Lesson 6

Welcome

Notes:

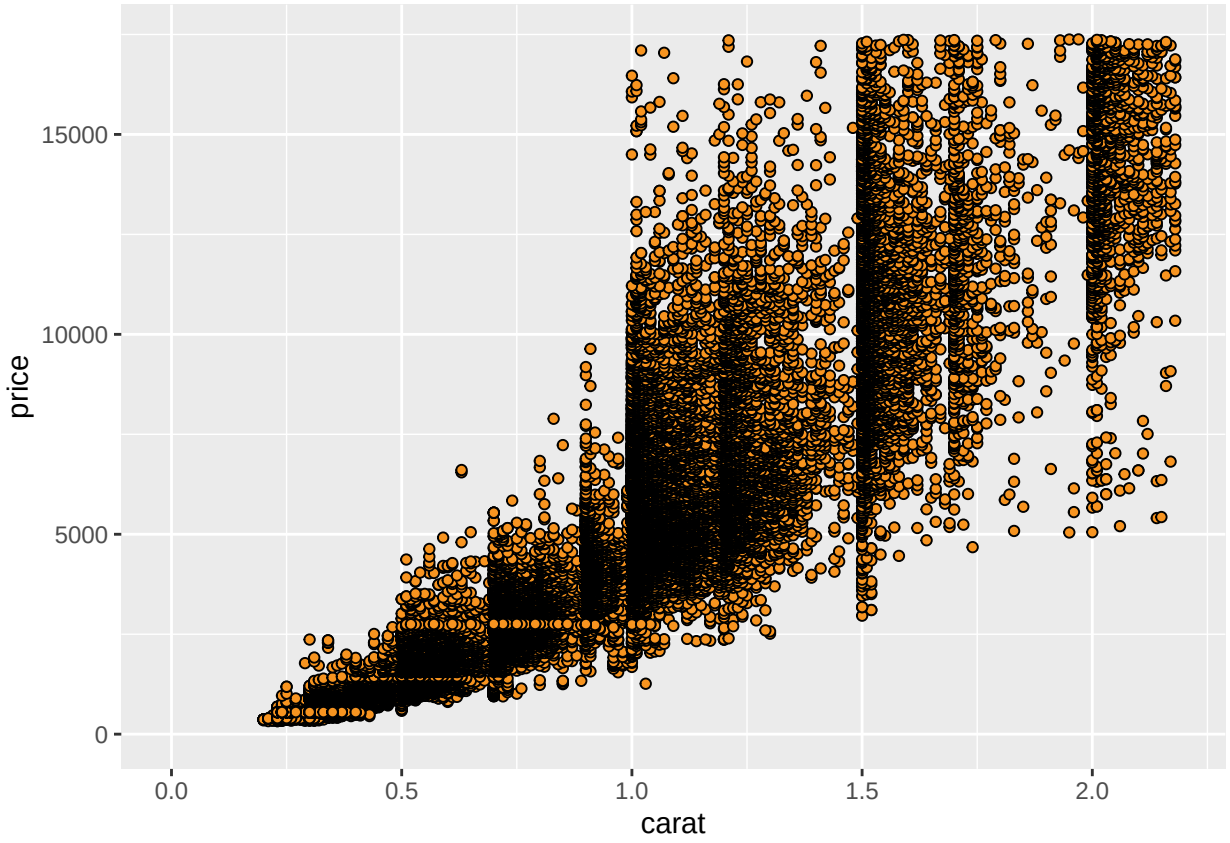
Scatterplot Review

```
library(ggplot2)
data(diamonds)
names(diamonds)
```

```
##  [1] "carat"   "cut"     "color"   "clarity" "depth"   "table"   "price"
##  [8] "x"       "y"       "z"
```

```
# Let's start by examining 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).

# Limit the x-axis and y-axis to omit the top 1% of values.

ggplot(aes(x = carat, y = price), data = diamonds) +
  geom_point(fill = I('#F79420'), color = I('black'), shape = 21) +
  xlim(0, quantile(diamonds$carat, 0.99)) +
  ylim(0, quantile(diamonds$price, 0.99))
```

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




```
ggsave('priceCarat.png')
```

```
## Saving 6.5 x 4.5 in image
```

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

Price and Carat Relationship

Response:

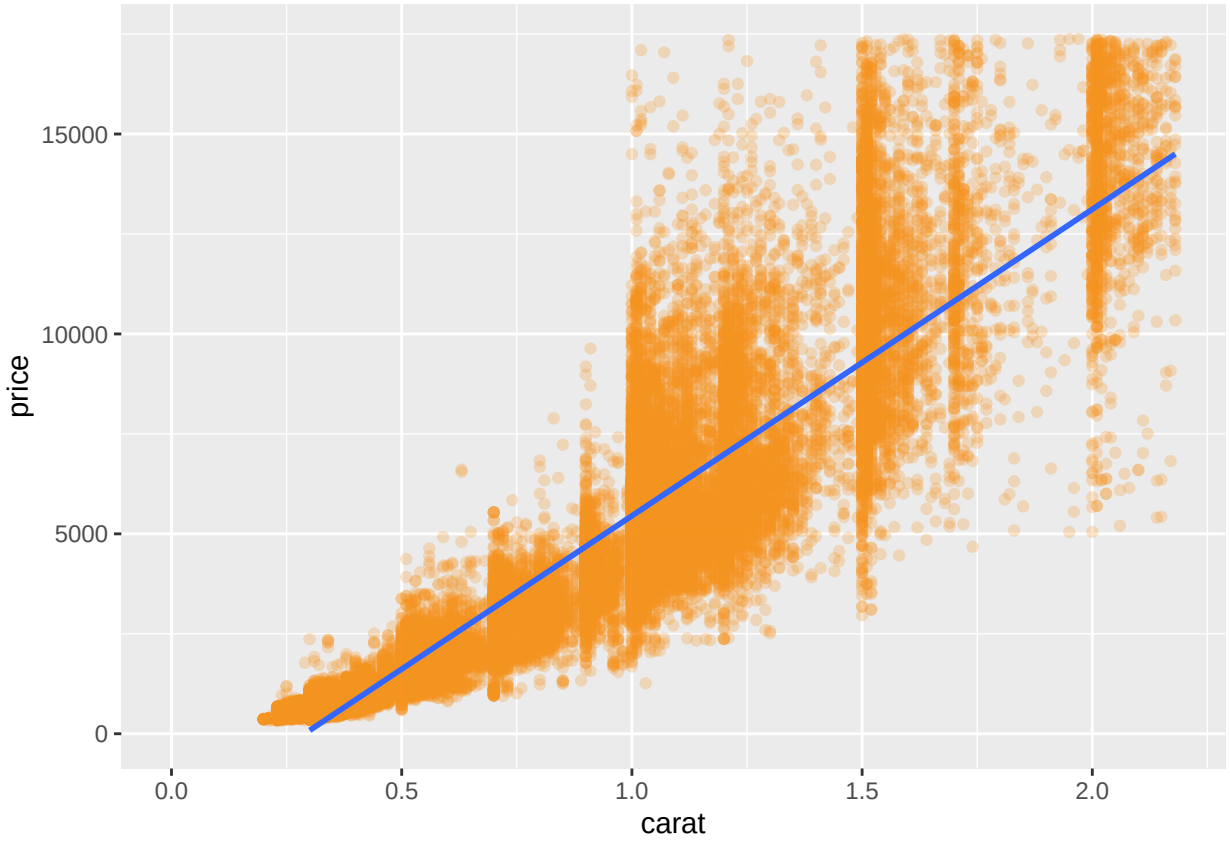
```
ggplot(aes(x = carat, y = price), data = diamonds) +
  geom_point(color = '#F79420', alpha = 1/4) +
  geom_smooth(method = 'lm') +
  scale_x_continuous(lim = c(0, quantile(diamonds$carat, 0.99))) +
  scale_y_continuous(lim = c(0, quantile(diamonds$price, 0.99)))
```

```
## `geom_smooth()` using formula 'y ~ x'
```

```
## 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).
```




```
ggsave('Linear.png')
```

```
## Saving 6.5 x 4.5 in image
## `geom_smooth()` using formula 'y ~ x'

## 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).
```

Frances Gerety

Notes:

A diamonds is

A diamonds is Forever.

The Rise of Diamonds

Notes:

---

ggpairs Function

Notes:

```
# install these if necessary
#install.packages('GGally')
#install.packages('scales')
#install.packages('memisc')
#install.packages('lattice')
#install.packages('MASS')
#install.packages('car')
#install.packages('reshape')
#install.packages('plyr')

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

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
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 object is masked from 'package:ggplot2':
##
##     syms
```

