HW03p

Evangeline Szpylka

April 13, 2018

```
knitr::opts_chunk$set(error = TRUE) #this allows errors to be printed into the PDF
```

1. Load package `ggplot2` below using `pacman`.

```
pacman::p_load(ggplot2)
rm(list = ls())
```

The dataset `diamonds` is in the namespace now as it was loaded with the `ggplot2` package. Run the following code and write about the dataset below.

```
?diamonds
```

```
## starting httpd help server ... done
```

```
str(diamonds)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    53940 obs. of  10 variables:
##  $ carat  : num  0.23 0.21 0.23 0.29 0.31 0.24 0.24 0.26 0.22 0.23 ...
##  $ cut    : Ord.factor w/ 5 levels "Fair"<"Good"<..: 5 4 2 4 2 3 3 3 1 3 ...
##  $ color  : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 2 2 6 7 7 6 5 2 5 ...
##  $ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 3 5 4 2 6 7 3 4 5 ...
##  $ depth  : num  61.5 59.8 56.9 62.4 63.3 62.8 62.3 61.9 65.1 59.4 ...
##  $ table  : num  55 61 65 58 58 57 57 55 61 61 ...
##  $ price  : int  326 326 327 334 335 336 336 337 337 338 ...
##  $ x      : num  3.95 3.89 4.05 4.2 4.34 3.94 3.95 4.07 3.87 4 ...
##  $ y      : num  3.98 3.84 4.07 4.23 4.35 3.96 3.98 4.11 3.78 4.05 ...
##  $ z      : num  2.43 2.31 2.31 2.63 2.75 2.48 2.47 2.53 2.49 2.39 ...
```

```
diamonds$cut = factor(as.character(diamonds$cut))
diamonds$color = factor(as.character(diamonds$color))
diamonds$clarity = factor(as.character(diamonds$clarity))
```

The "diamonds" dataset has 10 categories, which describe price, the 4 "c"'s of diamonds, length, width, and depth of the diamond, the percentage of depth relative to the average of the length and width, and width of diamond relative to its widest point. Cut, color, and clarity are measured with levels, price is measured as an integer, and the other categories appear to be continuously measured.

What is n, p, what do the features mean, what is the most likely response metric and why?

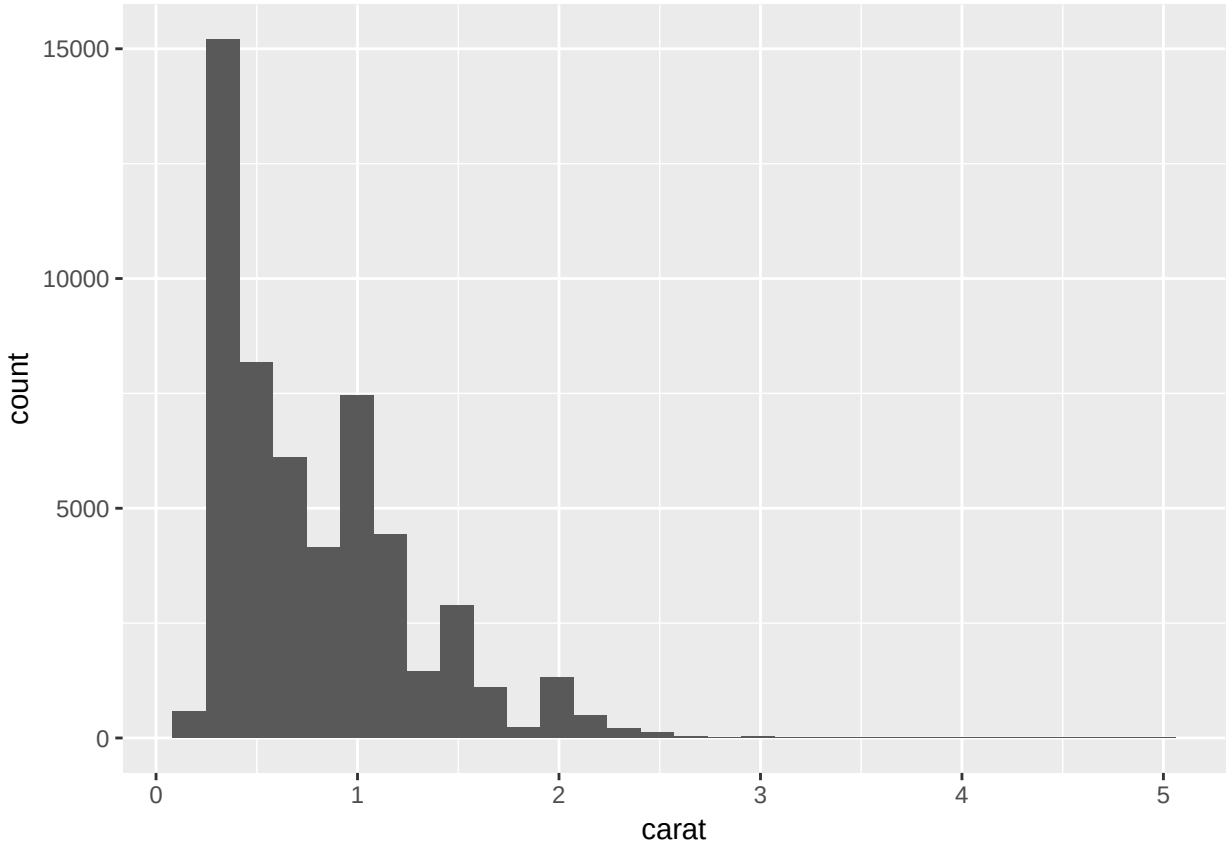
Here, $n = 53{,}940$, which is the number of observed diamonds and $p = 10$, which are the number of features being observed per diamond. The features are all measurable qualities of diamonds, such as the carat weight, the length, width and depth of each diamond, the cut and color and clarity, and depth and table proportions. The most likely response metric is price because the other 9 properties will allow us to predict how much a diamond will be worth, given certain features.

Regardless of what you wrote above, the variable `price` will be the response variable going forward.

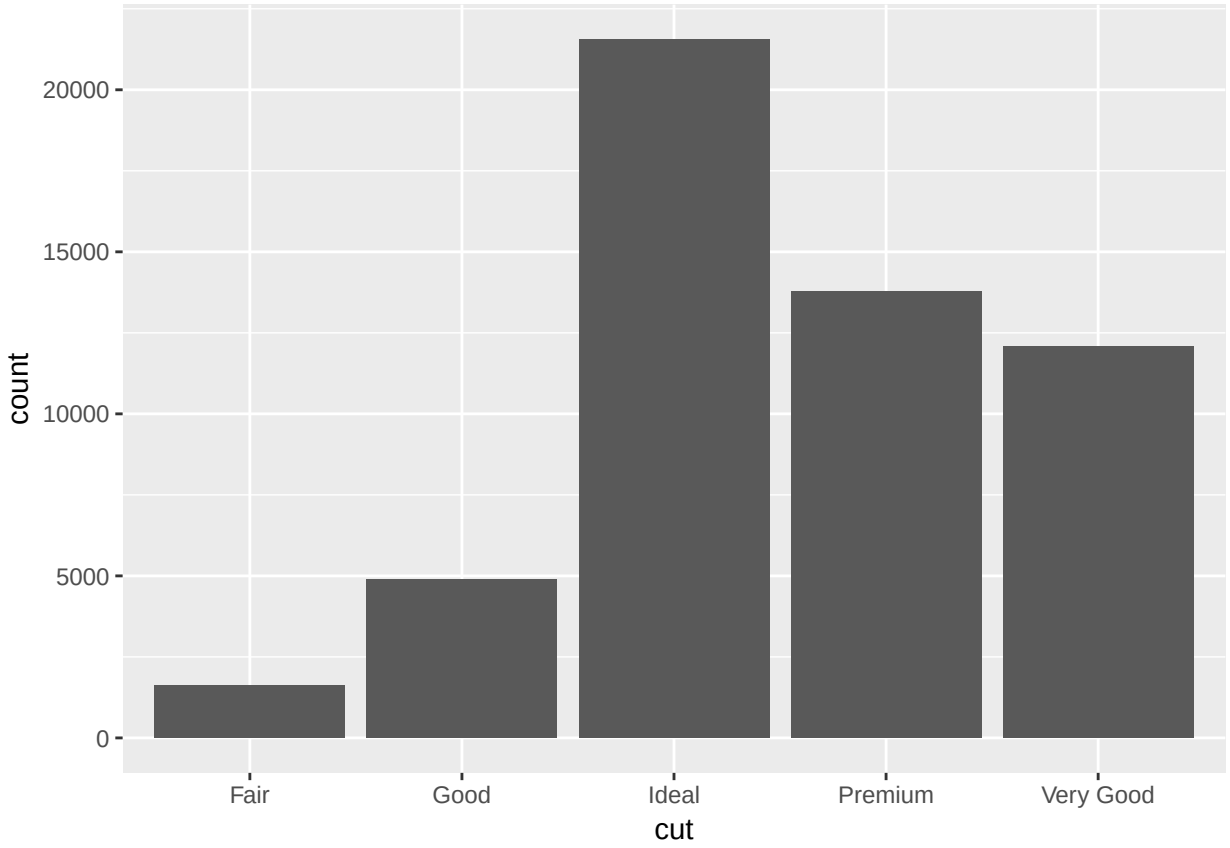
Use `ggplot` to look at the univariate distributions of *all* predictors. Make sure you handle categorical predictors differently from continuous predictors.

```
ggplot(diamonds, aes(carat)) +
  geom_histogram() #cont
```

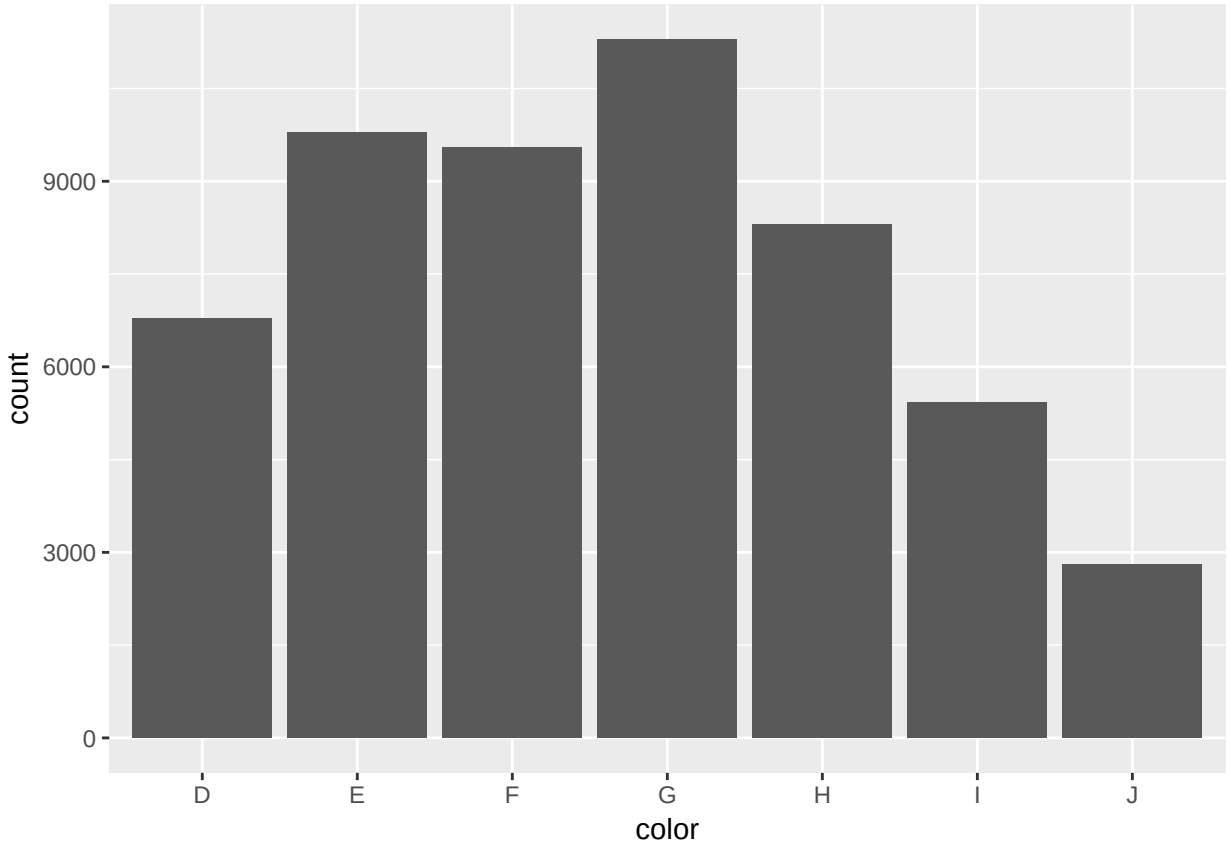
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



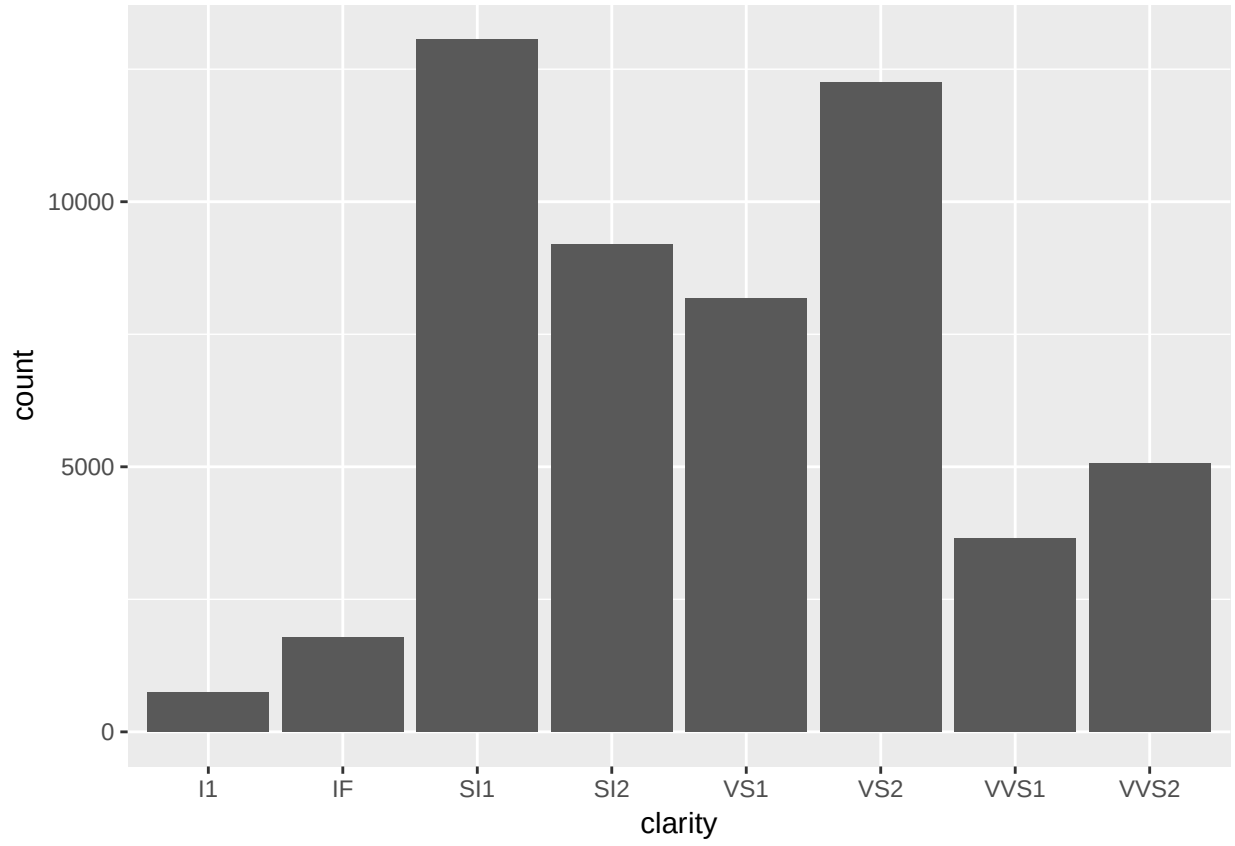

```
ggplot(diamonds, aes(cut)) +
  geom_bar() #ordinal
```




```
ggplot(diamonds, aes(color)) +
  geom_bar() #ordinal
```

