HW03p

Greg Maghakian

April 13, 2018

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

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

```
pacman::p_load(ggplot2)
diamonds$cut = factor(as.character(diamonds$cut))
diamonds$color = factor(as.character(diamonds$color))
diamonds$clarity = factor(as.character(diamonds$clarity))
```

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
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    : Factor w/ 5 levels "Fair","Good",..: 3 4 2 4 2 5 5 5 1 5 ...
##  $ color  : Factor w/ 7 levels "D","E","F","G",..: 2 2 2 6 7 7 6 5 2 5 ...
##  $ clarity: Factor w/ 8 levels "I1","IF","SI1",..: 4 3 5 6 4 8 7 3 6 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 ...
```

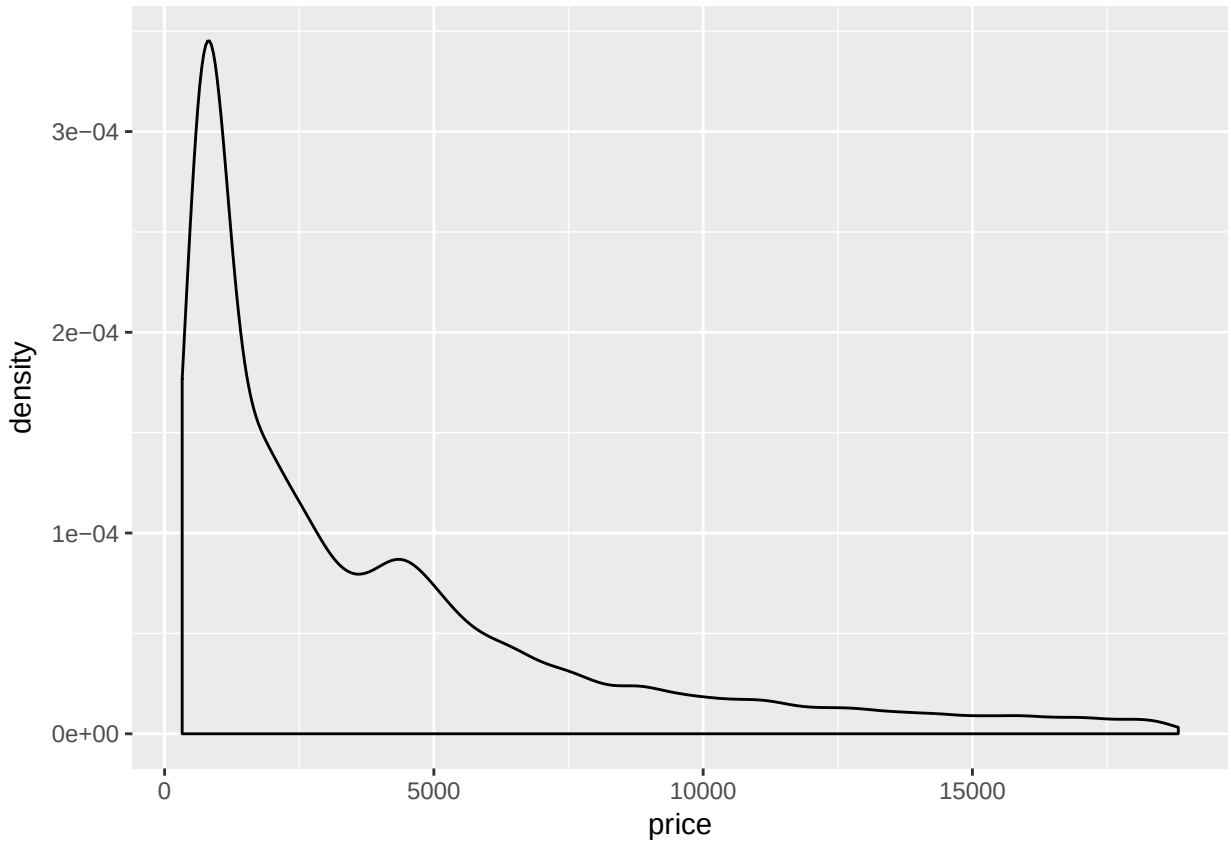
What is n, p, what do the features mean, what is the most likely response metric and why?

n is 53940 observations of diamonds and p is 10 features for diamonds. These features include price, carat, cut, clarity, length (x), width (y), depth (z), total depth percentage, and width of top of diamond relative to widest point. I believe price would be the response metric as it would make the most sense for us to want to observe and predict for the diamond industry. We can view how variations in features such as cut and clarity will change the price of the diamond.

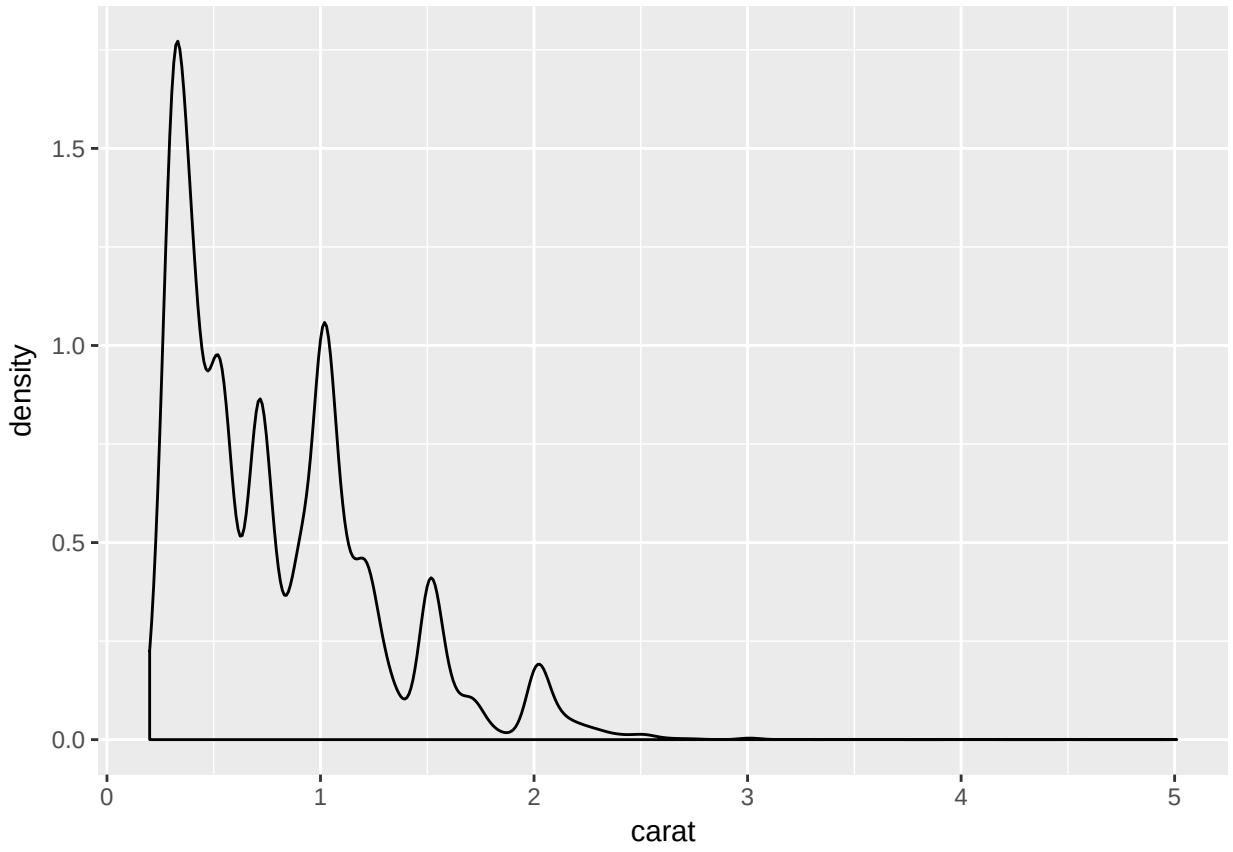
Regardless of what you wrote above, the variable `price` will be the response variable going forward.

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

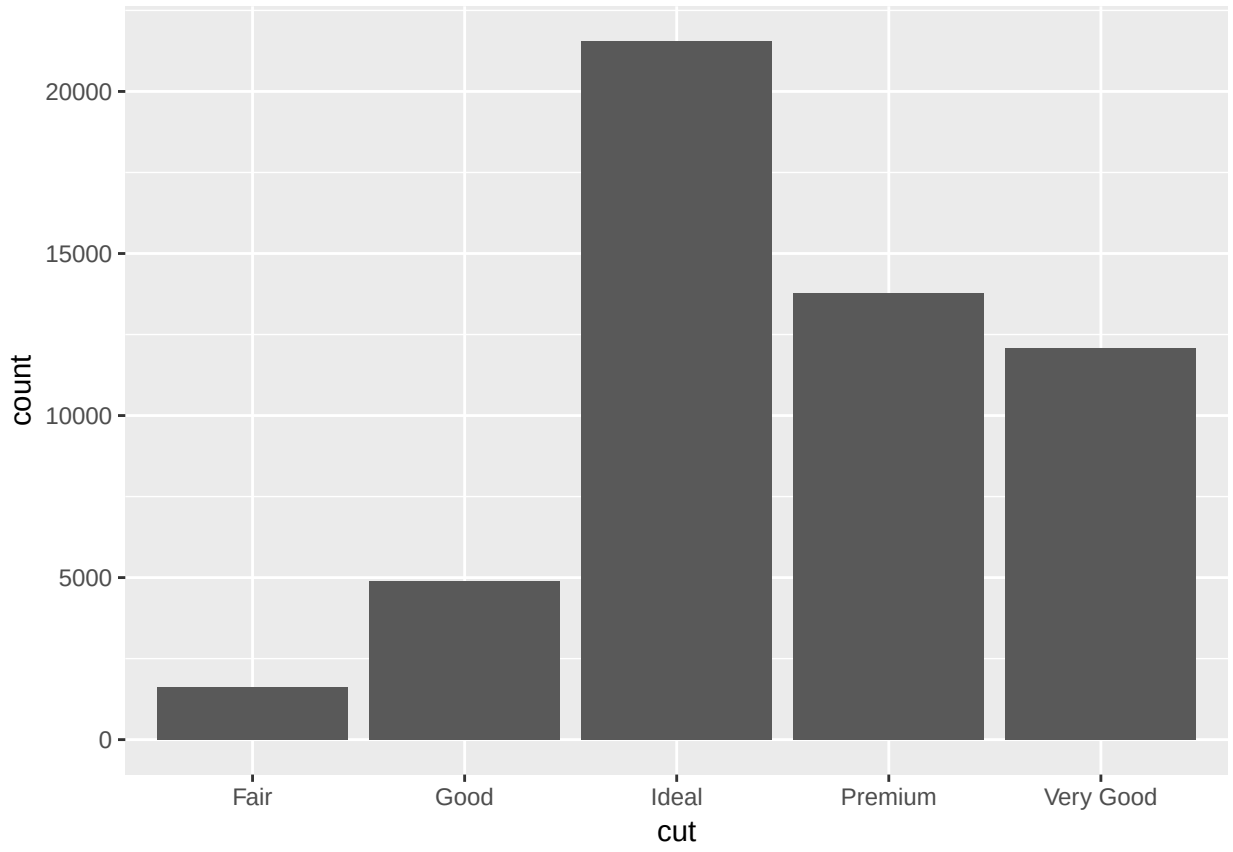
```
ggplot(diamonds, aes(price))+geom_density()
```



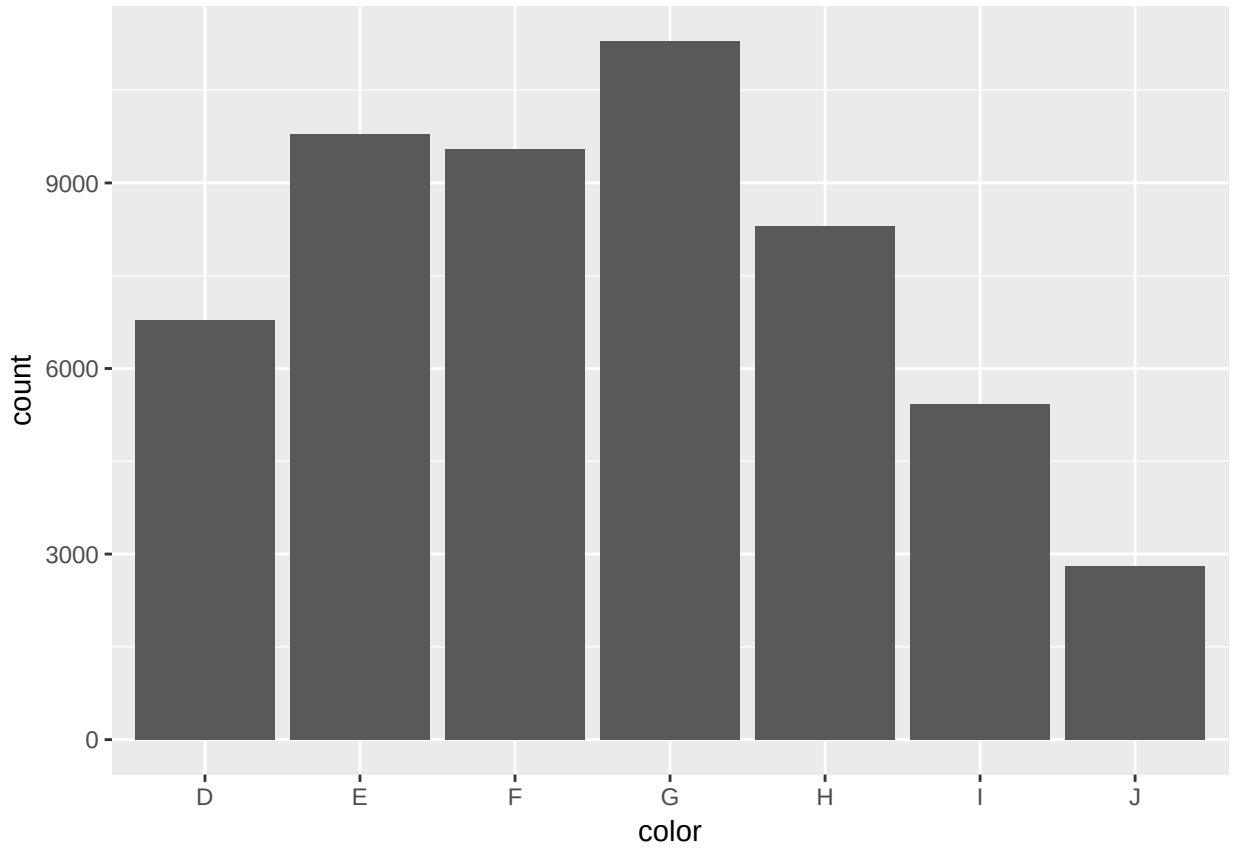

