# Linear Regression

Silvie Cinková

2025-08-07

## Table of contents

# 1 Diamonds

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v readr     2.1.5
v forcats   1.0.0     v stringr   1.5.1
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
v purrr     1.0.4
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to beco
```

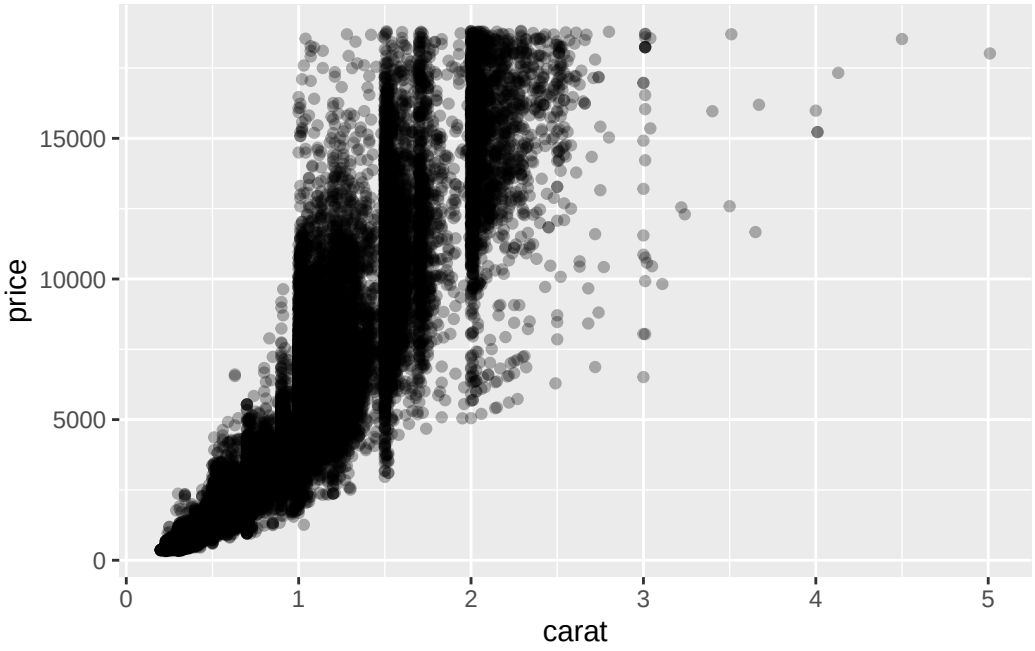
```
library(ggfortify)
diamonds <- ggplot2::diamonds
colnames(diamonds)
```

```
[1] "carat"   "cut"     "color"   "clarity" "depth"   "table"   "price"
[8] "x"       "y"       "z"
```

## 2 Carat vs. price

How much does a diamond's weight in carats affect its price?

```
ggplot(diamonds, aes(x = carat, y = price)) + geom_point(alpha = 0.3)
```



## 3 Correlation

```
cor <- cor.test(x = diamonds$carat, y = diamonds$price, method = "pearson")
 ↪  #default
cor
```

```
    Pearson's product-moment correlation

data:  diamonds$carat and diamonds$price
t = 551.41, df = 53938, p-value < 2.2e-16
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.9203098 0.9228530
sample estimates:
      cor
0.9215913
```

There are several methods to compute correlation. The most common correlation is Pearson's correlation, the default method in the `cor.test` function.

`p-value < 2.2e-16` (R's equivalent of effectively zero): the smaller, the more statistically significant the result. You decide where you put the threshold. A usual threshold in social sciences and humanities is 0.05.
`cor`: the direction and strength of the correlation. Positive: the more X, the more Y. Negative: the more X, the less Y. Range: -1 (perfect negative correlation) - +1 (perfect positive correlation). Zero: no correlation. Correlation tests detect only a linear association between variables. When two variables have a very tight "curvy" mutual association, with correlation you fail to detect it.
With the absolute `cor` value near 1, you only know that the mutual association is strong, but you are not able to tell how much Y changes when X changes.

## 4 Extract values from `cor.test`

```
str(cor)
```

```
List of 9
 $ statistic  : Named num 551
  ..- attr(*, "names")= chr "t"
 $ parameter  : Named int 53938
```

```
  ..- attr(*, "names")= chr "df"
 $ p.value    : num 0
 $ estimate   : Named num 0.922
  ..- attr(*, "names")= chr "cor"
 $ null.value : Named num 0
  ..- attr(*, "names")= chr "correlation"
 $ alternative: chr "two.sided"
 $ method     : chr "Pearson's product-moment correlation"
 $ data.name  : chr "diamonds$carat and diamonds$price"
 $ conf.int   : num [1:2] 0.92 0.923
  ..- attr(*, "conf.level")= num 0.95
 - attr(*, "class")= chr "htest"
```

`cor.test` returns two outputs. When you do not save it in an object, you will see only the message on the previous slide. But when you do save it, you will have the same information structured in a named list.

## 5 The `broom` library

- convenience library for statistical tests
- extracts values from (many) common test functions
- as data frames
- functions `tidy`, `augment`, and `glance`

