HW03p

[Your Name Goes Here]

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)
```

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's a dataset of a sample of about 54,000 diamonds and their various features.

```
?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 ...
```

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

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

$n = 53940$, $p = 10$.

Price: cost of diamond in USD, ($326-$18,823) Carat: unit of weight for diamonds, continous, range:(0.2-5.01). Cut: shape the diamond is in, ordinal factor (fair-ideal) Color: tells about the purity of diamonds, yellow(J) means low quality, purest(D) means best quality Clarity: how clear the diamond is, clarity implies wuality, x: length of diamond in millimeters, (0-10.74) y: width in millimeters, (0-58.9) z: depth in millimeters, (0-.31.8) Depth: total depth percentage, (43-79) Table: width of top of diamond relative to widest point, (43-95)

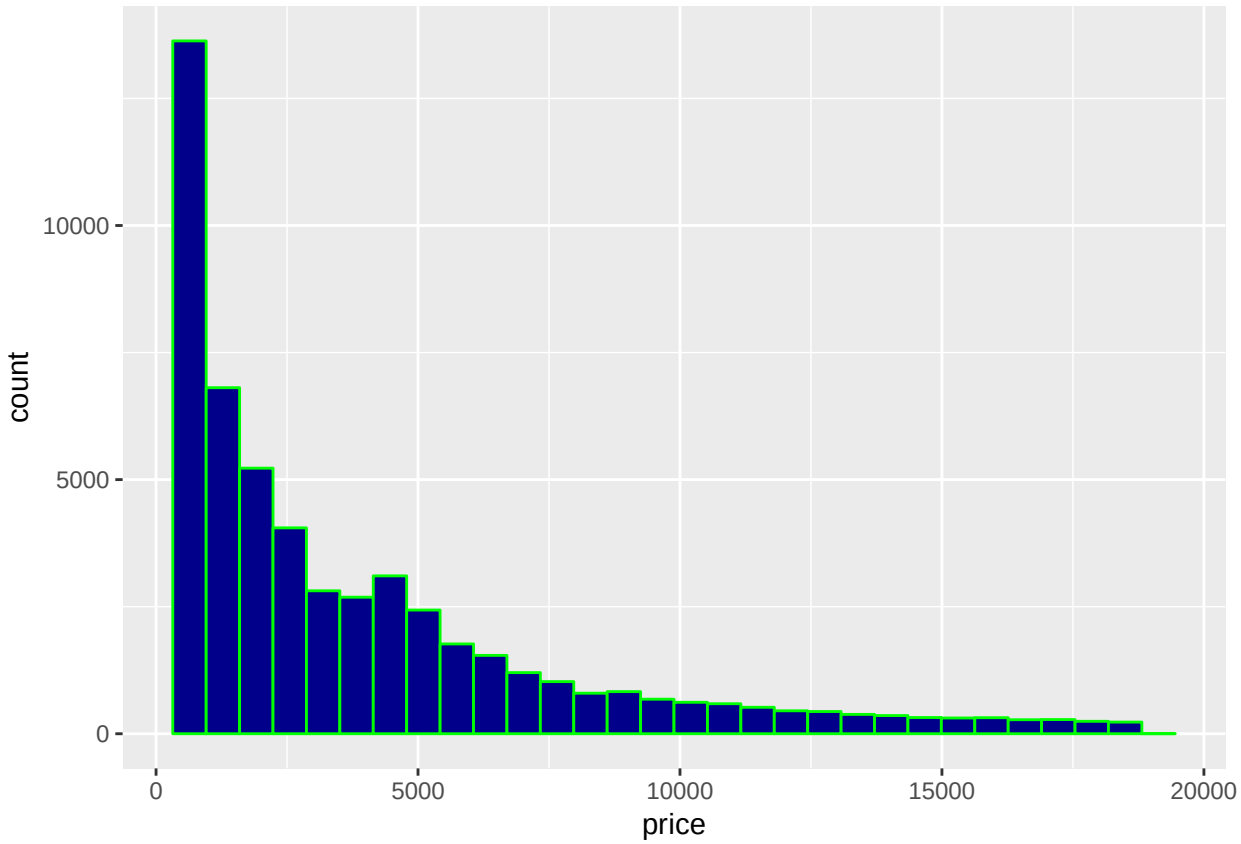
The most likely respnse metric is price because the dataset is labeled "Price of . . . "

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.

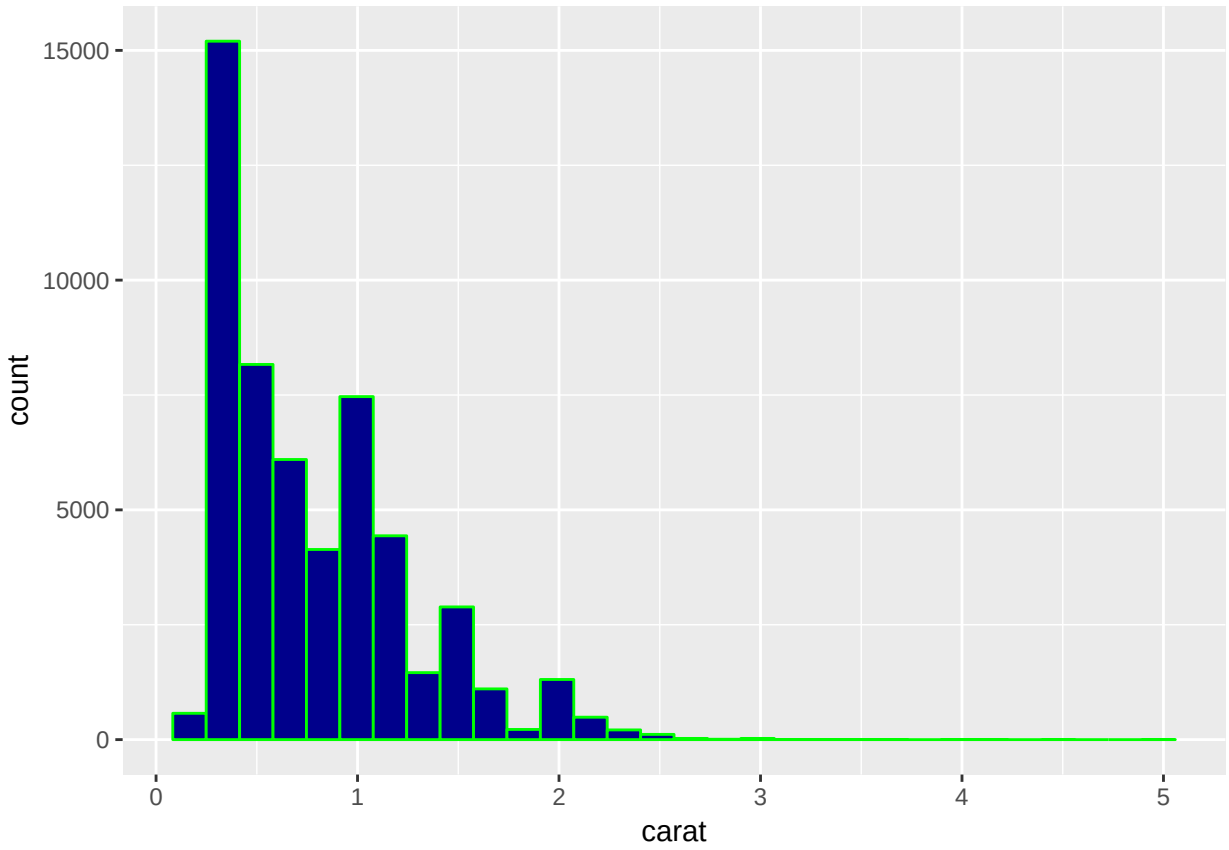
```
diamonds = as.data.frame(diamonds)
base = ggplot(diamonds)
base + aes(price) + geom_histogram(col="green", fill = "darkblue")
```

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

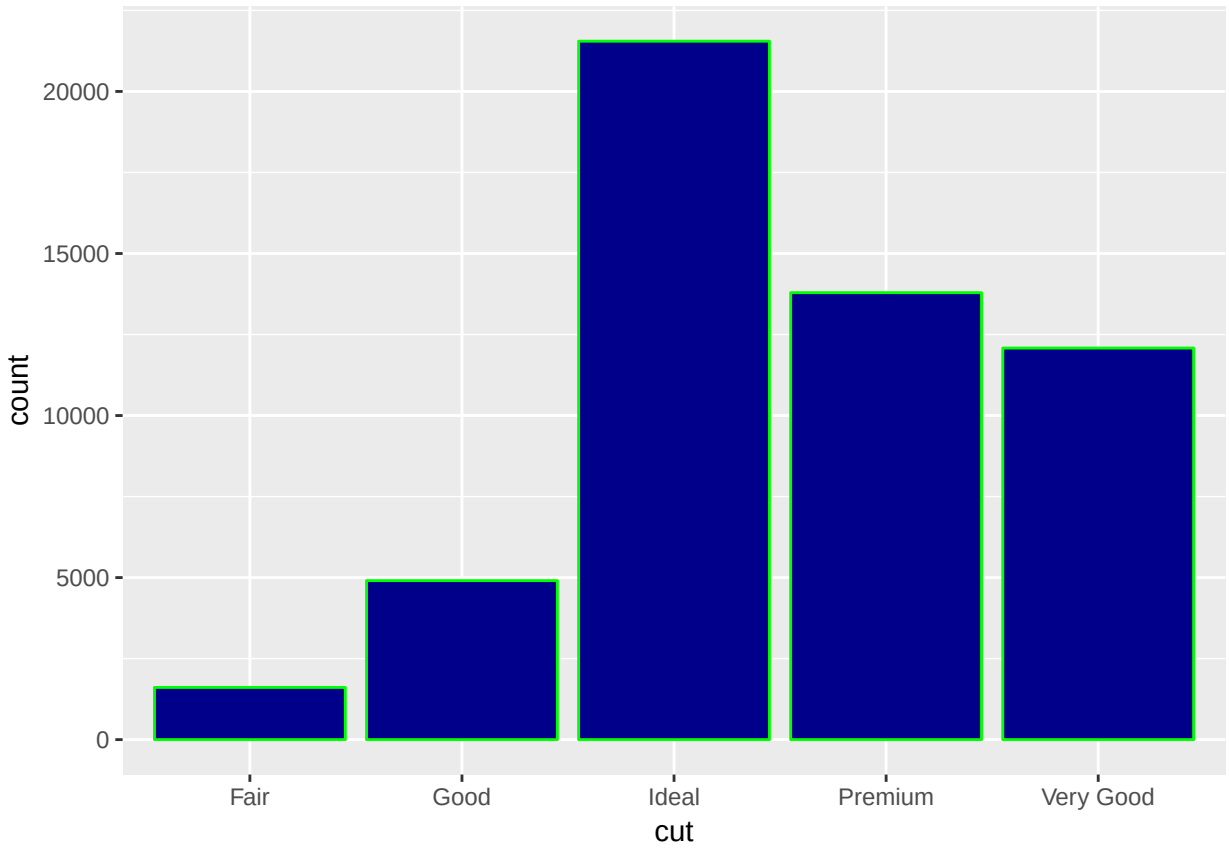



