DATA 473 Project Report

Hien Nguyen, 300199540

18 June 2021

Introduction

The chosen dataset for my project is dataset 1, which involves predicting cubic zirconia's prices, analyzing the data to provide more insights, with a particular focus on which predictor has a more significant impact over the response variable `price` and how a change in those predictors affect the price change.

In order to build a model that satisfies the above requirements, methods such as Linear Regression and Generalized Additive Model (GAM) will be performed. However, before we can decide which model is best, some data analyzing steps will need to be taken. Detailed graphs such as those in the EDA, the Residuals vs Fitted plot, the Q-Q plot, the Scale-Location plot, the Residuals vs Leverage plot, those plots representing response-predictor relationship, and those acquired during the GAM check process, will all be included. Additionally, R-squared, adjusted R-squared, global usefulness test, hypothesis test, and other methods to check regression assumptions will also be closely examined. A factor that would help to get a better understanding of the price such as interaction, will also be investigated. Finally, for the model selection stage, methods such as AIC, BIC, and Mallow's Cp will be used to determine which model we can use to go forward with.

Regarding the prediction model, it is essential to use cross-validation with best subset selection to avoid over-fitting, hence achieve a better result in predicting the cubic zirconia's prices. However, other popular methods such as Ridge regression and LASSO will also compare the result with the best subset selection method.

The whole process will be accompanied by an interpretation of the results. Finally, this data analysis process will decide what course of action would need to be taken to increase profit and other related studies will also be discussed.

Data description, exploratory data analysis (EDA) and methods

Looking at the data structure and summary drawn from the function `str` and `summary` , we can tell that there are 7 numeric variables and 3 categorical variables that we will need to change to factors. However, there are 697 NA in the dataset. According to a well-known rule of thumb, if the percentage missing is low, less than 5%, then removing them would not affect our statistical analysis. Therefore I chose to remove NA from the dataset altogether.

The insight taken from the EDA in Appendix 2 is as below:

- Price is right-skewed; thus, log transformation is considered.
- Based on the correlation coefficient, the strongest predictors of price are carat, x, y, and z - a variable selection procedure should be implemented.
- There are strong correlations between some pairs of predictors - multicollinearity should be investigated.
- Price increases when carat decrease.
- There is potential non-linearity in the relationship between price and each of the numerical predictors. Polynomial or smooth spline regression should be considered.

- According to the boxplot of `cut` , the Premium cut has the highest average price. Though Ideal cut is the best cut, it has the lowest average price. Meanwhile, Fair cut has the second-highest average price, despite being the worse cut out of 5 categories. Additionally, the difference between fair and premium cut is also very small. This is not very intuitive, a closer look at the category is necessary.
- The boxplot for `color` and `clarity` represents the same issue as `cut` , D is the best color and J is the worst, but J has the highest average price and D lowest. The best order for `clarity` is IF, VVS1, VVS2, VS1, VS2, SI1, SI2, I1. However, the price is increasing instead of decreasing.
- The EDA does not tell us anything about the possible interactions between predictors, but these should be investigated.

The first three tables in Appendix 2 examine the max, median, average, and min price of the three category variables. The median and average prices are distributed randomly between each sub-category and they have very similar min and max values. It is clear that price does not depend much on these features, this is somewhat reasonable because cubic zirconia is a man-made stone, produced in the lab and not like a diamond which is sourced from nature; therefore, 99% of them achieve good `cut` , `color` and `clarity` , and if there is any difference, it won't be visible to our naked eyes.

The R-square (0.9208) and Adjusted-R square (0.9207) obtained from `fit1` linear model (Appendix 4) are very similar, suggesting that no predictor is redundant. Furthermore, to quantify our uncertainty about the corresponding population regression coefficients and to make sure our model has at least one non-zero coefficient, we test the global hypotheses:

- H0: $\beta 1$=$\beta 2$= … =βp = 0
- H1: at least one βj = 0

According to Model Assessment Summary table, We find F = 13263 with 23 and 26246 d.o.f and p−value < 2.2e-16. There is very strong evidence to reject H0, and there is no evidence that all regression coefficients are zero in the population. It is worth going on further analyze and interpret a model of `price` against the predictors.

The plot in Appendix 5 shows that there is evidence of non-linearity, non-normality, and non-constant variance with one influential observation #25796. Log transformation of the response is used to help with assumptions violation.

After another check of Residuals vs Leverage plot, there are still some influential points such as #17507, #5822 and #25796, I decided to remove them because influential point is an outlier that dramatically affects the slope of the regression line and it will cause the coefficient of determination to be bigger, sometimes, smaller. As expected, after removing the three outliers, R-square and Adjusted R-squared both increase from 0.9207 to 0.9857. Since we changed the model, we need to check the Residuals vs Leverage plot again to see if it works. #17507, #5822, and #25796 are no longer appear outside of Cook's distance, however, there is still one influential point, so the same process is applied again. The new model is store in `new_fit3` , and R-square and Adjusted R-squared increase from 0.9857 to 0.9883.

We continue to use hypotheses test to check normality assumption:

- H0: The sample comes from a normal distribution
- H1: The sample does not come from a normal distribution

The Shapiro-Wilk statistic does not work for a dataset with more than 5000 observations; therefore, only the first 5000 rows of `new_cubiz2` will be used for this test. Moreover, Anderson-Darling statistic is also included to perform the test on the whole dataset. The p-values from both tests are very small and very close to 0 (Appendix 5.1). As a result, H0 is rejected, and we conclude that the sample does not come from a normal distribution.

On the other hand, the Breusch-Pagan statistic is used to test the hypotheses:

- H0: Homoscedasticity is present

- H1: Heteroscedasticity is present

The Breusch-Pagan also reject the null hypotheses, confirming that the residuals are not distributed with equal variance.

The original `cubiz` dataset does not have time series or spatial data, as such, autocorrelation is not violated.

As shown in VIF Values table (Appendix 5.2), `x` , `y` , `z` have severe multicollinearity since their values of GVIF^(1/(2*Df)) are much more than 10. With `x` has a lower VIF score than `y` and `z` , only `x` will be retained in the model. Though this change reduces R-square and Adjusted R-squared from 0.9883 to 0.9879, there is no more evidence of multicollinearity, the assumption is honored, and that is more important.

Looking at the plot of residuals against each predictor in Appendix 6.1, we can see an indication of non-linear patterns in `carat` and `x` . To identify suitable transformations, I plot `log(price)` against each of these two predictors (see plot “aa” for `carat` and “dd” for `x`). The non-monotonic patterns in the plots of `log(price)` against each of `carat` and `length (x)` suggest polynomial transformations would be suitable for both predictors. Additionally, there is not many turning points, therefore, polynomial degree 2 should be sufficient. The transformations indeed increase R-square and Adjusted R-squared from 0.9879 to 0.9890.

The Model Assumption plots in Appendix 6.2 indicate that the violation of assumptions has been significantly reduced. There is no influential point, and the plots show some sign of linearity and homoscedasticity. However, the q-q plot shows tails are heavier than in a normal distribution. There is still potential non-normality in the residuals. As a result, generalized additive model (GAM) is considered.

Before fitting GAM, we could first explore interaction term. The bigger the size, the heavier the stone measured in carat, one possible interaction could be `carat:x` . As represented in Appendix 6.2, when all other predictors are held constant, log(price) increases as the carat weight increases for those cubic zirconia stones with lengths in the range from 3 to 8mm. The stones with the shortest length 3mm have the highest log(price) (as `carat` increasing) indicated by the steepest slope and the slope gets less steep as `carat` increases and `x` reaches 8mm. On the contrary, log(price) decreases as the carat weight increases for stones with 9 to 10mm lengths. Furthermore, after checking `fit6` - the model with interaction, we can see that R-square and Adjusted R-squared raise from 0.9890 to 0.9891. This is a good sign, and interaction should be included.

Based on the GAM model fitted in Appendix 6.4, we can conclude that all smooth terms for `carat` , `depth` , `table` and `x` are non-linear and significant, since their edf are bigger than 1 and p-values are close to 0.

In the Appendix 6.4 plots, `carat` and `x` have the most wiggly curve and have the highest edf values of 7.9 and 8.9. For categorical predictors, the confidence interval of all the `cut` , `color` , and `clarity` types exclude zero, reflecting a statistically significant difference (at the 5% significance level) in log(price) between each reference level and its fellow sub-categories.

Model-checking with `gam.check()` shows that the histogram is symmetric, but the q-q plot still shows that tails are heavier than the normal distribution. Therefore, there is potential non-normality in the residuals. The Resid vs linear pred plot also shows evidence of non-constant variance. Moreover, the k-index of `depth` and `table` are close to 1 and their edfs are not close to k’, so we could accept their current number of basis functions. However, `carat` and `x` are different, though their k-index are much lower than 1, their edf values are very close to k’. Hence, it is best to refit the model with higher k values to double-check these conclusions. After raising the k values from the default number (at k = 9) to k equal to 20, as expected, the number of basis functions increase significantly from 7.9 to 15.8 for `carat` and 8.9 to 17.5 for `x` . On the other hand, the basis functions for `depth` and `table` only increase slightly. By setting the k value to 20, the residual diagnostic plots now show more sign of linearity, constant variance, and normality.

When we start the model selection process, we denote the model that has the lowest AIC value with all predictors as model B and the model with the second smallest AIC value as model A (excludes `table`). According to the rule of thumb, when the difference is larger than 10 (in this case, it is 79), we prefer model B, all predictors are retained. Additionally, BIC and Mallow's Cp also yield the same result as AIC (see Appendix 7.1, 7.2 and 7.3). As such, the chosen model in terms of AIC, BIC, and Mallow's Cp would have all eight predictors, `carat`, `cut`, `color`, `clarity`, `depth`, `table`, `x` and `carat:x`.

About model selection for GAM model `fit.gam2`, p-values shown in the summary table are all very small and close to 0. This means that all predictors are significant in predicting the response. Similarly to linear model, GAM model should also include `carat`, `cut`, `color`, `clarity`, `depth`, `table`, `x` and `carat:x` (Appendix 7.4).

The next question is whether we should use the linear model or the GAM model - in other words, does the additional complexity of the latter result in substantially improved fit. The comparison tables in Appendix 7.7 suggest that GAM model is the winner as it has much lower values for both AIC and BIC methods.

Some prediction settings are represented in Appendix 9, such as best subset selection using cross-validation, Ridge regression, and LASSO regression. These methods aim to find the best prediction model with the lowest test MSE. LASSO gives the lowest test MSE of 822,217 (see Appendix 9.2 for the prediction coefficients).

Discussion and conclusion inference

Conclusion

- Looking at the Linear models, we can see that the carat weight and the length have the most significant impact on price.
- Looking at the GAM model, the reference level for cut is Fair, for color is D and for clarity is IF.
- Cut Fair is the cheapest cut compared to other cut types, and D is the most expensive color compared to other color types. Meanwhile, clarity IF is the most expensive among other clarity types.
- The biggest difference among these categorical variables is cut Fair versus cut Premium and Ideal; color D versus color G, H, I, J; and clarity IF versus VS1, VS2, SI1, SI2 and I1. Therefore, color D, E, F, clarity IF, VVS1, VVS2 and cut Premium, Ideal are the features that would bring more revenue. Producing stones with these features and focusing on marketing them will bring more profit to the company.
- We expect the price to increase by a multiplicative factor of 0.15 (15%) for each Ideal cut stone and 0.11 (11%) for each Premium one, compared to each Fair one, while holding all other predictors constant.
- We expect the price to decrease by a multiplicative factor of 0.6 (40%) for each color J stone and 0.85 (14%) for each with color G , compared to each with color D, while holding all other predictors constant.
- We expect the price to decrease by a multiplicative factor of 0.35 (65%) for each I1 clarity stone and 0.76 (23%) for each VS1 graded one, compared to each IF graded one, while holding all other predictors constant.
- We already knew that, log(price) increases as the carat weight increases for those cubic zirconia stones with length in the range from 3 to 8mm. Hence, the company should not prioritize any activity on any heavy and bigger stone than 8mm length. Instead, they should put their effort in smaller stone's lengths, especially those around 3, 4 and 5mm, as the more carat weight those stones get, the higher the log(price).

Discussion

I found two analyses on this dataset. Each of them uses very different approach from mine and each other, though they both use Python instead of R.

The first model was built by Sindiri (towardsdatascience.com, 2020), he has somewhat a similar process to mine, however, he focused mainly on the prediction model, performing the exploratory data analysis (EDA), preparing the dataset for training, creating a linear regression model, training the model to fit the data and finally, making predictions using the trained model. After analyzing the EDA, he concluded that `x`, `y` and `z` have a strong correlation with `price`, while `depth` has a very weak relation, thus he decided to drop this variable. He chose Pytorch library as his tool for building the prediction model. Hence, he needed to transform the data frame dataset to the Tensor dataset and then split them into training set and validation set. Furthermore, training set was used to train the model and tune the hyperparameters (learning rate, epochs, batch size). After all these hyperparameters were locked in, he used them to minimize the loss function and then used the final model to test the validation set.

Muralidharan built the second model. His initial steps were very similar to mine and Sindiri's, such as examining the data by looking through the dataset structure, finding out the predictor's data types, and checking NA. Surprisingly their datasets do not have any NA. In particular, he went into great detail and examined each variable by using various chart types. His EDA was quite similar to the first one, they both used `sns.pairplot` and `sns.heatmap`, while I combined both of them in one using `pairs.panels`. Muralidharan also performed data scaling and looked for VIF values before and after scaling. He also removed outliers that appeared in each variable's boxplot. He then split the data into two (train and test dataset) and apply Linear Regression using Sklearn package. According to his model summary, `depth` has a p-value larger than significance level alpha equal to 0.05, `depth` was dropped out from the model. Muralidharan's findings and mine are pretty similar in suggesting that the Ideal, Premium, and Very Good types of cut would bring profits, whereas Fair cut and I1 clarity would not.

References

Sindiri V. (2020, June 16). Diamond price prediction based on their cut, colour, clarity, price with PyTorch. https://towardsdatascience.com/diamond-price-prediction-based-on-their-cut-colour-clarity-price-with-pytorch-1e0353d2503b (https://towardsdatascience.com/diamond-price-prediction-based-on-their-cut-colour-clarity-price-with-pytorch-1e0353d2503b)