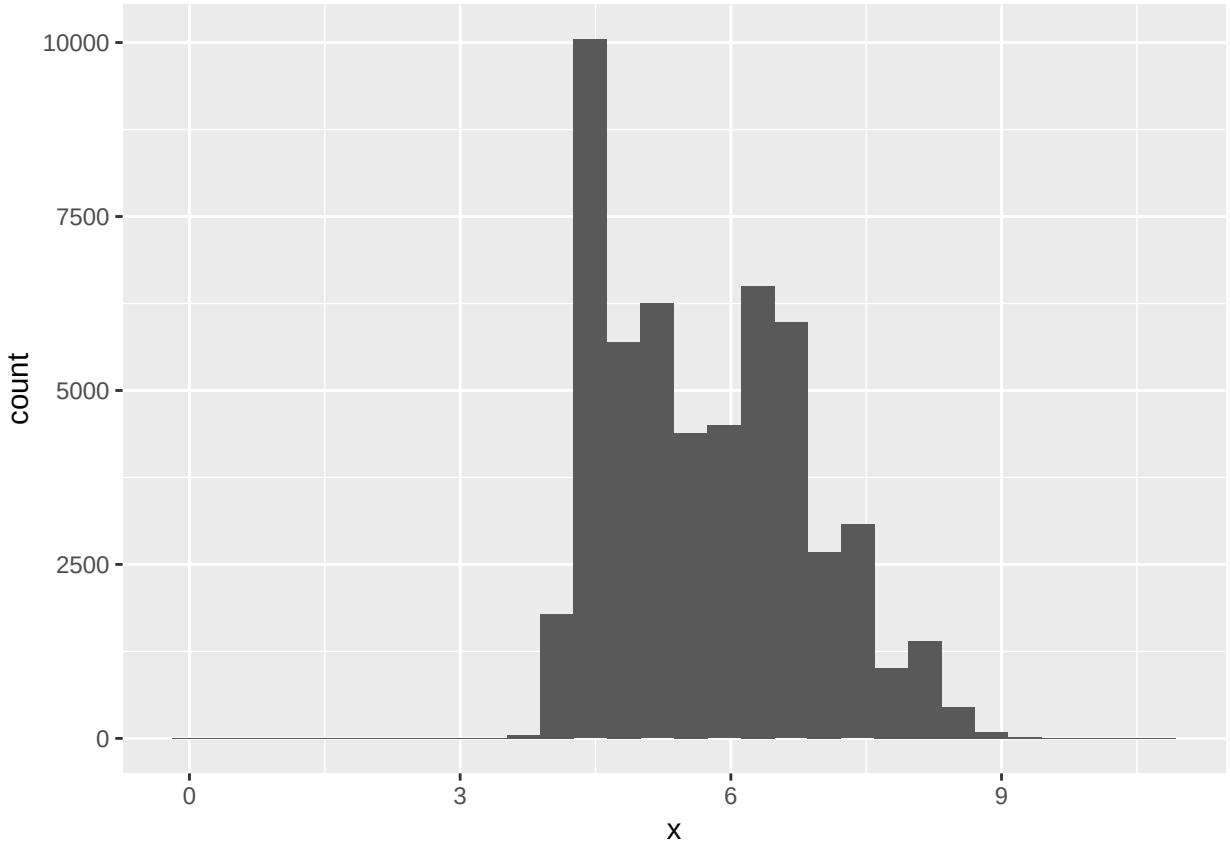



```
ggplot(diamonds, aes(clarity)) +
  geom_bar() #ordinal
```



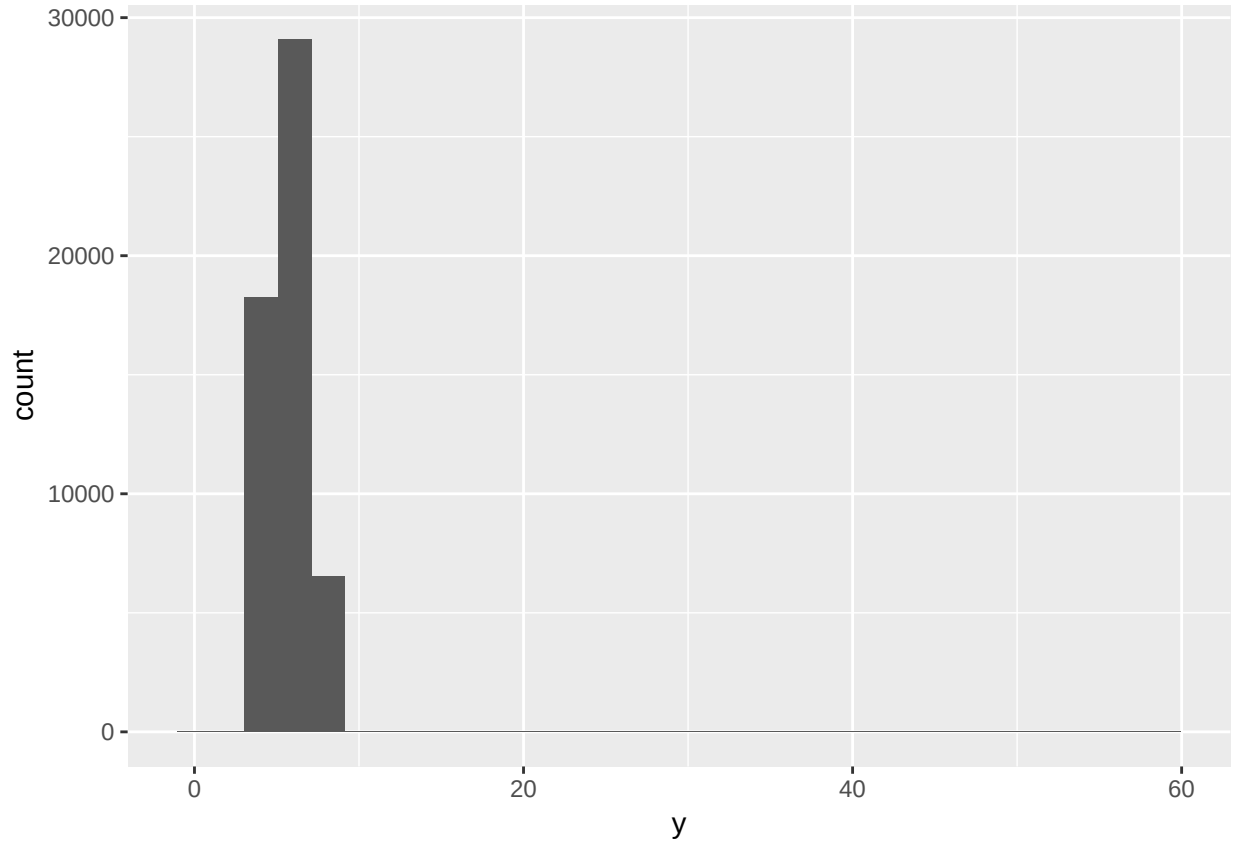

```
ggplot(diamonds, aes(x)) +
  geom_histogram() #cont
```

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



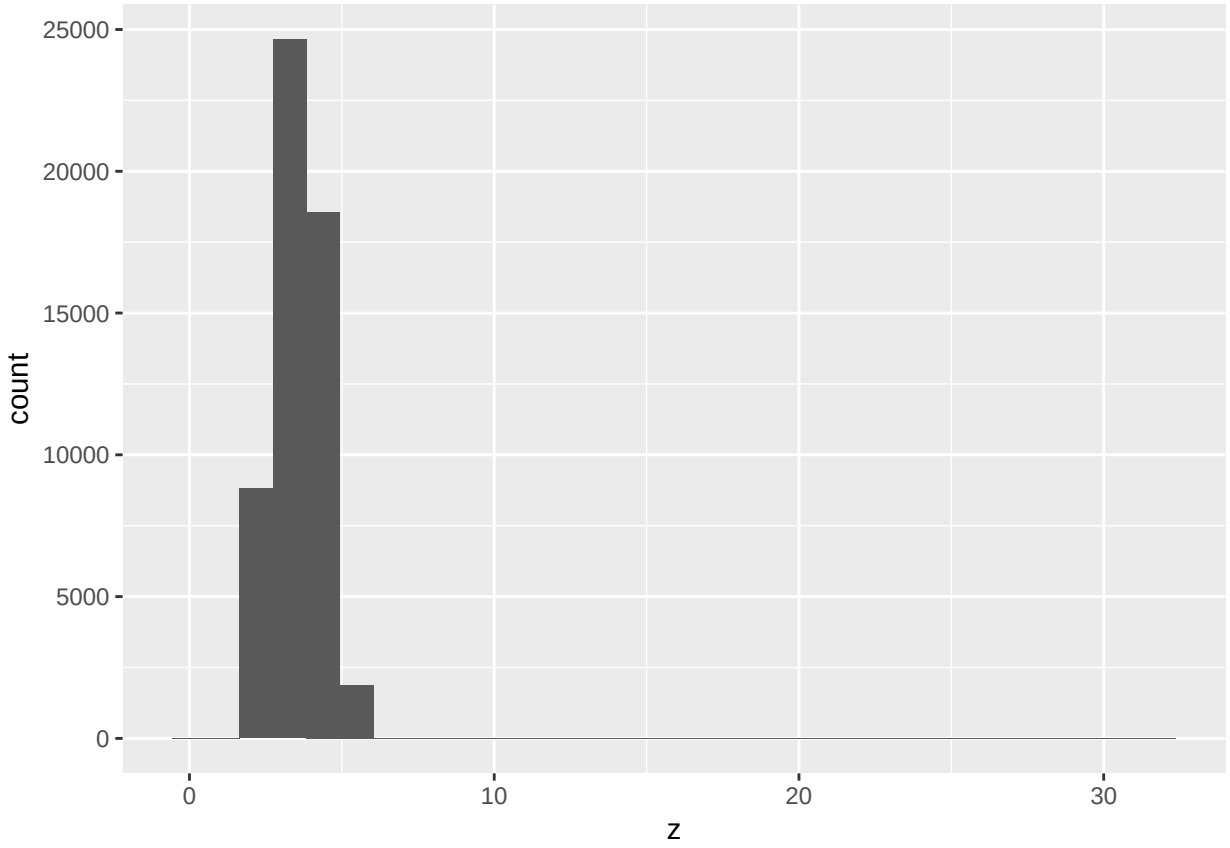

```
ggplot(diamonds, aes(y)) +
  geom_histogram() #cont
```

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



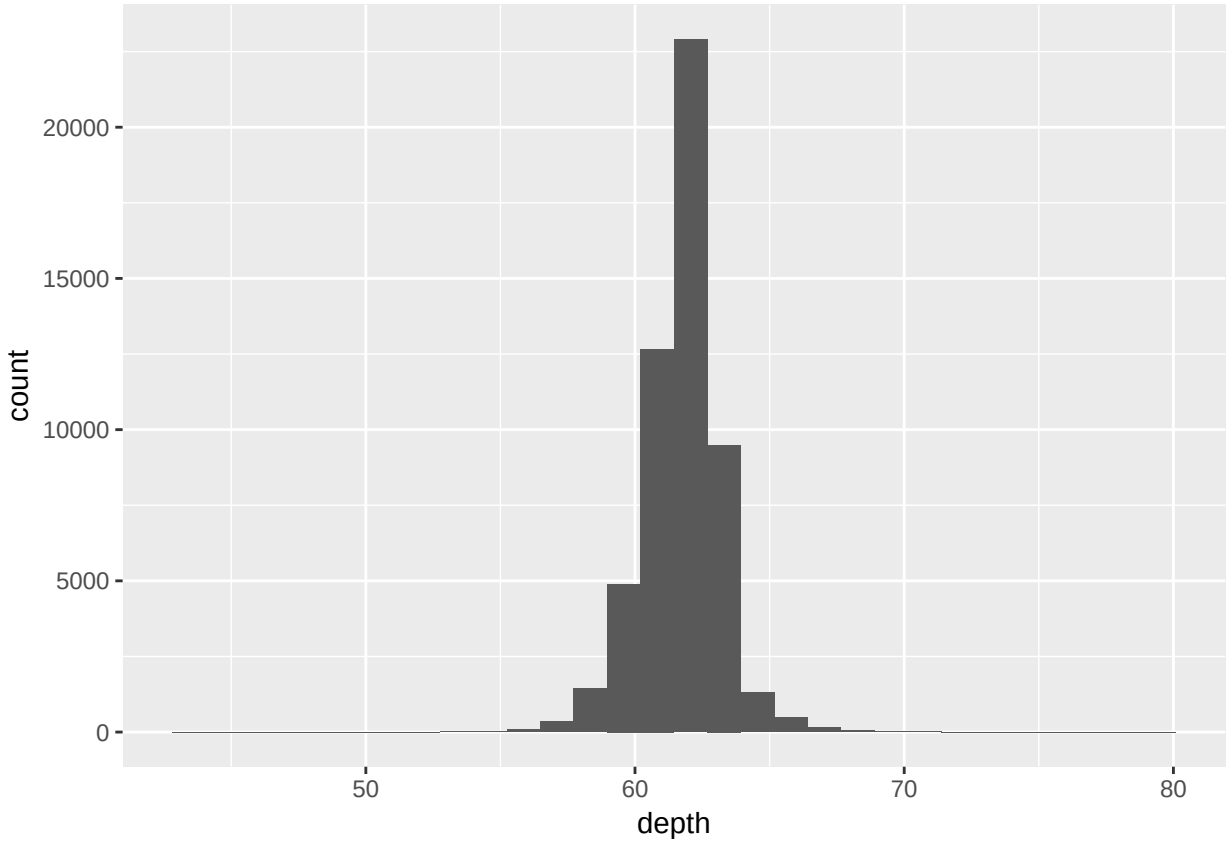

```
ggplot(diamonds, aes(z)) +
  geom_histogram() #cont
```

