Lab4-Stat131A-ColinAsbill

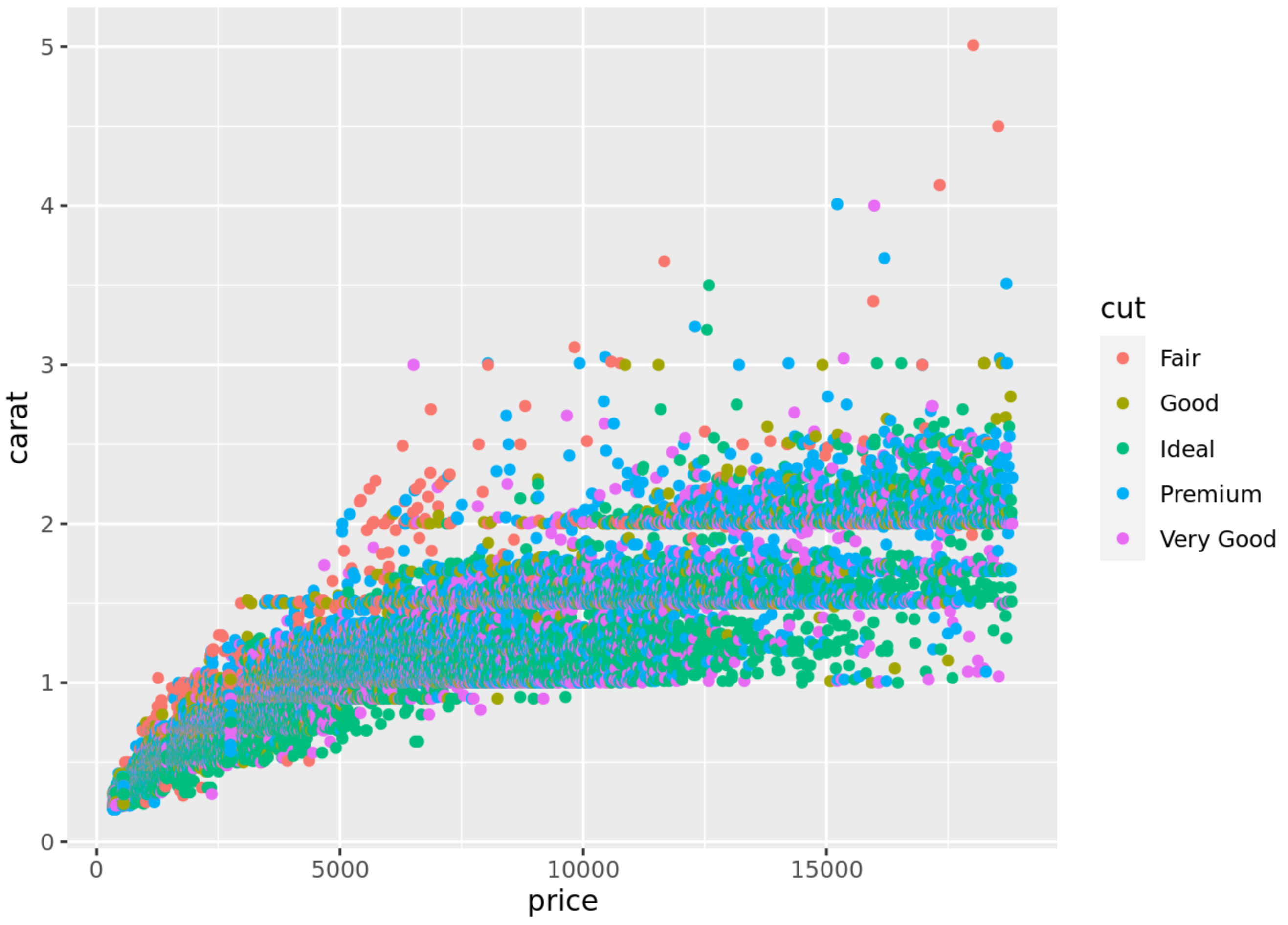
AUTHOR
Colin Asbill

```
library(tidyverse)
library(broom)
library(GGally)
library(leaps)
diamonds <- read_csv("diamonds.csv")
```

Part 1 EDA and First Fit

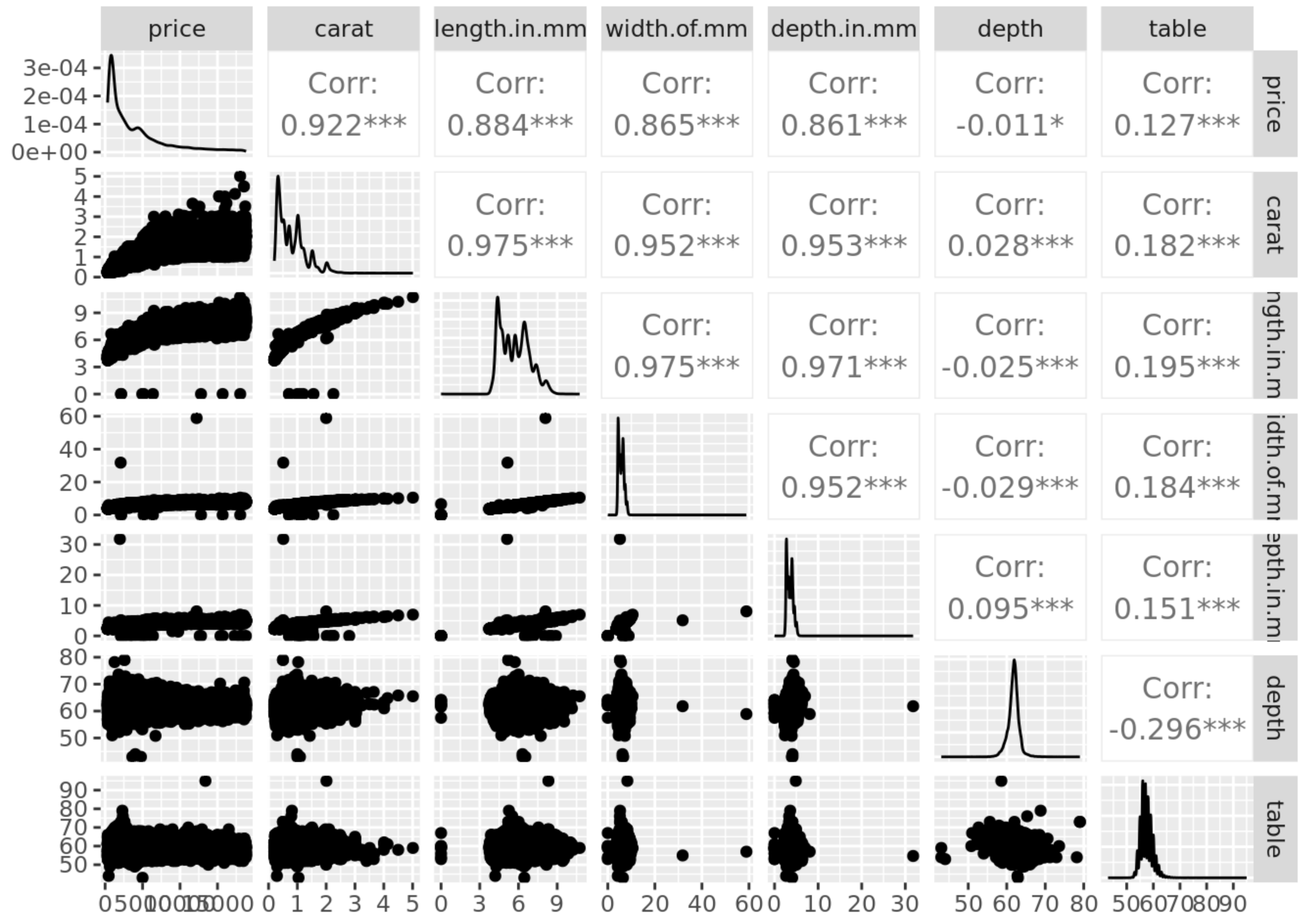
Question 1

```
diamonds %>%
  ggplot(aes(x = price, y = carat)) +
  geom_point(aes(color = cut))
```

