Week 8 - Homework

STAT 420, Summer 2018, Connor Segneri - segneri3

Contents

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.

```
diagnostics = function(model, pcol='grey', lcol='dodgerblue', alpha=0.05, plotit=TRUE, testit=TRUE) {
  output = list()

  if (testit) {
    output$p_val = shapiro.test(resid(model))$p.value
```

```
    if (output$p_val < alpha) {
      output$decision = 'Reject'
    } else {
      output$decision = 'Fail to Reject'
    }
  }

  if (plotit) {
    par(mfrow = c(1,2))

    plot(fitted(model), resid(model), col = pcol, pch = 20,
      xlab = "Fitted", ylab = "Residuals", main = "Fitted versus Residuals")
    abline(h = 0, col = lcol, lwd = 2)

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

  output
}
```

(b) Run the following code.

```
set.seed(420)

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)
```

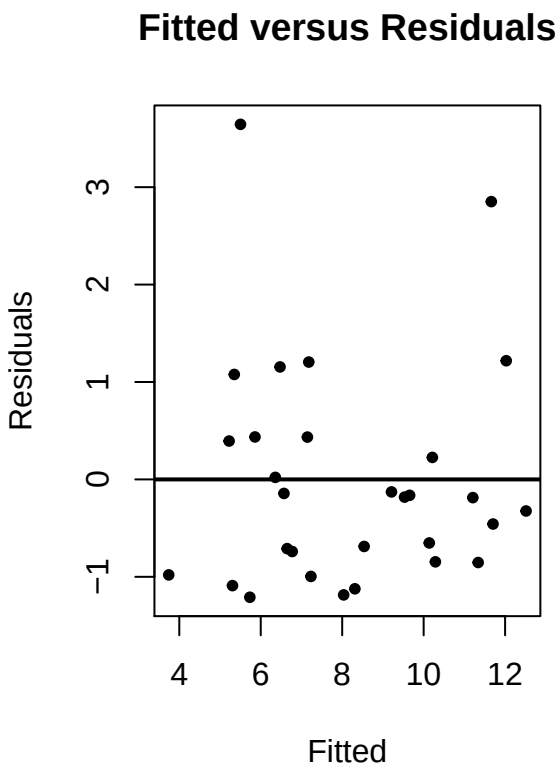
```
diagnostics(fit_1, plotit = FALSE)$p_val
```

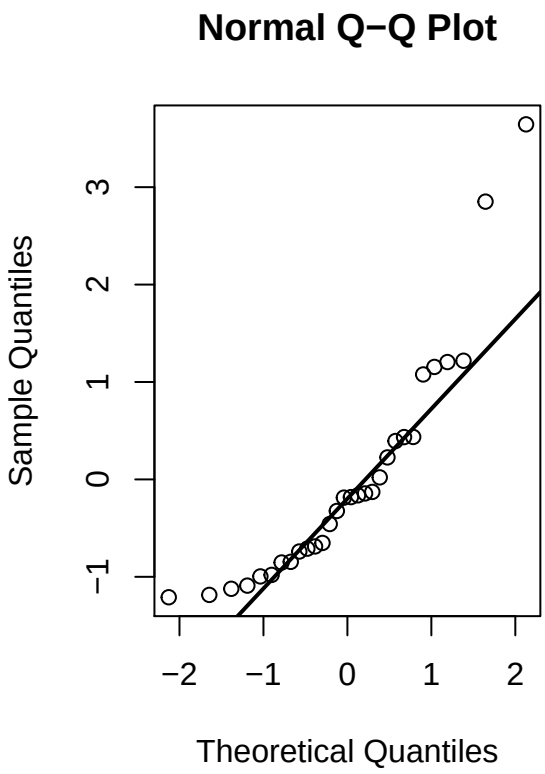
```
## [1] 0.0004299
```

```
diagnostics(fit_2, plotit = FALSE)$decision
```

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

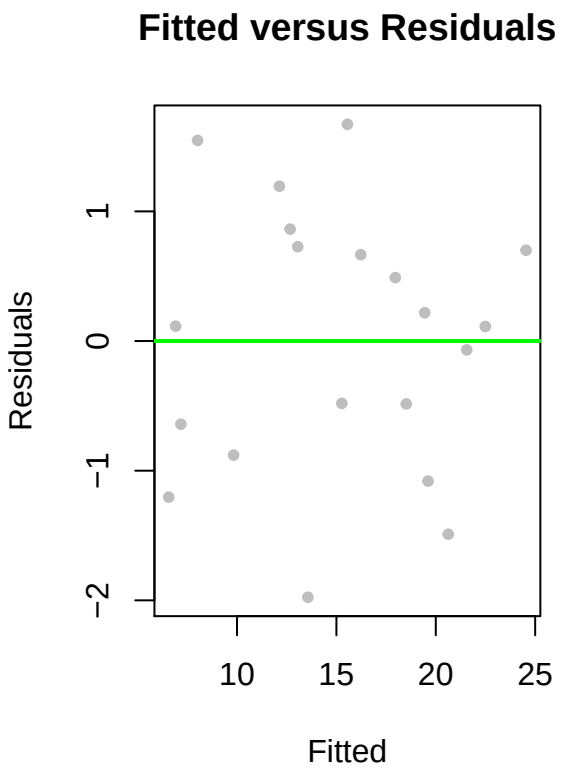
```
diagnostics(fit_1, testit = FALSE, pcol = "black", lcol = "black")
```