```
ggplot(diamonds,aes(carat))+geom_density()
```



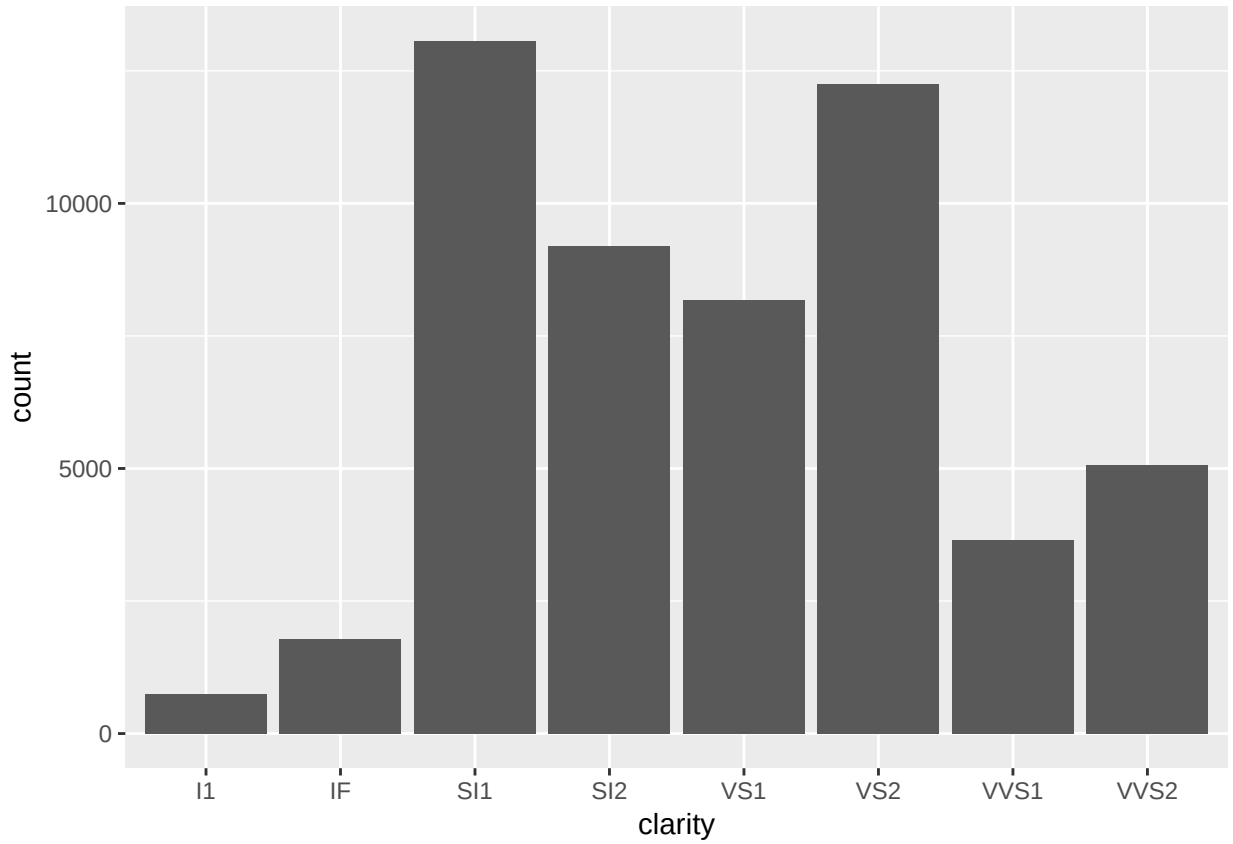

```
ggplot(diamonds,aes(cut))+geom_bar()
```



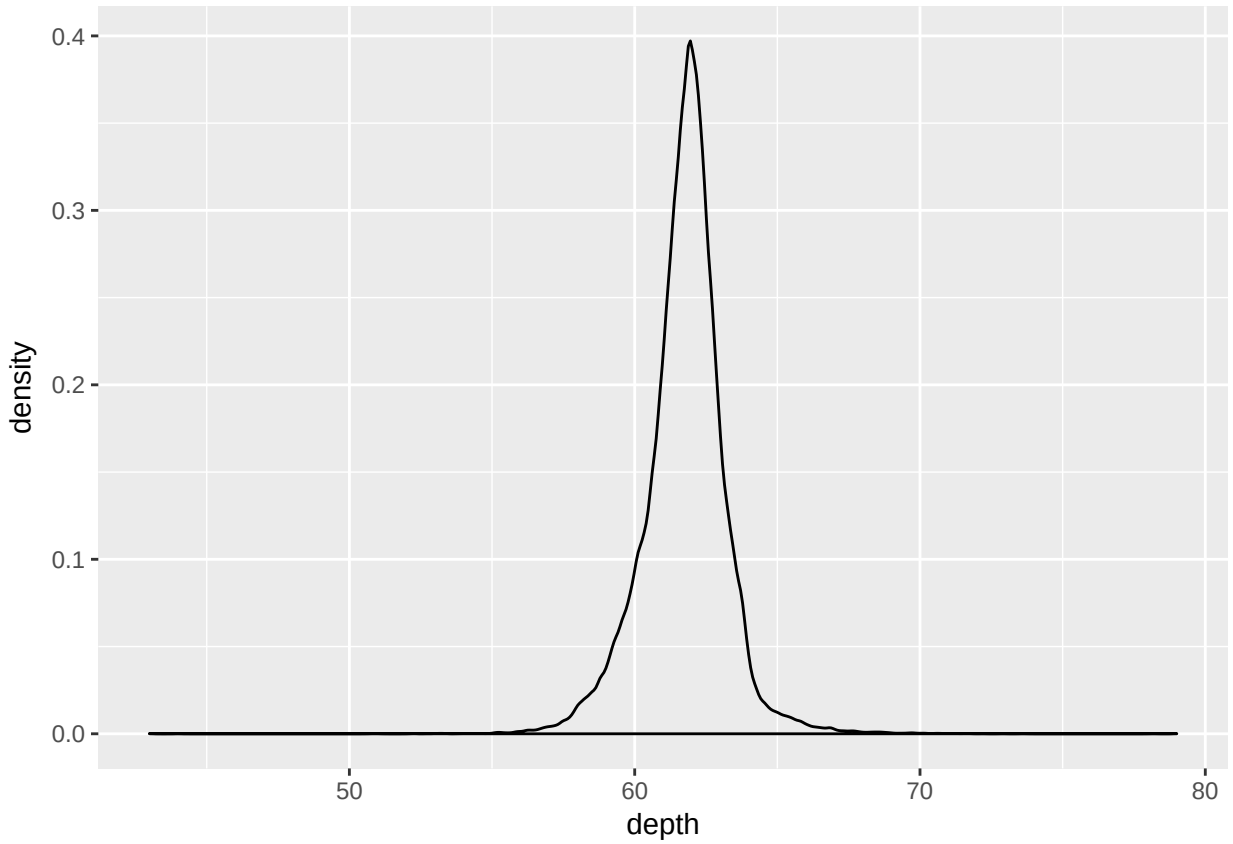

```
ggplot(diamonds,aes(color))+geom_bar()
```



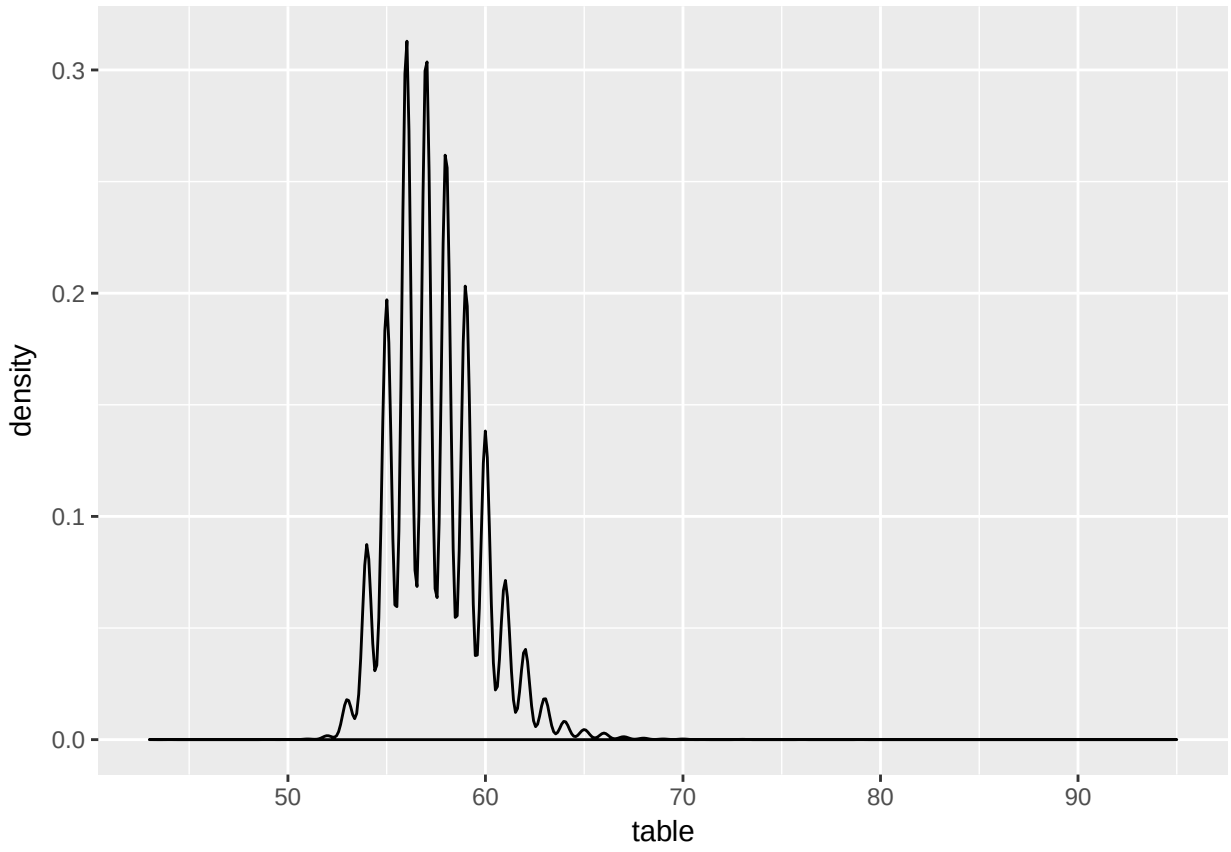

```
ggplot(diamonds,aes(clarity))+geom_bar()
```



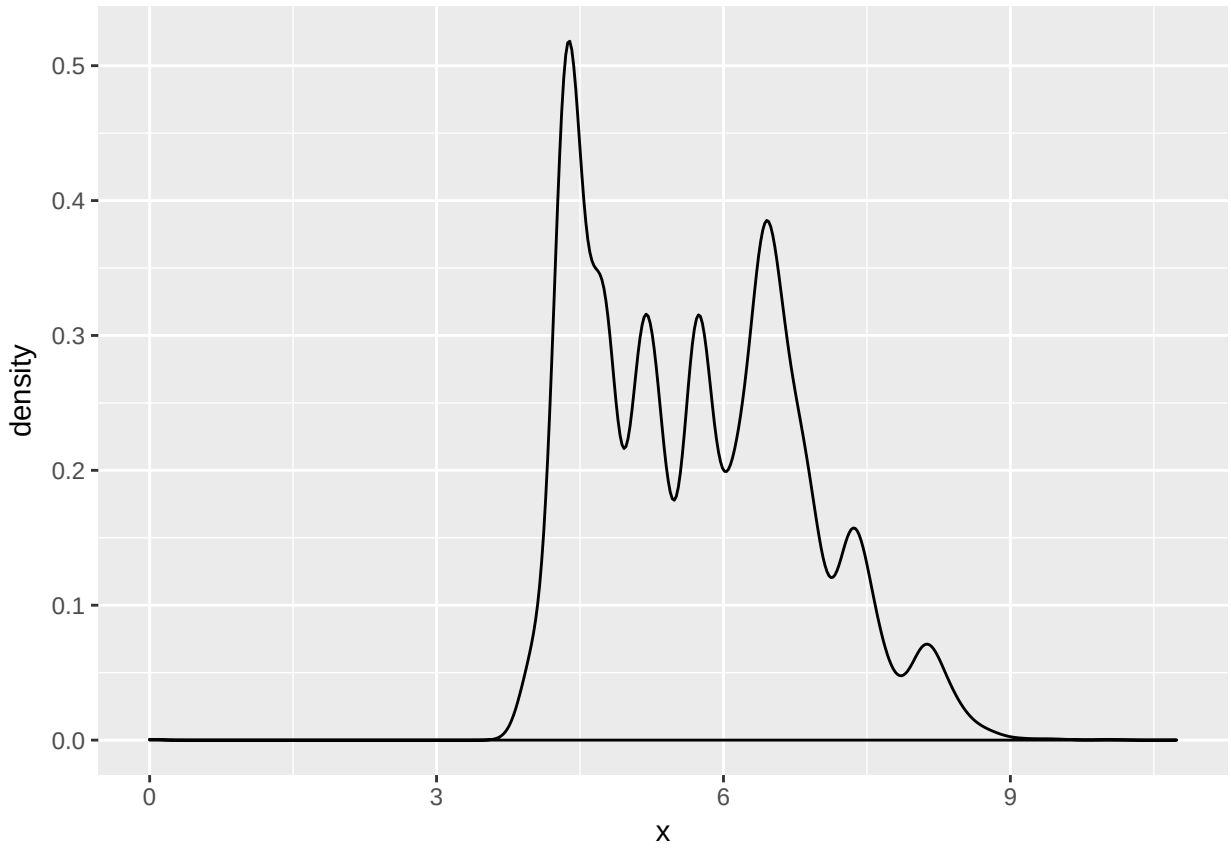

