Analysis of the Diamonds Dataset

Jon Kinsey

Wed Dec 3 18:35:55 2014

```r
# The diamonds dataset contains the prices and other attributes
# of almost 54,000 diamonds.
#
# The variables are as follows:
# * price. price in US dollars (\$326-\$18,823)
# * carat. weight of the diamond (0.2-5.01)
# * cut. quality of the cut (Fair, Good, Very Good, Premium, Ideal)
# * colour. diamond colour, from J (worst) to D (best)
# * clarity. a measurement of how clear the diamond is (I1 (worst), SI1, SI2,
#   VS1, VS2, VVS1, VVS2, 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)
#
library(ggplot2)
library(gridExtra)
```

```
## Loading required package: grid
```

```r
library(scales)
library(memisc)
```

```
## Loading required package: lattice
## Loading required package: MASS
##
## Attaching package: 'memisc'
##
## The following object is masked from 'package:scales':
##
##     percent
##
## The following objects are masked from 'package:stats':
##
##     contr.sum, contr.treatment, contrasts
##
## The following object is masked from 'package:base':
##
##     as.array
```

```r
data(diamonds)

#View(diamonds)
table(diamonds$cut)
```

```
##
##      Fair      Good Very Good   Premium     Ideal
##      1610      4906     12082     13791     21551
```

```r
summary(diamonds$price)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     326     950    2400    3930    5320   18800
```

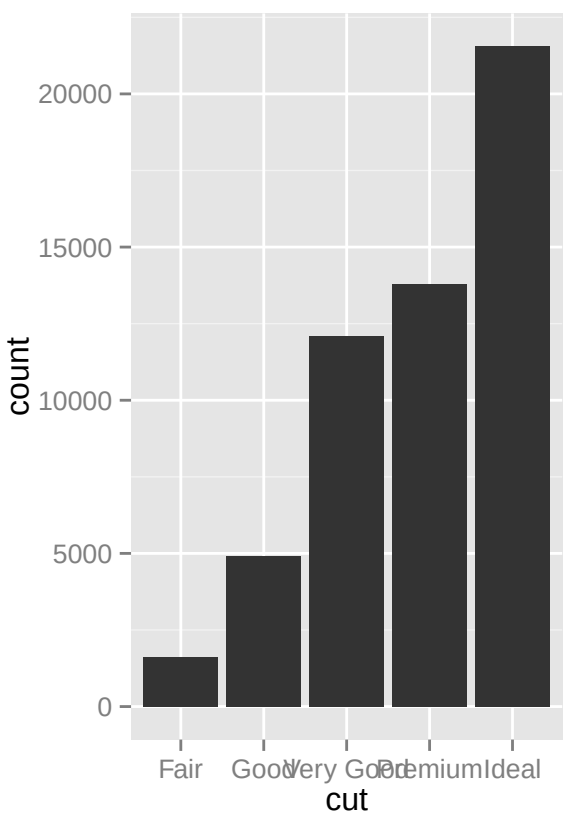
```r
#
# Before looking at any plots lets take a look at a sample
set.seed(42)
smallset<-diamonds[sample(nrow(diamonds),1000),]
head(smallset)
```

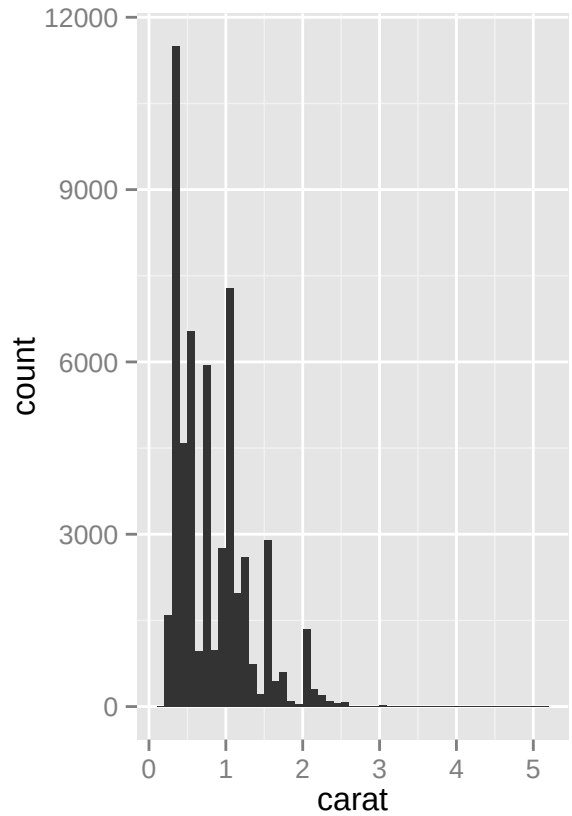
```
##       carat       cut color clarity depth table price    x    y    z
## 49345  0.71 Very Good     H     SI1  62.5    60  2096 5.68 5.75 3.57
## 50545  0.79   Premium     H     SI1  61.8    59  2275 5.97 5.91 3.67
## 15434  1.03     Ideal     F     SI1  62.4    57  6178 6.48 6.44 4.03
## 44792  0.50     Ideal     E     VS2  62.2    54  1624 5.08 5.11 3.17
## 34614  0.27     Ideal     E     VS1  61.6    56   470 4.14 4.17 2.56
## 27998  0.30   Premium     E     VS2  61.7    58   658 4.32 4.34 2.67
```

```r
summary(smallset)
```

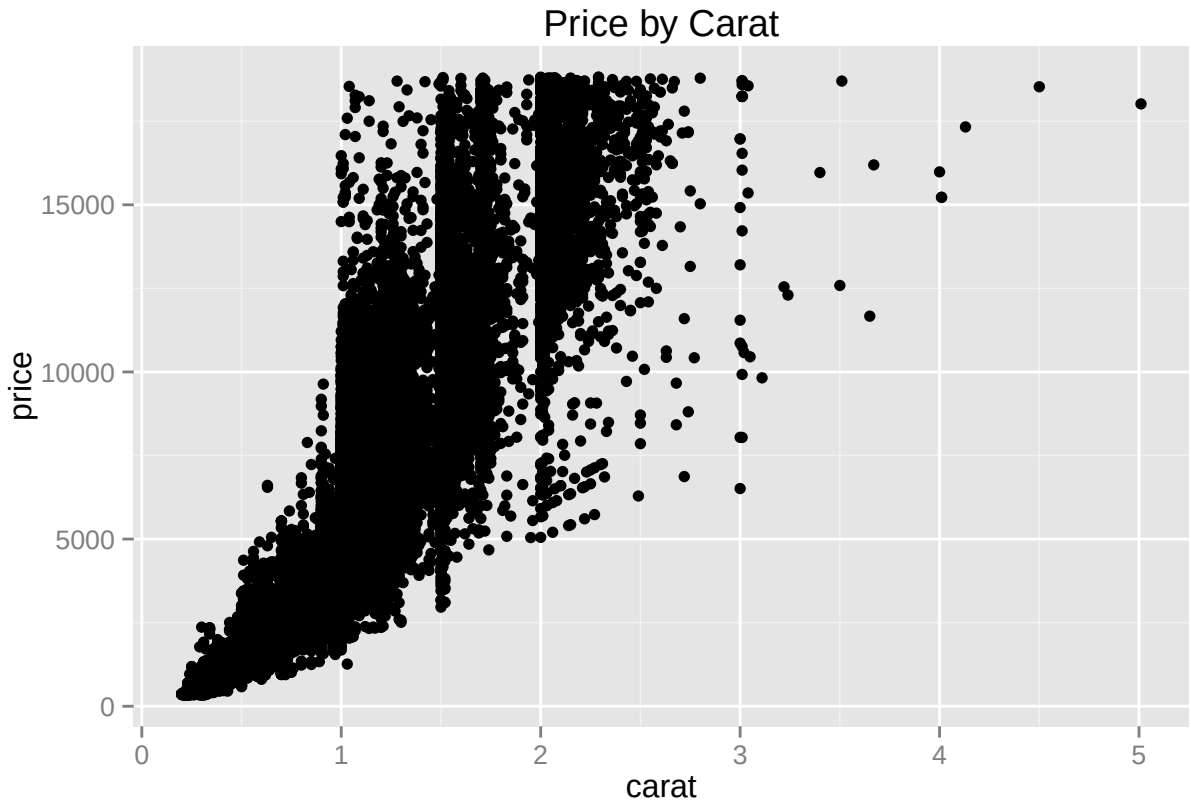
```
##      carat              cut      color      clarity        depth
##  Min.   :0.220   Fair     : 28   D:121   SI1    :258   Min.   :55.2
##  1st Qu.:0.400   Good     : 88   E:186   VS2    :231   1st Qu.:61.0
##  Median :0.710   Very Good:227   F:164   SI2    :175   Median :61.8
##  Mean   :0.819   Premium  :257   G:216   VS1    :141   Mean   :61.7
##  3rd Qu.:1.070   Ideal    :400   H:154   VVS2   : 91   3rd Qu.:62.5
##  Max.   :2.660                    I:106   VVS1   : 67   Max.   :72.2
##                                   J: 53   (Other): 37
##      table          price             x               y
##  Min.   :50.1   Min.   :  342   Min.   :3.85   Min.   :3.84
##  1st Qu.:56.0   1st Qu.:  990   1st Qu.:4.74   1st Qu.:4.76
##  Median :57.0   Median : 2595   Median :5.75   Median :5.78
##  Mean   :57.4   Mean   : 4111   Mean   :5.79   Mean   :5.79
##  3rd Qu.:59.0   3rd Qu.: 5495   3rd Qu.:6.60   3rd Qu.:6.61
##  Max.   :65.0   Max.   :18795   Max.   :8.83   Max.   :8.87
##
##        z
##  Min.   :2.33
##  1st Qu.:2.92
##  Median :3.55
##  Mean   :3.57
##  3rd Qu.:4.07
##  Max.   :5.58
##
```

```r
#
# In the first plots we lets look at the no. of diamonds vs cut and carat
# using qplot and adjusting the bin width
p1 <- qplot(x=cut, data = diamonds)
p2 <- qplot(x=carat, data = diamonds, binwidth = .1)
# Plot them together using grid.arrange
grid.arrange(p1, p2, ncol = 2)
```