```
cor %>% broom::tidy()
```

```
# A tibble: 1 x 8
  estimate statistic p.value parameter conf.low conf.high method     alternative
     <dbl>     <dbl>   <dbl>     <int>    <dbl>     <dbl> <chr>      <chr>
1    0.922      551.       0     53938    0.920     0.923 Pearson's~ two.sided
```

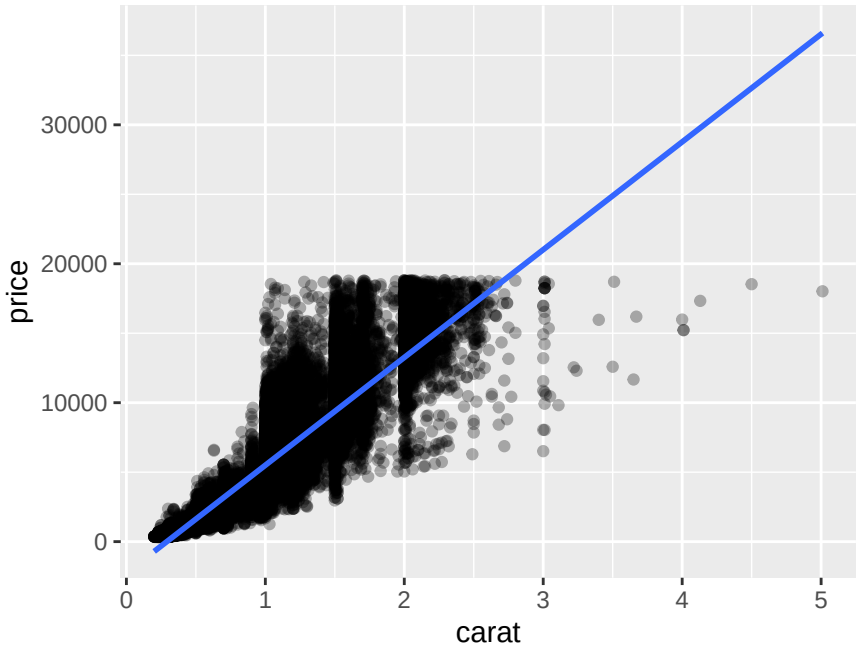
Now you know for sure that there is a strong relationship between the two variables. But `cor.test` will not tell you how much Y grows when X grows. You might want to know this to estimate the price of a diamond you have not seen in your data. This is called prediction. For predictions, you need a **regression model**. We are going to talk through the simplest one, the linear regression model. You can use the linear model when your response variable (the one you imagine on the Y-axis of a plot) is continuous (must be numeric).
You have to use other models when your response variable is categorical. These are called logistic regression models. There is a special logistic regression model for a binary response variable (e.g. TRUE, FALSE): so-called **binomial model** and yet another one when the

categories are more than two ( **multinomial model**). Back to our simple linear regression models.

## 6 `carat` vs. `price`

- linear model with `ggplot2`

```
diamonds_plot <- diamonds %>% ggplot(aes(x = carat, y = price)) +
  geom_point(alpha = 0.3) +
  geom_smooth(method = "lm", se = TRUE, formula = y ~ x) +
  coord_fixed(ratio = 1/10000)
diamonds_plot
```




`ggplot2` generates a linear model (blue line) with `geom_smooth`, `method = "lm"`. To construct the line, it must have computed the intercept (where the line crosses the Y-axis) and the slope coefficient (how much tilt). The algorithm behind picked the intercept and the slope coefficient so that the line goes as close to all data points as possible. The distance of each point from the line is measured as the square of the difference between the observed value and the line. This method is also called *Least squares*, because you can imagine that the algorithm draws a line and then draws draws squares from the line to the point and measures their total surface.

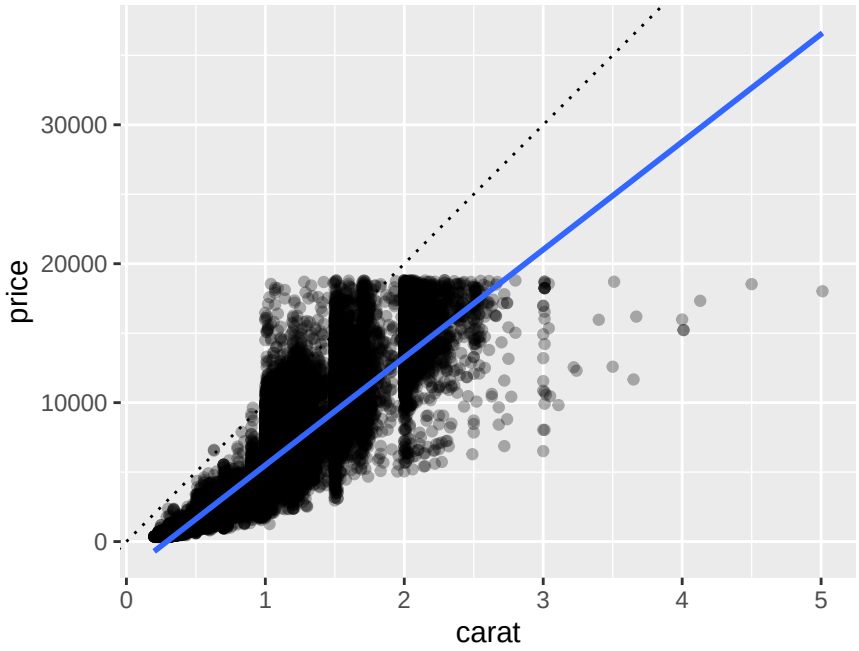
It repeats drawing lines and totaling the square distances to all points until it finds the line with the smallest total surface of these squares.

To this plot, I have added `coord_fixed()` with `ratio = 10000/1`. The distance between ticks becomes equally long on both axes (ten thousand dollars on the Y-axis equally long as one carat on the X-axis), which makes the visual estimation of the slope easier.

## 7 Slope and Intercept in detail

```
diamonds_plot +
  geom_abline(slope = 10000, intercept = 0, linetype = 3) +
  coord_fixed(ratio = 1/10000)
```

```
Coordinate system already present. Adding new coordinate system, which will
replace the existing one.
```




The slope is the amount by which Y changes when X raises. If each carat increased the price by ten thousand dollars, the slope would look like the **dotted** line in this plot and you could calculate the price from carats like this: `y = x * 10000 + 0`. The zero after `+` means that when your diamond weights exactly 0 carats, you pay exactly 0 dollars. In terms of the plot,

the zero is the value on the Y-axis that you get for a zero on the X-axis. The point where the line cuts the Y-axis is called **Intercept**. As you can see, the line of our linear model has a different slope and a different intercept. When the intercept is not zero, you can hardly estimate the slope visually. To calculate the slope of a line, we need the x and y values of two points. The slope would be the difference of the y values divided by the difference of the x values, always subtracting the point more to the left (smaller x) from the point more to the right (larger x). Mnemonic: **Rise (Y) over Run (X)** (difference of Y above, difference of X below in the fraction). Let us visually estimate these two points. For instance, a one-carat diamond would cost about $6,000 and a 3-carat diamond about $21,000. So it would be `(21000 - 6000)/(3-1) = 7500`. This means that each one-carat increase would correspond to a $7,500 increase.

## 8 Slope and intercept computed