Muralidharan N. Linear Regression

Appendix

1. Organise the data for analysis

```
data <- read.csv('cubicz.csv')
cubiz <- data[,-1]
head(cubiz)
```

```
##   carat       cut color clarity depth table    x    y    z price
## 1  0.30     Ideal     E     SI1  62.1    58 4.27 4.29 2.66   499
## 2  0.33   Premium     G      IF  60.8    58 4.42 4.46 2.70   984
## 3  0.90 Very Good     E    VVS2  62.2    60 6.04 6.12 3.78  6289
## 4  0.42     Ideal     F     VS1  61.6    56 4.82 4.80 2.96  1082
## 5  0.31     Ideal     F    VVS1  60.4    59 4.35 4.43 2.65   779
## 6  1.02     Ideal     D     VS2  61.5    56 6.46 6.49 3.99  9502
```

```
str(cubiz)
```

```
## 'data.frame':    26967 obs. of  10 variables:
##  $ carat  : num  0.3 0.33 0.9 0.42 0.31 1.02 1.01 0.5 1.21 0.35 ...
##  $ cut    : chr  "Ideal" "Premium" "Very Good" "Ideal" ...
##  $ color  : chr  "E" "G" "E" "F" ...
##  $ clarity: chr  "SI1" "IF" "VVS2" "VS1" ...
##  $ depth  : num  62.1 60.8 62.2 61.6 60.4 61.5 63.7 61.5 63.8 60.5 ...
##  $ table  : num  58 58 60 56 59 56 60 62 64 57 ...
##  $ x      : num  4.27 4.42 6.04 4.82 4.35 6.46 6.35 5.09 6.72 4.52 ...
##  $ y      : num  4.29 4.46 6.12 4.8 4.43 6.49 6.3 5.06 6.63 4.6 ...
##  $ z      : num  2.66 2.7 3.78 2.96 2.65 3.99 4.03 3.12 4.26 2.76 ...
##  $ price  : int  499 984 6289 1082 779 9502 4836 1415 5407 706 ...
```

```
summary(cubiz)
```

```
##      carat             cut                 color              clarity         
##  Min.   :0.2000   Length:26967       Length:26967       Length:26967      
##  1st Qu.:0.4000   Class :character   Class :character   Class :character  
##  Median :0.7000   Mode  :character   Mode  :character   Mode  :character  
##  Mean   :0.7984                                                           
##  3rd Qu.:1.0500                                                           
##  Max.   :4.5000                                                           
##                                                                           
##      depth           table             x                y         
##  Min.   :50.80   Min.   :49.00   Min.   : 0.00   Min.   : 0.000  
##  1st Qu.:61.00   1st Qu.:56.00   1st Qu.: 4.71   1st Qu.: 4.710  
##  Median :61.80   Median :57.00   Median : 5.69   Median : 5.710  
##  Mean   :61.75   Mean   :57.46   Mean   : 5.73   Mean   : 5.734  
##  3rd Qu.:62.50   3rd Qu.:59.00   3rd Qu.: 6.55   3rd Qu.: 6.540  
##  Max.   :73.60   Max.   :79.00   Max.   :10.23   Max.   :58.900  
##  NA's   :697                                                      
##        z              price      
##  Min.   : 0.000   Min.   :  326  
##  1st Qu.: 2.900   1st Qu.:  945  
##  Median : 3.520   Median : 2375  
##  Mean   : 3.538   Mean   : 3940  
##  3rd Qu.: 4.040   3rd Qu.: 5360  
##  Max.   :31.800   Max.   :18818  
## 
```

```
# remove NA
cubiz <- cubiz[!is.na(cubiz$depth),]
summary(cubiz)
```

```
##      carat           cut                color             clarity
##  Min.   :0.200   Length:26270       Length:26270       Length:26270
##  1st Qu.:0.400   Class :character   Class :character   Class :character
##  Median :0.700   Mode  :character   Mode  :character   Mode  :character
##  Mean   :0.798
##  3rd Qu.:1.050
##  Max.   :4.500
##      depth           table              x                y
##  Min.   :50.80   Min.   :49.00   Min.   : 0.000   Min.   : 0.000
##  1st Qu.:61.00   1st Qu.:56.00   1st Qu.: 4.710   1st Qu.: 4.720
##  Median :61.80   Median :57.00   Median : 5.690   Median : 5.700
##  Mean   :61.75   Mean   :57.46   Mean   : 5.729   Mean   : 5.733
##  3rd Qu.:62.50   3rd Qu.:59.00   3rd Qu.: 6.550   3rd Qu.: 6.540
##  Max.   :73.60   Max.   :79.00   Max.   :10.230   Max.   :58.900
##        z              price
##  Min.   :0.000   Min.   :  326
##  1st Qu.:2.900   1st Qu.:  945
##  Median :3.520   Median : 2375
##  Mean   :3.537   Mean   : 3938
##  3rd Qu.:4.040   3rd Qu.: 5361
##  Max.   :8.060   Max.   :18818
```

```r
library(forcats)

cubiz$cut <- factor(cubiz$cut, levels = c('Fair', 'Good', 'Very Good', 'Premium',
                                  'Ideal'))
cubiz$color <- factor(cubiz$color)
cubiz$clarity <- factor(cubiz$clarity, levels = c('IF', 'VVS1', 'VVS2', 'VS1',
                                          'VS2', 'SI1', 'SI2', 'I1'))

str(cubiz)
```

```
## 'data.frame':    26270 obs. of  10 variables:
##  $ carat  : num  0.3 0.33 0.9 0.42 0.31 1.02 1.01 0.5 1.21 0.35 ...
##  $ cut    : Factor w/ 5 levels "Fair","Good",..: 5 4 3 5 5 5 2 4 2 5 ...
##  $ color  : Factor w/ 7 levels "D","E","F","G",..: 2 4 2 3 3 1 5 2 5 3 ...
##  $ clarity: Factor w/ 8 levels "IF","VVS1","VVS2",..: 6 1 3 4 2 5 6 6 6 5 ...
##  $ depth  : num  62.1 60.8 62.2 61.6 60.4 61.5 63.7 61.5 63.8 60.5 ...
##  $ table  : num  58 58 60 56 59 56 60 62 64 57 ...
##  $ x      : num  4.27 4.42 6.04 4.82 4.35 6.46 6.35 5.09 6.72 4.52 ...
##  $ y      : num  4.29 4.46 6.12 4.8 4.43 6.49 6.3 5.06 6.63 4.6 ...
##  $ z      : num  2.66 2.7 3.78 2.96 2.65 3.99 4.03 3.12 4.26 2.76 ...
##  $ price  : int  499 984 6289 1082 779 9502 4836 1415 5407 706 ...
```

```r
summary(cubiz)
```

```
##      carat               cut        color      clarity         depth      
##  Min.   :0.200   Fair     :  757   D:3268   SI1    :6408   Min.   :50.80  
##  1st Qu.:0.400   Good     : 2382   E:4793   VS2    :5925   1st Qu.:61.00  
##  Median :0.700   Very Good: 5878   F:4612   SI2    :4447   Median :61.80  
##  Mean   :0.798   Premium  : 6707   G:5529   VS1    :3991   Mean   :61.75  
##  3rd Qu.:1.050   Ideal    :10546   H:3991   VVS2   :2479   3rd Qu.:62.50  
##  Max.   :4.500                     I:2676   VVS1   :1791   Max.   :73.60  
##                                    J:1401   (Other):1229                  
##      table             x                y                z        
##  Min.   :49.00   Min.   : 0.000   Min.   : 0.000   Min.   :0.000  
##  1st Qu.:56.00   1st Qu.: 4.710   1st Qu.: 4.720   1st Qu.:2.900  
##  Median :57.00   Median : 5.690   Median : 5.700   Median :3.520  
##  Mean   :57.46   Mean   : 5.729   Mean   : 5.733   Mean   :3.537  
##  3rd Qu.:59.00   3rd Qu.: 6.550   3rd Qu.: 6.540   3rd Qu.:4.040  
##  Max.   :79.00   Max.   :10.230   Max.   :58.900   Max.   :8.060  
##                                                                   
##      price      
##  Min.   :  326  
##  1st Qu.:  945  
##  Median : 2375  
##  Mean   : 3938  
##  3rd Qu.: 5361  
##  Max.   :18818  
## 
```

2. EDA

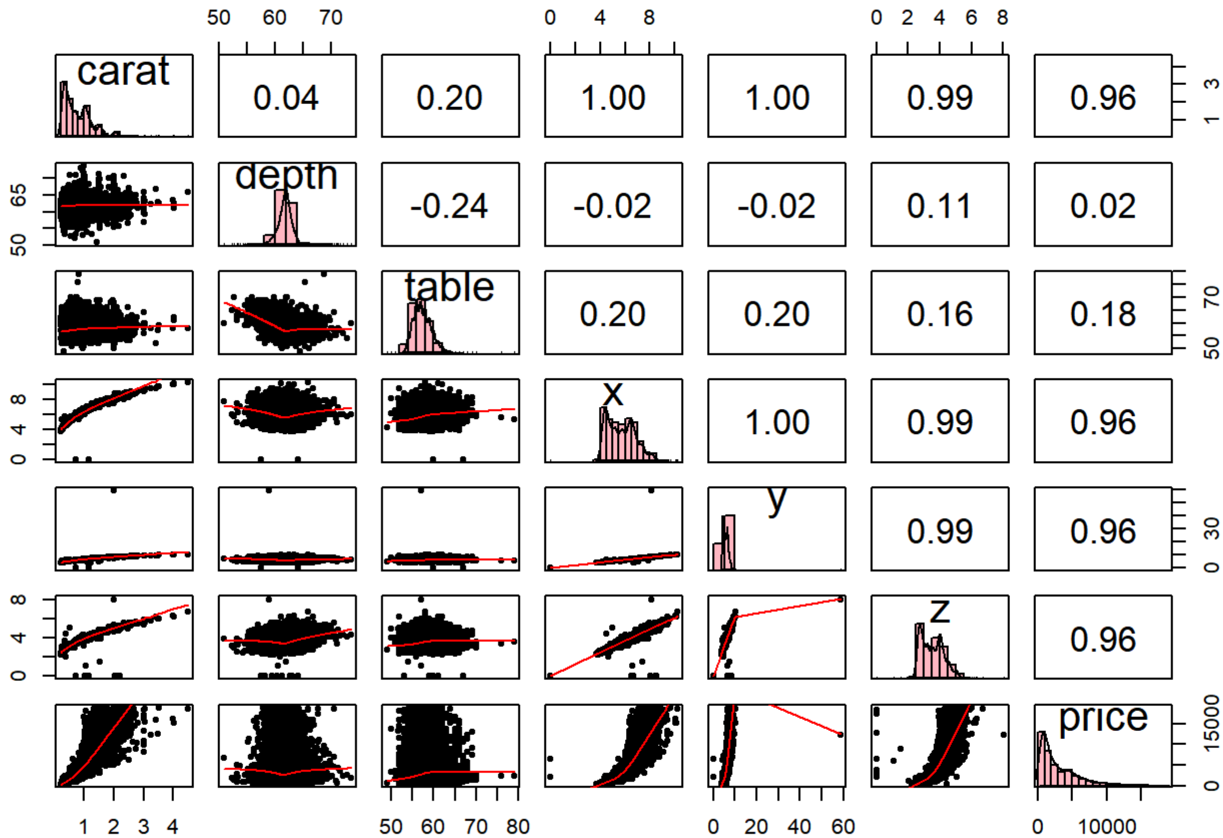
```
library(dplyr)
```

```
## 
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
## 
##     filter, lag
```

```
## The following objects are masked from 'package:base':
## 
##     intersect, setdiff, setequal, union
```

```
library(psych)

cubiz%>%
  dplyr::select(where(is.numeric))%>%
  pairs.panels(method='spearman', hist.col='lightpink', density=TRUE,
               ellipse=FALSE)
```

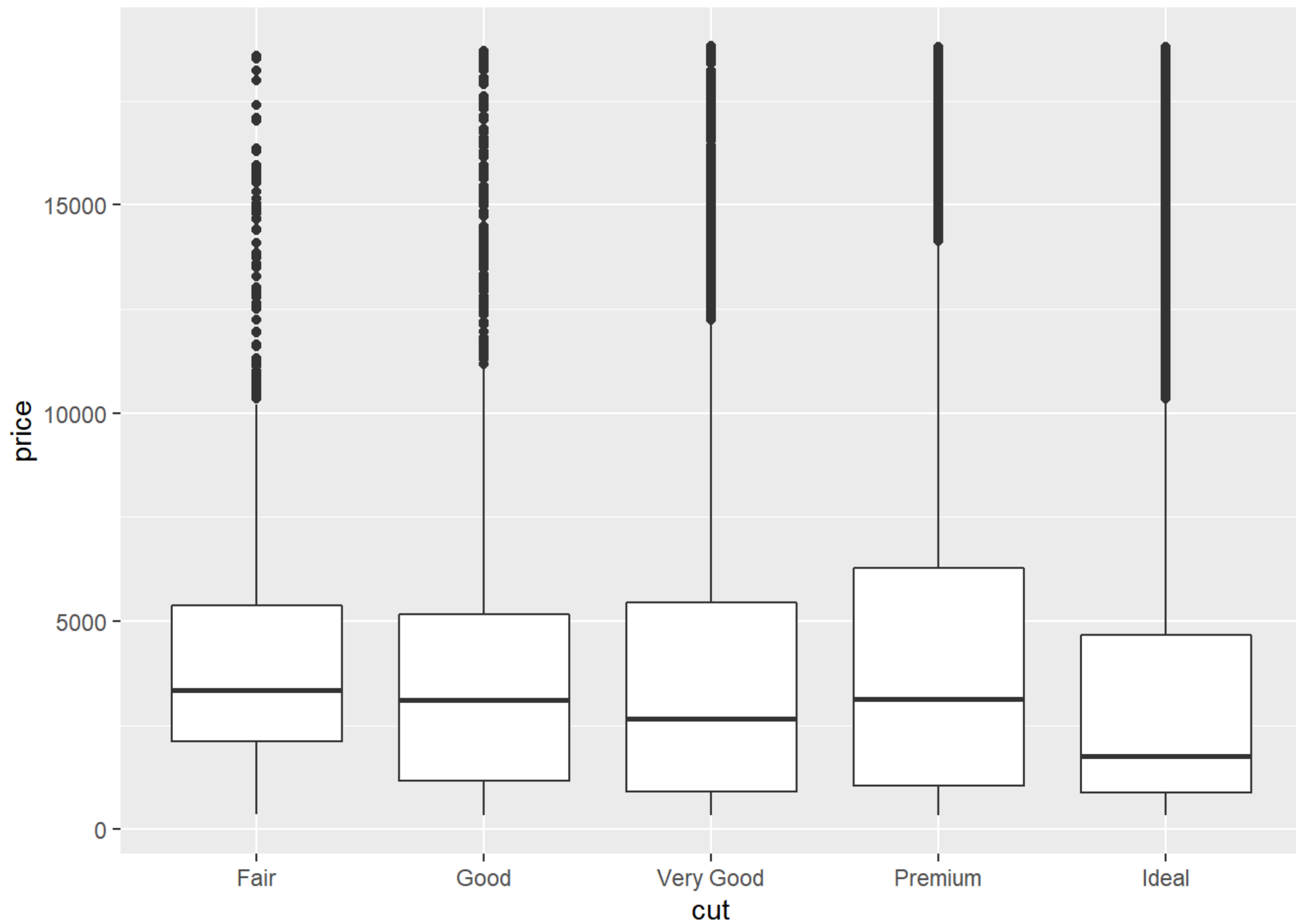



```r
library(ggplot2)
```

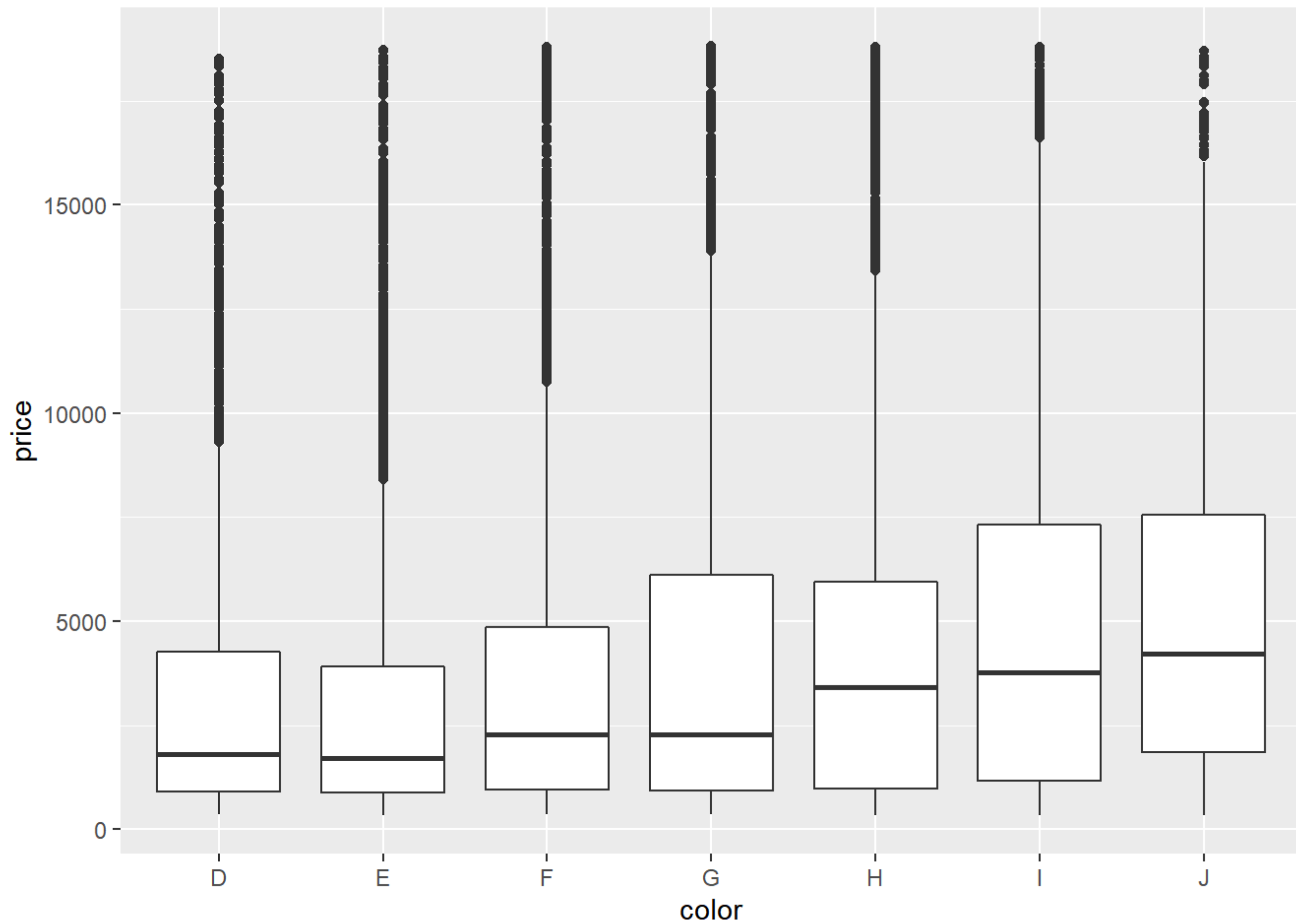
```
##
## Attaching package: 'ggplot2'
```

```
## The following objects are masked from 'package:psych':
##
##     %+%, alpha
```

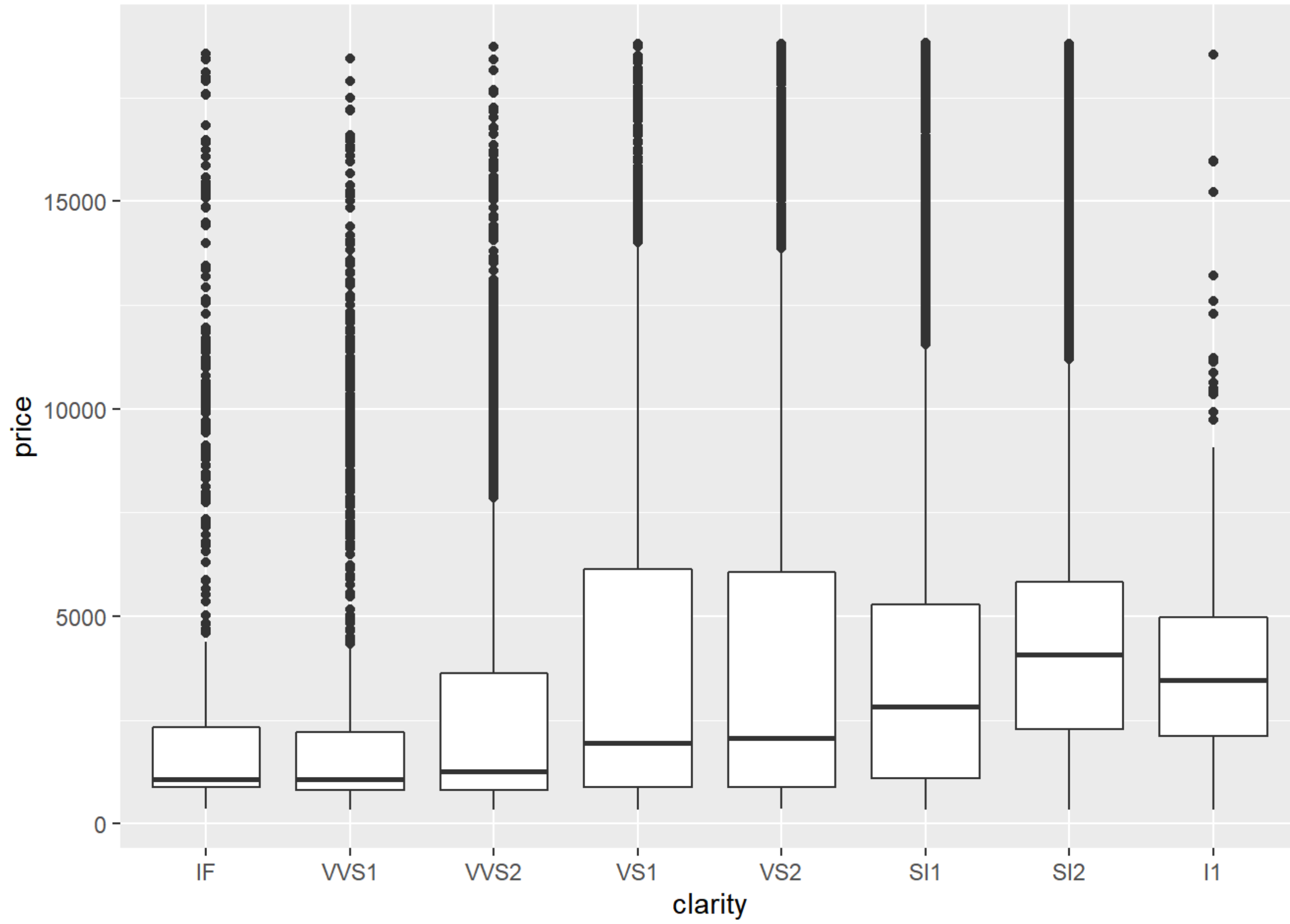
```r
ggplot(cubiz,aes(x=cut, y=price)) + geom_boxplot()
```




```
ggplot(cubiz,aes(x=color, y=price)) + geom_boxplot()
```




```
ggplot(cubiz,aes(x=clarity, y=price)) + geom_boxplot()
```




```
library(pander)

#By cut
cut_price <- cubiz %>% select(cut, price)

pander(cut_price %>%
  group_by(cut) %>%
  summarise(
    MaxPriceByCut = max(price),
    MedianPriceByCut = median(price),
    AveragePriceByCut = mean(price),
    MinPriceByCut = min(price)
  ) %>%
  arrange(cut))
```

cut	MaxPriceByCut	MedianPriceByCut	AveragePriceByCut	MinPriceByCut
Fair	18574	3348	4559	369
Good	18707	3110	3952	335
Very Good	18818	2648	4032	336
Premium	18795	3118	4544	326
Ideal	18804	1764	3452	326

```
#By clarity
clarity_price <- cubiz %>% select(clarity, price)

pander(clarity_price %>%
  group_by(clarity) %>%
  summarise(
    MaxPriceByClarity = max(price),
    MedianPriceByClarity = median(price),
    AveragePriceByClarity = mean(price),
    MinPriceByClarity = min(price)
  ) %>%
  arrange(clarity))
```

