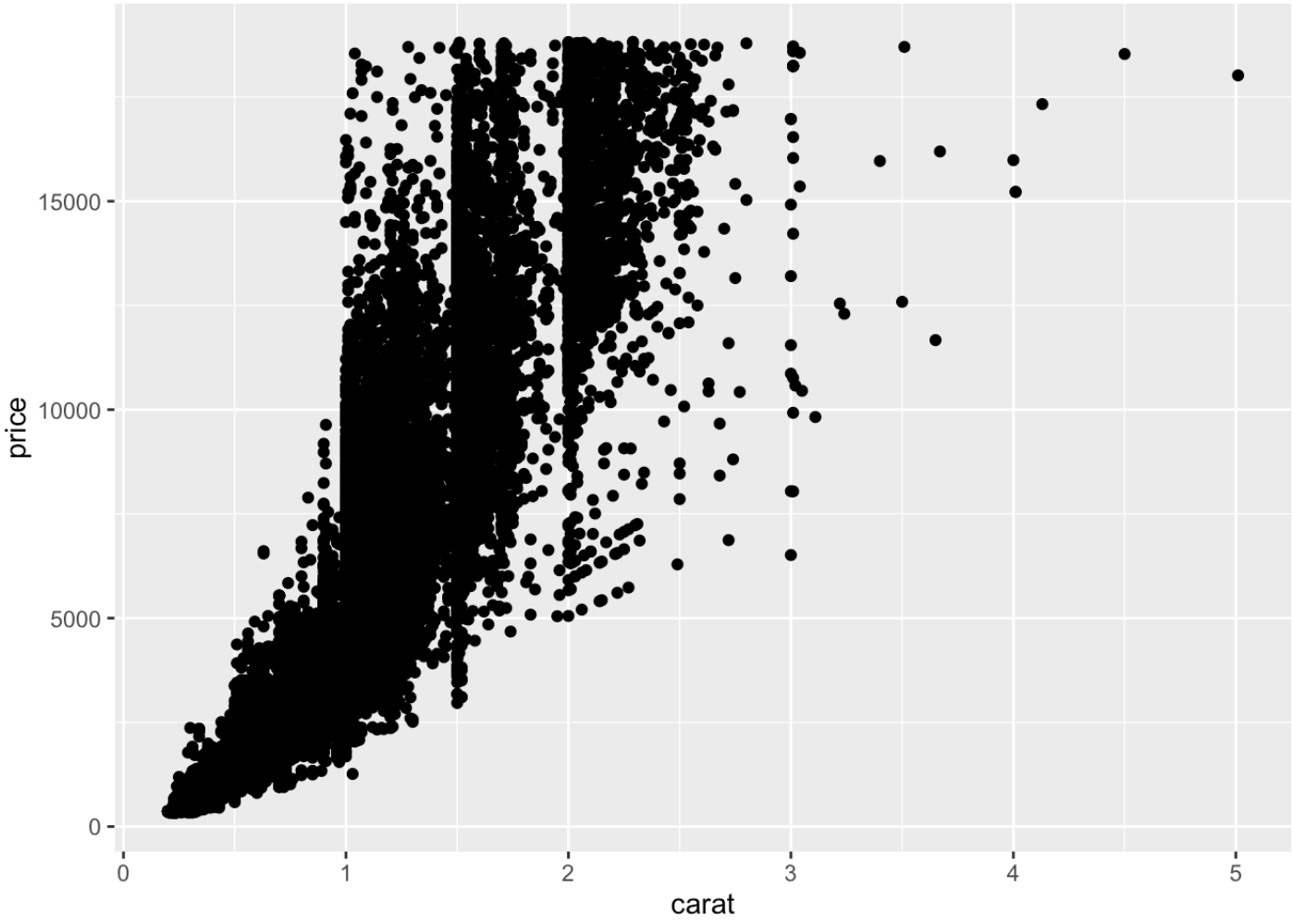
Lesson 6

Welcome

Notes:

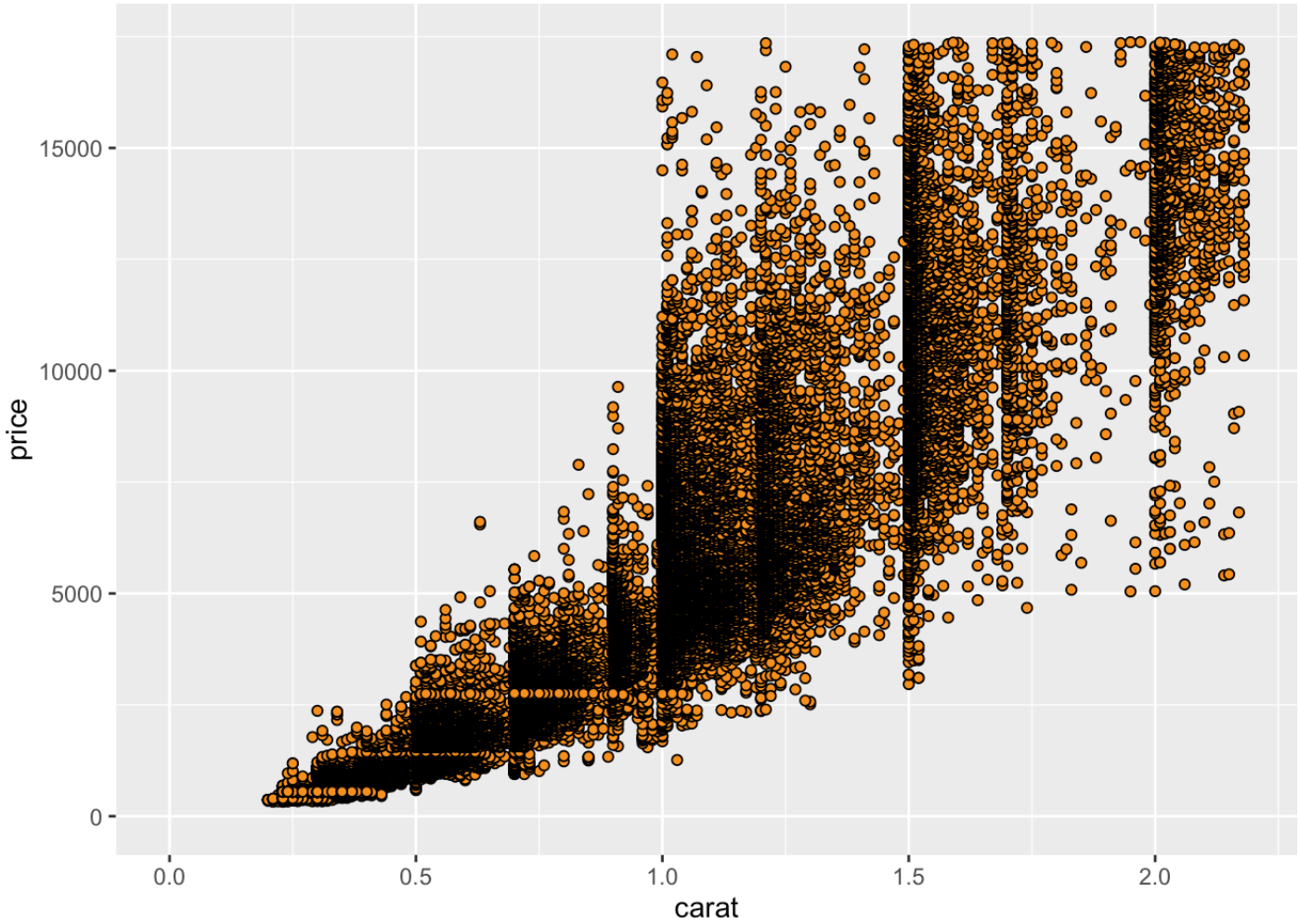
Scatterplot Review

```
library('ggplot2')
data(diamonds)

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




