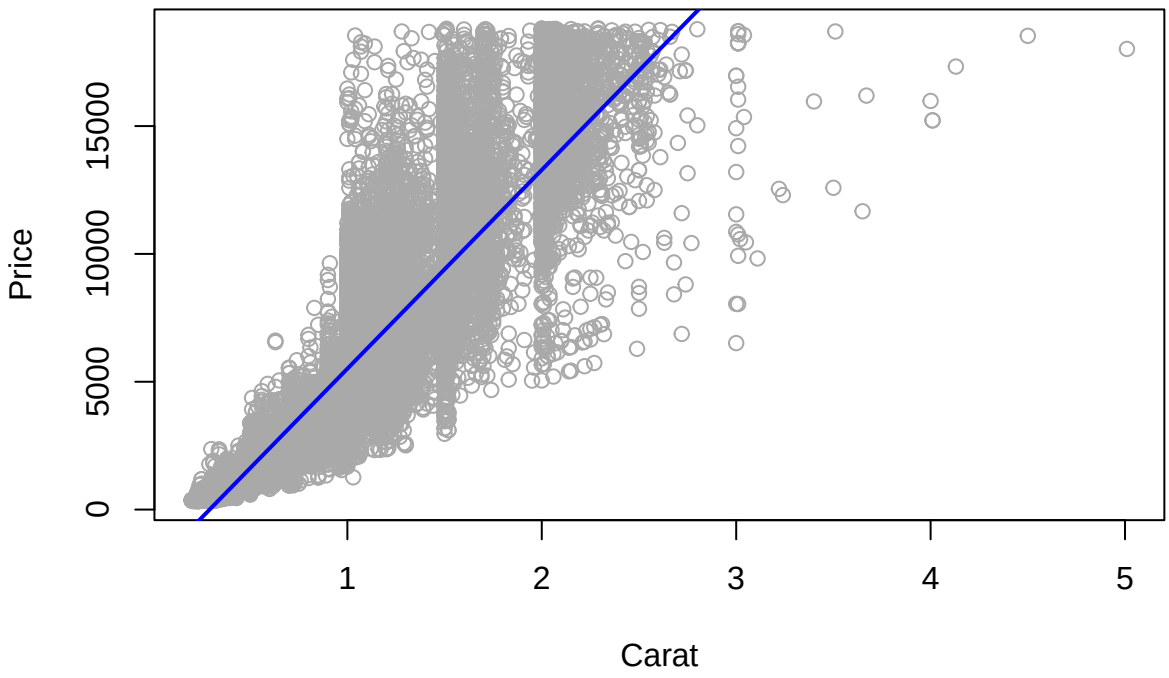
Simple linear regression

Since we have a feature which is strongly correlated with our target, it is worth fitting a simple linear regression, where we can also visualize the model in a nice bi-dimensional plot. After a 70-30 split between training and test set, we perform OLS on the training set and obtain the following regression of `price` on `carat`:

```
##
## Call:
## lm(formula = price ~ carat, data = dta, subset = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -18672.5   -811.9    -19.0    541.2  12721.1
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2265.88      15.62  -145.0   <2e-16 ***
## carat        7775.72      16.84   461.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1550 on 37753 degrees of freedom
## Multiple R-squared:  0.8496, Adjusted R-squared:  0.8496
## F-statistic: 2.133e+05 on 1 and 37753 DF,  p-value: < 2.2e-16
```

```
## [1] "RMSE on the training set: 1550.242"
```

First of all, we notice all the coefficients are extremely significant, with a p-value which is basically zero. Despite the simplicity, the model performs pretty good on the test set, with a R^2 of approximately 85% and a RMSE of 1550 USD. This means we are able to explain 85% of the variability of diamonds' price using just their carat; moreover, the model predicts the diamonds' price with an average error of 1550 USD in the training set. The fact that both the response and the predictor are in levels allows for a nice interpretation of the coefficient, that is the effect of a one-unit increase in carat leads (on average) to an increase of 7775.72 USD in the diamond's price. Let's now visualize the model in the bi-dimensional space defined by the response and the predictor.