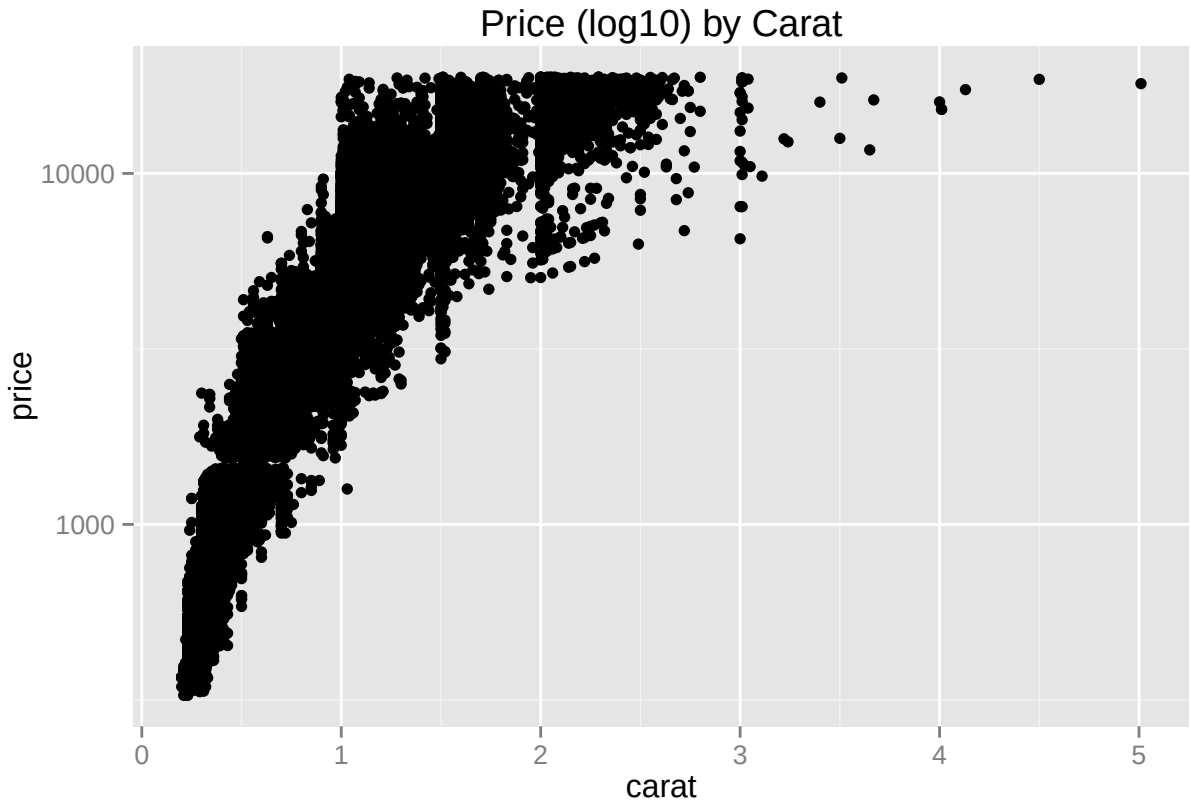




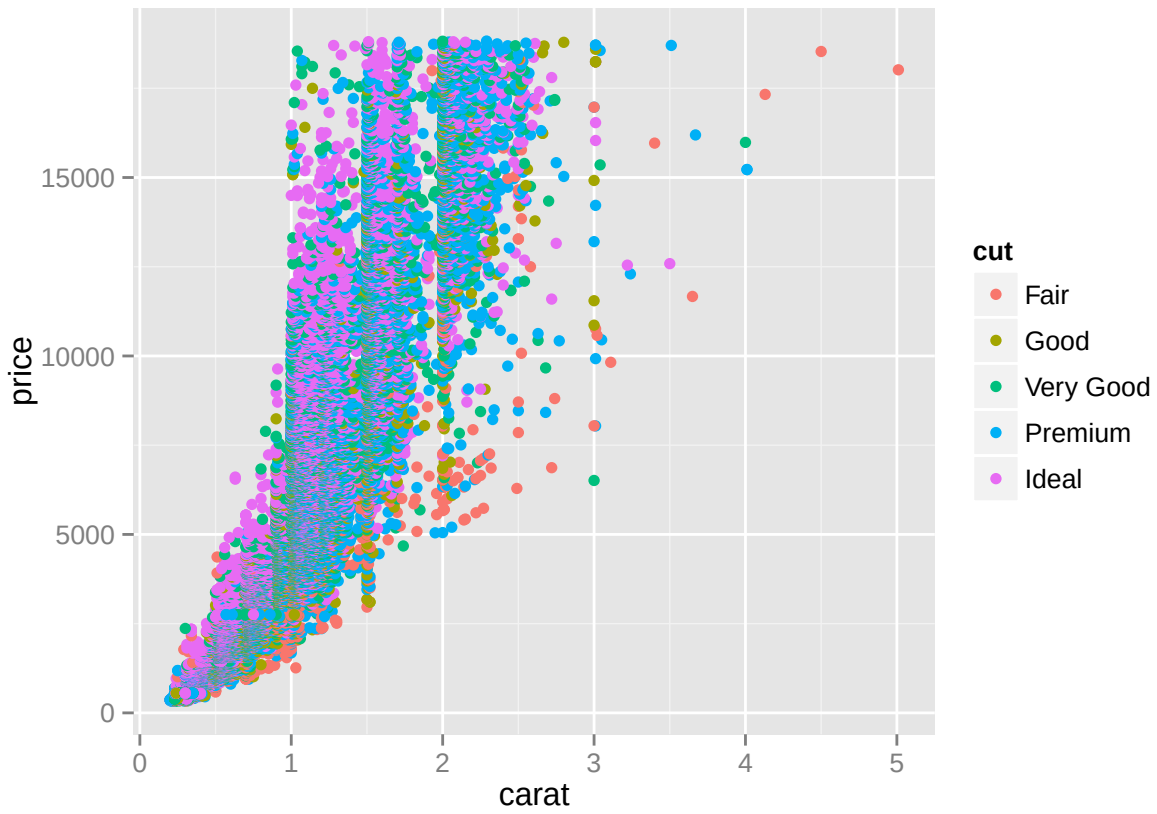

```
# Here, we see the counts are highest for the Ideal cut
# and the distribution is skewed toward a carat less than 1
#
# Now lets take a look at the price per carat first on a linear scale
# then on a log scale for the y-axis
qplot(carat, price, data=diamonds) +
  ggtitle('Price by Carat')
```



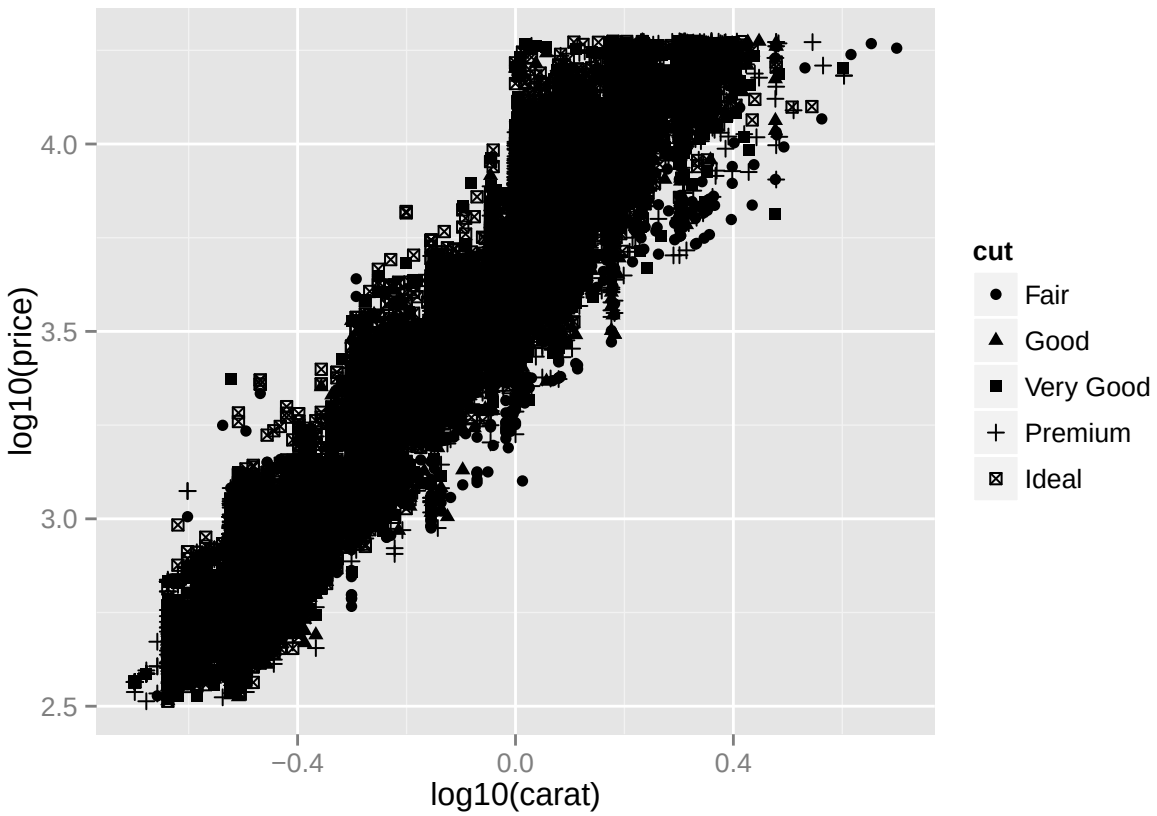

```
qplot(carat, price, data=diamonds) +
  scale_y_continuous(trans = log10_trans() ) +
  ggtitle('Price (log10) by Carat')
```



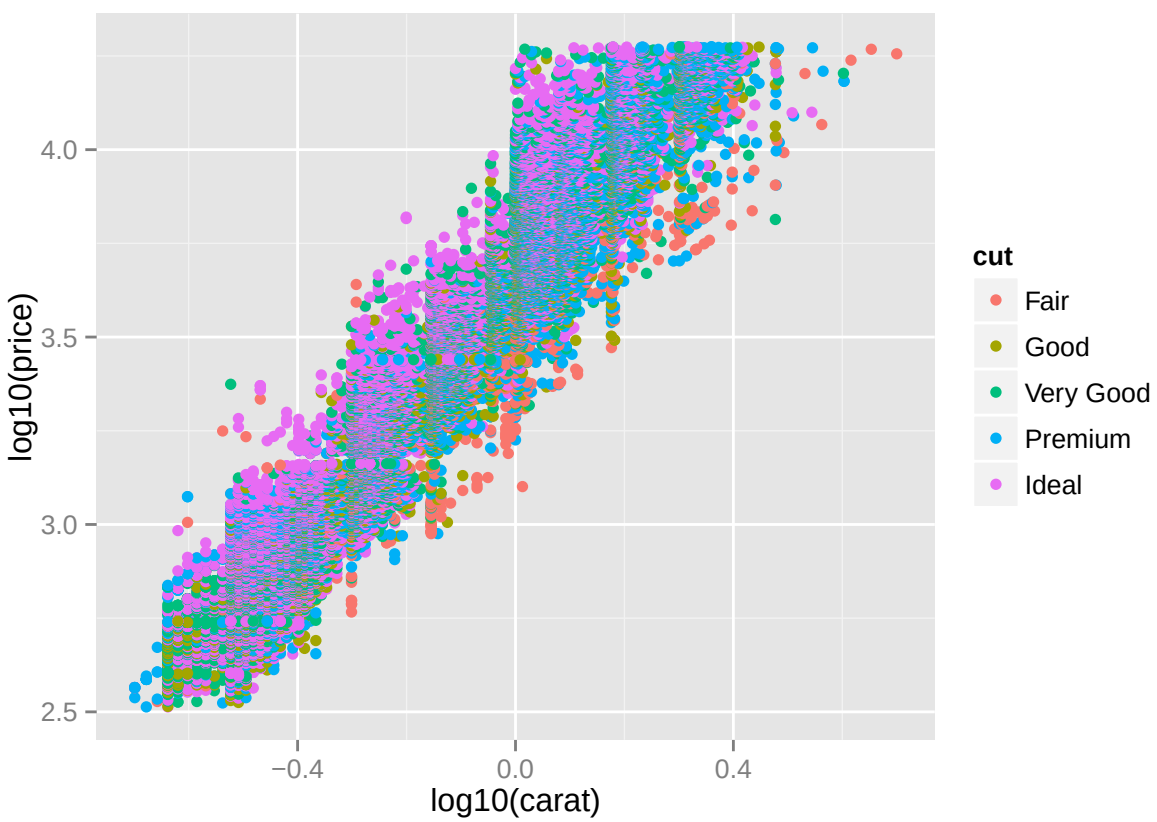

```
#
# Below is a plot of the price vs carat and using color for each of the
# five cuts using ggplot which offers more flexibility than qplot.
ggplot(diamonds, aes(x=carat, y=price, color=cut)) + geom_point()
```



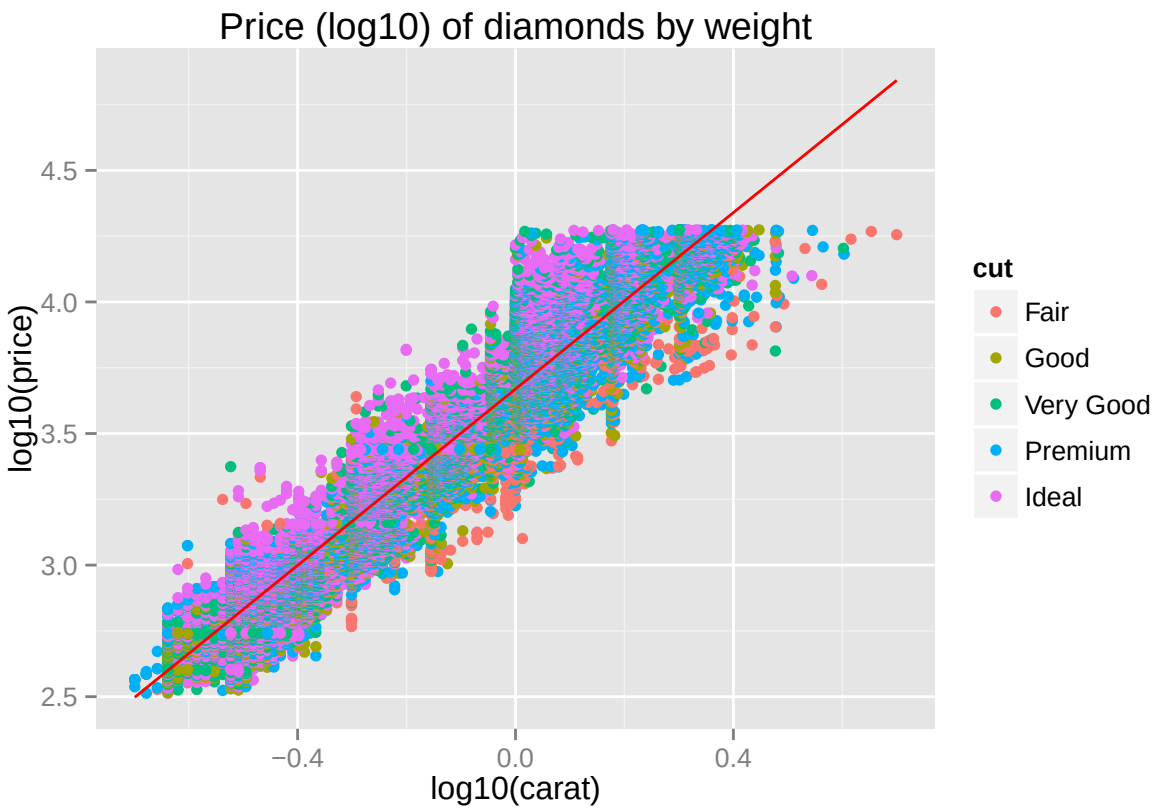

```
# Looks somewhat linear but hard to tell.
# Lets try plottin it on a log scale using qplot with log10
# first in BW using shape and also with color
# A title is created using main
qplot(log10(carat), log10(price), shape = cut, data = diamonds)
```




```
qplot(log10(carat), log10(price), color = cut, data = diamonds)
```

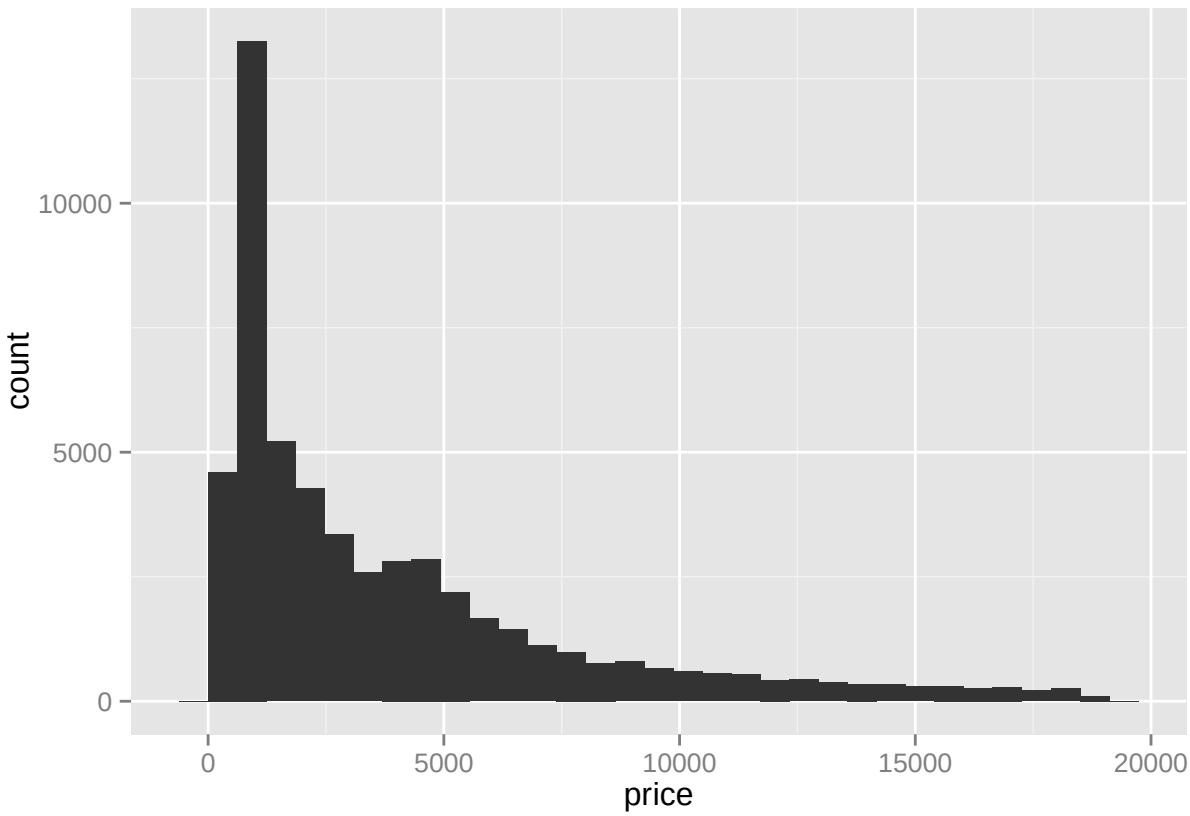


```r
# Using log10 the plot now looks linear
#
# Now, use geom_smooth to add a nice linear fit
qplot(log10(carat), log10(price), color = cut, data = diamonds,
  main="Price (log10) of diamonds by weight") +
  geom_smooth(method = "lm", color = "red")
```

