HW03p

Chaim Eisenbach

April 13, 2018

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

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

```
#TO-DO
pacman::p_load(ggplot2, quantreg)
```

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.

```
data(diamonds)
?diamonds
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 ...
```

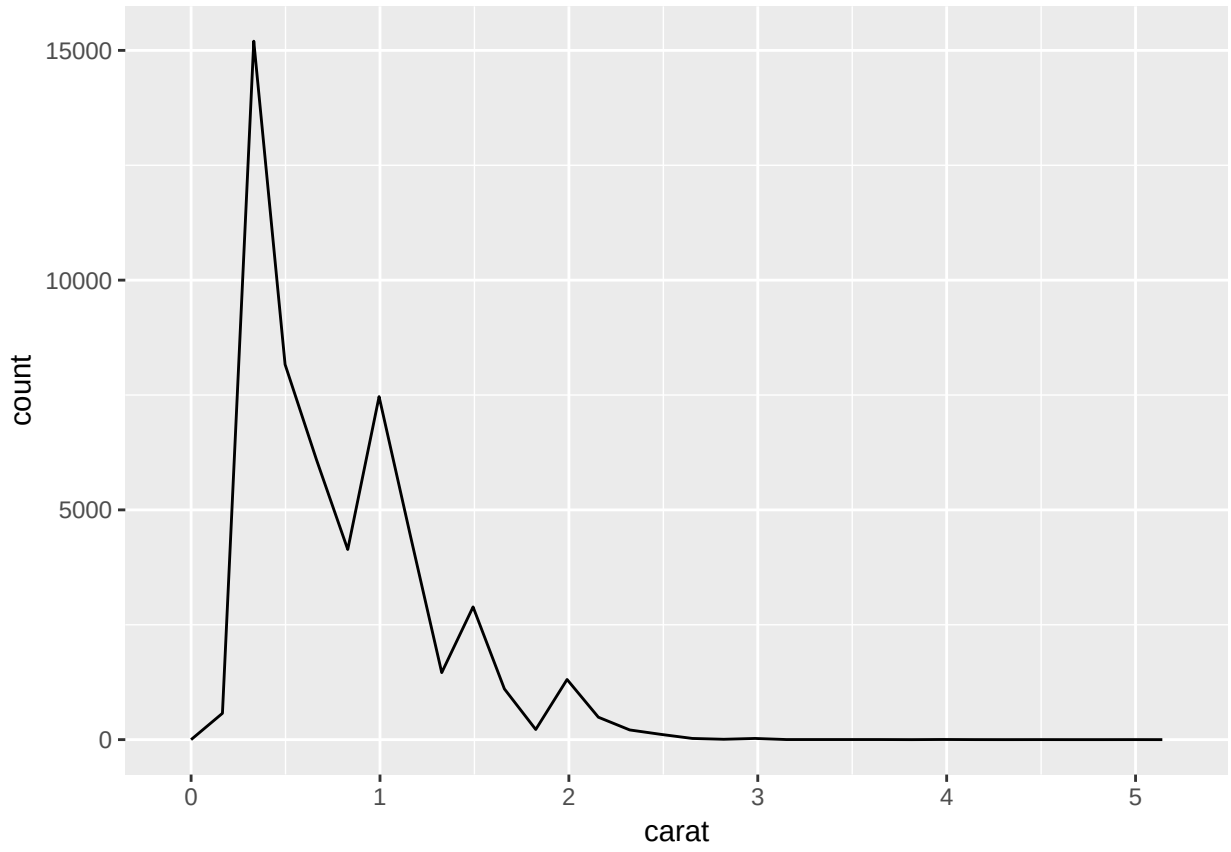
What is n, p, what do the features mean, what is the most likely response metric and why? n is 53940 and p is 10 Response metric would be price

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.

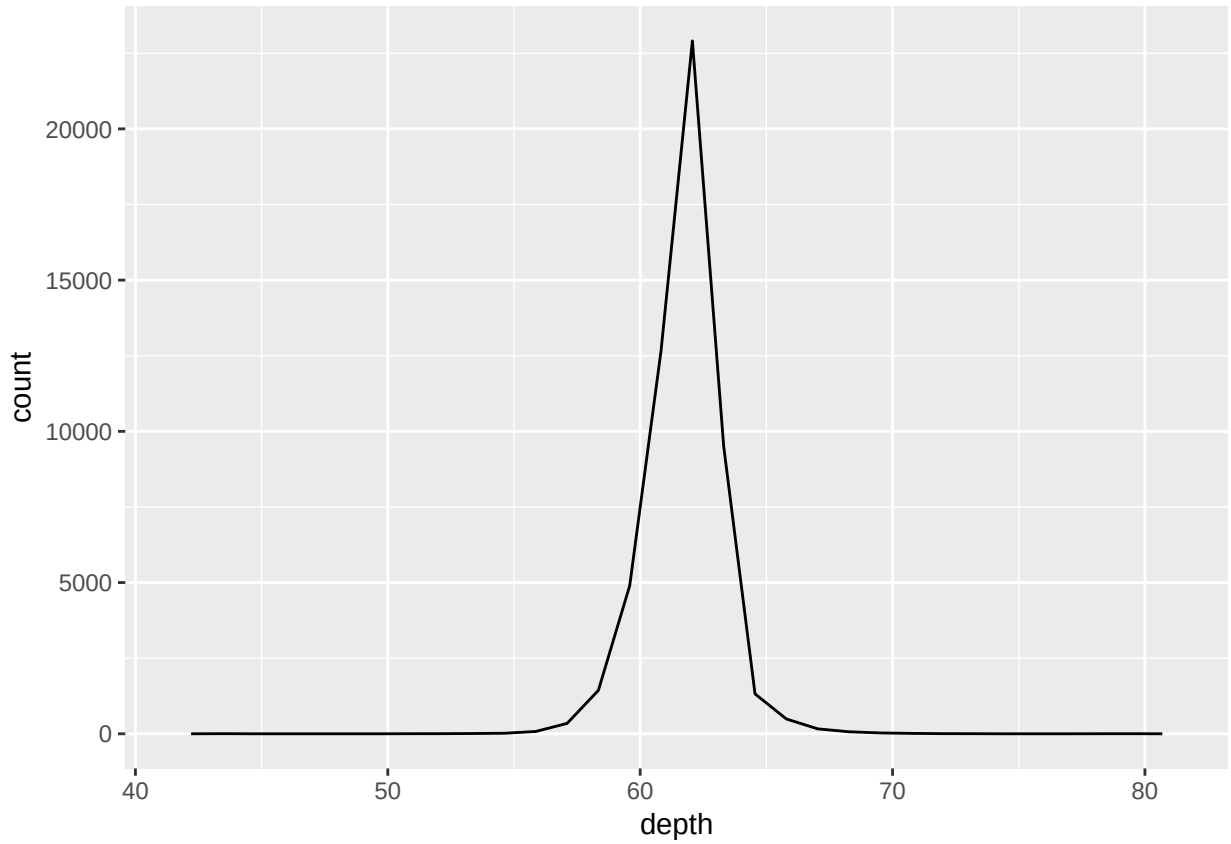
```
#TO-DO
ggplot(diamonds, aes(carat)) + geom_freqpoly()
```

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



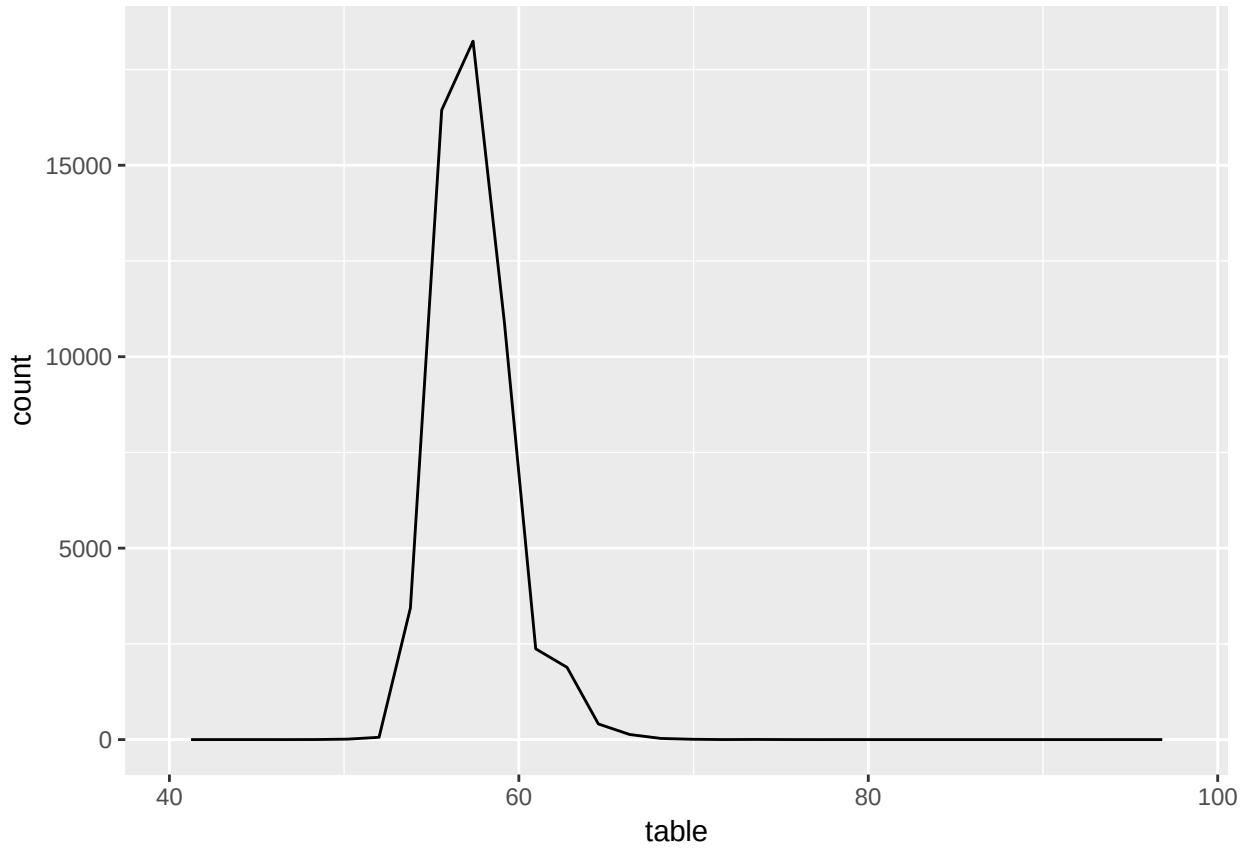

```
ggplot(diamonds, aes(depth)) + geom_freqpoly()
```

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



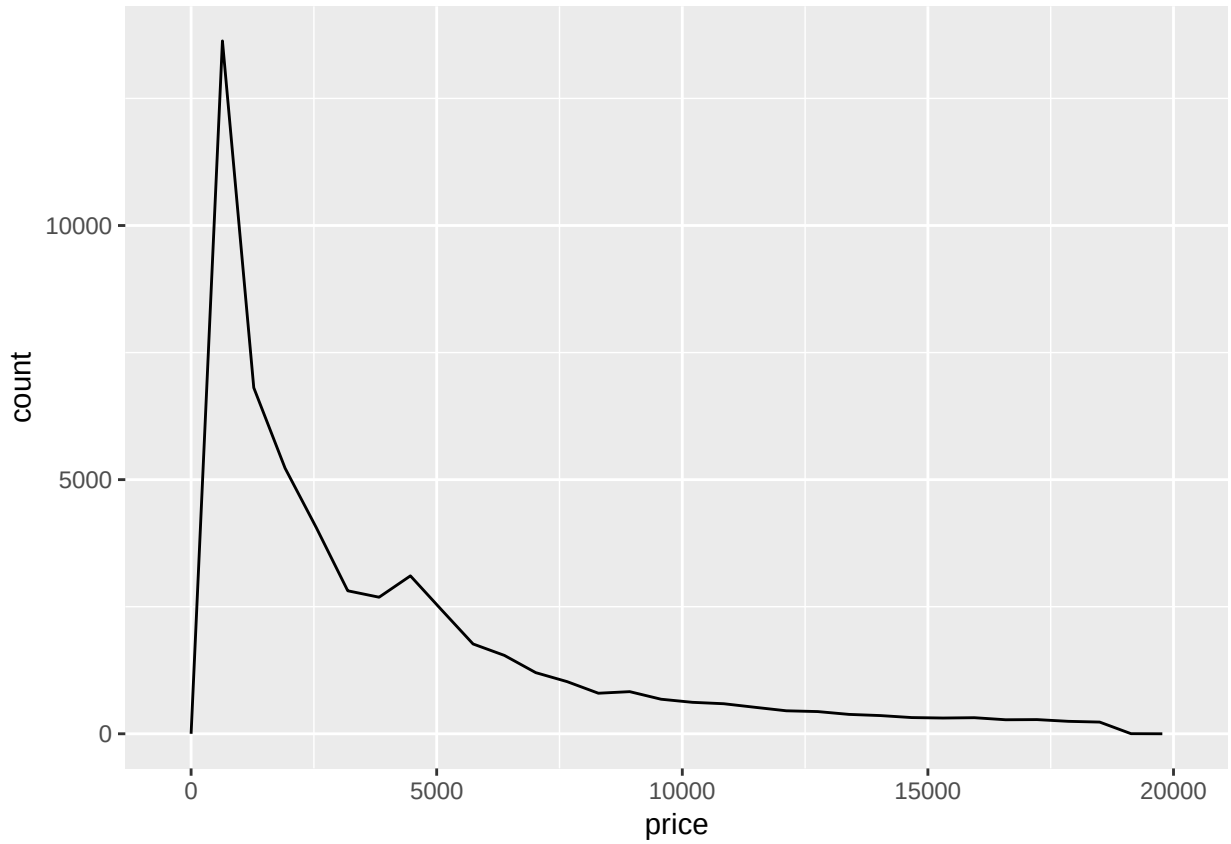

```
ggplot(diamonds, aes(table)) + geom_freqpoly()
```

