HW03p

Launy Schweiger

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

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. It is a data set containing information about different diamonds attributes such as quality of cut, carat color, size, and the price

```
?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 ...
```

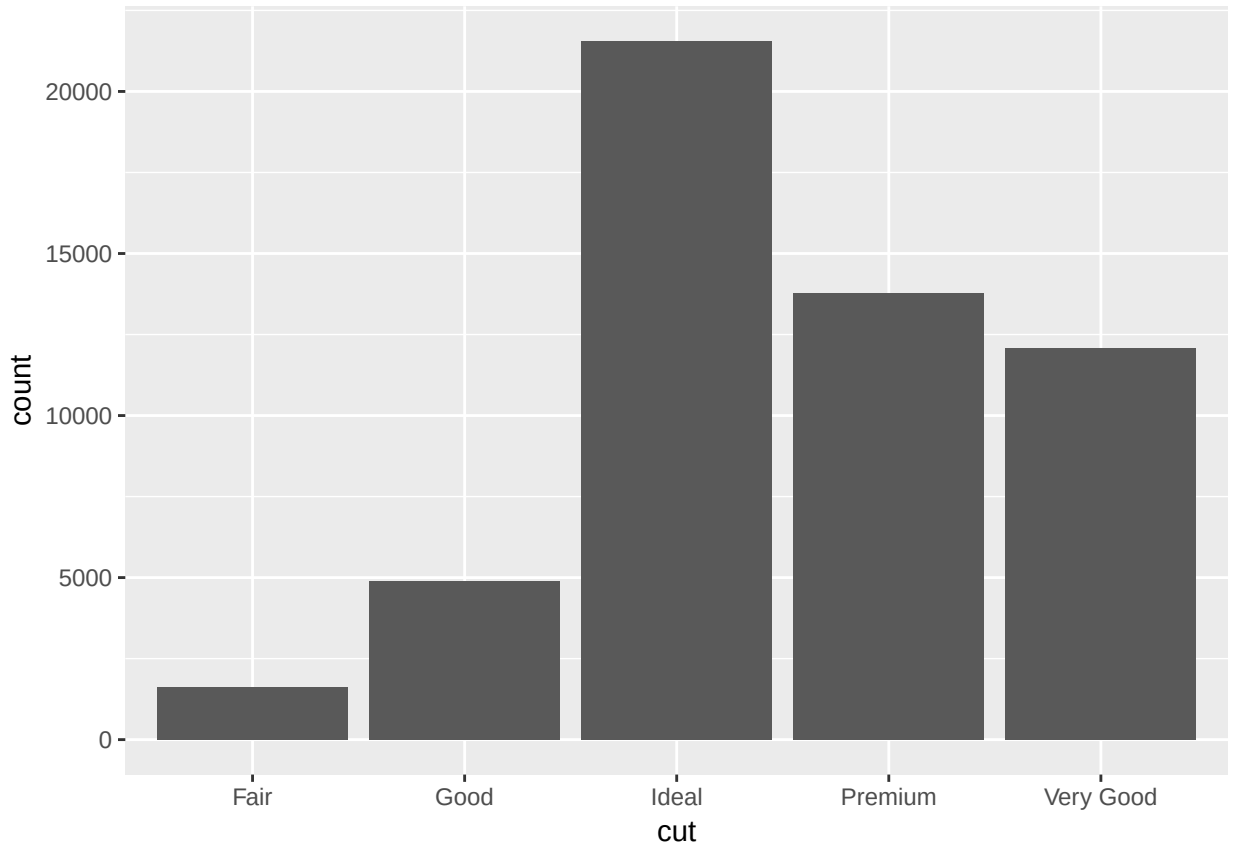
What is n, p, what do the features mean, what is the most likely response metric and why?

n= 53940 p=10 price since that is how we determine if the diamonds is "good" or not.

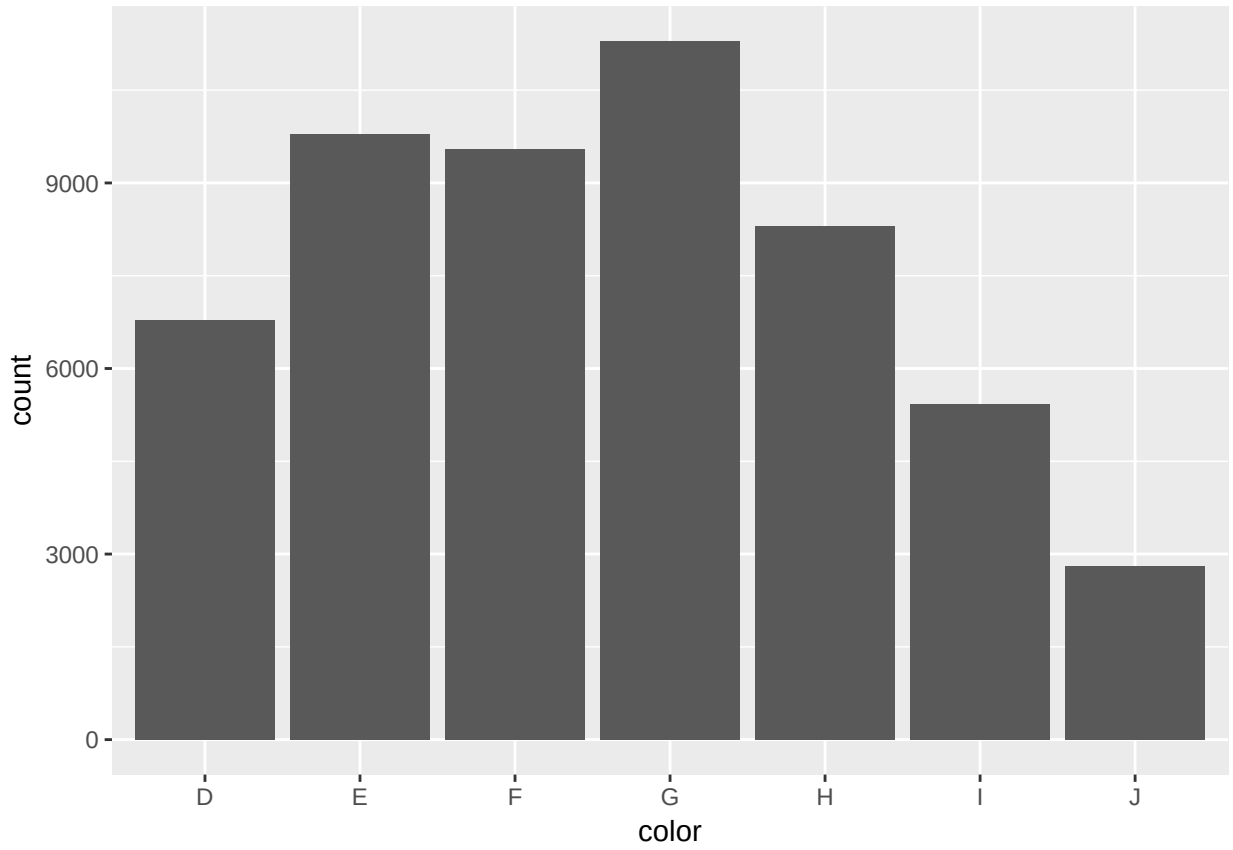
Regardless of what you wrote above, the variable `price` will be the response variable going forward.

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

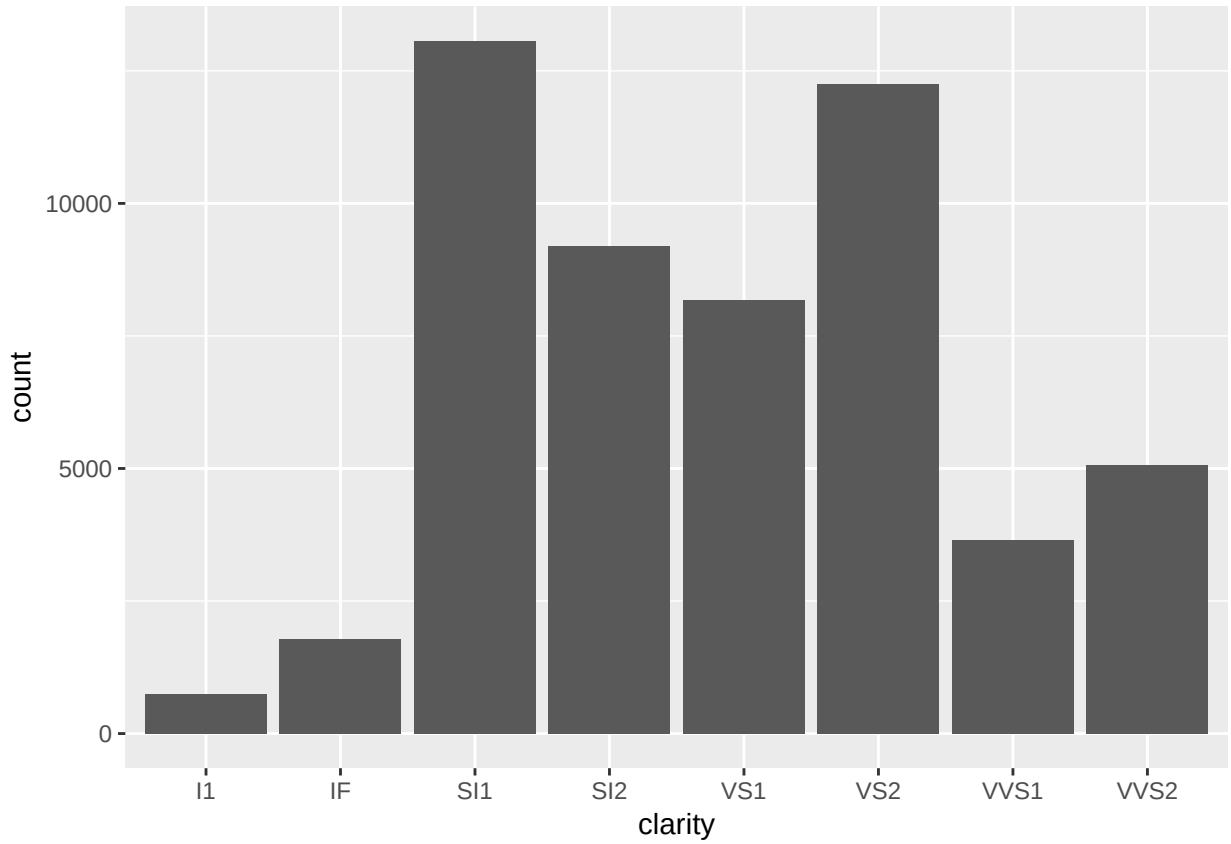
```
ggplot(data=diamonds)+ geom_bar(mapping=aes(x=cut))
```