Table continues below

clarity	MaxPriceByClarity	MedianPriceByClarity	AveragePriceByClarity
IF	18552	1064	2749
VVS1	18445	1061	2487
VVS2	18718	1262	3276
VS1	18795	1949	3843
VS2	18791	2066	3959
SI1	18818	2809	4010
SI2	18804	4077	5081
I1	18531	3459	3909

MinPriceByClarity
369
336
336
338
357
326
326
345

```
#By color
color_price <- cubiz %>% select(color, price)

pander(color_price %>%
  group_by(color) %>%
  summarise(
    MaxPriceByColor = max(price),
    MedianPriceByColor = median(price),
    AveragePriceByColor = mean(price),
    MinPriceByColor = min(price)
  ) %>%
  arrange(color))
```

Table continues below

color	MaxPriceByColor	MedianPriceByColor	AveragePriceByColor
D	18526	1802	3187
E	18731	1707	3081
F	18791	2267	3692
G	18818	2275	4009
H	18795	3417	4482
I	18795	3764	5139
J	18701	4226	5307

MinPriceByColor
357
326
357
361
337
336
335

3. Fitting linear model with no interactions

```
fit1 <- lm(price ~ carat + cut + color + clarity + depth + table + x + y + z,
           data=cubiz)
```

4. Summary from fit1 above

```
summ.fit1 <- summary(fit1)
Rsq<-summ.fit1$r.squared
AdjRsq<-summ.fit1$adj.r.squared
fit0 <- lm(price~1, data=cubiz)
lrt.fit1 <- anova(fit0, fit1)
Fval <- lrt.fit1$F[2]
pval <- lrt.fit1$`Pr(>F)`[2]
Statistic <- c("F-statistic", "p-value", "R-squared","Adj. R-squared")
Value <- c(Fval,pval,Rsq,AdjRsq)
fit1.res <- data.frame(Statistic,Value)

pander(fit1.res, digits=4,caption="Model assessment summary")
```

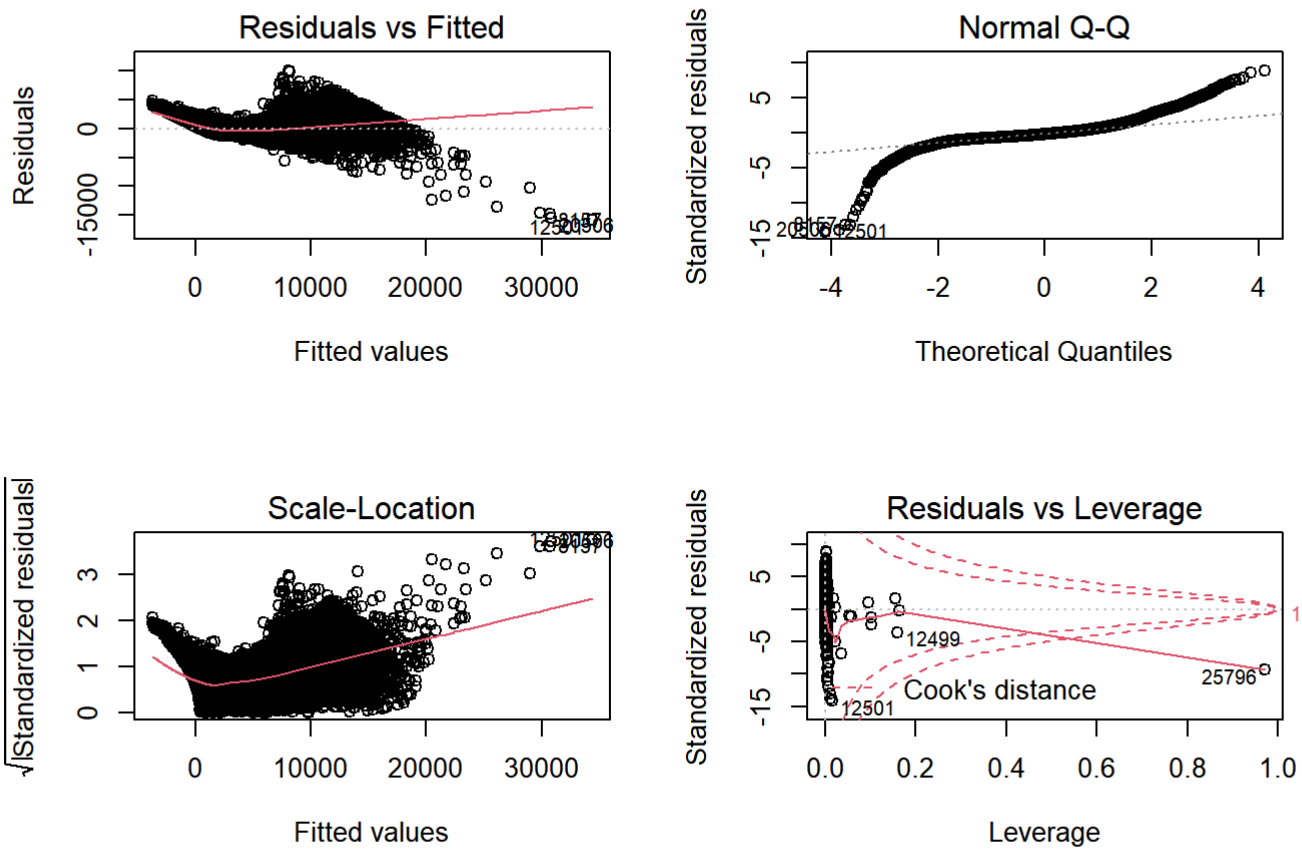
Model assessment summary

Statistic	Value
F-statistic	13263
p-value	0
R-squared	0.9208
Adj. R-squared	0.9207

5. Check the regression assumptions for this initial model

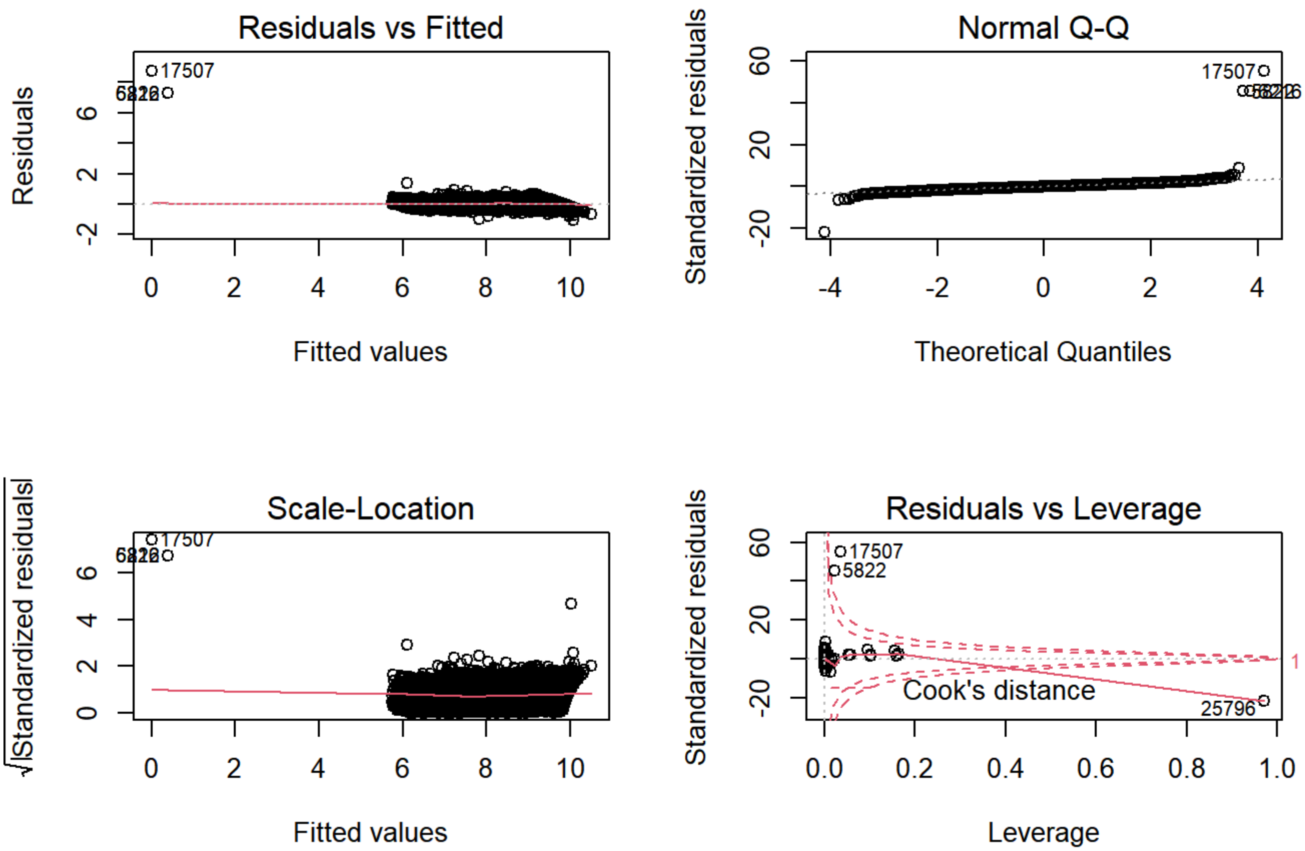
```
par(mfrow=c(2,2))
plot(fit1)
```

```
## Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produced

## Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produced
```




```
# log transformation
fit2 <- lm(log(price) ~ carat + cut + color + clarity + depth + table + x + y + z,
           data=cubiz)

# re-check assumptions
par(mfrow=c(2,2))
plot(fit2)
```

```
## Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produced

## Warning in sqrt(crit * p * (1 - hh)/hh): NaNs produced
```

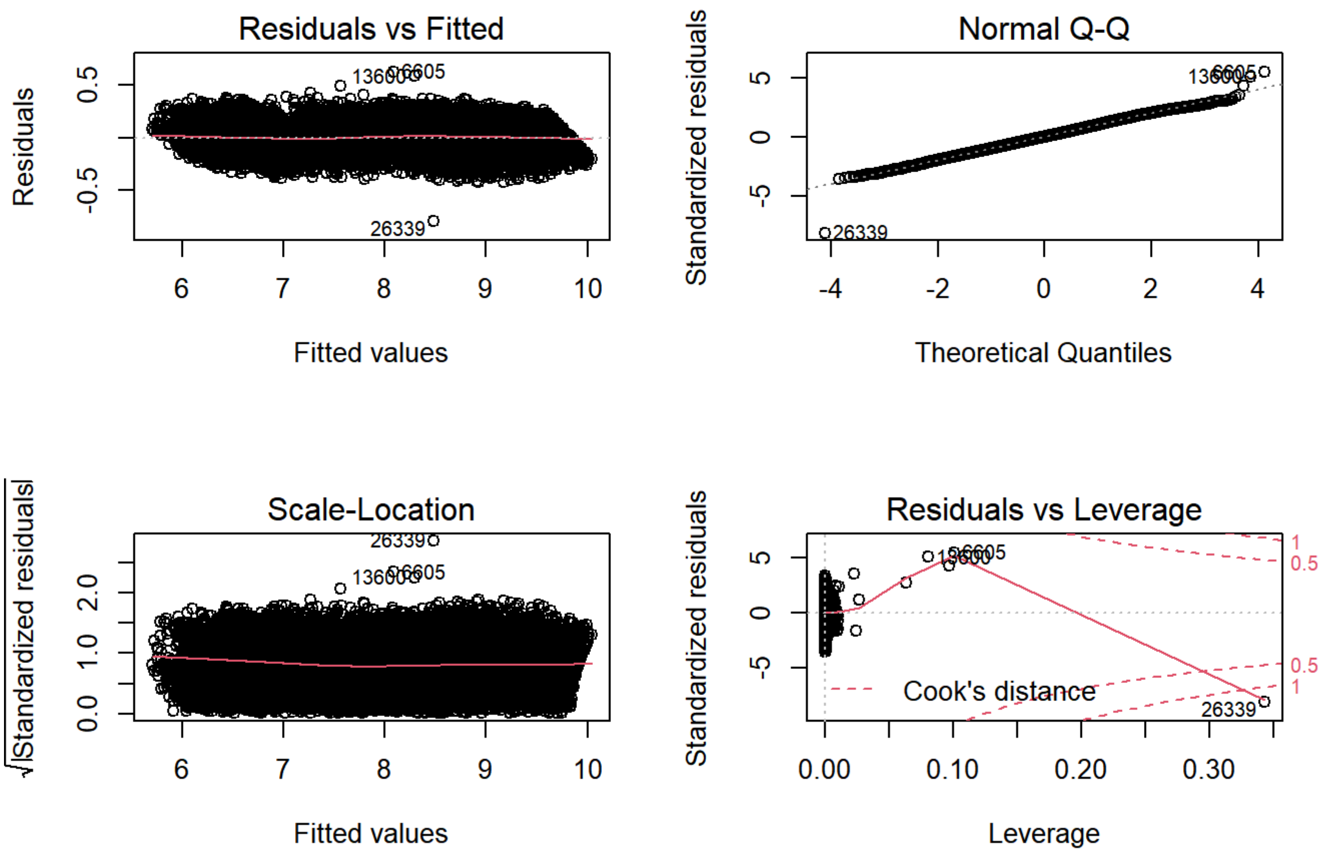



```
#remove newly appear influential points
HighLeverage <- cooks.distance(fit2) > (4/nrow(cubiz))
LargeResiduals <- rstudent(fit2) > 3
new_cubiz <- cubiz[!HighLeverage & !LargeResiduals,]

fit3 <- lm(log(price) ~ carat + cut + color + clarity + depth + table + x + y + z,
           data=new_cubiz)

# function to create Rsq and AdjRsq summary table
compare_rsq_summary <- function(Rsq1, AdjRsq1, Rsq2, AdjRsq2, table_name)
{
  Rsq1 <- Rsq1
  Rsq2 <- Rsq2
  AdjRsq1 <- AdjRsq1
  AdjRsq2 <- AdjRsq2
  Statistic <- c('R-squared', 'Adjusted R-squared')
  Model.fit1 <- c(Rsq1, AdjRsq1)
  Model.fit2 <- c(Rsq2, AdjRsq2)
  mod.summ <- data.frame(Statistic, Model.fit1, Model.fit2)
  names(mod.summ) <- c("Statistic", "Previous Model",
                        "New Model")
  pander(mod.summ, digits=5, caption = table_name)
}

compare_rsq_summary(0.9208, 0.9207, 0.9857, 0.9857,
                    "Model comparison between with and without influential points")
```

Model comparison between with and without influential points

Statistic	Previous Model	New Model
R-squared	0.9208	0.9857
Adjusted R-squared	0.9207	0.9857

```
par(mfrow=c(2,2))
plot(fit3)
```