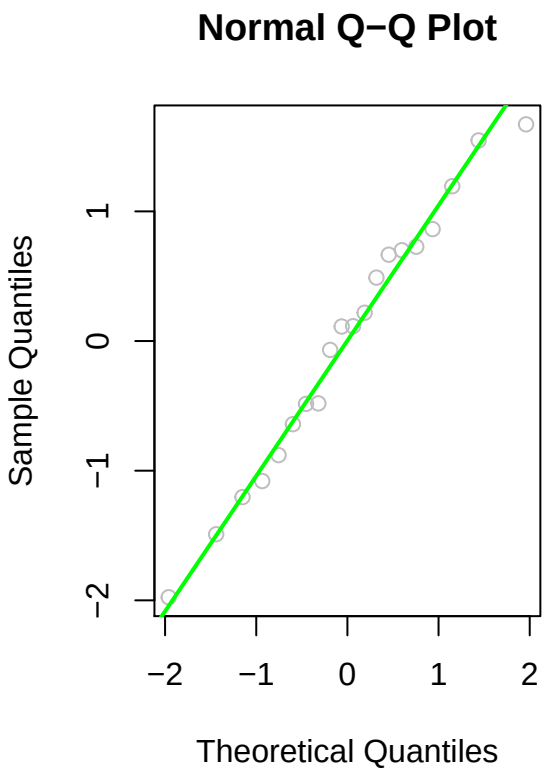





```
## list()
```

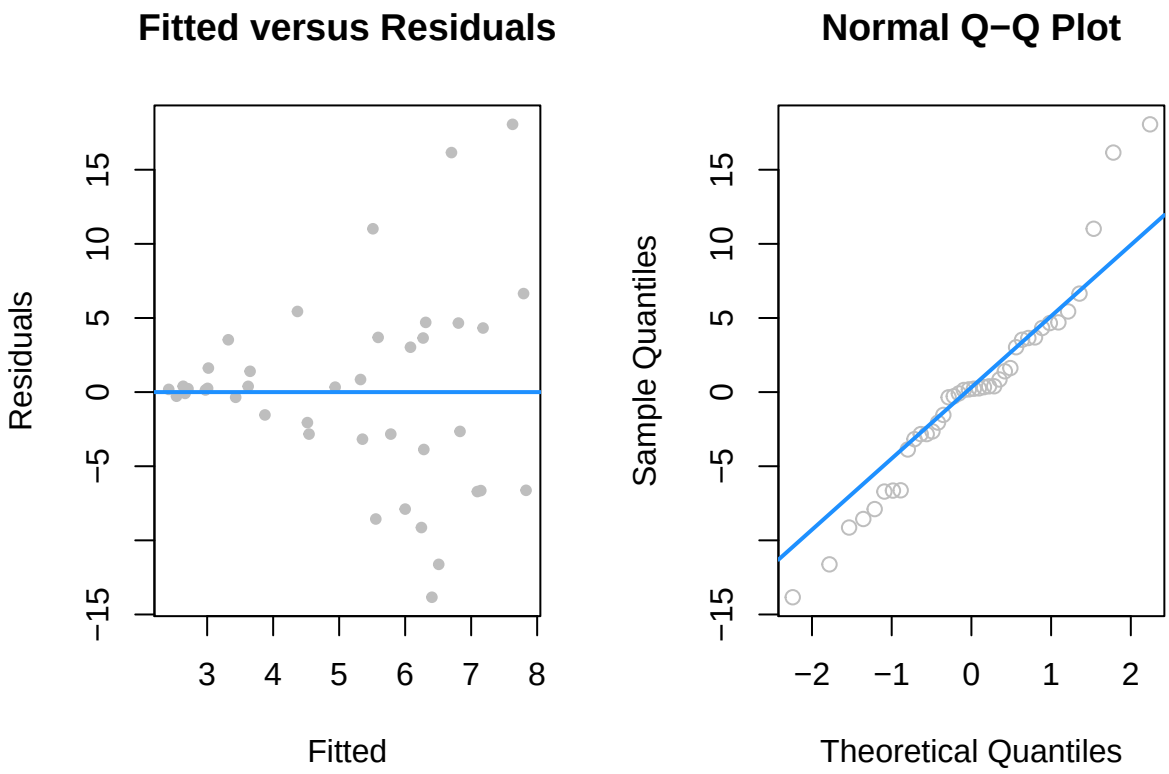
```
diagnostics(fit_2, testit = FALSE, pcol = "grey", lcol = "green")
```






```
## list()
```

```
diagnostics(fit_3)
```




```
## $p_val
## [1] 0.09105
##
## $decision
## [1] "Fail to Reject"
```

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)
```

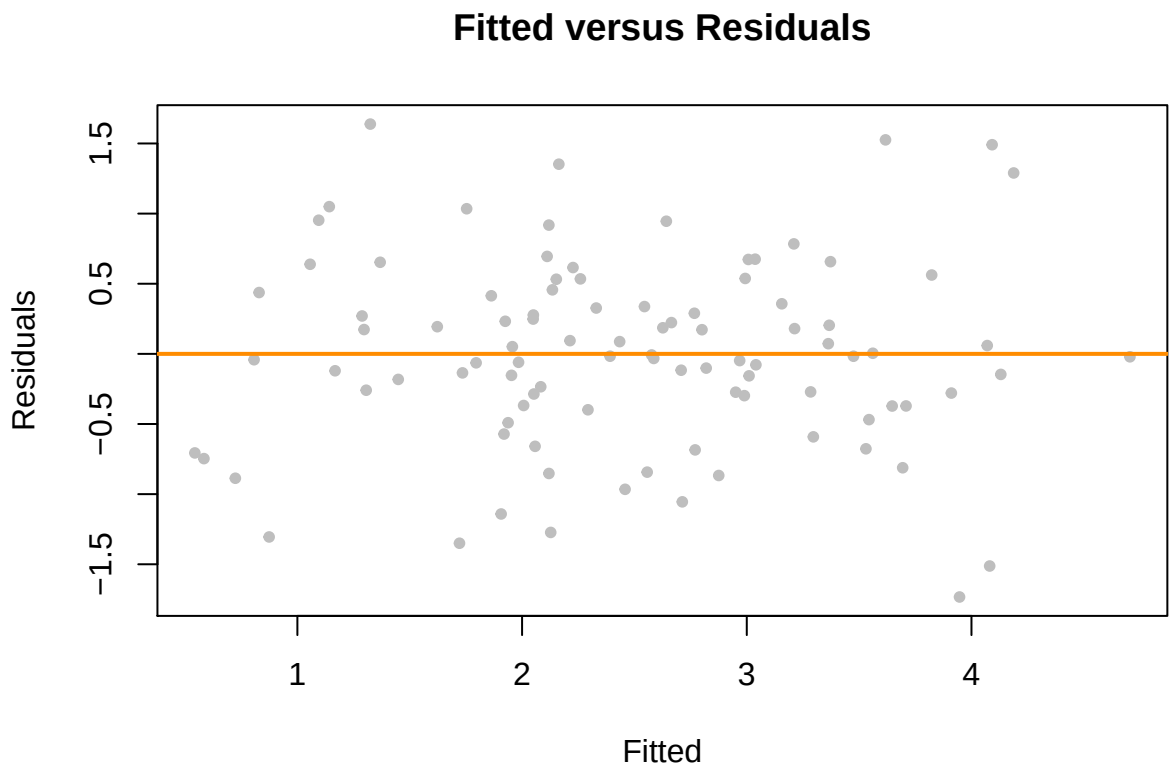
(a) Fit an additive multiple regression model with `lpsa` as the response and the remaining variables in the `prostate` dataset as predictors. Report the R^2 value for this model.

