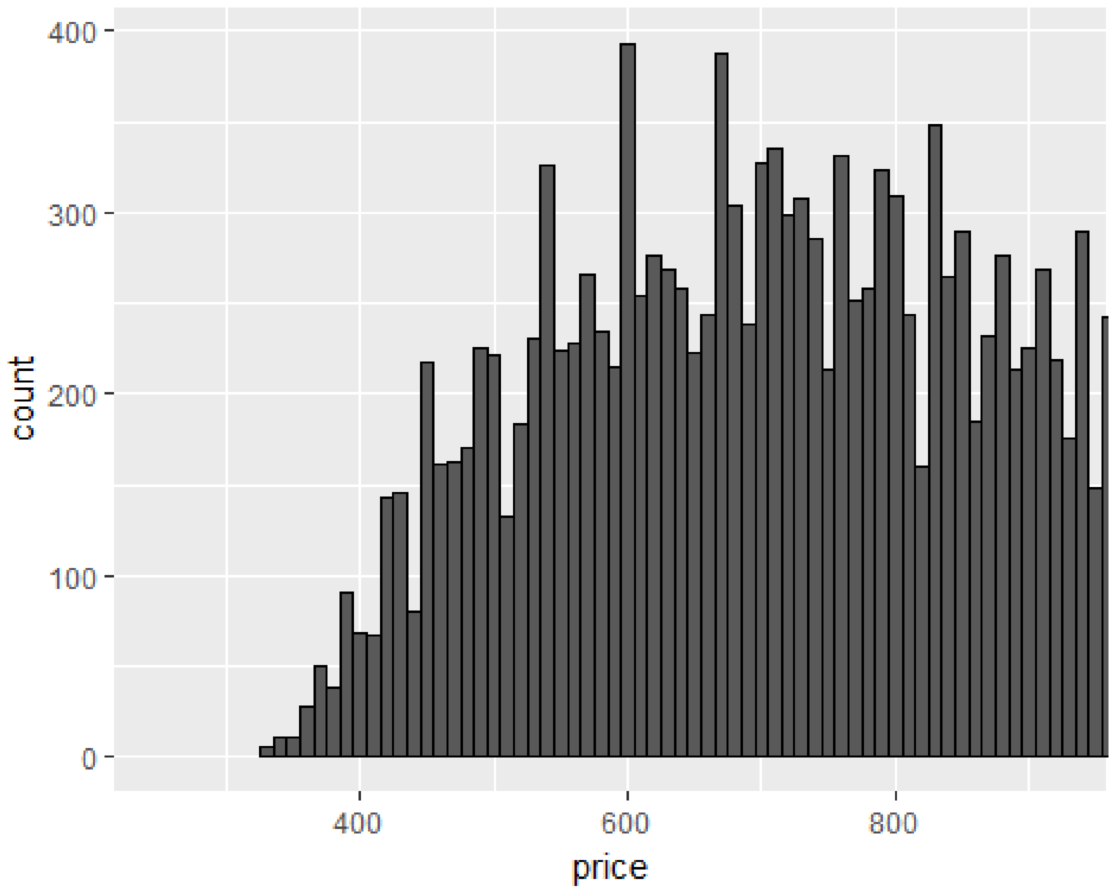
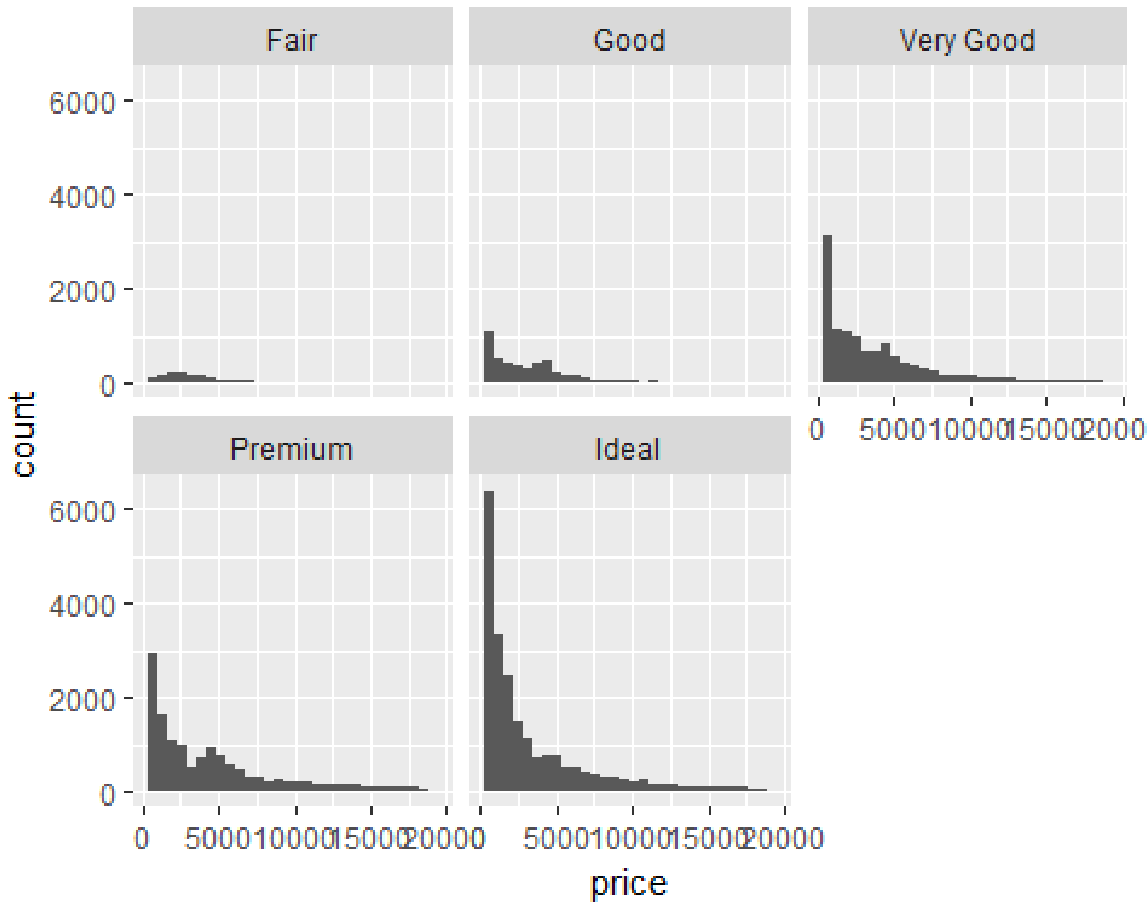
Diamonds

```
#Diamonds at certain prices
subset(diamonds, price<500)

## # A tibble: 1,729 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  4.20  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
##  7  0.24 Very Good     I    VVS1  62.3    57   336  3.95  3.98  2.47
##  8  0.26 Very Good     H     SI1  61.9    55   337  4.07  4.11  2.53
##  9  0.22      Fair     E     VS2  65.1    61   337  3.87  3.78  2.49
## 10  0.23 Very Good     H     VS1  59.4    61   338  4.00  4.05  2.39
## # ... with 1,719 more rows

subset(diamonds, price<250)

## # A tibble: 0 x 10
## # ... with 10 variables: carat <dbl>, cut <ord>, color <ord>,
## #   clarity <ord>, depth <dbl>, table <dbl>, price <int>, x <dbl>,
## #   y <dbl>, z <dbl>

subset(diamonds, price>=15000)

## # A tibble: 1,656 x 10
##    carat       cut color clarity depth table price     x     y     z
##    <dbl>     <ord> <ord>   <ord> <dbl> <dbl> <int> <dbl> <dbl> <dbl>
##  1  1.60     Ideal     G     VS2  61.9    56 15000  7.53  7.47  4.64
##  2  1.54   Premium     E     VS2  62.3    58 15002  7.31  7.39  4.58
##  3  1.19     Ideal     F    VVS1  61.5    55 15005  6.82  6.84  4.20
##  4  2.10   Premium     I     SI1  61.5    57 15007  8.25  8.21  5.06
##  5  1.69     Ideal     D     SI1  60.8    57 15011  7.69  7.71  4.68
##  6  1.50 Very Good     G    VVS2  62.9    56 15013  7.22  7.32  4.57
##  7  1.73 Very Good     G     VS1  62.8    57 15014  7.57  7.72  4.80
##  8  2.02   Premium     G     SI2  63.0    59 15014  8.05  7.95  5.03
##  9  2.05 Very Good     F     SI2  61.9    56 15017  8.13  8.18  5.05
## 10  1.50 Very Good     F     VS1  61.6    58 15022  7.35  7.43  4.55
## # ... with 1,646 more rows
```