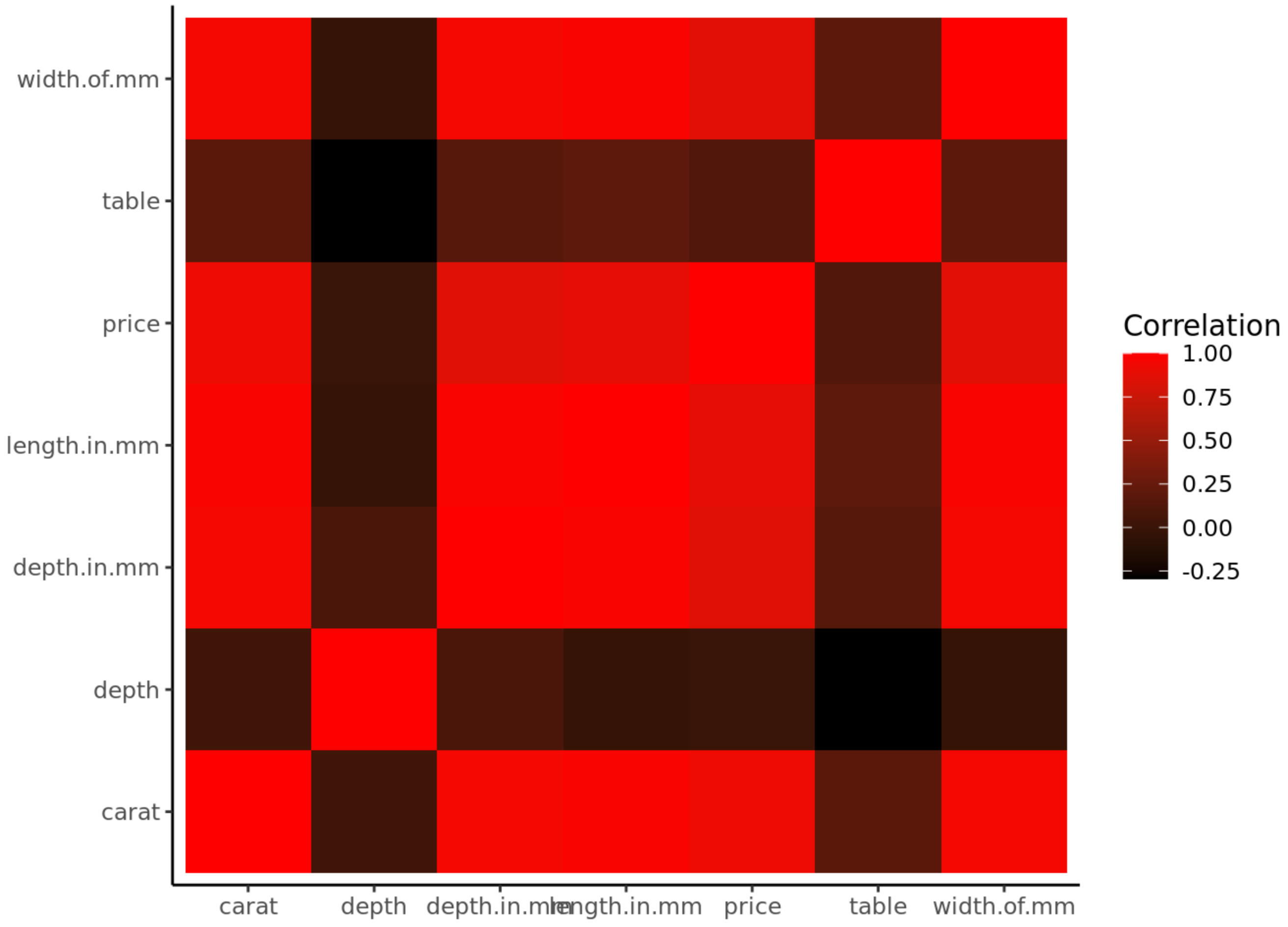


Looking at this graph that compares price with the carat of the diamonds we can see that carat and price have correlation, but the cut variable might be affected by outliers such as the 5 carat diamonds in the top right being very expensive, but having the lowest quality cut.

```
diamonds %>%
  select(price, carat, length.in.mm, width.of.mm, depth.in.mm, depth, table) %>%
  ggpairs()
```



```
diamonds_new <- diamonds %>% select(price, carat, length.in.mm, width.of.mm, depth.in.mm, depth, 
as.data.frame(cor(diamonds_new)) %>%
  rownames_to_column("Variables_1") %>%
  pivot_longer(-c(Variables_1), names_to = "Variables_2", values_to = "Correlation") %>%
  ggplot(mapping = aes(x = Variables_1, y = Variables_2)) +
  geom_tile(aes(fill = Correlation)) +
  scale_fill_gradient(low = "black", high = "red")+
  theme_classic() +
  theme(axis.title.x = element_blank(),
        axis.title.y = element_blank())
```



We can see Price is not very correlated with table and depth, while the rest of the continuous variables have similar levels of correlation

Question 2

```
lm_d <- lm(price ~ carat + depth + table + length.in.mm + width.of.mm + depth.in.mm, data = diamon
tidy(lm_d) %>%
  select(term, estimate, std.error)
```

```
# A tibble: 7 × 3
  term         estimate std.error
  <chr>           <dbl>     <dbl>
1 (Intercept)   20849.     448.  
2 carat         10686.      63.2 
3 depth          -203.       5.50
4 table          -102.       3.08
5 length.in.mm  -1316.      43.1 
6 width.of.mm      66.3     25.5 
7 depth.in.mm      41.6     44.3 

  Coefficient  std.error
```

carat 10686.3091 63.200807

depth -203.1541 5.503984

table -102.4457 3.084213

length.in.mm -1315.6678 43.070264

width.of.mm 66.3216 25.523021

depth.in.mm 41.6277 44.304632

Question 3

price = 20849.3164 + 10686.3091*carat* + *-203.1541*depth + -102.4457*table* + *-1315.6678*length.in.mm + 66.3216*width.of.mm* + *41.6277*depth.in.mm

Coefficient Interpretation

Question 1

For every one unit increase in carat there is a corresponding $10,686.3091 increase in price

Question 2

```
diamonds$carat_standardized <- (diamonds$carat - mean(diamonds$carat))/(sd(diamonds$carat))
diamonds$depth_standardized <- (diamonds$depth - mean(diamonds$depth))/(sd(diamonds$depth))
diamonds$table_standardized <- (diamonds$table - mean(diamonds$table))/(sd(diamonds$table))
diamonds$length_standardized <- (diamonds$length.in.mm - mean(diamonds$length.in.mm))/(sd(diamond
diamonds$width_standardized <- (diamonds$width.of.mm - mean(diamonds$width.of.mm))/(sd(diamonds$w
diamonds$depthmm_standardized <- (diamonds$depth.in.mm - mean(diamonds$depth.in.mm))/(sd(diamonds

lm_d_standard <- lm(price ~ carat_standardized + depth_standardized + table_standardized + length_
tidy(lm_d_standard) %>%
  select(term, estimate, std.error)
```

```
# A tibble: 7 × 3
  term                 estimate std.error
  <chr>                   <dbl>     <dbl>
1 (Intercept)             3933.      6.45
2 carat_standardized      5065.     30.0
3 depth_standardized      -291.      7.89
4 table_standardized      -229.      6.89
5 length_standardized    -1476.     48.3
6 width_standardized        75.7    29.2
7 depthmm_standardized      29.4    31.3
```

You can not interpret standardized coefficients the same way as non standardized because the standardization transforms the values and now you can only see how each value is weighted in impact towards price.

Question 3

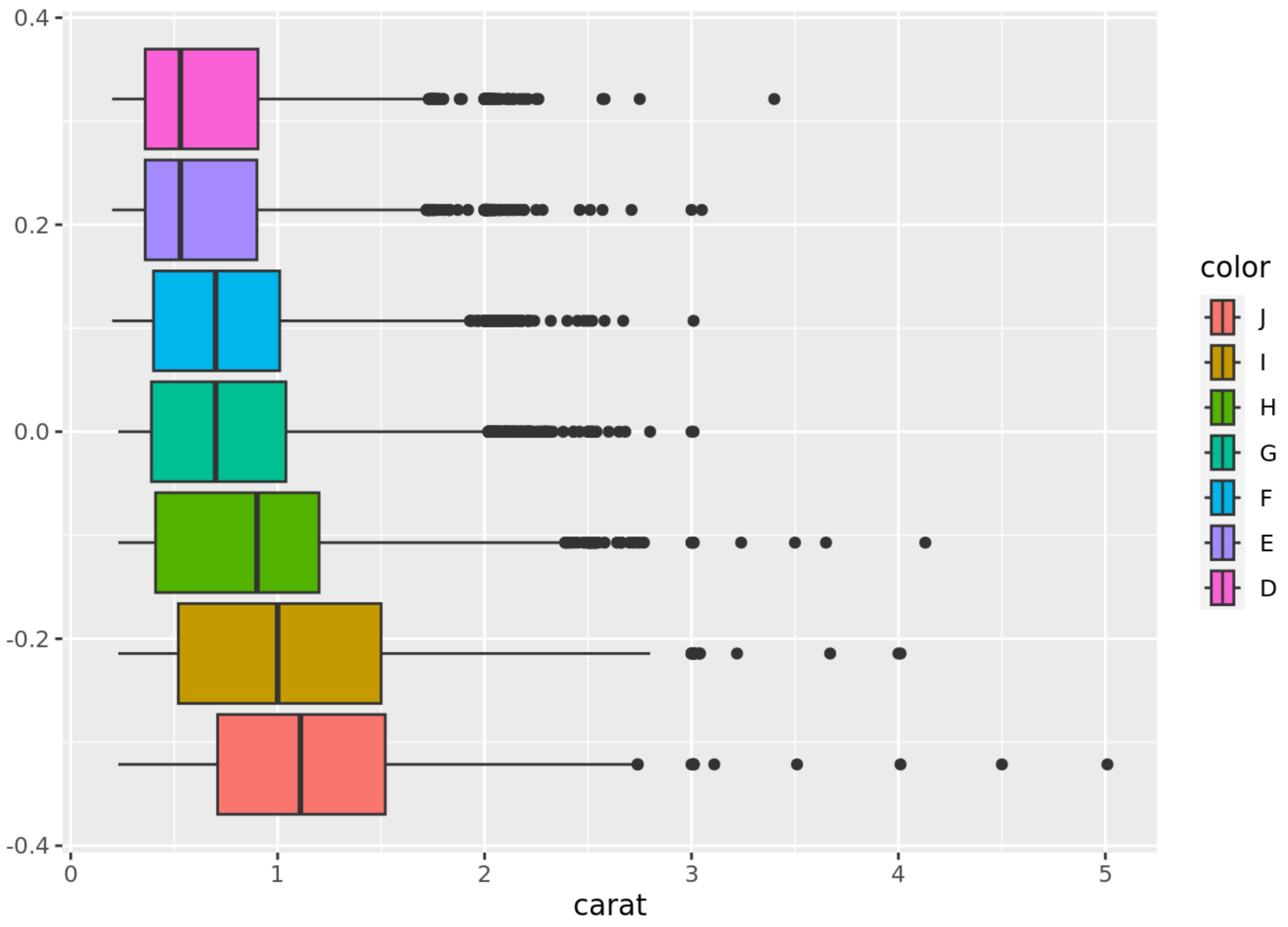
The seller should first look at the second model to determine how impactful each variable is in the model and see that carat size has the biggest impact on price. Then the seller should input the different carat sizes into the first model to get an estimated price to negotiate with the diamond store clerk. Only the first model can be used to estimate price.