```
lm_carat_price <- lm(formula = price ~ carat , data = diamonds)
lm_carat_price
```

```
Call:
lm(formula = price ~ carat, data = diamonds)

Coefficients:
(Intercept)        carat
      -2256         7756
```

`stats::lm` - in most installations the `stats` library loads automatically whenever you launch R. Formula: response variable (Y-axis variable) ~ explanatory variable(s) (X-axis and possibly other variables except Y)

Look at the result: the slope coefficient of `carat` is 7,756. Our visual estimation resulted in 7,500. That was rather close!

## 9 `lm` detail

```
lm_carat_price %>% summary()
```

```
Call:
lm(formula = price ~ carat, data = diamonds)
```

```
Residuals:
     Min       1Q   Median       3Q      Max
-18585.3   -804.8    -18.9    537.4  12731.7

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -2256.36      13.06  -172.8   <2e-16 ***
carat        7756.43      14.07   551.4   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1549 on 53938 degrees of freedom
Multiple R-squared:  0.8493,    Adjusted R-squared:  0.8493
F-statistic: 3.041e+05 on 1 and 53938 DF,  p-value: < 2.2e-16
```

Start investigating the report from the bottom up. 1. Look at the p-value at the bottom: It is tiny, so the model is statistically significant (it has enough data to rely on). 2. Look at the (adjusted) R-squared: This is the proportion of data variance explained by the model. This model explains 84.93% of the data variance with this single explanatory variable. So `carat` is an extremely strong *predictor* of `price`.
3. Look at the *Residual standard error*: it says how much wrong the model typically is, in the same units as the response variable. So here we learn that the predicted price is typically wrong by $1549. More about "residual" later.

Now look at the Coefficients section. It lists the Intercept and `carat`, which is our only explanatory variable. The first column is Estimate. This contains the values of intercept and slope coefficient. The intercept value at -2256.36 appears to suggest that the when you enter the jewelry store without buying any of these diamonds, they will pay you $2256.36! This artifact is easy to interpret: in fact, the relationship between `carat` and `price` is not perfectly linear. In smaller diamonds (below one carat), the price does not increase as steeply as in the heavier diamonds. At the same time, there are enough heavier diamonds with that steep price increase trend that the model must take it into account and the line becomes funny with smaller diamonds.

The Error column captures the standard error of each coefficients. Imagine hundreds of such large diamond collections, gathered by the same methods. The standard error makes a qualified guess how much the coefficient's value would fluctuate in this sample of samples. Here it says that in the different collections the carat coefficients would probably differ by $14.07.

The last column shows you the statistical significance of the coefficient value (that is, was there enough data to tell?) Below the table is a legend showing at which p-values the `lm` function sets significance thresholds. Three asterisks means highly significant, two less significant, etc. But again, you decide where to put your threshold. Just mind to set it (mentally) before

you run the model. Setting a (higher) significance threshold ex post to make your results significant amounts to academic cheating!!!

The p-value column is actually called `Pr(>|t|)`. This means the p-value of a t-test, and it is associated with the values in the column called `t value`. In this context, the t-test gives a p-value for how likely it is to observe such a t-value (and more extreme) if the true effect were zero and the variable could be deleted without affecting the values of the response variable. Here the probability that the effect of `carat` being zero (and hence `carat` a superfluous, irrelevant variable) is effectively zero (that is, it is sure that `carat` has some effect), and therefore it is marked with three asterisks.

Residuals: the model generates a predicted value for each observed value. The predicted values form the regression line. The residual is the difference between the observed value and the predicted value (observed minus predicted). The Residuals section shows their five-number summary. They ought to be (close to) normally distributed (Gaussian distribution), with median being as close to the mean as possible and close to zero (because we ideally want as little error as possible - cf. *least squares*!) .

## 10 Display Coefficients with `broom::tidy`

- sometimes masked by other packages
- call explicitly `broom::tidy`

```
lm_carat_price %>% broom::tidy()
```

```
# A tibble: 2 x 5
  term        estimate std.error statistic p.value
  <chr>          <dbl>     <dbl>     <dbl>   <dbl>
1 (Intercept)   -2256.      13.1     -173.       0
2 carat          7756.      14.1      551.       0
```

## 11 Observed vs. predicted values

- model predicts a value for each observed data point
- predicted values form the regression line
- `residual = observed value - predicted value`

## 12 Model values: observed, fitted

```
values_lm_carat_price <- lm_carat_price %>% broom::augment()
glimpse(values_lm_carat_price)
```

```
Rows: 53,940
Columns: 8
$ price      <int> 326, 326, 327, 334, 335, 336, 336, 337, 337, 338, 339, 340,~
$ carat      <dbl> 0.23, 0.21, 0.23, 0.29, 0.31, 0.24, 0.24, 0.26, 0.22, 0.23,~
$ .fitted    <dbl> -472.382688, -627.511200, -472.382688, -6.997151, 148.13136~
$ .resid     <dbl> 798.3827, 953.5112, 799.3827, 340.9972, 186.8686, 730.8184,~
$ .hat       <dbl> 4.515399e-05, 4.706148e-05, 4.515399e-05, 3.982758e-05, 3.8~
$ .sigma     <dbl> 1548.572, 1548.571, 1548.572, 1548.576, 1548.576, 1548.573,~
$ .cooksd    <dbl> 6.001647e-06, 8.922178e-06, 6.016690e-06, 9.656791e-07, 2.7~
$ .std.resid <dbl> 0.51557559, 0.61575429, 0.51622136, 0.22020685, 0.12067468,~
```

## 13 `.resid` (Residuals)

- `.resid` = `.fitted` minus `price`

```
values_lm_carat_price %>% select(price, .fitted, .resid) %>%
  slice_head(n = 10) %>% mutate(my_residuals = price - .fitted)
```

```
# A tibble: 10 x 4
   price .fitted .resid my_residuals
   <int>   <dbl>  <dbl>        <dbl>
 1   326 -472.     798.         798.
 2   326 -628.     954.         954.
 3   327 -472.     799.         799.
 4   334   -7.00   341.         341.
 5   335  148.     187.         187.
 6   336 -395.     731.         731.
 7   336 -395.     731.         731.
 8   337 -240.     577.         577.
 9   337 -550.     887.         887.
10   338 -472.     810.         810.
```

```
lm_carat_price %>% broom::glance()
```

```
# A tibble: 1 x 12
  r.squared adj.r.squared sigma statistic p.value    df   logLik     AIC     BIC
      <dbl>         <dbl> <dbl>     <dbl>   <dbl> <dbl>    <dbl>   <dbl>   <dbl>
1     0.849         0.849 1549.   304051.       0     1 -472730. 945467. 945493.
# i 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

sigma = Residual Standard Error (you saw it in in the summary).