```
base + aes(carat) + geom_histogram(col="green", fill = "darkblue")
```

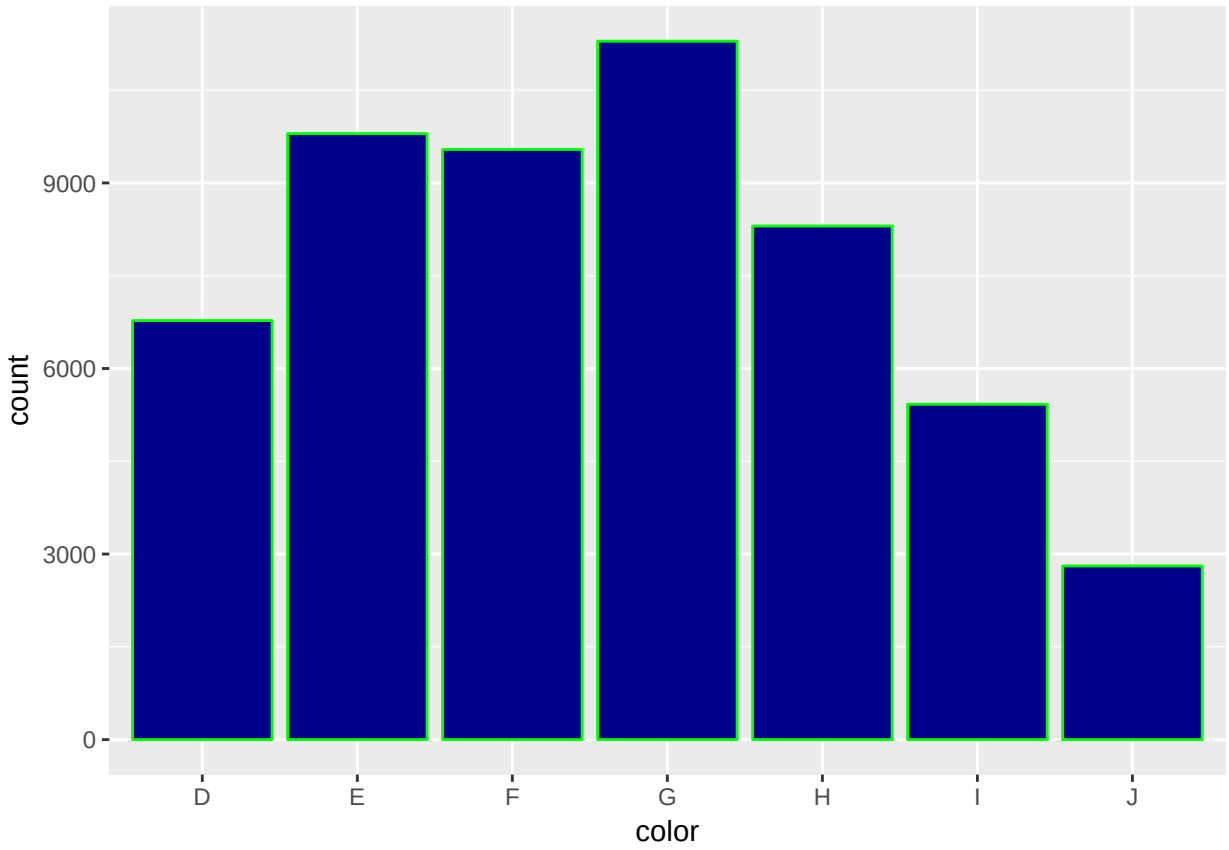
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



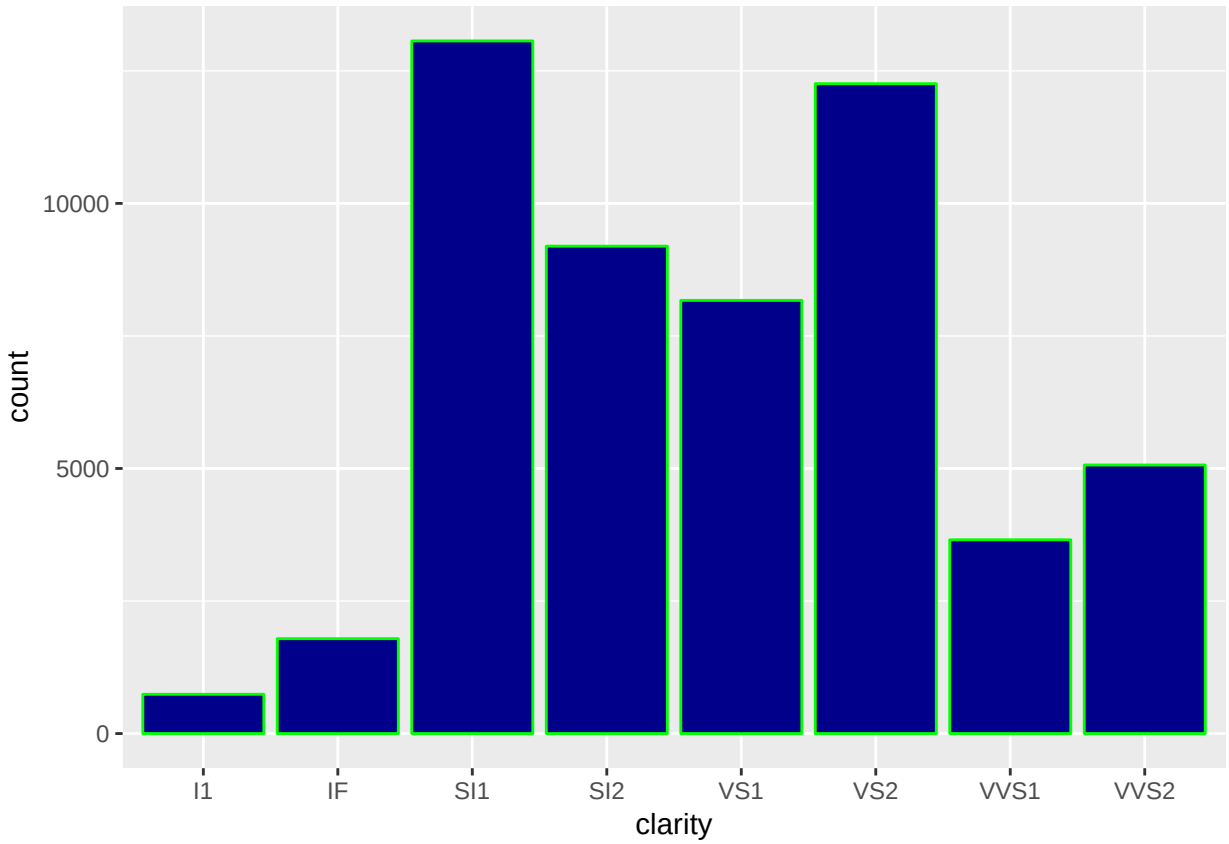

```
base + aes(cut) + geom_bar(col="green", fill = "darkblue")
```




```
base + aes(color) + geom_bar(col="green", fill = "darkblue")
```

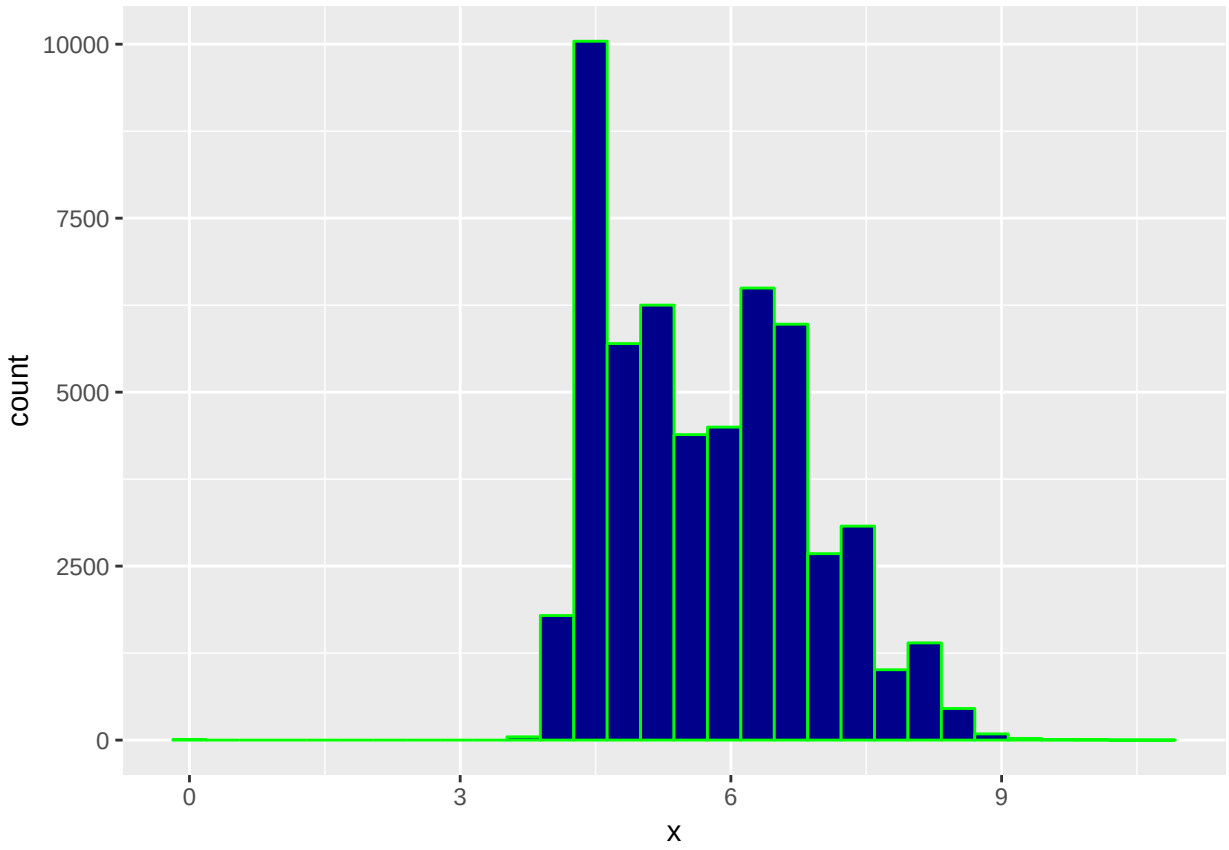