Question 4

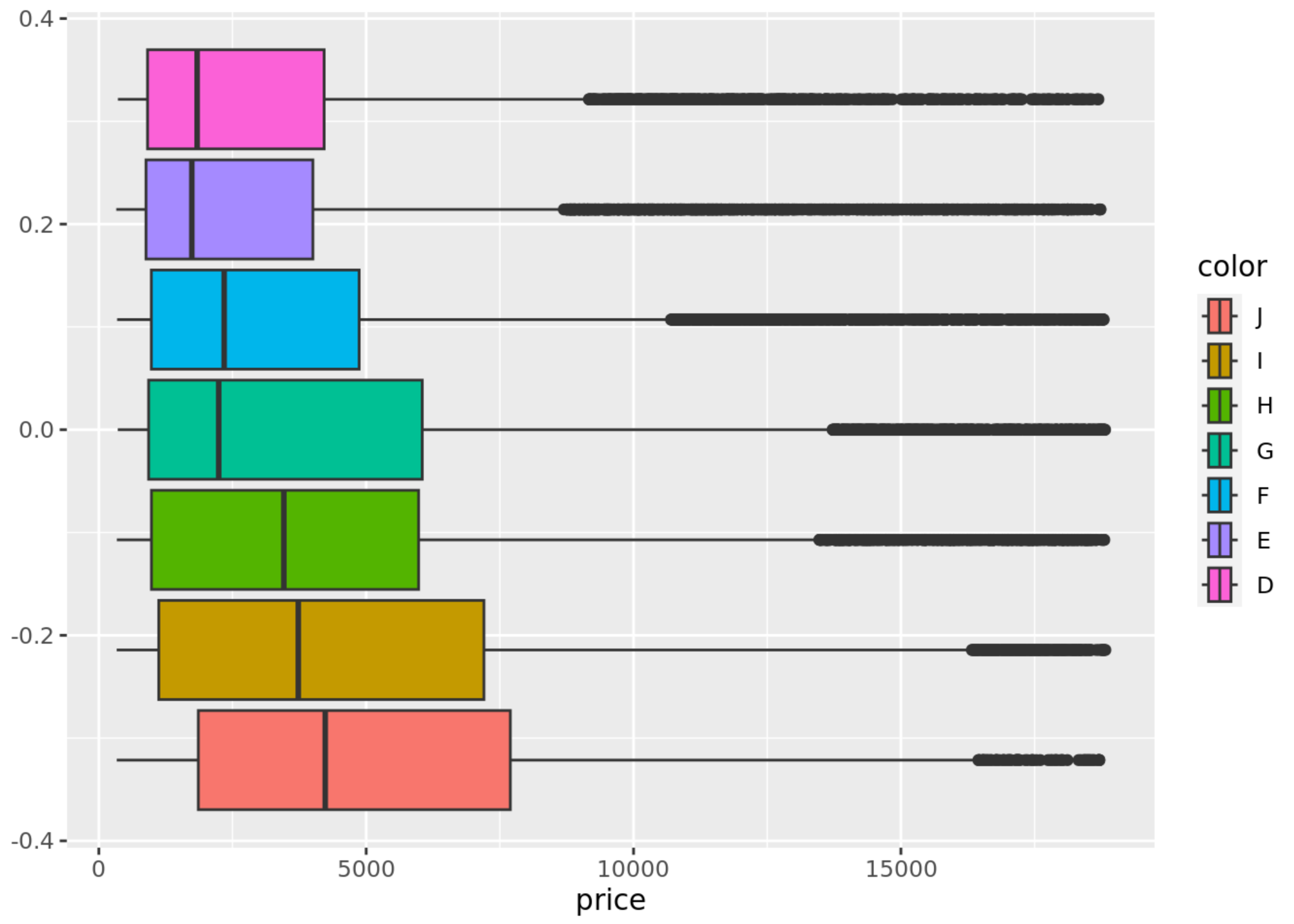
```
diamonds$cut = factor(diamonds$cut, levels = c("Fair", "Good", "Very Good", "Premium", "Ideal"))
diamonds$color = factor(diamonds$color, levels = c("J", "I", "H", "G", "F", "E", "D"))
diamonds$clarity = factor(diamonds$clarity, levels = c("I1", "SI1", "SI2", "VS1", "VS2", "VVS1",
```

Question 5

```
diamonds %>%
  ggplot(aes(x = carat, fill = color)) +
  geom_boxplot()
```



```
diamonds %>%
  ggplot(aes(x = price, fill = color)) +
  geom_boxplot()
```



Question 6

```
lmcat <- lm(price ~ carat + color, data = diamonds)
tidy(lmcat)
```

```
# A tibble: 8 × 5
  term        estimate std.error statistic   p.value
  <chr>          <dbl>     <dbl>     <dbl>     <dbl>
1 (Intercept)   -4051.      32.2    -126.  0
2 carat          8067.      14.0     575.  0
3 colorI          859.      34.3      25.1 8.13e-138
4 colorH         1182.      32.3      36.6 1.82e-289
5 colorG         1829.      31.5      58.0 0
6 colorF         1834.      32.2      57.0 0
7 colorE         1821.      32.3      56.4 0
8 colorD         1914.      33.8      56.7 0
```

price = intercept + 8066.62 * Carat + 858.74*colorI + 1182.23*colorH + 1828.93 *colorG + 1834.20*colorF + 1820.69 * colorE + 1914.47 * colorD

All the color variables are indicators that compare their value vs the baseline colorJ, so colorD = 1 if the Color of the Diamond is D and 0 otherwise. colorD increases the price of the Diamond by $1914.47 compared to the price if the Diamond was colorJ.

Question 7

As the color increases/gets closure to D the price increases with the exception of ColorE where there is a slight dip, but this could be the result of a small sample size skewing the data because there is more higher carat colorF. This suggests that color has a postive association with price.

Question 8

```
lmint <- lm(price ~ carat:color, data = diamonds)
tidy(lmint)
```

```
# A tibble: 8 × 5
  term          estimate std.error statistic p.value
  <chr>            <dbl>     <dbl>     <dbl>   <dbl>
1 (Intercept)     -2557.      12.9     -198.       0
2 carat:colorJ     6847.      22.8      300.       0
3 carat:colorI     7524.      19.3      390.       0
4 carat:colorH     7699.      18.6      414.       0
5 carat:colorG     8508.      19.9      427.       0
6 carat:colorF     8563.      22.4      383.       0
7 carat:colorE     8500.      24.6      346.       0
8 carat:colorD     8638.      28.0      308.       0
```

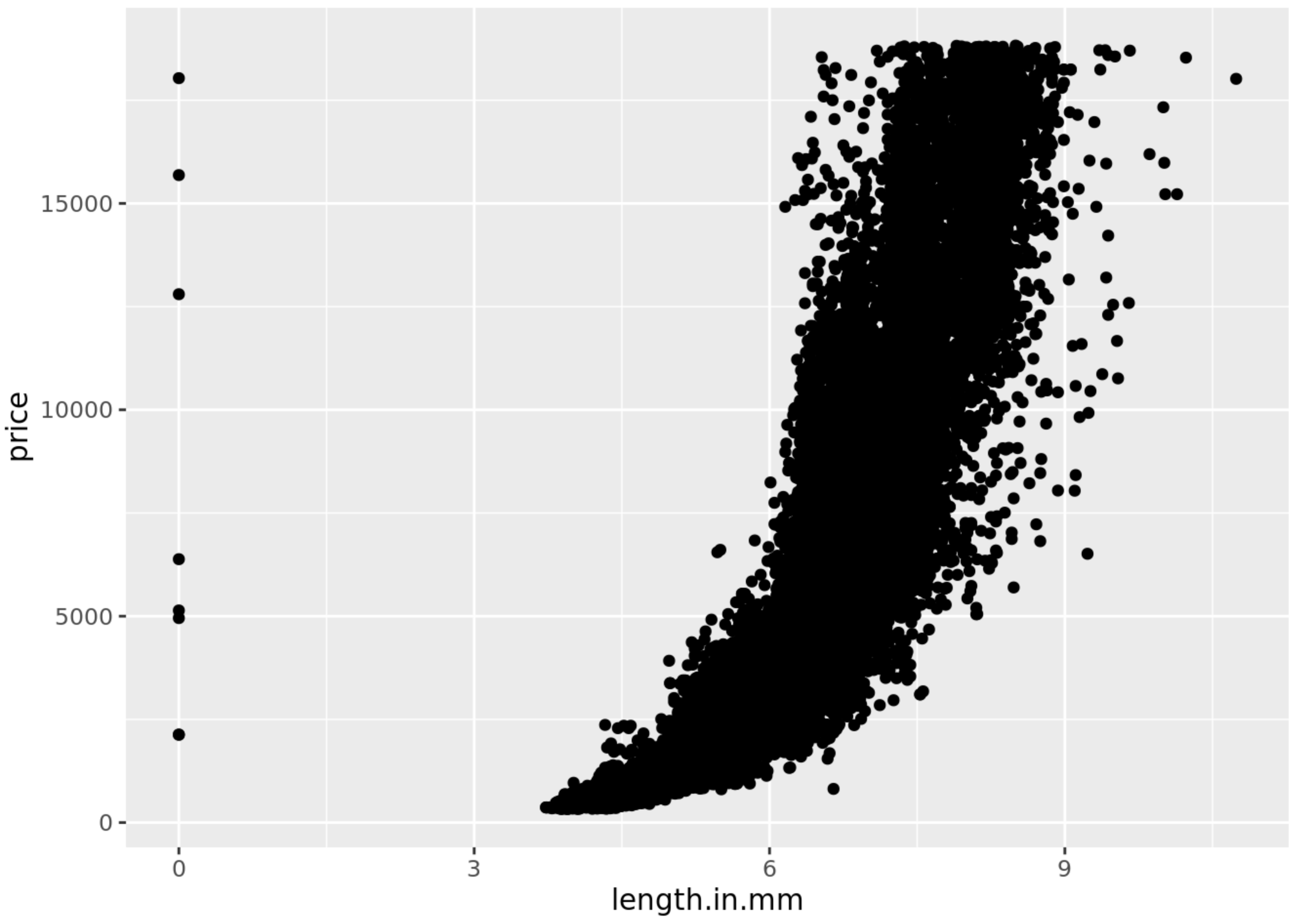
The interaction term between carat and colorD can be interpreted as a one unit increase in carat results in a 8638.10 increase in price if the color is colorD.