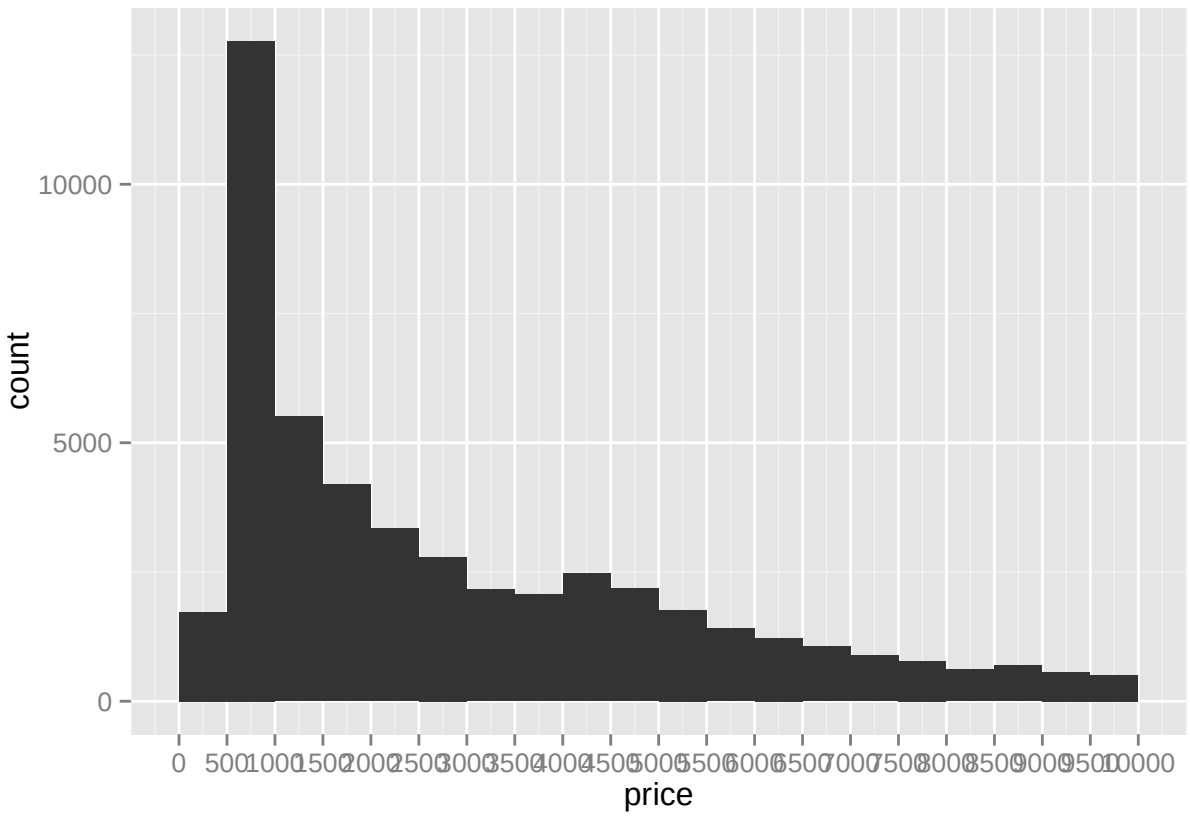



```r
#
# Next lets make some plots of the no. of diamonds vs price
qplot(x=price, data=diamonds)
```

```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```

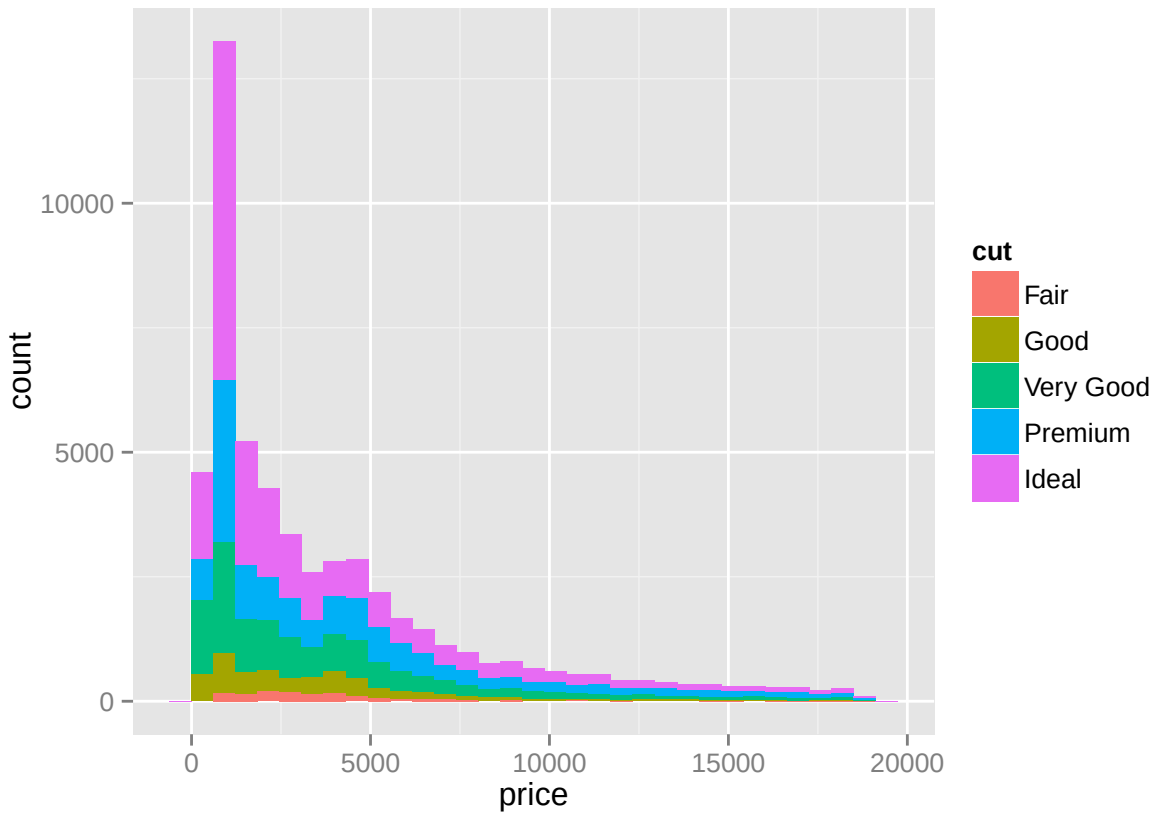



```
# we can see that the prices for diamonds are heavily skewed with the
# most common price at around $700.
# Lets use scale_x_continuous to zoom in on the x-axis
qplot(x=price, data=diamonds, binwidth=500) +
  scale_x_continuous(limits = c(0, 10000), breaks=seq(0,10000,500))
```

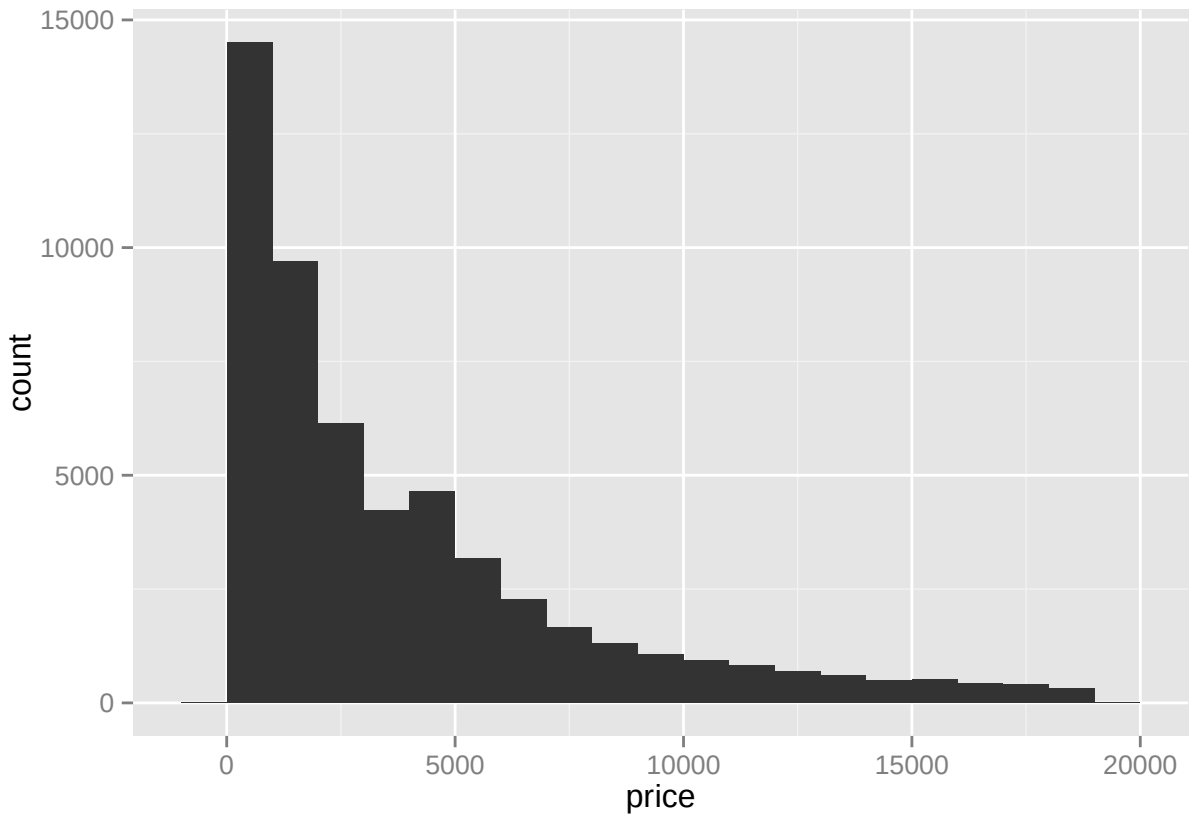



```r
#
# Using ggplot with the aesthetic wrapper we have a color overlay for each cut
ggplot(diamonds, aes(x=price, fill=cut)) + geom_histogram()
```

```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```

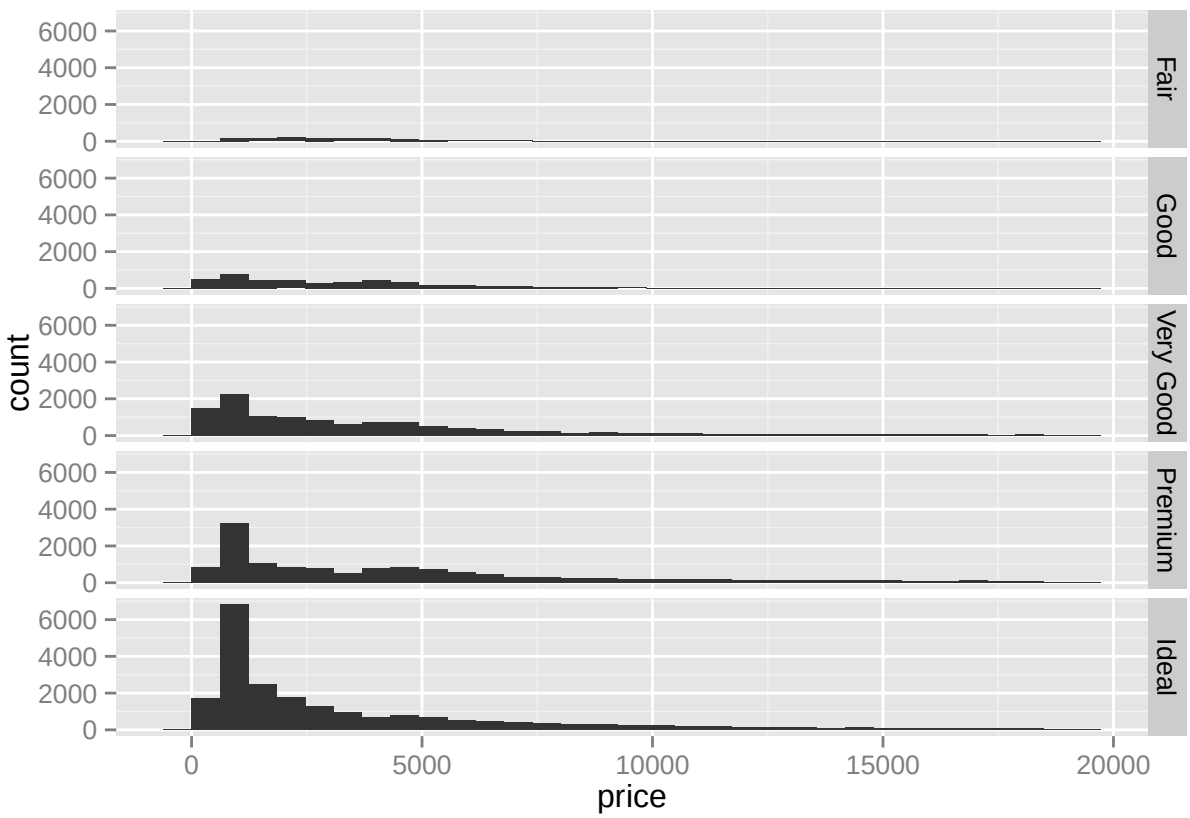



```
ggplot(diamonds, aes(x=price)) + geom_histogram(binwidth=1000)
```