## 14 Is my model reliable?

- standardized residuals ought to be *roughly* normally distributed
- Normality tests can be too strict
  - `shapiro.test` (3 - 5,000 data points)
  - `ks.test` for larger data sets
- Use plots with ggfortify and autoplot

## 15 Shapiro test on standardized residuals

- returns probability of your distribution being normal
- make a random sample of max 4,999 data points

```
values_lm_carat_price$.std.resid %>%
  sample(size = 4999) %>%
  shapiro.test()
```

```
    Shapiro-Wilk normality test

data:  .
W = 0.8932, p-value < 2.2e-16
```

## 16 Kolmogorov-Smirnov on standardized residuals

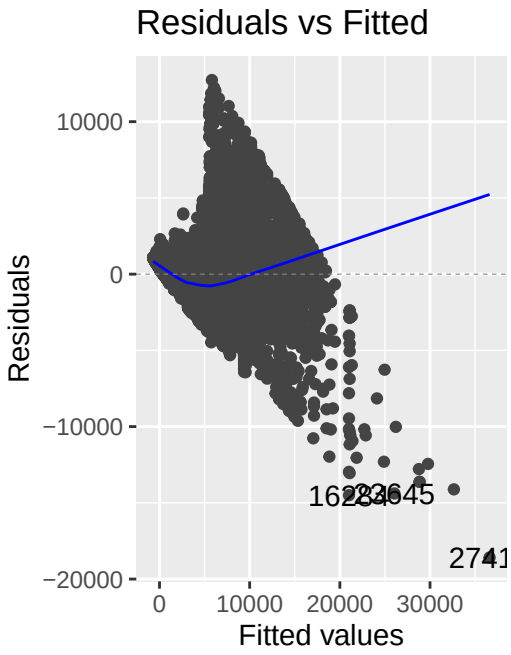
```
values_lm_carat_price$.std.resid %>%
  ks.test(y = "pnorm", sd = sd(.), mean = mean(.))
```

```
Warning in ks.test.default(., y = "pnorm", sd = sd(.), mean = mean(.)): ties
should not be present for the one-sample Kolmogorov-Smirnov test


    Asymptotic one-sample Kolmogorov-Smirnov test

data:  .
D = 0.15254, p-value < 2.2e-16
alternative hypothesis: two-sided
```

So the distribution of standardized residuals is not normal, but let's check the plots

## 16.1 `ggfortify` and `autoplot`

```
autoplot(lm_carat_price, which = 1)
```




The scatterplot checks two key assumptions: 1. Linearity: Relationship between predictors and response is linear. 2. Homoscedasticity: Residual variance is constant across fitted values. A good model would show a random scatter of points around the horizontal line at 0, without clear patterns, curves, or funnels, but with a roughly equal spread across the range of fitted values. Our model is actually not very good, even though it explains so much data variation:

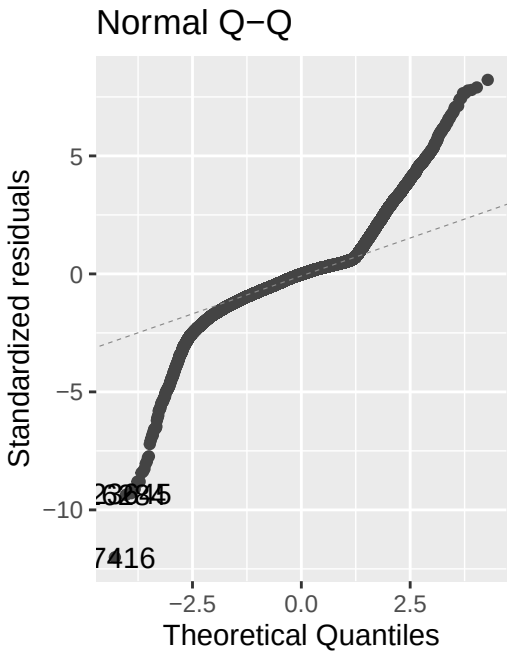
- It shows a funnel shape: The higher the fitted values, the greater the spread. - It has many outliers far from 0.

The blue curve is a loess (similar to gam) curve of the trend in residuals across the fitted values. When it lies flat close to zero, the residuals stay small. In our case: In predictions between roughly $2,000 and $10,000 it the residuals tend to be slightly negative. Because residuals are computed as observed minus fitted, these residuals are negative because the model guessed the price higher than the actual (observed) price. This error is not so big, but look at the most expensive diamonds: the curve goes steeply up. This means that the model seriously underestimates the prices of the expensive diamonds.

## 17 Quantile-quantile plot

- Standardized residuals ought to lie on the dotted line.

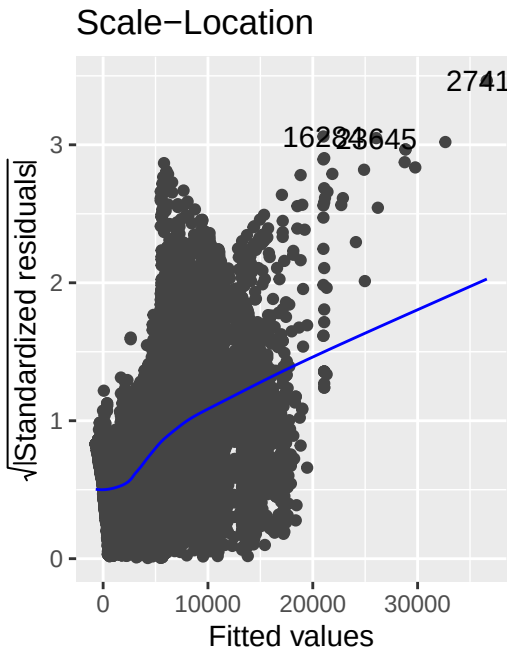
```
lm_carat_price %>% autoplot(which  = 2)
```




The model seems to perform reasonably well on the middle-prized diamonds, but fails in the tails. This is very common. On the other hand, the tails are very densely populated here, so it is not little data that we model incorrectly.

## 18 Scale-Location plot

```
lm_carat_price %>% autoplot(which = 3)
```

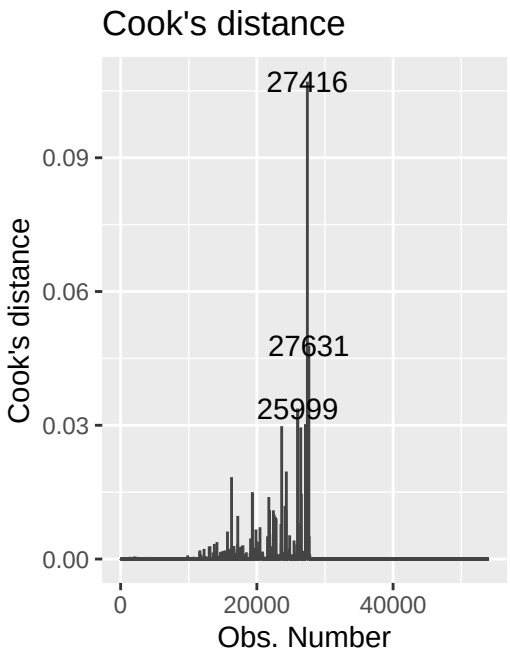



This plot mainly checks homoscedasticity (equal variance of residuals). The curve just shows the trend of the points. Again, the distribution should be flat horizontal around zero. Ours is curved upward. This is very common and indicates that the variance of the residuals increases with fitted values (heteroscedasticity).

## 19 Cook's distance

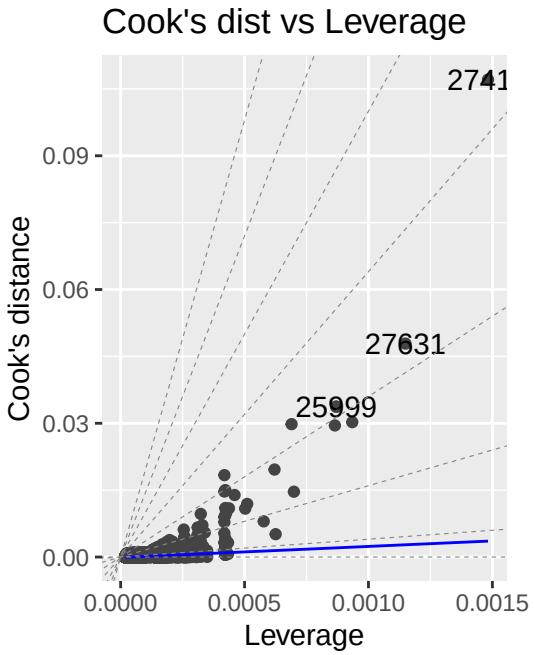
- distance = influence of the observation on the model
- combines *leverage* (how far predictor from mean) and *residual size* (guessing error)
- observations worth investigating individually

```
lm_carat_price %>% autoplot(which = 4)
```




## 20

```
lm_carat_price %>% autoplot(which = 6)
```




This plot shows which points had high Cook's distance mainly due to their leverage. It looks as if all of them had. They were perhaps not excessively poorly fitted (worth investigating).

## 21 Potential improvements

- add another predictor (often a categorical one is very helpful)
- log-transform the response variable (remedies heteroscedasticity)
- add a nonlinear term (e.g. x square)

## 22 Add a predictor

```
lm_carat_cut_price <- lm(formula = price ~ carat + cut, data = diamonds)
broom::glance(lm_carat_cut_price) %>% select(c(1:3, 6, 8, 9))
```

```
# A tibble: 1 x 6
  r.squared adj.r.squared sigma    df      AIC      BIC
      <dbl>         <dbl> <dbl> <dbl>    <dbl>    <dbl>
1     0.856         0.856 1511.     5 942854. 942916.
```

Just `price ~ carat`

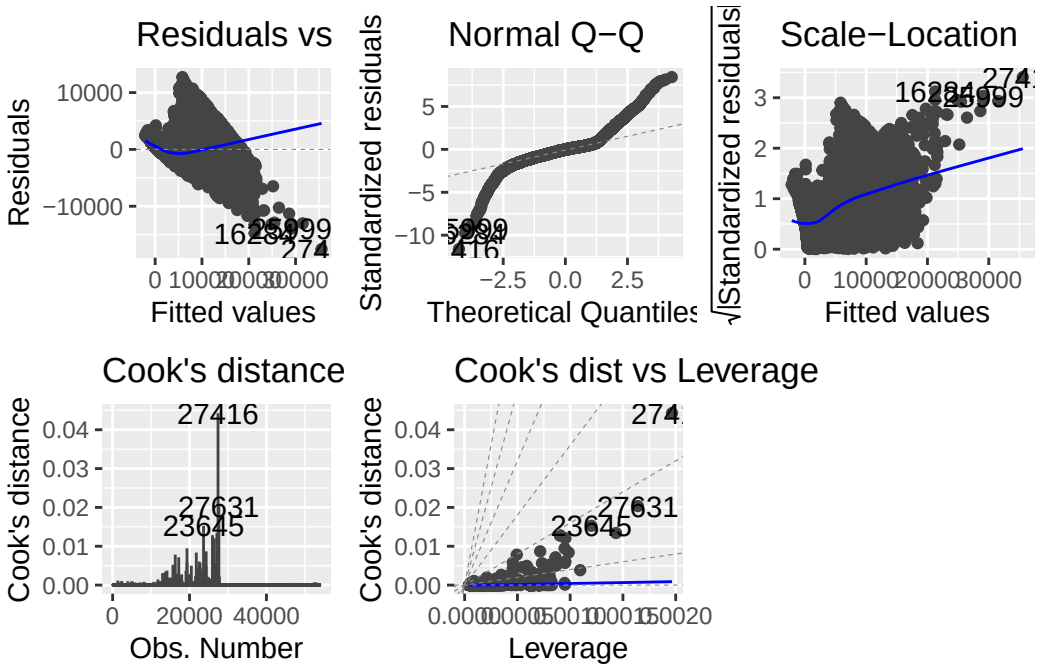
```
broom::glance(lm_carat_price)  %>% select(c(1:3, 6, 8, 9))
```

```
# A tibble: 1 x 6
  r.squared adj.r.squared sigma    df      AIC      BIC
      <dbl>         <dbl> <dbl> <dbl>    <dbl>    <dbl>
1     0.849         0.849 1549.     1 945467. 945493.
```