```
ggplot(data = diamonds, aes(carat, price)) +
  scale_x_continuous(lim = c(0, quantile(diamonds$carat, 0.99))) +
  scale_y_continuous(lim = c(0, quantile(diamonds$price, 0.99))) +
  geom_point(fill = I('#F79420'), color = I('black'), shape = 21)
```

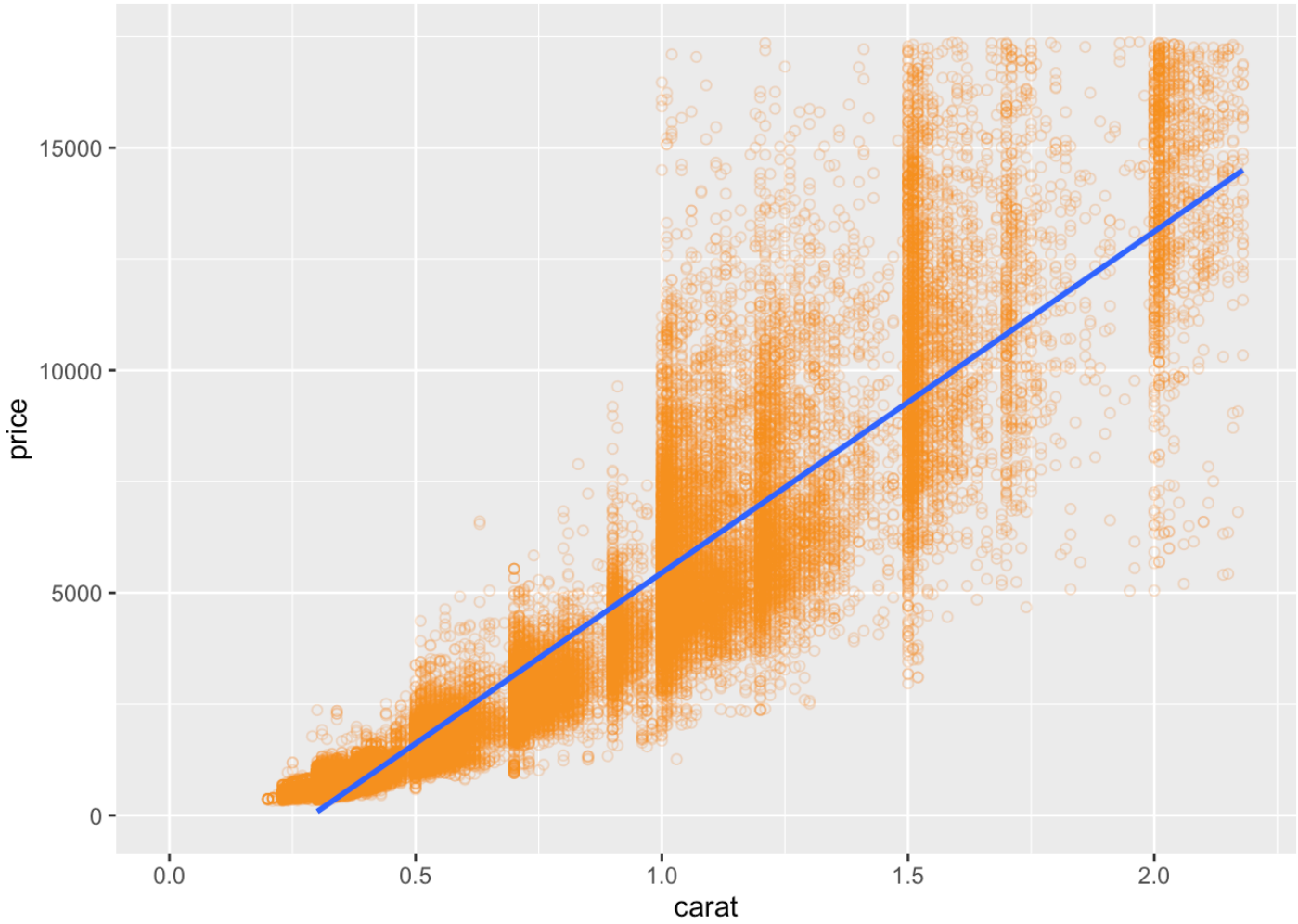
```
## Warning: Removed 926 rows containing missing values (geom_point).
```




```
 # with(diamonds, cor.test(price, carat))



#  add smooth line

ggplot(data = diamonds, aes(carat, price)) +
  scale_x_continuous(lim = c(0, quantile(diamonds$carat, 0.99))) +
  scale_y_continuous(lim = c(0, quantile(diamonds$price, 0.99))) +
  geom_point(color = I('#F79420'), alpha = 1/4, shape = 21) +
  stat_smooth(method = 'lm')
```

```
## Warning: Removed 926 rows containing non-finite values (stat_smooth).

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

```
## Warning: Removed 4 rows containing missing values (geom_smooth).
```




```
# put the trend line first and it was buried underneath the points

#add trendline to check how well the data fits around the trend line (linear trendl
ine)

#notice how the linear trendline is not aligned with the data in some key places wh
ere the line should be lower and higher


 # with(diamonds, cor.test(price, carat))
```

Price and Carat Relationship

Response:

Exponential non-linear patter. Blocks at every whole number

Frances Gerety

Notes:

A diamonds is

The Rise of Diamonds

Notes:

---

ggpairs Function

Notes: lower triangle -> grouped histograms -> qual/qual pairs scatter plots -> quant/quant pairs

upper triangle -> grouped histograms -> qual/qual (x value for grouping pairs) boxplot -> qual/quant pairs

diagonal = corrs for quant/quant pairs

```
# install these if necessary
# install.packages('GGally')
# install.packages('scales')
# install.packages('memisc') #summarise regression
# install.packages('lattice')
# install.packages('MASS')
# install.packages('car') #recode variables
# install.packages('reshape') #reshape/wrangle
# install.packages('plyr') #summaries/transforms

# load the ggplot graphics package and the others
library(ggplot2)
library(GGally)
library(scales)
library(memisc)
```

```
## Loading required package: lattice
```

```
## Loading required package: MASS
```

```
##
## Attaching package: 'memisc'
```

```
## The following object is masked from 'package:scales':
##
##     percent
```