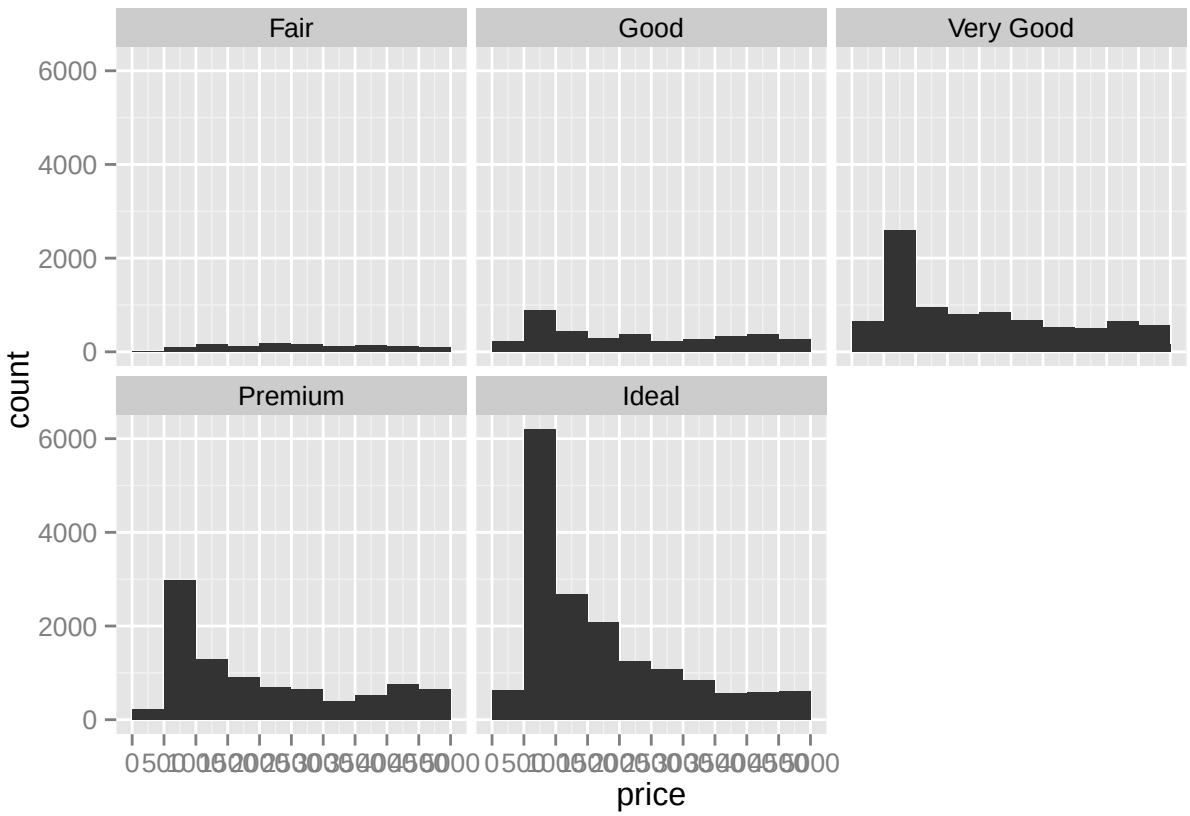
```r
#
# Lets break up the histogram of diamond prices by cut using facet
# to yield 5 separate plots using qplot for simplicity. Recall the quality of
# the cuts we have are: Fair, Good, Very Good, Premium, Ideal
qplot(x = price, data = diamonds) +  facet_grid(cut ~ .)
```

```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```



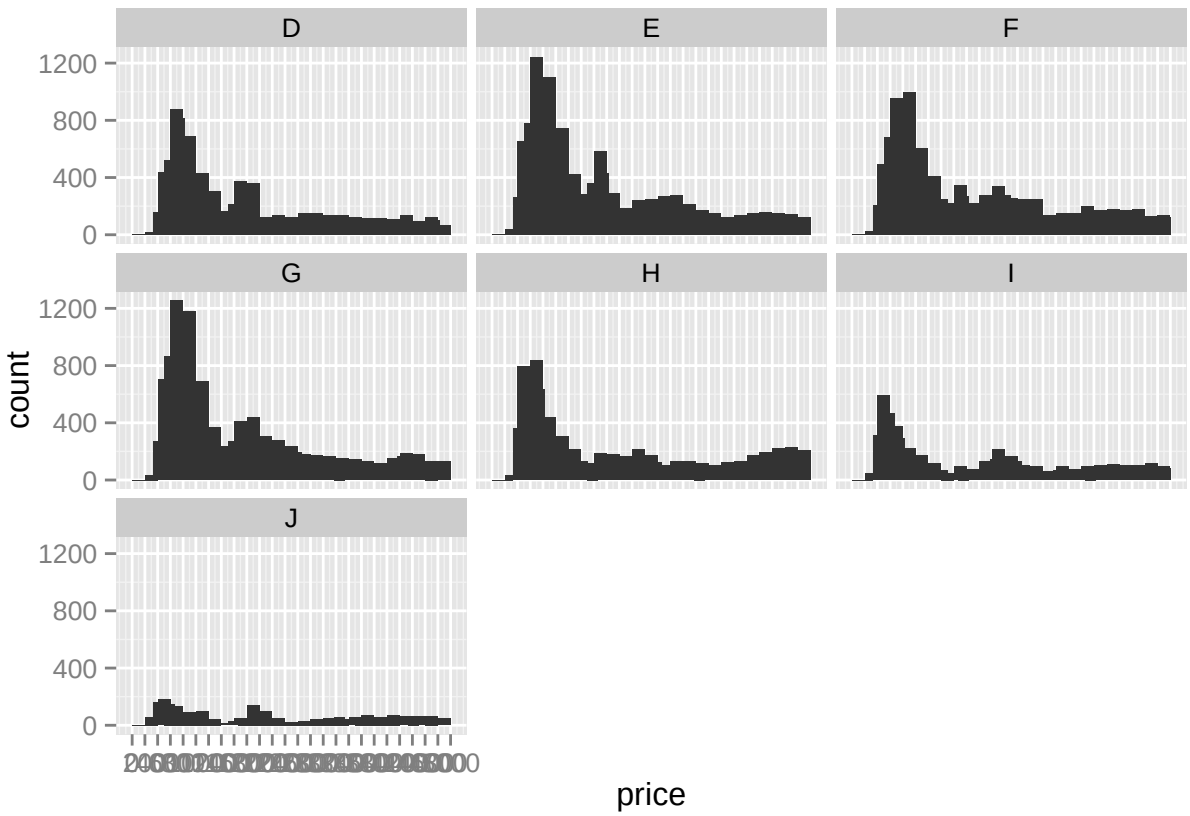

```r
# Lets limit the x-axis and take another look. But lets use facet_wrap
# which strings together ggplots in different frames based in a single variable
# Facet by cut:
qplot(x=price, data=diamonds, binwidth=500) +
  scale_x_continuous(limits = c(0, 5000), breaks=seq(0,5000,500)) +
  geom_histogram() +
  facet_wrap(~cut)
```

```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```



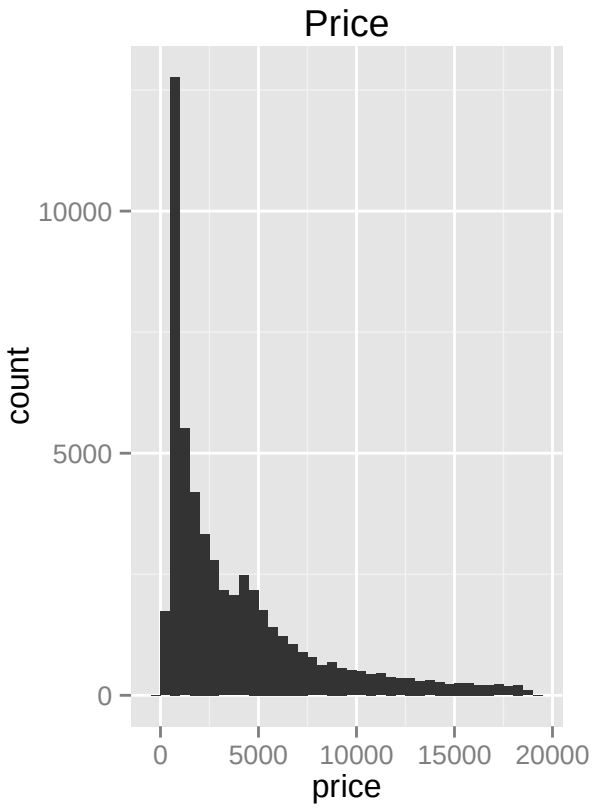

```r
# and by color:
qplot(x=price, data=diamonds, binwidth=200) +
  scale_x_continuous(limits = c(0, 5000), breaks=seq(0,5000,200)) +
  geom_histogram() +
  facet_wrap(~color)
```

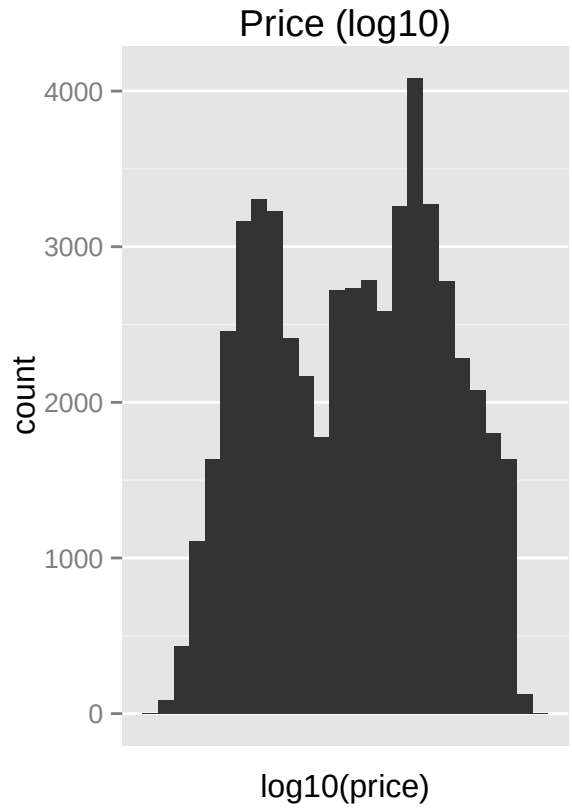
```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```




```
#
# We can put 2 plots together using grid.arrange from the gridExtra package
# Lets compare price and log10(price)
library(gridExtra)
#
plot1 <- qplot(x=price, data=diamonds, binwidth=500) +
  ggtitle('Price')
#
plot2 <- qplot(x=log10(price), data=diamonds, binwidth=0.01) +
  scale_x_log10() +
  ggtitle('Price (log10)')
#
grid.arrange(plot1, plot2, ncol=2)
```

```
## Warning: position_stack requires constant width: output may be incorrect
```






```r
#
#
# We can look numerically at the price per cut using the "by" command:
by(diamonds$price, diamonds$cut, summary)
```

```
## diamonds$cut: Fair
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     337    2050    3280    4360    5210   18600
## ------------------------------------------------------------
## diamonds$cut: Good
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     327    1140    3050    3930    5030   18800
## ------------------------------------------------------------
## diamonds$cut: Very Good
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     336     912    2650    3980    5370   18800
## ------------------------------------------------------------
## diamonds$cut: Premium
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     326    1050    3180    4580    6300   18800
## ------------------------------------------------------------
## diamonds$cut: Ideal
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     326     878    1810    3460    4680   18800
```

```r
#
# Here's the max & min per cut
by(diamonds$price, diamonds$cut, max)
```

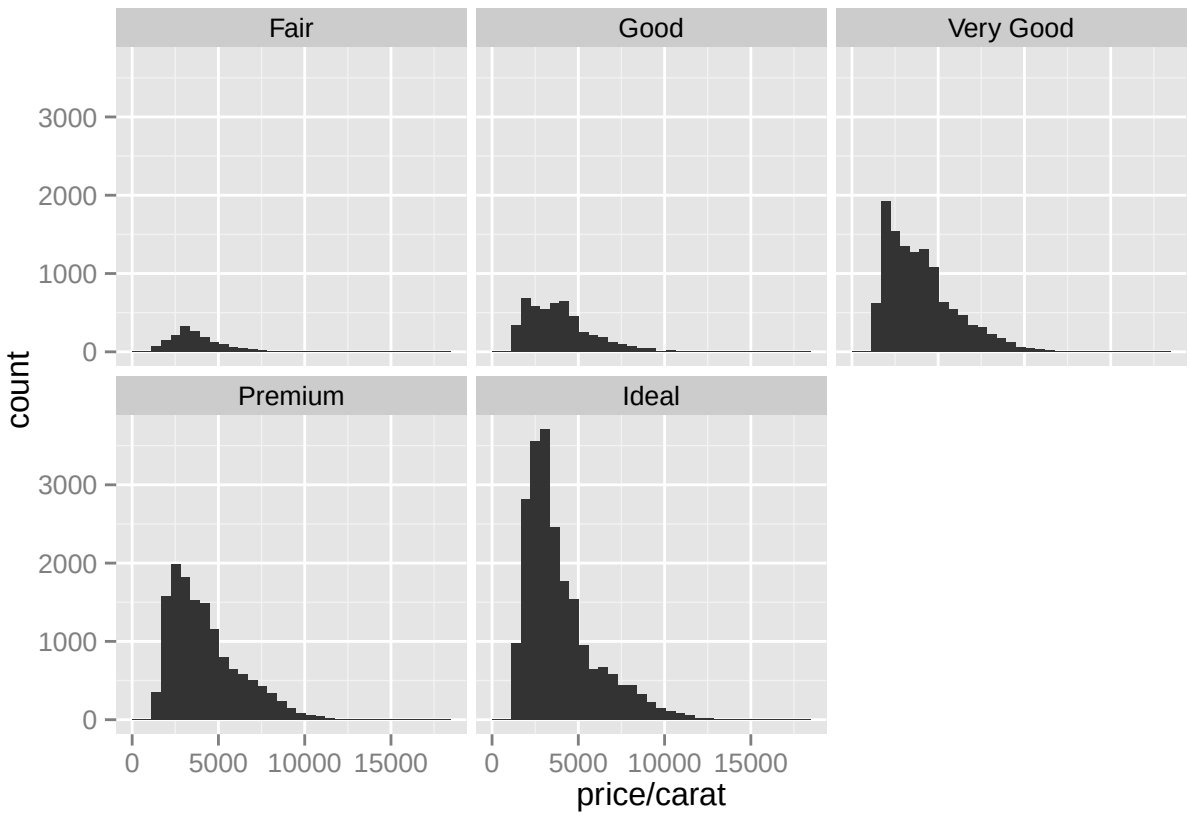
```
## diamonds$cut: Fair
## [1] 18574
## ------------------------------------------------------------
## diamonds$cut: Good
## [1] 18788
## ------------------------------------------------------------
## diamonds$cut: Very Good
## [1] 18818
## ------------------------------------------------------------
## diamonds$cut: Premium
## [1] 18823
## ------------------------------------------------------------
## diamonds$cut: Ideal
## [1] 18806
```

