Week 8 - Homework

STAT 420, Summer 2022, Ilya Andreev, iandre3@illinois.edu

Exercise 1 (Writing Functions)

(a) Write a function named `diagnostics` that takes as input the arguments:

- `model`, an object of class `lm()`, that is a model fit via `lm()`
- `pcol`, for controlling point colors in plots, with a default value of `grey`
- `lcol`, for controlling line colors in plots, with a default value of `dodgerblue`
- `alpha`, the significance level of any test that will be performed inside the function, with a default value of `0.05`
- `plotit`, a logical value for controlling display of plots with default value `TRUE`
- `testit`, a logical value for controlling outputting the results of tests with default value `TRUE`

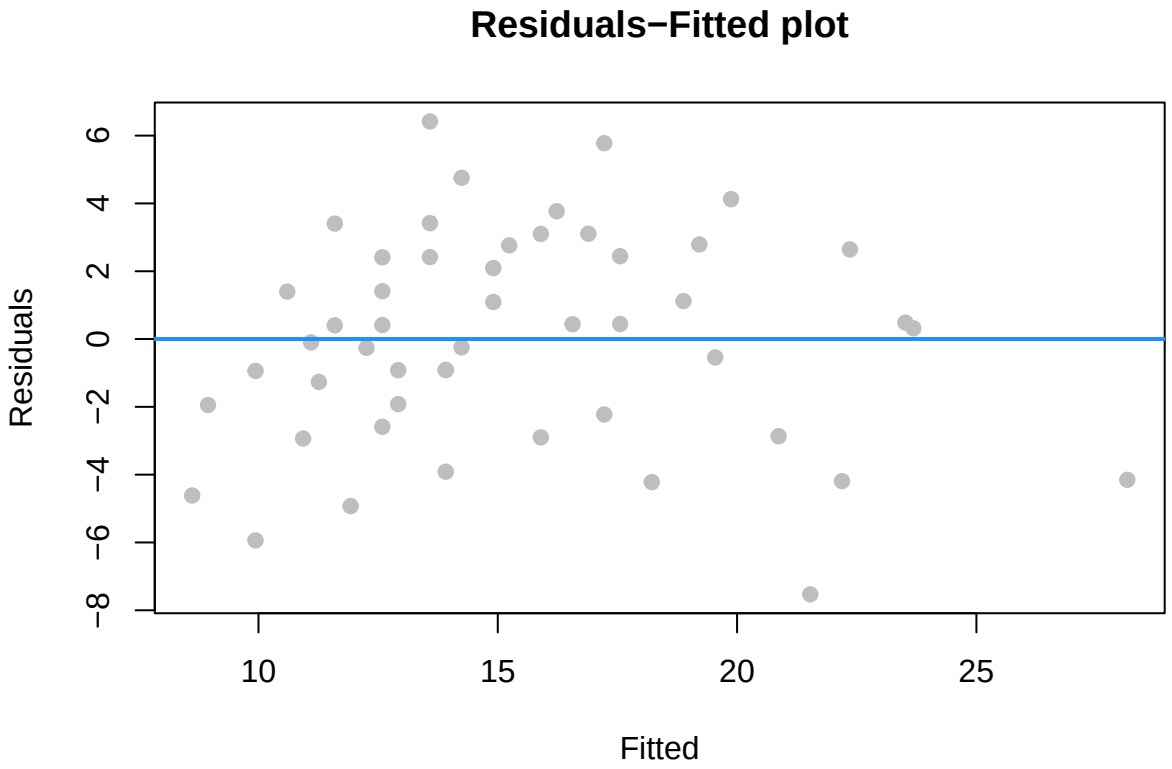
The function should output:

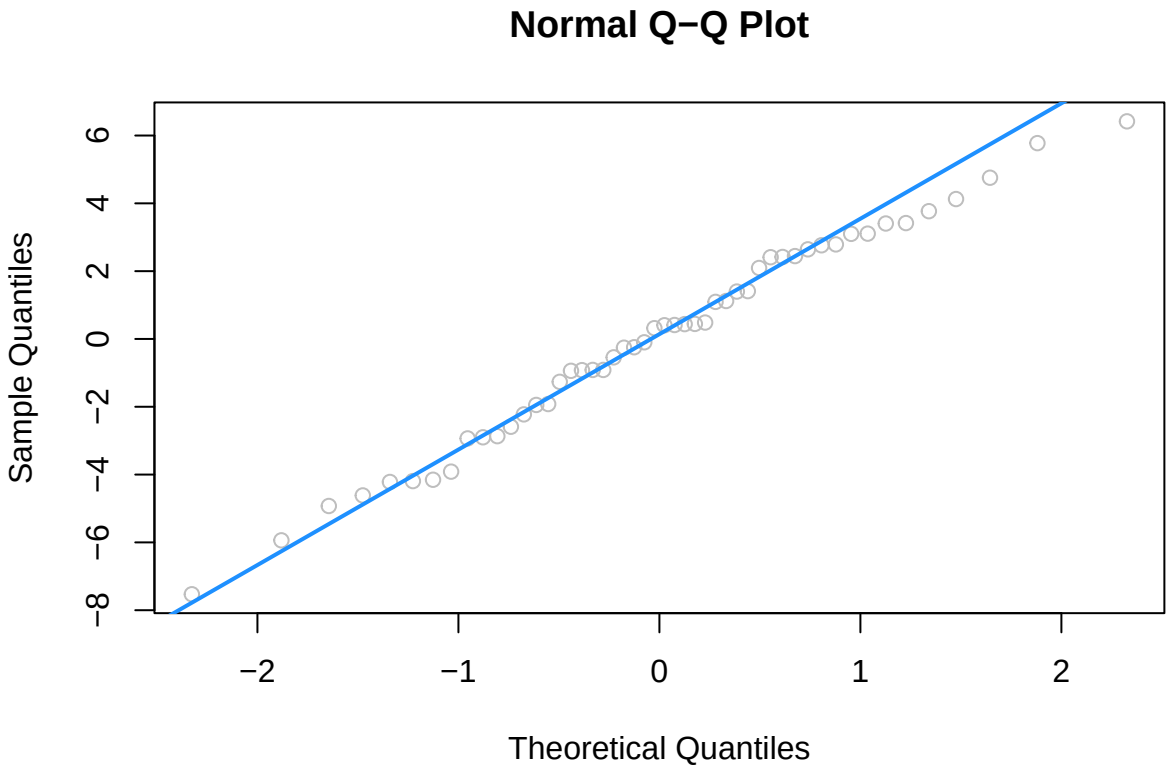
- A list with two elements when `testit` is `TRUE`:
  - `p_val`, the p-value for the Shapiro-Wilk test for assessing normality
  - `decision`, the decision made when performing the Shapiro-Wilk test using the `alpha` value input to the function. "Reject" if the null hypothesis is rejected, otherwise "Fail to Reject."
- Two plots, side-by-side, when `plotit` is `TRUE`:
  - A fitted versus residuals plot that adds a horizontal line at $y = 0$, and labels the x-axis "Fitted" and the y-axis "Residuals." The points and line should be colored according to the input arguments. Give the plot a title.
  - A Normal Q-Q plot of the residuals that adds the appropriate line using `qqline()`. The points and line should be colored according to the input arguments. Be sure the plot has a title.

Consider using this function to help with the remainder of the assignment as well.

```
m = lm(speed ~ dist, cars)

diagnostics = function(model,
                       pcol="grey",
                       lcol="dodgerblue",
                       alpha=0.05,
                       plotit=TRUE,
                       testit=TRUE) {
  if (plotit) {
    plot(fitted(model), resid(model), col=pcol,
     pch=20, cex=1.5, xlab="Fitted", ylab="Residuals", main="Residuals-Fitted plot")
    abline(h=0, lty=1, col=lcol, lwd=2)

    qqnorm(resid(model), main="Normal Q-Q Plot", col=pcol)
    qqline(resid(model), col=lcol, lwd=2)
  }

    if (testit) {
```

```
    p = shapiro.test(resid(model))[[2]]
    decision = ""
    if (p < alpha) {
      decision = "Reject"
    } else {
      decision = "Fail to Reject"
    }

    list(p_val=p, decision=decision)
  }
}

diagnostics(m)
```






```
## $p_val
## [1] 0.8696
##
## $decision
## [1] "Fail to Reject"
```

(b) Run the following code.

```
set.seed(40)

data_1 = data.frame(x = runif(n = 30, min = 0, max = 10),
                    y = rep(x = 0, times = 30))
data_1$y = with(data_1, 2 + 1 * x + rexp(n = 30))
fit_1 = lm(y ~ x, data = data_1)

data_2 = data.frame(x = runif(n = 20, min = 0, max = 10),
                    y = rep(x = 0, times = 20))
data_2$y = with(data_2, 5 + 2 * x + rnorm(n = 20))
fit_2 = lm(y ~ x, data = data_2)

data_3 = data.frame(x = runif(n = 40, min = 0, max = 10),
                    y = rep(x = 0, times = 40))
data_3$y = with(data_3, 2 + 1 * x + rnorm(n = 40, sd = x))
fit_3 = lm(y ~ x, data = data_3)
```

```
diagnostics(fit_1, plotit = FALSE)$p_val
diagnostics(fit_2, plotit = FALSE)$decision
diagnostics(fit_1, testit = FALSE, pcol = "black", lcol = "black")
diagnostics(fit_2, testit = FALSE, pcol = "grey", lcol = "green")
diagnostics(fit_3)
```

Exercise 2 (Prostate Cancer Data)

For this exercise, we will use the `prostate` data, which can be found in the `faraway` package. After loading the `faraway` package, use `?prostate` to learn about this dataset.

```
library(faraway)
```

(a) Fit an additive multiple regression model with `lpsa` as the response and the remaining variables in the `prostate` dataset as predictors. Report the R^2 value for this model.

```
original_model = lm(lpsa ~ ., prostate)
summary(original_model)$r.squared
```

```
## [1] 0.6548
```

(b) Check the constant variance assumption for this model. Do you feel it has been violated? Justify your answer.