```
ggplot(diamonds,aes(depth))+geom_density()
```



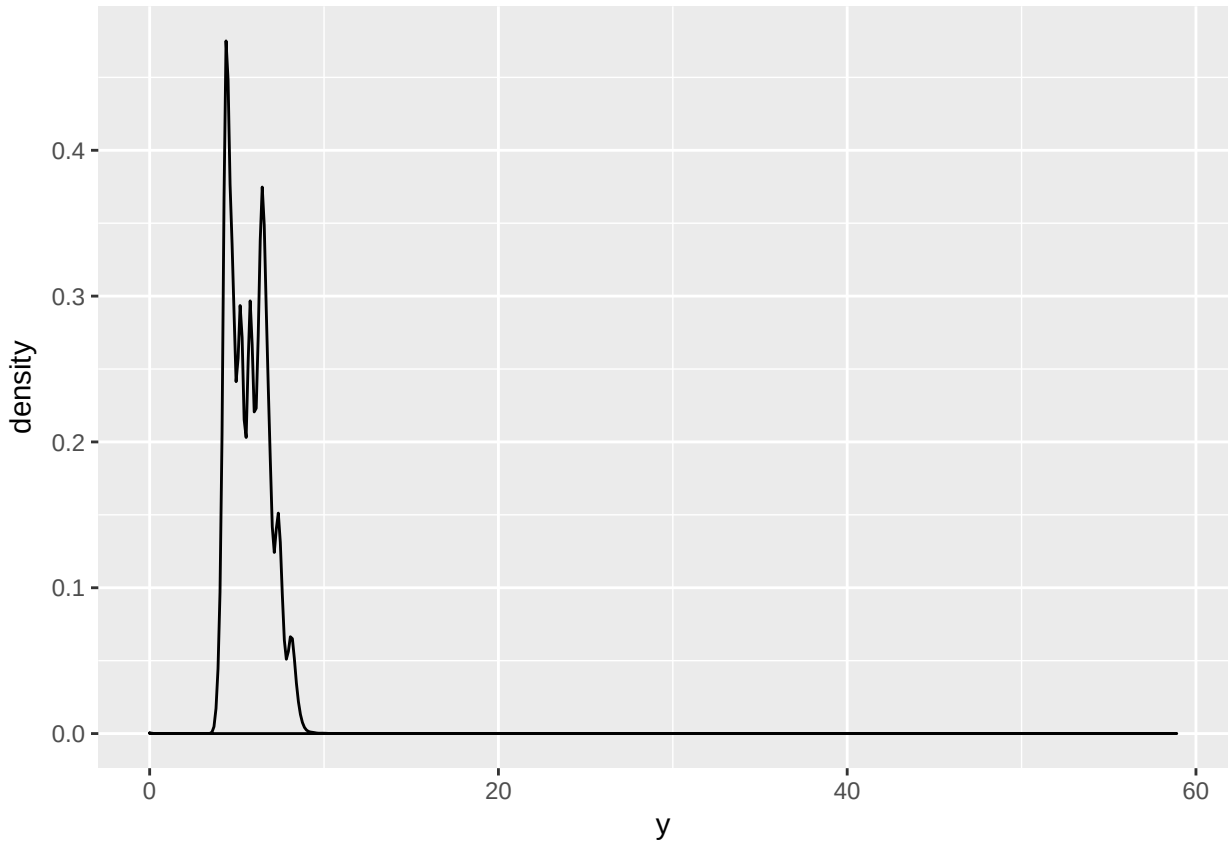

```
ggplot(diamonds,aes(table))+geom_density()
```



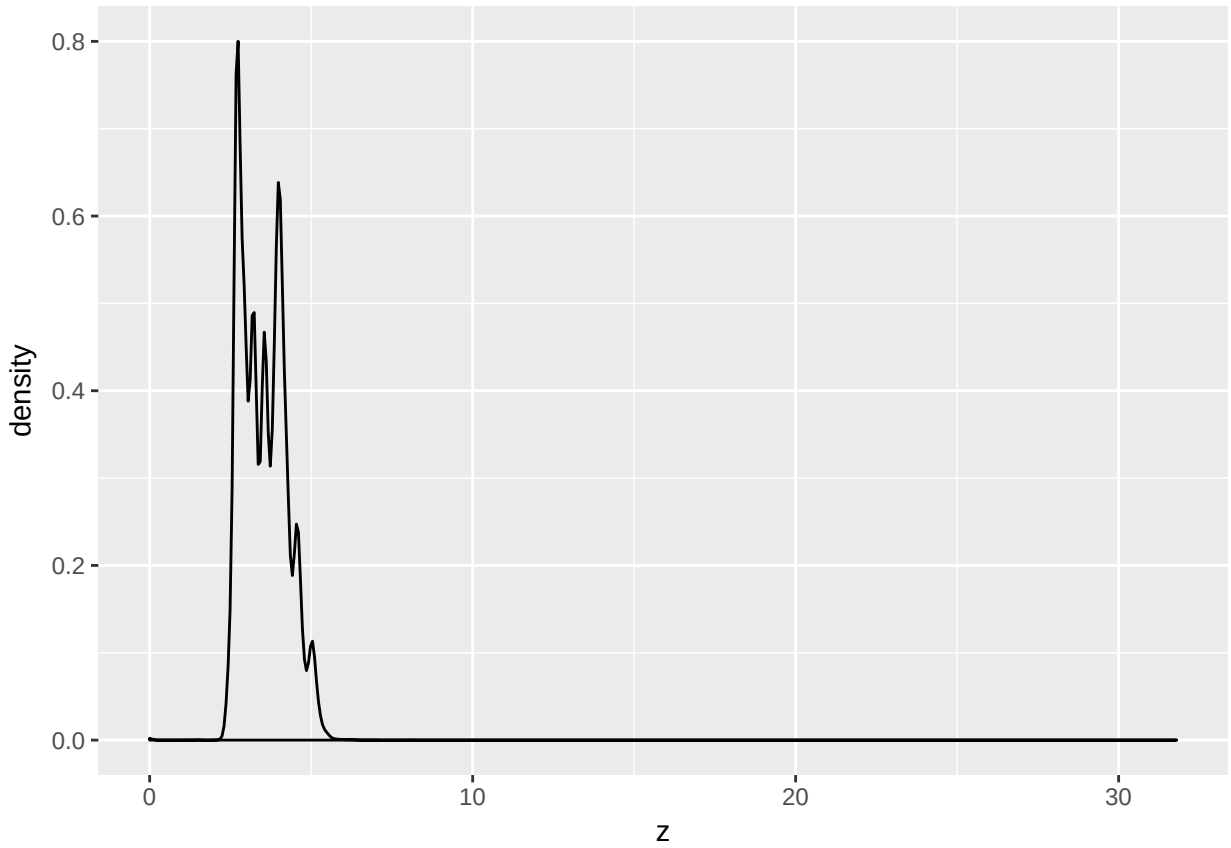

```
ggplot(diamonds,aes(x))+geom_density()
```




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

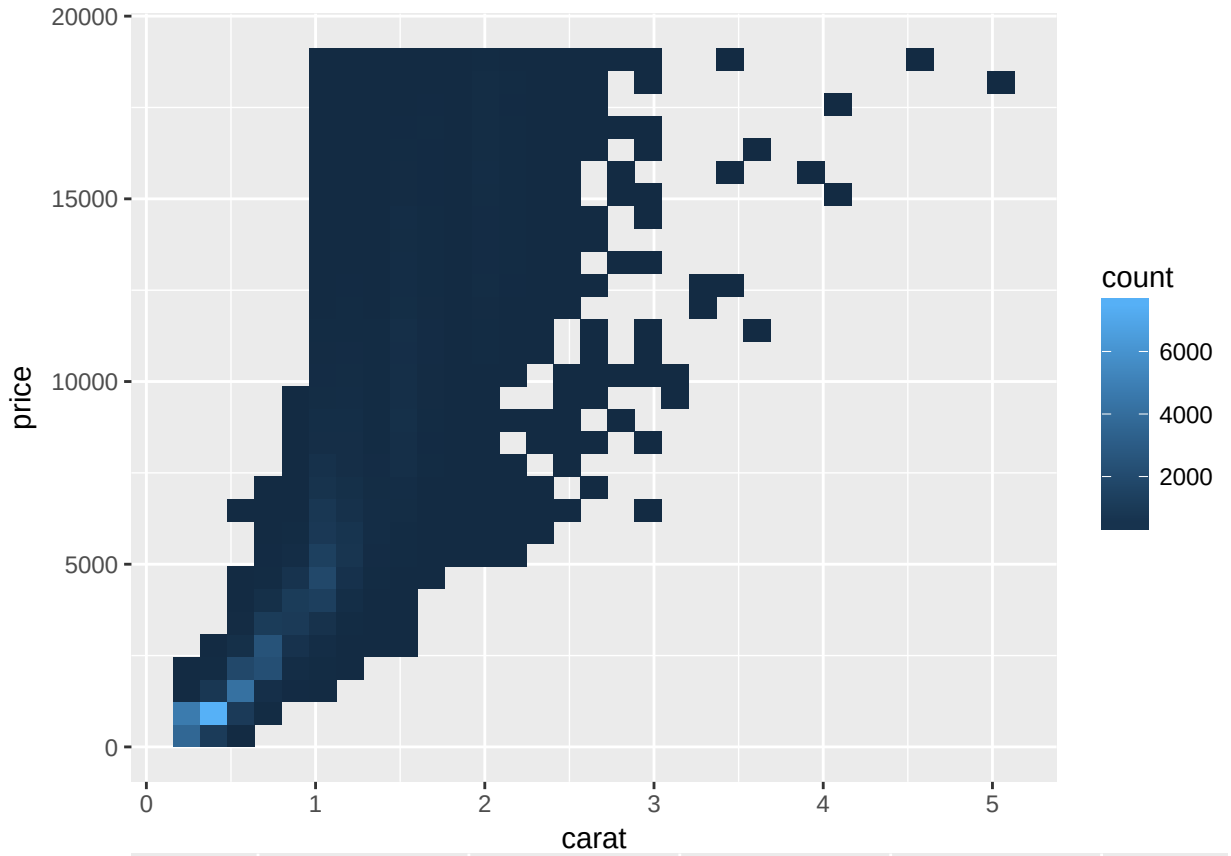


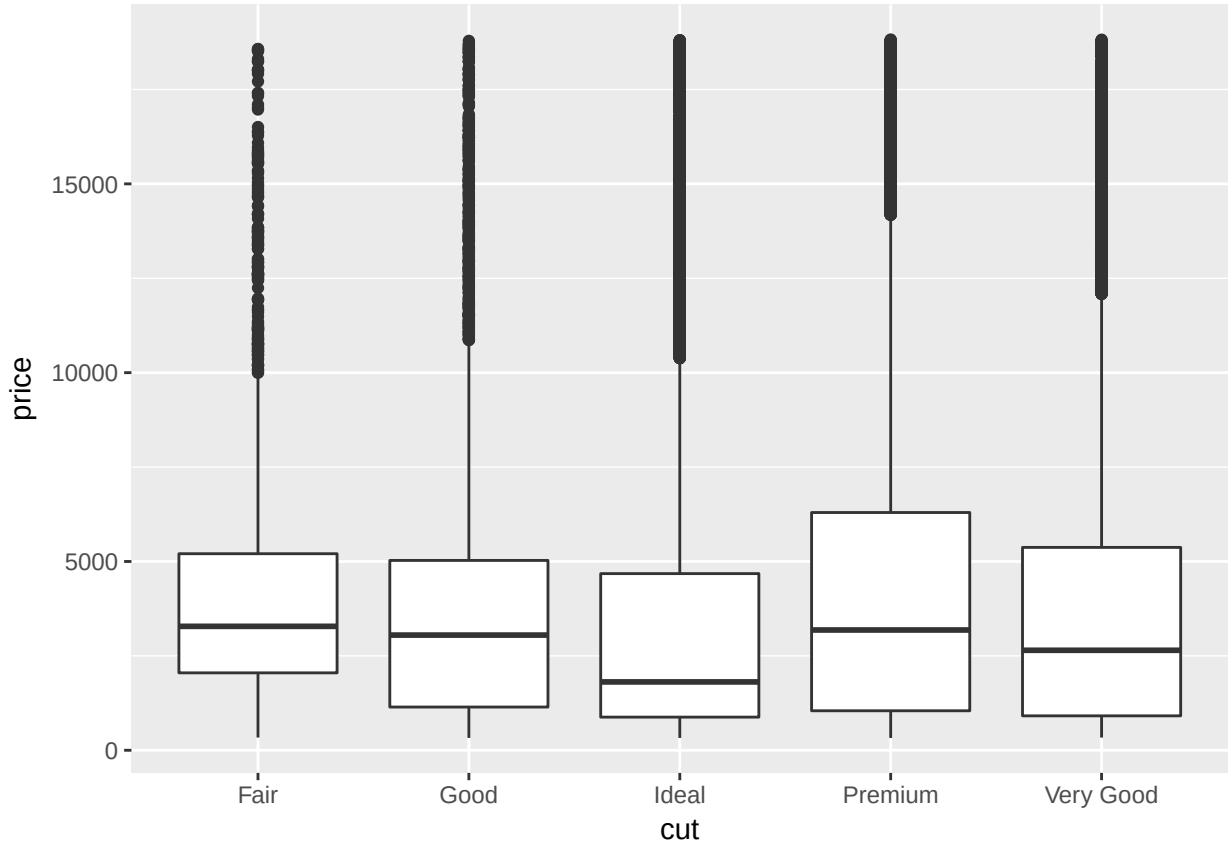