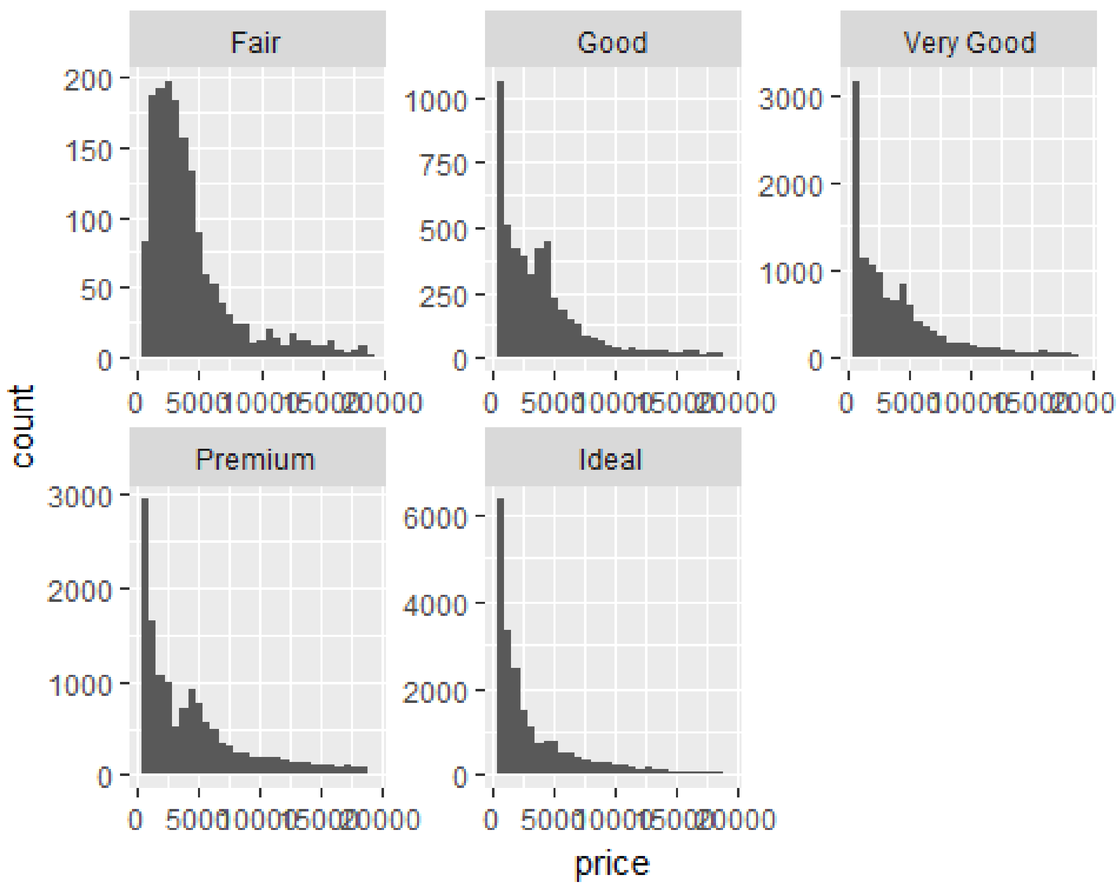
```r
#Exploring the largest peak area in the price histogram
ggplot(diamonds, aes(x=price)) +geom_histogram(color='black', binwidth=10)+
  coord_cartesian(xlim = c(250,925))
```



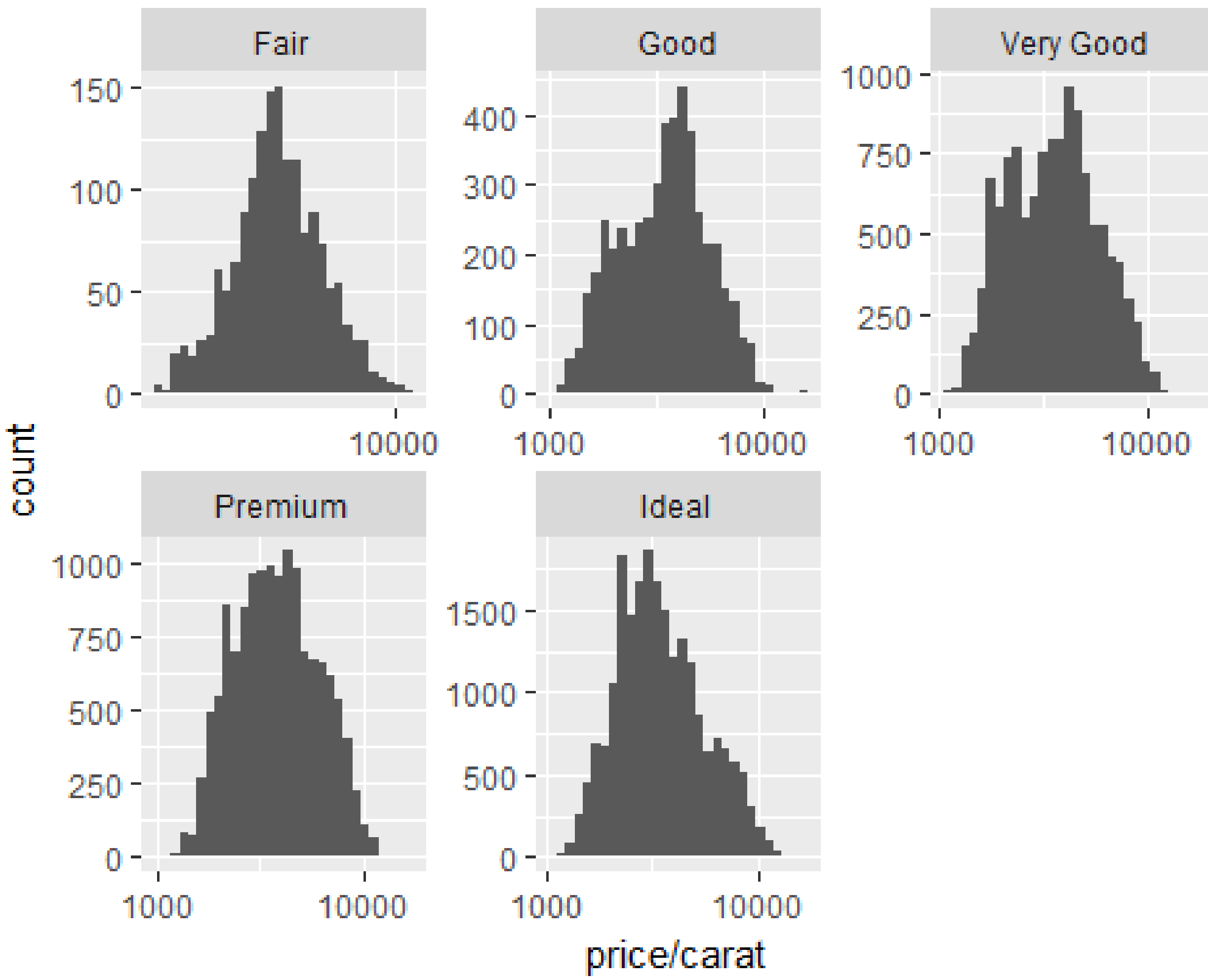
```
#Creating histogram of diamond prices by cut
ggplot(data = diamonds)+
  geom_histogram(aes(x = price)) +
  facet_wrap(~ cut)

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



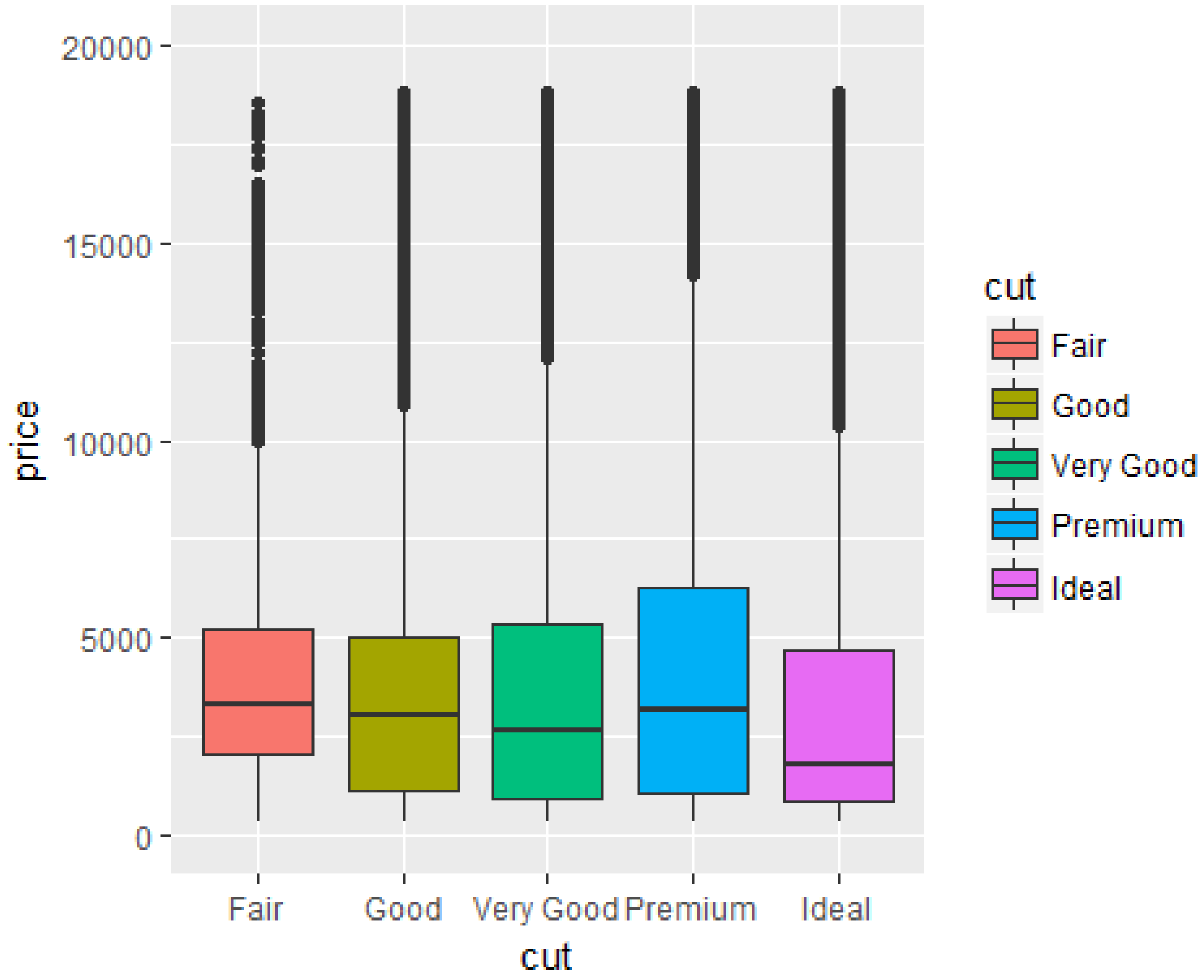
```
#Getting Median stats by cut
aggregate(. ~cut, data=diamonds, FUN=median)