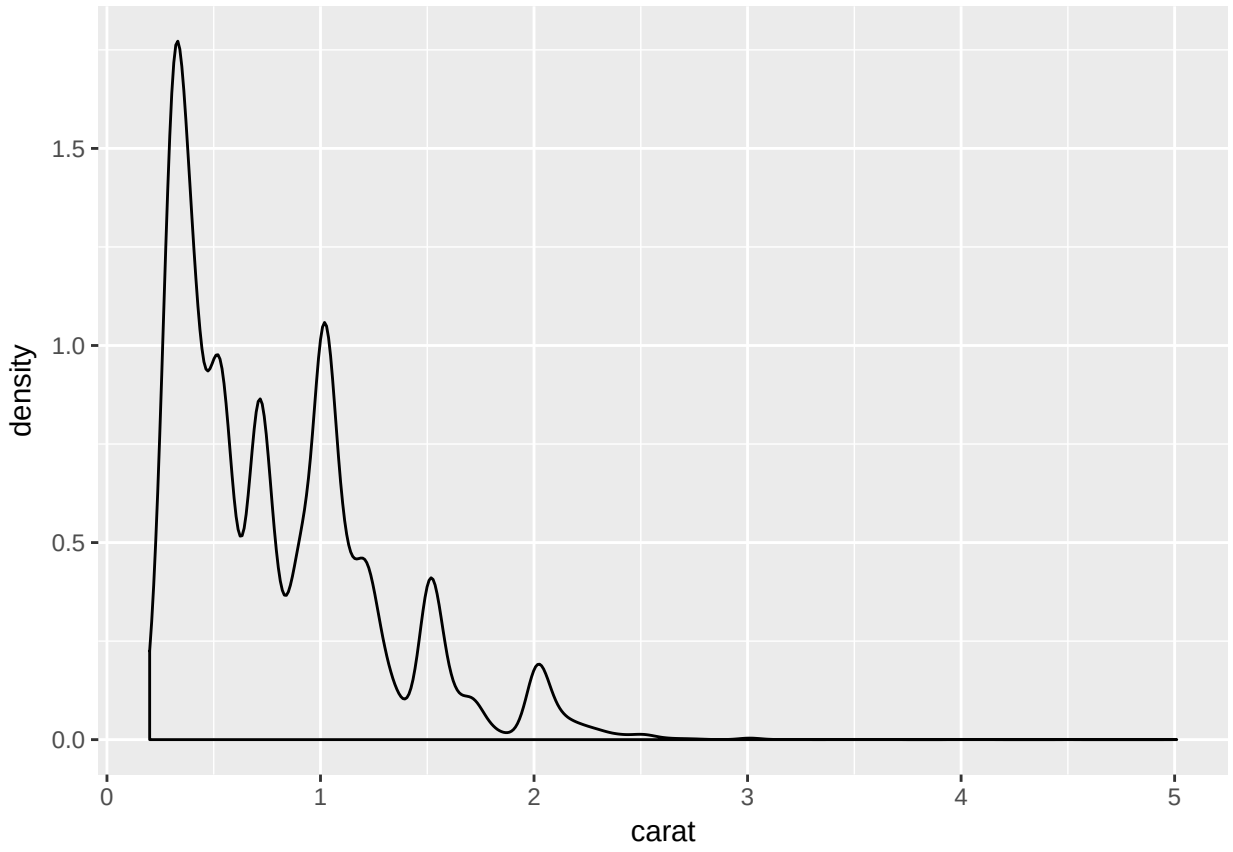

```
ggplot(data=diamonds)+ geom_bar(mapping=aes(x=color))
```



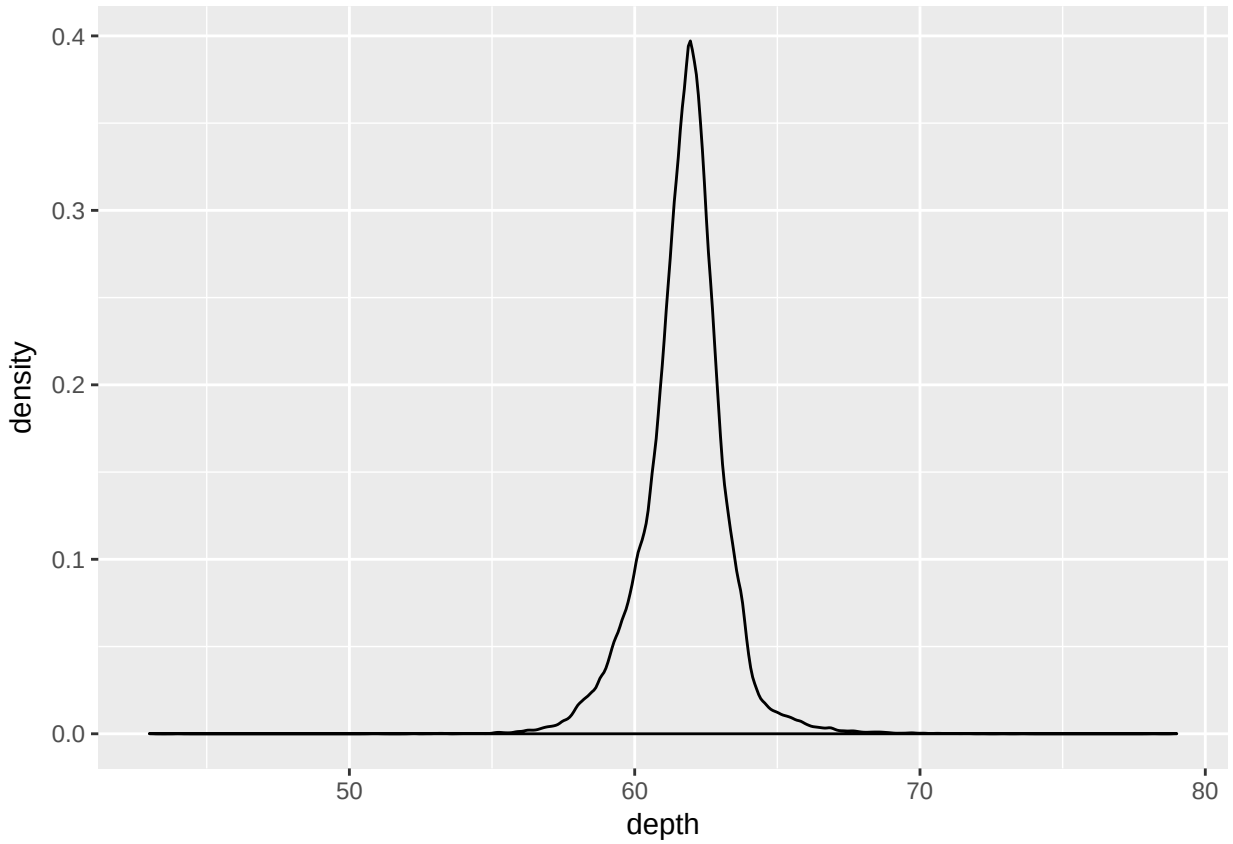

```
ggplot(data=diamonds)+ geom_bar(mapping=aes(x=clarity))
```



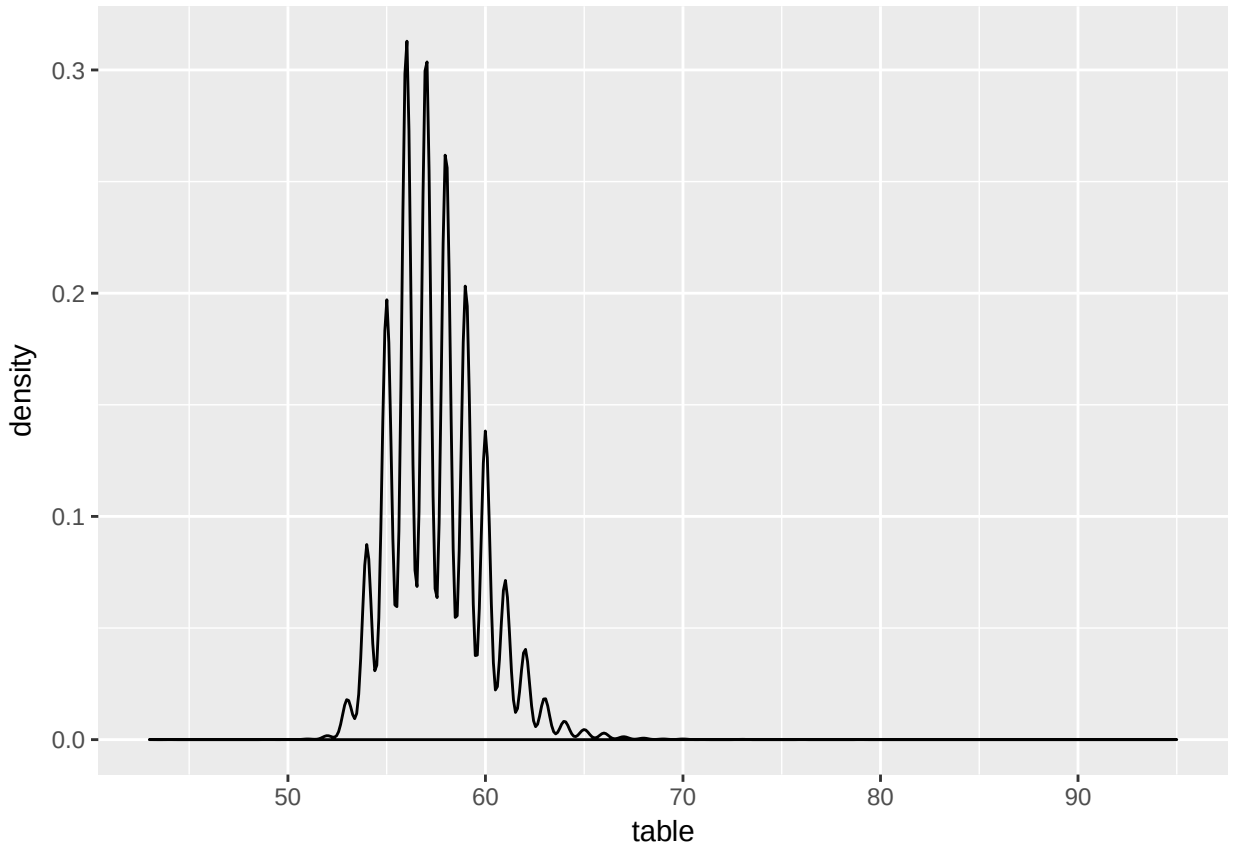

```
ggplot(data=diamonds)+ geom_density(mapping=aes(x=carat))
```