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



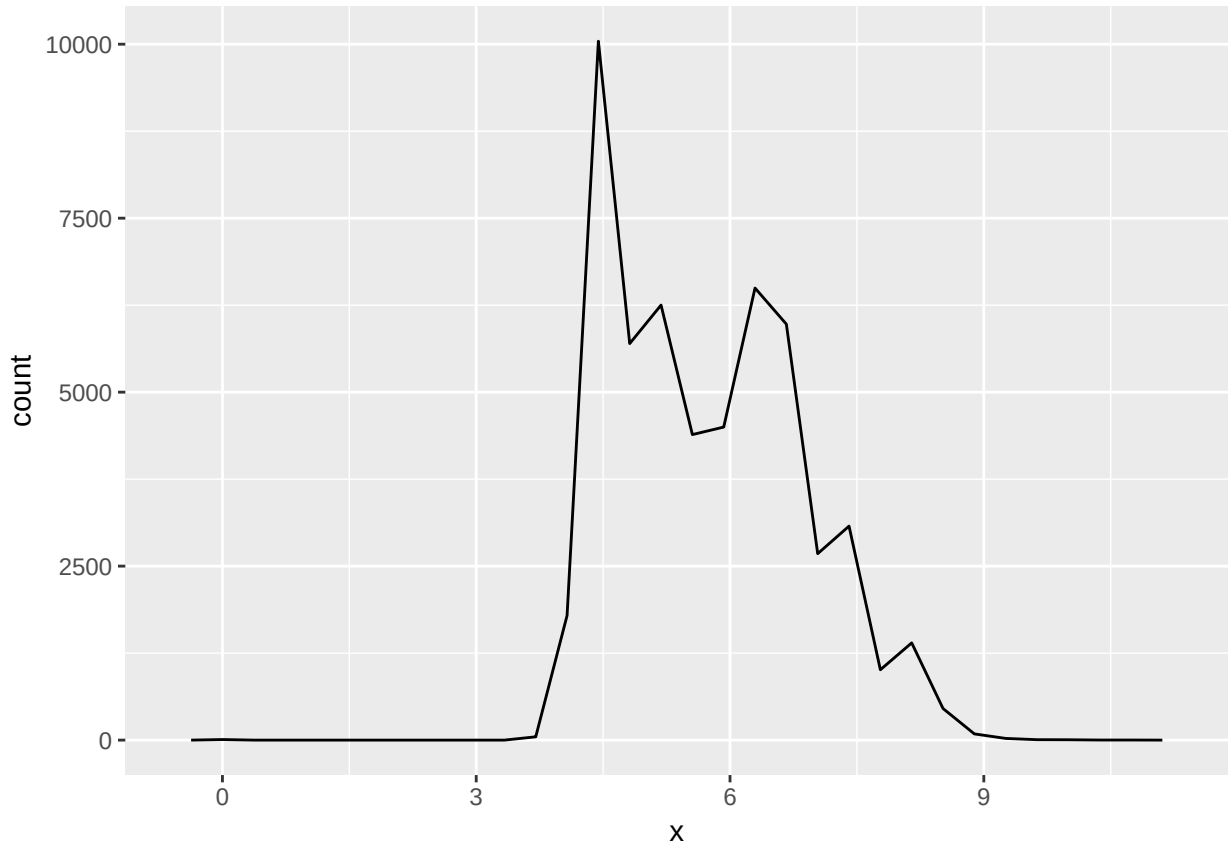

```
ggplot(diamonds, aes(price)) + geom_freqpoly()
```

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



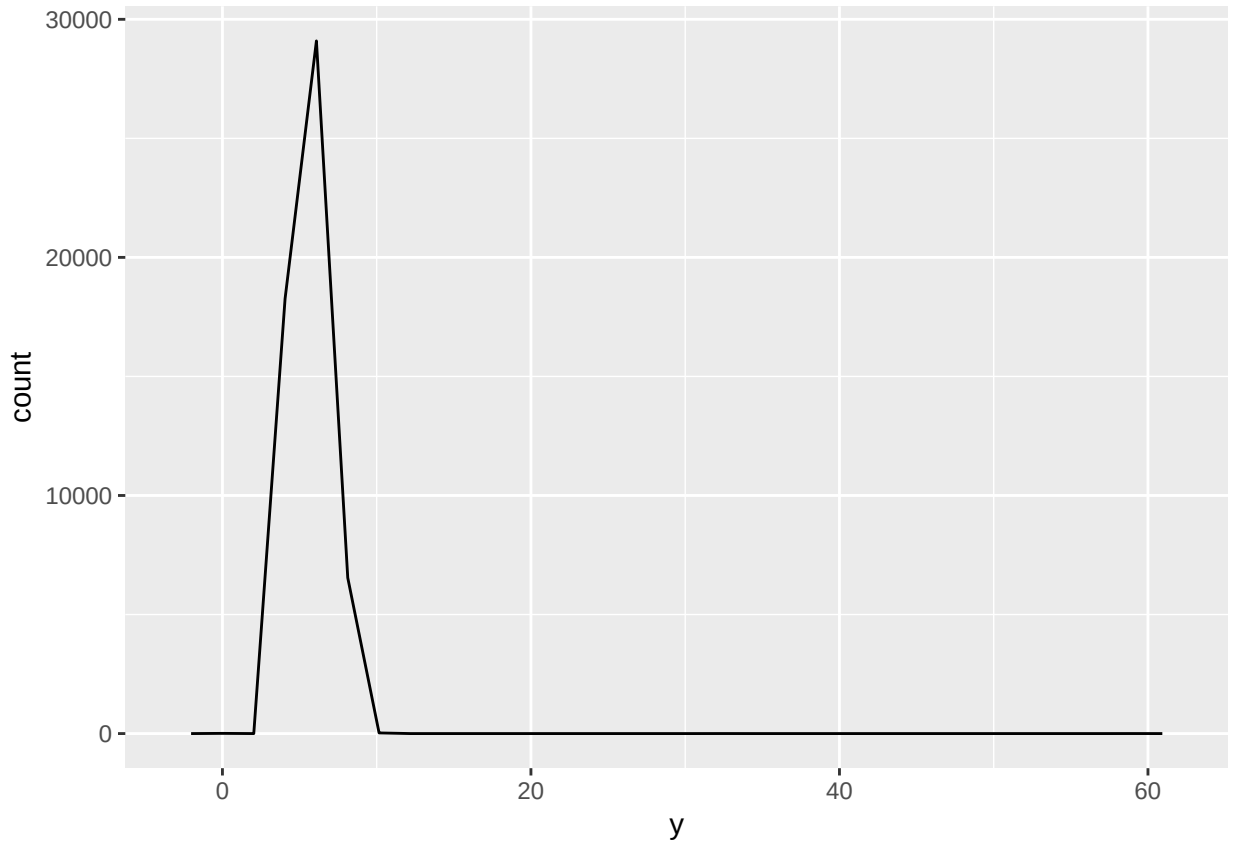

```
ggplot(diamonds, aes(x)) + geom_freqpoly()
```

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



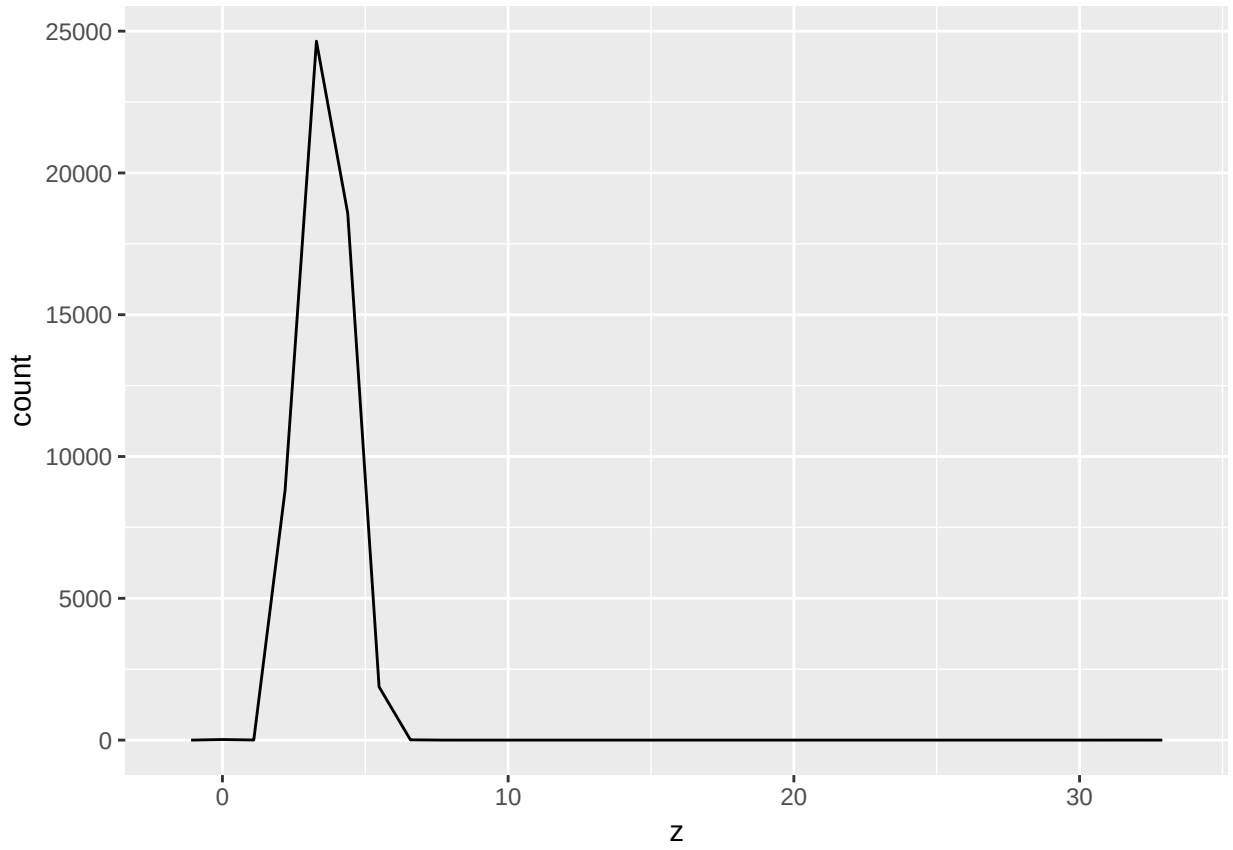

```
ggplot(diamonds, aes(y)) + geom_freqpoly()
```