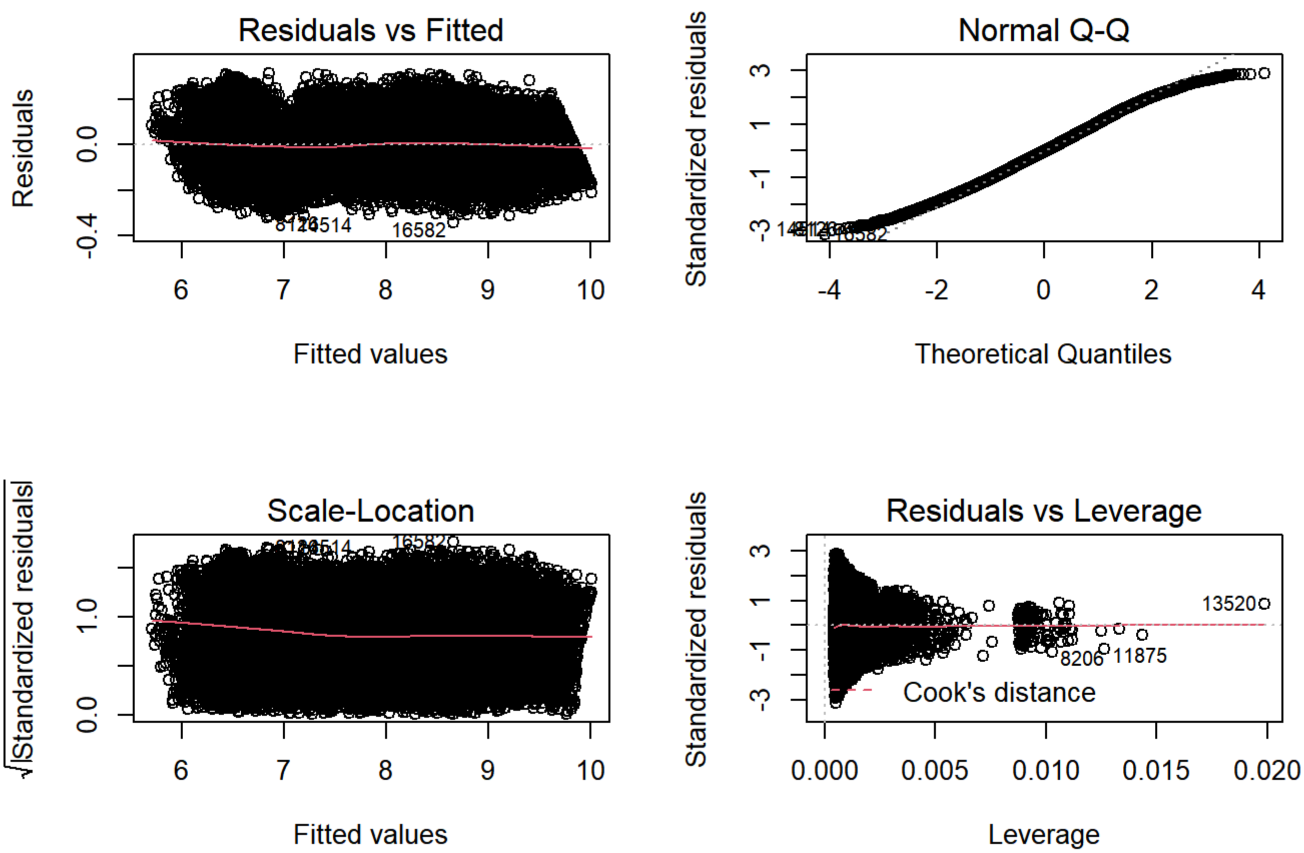



```
# remove influential point #26339
HighLeverage <- cooks.distance(fit3) > (4/nrow(cubiz))
LargeResiduals <- rstudent(fit3) > 3
new_cubiz2 <- new_cubiz[!HighLeverage & !LargeResiduals,]

#fit the model without the influential point #26339 and others
new_fit3 <- lm(log(price) ~ carat + cut + color + clarity + depth + table + x + y + z,
           data=new_cubiz2)

compare_rsq_summary(0.9857, 0.9857, 0.9883, 0.9883,
                    "Model comparison between with and without influential points (con
t.)")
```

Model comparison between with and without influential points (cont.)

Statistic	Previous Model	New Model
R-squared	0.9857	0.9883
Adjusted R-squared	0.9857	0.9883

```
#plot again to see if the point has been removed and if there is any more influential po
ints appear
par(mfrow=c(2,2))
plot(new_fit3)
```



5.1 Hypothesis test

```
# use Anderson-Darling normality test
library(nortest)
pander(ad.test(new_fit3$res))
```

Anderson-Darling normality test: `new_fit3$res`

Test statistic	P value
9.663	2.274e-23 * * *

```
# apply Shapiro test to only first 5000 observations
new_cubiz3 <- new_cubiz2[1:5000,]
fit.cubiz <- lm(log(price) ~ carat + cut + color + clarity + depth + table + x + y + z,
           data=new_cubiz3)
pander(shapiro.test(fit.cubiz$res))
```

Shapiro-Wilk normality test: `fit.cubiz$res`

Test statistic	P value
0.9968	8.22e-09 * * *

```
# Breusch-Pagan test
library(lmtest)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
pander(bptest(new_fit3))
```

studentized Breusch-Pagan test: `new_fit3`

Test statistic	df	P value
791.8	23	1.895e-152 * * *

5.2 Check multicollinearity

```
library(car)
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:psych':
##
##     logit
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
library(knitr)
pander(car::vif(new_fit3), digits=2, caption='VIF values')
```

VIF values

	GVIF	Df	GVIF^(1/(2*Df))
carat	29	1	5.4
cut	2.4	4	1.1
color	1.2	6	1
clarity	1.3	7	1
depth	32	1	5.7
table	1.8	1	1.4
x	1185	1	34
y	1233	1	35
z	2369	1	49

```
# remove y and z
new_cubiz2$y <- NULL
new_cubiz2$z <- NULL

# fit new model without y and z
fit4 <- lm(log(price) ~ carat + cut + color + clarity + depth + table + x,
           data=new_cubiz2)

pander(car::vif(fit4), digits=2, caption='VIF values')
```

VIF values

	GVIF	Df	GVIF^(1/(2*Df))
carat	29	1	5.4
cut	1.9	4	1.1
color	1.2	6	1
clarity	1.3	7	1
depth	1.4	1	1.2
table	1.8	1	1.4

	GVIF	Df	GVIF^(1/(2*Df))
x	29	1	5.4

```
compare_rsq_summary(0.9883, 0.9883, 0.9879, 0.9879,
                    "Model comparison between incl. and excl. Width and Height")
```

Model comparison between incl. and excl. Width and Height

Statistic	Previous Model	New Model
R-squared	0.9883	0.9879
Adjusted R-squared	0.9883	0.9879

6. Interactions, transformations and smooth term

6.1 Plot residuals against each of the predictor

```
library(broom)

fit4 <- lm(log(price) ~ carat + cut + color + clarity + depth + table + x,
           data=new_cubiz2) %>%
  augment()
a<-ggplot(fit4, aes(x=carat, y=.std.resid)) +
  geom_point() + geom_smooth(method=NULL) +
  labs(x='Carat', y='Residuals') +
  theme_bw()
b<-ggplot(fit4, aes(x=depth, y=.std.resid)) +
  geom_point() + geom_smooth(method=NULL) +
  labs(x='Depth', y='Residuals') +
  theme_bw()
c<-ggplot(fit4, aes(x=table, y=.std.resid)) +
  geom_point() + geom_smooth(method=NULL) +
  labs(x='Table', y='Residuals') +
  theme_bw()
d<-ggplot(fit4, aes(x=x, y=.std.resid)) +
  geom_point() + geom_smooth(method=NULL) +
  labs(x='Z', y='Residuals') +
  theme_bw()
library(gridExtra)
```

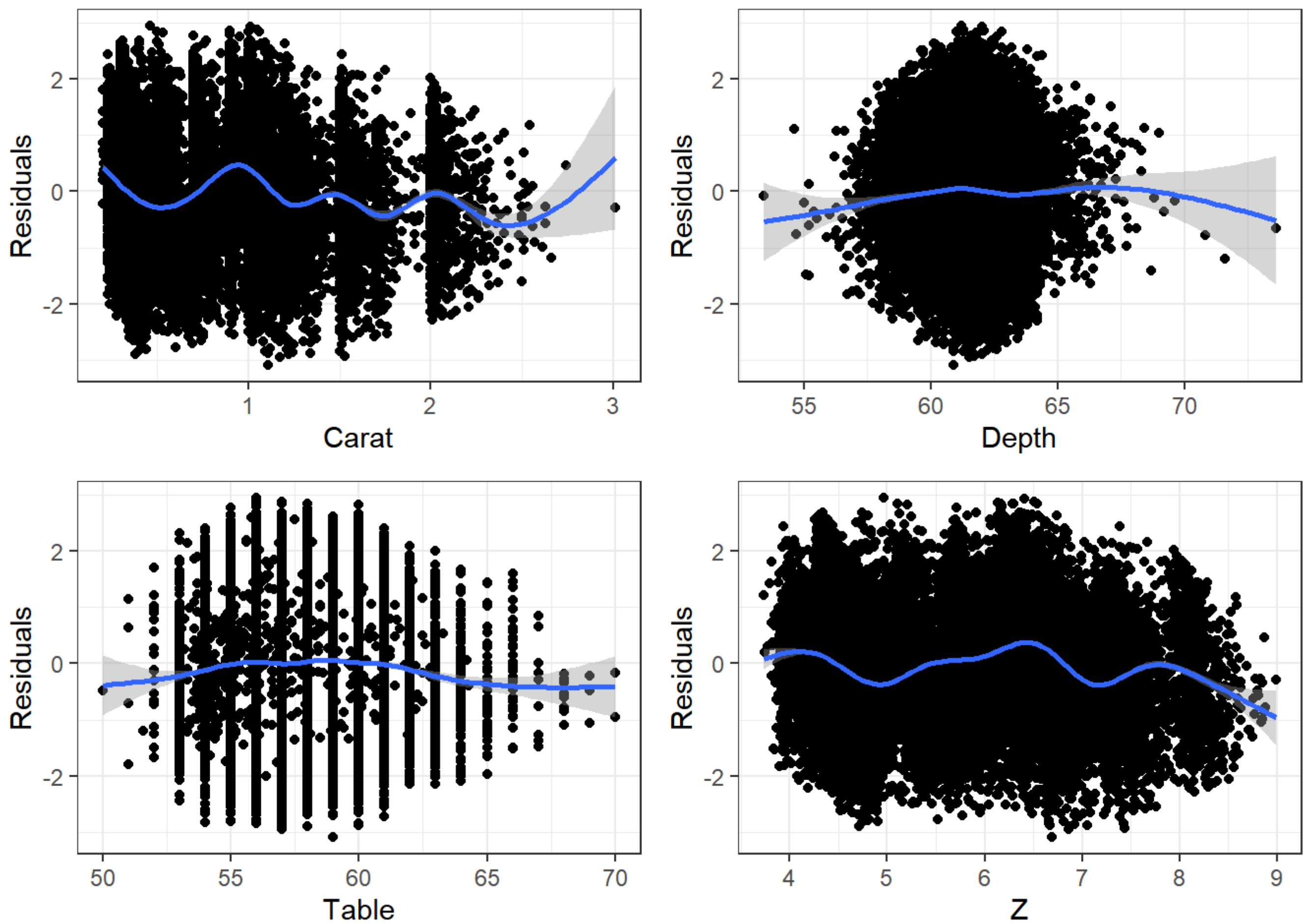
```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
grid.arrange(a,b,c,d, nrow=2)
```

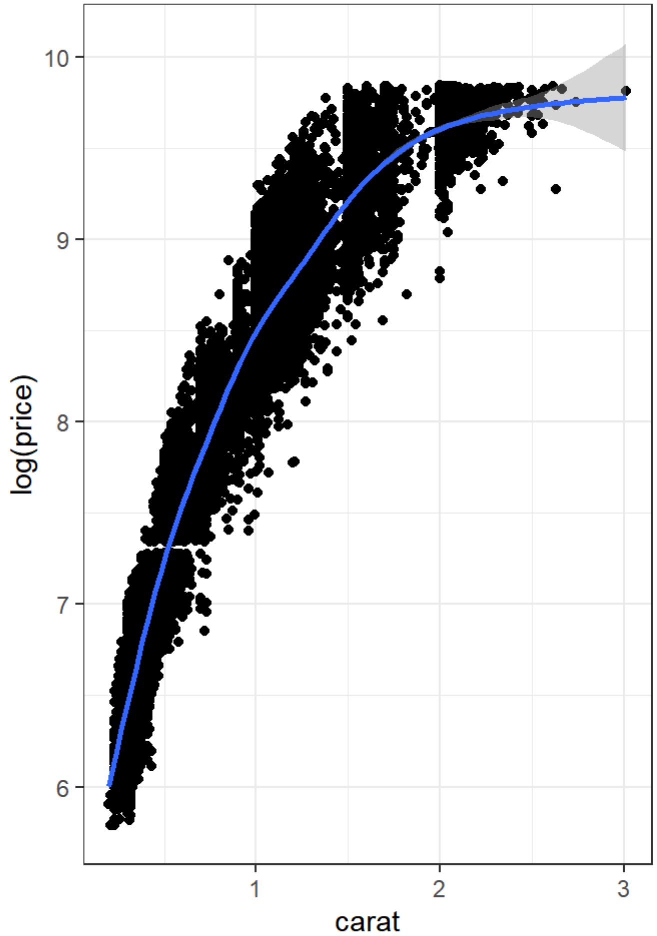
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

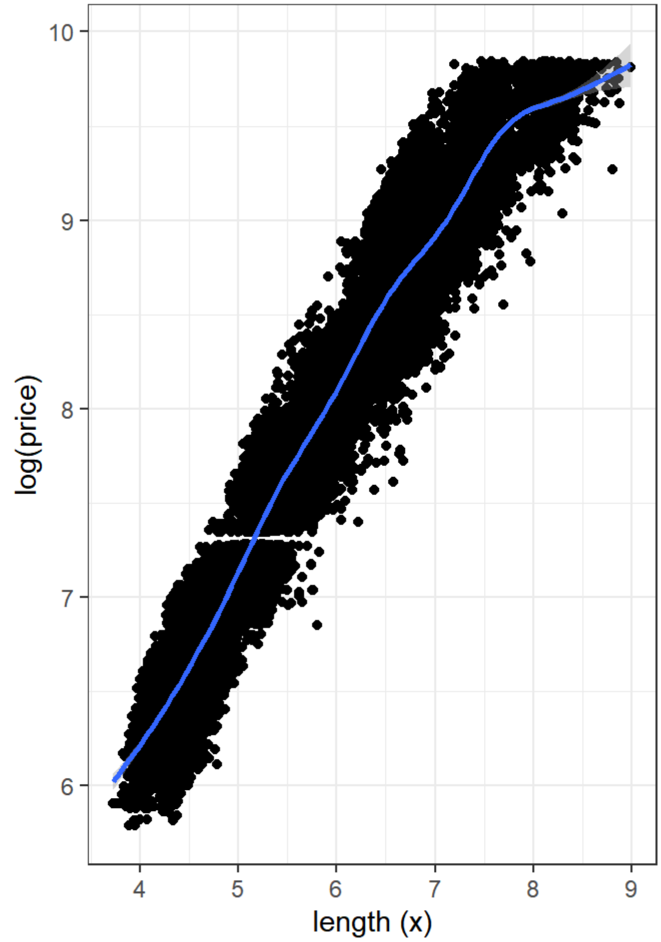
```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
# plot log(price) against each of these 2 predictors

names(fit4)[2] <- "log.price"

aa <- ggplot(fit4, aes(x=carat, y=log.price)) +
  geom_point() + geom_smooth(method=NULL) +
  labs(x='carat', y='log(price)') +
  theme_bw()
dd <- ggplot(fit4, aes(x=x, y=log.price)) +
  geom_point() + geom_smooth(method=NULL) +
  labs(x='length (x)', y='log(price)') +
  theme_bw()
grid.arrange(aa,dd, nrow=1)
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```





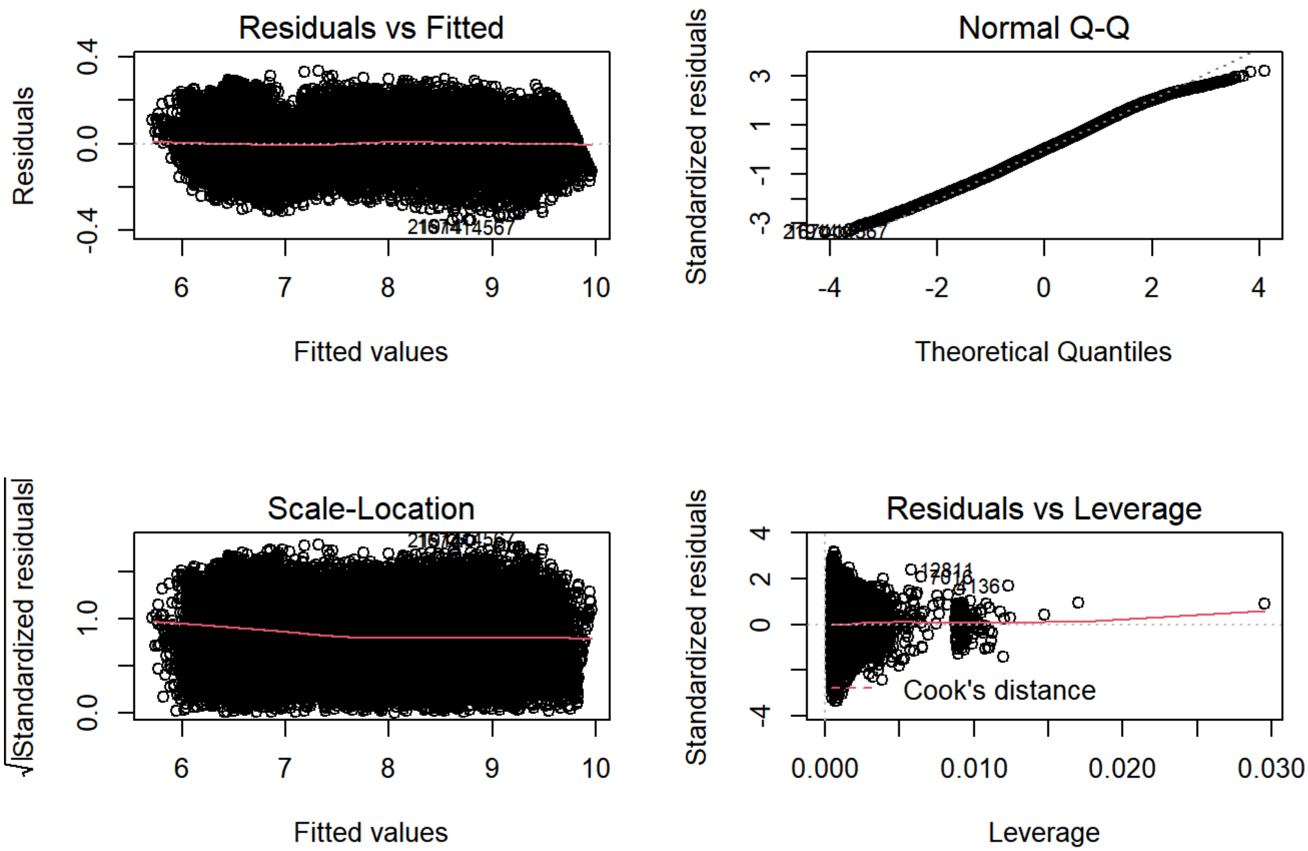
6.2 Predictor transformation