```
model_a = lm(lpsa ~ ., data = prostate)
summary(model_a)$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)
plot(fitted(model_a), resid(model_a), col = "grey", pch = 20,
     xlab = "Fitted", ylab = "Residuals", main = "Fitted versus Residuals")
abline(h = 0, col = "darkorange", lwd = 2)
```



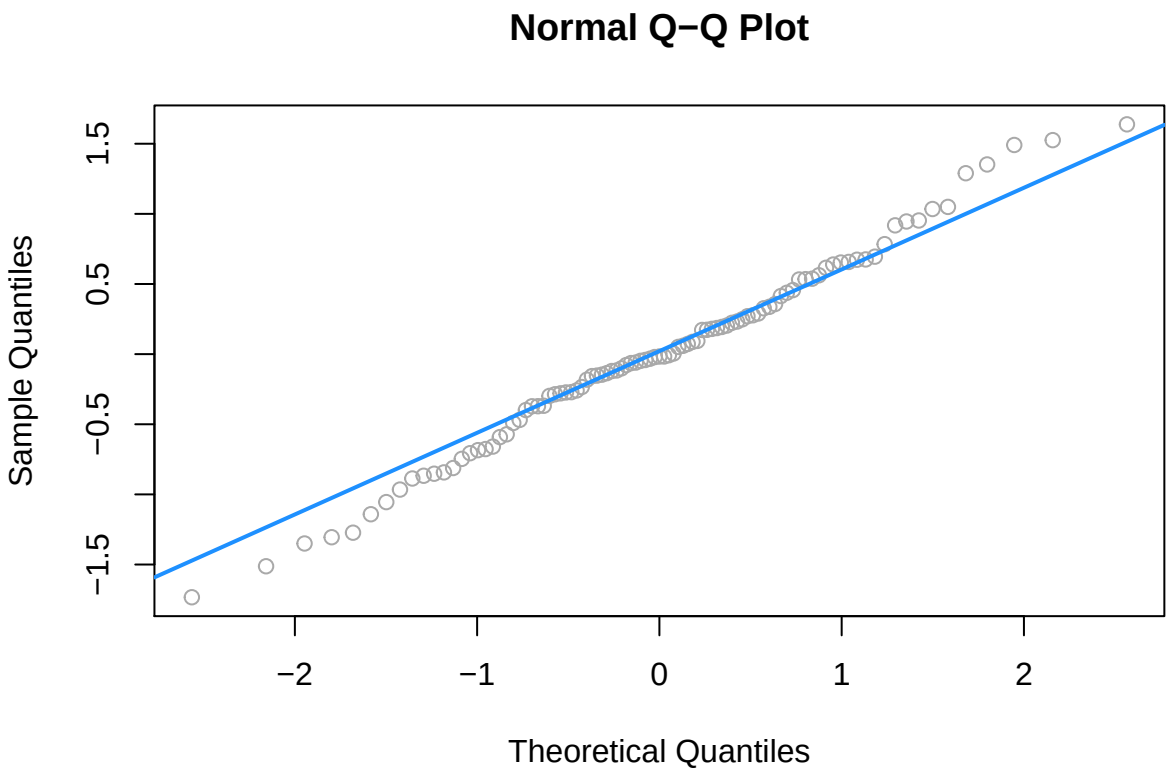

```
bptest(model_a)
```

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

Looking at the fitted versus residuals plot and the p-value of the Breusch-Pagan test, it doesn't look like the constant variance assumption is violated.

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

```
qqnorm(resid(model_a), main = "Normal Q-Q Plot", col = "darkgrey")
qqline(resid(model_a), col = "dodgerblue", lwd = 2)
```




```
shapiro.test(resid(model_a))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(model_a)
## W = 0.99, p-value = 0.8
```

Looking at the Normal Q-Q Plot and the p-value for the Shapiro-Wilk test, it seems as though the data comes from a normal distribution.

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

I'll define "high leverage" as twice the average leverage. These observations with their leverages can be seen below.

```
hatvalues(model_a)[hatvalues(model_a) > 2 * mean(hatvalues(model_a))]
```

```
##     32     37     41     74     92
## 0.3305 0.2184 0.2410 0.1912 0.2092
```

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

I'll define "influential" as having a Cook's Distance value greater than `4/n`, where `n` is the number of observations. These observations and their Cook Distances can be seen below.

```
cooks.distance(model_a)[cooks.distance(model_a) > 4 / length(fitted(model_a))]
```

```
##      32      39      47      69      95      96      97
## 0.12270 0.05202 0.10574 0.10054 0.09874 0.05594 0.07378
```

(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.

```
model_f = lm(lpsa ~ ., data = prostate, subset = !(cooks.distance(model_a) > 4 / length(fitted(model_a)))

coef(model_a)
```

```
## (Intercept)      lcavol     lweight         age        lbph         svi
##    0.669337    0.587022    0.454467   -0.019637    0.107054    0.766157
##         lcp     gleason       pgg45
##   -0.105474    0.045142    0.004525
```

```
coef(model_f)
```

```
## (Intercept)      lcavol     lweight         age        lbph         svi
##    -0.24608     0.56499     0.54443    -0.01856     0.13328     0.74475
##         lcp     gleason       pgg45
##    -0.15542     0.11472     0.00665
```

All of the coefficients vary, some more drastically than others. The sign of the intercept changed.

(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.

```
influential = prostate[cooks.distance(model_a) > 4 / length(fitted(model_a)),]

predict(model_a, newdata = influential)
```

```
##    32    39    47    69    95    96    97
## 2.875 3.947 4.081 1.325 3.618 4.188 4.092
```

```
predict(model_f, newdata = influential)
```

```
##    32    39    47    69    95    96    97
## 3.106 3.898 4.284 1.340 3.327 4.220 3.905
```

The two sets of predictions didn't vary too drastically, although there is a noticeable change across every prediction.

Exercise 3 (Why Bother?)

Why do we care about violations of assumptions? One key reason is that the distributions of the parameter estimators 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 $\epsilon \sim N(0, 1)$.

```
library(lmtest)
set.seed(1)
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(1)
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 = 8.4, df = 2, p-value = 0.01
```

(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 = 19951015
set.seed(birthday)
```

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.)

```
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)

  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_1[i] = summary(fit_1)$coefficients[3,4]
  p_val_2[i] = summary(fit_2)$coefficients[3,4]
}
```

(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.

```
p_val_df = data.frame(
  less_than_0.01 = c(
    length(p_val_1[p_val_1 < 0.01]) / length(p_val_1),
    length(p_val_2[p_val_2 < 0.01]) / length(p_val_2)
  ),
  less_than_0.05 = c(
    length(p_val_1[p_val_1 < 0.05]) / length(p_val_1),
    length(p_val_2[p_val_2 < 0.05]) / length(p_val_2)
  ),
  less_than_0.10 = c(
    length(p_val_1[p_val_1 < 0.10]) / length(p_val_1),
    length(p_val_2[p_val_2 < 0.10]) / length(p_val_2)
  )
)
rownames(p_val_df) = c("p_val_1", "p_val_2")

library(knitr)
library(kableExtra)

p_val_df %>%
  kable() %>%
  kable_styling()
```

	less_than_0.01	less_than_0.05	less_than_0.10
p_val_1	0.0088	0.0460	0.0976
p_val_2	0.0352	0.1064	0.1796

The data frame above shows the proportions of p-values that are less than `0.01`, `0.05`, and `0.10` for the p-values used to determine $\beta_2 = 0$ for both of the models defined in this exercise. For example, this data frame shows that `.88%` of p-values for the first model were less than `0.01`. This is the proportion of models where we would fail to reject the null hypothesis, and the individual value of $beta_2$ would not be considered useful for the predictive model.