The richer model explains slightly more variance (r.squared) and is slightly more precise (sigma is smaller). At the same time, AIC and BIC improved (decreased). Lower AIC or BIC mean that the model is better for prediction than the old model. These scores signal risk of overfitting.

## 23 Plot diagnostics for the enriched model

```
autoplot(lm_carat_cut_price, which = c(1,2,3,4,6), ncol = 3)
```



Not better. We should possibly try the log transformation of the response variable. Note that neither sigma, nor AIC and BIC will be comparable with models without the log transformation.

## 24 Log transformation of the response variable

```
lm_log_carat_price <- lm(formula = log(price) ~ (carat), data = diamonds)
broom::glance(lm_log_carat_price) %>% select(c(1:3, 5, 8:9))
```

```
# A tibble: 1 x 6
  r.squared adj.r.squared sigma p.value    AIC    BIC
      <dbl>         <dbl> <dbl>   <dbl>  <dbl>  <dbl>
1     0.847         0.847 0.397       0 53464. 53491.
```

```
broom::glance(lm_log_carat_price) %>% select(c(1:3, 5, 8:9))
```

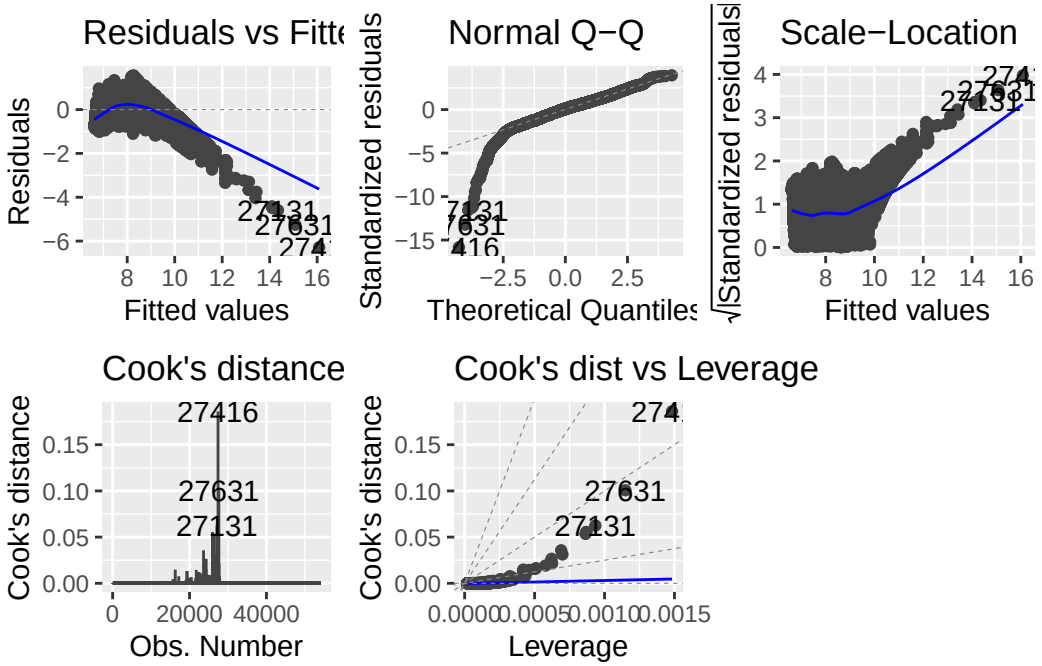
```
# A tibble: 1 x 6
  r.squared adj.r.squared sigma p.value    AIC    BIC
      <dbl>         <dbl> <dbl>   <dbl>  <dbl>  <dbl>
1     0.847         0.847 0.397       0 53464. 53491.
```

sligthly weaker (lower R2) than without the transformation

## 25 log-transformed response

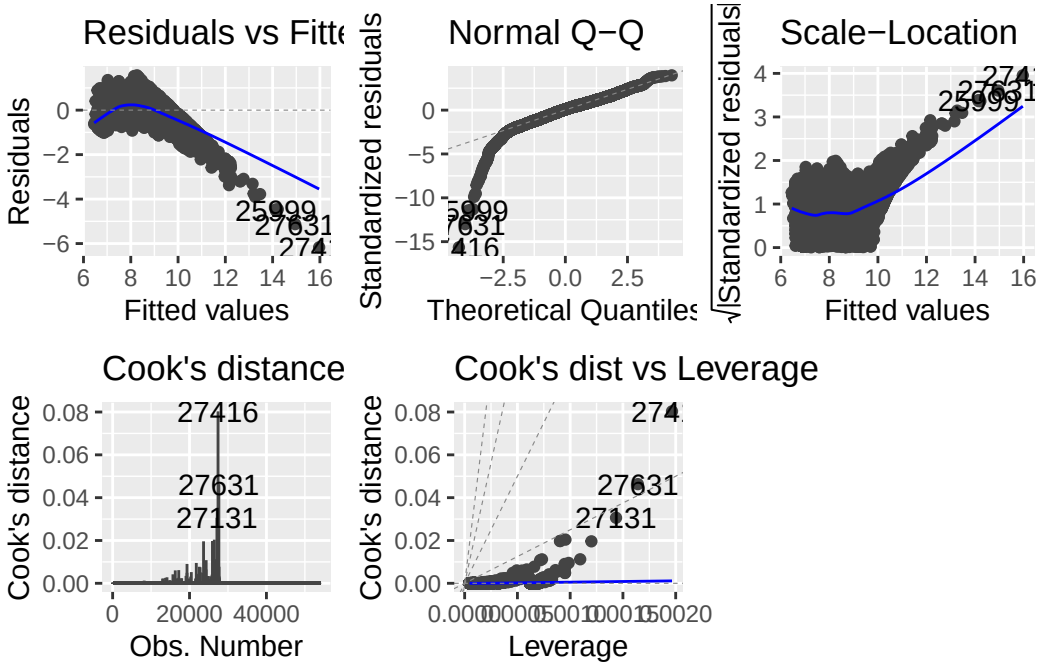
- better QQ fit

```
autoplot(lm_log_carat_price, which = c(1,2,3,4,6), ncol = 3)
```