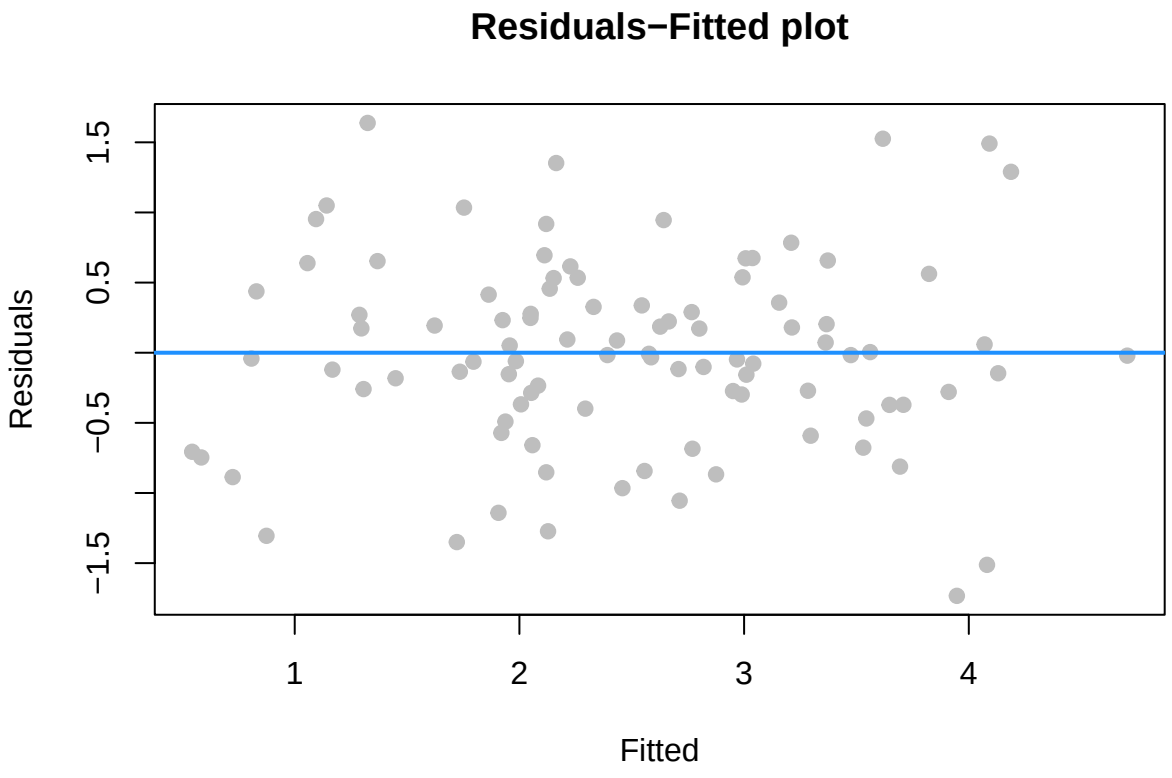
```
library(lmtest)
```

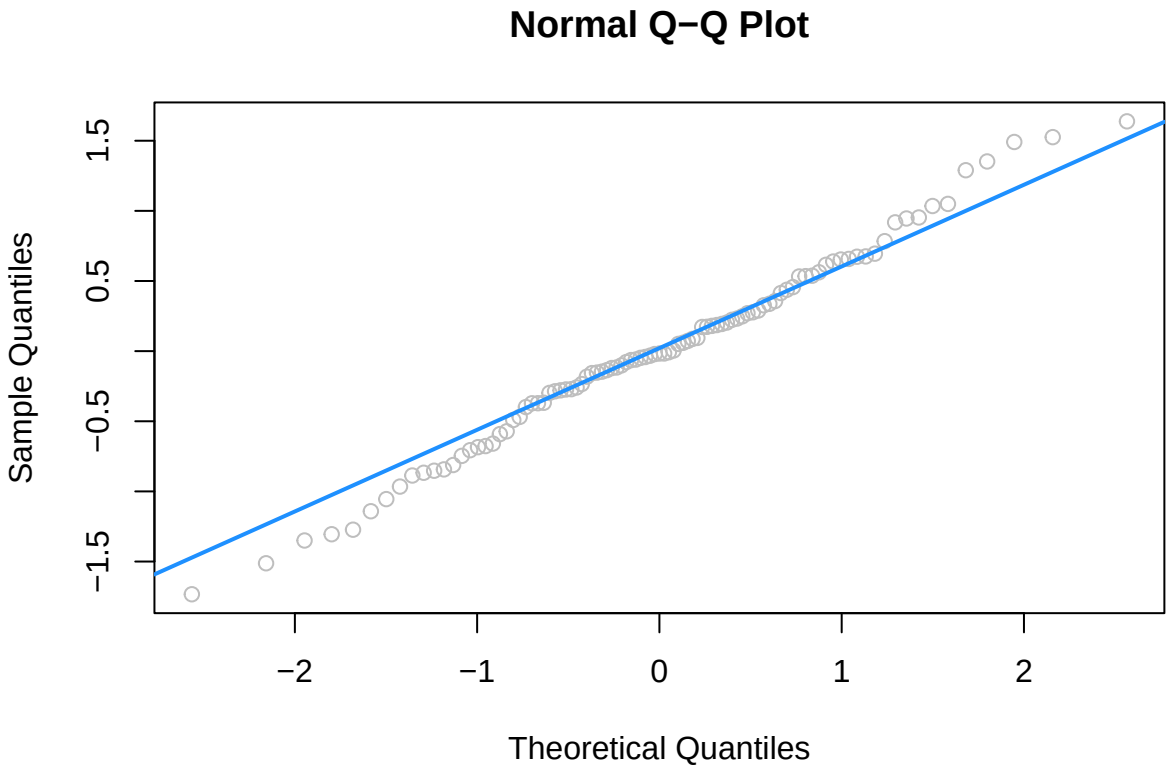
```
## Loading required package: zoo
```

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

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

```
diagnostics(original_model)
```






```
## $p_val
## [1] 0.7721
##
## $decision
## [1] "Fail to Reject"
```

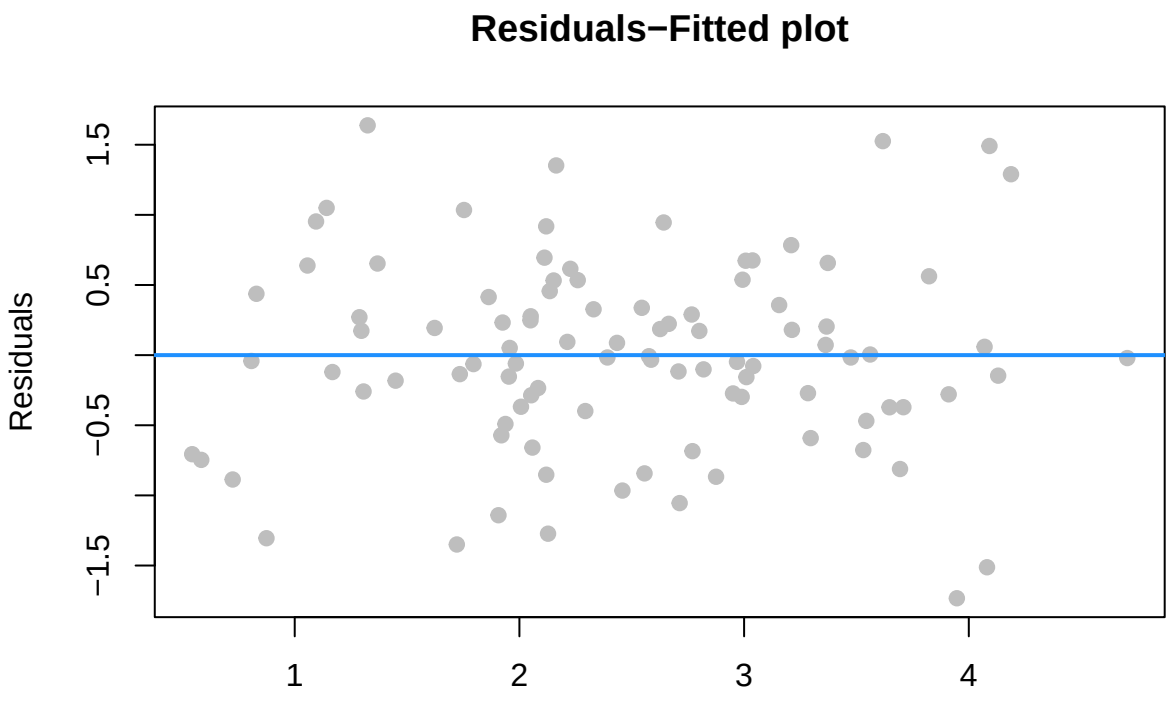
```
bptest(original_model)
```

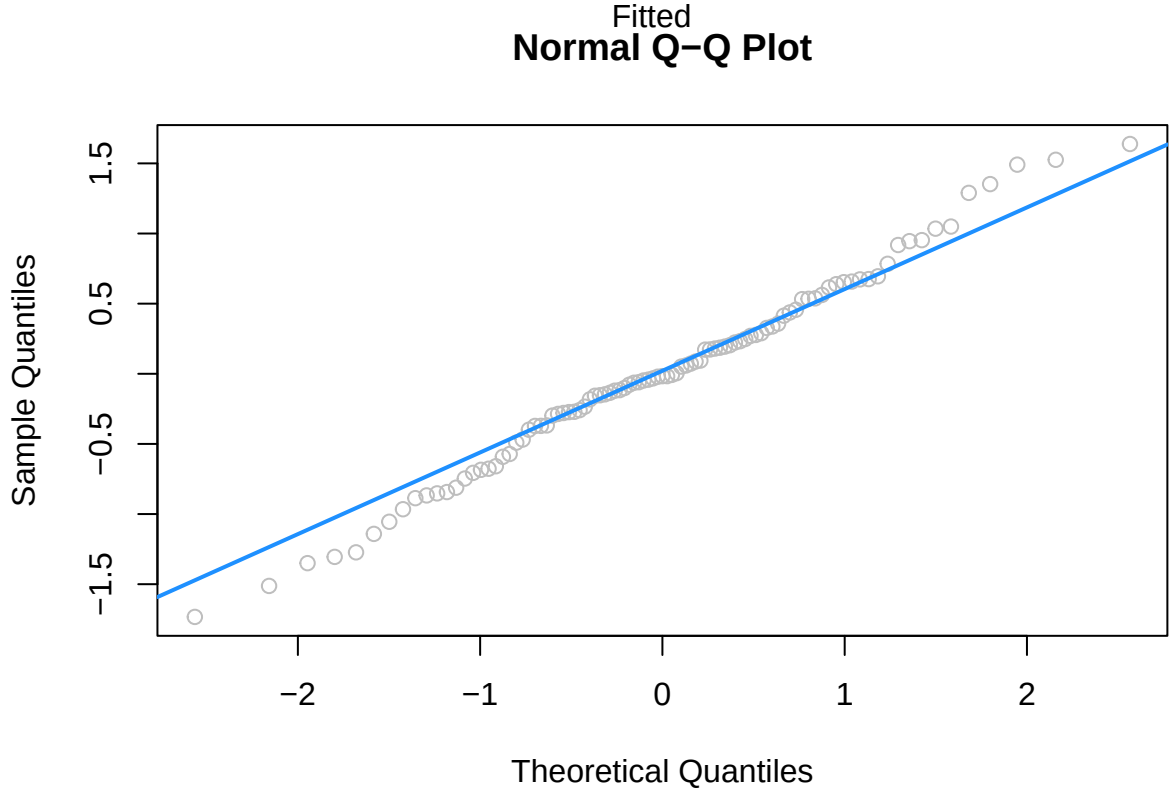
```
##
##  studentized Breusch-Pagan test
##
## data:  original_model
## BP = 10, df = 8, p-value = 0.3
```

Judging visually by the Residuals-Fitted plot and by the Breusch-Pagan Test's p-value of 0.4, we do not reject the homoscedasticity assumption.

(c) Check the normality assumption for this model. Do you feel it has been violated? Justify your answer.

```
diagnostics(original_model)
```






```
## $p_val
## [1] 0.7721
##
## $decision
## [1] "Fail to Reject"
```

While there is some mismatch between low and high theoretical quantiles and their corresponding sample quantiles, the Shapiro-Wilk test tells us that there is not enough evidence to reject the normality hypothesis.

(d) Check for any high leverage observations. Report any observations you determine to have high leverage.

```
prostate[hatvalues(original_model) > 2 * mean(hatvalues(original_model)),]
```

```
##    lcavol lweight age    lbph svi    lcp gleason pgg45  lpsa
## 32 0.1823   6.108  65  1.7047   0 -1.386       6     0 2.008
## 37 1.4231   3.657  73 -0.5798   0  1.658       8    15 2.158
## 41 0.6206   3.142  60 -1.3863   0 -1.386       9    80 2.298
## 74 1.8390   3.237  60  0.4383   1  1.179       9    90 3.075
## 92 2.5329   3.678  61  1.3481   1 -1.386       7    15 4.130
```

There are 5 points whose leverage is twice as large as the average leverage.