```
base + aes(clarity) + geom_bar(col="green", fill = "darkblue")
```



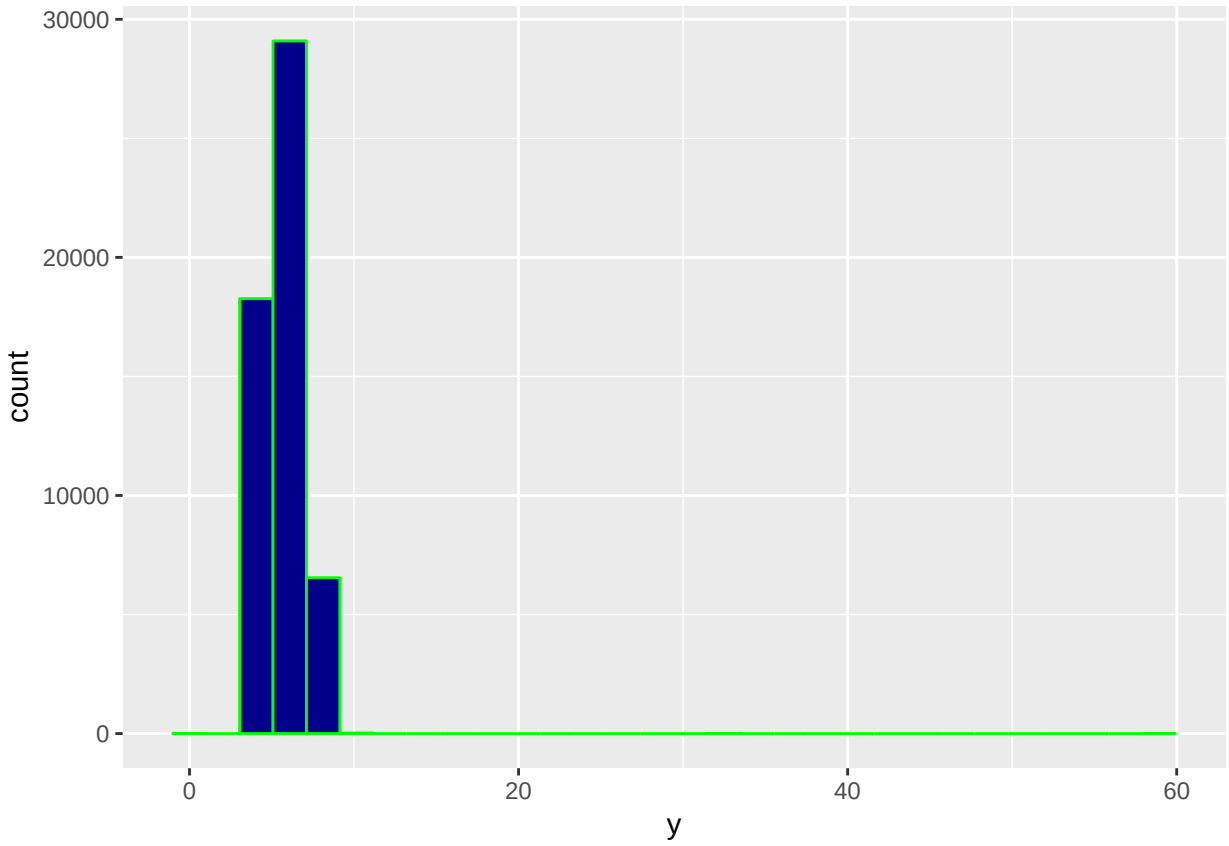

```
base + aes(x) + geom_histogram(col="green", fill = "darkblue")
```

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



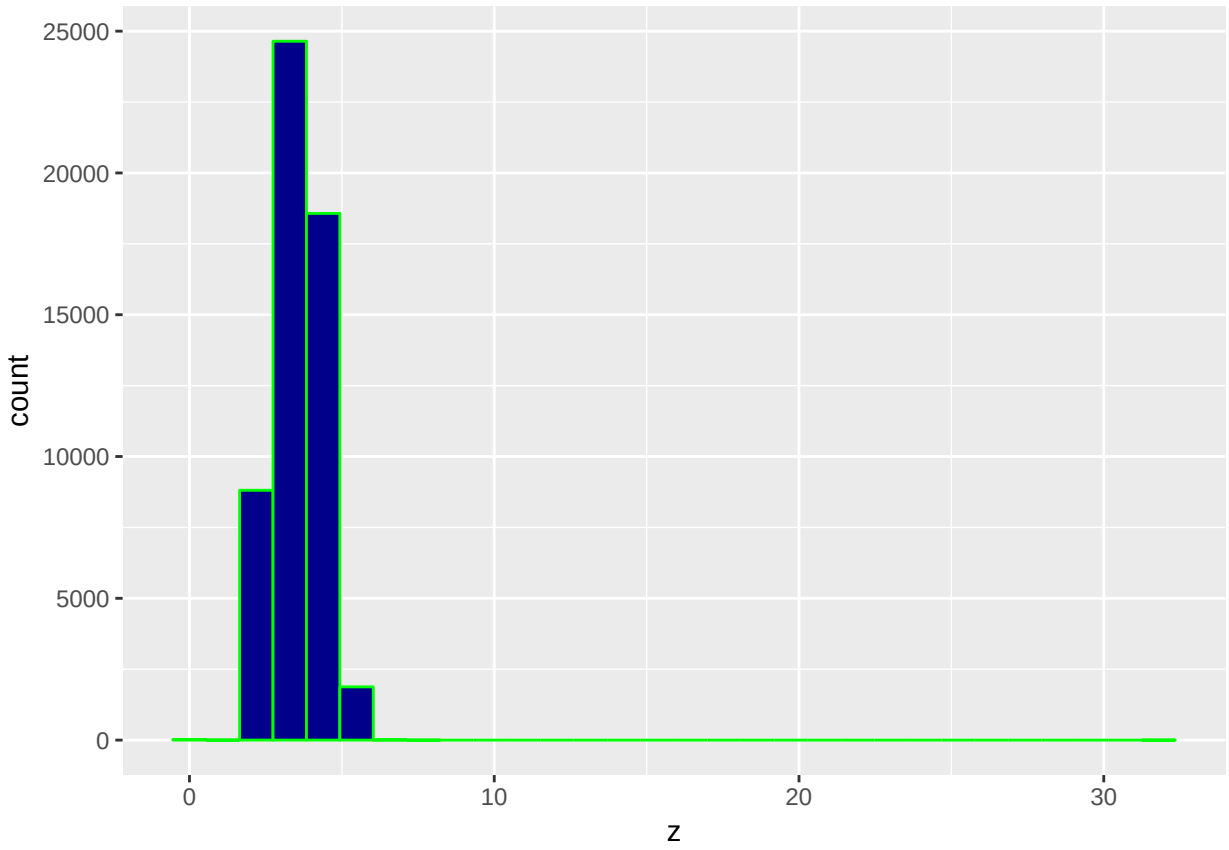

```
base + aes(y) + geom_histogram(col="green", fill = "darkblue")
```

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



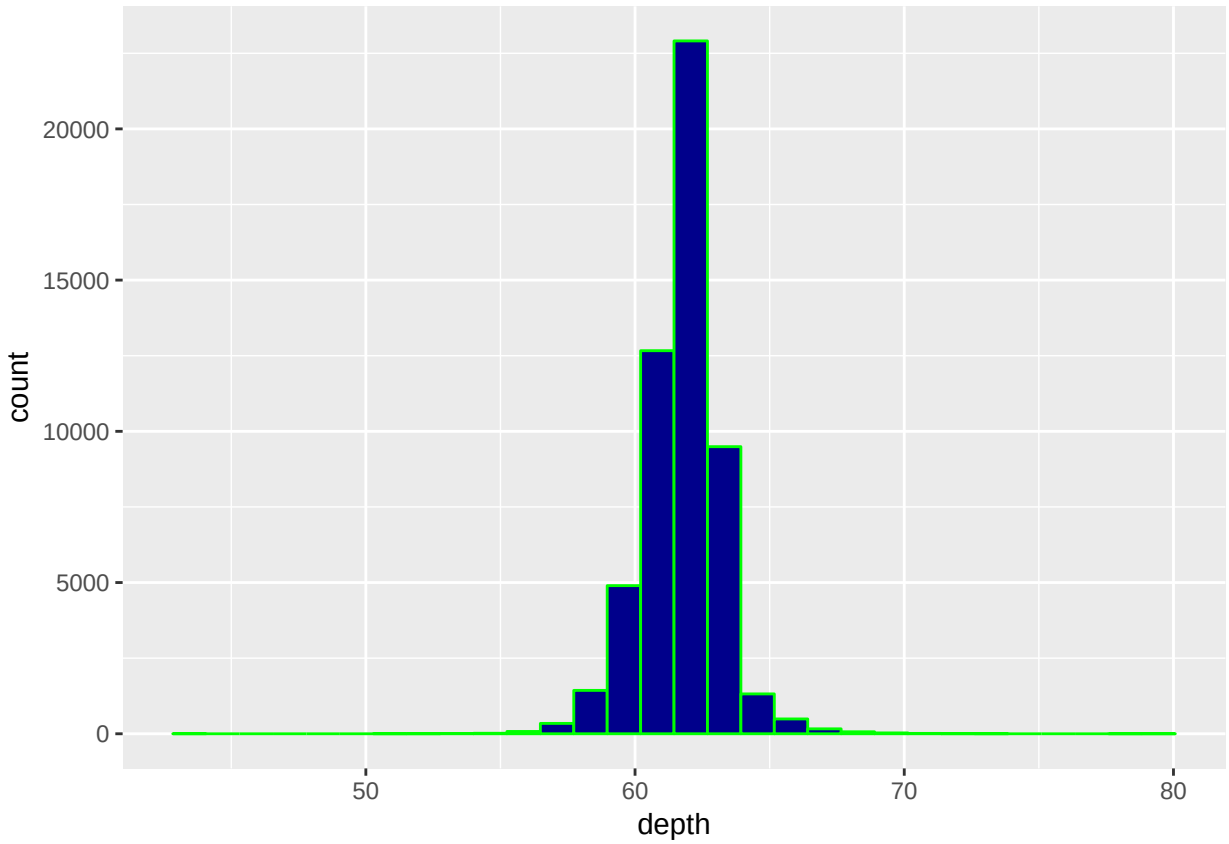