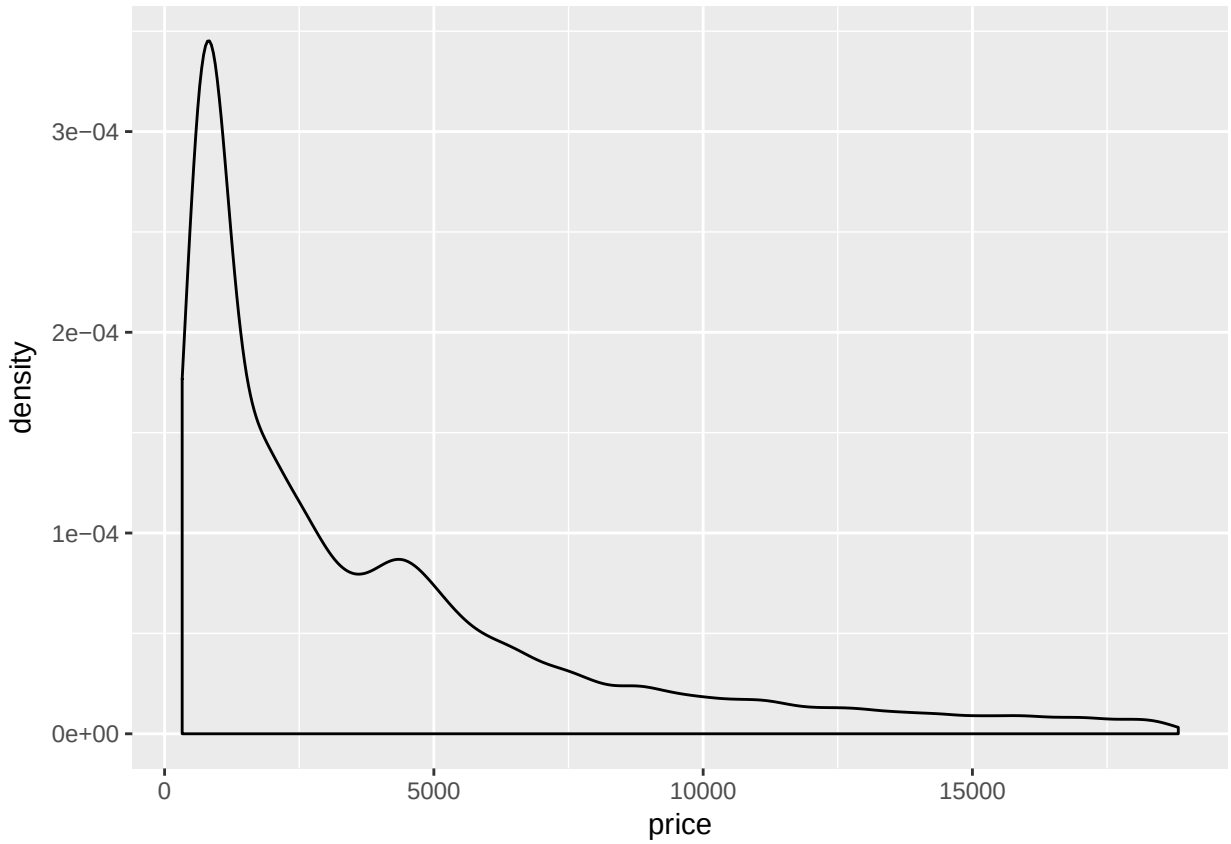

```
ggplot(data=diamonds)+ geom_density(mapping=aes(x=depth))
```



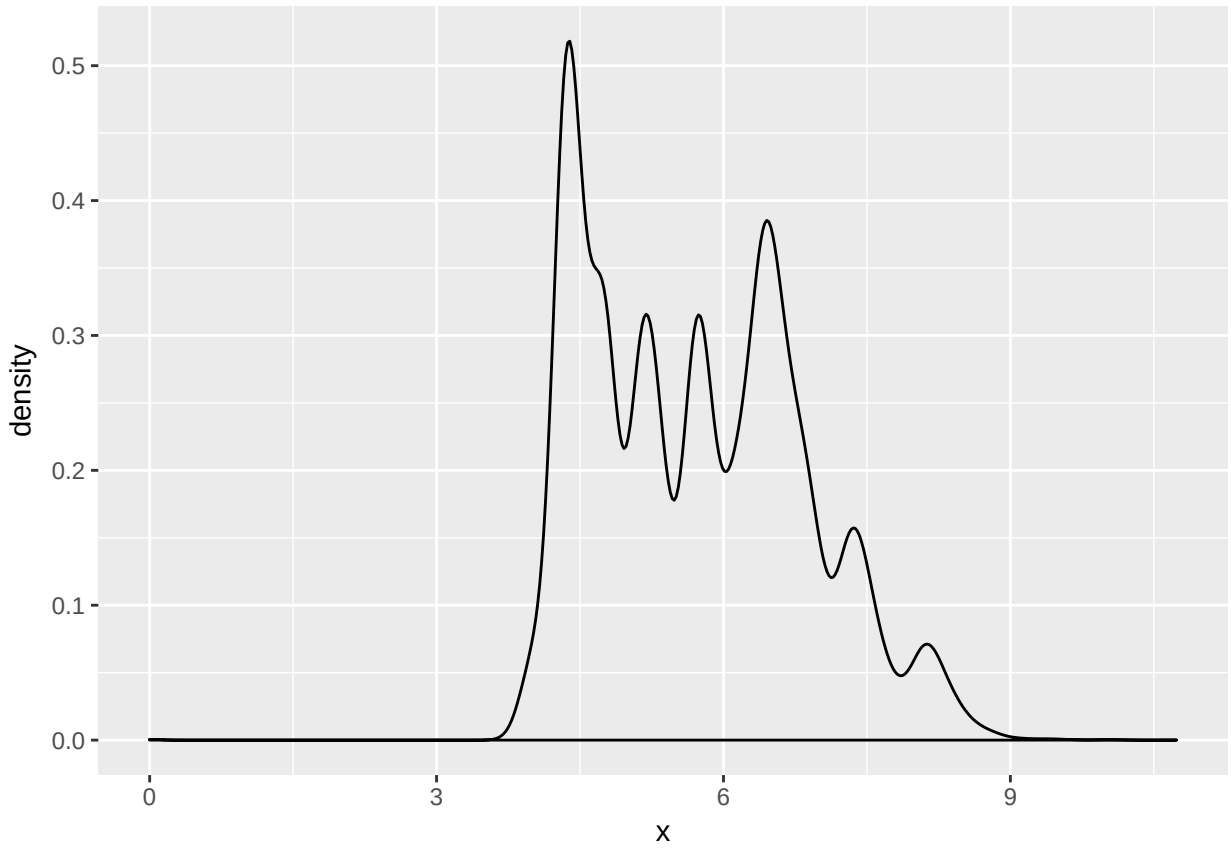

```
ggplot(data=diamonds)+ geom_density(mapping=aes(x=table))
```



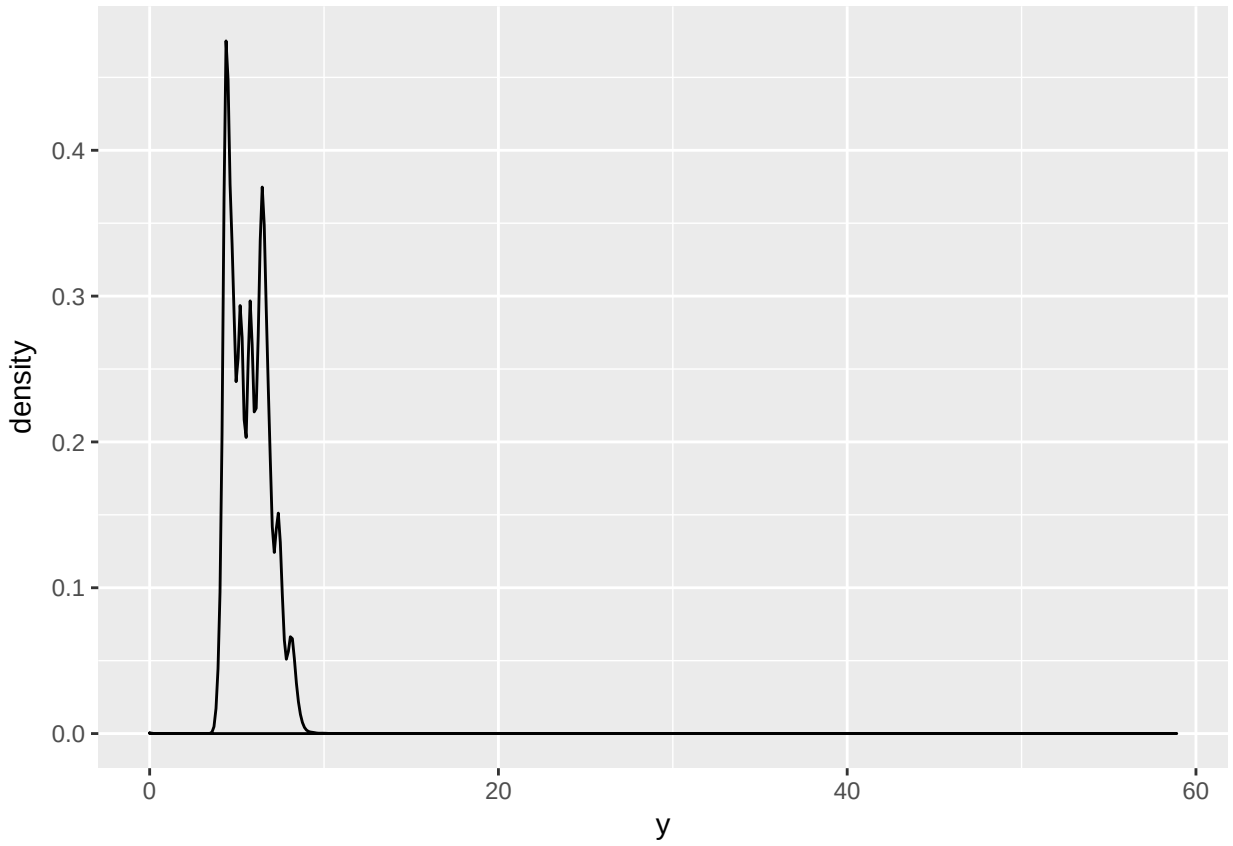

```
ggplot(data=diamonds)+ geom_density(mapping=aes(x=price))
```



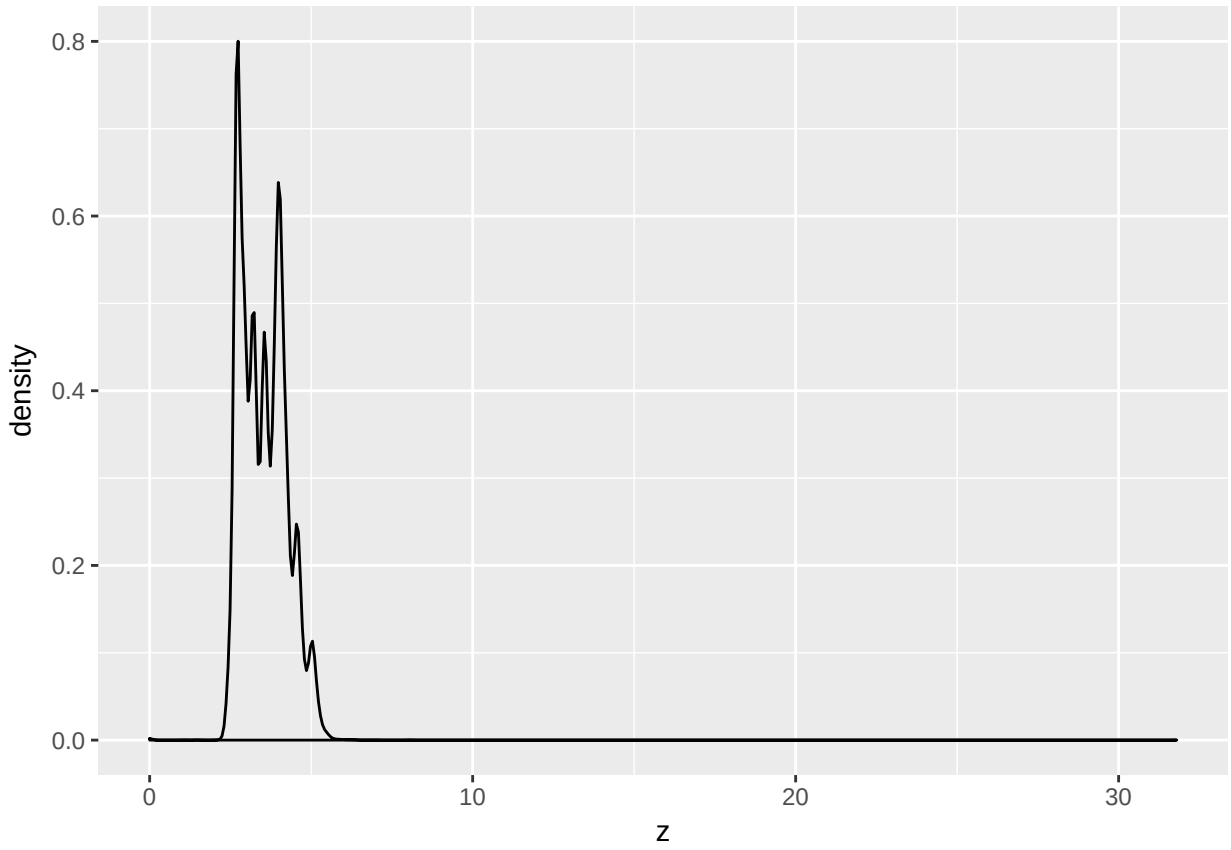

```
ggplot(data=diamonds)+ geom_density(mapping=aes(x=x))
```




```
ggplot(data=diamonds)+ geom_density(mapping=aes(x=y))
```