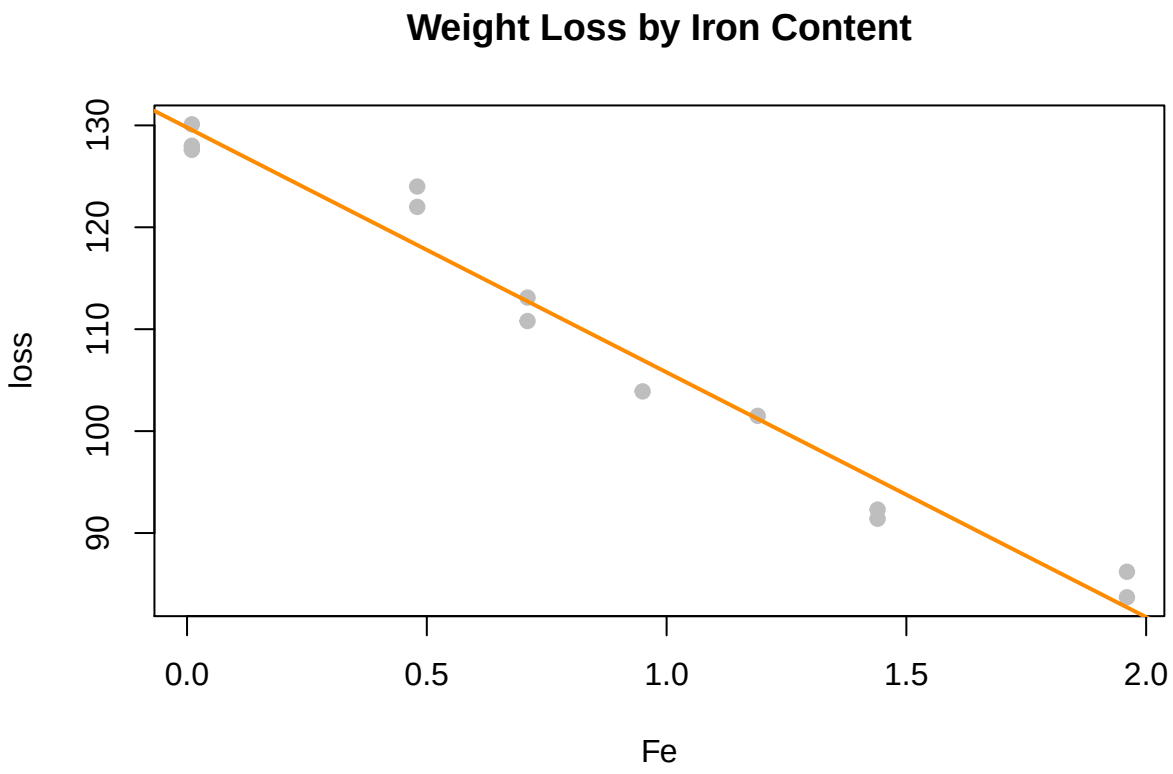
Exercise 4 (Corrosion Data)

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 scatter plot and add the fitted line. Check the assumptions of this model.

```
model_a = lm(loss ~ Fe, data = corrosion)

plot(loss ~ Fe, data = corrosion, col = "grey", pch = 20, cex = 1.5,
     main = "Weight Loss by Iron Content")
abline(model_a, col = "darkorange", lwd = 2)
```




```
bptest(model_a)
```

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

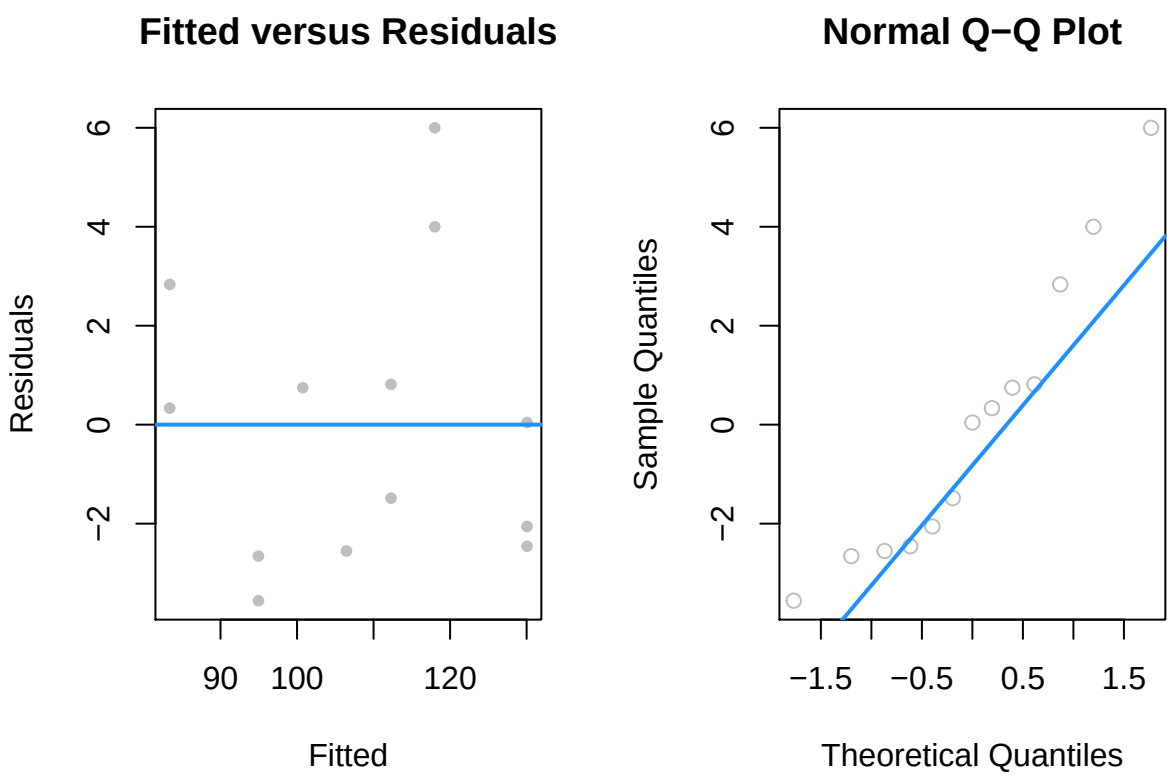
```
shapiro.test(resid(model_a))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(model_a)
## W = 0.93, p-value = 0.4
```

None of the assumptions in this model are suspect.

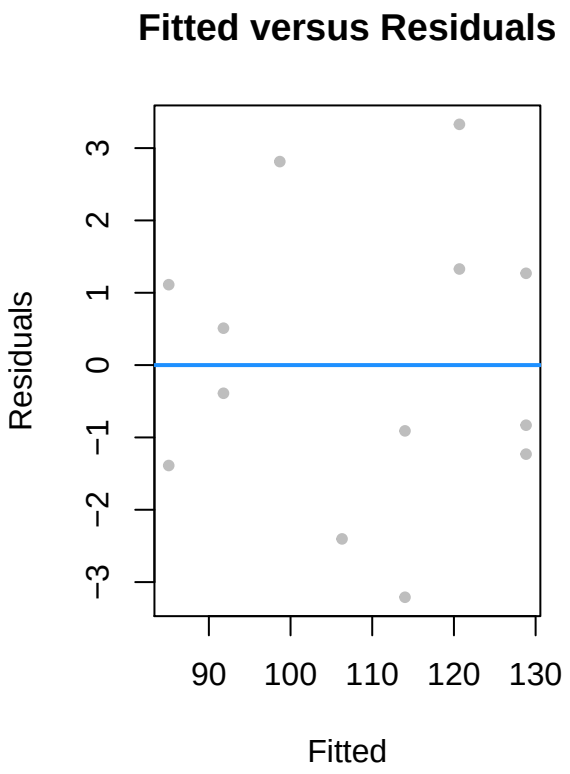
(b) Fit higher order polynomial models of degree 2, 3, and 4. For each, plot a fitted versus residuals plot and comment on the constant variance assumption. Based on those plots, which of these three models do you think are acceptable? Use a statistical test(s) to compare the models you just chose. Based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

