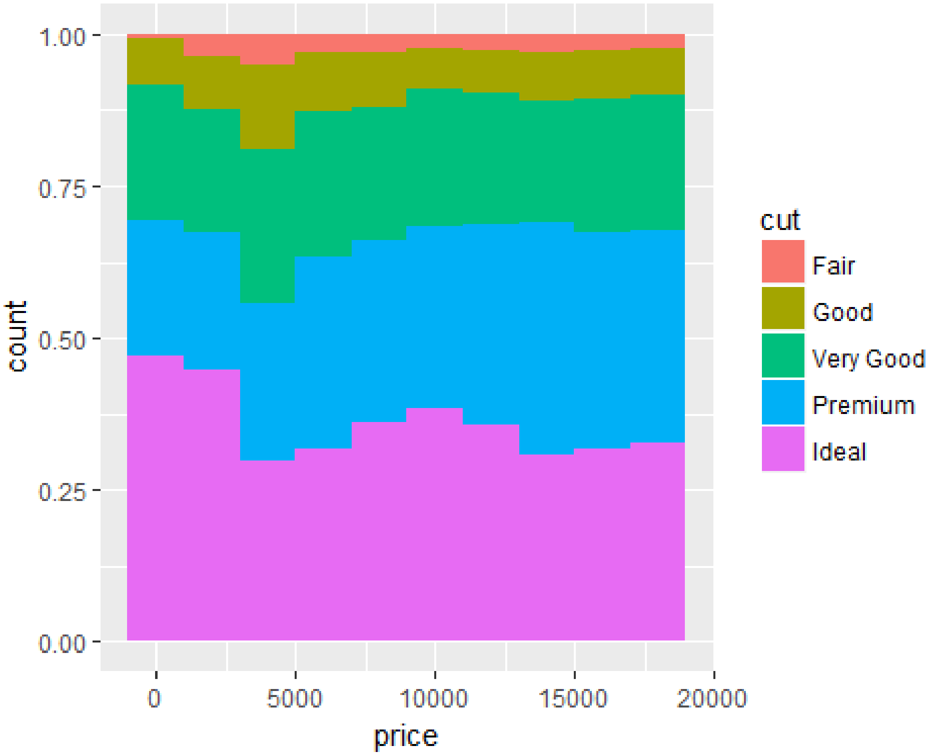
```
ggplot(data = diamonds, aes(x = price, fill = cut)) +geom_histogram(binwidth
= 2000)
```



```
ggplot(data = diamonds, aes(x = price, fill = cut)) +geom_histogram(binwidth
= 2000, position = "dodge")
```

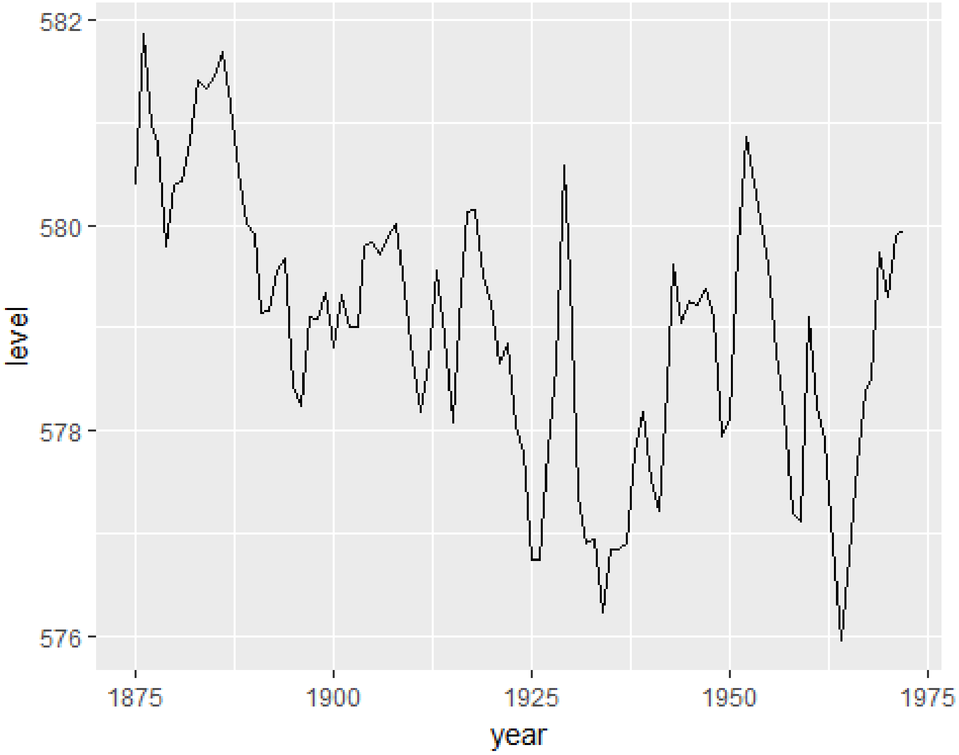


```
ggplot(data = diamonds, aes(x = price, fill = cut)) +geom_histogram(binwidth
= 2000, position = "fill")
```

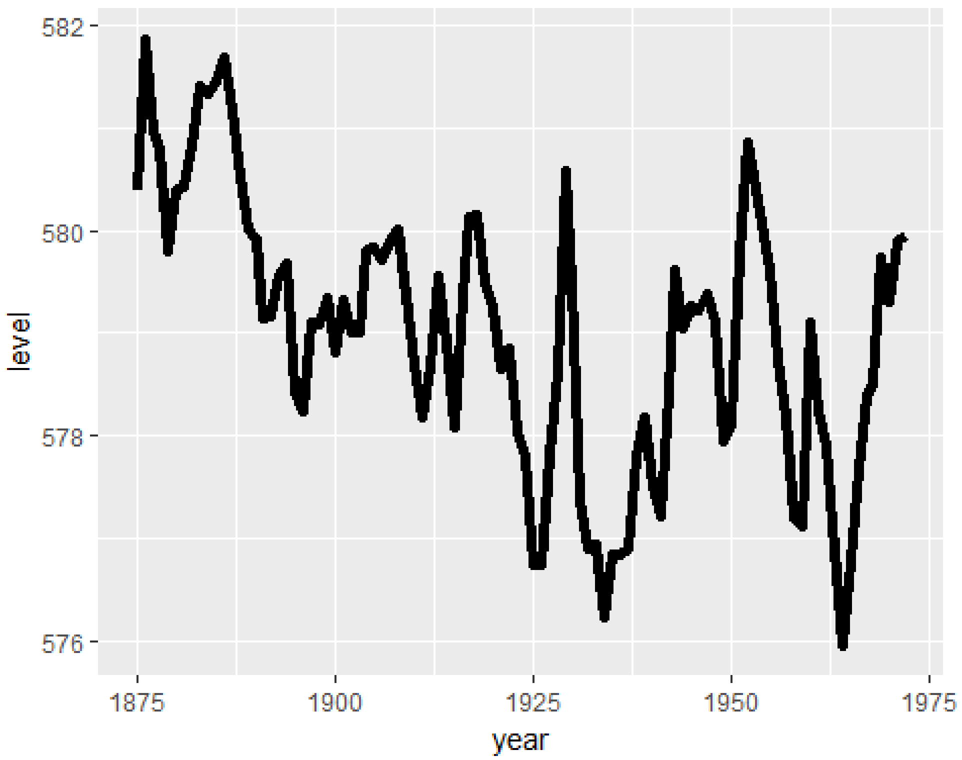


Line Plot Lake Huron level over time

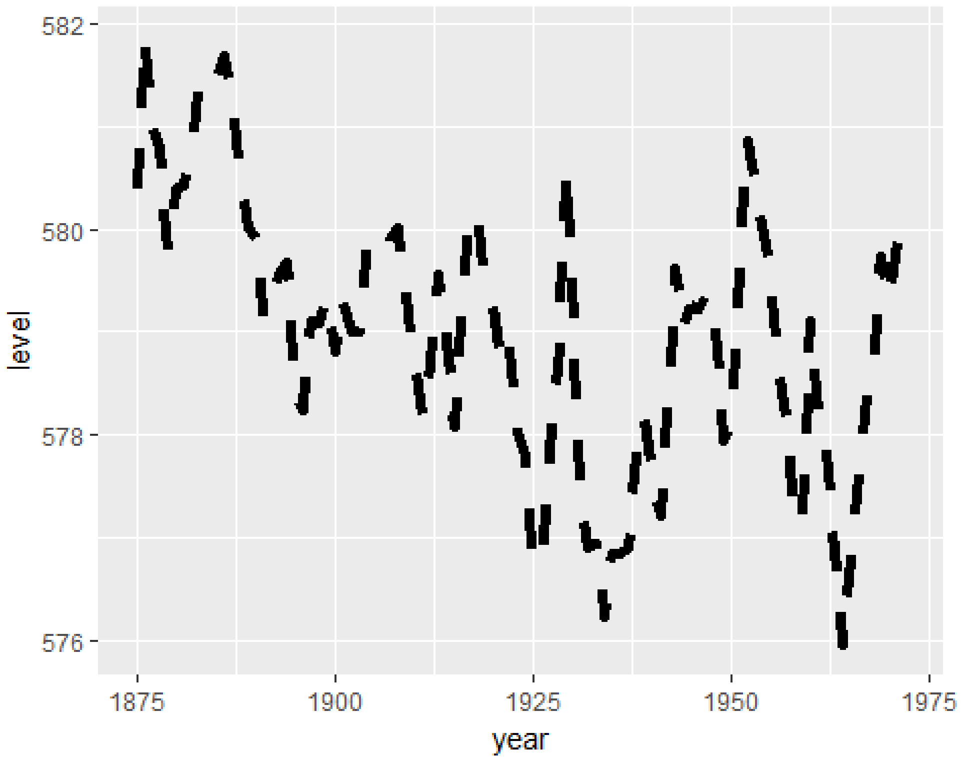
```
head(LakeHuron)

## [1] 580.38 581.86 580.97 580.80 579.79 580.39

huron <- data.frame(year = 1875:1972, level = as.vector(LakeHuron))
head(huron)

##   year  level
## 1 1875 580.38
## 2 1876 581.86
## 3 1877 580.97
## 4 1878 580.80
## 5 1879 579.79
## 6 1880 580.39

ggplot(data = huron, aes(x = year, y = level)) + geom_line()
```



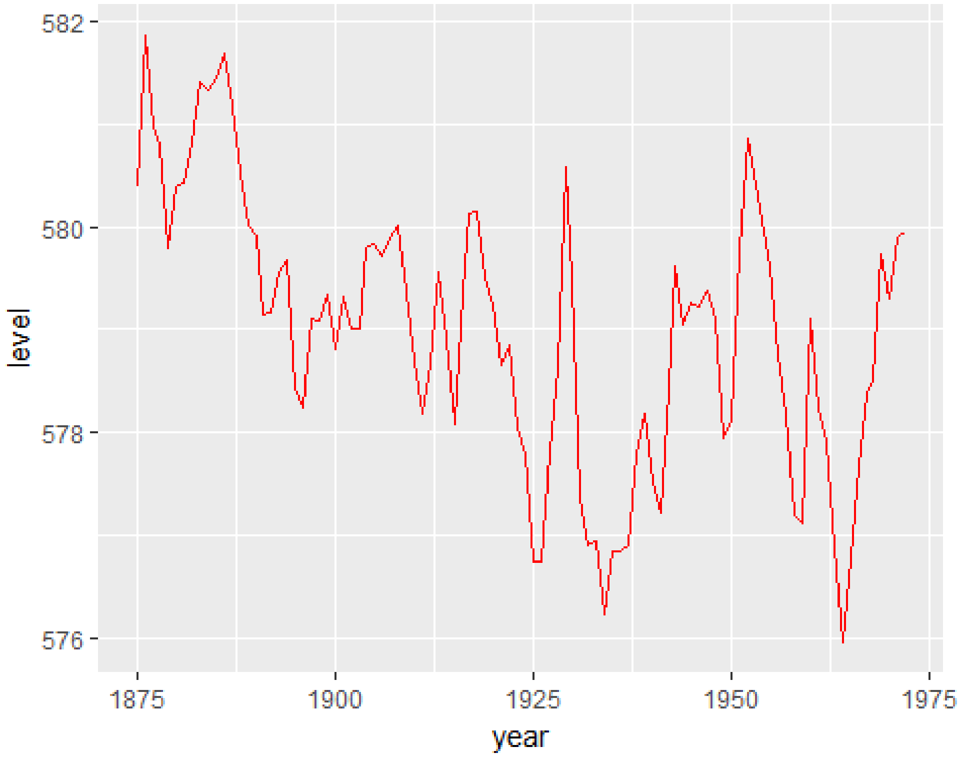
```
ggplot(data = huron, aes(x = year, y = level)) + geom_line(size = 2)
```



```
ggplot(data = huron, aes(x = year, y = level)) + geom_line(size = 2, linetype
= "dashed")
```

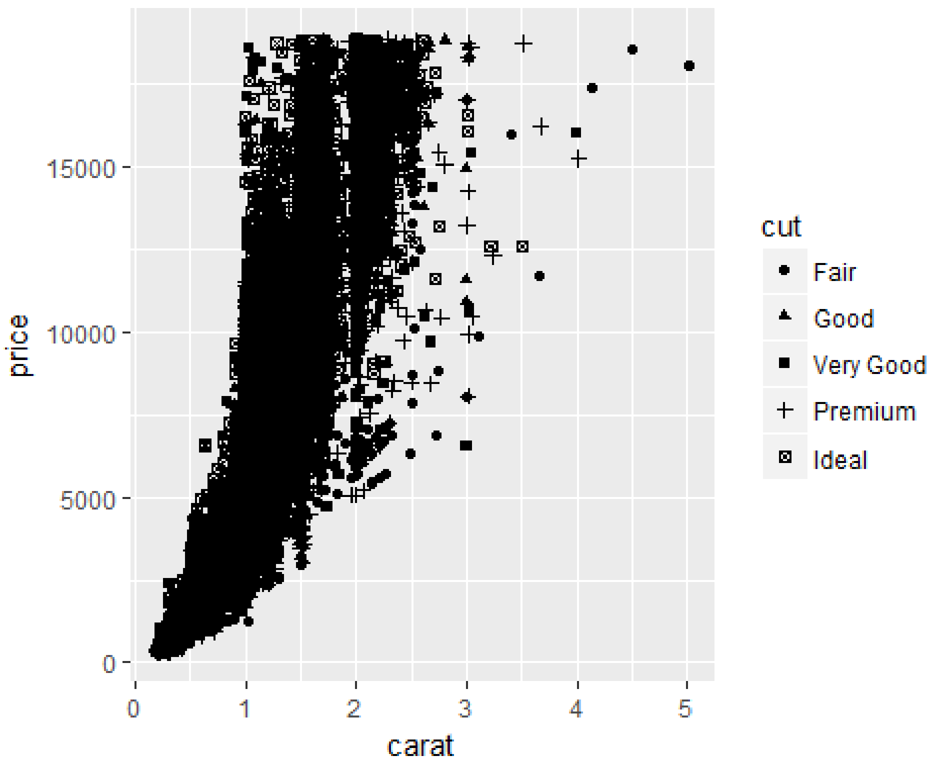


```
ggplot(data = huron, aes(x = year, y = level)) + geom_line(color = "red")
```