##         cut carat color clarity depth table  price     x    y    z
## 1      Fair  1.00     4       3  65.0    58 3282.0 6.175 6.10 3.97
## 2      Good  0.82     3       3  63.4    58 3050.5 5.980 5.99 3.70
## 3 Very Good  0.71     3       4  62.1    58 2648.0 5.740 5.77 3.56
## 4   Premium  0.86     4       4  61.4    59 3185.0 6.110 6.06 3.72
## 5     Ideal  0.54     4       4  61.8    56 1810.0 5.250 5.26 3.23

#Getting Highest price diamond by cut
aggregate(. ~cut, data=diamonds, FUN=max)

##         cut carat color clarity depth table price     x     y     z
## 1      Fair  5.01     7       8  79.0    95 18574 10.74 10.54  6.98
## 2      Good  3.01     7       8  67.0    66 18788  9.44  9.38  5.79
## 3 Very Good  4.00     7       8  64.9    66 18818 10.01  9.94 31.80
## 4   Premium  4.01     7       8  63.0    62 18823 10.14 58.90  8.06
## 5     Ideal  3.50     7       8  66.7    63 18806  9.65 31.80  6.03

#The previous plot showed the distributions as different so freeing scales
ggplot(data = diamonds)+
  geom_histogram(aes(x = price)) +
  facet_wrap(~ cut, scales='free')
```



```r
#Looking at the price per carrat with log10
ggplot(data = diamonds)+
  geom_histogram(aes(x = price/carat)) +
  facet_wrap(~ cut, scales='free')+
  scale_x_log10()
```



```
#Creating a box plot for price based on cut
ggplot(diamonds, aes(x=cut, y=price, fill=cut))+
  geom_boxplot()+
  coord_cartesian(ylim = c(0,20000))
```

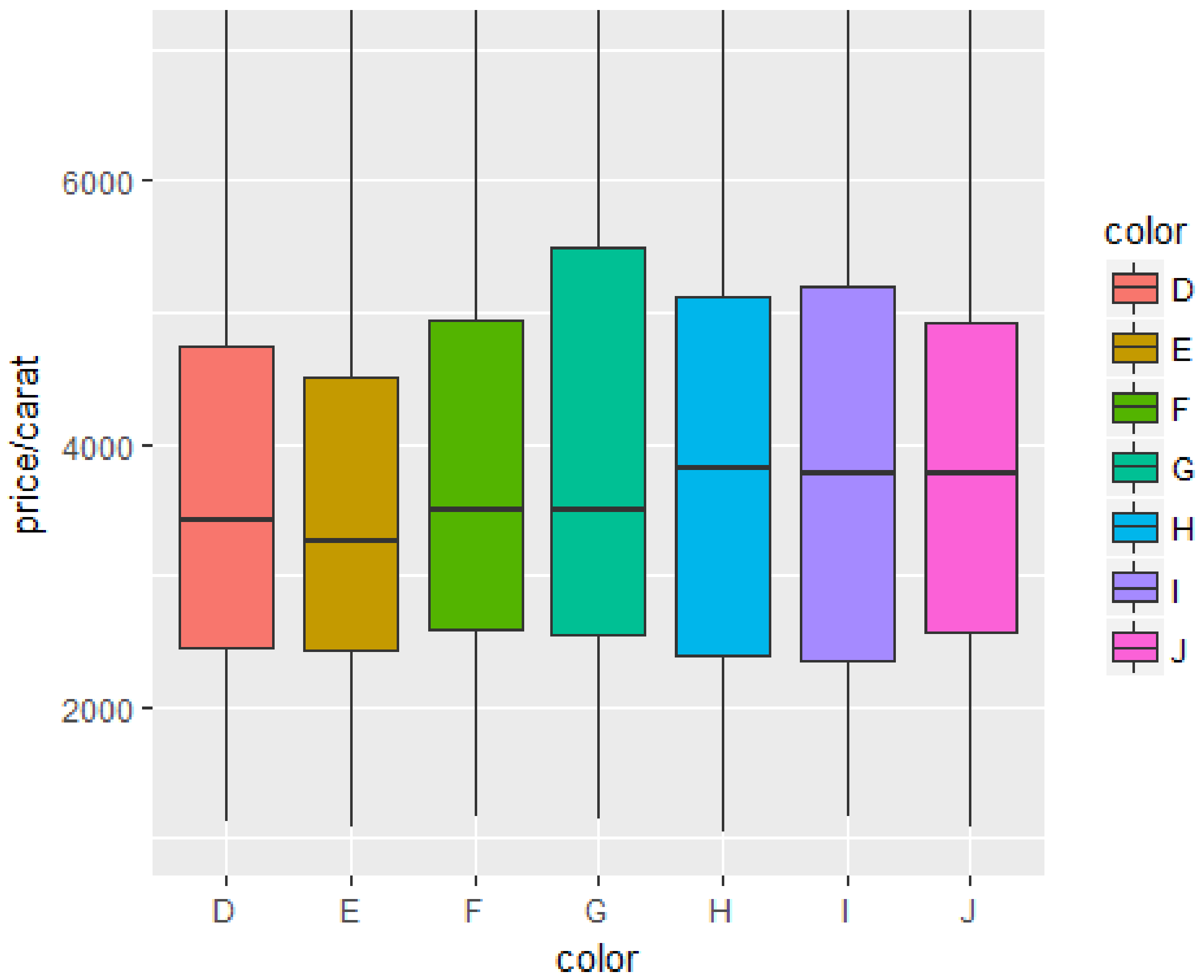


```r
#Summarizing price data by color
by(diamonds$price, diamonds$color, summary)

## diamonds$color: D
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     357     911    1838    3170    4214   18693
## --------------------------------------------------------
## diamonds$color: E
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     326     882    1739    3077    4003   18731
## --------------------------------------------------------
## diamonds$color: F
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     342     982    2344    3725    4868   18791
## --------------------------------------------------------
## diamonds$color: G
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     354     931    2242    3999    6048   18818
## --------------------------------------------------------
## diamonds$color: H
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     337     984    3460    4487    5980   18803
## --------------------------------------------------------
## diamonds$color: I
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     334    1120    3730    5092    7202   18823
## --------------------------------------------------------
## diamonds$color: J
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     335    1860    4234    5324    7695   18710

#Getting individual IQR's based on certain color
IQR(subset(diamonds, color=='D')$price)

## [1] 3302.5

IQR(subset(diamonds, color=='J')$price)

## [1] 5834.5
```

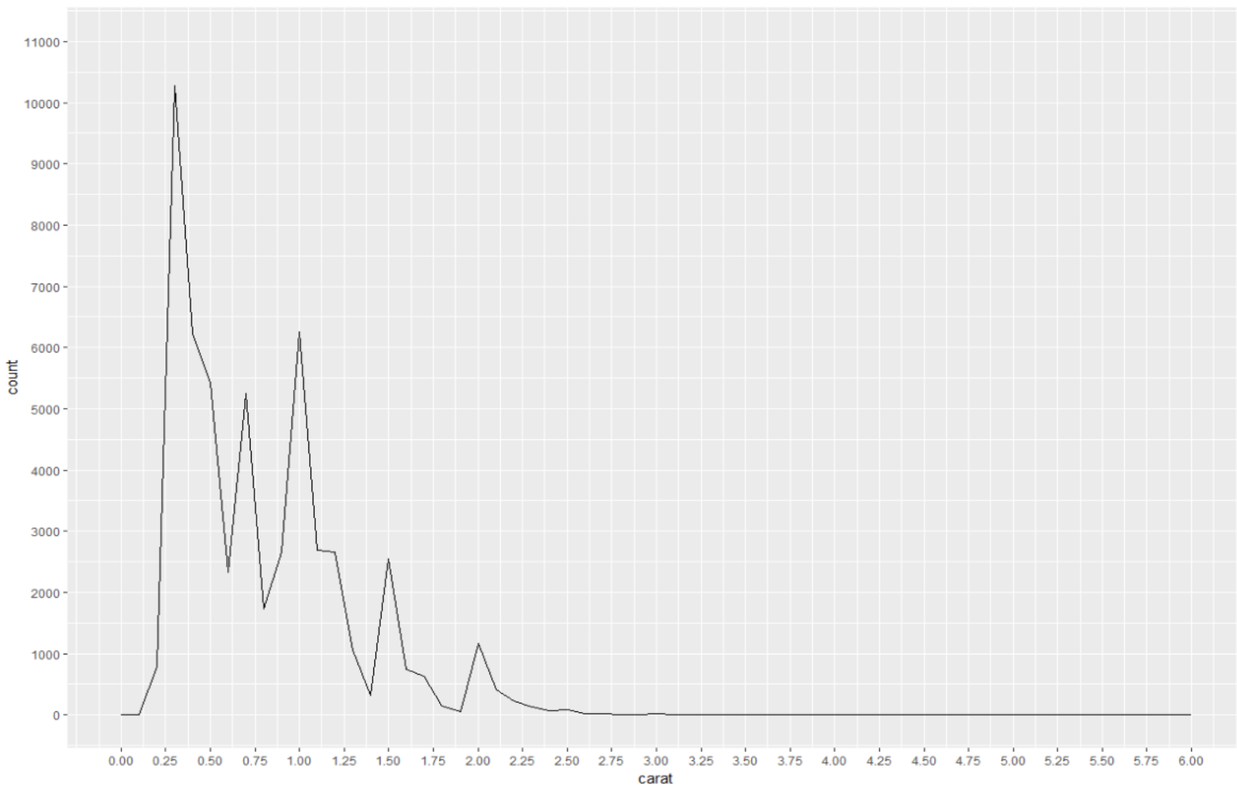
```r
#Getting price IQR by color
by(diamonds$price,diamonds$color,IQR)
```