```
base + aes(z) + geom_histogram(col="green", fill = "darkblue")
```

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



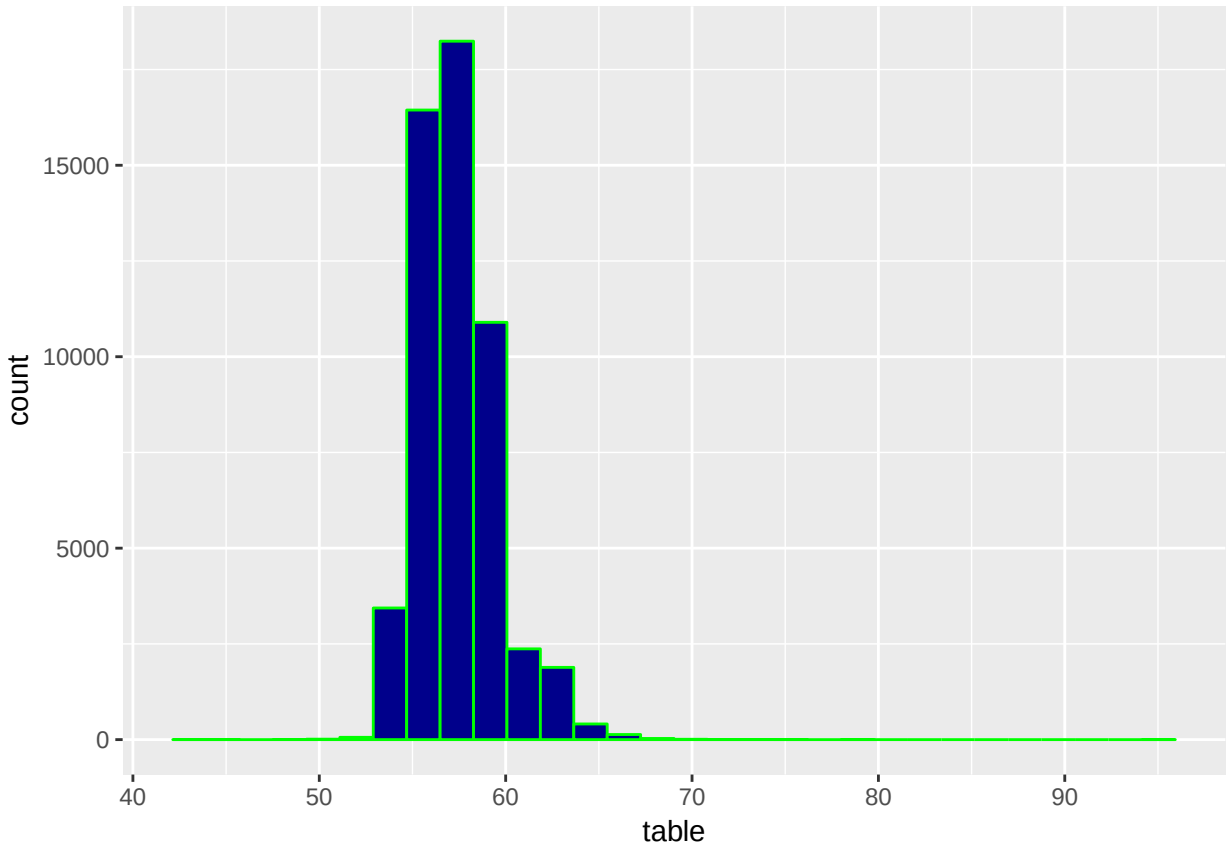

```
base + aes(depth) + geom_histogram(col="green", fill = "darkblue")
```

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



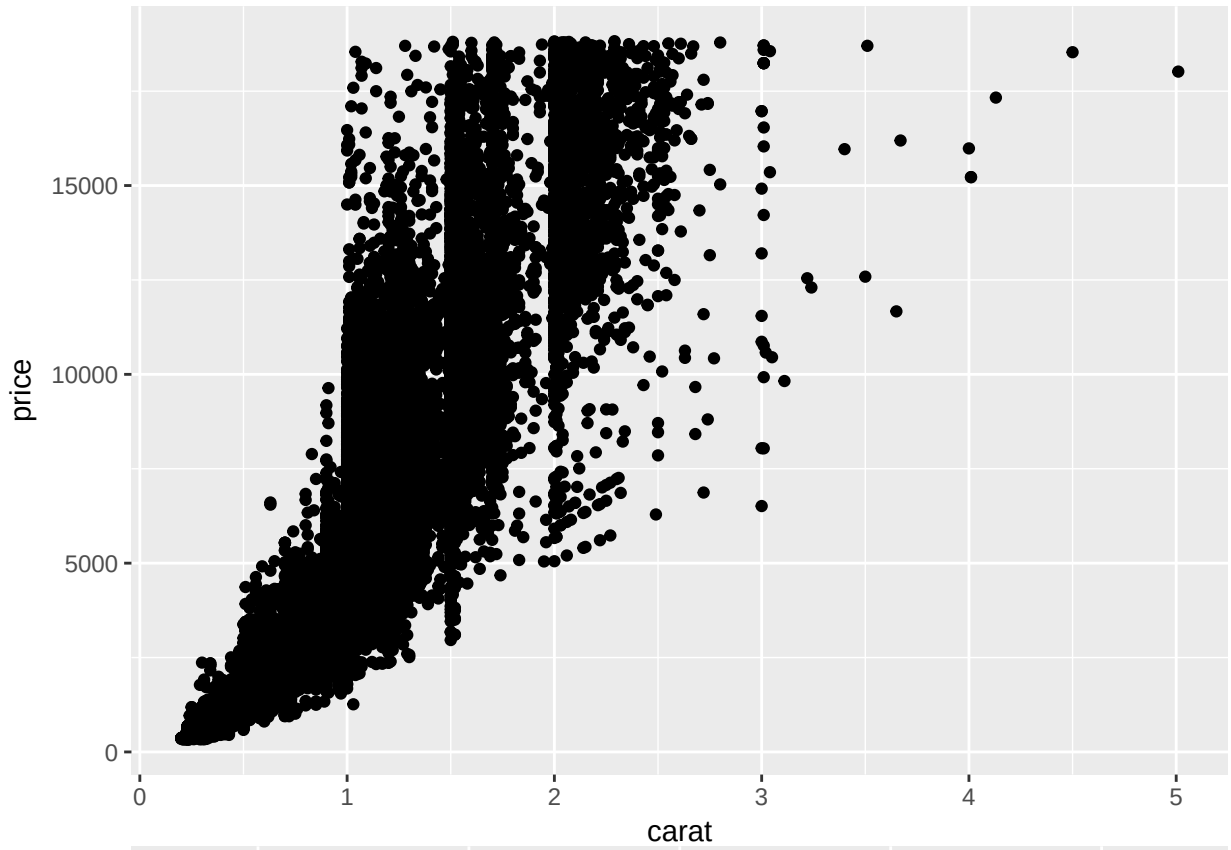