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

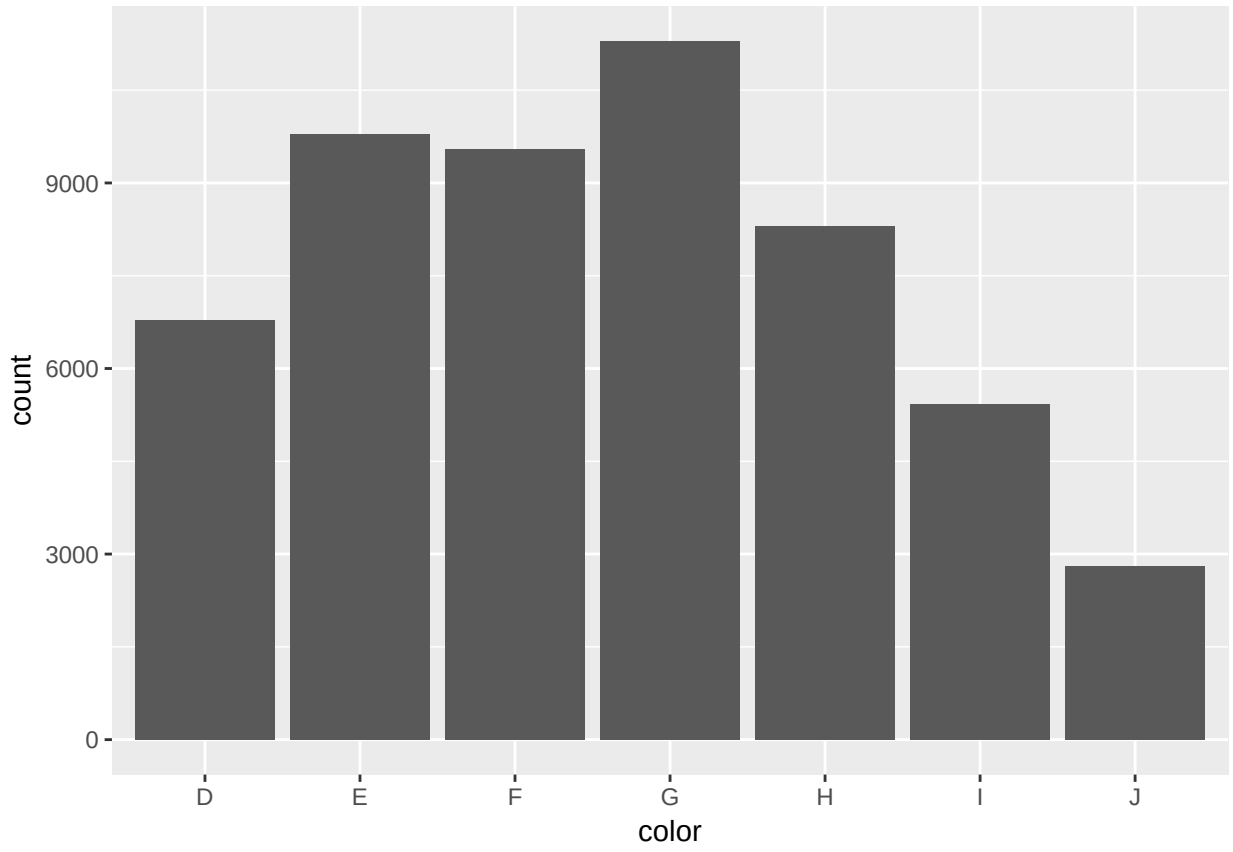



```
ggplot(diamonds, aes(z)) + geom_freqpoly()
```

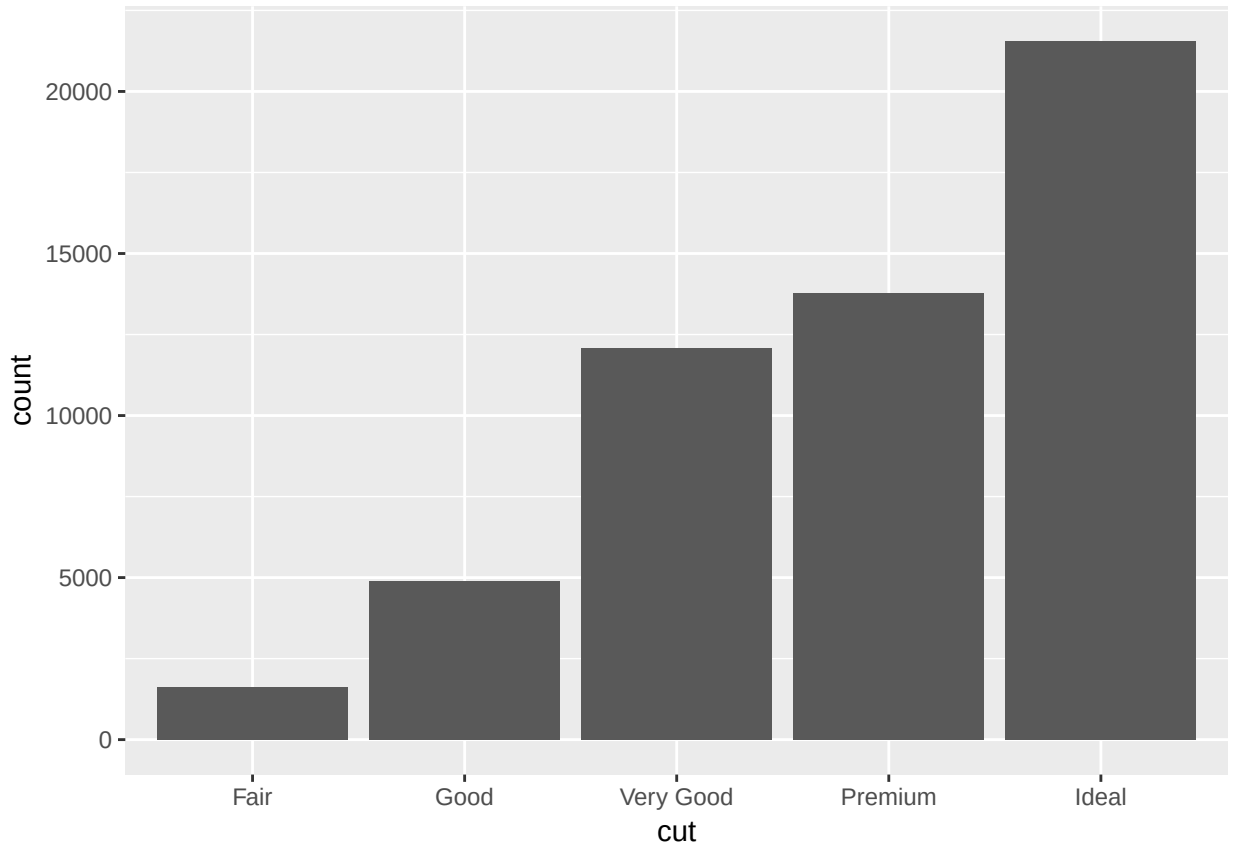
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



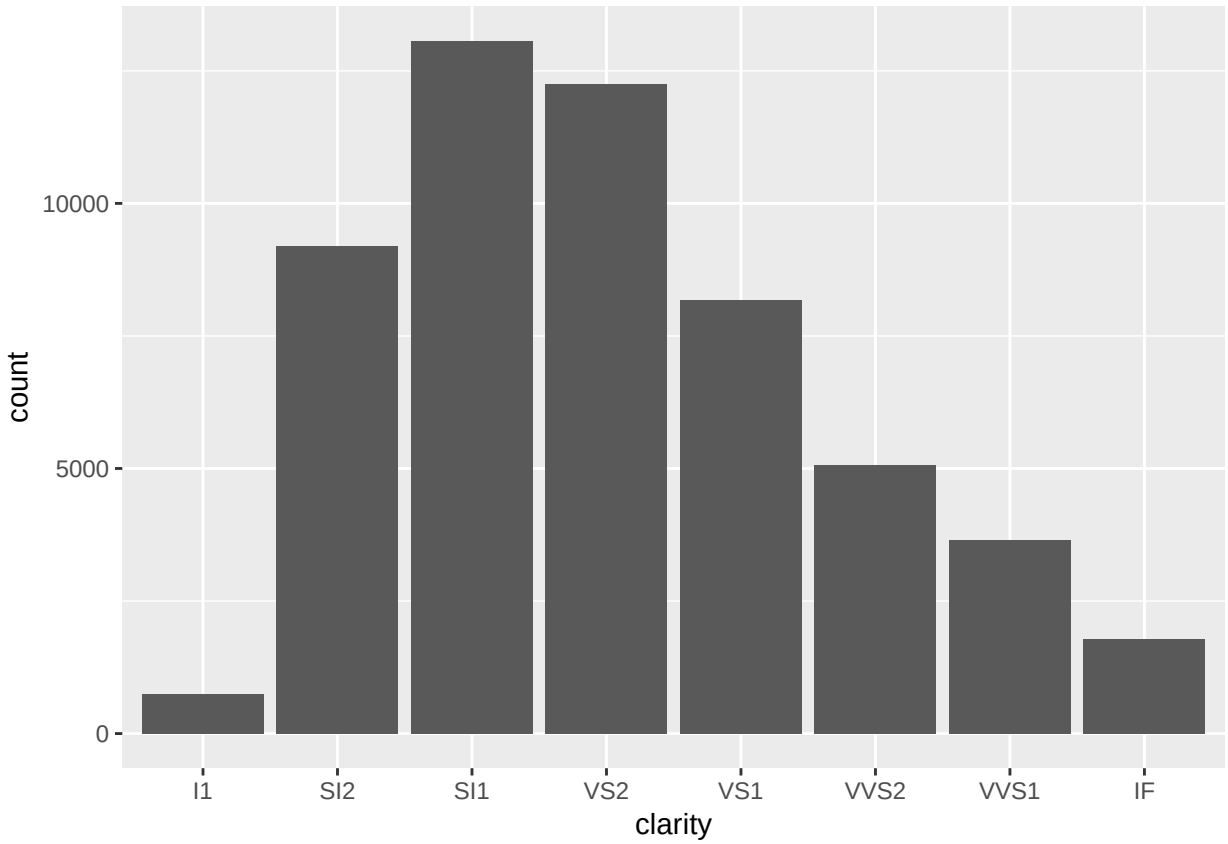

```
ggplot(diamonds, aes(color)) + geom_bar()
```




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



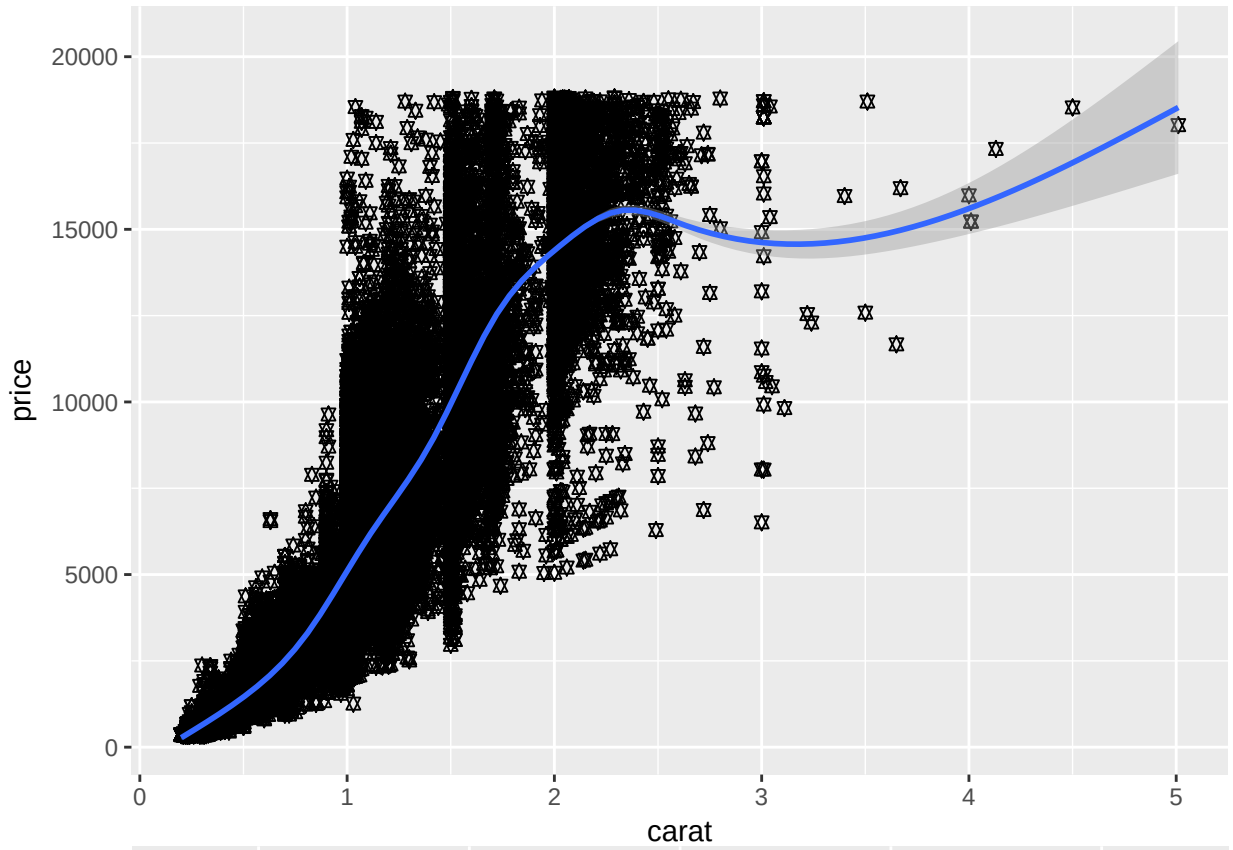

```
ggplot(diamonds, aes(clarity)) + geom_bar()
```