```
## diamonds$color: D
## [1] 3302.5
## ------------------------------------------------------------
## diamonds$color: E
## [1] 3121
## ------------------------------------------------------------
## diamonds$color: F
## [1] 3886.25
## ------------------------------------------------------------
## diamonds$color: G
## [1] 5117
## ------------------------------------------------------------
## diamonds$color: H
## [1] 4996.25
## ------------------------------------------------------------
## diamonds$color: I
## [1] 6081.25
## ------------------------------------------------------------
## diamonds$color: J
## [1] 5834.5
```

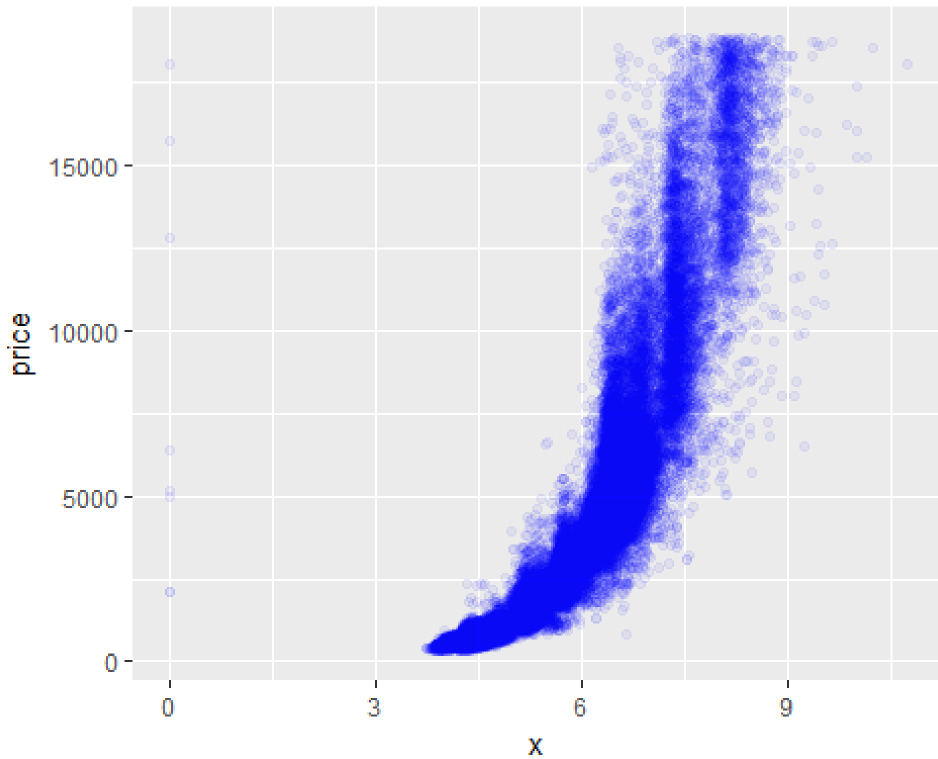
```r
#Looking at price per carat of diamonds across colors
ggplot(diamonds, aes(x=color, y=price/carat, fill=color))+
  geom_boxplot()+
  coord_cartesian(ylim = c(1000,7000))
```



```r
#Investigating carats using a frequency polygon

ggplot(diamonds, aes(x=carat))+
  geom_freqpoly(binwidth=0.1)+
  scale_x_continuous(limits=c(0,6), breaks=seq(0,6,0.25))+
  scale_y_continuous(limits=c(0,11000), breaks=seq(0,11000,1000))
```

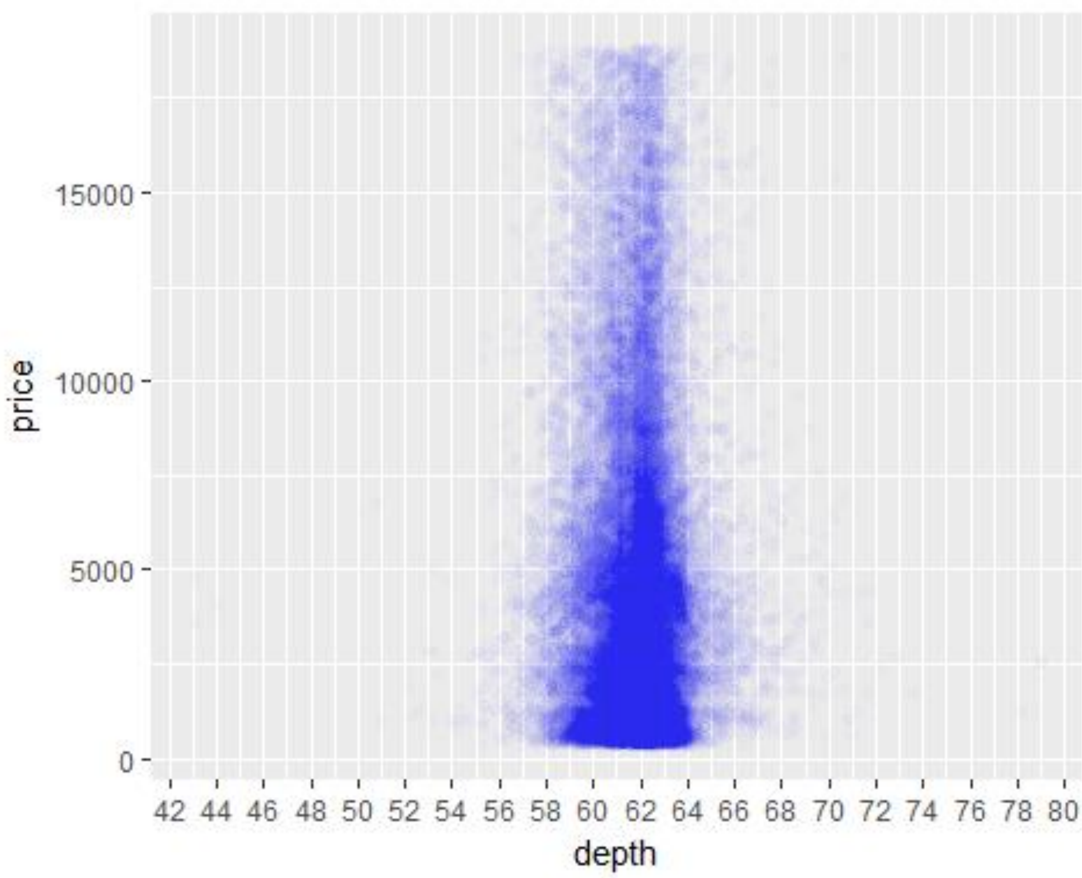


```r
ggplot(diamonds, aes(x=x, y=price))+geom_point(color="blue", alpha=1/20)
```

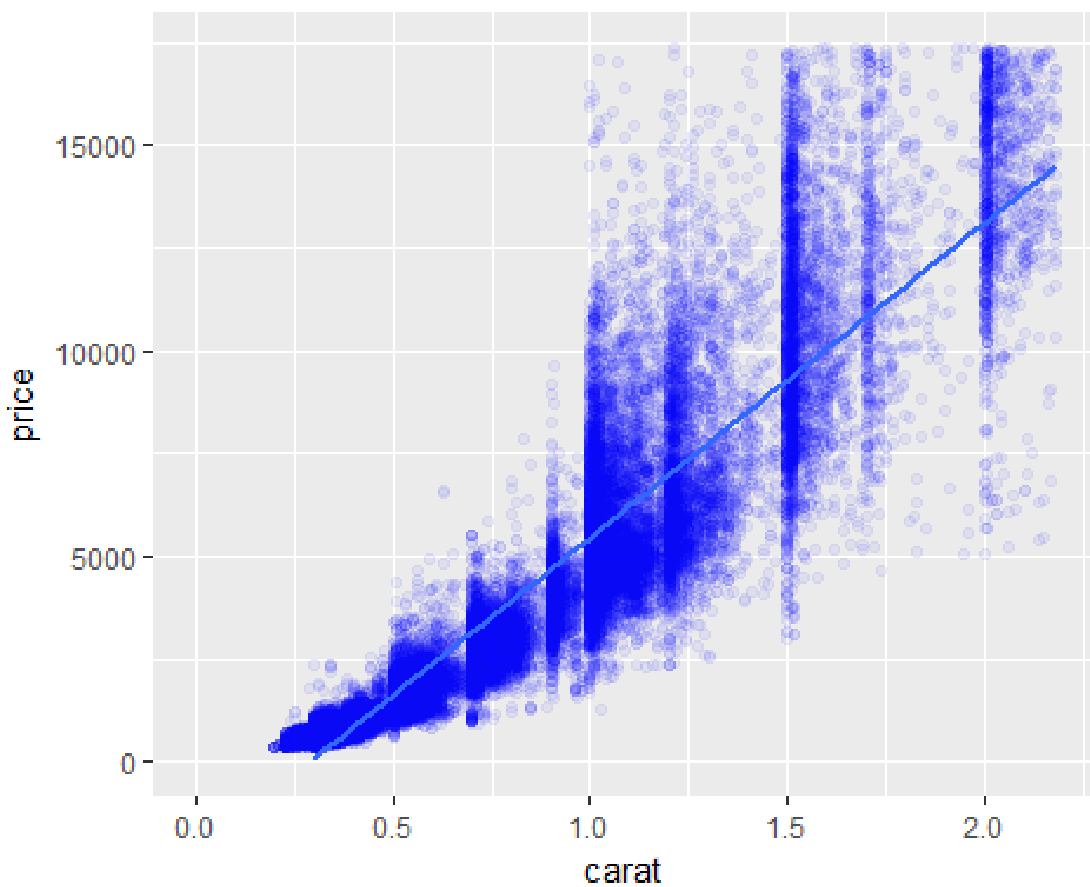