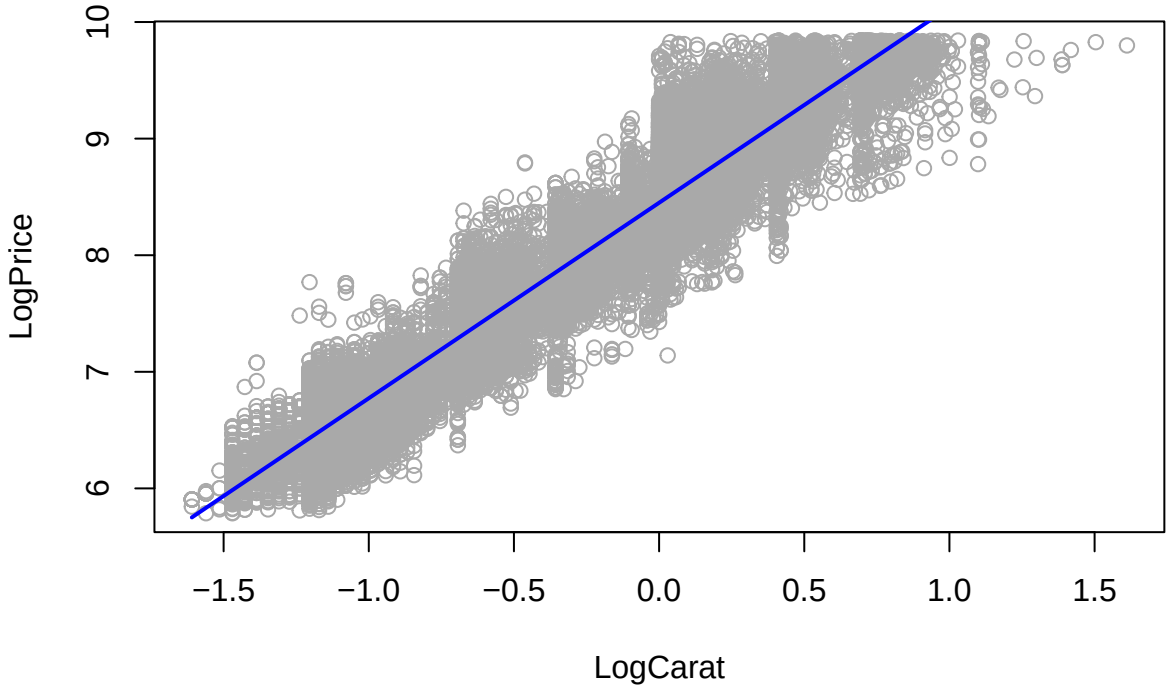


This plot makes clear why visualization is so important: even though the number shown in the summary model looked pretty good, the plots highlights the fact that we are totally missing the target after the level of 3 carats because, while the line keeps increasing, there is not much difference between the price of a 3-carat diamond and a 5-carat one. Notice this happens even though there were several training observations above the level of 3 carats. From the descriptive statistics, we know both `carat` and `price` have a left-skewed distribution, so taking logarithm may help in getting better performances. We therefore fit a new regression, this time considering log-transformed variables. The resulting model's summary is shown below.