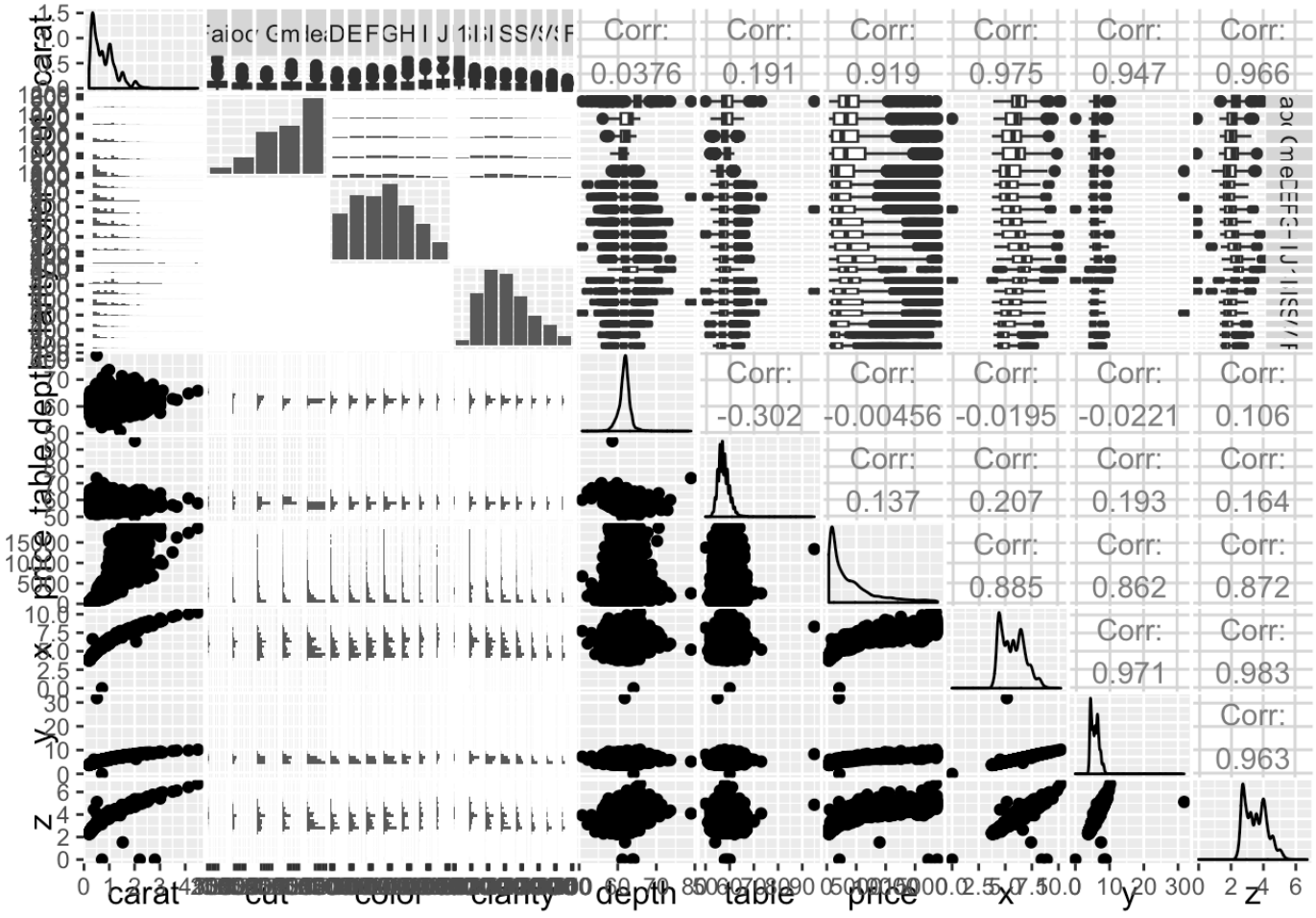
```
## The following objects are masked from 'package:stats':
##
##     contr.sum, contr.treatment, contrasts
```

```
## The following object is masked from 'package:base':
##
##     as.array
```

```
# sample 10,000 diamonds from the data set
set.seed(20022012)
diamond_samp <- diamonds[sample(1:length(diamonds$price), 10000), ]
ggpairs(diamond_samp)
```

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

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




```
help("wrap", package = "GGally")
```

What are some things you notice in the ggpairs output? Response: Price/Clarity | Price/Colour Critical factor is the carat (size) ***

The Demand of Diamonds

Notes: Price/Size - nonlinear Often the case that using substantive knowledge can lead to fruitful transformations

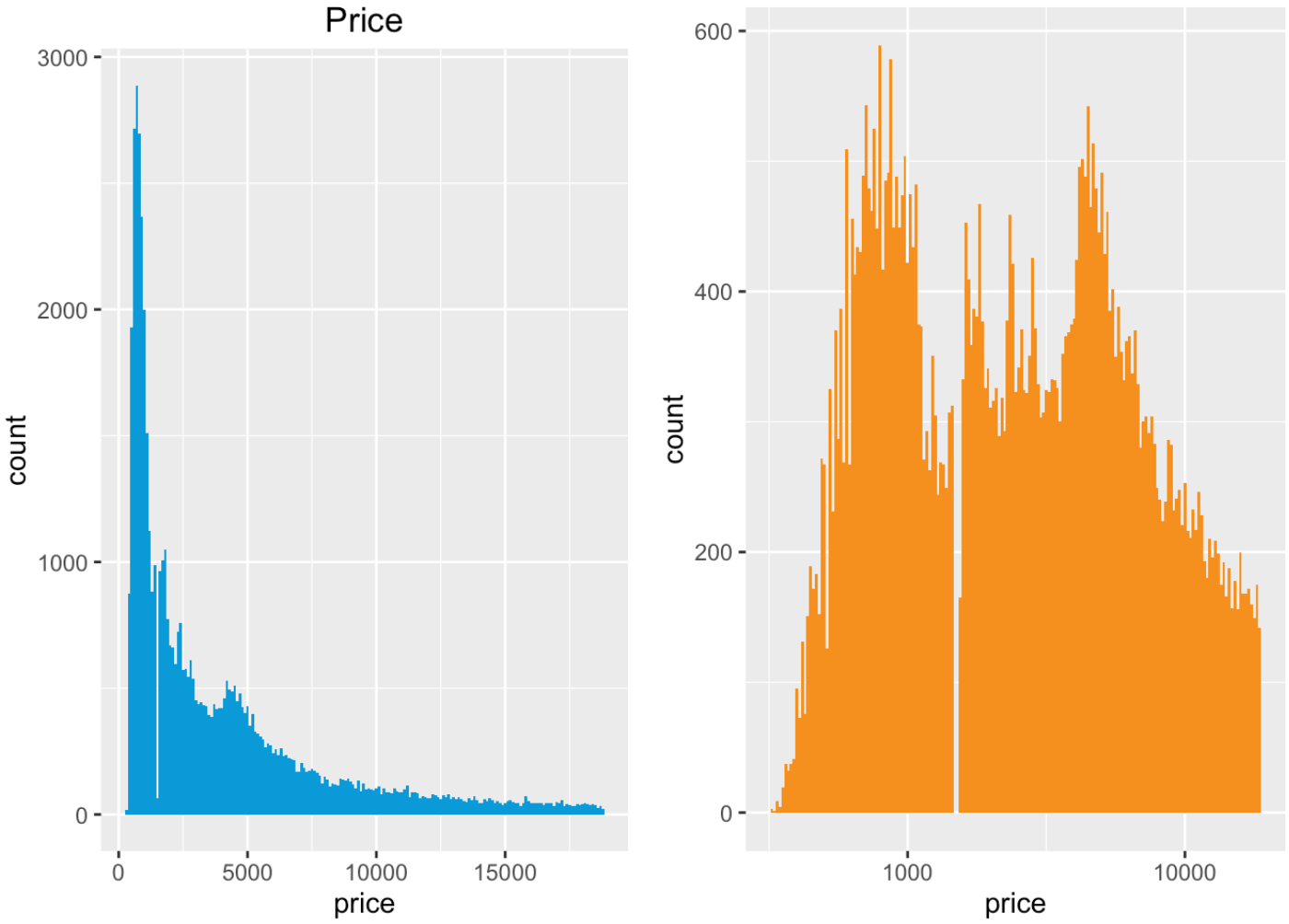
Path dependence Multiplicative processes like year on year inflation