```r
#Calculating correlations
with(diamonds, cor.test(x, price))
##  Pearson's product-moment correlation
##
## data:  x and price
## t = 440.16, df = 53938, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.8825835 0.8862594
## sample estimates:
##       cor
## 0.8844352

with(diamonds, cor.test(y, price))
##  Pearson's product-moment correlation
##
## data:  y and price
## t = 401.14, df = 53938, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.8632867 0.8675241
## sample estimates:
##       cor
## 0.8654209
```

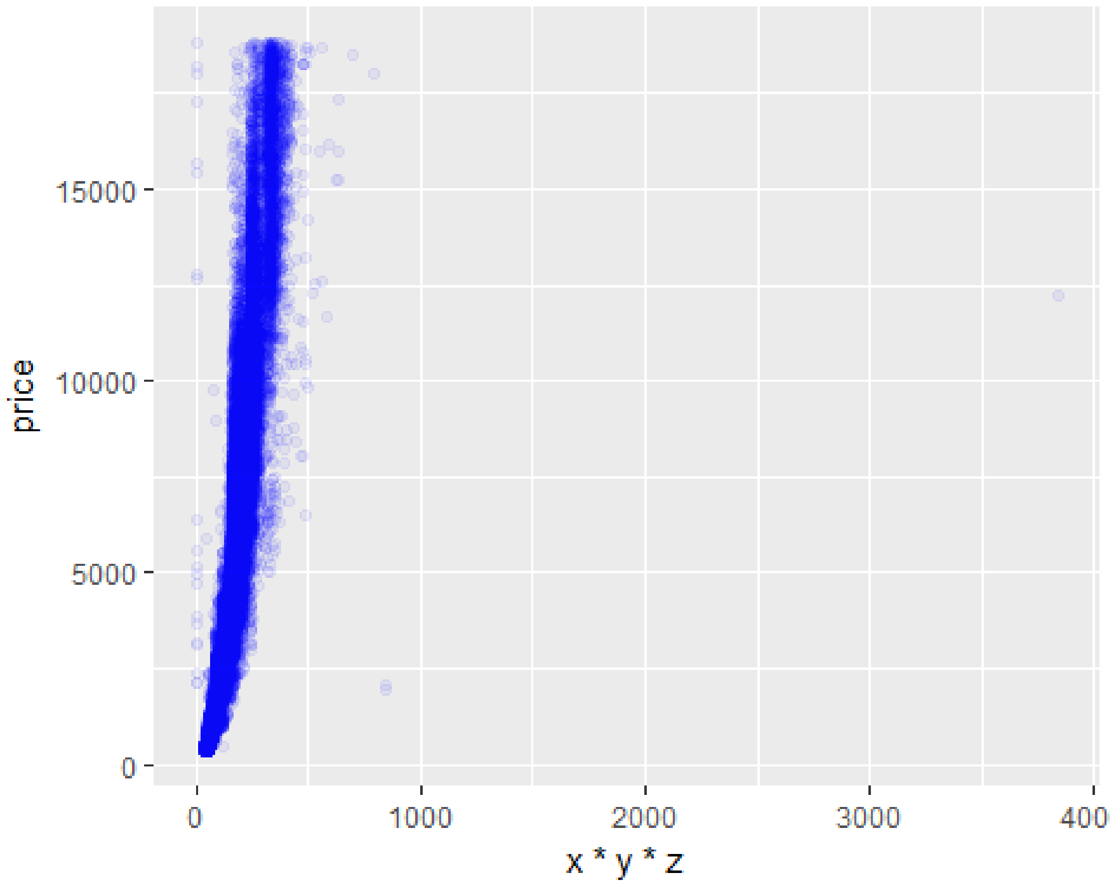
```r
with(diamonds, cor.test(z, price))

##
##  Pearson's product-moment correlation
##
## data:  z and price
## t = 393.6, df = 53938, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.8590541 0.8634131
## sample estimates:
##       cor
## 0.8612494

ggplot(diamonds, aes(x=depth, y=price)) +geom_point(color='blue', alpha=1/100)+
  scale_x_continuous(breaks = seq(0,80,2))
```



```r
with(diamonds, cor.test(x=depth, y=price))

##
##  Pearson's product-moment correlation
##
## data:  depth and price
## t = -2.473, df = 53938, p-value = 0.0134
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.019084756 -0.002208537
## sample estimates:
##        cor
## -0.0106474

ggplot(diamonds, aes(x=carat, y=price))+geom_point(color='blue', alpha=1/20)+
  scale_x_continuous(limits = c(0, quantile(diamonds$carat, 0.99)))+
  scale_y_continuous(limits = c(0, quantile(diamonds$price, 0.99)))+
  stat_smooth(method = 'lm')
```

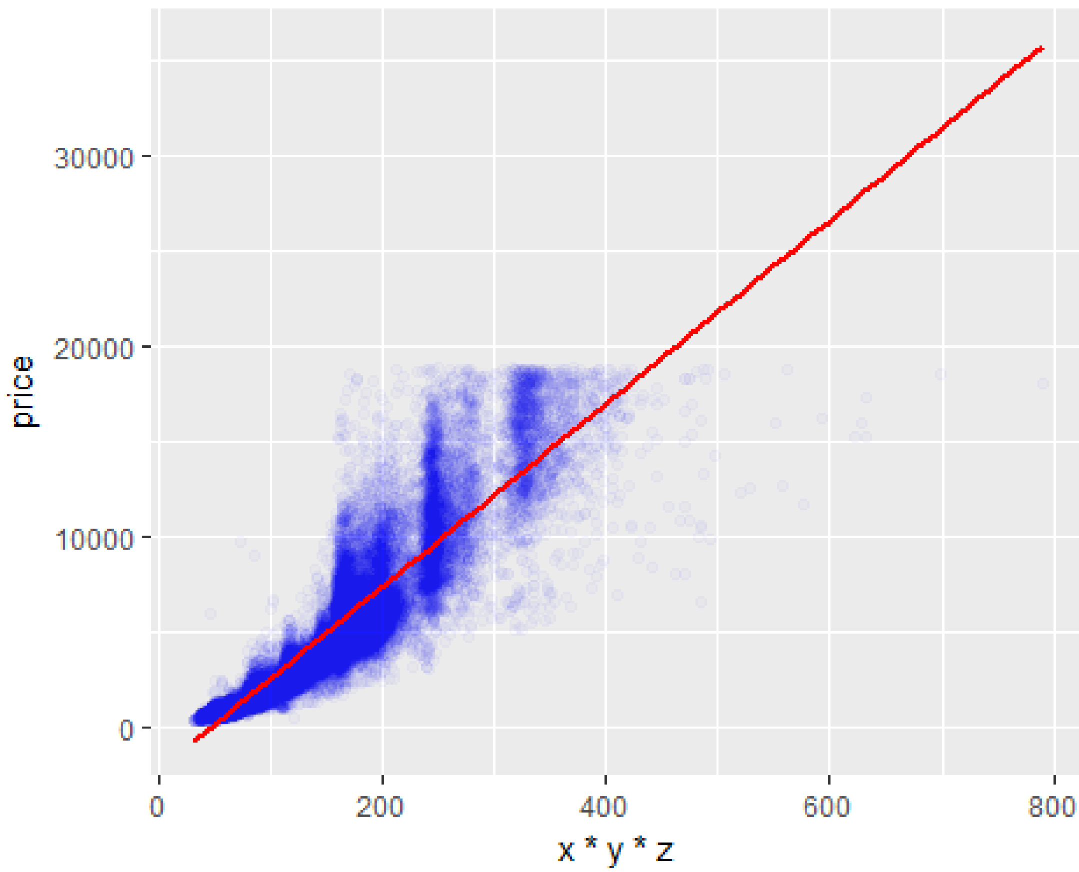