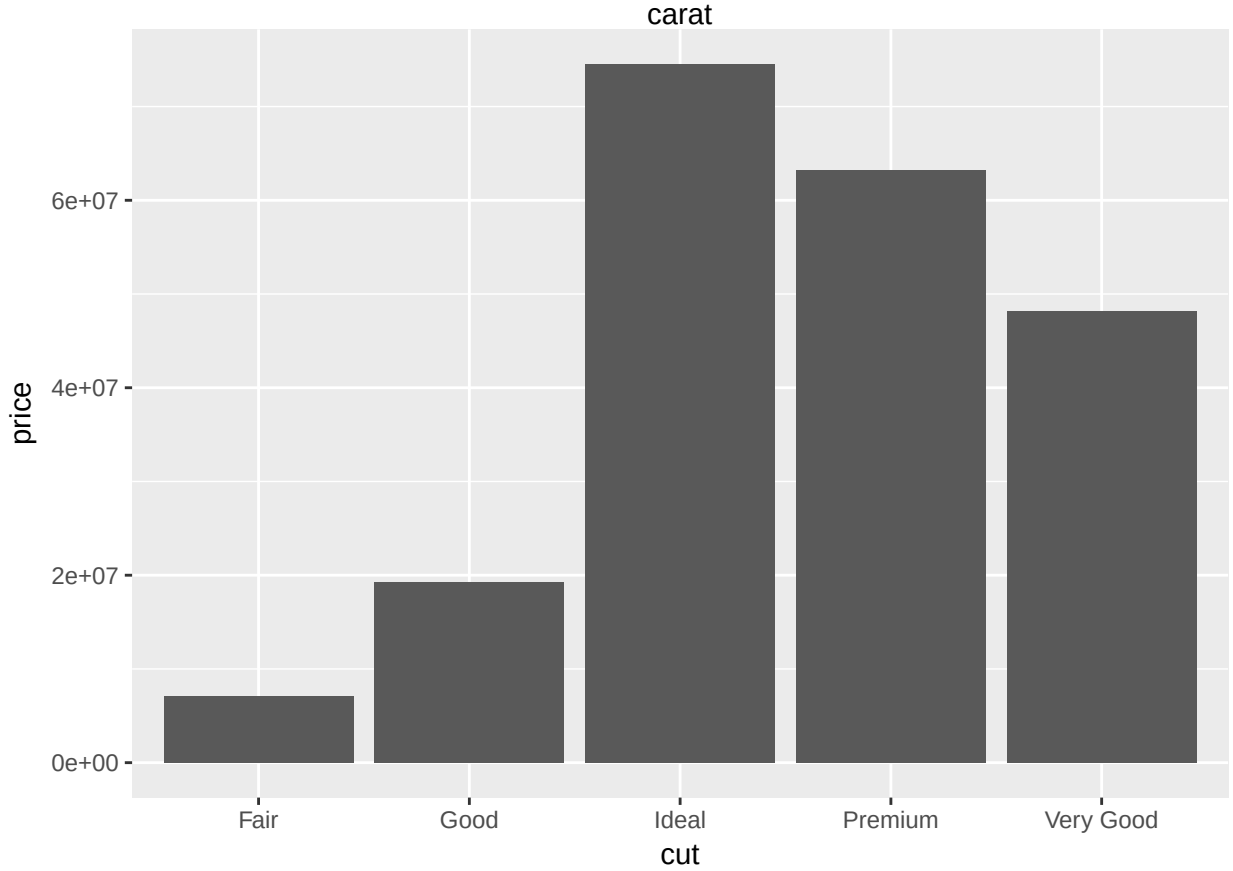


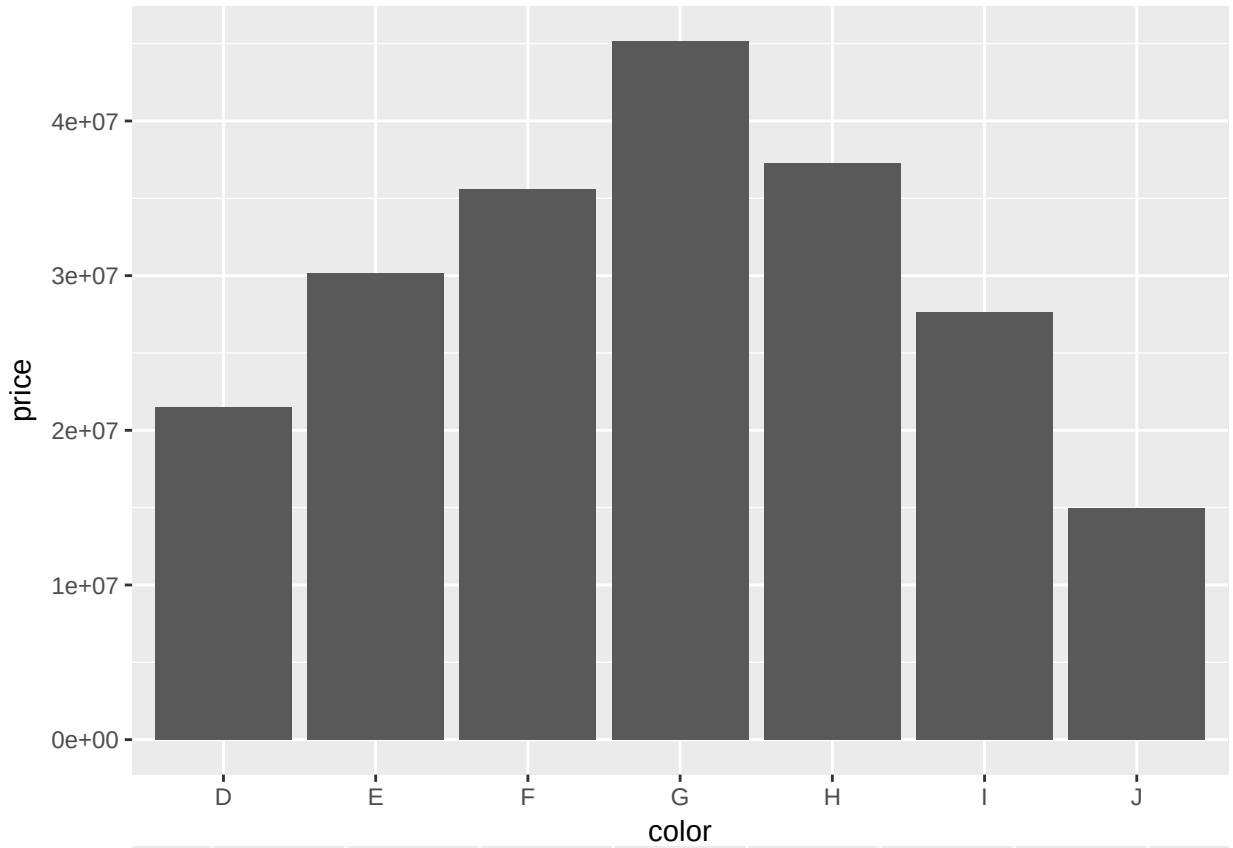
Use `ggplot` to look at the bivariate distributions of the response versus *all* predictors. Make sure you handle categorical predictors differently from continuous predictors. This time employ a for loop when an logic that handles the predictor type.