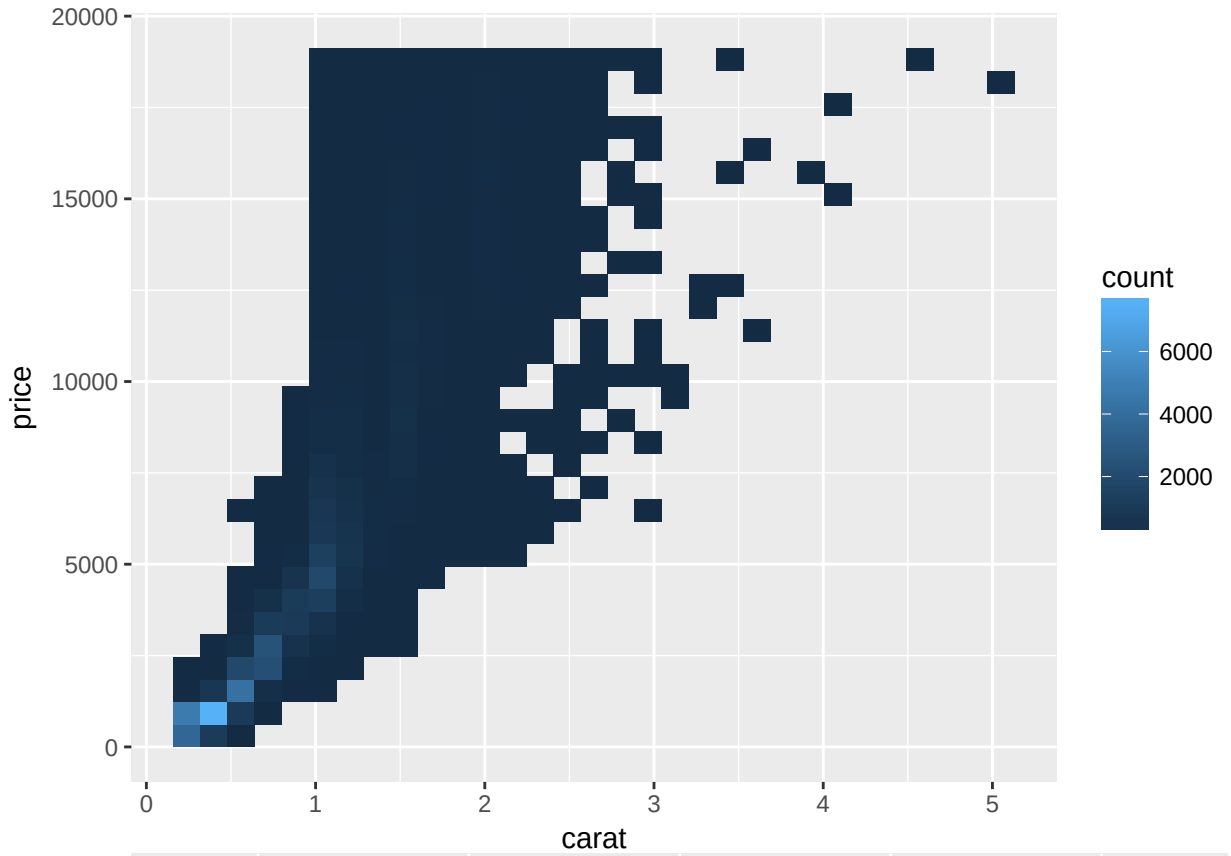


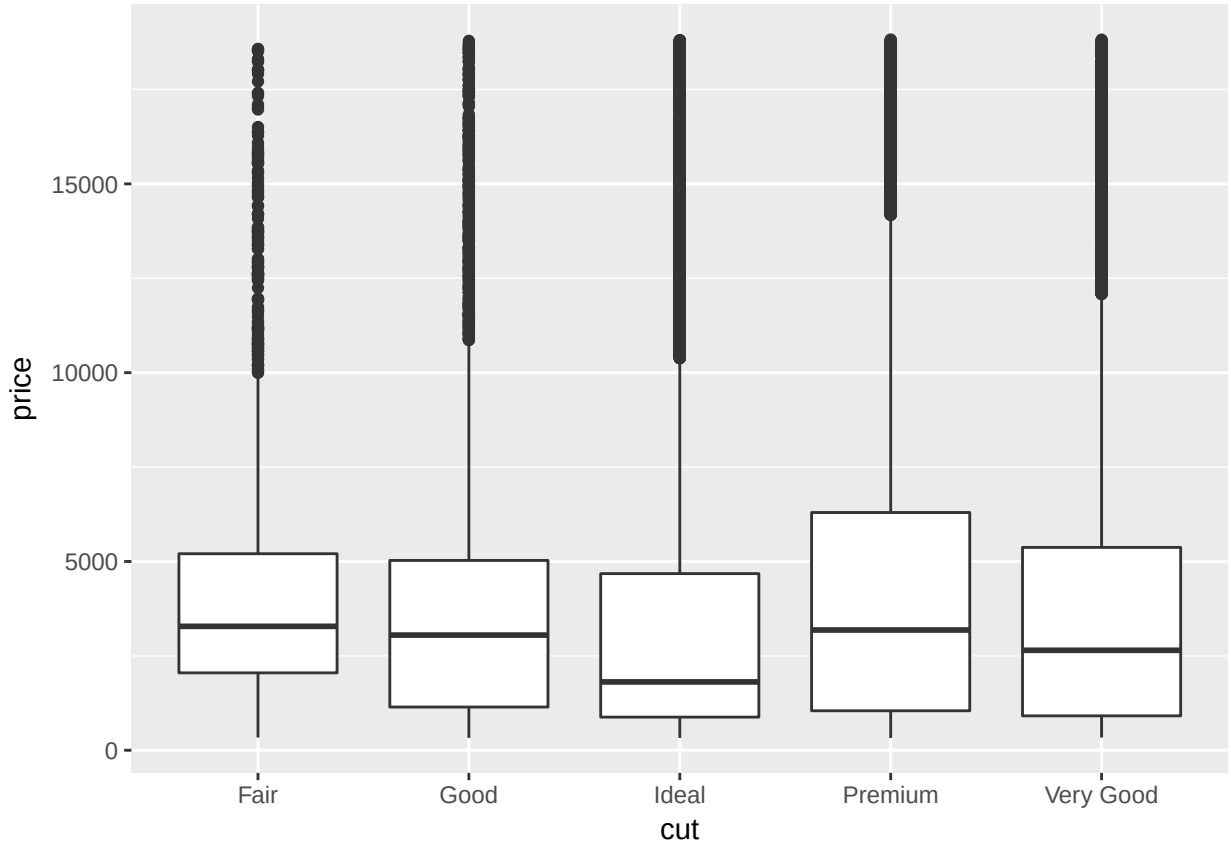

```
ggplot(data=diamonds)+ geom_density(mapping=aes(x=z))
```