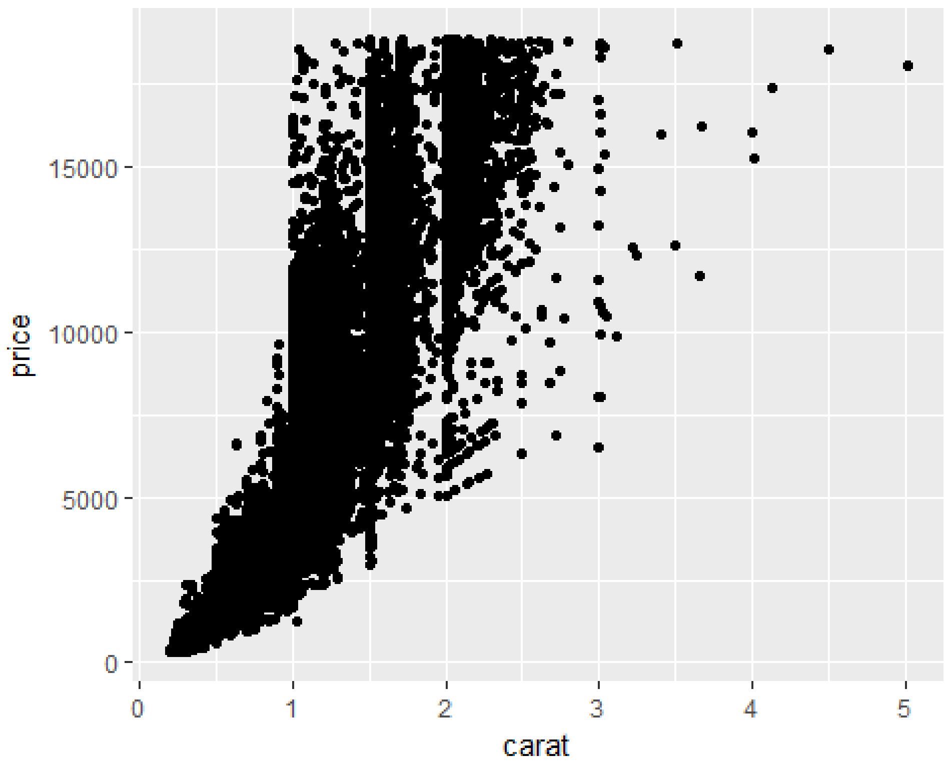
4. Facets

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

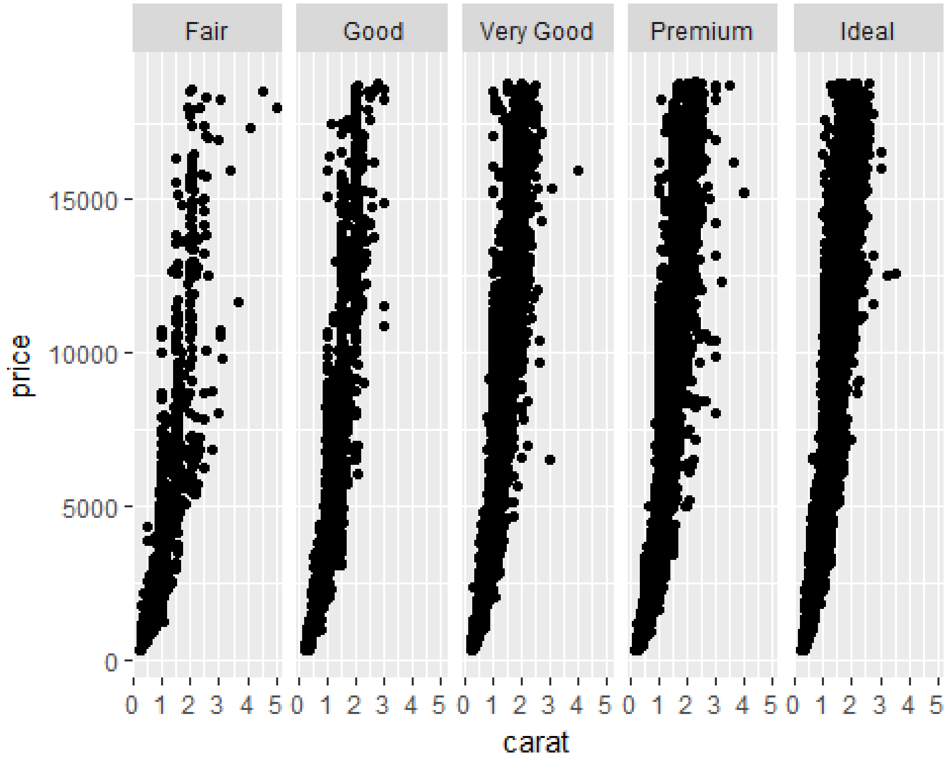


If you need to draw a scatter plot for each cut type in the separate window use facet_grid(y~x)

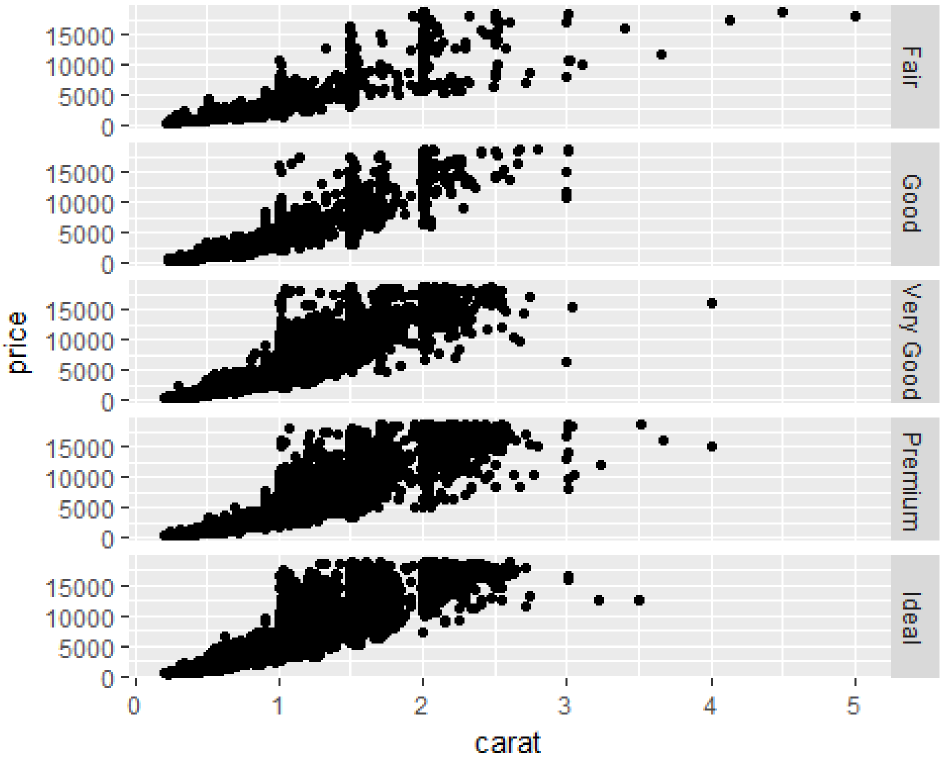
```
p <- ggplot(data = diamonds, aes(x = carat, y = price)) + geom_point()

p
```



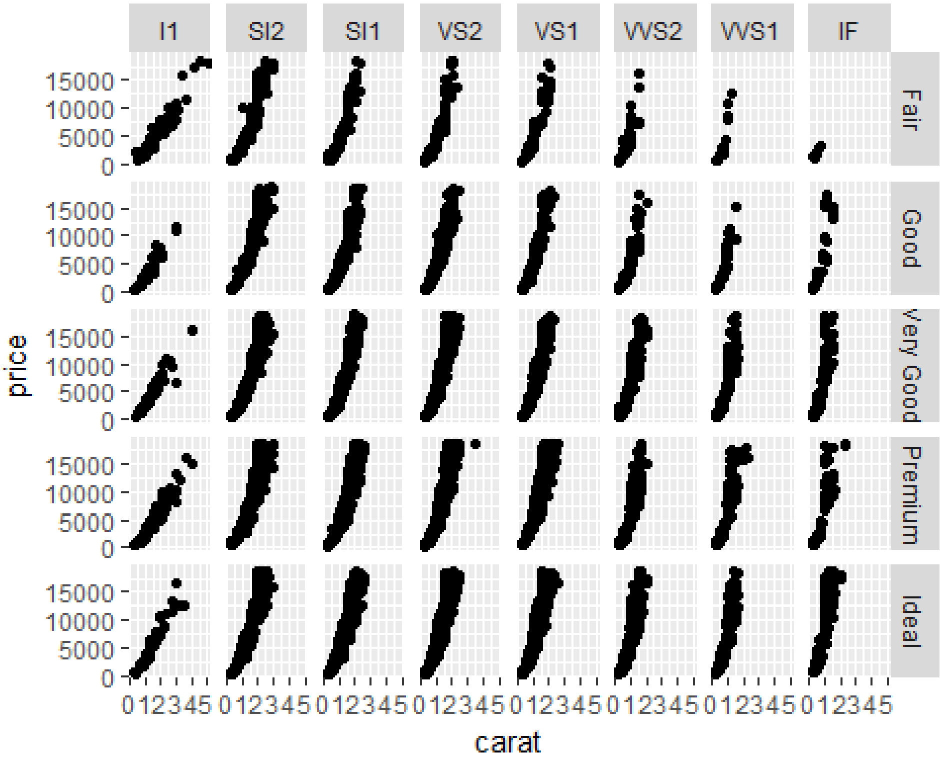
```
p + facet_grid(.~cut)
```



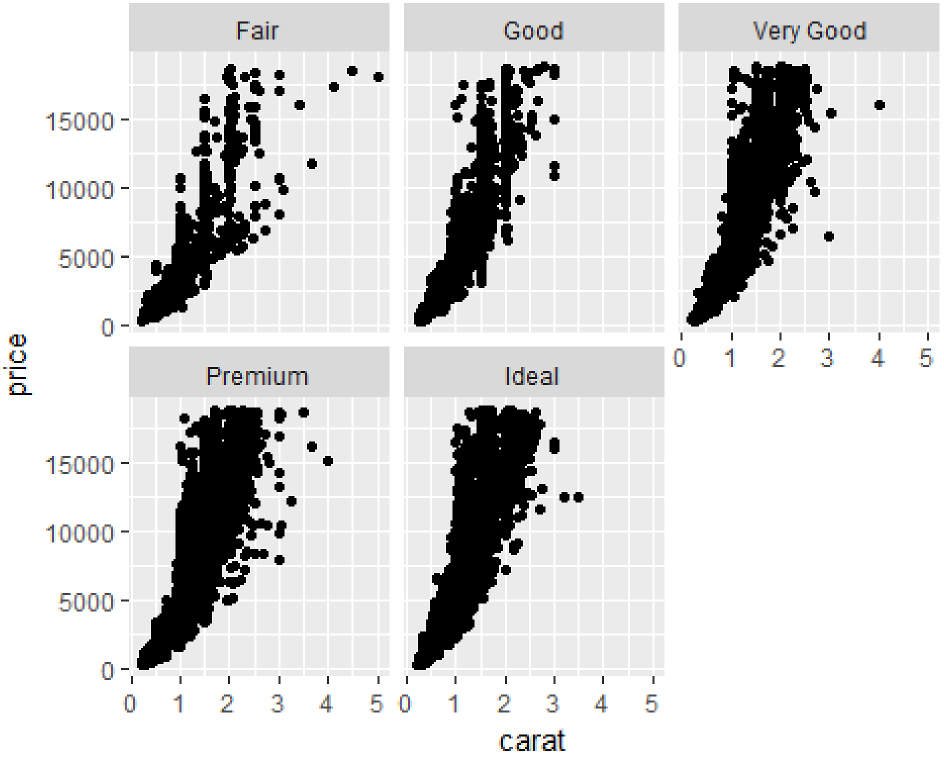
```
p + facet_grid(cut~.)
```



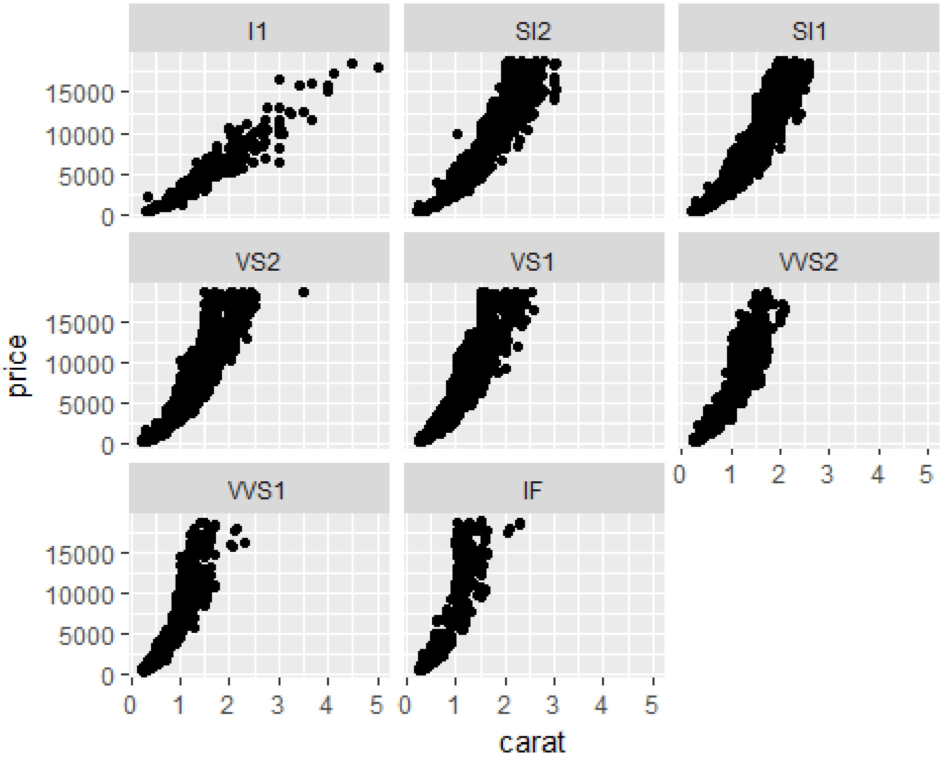
```
p + facet_grid(cut~clarity)
```



```
p +facet_wrap(~cut)
```