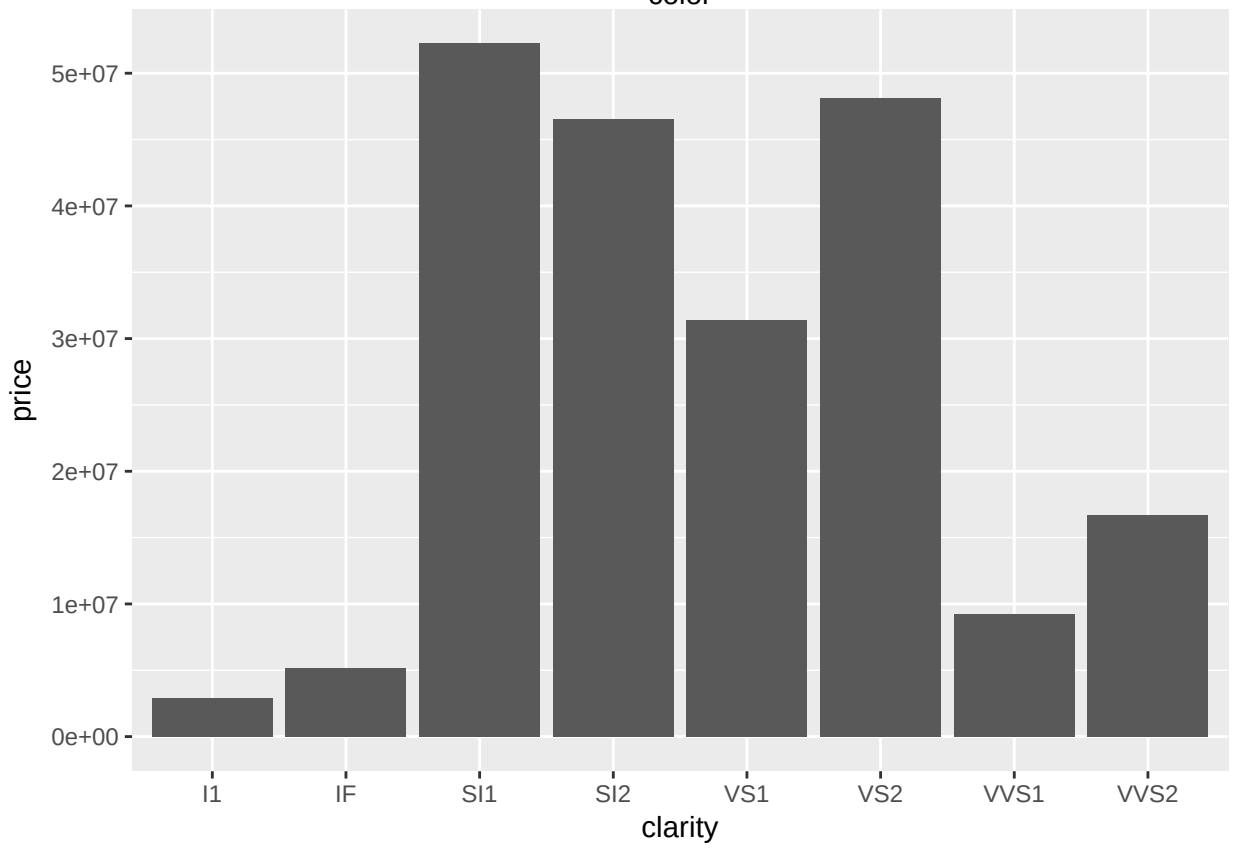
```
#TO-DO
for (i in 1:10){
  j = as.vector(as.matrix(diamonds[,i]))

  if(typeof(j) == "character"){
   print(ggplot(diamonds,aes(x = j, y = price)) +  xlab(colnames(diamonds[,i]))+ geom_bar(stat = "ident
  } else { print(ggplot(diamonds,aes(x = j, y = price)) + xlab(colnames(diamonds[,i]))+geom_point(shape
  }

}
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

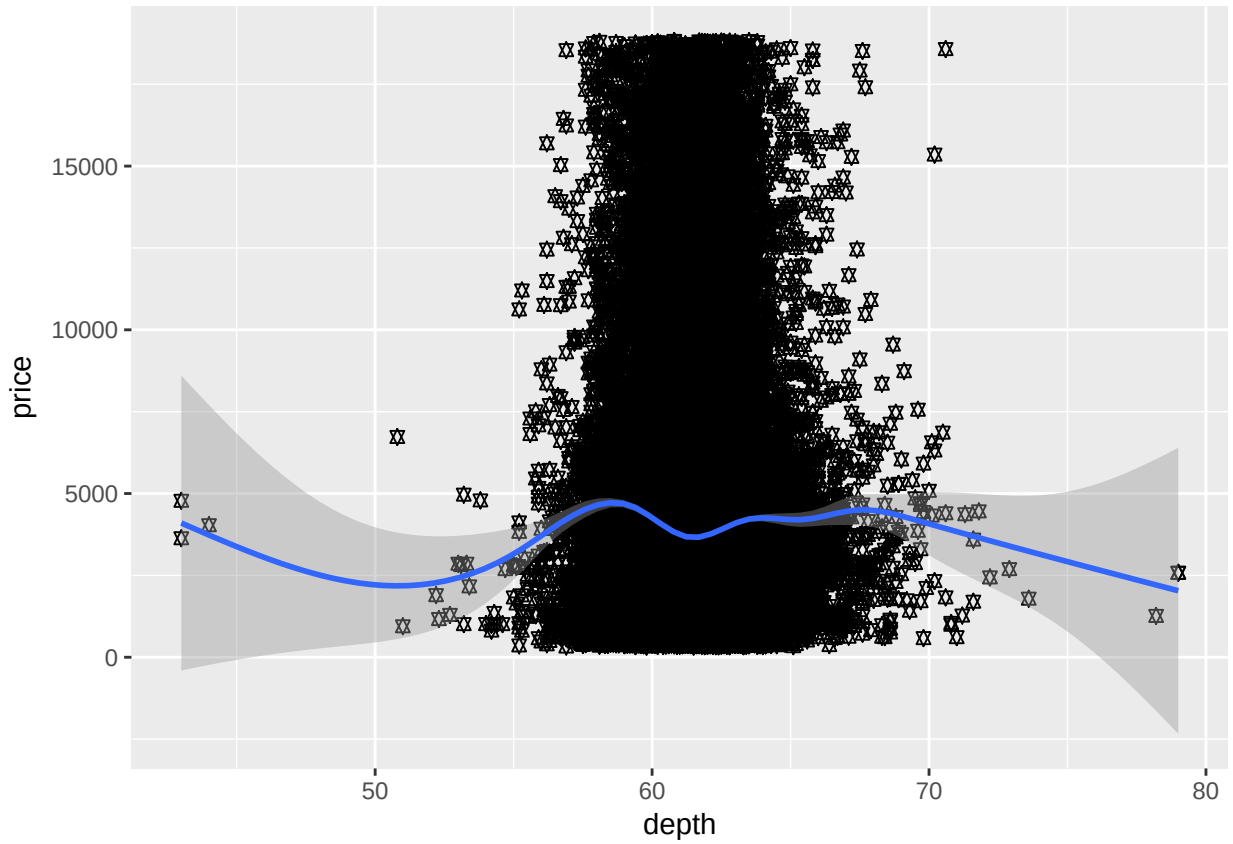

price
20000
15000
10000
5000
0
0
1
2
3
4
5
carat


price
6e+07
4e+07
2e+07
0e+00
Fair
Good
Ideal
Premium
Very Good
cut

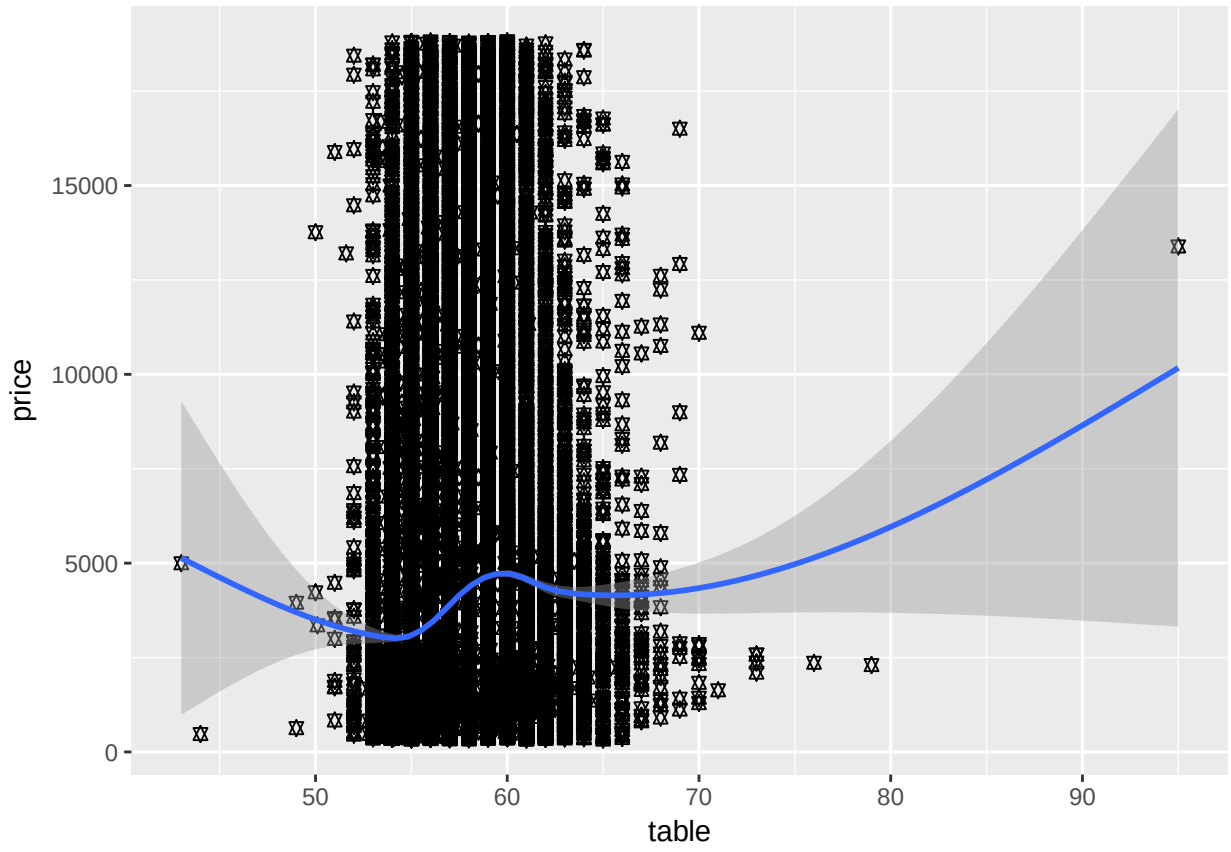




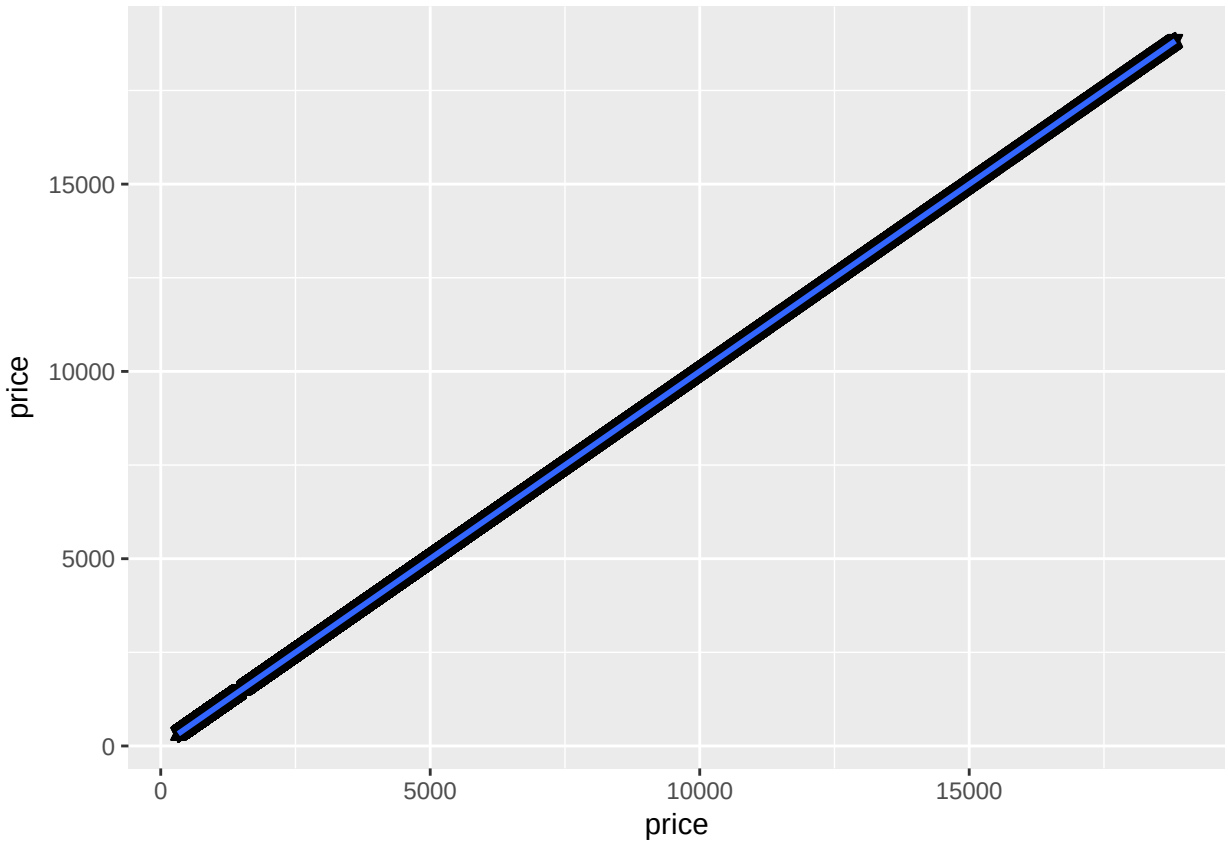

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



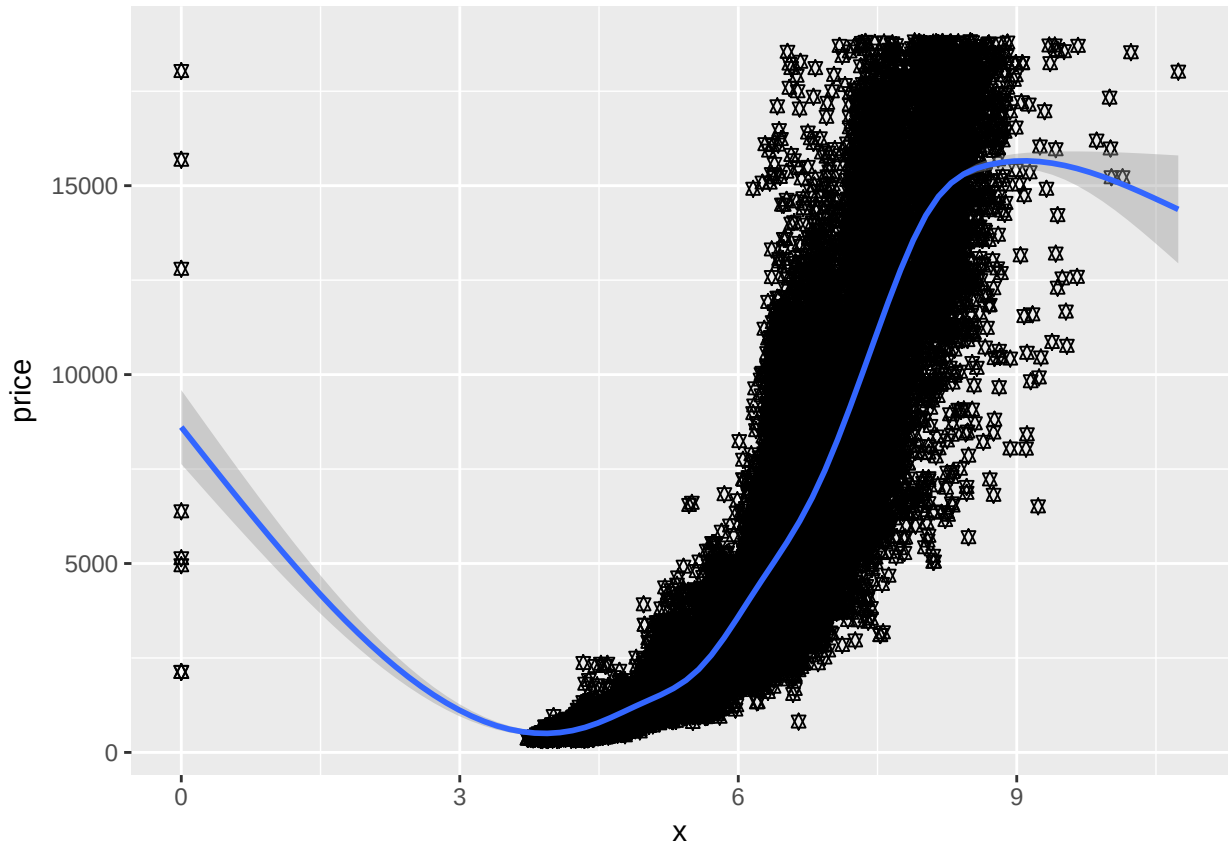

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



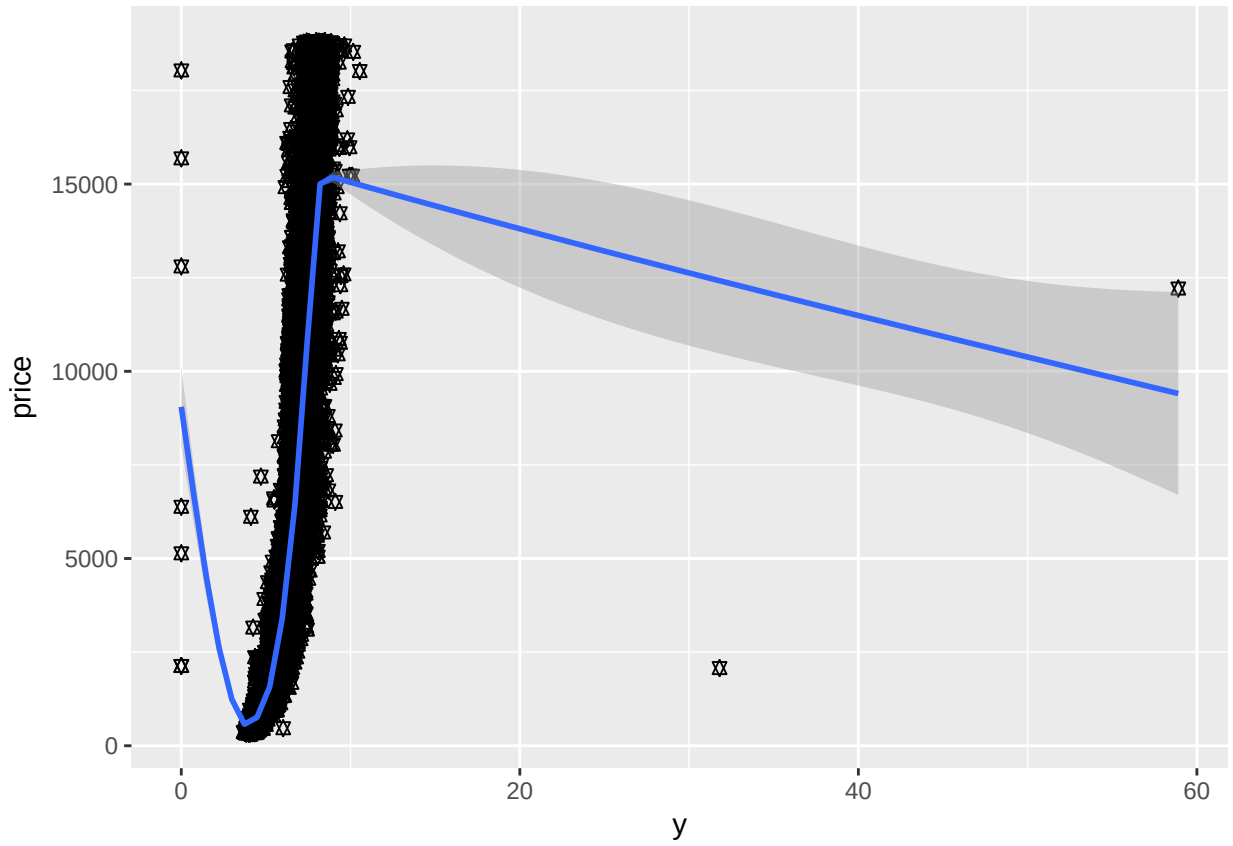

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



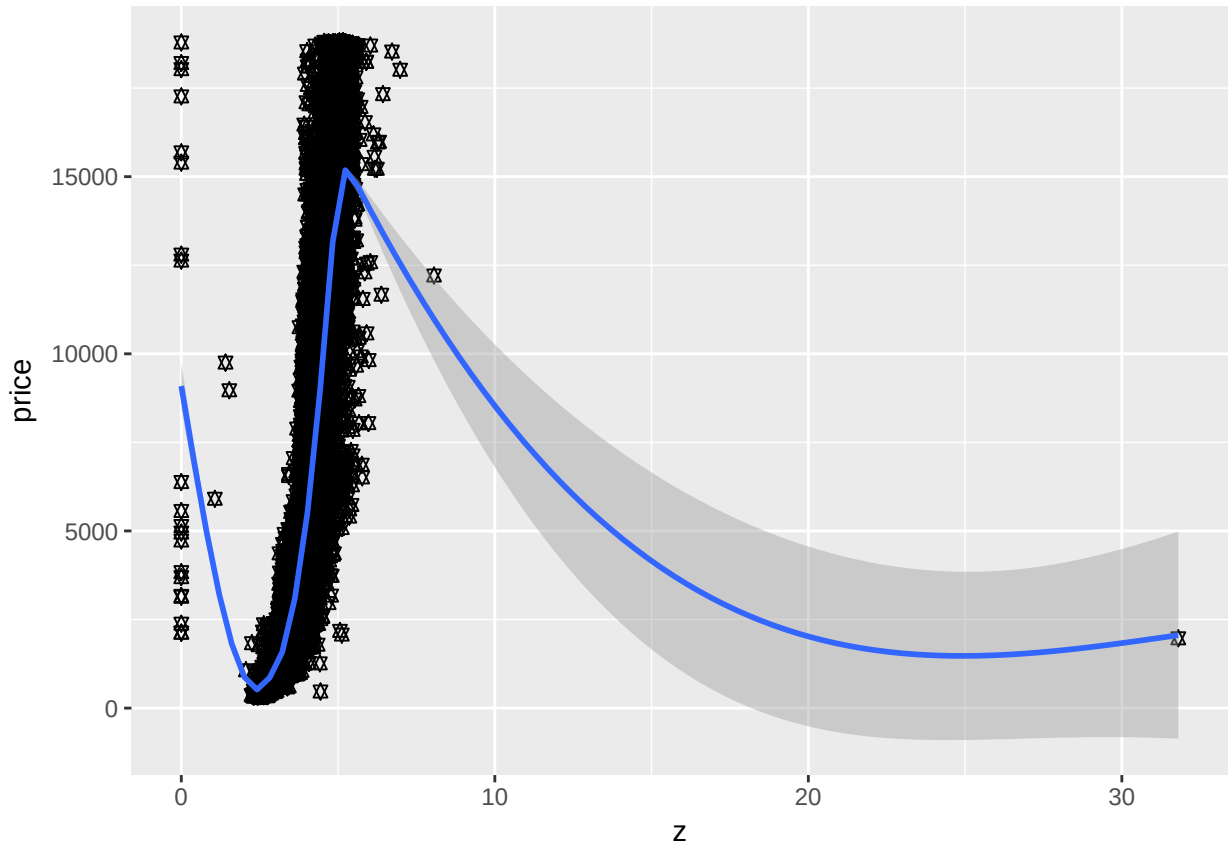

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

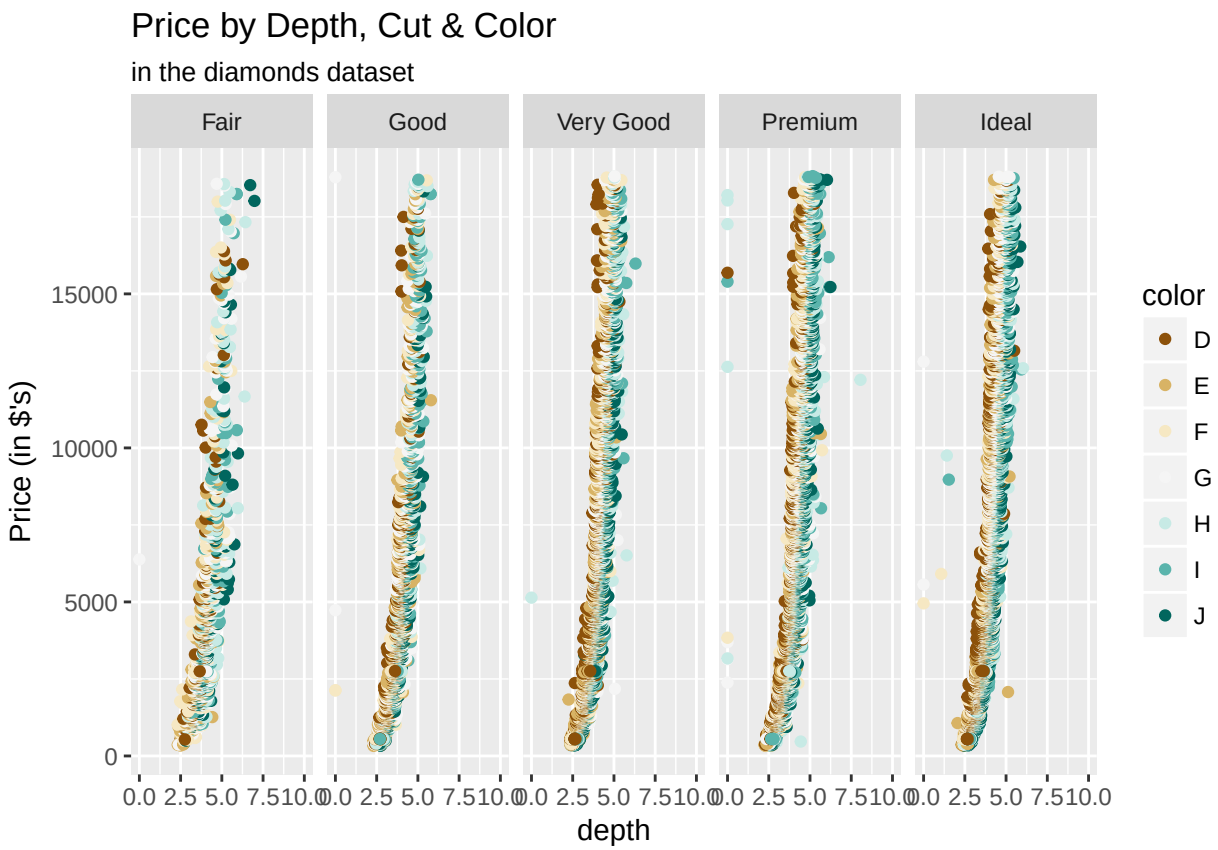


Does depth appear to be mostly independent of price? Yes **TO-DO

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