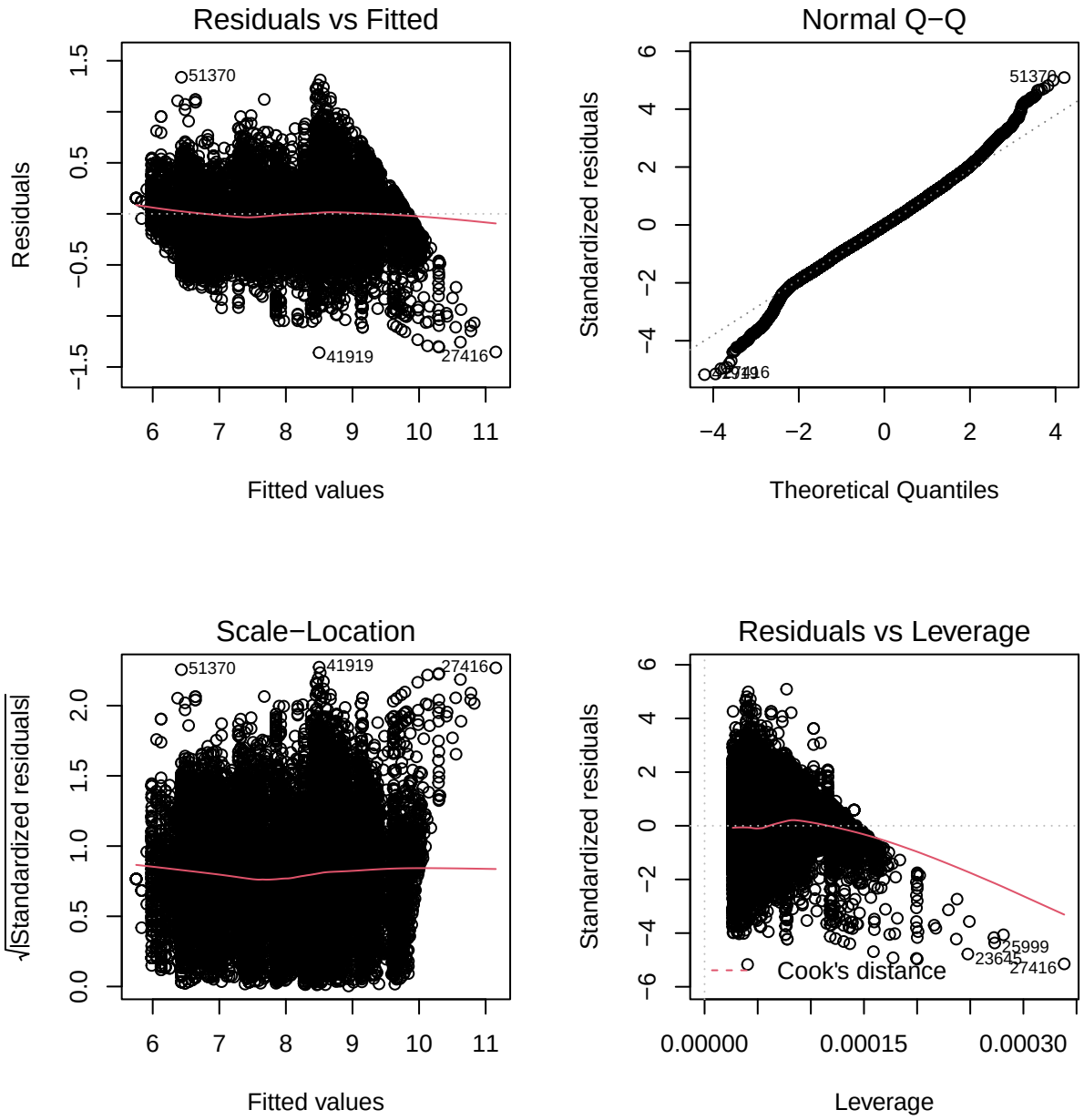
```
##
## Call:
## lm(formula = log(price) ~ log(carat), data = dta, subset = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.35899 -0.16979 -0.00602  0.16681  1.33790
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 8.449883   0.001633  5175.4   <2e-16 ***
## log(carat)  1.676806   0.002314   724.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2629 on 37753 degrees of freedom
## Multiple R-squared:  0.933,  Adjusted R-squared:  0.933
## F-statistic: 5.253e+05 on 1 and 37753 DF,  p-value: < 2.2e-16
```

```
## [1] "MSE on the training set: 0.06909"
```