```
ggplot(diamonds,aes(z))+geom_density()
```

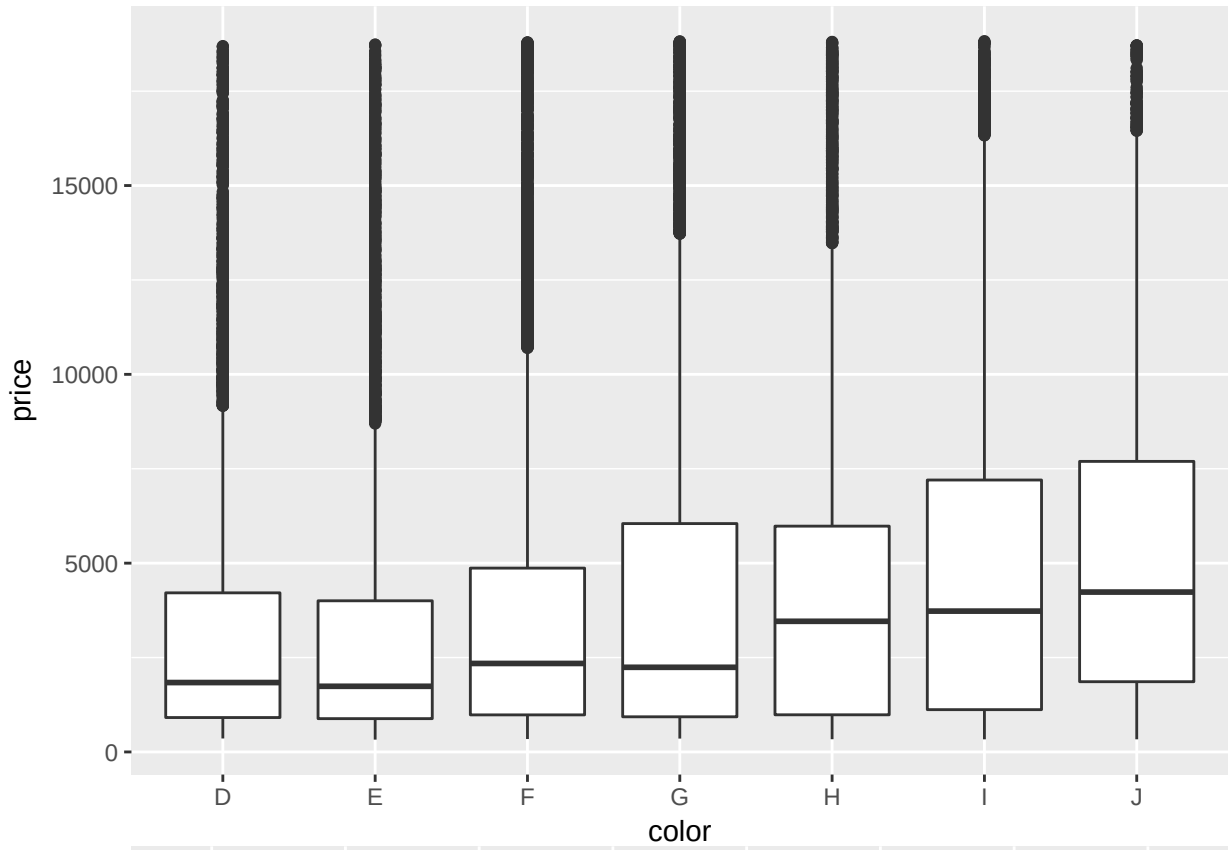


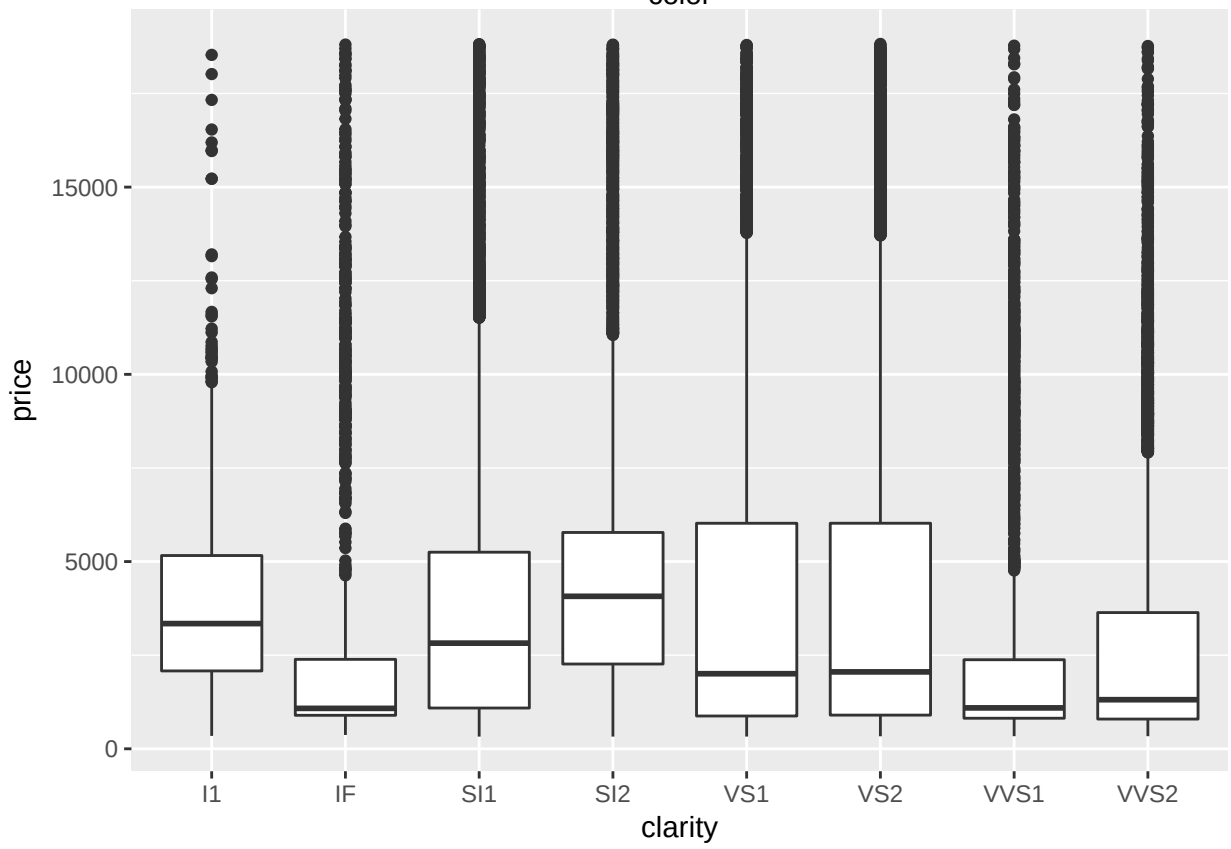
Use `ggplot` to look at the bivariate distributions of the response versus *all* predictors. Make sure you handle categorical predictors differently from continuous predictors. This time employ a for loop when an logic that handles the predictor type.

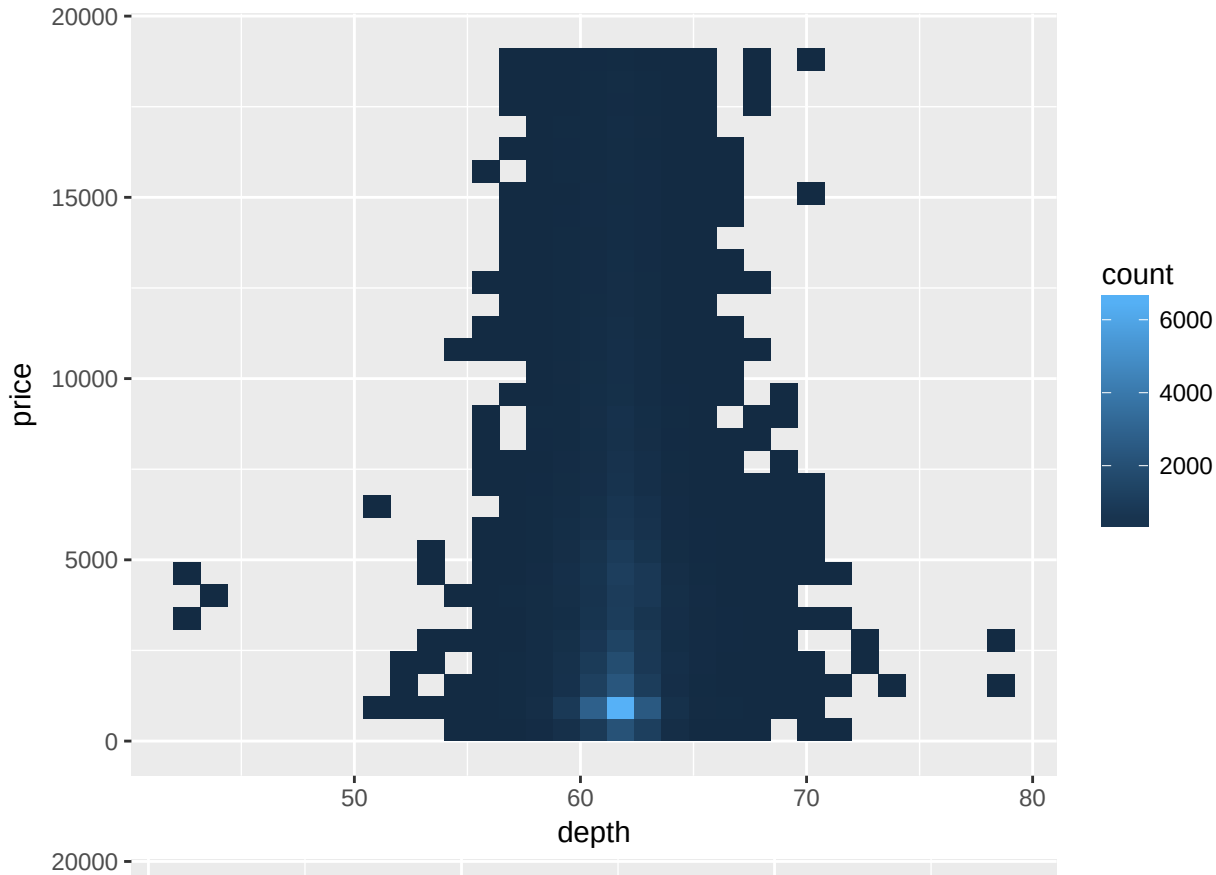
```
for(i in 1:ncol(diamonds)) {
  if(names(diamonds)[i]=="price"){
   next
  }
    if(is.numeric(diamonds[[i]])){
    print(ggplot(diamonds,aes(diamonds[[i]],price))+geom_bin2d()+xlab(names(diamonds)[i]))
  } else{
    print(ggplot(diamonds,aes(diamonds[[i]],price))+geom_boxplot()+xlab(names(diamonds)[i]))
  }