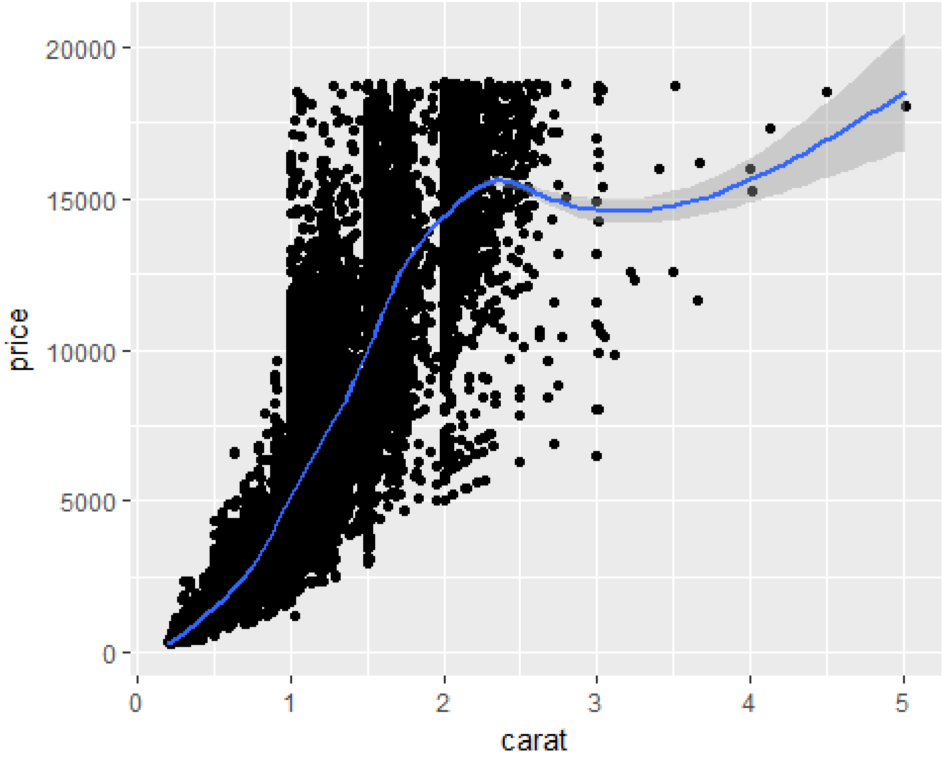


```
p +facet_wrap(~clarity)
```

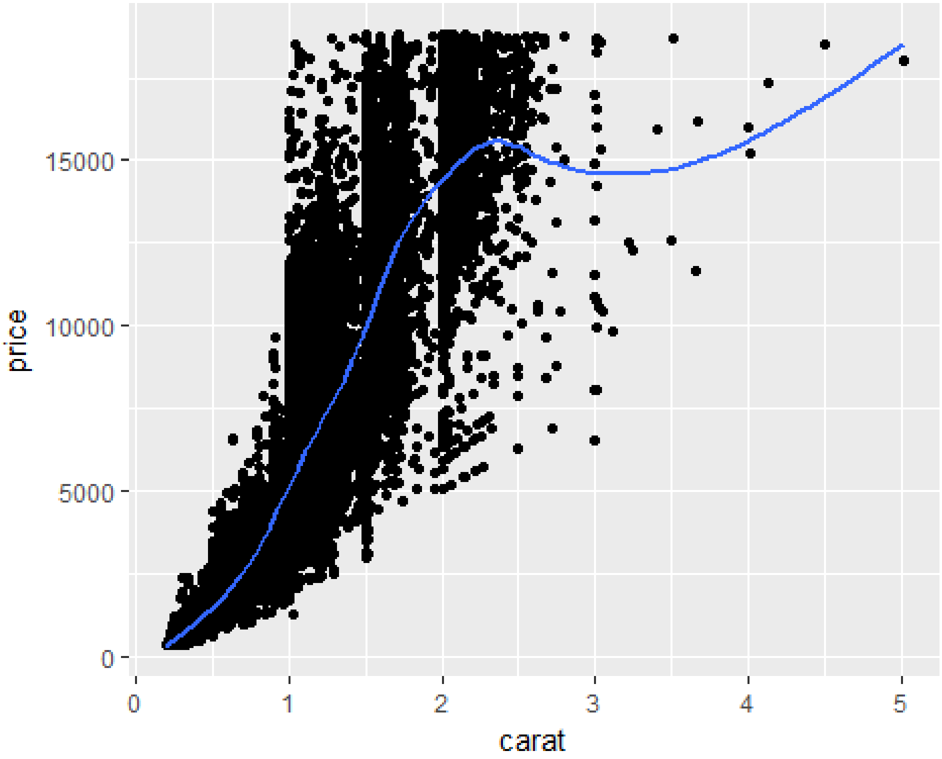


5. Statistics

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

p + geom_smooth()

## `geom_smooth()` using method = 'gam'
```

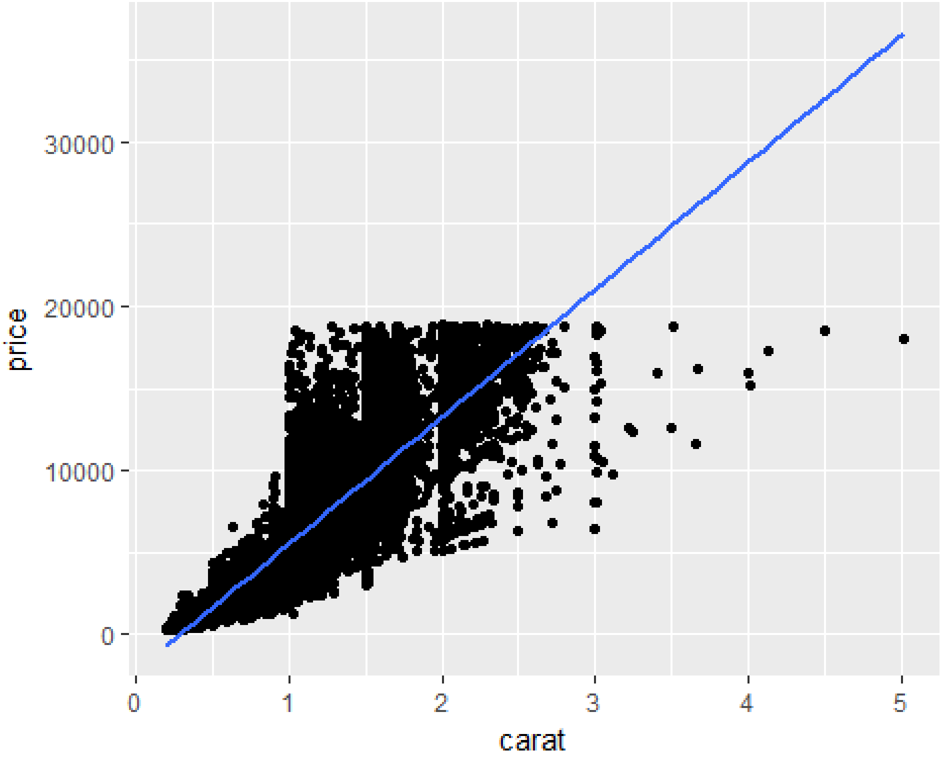


```
p + geom_smooth(se = FALSE)

## `geom_smooth()` using method = 'gam'
```

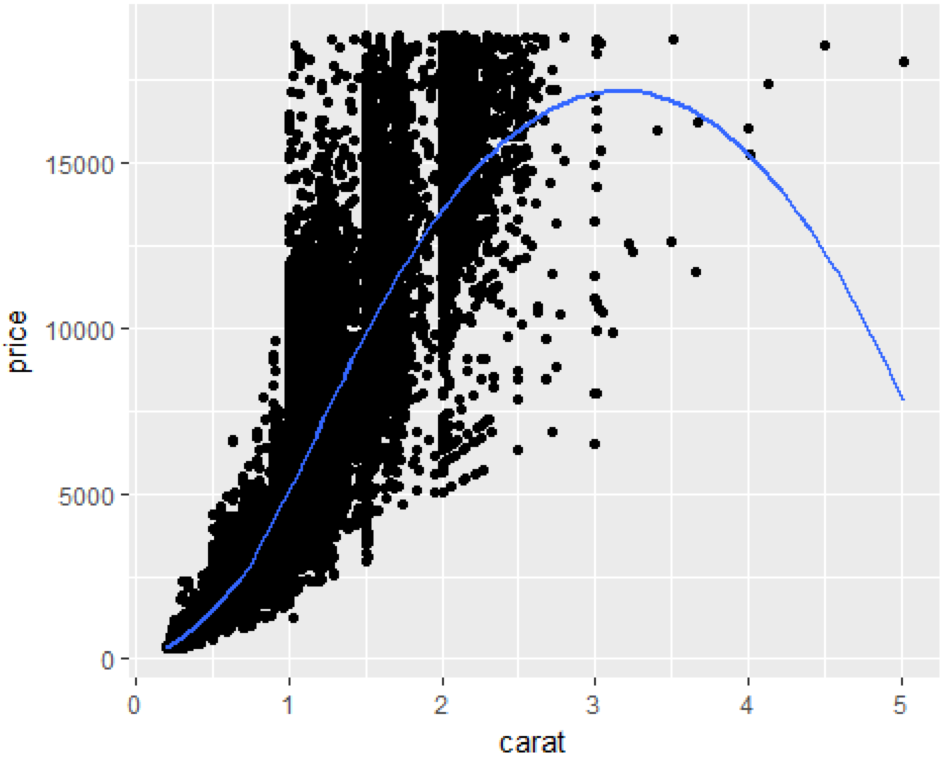


Methods for stat_smooth: lm", "glm", "gam", "loess", "rlm"

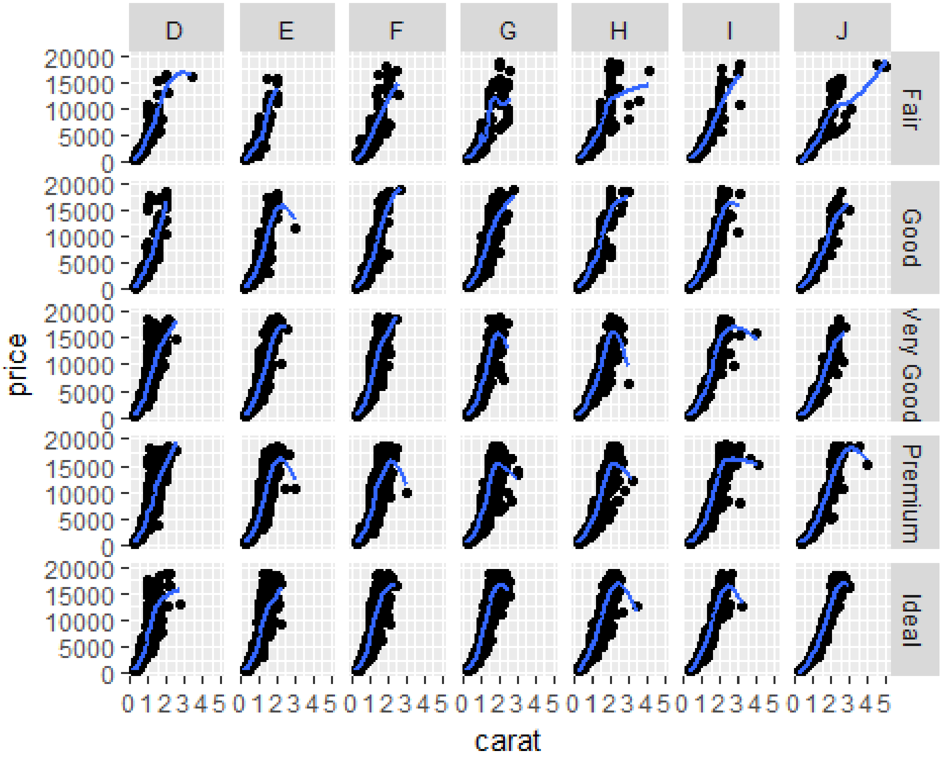
```
p + geom_smooth(method = "lm", se= FALSE)
```



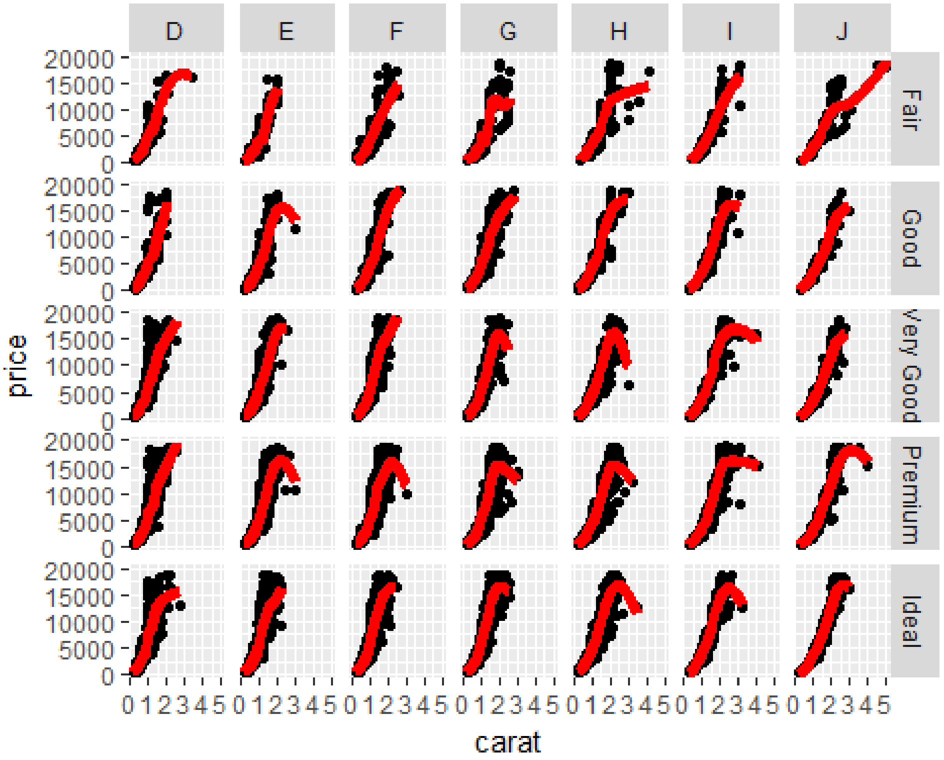
```
p + stat_smooth (method = "loess", se = FALSE)
```