```
base + aes(table) + geom_histogram(col="green", fill = "darkblue")
```

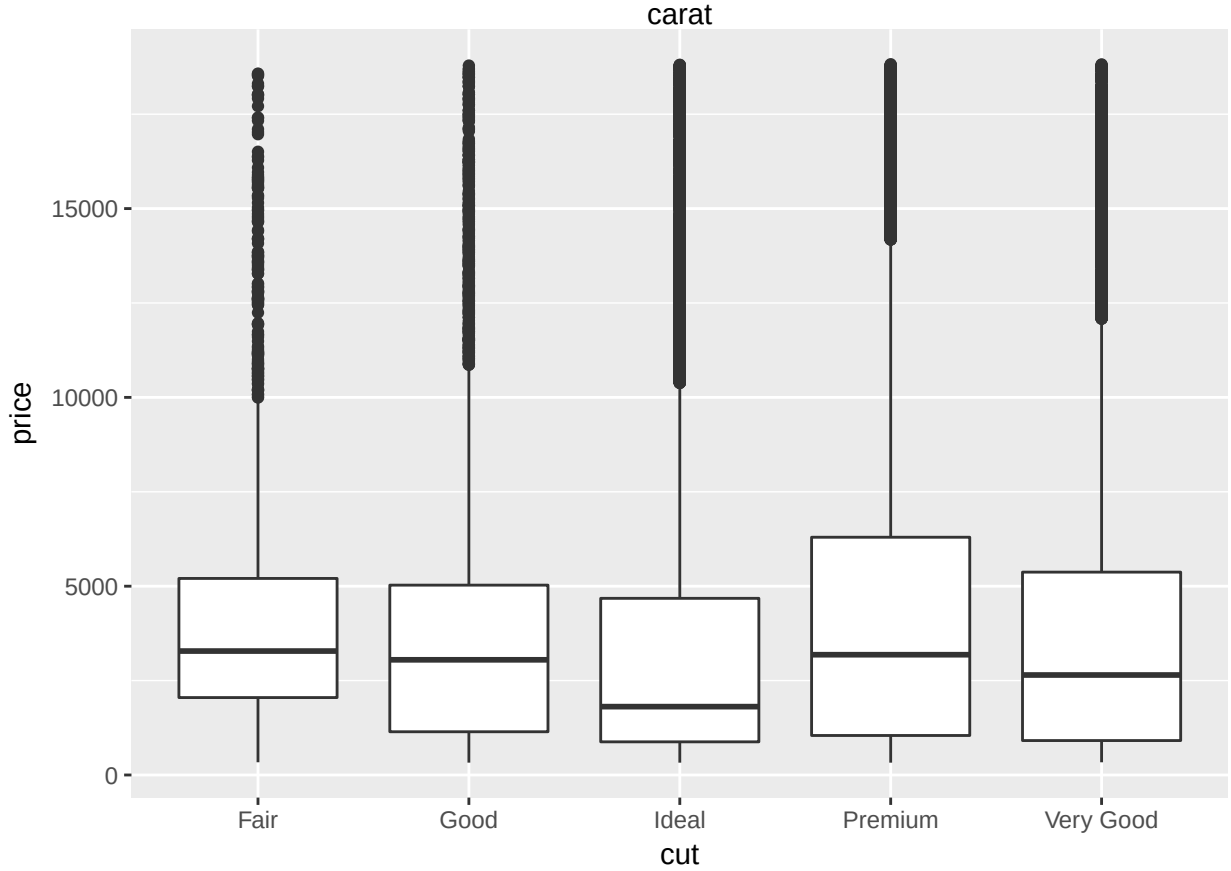
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

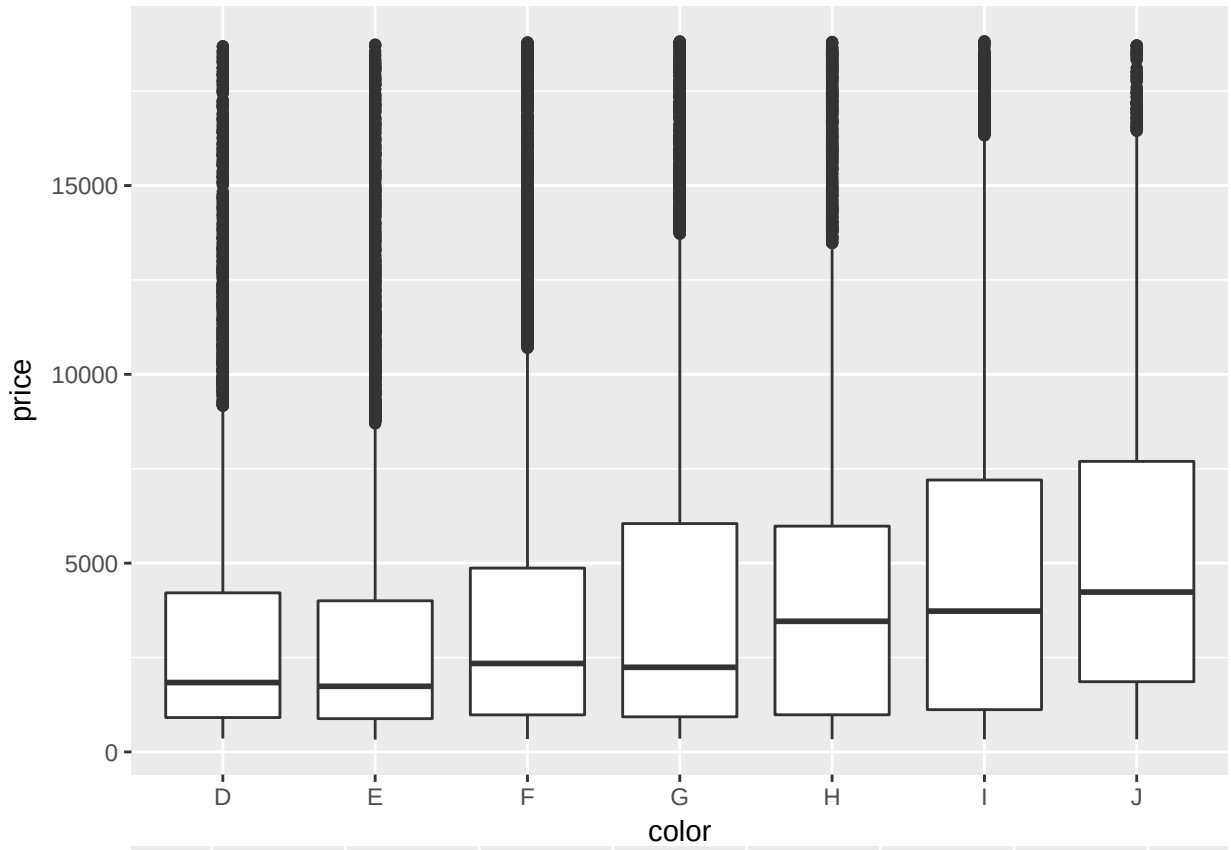


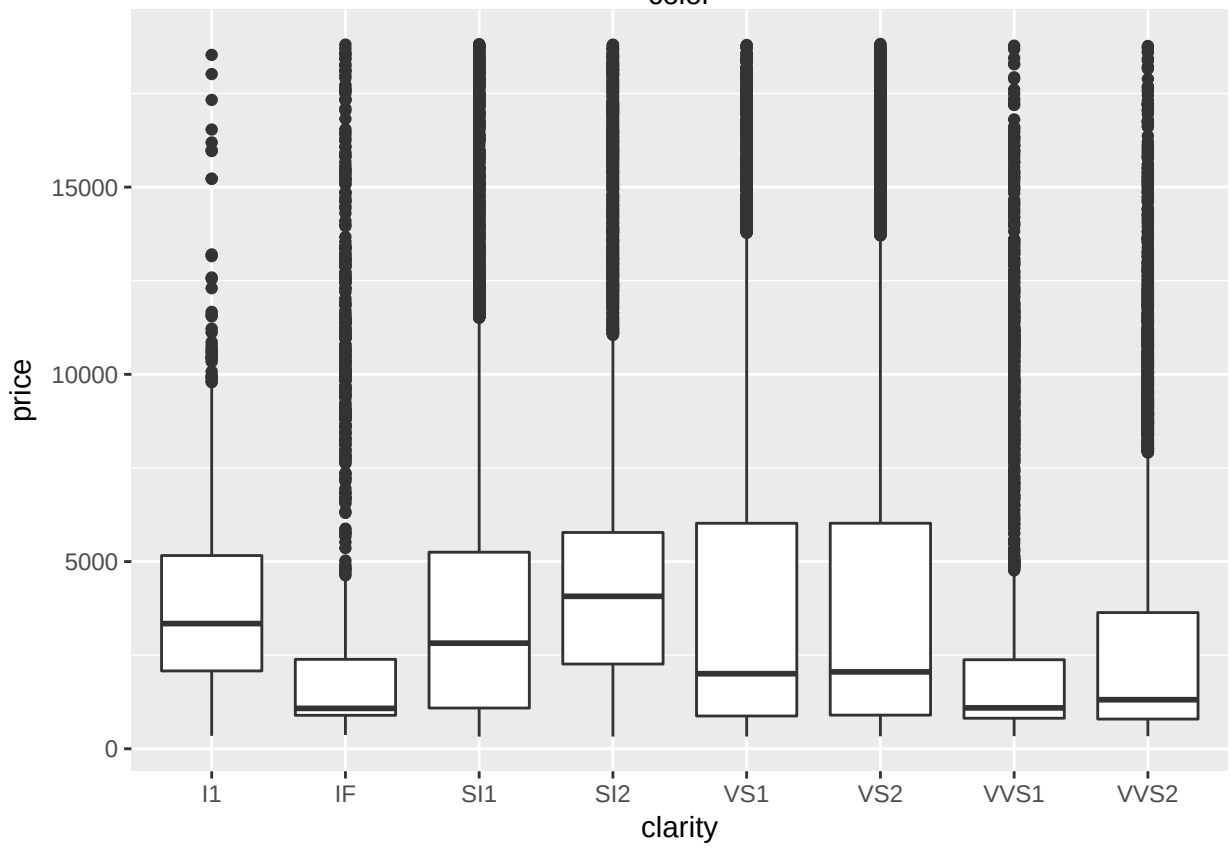
Use `ggplot` to look at the bivariate distributions of the response versus *all* predictors. Make sure you handle categorical predictors differently from continuous predictors. This time employ a for loop when an logic that handles the predictor type.

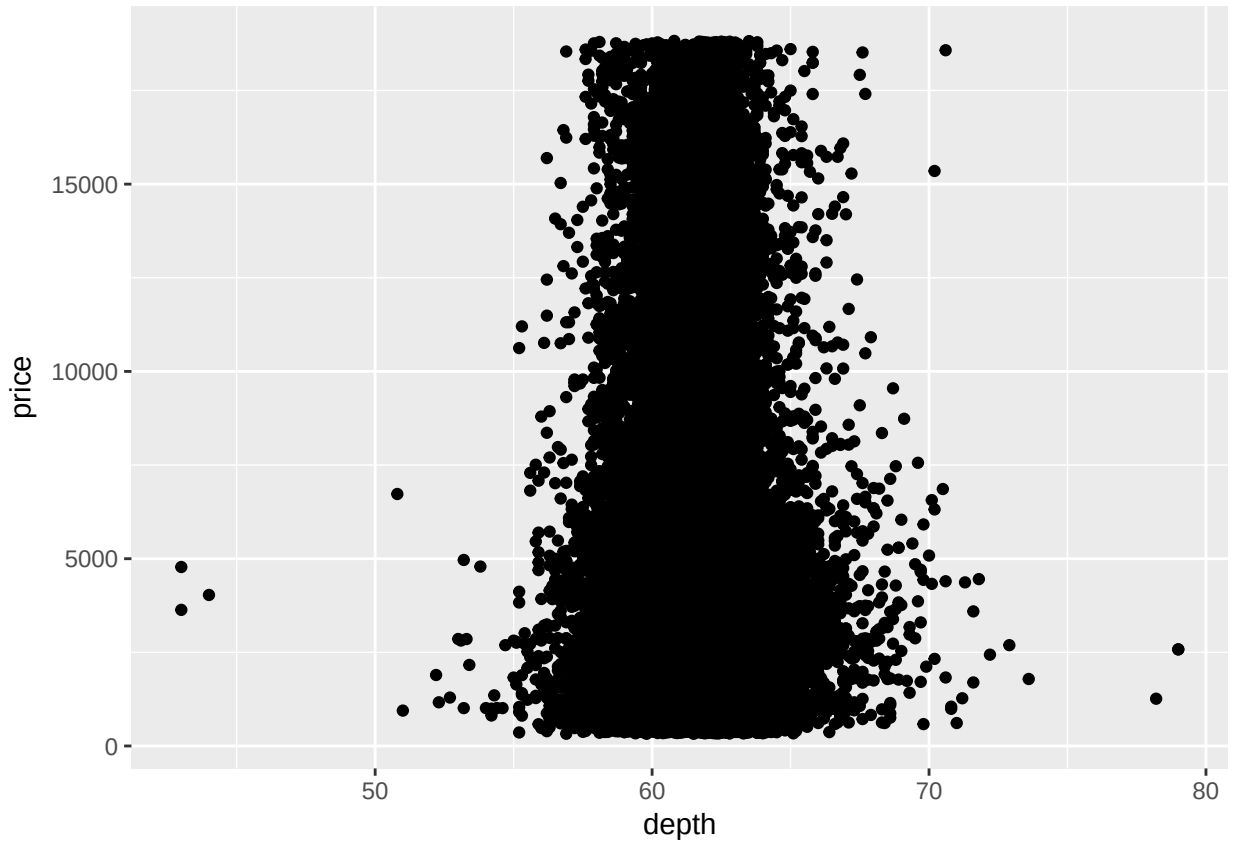
```
#TO-DO
basic_bivar = ggplot(diamonds, aes(y = price))
names=colnames(diamonds)
for(i in 1:ncol(diamonds)) {
  if(is.numeric(diamonds[[i]])) {
    object = basic_bivar + aes(x = diamonds[[i]]) + geom_point() + xlab(names[i])
  }else {object = basic_bivar + aes(x = diamonds[[i]]) + geom_boxplot() + xlab(names[i])}
  plot(object)
  }
```

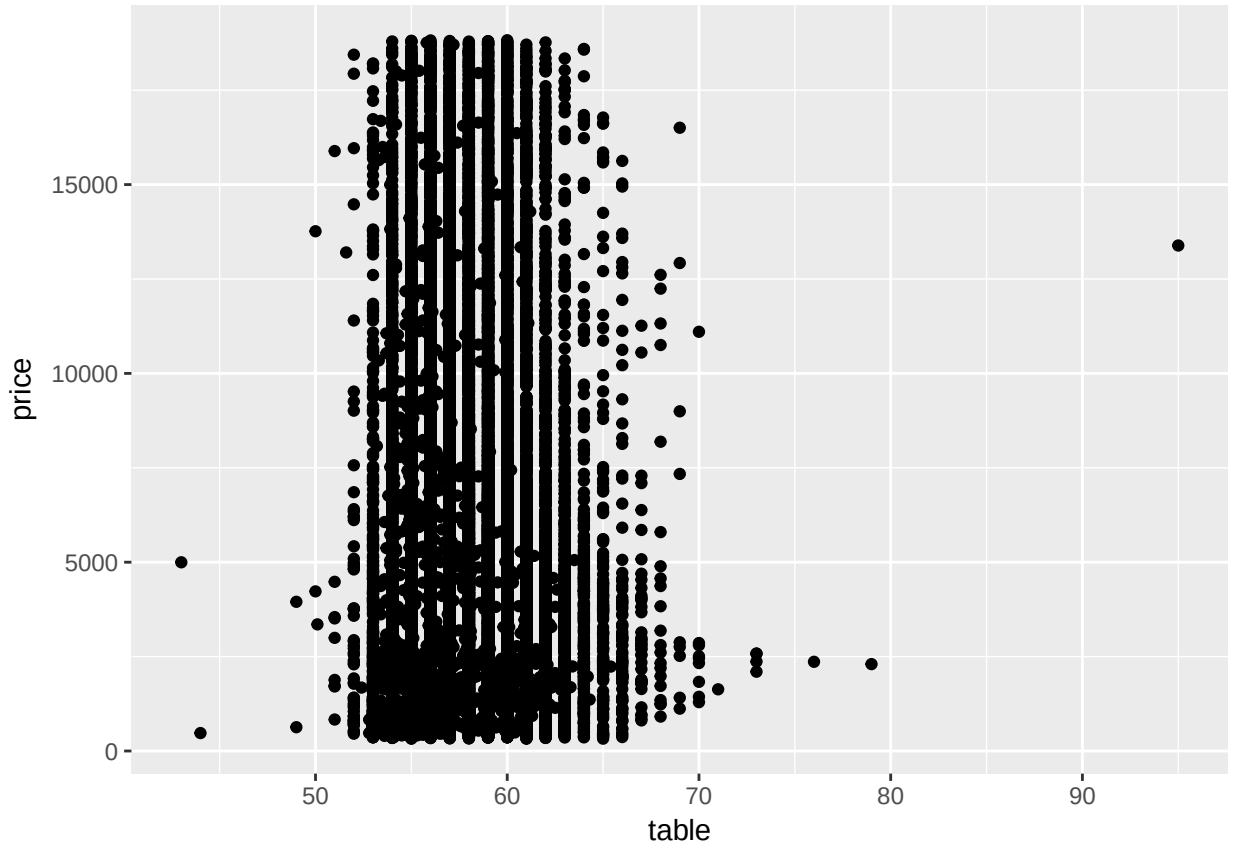