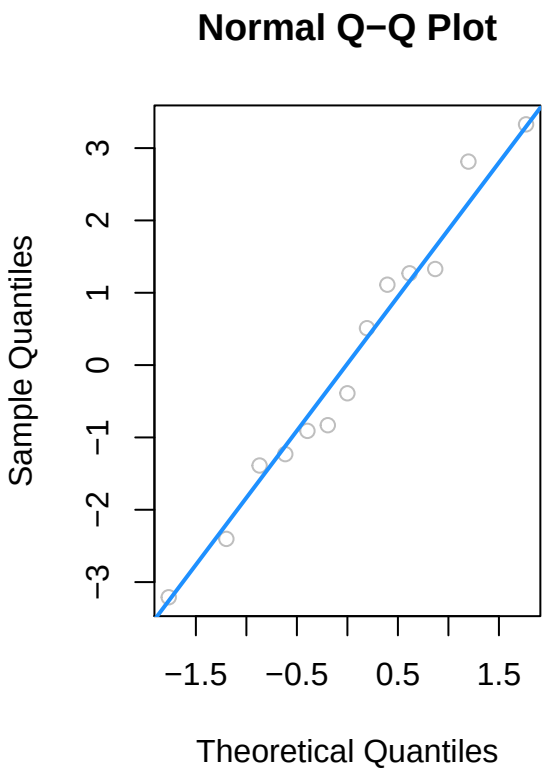
```r
model_b_2 = lm(loss ~ Fe + I(Fe^2), data = corrosion)
model_b_3 = lm(loss ~ Fe + I(Fe^2) + I(Fe^3), data = corrosion)
model_b_4 = lm(loss ~ Fe + I(Fe^2) + I(Fe^3) + I(Fe^4), data = corrosion)

diagnostics(model_b_2)
```




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

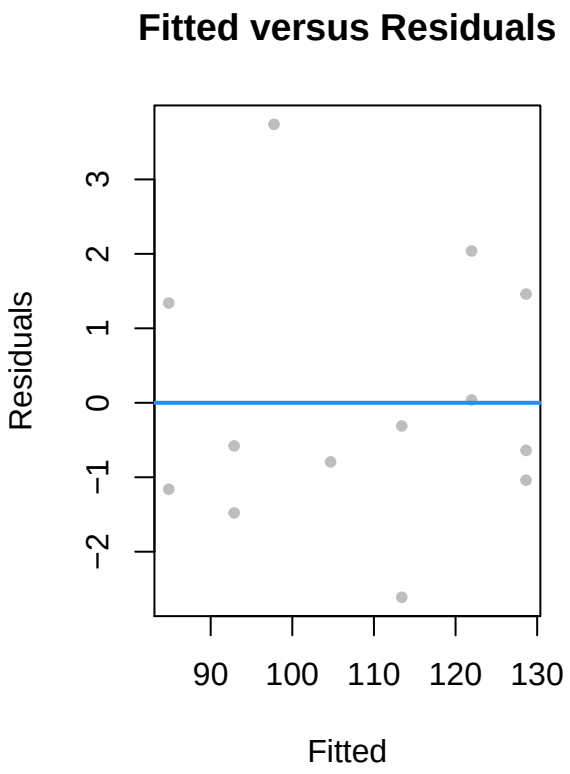
```r
diagnostics(model_b_3)
```

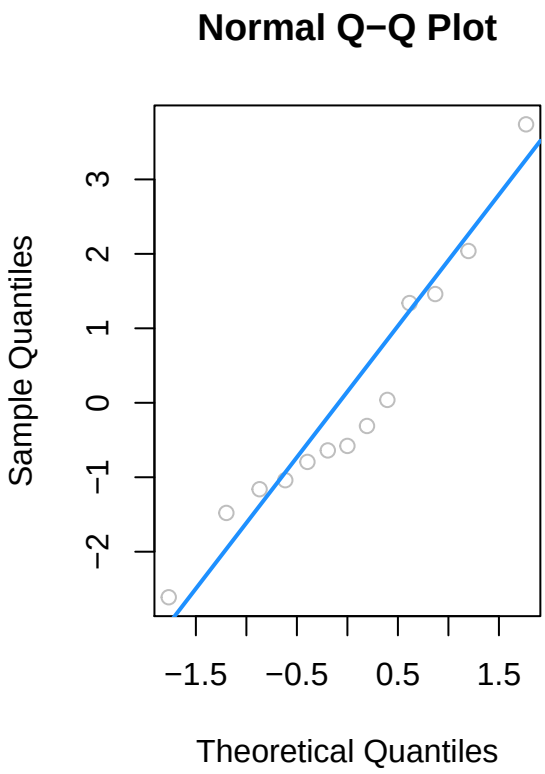





```
## $p_val
## [1] 0.9085
##
## $decision
## [1] "Fail to Reject"
```

```
diagnostics(model_b_4)
```






```
## $p_val
## [1] 0.4022
##
## $decision
## [1] "Fail to Reject"
```

The constant variance assumption looks the best in model 2, with degree up to 3. The model that has degree 2 also seems acceptable, but the fitted versus residuals plot does not look as good as the model with degree up to 3.

```
anova(model_b_2, model_b_3)
```

```
## Analysis of Variance Table
##
## Model 1: loss ~ Fe + I(Fe^2)
## Model 2: loss ~ Fe + I(Fe^2) + I(Fe^3)
##   Res.Df   RSS Df Sum of Sq  F Pr(>F)
## 1     10 100.1
## 2      9  45.1  1        55 11  0.009 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Based on the anova test above, I would choose the model with degree up to 3.

```
shapiro.test(resid(model_b_3))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(model_b_3)
## W = 0.97, p-value = 0.9
```

Based on the Shapiro-Wilk test above, it does not seem that the normality assumption of the model is suspect.

I'll define "influential" as having a Cook's Distance value greater than `4/n`, where `n` is the number of observations. These observations and their Cook Distances can be seen below.

```
cooks.distance(model_b_3)[cooks.distance(model_b_3) > 4 / length(fitted(model_b_3))]
```