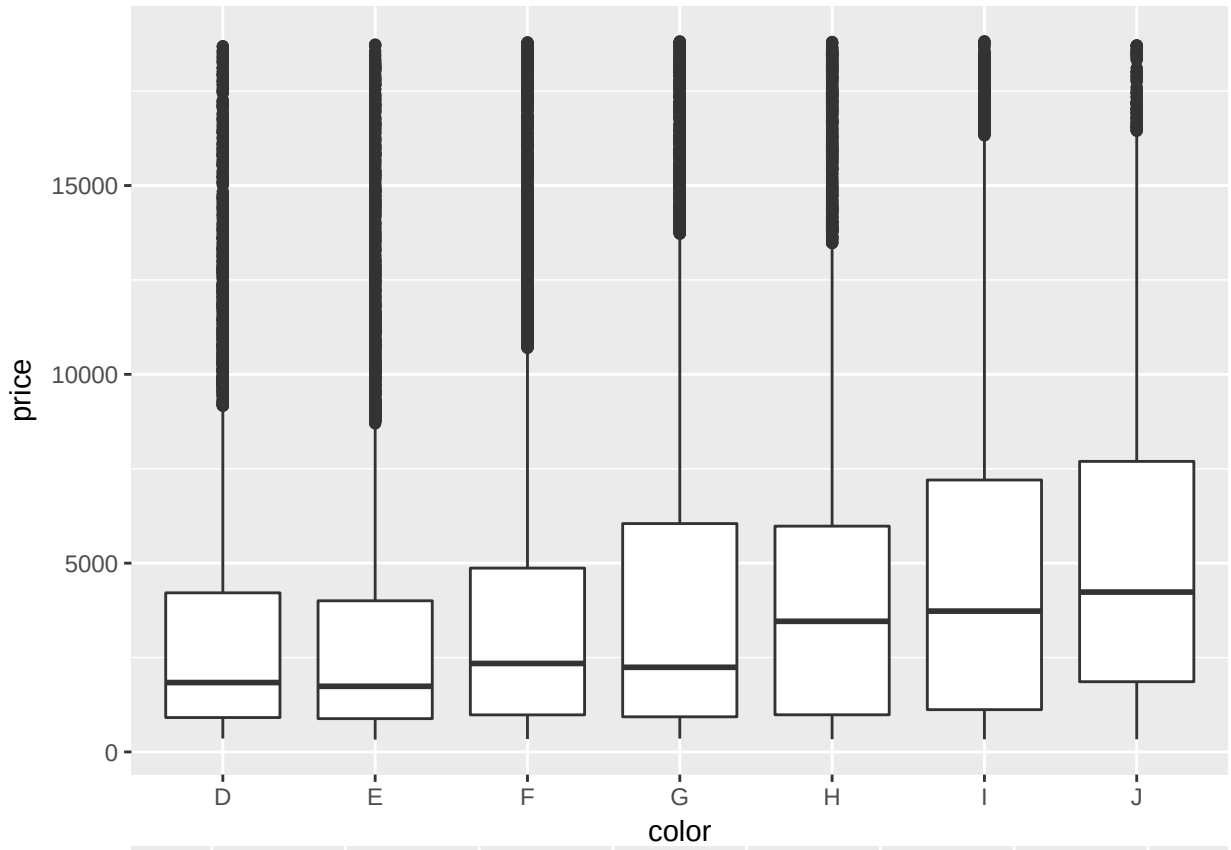


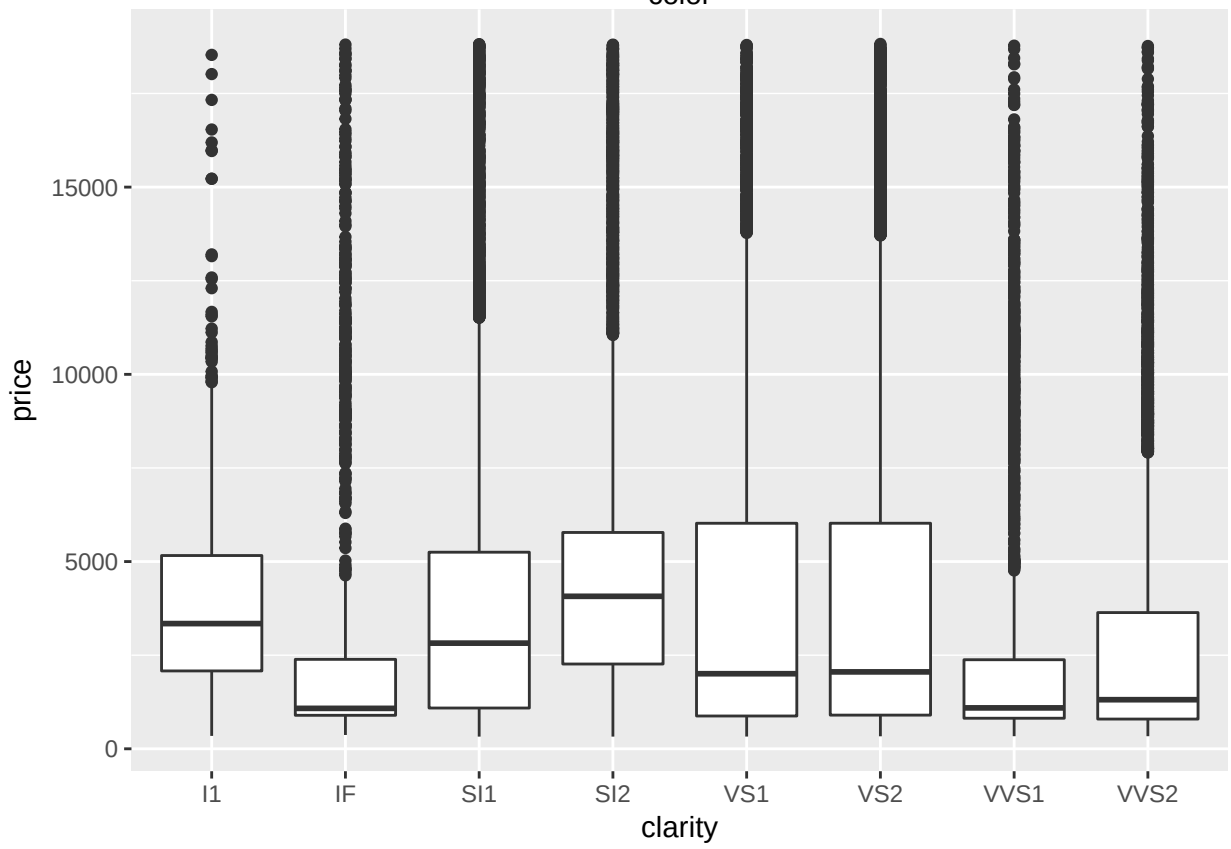
Use `ggplot` to look at the bivariate distributions of the response versus *all* predictors. Make sure you handle categorical predictors differently from continuous predictors. This time employ a for loop when an logic that handles the predictor type.

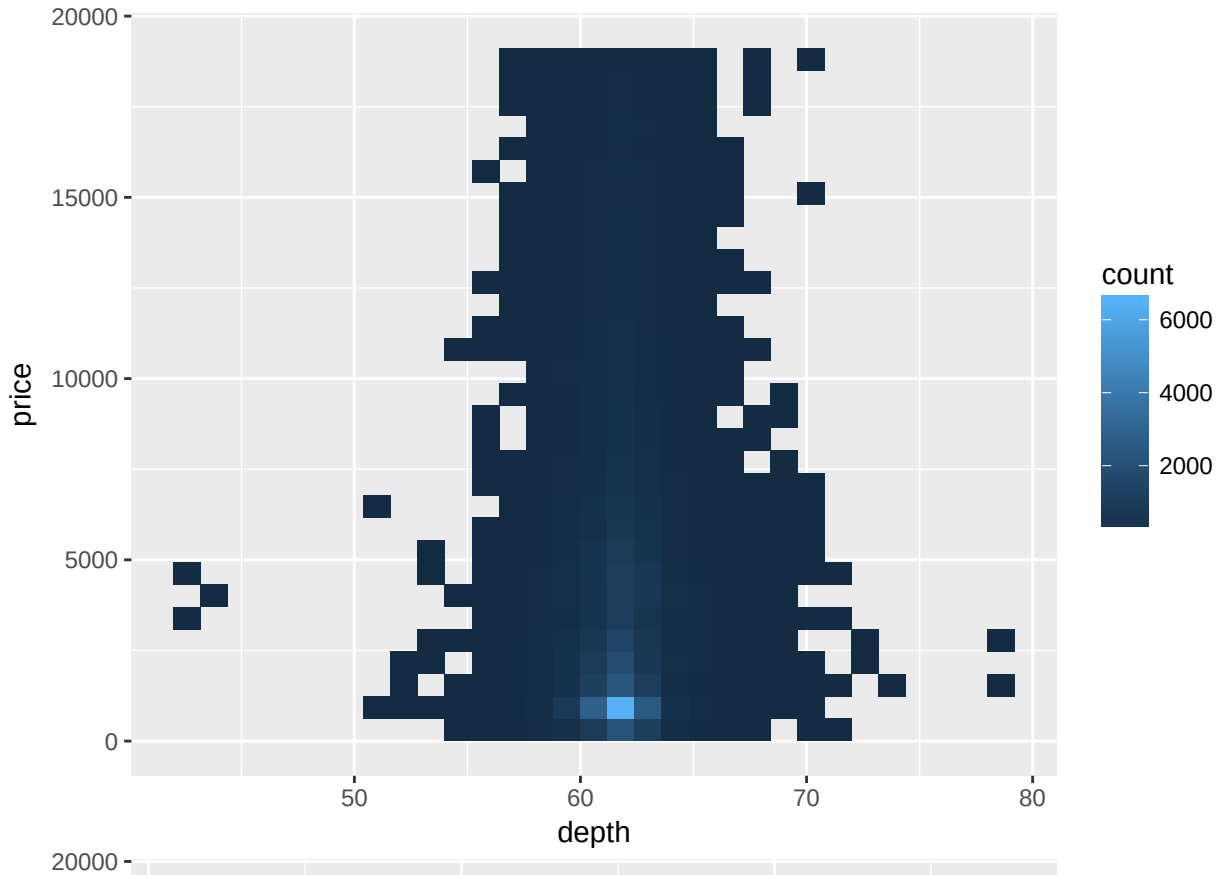
```
plot=ggplot(data=diamonds)
for(i in seq(1,10)){
  if((names(diamonds)[i])=="price"){next}
    if(is.numeric(diamonds[[i]])){g=(plot+geom_bin2d(mapping=aes(x=diamonds[[i]],y=price))+xlab((names(
    print(g)}
    if(is.factor(diamonds[[i]])){g=(plot+geom_boxplot(mapping=aes(x=diamonds[[i]],y=price))+xlab((names
    print(g)}
  }
```