The distribution monetary variable will be highly skewed due to path dependence or multiplicative processes. Good idea to look into compressing any such variable by putting it on a log scale. (Research later*)

```
plot1 <- ggplot(data = diamonds, aes(price)) +
  geom_histogram(binwidth = 100, fill = I('#099DD9')) +
  ggtitle("Price")

plot2 <- ggplot(data = diamonds, aes(price)) +
  geom_histogram(binwidth = 0.01, fill = I('#F79420')) +
  scale_x_log10()
  ggtitle("Price (log10)")
```

```
## $title
## [1] "Price (log10)"
##
## attr(,"class")
## [1] "labels"
```

```
library(gridExtra)
library(grid)
grid.arrange(plot1, plot2, ncol = 2)
```



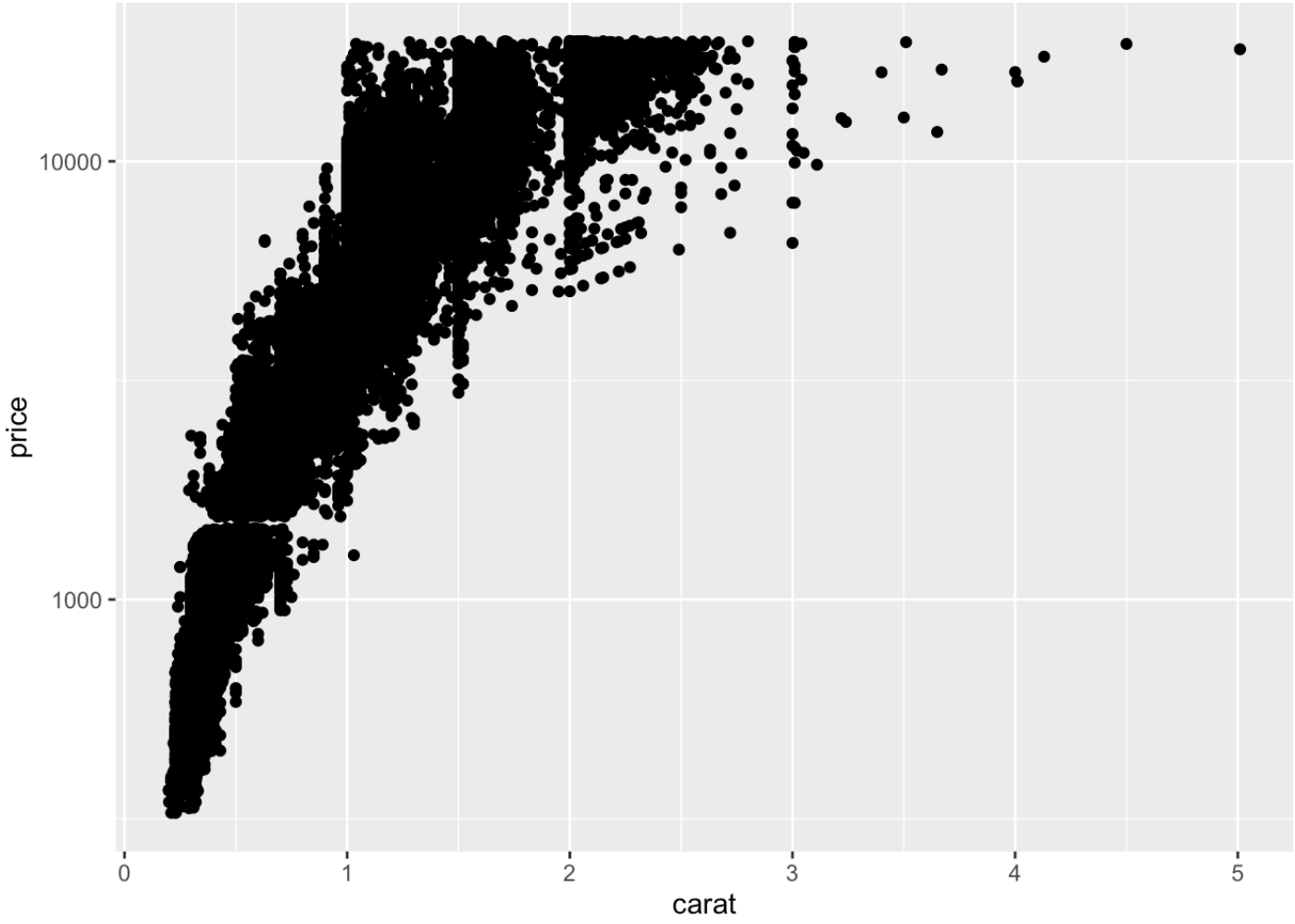
Connecting Demand and Price Distributions

Notes: The raw prices are skewed to the right. With a long tail.

The log10 distribution is closer to a normal distribution. Also showing signs of bimodality; where there may be two different sets of consumers who purchase at drastically different price points.