```r
ggplot(diamonds, aes(x= x*y*z, y=price)) +geom_point(alpha = 1/20, 
color='blue')
```



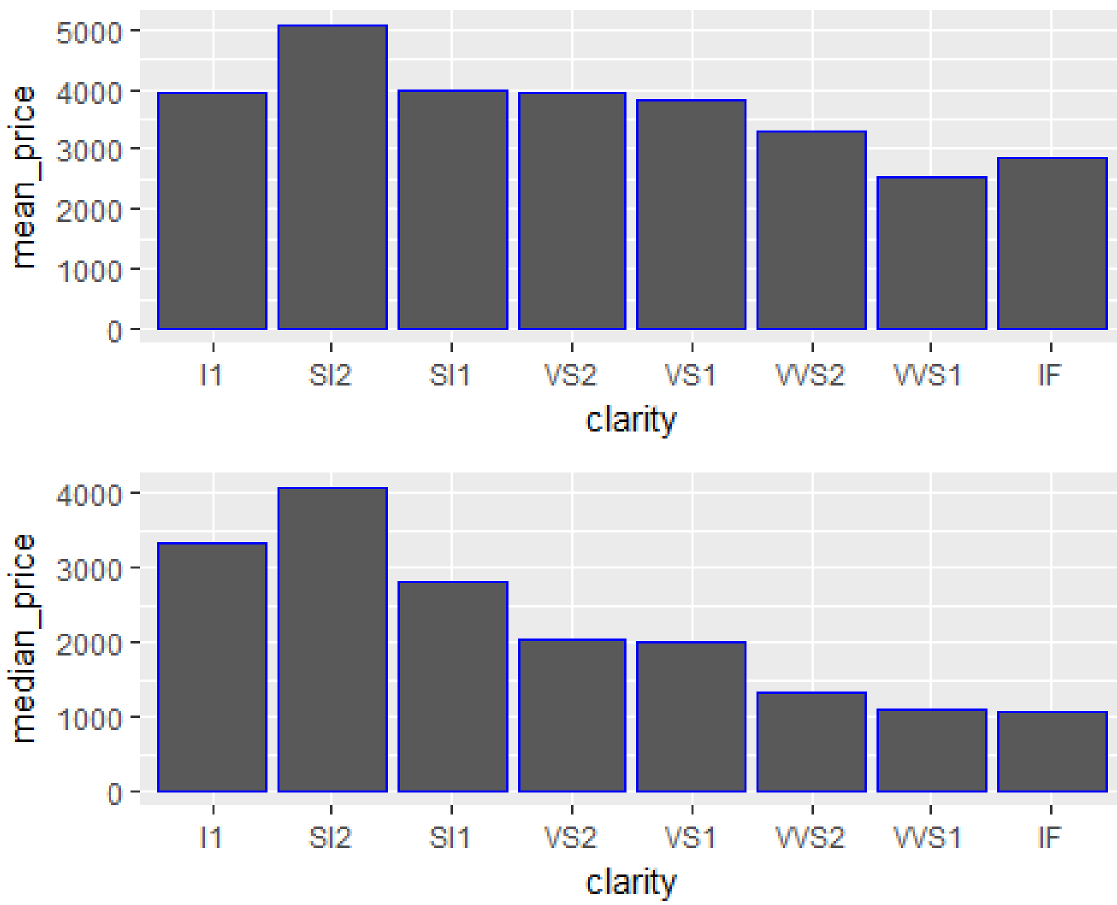

```r
#Calculating volume with variables x, y,z
#Getting correlation but not including volume of 0 or 800+
diamonds$volume <- diamonds$x * diamonds$y * diamonds$z
with(subset(diamonds,diamonds$volume>0 & diamonds$volume<800), 
cor.test(volume, price))

## 
##  Pearson's product-moment correlation
## 
## data:  volume and price
## t = 559.19, df = 53915, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9222944 0.9247772
## sample estimates:
##       cor 
## 0.9235455
```

```r
ggplot(subset(diamonds,diamonds$volume>0 & diamonds$volume<800), aes(x=
x*y*z, y=price)) +
  geom_point(alpha = 1/50, color='blue')+
  geom_smooth(method = 'lm', color='red')
```



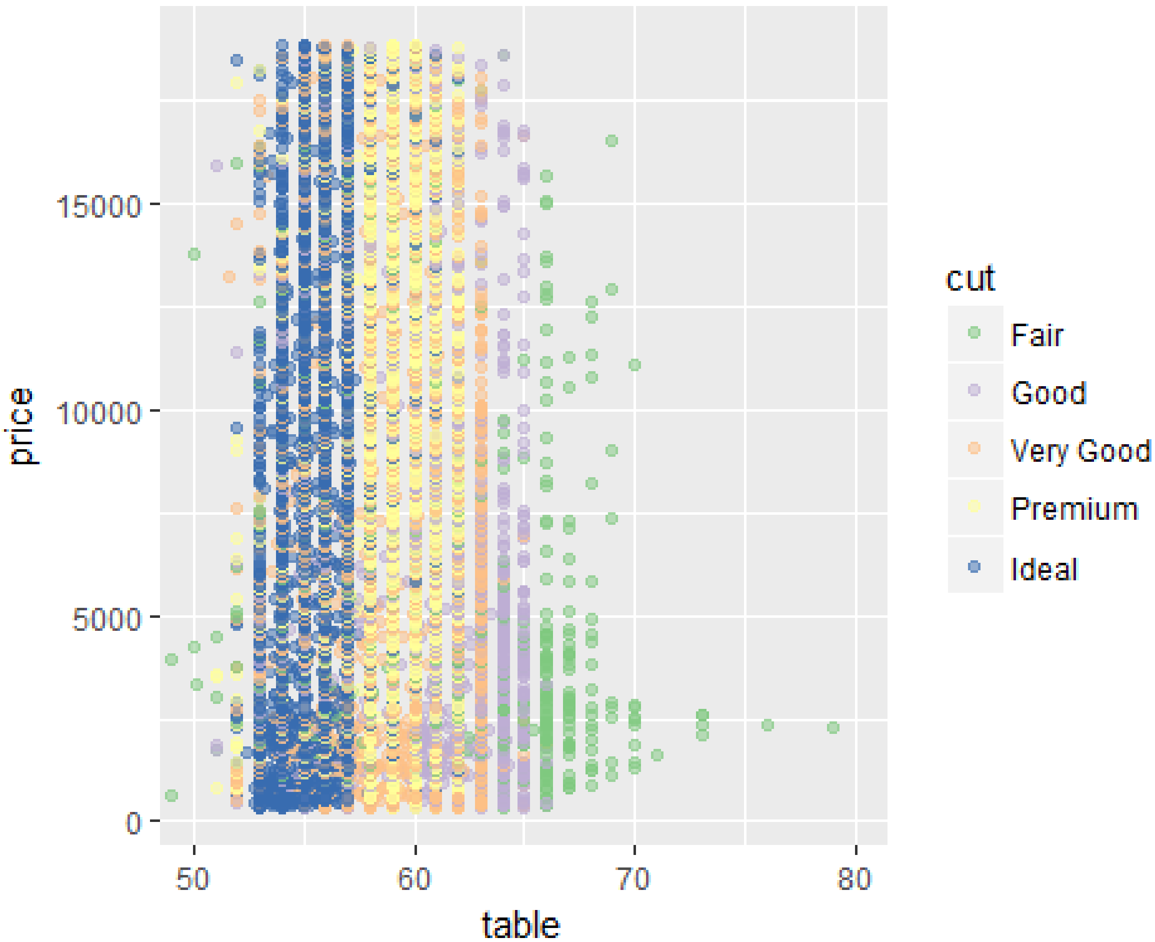

```r
diamondsByClarity <- diamonds %>%
  group_by(clarity) %>%
  summarise(mean_price = mean(price),
            median_price = median(price),
            min_price = min(price),
            max_price = max(price),
            n = n()) %>%
arrange(clarity)
```

```r
library(gridExtra)