```
#TO-DO

graph1 = ggplot(diamonds, aes(x = z, y = price)) +
  ggtitle("Price by Depth, Cut & Color", subtitle = "in the diamonds dataset") +
  ylab("Price (in $'s)") +
  xlab("depth") +
   xlim(0, 10) +

  geom_point(aes(col = color)) +scale_color_brewer(type = "div")+
  facet_grid(. ~ cut)

graph1
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```



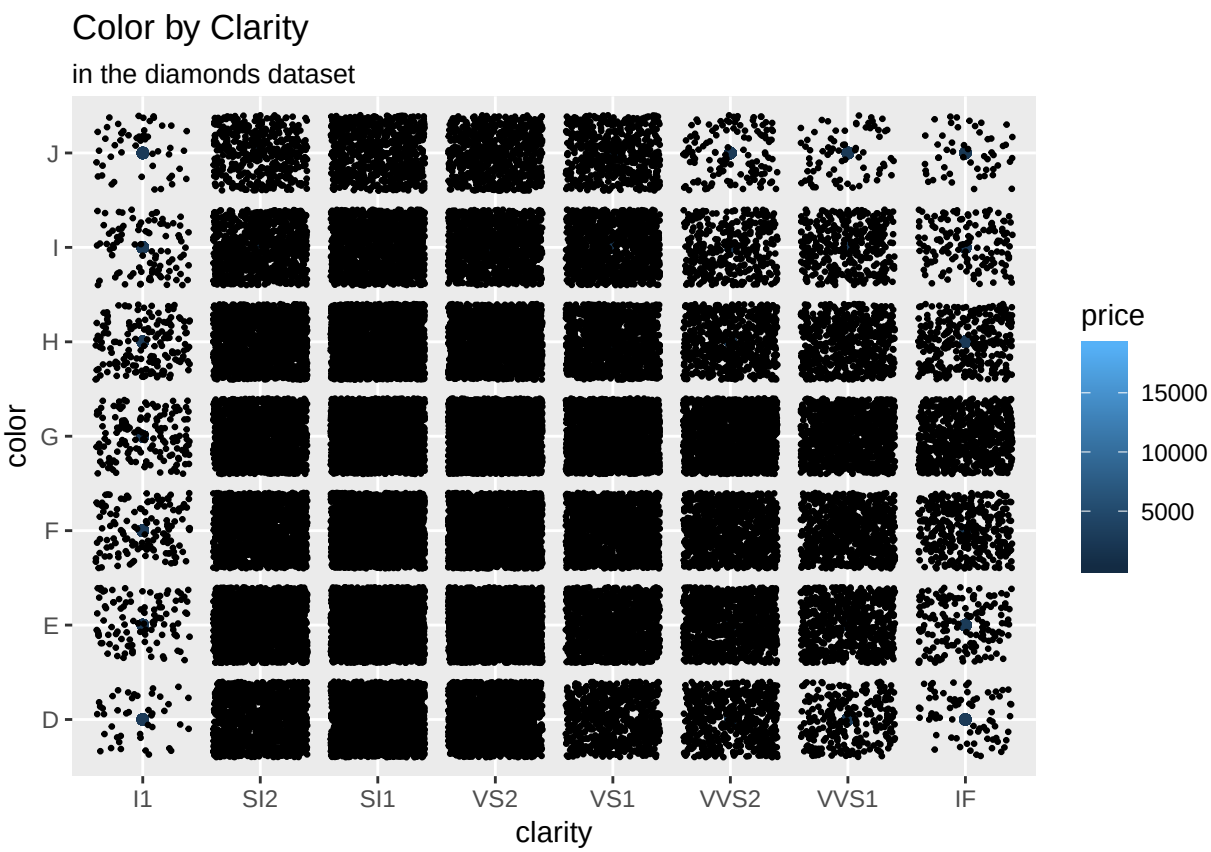
Does diamond color appear to be independent of diamond depth? Yes