(e) Check for any influential observations. Report any observations you determine to be influential.

```
prostate[cooks.distance(original_model) > 4 / length(cooks.distance(original_model)),]
```

```
##     lcavol lweight age    lbph svi    lcp gleason pgg45  lpsa
## 32  0.1823   6.108  65  1.7047   0 -1.386       6     0 2.008
## 39  2.6610   4.085  68  1.3737   1  1.833       7    35 2.214
## 47  2.7279   3.995  79  1.8795   1  2.657       9   100 2.569
## 69 -0.4463   4.409  69 -1.3863   0 -1.386       6     0 2.963
## 95  2.9074   3.396  52 -1.3863   1  2.464       7    10 5.143
## 96  2.8826   3.774  68  1.5581   1  1.558       7    80 5.478
## 97  3.4720   3.975  68  0.4383   1  2.904       7    20 5.583
```

There are 7 points with high influence.

(f) Refit the additive multiple regression model without any points you identified as influential. Compare the coefficients of this fitted model to the previously fitted model.

```
summary(original_model)
```

```
##
## Call:
## lm(formula = lpsa ~ ., data = prostate)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -1.733 -0.371 -0.017  0.414  1.638
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.66934    1.29639    0.52   0.6069
## lcavol       0.58702    0.08792    6.68  2.1e-09 ***
## lweight      0.45447    0.17001    2.67   0.0090 **
## age         -0.01964    0.01117   -1.76   0.0823 .
## lbph         0.10705    0.05845    1.83   0.0704 .
## svi          0.76616    0.24431    3.14   0.0023 **
## lcp         -0.10547    0.09101   -1.16   0.2496
## gleason      0.04514    0.15746    0.29   0.7750
## pgg45        0.00453    0.00442    1.02   0.3089
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.708 on 88 degrees of freedom
```

```
## Multiple R-squared:  0.655,  Adjusted R-squared:  0.623
## F-statistic: 20.9 on 8 and 88 DF,  p-value: <2e-16
```

```
pruned_model = lm(lpsa ~ .,
                  prostate,
                  subset=cooks.distance(original_model) < 4 / length(cooks.distance(original_model)))
summary(pruned_model)
```

```
##
## Call:
## lm(formula = lpsa ~ ., data = prostate, subset = cooks.distance(original_model) <
##     4/length(cooks.distance(original_model)))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.2501 -0.2838  0.0045  0.3846  1.3773
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.24608    1.17544   -0.21  0.83470
## lcavol       0.56499    0.07666    7.37  1.3e-10 ***
## lweight      0.54443    0.17842    3.05  0.00308 **
## age         -0.01856    0.00956   -1.94  0.05554 .
## lbph         0.13328    0.05172    2.58  0.01179 *
## svi          0.74475    0.21193    3.51  0.00072 ***
## lcp         -0.15542    0.07901   -1.97  0.05261 .
## gleason      0.11472    0.13678    0.84  0.40407
## pgg45        0.00665    0.00399    1.67  0.09961 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.589 on 81 degrees of freedom
## Multiple R-squared:  0.721,  Adjusted R-squared:  0.693
## F-statistic: 26.1 on 8 and 81 DF,  p-value: <2e-16
```

We can notice that the p-values for the significance of most coefficients have dropped.

(g) Create a data frame that stores the observations that were "removed" because they were influential. Use the two models you have fit to make predictions with these observations. Comment on the difference between these two sets of predictions.

```
newdata = data.frame(
  prostate[cooks.distance(original_model) > 4 / length(cooks.distance(original_model)),])
predict(original_model, newdata=newdata, level=0.9, interval="confidence")
```

```
##      fit    lwr   upr
## 32 2.875 2.1982 3.552
## 39 3.947 3.6399 4.254
## 47 4.081 3.6239 4.538
## 69 1.325 0.9024 1.747
## 95 3.618 3.1749 4.060
## 96 4.188 3.7831 4.593
## 97 4.092 3.6893 4.495
```

```
predict(pruned_model, newdata=newdata, level=0.9, interval="confidence")
```

```
##      fit    lwr   upr
```

```
## 32 3.106 2.3764 3.837
## 39 3.898 3.5957 4.200
## 47 4.284 3.8557 4.712
## 69 1.340 0.9083 1.772
## 95 3.327 2.9001 3.754
## 96 4.220 3.8469 4.593
## 97 3.905 3.5115 4.299
```

Exercise 3 (Why Bother?)

Why do we care about violations of assumptions? One key reason is that the distributions of the parameter esimators that we have used are all reliant on these assumptions. When the assumptions are violated, the distributional results are not correct, so our tests are garbage. **Garbage In, Garbage Out!**

Consider the following setup that we will use for the remainder of the exercise. We choose a sample size of 50.

```
n = 50
set.seed(420)
x_1 = runif(n, 0, 5)
x_2 = runif(n, -2, 2)
```

Consider the model,

$$Y = 4 + 1x_1 + 0x_2 + \epsilon.$$

That is,

- $\beta_0 = 4$
- $\beta_1 = 1$
- $\beta_2 = 0$

We now simulate `y_1` in a manner that does **not** violate any assumptions, which we will verify. In this case $\epsilon \sim N(0, 1)$.

```
set.seed(83)
library(lmtest)
y_1 = 4 + 1 * x_1 + 0 * x_2 + rnorm(n = n, mean = 0, sd = 1)
fit_1 = lm(y_1 ~ x_1 + x_2)
bptest(fit_1)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  fit_1
## BP = 4.4, df = 2, p-value = 0.1
```

Then, we simulate `y_2` in a manner that **does** violate assumptions, which we again verify. In this case $\epsilon \sim N(0, \sigma = |x_2|)$.

```
set.seed(83)
y_2 = 4 + 1 * x_1 + 0 * x_2 + rnorm(n = n, mean = 0, sd = abs(x_2))
fit_2 = lm(y_2 ~ x_1 + x_2)
bptest(fit_2)
```

```
##
##  studentized Breusch-Pagan test
```

```
##
## data:  fit_2
## BP = 4.9, df = 2, p-value = 0.08
```

(a) Use the following code after changing `birthday` to your birthday.

```
num_sims = 2500
p_val_1 = rep(0, num_sims)
p_val_2 = rep(0, num_sims)
birthday = 19991124
set.seed(birthday)

for (i in 1:num_sims) {
  y_1 = 4 + 1 * x_1 + 0 * x_2 + rnorm(n = n, mean = 0, sd = 1)
  fit_1 = lm(y_1 ~ x_1 + x_2)
  p_val_1[i] = summary(fit_1)$coef[3, 4]
  y_2 = 4 + 1 * x_1 + 0 * x_2 + rnorm(n = n, mean = 0, sd = abs(x_2))
  fit_2 = lm(y_2 ~ x_1 + x_2)
  p_val_2[i] = summary(fit_2)$coef[3, 4]
}
```

Repeat the above process of generating `y_1` and `y_2` as defined above, and fit models with each as the response `2500` times. Each time, store the p-value for testing,

$$\beta_2 = 0,$$

using both models, in the appropriate variables defined above. (You do not need to use a data frame as we have in the past. Although, feel free to modify the code to instead use a data frame.)

(b) What proportion of the `p_val_1` values is less than 0.01? Less than 0.05? Less than 0.10? What proportion of the `p_val_2` values is less than 0.01? Less than 0.05? Less than 0.10? Arrange your results in a table. Briefly explain these results.

```
m11 = sum(p_val_1 < 0.01) / num_sims
m12 = sum(p_val_1 < 0.05) / num_sims
m13 = sum(p_val_1 < 0.10) / num_sims

m21 = sum(p_val_2 < 0.01) / num_sims
m22 = sum(p_val_2 < 0.05) / num_sims
m23 = sum(p_val_2 < 0.10) / num_sims

knitr::kable(data.frame("Cutoff"=c(0.01, 0.05, 0.10), "Model 1"=c(m11, m12, m13), "Model 2"=c(m21, m22,
```

Cutoff	Model.1	Model.2
0.01	0.0092	0.0356
0.05	0.0460	0.1116
0.10	0.1012	0.1796

In the second model, which violates our LINE assumptions, we are much more prone to assuming that β_2 is a non-zero parameter.

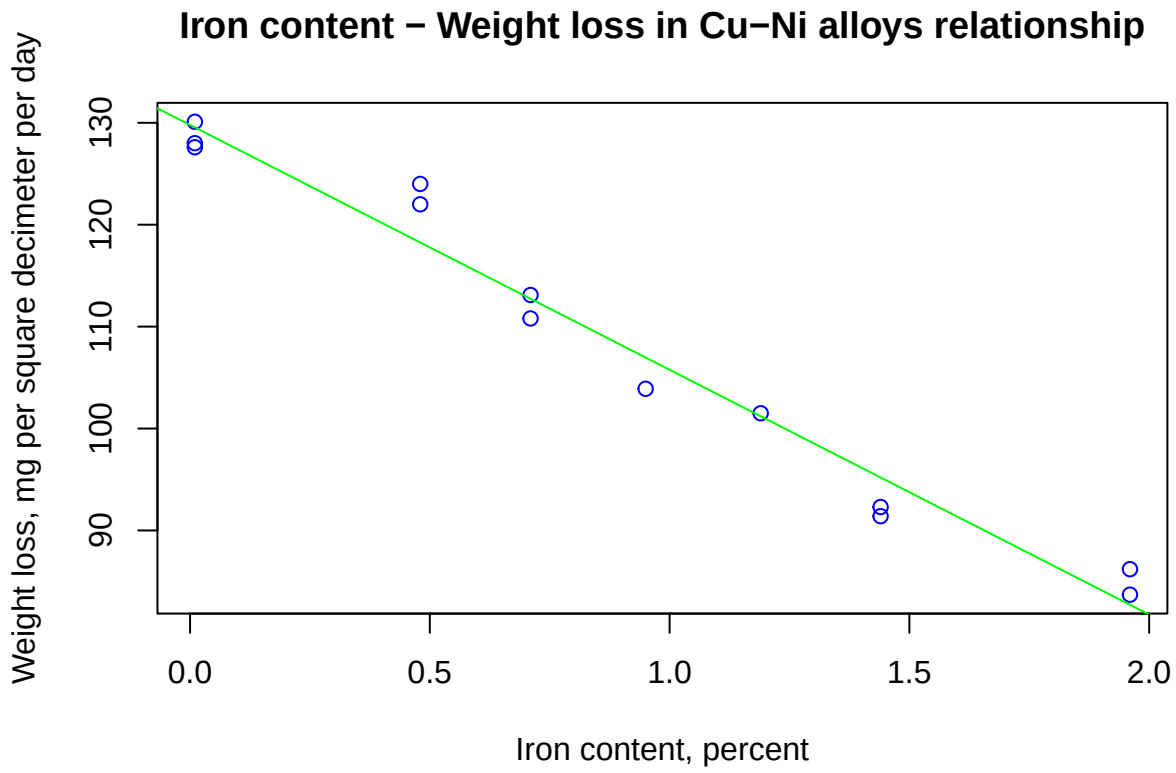
Exercise 4 (Corrosion Data)

For this exercise, we will use the `corrosion` data, which can be found in the `faraway` package. After loading the `faraway` package, use `?corrosion` to learn about this dataset.