price
15000
10000
5000
0
0
1
2
3
4
5
carat

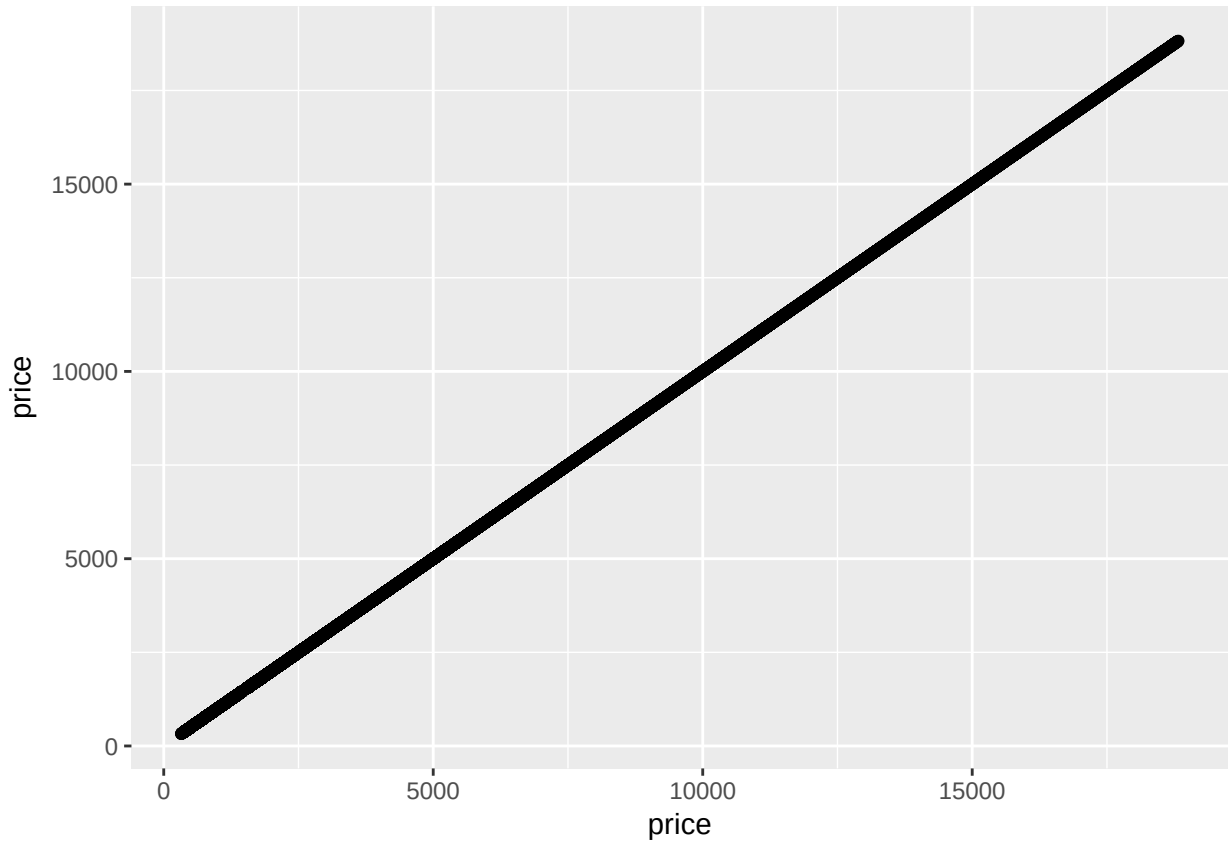

price
15000
10000
5000
0
Fair
Good
Ideal
Premium
Very Good
cut

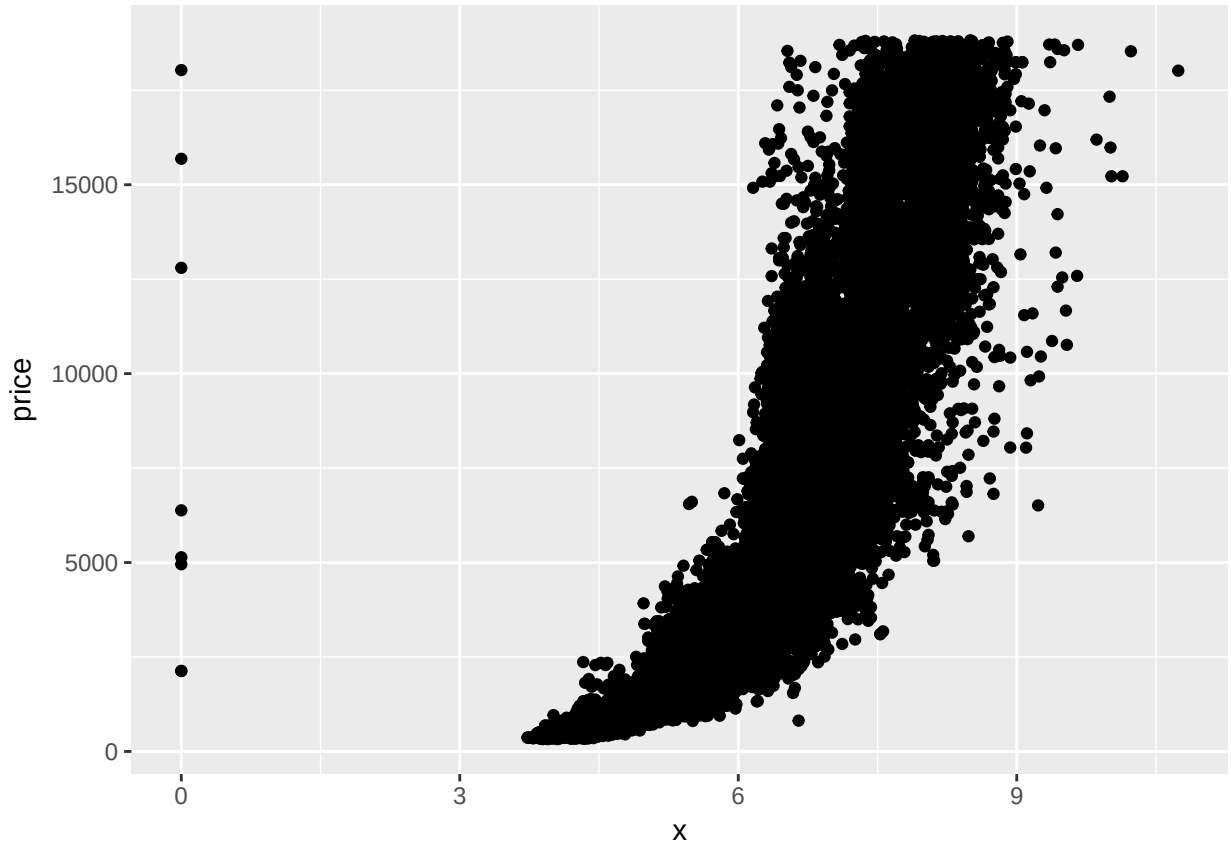

price
15000
10000
5000
0
D
E
F
G
H
I
J
color

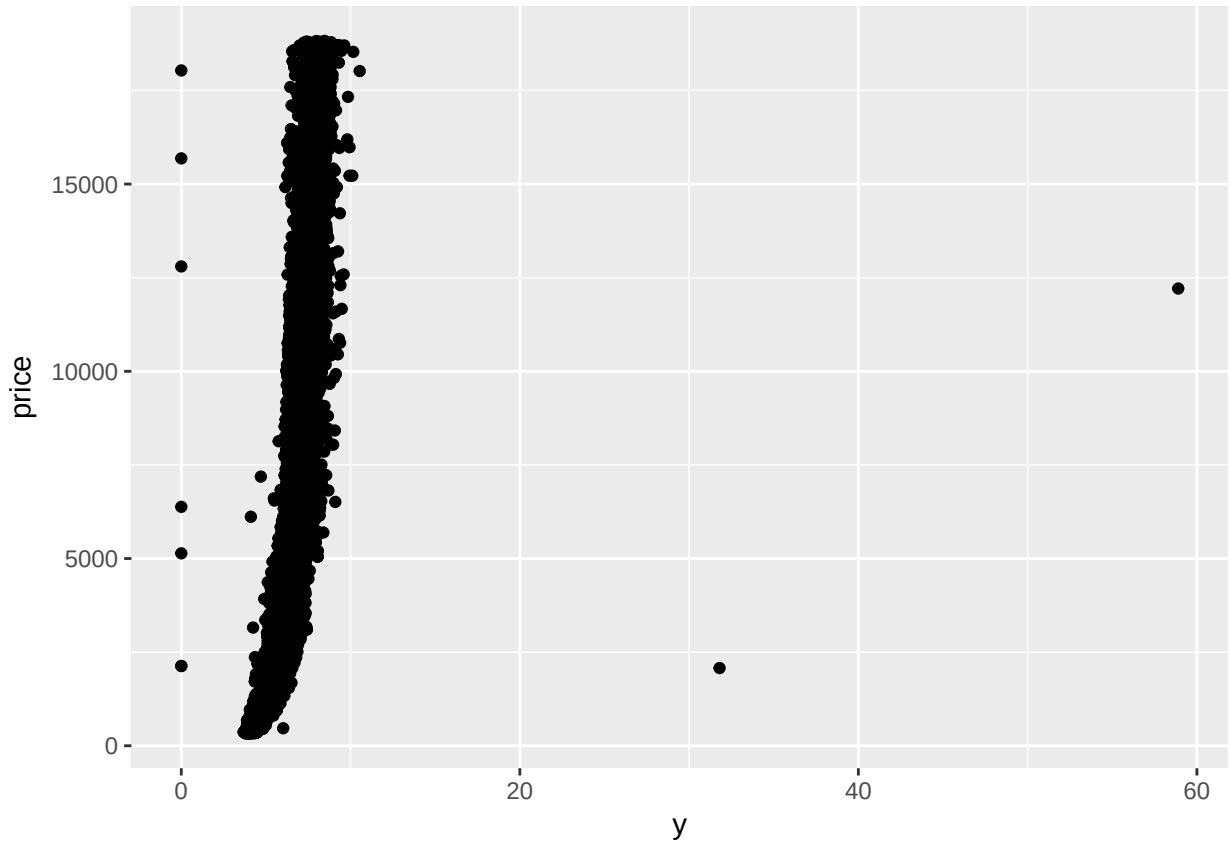

price
15000
10000
5000
0
I1
IF
SI1
SI2
VS1
VS2
VVS1
VVS2
clarity

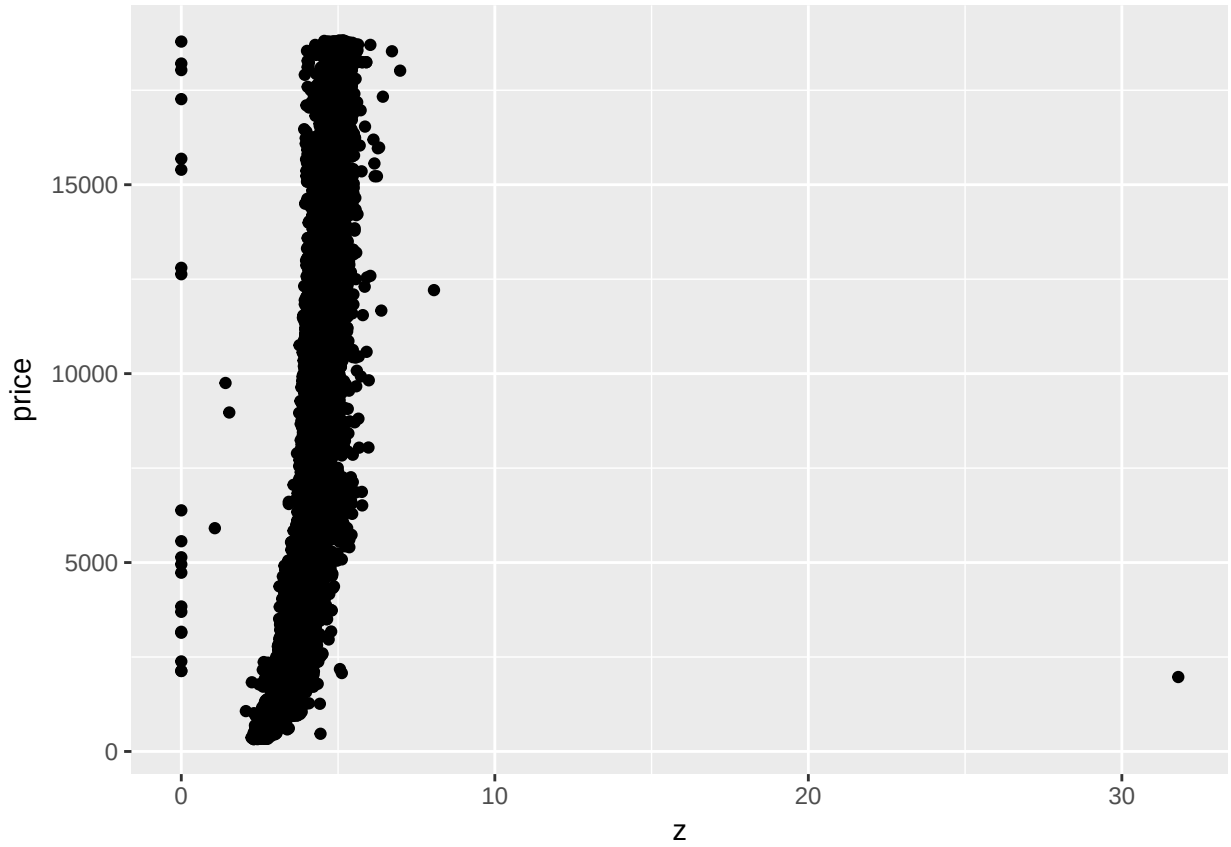

price
15000
10000
5000
0
50
60
70
80
depth


price
15000
10000
5000
0
50
60
70
80
90
table


15000
10000
5000
0
price
0
5000
10000
15000
price


15000
10000
5000
0
price
0
3
6
9
x





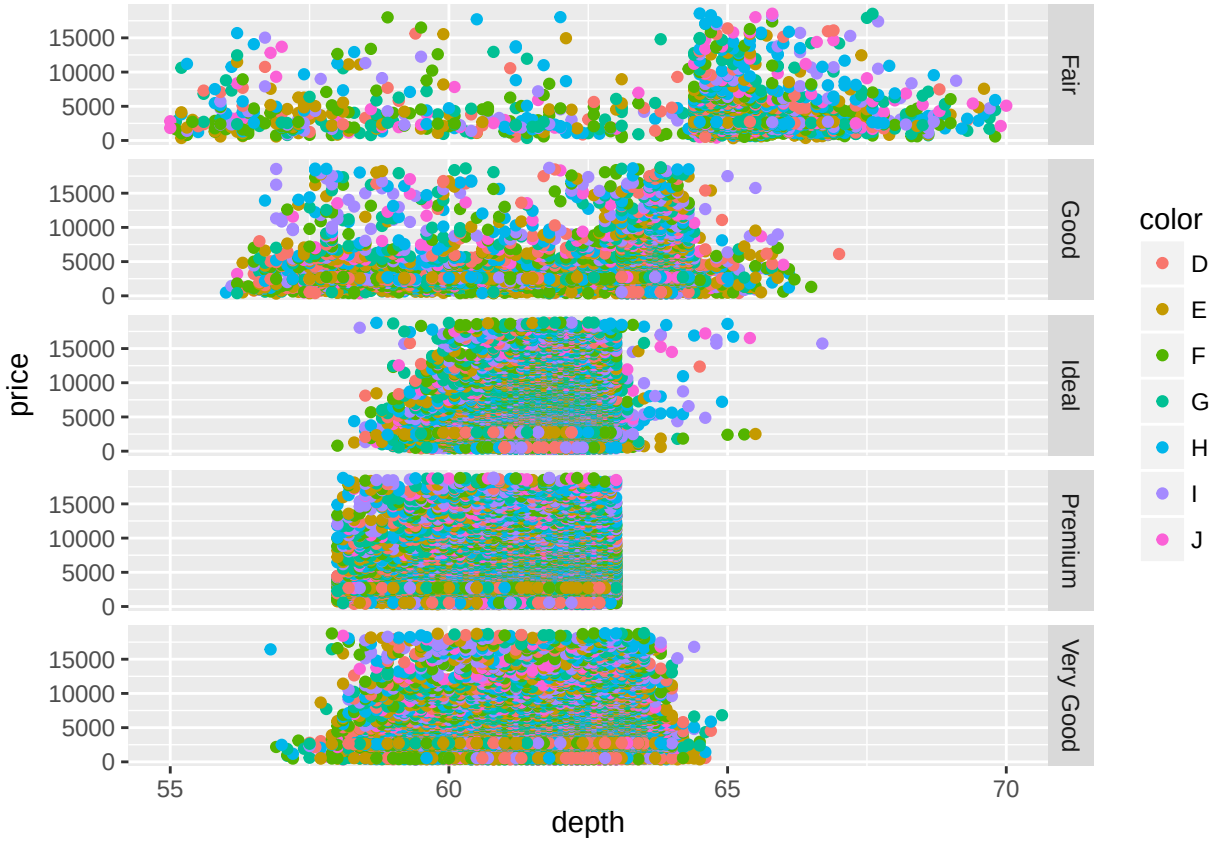
Does depth appear to be mostly independent of price?

Depth appears to be mostly independent of price, it has a "normal shape" to it. Also depth here, has nothing to do with size, it's a percetange so it should be unrelated to the price.

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