b1 <- ggplot(diamondsByClarity, aes(x=clarity, y=mean_price))+
geom_bar(stat='identity', color='blue')
b2 <- ggplot(diamondsByClarity, aes(x=clarity, y=median_price))+
geom_bar(stat='identity', color='blue')
grid.arrange(b1,b2)
```

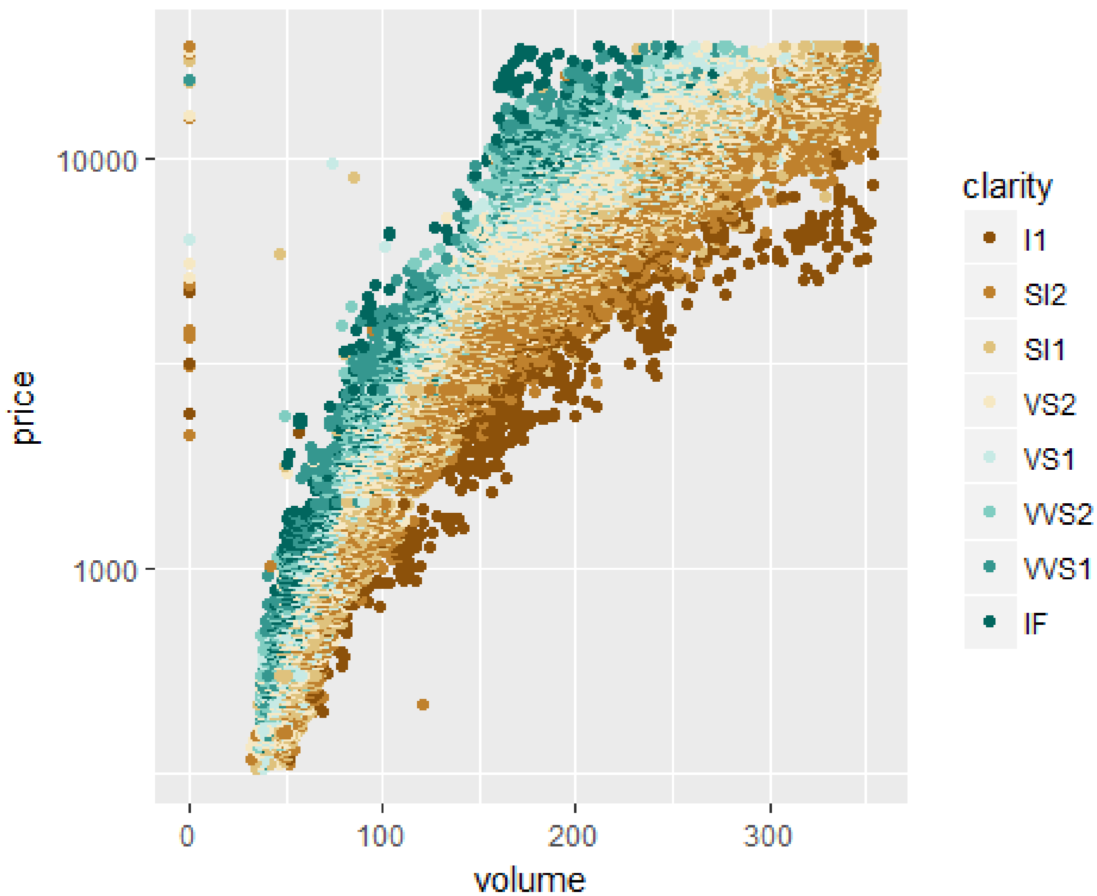


```
#Plotting the histograms in each color with a facet wrap showing the
different cuts
ggplot(diamonds, aes(x=price, fill=cut))+ geom_histogram()+
facet_wrap(~color)+
  scale_fill_brewer(type = 'qual')+
  scale_x_log10()
```

```
#Creating a scatter plot of the table value vs price
#Table reflects the width of hte top diamond
ggplot(diamonds, aes(x=table, y=price)) +
  geom_point(aes(color=cut), alpha=1/2)+
  scale_color_brewer(type='qual')+
  coord_cartesian(xlim=c(50,80))
```

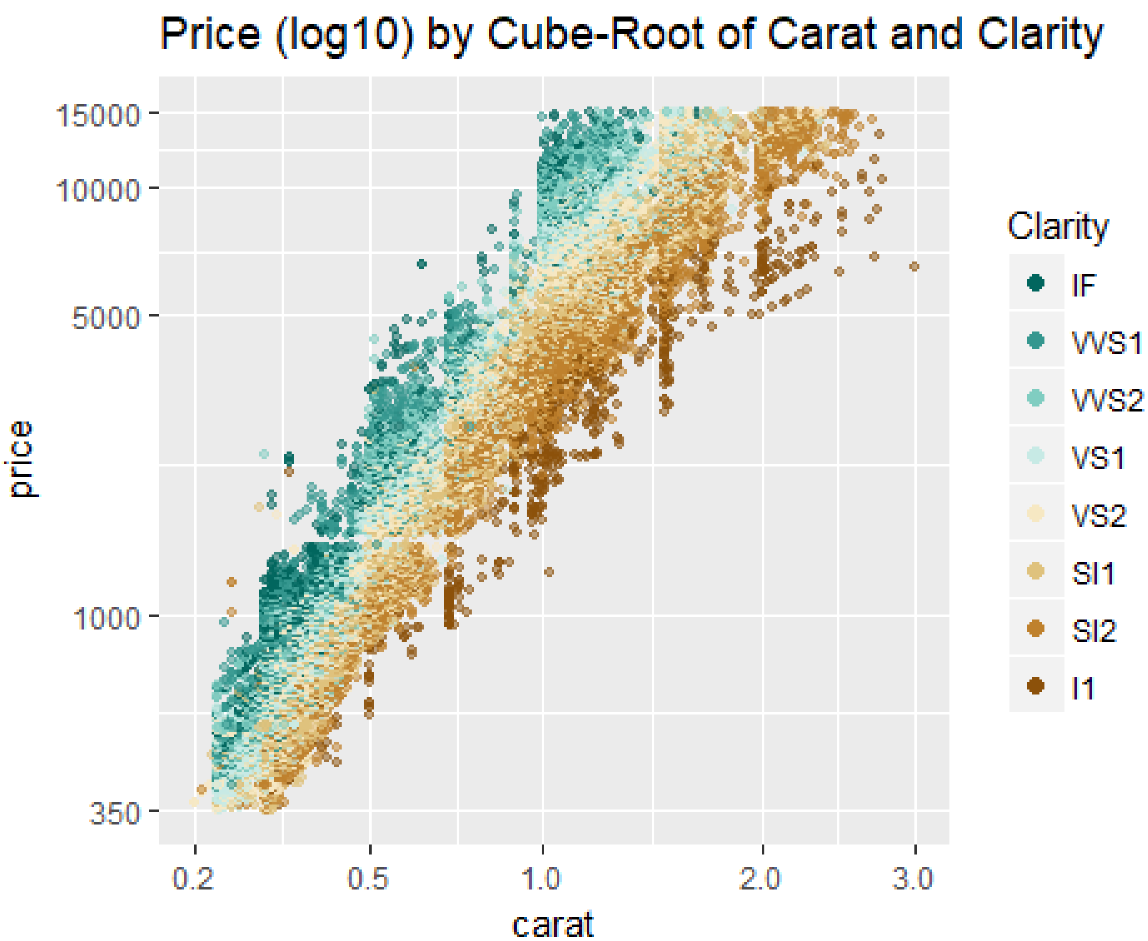


```r
#Adding a volume variable with transform
diamonds <- transform(diamonds, volume = x*y*z)

#Plotting price (in a log10 scale) in terms of volume colored by clarity
ggplot(subset(diamonds, volume < quantile(volume, 0.99)), aes(x=volume,
y=price))+
  geom_point(aes(color = clarity))+
  scale_color_brewer(type = 'div')+
  scale_y_log10()
```

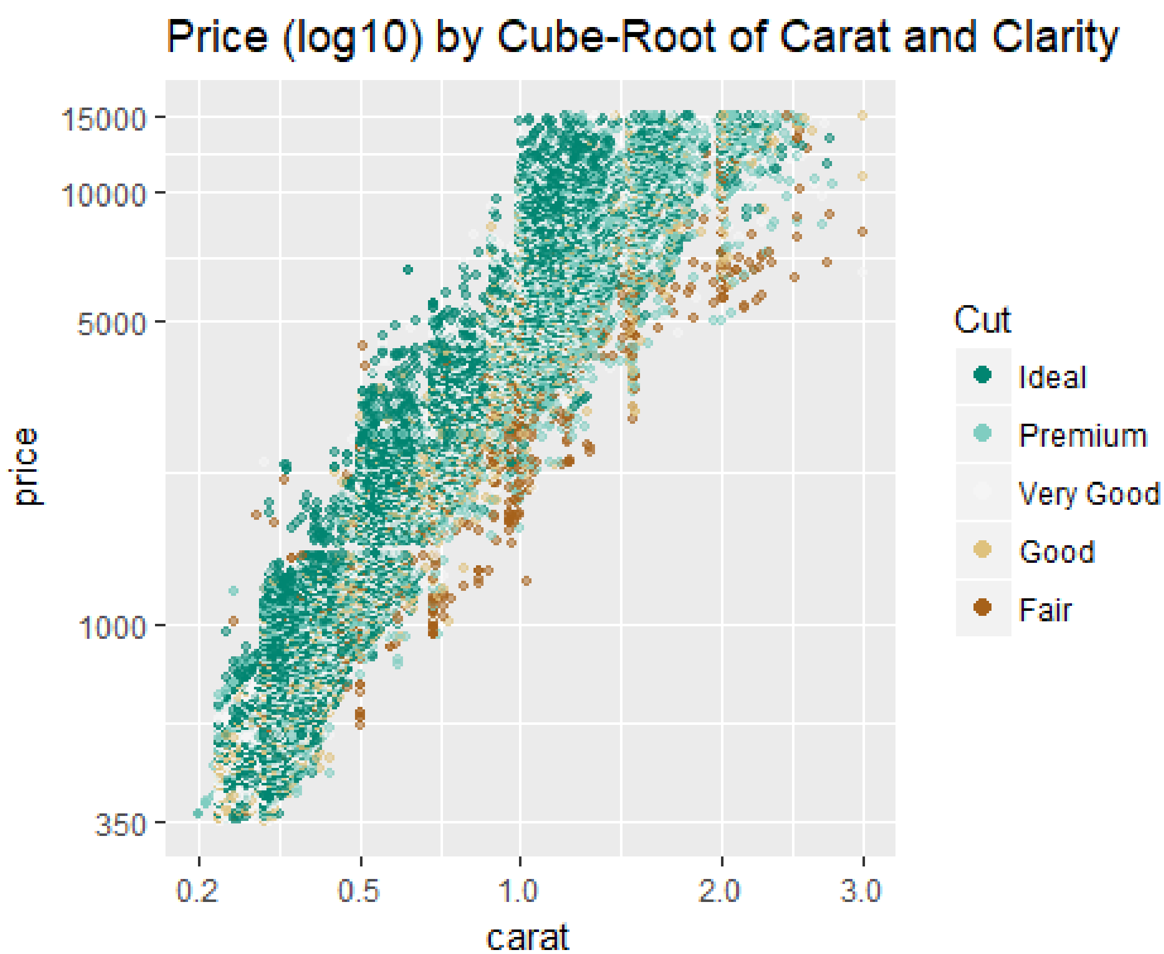