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

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

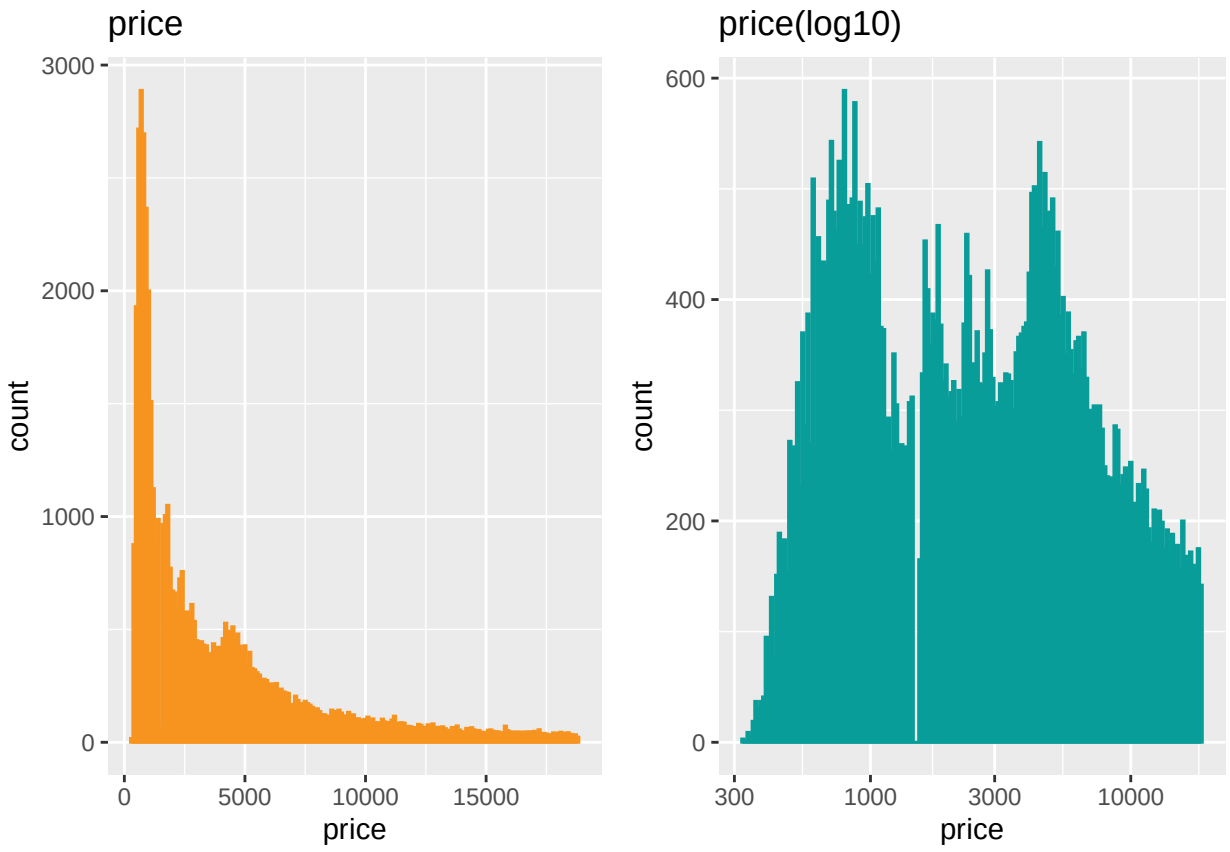
```
# sample 10,000 diamonds from the data set
set.seed(20022012)
diamond_samp = diamonds[sample(1:length(diamonds$price), 10000), ]

# it needs a long time.
#ggpairs(diamond_samp,
#        lower = list(continuous = wrap('points', shape = I('.'))),
#        upper = list(continuous = wrap('box', outlier.shape = I('.'))))
```

What are some things you notice in the ggpairs output? Response:

The Demand of Diamonds

Notes:

```
# Create two histograms of the price variable
# and place them side by side on one output image.

# We've put some code below to get you started.

# The first plot should be a histogram of price
# and the second plot should transform
# the price variable using log10.

# Set appropriate bin widths for each plot.
# ggtitle() will add a title to each histogram.

# You can self-assess your work with the plots
# in the solution video.
# ================================================== #
library(gridExtra)

plot1 = ggplot(aes(x = price), data = diamonds) +
  geom_histogram(binwidth = 100, color = '#F79420') +
  ggtitle('price')

plot2 = ggplot(aes(x = price), data = diamonds) +
  geom_histogram(binwidth = 0.01,  color = '#099D99') +
  scale_x_log10() +
  ggtitle('price(log10)')

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




```
ggsave('histPrice2.png')
```

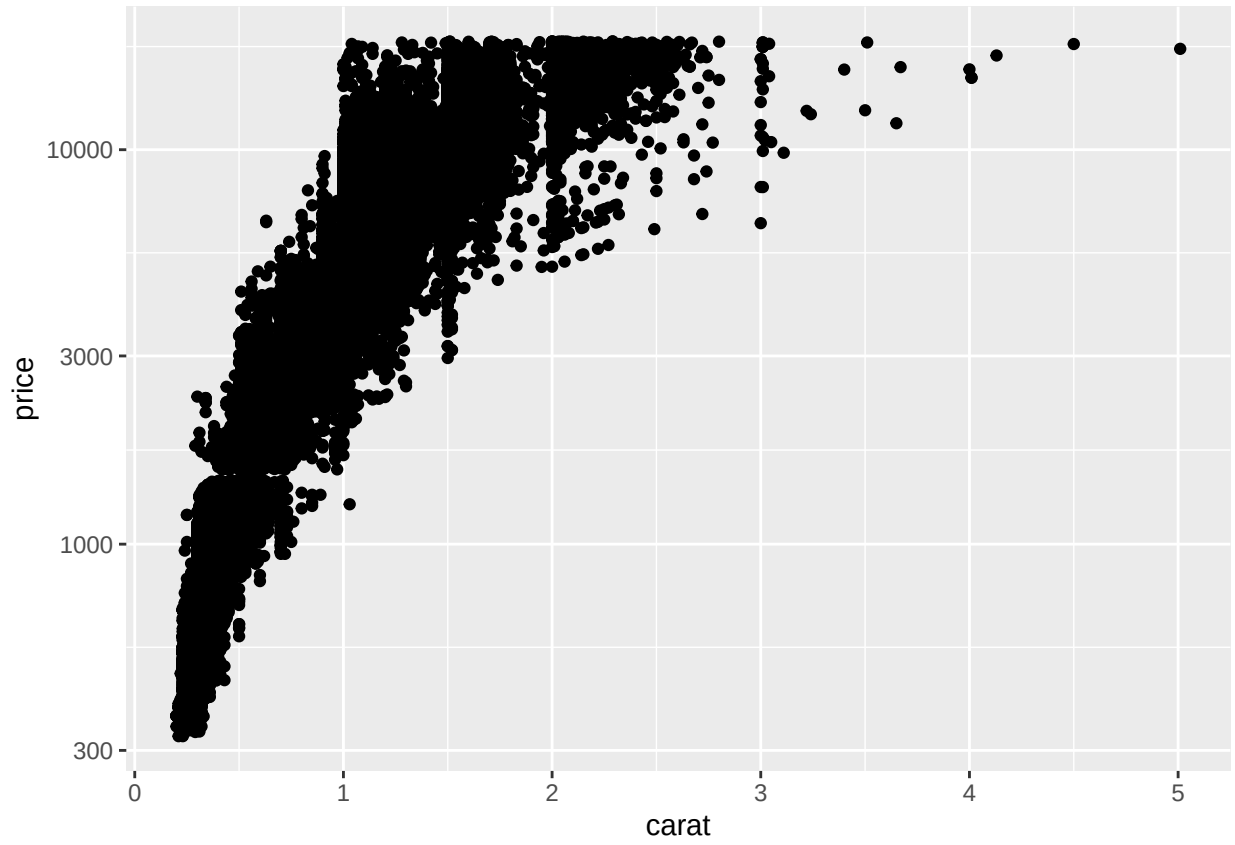
```
## Saving 6.5 x 4.5 in image
```

Connecting Demand and Price Distributions

Notes:

Scatterplot Transformation