```
ggplot(diamonds, aes(x = depth, y = price)) + xlim(55,70) + ylim(300, 19000) + geom_point(aes(col = col
```

```
## Warning: Removed 45 rows containing missing values (geom_point).
```



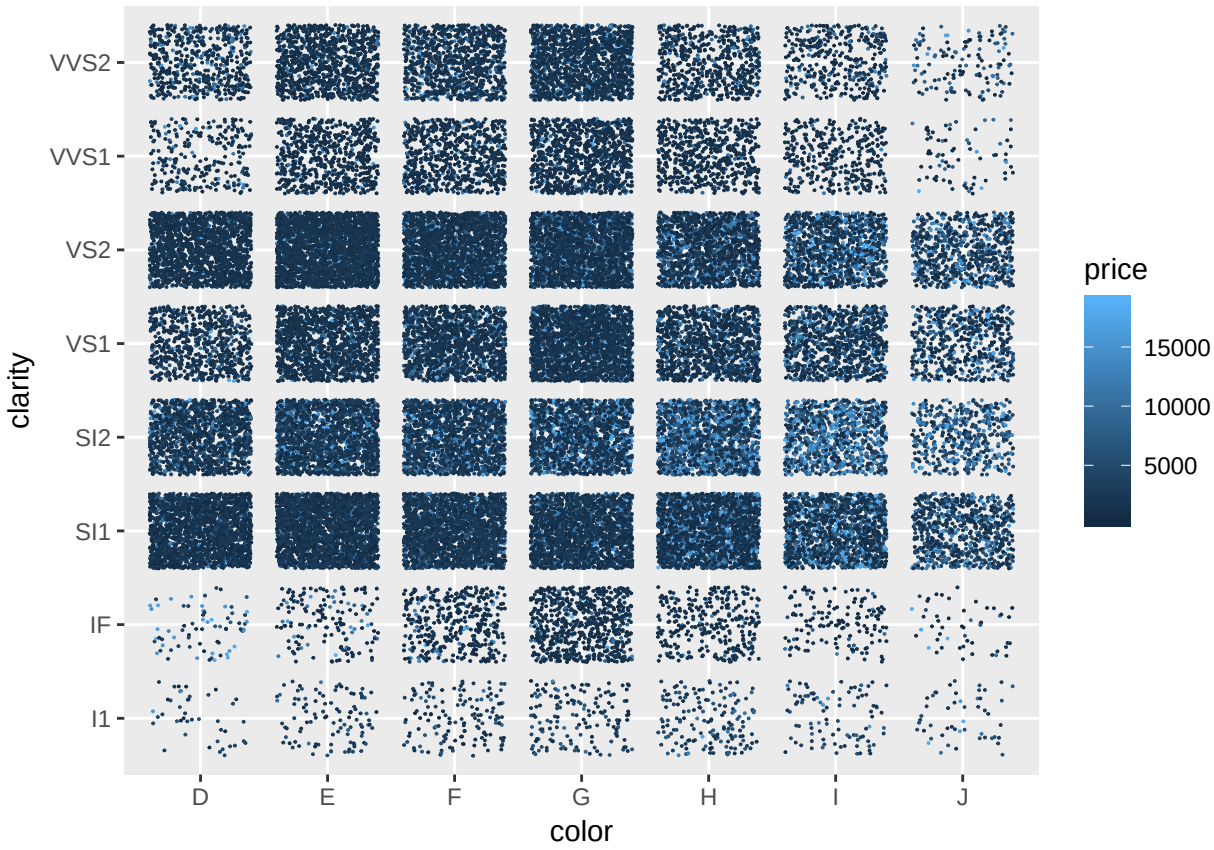
Does diamond color appear to be independent of diamond depth? Color and depth of diamonds appear to be independent because the distribution of colors along the x-axis seem about the same.

Does diamond cut appear to be independent of diamond depth? No, cut and depth appear to be dependent. The shape of the diamon depth distribution gets closer together as quality of cut increases.

Do these plots allow you to assess well if diamond cut is independent of diamond price? Yes / no No, it's hard to see the count in each graph. A bar graph would be better probably.

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 = .02)
```



Does diamond clarity appear to be mostly independent of diamond color? Clarity and Color appear mostly indepedent here. It's hard to tell the distribution of price in the densely populated boxes because they dots overlap and the darker colors are more obvious to the naked eye.