Scatterplot Transformation

```
ggplot(data = diamonds, aes(carat, price)) +
  geom_point(fill = I('#F79420')) +
  scale_y_continuous(trans = log10_trans())
```




```
  ggtitle("Price (log10) by Carat")
```

```
## $title
## [1] "Price (log10) by Carat"
##
## attr(,"class")
## [1] "labels"
```

```
  # because carats are measuring volume, we are getting exp. dispersion, so let's u
se the cubed root of carat cutting el*w*h
```

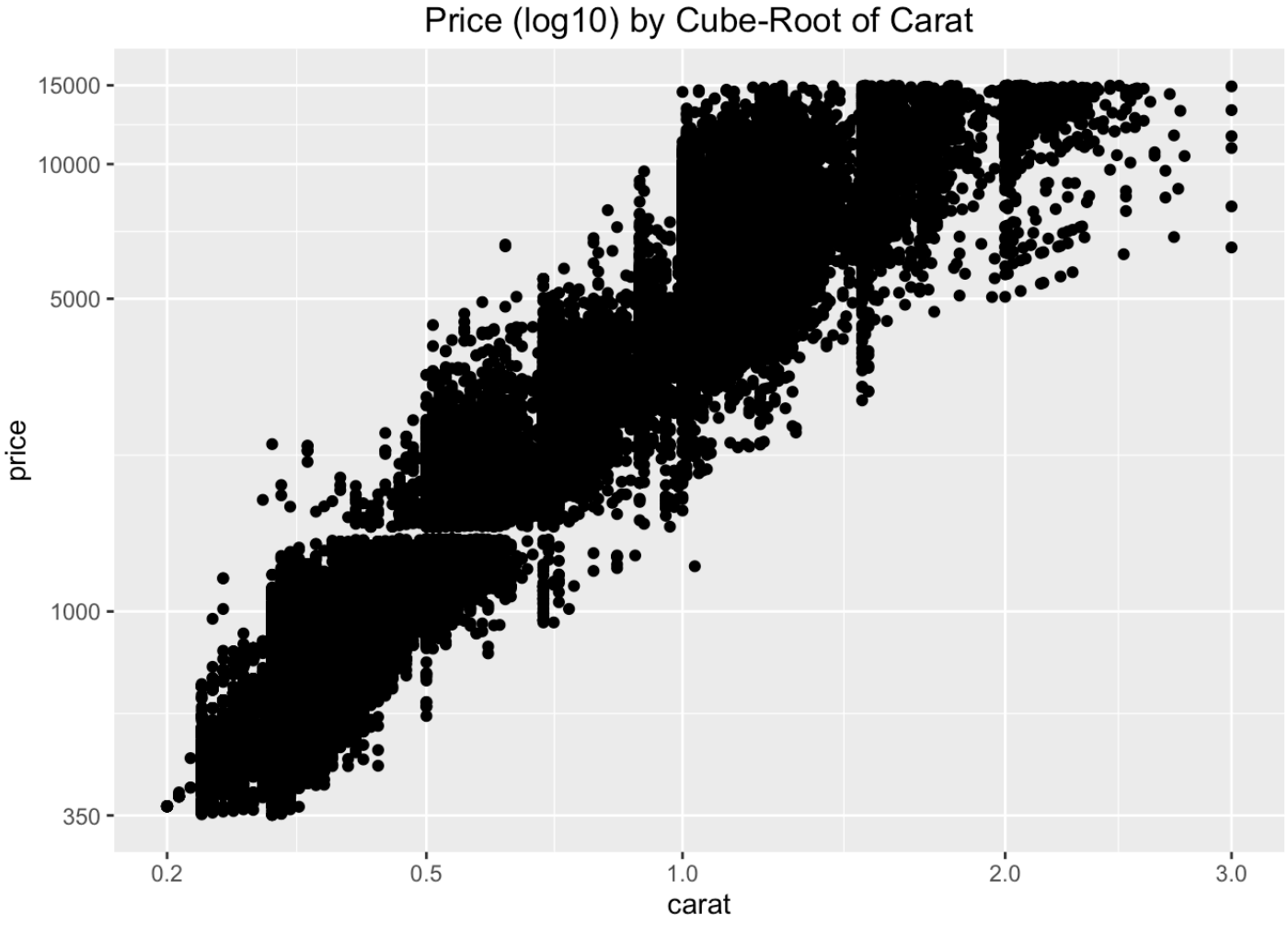
Create a new function to transform the carat variable

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

Use the cuberoot_trans function

```
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 = log10_trans(), limits = c(350, 15000),
                     breaks = c(350, 1000, 5000, 10000, 15000)) +
  ggtitle('Price (log10) by Cube-Root of Carat')
```

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



Overplotting Revisited

```
head(sort(table(diamonds$carat), decreasing = T))
```

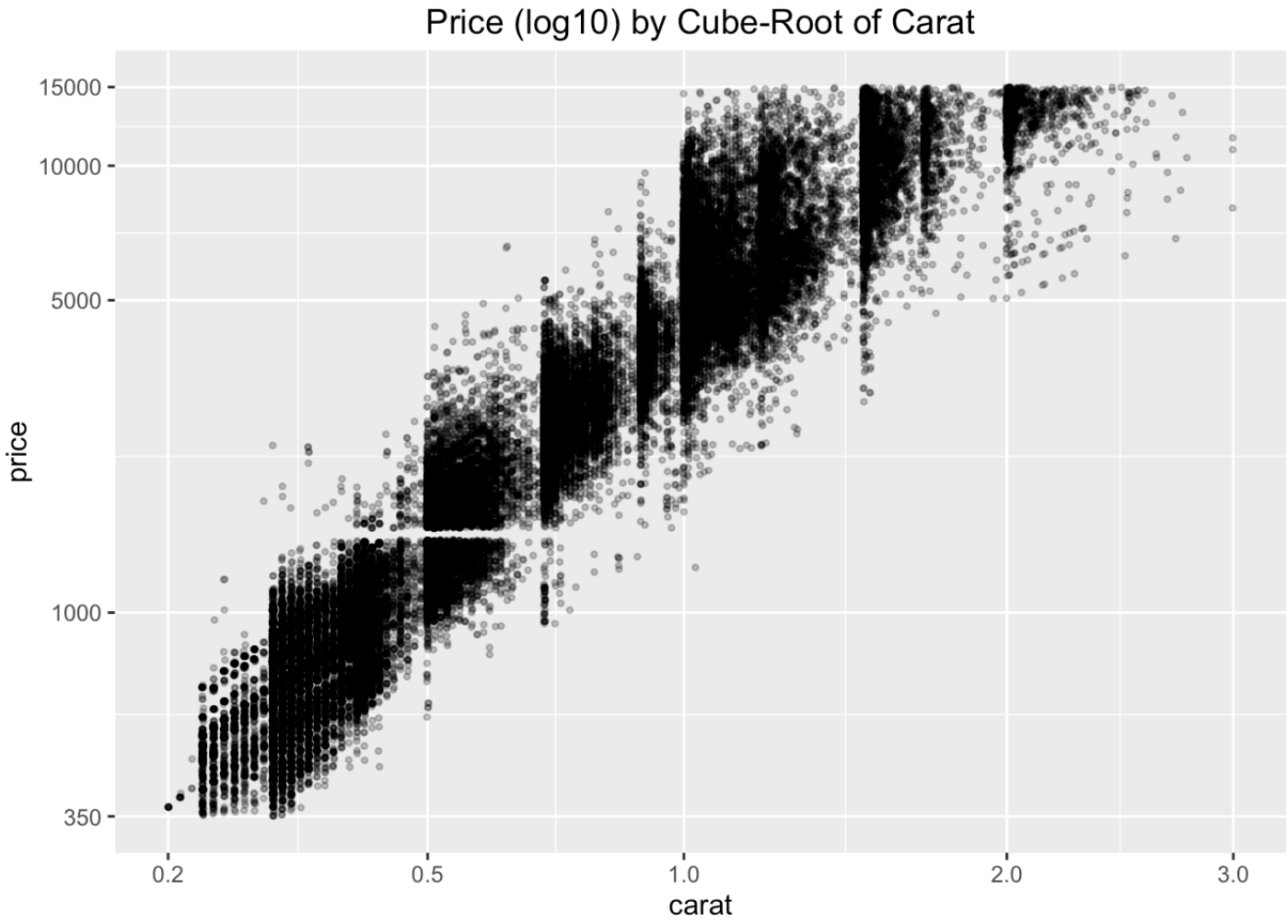
```
##
##  0.3 0.31 1.01  0.7 0.32    1
## 2604 2249 2242 1981 1840 1558
```

```
head(sort(table(diamonds$price), decreasing = T))
```

```
##
## 605 802 625 828 776 698
## 132 127 126 125 124 121
```

```
ggplot(aes(carat, price), data = diamonds) +
  geom_jitter(alpha = 5/20, 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 = log10_trans(), limits = c(350, 15000),
                     breaks = c(350, 1000, 5000, 10000, 15000)) +
  ggtitle('Price (log10) by Cube-Root of Carat')
```