```
p + facet_grid(cut~color)  + geom_smooth(se= FALSE)

## `geom_smooth()` using method = 'gam'
```



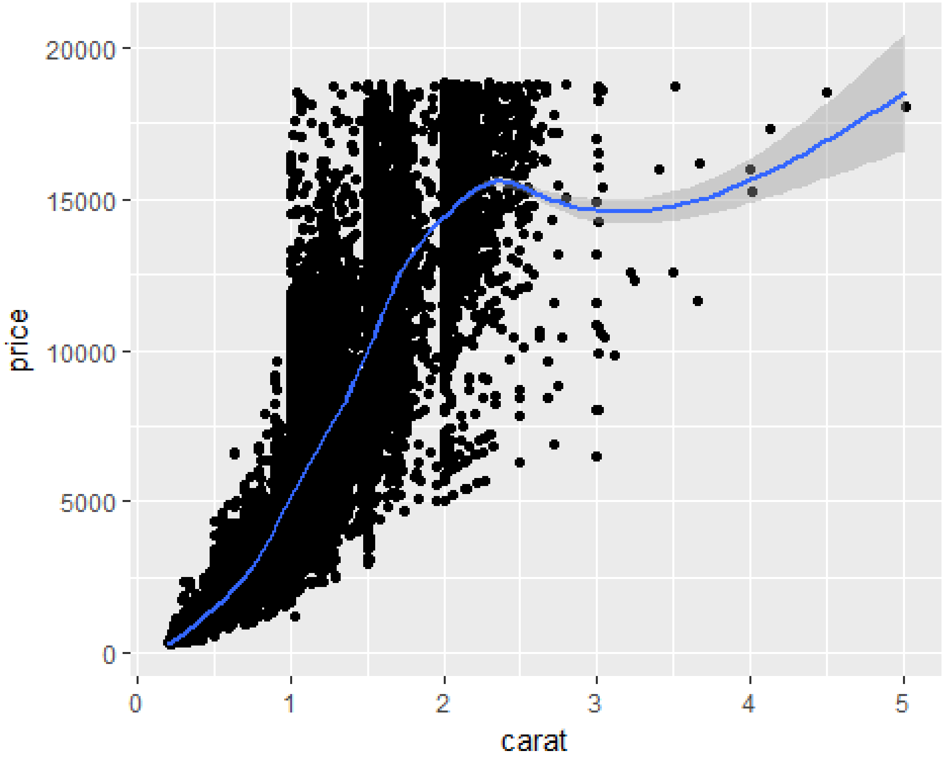
```
p + facet_grid(cut~color)  + geom_smooth(se= FALSE, color = "red", size = 2)

## `geom_smooth()` using method = 'gam'
```



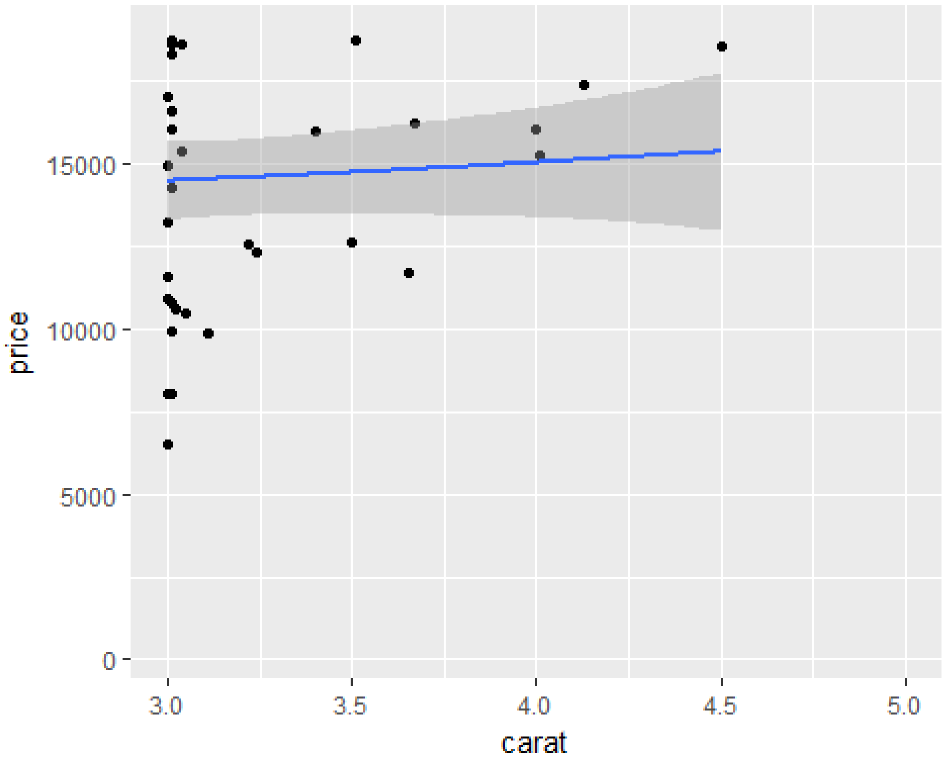
6. Coordinate System

Zoom

```
p <- ggplot(data = diamonds, aes(x = carat, y = price)) + geom_point() +
geom_smooth()
p

## `geom_smooth()` using method = 'gam'
```

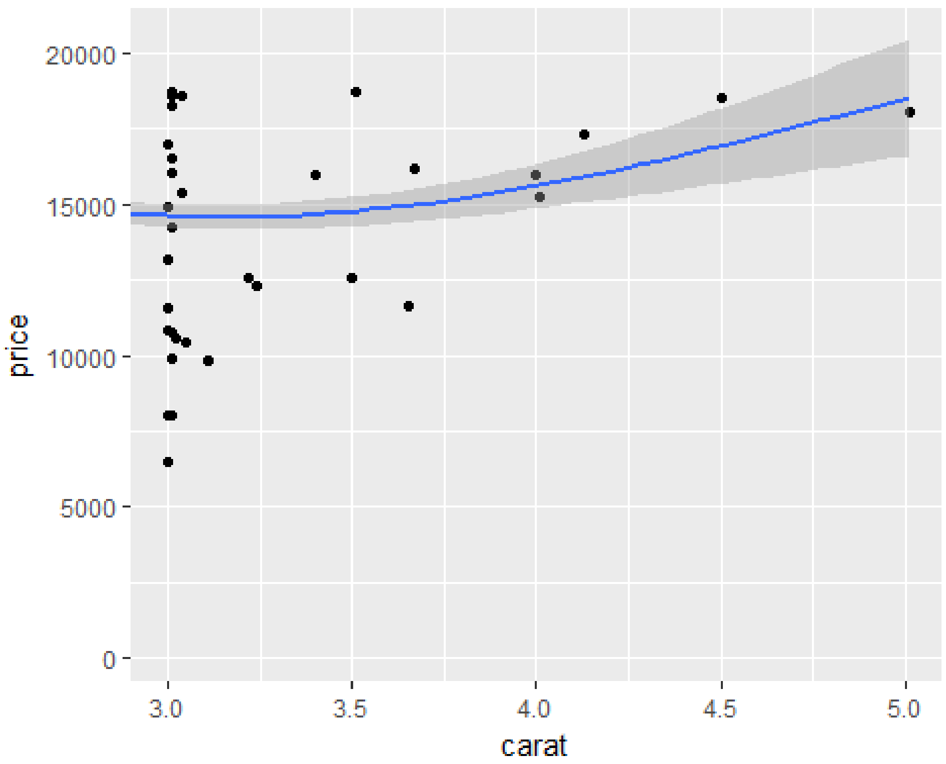


```
p + xlim(c(3, 5))

## `geom_smooth()` using method = 'gam'

## Warning: Removed 53901 rows containing non-finite values (stat_smooth).

## Warning: Removed 53901 rows containing missing values (geom_point).
```

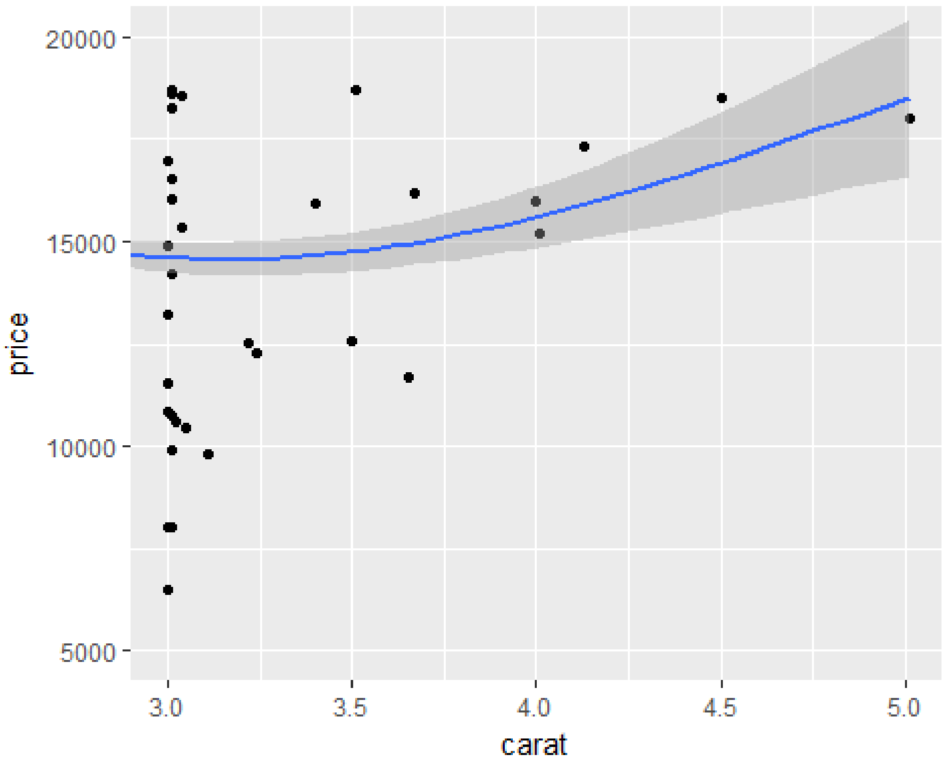


The above snapshot ignored all points beyond the limit

```
p + coord_cartesian(xlim = c(3,5))

## `geom_smooth()` using method = 'gam'
```



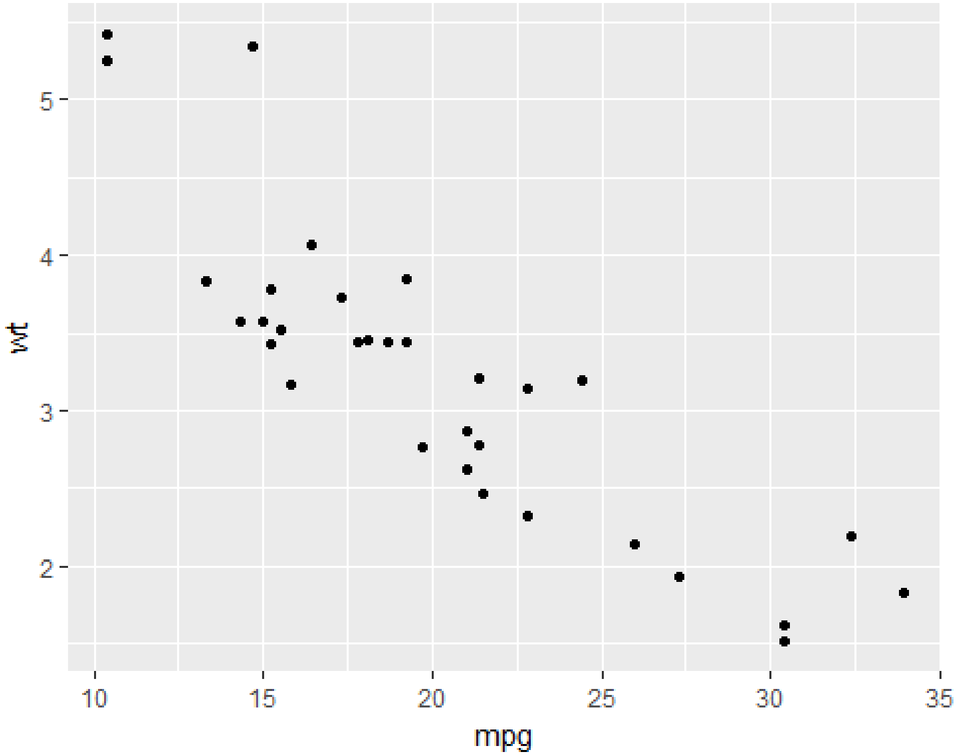
```
p + coord_cartesian(xlim = c(3,5), ylim = c(5000,20000))