Question 9

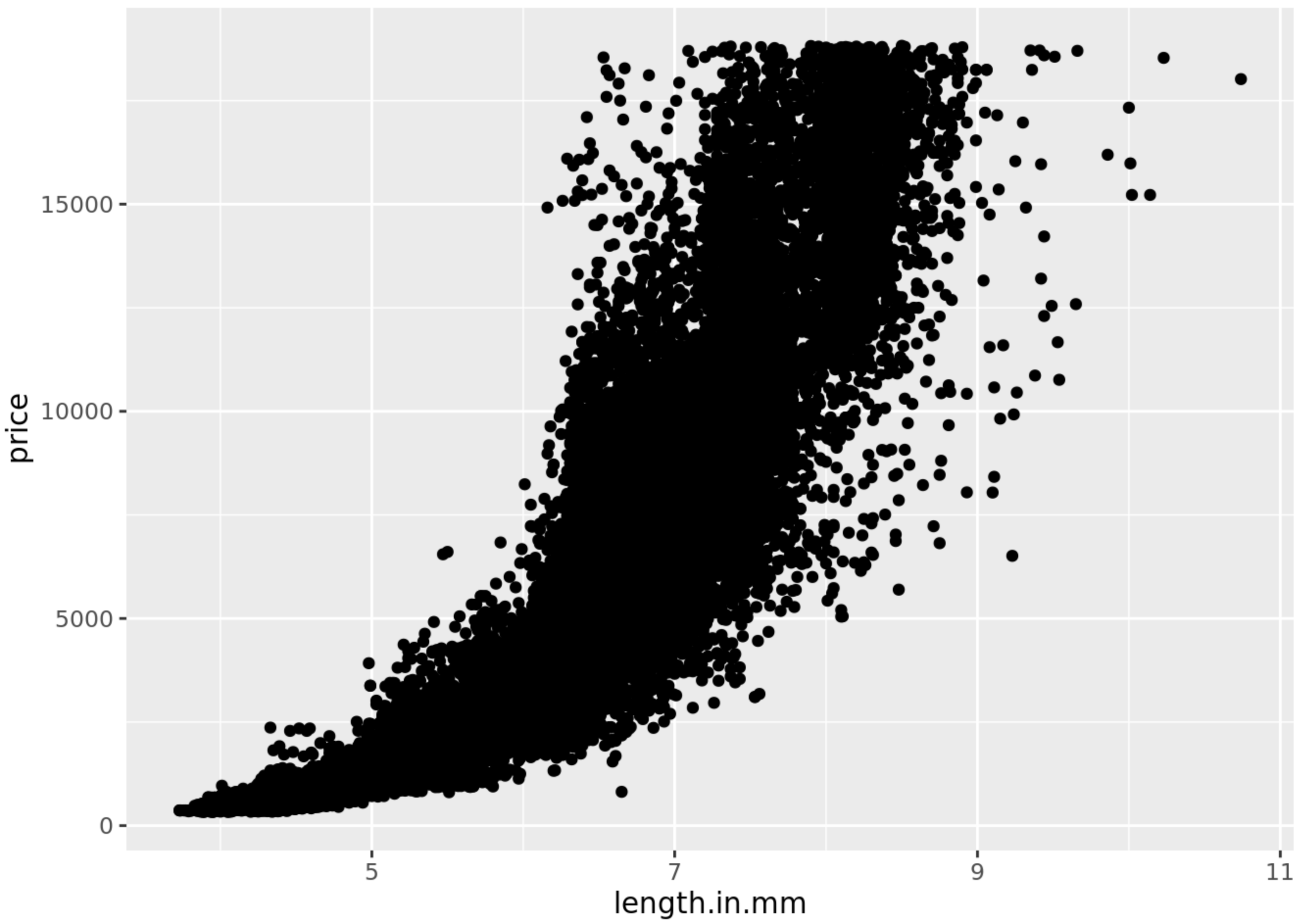
```
lm_d_standard2 <- lm(price ~ length_standardized + width_standardized + depthmm_standardized, dat
tidy(lm_d_standard2) %>%
  select(term, estimate, std.error)
```

```
# A tibble: 4 × 3
  term                 estimate std.error
  <chr>                   <dbl>     <dbl>
1 (Intercept)             3933.      8.01
2 length_standardized     3130.     46.0
3 width_standardized       250.     36.1
4 depthmm_standardized     159.     33.6
```

```
diamonds%>%
ggplot(aes(x = length.in.mm, y= price)) +
  geom_point()
```



```
diamonds_filter <- diamonds %>% filter(length.in.mm > 0)
diamonds_filter %>%
  ggplot(aes(x = length.in.mm, y= price)) +
  geom_point()
```



```
lm_d2 <- lm(price ~ length.in.mm + width.of.mm, data = diamonds_filter)
tidy(lm_d2)
```

```
# A tibble: 3 × 5
  term         estimate std.error statistic  p.value
  <chr>           <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)   -14194.      41.3   -343.   0
2 length.in.mm    2958.      31.8     93.0  0
3 width.of.mm      204.      31.2      6.53 6.64e-11
```

Looking at the standardized model we can see that length.in.mm is contributing the most to the price by far with these three variables, with a value more than 10 times the next closest, so these other variables don't add much to the model, but I'll include width so there is a second variable. Then looking at the scatter plot we should filter by length > 0 to eliminate those outliers. The model is just price = intercept + 2958.266 * length.in.mm + 203.848* width.of.mm

Question 10

```
lm_d2 <- lm(price ~ length.in.mm + width.of.mm, data = diamonds_filter)
tidy(lm_d2)
```

```
# A tibble: 3 × 5
  term         estimate std.error statistic  p.value
  <chr>            <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)   -14194.       41.3    -343.   0
2 length.in.mm    2958.       31.8      93.0  0
3 width.of.mm      204.       31.2       6.53 6.64e-11
```

A one unit increase in length.in.mm leads to a 2958.266 increase in price

Inference Wine

Question 1

```
wine <- read_csv("winequality-red.csv")
```

```
Rows: 1599 Columns: 12
── Column specification ────────────────────────────────────────────────────────
Delimiter: ","
dbl (12): fixed.acidity, volatile.acidity, citric.acid, residual.sugar, chlo...

ℹ Use `spec()` to retrieve the full column specification for this data.
ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
lmW <- lm(quality ~ fixed.acidity + volatile.acidity + citric.acid + residual.sugar + chlorides +
tidy(lmW) %>%
  select(term, estimate, std.error)
```

```
# A tibble: 12 × 3
   term                 estimate std.error
   <chr>                   <dbl>     <dbl>
 1 (Intercept)          22.0     21.2
 2 fixed.acidity         0.0250   0.0259
 3 volatile.acidity     -1.08     0.121
 4 citric.acid          -0.183    0.147
 5 residual.sugar        0.0163   0.0150
 6 chlorides            -1.87     0.419
 7 free.sulfur.dioxide   0.00436  0.00217
 8 total.sulfur.dioxide -0.00326  0.000729
 9 density             -17.9     21.6
10 pH                   -0.414    0.192
11 sulphates             0.916    0.114
12 alcohol               0.276    0.0265
```

Question 2

```
tidy(lmW)
```

```
# A tibble: 12 × 5
   term                 estimate std.error statistic  p.value
   <chr>                   <dbl>     <dbl>     <dbl>    <dbl>
 1 (Intercept)          22.0     21.2          1.04  3.00e- 1
 2 fixed.acidity         0.0250   0.0259       0.963 3.36e- 1
 3 volatile.acidity     -1.08     0.121       -8.95  9.87e-19
 4 citric.acid          -0.183    0.147       -1.24  2.15e- 1
 5 residual.sugar        0.0163   0.0150       1.09  2.76e- 1
 6 chlorides            -1.87     0.419       -4.47  8.37e- 6
 7 free.sulfur.dioxide   0.00436  0.00217      2.01  4.47e- 2
 8 total.sulfur.dioxide -0.00326  0.000729    -4.48  8.00e- 6
 9 density             -17.9     21.6         -0.827 4.09e- 1
10 pH                   -0.414    0.192       -2.16  3.10e- 2
11 sulphates             0.916    0.114        8.01  2.13e-15
12 alcohol               0.276    0.0265      10.4   1.12e-24
```

Null Hypothesis for pH would be pH = 0, the test stat is equal to the estimate/std.error, so -0.4136/1.91e-01 the p-value is 3.1 * e-02 the t-stat is above a magnitude of 2 so meaning that we should reject the null. A t-stat of 2 is statistically significant to the 5% level.

Question 3

```r
library(dplyr)

# Bootstrap function
MAKE_LM_BOOTSTRAP_STATS <- function(x_columns, y) {
  indices <- seq(from = 1, to = nrow(x_columns), by = 1)
  selected <- sample(x = indices, size = length(indices), replace = TRUE)

  boot_x <- x_columns %>%
    slice(selected)
  boot_y <- y[selected]

  boot <- data.frame(boot_x, y = boot_y)

  linear_model <- lm(y ~ ., data = boot)
  stat <- coef(linear_model)

  return(stat)
}

x <- wine %>% select(fixed.acidity , volatile.acidity , citric.acid , residual.sugar , chlorides
y <- wine$quality

# Generate bootstrap distributions
bootstrap_distributions <-
  as.data.frame(replicate(n = 1000, MAKE_LM_BOOTSTRAP_STATS(x_columns = x, y = y)))

bootstrap_fixed <- as.numeric(bootstrap_distributions["fixed.acidity",])
bootstrap_volatile <- as.numeric(bootstrap_distributions["volatile.acidity",])
```

```
bootstrap_citric <- as.numeric(bootstrap_distributions["citric.acid",])
bootstrap_sugar <- as.numeric(bootstrap_distributions["residual.sugar",])
bootstrap_chlorides <- as.numeric(bootstrap_distributions["chlorides",])
bootstrap_free <- as.numeric(bootstrap_distributions["free.sulfur.dioxide",])
bootstrap_total <- as.numeric(bootstrap_distributions["total.sulfur.dioxide",])
bootstrap_density <- as.numeric(bootstrap_distributions["density",])
bootstrap_pH <- as.numeric(bootstrap_distributions["pH",])
bootstrap_sulphates <- as.numeric(bootstrap_distributions["sulphates",])
bootstrap_alcohol <- as.numeric(bootstrap_distributions["alcohol",])

# Compute 95% confidence intervals
CI_fixed <- quantile(bootstrap_fixed, c(0.025, 0.975))
CI_volatile <- quantile(bootstrap_volatile, c(0.025, 0.975))
CI_citric <- quantile(bootstrap_citric, c(0.025, 0.975))
CI_chlorides <- quantile(bootstrap_chlorides, c(0.025, 0.975))
CI_sugar <- quantile(bootstrap_sugar, c(0.025, 0.975))
CI_free <- quantile(bootstrap_free, c(0.025, 0.975))
CI_total <- quantile(bootstrap_total, c(0.025, 0.975))
CI_density <- quantile(bootstrap_density, c(0.025, 0.975))
CI_pH <- quantile(bootstrap_pH, c(0.025, 0.975))
CI_sulphates <- quantile(bootstrap_sulphates, c(0.025, 0.975))
CI_alcohol <- quantile(bootstrap_alcohol, c(0.025, 0.975))
print(CI_fixed)
```

```
       2.5%       97.5% 
-0.03791511  0.08506279 
```

```
print(CI_volatile)
```

```
      2.5%      97.5% 
-1.3399732 -0.8070485 
```

```
print(CI_citric)
```

```
      2.5%      97.5% 
-0.4590801  0.1030521 
```

```
print(CI_chlorides)
```

```
      2.5%      97.5% 
-2.8197102 -0.9443352 
```

```
print(CI_sugar)
```

```
       2.5%       97.5% 
-0.02250166  0.05643550 
```

```
print(CI_free)
```

```
         2.5%         97.5%
-3.745692e-05  8.602863e-03
```

```
print(CI_total)
```

```
        2.5%        97.5%
-0.004773532 -0.001926829
```

```
print(CI_density)
```

```
     2.5%     97.5%
-67.25123  32.00337
```

```
print(CI_pH)
```

```
         2.5%         97.5%
-0.8236101273  0.0008394741
```

```
print(CI_sulphates)
```

```
     2.5%     97.5%
0.6815787 1.2214198
```

```
print(CI_alcohol)
```

```
     2.5%     97.5%
0.2216677 0.3303539
```

```
confint(lmW, level = 0.95)
```

```
                             2.5 %       97.5 %
(Intercept)          -1.960710e+01 63.537517843
fixed.acidity        -2.590639e-02  0.075887499
volatile.acidity     -1.321126e+00 -0.846054953
citric.acid          -4.712441e-01  0.106116245
residual.sugar       -1.309474e-02  0.045757280
chlorides            -2.696632e+00 -1.051817956
free.sulfur.dioxide   1.024314e-04  0.008620235
total.sulfur.dioxide -4.693951e-03 -0.001835208
density              -6.031362e+01 24.551294542
pH                   -7.894637e-01 -0.037842600
sulphates             6.920661e-01  1.140602768
alcohol               2.242512e-01  0.328144192
```