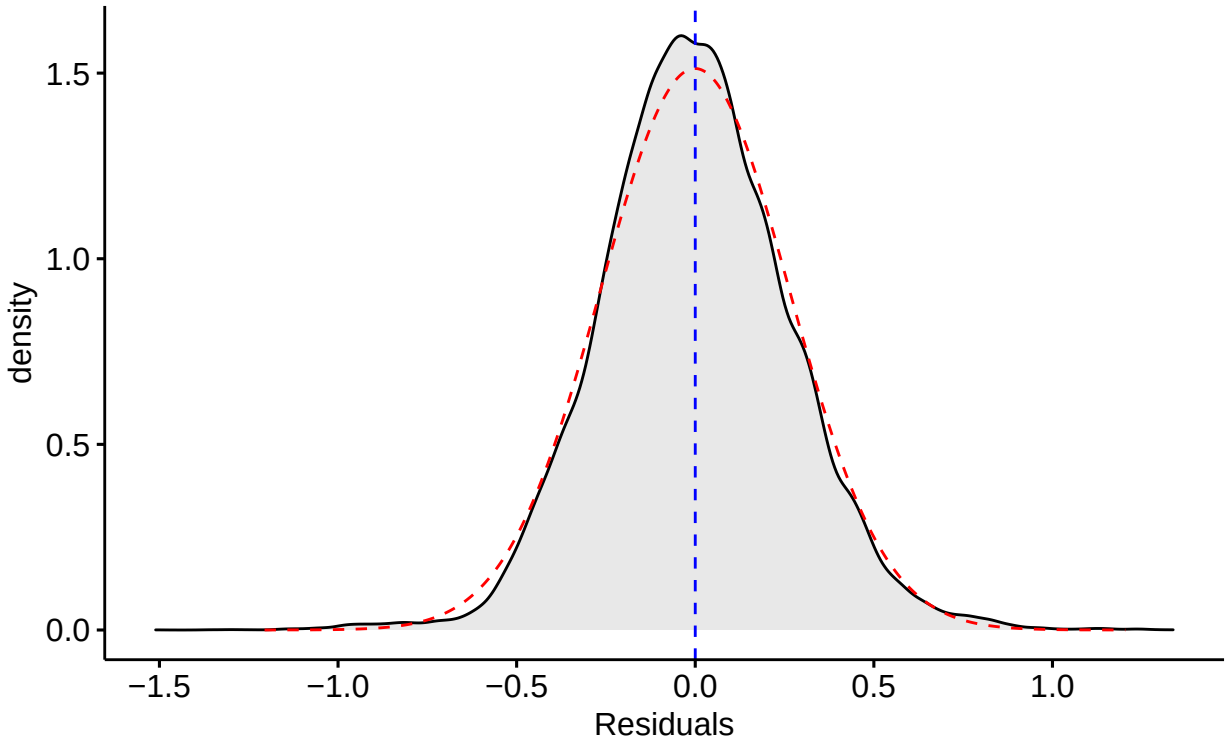
Indeed, significant improvements were achieved. The R^2 statistic is now 93%, and the training MSE is 0.069 (RMSE = 0.26). Clearly, this is a log-scaled MSE, so it is not comparable with the one of the previous model; however, the fact that the response now varies between 5.78 and 9.84 suggests that we have obtained a slightly better model, also in terms of training RMSE. Still, all the coefficients are strongly significant. Finally, this simple model yields again a nice interpretation: according to this log-regression, an increase of 1% in carat causes an increase of 1.67% in the diamond's price. We now show the model in the bi-dimensional space, just like we did before.



We are still missing the observations with extreme values for our predictor. However, the fact we reduced the target variation range led to a significant improvement in the model. Also visually, the relationship between the predictor and the response looks more linear. Let's now turn to the diagnostic of the model.



The `Residuals vs Fitted` graph highlights that there is still some pattern left in the residuals, especially for large values of our target, that are systematically overestimated by the model. This trend is even more evident in the `Residuals vs Leverage` plot. These issues are probably due to the simplicity of the *linear* model. The `Scale-Location` plot is quite reassuring for what concerns the homoskedasticity assumption, while the Q-Q plot does not seem very clear. We therefore plot the residuals' distribution together with the normal distribution to assess residuals' normality.