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



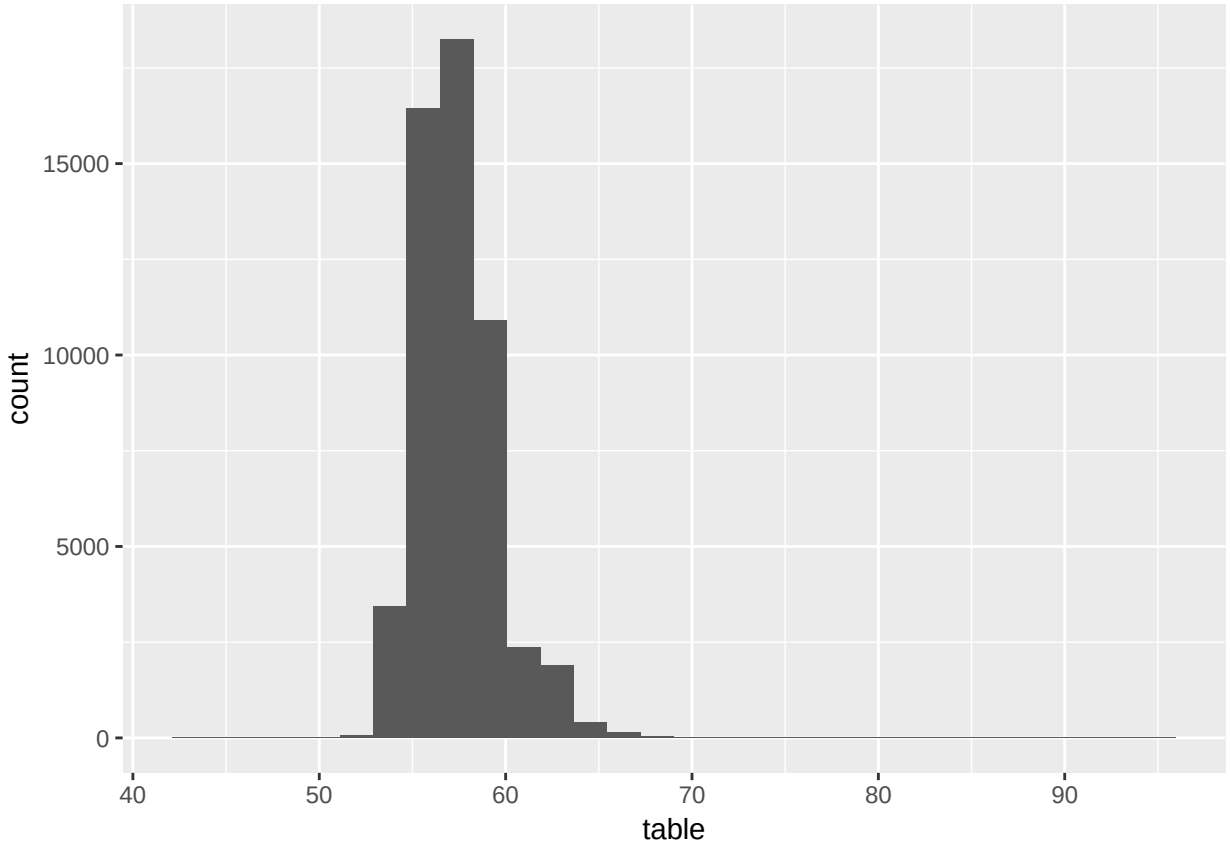

```
ggplot(diamonds, aes(depth)) +
  geom_histogram() #cont
```

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



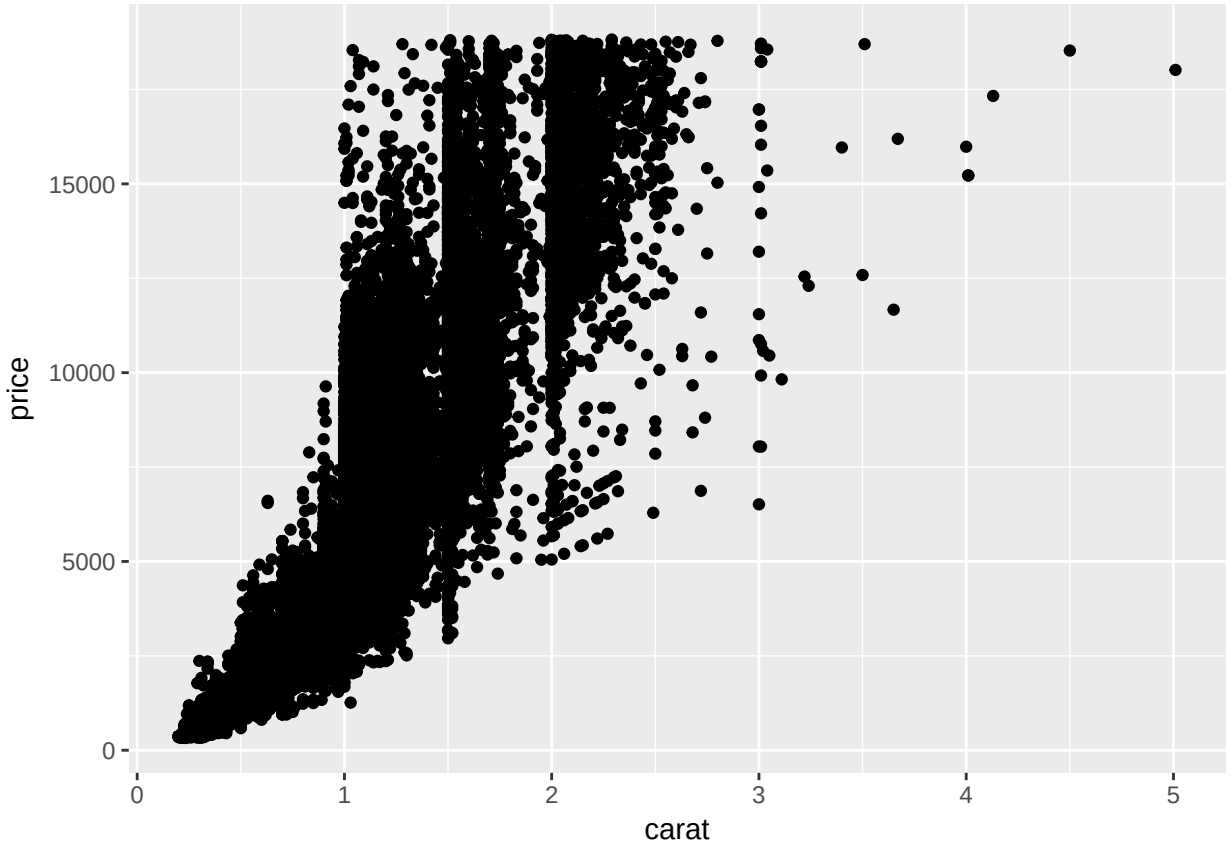

```
ggplot(diamonds, aes(table)) +
  geom_histogram() #cont
```

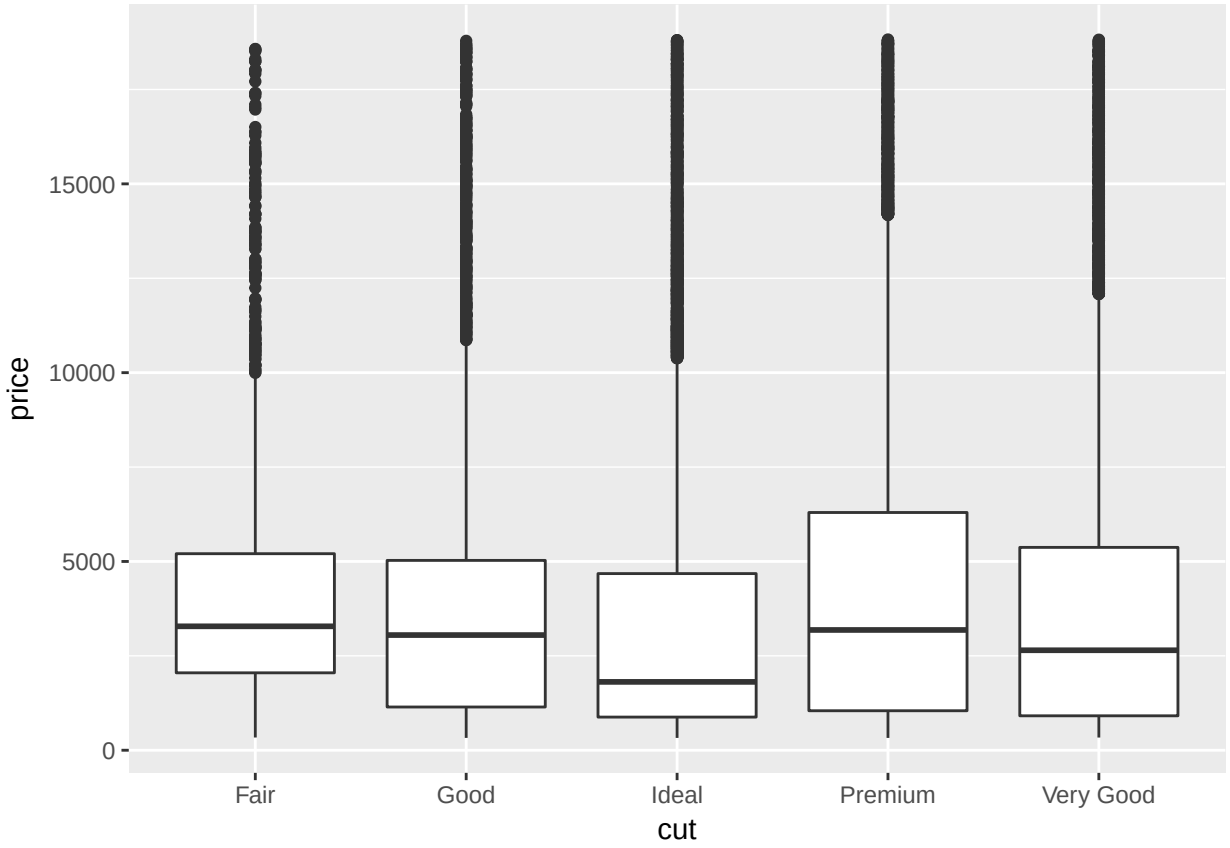
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

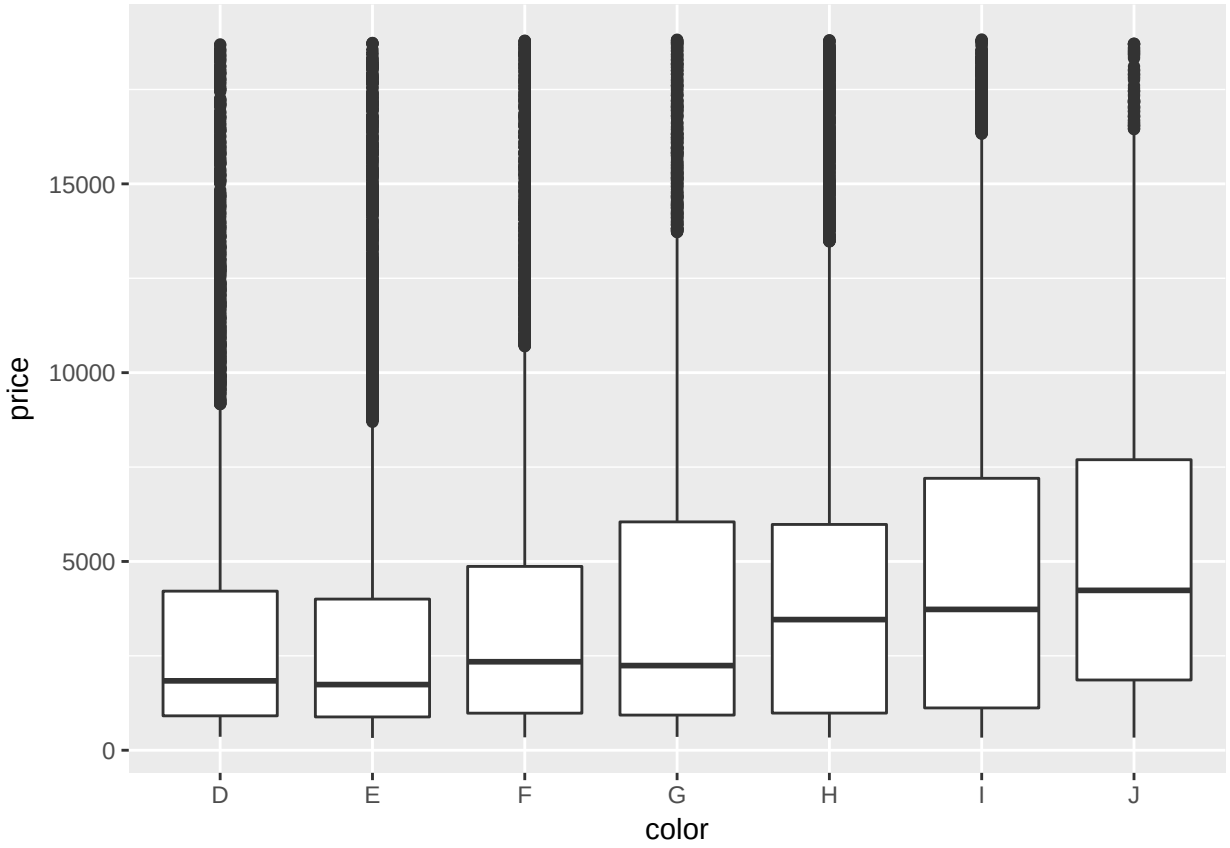


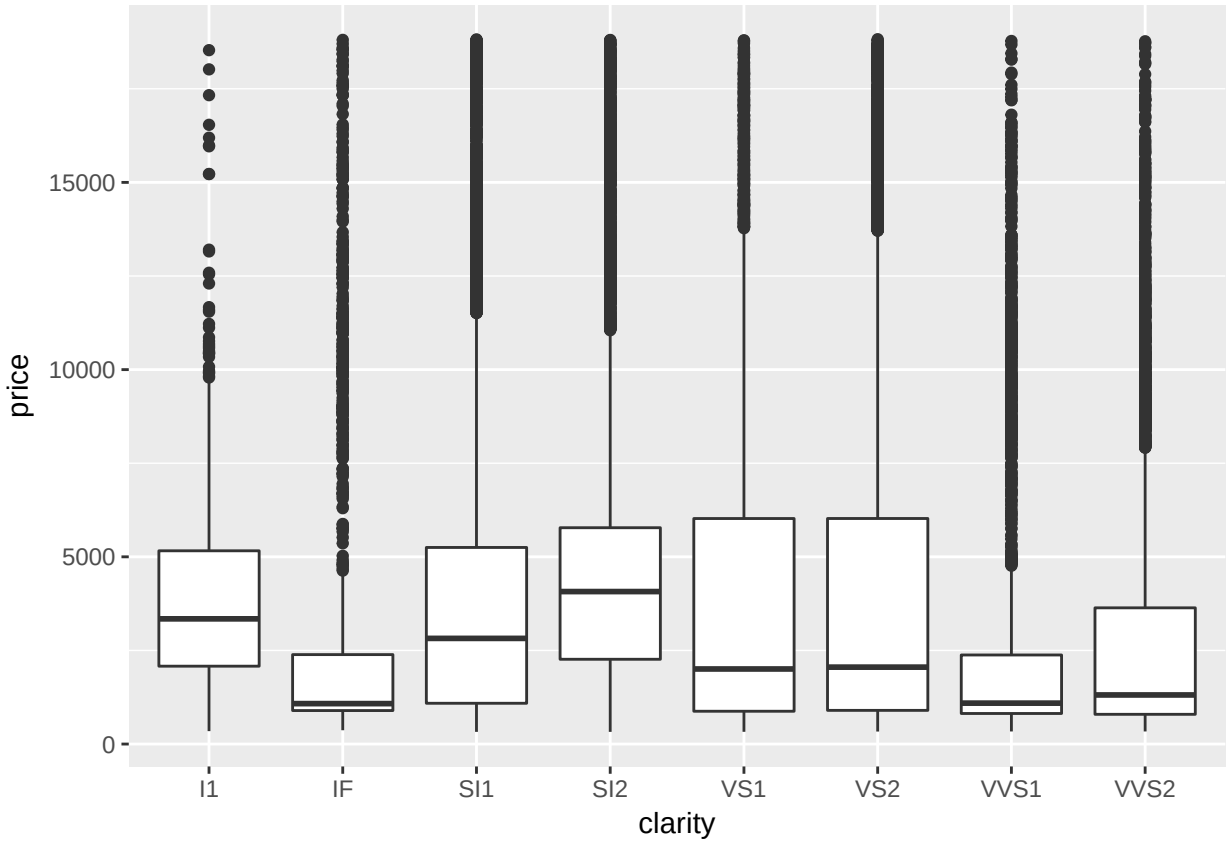
Use `ggplot` to look at the bivariate distributions of the response versus *all* predictors. Make sure you handle categorical predictors differently from continuous predictors. This time employ a for loop when an logic that handles the predictor type.