## 26 log-transformed response + cut

```
lm_log_carat_cut_price <- lm(formula = log(price)~ carat + cut, data =
↪  diamonds)
autoplot(lm_log_carat_cut_price, which = c(1,2, 3,4,6), ncol = 3)
```

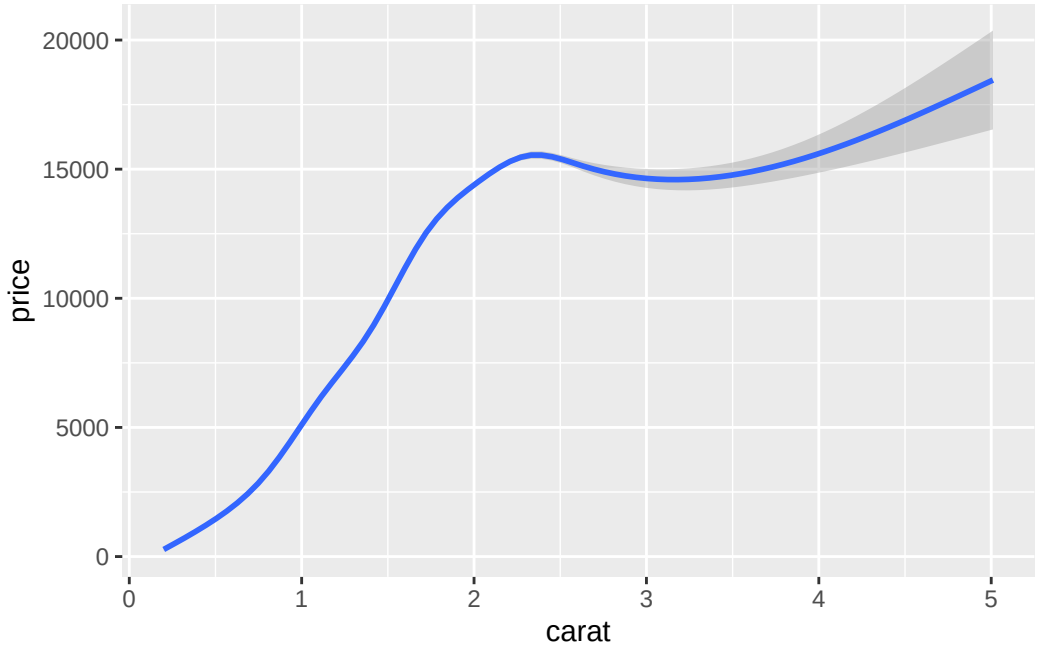



adding the predictor `cut` improves AIC and BIC, but does not resolve the linearity or heteroscedasticity problem.

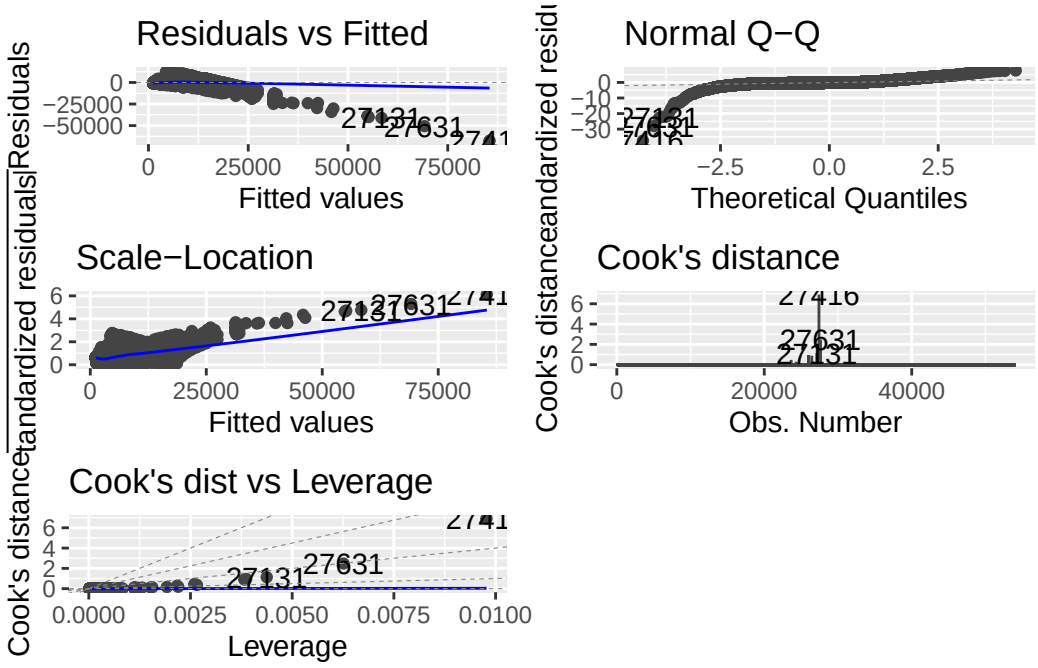
## 27 Add a non-linear term

this curve looks like $x^2$, right?

```
diamonds %>% ggplot(aes(x = carat, y = price)) +
  geom_smooth(method = "gam")
```




```
lm_carat_price_x2 <- lm(formula = price ~ I(carat^2), data = diamonds)
autoplot(lm_carat_price_x2, which = c(1,2,3,4,6))
```



```
    broom::glance(lm_carat_price_x2) %>% select(1:3,4, 8:9)
```

```
# A tibble: 1 x 6
  r.squared adj.r.squared sigma statistic     AIC     BIC
      <dbl>         <dbl> <dbl>     <dbl>   <dbl>   <dbl>
1     0.794         0.794 1812.   207606. 962398. 962425.
```

This makes better fit of standardized residuals but worsens everything else.