```
ggplot(aes(x = carat, y = price), data = diamonds) +
  geom_point() +
  scale_y_log10()
```




```
  ggtitle('price(log10) by carat')
```

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

```
ggsave('priceCarat2.png')
```

```
## Saving 6.5 x 4.5 in image
```

Create a new function to transform the carat variable

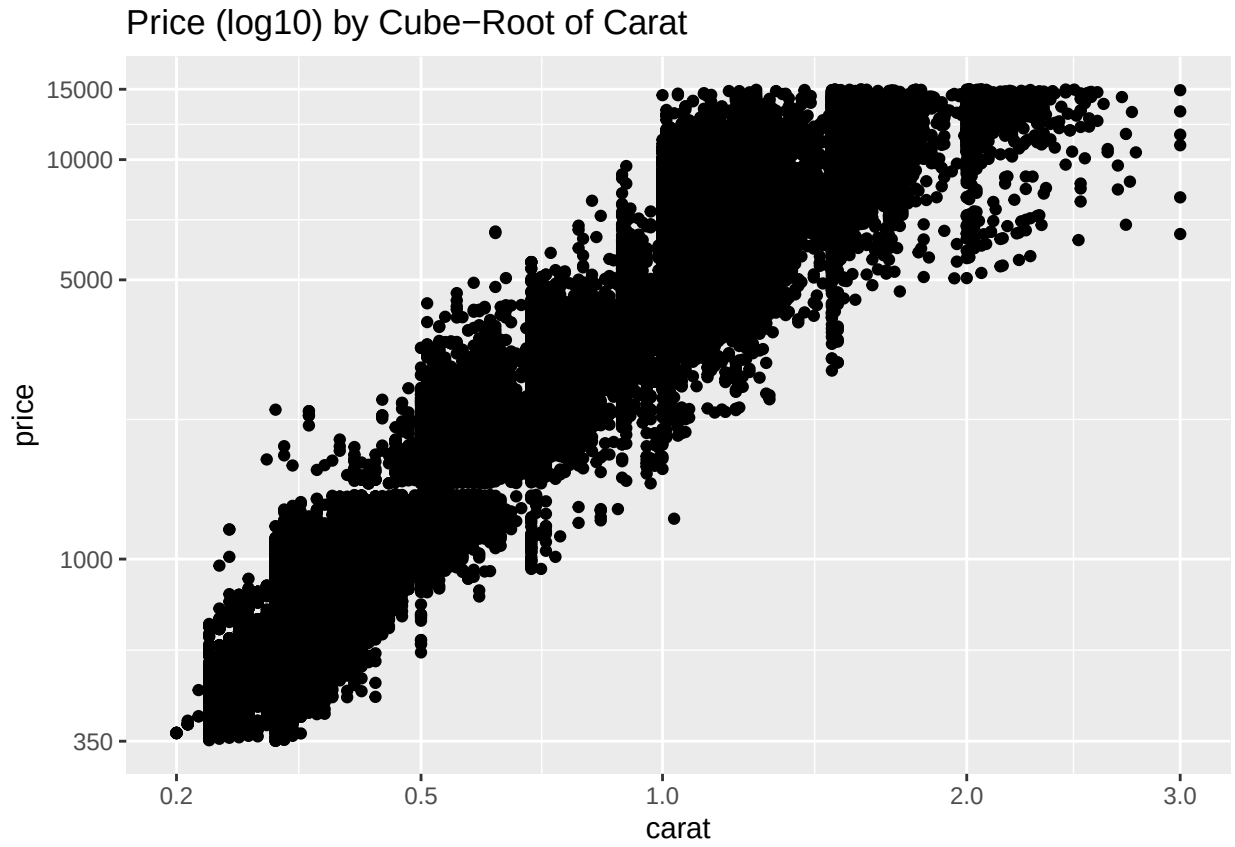
```
cuberoot_trans = function() trans_new('cuberoot', transform = function(x) x^(1/3),
                                      inverse = function(x) x^3)
```

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).
```




```
ggsave('scatterlog10.png')
```

```
## Saving 6.5 x 4.5 in image
```