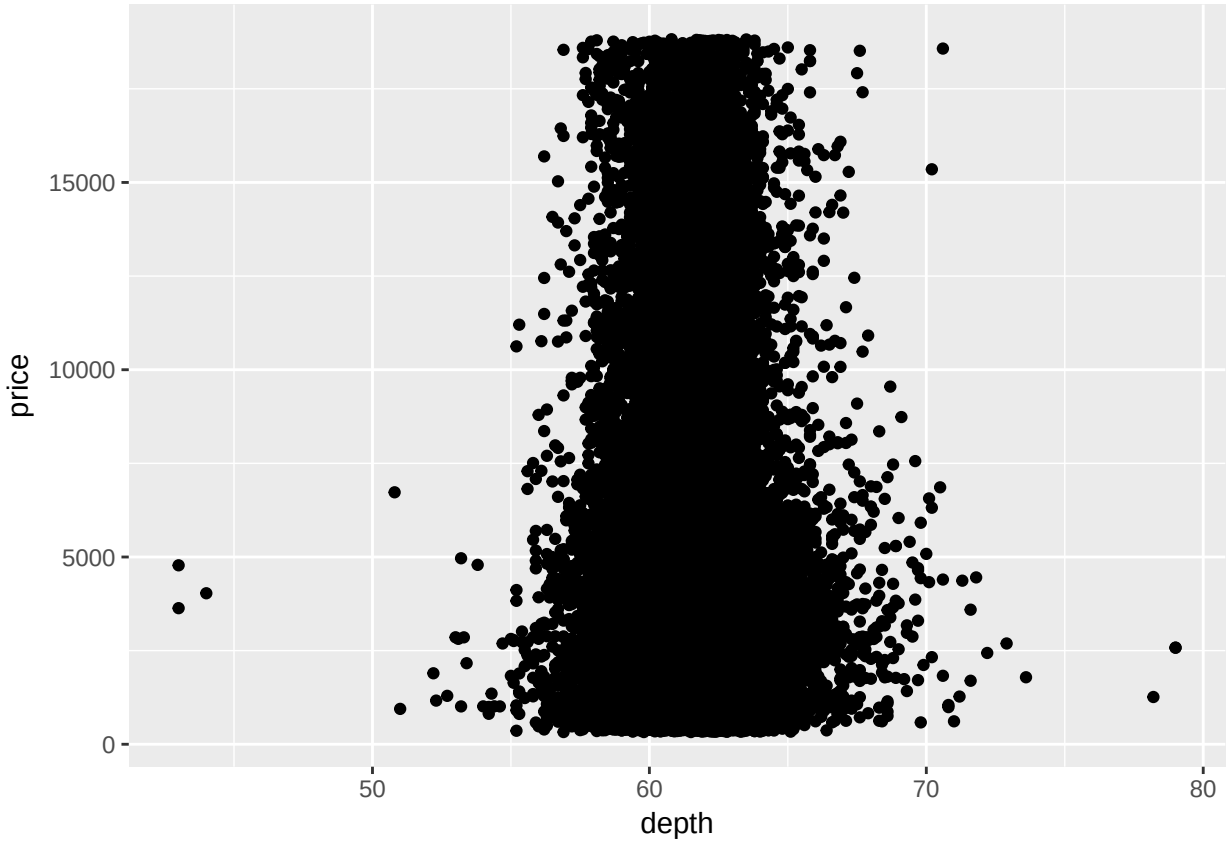
```
for (feature in setdiff(names(diamonds), "price")){
  if(class(diamonds[[feature]]) == "factor"){
    plot(ggplot(diamonds, aes(x = diamonds[[feature]], y = price)) + geom_boxplot() + xlab(feature))
} else {print(ggplot(diamonds, aes(x = diamonds[[feature]], y = price)) +
  geom_point() + xlab(feature))
  }
}
```