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

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

What is b, R^2 and the RMSE? What was the standard error of price originally? $b = 5764, -29.6$, $R^2 = 0.00011$, $RMSE = 3989.251$ $S_e = 3989.44$

```
coef(lm_two) #b
```

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

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

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

```
summary(lm_two)$sigma #RMSE
```

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

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

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

Are these metrics expected given the appropriate or relevant visualization(s) above? Yes these metrics were expected. The R^2 is almost 0 and $RMSE$ is relatively high, implying the price and depth are idependent.

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

```
lm_two_car = lm(price ~ carat, diamonds)
```

What is b, R^2 and the RMSE? What was the standard error of price originally? $b = [-2256.361, 7756.426]$, $R^2 = 0.85$, $RMSE = 1548.6$, $S_e = 3989.44$

```
coef(lm_two_car) #b
```

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

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

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

```
summary(lm_two_car)$sigma #RMSE
```

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

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

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

Are these metrics expected given the appropriate or relevant visualization(s) above? Yes, the data almost follows a straight line, hence the high R^2 value.

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

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

What is b, R^2 and the RMSE? What was the standard error of price originally? $b = [3169.95410 - 93.20162554.93230829.181581316.715101921.920862153.86392]$, $R^2 = 0.031$, $RMSE = 3926.78$, $S_e = 3989.44$

```
coef(anov_mod) #b
```

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

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

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

```
summary(anov_mod)$sigma #RMSE
```

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

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

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

Are these metrics expected given the appropriate or relevant visualization(s) above? No these mterics are not expected. The color of the diamonds is rated D-J(best to worst) and intuitively, better diamonds cost more money. However, this anova model returned the highest slope for the worst color diamond and the second highest slope for the second worst color etc.

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

Color is a categorical variable. In the modelling process above, the categorical variable got "dummified". Each color or 'level' became its' own variable, so each gets its' own value in b.

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.

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

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

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

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

Extrapolating in this model would be creating additional color classes. **TO-DO

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

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

What is b, R^2 and the RMSE? Also - provide an approximate 95% interval for predictions using the empirical rule. $R^2 = 0.92$, $RMSE = 1130.1$

```
coef(lm_four) #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
```

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

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

```
summary(lm_four)$sigma #RMSE
```

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

Interpret all entries in the vector b. For each unit increment in each x, the price will increase (or decrease if is negative) by the coressponding value of b

Are these metrics expected given the appropriate or relevant visualization(s) above? Can you tell from the visualizations? I can't tell from from the visualizations. This model be a hyper plane in a high dimension, thus being impossible to visualize.

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

R^2 is so high because there are so many predictors.

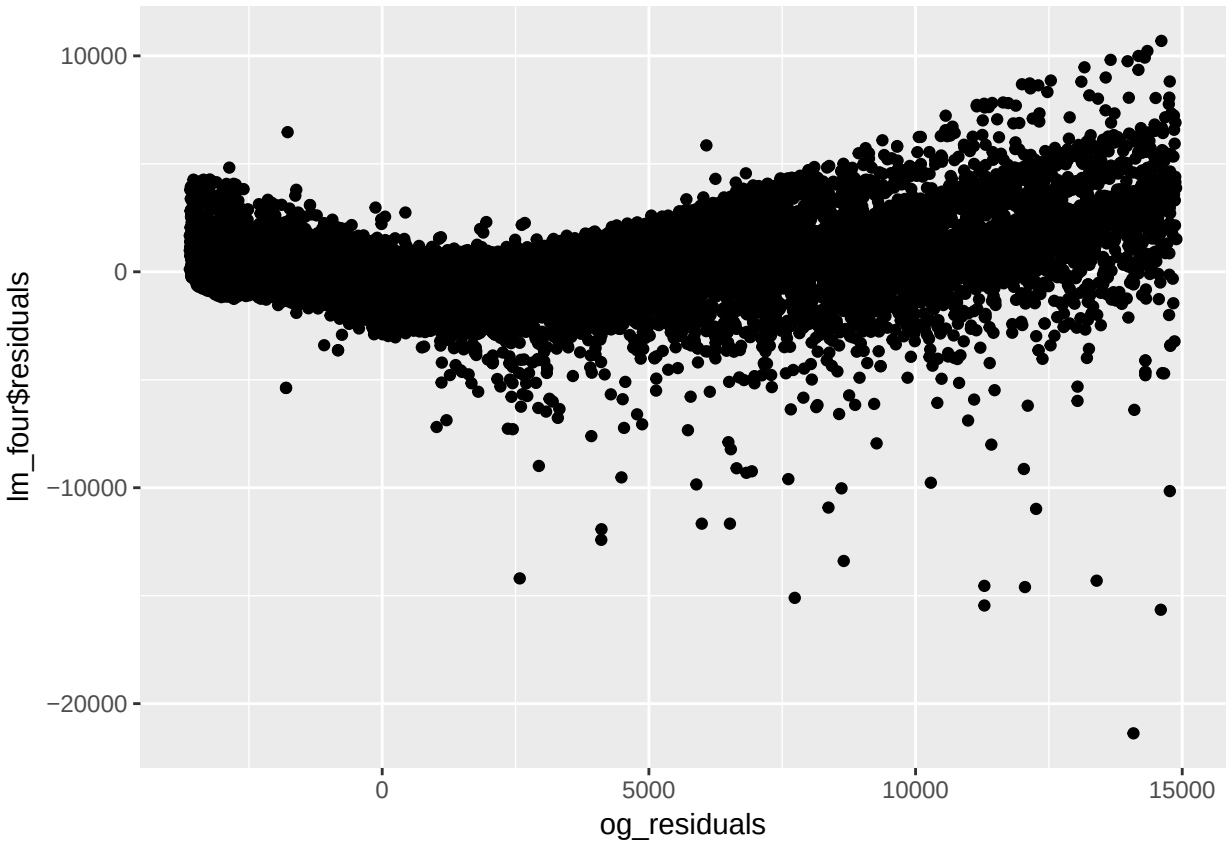
Do you think you overfit? Comment on why or why not but do not do any numerical testing or coding.

I think this is an overfit because we are building the model with all the data points given. when we are given new data on diamonds this model will fail because it was tailored too finely.

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

```
og_residuals = diamonds$price - mean(diamonds$price)

ggplot(diamonds) +aes(og_residuals, lm_four$residuals) + geom_jitter()
```



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

It seems more exponential

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

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

What is b, R^2 and the RMSE?

$b = [21804.31, 1536647.48, -76105.46, 538.33, 807.51, 747.70, 678.32, -209.44, -284.55, -496.85, -997.60, -1469.25, -2357.80, 5$
$R^2 = 0.92$, $RMSE = 1118.162$

```
b_five = lm_five$coefficients
r_sq_five = summary(lm_five)$r.squared
rmse_five = summary(lm_five)$sigma
```

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.

**TO-DO

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

In model_4, $R^2 = 0.920$ and $RMSE = 1130.094$. Model_5 is a slight improvement on model_4 because R^2 increased to 0.972 and $RMSE$ decreased to 1118.16

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

```
g <- function(features_matrix) {
  predict(lm_five, features_matrix)
}
```

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

```
lm_six = 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 an ordinal factor and not continous, so we can't make it into a plolynomial.

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
y = diamonds$price
X = cbind(1, diamonds[,-7]) #add 1 vector, drop pricce from matrix

#turn factors into numerics
X$cut = as.numeric(X$cut)
X$color = as.numeric(X$color)
X$clarity = as.numeric(X$clarity)

X = as.matrix(X)
XtX = t(X) %*% X #x transpose x
XtXinv = solve(XtX)#xtranspose x inverse(full rank)
```

What is b, R^2 and the RMSE? $b = [15902.9510978.2870.69127 - 266.4524287.8468 - 154.2983 - 93.31619 - 1184.92547.26877 - 1.688061]$, $RMSE = 99329.34$, $R^2 = 0.885$

```
b_seven = XtXinv %*% t(X) %*% y #b

#RMSE
yhat = X %*% b_seven #predictions
e = y- yhat #residuals
SSE = t(e) %*% e
MSE = 1/(ncol(X)) * SSE
RMSE = sqrt(MSE)
#R^2
s_sq_y = var(y)
s_sq_e = var(e)
Rsq = (s_sq_y - s_sq_e) / s_sq_y
```

/Are they the same as in #4?/

**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.

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

z = t(Q) %*% y
b_qr = solve(R, z) #b for qr
all.equal(b_seven, b_qr) #check if b's are equal
```

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

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

```
tran_x = t(X)
H = X %*% XtXinv %*% tran_x
yhat_qr = H %*% y
e_qr = y-yhat_qr
```

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

```
#a
all.equal(yhat_qr + e_qr, as.matrix(y))
```

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

```
#b
cor(yhat_qr, e_qr) #correlation 0 implies orthogonal
```

```
##                [,1]
## [1,] -2.062143e-12
```

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

```
lm_eight = lm(price ~ . * . + poly(carat, 5, raw = TRUE) + poly(depth, 5, raw = TRUE) + poly(table, 5, r
```

Report R^2, RMSE, the standard error of the residuals (s_e) but you do not need to report b. $R^2 = 0.97$, $RMSE = 659.2$, $S_e = 657.6$

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

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

```
summary(lm_eight)$sigma #RMSE
```

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

```
sd(lm_eight$residuals) #S_e
```

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

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

```
#TO-DO
```

How many diamonds have predictions that are wrong by $1,000 or more ? 4583 diamonds have prediction that are wrong by 1000 or more

```
j = 0
for(i in lm_eight$residuals) {
  if(abs(i) > 1000) {
    j = j + 1
  }
}
j
```

```
## [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.*

**TO-DO

What is the degrees of freedom in this model?

```
#TO-DO
```

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

**TO-DO

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

**TO-DO

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

** TO-DO

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.

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

```
X = lm_eight$model #set x equal to the D created in previous question
k=10
n=53940

test_index = sample(1:n, 1/k*n)
train_index = setdiff(1:n, test_index)

x_test = X[test_index,]
y_test =
```

```
## Error: <text>:10:0: unexpected end of input
## 8: x_test = X[test_index,]
## 9: y_test =
##    ^
```

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?

** 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(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:

```
#TO-DO

mod = lm(y_train ~ X_train)
```

```
## Error in eval(predvars, data, env): object 'y_train' not found
```

```
y_hat_oos = predict(mod, X_test)
```

```
## Error in predict.lm(mod, X_test): object 'X_test' not found
```