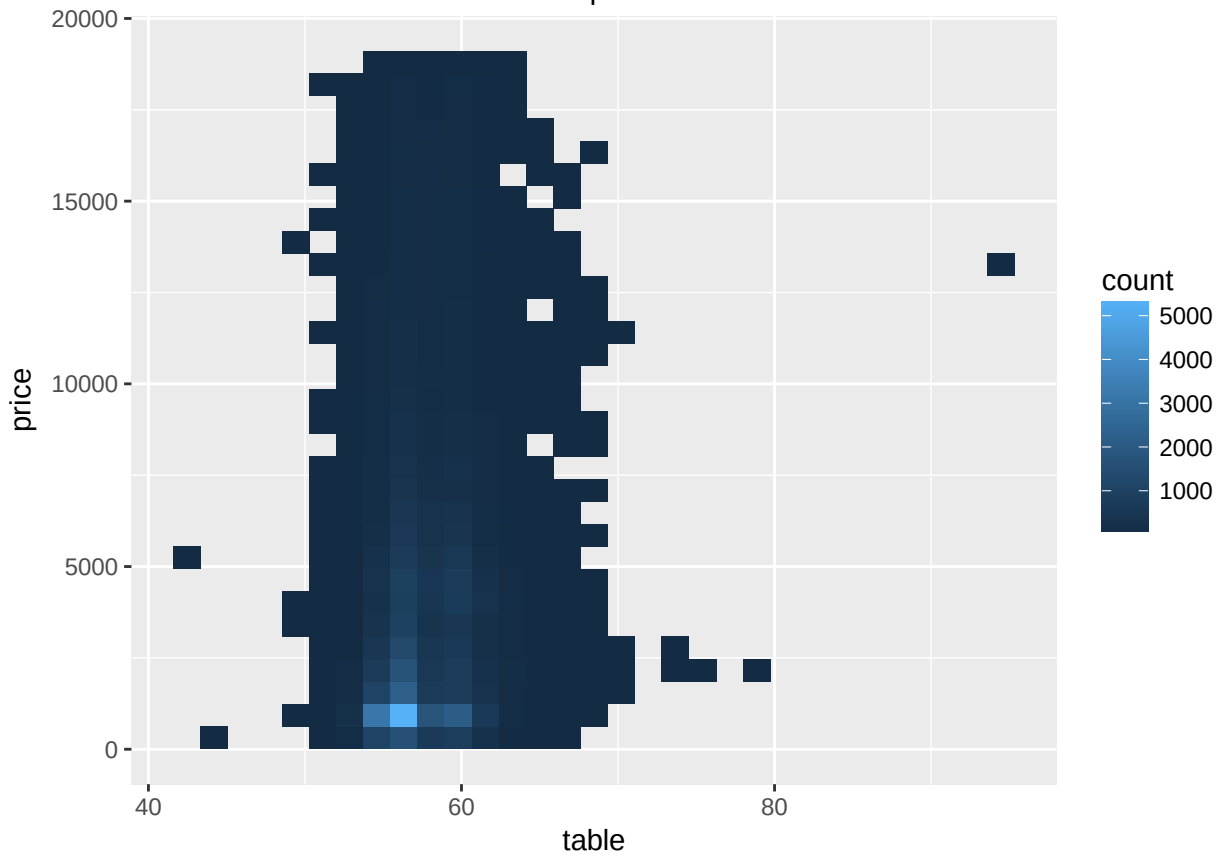

20000
15000
10000
5000
0
price
0
1
2
3
4
5
carat
count
6000
4000
2000

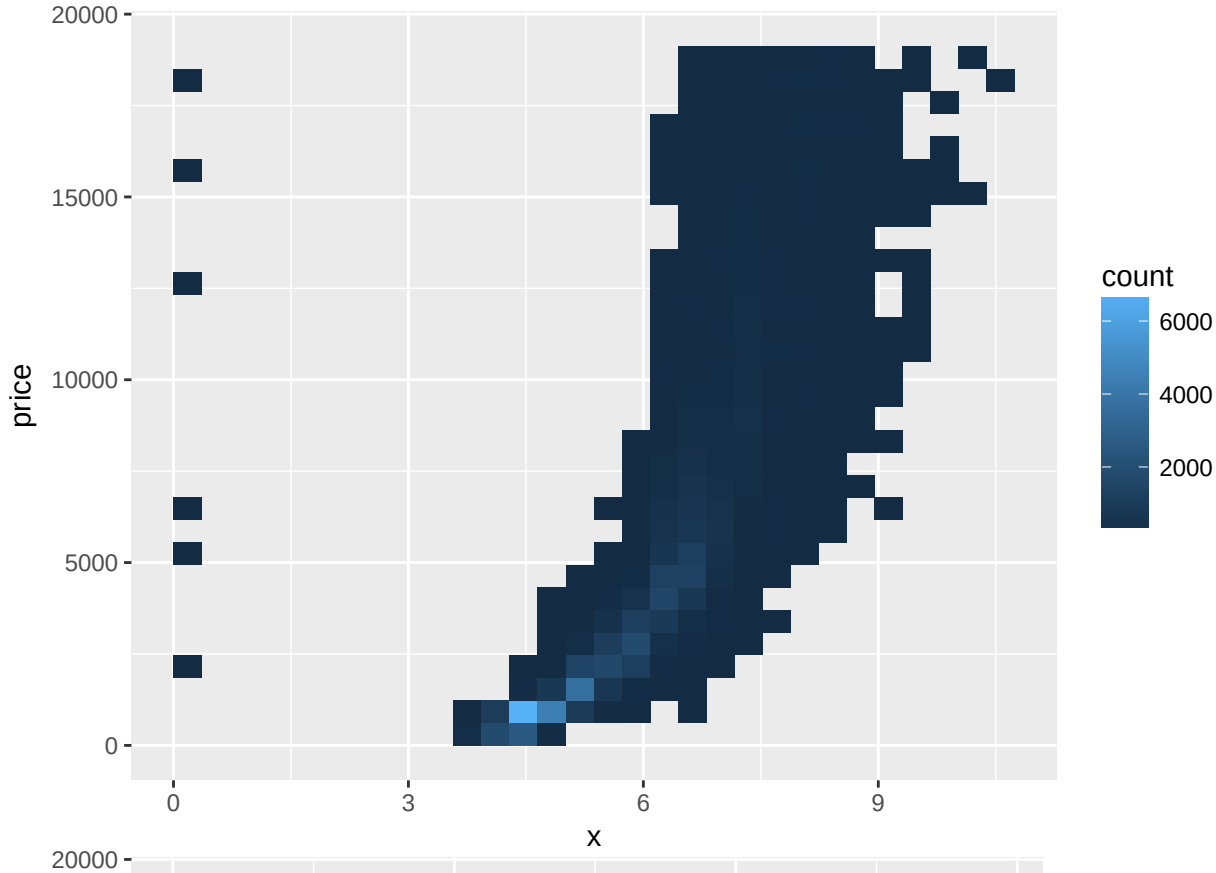

15000
10000
5000
0
price
Fair
Good
Ideal
Premium
Very Good
cut

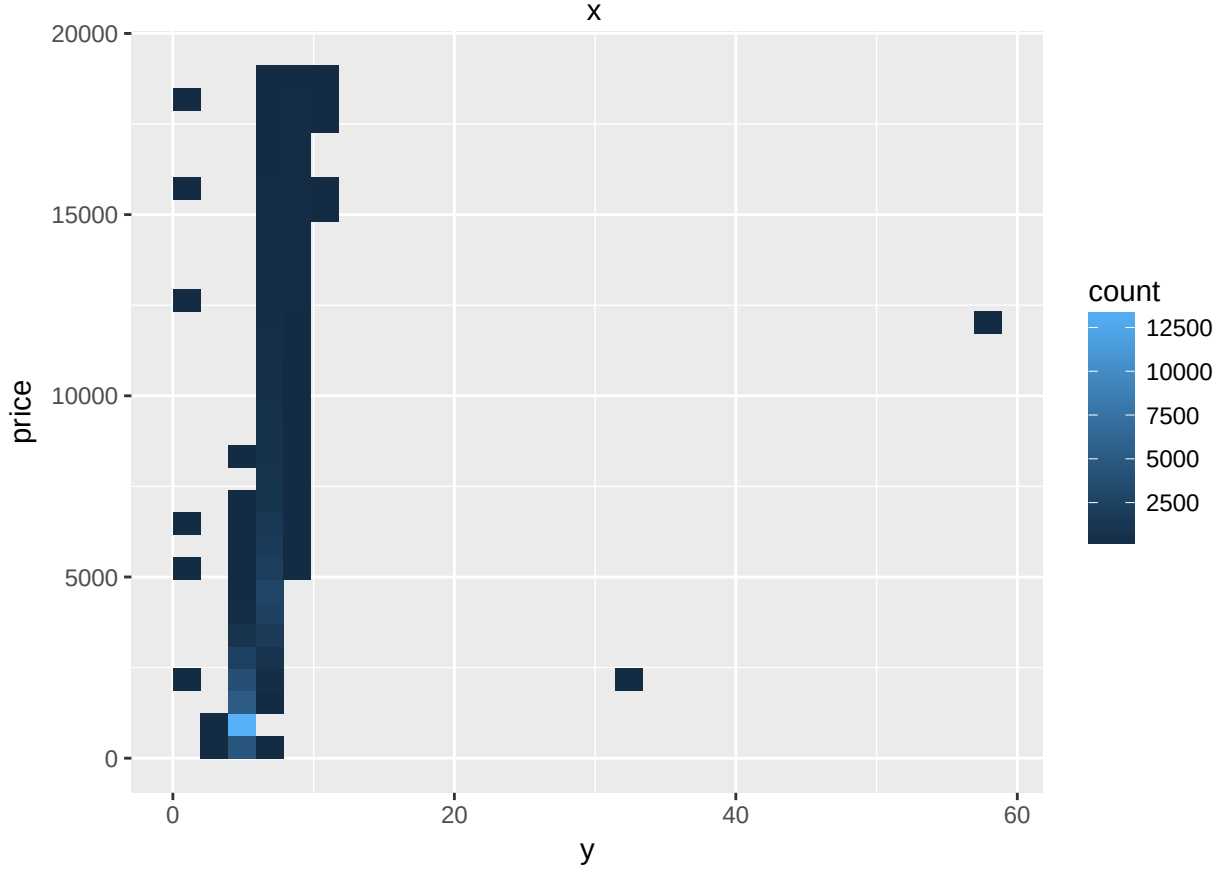

price
15000
10000
5000
0
D
E
F
G
H
I
J
color

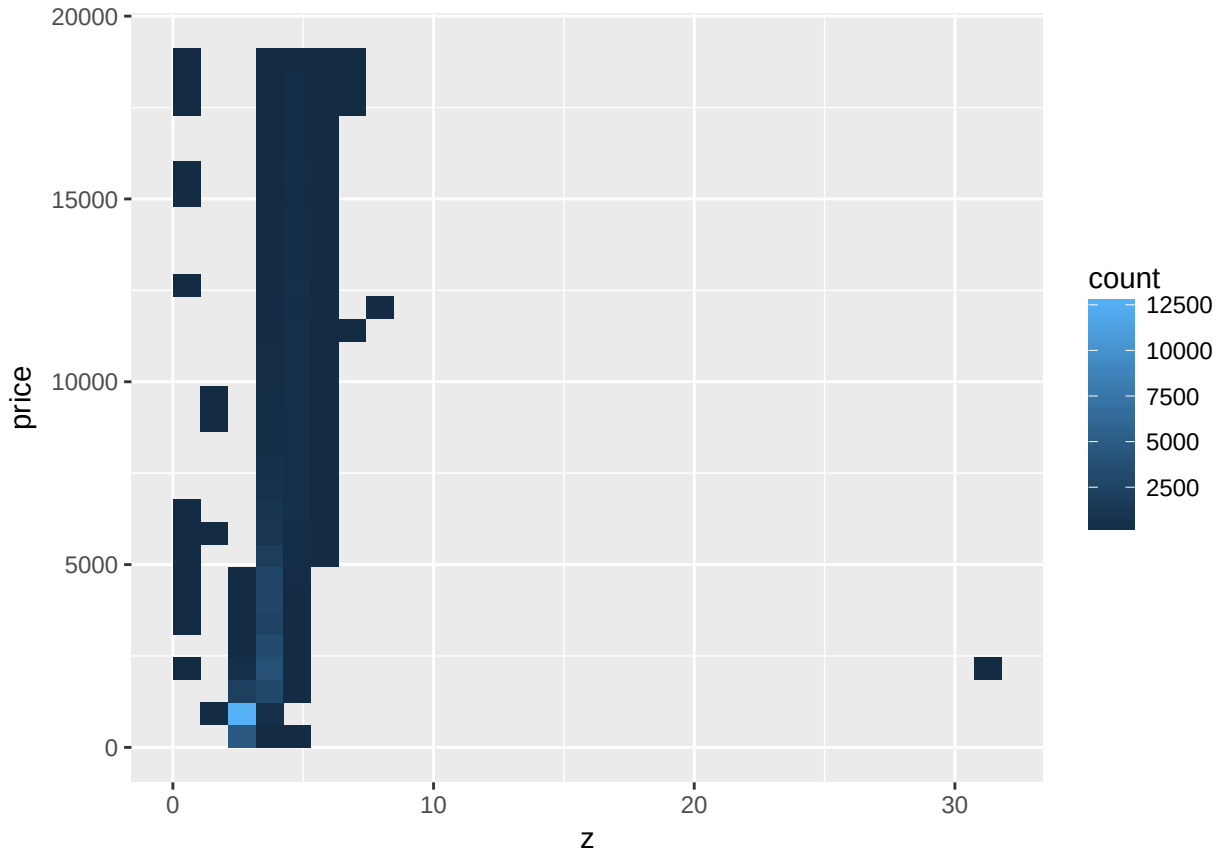

price
15000
10000
5000
0
I1
IF
SI1
SI2
VS1
VS2
VVS1
VVS2
clarity


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



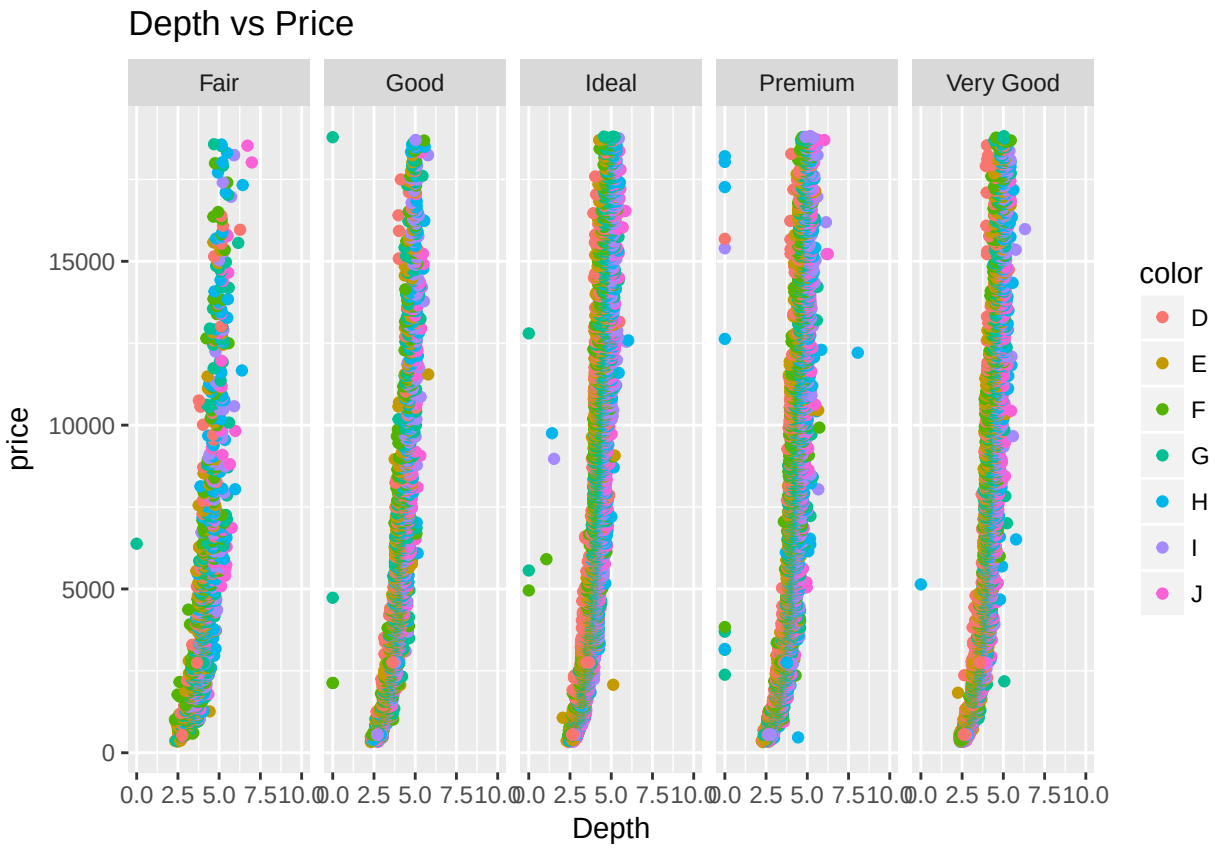
Does depth appear to be mostly independent of price?

Yes

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

```
ggplot(data=diamonds)+geom_point(mapping=aes(x=z,y=price,col=color))+facet_grid(. ~cut)+ggtitle("Depth
```

```
## 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?

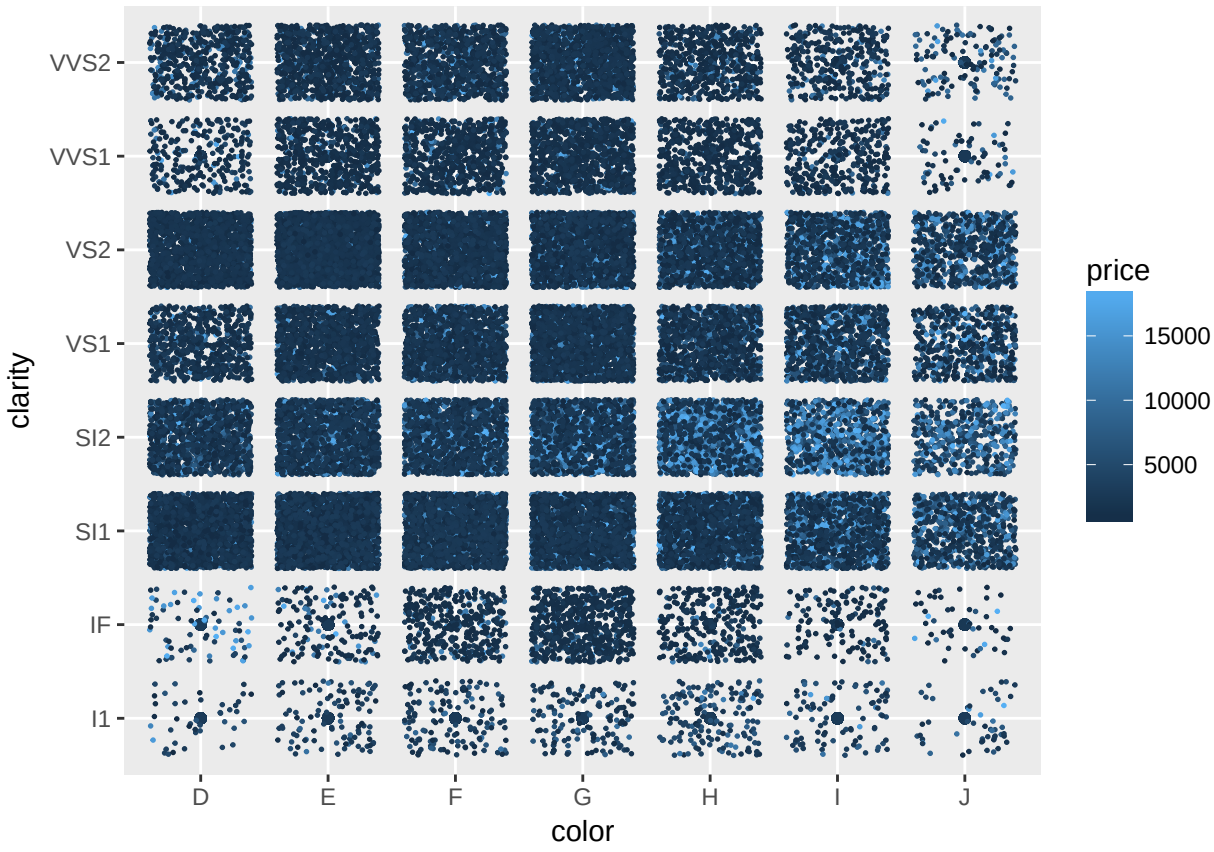
no

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(x=color,y=clarity,col=price))+geom_point()+geom_jitter(size=.3)
```



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.

```
dp=lm(diamonds$price~diamonds$depth)
```

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

```
b=summary(dp)$coef
b
```

```
##                  Estimate Std. Error   t value     Pr(>|t|)
## (Intercept)    5763.66772   740.5563  7.782889 7.214180e-15
## diamonds$depth  -29.64997    11.9897 -2.472953 1.340325e-02
```

```
r=summary(dp)$r.squared
r
```

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

```
rmse=summary(dp)$sigma
rmse
```

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

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

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

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

yes

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

```
cp=lm(diamonds$price~diamonds$carat)
```

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

```
b=summary(cp)$coef
b
```

```
##                 Estimate Std. Error   t value Pr(>|t|)
## (Intercept)    -2256.361   13.05535 -172.8304        0
## diamonds$carat  7756.426   14.06658  551.4081        0
```

```
r=summary(cp)$r.squared
r
```

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

```
rmse=summary(cp)$sigma
rmse
```

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

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

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

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

yes

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

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

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

```
b=summary(anova_lm)$coef
b
```

```
##        Estimate Std. Error   t value Pr(>|t|)
## colorD 3169.954   47.70694  66.44639        0
## colorE 3076.752   39.67251  77.55377        0
## colorF 3724.886   40.19912  92.66090        0
## colorG 3999.136   36.95309 108.22195        0
## colorH 4486.669   43.09159 104.11936        0
## colorI 5091.875   53.32814  95.48195        0
## colorJ 5323.818   74.10332  71.84318        0
```

```
r=summary(anova_lm)$r.squared
r
```