}
```

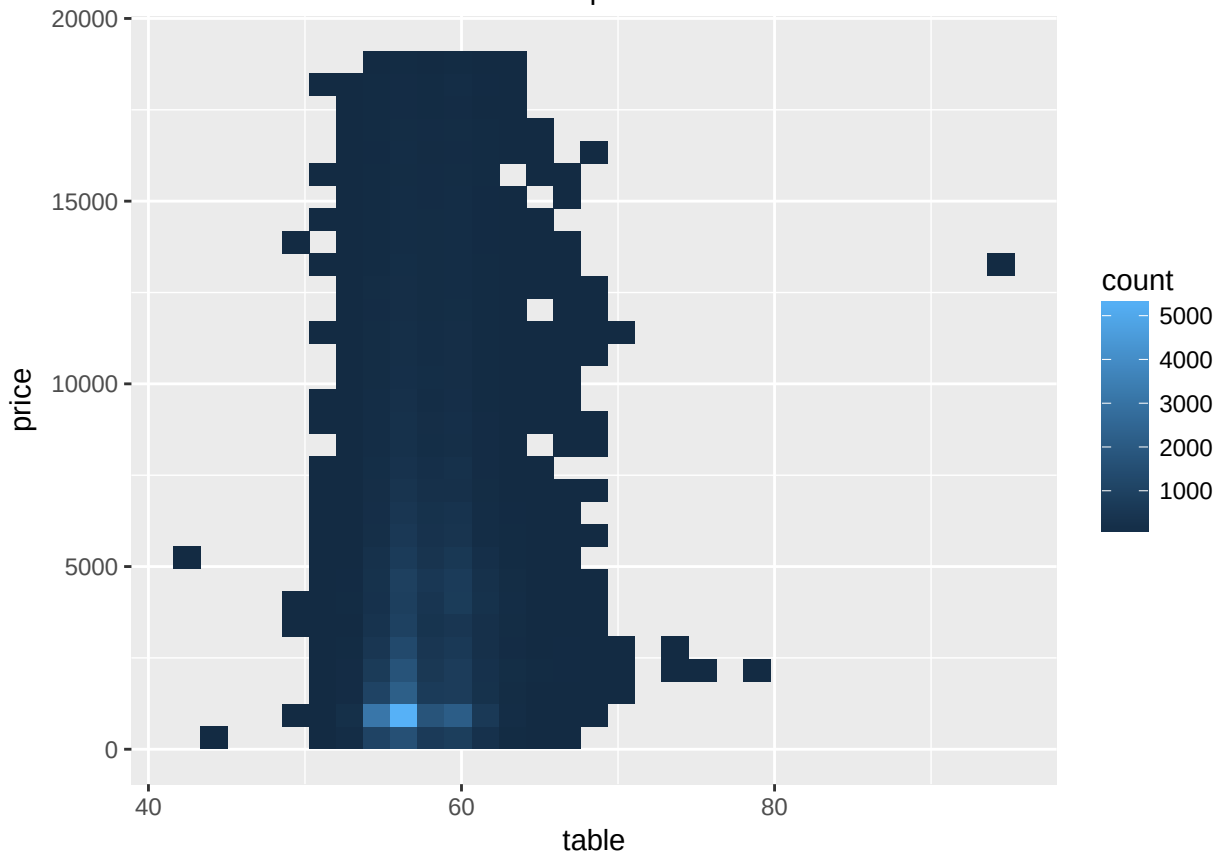

20000
15000
10000
5000
0
price
0
1
2
3
4
5
carat
count
6000
4000
2000

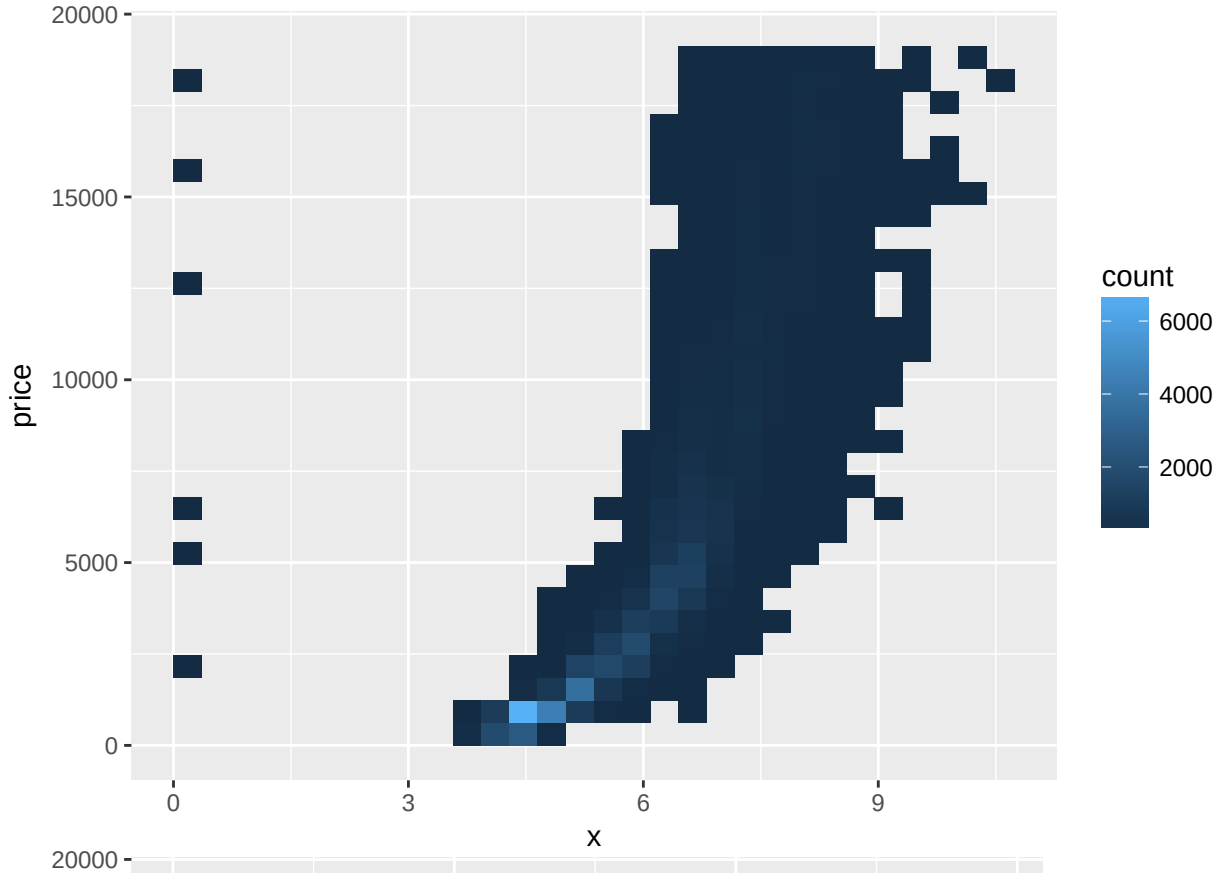

15000
10000
5000
0
price
Fair
Good
Ideal
Premium
Very Good
cut

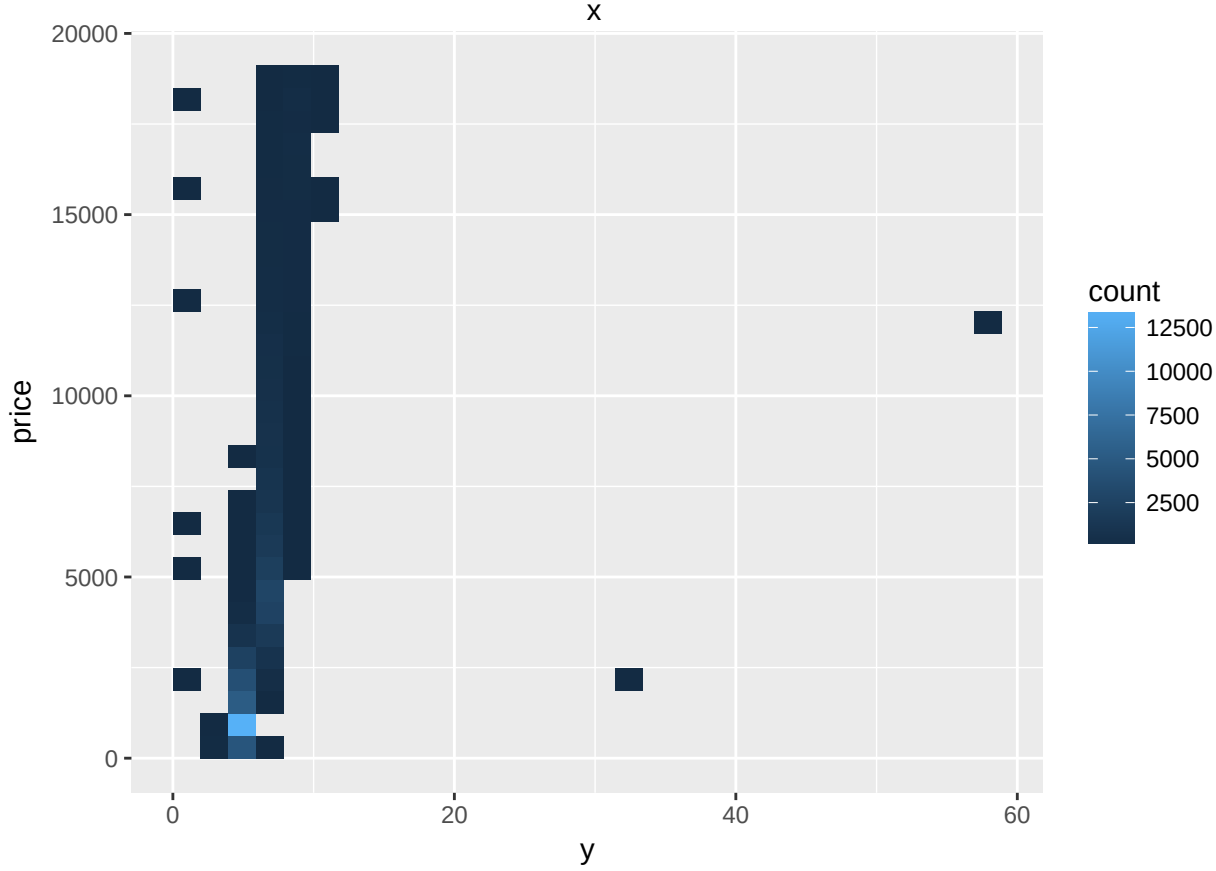

price
15000
10000
5000
0
D
E
F
G
H
I
J
color

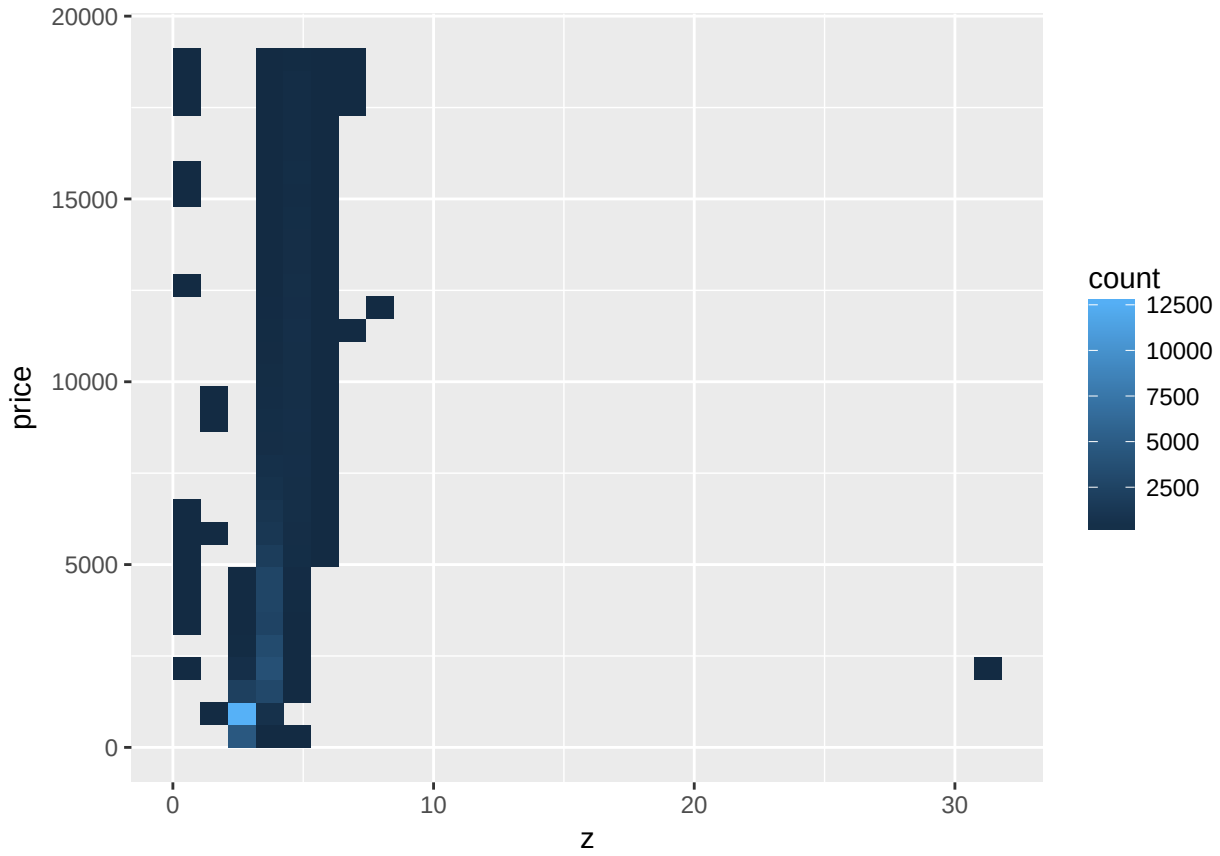

price
15000
10000
5000
0
I1
IF
SI1
SI2
VS1
VS2
VVS1
VVS2
clarity


20000
15000
10000
5000
0
price
50
60
70
80
depth
count
6000
4000
2000


20000
15000
10000
5000
0
price
40
60
80
table
count
5000
4000
3000
2000
1000


price
20000
15000
10000
5000
0
0
3
6
9
x
count
6000
4000
2000


price
20000
15000
10000
5000
0
0
20
40
60
y
count
12500
10000
7500
5000
2500

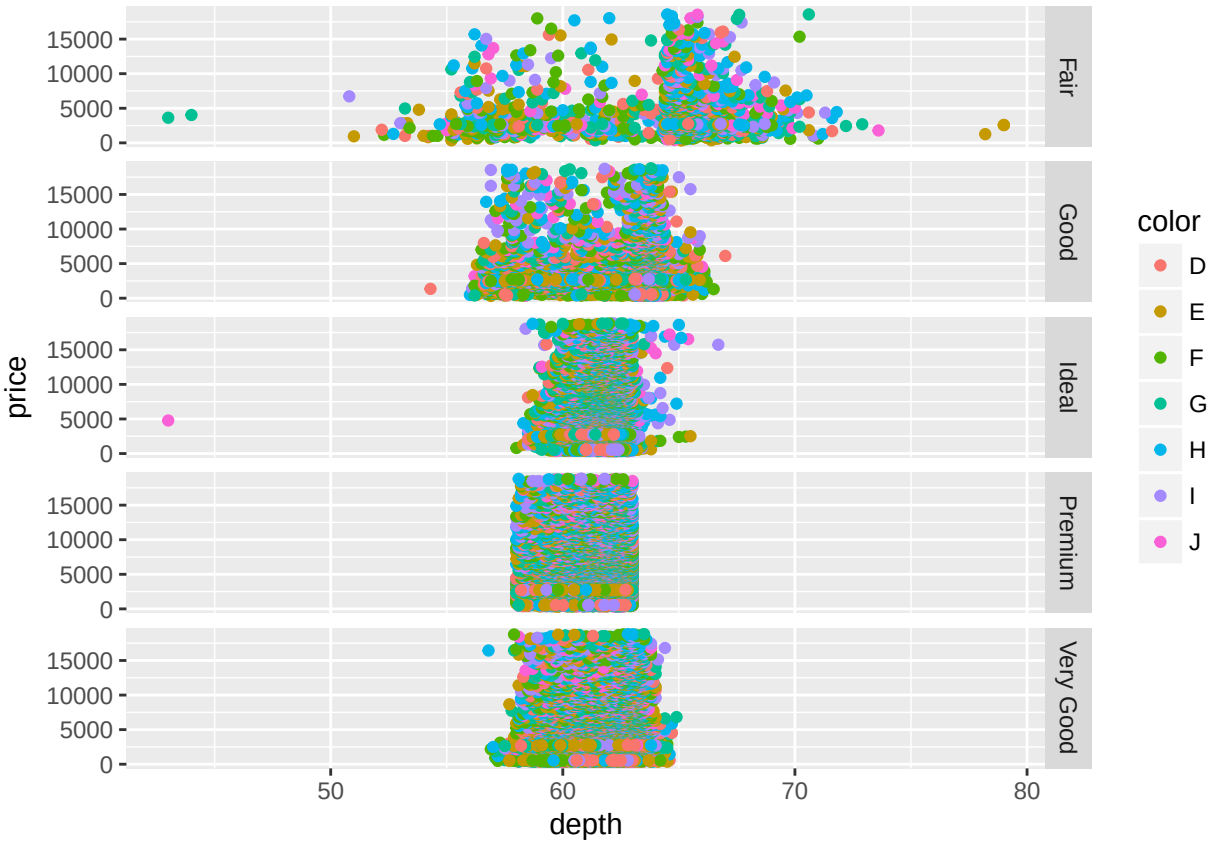


Does depth appear to be mostly independent of price?

Yes

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