## `geom_smooth()` using method = 'gam'
```



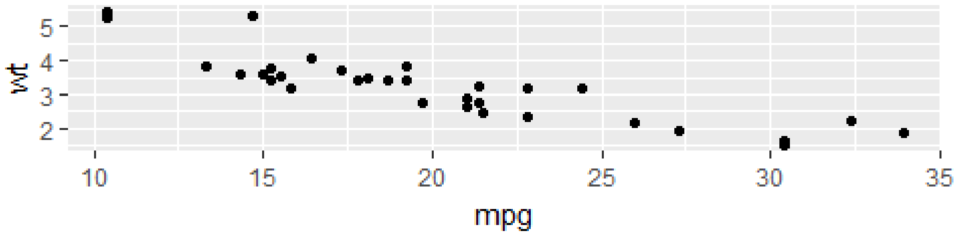
Scale

The default,ratio = 1, ensures that one unit on the x-axis is the same length as one unit on the y-axis.

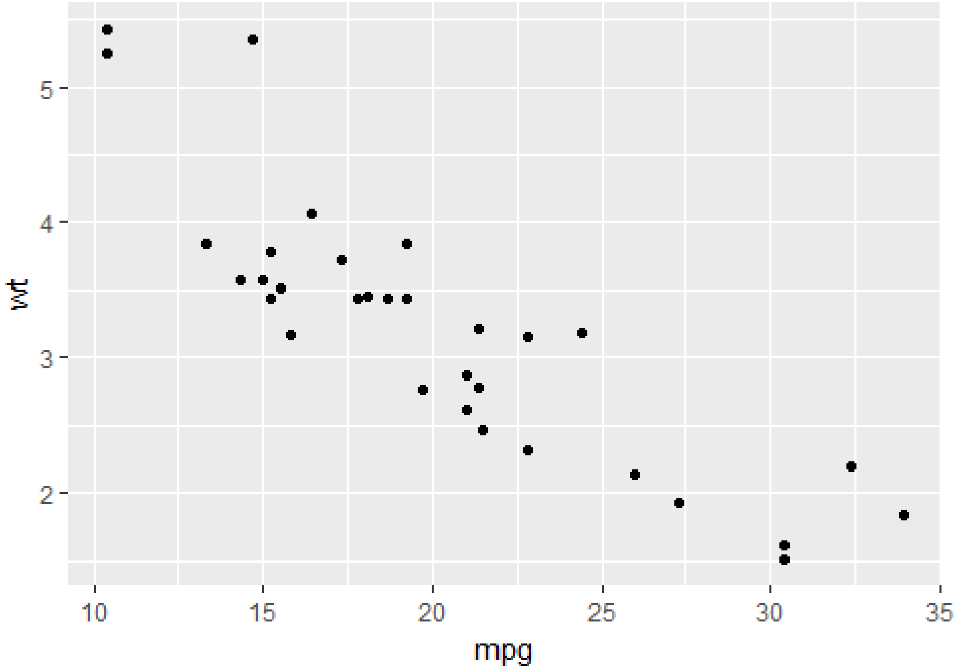
```
p <- ggplot(mtcars, aes(mpg, wt)) + geom_point()
p
```



```
p + coord_fixed(ratio = 1)
```

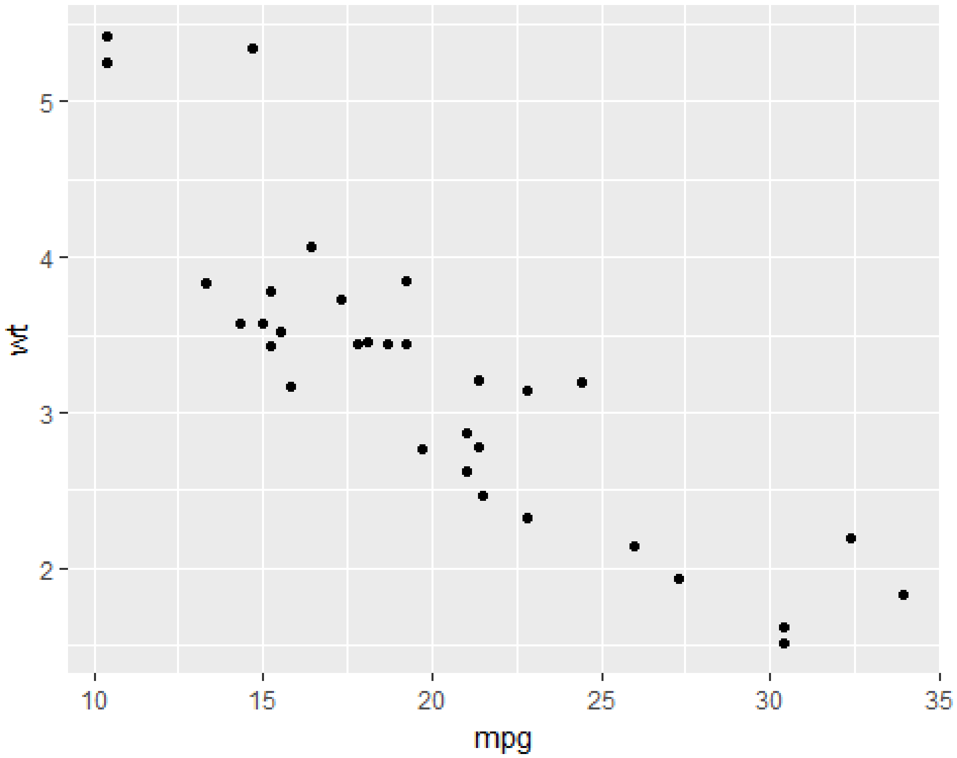


```
p + coord_fixed(ratio = 4)
```

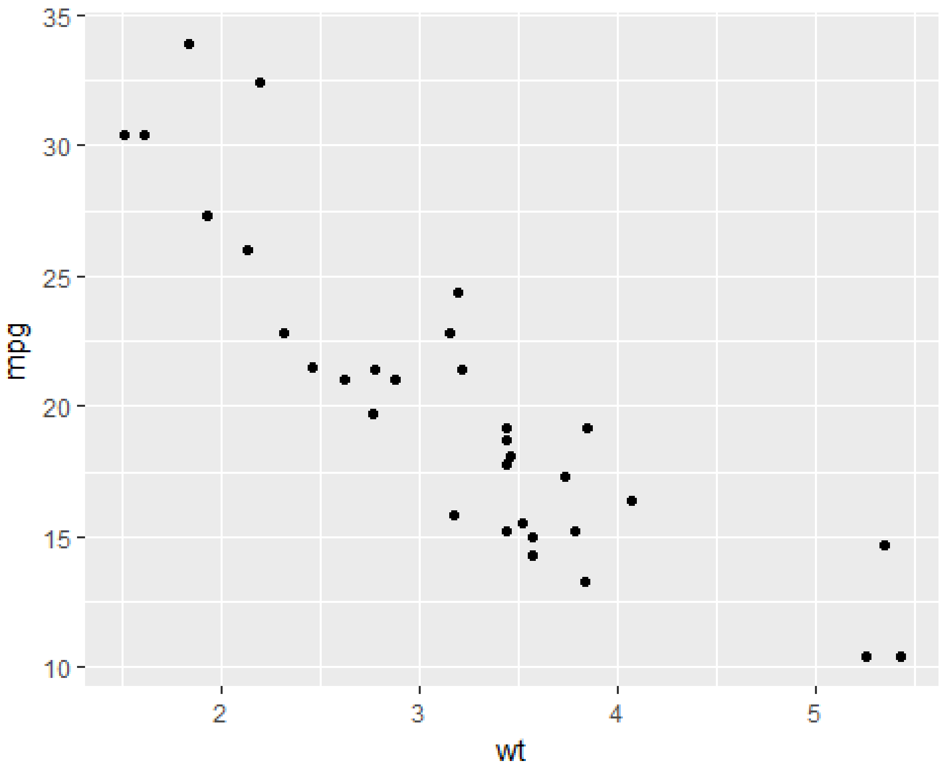


Flip

```
p <- ggplot(mtcars, aes(mpg, wt)) + geom_point()
p
```

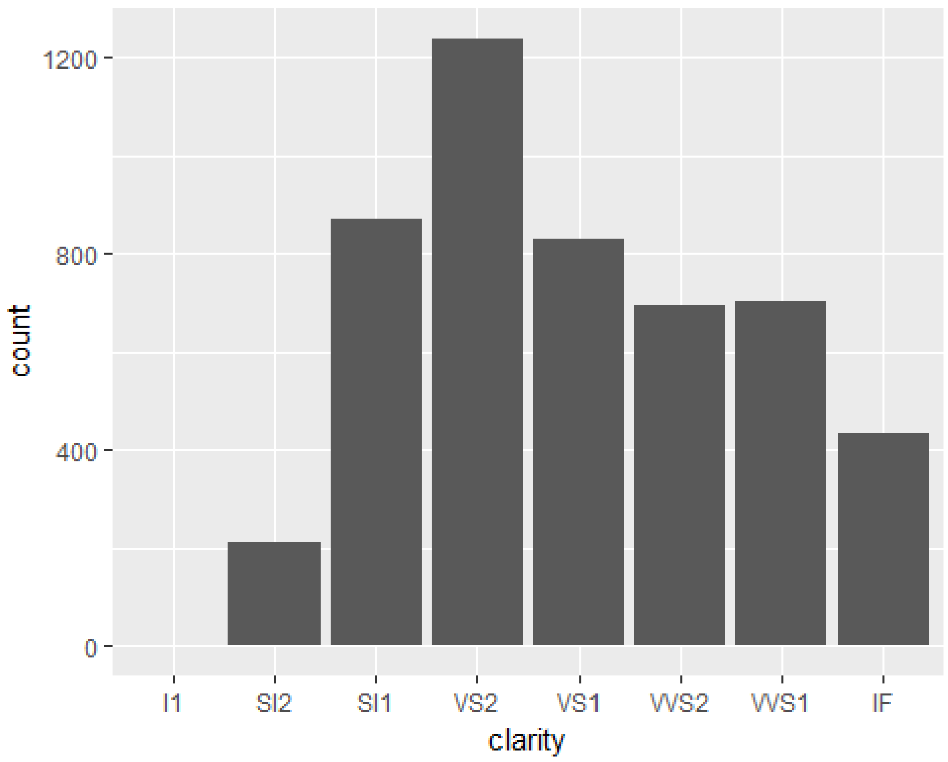