## 28 Just add another predictor without any other transformations

```
    lm(formula = price ~ carat + cut + clarity, data = diamonds) %>%
      broom::glance() %>% select(1:3,4, 8:9)
```

```
# A tibble: 1 x 6
  r.squared adj.r.squared sigma statistic     AIC     BIC
      <dbl>         <dbl> <dbl>     <dbl>   <dbl>   <dbl>
1     0.897         0.897 1281.    39107. 925009. 925133.
```

```
    lm(formula = price ~ carat + cut, data = diamonds)
```

```
Call:
lm(formula = price ~ carat + cut, data = diamonds)

Coefficients:
(Intercept)        carat        cut.L        cut.Q        cut.C        cut^4
   -2701.38      7871.08      1239.80      -528.60       367.91        74.59
```

```
    broom::glance(lm_carat_cut_price) %>% select(c(1:3, 6, 8, 9))
```

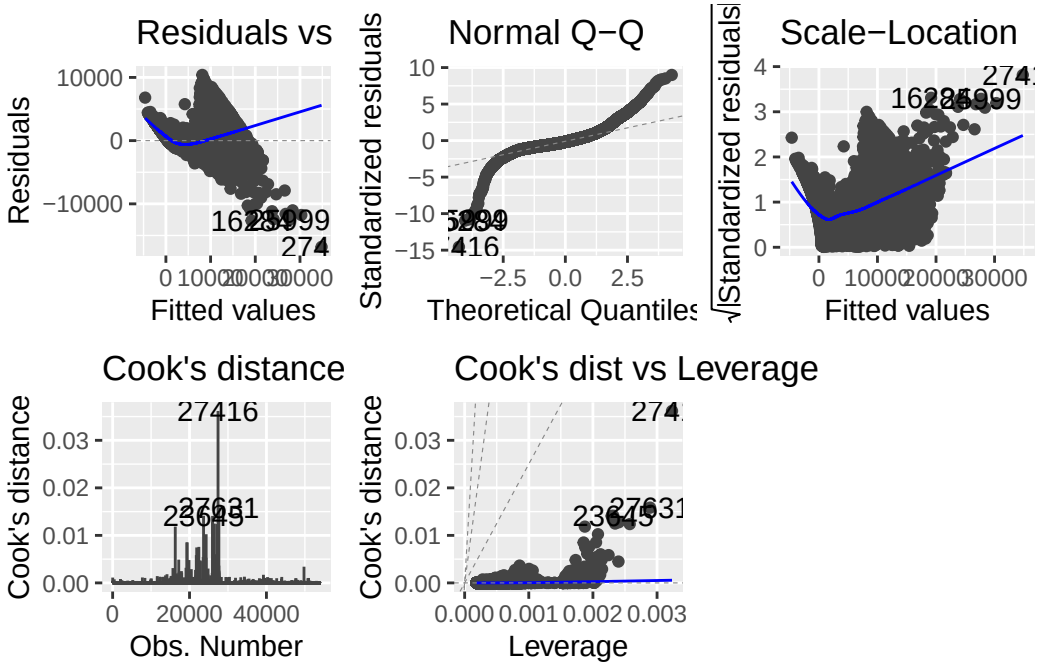
```
# A tibble: 1 x 6
  r.squared adj.r.squared sigma    df     AIC     BIC
      <dbl>         <dbl> <dbl> <dbl>   <dbl>   <dbl>
1     0.856         0.856 1511.     5 942854. 942916.
```

```
    lm(formula = price ~ carat + cut + clarity + color, data = diamonds) %>%
      broom::glance() %>% select(1:3,4, 8:9)
```

```
# A tibble: 1 x 6
  r.squared adj.r.squared sigma statistic     AIC     BIC
      <dbl>         <dbl> <dbl>     <dbl>   <dbl>   <dbl>
1     0.916         0.916 1157.    32641. 914023. 914201.
```

## 29 Diagnostic plots for the richest model

```
    lm(formula = price ~ carat + cut + clarity + color, data = diamonds) %>%
      autoplot(which = c(1:4, 6), ncol = 3)
```



## 30 Interaction between terms

```
richest2 <- lm(formula = price ~ carat + cut *  clarity + color, data =
 ↪  diamonds)
broom::glance(richest2)
```

```
# A tibble: 1 x 12
  r.squared adj.r.squared sigma statistic p.value    df   logLik     AIC     BIC
      <dbl>         <dbl> <dbl>     <dbl>   <dbl> <dbl>    <dbl>   <dbl>   <dbl>
1     0.917         0.916 1153.    12862.       0    46 -456806. 913707. 914134.
# i 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

```
broom::tidy(richest2) %>% filter(p.value < 0.05) %>%
 ↪  arrange(desc(abs(estimate)), p.value)
```

```
# A tibble: 29 x 5
   term            estimate std.error statistic   p.value
   <chr>              <dbl>     <dbl>     <dbl>     <dbl>
 1 carat              8888.      12.0    740.   0
 2 clarity.L          4391.      58.2     75.4  0
 3 (Intercept)       -3653.      18.6   -196.   0
 4 color.L           -1910.      17.7   -108.   0
 5 clarity.Q         -1711.      53.8    -31.8  4.89e-220
 6 clarity.C           922.      48.3     19.1  8.54e- 81
 7 cut.L               646.      43.3     14.9  3.32e- 50
 8 color.Q            -631.      16.1    -39.2  0
 9 cut.C:clarity.L    -440.     110.      -3.98 6.90e-  5
10 cut.L:clarity.L    -433.     162.      -2.68 7.42e-  3
# i 19 more rows
```

If estimate $> 0$: The effect of X1X_1X1 becomes stronger (more positive) as X2X_2X2 increases. If estimate $<0$: The effect of X1X_1X1 becomes weaker (or more negative) as X2X_2X2 increases. I should have normalized the predictor values before. Now they are not interpretable (how much is one of them when the other is 0). Normalization would change this to how much is one when the other is average. Q, C, L: Quadratic, Cubic, Linear.