```
ggplot(diamonds, aes(x = depth, y = price)) +
  geom_point(size=.1) +
  facet_grid(cut~.)+ geom_point(aes(col = color))
```



Does diamond color appear to be independent of diamond depth?

Yes it appears to be

Does diamond cut appear to be independent of diamond depth?

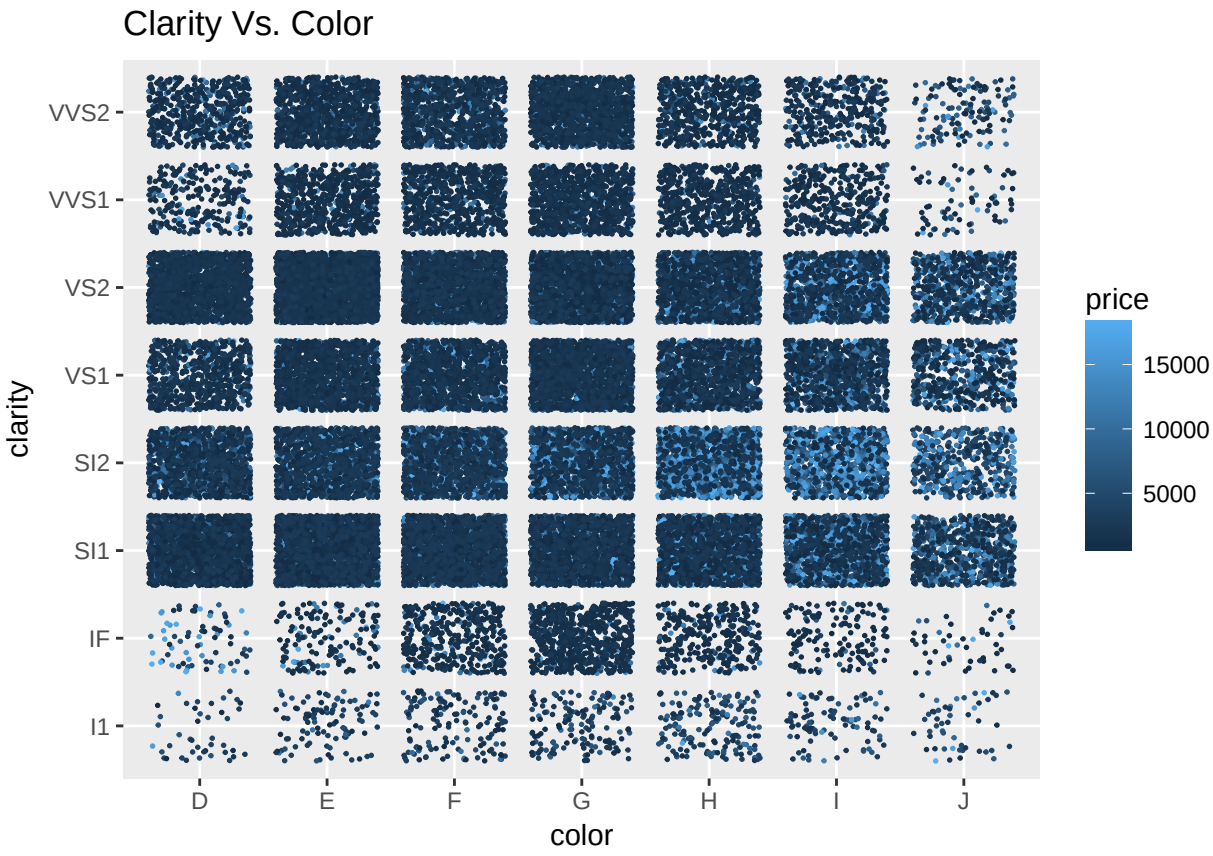
No it doesn't. We can see variation in depth based on the cut. i.e fair vs premium.

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.

```
ggplot(diamonds, aes(color,clarity))+geom_jitter(aes(col=price),size=.3)+ggtitle("Clarity Vs. Color")
```



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.

```
regression1=lm(price~depth,data=diamonds)
regression1
```

```
##
## 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?

```
regression1$coefficients
```

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

```
summary(regression1)$r.squared
```

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

```
summary(regression1)$sigma
```

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

```
stderror1=sd(diamonds$price)
stderror1
```

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

b_0=5763.67 and b_1= -29.65 R^2 is .000113 or .01% RMSE is 3989.251 The standard error of price is 3989.44

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

Yes as we saw that depth and price are relatively independent from each other. Our low R^2 and larger RMSE supports this.

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

```
regression2=lm(price~carat,data=diamonds)
regression2
```

```
##
## 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?

```
regression2$coefficients
```

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

```
summary(regression2)$r.squared
```

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

```
summary(regression2)$sigma
```

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

```
stderror2=sd(diamonds$price)
stderror2
```

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

b_0 is -2256.361 and b_1 is 7756.426 R^2 is .8493 or 84.93% RMSE is 1548.562 standard error is 3989.44

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

Yes we can view that there is a relationship between the two and this is reflected in high R^2 and lower RMSE.

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

```
anova = lm(price ~ color, diamonds)
anova
```

```
##
## 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?

```
anova$coefficients
```

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

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

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

```
summary(anova)$sigma
```

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

```
stderror3=sd(diamonds$price)
stderror3
```

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

our $b's$ are 3169.95410 -93.20162 554.93230 829.18158 1316.71510 1921.92086 2153.86392 R^2 is .5087 or 50.87% RMSE is 3926.777 stderror is 3989.44

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

Yes, our visualization shows that color does change price but it seems that it is not by much. This is displayed in our 50% R^2 and somewhat large RMSE.

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

This is because our feature is a categorical variable with 7 levels.

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.

```
anova
```

```
##
## 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
```

```
D=(subset(diamonds,subset= diamonds$color=="D"))
avg=mean(D$price)
E=(subset(diamonds,subset= diamonds$color=="E"))
mean(E$price)-avg
```

```
## [1] -93.20162
```

```
F=(subset(diamonds,subset= diamonds$color=="F"))
mean(F$price)-avg
```

```
## [1] 554.9323
```

```
G=(subset(diamonds,subset= diamonds$color=="G"))
mean(G$price)-avg
```

```
## [1] 829.1816
```

```
H=(subset(diamonds,subset= diamonds$color=="H"))
mean(H$price)-avg
```

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

```
I=(subset(diamonds,subset= diamonds$color=="I"))
mean(I$price)-avg
```

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

```
J=(subset(diamonds,subset= diamonds$color=="J"))
mean(J$price)-avg
```

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

These sample averages are our b vector. However, since we included the intercept, we would need to subtract the intercept from all others to get the $b's$ to match up correctly.

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

```
anova1 = lm(price ~ 0+color, diamonds)
anova1
```

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

```
anova1$coefficients
```

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

```
D=(subset(diamonds,subset= diamonds$color=="D"))
mean(D$price)
```

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

```
E=(subset(diamonds,subset= diamonds$color=="E"))
mean(E$price)
```

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

```
F=(subset(diamonds,subset= diamonds$color=="F"))
mean(F$price)
```

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

```
G=(subset(diamonds,subset= diamonds$color=="G"))
mean(G$price)
```

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

```
H=(subset(diamonds,subset= diamonds$color=="H"))
mean(H$price)
```

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

```
I=(subset(diamonds,subset= diamonds$color=="I"))
mean(I$price)
```

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

```
J=(subset(diamonds,subset= diamonds$color=="J"))
mean(J$price)
```

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

Here we see that the b vectors match perfectly to our sample averages and we can read them directly from our coefficients since we excluded the intercept.

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

Extrapolation would be perhaps getting data on a new color of diamond that is out of our X space for data. This could result in a poor prediction as it is out of range and could be a drastically different price!

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

```
regression3=lm(price~.,data=diamonds)
regression3
```

```
##
## 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.

```
regression3$coefficients
```

```
##  (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(regression3)$r.squared
```

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

```
summary(regression3)$sigma
```

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

95% of the entries are plus/minus 2*rmse or plus/minus 2260.188 or our real diamond price is in a $4,500 range of our prediction.

Interpret all entries in the vector b. Ceteris Paribus... The intercept allows us to see that a fair cut, D color, I1 clarity diamond is on average $2184.48. On average, an increase in carat by 1 will increase price by $11256.98. Compared to a fair cut diamond, a good cut diamond's price is $579.75 more on average. Compared to a fair cut diamond, an ideal cut diamond's price is $832.91 more on average. Compared to a fair cut diamond, a premium cut diamond's price is $762.14 more on average. Compared to a fair cut diamond, a very good cut diamond's price is $726.78 more on average. Compared to a D color diamond, an E color diamond is $209.12 less on average. Compared to a D color diamond, an F color diamond is $272.85 less on average. Compared to a D color diamond, a G color diamond is $482.04 less on average. Compared to a D color diamond, an H color diamond is $980.27 less on average. Compared to a D color diamond, an I color diamond is $1466.24 less on average. Compared to a D color diamond, a J color diamond is $2369.40 less on average. Compared to clarity I1 diamond a clarity IF diamond is $5345.10 more on average. Compared to clarity I1 diamond a clarity SI1 diamond is $3665.47 more on average. Compared to clarity I1 diamond a clarity SI2 diamond is $2702.59 more on average. Compared to clarity I1 diamond a clarity VS1 diamond is $4578.40 more on average. Compared to clarity I1 diamond a clarity VS2 diamond is $4267.22 more on average. Compared to clarity I1 diamond a clarity VVS1 diamond is $5007.76 more on average. Compared to clarity I1 diamond a clarity VVS2 diamond is $4950.81 more on average. On average, an increase in depth by 1% point will decrease price by $63.81. On average, an increase in table by 1 will decrease price by $26.47. On average, an increase in x by 1 will decrease price by $1008.26. On average, an increase in y by 1 will increase price by $9.61. On average, an increase in z by 1 will decrease price by $50.12.

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

Yes it is. We can make out some of these metrics in the visualizations above. Some are not that clear and less pronounced.

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

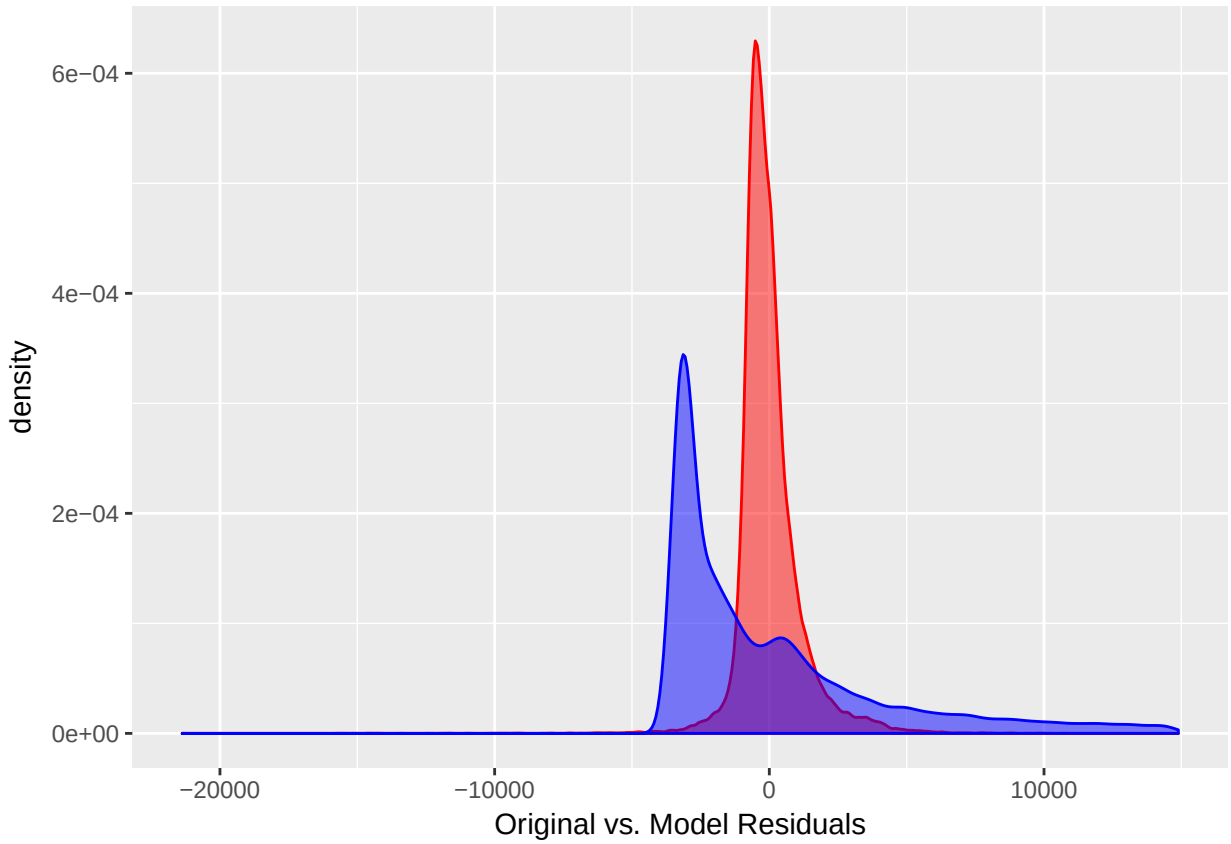
I think R^2 is high because we are including very relevant regressors that explain how price is determined for the diamond market

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

No I do not believe we overfit because we have over 50,000 observations with only 9 features that I believe are closely related to predicting diamond price.

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

```
b=regression3$residuals

original=diamonds$price-mean(diamonds$price)


ggplot(data=diamonds, aes(b))+geom_density(col="red", fill="red", alpha=.5)+geom_density(aes(original),
```



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

Price vs. carat does not appear linear based off of the visualization in Question 2. It appears to be more exponential.

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

```
regression4=lm(price~.+I(diamonds$carat^2),data=diamonds)
```

What is b, R^2 and the RMSE?