```
p + coord_flip()
```

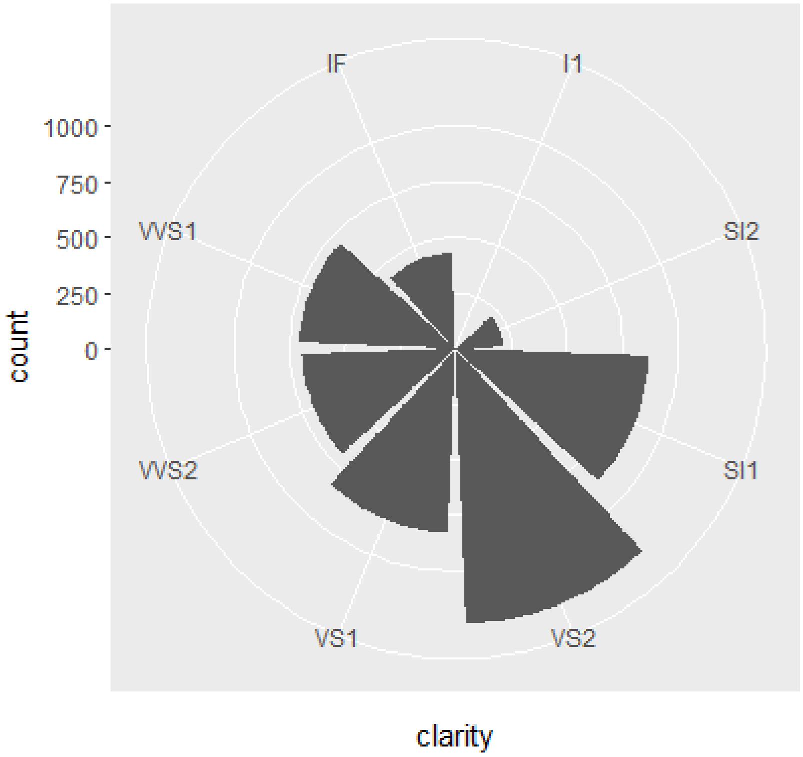


Polar - Pie Chart

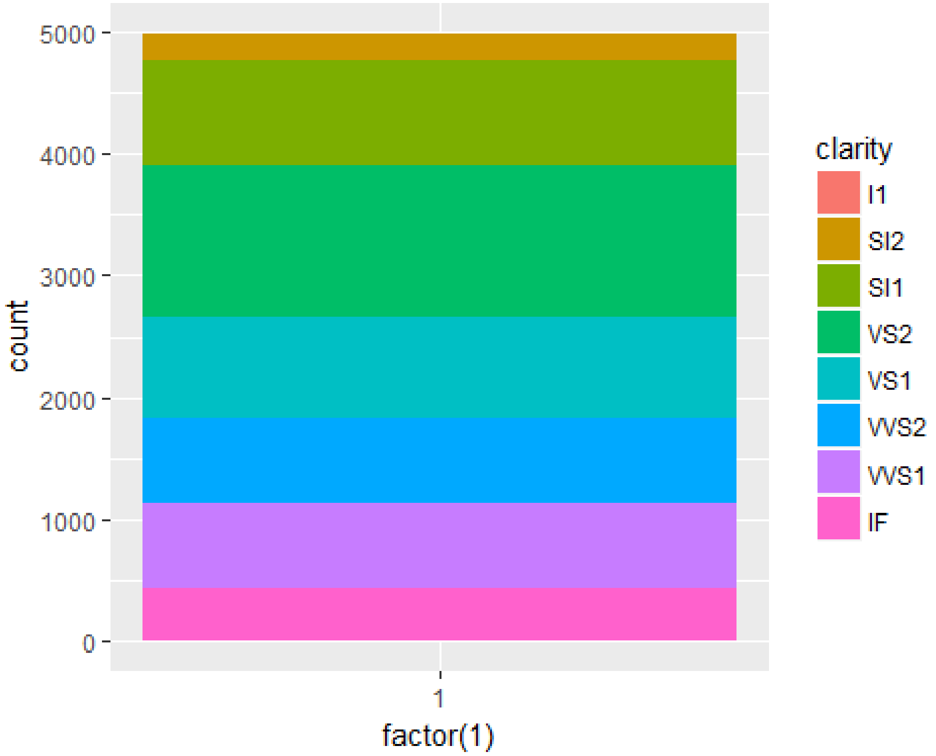
```
p <- ggplot(data = d2931, aes(x = clarity)) + geom_bar()
p
```



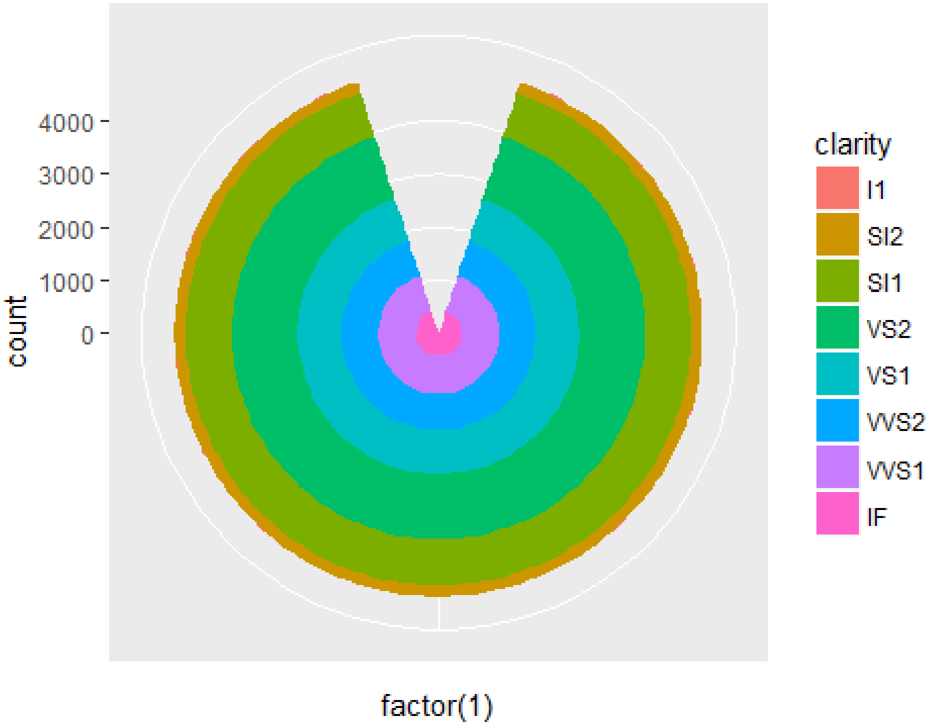
```
p + coord_polar()
```



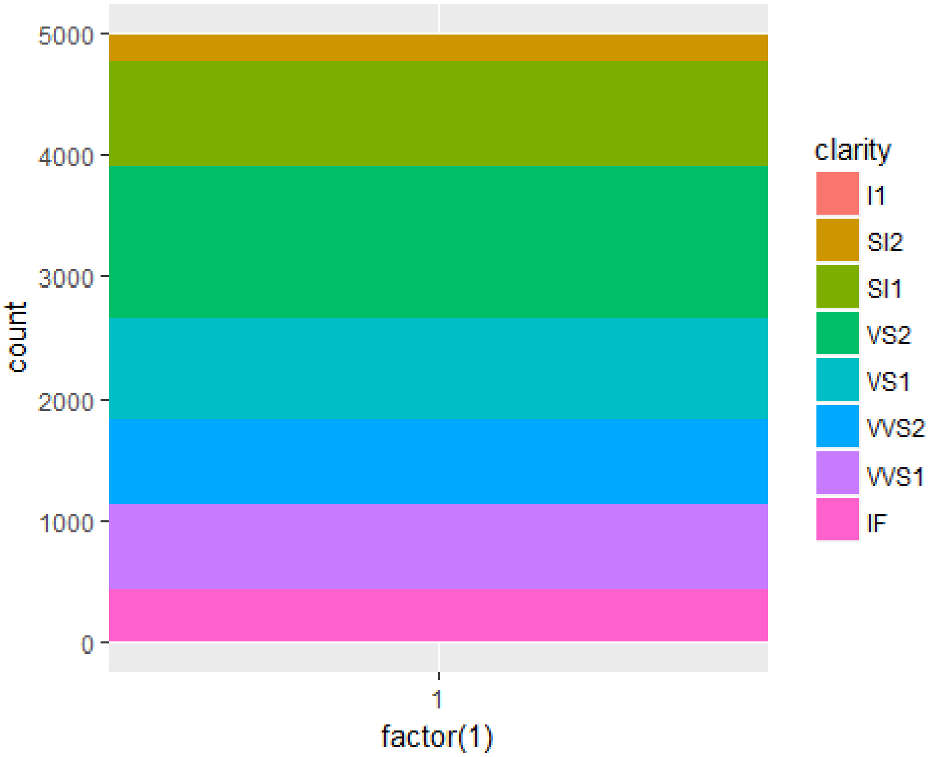
```
p <- ggplot(data = d2931, aes(x = factor(1), fill = clarity)) + geom_bar()
p
```



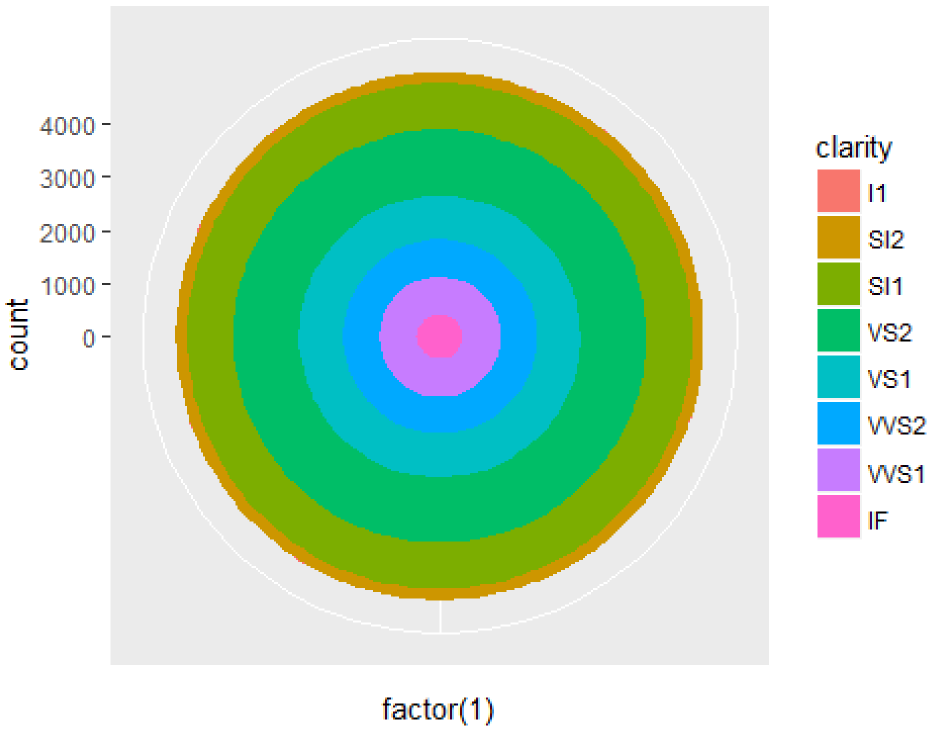
```
p + coord_polar()
```



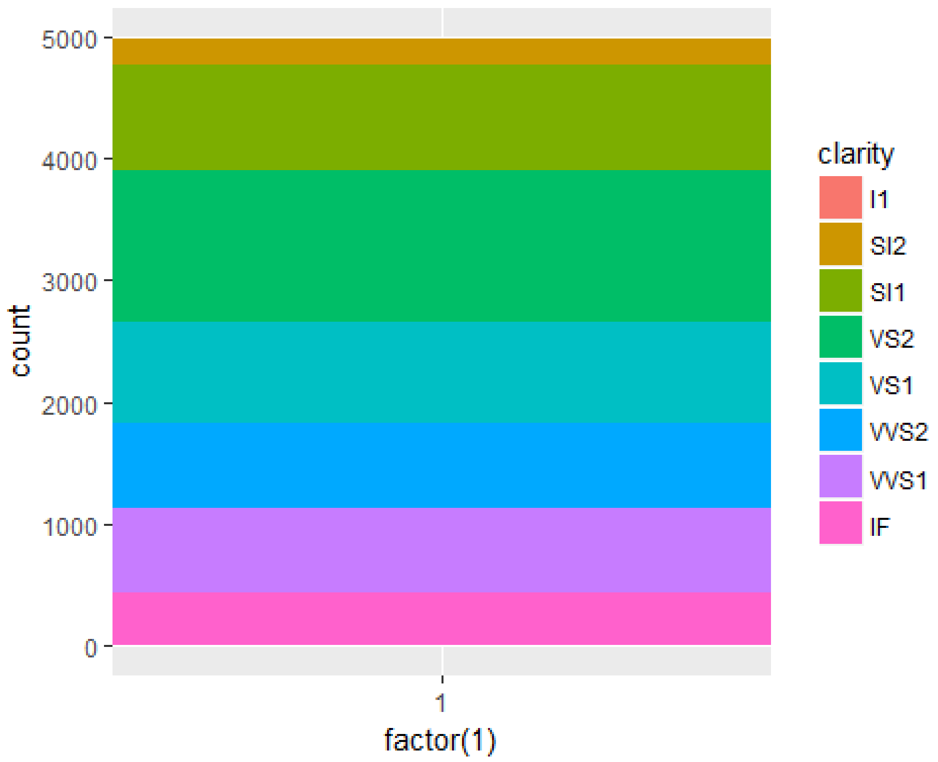
```
p <- ggplot(data = d2931, aes(x = factor(1), fill = clarity)) +
geom_bar(width = 1)
p
```



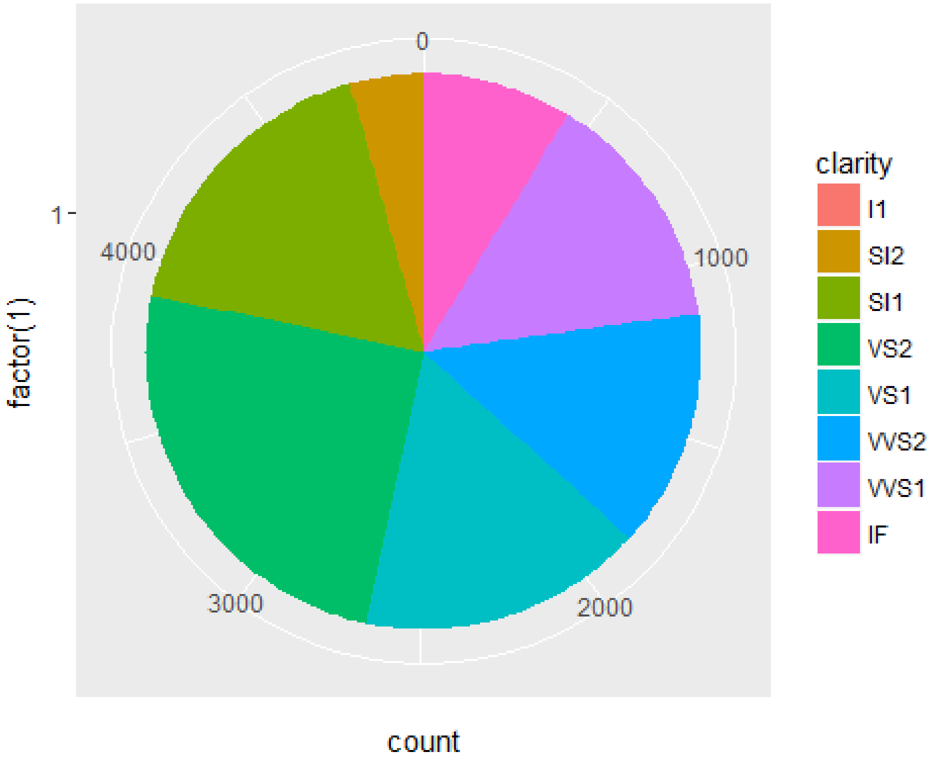
```
p + coord_polar()
```



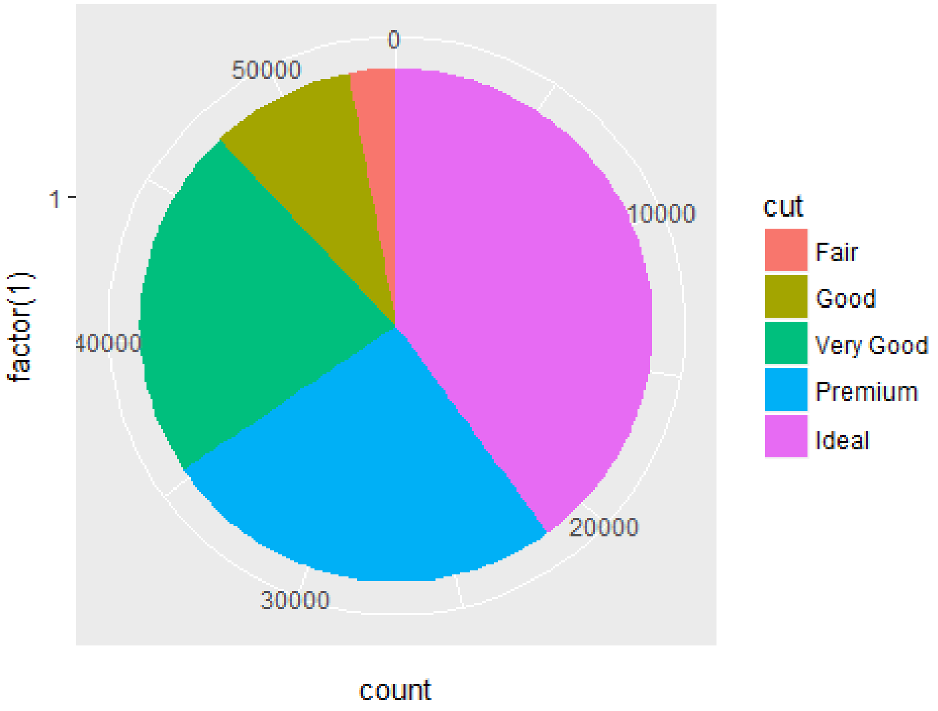
```
p <- ggplot(data = d2931, aes(x = factor(1), fill = clarity)) +
geom_bar(width = 1)
p
```