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

Overplotting Revisited

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

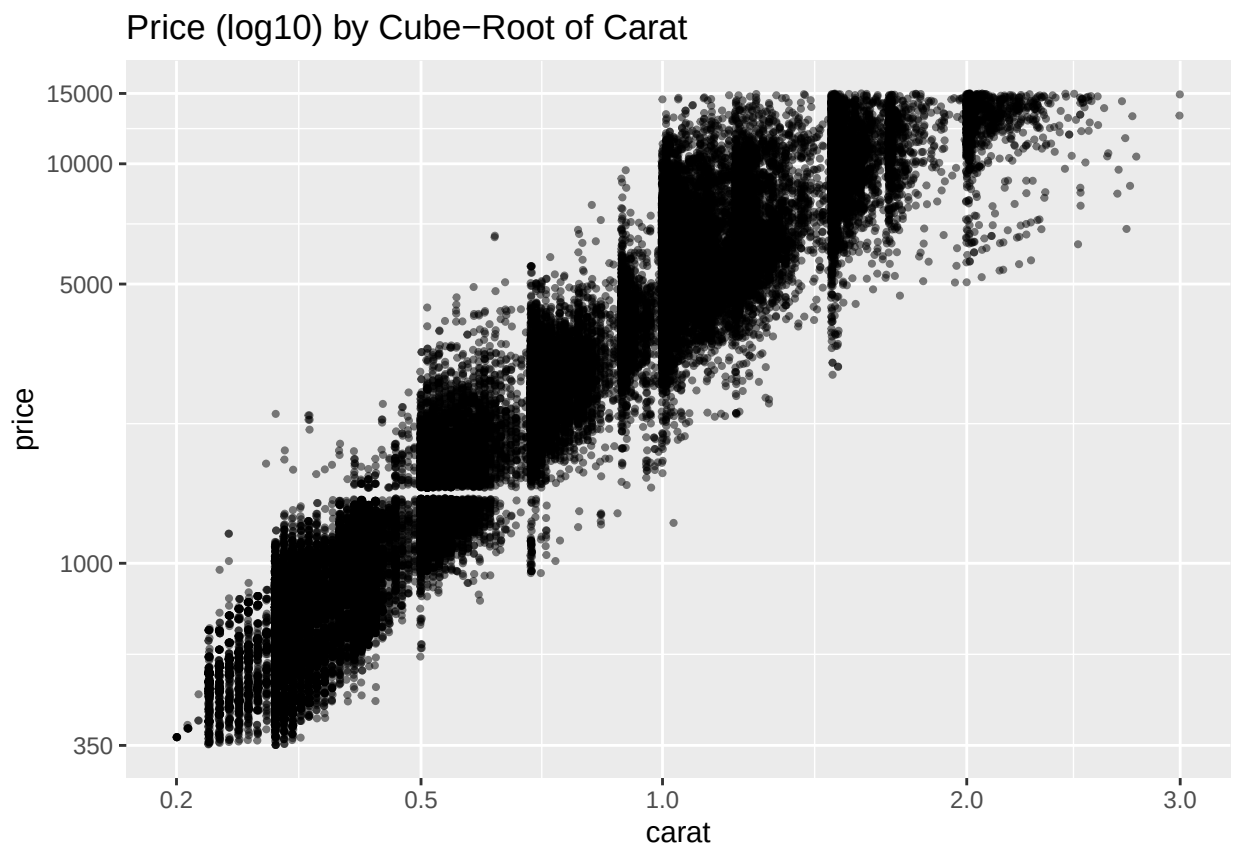
```
##
##  0.3 0.31 1.01  0.7 0.32    1
## 2604 2249 2242 1981 1840 1558
```

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

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

```r
# Add a layer to adjust the features of the
# scatterplot. Set the transparency to one half,
# the size to three-fourths, and jitter the points.

# If you need hints, see the Instructor Notes.
# There are three hints so scroll down slowly if
# you don't want all the hints at once.
# ================================================ #
ggplot(aes(carat, price), data = diamonds) +
  geom_jitter(alpha = 1/2, size = 3/4) +
  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).
```



```
ggsave('scatterxy.png')
```

```
## Saving 6.5 x 4.5 in image
```