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



Other Qualitative Factors

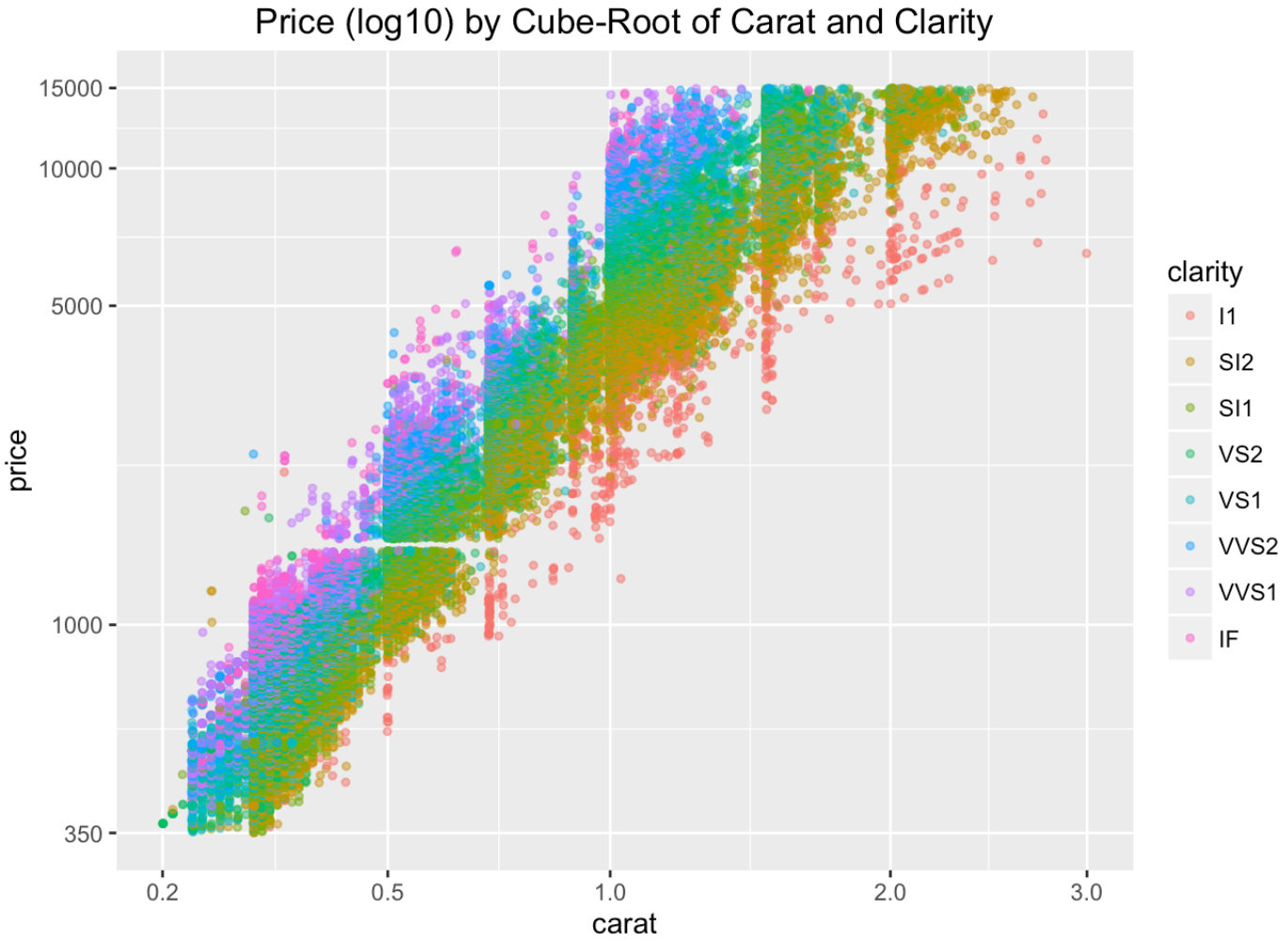
Notes: Use domain knowledge to understand features from consumers and makers. ***

Price vs. Carat and Clarity

Alter the code below.

```
# install and load the RColorBrewer package
# install.packages('RColorBrewer')
library(RColorBrewer)
?geom_point
ggplot(aes(x = carat, y = price), data = diamonds) +
  geom_point(aes(color = clarity), alpha = 0.5, size = 1, 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 = 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).
```