```
## [1] 0.5087167
```

```
rmse=summary(anova_lm)$sigma
rmse
```

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

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

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

yes

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

multiple factors aka each different color has its own estimate

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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

nothing since color is catagorical

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

```
all=lm(price~.,diamonds)
all
```

```
##
## 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.

```
b=all$coef
b
```

```
##  (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
```

```
r=summary(all)$r.squared
r
```

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

```
rmse=summary(all)$sigma
rmse
```

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

```
ci=2*rmse
ci
```

```
## [1] 2260.189
```

Interpret all entries in the vector b. one unit increase of carat will increase 11256 one unit increase of depth will decrease -63 one unit increase of table will decrease -26 compared to the fair cut, cutgood will be 579 more on average compared to the fair cut, cut ideal will be 832 more on average compared to the fair cut, cut premium will be 762 more on average compared to the fair cut, cut very good will be 726 more on average compared to the color D, E will be -209 less on average compared to the color D, F will be -272 less on average compared to the color D, G will be -482 less on average compared to the color D, H will be -980 less on average compared to the color D, I will be -1466 less on average compared to the color D, J will be -2369 less on average compared to the clarity I1, IF will be 5345 more on average compared to the clarity I1, SI1 will be 3665 more on average compared to the clarity I1, SI2 will be 2702 more on average compared to the clarity I1, VS1 will be 4578 more on average compared to the clarity I1, VS2 will be 4267 more on average compared to the clarity I1, VVS1 will be 5007 more on average compared to the clarity I1, VVS2 will be 4950 more on average

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

yes

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

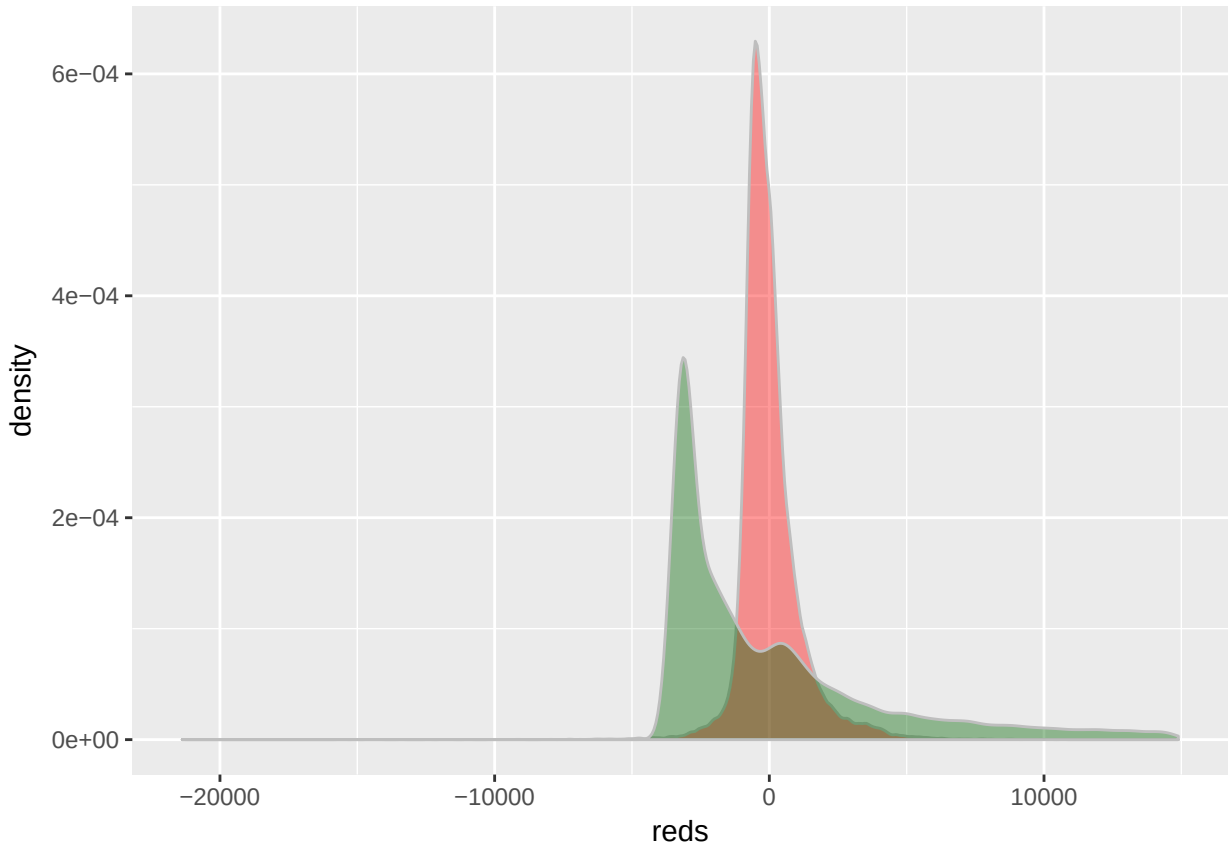
It is high since we are explaining using many features and sub features that can explain the price of a dimaond

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

no, since we have to many features and sub features that explain all the differences in price.

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

```
reds=all$residuals
null_r=(diamonds$price-mean(diamonds$price))
ggplot(data=diamonds,aes(reds))+geom_density(col = "grey", fill = "red", alpha = 0.4)+geom_density(mapp
```



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

no looks more exponential

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

```
poly1=lm(diamonds$price~.+I(diamonds$carat^2),data=diamonds)
```

What is b, R^2 and the RMSE?