```
## named numeric(0)
```

It seems as though there are no influential observations.

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.

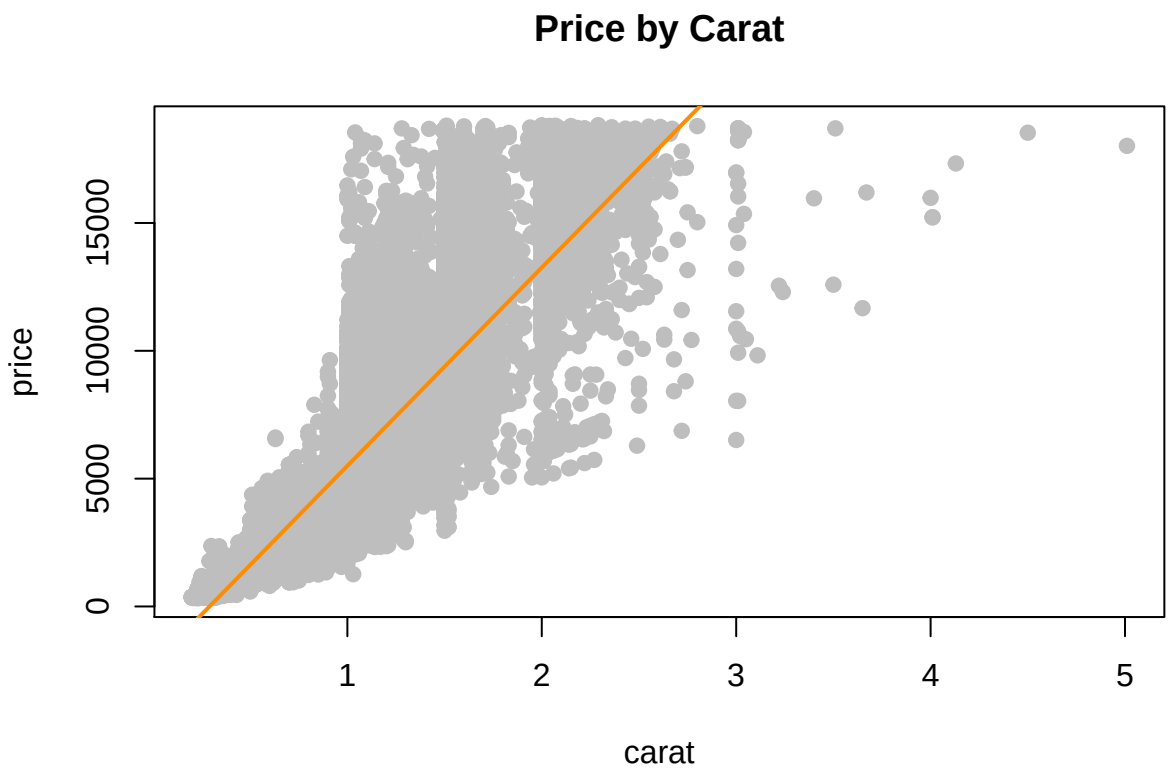
```
model_a = lm(price ~ carat, data = diamonds)
summary(model_a)
```

```
##
## 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
```

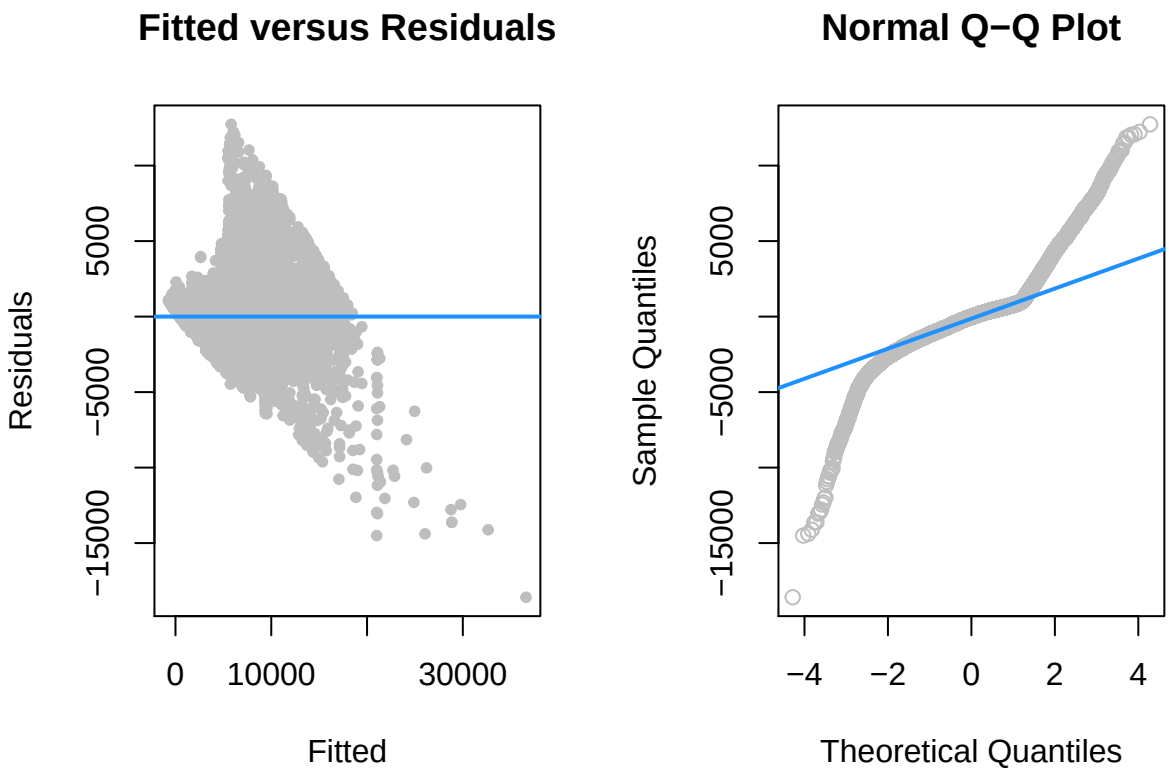
```
##
## Residual standard error: 1550 on 53938 degrees of freedom
## Multiple R-squared:  0.849,  Adjusted R-squared:  0.849
## F-statistic: 3.04e+05 on 1 and 53938 DF,  p-value: <2e-16
```

(b) Plot a scatter plot 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(price ~ carat, data = diamonds, col = "grey", pch = 20, cex = 1.5,
     main = "Price by Carat")
abline(model_a, col = "darkorange", lwd = 2)
```




```
diagnostics(model_a, testit = FALSE)
```