```
fit5 <- lm(log(price) ~ poly(carat, 2) + cut + color + clarity + depth + table + poly(x,
2),
                   data=new_cubiz2)

compare_rsq_summary(0.9879, 0.9879, 0.9890, 0.9890,
                    "Model comparison between before and after predictors transformatio
n")
```

Model comparison between before and after predictors transformation

Statistic	Previous Model	New Model
R-squared	0.9879	0.989
Adjusted R-squared	0.9879	0.989

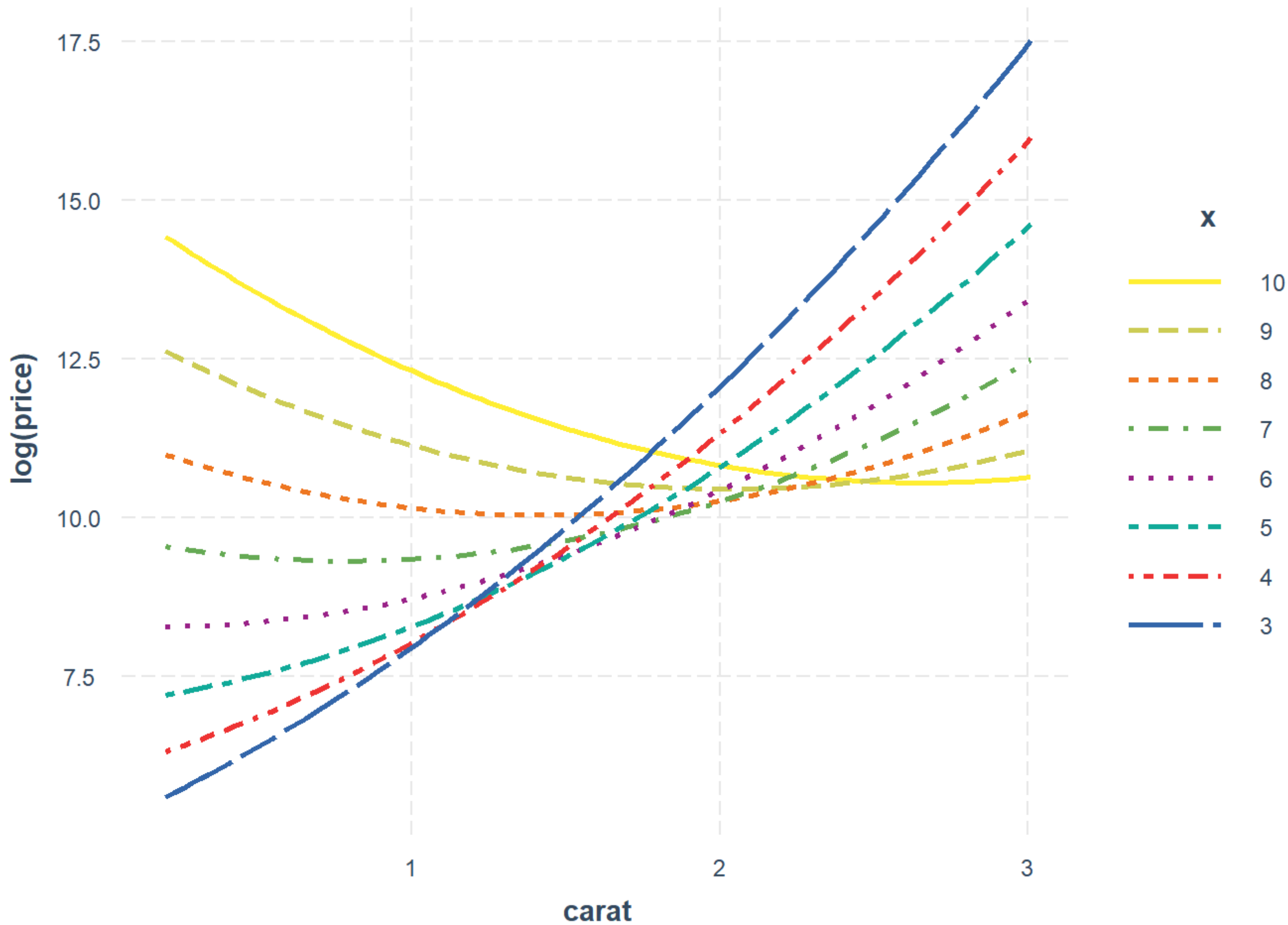
```
par(mfrow=c(2,2))
plot(fit5)
```



6.3 Interaction term

```
library(interactions)
fit6 <- lm(log(price) ~ poly(carat, 2) + cut + color + clarity + depth + table + poly(x, 2) + carat:x,
           data=new_cubiz2)

interact_plot(fit6, pred=carat, modx=x,
              modx.values = c(3,4,5,6,7,8,9,10), colors='Qual1')
```

```
## Using data new_cubiz2 from global environment. This could cause incorrect
## results if new_cubiz2 has been altered since the model was fit. You can
## manually provide the data to the "data =" argument.
```




```
compare_rsq_summary(0.9890, 0.9890, 0.9891, 0.9891,
                    "Model comparison between without and with interaction term")
```

Model comparison between without and with interaction term

Statistic	Previous Model	New Model
R-squared	0.989	0.9891
Adjusted R-squared	0.989	0.9891

6.4 Fit GAM

```
library(mgcv)
```

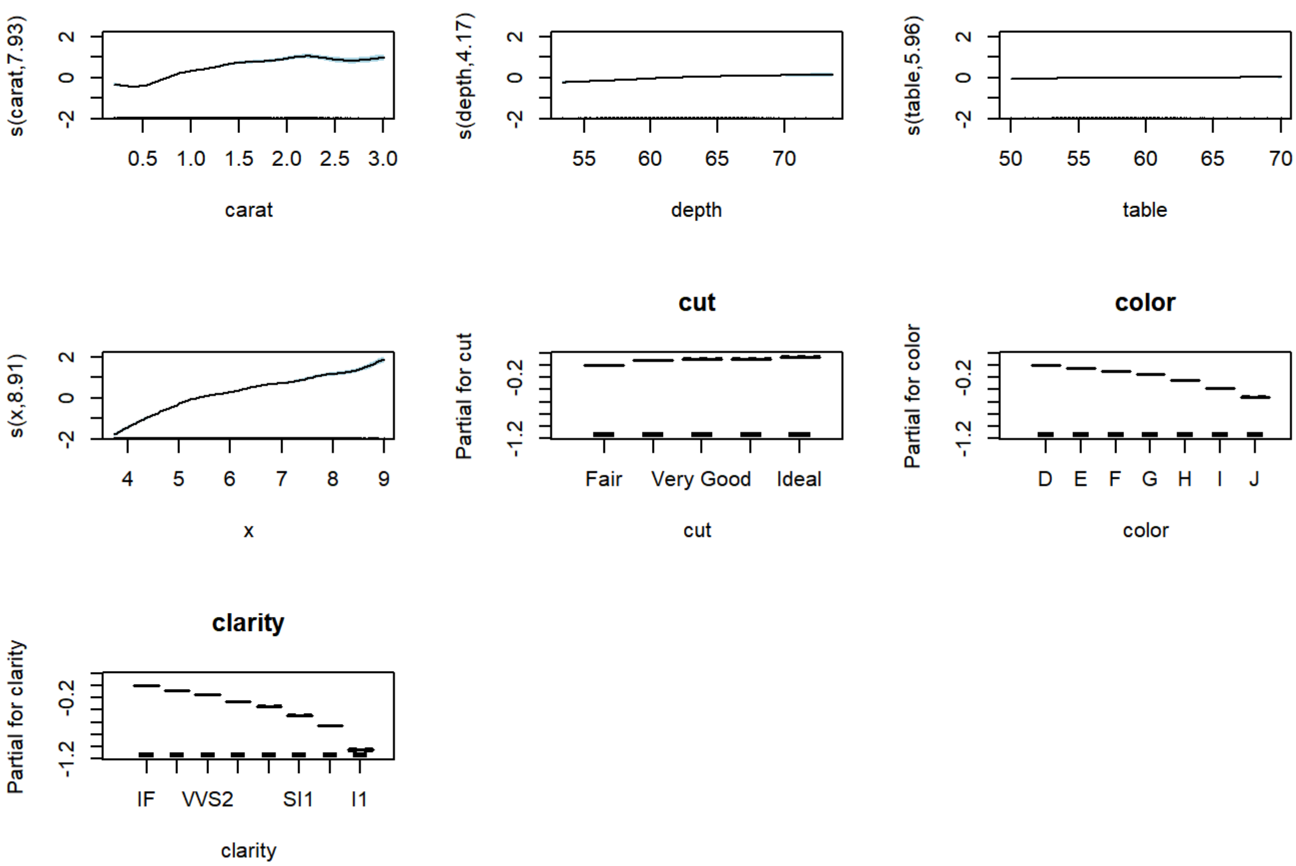
```
## Loading required package: nlme
```

```
##
## Attaching package: 'nlme'
```

```
## The following object is masked from 'package:dplyr':
##
##     collapse
```

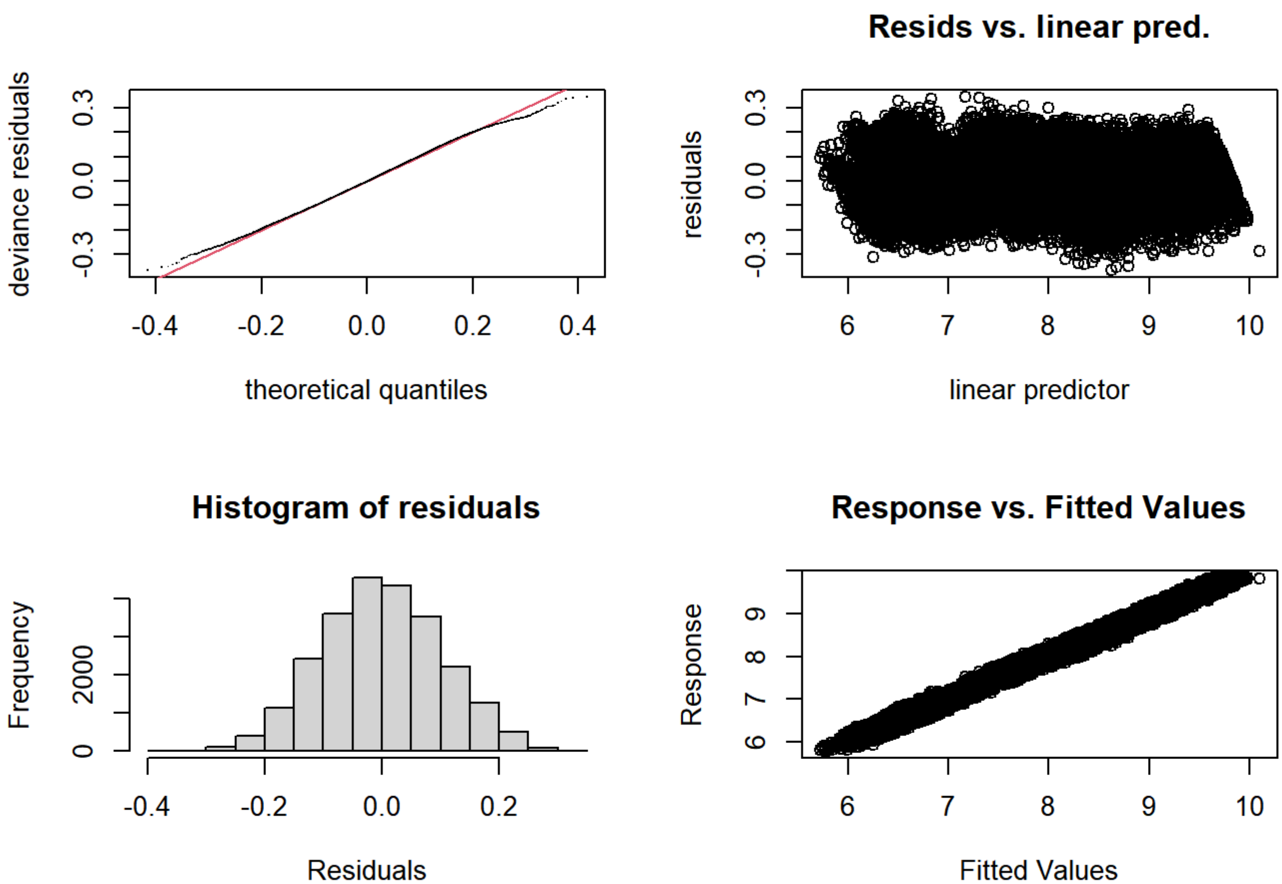
```
## This is mgcv 1.8-33. For overview type 'help("mgcv-package")'.
```

```
fit.gam <- gam(log(price) ~ s(carat) + cut + color + clarity + s(depth) +
                 s(table) + s(x), data=new_cubiz2, method="REML", select=T)

plot(fit.gam, all.terms = TRUE,rug=TRUE,
     pch=19, cex=0.65, scheme = 1, shade.col="lightblue", page=1)
```



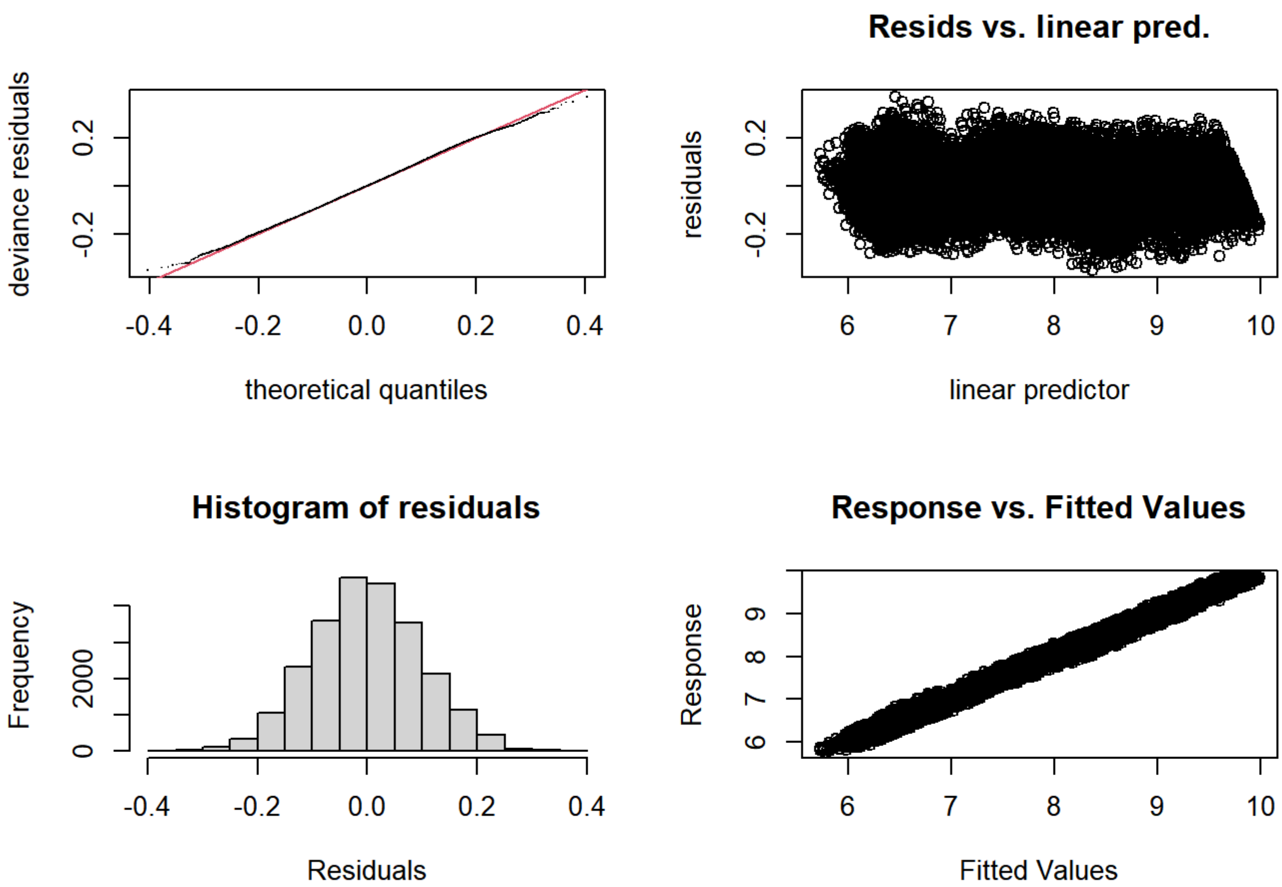
6.5 GAM model checking

```
par(mfrow=c(2,2))
gam.check(fit.gam, k.rep=2000)
```




```
##
## Method: REML   Optimizer: outer newton
## full convergence after 27 iterations.
## Gradient range [-3.430199e-05,0.001796894]
## (score -20847.27 & scale 0.0103028).
## eigenvalue range [-0.00162425,12082].
## Model rank =  54 / 54
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##              k'  edf k-index p-value
## s(carat) 9.00 7.93    0.91  <2e-16 ***
## s(depth) 9.00 4.17    1.00    0.44
## s(table) 9.00 5.96    1.00    0.54
## s(x)     9.00 8.91    0.92  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# GAM model with k=20
fit.gam2 <- gam(log(price) ~ s(carat, k=20) + cut + color + clarity + s(depth, k=20) +
                  s(table, k=20) + s(x, k=20), data=new_cubiz2, method="REML", select=T)

par(mfrow=c(2,2))
gam.check(fit.gam2, k.rep=2000)
```




```
##
## Method: REML   Optimizer: outer newton
## full convergence after 6 iterations.
## Gradient range [-0.003449634,0.003145278]
## (score -21511.86 & scale 0.00972478).
## eigenvalue range [-0.002082788,12082.02].
## Model rank =  94 / 94
##
## Basis dimension (k) checking results. Low p-value (k-index<1) may
## indicate that k is too low, especially if edf is close to k'.
##
##              k'   edf k-index p-value
## s(carat) 19.00 15.82    0.94  <2e-16 ***
## s(depth) 19.00  5.85    1.01    0.80
## s(table) 19.00  7.81    1.01    0.84
## s(x)     19.00 17.58    1.00    0.54
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
sum.gam <- summary(fit.gam)
sum.gam2 <- summary(fit.gam2)

gam_edf_carat <- sum.gam$edf[1]
gam_edf_depth <- sum.gam$edf[2]
gam_edf_table <- sum.gam$edf[3]
gam_edf_x <- sum.gam$edf[4]
gam2_edf_carat <- sum.gam2$edf[1]
gam2_edf_depth <- sum.gam2$edf[2]
gam2_edf_table <- sum.gam2$edf[3]
gam2_edf_x <- sum.gam2$edf[4]

Predictors <- c("s(carat)", "s(depth)", "s(table)","s(x)")
Model.GAM <- c(gam_edf_carat, gam_edf_depth, gam_edf_table, gam_edf_x)
Model.GAM2 <- c(gam2_edf_carat, gam2_edf_depth, gam2_edf_table, gam2_edf_x)
mod.summary <- data.frame(Predictors, Model.GAM, Model.GAM2)
names(mod.summary) <- c("Predictors", "Model fit.gam (k=9)", "Model fit.gam2 (k=20)")

pander(mod.summary, digits=3,caption="Model comparison between k = 9 and k = 20")
```

Model comparison between k = 9 and k = 20

Predictors	Model fit.gam (k=9)	Model fit.gam2 (k=20)
s(carat)	7.93	15.8
s(depth)	4.17	5.85
s(table)	5.96	7.81
s(x)	8.91	17.6

7. Simplify the model in stage: model selection

7.1 AIC for Linear Model

```
fit6 <- lm(log(price) ~ poly(carat, 2) + cut + color + clarity + depth + table + poly(x,
2) + carat:x,
           data=new_cubiz2)

step(fit6, direction="both")
```

```
## Start:  AIC=-108849.6
## log(price) ~ poly(carat, 2) + cut + color + clarity + depth +
##     table + poly(x, 2) + carat:x
##
##                  Df Sum of Sq    RSS     AIC
## <none>                        267.74 -108850
## - table           1      0.89 268.63 -108771
## - carat:x         1      2.20 269.94 -108654
## - depth           1      8.99 276.73 -108053
## - poly(carat, 2)  2     12.53 280.27 -107748
## - cut             4     14.19 281.94 -107609
## - poly(x, 2)      2     98.58 366.32 -101273
## - color           6    390.99 658.73  -87091
## - clarity         7    658.97 926.72  -78839
```

```
##
## Call:
## lm(formula = log(price) ~ poly(carat, 2) + cut + color + clarity +
##     depth + table + poly(x, 2) + carat:x, data = new_cubiz2)
##
## Coefficients:
##     (Intercept)  poly(carat, 2)1  poly(carat, 2)2          cutGood
##        10.14655        411.81071         25.75743          0.09344
##    cutVery Good       cutPremium         cutIdeal           colorE
##         0.12054          0.13388          0.16751         -0.05289
##          colorF           colorG           colorH           colorI
##        -0.09376         -0.15411         -0.25223         -0.37856
##          colorJ      clarityVVS1      clarityVVS2       clarityVS1
##        -0.52666         -0.08878         -0.15502         -0.27043
##      clarityVS2       claritySI1       claritySI2        clarityI1
##        -0.34354         -0.48998         -0.65766         -1.07009
##           depth            table     poly(x, 2)1      poly(x, 2)2
##         0.02696          0.00396        222.66959         17.44799
##         carat:x
##        -0.79842
```

```
fit6.AIC.A <- -108771
fit6.AIC.B <- -108850
fit6.AIC.difference <- fit6.AIC.A - fit6.AIC.B
fit6.AIC.difference
```

```
## [1] 79
```

7.2 BIC for Linear Model

```
step(fit6, direction="both", k=log(nrow(new_cubiz2)))
```

```
## Start:  AIC=-108647.3
## log(price) ~ poly(carat, 2) + cut + color + clarity + depth +
##     table + poly(x, 2) + carat:x
##
##                  Df Sum of Sq    RSS     AIC
## <none>                        267.74 -108647
## - table           1      0.89 268.63 -108577
## - carat:x         1      2.20 269.94 -108460
## - depth           1      8.99 276.73 -107859
## - poly(carat, 2)  2     12.53 280.27 -107562
## - cut             4     14.19 281.94 -107439
## - poly(x, 2)      2     98.58 366.32 -101087
## - color           6    390.99 658.73  -86937
## - clarity         7    658.97 926.72  -78693
```

```
##
## Call:
## lm(formula = log(price) ~ poly(carat, 2) + cut + color + clarity +
##     depth + table + poly(x, 2) + carat:x, data = new_cubiz2)
##
## Coefficients:
##     (Intercept)  poly(carat, 2)1  poly(carat, 2)2          cutGood
##        10.14655        411.81071         25.75743          0.09344
##    cutVery Good       cutPremium          cutIdeal           colorE
##         0.12054          0.13388          0.16751         -0.05289
##          colorF           colorG            colorH           colorI
##        -0.09376         -0.15411          -0.25223         -0.37856
##          colorJ      clarityVVS1       clarityVVS2       clarityVS1
##        -0.52666         -0.08878          -0.15502         -0.27043
##      clarityVS2       claritySI1        claritySI2        clarityI1
##        -0.34354         -0.48998          -0.65766         -1.07009
##           depth            table      poly(x, 2)1      poly(x, 2)2
##         0.02696          0.00396         222.66959         17.44799
##         carat:x
##        -0.79842
```

```
fit6.BIC.A <- -108577
fit6.BIC.B <- -108647
fit6.BIC.difference <- fit6.BIC.A - fit6.BIC.B
fit6.BIC.difference
```

```
## [1] 70
```

7.3 Mallow’s Cp for Linear Model

```r
par(mfrow=c(1,1))

library(leaps)
x <- model.matrix(fit6)[,-1]
y <- log(new_cubiz2$price)
models <- leaps(x,y)
library(faraway)
```

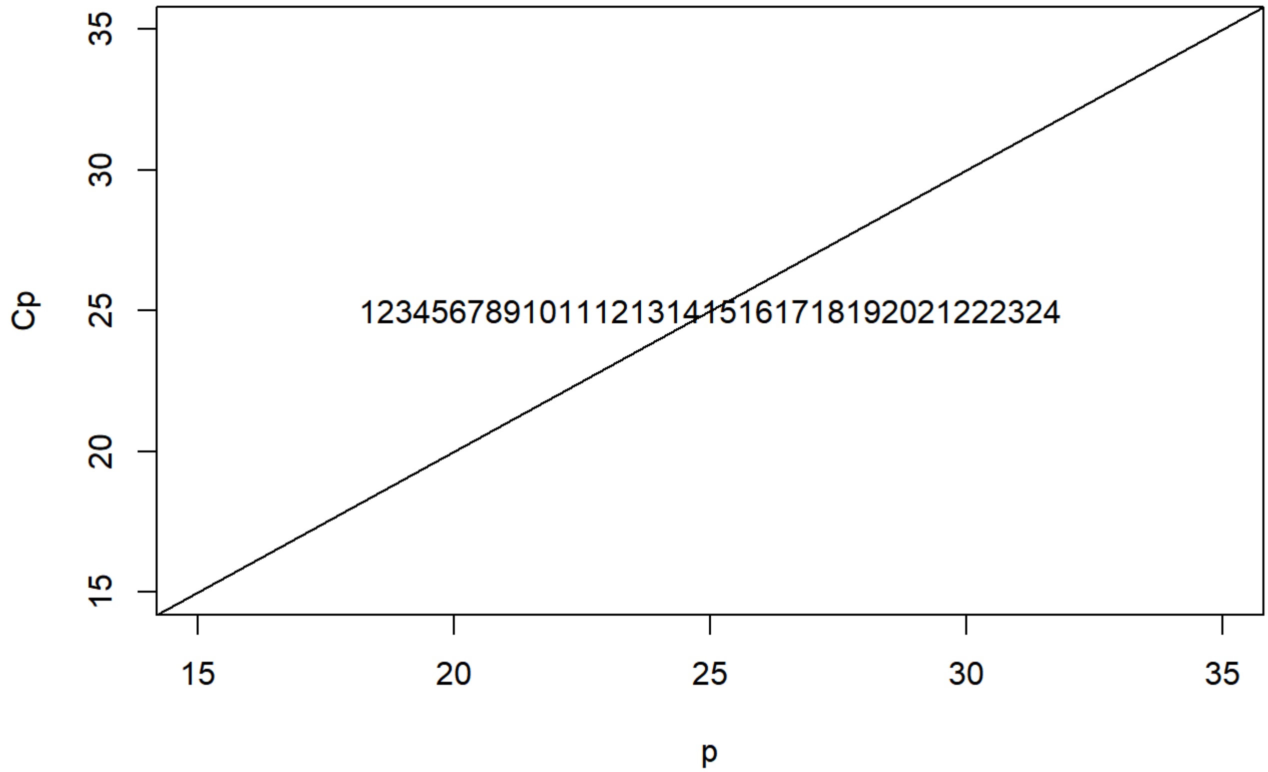
```
## Registered S3 methods overwritten by 'lme4':
##   method                          from
##   cooks.distance.influence.merMod car
##   influence.merMod                car
##   dfbeta.influence.merMod         car
##   dfbetas.influence.merMod        car
```

```
##
## Attaching package: 'faraway'
```

```
## The following objects are masked from 'package:car':
##
##     logit, vif
```

```
## The following object is masked from 'package:psych':
##
##     logit
```

```r
Cpplot(models)
```




```
#find the lowest Cp position in the output
idx <- which(models$Cp == min(models$Cp))
cols_to_filter <- models$which[idx,]
winner <- x[idx,]
pander(filter(as.data.frame(winner), cols_to_filter), caption = "Chosen predictors by Ma
llow's Cp")
```

Chosen predictors by Mallow’s Cp

	winner
poly(carat, 2)1	-0.005165
poly(carat, 2)2	0.002412
cutGood	0
cutVery Good	0
cutPremium	0
cutIdeal	1
colorE	0
colorF	0
colorG	0

	winner
colorH	0
colorI	0
colorJ	0
clarityVVS1	0
clarityVVS2	0
clarityVS1	0
clarityVS2	1
claritySI1	0
claritySI2	0
clarityI1	0
depth	62.8
table	55
poly(x, 2)1	-0.005577
poly(x, 2)2	0.0005876
carat:x	1.939

7.4 Model selection for GAM

```
fit.gam2 <- gam(log(price) ~ s(carat, k=20) + cut + color + clarity + s(depth, k=20) +
                  s(table, k=20) + s(x, k=20), data=new_cubiz2, method="REML", select=T)
summary(fit.gam2)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## log(price) ~ s(carat, k = 20) + cut + color + clarity + s(depth,
##     k = 20) + s(table, k = 20) + s(x, k = 20)
##
## Parametric coefficients:
##                Estimate Std. Error  t value Pr(>|t|)
## (Intercept)    8.172122   0.009078  900.256   <2e-16 ***
## cutGood        0.071434   0.007598    9.402   <2e-16 ***
## cutVery Good   0.093132   0.008114   11.478   <2e-16 ***
## cutPremium     0.105691   0.008251   12.809   <2e-16 ***
## cutIdeal       0.137290   0.008343   16.455   <2e-16 ***
## colorE        -0.052562   0.002341  -22.448   <2e-16 ***
## colorF        -0.093756   0.002376  -39.458   <2e-16 ***
## colorG        -0.155312   0.002323  -66.864   <2e-16 ***
## colorH        -0.251438   0.002492 -100.917   <2e-16 ***
## colorI        -0.374467   0.002800 -133.730   <2e-16 ***
## colorJ        -0.516728   0.003581 -144.304   <2e-16 ***
## clarityVVS1   -0.083993   0.004482  -18.738   <2e-16 ***
## clarityVVS2   -0.148547   0.004299  -34.557   <2e-16 ***
## clarityVS1    -0.265580   0.004113  -64.574   <2e-16 ***
## clarityVS2    -0.343711   0.004038  -85.129   <2e-16 ***
## claritySI1    -0.488679   0.004080 -119.781   <2e-16 ***
## claritySI2    -0.659342   0.004250 -155.143   <2e-16 ***
## clarityI1     -1.051384   0.009959 -105.572   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##              edf Ref.df       F p-value
## s(carat) 15.821     19  42.827  <2e-16 ***
## s(depth)  5.850     19  28.952  <2e-16 ***
## s(table)  7.807     19   7.329  <2e-16 ***
## s(x)     17.575     19 158.369  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =   0.99   Deviance explained =   99%
## -REML = -21512  Scale est. = 0.0097248  n = 24182
```

7.6 Fit LM using GAM

```
library(mgcv)
fit6.lm <- gam(log(price) ~ poly(carat, 2) + cut + color + clarity + depth + table + pol
y(x, 2) + carat:x,
               data=new_cubiz2, method="REML")
```

7.7 Comparing AIC & BIC between Linear Model and GAM

```
pander(AIC(fit6.lm, fit.gam2), caption="")
```

	df	AIC
fit6.lm	26	-40222
fit.gam2	67.67	-43341

```
pander(BIC(fit6.lm, fit.gam2), caption="")
```

	df	BIC
fit6.lm	26	-40012
fit.gam2	67.67	-42794

8. Using cross-validation to avoid over-fitting

```
predict.regsubsets <- function(object,newdata,id,...){
  form <- as.formula(object$call[[2]])
  mat <- model.matrix(form,newdata)
  coefi <- coef(object,id=id)
  xvars <- names(coefi)
  mat[,xvars]%*%coefi
}

k <- 10
set.seed(123)
folds <- sample(1:k, nrow(new_cubiz2), replace=TRUE)
cv.errors <- matrix(NA,k,21, dimnames=list(NULL,paste(1:21)))

for(j in 1:k) {
  best.fit <- regsubsets(price~.,data=new_cubiz2[folds!=j, ],nvmax=21)

  for(i in 1:21) {
    pred <- predict.regsubsets(best.fit,new_cubiz2[folds==j,],id=i)
    cv.errors[j,i] <- mean((new_cubiz2$price[folds==j]-pred)^2)
  }
}

mean.cv.errors <- apply(cv.errors,2,mean)
pander(mean.cv.errors, caption="Number of predictors and corresponding test MSE")
```

Table continues below

1	2	3	4	5	6	7	8
1881891	1638174	1438461	1276590	1160683	1051624	977074	921645

Table continues below

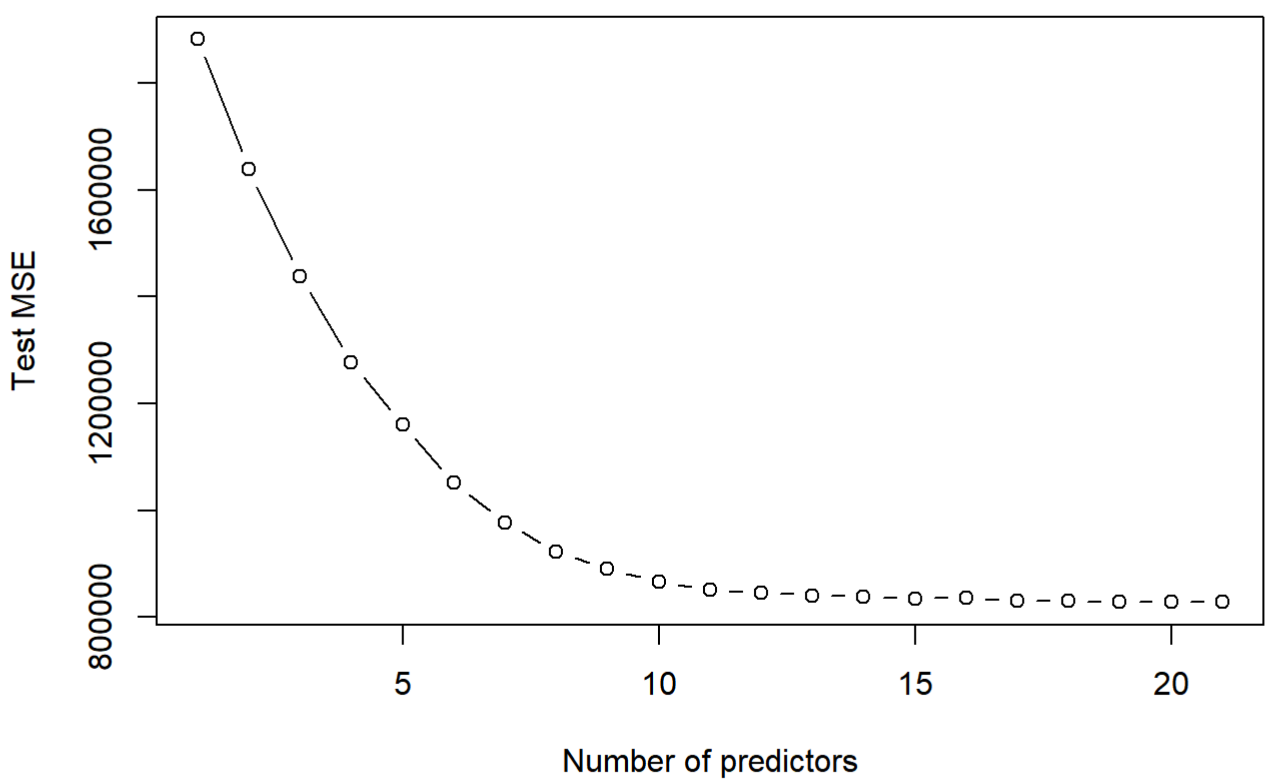
9	10	11	12	13	14	15	16	17
890427	866661	850997	846352	840693	837839	834821	835773	830979

18	19	20	21
829962	828235	828373	828198

```
which.min(mean.cv.errors)
```

```
## 21
## 21
```

```
par(mfrow=c(1,1))
plot(mean.cv.errors, type="b", xlab="Number of predictors", ylab="Test MSE")
```




```
reg.best <- regsubsets(price~.,data=new_cubiz2 ,nvmax=21)
coef(reg.best,21)
```

```
##   (Intercept)        carat      cutGood cutVery Good   cutPremium     cutIdeal 
##   16203.45065  14519.23200    458.89641    531.18968    643.78817    706.52124 
##        colorE       colorF       colorG       colorH       colorI       colorJ 
##    -155.70555   -197.30904   -341.16656   -881.58316  -1486.49335  -2483.96208 
##   clarityVVS1  clarityVVS2   clarityVS1   clarityVS2   claritySI1   claritySI2 
##    -102.56834    -77.85549   -323.71923   -623.43608  -1208.99513  -2159.17566 
##     clarityI1        depth        table            x 
##   -4149.53042   -126.93391    -35.50522  -2283.41917
```

9. Shrinkage methods

9.1 Ridge regression with cross-validation