```
poly1$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
```

```
r=summary(poly1)$r.squared
r
```

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

```
rmse=summary(poly1)$sigma
rmse
```

```
## [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.

one unit increase of carat will increase 16114 one unit increase of depth will decrease -116 one unit increase of table will decrease -36 compared to the fair cut, cutgood will be 538 more on average compared to the fair cut, cut ideal will be 807 more on average compared to the fair cut, cut premium will be 747 more on average compared to the fair cut, cut very good will be 678 more on average compared to the color D, E will be -209 less on average compared to the color D, F will be -284 less on average compared to the color D, G will be -496 less on average compared to the color D, H will be -997 less on average compared to the color D, I will be -1469 less on average compared to the color D, J will be -2357 less on average compared to the clarity I1, IF will be 5243 more on average compared to the clarity I1, SI1 will be 3565 more on average compared to the clarity I1, SI2 will be 2605 more on average compared to the clarity I1, VS1 will be 4475 more on average compared to the clarity I1, VS2 will be 4163 more on average compared to the clarity I1, VVS1 will be 5904 more on average compared to the clarity I1, VVS2 will be 4843 more on average on average $carat^2$ will be -1028 since after certain point our polynomial will start to give inaccurate responses

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

yes r^2 went up and rmse went down

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

```
X=model.matrix(poly1,diamonds)
b=coef(poly1)
```

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[23
 sum
}
```

cannot get function to work proply, dont know why?

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

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

```
## 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 not a numeric vector but made of discrete factors

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
reglm=lm(price~.,data=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?

```
RMSE
```

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

```
Rsq
```

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

```
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
```

Are they the same as in #4? yes

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)
#X1=Q%*%R
yhat_via_Q = Q%*%t(Q)%*%y1
head(yhat_via_Q)
```

```
##           [,1]
## [1,]  950.4536
## [2,] 2012.7517
## [3,] 4961.2134
## [4,] 1768.8883
## [5,] 8062.3744
## [6,] 7895.2217
```

```
bq=solve(R)%*%t(Q)%*%y1
bq
```

```
##                       [,1]
## (Intercept)  -12957.90102
## carat         10946.25569
## cutGood         656.05447
## cutIdeal        909.30148
## cutPremium      782.72600
## cutVery Good    811.53299
## colorE         -136.90027
## colorF         -217.88553
## colorG         -364.47948
## colorH         -975.13159
## colorI        -1470.42539
## colorJ        -2651.44761
## clarityIF      4517.51082
## claritySI1     2607.10726
## claritySI2     1576.96857
## clarityVS1     3470.59267
## clarityVS2     3221.08352
## clarityVVS1    4362.89897
## clarityVVS2    3857.10005
## depth           186.30387
## table           -34.59916
## x               172.05919
## y              1797.73377
## z             -4468.54923
```

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

```
XtX = t(X1) %*% X1
XtXinv = solve(XtX)
H = X1 %*% XtXinv %*% t(X1)
yhat = H %*% yhat_via_Q
head(yhat)
```

```
##            [,1]
## 17045  950.4536
## 46812 2012.7517
## 3469  4961.2134
## 45020 1768.8883
## 19894 8062.3744
## 20225 7895.2217
```

```
yhat = H %*% y1
e = y1 - yhat
e_q=y1-yhat_via_Q
```

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(e+yhat), y1) #(a)
expect_equal(sum(t(yhat_via_Q) %*% e_q),0,tol=1e-4) #(b)
expect_equal(sum(H %*% e_q),0) #(c)
expect_equal (as.vector(t(yhat_via_Q)%*%H),as.vector(t(yhat_via_Q)))#(d)
expect_equal(sum(t(yhat_via_Q)%*%e_q), 0 , tol = 1e-4) #(e)
y1bar=mean(y1)
SSR=sum((yhat_via_Q-y1bar)^2)
SSE=sum(e_q^2)
SST=sum((y1-y1bar)^2)
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.

```
#mod8=lm(price~.*.+I(carat^5)+I(x^5)+I(y^5)+I(z^5)*.+I(depth^5)+I(table^5),diamonds)
mod8p=lm(price~.*.+poly(carat,5,raw=TRUE)+poly(x,5,raw=TRUE)+poly(y,5,raw=TRUE)+poly(z,5,raw=TRUE)+poly
```

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

```
summary(mod8p)$r.squared
```

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

```
summary(mod8p)$sigma
```

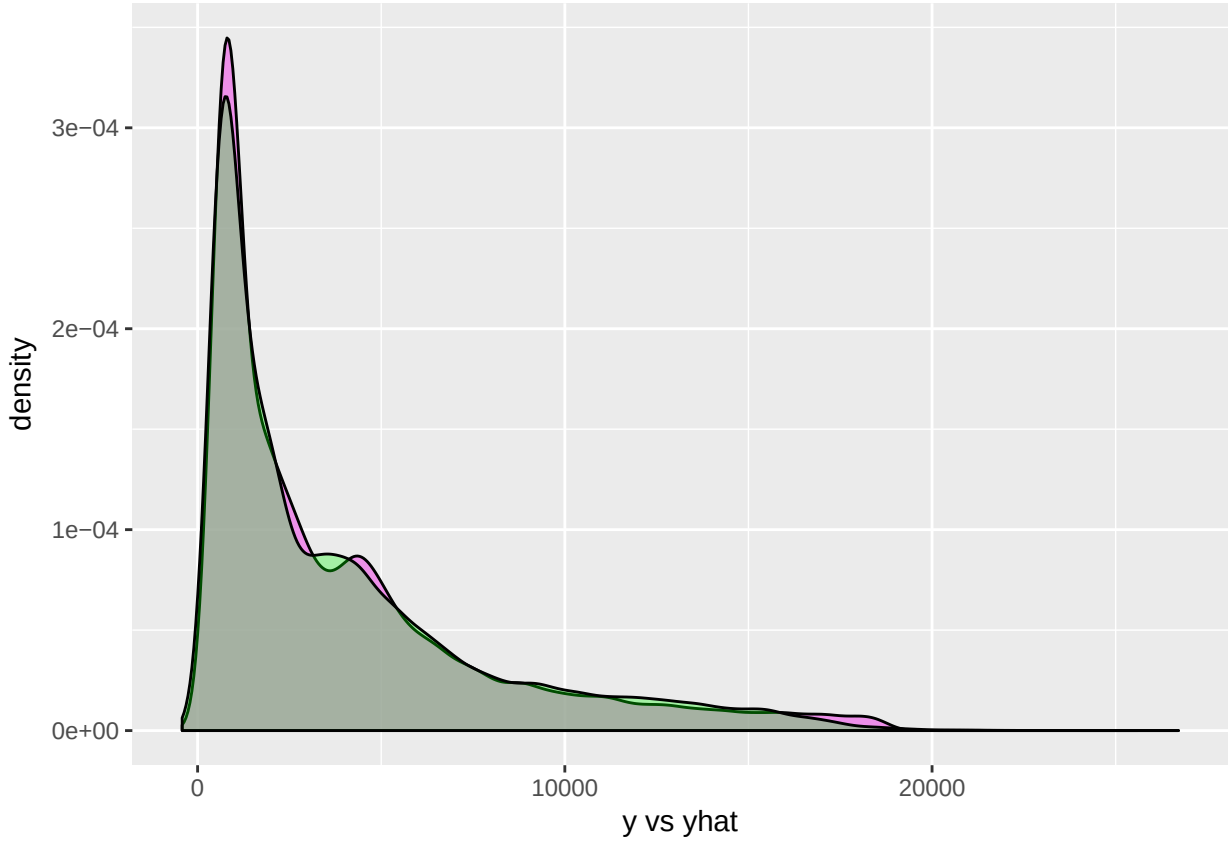
```
## [1] 659.2249
```

```
sd(mod8p$residuals)
```

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

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

```
y8=diamonds$price
yhat8=as.vector(y-resid(mod8p))
ggplot(data=diamonds,aes(y8))+geom_density(fill="orchid2",alpha=.8)+geom_density(aes(yhat8),fill="green
```



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.

no since the range is about 2500 per guess since the mean is about 3900

What is the degrees of freedom in this model?

```
#length(mod8$coefficients)
length(mod8p$coefficients)
```

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

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

no, we have a large h^* to include polynimals and interaction terms.

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

yes since they overlap all most ever where

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

scarcity,size, supply and demand,shinynes,Phosphorescent of the diamond, color of Phosphorescent.

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.

what color the person likes or shape, what reigon the diamond is from

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_in=sample(1:n,size=n*1/K)
train_in=sample(setdiff(1:n,test_in))



#make sure we did this right:
pacman::p_load(testthat)
expect_equal(1 : n, sort(c(test_in, train_in)))

diamonds_train = diamonds[train_in, ]
diamonds_test = diamonds[test_in, ]

mod8p10=lm(price~.*.+poly(carat,5,raw=TRUE)+poly(x,5,raw=TRUE)+poly(y,5,raw=TRUE)+poly(z,5,raw=TRUE)+po
y_train=predict(mod8p10,diamonds_test)
```

```
## Warning in predict.lm(mod8p10, diamonds_test): prediction from a rank-
## deficient fit may be misleading
```

```
y_test=diamonds_test$price
s_e_s=sd(y_train-y_test)
s_e_s
```

```
## [1] 1229.627
```

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

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:

```
tot=rep(NA,100)

for(i in 1:100){
  X=as.data.frame(matrix(c(x,rnorm(n*i)),ncol=i+1))
  K=10
  testind=sample(1:n,size=n*1/K)
  trainind=sample(setdiff(1:n,testind))
  train=X[trainind,]
  test=X[testind,]
  yhtrain=y[trainind]
  yhtest=y[testind]
  mod=lm(yhtrain~.,train)
  yhoss=predict(mod,test)
  osse=sd(yhtest-yhoss)
  tot[i]=osse
}
```

```
## 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
```

```
tot
```

```
##   [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
```

it looks like around 75 predictors we start to overfitt