```
p + coord_polar(theta = "y")
```

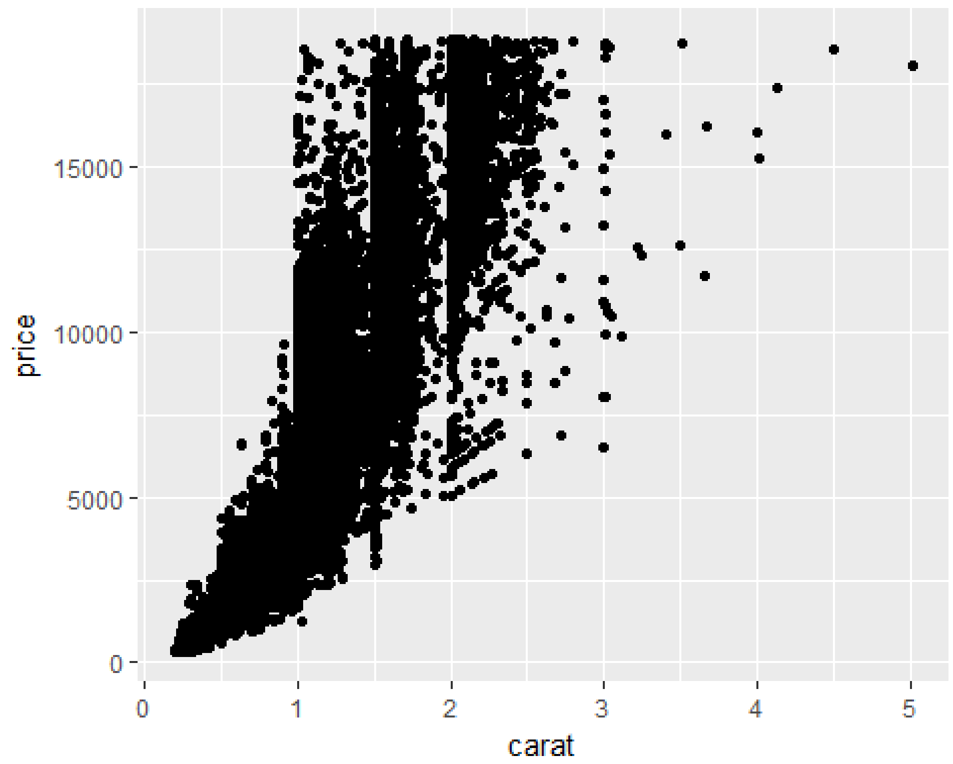


```
ggplot(data = diamonds) +
  geom_bar(mapping = aes(x = factor(1), fill = cut), width = 1) +
  coord_polar(theta = "y")
```

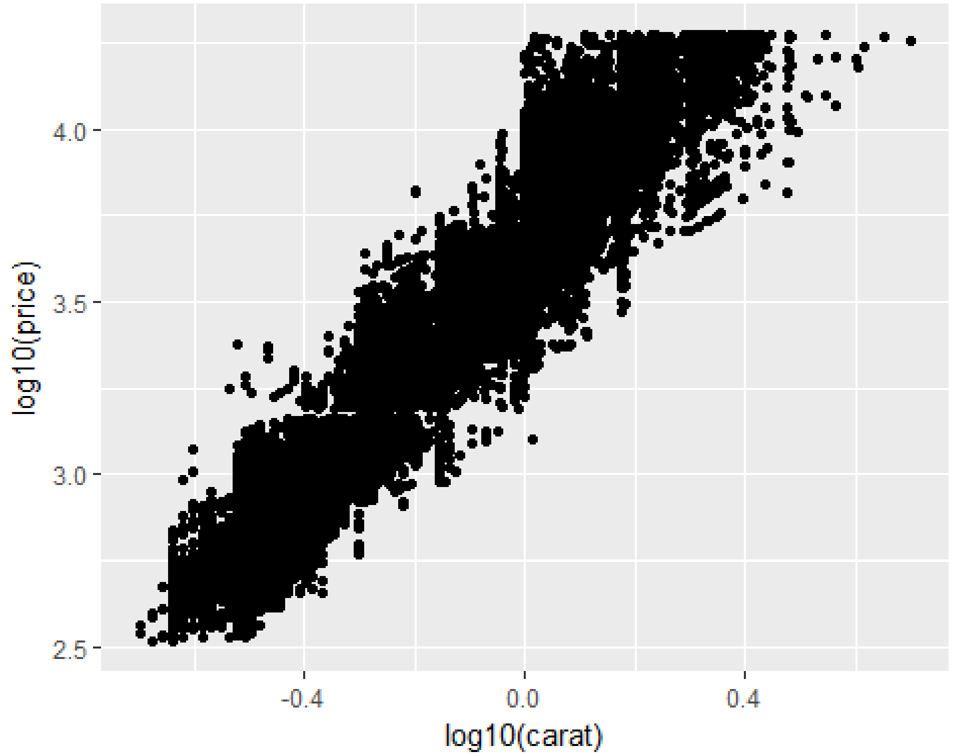


Transform

```
ggplot(diamonds, aes(carat, price)) + geom_point()
```



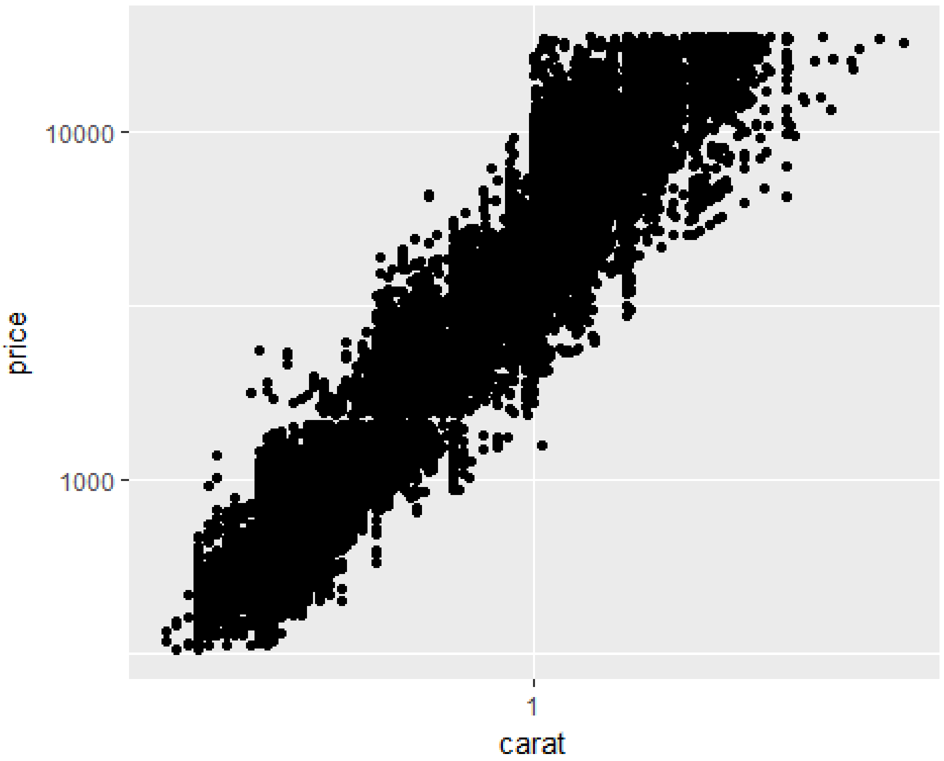
```
ggplot(diamonds, aes(log10(carat), log10(price))) + geom_point()
```



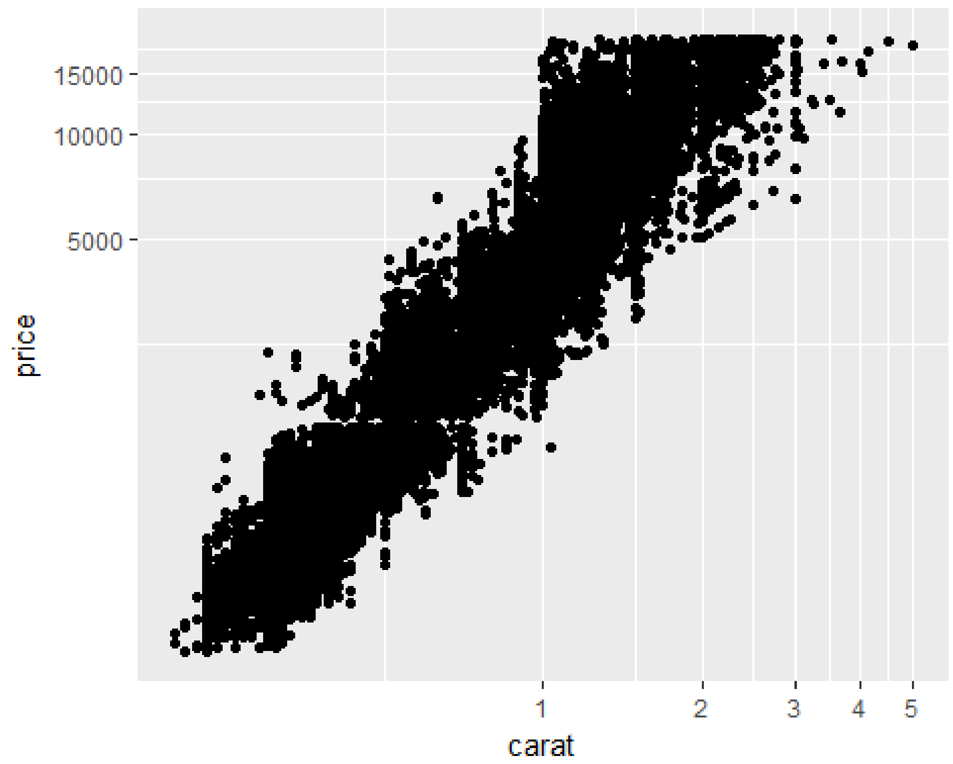
log(1) = 0, log(10)=1, log(1000) = 3

Tranform scale

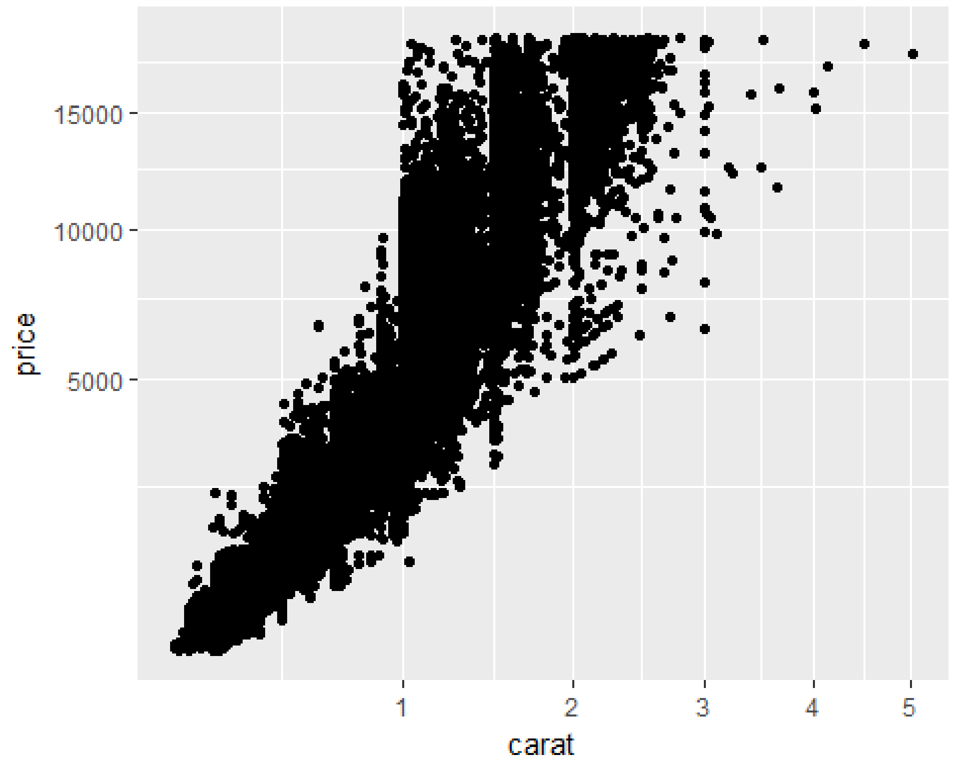
```
ggplot(diamonds, aes(carat, price)) + geom_point() +
  scale_x_log10() +
  scale_y_log10()
```



```
ggplot(diamonds, aes(carat, price)) + geom_point() +
  coord_trans(x = "log10", y = "log10")
```

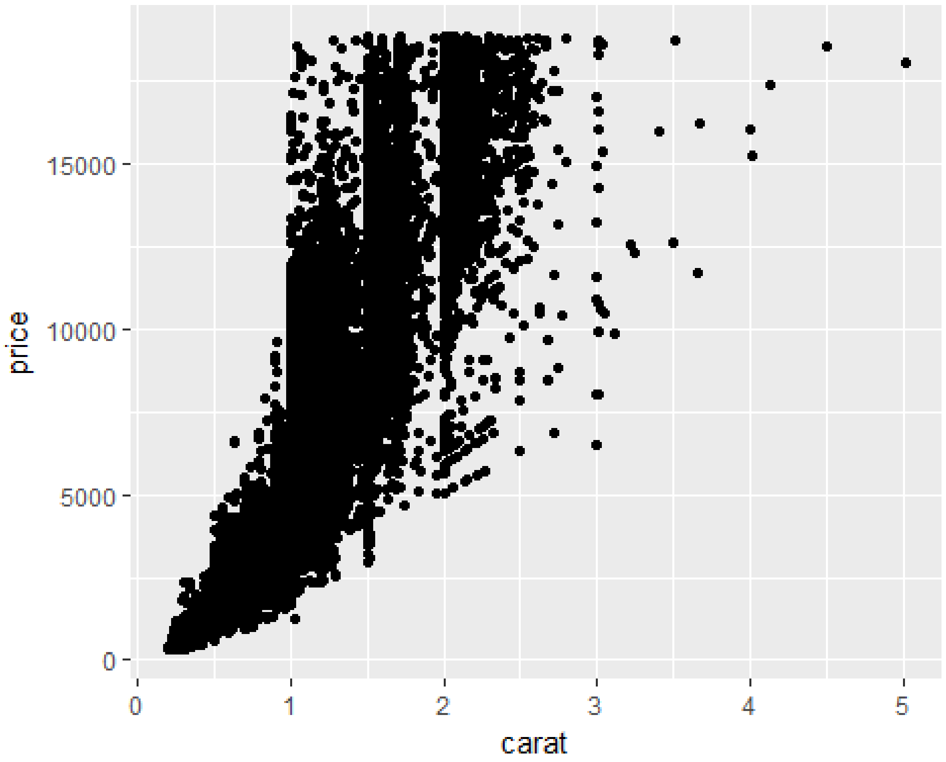