Does diamond cut appear to be independent of diamond depth? Not necessarily

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

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.

```
#TO-DO
graph2 = ggplot(diamonds, aes(x = clarity, y = color)) +
  ggtitle("Color by Clarity", subtitle = "in the diamonds dataset") +
  ylab("color") +
  xlab("clarity") +
  geom_point(aes(col = price)) +

  geom_jitter(size = .5)
graph2
```



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

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

```
mod = lm(price ~  depth, diamonds)
```

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

```
coef(mod)
```

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

```
summary(mod)$r.squared
```

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

```
summary(mod)$sigma
```

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

```
sd(diamonds$price)
```

```
## [1] 3989.44
```

Are these metrics expected given the appropriate or relevant visualization(s) above? Yes, we saw price is probably independant of depth. **TO-DO

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

```
#TO-DO
mod2 = lm(price ~  carat, diamonds)
```

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

```
#TO-DO
coef(mod2)
```

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

```
summary(mod2)$r.squared
```

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

```
summary(mod2)$sigma
```

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

```
sd(diamonds$price)
```

```
## [1] 3989.44
```

Are these metrics expected given the appropriate or relevant visualization(s) above? Yes **TO-DO

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

```
#TO-DO
diamonds$color = factor(as.character(diamonds$color))
anova_mod1 = lm(price ~ color, diamonds)
```

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

```
#TO-DO
coef(anova_mod1)
```

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

```
summary(anova_mod1)$r.squared
```

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

```
summary(anova_mod1)$sigma
```

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

```
sd(diamonds$price)
```

```
## [1] 3989.44
```

Are these metrics expected given the appropriate or relevant visualization(s) above? Essentially **TO-DO

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

We used a categorical predictor

**TO-DO

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.

```
#TO-DO
b = 3169.954
abs(b - mean(diamonds$price[diamonds$color == "D"]) )
```

```
## [1] 9.594096e-05
```

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

```
## [1] 93.20152
```

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

```
## [1] 554.9324
```

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

```
## [1] 829.1817
```

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

```
## [1] 1316.715
```

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

```
## [1] 1921.921
```

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

```
## [1] 2153.864
```

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

```
#TO-DO
anova_mod2 = lm(price ~ 0 + color, diamonds)
coef(anova_mod2)
```

```
##   colorD   colorE   colorF   colorG   colorH   colorI   colorJ
## 3169.954 3076.752 3724.886 3999.136 4486.669 5091.875 5323.818
```

```
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. You can't really extrapolate from a categorical predictor. **TO-DO

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

```
#TO-DO
diamonds$cut = factor(as.character(diamonds$cut))
diamonds$clarity = factor(as.character(diamonds$clarity))
mod3 = lm(price ~ ., diamonds)
```

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

```
#TO-DO
coef(mod3)
```

```
##  (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
```

```
summary(mod3)$r.squared
```

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

```
summary(mod3)$sigma
```

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

95% of predictions are + or - $2260.

Interpret all entries in the vector b.

The intercept is color D, cut Fair, clarity I1. Each slope for each variable is the increase or decrease in price from that variable. **TO-DO

Are these metrics expected given the appropriate or relevant visualization(s) above? Can you tell from the visualizations? It's difficult to tell from the visualizations, except for carat **TO-DO

Comment on why R^2 is high. Think theoretically about diamonds and what you know about them. Knowing all the important factors that go into the price of a diamond, as opposed to knowing just one factor, enables us to more accurately assess the price. **TO-DO

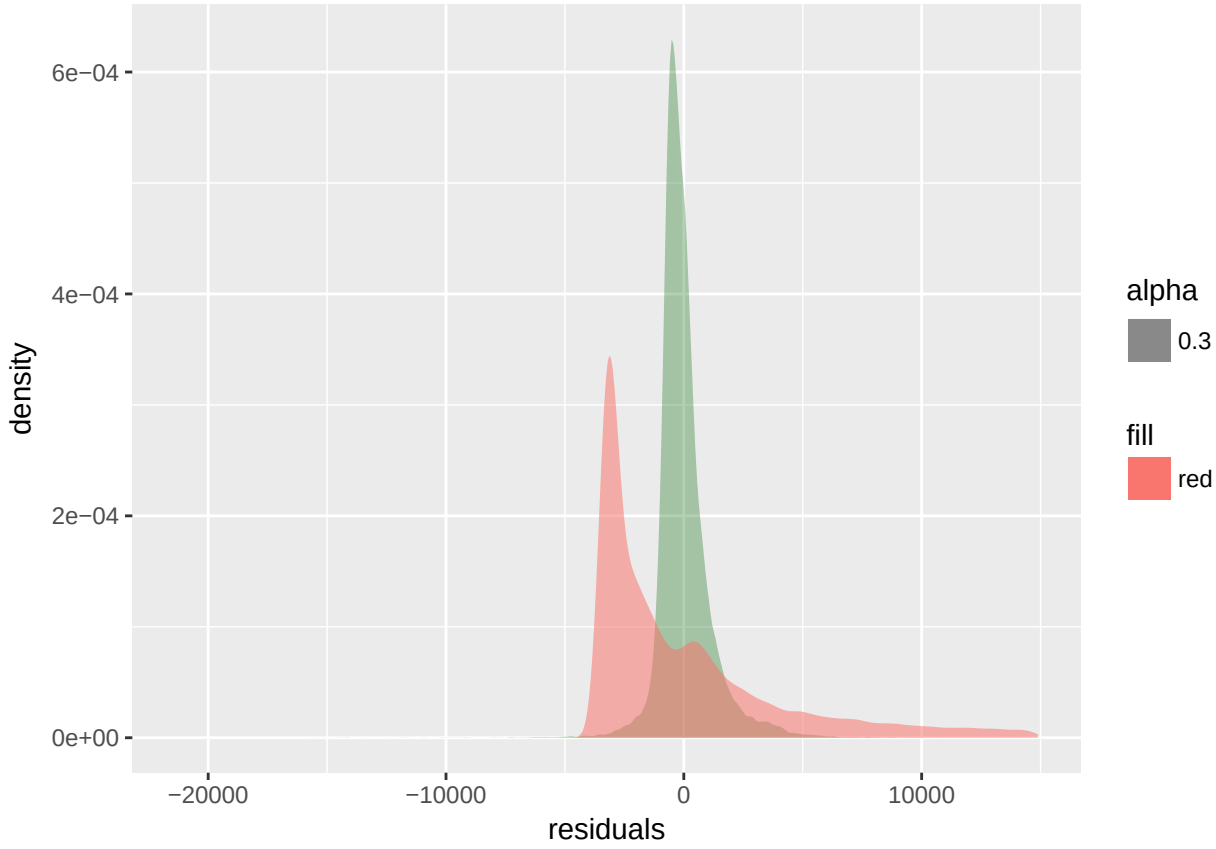
Do you think you overfit? Comment on why or why not but do not do any numerical testing or coding.

Not particularly, there is a large enough sample size that this model could probably fit any diamond we'd want to predict the price of.

**TO-DO

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

```
#TO-DO
ggplot(data.frame(null_residuals = diamonds$price - mean(diamonds$price), residuals = resid(mod3))) +
  stat_density(aes(x = residuals), fill = "darkgreen", alpha = 0.3) +
  stat_density(aes(x = null_residuals, fill = "red", alpha = 0.3))
```



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

Yes-ish, but is probably exponential or something like that.

** TO-DO

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

```
#TO-DO

mod4 = lm(price ~ cut+color+clarity+x+y+z+depth+table +poly(carat, 2, raw = TRUE), diamonds)
```

What is b, R^2 and the RMSE?

```
#TO-DO
coef(mod4)
```

```
##                 (Intercept)                      cutGood
##                  9807.97904                   538.33407
##                    cutIdeal                 cutPremium
##                   807.51616                   747.69518
##                cutVery Good                      colorE
##                   678.31993                  -209.43992
##                      colorF                      colorG
##                  -284.54706                  -496.84716
##                      colorH                      colorI
##                  -997.60127                 -1469.25151
##                      colorJ                    clarityIF
##                 -2357.79746                  5243.52276
```

```
##                  claritySI1                  claritySI2
##                  3565.41193                  2605.54013
##                  clarityVS1                  clarityVS2
##                  4475.44424                  4163.34947
##                 clarityVVS1                 clarityVVS2
##                  4904.22750                  4843.80493
##                           x                           y
##                -2123.00617                   -23.46172
##                           z                       depth
##                   -83.11272                  -116.22729
##                       table poly(carat, 2, raw = TRUE)1
##                   -36.37384                 16144.75809
## poly(carat, 2, raw = TRUE)2
##                 -1028.81806
```

```
summary(mod4)$r.squared
```

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

```
summary(mod4)$sigma
```

```
## [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. The intercept is color D, cut Fair, clarity I1. Each slope for each variable is the increase or decrease in price from that variable. carat^2 is negative because when we extrapolate a polynomial it will eventually go down. **TO-DO

Is this an improvement over the model in #4? Yes/no and why. Yes. The rmse is lower and r-squared is higher **TO-DO

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

```
#TO-DO
b = coef(mod4)
g = function(x_star){
  x_star_clarity = x_star["clarity"]
  x_star_color = x_star["color"]
  x_star_cut = x_star["cut"]

  yhat ={  b["(Intercept)"] + x_star["carat"]*b["carat"] + x_star["carat"]^2*b["carat"]
      if(as.numeric(x_star_clarity) == 1){+b["clarityI1"]}
     if(as.numeric(x_star_clarity) == 2) {+b["claritySI2"]}
    if(as.numeric(x_star_clarity) == 3) {+b["claritySI1"]}
      if(as.numeric(x_star_clarity) == 4) {+b["clarityVS2"]}
     if(as.numeric(x_star_clarity) == 5) {+b["clarityVS1"]}
     if(as.numeric(x_star_clarity) == 6) {+b["clarityVVS2"]}
      if(as.numeric(x_star_clarity) == 7) {+b["clarityVVS1"]}
     if(as.numeric(x_star_clarity) == 8) {+b["clarityIF"]}
      if(as.numeric(x_star_color) == 1) {+b["colorD"]}
     if(as.numeric(x_star_color) == 2) {+b["colorE"]}
      if(as.numeric(x_star_color) == 3) {+b["colorF"]}
       if(as.numeric(x_star_color) == 4) {+b["colorG"]}
     if(as.numeric(x_star_color) == 5) {+b["colorH"]}
    if(as.numeric(x_star_color) == 6) {+b["colorI"]}
    if(as.numeric(x_star_color) == 7) {+b["colorJ"]}
     if(as.numeric(x_star_cut) == 1) {+b["colorFair"]}
```

```
     if(as.numeric(x_star_cut) == 2) {+b["colorGood"]}
    if(as.numeric(x_star_cut) == 3) {+b["colorVery Good"]}
     if(as.numeric(x_star_cut) == 4) {+b["colorPremium"]}
    if(as.numeric(x_star_cut) == 5) {+b["colorIdeal"]}
    +x_star["cutPremium"]*b["color"] + b["x"]*x_star["x"] + b["y"]*x_star["y"]+b["z"]*x_star["z"]
  yhat
}
#g = function(x_star){
#  x_star = model.matrix(mod4, diamonds)
#  yhat = x_star%*%b
#  yhat
#}
test_diamond = diamonds[20001,]
prediction = predict(mod4,test_diamond)
prediction
```

```
##        1
## 9302.685
```

```
g(test_diamond)
```

```
## Error in (function (..., row.names = NULL, check.rows = FALSE, check.names = TRUE, : row names conta
```

```
n = 53000
#test_diamond["clarityVS1"]
test_indices = sample(1 : n, 1)
X_test = (diamonds[test_indices, ])
X_test$price = NULL
g(X_test)
```

```
## Error in (function (..., row.names = NULL, check.rows = FALSE, check.names = TRUE, : row names conta
```

```
b["cutGood"]+b["cutPremium"]+b["cutIdeal"]+b["colorE"]+b["colorF"]+b["colorG"]+b["colorH"]+b["colorI"]+
```

```
##  cutGood
## 30554.81
```

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

```
degree_2_poly_mod1 = lm(price ~ poly(color, 2, raw = TRUE), diamonds)
```

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

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

Why did this throw an error? Color is categorical **TO-DO

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
data("diamonds")
#diamonds$cutFair = ifelse(diamonds$cut =="Fair", 1,0)
#diamonds$cutGood = ifelse(diamonds$cut =="Good", 1,0)
#diamonds$cutVery_good = ifelse(diamonds$cut =="Very Good", 1,0)
#diamonds$cutPremium = ifelse(diamonds$cut =="Premium", 1,0)
#diamonds$cutIdeal = ifelse(diamonds$cut =="Ideal", 1,0)
#diamonds$colorD = ifelse(diamonds$color =="D", 1,0)
#diamonds$colorE = ifelse(diamonds$color =="E", 1,0)
```