```
regression4$coefficients
```

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

```
## I(diamonds$carat^2)
##         -1028.81806
```

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.

Ceteris Paribus...

The intercept allows us to see that a fair cut, D color, I1 clarity diamond is on average $9807.98. On average, an increase in carat by 1 will increase price by $16144.76. Compared to a fair cut diamond, a good cut diamond's price is $538.33 more on average. Compared to a fair cut diamond, an ideal cut diamond's price is $807.52 more on average. Compared to a fair cut diamond, a premium cut diamond's price is $747.70 more on average. Compared to a fair cut diamond, a very good cut diamond's price is $678.32 more on average. Compared to a D color diamond, an E color diamond is $209.45 less on average. Compared to a D color diamond, an F color diamond is $284.55 less on average. Compared to a D color diamond, a G color diamond is $496.85 less on average. Compared to a D color diamond, an H color diamond is $997.60 less on average. Compared to a D color diamond, an I color diamond is $1469.25 less on average. Compared to a D color diamond, a J color diamond is $2357.80 less on average. Compared to clarity I1 diamond a clarity IF diamond is $5243.52 more on average. Compared to clarity I1 diamond a clarity SI1 diamond is $3565.41 more on average. Compared to clarity I1 diamond a clarity SI2 diamond is $2605.54 more on average. Compared to clarity I1 diamond a clarity VS1 diamond is $4475.44 more on average. Compared to clarity I1 diamond a clarity VS2 diamond is $4163.35 more on average. Compared to clarity I1 diamond a clarity VVS1 diamond is $4904.23 more on average. Compared to clarity I1 diamond a clarity VVS2 diamond is $4843.80 more on average. On average, an increase in depth by 1% point will decrease price by $116.23. On average, an increase in table by 1 will decrease price by $36.37. On average, an increase in x by 1 will decrease price by $2123.01. On average, an increase in y by 1 will decrease price by $23.46. On average, an increase in z by 1 will decrease price by $83.11. On average, after a certain carat size, our diamond price starts displaying diminishing returns of $1028.82

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

```
summary(regression3)$r.squared
```

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

```
summary(regression4)$r.squared
```

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

```
summary(regression3)$sigma
```

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

```
summary(regression4)$sigma
```

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

Yes it's an improvement as R^2 increased and RMSE decreased.

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

The inputs should be in following form g(c(number,cut number, color number, clarity number,x number, y number, z number, depth number,table number)) where cut number is 1=good, 2=very good, 3=premium , 4=ideal color number is 1=E,2=F,3=G,4=H,5=I,6=J clarity number is 1=SI1, 2=SI2,3=VS1,4=VS2,5=VSS1,6=VVS2

```
g=function(xnew){
  if(xnew[2]==(1|2|3|4)){next} else{return(NULL)}
  if(xnew[3]==(1|2|3|4|5|6)){next}else{return(NULL)}
  if(xnew[4]==(1|2|3|4|5|6)){next} else{return(NULL)}
  b=coef(poly1)
    cuts=paste("cut",xnew[2],sep="")
```

```
    col=paste("color",xnew[3],sep="")
    cla=paste("clarity",xnew[4],sep="")
    b[cla]
 sum=(b["(Intercept)"]+b[2]*xnew[1]+b[cuts]+b[col]+b[cla]+b[20]*xnew[5]+b[21]*xnew[6]+b[22]*xnew[7]+b[2
 sum
}
```

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

```
lm(price~poly(diamonds$color,2,raw=TRUE),data=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?

It doesn't make sense to square a categorical because they are just dummies that return 0 or 1 when squared which will result in linear dependence.

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.

```
y=diamonds$price
regressionlm=lm(price~.,diamonds)
X=model.matrix(lm(price~.,data = diamonds),diamonds)
b=solve(t(X)%*%X)%*%t(X)%*%y
yhat=X%*%b
e=y-yhat
SSE=t(e)%*%e
p=nrow(X)-ncol(X)
MSE=(1/p)*SSE
RMSE=sqrt(MSE)
s_sq_y = var(y)
s_sq_e = var(e)
Rsq = (s_sq_y - s_sq_e) / s_sq_y
```

What is b, R^2 and the RMSE?

```
b
```

```
##                       [,1]
## (Intercept)   2184.477351
## carat        11256.978308
## cutGood        579.751446
## cutIdeal       832.911845
## cutPremium     762.143950
## cutVery Good   726.782591
## colorE        -209.118085
## colorF        -272.853832
## colorG        -482.038904
## colorH        -980.266675
## colorI       -1466.244474
## colorJ       -2369.398063
## clarityIF     5345.102246
## claritySI1    3665.472080
## claritySI2    2702.586294
## clarityVS1    4578.397915
```

```
## clarityVS2    4267.223565
## clarityVVS1   5007.759045
## clarityVVS2   4950.814072
## depth          -63.806100
## table          -26.474085
## x            -1008.261098
## y                9.608886
## z              -50.118891
```

```
Rsq
```

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

```
RMSE
```

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

Are they the same as in #4?

Yes they are!

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.

```
indices = sample(1 : nrow(X), 2000)
X1 = X[indices, ]
y1 = y[indices]
rm(indices)
qrX = qr(X1)
Q = qr.Q(qrX)
R = qr.R(qrX)
yhatQ= Q %*% t(Q) %*% y1
bq=solve(R)%*%t(Q)%*%y1

bq
```

```
##                       [,1]
## (Intercept)  -40457.55161
## carat         11356.42099
## cutGood         466.45104
## cutIdeal        809.65736
## cutPremium      679.08501
## cutVery Good    626.66404
## colorE         -261.49963
## colorF         -400.88031
## colorG         -533.31775
## colorH         -981.02715
## colorI        -1597.98318
## colorJ        -2538.15012
## clarityIF      5276.45408
## claritySI1     3650.82966
## claritySI2     2810.30333
## clarityVS1     4499.99492
## clarityVS2     4299.26883
## clarityVVS1    4978.22887
## clarityVVS2    4907.37944
```

```
## depth           626.63790
## table           -24.15131
## x              2275.14775
## y              4032.67328
## z            -11897.20211
```

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

```
H=X1%*%solve(t(X1)%*%X1)%*%t(X1)
yhatQ= H %*% y1
e1=y1-yhatQ
```

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)
expect_equal(as.vector(yhat+e),y) #A
expect_equal(sum(t(yhatQ)%*%e1),0,tol=1e-4) #B
```

```
## Error: sum(t(yhatQ) %*% e1) not equal to 0.
## 1/1 mismatches
## [1] -0.167 - 0 == -0.167
```

```
expect_equal(sum(H%*%e1),0,tol=1e-4) #C
expect_equal(as.vector(t(yhatQ)%*%H),as.vector(t(yhatQ)))#D
expect_equal(sum(t(e1)%*%yhatQ),0,tol=1e-4) #E
```

```
## Error: sum(t(e1) %*% yhatQ) not equal to 0.
## 1/1 mismatches
## [1] -0.167 - 0 == -0.167
```

```
y1bar=mean(y1)
SST = sum((y1 - y1bar)^2)
SSR = sum((yhatQ - y1bar)^2)
SSE = sum(e1^2)
SSR + SSE
```

```
## [1] 33089432226
```

```
SST
```

```
## [1] 33089432226
```

```
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.

```
regression8=lm(price~.*.+poly(carat,5,raw=TRUE)+poly(x,5,raw=TRUE)+poly(y,5,raw=TRUE)+poly(z,5,raw=TRUE)


regression8
```

```
##
## Call:
## lm(formula = price ~ . * . + poly(carat, 5, raw = TRUE) + poly(x,
##     5, raw = TRUE) + poly(y, 5, raw = TRUE) + poly(z, 5, raw = TRUE) +
```

```
##     poly(depth, 5, raw = TRUE) + poly(table, 5, raw = TRUE),
##     data = diamonds)
##
## Coefficients:
##                  (Intercept)                        carat
##                    4.054e+05                    6.230e+03
##                      cutGood                     cutIdeal
##                   -4.859e+03                   -2.421e+03
##                   cutPremium                cutVery Good
##                   -3.750e+03                   -6.158e+03
##                       colorE                       colorF
##                   -6.337e+02                   -6.563e+02
##                       colorG                       colorH
##                    4.200e+02                    2.342e+03
##                       colorI                       colorJ
##                    7.890e+03                    1.609e+04
##                    clarityIF                   claritySI1
##                   -8.488e+03                    8.354e+03
##                   claritySI2                   clarityVS1
##                    9.361e+03                    5.044e+03
##                   clarityVS2                  clarityVVS1
##                    6.461e+03                    7.376e+03
##                  clarityVVS2                        depth
##                    1.137e+03                   -2.965e+04
##                        table                            x
##                   -5.231e+03                   -1.832e+04
##                            y                            z
##                    1.637e+04                    9.462e+03
## poly(carat, 5, raw = TRUE)1  poly(carat, 5, raw = TRUE)2
##                             NA                   -5.087e+03
## poly(carat, 5, raw = TRUE)3  poly(carat, 5, raw = TRUE)4
##                    4.757e+03                   -1.363e+03
## poly(carat, 5, raw = TRUE)5      poly(x, 5, raw = TRUE)1
##                    1.394e+02                           NA
##     poly(x, 5, raw = TRUE)2      poly(x, 5, raw = TRUE)3
##                    3.702e+03                   -8.285e+02
##     poly(x, 5, raw = TRUE)4      poly(x, 5, raw = TRUE)5
##                    8.590e+01                   -3.782e+00
##     poly(y, 5, raw = TRUE)1      poly(y, 5, raw = TRUE)2
##                             NA                   -4.242e+03
##     poly(y, 5, raw = TRUE)3      poly(y, 5, raw = TRUE)4
##                    4.138e+02                   -1.337e+01
##     poly(y, 5, raw = TRUE)5      poly(z, 5, raw = TRUE)1
##                    1.269e-01                           NA
##     poly(z, 5, raw = TRUE)2      poly(z, 5, raw = TRUE)3
##                   -2.870e+03                    9.650e+02
##     poly(z, 5, raw = TRUE)4      poly(z, 5, raw = TRUE)5
##                   -1.039e+02                    2.401e+00
## poly(depth, 5, raw = TRUE)1  poly(depth, 5, raw = TRUE)2
##                             NA                    1.024e+03
## poly(depth, 5, raw = TRUE)3  poly(depth, 5, raw = TRUE)4
##                   -1.701e+01                    1.356e-01
## poly(depth, 5, raw = TRUE)5  poly(table, 5, raw = TRUE)1
##                   -4.157e-04                           NA
```

```
## poly(table, 5, raw = TRUE)2  poly(table, 5, raw = TRUE)3
##                    1.238e+02                    -1.355e+00
## poly(table, 5, raw = TRUE)4  poly(table, 5, raw = TRUE)5
##                    6.312e-03                    -8.386e-06
##                carat:cutGood                carat:cutIdeal
##                   -2.287e+02                     5.008e+02
##             carat:cutPremium           carat:cutVery Good
##                    6.003e+02                     1.376e+02
##                carat:colorE                 carat:colorF
##                    1.875e+02                     4.230e+02
##                carat:colorG                 carat:colorH
##                   -5.386e+02                    -1.323e+03
##                carat:colorI                 carat:colorJ
##                   -1.260e+03                    -2.854e+03
##              carat:clarityIF              carat:claritySI1
##                    8.784e+03                     6.550e+03
##              carat:claritySI2              carat:clarityVS1
##                    4.742e+03                     8.094e+03
##              carat:clarityVS2             carat:clarityVVS1
##                    7.656e+03                     1.084e+04
##             carat:clarityVVS2                  carat:depth
##                    9.363e+03                    -5.221e+01
##                 carat:table                        carat:x
##                    4.395e+00                    -1.454e+03
##                   carat:y                         carat:z
##                    8.960e+02                     3.787e+02
##               cutGood:colorE               cutIdeal:colorE
##                    1.221e+01                    -6.949e+01
##            cutPremium:colorE           cutVery Good:colorE
##                   -1.060e+02                    -5.184e+01
##               cutGood:colorF               cutIdeal:colorF
##                   -4.395e+01                    -2.485e+01
##            cutPremium:colorF           cutVery Good:colorF
##                   -1.041e+02                    -2.899e+01
##               cutGood:colorG               cutIdeal:colorG
##                    1.388e+01                    -4.027e+01
##            cutPremium:colorG           cutVery Good:colorG
##                   -9.727e+01                    -1.632e+01
##               cutGood:colorH               cutIdeal:colorH
##                   -7.056e+01                    -2.203e+02
##            cutPremium:colorH           cutVery Good:colorH
##                   -3.497e+02                    -1.588e+02
##               cutGood:colorI               cutIdeal:colorI
##                   -2.810e+02                    -4.437e+02
##            cutPremium:colorI           cutVery Good:colorI
##                   -5.560e+02                    -3.144e+02
##               cutGood:colorJ               cutIdeal:colorJ
##                   -2.820e+02                    -4.631e+02
##            cutPremium:colorJ           cutVery Good:colorJ
##                   -5.658e+02                    -3.431e+02
##            cutGood:clarityIF            cutIdeal:clarityIF
##                    1.284e+03                     1.055e+03
##         cutPremium:clarityIF        cutVery Good:clarityIF
##                    1.391e+03                     1.503e+03
```

```
##          cutGood:claritySI1          cutIdeal:claritySI1
##                   -1.049e+01                   -3.745e+02
##       cutPremium:claritySI1      cutVery Good:claritySI1
##                    2.818e+01                    7.656e+01
##          cutGood:claritySI2          cutIdeal:claritySI2
##                   -1.592e+01                   -4.219e+02
##       cutPremium:claritySI2      cutVery Good:claritySI2
##                    5.259e+01                    4.263e+01
##          cutGood:clarityVS1          cutIdeal:clarityVS1
##                    1.870e+02                   -1.140e+02
##       cutPremium:clarityVS1      cutVery Good:clarityVS1
##                    3.136e+02                    2.821e+02
##          cutGood:clarityVS2          cutIdeal:clarityVS2
##                    2.672e+01                   -2.773e+02
##       cutPremium:clarityVS2      cutVery Good:clarityVS2
##                    1.645e+02                    1.490e+02
##         cutGood:clarityVVS1         cutIdeal:clarityVVS1
##                   -1.904e+02                   -4.282e+02
##      cutPremium:clarityVVS1     cutVery Good:clarityVVS1
##                    3.692e+01                    1.325e+00
##         cutGood:clarityVVS2         cutIdeal:clarityVVS2
##                    1.865e+02                   -4.217e+01
##      cutPremium:clarityVVS2     cutVery Good:clarityVVS2
##                    3.662e+02                    3.455e+02
##               cutGood:depth               cutIdeal:depth
##                    5.279e+01                    1.690e+01
##            cutPremium:depth           cutVery Good:depth
##                    5.632e+01                    7.983e+01
##               cutGood:table               cutIdeal:table
##                    3.090e+00                    6.431e+00
##            cutPremium:table           cutVery Good:table
##                   -8.201e+00                   -2.543e+00
##                    cutGood:x                    cutIdeal:x
##                   -1.417e+03                   -1.888e+03
##                 cutPremium:x                cutVery Good:x
##                   -8.365e+02                   -7.837e+02
##                    cutGood:y                    cutIdeal:y
##                    1.986e+03                    1.826e+03
##                 cutPremium:y                cutVery Good:y
##                    9.612e+02                    1.436e+03
##                    cutGood:z                    cutIdeal:z
##                   -4.343e+02                    4.846e+02
##                 cutPremium:z                cutVery Good:z
##                   -5.835e+01                   -6.641e+02
##            colorE:clarityIF             colorF:clarityIF
##                   -2.378e+03                   -2.793e+03
##            colorG:clarityIF             colorH:clarityIF
##                   -3.692e+03                   -4.954e+03
##            colorI:clarityIF             colorJ:clarityIF
##                   -6.002e+03                   -7.705e+03
##           colorE:claritySI1            colorF:claritySI1
##                   -1.326e+01                   -1.679e+02
##           colorG:claritySI1            colorH:claritySI1
##                   -4.906e+02                   -1.110e+03
```

```
##          colorI:claritySI1          colorJ:claritySI1
##                -1.828e+03                -2.893e+03
##          colorE:claritySI2          colorF:claritySI2
##                 8.615e+01                 8.260e+00
##          colorG:claritySI2          colorH:claritySI2
##                -1.792e+02                -4.798e+02
##          colorI:claritySI2          colorJ:claritySI2
##                -8.828e+02                -1.355e+03
##          colorE:clarityVS1          colorF:clarityVS1
##                -1.577e+01                -1.599e+02
##          colorG:clarityVS1          colorH:clarityVS1
##                -6.249e+02                -1.831e+03
##          colorI:clarityVS1          colorJ:clarityVS1
##                -2.804e+03                -4.171e+03
##          colorE:clarityVS2          colorF:clarityVS2
##                -3.431e+01                -1.053e+02
##          colorG:clarityVS2          colorH:clarityVS2
##                -5.751e+02                -1.689e+03
##          colorI:clarityVS2          colorJ:clarityVS2
##                -2.592e+03                -3.868e+03
##         colorE:clarityVVS1         colorF:clarityVVS1
##                -2.094e+02                -4.080e+02
##         colorG:clarityVVS1         colorH:clarityVVS1
##                -1.184e+03                -2.360e+03
##         colorI:clarityVVS1         colorJ:clarityVVS1
##                -3.395e+03                -5.216e+03
##         colorE:clarityVVS2         colorF:clarityVVS2
##                -2.438e+02                -4.505e+02
##         colorG:clarityVVS2         colorH:clarityVVS2
##                -1.104e+03                -2.408e+03
##         colorI:clarityVVS2         colorJ:clarityVVS2
##                -3.431e+03                -5.131e+03
##               colorE:depth               colorF:depth
##                  1.729e+01                  2.413e+01
##               colorG:depth               colorH:depth
##                  1.918e+01                  1.795e+01
##               colorI:depth               colorJ:depth
##                 -1.106e+01                 -1.145e+02
##               colorE:table               colorF:table
##                  6.595e+00                  1.331e+01
##               colorG:table               colorH:table
##                  1.296e+01                  1.831e+01
##               colorI:table               colorJ:table
##                  2.725e+00                  6.966e-01
##                   colorE:x                   colorF:x
##                  3.922e+02                  1.094e+02
##                   colorG:x                   colorH:x
##                  3.021e+02                  6.157e+02
##                   colorI:x                   colorJ:x
##                  6.798e+02                  9.674e+02
##                   colorE:y                   colorF:y
##                  3.276e+00                  2.695e+02
##                   colorG:y                   colorH:y
##                 -1.294e+02                 -3.686e+02
```

```
##                  colorI:y                  colorJ:y
##                -8.366e+02                -2.413e+03
##                  colorE:z                  colorF:z
##                -9.404e+02                -1.205e+03
##                  colorG:z                  colorH:z
##                -8.408e+02                -1.205e+03
##                  colorI:z                  colorJ:z
##                -1.202e+03                 9.872e+02
##             clarityIF:depth           claritySI1:depth
##                    1.617e+02                 -7.211e+01
##            claritySI2:depth           clarityVS1:depth
##                   -8.744e+01                 -2.868e+01
##            clarityVS2:depth          clarityVVS1:depth
##                   -4.552e+01                 -3.536e+01
##           clarityVVS2:depth            clarityIF:table
##                    2.714e+01                 -6.155e+01
##            claritySI1:table           claritySI2:table
##                   -5.992e+01                 -5.755e+01
##            clarityVS1:table           clarityVS2:table
##                   -7.094e+01                 -6.565e+01
##           clarityVVS1:table          clarityVVS2:table
##                   -8.197e+01                 -6.313e+01
##                 clarityIF:x                claritySI1:x
##                    4.551e+03                  1.746e+03
##                claritySI2:x               clarityVS1:x
##                    4.796e+02                  1.784e+03
##                clarityVS2:x              clarityVVS1:x
##                    1.744e+03                  2.118e+03
##               clarityVVS2:x                clarityIF:y
##                    2.205e+03                 -2.099e+03
##                claritySI1:y               claritySI2:y
##                   -2.169e+03                 -1.112e+03
##                clarityVS1:y               clarityVS2:y
##                   -1.776e+03                 -2.020e+03
##               clarityVVS1:y              clarityVVS2:y
##                   -2.121e+03                 -1.522e+03
##                 clarityIF:z                claritySI1:z
##                   -3.649e+03                 -3.724e+02
##                claritySI2:z               clarityVS1:z
##                    8.304e+01                 -7.941e+02
##                clarityVS2:z              clarityVVS1:z
##                   -4.983e+02                 -1.253e+03
##               clarityVVS2:z                depth:table
##                   -1.888e+03                  2.843e+00
##                     depth:x                    depth:y
##                    3.092e+01                  2.707e+01
##                     depth:z                    table:x
##                   -7.032e+01                  1.250e+02
##                     table:y                    table:z
##                   -1.158e+02                 -2.342e+01
##                         x:y                        x:z
##                    9.268e+02                 -2.373e+02
##                         y:z
##                    7.330e+01
```