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

Other Qualitative Factors

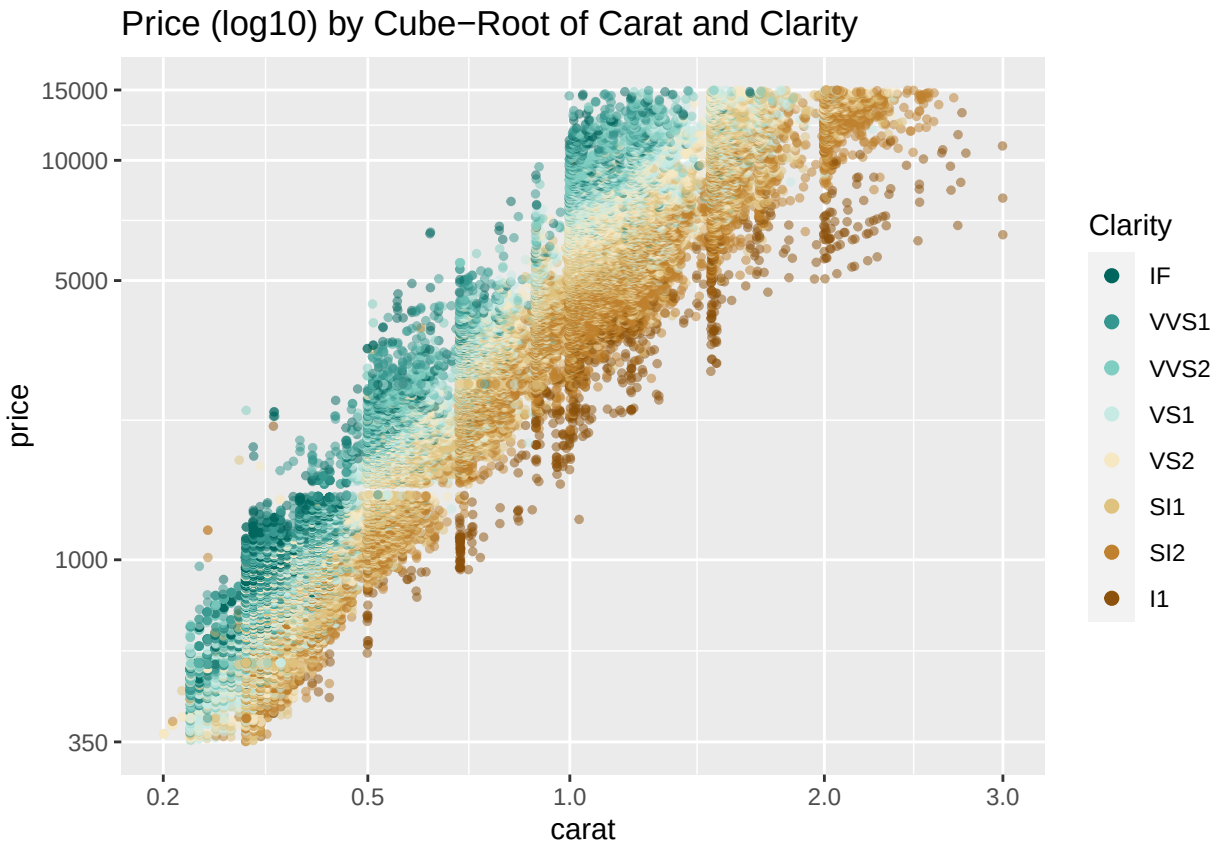
Notes:

Price vs. Carat and Clarity

Alter the code below.

```
# install and load the RColorBrewer package
# install.packages('RColorBrewer')

# ========================================================== #
# Adjust the code below to color the points by clarity.

# A layer called scale_color_brewer() has
# been added to adjust the legend and
# provide custom colors.

# See if you can figure out what it does.
# Links to resources are in the Instructor Notes.

# You will need to install the package RColorBrewer
# in R to get the same colors and color palettes.
# ========================================================== #
library(ggplot2)
library(RColorBrewer)

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

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')
```

```
## Warning: Removed 1690 rows containing missing values (geom_point).
```




```
ggsave('caratClarity.png')
```

```
## Saving 6.5 x 4.5 in image
```

```
## Warning: Removed 1690 rows containing missing values (geom_point).
```