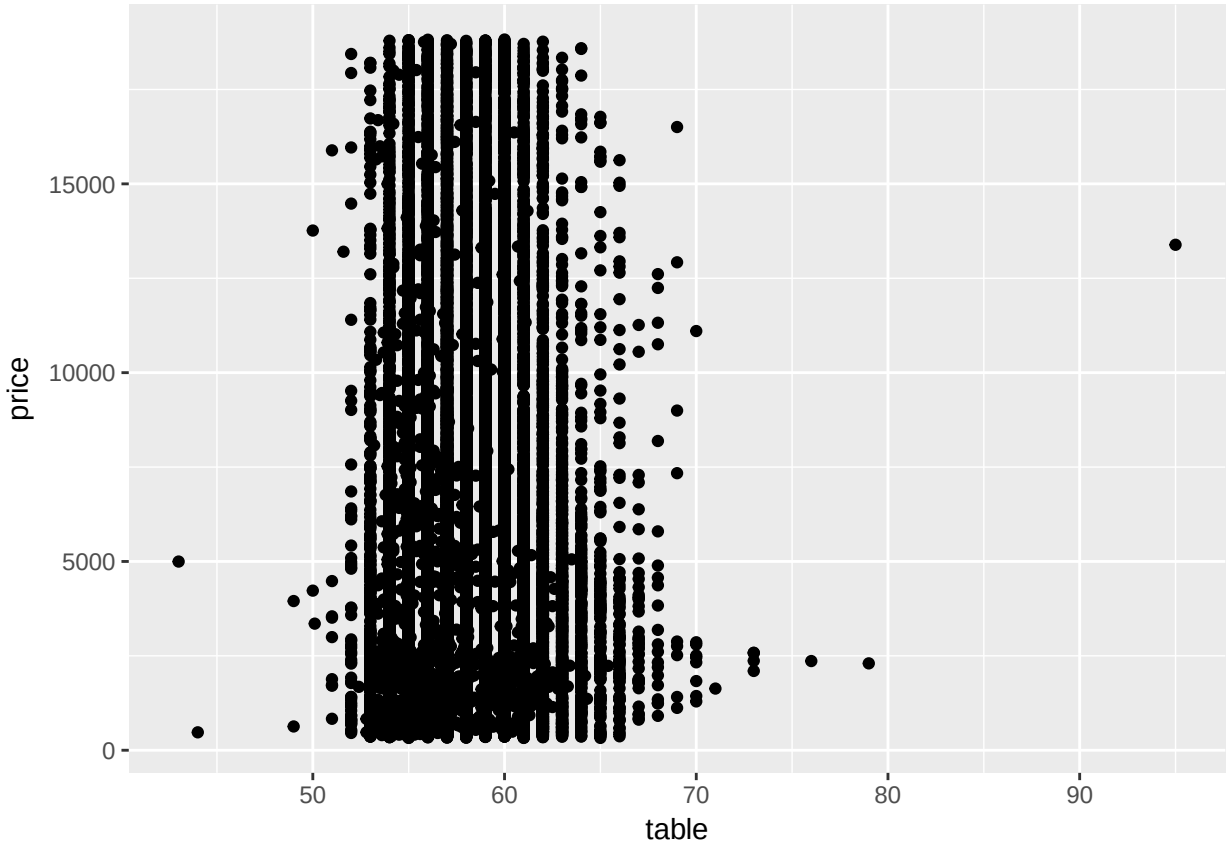

price
15000
10000
5000
0
0
1
2
3
4
5
carat

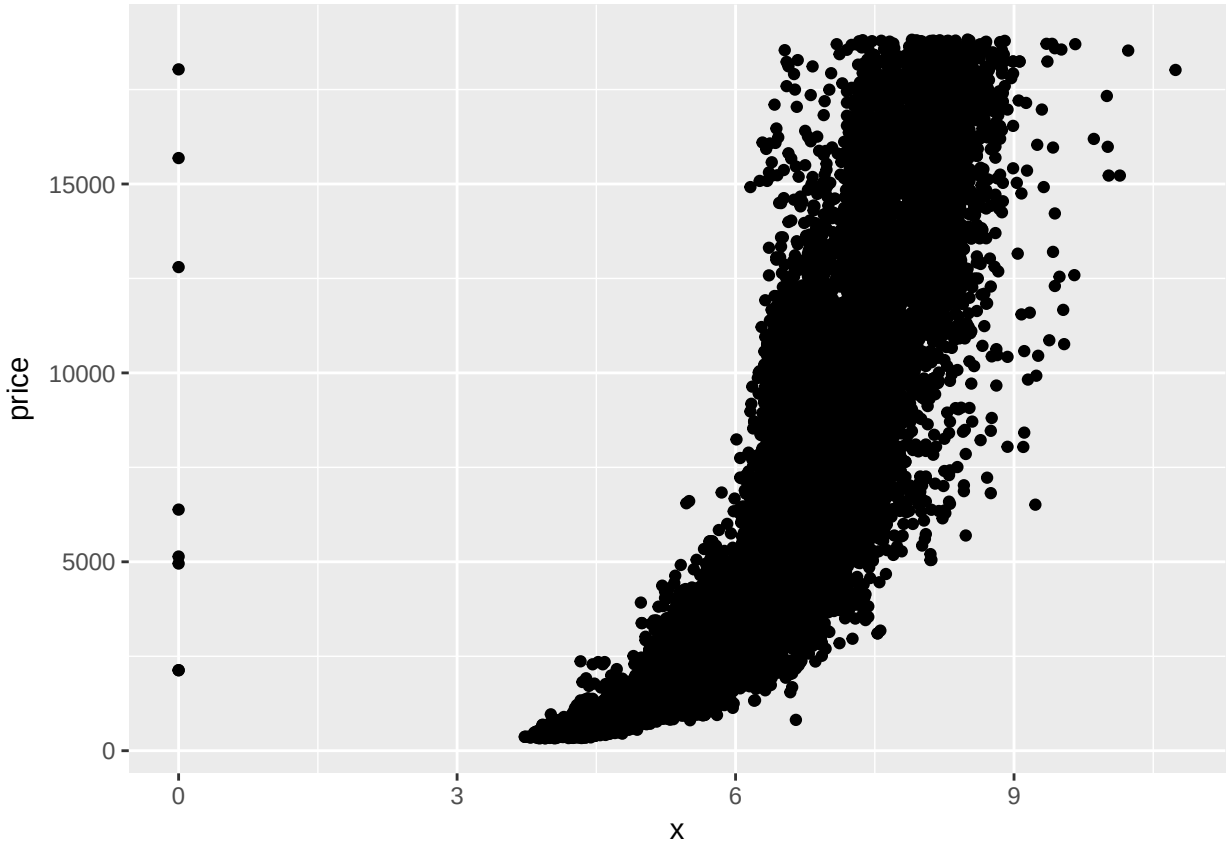

price
15000
10000
5000
0
Fair
Good
Ideal
Premium
Very Good
cut

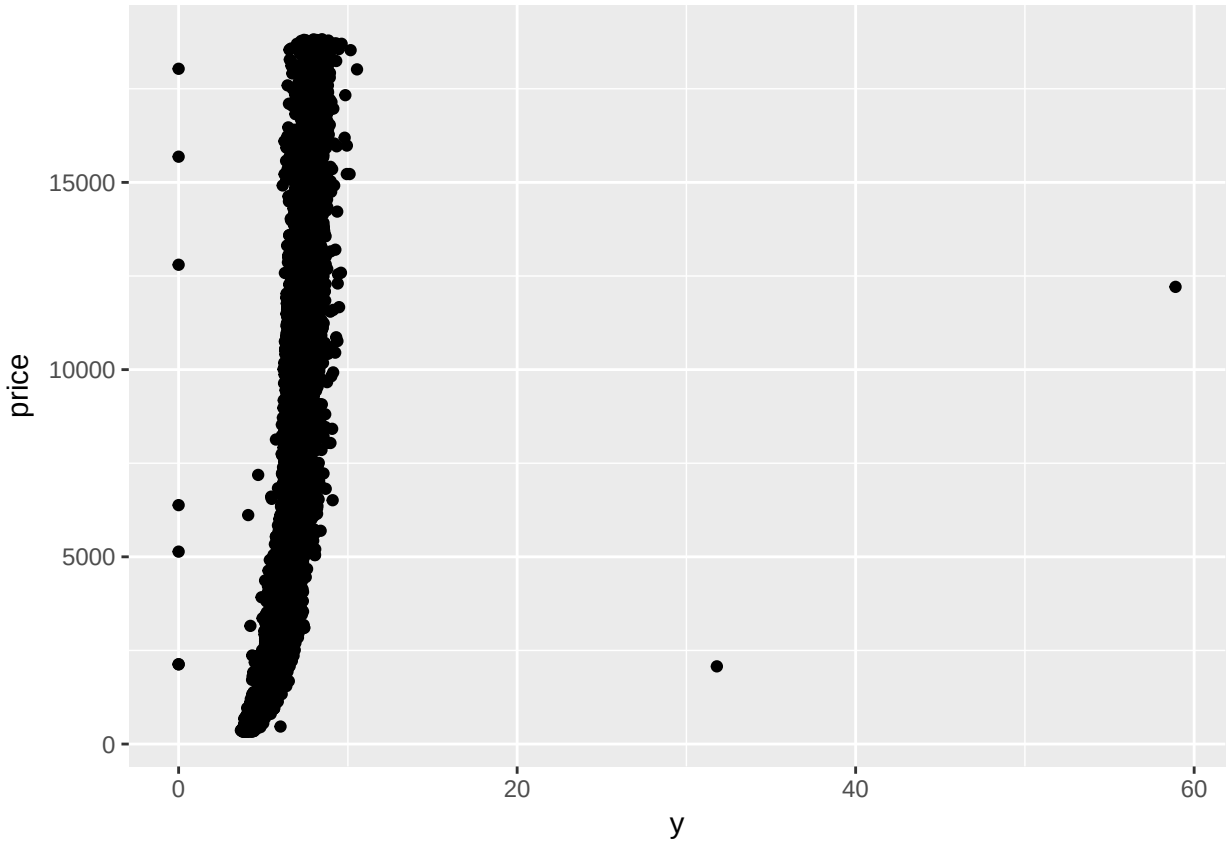

price
15000
10000
5000
0
D
E
F
G
H
I
J
color

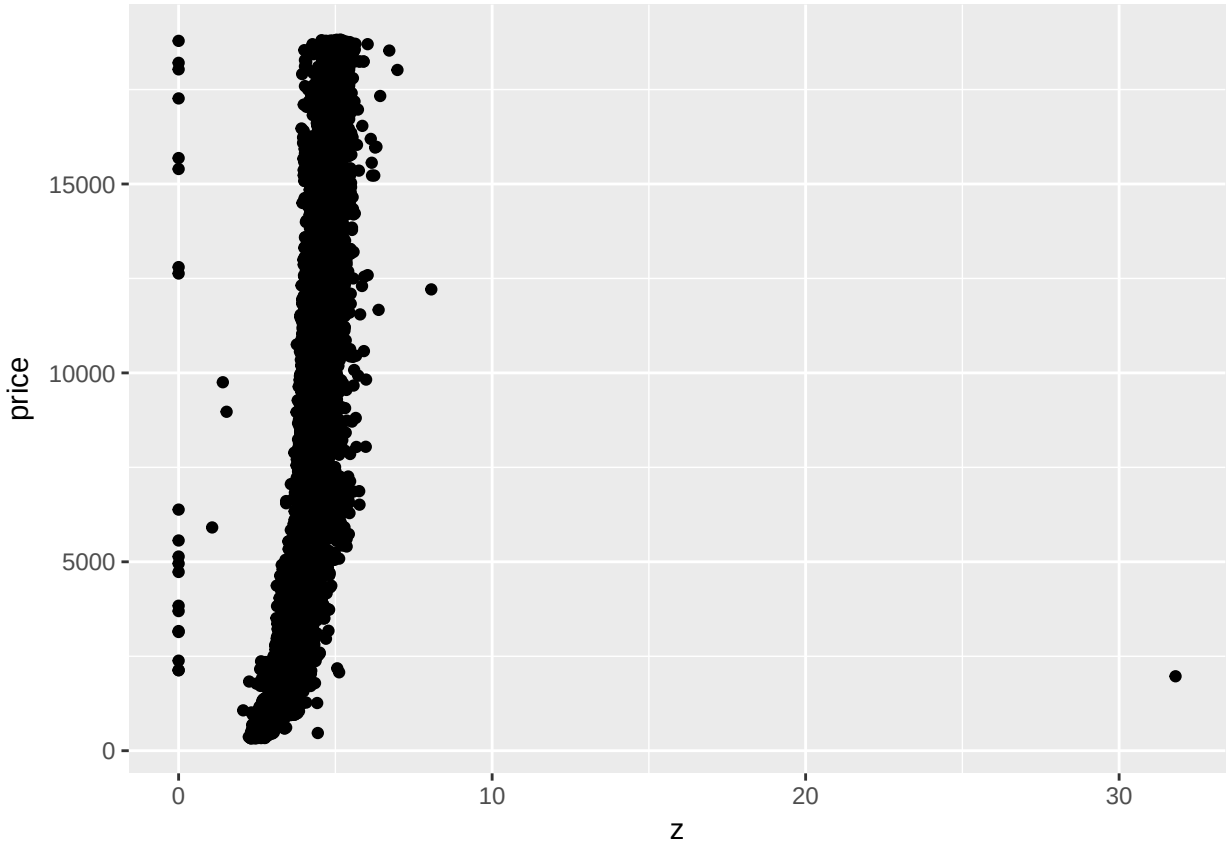

price
15000
10000
5000
0
I1
IF
SI1
SI2
VS1
VS2
VVS1
VVS2
clarity


price
15000
10000
5000
0
50
60
70
80
depth


price
15000
10000
5000
0
50
60
70
80
90
table


price
15000
10000
5000
0
0
3
6
9
x


price
15000
10000
5000
0
0
20
40
60
y



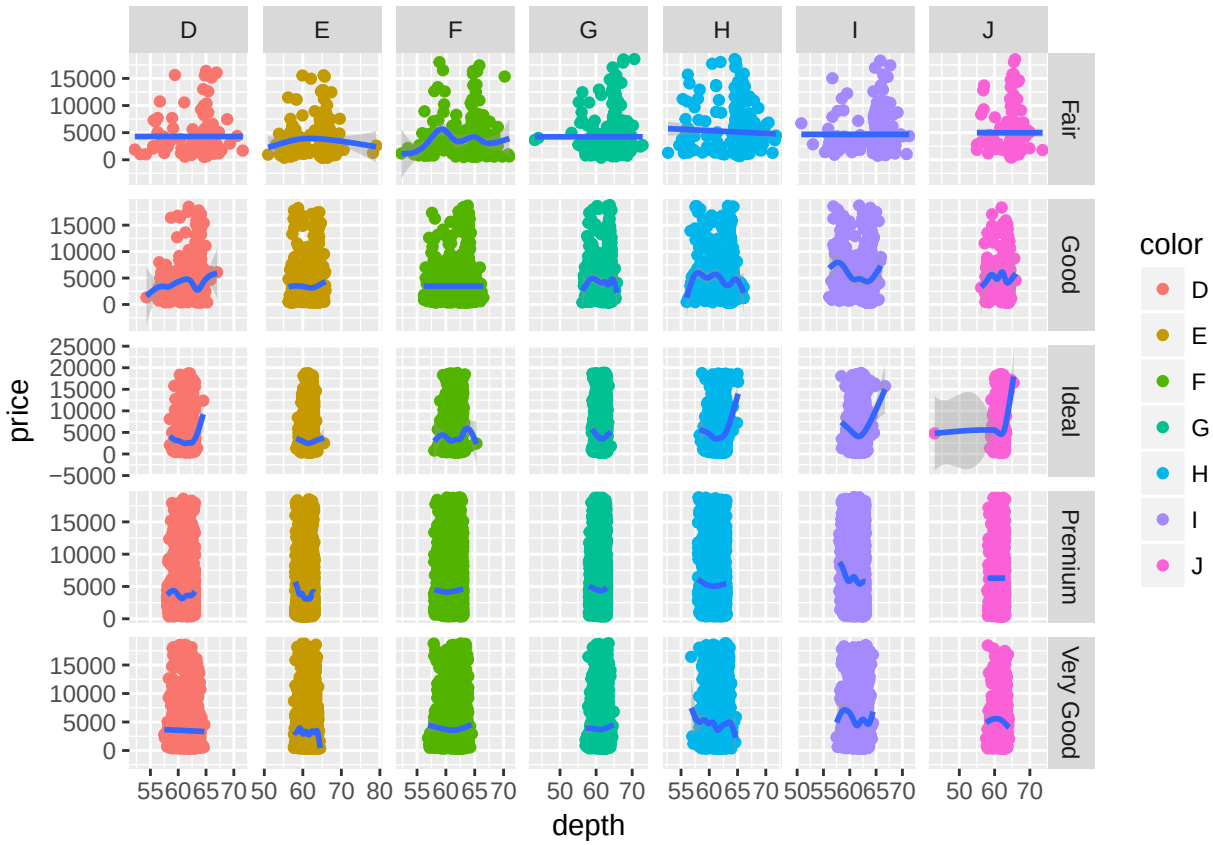
Does depth appear to be mostly independent of price?

Yes, depth appears to be mostly independent of price, since it is taking on many different prices at the same depths (around 62-63).

Look at depth vs price by predictors cut (using faceting) and color (via different colors).

```
ggplot(diamonds, aes(x = depth, y = price)) +
  geom_point(aes(col = color)) +
  geom_smooth() +
  facet_grid(cut ~ color, scales = "free")
```

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



Does diamond color appear to be independent of diamond depth?

Color appears to be independent of diamond depth because no matter what color we are in, the variance of the diamond depth appears to be the similar across all diamonds of the same color.

Does diamond cut appear to be independent of diamond depth?

The cut appears to be dependent of depth because the "better" the cut, the more depth is centered around the mean. The lower the cut, the more the depth of the diamond varies.

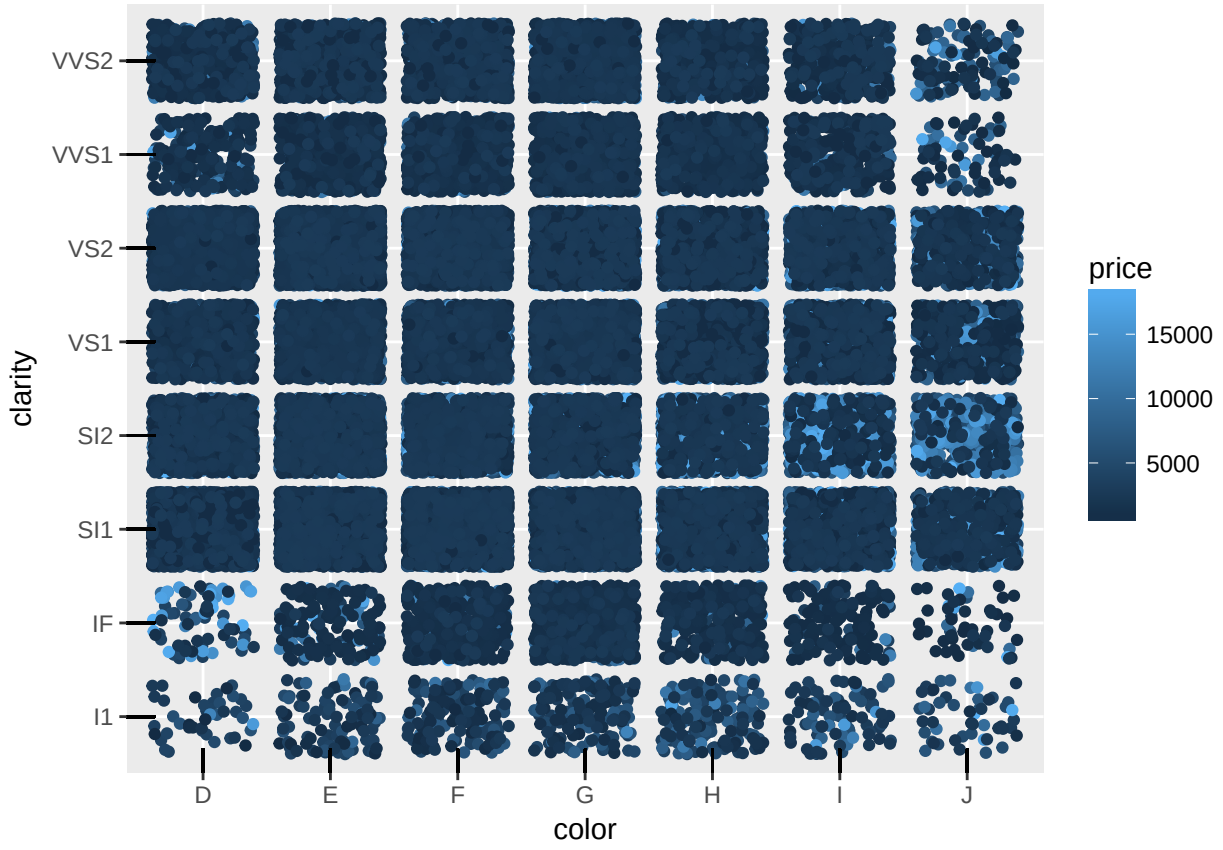
Do these plots allow you to assess well if diamond cut is independent of diamond price? Yes / no

Yes, it allows us to assess if the cut is independent of price.

We never discussed in class bivariate plotting if both variables were categorical. Use the geometry "jitter" to visualize color vs clarity. Visualize price using different colors. Use a small sized dot.

```
ggplot(diamonds, aes(x = color, y = clarity), size = "small") +
  geom_point(position = "jitter", aes(color = price)) +
  geom_smooth() +
  geom_rug()
```

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



Does diamond clarity appear to be mostly independent of diamond color?

Yes, diamond clarity appears to be mostly independent of diamond color. There are few diamonds of low color and low clarity, few of high color and high clarity and yet many of average clarity and average color. There is no "specific" trend that follows between the two features.

2. Use `lm` to run a least squares linear regression using depth to explain price.

```
reg1 = lm(price ~ depth, data = diamonds)
reg1
```

```
##
## Call:
## lm(formula = price ~ depth, data = diamonds)
##
## Coefficients:
## (Intercept)        depth
##     5763.67       -29.65
```

What is b, R^2 and the RMSE? What was the standard error of price originally?

```
b1 = reg1$coefficients
b1
```

```
## (Intercept)       depth
##  5763.66772   -29.64997
```

```
rsq1 = summary(reg1)$r.squared
rsq1
```

```
## [1] 0.0001133672
```

```
rmse1 = summary(reg1)$sigma
rmse1
```

```
## [1] 3989.251
```

```
se1 = sd(reg1$residuals)
se1
```

```
## [1] 3989.214
```

Are these metrics expected given the appropriate or relevant visualization(s) above?

Given that the "best" looking line to approximate this data would be some sort of vertical line, it makes sense that R^2 would be very low, and the RMSE / standard error would be extremely high because the linear model is not doing a "great" job of modelling the data.

Use `lm` to run a least squares linear regression using carat to explain price.

```
reg2 = lm(price ~ carat, data = diamonds)
reg2
```

```
##
## Call:
## lm(formula = price ~ carat, data = diamonds)
##
## Coefficients:
## (Intercept)        carat
##       -2256         7756
```

What is b, R^2 and the RMSE? What was the standard error of price originally?

```
b2 = reg2$coefficients
b2
```

```
## (Intercept)       carat
##   -2256.361    7756.426
```

```
rsq2 = summary(reg2)$r.squared
rsq2
```

```
## [1] 0.8493305
```

```
rmse2 = summary(reg2)$sigma
rmse2
```

```
## [1] 1548.562
```

```
se2 = sd(reg2$residuals)
se2
```

```
## [1] 1548.548
```

Are these metrics expected given the appropriate or relevant visualization(s) above?

Given the visualization above, these metrics make sense as well because there is an upward trend that the linear model can capture and model.

3. Use `lm` to run a least squares anova model using color to explain price.

```
reg3 = lm(price ~ color, data = diamonds)
reg3
```

```
##
## Call:
## lm(formula = price ~ color, data = diamonds)
##
## Coefficients:
## (Intercept)       colorE       colorF       colorG       colorH
##      3170.0        -93.2        554.9        829.2       1316.7
##      colorI       colorJ
##      1921.9       2153.9
```

What is b, R^2 and the RMSE? What was the standard error of price originally?

```
b3 = reg3$coefficients
b3
```

```
## (Intercept)      colorE      colorF      colorG      colorH      colorI
##  3169.95410   -93.20162   554.93230   829.18158  1316.71510  1921.92086
##      colorJ
##  2153.86392
```

```
rsq3 = summary(reg3)$r.squared
rsq3
```

```
## [1] 0.03127542
```

```
rmse3 = summary(reg3)$sigma
rmse3
```

```
## [1] 3926.777
```

```
se3 = sd(reg3$residuals)
se3
```

```
## [1] 3926.558
```

Are these metrics expected given the appropriate or relevant visualization(s) above?

Given the above visualizations, this would make sense because there is no "significant" looking trend between the color of a diamond and its price. The better the color is, the more outliers there are compared to the box plot (which has the average within). The worse the color, the fewer outliers there are.

Our model only included one feature - why are there more than two estimates in b?

There are more than two estimates in b because color has 7 "options" which are translated into 6 dummy variables and an intercept, which is the reference category.

Verify that the least squares linear model fit gives the sample averages of each price given color combination. Make sure to factor in the intercept here.

```
avgD = b3[1]
avgD
```

```
## (Intercept)
##    3169.954
```

```
avgE = b3[1]+b3[2]
avgE
```

```
## (Intercept)
##    3076.752
```

```
avgF = b3[1]+b3[3]
avgF
```

```
## (Intercept)
##    3724.886
```

```
avgG = b3[1]+b3[4]
avgG
```

```
## (Intercept)
##    3999.136
```

```
avgH = b3[1]+b3[5]
avgH
```

```
## (Intercept)
##    4486.669
```

```
avgI = b3[1]+b3[6]
avgI
```

```
## (Intercept)
##    5091.875
```

```
avgJ = b3[1]+b3[7]
avgJ
```

```
## (Intercept)
##    5323.818
```

```
#to make sure that the sample averages are equivalent to above
mean(diamonds$price[diamonds$color == "D"])
```

```
## [1] 3169.954
```

```
mean(diamonds$price[diamonds$color == "E"])
```

```
## [1] 3076.752
```

```
mean(diamonds$price[diamonds$color == "F"])
```

```
## [1] 3724.886
```

```
mean(diamonds$price[diamonds$color == "G"])
```

```
## [1] 3999.136
```

```
mean(diamonds$price[diamonds$color == "H"])
```

```
## [1] 4486.669
```

```
mean(diamonds$price[diamonds$color == "I"])
```

```
## [1] 5091.875
```

```
mean(diamonds$price[diamonds$color == "J"])
```

```
## [1] 5323.818
```

Fit a new model without the intercept and verify the sample averages of each colors' prices *directly* from the entries of vector b.

```
reg3a = lm(price ~ 0 + color, data = diamonds)
reg3a
```

```
##
## Call:
## lm(formula = price ~ 0 + color, data = diamonds)
##
## Coefficients:
## colorD  colorE  colorF  colorG  colorH  colorI  colorJ
##   3170    3077    3725    3999    4487    5092    5324
```

```
b3a = reg3a$coefficients
#to make sure that the sample averages are equivalent to above
mean(diamonds$price[diamonds$color == "D"])
```

```
## [1] 3169.954
```

```
mean(diamonds$price[diamonds$color == "E"])
```

```
## [1] 3076.752
```

```
mean(diamonds$price[diamonds$color == "F"])
```

```
## [1] 3724.886
```

```
mean(diamonds$price[diamonds$color == "G"])
```

```
## [1] 3999.136
```

```
mean(diamonds$price[diamonds$color == "H"])
```

```
## [1] 4486.669
```

```
mean(diamonds$price[diamonds$color == "I"])
```

```
## [1] 5091.875
```

```
mean(diamonds$price[diamonds$color == "J"])
```

```
## [1] 5323.818
```

What would extrapolation look like in this model? We never covered this in class explicitly.

Extrapolation would be to predict a new diamond's price based on its color. In this case, assuming all other features were constant, we could only predict the price of a diamond given that it had one of these colors. Since we don't have any other info about other features, we could say that the color is the "starting price", like a base value. Given only a colorD diamond, we could say that while building the price, we would start at $3,169.95 and either increase or decrease the price as we add other features. However, we cannot predict anything for a diamond that does not have one of the colors listed.

4. Use `lm` to run a least squares linear regression using all available features to explain diamond price.

```
reg4 = lm(price ~ ., data = diamonds)
reg4
```

```
##
## Call:
## lm(formula = price ~ ., data = diamonds)
##
## Coefficients:
##  (Intercept)         carat       cutGood      cutIdeal    cutPremium
```

```
##     2184.477     11256.978       579.751       832.912       762.144
## cutVery Good        colorE        colorF        colorG        colorH
##      726.783      -209.118      -272.854      -482.039      -980.267
##       colorI        colorJ     clarityIF    claritySI1    claritySI2
##    -1466.244     -2369.398      5345.102      3665.472      2702.586
##   clarityVS1    clarityVS2   clarityVVS1   clarityVVS2         depth
##     4578.398      4267.224      5007.759      4950.814       -63.806
##        table             x             y             z
##      -26.474     -1008.261         9.609       -50.119
```

What is b, R^2 and the RMSE? Also - provide an approximate 95% interval for predictions using the empirical rule.

```
b4 = reg4$coefficients
b4
```

```
##  (Intercept)        carat      cutGood     cutIdeal   cutPremium
##  2184.477350 11256.978307   579.751446   832.911845   762.143950
## cutVery Good       colorE       colorF       colorG       colorH
##   726.782591  -209.118085  -272.853832  -482.038904  -980.266675
##       colorI       colorJ    clarityIF   claritySI1   claritySI2
## -1466.244474 -2369.398063  5345.102246  3665.472080  2702.586294
##   clarityVS1   clarityVS2  clarityVVS1  clarityVVS2        depth
##  4578.397915  4267.223565  5007.759045  4950.814072   -63.806100
##        table            x            y            z
##   -26.474085 -1008.261098     9.608886   -50.118891
```

```
rsq4 = summary(reg4)$r.squared
rsq4
```

```
## [1] 0.9197915
```

```
rmse4 = summary(reg4)$sigma
rmse4
```

```
## [1] 1130.094
```

```
se4 = sd(reg4$residuals)
se4
```

```
## [1] 1129.853
```

```
ci4 = confint(reg4, level = 0.95)
ci4
```

```
##                    2.5 %      97.5 %
## (Intercept)   1384.40874  2984.54596
## carat        11161.66800 11352.28861
## cutGood        513.91060   645.59229
## cutIdeal       767.43293   898.39076
## cutPremium     698.97766   825.31024
## cutVery Good   663.59067   789.97451
## colorE        -244.18872  -174.04745
## colorF        -308.31567  -237.39199
## colorG        -516.76265  -447.31516
## colorH       -1017.18508  -943.34827
## colorI       -1507.72284 -1424.76611
## colorJ       -2420.61480 -2318.18132
## clarityIF     5245.09472  5445.10978
```

```
## claritySI1    3579.94902  3750.99514
## claritySI2    2616.70173  2788.47086
## clarityVS1    4491.08740  4665.70843
## clarityVS2    4181.27041  4353.17672
## clarityVVS1   4915.32566  5100.19243
## clarityVVS2   4860.93839  5040.68976
## depth          -72.69386   -54.91834
## table          -32.18095   -20.76722
## x            -1072.74095  -943.78125
## y              -28.28374    47.50152
## z             -115.75231    15.51453
```

Interpret all entries in the vector b.

Intercept: The predicted price of a diamond that has cutFair, colorD, clarityI1, with all other features equal to 0 is $2,184.48.
Carat: As carat increases by 1 unit, the predicted price of a diamond increases by $11,256.98 relative to the intercept, holding all other features constant. cutGood: The price of a cutGood diamond is $579.75 higher than a diamond with the intercept features, holding all other features constant.
cutIdeal: The price of a cutIdeal diamond is $832.91 higher than a diamond with the intercept features, holding all other features constant. cutPremium: The price of a cutPremium diamond is $762.14 higher than a diamond with the intercept features, holding all other features constant. cutVery_Good: The price of a cutVery_Good diamond is $726.78 higher than a diamond with the intercept features, holding all other features constant. colorE: The price of a colorE diamond is $209.12 less than a diamond with the intercept features, holding all other features constant. colorF: The price of a colorE diamond is $272.85 less than a diamond with the intercept features, holding all other features constant. colorG: The price of a colorE diamond is $482.04 less than a diamond with the intercept features, holding all other features constant. colorH: The price of a colorE diamond is $980.27 less than a diamond with the intercept features, holding all other features constant. colorI: The price of a colorE diamond is $1466.24 less than a diamond with the intercept features, holding all other features constant. colorJ: The price of a colorE diamond is $2369.40 less than a diamond with the intercept features, holding all other features constant. clarityIF: The price of a clarityIF diamond is $5345.10 higher than a diamond with the intercept features, holding all other features constant. claritySI1: The price of a claritySI1 diamond is $3665.47 higher than a diamond with the intercept features, holding all other features constant. claritySI2: The price of a claritySI2 diamond is $2702.59 higher than a diamond with the intercept features, holding all other features constant. clarityVS1: The price of a clarityVS1 diamond is $4578.40 higher than a diamond with the intercept features, holding all other features constant. clarityVS2: The price of a clarityVS2 diamond is $4267.22 higher than a diamond with the intercept features, holding all other features constant. clarityVVS1: The price of a clarityVVS1 diamond is $5007.76 higher than a diamond with the intercept features, holding all other features constant. clarityVVS2: The price of a clarityVVS2 diamond is $4950.81 higher than a diamond with the intercept features, holding all other features constant. depth: As depth increases by 1 unit, the price of a diamond decreases by $63.81 relative to the intercept, holding all other features constant. table: As table increases by 1 unit, the price of a diamond decreases by $26.47 relative to the intercept, holding all other features constant. x: As x increases by 1 unit, the price of a diamond decreases by $1008.26 relative to the intercept, holding all other features constant. y: As y increases by 1 unit, the price of a diamond increases by $9.61 relative to the intercept, holding all other features constant. z: As z increases by 1 unit, the price of a diamond decreases by $50.12 relative to the intercept, holding all other features constant.

Are these metrics expected given the appropriate or relevant visualization(s) above? Can you tell from the visualizations?

Given the visualizations from above, I think these metrics are expected because the lower the diamond clarity, the more the price will decrease. The better the cut, the higher the price. The better the color, the higher the price as well. Inutitively as carats increase price increases as well. We can also tell this from the visualizations based on how the data trends upwards or downwards.

Comment on why R^2 is high. Think theoretically about diamonds and what you know about them.

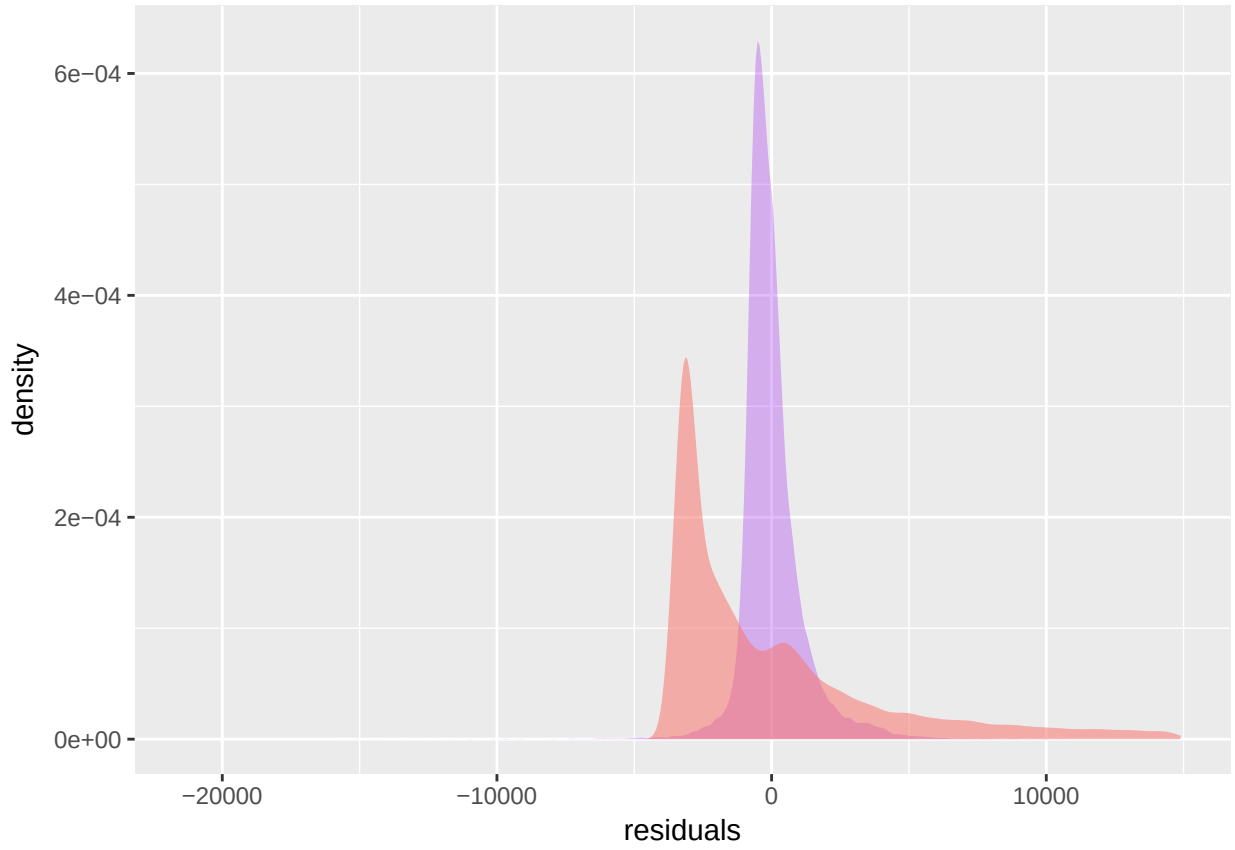
R^2 is high because we have many predictors here (that arise from the creation of the dummy variables), and these predictors are all features that influence the price of the diamond. They are easily observable, and they are easily "rankable" in the sense that there is a clear scale that people universally could declare one diamond's features to be better than another's with little to no disagreeing.

Do you think you overfit? Comment on why or why not but do not do any numerical testing or coding.

I think I overfit a tiny bit because there is more that goes into predicting the price of diamonds. For example, the location of where diamonds are being sold will influence how cheap/expensive the diamond will be. Countries where it is not a novelty may not charge as much as the United States because here, it is a big business. In addition, there will be price variability between sellers. One may sell on the cheaper side to secure the sale, while another may charge way above price to keep "exclusive". These factors are not accounted for in the regression and so this model may not have as great predicting power as the R^2 value suggests.

Create a visualization that shows the "original residuals" (i.e. the prices minus the average price) and the model residuals.

```
ggplot(data.frame(null_residuals = diamonds$price - mean(diamonds$price), residuals = residuals(reg4))) +
  stat_density(aes(x = residuals), fill = "purple", alpha = 0.3) +
  stat_density(aes(x = null_residuals, fill = "green", alpha = 0.3)) +
  theme(legend.position = "none")
```



5. Reference your visualizations above. Does price vs. carat appear linear?

Price vs. carat does not appear completely linear. It can be modelled linearly but there looks to be a slight curvature around the lower left hand side of the data plot.

Upgrade your model in #4 to use one polynomial term for carat.

```
reg5 = lm(price ~ . + I(carat^2), data = diamonds)
reg5
```

```
##
## Call:
## lm(formula = price ~ . + I(carat^2), data = diamonds)
##
## Coefficients:
##  (Intercept)         carat       cutGood      cutIdeal    cutPremium
##      9807.98      16144.76        538.33        807.52        747.70
## cutVery Good        colorE        colorF        colorG        colorH
##       678.32       -209.44       -284.55       -496.85       -997.60
##       colorI        colorJ     clarityIF    claritySI1    claritySI2
##     -1469.25      -2357.80       5243.52       3565.41       2605.54
##   clarityVS1    clarityVS2   clarityVVS1   clarityVVS2         depth
##      4475.44       4163.35       4904.23       4843.80       -116.23
##        table             x             y             z    I(carat^2)
##       -36.37      -2123.01        -23.46        -83.11      -1028.82
```

What is b, R^2 and the RMSE?

```
b5 = reg5$coefficients
b5
```

```
##  (Intercept)         carat       cutGood      cutIdeal    cutPremium
##   9807.97904  16144.75809    538.33407    807.51616    747.69518
## cutVery Good       colorE       colorF       colorG       colorH
##    678.31993   -209.43992   -284.54706   -496.84716   -997.60127
##       colorI       colorJ    clarityIF   claritySI1   claritySI2
##  -1469.25151  -2357.79746   5243.52276   3565.41193   2605.54013
##   clarityVS1   clarityVS2  clarityVVS1  clarityVVS2        depth
##   4475.44424   4163.34947   4904.22750   4843.80493   -116.22729
##        table            x            y            z   I(carat^2)
##    -36.37384  -2123.00617    -23.46172    -83.11272  -1028.81806
```

```
rsq5 = summary(reg5)$r.squared
rsq5
```

```
## [1] 0.9214777
```

```
se5 = summary(reg5)$sigma
se5
```

```
## [1] 1118.162
```

Interpret each element in b just like previously. You can copy most of the text from the previous question but be careful. There is one tricky thing to explain.

Intercept: The predicted price of a diamond that has cutFair, colorD, clarityI1, with all other features equal to 0 is $9807.98.
Carat: As carat increases by 1 unit, the predicted price of a diamond increases by ($16,144.76 minus the number of carats times $1028.82) relative to the intercept, holding all other features constant. cutGood: The price of a cutGood diamond is $538.33 higher than a diamond with the intercept features, holding all other features constant.
cutIdeal: The price of a cutIdeal diamond is $807.52 higher than a diamond with the intercept features, holding all other features constant. cutPremium: The price of a cutPremium diamond is $747.70 higher than a diamond with the intercept features, holding all other features constant. cutVery_Good: The price of a cutVery_Good diamond is $678.32 higher than a diamond with the intercept features, holding all other

features constant. colorE: The price of a colorE diamond is $209.44 less than a diamond with the intercept features, holding all other features constant. colorF: The price of a colorE diamond is $284.55 less than a diamond with the intercept features, holding all other features constant. colorG: The price of a colorE diamond is $496.85 less than a diamond with the intercept features, holding all other features constant. colorH: The price of a colorE diamond is $997.60 less than a diamond with the intercept features, holding all other features constant. colorI: The price of a colorE diamond is $1469.25 less than a diamond with the intercept features, holding all other features constant. colorJ: The price of a colorE diamond is $2357.80 less than a diamond with the intercept features, holding all other features constant. clarityIF: The price of a clarityIF diamond is $5243.52 higher than a diamond with the intercept features, holding all other features constant. claritySI1: The price of a claritySI1 diamond is $3565.41 higher than a diamond with the intercept features, holding all other features constant. claritySI2: The price of a claritySI2 diamond is $2605.54 higher than a diamond with the intercept features, holding all other features constant. clarityVS1: The price of a clarityVS1 diamond is $4475.44 higher than a diamond with the intercept features, holding all other features constant. clarityVS2: The price of a clarityVS2 diamond is $4163.35 higher than a diamond with the intercept features, holding all other features constant. clarityVVS1: The price of a clarityVVS1 diamond is $4904.23 higher than a diamond with the intercept features, holding all other features constant. clarityVVS2: The price of a clarityVVS2 diamond is $4843.80 higher than a diamond with the intercept features, holding all other features constant. depth: As depth increases by 1 unit, the price of a diamond decreases by $116.23 relative to the intercept, holding all other features constant. table: As table increases by 1 unit, the price of a diamond decreases by $36.37 relative to the intercept, holding all other features constant. x: As x increases by 1 unit, the price of a diamond decreases by $2123.01 relative to the intercept, holding all other features constant. y: As y increases by 1 unit, the price of a diamond increases by $23.46 relative to the intercept, holding all other features constant. z: As z increases by 1 unit, the price of a diamond decreases by $83.11 relative to the intercept, holding all other features constant.

Is this an improvement over the model in #4? Yes/no and why.

This is a slight improvement over the model in #4, but mostly because by adding another feature to our model, we are just increasing the column space of the matrix, and in turn, the projection onto the column space gets a little bit larger.

Define a function g that makes predictions given a vector of the same features in $\mathbb{D}$.

```
#TO-DO
#did not get to
```

6. Use `lm` to run a least squares linear regression using a polynomial of color of degree 2 to explain price.

```
reg6 = lm(price ~ I(color^2), data=diamonds)
```

```
## Warning in Ops.factor(color, 2): '^' not meaningful for factors
```

```
## Error in lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...): 0 (non-NA) cases
```

```
reg6
```

```
## Error in eval(expr, envir, enclos): object 'reg6' not found
```

Why did this throw an error?

This threw an error because we are regressing on a categorical variable, and thus there would be no difference when just using the regular variable. There is no advantage to squaring.

7. Redo the model fit in #4 without using `lm` but using the matrix algebra we learned about in class. This is hard and requires many lines, but it's all in the notes.

```
#TO-DO
#outputting an error for the continuous variables, need to make dummies for new matrix
#left out one level on each feature b/c will go into intercept
diamondsnew = diamonds
```

```
diamondsnew$cutGood = ifelse(diamonds$cut == "Good", 1, 0)
diamondsnew$cutVeryGood = ifelse(diamonds$cut == "Very Good", 1, 0)
diamondsnew$cutPremium = ifelse(diamonds$cut == "Premium", 1, 0)
diamondsnew$cutIdeal = ifelse(diamonds$cut == "Ideal", 1, 0)
diamondsnew$colorE = ifelse(diamonds$color == "E", 1, 0)
diamondsnew$colorF = ifelse(diamonds$color == "F", 1, 0)
diamondsnew$colorG = ifelse(diamonds$color == "G", 1, 0)
diamondsnew$colorH = ifelse(diamonds$color == "H", 1, 0)
diamondsnew$colorI = ifelse(diamonds$color == "I", 1, 0)
diamondsnew$colorJ = ifelse(diamonds$color == "J", 1, 0)
diamondsnew$claritySI1 = ifelse(diamonds$clarity == "SI1", 1, 0)
diamondsnew$claritySI2 = ifelse(diamonds$clarity == "SI2", 1, 0)
diamondsnew$clarityVS1 = ifelse(diamonds$clarity == "VS1", 1, 0)
diamondsnew$clarityVS2 = ifelse(diamonds$clarity == "VS2", 1, 0)
diamondsnew$clarityVVS1 = ifelse(diamonds$clarity == "VVS1", 1, 0)
diamondsnew$clarityVVS2 = ifelse(diamonds$clarity == "VVS2", 1, 0)
diamondsnew$clarityIF = ifelse(diamonds$clarity == "IF", 1, 0)
diamondsnew$clarity = NULL #too many cols in my new matrix
diamondsnew$color = NULL #need to remove
diamondsnew$cut = NULL #need to remove

X = as.matrix(cbind(1, diamondsnew$x, diamondsnew$y, diamondsnew$z, diamondsnew$depth, diamondsnew$table

y = diamondsnew$price
indices = sample(1 : nrow(X), 2000)
X = X[indices, ]
y = y[indices]
rm(indices)

Xt = t(X)
XtX = Xt %*% X
XtXinv = solve(XtX)
b = XtXinv %*% Xt %*% y
yhat = X %*% b
```

What is b, R^2 and the RMSE?

```
e = y - yhat
I = diag(nrow(X))
H = X %*% XtXinv %*% t(X)
e_with_H = (I - H) %*% y
ybar = mean(y)
SST = sum((y - ybar)^2)
SSR = sum((yhat - ybar)^2)
SSE = sum(e^2)

b7 = XtXinv %*% t(X) %*% y
b7
```

```
##               [,1]
##  [1,] -3502.246230
##  [2,] -1946.970066
##  [3,]  1252.025076
##  [4,]  -587.526817
##  [5,]    -8.728568
```

```
##  [6,]    12.708458
##  [7,] 11452.409213
##  [8,]   642.602026
##  [9,]   797.213249
## [10,]   879.779643
## [11,]   966.939104
## [12,]  -380.135827
## [13,]  -363.552524
## [14,]  -664.080683
## [15,] -1051.143373
## [16,] -1630.881216
## [17,] -2548.515910
## [18,]  3700.631839
## [19,]  2687.026540
## [20,]  4735.562408
## [21,]  4309.554581
## [22,]  5026.833697
## [23,]  4953.709468
## [24,]  5290.450797
```

```
rsq7 = (SST - SSE) / (SST)
rsq7
```

```
## [1] 0.9257999
```

```
rmse7 = sqrt(mean(e^2))
rmse7
```

```
## [1] 1092.926
```

Are they the same as in #4?

They are close to the ones in #4.

Redo the model fit using matrix algebra by projecting onto an orthonormal basis for the predictor space Q and the Gram-Schmidt "remainder" matrix R. Formulas are in the notes. Verify b is the same.

```
qrX = qr(X)
Q = qr.Q(qrX)
R = qr.R(qrX)

sum(Q[, 1]^2)
```

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

```
sum(Q[, 2]^2)
```

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

```
Q[, 1] %*% Q[, 2]
```

```
##                [,1]
## [1,] -1.919038e-17
```

```
Q[, 2] %*% Q[, 3]
```

```
##                [,1]
## [1,] -3.637498e-17
```

```
yhat_via_Q = Q %*% t(Q) %*% y
```

Generate the vectors $\hat{y}$, e and the hat matrix H.

```
#in order of how they rely on one another
e = y - yhat
H = X %*% XtXinv %*% t(X)
yhat = H %*% y
```

In one line each, verify that (a) $\hat{y}$ and e sum to the vector y (the prices in the original dataframe), (b) $\hat{y}$ and e are orthogonal (c) e projected onto the column space of X gets annhilated, (d) $\hat{y}$ projected onto the column space of X is unaffected, (e) $\hat{y}$ projected onto the orthogonal complement of the column space of X is annhilated (f) the sum of squares residuals plus the sum of squares model equal the original (total) sum of squares

```
pacman::p_load(testthat) #will we need tolerance levels?
expect_equal(as.vector(yhat + e), y)
expect_equal(as.vector(t(yhat) %*% e), 0, tol = 1e-1)
```

```
## Error: as.vector(t(yhat) %*% e) not equal to 0.
## 1/1 mismatches
## [1] 0.186 - 0 == 0.186
```

```
expect_equal(sum(H %*% e), 0, tol = 1e-4)
expect_equal(H %*% yhat, yhat)
expect_equal(sum((I - H) %*% yhat), 0, tol = 1e-4)
expect_equal(SSR + SSE, SST)
```

8. Fit a linear least squares model for price using all interactions and also 5-degree polynomials for all continuous predictors.

```
reg8 = lm(price ~ . * . + I(carat^5) + I(x^5) + I(y^5) + I(z^5) + I(depth^5) + I(table^5), data = diamo
reg8
```

```
##
## Call:
## lm(formula = price ~ . * . + I(carat^5) + I(x^5) + I(y^5) + I(z^5) +
##     I(depth^5) + I(table^5), data = diamonds)
##
## Coefficients:
##               (Intercept)                     carat
##                 7.886e+03                -8.654e+03
##                   cutGood                   cutIdeal
##                -2.210e+03                 2.587e+03
##                cutPremium              cutVery Good
##                 1.239e+03                -5.837e+02
##                    colorE                    colorF
##                -3.183e+03                -1.005e+03
##                    colorG                    colorH
##                 4.405e+02                 3.880e+03
##                    colorI                    colorJ
##                 7.709e+03                 1.604e+04
##                 clarityIF                 claritySI1
##                 3.125e+03                 1.854e+04
##                claritySI2                 clarityVS1
##                 1.900e+04                 1.517e+04
##                clarityVS2                clarityVVS1
##                 1.589e+04                 1.885e+04
##               clarityVVS2                     depth
```

```
##               1.152e+04               -9.695e+01
##                   table                        x
##               4.760e+01               -1.177e+04
##                       y                        z
##               5.593e+03                1.398e+03
##              I(carat^5)                   I(x^5)
##              -1.285e+00               -2.151e-01
##                  I(y^5)                   I(z^5)
##              -5.606e-04                1.250e-03
##              I(depth^5)               I(table^5)
##              -8.022e-07               -5.144e-07
##           carat:cutGood           carat:cutIdeal
##              -5.969e+02                9.834e+02
##       carat:cutPremium       carat:cutVery Good
##               1.159e+03                4.975e+02
##           carat:colorE            carat:colorF
##               4.196e+02                3.194e+02
##           carat:colorG            carat:colorH
##              -2.446e+02               -9.256e+02
##           carat:colorI            carat:colorJ
##              -1.463e+03               -2.756e+03
##         carat:clarityIF         carat:claritySI1
##               1.100e+04                9.183e+03
##       carat:claritySI2         carat:clarityVS1
##               7.024e+03                1.039e+04
##       carat:clarityVS2        carat:clarityVVS1
##               9.963e+03                1.240e+04
##      carat:clarityVVS2              carat:depth
##               1.163e+04               -6.509e+01
##            carat:table                  carat:x
##               1.491e+01               -2.751e+03
##                 carat:y                  carat:z
##               4.128e+03                7.499e+02
##          cutGood:colorE          cutIdeal:colorE
##              -5.570e+01               -1.176e+02
##       cutPremium:colorE      cutVery Good:colorE
##              -1.514e+02               -1.183e+02
##          cutGood:colorF          cutIdeal:colorF
##              -1.050e+02               -7.233e+01
##       cutPremium:colorF      cutVery Good:colorF
##              -1.420e+02               -7.950e+01
##          cutGood:colorG          cutIdeal:colorG
##               8.177e+00               -4.707e+01
##       cutPremium:colorG      cutVery Good:colorG
##              -9.048e+01               -1.383e+01
##          cutGood:colorH          cutIdeal:colorH
##              -3.971e+01               -1.964e+02
##       cutPremium:colorH      cutVery Good:colorH
##              -3.234e+02               -1.323e+02
##          cutGood:colorI          cutIdeal:colorI
##              -2.386e+02               -4.252e+02
##       cutPremium:colorI      cutVery Good:colorI
##              -5.234e+02               -2.909e+02
##          cutGood:colorJ          cutIdeal:colorJ
```

```
##              -2.870e+02               -4.861e+02
##       cutPremium:colorJ       cutVery Good:colorJ
##              -5.951e+02               -3.342e+02
##       cutGood:clarityIF       cutIdeal:clarityIF
##               1.127e+03                9.441e+02
##    cutPremium:clarityIF    cutVery Good:clarityIF
##               1.287e+03                1.399e+03
##      cutGood:claritySI1      cutIdeal:claritySI1
##              -4.973e+00               -2.839e+02
##   cutPremium:claritySI1   cutVery Good:claritySI1
##               1.409e+02                1.759e+02
##      cutGood:claritySI2      cutIdeal:claritySI2
##              -3.045e+00               -3.170e+02
##   cutPremium:claritySI2   cutVery Good:claritySI2
##               1.953e+02                1.589e+02
##      cutGood:clarityVS1      cutIdeal:clarityVS1
##               1.605e+02               -6.973e+01
##   cutPremium:clarityVS1   cutVery Good:clarityVS1
##               3.873e+02                3.349e+02
##      cutGood:clarityVS2      cutIdeal:clarityVS2
##              -3.393e+00               -2.077e+02
##   cutPremium:clarityVS2   cutVery Good:clarityVS2
##               2.424e+02                2.159e+02
##     cutGood:clarityVVS1     cutIdeal:clarityVVS1
##              -2.014e+02               -4.114e+02
##  cutPremium:clarityVVS1  cutVery Good:clarityVVS1
##               5.178e+01                2.496e+01
##     cutGood:clarityVVS2     cutIdeal:clarityVVS2
##               1.475e+02               -2.017e+01
##  cutPremium:clarityVVS2  cutVery Good:clarityVVS2
##               3.940e+02                3.613e+02
##           cutGood:depth           cutIdeal:depth
##               1.665e+01               -3.731e+01
##        cutPremium:depth       cutVery Good:depth
##               1.777e+00                1.134e+01
##           cutGood:table           cutIdeal:table
##              -6.717e+00               -1.190e+00
##        cutPremium:table       cutVery Good:table
##              -1.147e+01               -8.239e+00
##                cutGood:x                cutIdeal:x
##              -1.839e+03               -2.690e+03
##            cutPremium:x            cutVery Good:x
##              -1.234e+03               -1.452e+03
##                cutGood:y                cutIdeal:y
##               2.286e+03                2.166e+03
##            cutPremium:y            cutVery Good:y
##               8.783e+02                1.542e+03
##                cutGood:z                cutIdeal:z
##              -7.442e+01                8.031e+02
##            cutPremium:z            cutVery Good:z
##               2.079e+02               -1.013e+02
##        colorE:clarityIF         colorF:clarityIF
##              -2.415e+03               -2.825e+03
##        colorG:clarityIF         colorH:clarityIF
```

```
##              -3.719e+03              -4.942e+03
##         colorI:clarityIF         colorJ:clarityIF
##              -5.852e+03              -7.872e+03
##        colorE:claritySI1        colorF:claritySI1
##              -5.529e+01              -2.202e+02
##        colorG:claritySI1        colorH:claritySI1
##              -5.330e+02              -1.110e+03
##        colorI:claritySI1        colorJ:claritySI1
##              -1.729e+03              -3.115e+03
##        colorE:claritySI2        colorF:claritySI2
##               3.439e+01              -6.759e+01
##        colorG:claritySI2        colorH:claritySI2
##              -2.360e+02              -4.829e+02
##        colorI:claritySI2        colorJ:claritySI2
##              -7.834e+02              -1.609e+03
##        colorE:clarityVS1        colorF:clarityVS1
##              -6.222e+01              -2.171e+02
##        colorG:clarityVS1        colorH:clarityVS1
##              -6.732e+02              -1.836e+03
##        colorI:clarityVS1        colorJ:clarityVS1
##              -2.691e+03              -4.370e+03
##        colorE:clarityVS2        colorF:clarityVS2
##              -8.473e+01              -1.632e+02
##        colorG:clarityVS2        colorH:clarityVS2
##              -6.272e+02              -1.696e+03
##        colorI:clarityVS2        colorJ:clarityVS2
##              -2.482e+03              -4.098e+03
##       colorE:clarityVVS1       colorF:clarityVVS1
##              -2.614e+02              -4.425e+02
##       colorG:clarityVVS1       colorH:clarityVVS1
##              -1.216e+03              -2.368e+03
##       colorI:clarityVVS1       colorJ:clarityVVS1
##              -3.270e+03              -5.373e+03
##       colorE:clarityVVS2       colorF:clarityVVS2
##              -3.028e+02              -5.021e+02
##       colorG:clarityVVS2       colorH:clarityVVS2
##              -1.142e+03              -2.416e+03
##       colorI:clarityVVS2       colorJ:clarityVVS2
##              -3.305e+03              -5.318e+03
##             colorE:depth             colorF:depth
##                6.203e+01               2.910e+01
##             colorG:depth             colorH:depth
##                2.845e+01               3.675e+00
##             colorI:depth             colorJ:depth
##               -1.841e+01              -1.096e+02
##             colorE:table             colorF:table
##                1.026e+01               1.164e+01
##             colorG:table             colorH:table
##                1.004e+01               1.713e+01
##             colorI:table             colorJ:table
##                3.612e+00               2.216e+00
##                 colorE:x                 colorF:x
##                2.295e+02               7.803e+01
##                 colorG:x                 colorH:x
```

```
##               2.701e+02               5.847e+02
##                colorI:x                colorJ:x
##               5.031e+02               1.209e+03
##                colorE:y                colorF:y
##               5.484e+02               4.640e+02
##                colorG:y                colorH:y
##              -3.194e+01              -5.299e+02
##                colorI:y                colorJ:y
##              -6.234e+02              -2.714e+03
##                colorE:z                colorF:z
##              -1.702e+03              -1.378e+03
##                colorG:z                colorH:z
##              -1.118e+03              -1.154e+03
##                colorI:z                colorJ:z
##              -1.086e+03               1.032e+03
##          clarityIF:depth         claritySI1:depth
##               5.360e+01              -1.481e+02
##         claritySI2:depth         clarityVS1:depth
##              -1.616e+02              -1.128e+02
##         clarityVS2:depth        clarityVVS1:depth
##              -1.220e+02              -1.604e+02
##        clarityVVS2:depth          clarityIF:table
##              -6.400e+01              -7.083e+01
##         claritySI1:table         claritySI2:table
##              -7.071e+01              -7.030e+01
##         clarityVS1:table         clarityVS2:table
##              -8.089e+01              -7.177e+01
##        clarityVVS1:table        clarityVVS2:table
##              -8.668e+01              -6.994e+01
##               clarityIF:x              claritySI1:x
##                3.981e+03               1.224e+03
##              claritySI2:x              clarityVS1:x
##              -7.343e+01               1.195e+03
##              clarityVS2:x             clarityVVS1:x
##                1.365e+03               1.523e+03
##             clarityVVS2:x              clarityIF:y
##                1.711e+03              -3.021e+03
##              claritySI1:y              claritySI2:y
##              -2.923e+03              -1.717e+03
##              clarityVS1:y              clarityVS2:y
##              -2.404e+03              -2.854e+03
##             clarityVVS1:y             clarityVVS2:y
##              -3.075e+03              -2.344e+03
##              clarityIF:z              claritySI1:z
##              -2.892e+03              -2.474e+02
##              claritySI2:z              clarityVS1:z
##                2.451e+02              -5.272e+02
##              clarityVS2:z             clarityVVS1:z
##              -2.430e+02              -3.511e+01
##             clarityVVS2:z               depth:table
##              -1.451e+03               1.161e+00
##                 depth:x                 depth:y
##                1.470e+02              -9.782e+01
##                 depth:z                 table:x
```

```
##               -1.409e+01                7.436e+01
##                  table:y                  table:z
##               -1.023e+02                3.226e+01
##                      x:y                      x:z
##                6.520e+02               -5.450e+02
##                      y:z
##                2.005e+02
```