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

```
summary(regression8)$r.squared
```

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

```
summary(regression8)$sigma
```

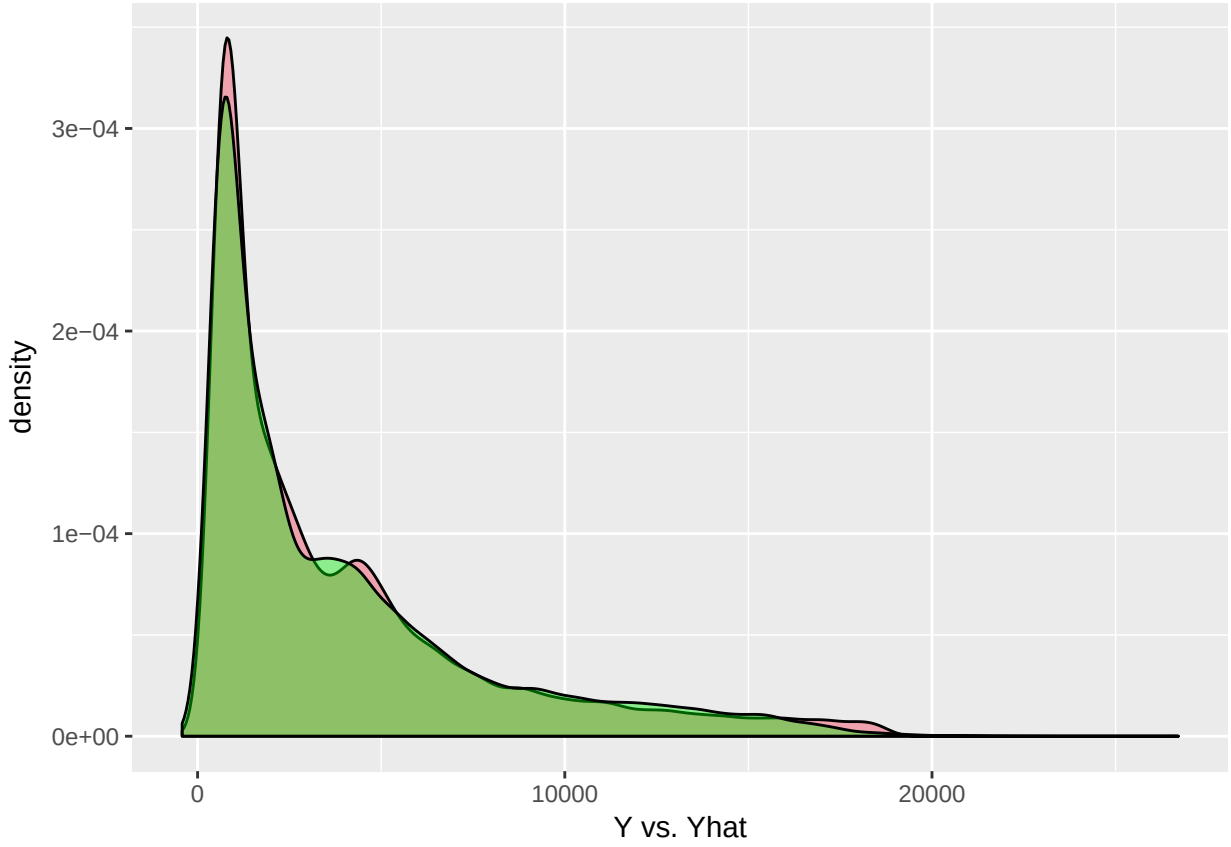
```
## [1] 659.2249
```

```
sd(regression8$residuals)
```

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

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

```
e8=regression8$resid

y8=diamonds$price
yhat8=(y8-e8)
ggplot(data=diamonds,aes(y8))+geom_density(fill="lightpink2",alpha=1)+geom_density(aes(yhat8),fill="gre
```



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

```
count8=0
for(i in 1:nrow(diamonds)){
  if((y8[i]-yhat8[i])>=1000 | (y8[i]-yhat8[i])<=-1000)
    count8=count8+1
  }
count8
```

```
## [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.

RMSE is 672.9627 which by using the empirical model is plus/minus 2*673=1346. This gives us a range of $$$2692 for our prediction of the diamond price. This isn't very good considering the mean diamond price is $$$3932.8

What is the degrees of freedom in this model?

```
length((regression8)$coefficients)
```

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

However, it might be 6 less than that due to the 6 poly of degree 1 terms that are duplicated which creates linear dependence.

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

Not necessarily as we have expanded our H set to include more than just lines. This means our new g is now something different. That doesn't mean we are closer to h^* as h^* has now changed to be more than just lines.

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

Yes we are moving closer to f because our model now has expanded to fit more than just lines– things like polynomials and interaction terms. We therefore are moving closer to being able to fit to f better. We are getting more complex.

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

Total diamond supply and demand for a given year. Phosphorescence. If it's synthetic or natural(if it's impossible to tell, then it should go down to the next question as information that could help).

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.

People's individual preferences about diamond shape, color, carat, etc.....

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)
train_indices = sample((setdiff(1 : n, test_indices)))

diamonds_train=diamonds[train_indices,]
diamonds_test=diamonds[test_indices,]
rm(test_indices,train_indices)



regression_train=lm(price~.*.+poly(carat,5,raw=TRUE)+poly(x,5,raw=TRUE)+poly(y,5,raw=TRUE)+poly(z,5,raw=

y_test=diamonds_test$price


y_hat_train=predict(regression_train,diamonds_test)
```

```
## Warning in predict.lm(regression_train, diamonds_test): prediction from a
## rank-deficient fit may be misleading
```

```
oosresiduals=sd(y_test-y_hat_train)
oosresiduals
```

```
## [1] 1289.858
```

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 I don't think theres overfitting as our oos residuals are close to our in sample residuals. They should be a little higher since creating our final g, we remove some estimation error by using more n. We would need a lot of degrees of freedom to overfit for over 50,000 observations I believe.

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:

```
total=rep(0,100)
for(p in 1:100){
  XX = matrix(c(x, rnorm(n * p)), ncol = p+1)
  X=as.data.frame(XX)
  K = 10
  testindex = sample(1 : n, size = n * 1 / K)
  trainindex = sample((setdiff(1 : n, testindex)))
  train=X[trainindex,]
  test= X[testindex,]
  ytrain=y[trainindex]
  ytest=y[testindex]
```

```r
rm(testindex,trainindex)

mod=lm(ytrain~.,train)
y_hat_oos = predict(mod, test)
oose=sd(ytest-y_hat_oos)
total[p]=oose

}
```

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

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

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

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

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

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

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

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

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

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

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

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

```r
total
```

```
##   [1]  1.6025653  1.1185090  1.2076205  0.9830602  1.1904426  1.2440194
##   [7]  1.4894051  1.4820439  1.3051065  1.0671769  1.1719074  1.2342231
##  [13]  1.2624284  0.8447519  1.1668474  0.9941222  1.6065389  0.8455472
##  [19]  1.3609121  1.5417182  1.4000576  1.0571265  0.9193502  1.3612588
##  [25]  1.1044001  1.2552116  1.9089554  1.1747716  1.5564625  1.3300331
##  [31]  2.1429253  0.9855483  0.8819150  0.7736053  1.4443179  1.9621217
##  [37]  1.0035524  0.9170596  1.2729892  1.6531014  1.3618136  1.4865604
##  [43]  2.0314095  1.2965171  2.3490197  1.1212855  2.0264717  2.3701738
```

```
##  [49]  1.7463930  1.8063244  1.1295992  2.7263814  2.0542179  1.5122838
##  [55]  2.6064978  1.7841266  2.0215423  2.0248213  1.9723193  2.3819699
##  [61]  0.8608265  1.2193403  2.6911385  3.3121550  1.8774050  2.2159971
##  [67]  2.3463006  2.0674036  2.1075791  3.1221284  2.5764568  3.0890694
##  [73]  2.6518045  1.7684177  3.0232122  2.9164130  4.0585416  3.1882017
##  [79]  2.0872358  3.3583247  2.8860323  7.1991646  3.3127373  3.0412318
##  [85] 13.0938915  2.8914350 60.6927765  8.8687830 68.0978298  7.5775654
##  [91] 29.6262187 29.8090232 27.4924767 17.3711330  7.7895626  6.9061769
##  [97]  5.2699336 25.0928087  7.5447032  5.2719697
```

Running this code, we can see that it looks like we start overfitting with around 70-80 fake predictors.