```r
by(diamonds$price, diamonds$cut, min)
```

```
## diamonds$cut: Fair
## [1] 337
## ------------------------------------------------------------
## diamonds$cut: Good
## [1] 327
## ------------------------------------------------------------
## diamonds$cut: Very Good
## [1] 336
## ------------------------------------------------------------
## diamonds$cut: Premium
## [1] 326
## ------------------------------------------------------------
## diamonds$cut: Ideal
## [1] 326
```

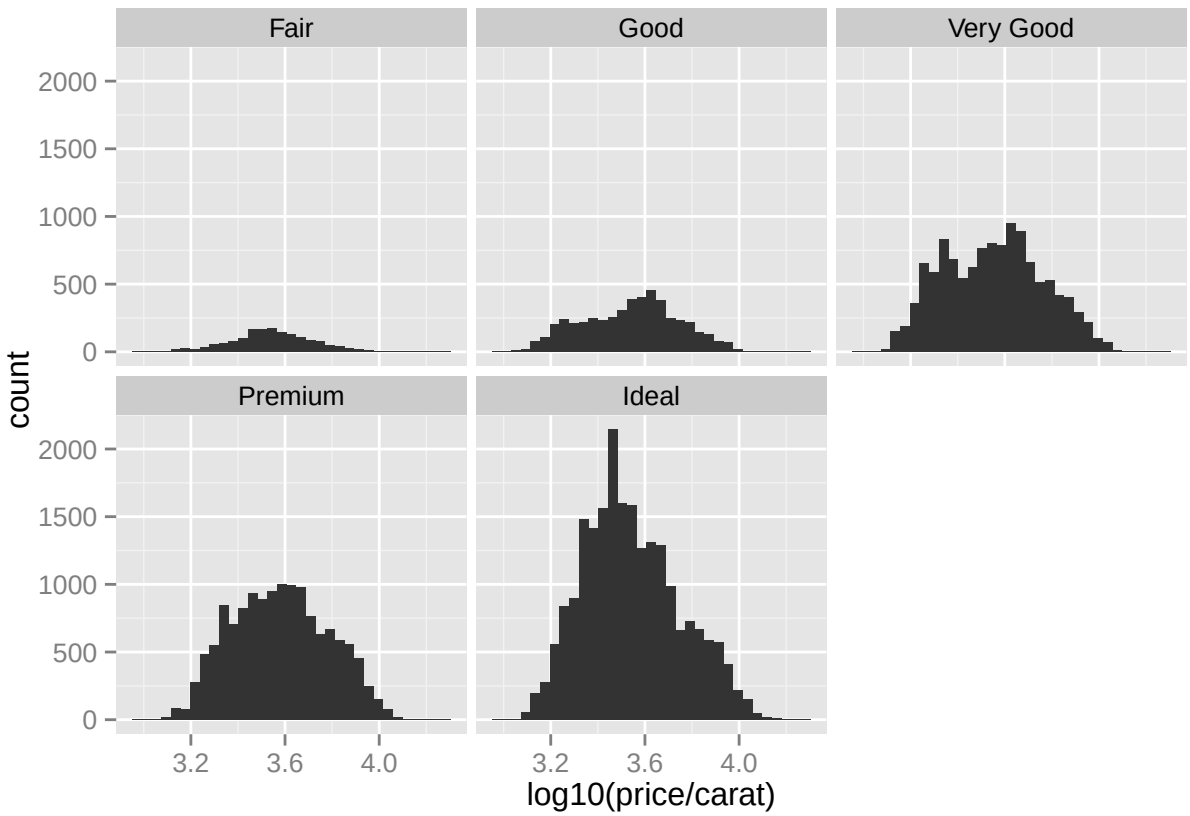
```r
#
# The 'Fair' and 'Good' diamonds appear to have different distributions
# compared to the better cut diamonds. They seem somewhat uniform
# on the left with long tails on the right. Let's look into this some more.
#
# Lets create a histogram of price per carat and facet it by cut
qplot(x = price/carat, data = diamonds) + facet_wrap(~cut)
```

```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```

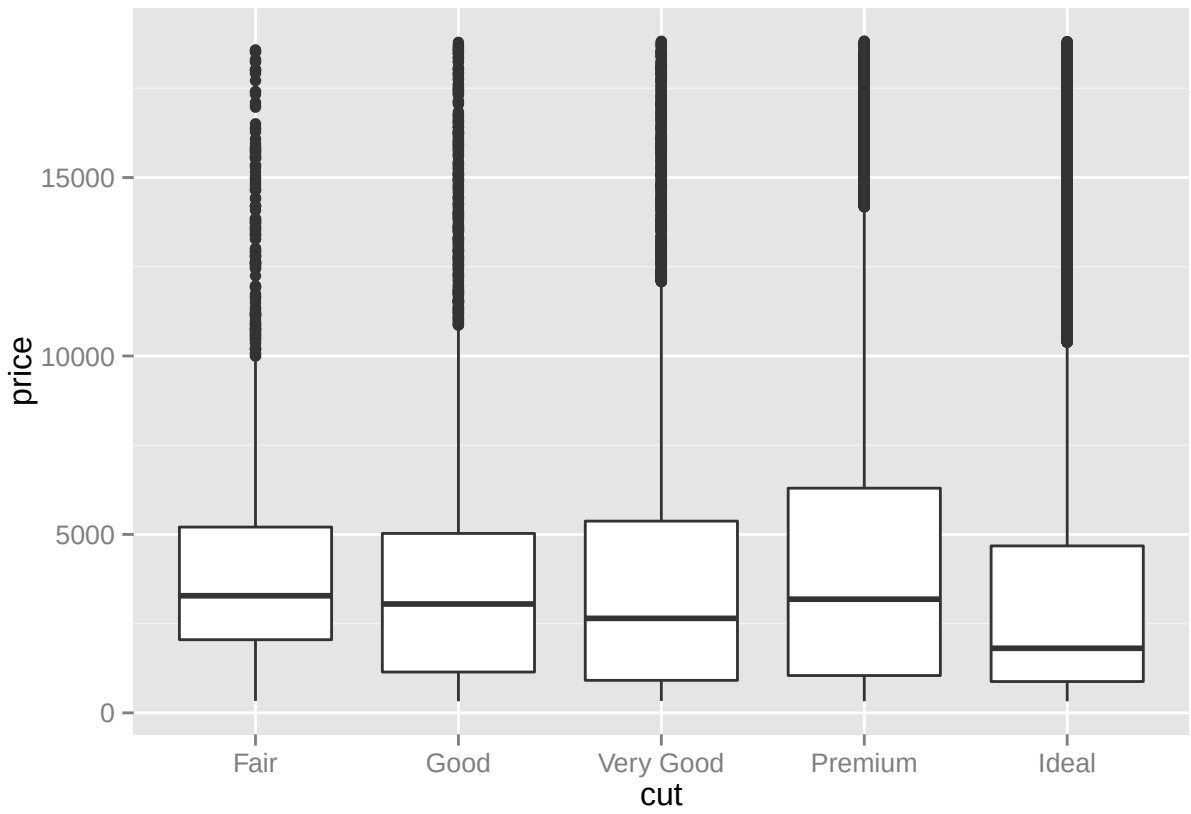



```r
qplot(x = log10(price/carat), data = diamonds) + facet_wrap(~cut)
```

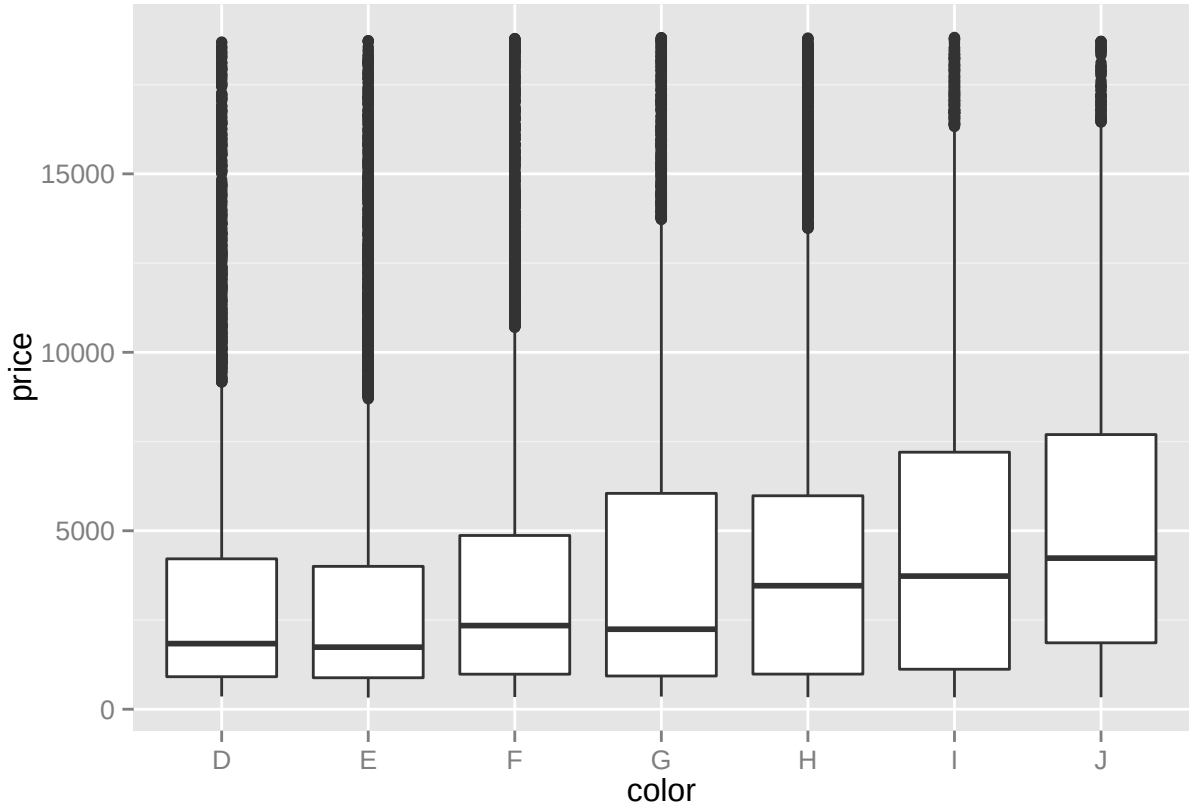
```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```



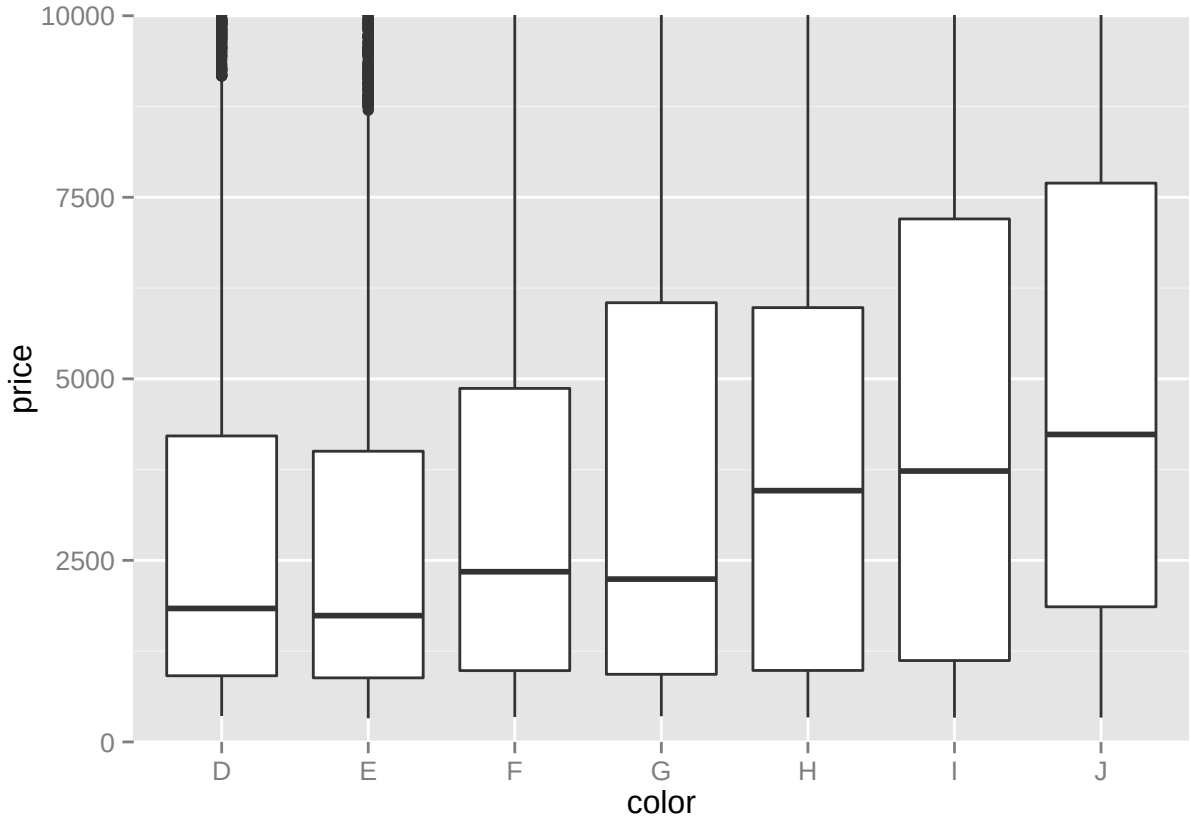

```
# These plots show that Very Good and Premium have normalize distributions
#
# Next, we investigate the price of diamonds using box plots,
# numerical summaries, and one of the following categorical
# variables: cut, clarity, or color.
#
qplot(x = cut, y = price, data = diamonds, geom = 'boxplot')
```




```
qplot(x = color, y = price, data = diamonds, geom = 'boxplot')
```

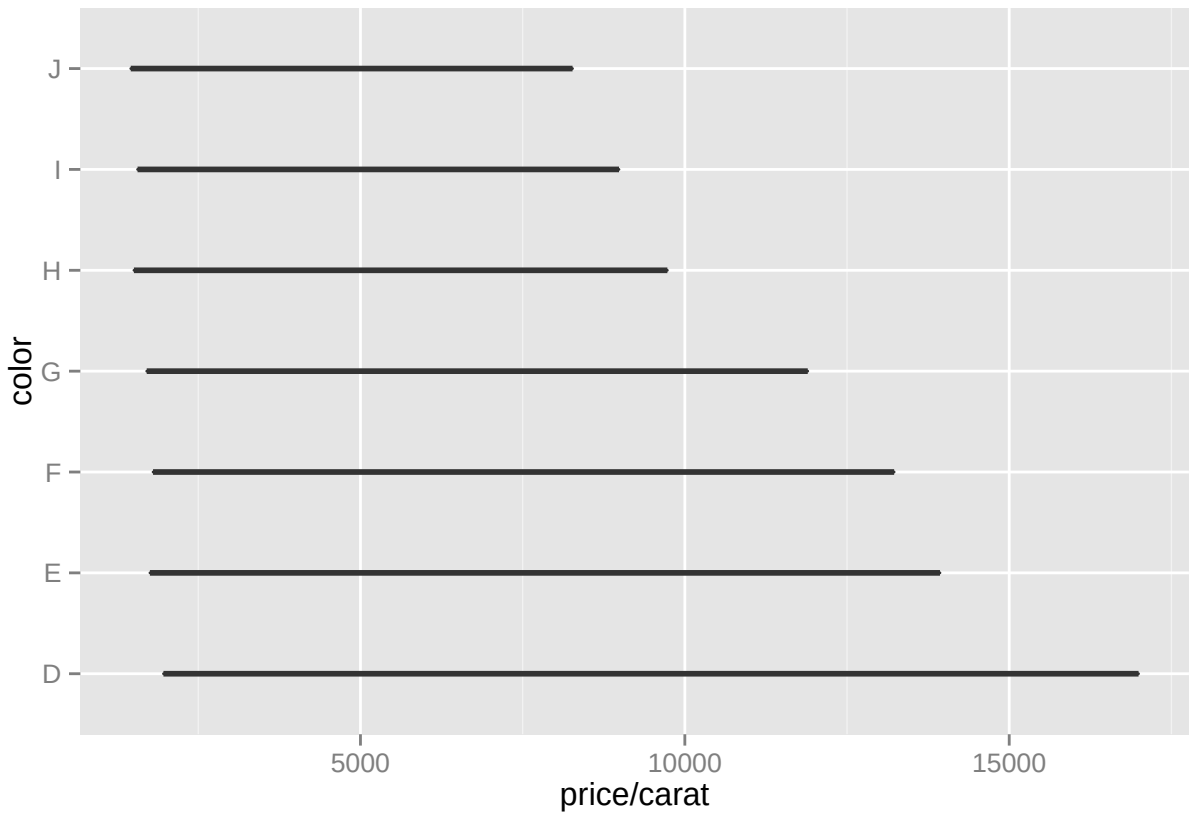


```r
# Lets zoom in a little
qplot(x = color, y = price, data = diamonds, geom = 'boxplot') +
  coord_cartesian(ylim = c(0,10000))
```




```r
#
qplot(x = price/carat, y = color, data = diamonds, geom = 'boxplot')
```

```
## Warning: position_dodge requires constant width: output may be incorrect
## Warning: position_dodge requires non-overlapping x intervals
```




```r
#
# Summary stats for each color :
by(diamonds$price, diamonds$color, summary)
```

```
## diamonds$color: D
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     357     911    1840    3170    4210   18700
## ------------------------------------------------------------
## diamonds$color: E
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     326     882    1740    3080    4000   18700
## ------------------------------------------------------------
## diamonds$color: F
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     342     982    2340    3720    4870   18800
## ------------------------------------------------------------
## diamonds$color: G
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     354     931    2240    4000    6050   18800
## ------------------------------------------------------------
## diamonds$color: H
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     337     984    3460    4490    5980   18800
## ------------------------------------------------------------
## diamonds$color: I
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     334    1120    3730    5090    7200   18800
```

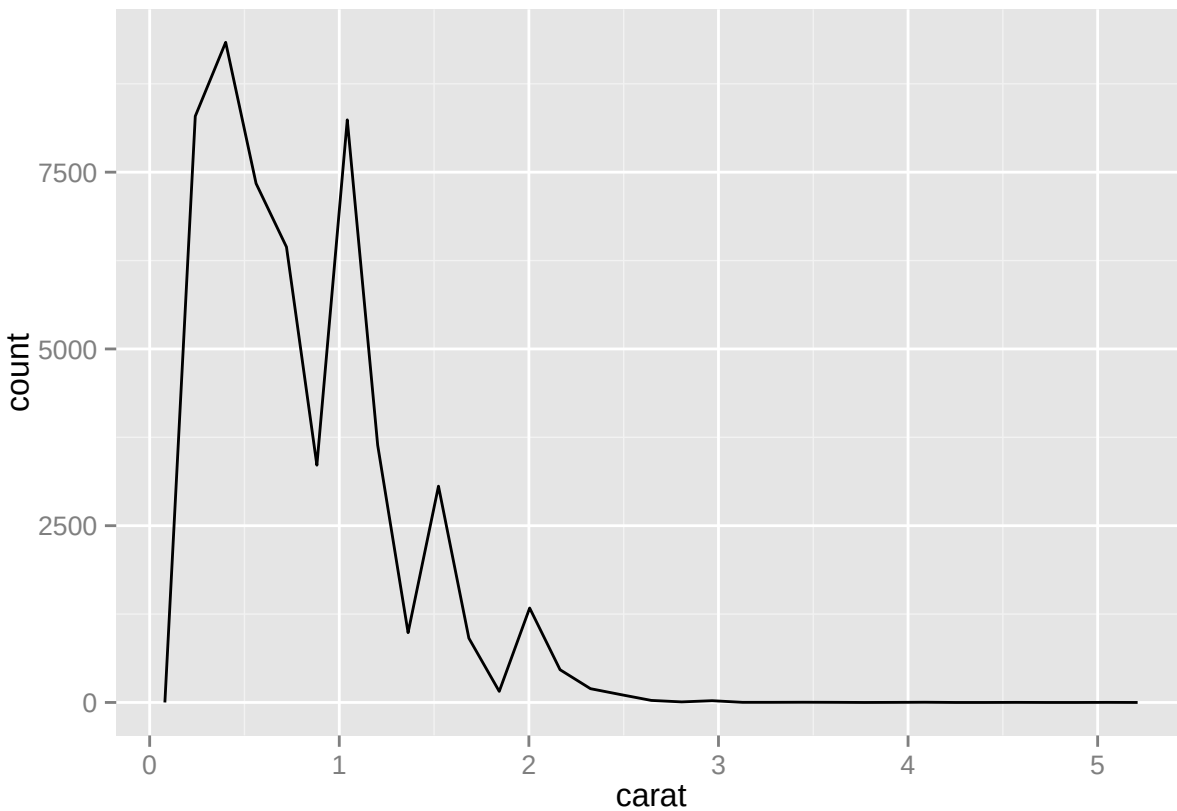
```
## ------------------------------------------------------------
## diamonds$color: J
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     335    1860    4230    5320    7700   18700
```

```r
#
IQR(subset(diamonds, color='D')$price)
```

```
## [1] 4374
```

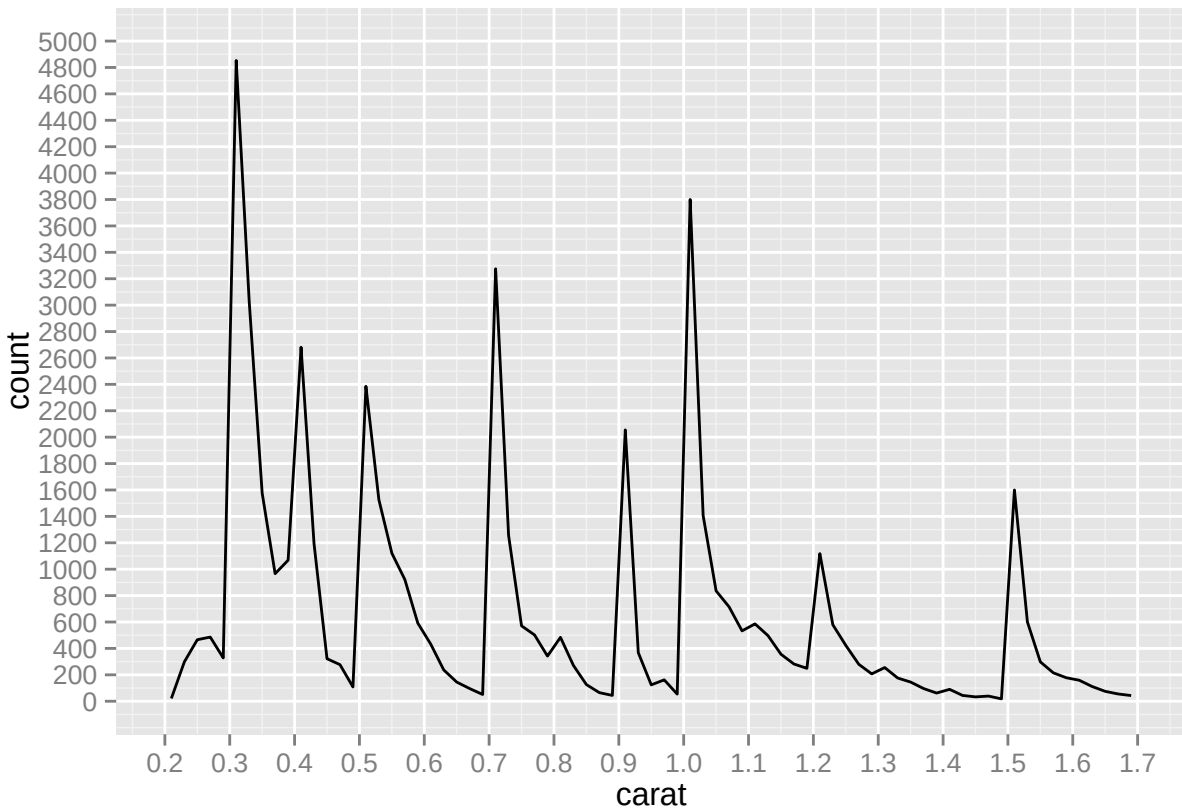
```r
#
# Here's a frequency polygon for counts vs carat
qplot(x = carat, data = diamonds, geom = 'freqpoly')
```

```
## stat_bin: binwidth defaulted to range/30. Use 'binwidth = x' to adjust this.
```




```r
qplot(x = carat, data = diamonds, geom = 'freqpoly', binwidth=0.02)+
  scale_x_continuous(lim = c(0.2,1.7), breaks = seq(0.2,1.7,0.1)) +
  scale_y_continuous(lim = c(0,5000), breaks = seq(0,5000,200))
```

```
## Warning: Removed 2 rows containing missing values (geom_path).
```




```r
#
# correlation coeffs between price and x,y,z
cor.test(diamonds$x, diamonds$price, method='pearson')
```

```
##
##  Pearson's product-moment correlation
##
## data:  diamonds$x and diamonds$price
## t = 440.2, df = 53938, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.8826 0.8863
## sample estimates:
##    cor
## 0.8844
```

```r
cor.test(diamonds$y, diamonds$price, method='pearson')
```

```
##
##  Pearson's product-moment correlation
##
## data:  diamonds$y and diamonds$price
## t = 401.1, df = 53938, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.8633 0.8675
## sample estimates:
```

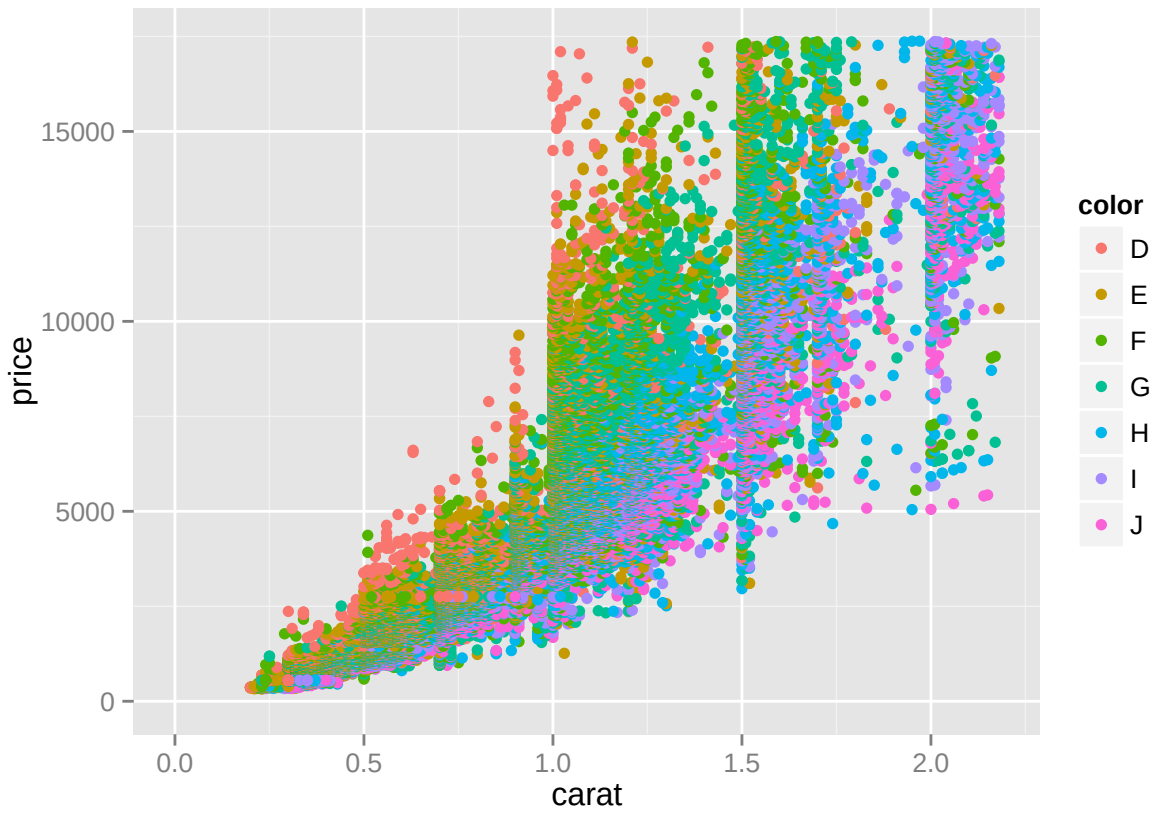
```
##    cor
## 0.8654
```

```r
cor.test(diamonds$z, diamonds$price, method='pearson')
```

```
##
##  Pearson's product-moment correlation
##
## data:  diamonds$z and diamonds$price
## t = 393.6, df = 53938, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.8591 0.8634
## sample estimates:
##    cor
## 0.8612
```

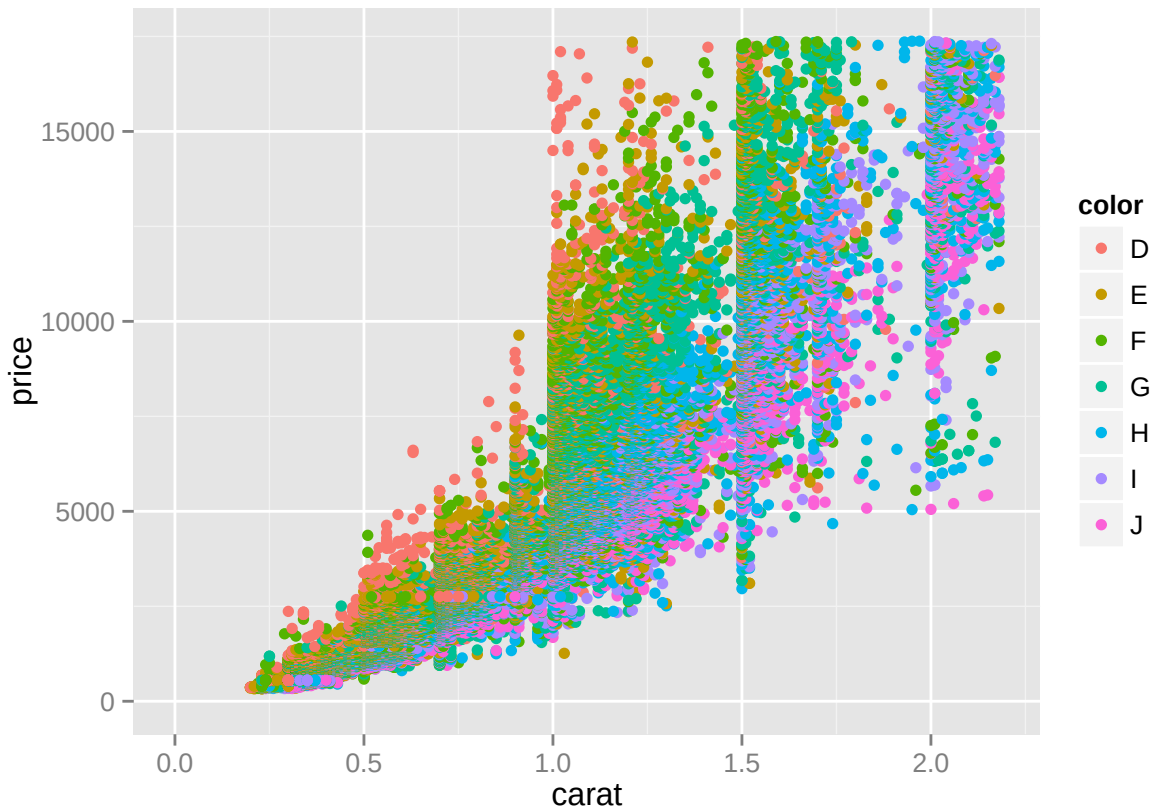
```r
#
# Now lets examine two variables in the data set.
# The scatterplot is a powerful tool to help you understand
# the relationship between two continuous variables.

# We can quickly see if the relationship is linear or not.
# In this case, we can use a variety of diamond
# characteristics to help us figure out whether
# the price advertised for any given diamond is
# reasonable or a rip-off.
#
# Let's consider the price of a diamond and it's carat weight.
# Create a scatterplot of price (y) vs carat weight (x). Here, we
# limit the x-axis and y-axis to omit the top 1% of values using quantile.
#
ggplot(aes(x=carat, y=price), data=diamonds) +
  xlim(0, quantile(diamonds$carat, 0.99)) +
  ylim(0, quantile(diamonds$price, 0.99)) +
  geom_point(aes(color = color))
```

```
## Warning: Removed 926 rows containing missing values (geom_point).
```



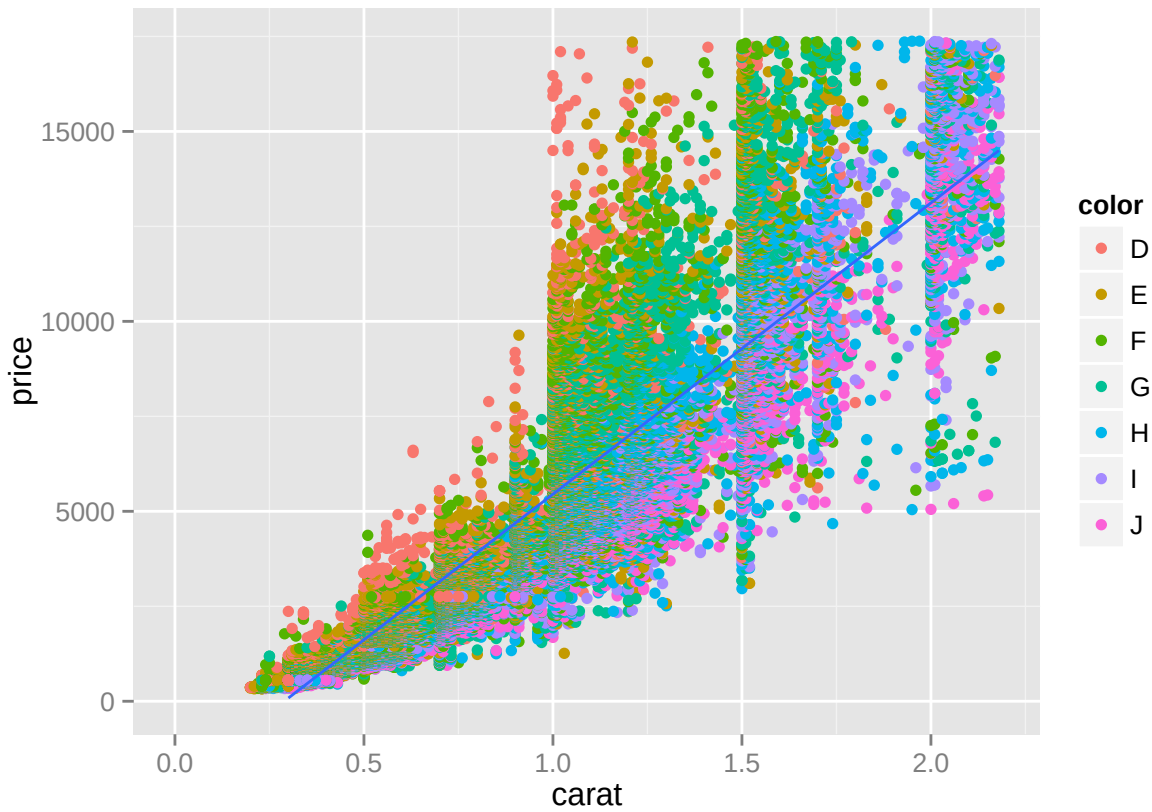

```
#
# using scale_continuous:
#
  ggplot(aes(x=carat, y=price), data=diamonds) +
  scale_x_continuous(lim = c(0, quantile(diamonds$carat, 0.99))) +
  scale_y_continuous(lim = c(0, quantile(diamonds$price, 0.99))) +
  geom_point(aes(color = color))
```

```
## Warning: Removed 926 rows containing missing values (geom_point).
```



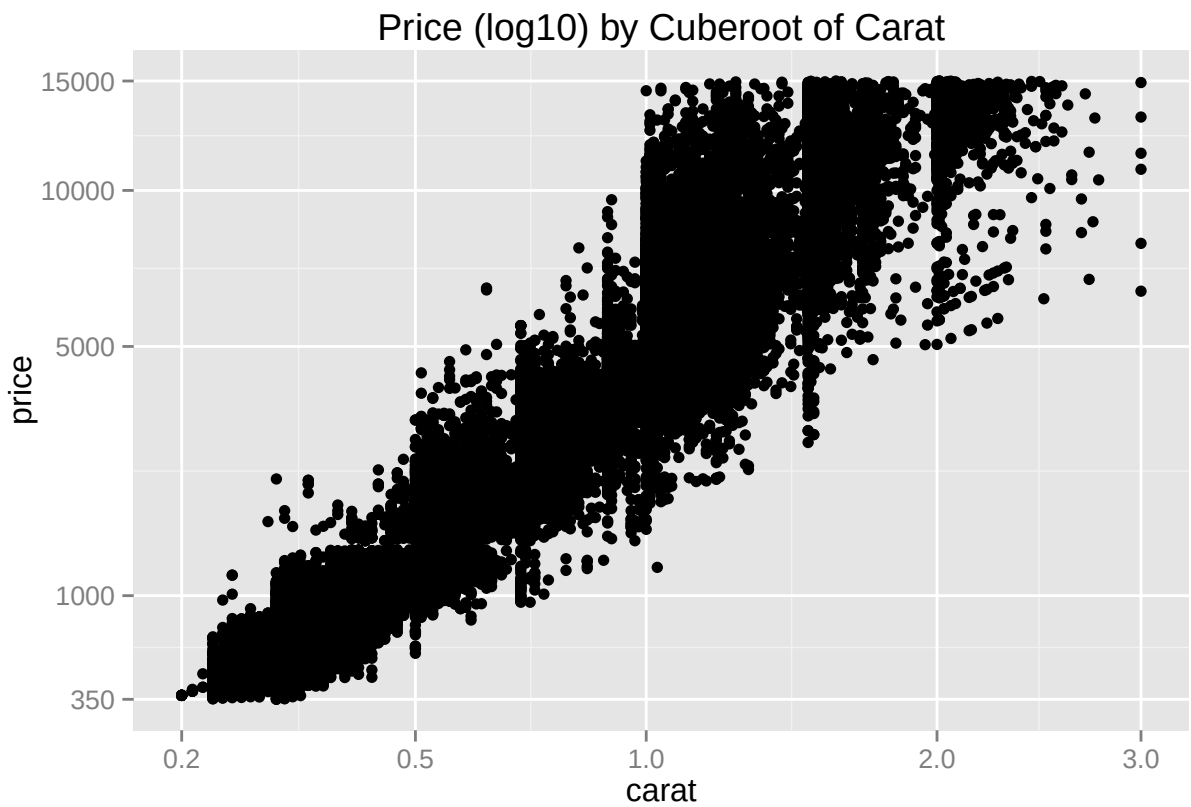

```
#
# We can create models using the lm() function where
#   lm ( outcome variable ~ explanatory variable)
# adding a linear fit,
#
ggplot(aes(x=carat, y=price), data=diamonds) +
  scale_x_continuous(lim = c(0, quantile(diamonds$carat, 0.99))) +
  scale_y_continuous(lim = c(0, quantile(diamonds$price, 0.99))) +
  geom_point(aes(color = color)) +
  stat_smooth(method = 'lm')
```

```
## Warning: Removed 926 rows containing missing values (stat_smooth).
## Warning: Removed 926 rows containing missing values (geom_point).
## Warning: Removed 4 rows containing missing values (geom_path).
```



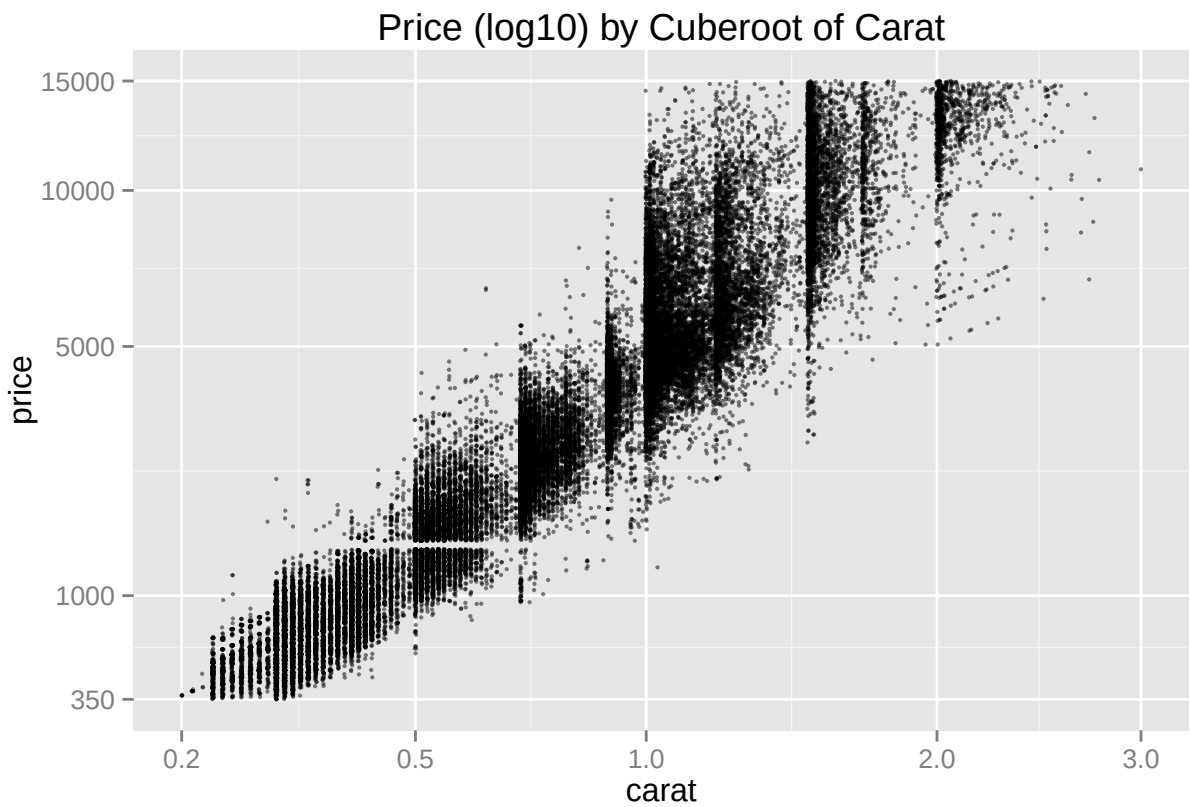

```
#
#
#Lets make a coordinate transformation an try looking at it using the
# cube root of the volume:
#
cuberoot_trans = function()trans_new('cuberoot',
                transform = function(x) x^(1/3),
                inverse = function(x) x^3)
#
ggplot(aes(carat, price), data = diamonds) +
  geom_point() +
  scale_x_continuous(trans = cuberoot_trans(), limits = c(0.2,3),
                     breaks = c(0.2, 0.5, 1, 2, 3)) +
  scale_y_continuous(trans = cuberoot_trans(), limits = c(350,15000),
                     breaks = c(350, 1000, 5000, 10000, 15000)) +
  ggtitle('Price (log10) by Cuberoot of Carat')
```

```
## Warning: Removed 1683 rows containing missing values (geom_point).
```



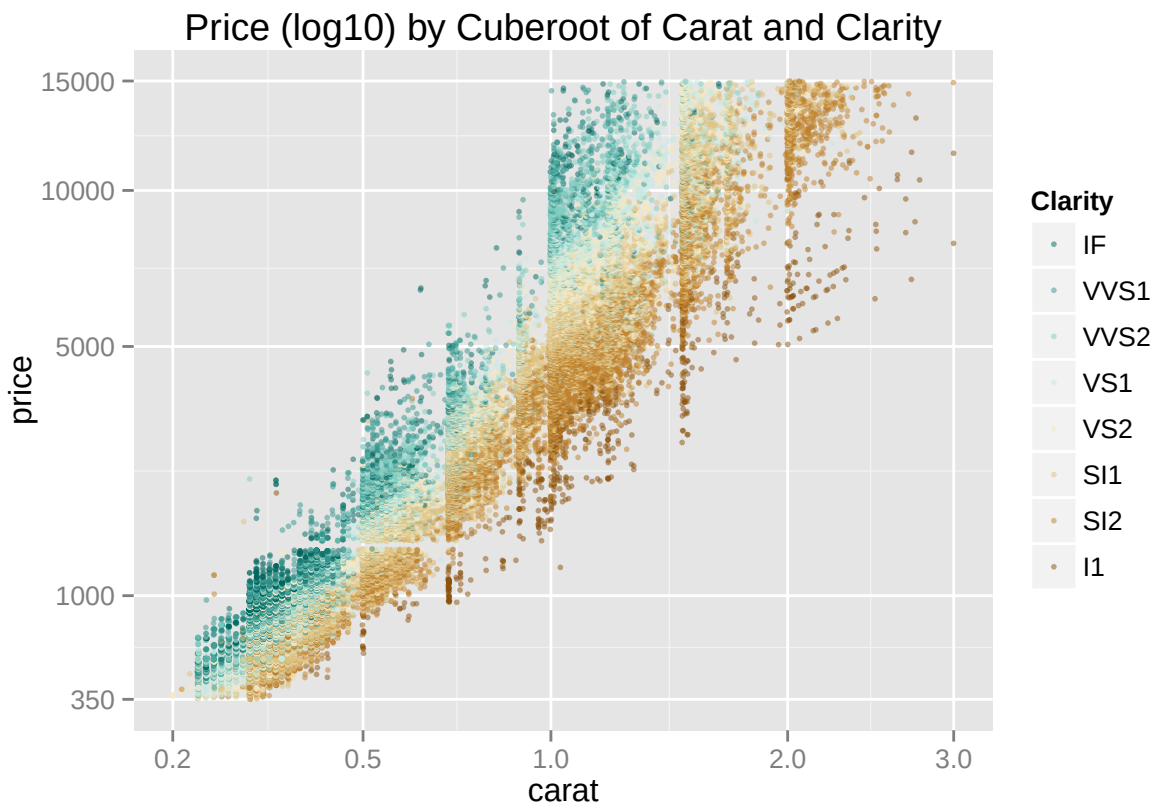

```
#
#So now we see its almost linear. Next lets address the over plotting.
#There are a lot, too many data points. Lets use alpha, size, jitter:
#
  ggplot(aes(carat, price), data = diamonds) +
  geom_point(alpha=1/2, size=0.75, position='jitter') +
  scale_x_continuous(trans = cuberoot_trans(), limits = c(0.2,3),
                     breaks = c(0.2, 0.5, 1, 2, 3)) +
  scale_y_continuous(trans = cuberoot_trans(), limits = c(350,15000),
                     breaks = c(350, 1000, 5000, 10000, 15000)) +
  ggtitle('Price (log10) by Cuberoot of Carat')
```

```
## Warning: Removed 1694 rows containing missing values (geom_point).
```



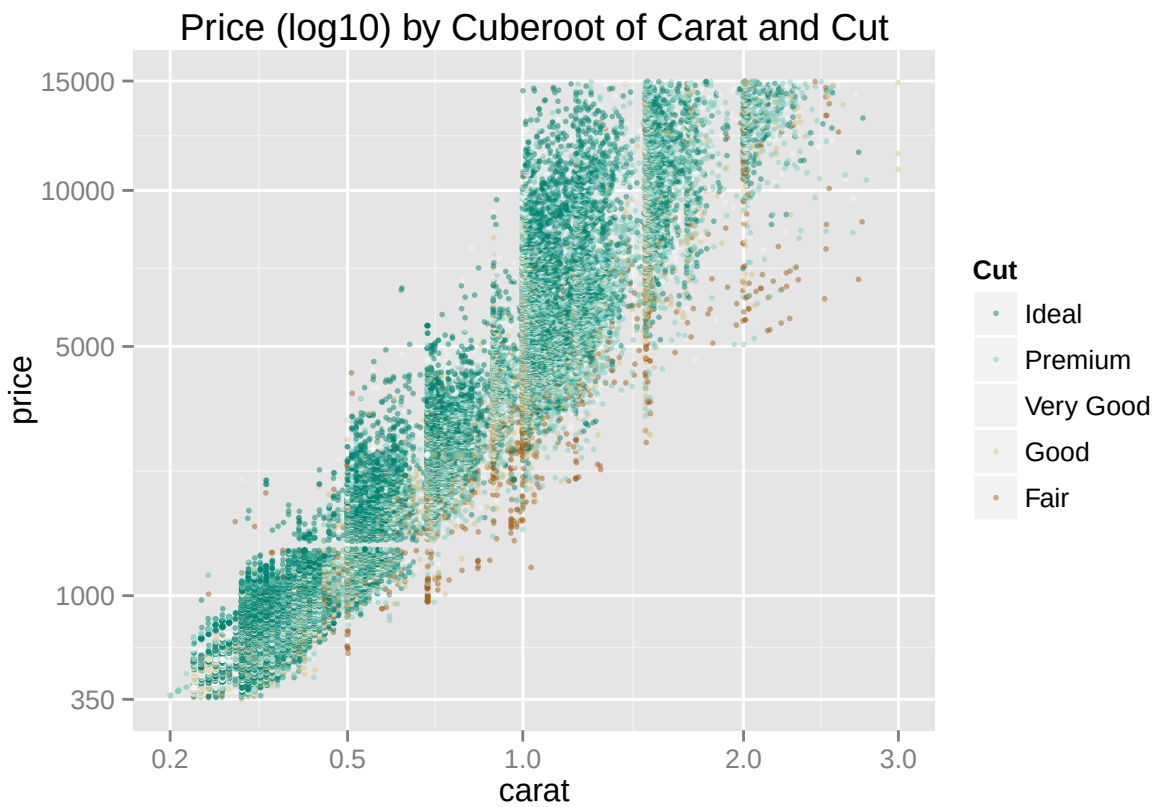

```
#
# Now lets use scale_color_brewer to adjust the legend and
# provide custom colors.
#
ggplot(aes(x=carat, y=price, color=clarity), data = diamonds) +
  geom_point(alpha=1/2, size=1, position='jitter') +
  scale_color_brewer(type = 'div',
                     guide = guide_legend(title = 'Clarity', reverse = TRUE,
                                          overide.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 = cuberoot_trans(), limits = c(350,15000),
                     breaks = c(350, 1000, 5000, 10000, 15000)) +
  ggtitle('Price (log10) by Cuberoot of Carat and Clarity')
```

```
## Warning: Removed 1691 rows containing missing values (geom_point).
```



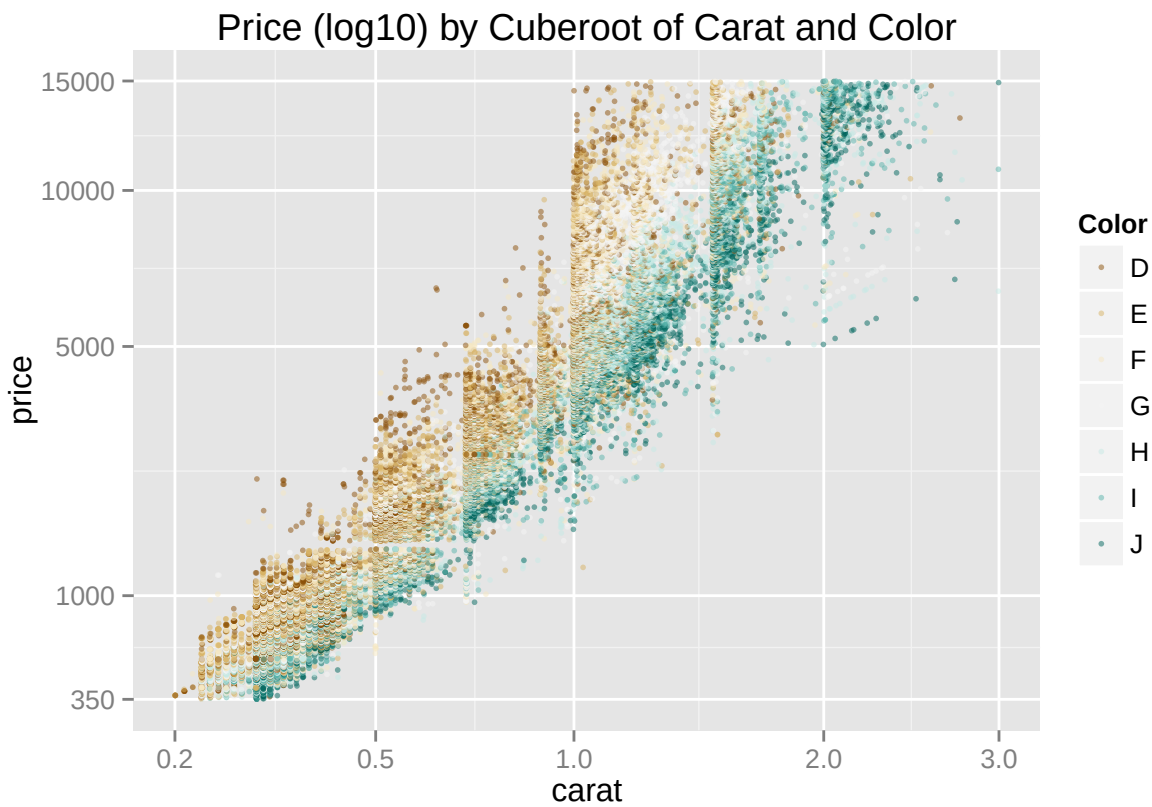

```
#
# This helps reduce the price per given carat. Lets look at the cut:
#
  ggplot(aes(x=carat, y=price, color=cut), data = diamonds) +
  geom_point(alpha=1/2, size=1, position='jitter') +
  scale_color_brewer(type = 'div',
                     guide = guide_legend(title = 'Cut', reverse = TRUE,
                                          overide.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 = cuberoot_trans(), limits = c(350,15000),
                     breaks = c(350, 1000, 5000, 10000, 15000)) +
  ggtitle('Price (log10) by Cuberoot of Carat and Cut')
```

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




```
#
# Don't see much price variation by cut. Lets look at diamond color
# by changing the aesthetic wrapper :
#
  ggplot(aes(x=carat, y=price, color=color), data = diamonds) +
  geom_point(alpha=1/2, size=1, position='jitter') +
  scale_color_brewer(type = 'div',
                     guide = guide_legend(title = 'Color', reverse = FALSE,
                                          overide.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 = cuberoot_trans(), limits = c(350,15000),
                     breaks = c(350, 1000, 5000, 10000, 15000)) +
  ggtitle('Price (log10) by Cuberoot of Carat and Color')
```

```
## Warning: Removed 1689 rows containing missing values (geom_point).
```




```
#
# Here I set reverse=FALSE so the best color is at the top of the list
# in the legend.
#
# Here we have log(price) as x and carat^(1/3) for y.
#
# Lets build a linear model for price with cut, color, and clarity added as variables. Carat was added
#
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
## =====================================================================
```

```r
#
# This gives us the model
#
# log(price) = 0.415 + 9.144 * (carat)^(1/3) - 1.093*carat + c1*cut + c2*color + c3*clarity
#
# Here 0.415 and 9.144 are given under m5. Note: This model is based on 2008 data.
# Can't simply adjust for inflation since the price at various carat has
# changed unevenly.
```