```
set.seed(123)

# split 80% of the data to train dataset and 20% to test dataset
train_index <- sample(length(new_cubiz2$carat), length(new_cubiz2$carat)*0.8)
train <- new_cubiz2[train_index,]
test <- new_cubiz2[-train_index,]
x_train <- model.matrix(price~.,train)[,c(-1)]
y_train <- train$price
x_test <- model.matrix(price~.,test)[,c(-1)]
y_test <- test$price
```

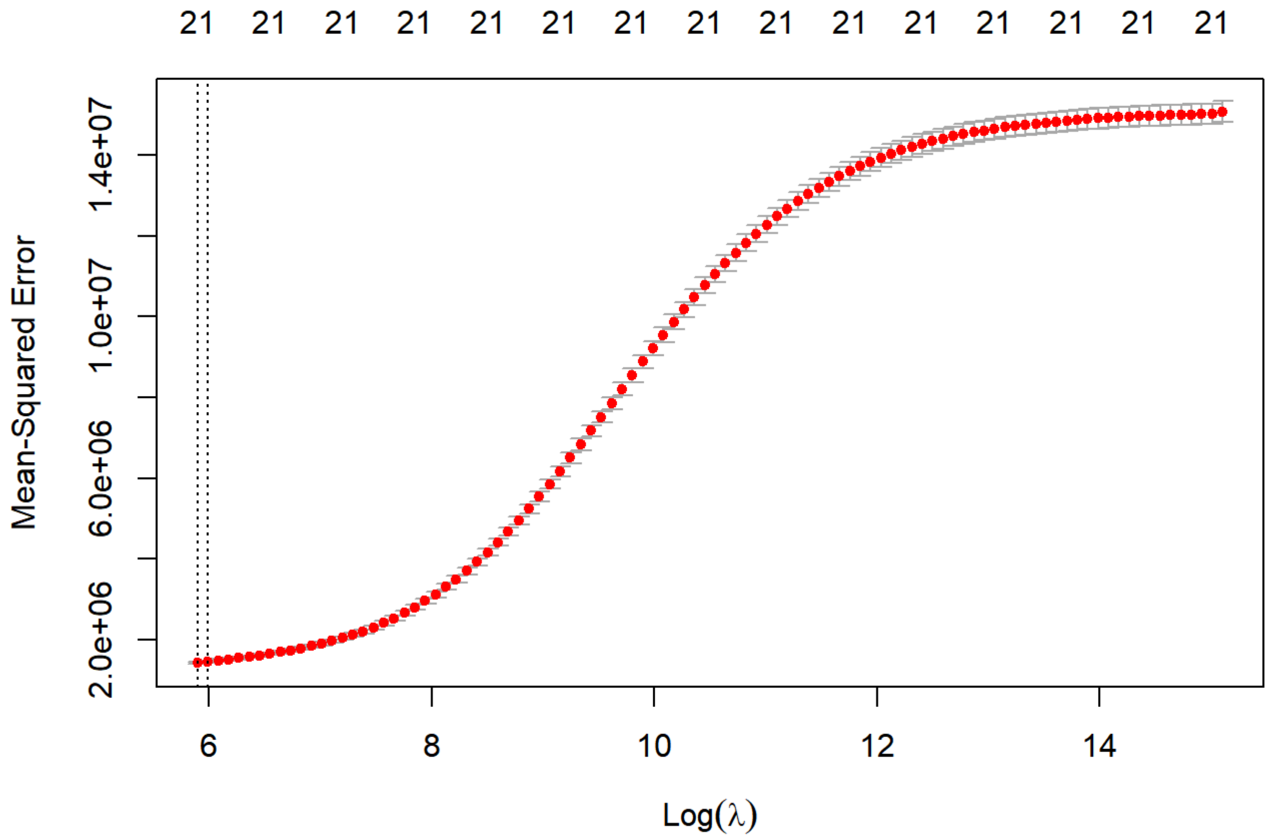
```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.0.5
```

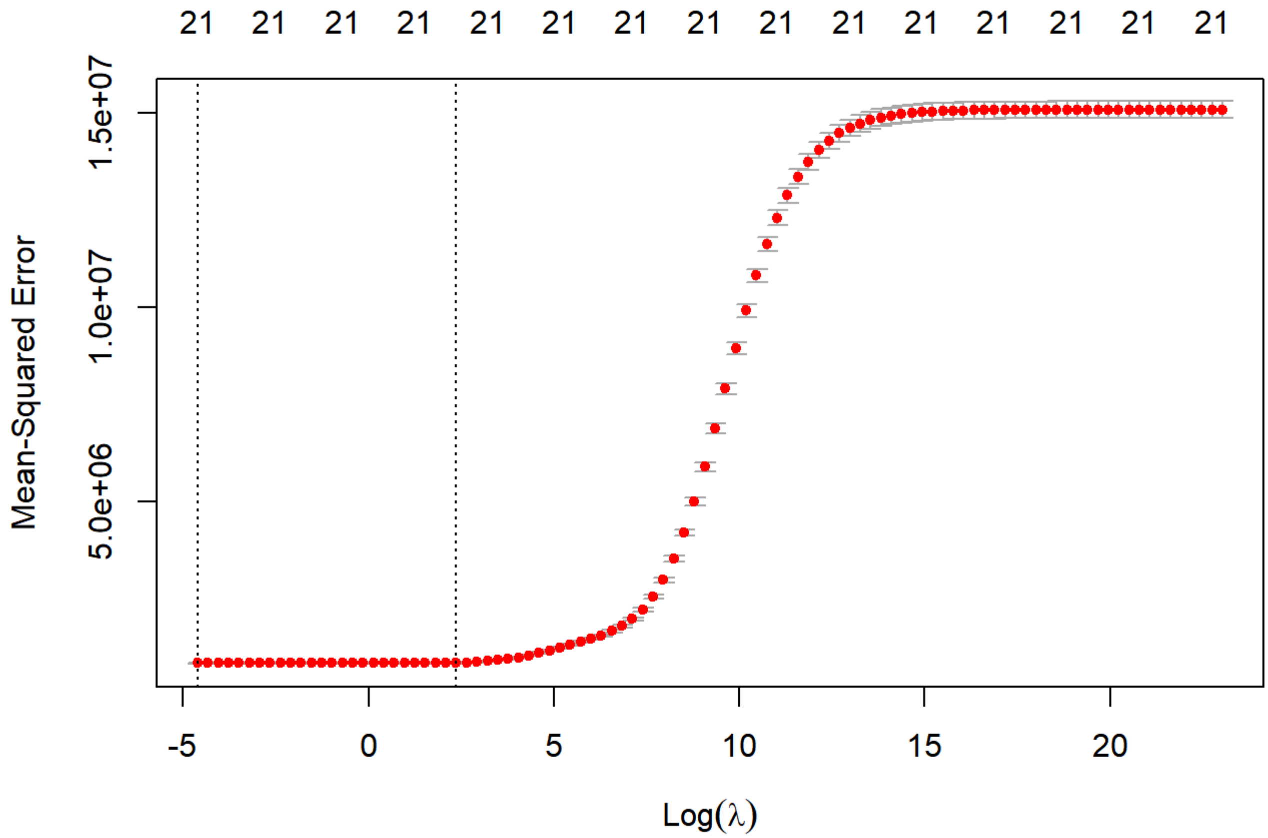
```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-1
```

```
cv.out.ridge <- cv.glmnet(x_train,y_train,alpha=0)
plot(cv.out.ridge)
```




```
# expand the range of plot
grid <- 10^seq(10,-2,length=100)

cv.out.ridge <- cv.glmnet(x_train,y_train, alpha=0, lambda=grid)

plot(cv.out.ridge)
```




```
bestlam.ridge <- cv.out.ridge$lambda.min
bestlam.ridge
```

```
## [1] 0.01
```

```
log(bestlam.ridge)
```

```
## [1] -4.60517
```

```
out.ridge <- glmnet(x_train,y_train,alpha=0)
predict(out.ridge,type="coefficients",s=bestlam.ridge)
```

```
## 22 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept)  -5730.611997
## carat         5771.942399
## cutGood        -79.166628
## cutVery Good   125.626106
## cutPremium     197.352068
## cutIdeal       241.123529
## colorE          34.897087
## colorF          19.543406
## colorG         -45.612737
## colorH        -445.478542
## colorI        -768.865549
## colorJ       -1502.668306
## clarityVVS1    553.680396
## clarityVVS2    535.210436
## clarityVS1     196.604310
## clarityVS2     -15.269256
## claritySI1    -620.470078
## claritySI2   -1315.787533
## clarityI1    -3144.956471
## depth            7.804591
## table          -19.553686
## x             1056.980036
```

```
lam1se.ridge <- cv.out.ridge$lambda.1se
lam1se.ridge
```

```
## [1] 10.72267
```

```
log(lam1se.ridge)
```

```
## [1] 2.37236
```

```
predict(out.ridge,type="coefficients",s=lam1se.ridge)
```

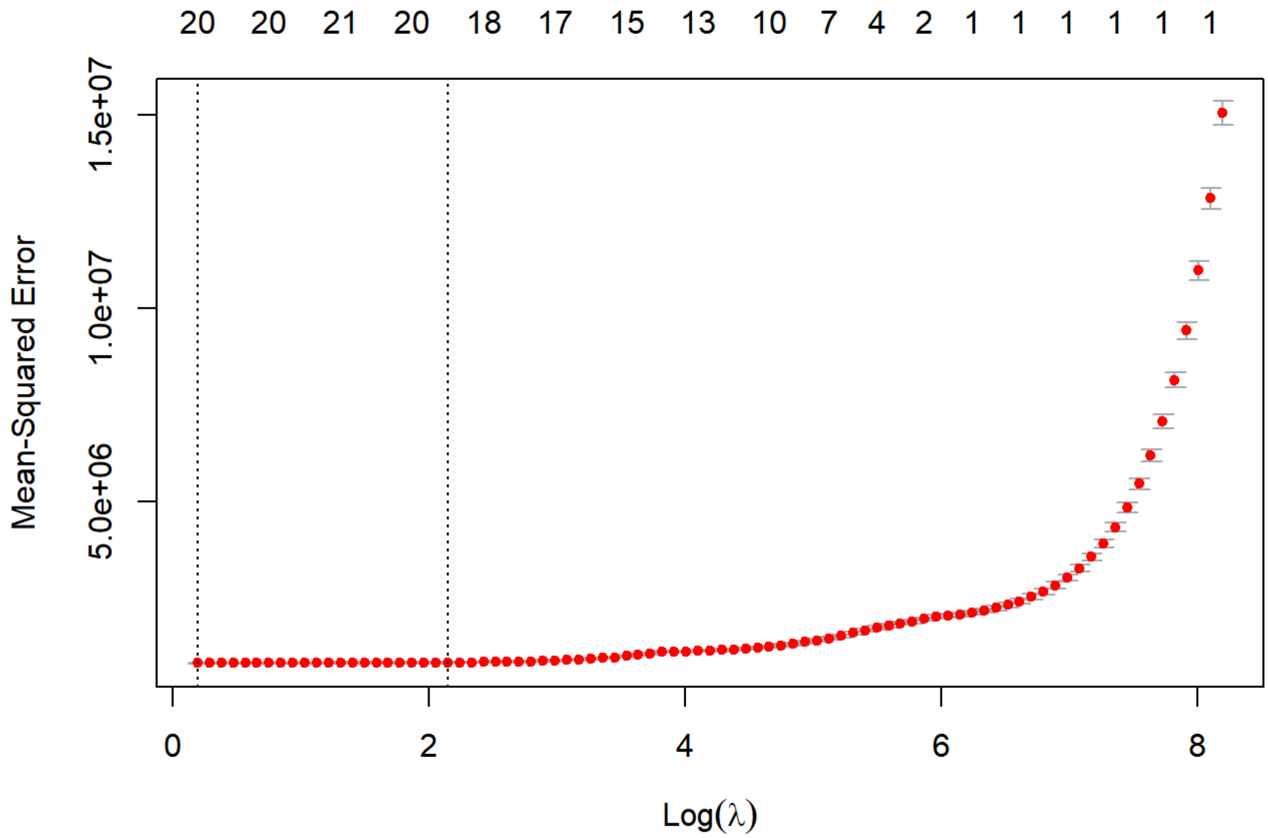
```
## 22 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept)  -5730.611997
## carat         5771.942399
## cutGood        -79.166628
## cutVery Good   125.626106
## cutPremium     197.352068
## cutIdeal       241.123529
## colorE          34.897087
## colorF          19.543406
## colorG         -45.612737
## colorH        -445.478542
## colorI        -768.865549
## colorJ       -1502.668306
## clarityVVS1    553.680396
## clarityVVS2    535.210436
## clarityVS1     196.604310
## clarityVS2     -15.269256
## claritySI1    -620.470078
## claritySI2   -1315.787533
## clarityI1    -3144.956471
## depth            7.804591
## table          -19.553686
## x             1056.980036
```

```
# test error
pred_ridge <- predict(out.ridge, s = bestlam.ridge, newx = x_test)
mse_ridge <- mean((pred_ridge - y_test)^2)
mse_ridge
```

```
## [1] 1481851
```

9.2 LASSO regression with cross-validation

```
cv.out.lasso=cv.glmnet(x_train,y_train,alpha=1)
plot(cv.out.lasso)
```




```
bestlam.lasso <- cv.out.lasso$lambda.min
bestlam.lasso
```

```
## [1] 1.21825
```

```
log(bestlam.lasso)
```

```
## [1] 0.1974158
```

```
out.lasso <- glmnet(x_train,y_train,alpha=1)
predict(out.lasso,type="coefficients",s=bestlam.lasso)
```

```
## 22 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept)  15667.41106
## carat        14257.57588
## cutGood        279.50038
## cutVery Good   381.55328
## cutPremium     494.18281
## cutIdeal       545.82374
## colorE        -125.02803
## colorF        -158.99511
## colorG        -303.42006
## colorH        -846.42330
## colorI       -1430.59407
## colorJ       -2431.04682
## clarityVVS1    -23.78502
## clarityVVS2      .
## clarityVS1    -240.02335
## clarityVS2    -532.12902
## claritySI1   -1119.54472
## claritySI2   -2063.79938
## clarityI1    -4047.89578
## depth         -123.96240
## table          -34.79833
## x            -2186.99564
```

```
lam1se.lasso <- cv.out.lasso$lambda.1se
lam1se.lasso
```

```
## [1] 8.594516
```

```
log(lam1se.lasso)
```

```
## [1] 2.151124
```

```
predict(out.lasso,type="coefficients",s=lam1se.lasso)
```

```
## 22 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept)  13871.15881
## carat        13345.15349
## cutGood        -68.23124
## cutVery Good     .
## cutPremium     102.19280
## cutIdeal       167.75976
## colorE           .
## colorF         -23.08983
## colorG        -153.10124
## colorH        -690.25551
## colorI       -1251.67084
## colorJ       -2226.81029
## clarityVVS1    186.73522
## clarityVVS2    217.53967
## clarityVS1       .
## clarityVS2    -281.15241
## claritySI1    -874.65844
## claritySI2   -1802.85707
## clarityI1    -3751.08980
## depth         -115.53707
## table          -35.39700
## x            -1830.92083
```

```
pred_lasso <- predict(out.lasso, s=bestlam.lasso, newx=x_test)
mse_lasso <- mean((pred_lasso - y_test)^2)
mse_lasso
```

```
## [1] 822217.1
```

```
Methods <- c("Ridge regression", "Cross-validation", "LASSO regression")
TestMSE <- c(mse_ridge, min(mean.cv.errors), mse_lasso)
testMSE_sum <- data.frame(Methods, TestMSE)

pander(testMSE_sum,caption="Prediction settings comparison")
```

Prediction settings comparison

Methods	TestMSE
Ridge regression	1481851
Cross-validation	828198
LASSO regression	822217

10. Models summary

```r
# fit1
summary(fit1)
```

```
##
## Call:
## lm(formula = price ~ carat + cut + color + clarity + depth +
##     table + x + y + z, data = cubiz)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15855.8   -593.8   -184.8    379.8  10045.9
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7428.165    653.997  11.358  < 2e-16 ***
## carat        11377.271     71.287 159.597  < 2e-16 ***
## cutGood        544.803     48.747  11.176  < 2e-16 ***
## cutVery Good   689.755     46.889  14.710  < 2e-16 ***
## cutPremium     715.194     46.860  15.262  < 2e-16 ***
## cutIdeal       803.884     48.751  16.490  < 2e-16 ***
## colorE        -209.459     25.751  -8.134 4.33e-16 ***
## colorF        -273.390     26.081 -10.482  < 2e-16 ***
## colorG        -476.060     25.469 -18.692  < 2e-16 ***
## colorH        -990.395     27.189 -36.427  < 2e-16 ***
## colorI       -1504.229     30.351 -49.562  < 2e-16 ***
## colorJ       -2368.007     37.236 -63.594  < 2e-16 ***
## clarityVVS1   -285.627     46.847  -6.097 1.10e-09 ***
## clarityVVS2   -334.450     44.820  -7.462 8.78e-14 ***
## clarityVS1    -710.380     42.828 -16.587  < 2e-16 ***
## clarityVS2   -1015.761     41.850 -24.272  < 2e-16 ***
## claritySI1   -1624.864     42.164 -38.536  < 2e-16 ***
## claritySI2   -2606.039     43.849 -59.432  < 2e-16 ***
## clarityI1    -5335.308     73.505 -72.585  < 2e-16 ***
## depth          -57.671      7.995  -7.214 5.59e-13 ***
## table          -28.993      4.216  -6.877 6.23e-12 ***
## x             -944.316     60.390 -15.637  < 2e-16 ***
## y               23.270     22.866   1.018  0.30885
## z             -243.373     88.866  -2.739  0.00617 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1133 on 26246 degrees of freedom
## Multiple R-squared:  0.9208, Adjusted R-squared:  0.9207
## F-statistic: 1.326e+04 on 23 and 26246 DF,  p-value: < 2.2e-16
```