Question 4

Sulphates, free sulfur and Alcohol have a positive effect on quality while, sugar, fixed.acidity and density can have a positive or negative effect depending on what end of the CI they are on. The results between the bootstrap and the parametric are fairly similar with the bounds being slightly different, but the the variables with a positive effect are all the same in both Confidence Intervals.

Question 5

```
lm1 <- lm(quality ~ fixed.acidity + volatile.acidity + citric.acid + chlorides + free.sulfur.diox
lm1 %>%
  glance() %>%
  select(df, df.residual, statistic, p.value)
```

```
# A tibble: 1 × 4
     df df.residual statistic   p.value
  <dbl>       <int>     <dbl>     <dbl>
1     9        1589      99.3 3.31e-147
```

```
lmW %>%
  glance() %>%
  select(df, df.residual, statistic, p.value)
```

```
# A tibble: 1 × 4
     df df.residual statistic   p.value
  <dbl>       <int>     <dbl>     <dbl>
1    11        1587      81.3 1.79e-145
```

Ho the variables residual.sugar and density are 0 and have no effect on quality

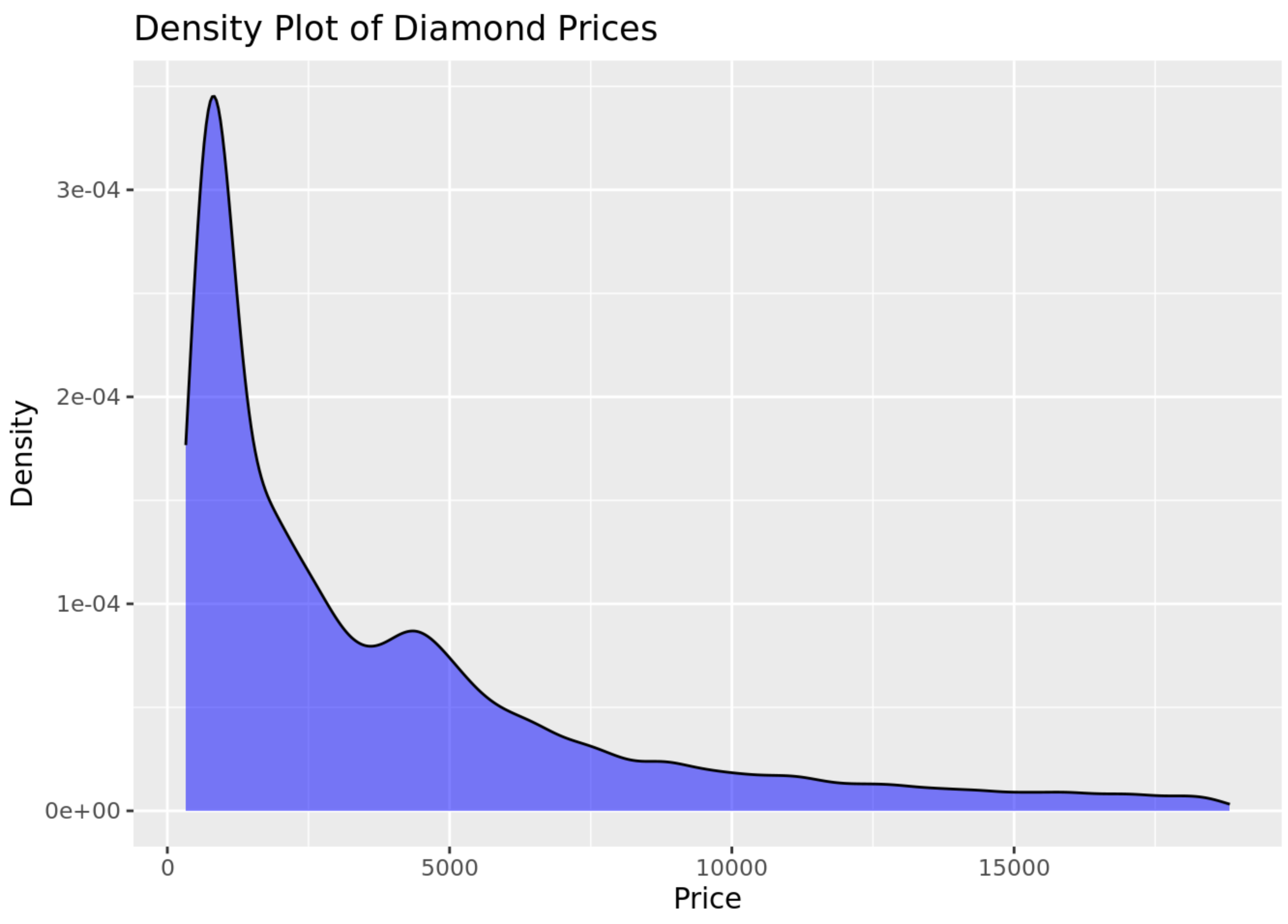
The model with residual.sugar and density has 2 more degrees of freedom and 2 more residuals, but is less accurate with a lower p.value and t-statistic. Therefore we should accept the null and not include these variables in the regression model.

Diagnostics

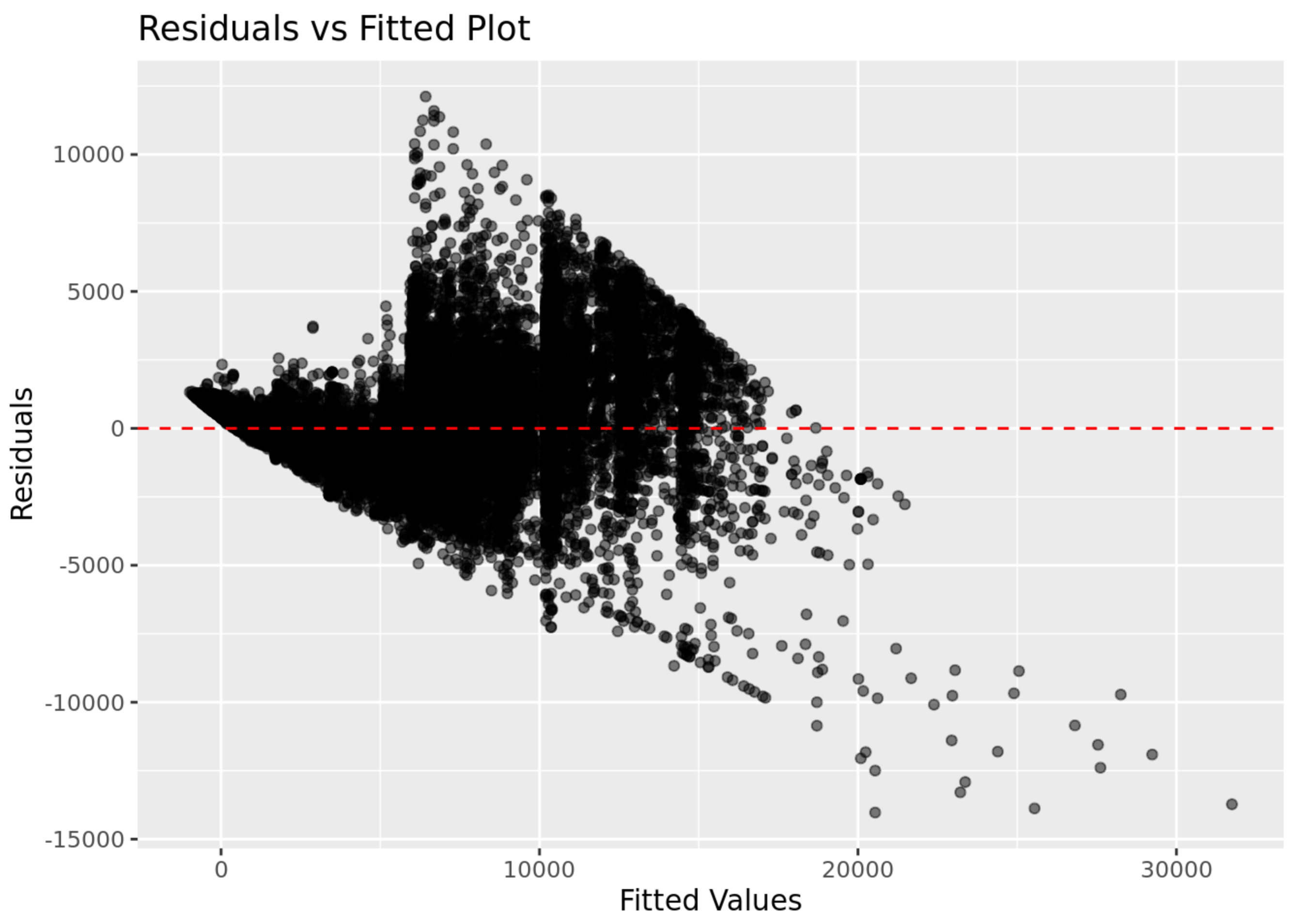
Question 1

```
diamonds$residuals <- residuals(lmint)
diamonds$fitted_values <- fitted(lmint)

ggplot(diamonds, aes(x = price)) +
  geom_density(fill = "blue", alpha = 0.5) +
  labs(x = "Price", y = "Density", title = "Density Plot of Diamond Prices")
```



```
ggplot(diamonds, aes(x = fitted_values, y = residuals)) +
  geom_point(alpha = 0.5) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  labs(x = "Fitted Values", y = "Residuals", title = "Residuals vs Fitted Plot")
```