```
#diamonds$colorE = ifelse(diamonds$color =="F", 1,0)
#diamonds$colorH = ifelse(diamonds$color =="H", 1,0)
#diamonds$colorI = ifelse(diamonds$color =="I", 1,0)
#diamonds$colorJ = ifelse(diamonds$color =="J", 1,0)
#diamonds$clarityIF = ifelse(diamonds$clarity =="IF",1,0)
#diamonds$clarityVVS1 = ifelse(diamonds$clarity =="VVS1", 1,0)
#diamonds$clarityVVS2 = ifelse(diamonds$clarity =="VVS2", 1,0)
#diamonds$clarityVS1 = ifelse(diamonds$clarity =="VS1", 1,0)
#diamonds$clarityVS2 = ifelse(diamonds$clarity =="VS2", 1,0)
#diamonds$claritySI1 = ifelse(diamonds$clarity =="SI1", 1,0)
#diamonds$claritySI2 = ifelse(diamonds$clarity =="SI2", 1,0)
#diamonds$cut = NULL
#diamonds$color = NULL
#diamonds$clarity = NULL


#diamonds1$price = NULL
#diamonds1$cutGood = NULL
#diamonds1$colorD = NULL
#diamonds1$clarityIF =NULL
#X = as.matrix(cbind(1,diamonds1))
#XtX = t(X) %*% X
#XtXinv = solve(XtX, tol =  1.73e-22)
#H = X %*% XtXinv %*% t(X)
#yhat = H %*% y
diamonds1 = diamonds[sample(nrow(diamonds), 0.08*nrow(diamonds)), ]
y = diamonds1$price
head(yhat)
```

```
## Error in head(yhat): object 'yhat' not found
```

```
X = model.matrix(price ~ . , diamonds1)
XtX = t(X) %*% X
XtXinv = solve(XtX, tol =  1.73e-22)
H = X %*% XtXinv %*% t(X)
yhat = H %*% y
```

What is b, R^2 and the RMSE?

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

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

```
## (Intercept)  1271.092338
## carat       11300.599362
## cut.L         582.802025
## cut.Q        -376.345205
## cut.C         198.038414
## cut^4          -7.311099
## color.L     -1808.218263
## color.Q      -627.373798
## color.C      -151.405038
## color^4        26.916010
## color^5      -149.804327
## color^6       -97.971497
## clarity.L    3913.107976
## clarity.Q   -1830.233129
## clarity.C     910.098588
## clarity^4    -300.136675
## clarity^5     293.539762
## clarity^6      53.280855
## clarity^7     138.609208
## depth          20.906184
## table         -32.929469
## x             715.664091
## y            -663.777992
## z           -1855.520292
```

```
(SST -SSE)/SST #R^2
```

```
## [1] 0.9245237
```

```
sqrt(mean(e^2)) #RMSE
```

```
## [1] 1093.563
```

Are they the same as in #4? Very close, but we sampled so it's not the same **TO-DO

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.

```
#TO-DO
qrX = qr(X)
Q = qr.Q(qrX)
R = qr.R(qrX)

sum(Q[, 1]^2) #normalized?
```

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

```
sum(Q[, 2]^2) #normalized?
```

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

```
Q[, 1] %*% Q[, 2] #orthogonal?
```

```
##               [,1]
## [1,] -1.198247e-16
```

```
Q[, 2] %*% Q[, 3] #orthogonal?
```

```
##              [,1]
## [1,] 2.001708e-17
```

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

```
##       [,1]
## 1 1036.561
## 2 1445.745
## 3 6094.748
## 4 6712.406
## 5 5843.386
## 6 6032.536
```

```
head(yhat_via_Q)
```

```
##          [,1]
## [1,] 1036.561
## [2,] 1445.745
## [3,] 6094.748
## [4,] 6712.406
## [5,] 5843.386
## [6,] 6032.536
```

```
bq = solve(R) %*% t(Q) %*% y
bq
```

```
##                     [,1]
## (Intercept)  1271.092340
## carat       11300.599362
## cut.L         582.802025
## cut.Q        -376.345205
## cut.C         198.038414
## cut^4          -7.311099
## color.L     -1808.218263
## color.Q      -627.373798
## color.C      -151.405038
## color^4        26.916010
## color^5      -149.804327
## color^6       -97.971497
## clarity.L    3913.107976
## clarity.Q   -1830.233129
## clarity.C     910.098588
## clarity^4    -300.136675
## clarity^5     293.539762
## clarity^6      53.280855
## clarity^7     138.609208
## depth          20.906184
## table         -32.929469
## x             715.664090
## y            -663.777992
## z           -1855.520291
```

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

```
#TO-DO
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

```
#TO-DO

pacman::p_load(testthat)
expect_equal(as.vector(e+yhat), y) #(a)
expect_equal(as.vector(t(yhat) %*% e),0, tol = .2) #(b)
```

```
## Error: as.vector(t(yhat) %*% e) not equal to 0.
## 1/1 mismatches
## [1] 0.312 - 0 == 0.312
```

```
expect_equal(sum(H %*% e),0 , tol = 1e-2) #(c)
expect_equal (H %*% yhat, yhat) #(d)
expect_equal(sum((I-H) %*% yhat), 0 , tol = 1e-4) #(e)
expect_equal(SSR + SSE,SST) # (f)
```

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

```
#TO-DO

data("diamonds")
mod5 = lm(price ~. * .  +poly(carat, 5, raw = TRUE)+poly(x, 5, raw = TRUE)+poly(y, 5, raw = TRUE)+poly(
```

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

```
#TO-DO
summary(mod5)$r.squared
```

```
## [1] 0.9728255
```

```
summary(mod5)$sigma
```

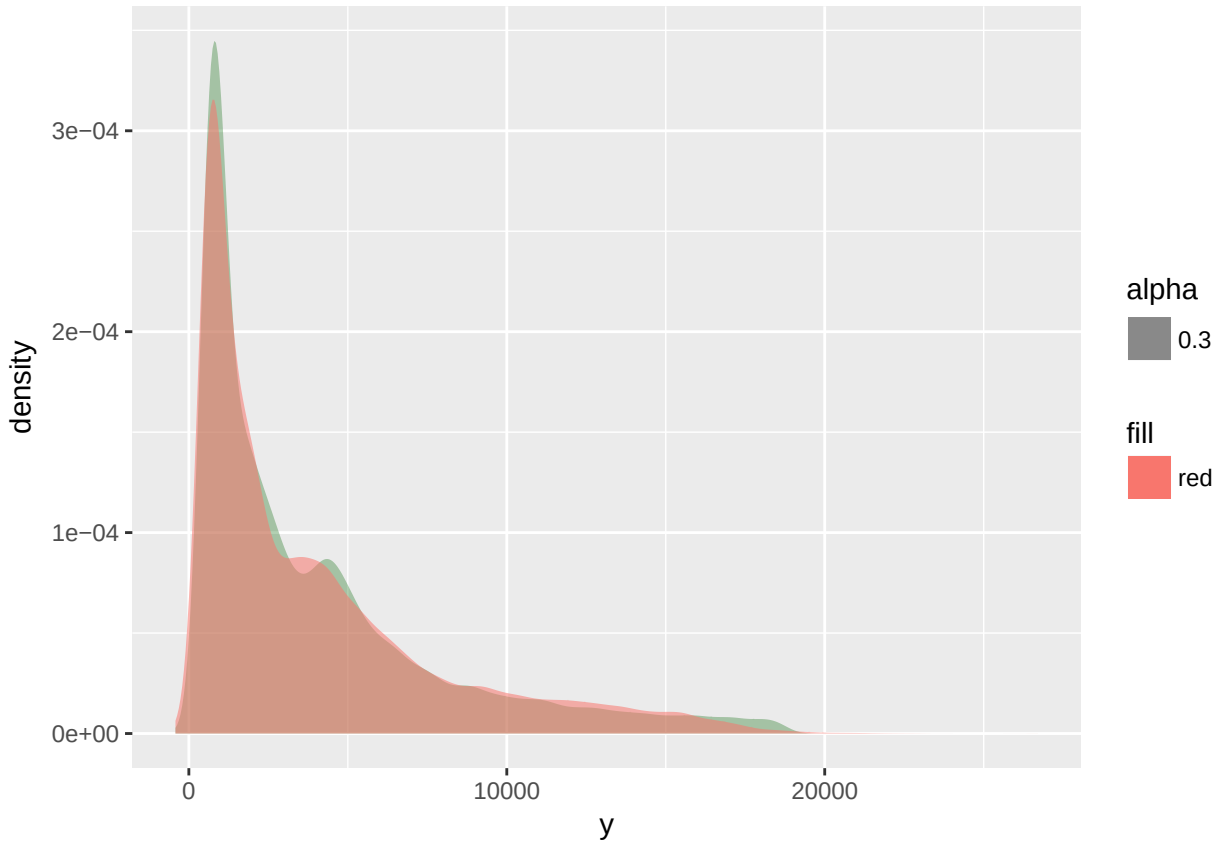
```
## [1] 659.2249
```

```
sd(mod5$residuals)
```

```
## [1] 657.6464
```

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

```
#TO-DO
y =(diamonds$price)
y_hat = as.vector(y - resid(mod5))
ggplot() +
  stat_density(aes(x = y), fill = "darkgreen", alpha = 0.3) +
  stat_density(aes(x = y_hat, fill = "red", alpha = 0.3))
```



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

```
#TO-DO
#abs(mod5$residuals)
sum(abs((mod5$residuals))>=1000)
```

```
## [1] 4583
```

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. Not terrible but it's still + or - 1300$ **TO-DO

What is the degrees of freedom in this model?

```
#TO-DO
#b = (coef(mod5) )
#b = b[!is.na(b)]
#length(b)
length(coef(mod5))
```

```
## [1] 265
```

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

Do you think g is close to f in this model? Yes / no and why? yes, because we are fitting it better.

What more degrees of freedom can you add to this model to make g closer to f? n-1 degrees of freedom.

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.

The source, a Tiffany diamond is more expensive than a Costco diamond.

** TO-DO

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

```
#TO-DO
#set.seed(1000)
n = 53940
K = 10 #i.e. the test set is 1/10th of the entire historical dataset
y = diamonds$price
#a simple algorithm to do this is to sample indices directly
test_indices = sample(1 : n, 1 / K * n)
train_indices = setdiff(1 : n, test_indices)

#now pull out the matrices and vectors based on the indices
X_train = diamonds[train_indices, ]
y_train = y[train_indices]
X_test = diamonds[test_indices, ]
y_test = y[test_indices]
mod6 = lm(price ~. * .  +poly(carat, 5, raw = TRUE)+poly(x, 5, raw = TRUE)+poly(y, 5, raw = TRUE)+poly(
summary(mod6)$r.squared
```

```
## [1] 0.9727865
```

```
sd(mod6$residuals)
```

```
## [1] 658.3537
```

```
y_hat_oos = predict(mod6, data.frame(X_test))
```

```
## Warning in predict.lm(mod6, data.frame(X_test)): prediction from a rank-
## deficient fit may be misleading
```

```
oos_residuals = y_test - y_hat_oos
1 - sum(oos_residuals^2) / sum((y_test - mean(y_test))^2)
```

```
## [1] 0.972445
```

```
sd(oos_residuals)
```

```
## [1] 660.1071
```

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? no ** TO-DO

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(1004)
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 = 1
X = matrix(c(x, rnorm(n * p_fake)), ncol = 1 + p_fake)

mod = lm(y ~ X)
summary(mod)$r.squared
```

```
## [1] 0.66882
```

```
sd(mod$residuals)
```

```
## [1] 0.9999428
```

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:

```
  test_indices = sample(1 : n, 1 / 10 * n)
  train_indices = setdiff(1 : n, test_indices)
  total_oose = c(2)
  for (i in 1:100){
    X=  cbind(X, rnorm(n))
    X_test = X[test_indices, ]
    X_train = X[train_indices, ]
    y_test = y[test_indices]
    y_train = y[train_indices]
    mod = lm(y_train ~.,data.frame(X_train))

    y_hat_oos = predict(mod, data.frame(X_test))
    oos_residuals = y_test - y_hat_oos
  #1 - sum(oos_residuals^2) / sum((y_test - mean(y_test))^2)
  #print(i)
    (summary(mod)$r.squared)
    (1 - sum(oos_residuals^2) / sum((y_test - mean(y_test))^2))
  #diff = sd(oos_residuals)-sd(mod$residuals)
  #print(sd(oos_residuals))
    total_oose = cbind(total_oose,sd(oos_residuals))
}
```

```
## 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

## 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

## 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

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

```r
min(total_oose)
```

```
## [1] 1.286785
```

Around 55 from my test, but it should just be 1, after all, one fake predictor is already overfitting.