```r
# fit2
pander(summary(fit2))
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-2.321	0.09326	-24.88	4.023e-135

	Estimate	Std. Error	t value	Pr(>|t|)
carat	-0.7655	0.01017	-75.3	0
cutGood	0.09547	0.006952	13.73	8.943e-43
cutVery Good	0.1223	0.006687	18.29	3.008e-74
cutPremium	0.107	0.006683	16.01	1.964e-57
cutIdeal	0.1555	0.006952	22.36	9.548e-110
colorE	-0.05747	0.003672	-15.65	6.001e-55
colorF	-0.0901	0.003719	-24.22	3.206e-128
colorG	-0.1563	0.003632	-43.04	0
colorH	-0.2585	0.003877	-66.68	0
colorI	-0.3824	0.004328	-88.35	0
colorJ	-0.5212	0.00531	-98.15	0
clarityVVS1	-0.09536	0.006681	-14.27	4.709e-46
clarityVVS2	-0.1635	0.006392	-25.58	1.37e-142
clarityVS1	-0.2845	0.006108	-46.58	0
clarityVS2	-0.3557	0.005968	-59.6	0
claritySI1	-0.4981	0.006013	-82.84	0
claritySI2	-0.6668	0.006253	-106.6	0
clarityI1	-1.108	0.01048	-105.7	0
depth	0.05157	0.00114	45.23	0
table	0.01006	0.0006012	16.74	1.537e-62
x	1.217	0.008612	141.4	0
y	0.006207	0.003261	1.904	0.05698
z	0.1062	0.01267	8.384	5.389e-17

Fitting linear model: log(price) ~ carat + cut + color + clarity + depth + table + x + y + z

Observations	Residual Std. Error	R^2	Adjusted R^2
26270	0.1615	0.9749	0.9749

```
# fit3
pander(summary(fit3))
```

	Estimate	Std. Error	t value	Pr(>|t|)

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-1.422	0.1552	-9.165	5.326e-20
carat	-0.924	0.008793	-105.1	0
cutGood	0.08238	0.006063	13.59	6.588e-42
cutVery Good	0.1082	0.00591	18.31	1.957e-74
cutPremium	0.1166	0.00583	20	2.43e-88
cutIdeal	0.15	0.006053	24.79	4.477e-134
colorE	-0.05362	0.002799	-19.16	3.208e-81
colorF	-0.09156	0.002839	-32.25	1.088e-223
colorG	-0.1535	0.002776	-55.27	0
colorH	-0.2514	0.002965	-84.79	0
colorI	-0.3776	0.003314	-114	0
colorJ	-0.5226	0.004147	-126	0
clarityVVS1	-0.08481	0.005198	-16.32	1.495e-59
clarityVVS2	-0.1556	0.004984	-31.23	4.417e-210
clarityVS1	-0.2761	0.004776	-57.81	0
clarityVS2	-0.3475	0.00468	-74.27	0
claritySI1	-0.4945	0.004722	-104.7	0
claritySI2	-0.6621	0.004915	-134.7	0
clarityI1	-1.073	0.01018	-105.4	0
depth	0.03175	0.002387	13.3	3.196e-40
table	0.01005	0.0004703	21.37	1.955e-100
x	0.8638	0.01939	44.54	0
y	0.1765	0.01813	9.731	2.433e-22
z	0.5295	0.03849	13.76	6.528e-43

Fitting linear model: log(price) ~ carat + cut + color + clarity + depth + table + x + y + z

Observations	Residual Std. Error	R^2	Adjusted R^2
25411	0.1211	0.9857	0.9857

```
# fit3_new
pander(summary(new_fit3))
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-0.5016	0.1956	-2.565	0.01033
carat	-0.9183	0.008397	-109.4	0
cutGood	0.08533	0.006146	13.88	1.17e-43
cutVery Good	0.1091	0.006008	18.15	3.858e-73
cutPremium	0.1203	0.005913	20.34	3.306e-91
cutIdeal	0.1525	0.006121	24.91	2.947e-135
colorE	-0.05363	0.002589	-20.72	1.679e-94
colorF	-0.09367	0.002627	-35.66	2.058e-271
colorG	-0.1535	0.002568	-59.77	0
colorH	-0.2507	0.002748	-91.24	0
colorI	-0.3806	0.003087	-123.3	0
colorJ	-0.5309	0.003934	-135	0
clarityVVS1	-0.08623	0.004948	-17.42	1.389e-67
clarityVVS2	-0.1533	0.004735	-32.37	4.467e-225
clarityVS1	-0.2701	0.004544	-59.44	0
clarityVS2	-0.3419	0.004463	-76.62	0
claritySI1	-0.4892	0.004506	-108.6	0
claritySI2	-0.6553	0.00469	-139.7	0
clarityI1	-1.06	0.01098	-96.54	0
depth	0.01643	0.003074	5.343	9.219e-08
table	0.01014	0.0004418	22.96	2.137e-115
x	0.7599	0.02211	34.36	1.125e-252
y	0.116	0.02268	5.115	3.17e-07
z	0.7987	0.05065	15.77	9.847e-56

Fitting linear model: log(price) ~ carat + cut + color + clarity + depth + table + x + y + z

Observations	Residual Std. Error	R^2	Adjusted R^2
24182	0.1092	0.9883	0.9883

```r
# fit4.org
fit4.org <- lm(log(price) ~ carat + cut + color + clarity + depth + table + x,
          data=new_cubiz2)
pander(summary(fit4.org))
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-3.258	0.06199	-52.55	0
carat	-0.8906	0.008468	-105.2	0
cutGood	0.1033	0.006153	16.79	6.217e-63
cutVery Good	0.132	0.005991	22.04	1.332e-106
cutPremium	0.122	0.006003	20.32	4.501e-91
cutIdeal	0.1684	0.006168	27.31	9.059e-162
colorE	-0.0535	0.002629	-20.35	2.731e-91
colorF	-0.09303	0.002668	-34.87	5.912e-260
colorG	-0.1534	0.002608	-58.83	0
colorH	-0.252	0.00279	-90.31	0
colorI	-0.3816	0.003135	-121.7	0
colorJ	-0.5307	0.003995	-132.8	0
clarityVVS1	-0.08817	0.005025	-17.55	1.714e-68
clarityVVS2	-0.1557	0.004809	-32.37	4.391e-225
clarityVS1	-0.2726	0.004614	-59.09	0
clarityVS2	-0.3458	0.00453	-76.33	0
claritySI1	-0.4935	0.004574	-107.9	0
claritySI2	-0.6609	0.004758	-138.9	0
clarityI1	-1.072	0.01114	-96.26	0
depth	0.06183	0.0006587	93.86	0
table	0.01005	0.0004477	22.46	1.502e-110
x	1.356	0.00351	386.4	0

Fitting linear model: log(price) ~ carat + cut + color + clarity + depth + table + x

Observations	Residual Std. Error	R^2	Adjusted R^2
24182	0.1109	0.9879	0.9879

```r
# fit5
pander(summary(fit5))
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	6.276	0.0754	83.24	0
poly(carat, 2)1	61.34	2.577	23.8	8.002e-124
poly(carat, 2)2	-9.132	0.3261	-28.01	6.81e-170
cutGood	0.08862	0.005873	15.09	3.272e-51
cutVery Good	0.115	0.005724	20.09	5.128e-89
cutPremium	0.1267	0.005729	22.11	3.031e-107
cutIdeal	0.1598	0.005882	27.17	3.979e-160
colorE	-0.05237	0.002505	-20.9	3.506e-96
colorF	-0.09292	0.002542	-36.55	9.812e-285
colorG	-0.1539	0.002486	-61.91	0
colorH	-0.2511	0.002661	-94.35	0
colorI	-0.3785	0.002988	-126.7	0
colorJ	-0.5255	0.00381	-137.9	0
clarityVVS1	-0.08736	0.004788	-18.24	7.356e-74
clarityVVS2	-0.1533	0.004582	-33.46	5.954e-240
clarityVS1	-0.2702	0.004397	-61.47	0
clarityVS2	-0.3433	0.004317	-79.53	0
claritySI1	-0.4892	0.004361	-112.2	0
claritySI2	-0.6552	0.004538	-144.4	0
clarityI1	-1.067	0.01061	-100.6	0
depth	0.02663	0.0009503	28.03	3.917e-170
table	0.004045	0.0004435	9.119	8.149e-20
poly(x, 2)1	107.5	2.557	42.03	0
poly(x, 2)2	-15.78	0.4435	-35.59	1.779e-270

Fitting linear model: log(price) ~ poly(carat, 2) + cut + color + clarity + depth + table + poly(x, 2)

Observations	Residual Std. Error	R^2	Adjusted R^2
24182	0.1057	0.989	0.989

```r
# fit6
pander(summary(fit6))
```

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	10.15	0.2851	35.59	1.97e-270
poly(carat, 2)1	411.8	25.04	16.45	1.808e-60
poly(carat, 2)2	25.76	2.5	10.3	7.798e-25
cutGood	0.09344	0.005859	15.95	5.794e-57
cutVery Good	0.1205	0.005714	21.09	7.024e-98
cutPremium	0.1339	0.005729	23.37	1.81e-119
cutIdeal	0.1675	0.005884	28.47	2.243e-175
colorE	-0.05289	0.002495	-21.2	8.299e-99
colorF	-0.09376	0.002533	-37.02	7.851e-292
colorG	-0.1541	0.002476	-62.25	0
colorH	-0.2522	0.002652	-95.11	0
colorI	-0.3786	0.002976	-127.2	0
colorJ	-0.5267	0.003795	-138.8	0
clarityVVS1	-0.08878	0.00477	-18.61	9.06e-77
clarityVVS2	-0.155	0.004565	-33.96	6.103e-247
clarityVS1	-0.2704	0.004379	-61.76	0
clarityVS2	-0.3435	0.0043	-79.9	0
claritySI1	-0.49	0.004344	-112.8	0
claritySI2	-0.6577	0.004523	-145.4	0
clarityI1	-1.07	0.01057	-101.2	0
depth	0.02696	0.0009467	28.47	2.003e-175
table	0.00396	0.0004418	8.965	3.333e-19
poly(x, 2)1	222.7	8.574	25.97	1.101e-146
poly(x, 2)2	17.45	2.402	7.263	3.901e-13
carat:x	-0.7984	0.05673	-14.07	8.346e-45

Fitting linear model: log(price) ~ poly(carat, 2) + cut + color + clarity + depth + table + poly(x, 2) + carat:x

Observations	Residual Std. Error	R^2	Adjusted R^2

Observations	Residual Std. Error	R^2	Adjusted R^2
24182	0.1053	0.9891	0.9891

```
# fit.gam
summary(fit.gam)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## log(price) ~ s(carat) + cut + color + clarity + s(depth) + s(table) +
##     s(x)
##
## Parametric coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   8.172627   0.008989  909.17   <2e-16 ***
## cutGood       0.074390   0.007421   10.03   <2e-16 ***
## cutVery Good  0.092931   0.007921   11.73   <2e-16 ***
## cutPremium    0.105729   0.008096   13.06   <2e-16 ***
## cutIdeal      0.137086   0.008199   16.72   <2e-16 ***
## colorE       -0.053409   0.002408  -22.18   <2e-16 ***
## colorF       -0.095257   0.002444  -38.98   <2e-16 ***
## colorG       -0.154822   0.002389  -64.80   <2e-16 ***
## colorH       -0.252134   0.002562  -98.41   <2e-16 ***
## colorI       -0.376412   0.002878 -130.79   <2e-16 ***
## colorJ       -0.519631   0.003677 -141.32   <2e-16 ***
## clarityVVS1  -0.085965   0.004611  -18.64   <2e-16 ***
## clarityVVS2  -0.150615   0.004420  -34.07   <2e-16 ***
## clarityVS1   -0.266605   0.004231  -63.02   <2e-16 ***
## clarityVS2   -0.341610   0.004152  -82.27   <2e-16 ***
## claritySI1   -0.488574   0.004197 -116.42   <2e-16 ***
## claritySI2   -0.658616   0.004371 -150.70   <2e-16 ***
## clarityI1    -1.058372   0.010235 -103.41   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##             edf Ref.df      F p-value
## s(carat) 7.930      9 120.61  <2e-16 ***
## s(depth) 4.168      9  70.25  <2e-16 ***
## s(table) 5.963      9  14.66  <2e-16 ***
## s(x)     8.906      9 258.51  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =   0.99   Deviance explained =   99%
## -REML = -20847  Scale est. = 0.010303  n = 24182
```

```r
# fit.gam2
summary(fit.gam2)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## log(price) ~ s(carat, k = 20) + cut + color + clarity + s(depth,
##     k = 20) + s(table, k = 20) + s(x, k = 20)
##
## Parametric coefficients:
##               Estimate Std. Error  t value Pr(>|t|)
## (Intercept)   8.172122   0.009078  900.256   <2e-16 ***
## cutGood       0.071434   0.007598    9.402   <2e-16 ***
## cutVery Good  0.093132   0.008114   11.478   <2e-16 ***
## cutPremium    0.105691   0.008251   12.809   <2e-16 ***
## cutIdeal      0.137290   0.008343   16.455   <2e-16 ***
## colorE       -0.052562   0.002341  -22.448   <2e-16 ***
## colorF       -0.093756   0.002376  -39.458   <2e-16 ***
## colorG       -0.155312   0.002323  -66.864   <2e-16 ***
## colorH       -0.251438   0.002492 -100.917   <2e-16 ***
## colorI       -0.374467   0.002800 -133.730   <2e-16 ***
## colorJ       -0.516728   0.003581 -144.304   <2e-16 ***
## clarityVVS1  -0.083993   0.004482  -18.738   <2e-16 ***
## clarityVVS2  -0.148547   0.004299  -34.557   <2e-16 ***
## clarityVS1   -0.265580   0.004113  -64.574   <2e-16 ***
## clarityVS2   -0.343711   0.004038  -85.129   <2e-16 ***
## claritySI1   -0.488679   0.004080 -119.781   <2e-16 ***
## claritySI2   -0.659342   0.004250 -155.143   <2e-16 ***
## clarityI1    -1.051384   0.009959 -105.572   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##             edf Ref.df       F p-value
## s(carat) 15.821     19  42.827  <2e-16 ***
## s(depth)  5.850     19  28.952  <2e-16 ***
## s(table)  7.807     19   7.329  <2e-16 ***
## s(x)     17.575     19 158.369  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =   0.99   Deviance explained =   99%
## -REML = -21512  Scale est. = 0.0097248  n = 24182
```

```r
# fit6.lm
summary(fit6.lm)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## log(price) ~ poly(carat, 2) + cut + color + clarity + depth +
##     table + poly(x, 2) + carat:x
##
## Parametric coefficients:
##                   Estimate Std. Error  t value Pr(>|t|)
## (Intercept)      1.015e+01  2.851e-01   35.589  < 2e-16 ***
## poly(carat, 2)1  4.118e+02  2.504e+01   16.449  < 2e-16 ***
## poly(carat, 2)2  2.576e+01  2.500e+00   10.302  < 2e-16 ***
## cutGood          9.344e-02  5.860e-03   15.948  < 2e-16 ***
## cutVery Good     1.205e-01  5.714e-03   21.093  < 2e-16 ***
## cutPremium       1.339e-01  5.729e-03   23.370  < 2e-16 ***
## cutIdeal         1.675e-01  5.884e-03   28.469  < 2e-16 ***
## colorE          -5.289e-02  2.495e-03  -21.196  < 2e-16 ***
## colorF          -9.376e-02  2.533e-03  -37.019  < 2e-16 ***
## colorG          -1.541e-01  2.476e-03  -62.247  < 2e-16 ***
## colorH          -2.522e-01  2.652e-03  -95.113  < 2e-16 ***
## colorI          -3.786e-01  2.976e-03 -127.214  < 2e-16 ***
## colorJ          -5.267e-01  3.795e-03 -138.765  < 2e-16 ***
## clarityVVS1     -8.878e-02  4.770e-03  -18.611  < 2e-16 ***
## clarityVVS2     -1.550e-01  4.565e-03  -33.957  < 2e-16 ***
## clarityVS1      -2.704e-01  4.379e-03  -61.759  < 2e-16 ***
## clarityVS2      -3.435e-01  4.300e-03  -79.901  < 2e-16 ***
## claritySI1      -4.900e-01  4.344e-03 -112.807  < 2e-16 ***
## claritySI2      -6.577e-01  4.523e-03 -145.411  < 2e-16 ***
## clarityI1       -1.070e+00  1.057e-02 -101.226  < 2e-16 ***
## depth            2.696e-02  9.467e-04   28.473  < 2e-16 ***
## table            3.960e-03  4.418e-04    8.965  < 2e-16 ***
## poly(x, 2)1      2.227e+02  8.574e+00   25.970  < 2e-16 ***
## poly(x, 2)2      1.745e+01  2.402e+00    7.263  3.9e-13 ***
## carat:x         -7.984e-01  5.673e-02  -14.073  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## R-sq.(adj) =  0.989   Deviance explained = 98.9%
## -REML = -20028  Scale est. = 0.011083  n = 24182
```

11. Coefficients calulation

```
# using fit.gam2
# cut Ideal
exp(0.137290)
```

```
## [1] 1.147161
```

```
proportional.change.Ideal <- exp(0.137290) - 1
proportional.change.Ideal
```

```
## [1] 0.1471608
```

```
# cut Premium
exp(0.105691)
```

```
## [1] 1.111478
```

```
proportional.change.Premium <- exp(0.105691) - 1
proportional.change.Premium
```

```
## [1] 0.1114784
```

```
# color J
exp(-0.516728)
```

```
## [1] 0.596469
```

```
proportional.change.J <- exp(-0.516728) - 1
proportional.change.J
```

```
## [1] -0.403531
```

```
# color G
exp(-0.155312)
```

```
## [1] 0.856148
```

```
proportional.change.G <- exp(-0.155312) - 1
proportional.change.G
```

```
## [1] -0.143852
```

```
# clarity I1
exp(-1.051384)
```

```
## [1] 0.3494538
```

```
proportional.change.I1 <- exp(-1.051384) - 1
proportional.change.I1
```

```
## [1] -0.6505462
```

```
# clarity VS1
exp(-0.265580)
```

```
## [1] 0.7667611
```

```
proportional.change.VS1 <- exp(-0.265580) - 1
proportional.change.VS1
```

```
## [1] -0.2332389
```