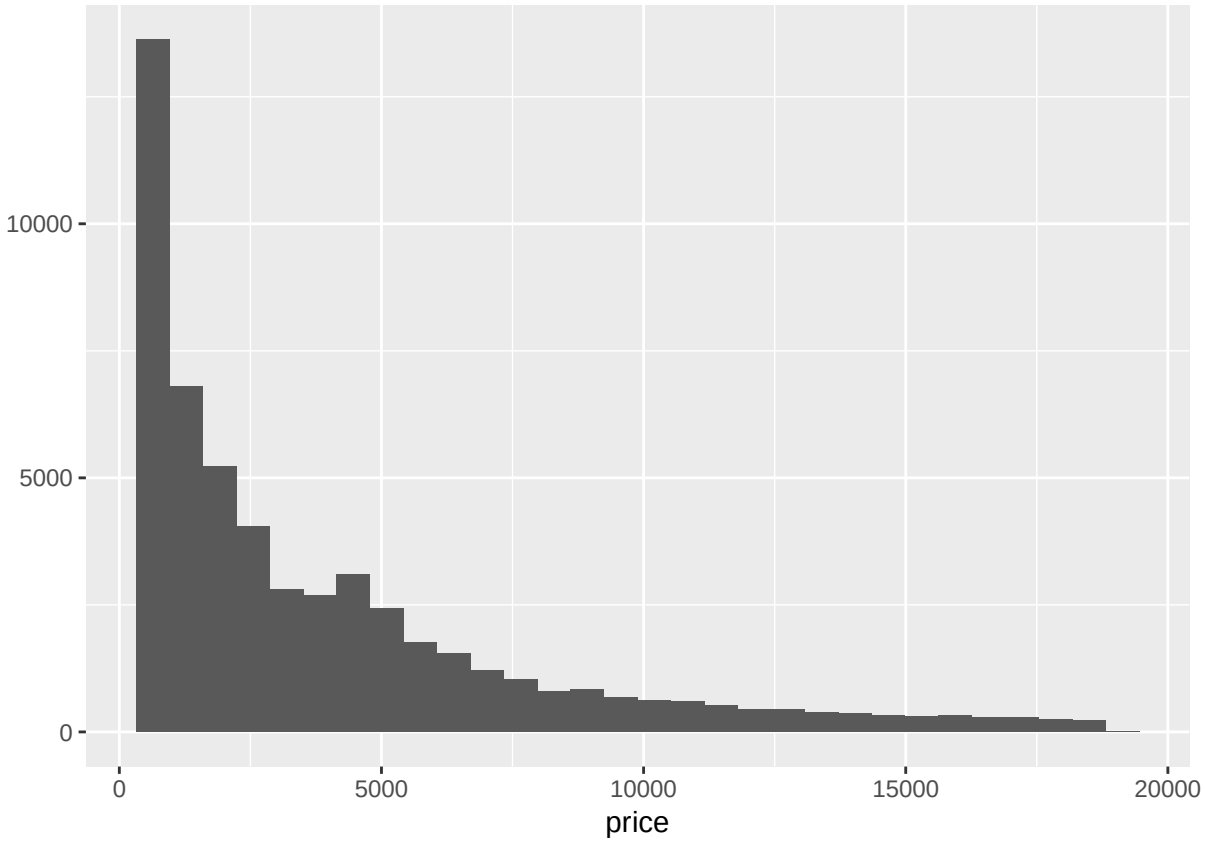



```
## list()
```

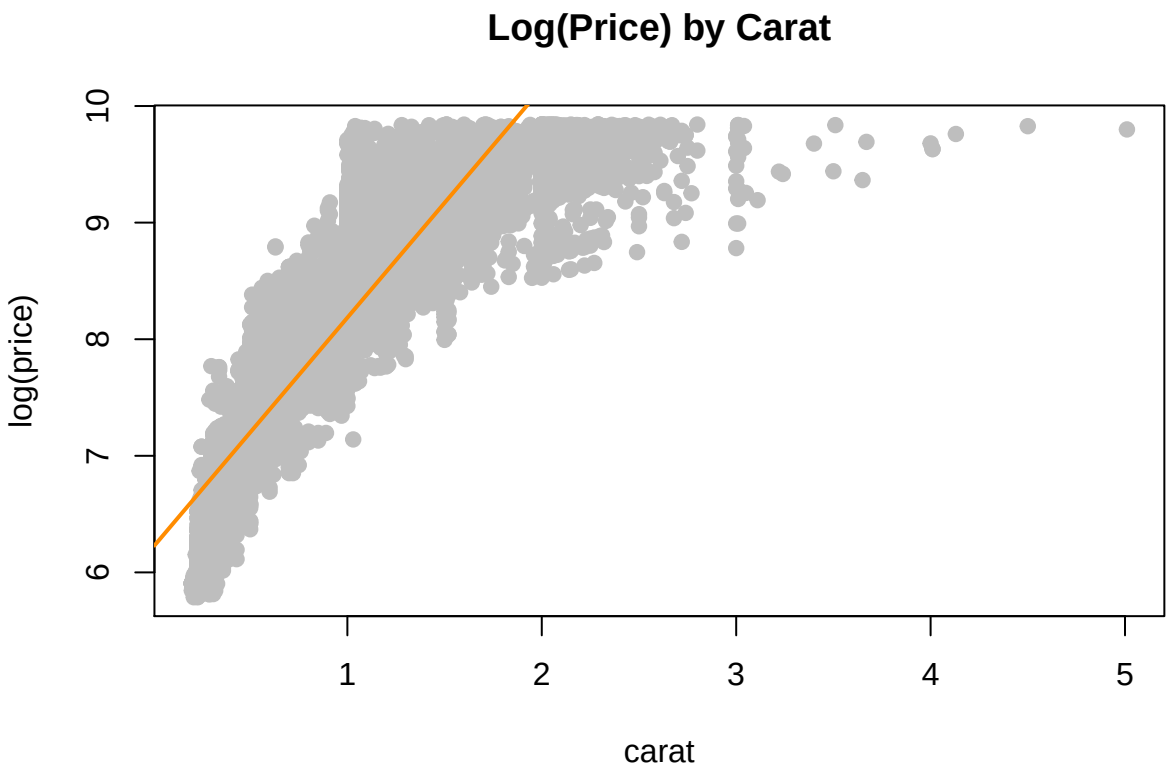
Looking at the two diagnostic plots above, it seems as though there is a non-constant variance. It also seems as though the normality assumption is suspect.

(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 scatter plot 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.

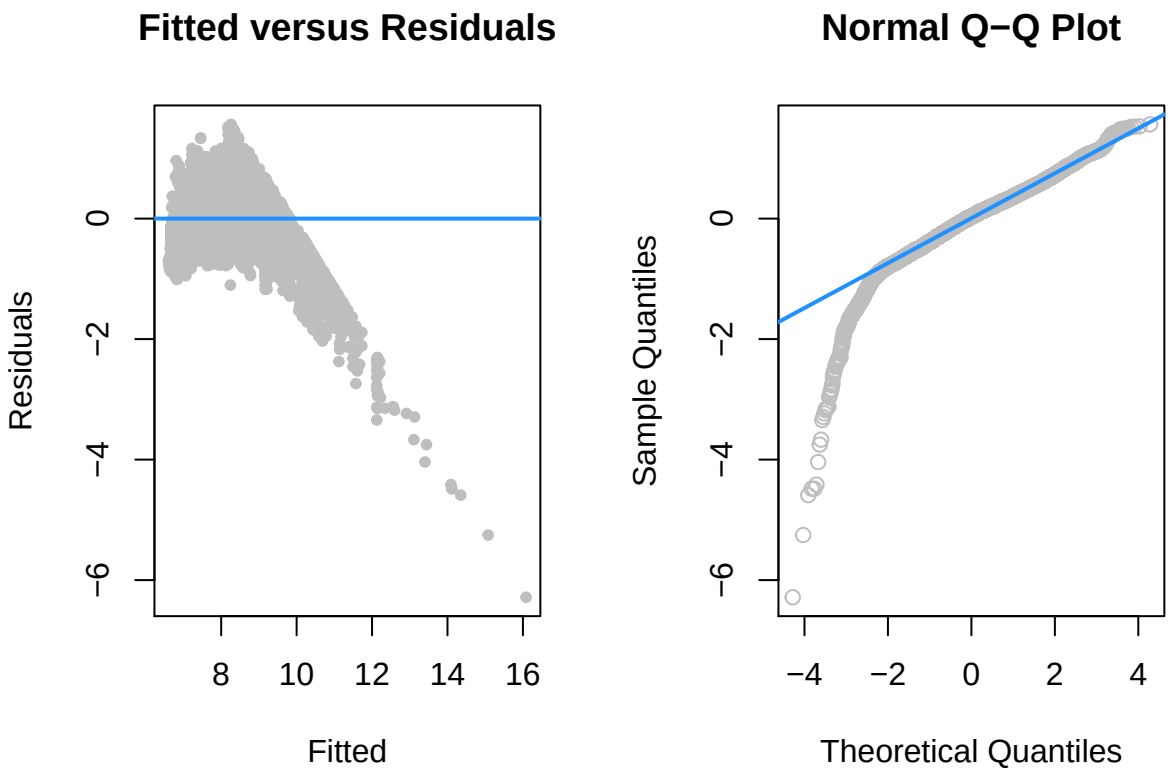
```
qplot(price, data = diamonds, bins = 30)
```




```
model_c = lm(log(price) ~ carat, data = diamonds)

plot(log(price) ~ carat, data = diamonds, col = "grey", pch = 20, cex = 1.5,
     main = "Log(Price) by Carat")
abline(model_c, col = "darkorange", lwd = 2)
```




```
diagnostics(model_c, testit = FALSE)
```