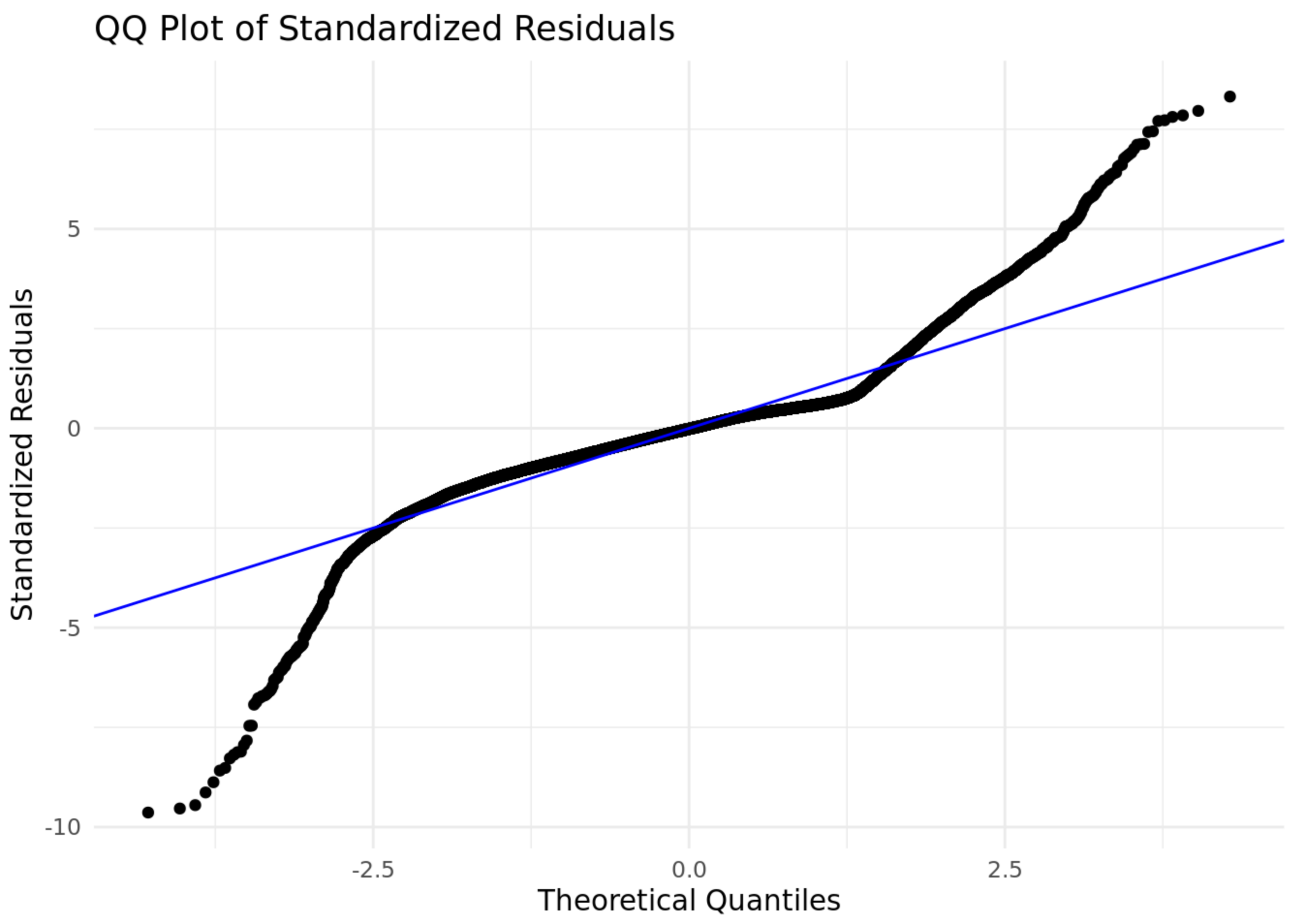


```
residuals_standardized <- rstandard(lmint)

qq_data <- qqnorm(residuals_standardized, plot.it = FALSE)

qq_df <- data.frame(Theoretical = qq_data$x, StandardizedResiduals = qq_data$y)

ggplot(qq_df, aes(x = Theoretical, y = StandardizedResiduals)) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0, color = "blue") +
  labs(title = "QQ Plot of Standardized Residuals",
       x = "Theoretical Quantiles",
       y = "Standardized Residuals") +
  theme_minimal()
```

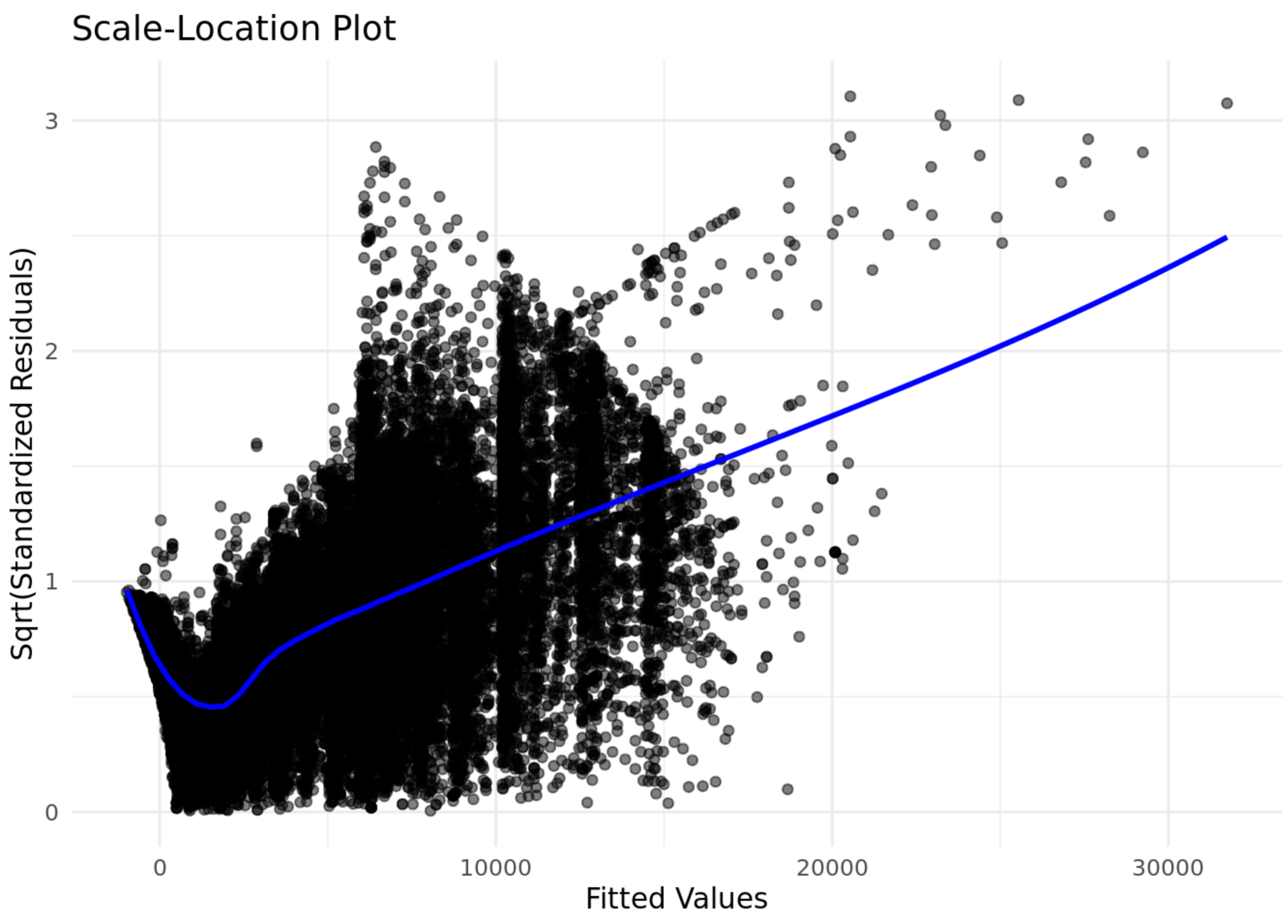


```r
sqrt_std_residuals <- sqrt(abs(rstandard(lmint)))

scale_location_df <- data.frame(
  FittedValues = fitted(lmint),
  SqrtStdResiduals = sqrt_std_residuals
)

ggplot(scale_location_df, aes(x = FittedValues, y = SqrtStdResiduals)) +
  geom_point(alpha = 0.5) +
  geom_smooth(method = "loess", color = "blue", se = FALSE) +
  labs(title = "Scale-Location Plot",
       x = "Fitted Values",
       y = "Sqrt(Standardized Residuals)") +
  theme_minimal()
```