Report R^2, RMSE, the standard error of the residuals (s_e) but you do not need to report b.

```
rsq8 = summary(reg8)$r.squared
rsq8
```

```
## [1] 0.9716716
```

```
rmse8 = summary(reg8)$sigma
rmse8
```

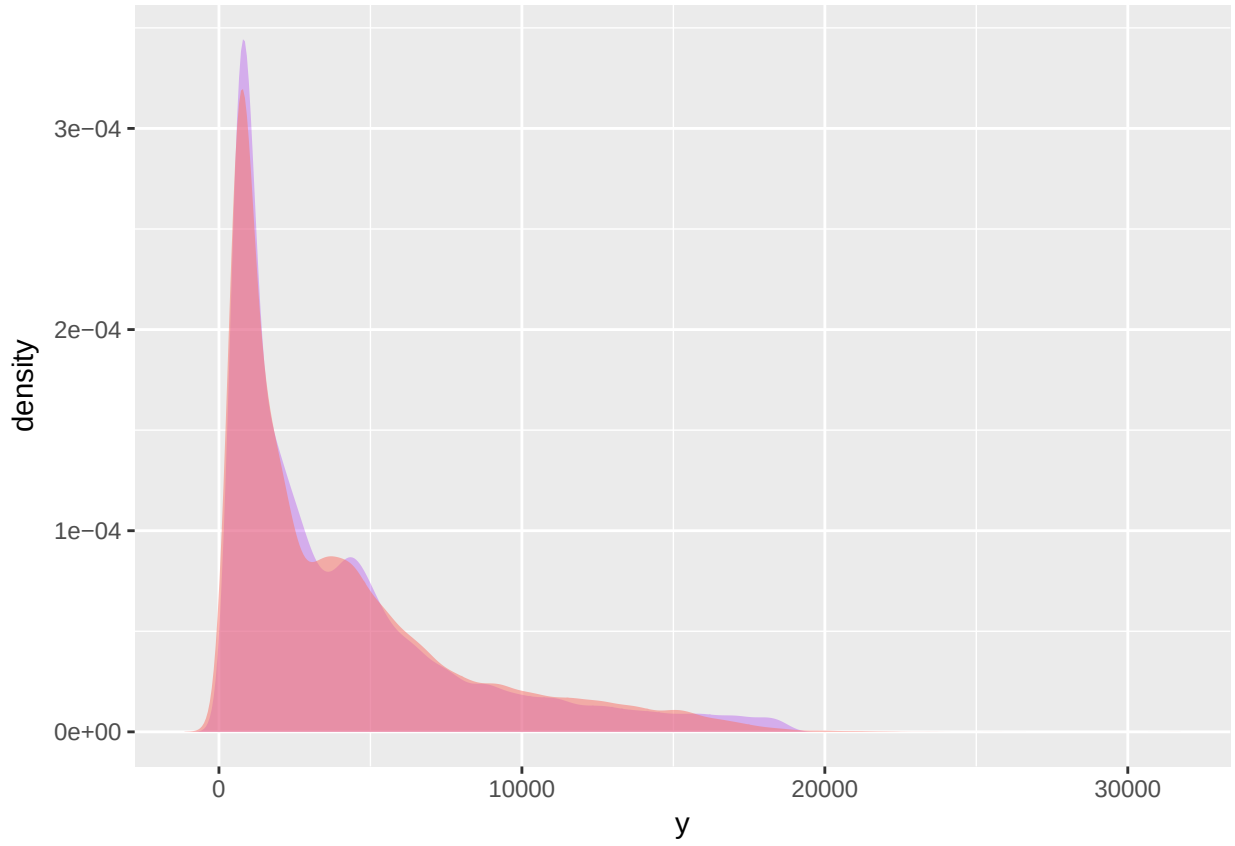
```
## [1] 672.9627
```

```
se8 = sd(reg8$residuals)
se8
```

```
## [1] 671.4638
```

Create an illustration of y vs. $\hat{y}$.

```
y = diamonds$price
y_hat = as.vector(y - residuals(reg8))
ggplot() +
  stat_density(aes(x = y), fill = "purple", alpha = 0.3) +
  stat_density(aes(x = y_hat, fill = "green", alpha = 0.3)) +
  theme(legend.position = "none")
```



How many diamonds have predictions that are wrong by $1,000 or more ?

```
wrong_predict = sum(abs(reg8$residuals) >= 1000)
wrong_predict
```

```
## [1] 4639
```

R^2 now is very high and very impressive. But is RMSE impressive? Think like someone who is actually using this model to e.g. purchase diamonds.

RMSE is not impressive here because according to the empirical rule, there is 95% confidence that our diamond price falls between ~$1300 below the average and ~1300 above the average. That is a large range where the seller can make all the difference (i.e. whether a person pays below market value or not).

What is the degrees of freedom in this model?

```
dfreg8 = length(reg8$coefficients)
dfreg8
```

```
## [1] 241
```

Do you think g is close to h^* in this model? Yes / no and why?

The difference between g and h^* is known as parameter estimation error when $f \notin \mathcal{H}$. g is not as close to h^* in this model as can be because there may be a decent amount of nonsense predictors, like the "fifth power" polynomials (in the many degrees of freedom the model has) which are fitting the errors, leading to overfitting.

Do you think g is close to f in this model? Yes / no and why?

The difference between g and f is known as the parameter estimation error when $f \in \mathcal{H}$, so since we are

talking about g relative to f, g may be closer because we aren't restricted to lines anymore, so maybe those "fifth" power polynomials are getting us slightly closer to f.

What more degrees of freedom can you add to this model to make g closer to f?

The most degrees of freedom we could theoretically add while still being functional would be such that the total number will equal to n-1 degrees of freedom. However, if we keep adding more degrees, we will continue adding to overfitting.

Even if you allowed for so much expressivity in $\mathcal{H}$ that f was an element in it, there would still be error due to ignorance of relevant information that you haven't measured. What information do you think can help? This is not a data science question - you have to think like someone who sells diamonds.

There are many practical factors that influence diamond prices like supply and demand, as well as cultural norms. Places where diamonds are not as "revered" will be lower in price compared to places where they are highly desired because simply put, if people truly want them, they may be willing to pay more for it so that they can have it.

9. Validate the model in #8 by reserving 10% of $\mathbb{D}$ as test data. Report oos standard error of the residuals

```
n = nrow(diamonds)
K = 10
test_indices = sample(1 : n, size = n * 1 / K)
master_train_indices = setdiff(1 : n, test_indices)
select_indices = sample(master_train_indices, size = n * 1 / K)
train_indices = setdiff(master_train_indices, select_indices)
rm(master_train_indices)

diamonds_train = diamonds[train_indices, ]
diamonds_select = diamonds[select_indices, ]
diamonds_test = diamonds[test_indices, ]
rm(test_indices, select_indices, train_indices)
y_select = diamonds_select$price #the true prices
reg9 = lm(price ~ . * . + I(carat^5) + I(x^5) + I(y^5) + I(z^5) + I(depth^5) + I(table^5), data = diamo
yhat_select_reg9 = predict(reg9, diamonds_select)
s_e_s = sd(yhat_select_reg9 - y_select)
s_e_s
```

```
## [1] 281262.6
```

Compare the oos standard error of the residuals to the standard error of the residuals you got in #8 (i.e. the in-sample estimate). Do you think there's overfitting?

I think there is some very slight overfitting because the oos standard error of the residuals is 680.0143, which is a little higher than the in-sample standard error of the residuals which is 671.4638. Predicting on the selection set was worse than making a model with all of the data, which signals some overfitting. However, this is not too terrible considering.

Extra-credit: validate the model via cross validation.

```
#TO-DO if you want extra credit
```

Is this result much different than the single validation? And, again, is there overfitting in this model?

** TO-DO

10. The following code (from plec 14) produces a response that is the result of a linear model of one predictor and random ϵ.

```
rm(list = ls())
set.seed(1003)
```

```
n = 100
beta_0 = 1
beta_1 = 5
xmin = 0
xmax = 1
x = runif(n, xmin, xmax)
#best possible model
h_star_x = beta_0 + beta_1 * x

#actual data differs due to information we don't have
epsilon = rnorm(n)
y = h_star_x + epsilon
```

We then add fake predictors. For instance, here is the model with the addition of 2 fake predictors:

```
p_fake = 2
X = matrix(c(x, rnorm(n * p_fake)), ncol = 1 + p_fake)
mod = lm(y ~ X)
```

Using a test set hold out, find the number of fake predictors where you can reliably say "I overfit". Some example code is below that you may want to use:

```
K = 5
test_indices = sample(1 : n, 1 / K * n) #amount for test
train_indices = setdiff(1 : n, test_indices) #amount out for train
total_oose = c(2) #need to initialize total_oose to something or else error
for(i in 1:80){ #80 b/c training data used
  X = cbind(X, rnorm(n))
  X_train = X[train_indices, ]
  y_train = y[train_indices]
  X_test = X[test_indices, ]
  y_test = y[test_indices]
  mod = lm(y_train ~., data.frame(X_train)) #error - must be data.frame, needs ~.,
  y_hat_oos = predict(mod, data.frame(X_test)) #error - must be data.frame
  oos_residuals = y_test - y_hat_oos #test data minus prediction
  1 - sum(oos_residuals^2) / sum((y_test - mean(y_test))^2) #from lec notes
  total_oose = cbind(total_oose, sd(oos_residuals)) #comparing the two
}
```

```
## Warning in predict.lm(mod, data.frame(X_test)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(mod, data.frame(X_test)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(mod, data.frame(X_test)): prediction from a rank-
## deficient fit may be misleading

## Warning in predict.lm(mod, data.frame(X_test)): prediction from a rank-
## deficient fit may be misleading
```

```
min(total_oose)
```

```
## [1] 1.172882
```

```
#if output cannot be seen, r output = 2 here
```