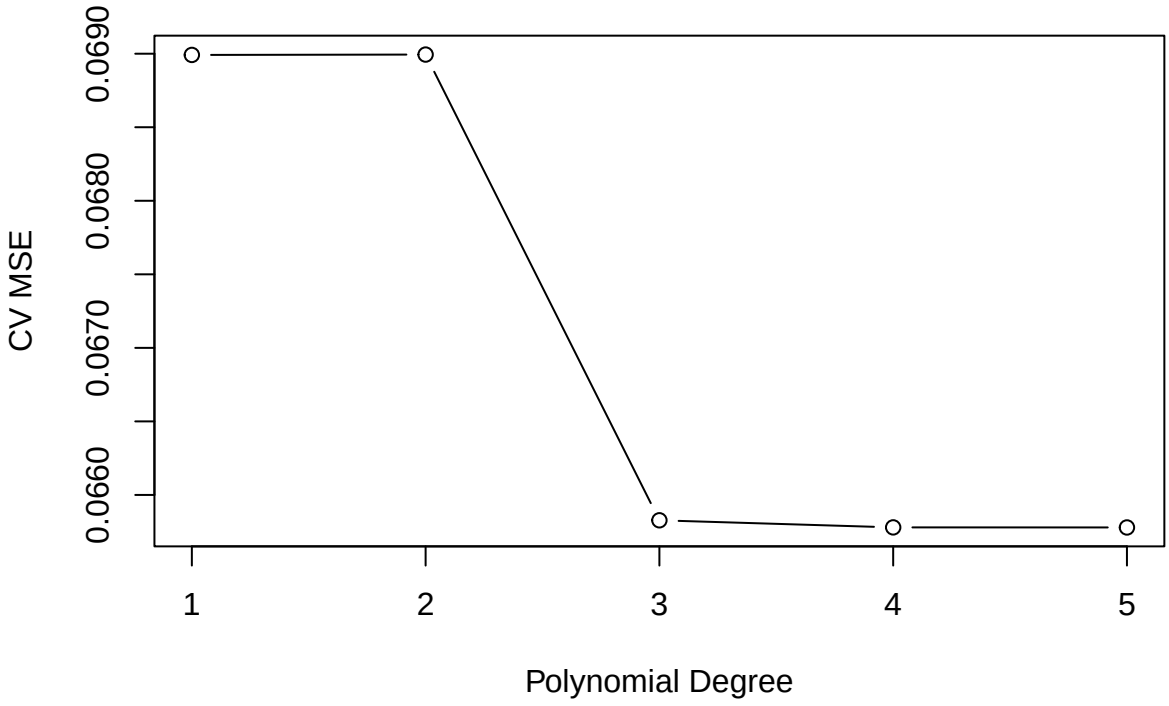
Normality assumption for the errors seems to be met by the model. Finally, we assess how the model performs on an independent test set.

```
## [1] "MSE on the test set: 0.06876"
```

The test MSE is basically the same as its training counterpart.

Simple ploynomial regression

We now relax the linearity assumption of traditional OLS by introducing as new predictors polynomial transformations of our original regressor. First of all, we use cross validation to select the optimal degree of the polynomial to be fitted.



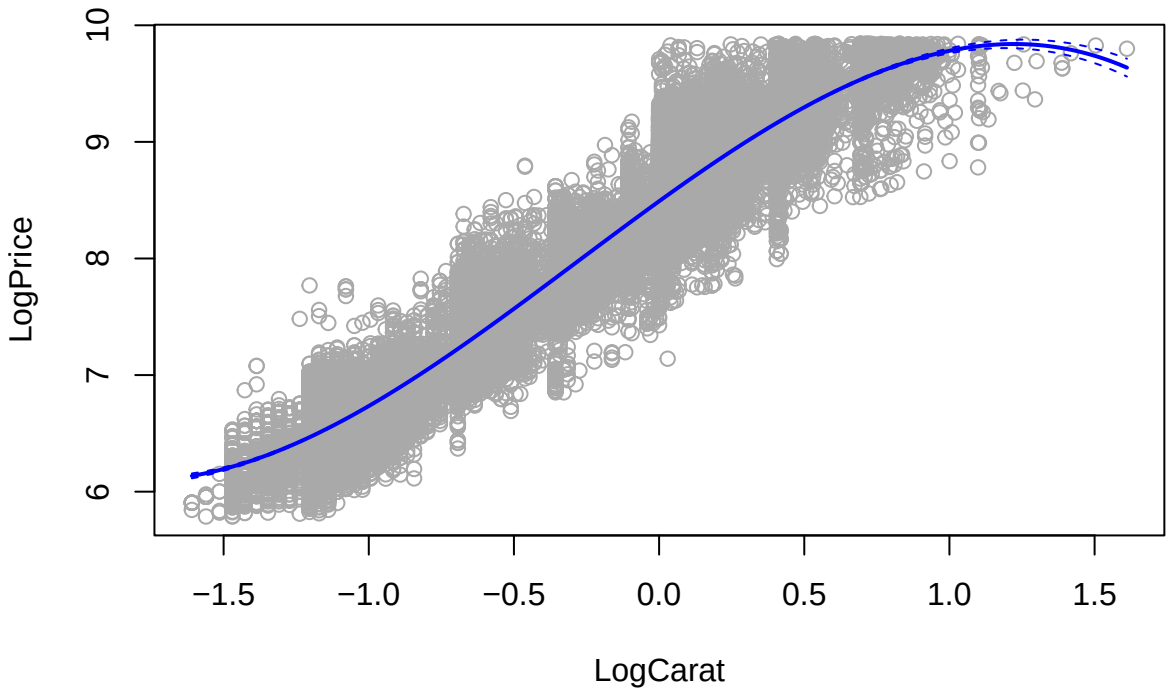
A third-degree polynomial seems to be the optimal choice, since while there is a sharp decrease in the cv MSE estimate between the second and the third degree, there is not much improvements when going beyond three. We therefore estimate the model regressing the logarithm of `price` on the cubic orthogonal polynomial of the logarithm of `carat`.

```
##
## Call:
## lm(formula = log(price) ~ poly(log(carat), 3), data = dta, subset = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.40332 -0.17403 -0.00116  0.17111  1.30199
##
## Coefficients:
##                         Estimate Std. Error  t value Pr(>|t|)
## (Intercept)             7.787506   0.001321 5897.333   <2e-16 ***
## poly(log(carat), 3)1 227.764035   0.306717  742.586   <2e-16 ***
## poly(log(carat), 3)2   0.146005   0.306387    0.477    0.634
## poly(log(carat), 3)3 -13.268384   0.306899  -43.234   <2e-16 ***
## ---
```