```r
ggplot(diamonds, aes(carat, price)) + geom_point() +
  coord_trans(x = "sqrt", y = "sqrt")
```

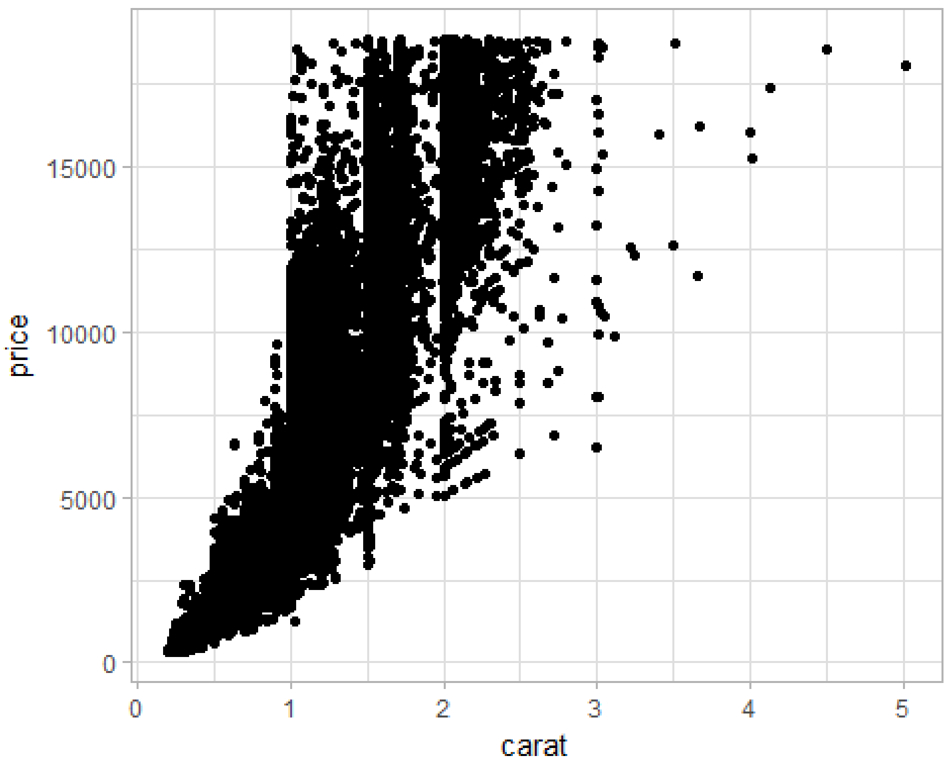


7. Theme

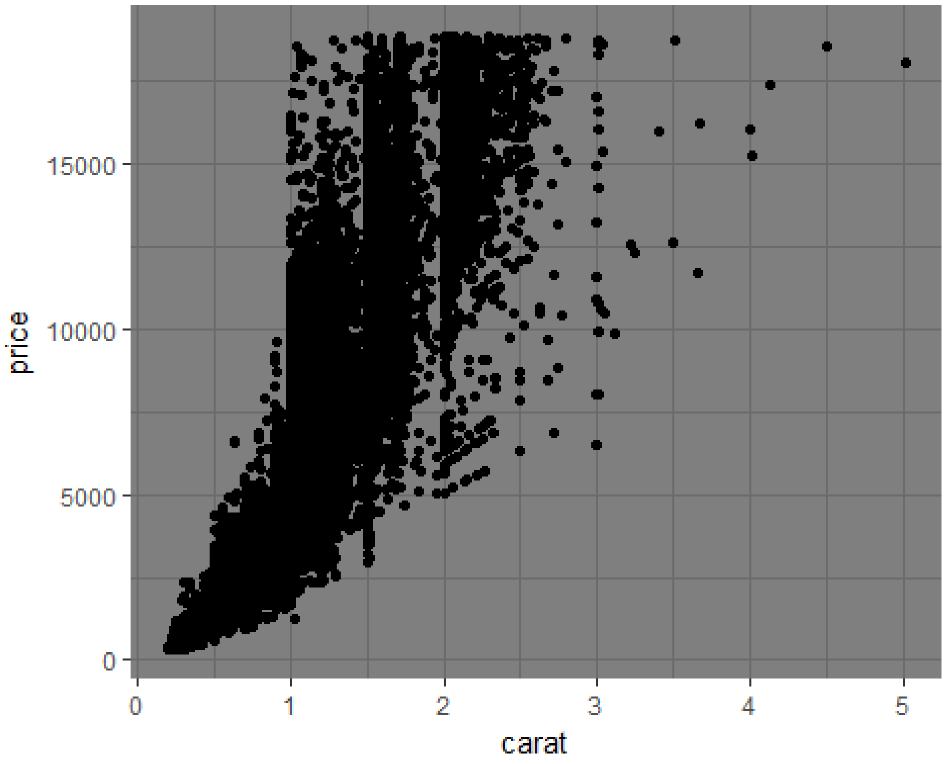
```
ggplot(diamonds, aes(carat, price)) + geom_point()
```



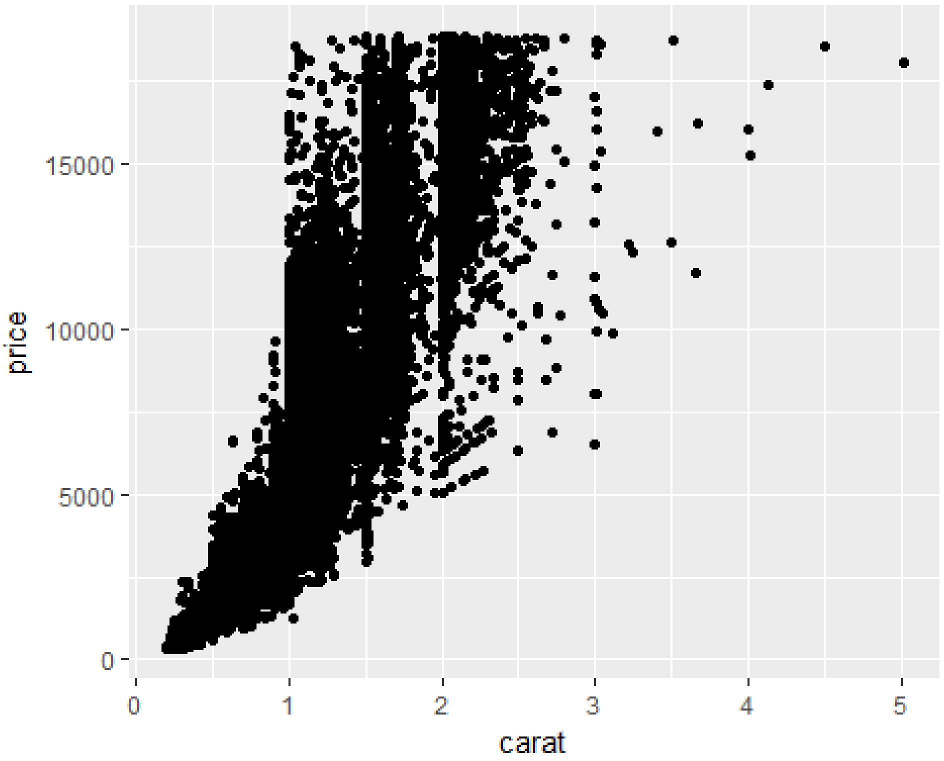
```
ggplot(diamonds, aes(carat, price)) + geom_point() + theme_light()
```



```
ggplot(diamonds, aes(carat, price)) + geom_point() + theme_dark()
```



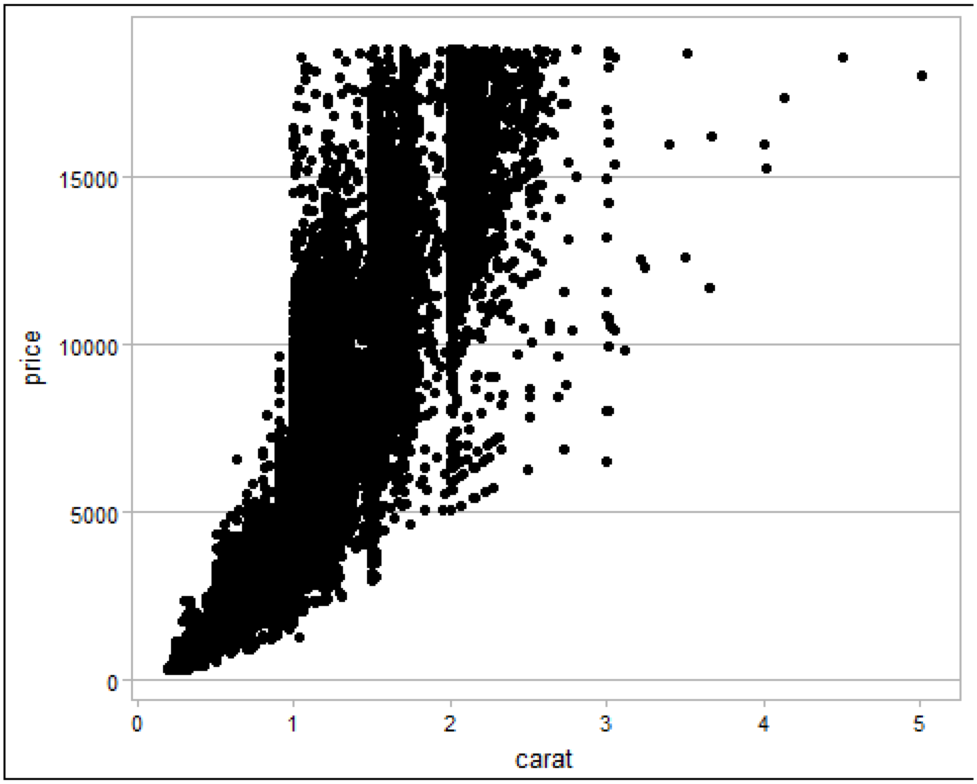
```
ggplot(diamonds, aes(carat, price)) + geom_point() + theme_gray()
```



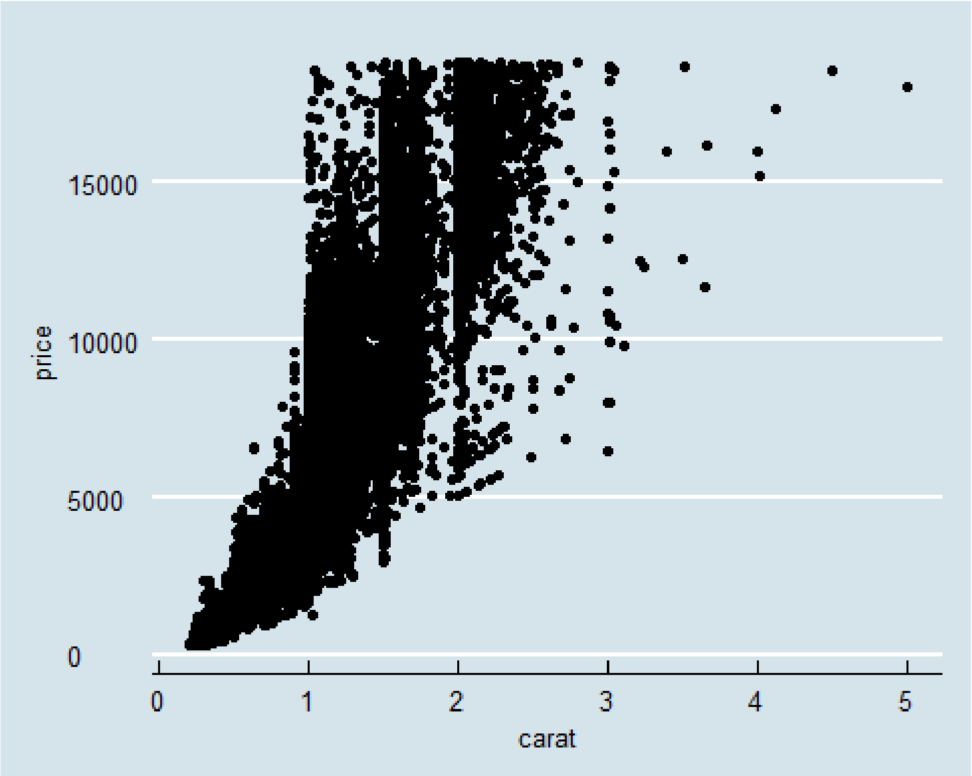
install.packages("ggthemes")

```
## Warning: package 'ggthemes' was built under R version 3.4.3
```

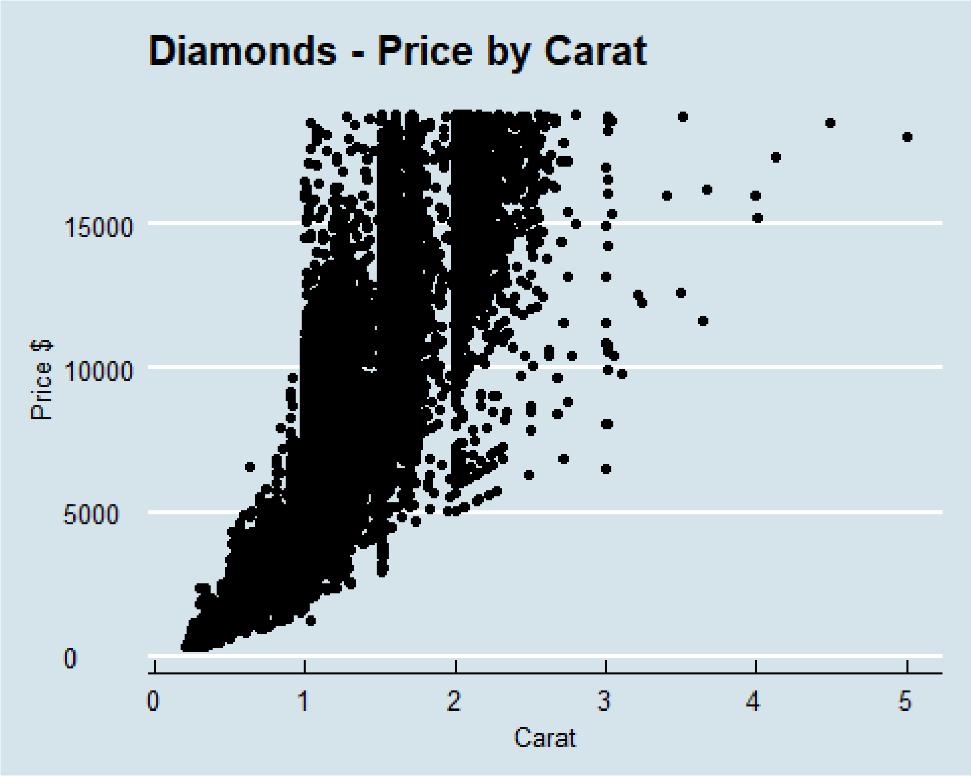
```
ggplot(diamonds, aes(carat, price)) + geom_point() + theme_calc()
```



```
ggplot(diamonds, aes(carat, price)) + geom_point() + theme_economist()
```



```
ggplot(diamonds, aes(carat, price)) + geom_point() + theme_economist() +
labs(title= "Diamonds - Price by Carat",
     y="Price $", x = "Carat")
```


Diamonds - Price by Carat
Price $
Carat