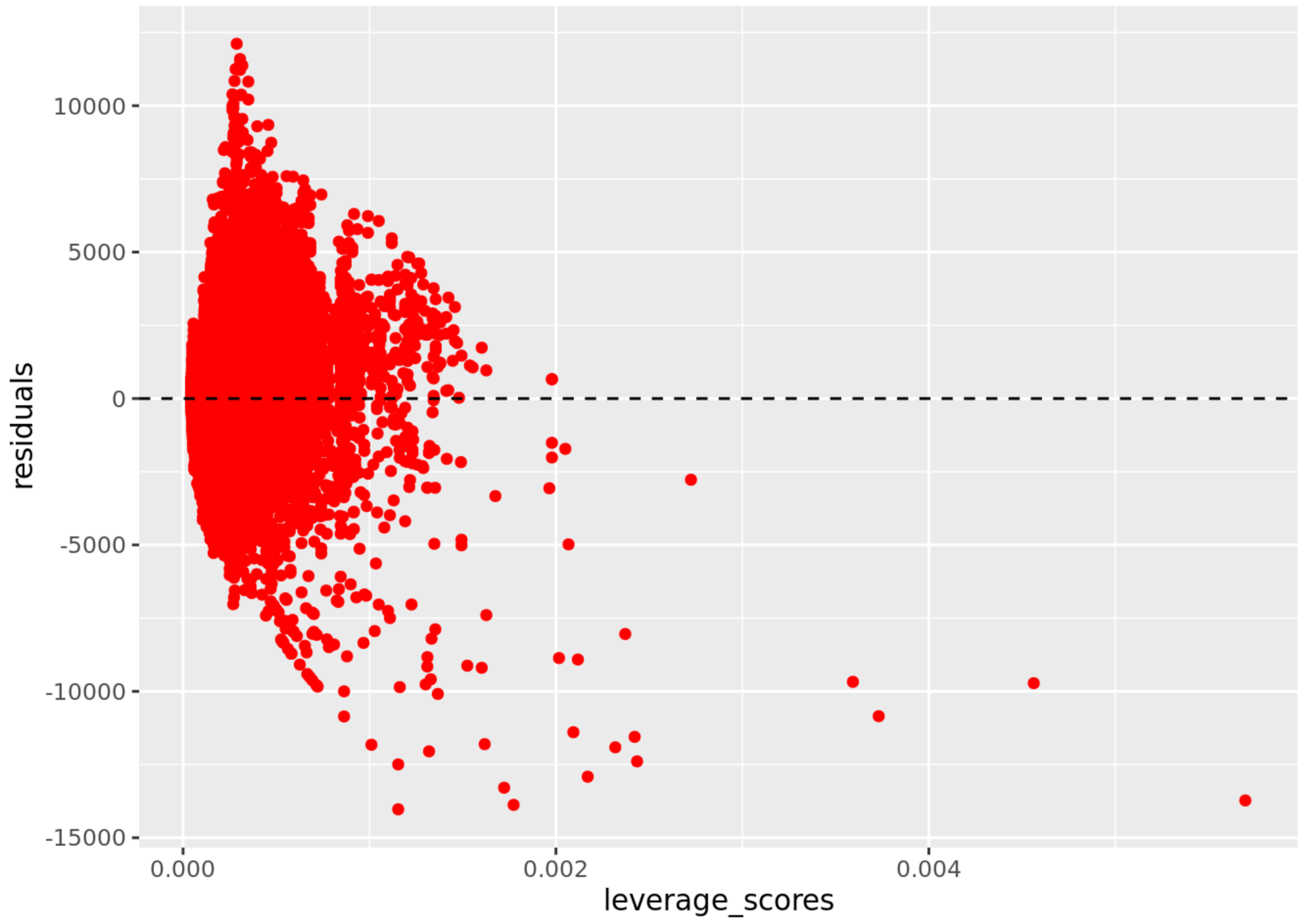
```
`geom_smooth()` using formula = 'y ~ x'
```



These three graphs show that this diamonds data set violates most assumptions, the residuals vs fitted graph shows that there is a ton of variance that isn't evenly distributed especially past 15000 violating homoscedasticity assumption. The QQ plot shows that the error terms are not normally distributed around the tails of the graph swinging wildly in an S like pattern. The scale location plot shows again the problem with uneven variance.

Question 2

```
leverage_scores <- hatvalues(lmint)
diamonds %>%
  ggplot(aes(x = leverage_scores, y = residuals)) +
  geom_point(color = "red") +
  geom_hline(yintercept = 0, linetype = "dashed")
```

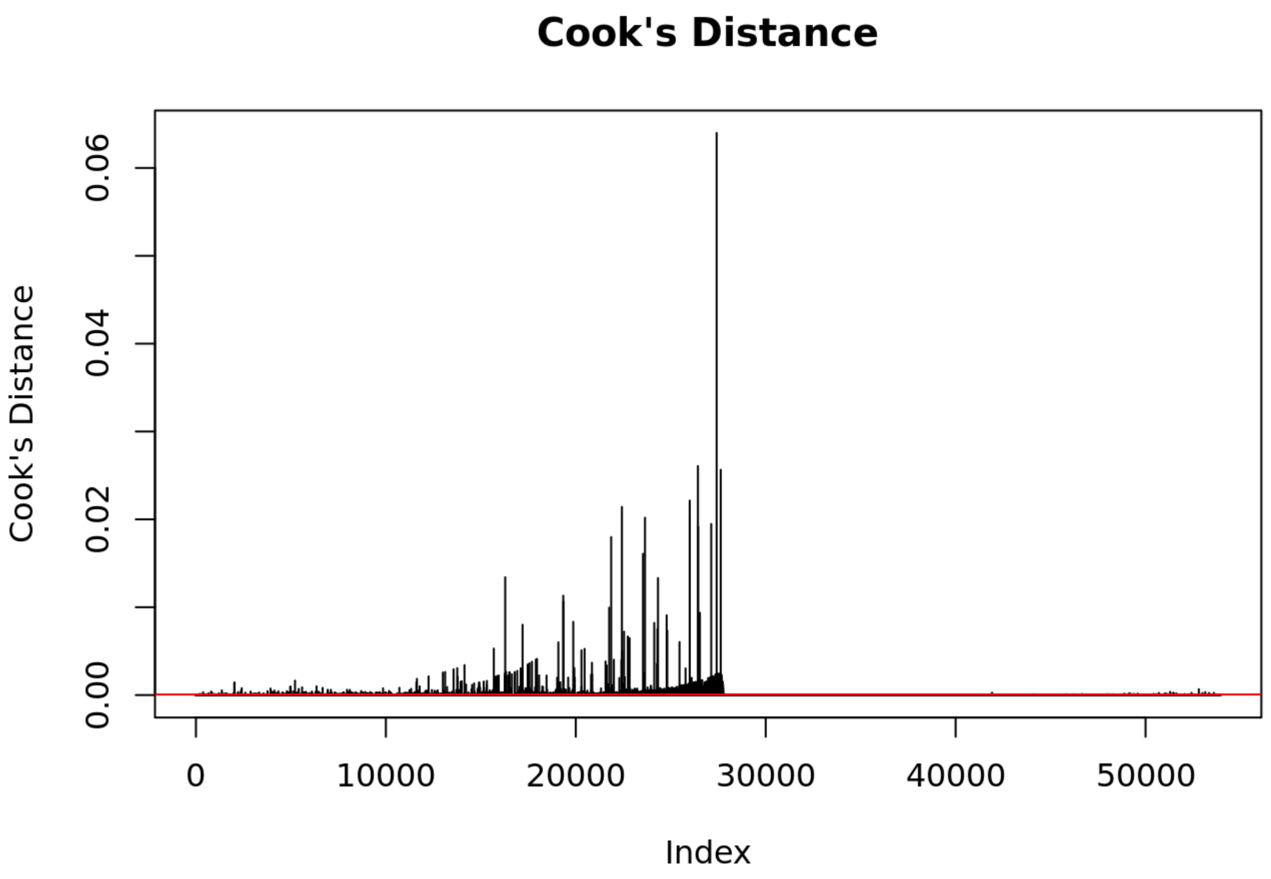


```
diamonds %>%
  mutate(leverages = hatvalues(lmint)) %>%
  filter(leverages > 0.002)
```

```
# A tibble: 15 × 19
   carat cut       color clarity depth table price length.in.mm width.of.mm
   <dbl> <fct>     <fct> <fct>   <dbl> <dbl> <dbl>        <dbl>       <dbl>
 1  3.11 Fair      J     I1       65.9    57  9823         9.15        9.02
 2  3.05 Premium   E     I1       60.9    58 10453         9.26        9.25
 3  3    Good      E     I1       64.2    65 11548         9.08        8.96
 4  2.75 Ideal     D     I1       60.9    57 13156         9.04        8.98
 5  2.58 Very Good D     SI2      58.9    63 14749         9.08        9.01
 6  4.01 Premium   I     I1       61      61 15223        10.1        10.1
 7  4.01 Premium   J     I1       62.5    62 15223        10.0         9.94
 8  3.4  Fair      D     I1       66.8    52 15964         9.42        9.34
 9  4    Very Good I     I1       63.3    58 15984        10.0         9.94
10  3.67 Premium   I     I1       62.4    56 16193         9.86        9.81
11  4.13 Fair      H     I1       64.8    61 17329        10           9.85
12  2.57 Premium   D     SI2      58.9    58 17924         8.99        8.94
13  5.01 Fair      J     I1       65.5    59 18018        10.7        10.5
14  4.5  Fair      J     I1       65.8    58 18531        10.2        10.2
15  3.51 Premium   J     VS2      62.5    59 18701         9.66        9.63
# ℹ 10 more variables: depth.in.mm <dbl>, carat_standardized <dbl>,
#   depth_standardized <dbl>, table_standardized <dbl>,
#   length_standardized <dbl>, width_standardized <dbl>,
#   depthmm_standardized <dbl>, residuals <dbl>, fitted_values <dbl>,
#   leverages <dbl>
```

```
cooks_distance <- cooks.distance(lmint)

plot(cooks_distance, type="h", main="Cook's Distance", ylab="Cook's Distance")
abline(h=4/(nrow(diamonds)), col="red")
```



```
threshold <- 4 / length(cooks_distance)


outliers <- cooks_distance > threshold

diamonds_cleaned <- diamonds[!outliers, ]
```

Based on the graph we will say that points above 0.002 have a high leverage, filtering by this we can see that diamonds with fair cut and J color have the highest leverage, so there could be something different about these diamonds that is causing a lot of error. A diamond with a fair cut and J color should typically be a lower price so something about these diamonds makes the much more expensive than a normal J color fair cut diamond.