```
ggplot(diamonds, aes(x=cut, y=price)) +geom_jitter(aes(color=color))+
  scale_color_brewer(type = 'div')+
  facet_wrap(~ clarity)
```

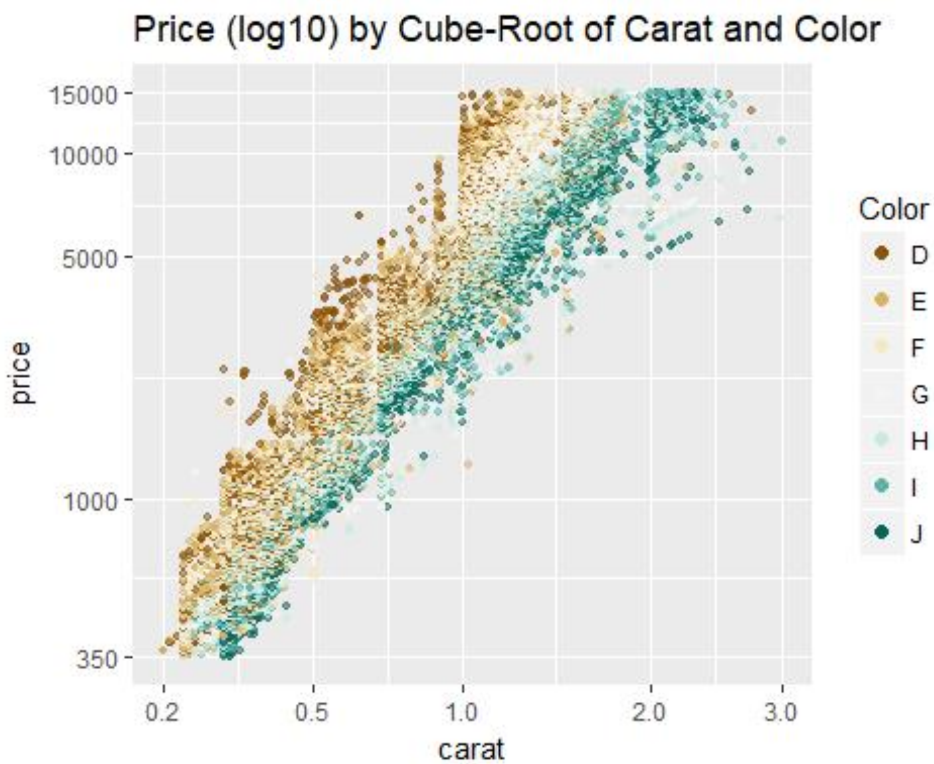
```
cuberoot_trans = function() trans_new('cuberoot', transform = function(x)
x^(1/3),
                                      inverse = function(x) x^3)

#Looking at the carat with price on a log10 scale
library(scales)
ggplot(aes(x = carat, y = price, color=clarity), data = diamonds) +
  geom_point(alpha = 0.5, size = 1, position = 'jitter') +
  scale_color_brewer(type = 'div',
    guide = guide_legend(title = 'Clarity', reverse = T,
    override.aes = list(alpha = 1, size = 2))) +
  scale_x_continuous(trans = cuberoot_trans(), limits = c(0.2, 3),
    breaks = c(0.2, 0.5, 1, 2, 3)) +
  scale_y_continuous(trans = log10_trans(), limits = c(350, 15000),
    breaks = c(350, 1000, 5000, 10000, 15000)) +
  ggtitle('Price (log10) by Cube-Root of Carat and Clarity')
```



```
#Also looking at it colored by cut
ggplot(aes(x = carat, y = price, color = cut), data = diamonds) +
  geom_point(alpha = 0.5, size = 1, position = 'jitter') +
  scale_color_brewer(type = 'div',
                     guide = guide_legend(title = 'Cut', reverse = T,
                                          override.aes = list(alpha = 1, size
= 2))) +
  scale_x_continuous(trans = cuberoot_trans(), limits = c(0.2, 3),
                     breaks = c(0.2, 0.5, 1, 2, 3)) +
  scale_y_continuous(trans = log10_trans(), limits = c(350, 15000),
                     breaks = c(350, 1000, 5000, 10000, 15000)) +
  ggtitle('Price (log10) by Cube-Root of Carat and Clarity')

## Warning: Removed 1693 rows containing missing values (geom_point).
```



```r
ggplot(aes(x = carat, y = price, color = color), data = diamonds) +
  geom_point(alpha = 0.5, size = 1, position = 'jitter') +
  scale_color_brewer(type = 'div',
                     guide = guide_legend(title = 'Color', reverse = FALSE,
                                          override.aes = list(alpha = 1, size
= 2))) +
  scale_x_continuous(trans = cuberoot_trans(), limits = c(0.2, 3),
                     breaks = c(0.2, 0.5, 1, 2, 3)) +
  scale_y_continuous(trans = log10_trans(), limits = c(350, 15000),
                     breaks = c(350, 1000, 5000, 10000, 15000)) +
  ggtitle('Price (log10) by Cube-Root of Carat and Color')
```




```r
#'I' makes it as is before using it in the regression, rather than as part of the forumula
suppressWarnings(library(memisc))

m1 <- lm(I(log(price)) ~ I(carat^(1/3)), data= diamonds)
m2 <- update(m1, ~ . + carat)
m3 <- update(m2, ~ . + cut)
m4 <- update(m3, ~ . + color)
m5 <- update(m4, ~ . + clarity)
mtable(m1, m2, m3, m4, m5)
## Calls:
## m1: lm(formula = I(log(price)) ~ I(carat^(1/3)), data = diamonds)
## m2: lm(formula = I(log(price)) ~ I(carat^(1/3)) + carat, data = diamonds)
## m3: lm(formula = I(log(price)) ~ I(carat^(1/3)) + carat + cut, data = diamonds)
## m4: lm(formula = I(log(price)) ~ I(carat^(1/3)) + carat + cut + color,
##     data = diamonds)
## m5: lm(formula = I(log(price)) ~ I(carat^(1/3)) + carat + cut + color +
##     clarity, data = diamonds)
##
```

```
##
=============================================================================================
##                          m1             m2             m3             m4             m5
## ---------------------------------------------------------------------------------------------
-
##   (Intercept)          2.821***       1.039***       0.874***      0.932***       0.415***
##                       (0.006)        (0.019)        (0.019)       (0.017)        (0.010)
##   I(carat^(1/3))       5.558***       8.568***       8.703***      8.438***       9.144***
##                       (0.007)        (0.032)        (0.031)       (0.028)        (0.016)
##   carat                              -1.137***      -1.163***     -0.992***      -1.093***
##                                      (0.012)        (0.011)       (0.010)        (0.006)
##   cut: .L                                            0.224***      0.224***       0.120***
##                                                     (0.004)       (0.004)        (0.002)
##   cut: .Q                                           -0.062***     -0.062***      -0.031***
##                                                     (0.004)       (0.003)        (0.002)
##   cut: .C                                            0.051***      0.052***       0.014***
##                                                     (0.003)       (0.003)        (0.002)
##   cut: ^4                                            0.018***      0.018***      -0.002
##                                                     (0.003)       (0.002)        (0.001)
##   color: .L                                                       -0.373***      -0.441***
##                                                                   (0.003)        (0.002)
##   color: .Q                                                       -0.129***      -0.093***
##                                                                   (0.003)        (0.002)
##   color: .C                                                        0.001         -0.013***
##                                                                   (0.003)        (0.002)
##   color: ^4                                                        0.029***       0.012***
##                                                                   (0.003)        (0.002)
##   color: ^5                                                       -0.016***      -0.003*
##                                                                   (0.003)        (0.001)
##   color: ^6                                                       -0.023***       0.001
##                                                                   (0.002)        (0.001)
##   clarity: .L                                                                     0.907***
##                                                                                  (0.003)
##   clarity: .Q                                                                    -0.240***
##                                                                                  (0.003)
##   clarity: .C                                                                     0.131***
##                                                                                  (0.003)
##   clarity: ^4                                                                    -0.063***
##                                                                                  (0.002)
##   clarity: ^5                                                                     0.026***
##                                                                                  (0.002)
##   clarity: ^6                                                                    -0.002
##                                                                                  (0.002)
##   clarity: ^7                                                                     0.032***
##                                                                                  (0.001)
## ---------------------------------------------------------------------------------------------
-
##   R-squared            0.924          0.935          0.939         0.951          0.984
##   adj. R-squared       0.924          0.935          0.939         0.951          0.984
##   sigma                0.280          0.259          0.250         0.224          0.129
##   F               652012.063     387489.366     138654.523     87959.467     173791.084
##   p                    0.000          0.000          0.000         0.000          0.000
##   Log-likelihood   -7962.499      -3631.319      -1837.416      4235.240      34091.272
##   Deviance          4242.831       3613.360       3380.837      2699.212        892.214
##   AIC              15930.999       7270.637       3690.832     -8442.481     -68140.544
##   BIC              15957.685       7306.220       3761.997     -8317.942     -67953.736
##   N                53940          53940          53940         53940          53940
##
=============================================================================================
```