Clarity and Price

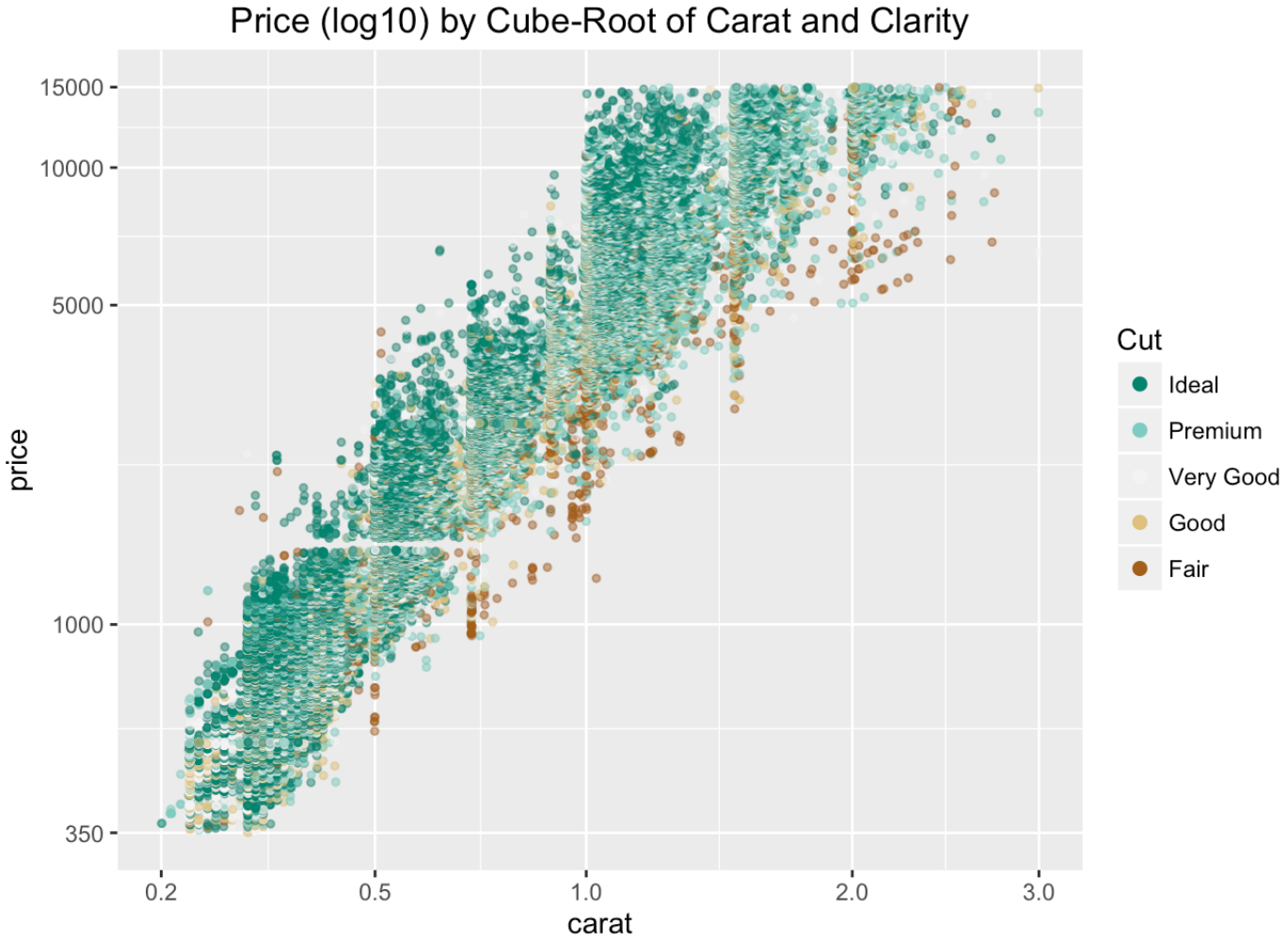
Response: Clarity has a positive relationship with price as the classes typically remain consistently stratified at the same relative price levels as the carats increase. ***

Price vs. Carat and Cut

Alter the code below.

```
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 1696 rows containing missing values (geom_point).
```



Cut and Price

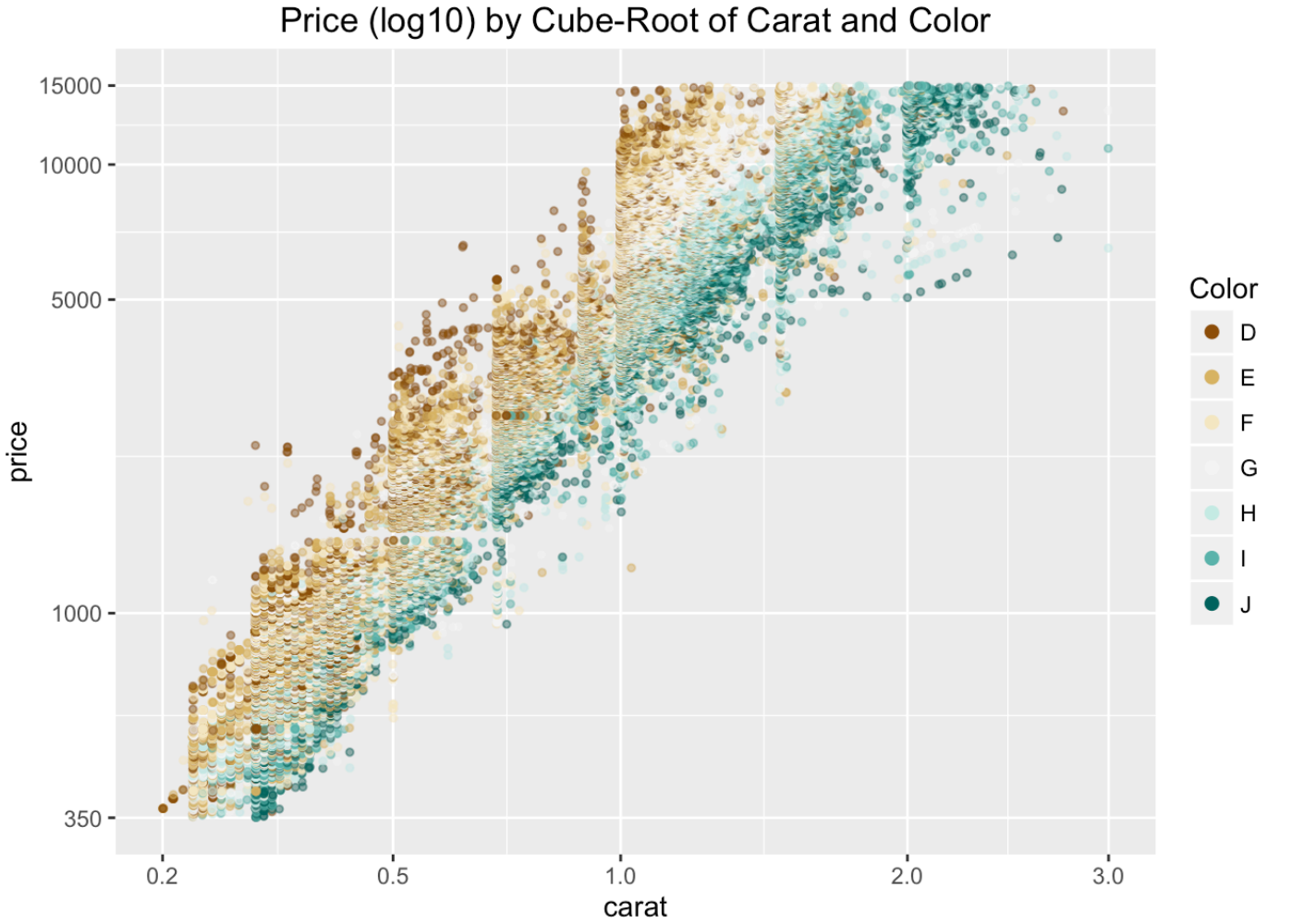
Response: Most of the data seems to be ideal/premium so there isn't much stratification diversity here as it looks outweighed by the sheer amount of higher end cuts.