Prediction

Question 1

```r
d3 <- diamonds %>% mutate(ln_width = log(width.of.mm),
             ln_price = log(price))
sum(is.infinite(d3$lnwidth))
```

```
Warning: Unknown or uninitialised column: `lnwidth`.
```

```
[1] 0
```

```r
sum(is.infinite(d3$lnprice))
```

```
Warning: Unknown or uninitialised column: `lnprice`.
```

```
[1] 0
```

```r
sum(is.nan(d3$ln_width))
```

```
[1] 0
```

```r
sum(is.nan(d3$ln_price))
```

```
[1] 0
```

```r
sum(is.na(d3$ln_width))
```

```
[1] 0
```

```r
sum(is.na(d3$ln_price))
```

```
[1] 0
```

```r
lm3 <-lm(log(price) ~ width.of.mm, data = diamonds)
tidy(lm3)
```

```
# A tibble: 2 × 5
  term        estimate std.error statistic p.value
  <chr>          <dbl>     <dbl>     <dbl>   <dbl>
1 (Intercept)    3.02    0.00786      384.       0
2 width.of.mm    0.832   0.00134      618.       0
```

I tried to take add the ln(price) and ln(width) to the diamonds data set then make a linear model of it but it kept giving me an error saying that Error in lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :

NA/NaN/Inf in ‘x’, so I’m just gonna interpret the regular regression and pretend its log log instead of log lin. For every 1% increase in width.of.mm there is a 0.83% increase in price holding all other variables constant.

Question 2

```
fake_data <- data.frame(width.of.mm = c(7,20))

predict(object = lm3, newdata = fake_data)
```

```
       1         2 
8.839234 19.651036 
```

```
predict(object = lm3, newdata = fake_data, interval = "prediction")
```

```
        fit       lwr      upr
1  8.839234  8.140099  9.53837
2 19.651036 18.950898 20.35117
```

Based on these intervals we can say with 95% confidence that the predicted price is between 8.14 and 9.53 for the first observation and 18.95 and 20.35 for the second.

Question 3

```
lm3 %>%
  glance() %>%
  select(r.squared)
```

```
# A tibble: 1 × 1
  r.squared
      <dbl>
1     0.876
```

The bounds for both prediction intervals are both about 1.4 so its hard to say, that one is more untrustworthy, we could fit an r-squared graph to see how accurate the prediction is when you add more variables, but we only have one variable and the starting r.squared it pretty high with 0.876 suggesting that most of the variability in price comes from width.of.mm.

Wine

Question 1

```
set.seed(1234)
train_pct <- .7
indices <- seq(from = 1, to = nrow(wine), by = 1)
```

```r
training_indices <- sample(x = indices, replace = FALSE,
                           size = nrow(wine)*train_pct)

wine_train <- wine %>%
  slice(training_indices)
wine_test <- wine %>%
  slice(-training_indices)
lmW1 <- lm(quality ~ alcohol, data = wine_train)
lmW2 <- lm(quality ~ alcohol + sulphates, data = wine_train)
lmW3 <- lm(quality ~ alcohol + sulphates + free.sulfur.dioxide , data = wine_train)
lmW4 <- lm(quality ~ alcohol + sulphates + free.sulfur.dioxide + fixed.acidity , data = wine_trai
lmW5 <- lm(quality ~ alcohol + sulphates + free.sulfur.dioxide + fixed.acidity + volatile.acidity
lmW6 <- lm(quality ~ alcohol + sulphates + free.sulfur.dioxide + fixed.acidity + volatile.acidity
lmW7 <- lm(quality ~ alcohol + sulphates + free.sulfur.dioxide + fixed.acidity + volatile.acidity
lmW8 <- lm(quality ~ alcohol + sulphates + free.sulfur.dioxide + fixed.acidity + volatile.acidity
lmW9 <- lm(quality ~ alcohol + sulphates + free.sulfur.dioxide + fixed.acidity + volatile.acidity
lmW10 <- lm(quality ~ alcohol + sulphates + free.sulfur.dioxide + fixed.acidity + volatile.acidit
predictions_1 <- predict(object = lmW1, newdata = wine_test)
predictions_2 <- predict(object = lmW2, newdata = wine_test)
predictions_3 <- predict(object = lmW3, newdata = wine_test)
predictions_4 <- predict(object = lmW4, newdata = wine_test)
predictions_5 <- predict(object = lmW5, newdata = wine_test)
predictions_6 <- predict(object = lmW6, newdata = wine_test)
predictions_7 <- predict(object = lmW7, newdata = wine_test)
predictions_8 <- predict(object = lmW8, newdata = wine_test)
predictions_9 <- predict(object = lmW9, newdata = wine_test)
predictions_10 <- predict(object = lmW10, newdata = wine_test)

best_k_model <-regsubsets(quality ~ ., wine)
summary(best_k_model)
```

```
Subset selection object
Call: regsubsets.formula(quality ~ ., wine)
11 Variables  (and intercept)
                     Forced in Forced out
fixed.acidity            FALSE      FALSE
volatile.acidity         FALSE      FALSE
citric.acid              FALSE      FALSE
residual.sugar           FALSE      FALSE
chlorides                FALSE      FALSE
free.sulfur.dioxide      FALSE      FALSE
total.sulfur.dioxide     FALSE      FALSE
density                  FALSE      FALSE
pH                       FALSE      FALSE
sulphates                FALSE      FALSE
alcohol                  FALSE      FALSE
1 subsets of each size up to 8
Selection Algorithm: exhaustive
         fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
```

```
1  ( 1 ) " "           " "              " "         " "            " "
2  ( 1 ) " "           "*"              " "         " "            " "
3  ( 1 ) " "           "*"              " "         " "            " "
4  ( 1 ) " "           "*"              " "         " "            " "
5  ( 1 ) " "           "*"              " "         " "            "*"
6  ( 1 ) " "           "*"              " "         " "            "*"
7  ( 1 ) " "           "*"              " "         " "            "*"
8  ( 1 ) " "           "*"              "*"         " "            "*"
         free.sulfur.dioxide total.sulfur.dioxide density pH  sulphates alcohol
1  ( 1 ) " "                 " "                  " "     " " " "       "*"
2  ( 1 ) " "                 " "                  " "     " " " "       "*"
3  ( 1 ) " "                 " "                  " "     " " "*"       "*"
4  ( 1 ) " "                 "*"                  " "     " " "*"       "*"
5  ( 1 ) " "                 "*"                  " "     " " "*"       "*"
6  ( 1 ) " "                 "*"                  " "     "*" "*"       "*"
7  ( 1 ) "*"                 "*"                  " "     "*" "*"       "*"
8  ( 1 ) "*"                 "*"                  " "     "*" "*"       "*"
```

Question 2

```r
k <- 10
fold_vector <-
  cut(1:nrow(wine), breaks = k, labels = FALSE)

random_folds <- sample(x = fold_vector, size = nrow(wine),
                       replace = FALSE)

wine <-wine %>%
  mutate(folds = random_folds)
ONE_CV_FOLD <- function(fold_number, formula_string)
{
  wine_train <- wine %>%
  filter(folds != fold_number)

  wine_test <- wine %>%
  filter(folds == fold_number)

  cv_linear_model <-
    do.call(what = "lm",
        args = list(formula = as.formula(formula_string), data = quote(wine_train)))

  cv_predictions <- predict(object= cv_linear_model, newdata = wine_test)

  observations <- wine_test %>%
  select(quality) %>%
  pull()

  MSE <- mean((cv_predictions-observations)^2)

  return(MSE)
}
MSEs1 <- sapply(unique(fold_vector), FUN = ONE_CV_FOLD,
               formula_string = "quality~ alcohol")
mean(MSEs1)
```

```
[1] 0.5057258
```

```
MSEs2 <- sapply(unique(fold_vector), FUN = ONE_CV_FOLD,
               formula_string = "quality~ alcohol + volatile.acidity")
mean(MSEs2)
```

```
[1] 0.4470504
```

```
MSEs3 <- sapply(unique(fold_vector), FUN = ONE_CV_FOLD,
               formula_string = "quality~ alcohol + volatile.acidity + sulphates")
mean(MSEs3)
```

```
[1] 0.4351985
```

```
MSEs4 <- sapply(unique(fold_vector), FUN = ONE_CV_FOLD,
               formula_string = "quality~ alcohol + volatile.acidity + sulphates + total.sulfur.di
mean(MSEs4)
```

```
[1] 0.4308383
```

```
MSEs5 <- sapply(unique(fold_vector), FUN = ONE_CV_FOLD,
               formula_string = "quality~ alcohol + volatile.acidity + sulphates + total.sulfur.di
mean(MSEs5)
```

```
[1] 0.4265202
```

```
MSEs6 <- sapply(unique(fold_vector), FUN = ONE_CV_FOLD,
               formula_string = "quality~ alcohol + volatile.acidity + sulphates + pH + chlorides
mean(MSEs6)
```

```
[1] 0.4231016
```

```
MSEs7 <- sapply(unique(fold_vector), FUN = ONE_CV_FOLD,
               formula_string = "quality~ alcohol + volatile.acidity + sulphates + total.sulfur.di
mean(MSEs7)
```

```
[1] 0.4231016
```

```
MSEs8 <- sapply(unique(fold_vector), FUN = ONE_CV_FOLD,
               formula_string = "quality~ alcohol + volatile.acidity + sulphates + total.sulfur.d
mean(MSEs8)
```

```
[1] 0.4222518
```

```
MSEs9 <- sapply(unique(fold_vector), FUN = ONE_CV_FOLD,
               formula_string = "quality~ alcohol + volatile.acidity + sulphates + total.sulfur.d
mean(MSEs9)
```

```
[1] 0.4222533
```

```
MSEs10 <- sapply(unique(fold_vector), FUN = ONE_CV_FOLD,
               formula_string = "quality~ alcohol + volatile.acidity + sulphates + total.sulfur.d
mean(MSEs10)
```

```
[1] 0.422882
```

Based on the MSEs decreasing with every variable added I would say the first model with only alcohol is the “best” model