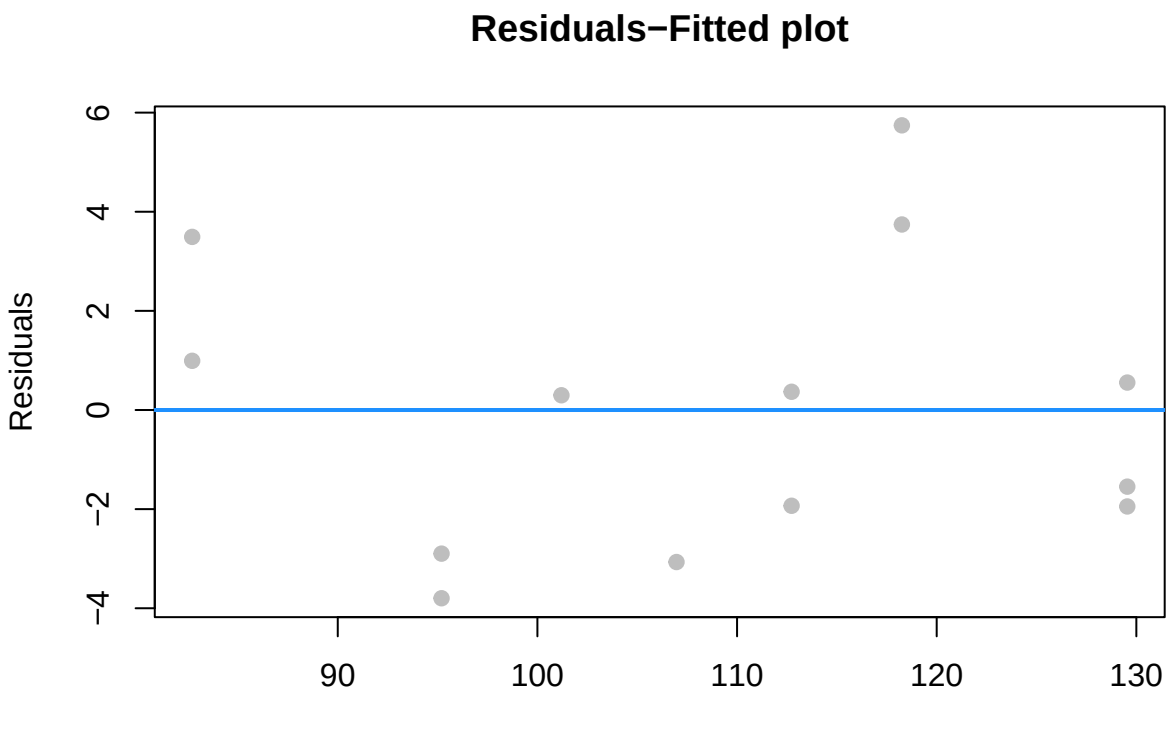
```
library(faraway)
```

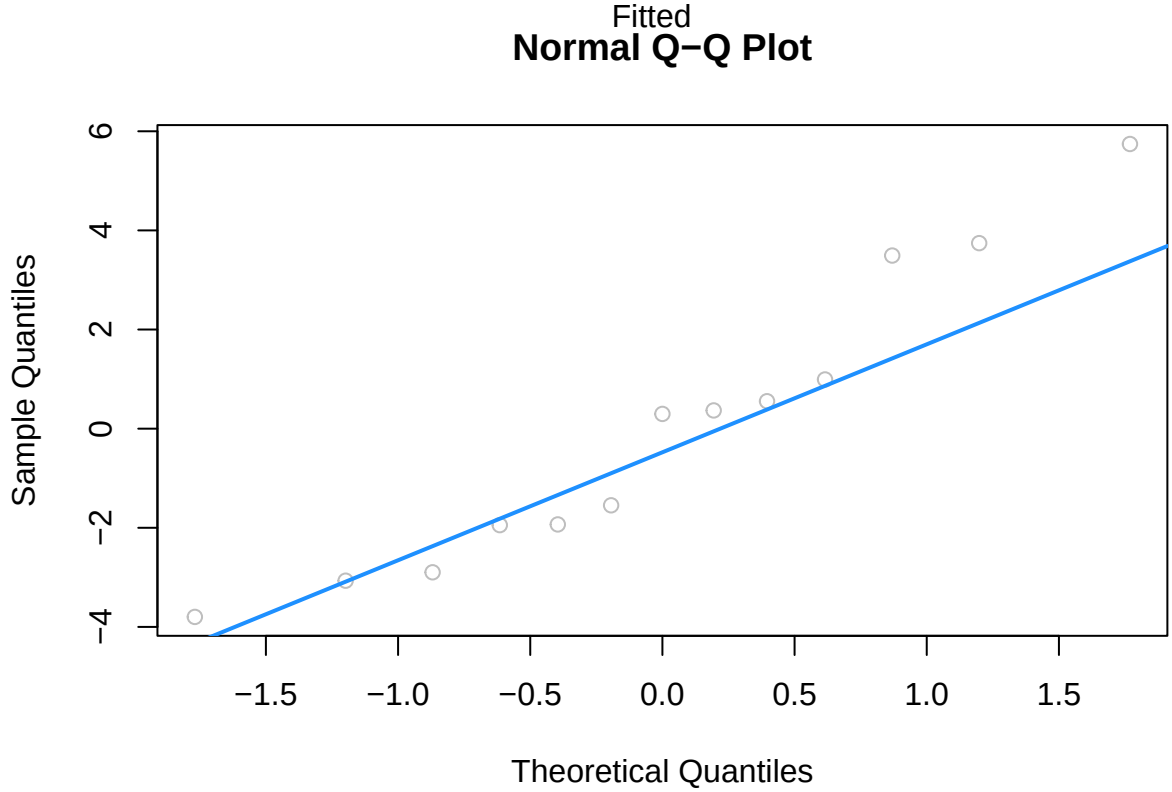
(a) Fit a simple linear regression with `loss` as the response and `Fe` as the predictor. Plot a scatterplot and add the fitted line. Check the assumptions of this model.

```
m = lm(loss ~ Fe, corrosion)
plot(corrosion$Fe,
     corrosion$loss,
     main="Iron content - Weight loss in Cu-Ni alloys relationship",
     xlab="Iron content, percent",
     ylab="Weight loss, mg per square decimeter per day",
     col="blue")
abline(m, col="green")
```




```
diagnostics(m)
```






```
## $p_val
## [1] 0.3733
##
## $decision
## [1] "Fail to Reject"
```

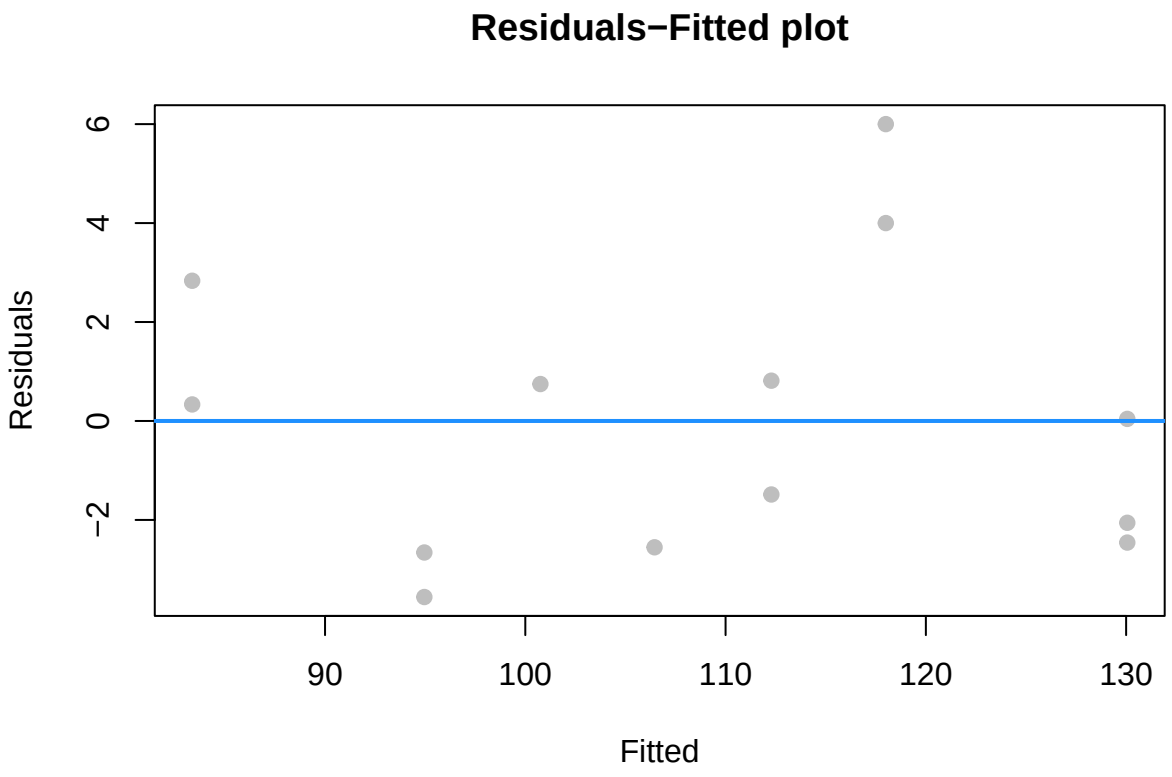
```
bptest(m)
```

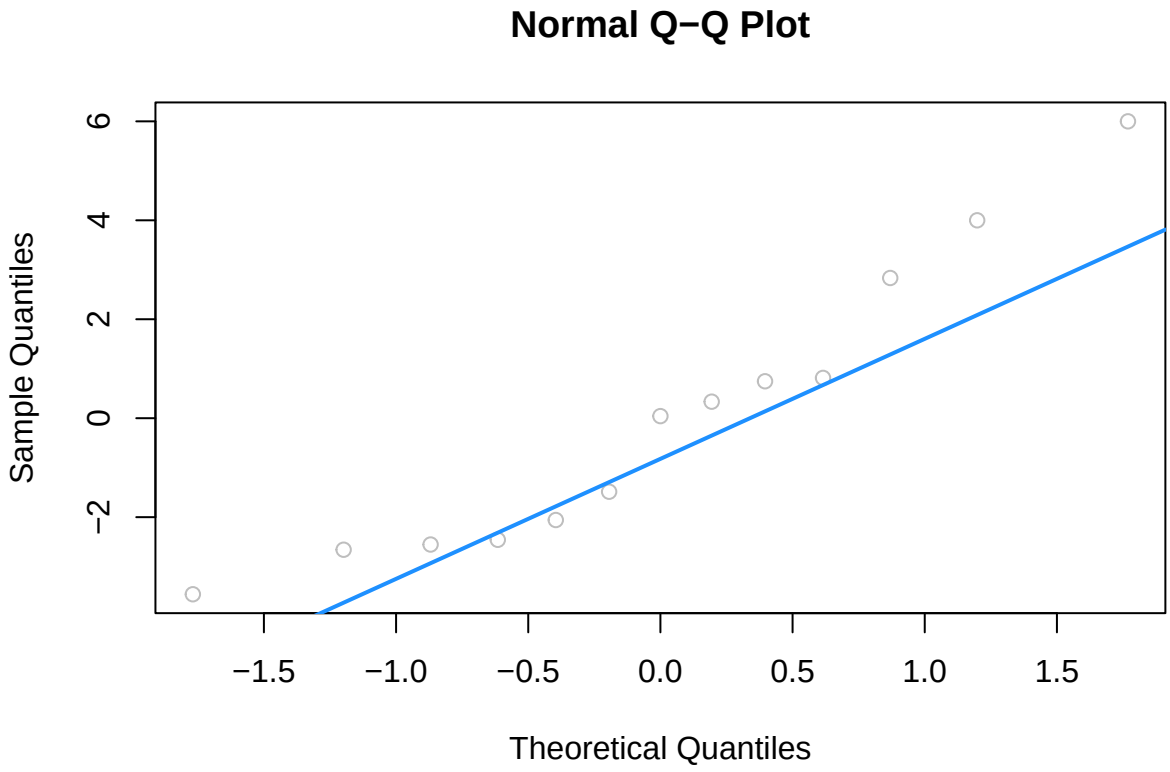
```
##
##  studentized Breusch-Pagan test
##
## data:  m
## BP = 0.025, df = 1, p-value = 0.9
```

Based on Shapiro-Wilk test and the Breusch-Pagan test, we do not reject the assumptions of homoscedasticity and normality.