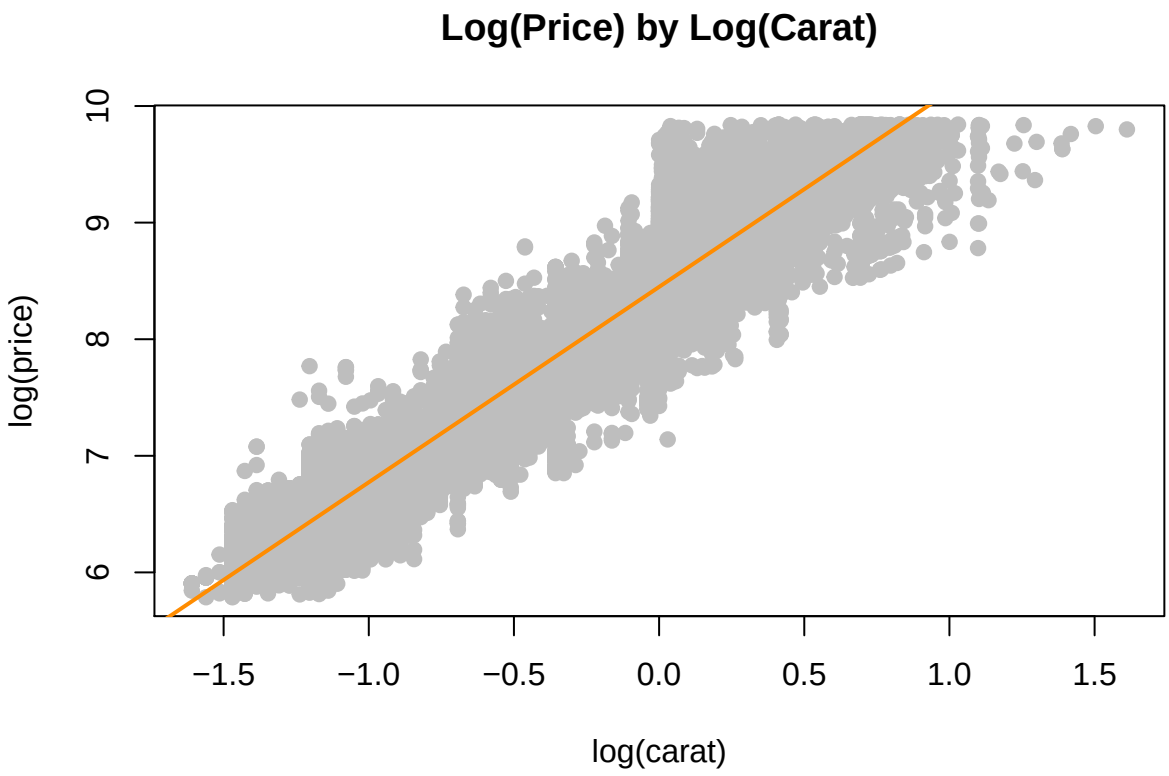



```
## list()
```

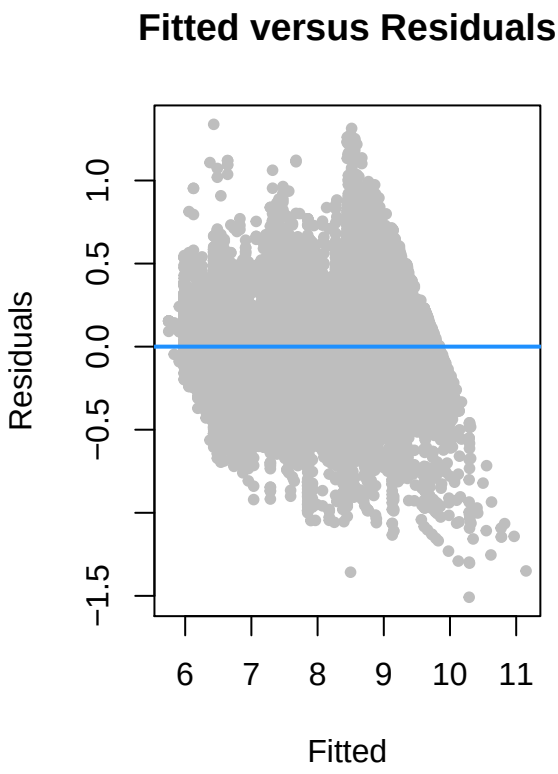
The diagnostics of the model with a log transformation of the response looks better than the previous model, but the assumptions still seem suspect.

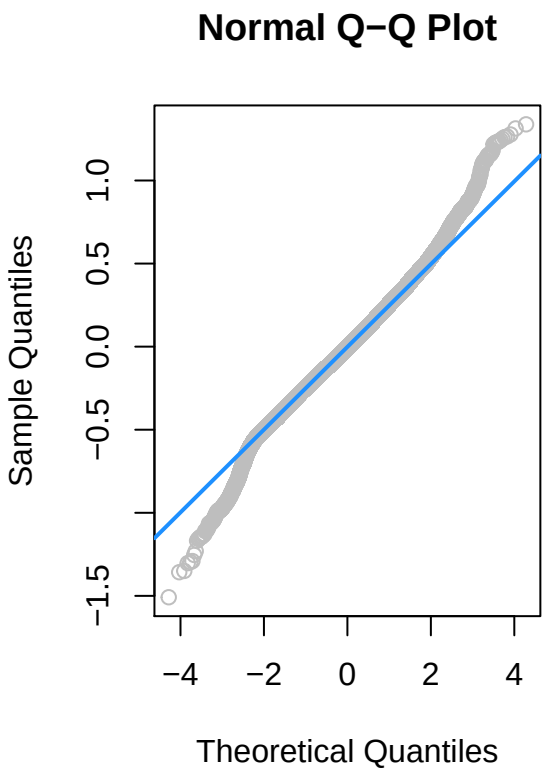
(d) Try adding log transformation of the predictor. Fit a model with a logged response and logged predictor, plot a scatter plot of log-price versus log-carat and add the line for the fitted model, then use a fitted versus residuals plot and/or a Q-Q plot to comment on the diagnostics of the model.

```
model_d = lm(log(price) ~ log(carat), data = diamonds)

plot(log(price) ~ log(carat), data = diamonds, col = "grey", pch = 20, cex = 1.5,
     main = "Log(Price) by Log(Carat)")
abline(model_d, col = "darkorange", lwd = 2)
```




```
diagnostics(model_d, testit = FALSE)
```






```
## list()
```

The diagnostics of this model no longer seem suspect. The normality assumption seems much more plausible, and it seems like there is constant variance in the residuals.

(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(model_d, interval = 'prediction', level = 0.99, newdata = data.frame(carat = log(3))))
```

```
##    fit  lwr   upr
## 1 5466 2779 10752
```