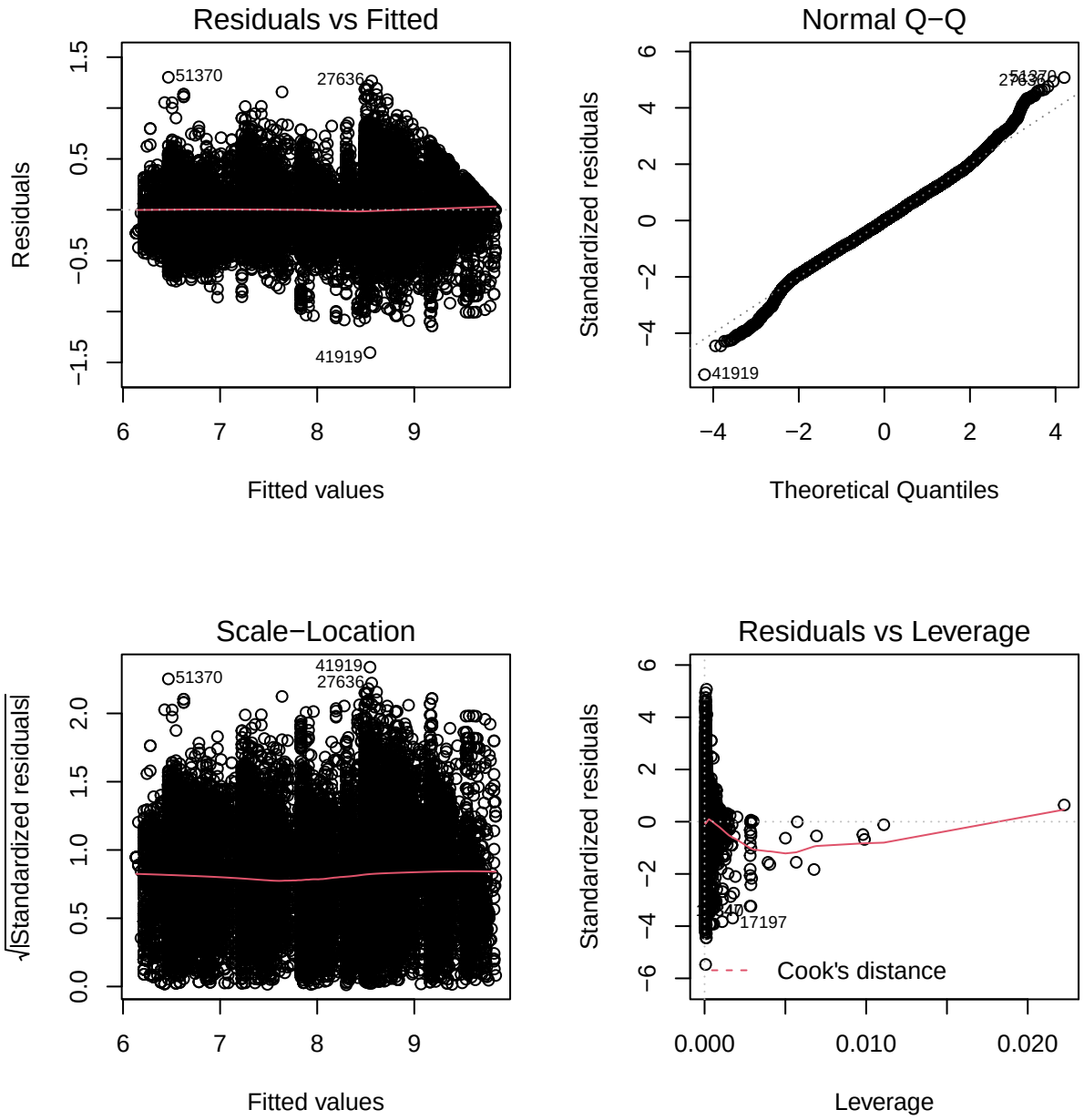
```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2566 on 37751 degrees of freedom
## Multiple R-squared:  0.9361, Adjusted R-squared:  0.9361
## F-statistic: 1.844e+05 on 3 and 37751 DF,  p-value: < 2.2e-16
```

```
## [1] "MSE on the training set: 0.06583"
```

Even though the squared term is not significant, we keep it because we include in the model the cubic one, which is of higher order. Notice the performance slightly improves, both in terms of R^2 and in terms of training MSE. Let's now visualize the model to see if, with the cubic fit, we are able to get better predictions for high-carat diamonds, which the simple linear model completely missed.



The cubic curve fits definitely better the data, especially at the boundaries of `carat`. Moreover, the confidence intervals do not go wild at the boundaries, as it often happens for polynomial regression, so we would say this is an overall better regression than the linear one. Let's now check the model's diagnostic.



The `Residuals vs Fitted` plot highlights the fact that, differently from before, there is no pattern left in the residuals. The model deals well also with high-leverage points, as shown by the `Residuals vs Leverage` plot, but still makes quite large errors. Again, the `Scale-Location` plot indicates that the homoskedasticity assumption is met, and also the Q-Q plot is reassuring about errors' normality. To make a proper comparison with the simple linear model of before, we look at the performance of the current model on an independent test set.

```
## [1] "MSE on the test set: 0.0658"
```

Including the squared and the cubic term in the regression led to a reduction in the test MSE of about 4.4%.