(b) Fit higher order polynomial models of degree 2, 3, and 4. For each, plot a fitted versus residuals plot and comment on the constant variance assumption. Based on those plots, which of these three models do you think are acceptable? Use a statistical test(s) to compare the models you just chose. Based on the test, which is preferred? Check the normality assumption of this model. Identify any influential observations of this model.

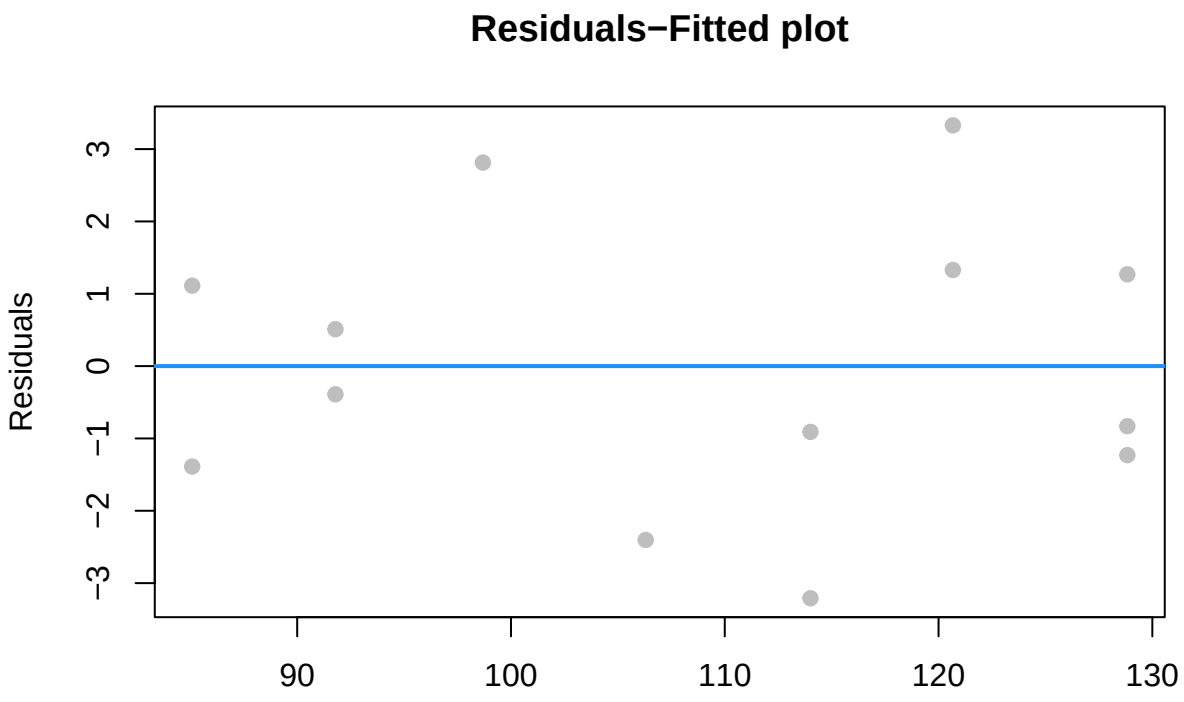
```
p2 = lm(loss ~ Fe + I(Fe ^ 2), corrosion)
p3 = lm(loss ~ Fe + I(Fe ^ 2) + I(Fe ^ 3), corrosion)
p4 = lm(loss ~ Fe + I(Fe ^ 2) + I(Fe ^ 3) + I(Fe ^ 4), corrosion)
diagnostics(p2)
```

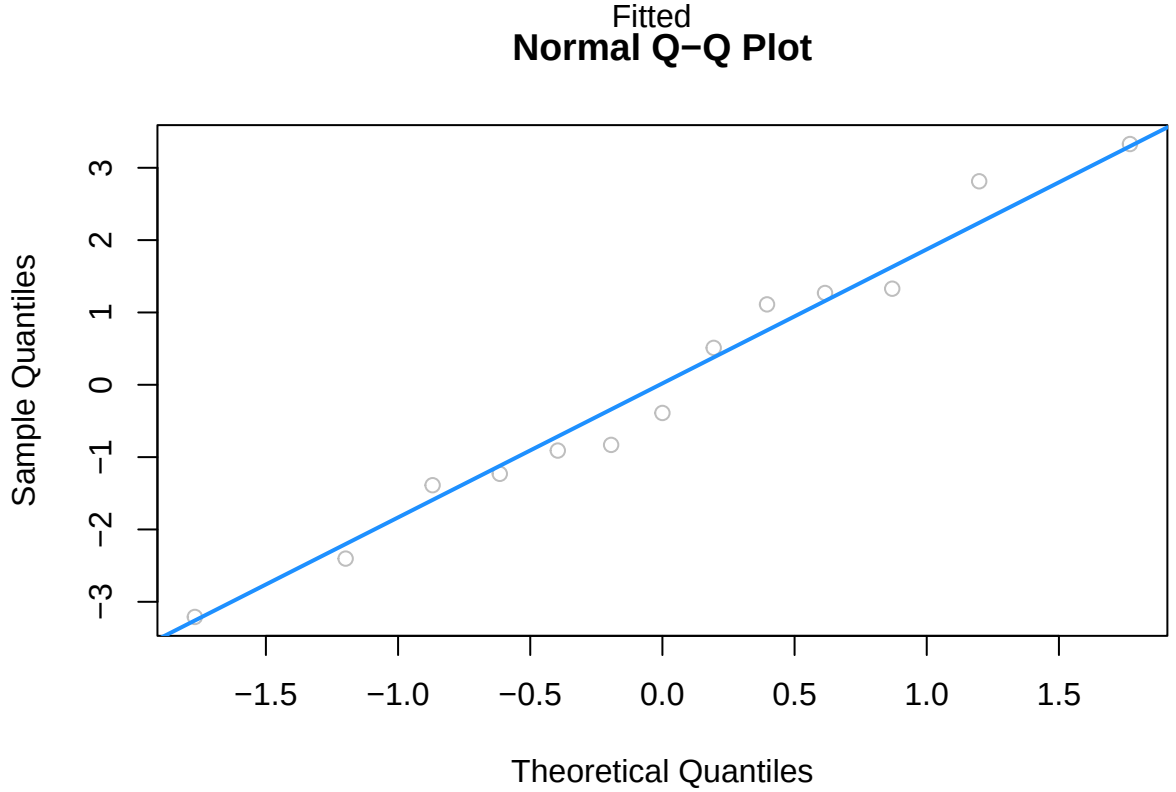





```
## $p_val
## [1] 0.2565
##
## $decision
## [1] "Fail to Reject"
```

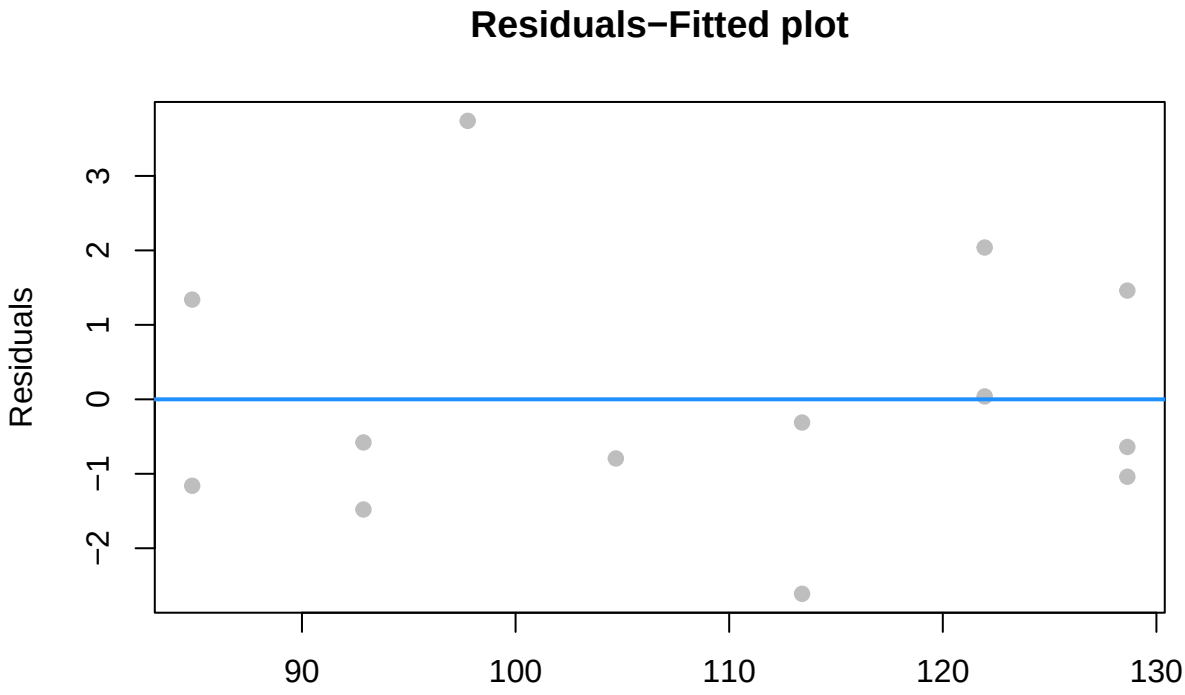
```
diagnostics(p3)
```

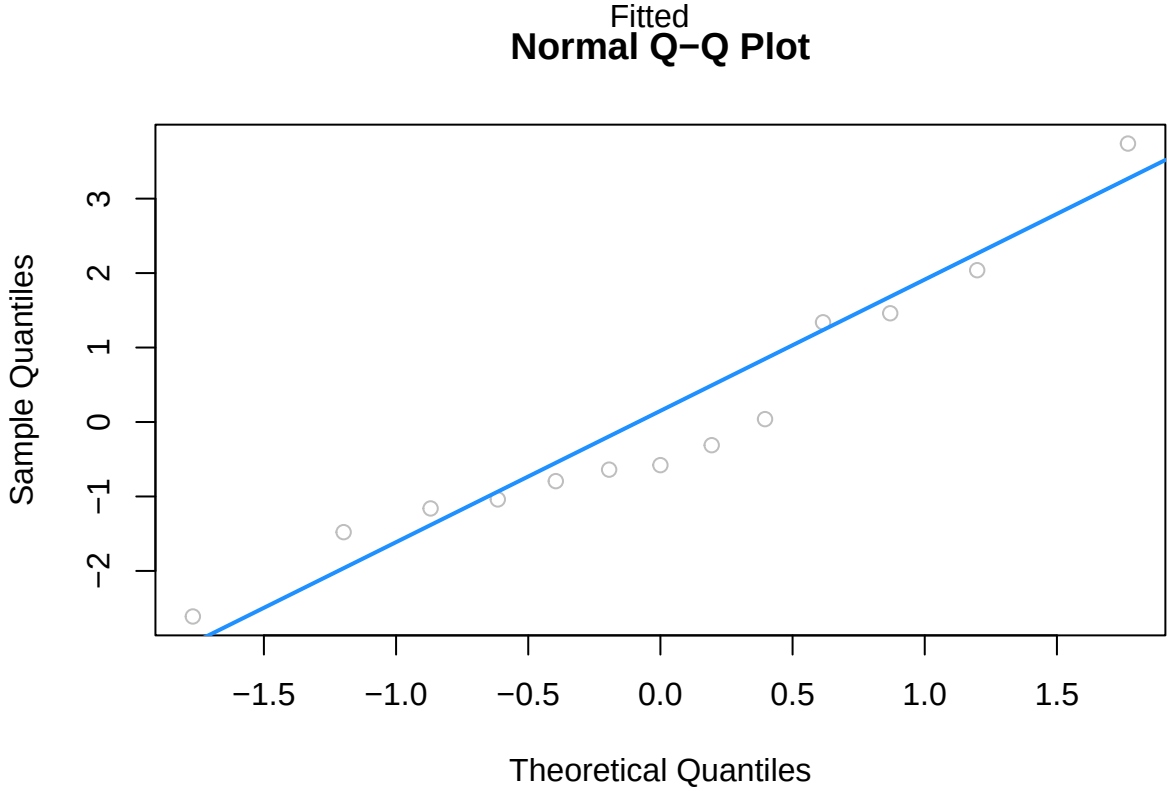





```
## $p_val
## [1] 0.9085
##
## $decision
## [1] "Fail to Reject"
```

```
diagnostics(p4)
```






```
## $p_val
## [1] 0.4022
##
## $decision
```

```
## [1] "Fail to Reject"
```

While the dataset is small and it is hard to visually make a call for one model being than the other, it appears to me that the 3rd degree polynomial is the least problematic in terms of the constant variance assumption.

```
corrosion[cooks.distance(p2) > 4 / length(cooks.distance(p2)),]
```

```
##      Fe loss
## 13 1.96 86.2
```

```
corrosion[cooks.distance(p3) > 4 / length(cooks.distance(p3)),]
```

```
## [1] Fe   loss
## <0 rows> (or 0-length row.names)
```

```
corrosion[cooks.distance(p4) > 4 / length(cooks.distance(p4)),]
```

```
##     Fe  loss
## 5 1.19 101.5
```

One can notice that unlike the 2nd- and 4th-degree polynomials, the 3rd-degree polynomial model has no influential points.

Exercise 5 (Diamonds)

The data set `diamonds` from the `ggplot2` package contains prices and characteristics of 54,000 diamonds. For this exercise, use `price` as the response variable y, and `carat` as the predictor x. Use `?diamonds` to learn more.

```
library(ggplot2)
```

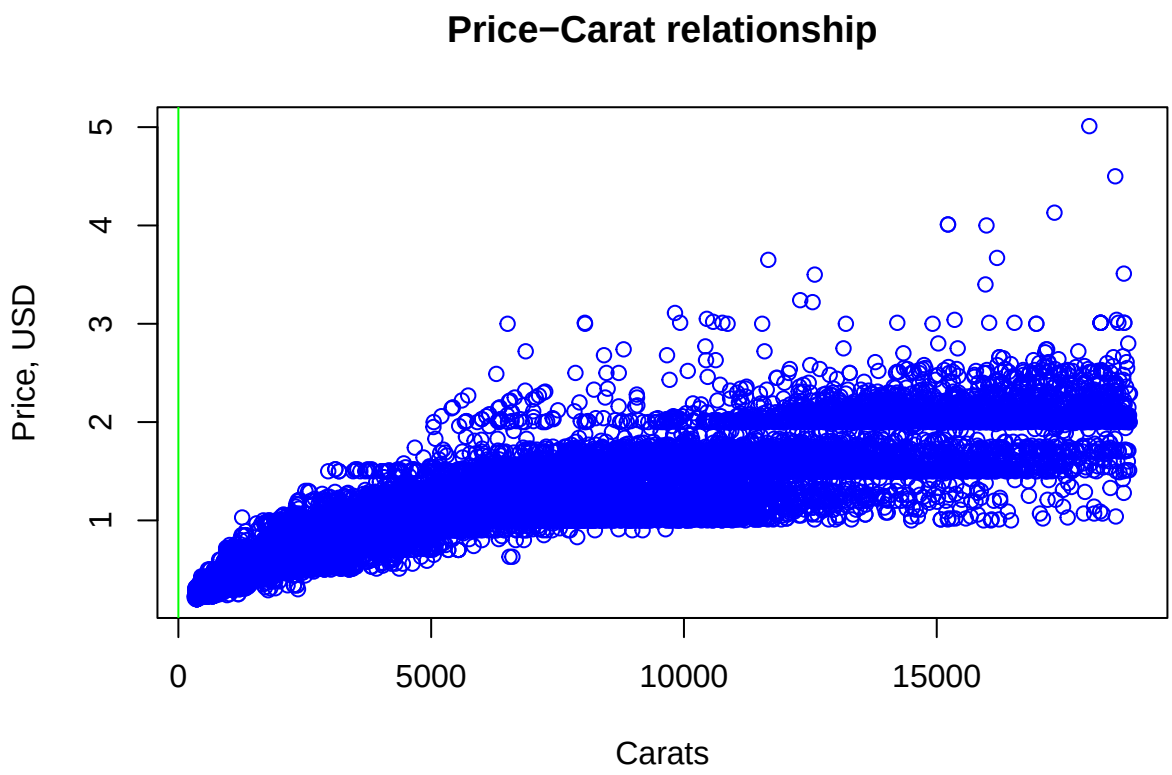
(a) Fit a linear model with `price` as the response variable y, and `carat` as the predictor x. Return the summary information of this model.

```
m = lm(price ~ carat, diamonds)
summary(m)
```

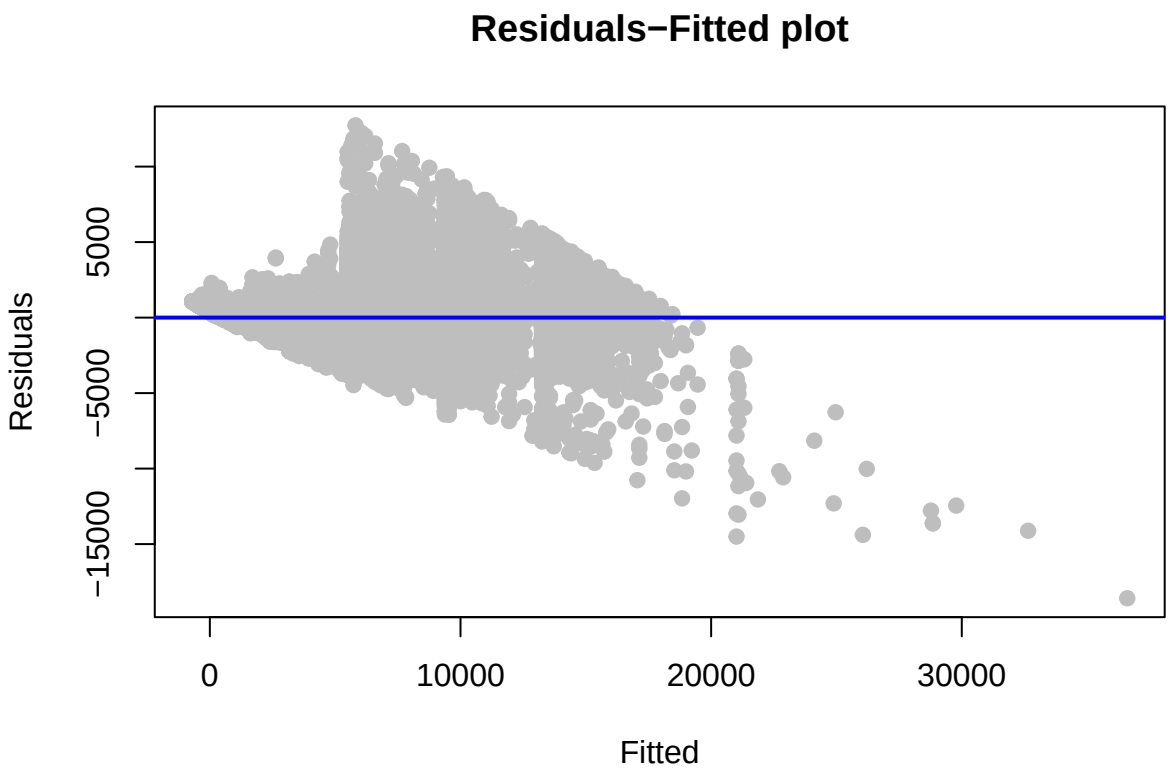
```
##
## Call:
## lm(formula = price ~ carat, data = diamonds)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -18585   -805    -19    537  12732
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -2256.4       13.1    -173   <2e-16 ***
## carat         7756.4       14.1     551   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1550 on 53938 degrees of freedom
## Multiple R-squared:  0.849,  Adjusted R-squared:  0.849
## F-statistic: 3.04e+05 on 1 and 53938 DF,  p-value: <2e-16
```

(b) Plot a scatterplot of price versus carat and add the line for the fitted model in part **(a)**. Using a fitted versus residuals plot and/or a Q-Q plot, comment on the diagnostics.

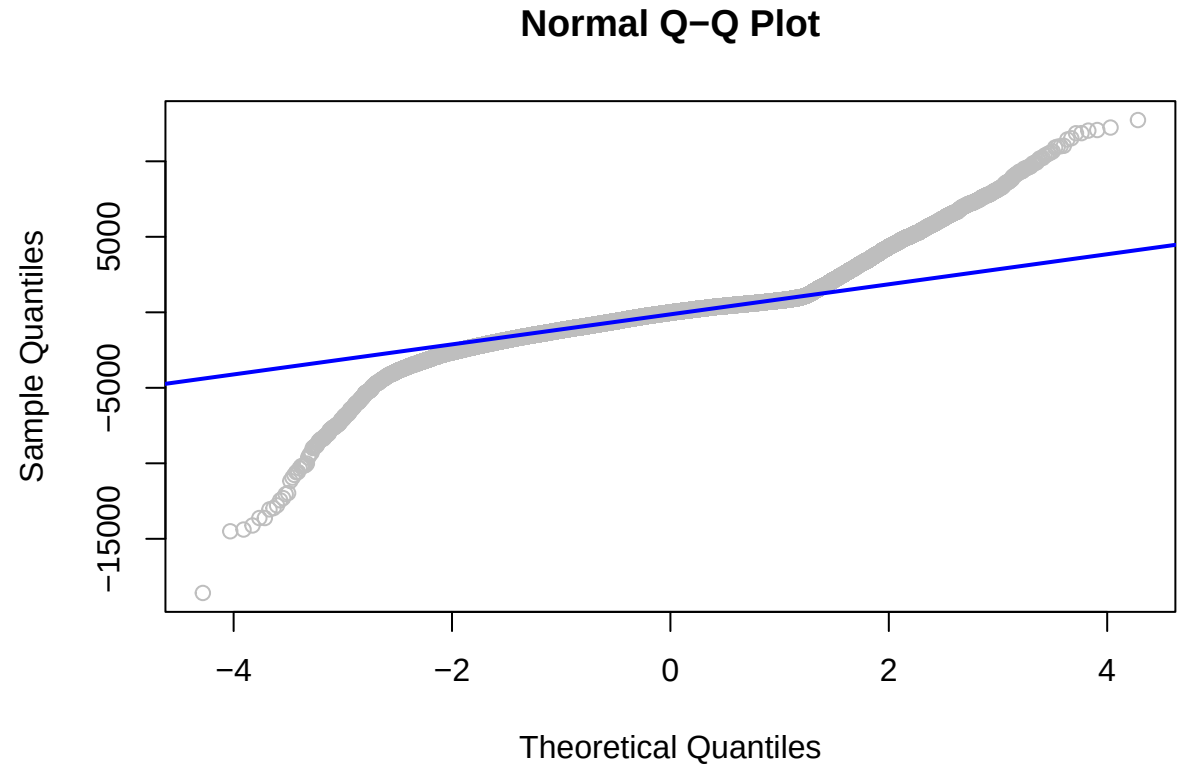
```
plot(diamonds$price,
     diamonds$carat,
     main="Price-Carat relationship",
     xlab="Carats",
     ylab="Price, USD",
     col="blue")
abline(m, col="green")
```




```
plot(fitted(m),
     resid(m),
     col="grey",
     pch=20,
     cex=1.5,
     xlab="Fitted",
     ylab="Residuals",
     main="Residuals-Fitted plot")
abline(h=0, lty=1, col="blue", lwd=2)
```




```
qqnorm(resid(m), main="Normal Q-Q Plot", col="grey")
qqline(resid(m), col="blue", lwd=2)
```

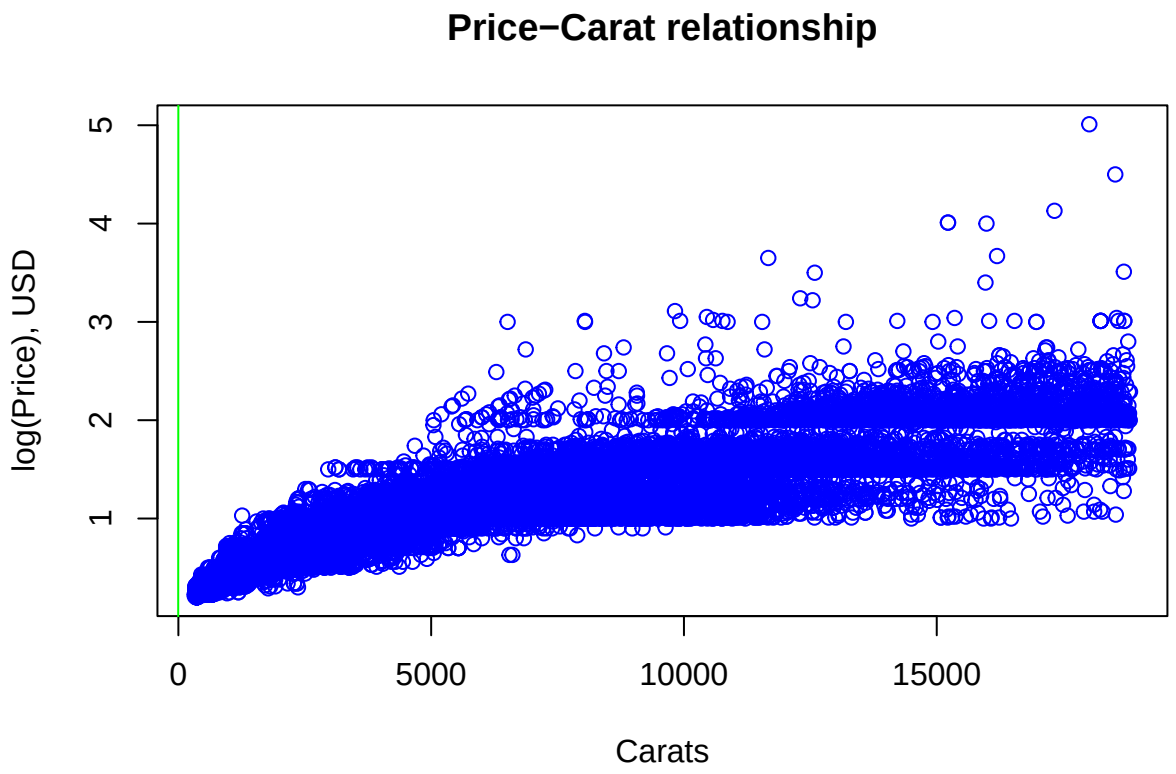


The dataset clearly fails the normality assumption and the equal variance assumption, leading to a poor fit.

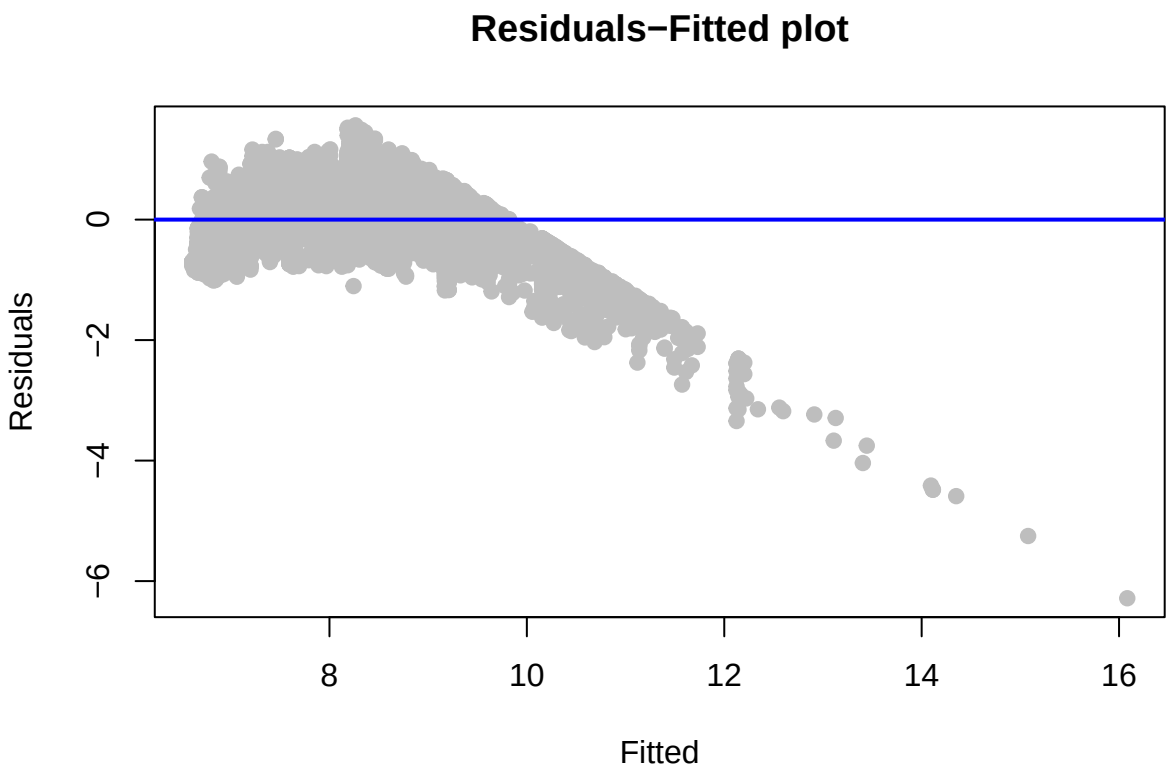
(c) Seeing as the price stretches over several orders of magnitude, it seems reasonable to try a log transformation

of the response. Fit a model with a logged response, plot a scatterplot of log-price versus carat and add the line for the fitted model, then use a fitted versus residuals plot and/or a Q-Q plot to comment on the diagnostics of the model.

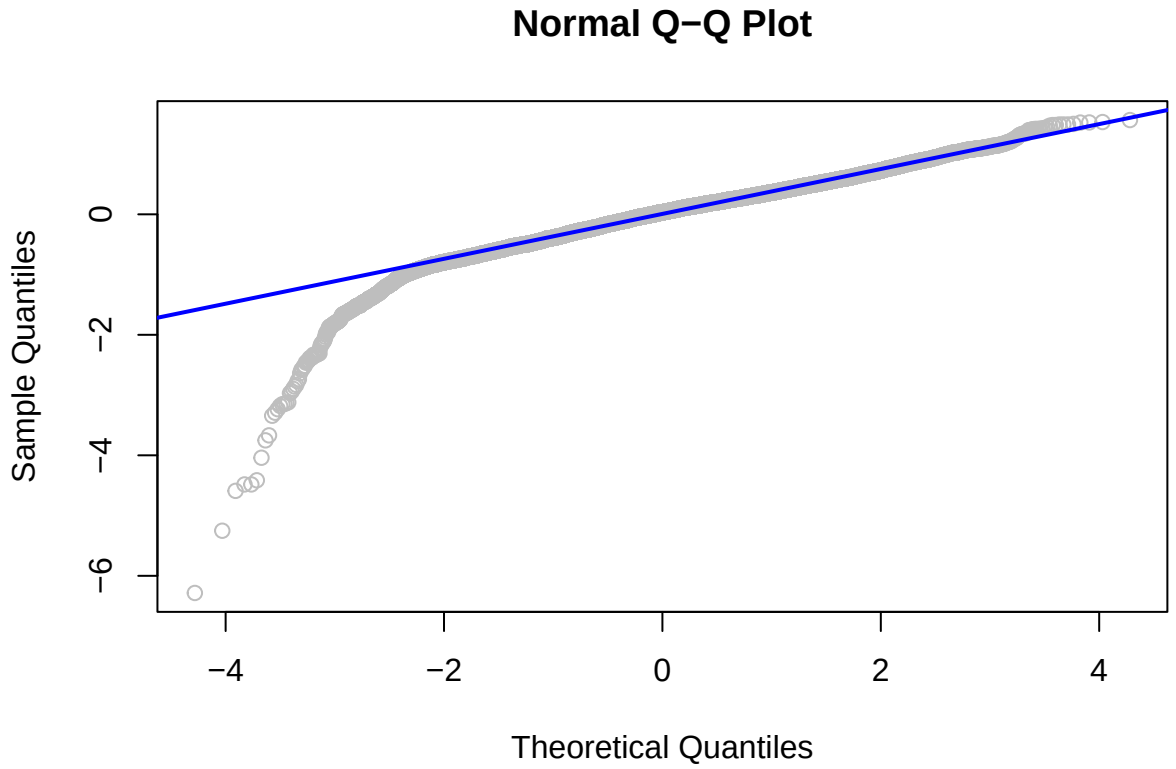
```
log_m = lm(log(price) ~ carat, diamonds)
plot(diamonds$price, diamonds$carat, main="Price-Carat relationship", xlab="Carats", ylab="log(Price), U
abline(log_m, col="green", untf=TRUE)
```




```
plot(fitted(log_m), resid(log_m), col="grey",
     pch=20, cex=1.5, xlab="Fitted", ylab="Residuals", main="Residuals-Fitted plot")
    abline(h=0, lty=1, col="blue", lwd=2)
```

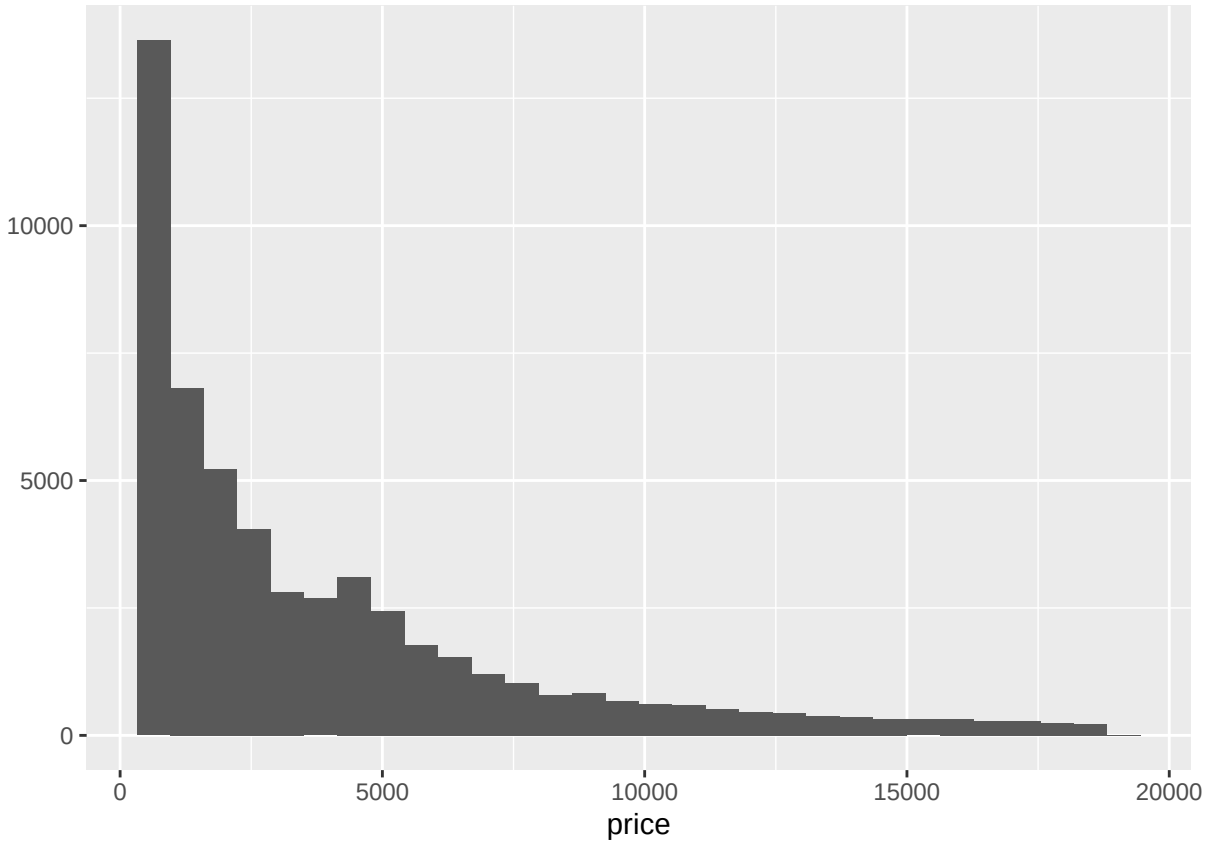



```
qqnorm(resid(log_m), main="Normal Q-Q Plot", col="grey")
qqline(resid(log_m), col="blue", lwd=2)
```



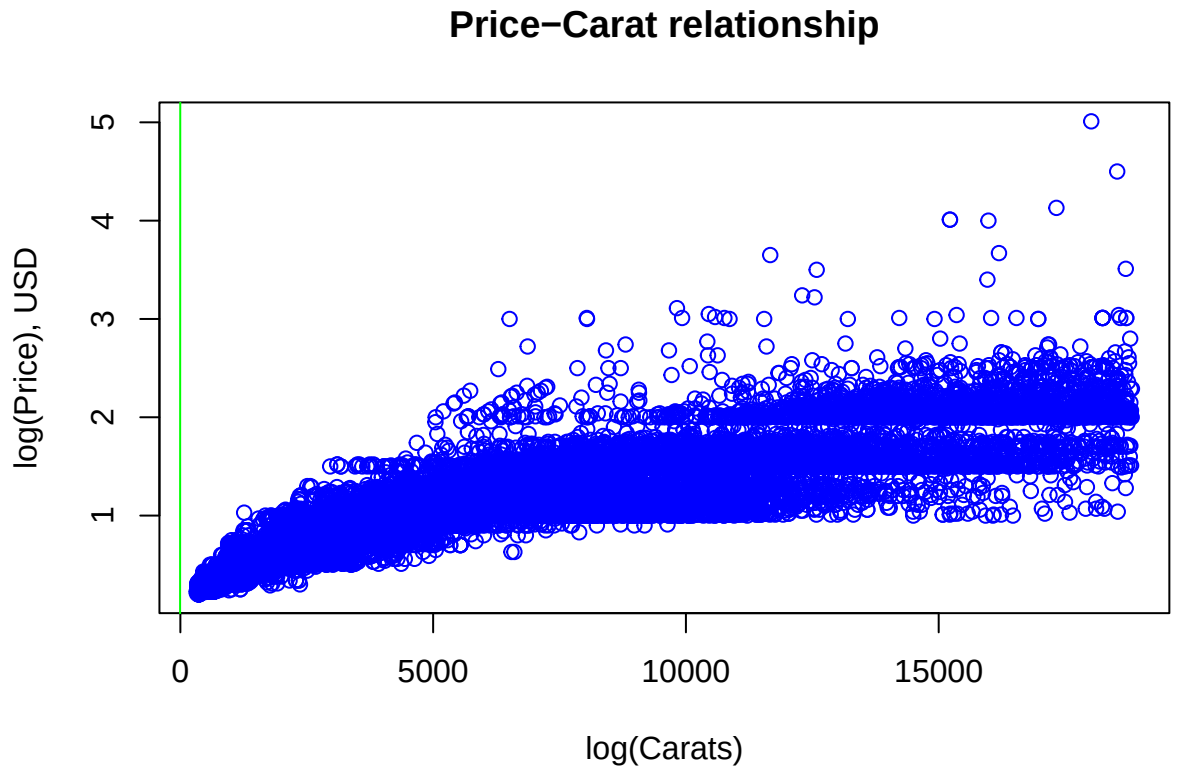
While a resonable guess, the log transformation clearly does not eliminate the issues with the normality and non-constant variance.

```
qplot(price, data = diamonds, bins = 30)
```

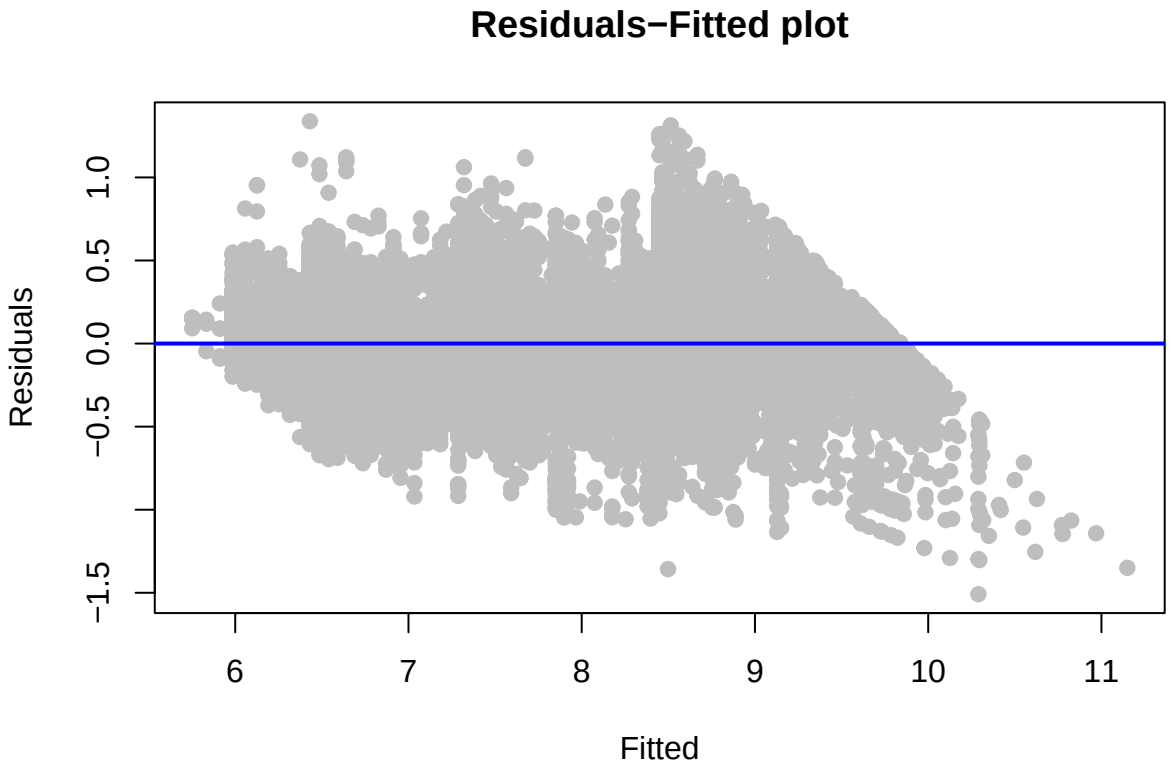


(d) Try adding log transformation of the predictor. Fit a model with a logged response and logged predictor, plot a scatterplot of log-price versus log-carat and add the line for the fitted model, then use a fitted versus residuals plot and/or a Q-Q plot to comment on the diagnostics of the model.

```
log_log_m = lm(log(price) ~ I(log(carat)), diamonds)
plot(diamonds$price, diamonds$carat, main="Price-Carat relationship", xlab="log(Carats)", ylab="log(Pri
abline(log_log_m, col="green", untf=TRUE)
```

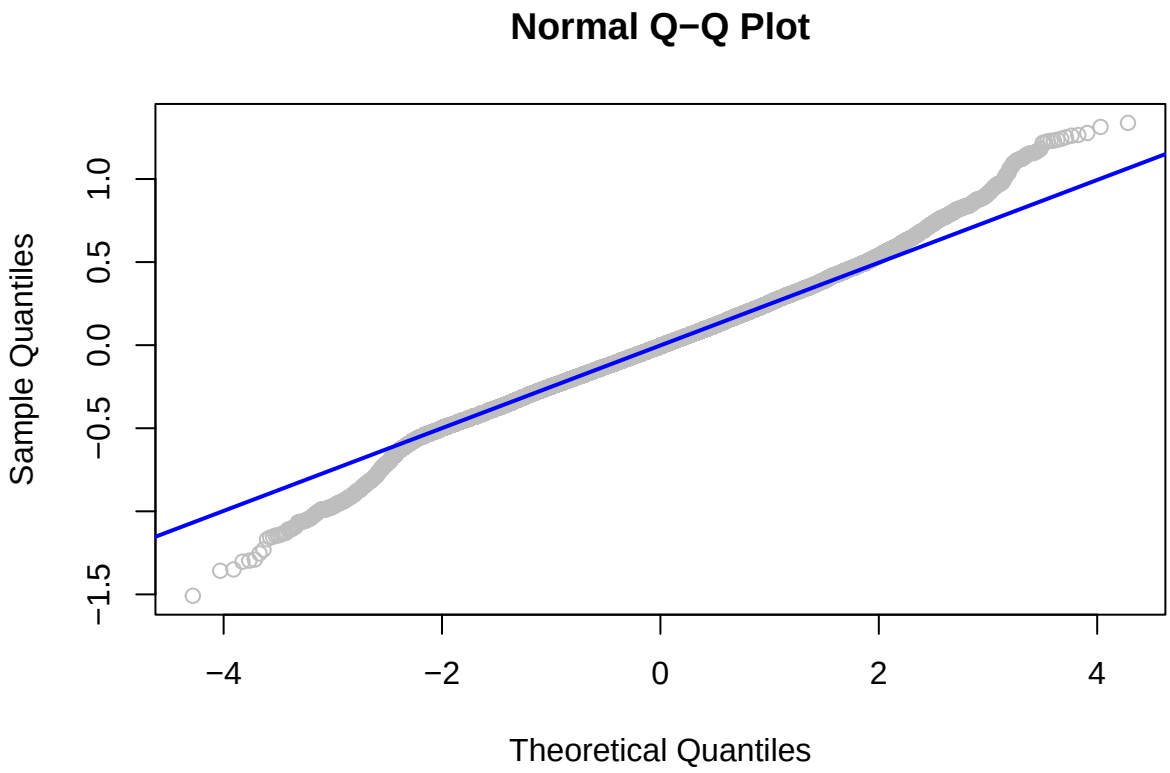



```
plot(fitted(log_log_m), resid(log_log_m), col="grey",
     pch=20, cex=1.5, xlab="Fitted", ylab="Residuals", main="Residuals-Fitted plot")
    abline(h=0, lty=1, col="blue", lwd=2)
```




```
qqnorm(resid(log_log_m), main="Normal Q-Q Plot", col="grey")
```

```
qqline(resid(log_log_m), col="blue", lwd=2)
```



(e) Use the model from part **(d)** to predict the price (in dollars) of a 3-carat diamond. Construct a 99% prediction interval for the price (in dollars).

```
exp(predict(log_log_m, newdata=data.frame(carat=c(3)), level=0.99, interval="prediction"))
```

```
##     fit   lwr   upr
## 1 29429 14959 57894
```