Clarity and Price

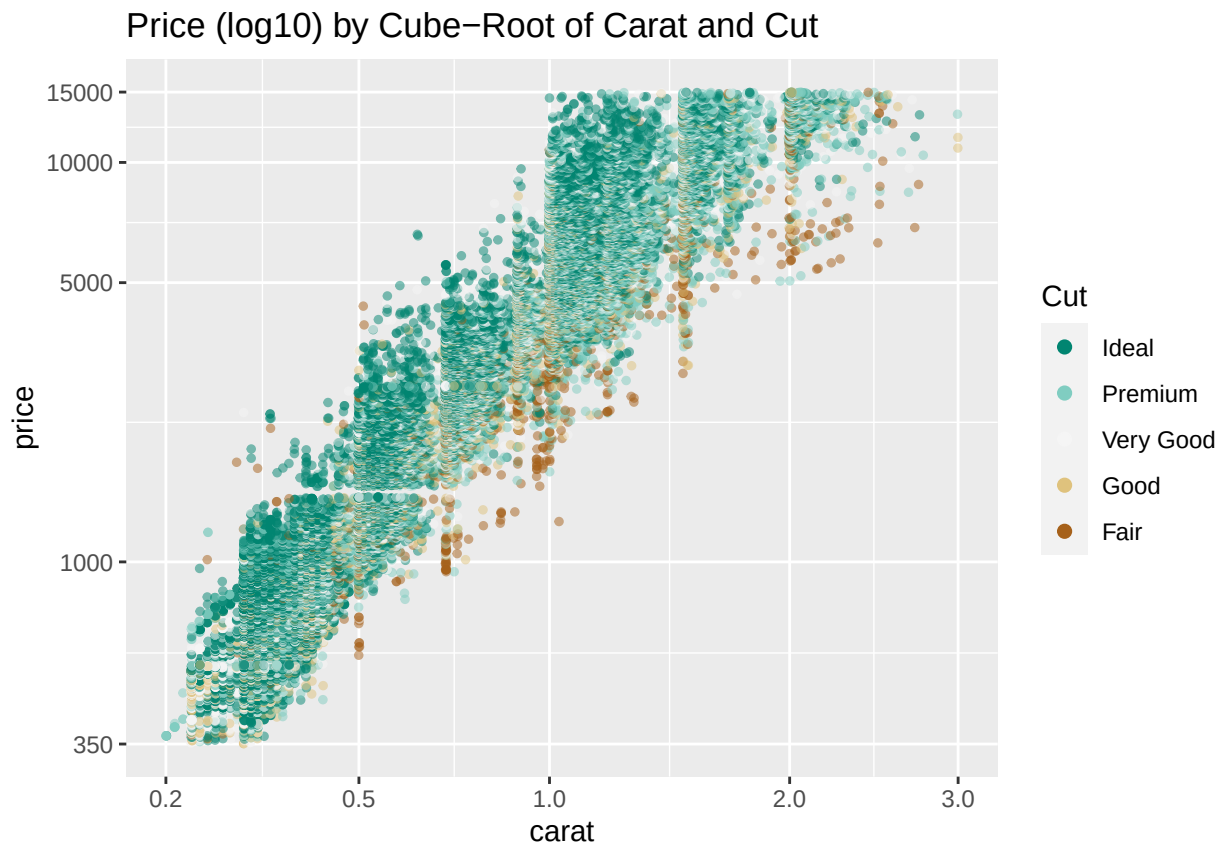
Response: Better clarity, more cheaper.

Price vs. Carat and Cut

Alter the code below.

```
# Let's look at cut and see if we find a similar result.

# Adjust the code below to color the points by cut.
# Change any other parts of the code as needed.
# ================================================== #

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 Cut')
```

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




```
ggsave('caratCut.png')
```

```
## Saving 6.5 x 4.5 in image
```

```
## Warning: Removed 1690 rows containing missing values (geom_point).
```

Cut and Price

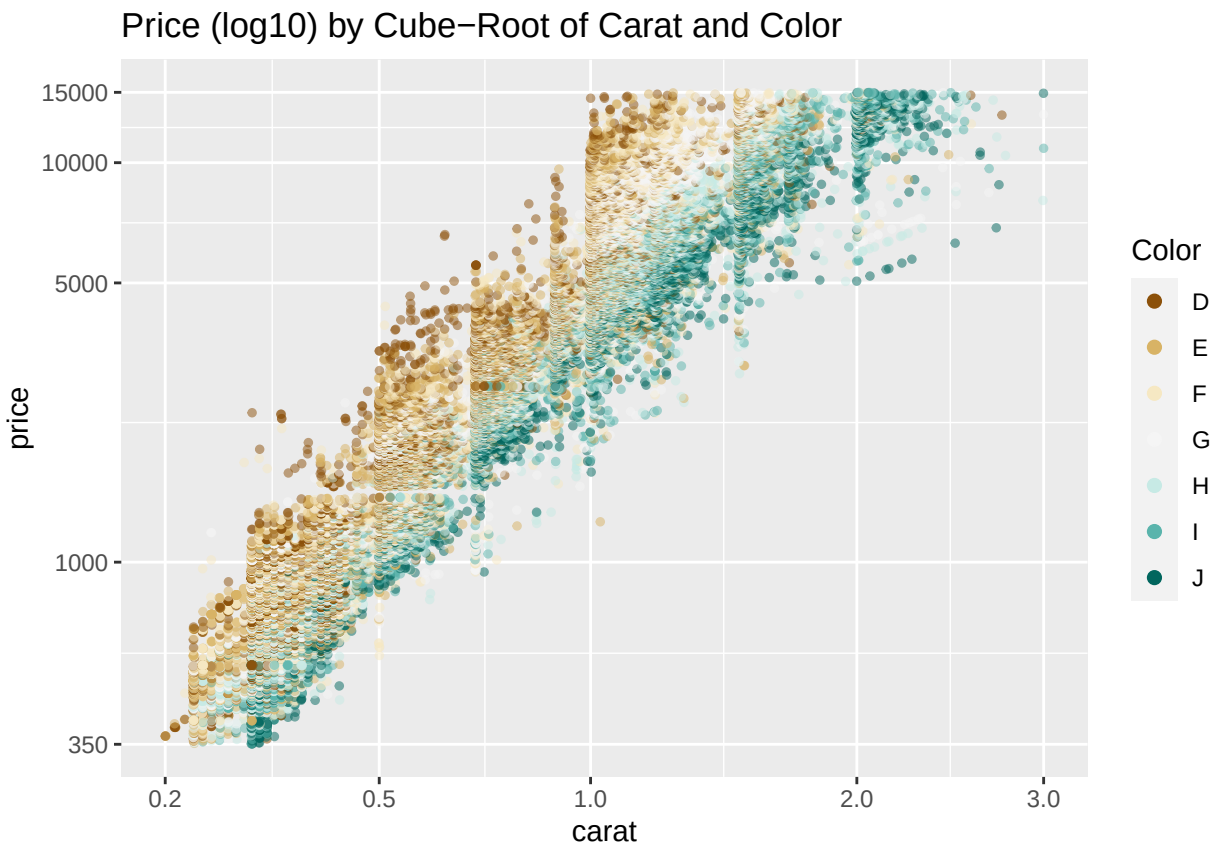
Response:

Price vs. Carat and Color

Alter the code below.

```
# Finally, let's use diamond color to color our plot.

# Adjust the code below to color the points by diamond colors
# and change the titles.
# ================================================================ #

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')
```

```
## Warning: Removed 1690 rows containing missing values (geom_point).
```




```
ggsave('caratColor.png')
```

```
## Saving 6.5 x 4.5 in image
```

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

Color and Price

Response:

Linear Models in R

Notes:

Response:

Building the Linear Model

Notes:

```
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, sdigits = 3)
```

```
##
## 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
##   N               53940         53940         53940         53940         53940
## ==========================================================================================
##   Significance: *** = p < 0.001; ** = p < 0.01; * = p < 0.05
```

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

Response:

A Bigger, Better Data Set

Notes:

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

library('bitops')
library('RCurl')

load("BigDiamonds.rda")
#diamondsurl = getBinaryURL("https://raw.github.com/solomonm/diamonds-data/master/BigDiamonds.Rda")
#load(rawConnection(diamondsurl))
```

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

Building a Model Using the Big Diamonds Data Set

Notes:

```
# Your task is to build five linear models like Solomon
# did for the diamonds data set only this
# time you'll use a sample of diamonds from the
# diamondsbig data set.

# Be sure to make use of the same variables
# (logprice, carat, etc.) and model
# names (m1, m2, m3, m4, m5).

# To get the diamondsbig data into RStudio
# on your machine, copy, paste, and run the
# code in the Instructor Notes. There's
# 598,024 diamonds in this data set!

# Since the data set is so large,
# you are going to use a sample of the
# data set to compute the models. You can use
# the entire data set on your machine which
# will produce slightly different coefficients
# and statistics for the models.

# This exercise WILL BE automatically graded.

# You can leave off the code to load in the data.
# We've sampled the data for you.
# You also don't need code to create the table output of the models.
# We'll do that for you and check your model summaries (R^2 values, AIC, etc.)

# Your task is to write the code to create the models.
#============================================================================== #
diamondsbig$logprice = log(diamondsbig$price)

m1 = lm(I(log(price)) ~ 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, sdigits = 3)
```

```
##
## Calls:
## m1: lm(formula = I(log(price)) ~ I(carat^(1/3)), data = diamondsbig[diamondsbig$price <
##     10000 & diamondsbig$cert == "GIA", ])
## m2: lm(formula = I(log(price)) ~ I(carat^(1/3)) + carat, data = diamondsbig[diamondsbig$price <
##     10000 & diamondsbig$cert == "GIA", ])
## m3: lm(formula = I(log(price)) ~ I(carat^(1/3)) + carat + cut, data = diamondsbig[diamondsbig$price
##     10000 & diamondsbig$cert == "GIA", ])
## m4: lm(formula = I(log(price)) ~ I(carat^(1/3)) + carat + cut + color,
##     data = diamondsbig[diamondsbig$price < 10000 & diamondsbig$cert ==
##         "GIA", ])
## m5: lm(formula = I(log(price)) ~ 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             0.888          0.892          0.899          0.937          0.969
##   N                338946         338946         338946         338946         338946
## ================================================================================================
##   Significance: *** = p < 0.001; ** = p < 0.01; * = p < 0.05
```

Predictions

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 workspace.
thisDiamond = data.frame(carat = 1.00, cut = "V.Good",
                         color = "I", clarity="VS1")
modelEstimate = predict(m5, newdata = thisDiamond,
                        interval="prediction", level = .95)

exp(modelEstimate)
```

```
##        fit     lwr      upr
## 1 5040.436 3730.34 6810.638
```

Evaluate how well the model predicts the BlueNile diamond's price. Think about the fitted point estimate as well as the 95% CI. Response: since lower is 8.224255, and upper is 8.826241, as well as fit = 8.525248, which locates in upper and lower. It suggests that the model prediction is good.

Final Thoughts

Notes:

Click **KnitHTML** to see all of your hard work and to have an html page of this lesson, your answers, and your notes!