Price vs. Carat and Color

Alter the code below.

```
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 = F,
                                          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')
```

```
## Warning: Removed 1688 rows containing missing values (geom_point).
```



Color and Price

Response: Yes, colour does seem to influece price, but it is also closely related to clarity, but doesn't seem semantically related. ***

Linear Models in R

Notes: predict with the lm() function

lm (y~x) outcome variable = y explanatory variable = x price ~ carat log(price) ~ carat^(1/3)

Response: log(price) ~ carat^(1/3)

Building the Linear Model

Notes:

```
# using the I wrapper ( I() ) variables, it is there to tell R to use the expressio
n before using the regression.
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            0.9        0.9        0.9        1.0        1.0
##   adj. R-squared       0.9        0.9        0.9        1.0        1.0
##   sigma                0.3        0.3        0.3        0.2        0.1
##   F               652012.1   387489.4   138654.5    87959.5   173791.1
##   p                    0.0        0.0        0.0        0.0        0.0
##   Log-likelihood   -7962.5    -3631.3    -1837.4     4235.2    34091.3
##   Deviance          4242.8     3613.4     3380.8     2699.2      892.2
##   AIC              15931.0     7270.6     3690.8    -8442.5   -68140.5
##   BIC              15957.7     7306.2     3762.0    -8317.9   -67953.7
##   N                53940      53940      53940      53940      53940
## ===========================================================================
```

Notice how adding cut to our model does not help explain much of the variance in the price of diamonds. This fits with our exploration earlier.

Model Problems

Video Notes:

Research: (Take some time to come up with 2-4 problems for the model) (You should 10-20 min on this)

Since we are including each of the features/vars, we may have some innerdependence that could overlap and be better outside the model. Finding the principal components might be key here. Using all the features may result in overfitting and only work on a very specific dataset.
We didn't include inflation rate or cultural trends that may be occuring because of this.

Response:

A Bigger, Better Data Set

Notes:

```
# install.packages('bitops')
# install.packages('RCurl')
library('bitops')
library('RCurl')

diamondsurl = '/Users/christianramsey/PycharmProjects/udacity/data\ analyst/P4\ Exp
lore\ and\ Summarize\ Data/EDA_Course_Materials/lesson6/BigDiamonds.Rda'
load((diamondsurl))
```

The code used to obtain the data is available here: https://github.com/solomonm/diamonds-data (https://github.com/solomonm/diamonds-data)

Building a Model Using the Big Diamonds Data Set

Notes:

```
diamondsbig$logprice = log(diamondsbig$price)
m1 <- lm(logprice ~ I(carat^(1/3)),
         data = diamondsbig[diamondsbig$price < 10000 & diamondsbig$cert == "GIA",]
)
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 = logprice ~ I(carat^(1/3)), data = diamondsbig[diamondsbig$price
<
##     10000 & diamondsbig$cert == "GIA", ])
## m2: lm(formula = logprice ~ I(carat^(1/3)) + carat, data = diamondsbig[diamondsb
ig$price <
```

```
##     10000 & diamondsbig$cert == "GIA", ])
## m3: lm(formula = logprice ~ I(carat^(1/3)) + carat + cut, data = diamondsbig[dia
mondsbig$price <
##     10000 & diamondsbig$cert == "GIA", ])
## m4: lm(formula = logprice ~ I(carat^(1/3)) + carat + cut + color,
##     data = diamondsbig[diamondsbig$price < 10000 & diamondsbig$cert ==
##         "GIA", ])
## m5: lm(formula = logprice ~ I(carat^(1/3)) + carat + cut + color +
##     clarity, data = diamondsbig[diamondsbig$price < 10000 & diamondsbig$cert ==
##     "GIA", ])
##
## ================================================================================
##                         m1          m2          m3          m4          m5
## --------------------------------------------------------------------------------
##   (Intercept)       2.671***    1.333***    0.949***    0.529***   -0.464***
##                    (0.003)     (0.012)     (0.012)     (0.010)     (0.009)
##   I(carat^(1/3))    5.839***    8.243***    8.633***    8.110***    8.320***
##                    (0.004)     (0.022)     (0.021)     (0.017)     (0.012)
##   carat                        -1.061***   -1.223***   -0.782***   -0.763***
##                                (0.009)     (0.009)     (0.007)     (0.005)
##   cut: V.Good                               0.120***    0.090***    0.071***
##                                            (0.002)     (0.001)     (0.001)
##   cut: Ideal                                0.211***    0.181***    0.131***
##                                            (0.002)     (0.001)     (0.001)
##   color: K/L                                            0.123***    0.117***
##                                                        (0.004)     (0.003)
##   color: J/L                                            0.312***    0.318***
##                                                        (0.003)     (0.002)
##   color: I/L                                            0.451***    0.469***
##                                                        (0.003)     (0.002)
##   color: H/L                                            0.569***    0.602***
##                                                        (0.003)     (0.002)
##   color: G/L                                            0.633***    0.665***
##                                                        (0.003)     (0.002)
##   color: F/L                                            0.687***    0.723***
##                                                        (0.003)     (0.002)
##   color: E/L                                            0.729***    0.756***
##                                                        (0.003)     (0.002)
##   color: D/L                                            0.812***    0.827***
##                                                        (0.003)     (0.002)
##   clarity: I1                                                       0.301***
##                                                                    (0.006)
##   clarity: SI2                                                      0.607***
##                                                                    (0.006)
##   clarity: SI1                                                      0.727***
##                                                                    (0.006)
##   clarity: VS2                                                      0.836***
##                                                                    (0.006)
##   clarity: VS1                                                      0.891***
##                                                                    (0.006)
##   clarity: VVS2                                                     0.935***
```

```
##                                                                      (0.006)
##   clarity: VVS1                                                       0.995***
##                                                                      (0.006)
##   clarity: IF                                                         1.052***
##                                                                      (0.006)
## ------------------------------------------------------------------------------
##   R-squared             0.9        0.9       0.9        0.9          1.0
##   adj. R-squared        0.9        0.9       0.9        0.9          1.0
##   sigma                 0.3        0.3       0.3        0.2          0.2
##   F               2700903.7  1406538.3  754405.4   423311.5     521161.4
##   p                     0.0        0.0       0.0        0.0          0.0
##   Log-likelihood   -60137.8   -53996.3  -43339.8    37830.4     154124.3
##   Deviance          28298.7    27291.5   25628.3    15874.9       7992.7
##   AIC              120281.6   108000.5   86691.6   -75632.8    -308204.5
##   BIC              120313.8   108043.5   86756.0   -75482.6    -307968.4
##   N                338946     338946    338946     338946       338946
## ==============================================================================
```

Predictions

we need to expontiate the result of our model since we took the log of price

Example Diamond from BlueNile: Round 1.00 Very Good I VS1 $5,601

```
#Be sure you've loaded the library memisc and have m5 saved as an object in your wo
rkspace.
thisDiamond = data.frame(carat = 1.00, cut = "V.Good",
                         color = "I", clarity="VS1")
modelEstimate = predict(m5, newdata = thisDiamond,
                        interval="prediction", level = .95)
```

Evaluate how well the model predicts the BlueNile diamond's price. Think about the fitted point estimate as well as the 95% CI.

Final Thoughts

Notes: