Week 8 - Homework

STAT 420, Summer 2018, Unger

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.

Solution:

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

  if (plotit == TRUE) {

    # side-by-side plots (one row, two columns)
    par(mfrow = c(1, 2))

    # fitted versus residuals
    plot(fitted(model), resid(model),
         col = pcol, pch = 20, cex = 1.5,
         xlab = "Fitted", ylab = "Residuals",
         main = "Fitted versus Residuals")
    abline(h = 0, col = lcol, lwd = 2)
    grid()

    # qq-plot
    qqnorm(resid(model), col = pcol, pch = 20, cex = 1.5)
    qqline(resid(model), col = lcol, lwd = 2)
    grid()
```

```
  }

  if (testit == TRUE) {
    # p-value and decision
    p_val = shapiro.test(resid(model))$p.value
    decision = ifelse(p_val < alpha, "Reject", "Fail to Reject")
    list(p_val = p_val, decision = decision)
  }

}
```

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

Solution:

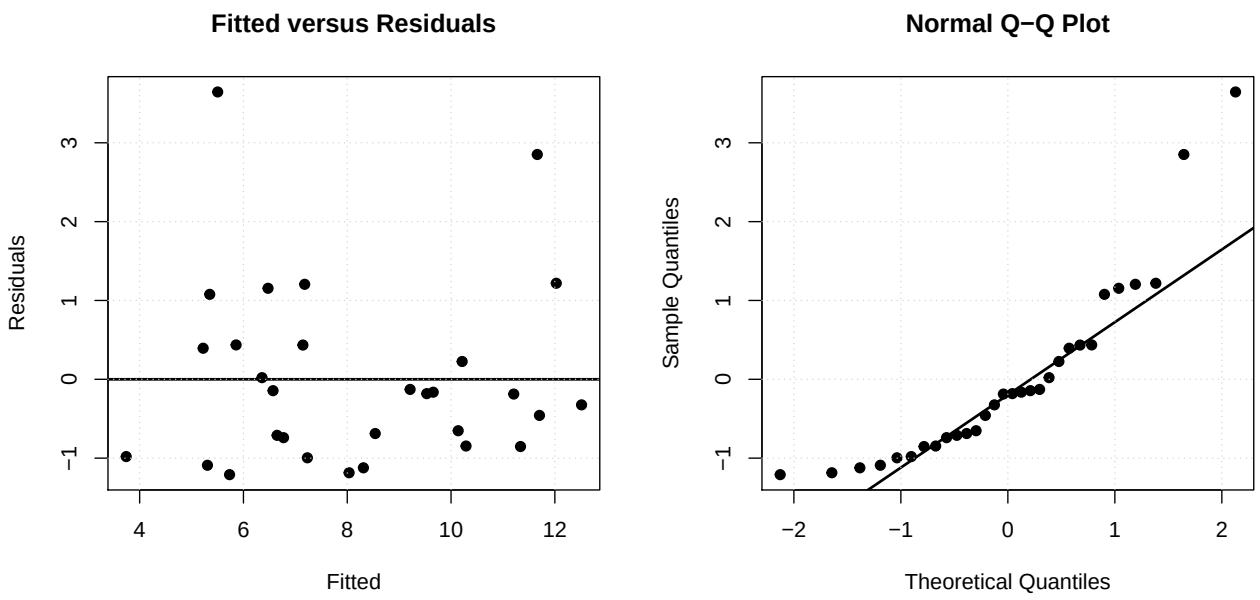
```
diagnostics(fit_1, plotit = FALSE)$p_val
```

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

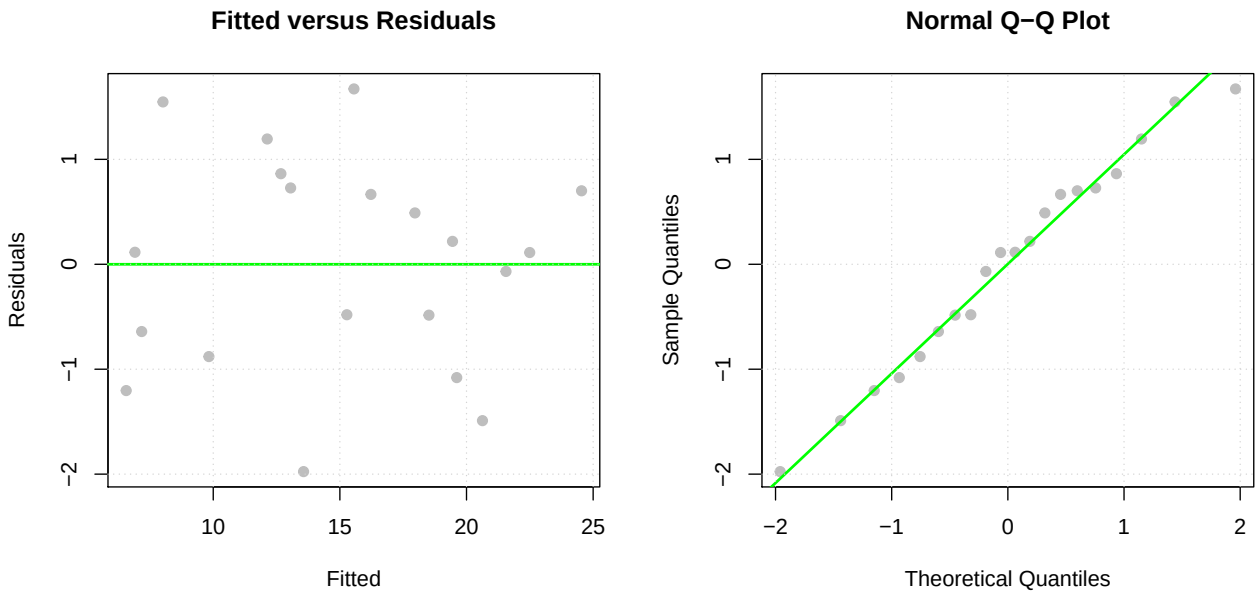
```
diagnostics(fit_2, plotit = FALSE)$decision
```

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

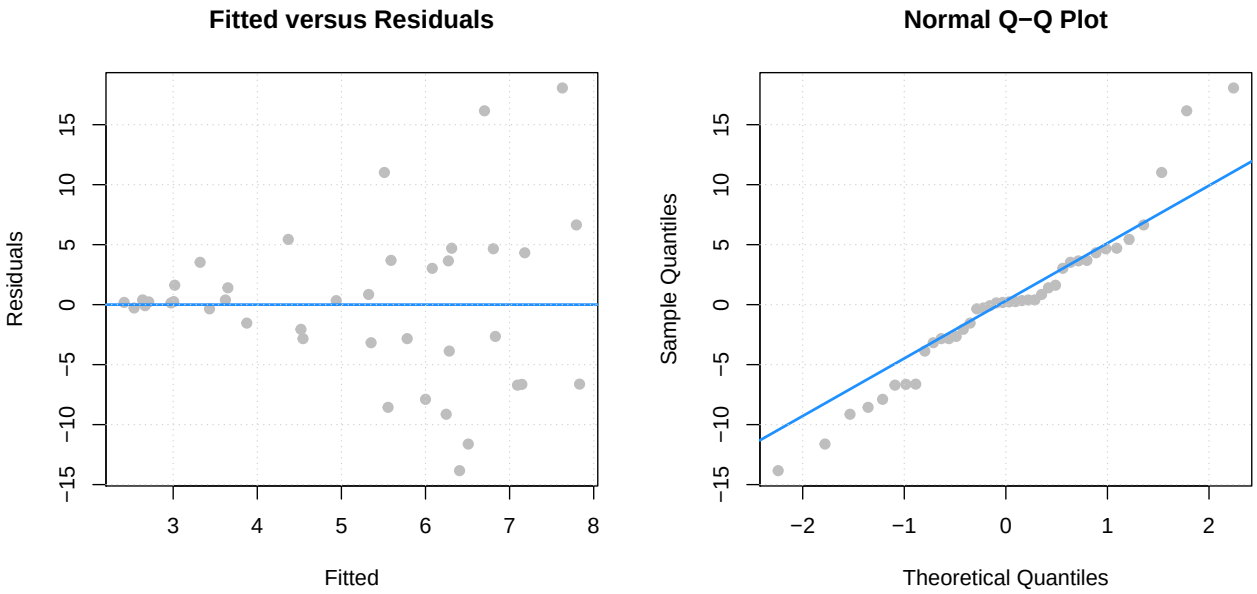
```
diagnostics(fit_1, testit = FALSE, pcol = "black", lcol = "black")
```




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




```
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.

Solution:

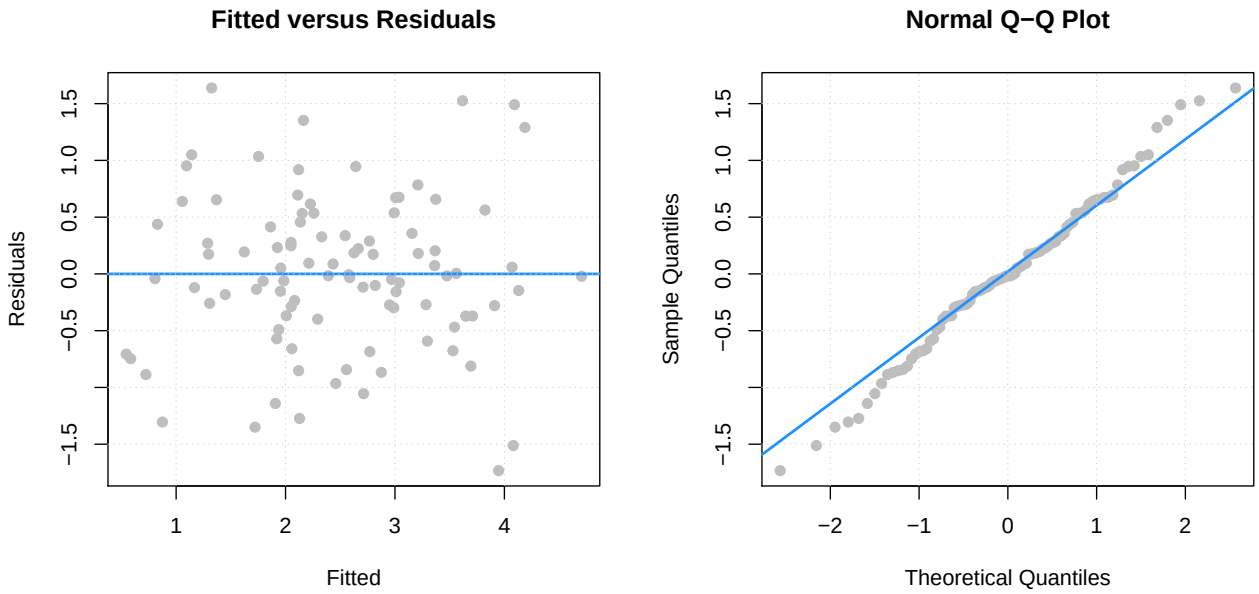
```
prostate_mod = lm(lpsa ~ ., data = prostate)
summary(prostate_mod)$r.squared
```

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

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

Solution:

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




```
library(lmtest)
bptest(prostate_mod)
```

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

The fitted versus residuals plot looks pretty good. We see the Breusch–Pagan test does not reject, so we choose to believe that the constant variance assumption has not been violated.

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

Solution:

```
diagnostics(prostate_mod, plotit = FALSE)
```

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

The points fall very close to the line in the Q-Q Plot output in **(b)**, and the Shapiro-Wilk test does not reject at any reasonable α, so we choose to believe that the normality assumption has not been violated.

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

Solution:

```
prostate_mod_lev = hatvalues(prostate_mod)
prostate_mod_lev_mean = mean(prostate_mod_lev)
prostate_mod_lev[prostate_mod_lev > 2 * prostate_mod_lev_mean]
```

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

Based on the heuristic given in the text, we say that these observations are points of high leverage. They have greater **potential** for a large influence on the model fit.

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

Solution:

```
prostate_mod_cook = cooks.distance(prostate_mod)
prostate_mod_cook[prostate_mod_cook > 4 / length(prostate_mod_cook)]
```

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

The above observations (with their Cook's distance reported) are considered influential according to the heuristic given in the text. Note that some of these observations had been previously identified as observations with large leverage, but not each of these observations.

(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.

Solution:

```
prostate_mod_sub = lm(lpsa ~ ., data = prostate,
                      subset = prostate_mod_cook <= 4 / length(prostate_mod_cook))
(coef(prostate_mod) - coef(prostate_mod_sub)) / coef(prostate_mod)
```

```
## (Intercept)      lcavol     lweight         age        lbph         svi
##     1.36764     0.03752    -0.19794     0.05469    -0.24497     0.02794
##         lcp     gleason       pgg45
##    -0.47353    -1.54143    -0.46954
```

Here, we calculate the relative change in the coefficients. We see that for some coefficients, such as `gleason`, the change can be rather large in magnitude. We note that this is done to illustrate the effect of influence. Removing an observation simply because it is influential should not be done in practice.

(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.

Solution:

```
prostate_removed = prostate[prostate_mod_cook > 4 / length(prostate_mod_cook), ]

(pred_all = predict(prostate_mod, prostate_removed))
```

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

```
(pred_sub = predict(prostate_mod_sub, prostate_removed))
```

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

```
(pred_all - pred_sub) / pred_all
```

```
##        32        39        47        69        95        96        97
## -0.080430  0.012407 -0.049663 -0.011689  0.080327 -0.007716  0.045618
```

Compared to the change in the estimated regression coefficients, the change in predicted `lpsa` is rather small. Why could this be? That's something we will touch on in an upcoming chapter!

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(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 = 19081014
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.)

Solution:

```
for (i in 1:num_sims) {
  y_1        = 4 + 1 * x_1 + 0 * x_2 + rnorm(n)
  fit_1      = lm(y_1 ~ x_1 + x_2)
  p_val_1[i] = summary(fit_1)$coef["x_2", "Pr(>|t|)"]

  y_2        = 4 + 1 * x_1 + 0 * x_2 + rnorm(n, 0, abs(x_2))
  fit_2      = lm(y_2 ~ x_1 + x_2)
  p_val_2[i] = summary(fit_2)$coef["x_2", "Pr(>|t|)"]
}
```

(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.

Solution:

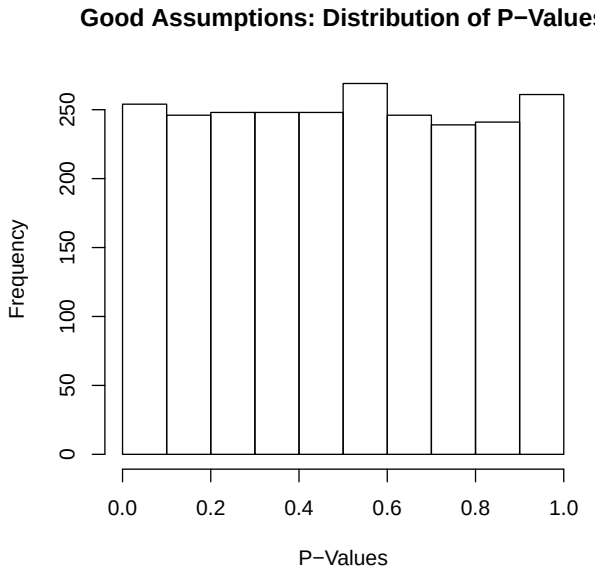
```
results = data.frame(
  alpha = c(0.01, 0.05, 0.10),
  good  = c(mean(p_val_1 < 0.01), mean(p_val_1 < 0.05), mean(p_val_1 < 0.10)),
  bad   = c(mean(p_val_2 < 0.01), mean(p_val_2 < 0.05), mean(p_val_2 < 0.10))
)
colnames(results) = c("alpha", "good assumptions", "assumptions violated")
knitr::kable(results)
```

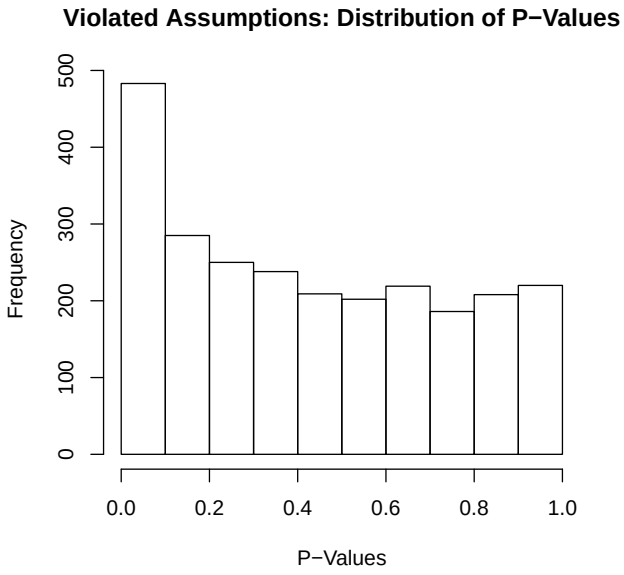
alpha	good assumptions	assumptions violated
0.01	0.0096	0.0416
0.05	0.0528	0.1192
0.10	0.1016	0.1932

The results of `p_val_1` are roughly what we would expect. Because $\beta_2 = 0$ is true in this simulation, we expect α of these simulations to reject by chance. We see these values are very far off for `p_val_2`, which is a result of the violation of assumptions. This is why we should only perform inference when assumptions are not violated.

Below we see the simulated distribution of the p-values under the two situations. When assumptions are met, the p-values follow the uniform distribution that we expect.

```
par(mfrow = c(1, 2))
hist(p_val_1, main = "Good Assumptions: Distribution of P-Values", xlab = "P-Values")
hist(p_val_2, main = "Violated Assumptions: Distribution of P-Values", xlab = "P-Values")
```





Exercise 4 (Corrosion Data)

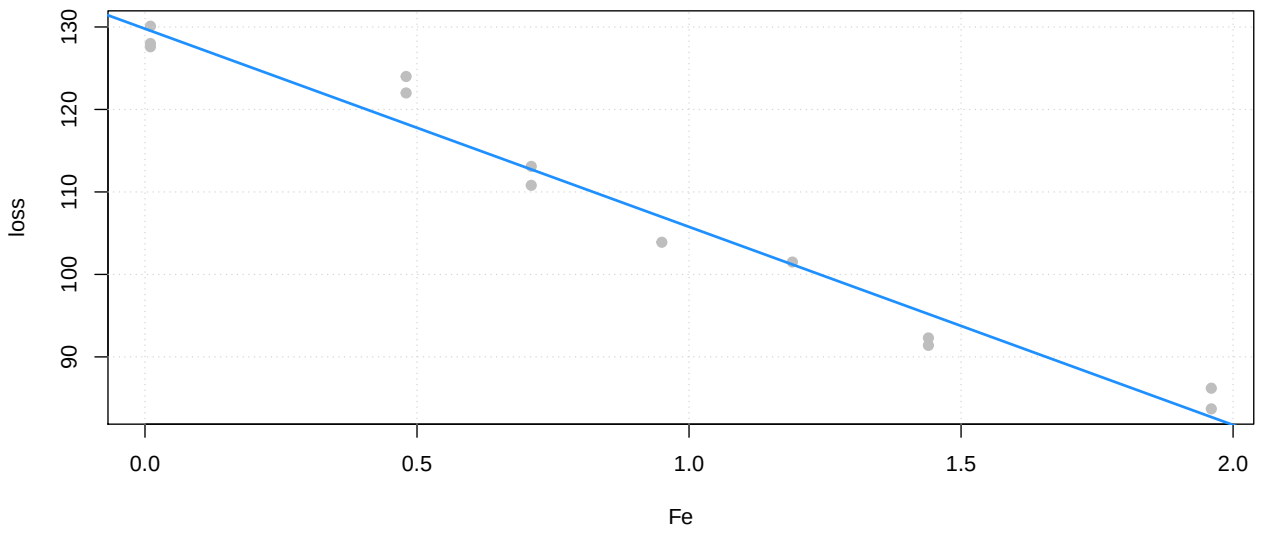
For this exercise, we will use the `corrosion` data, which can be found in the `faraway` package. After loading the `faraway` package, use `?corrosion` to learn about this dataset.

```
library(faraway)
```

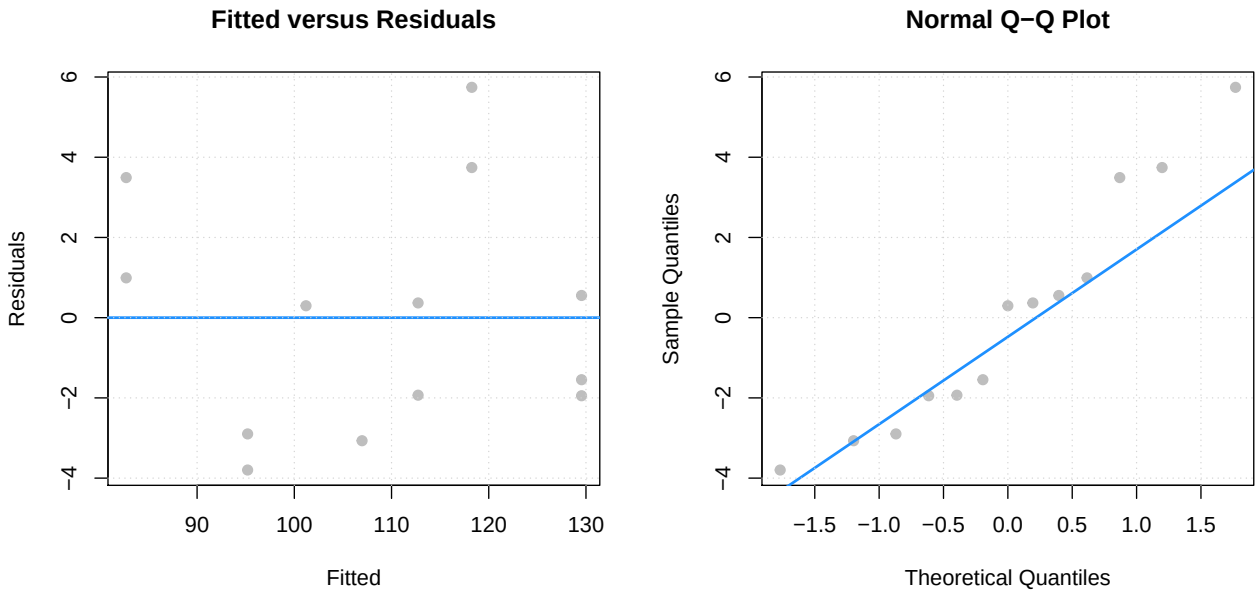
(a) Fit a simple linear regression with `loss` as the response and `Fe` as the predictor. Plot a scatterplot and add the fitted line. Check the assumptions of this model.

Solution:

```
corr_mod_1 = lm(loss ~ Fe, data = corrosion)
plot(loss ~ Fe, data = corrosion, col = "grey", pch = 20, cex = 1.5)
grid()
abline(corr_mod_1, col = "dodgerblue", lwd = 2)
```



```
diagnostics(corr_mod_1, testit = FALSE)
```

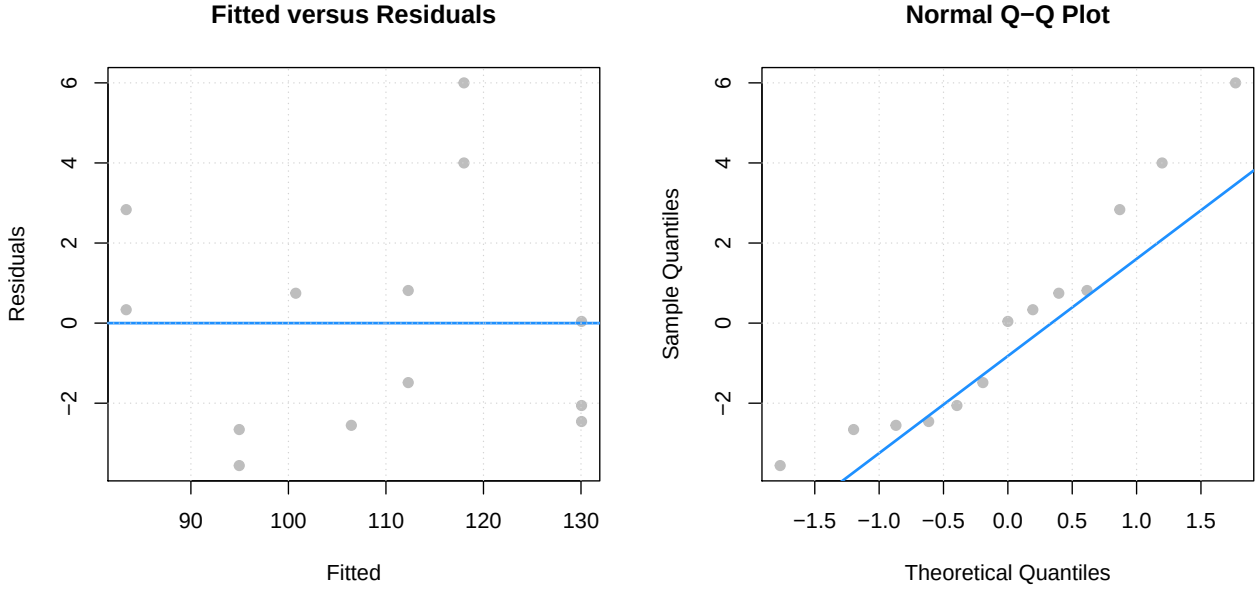


The Q-Q plot looks okay, not great. (Although with so little data, it is very hard to decide.) The fitted versus residuals plot, however, seem to have a pattern. Something weird is happening there.

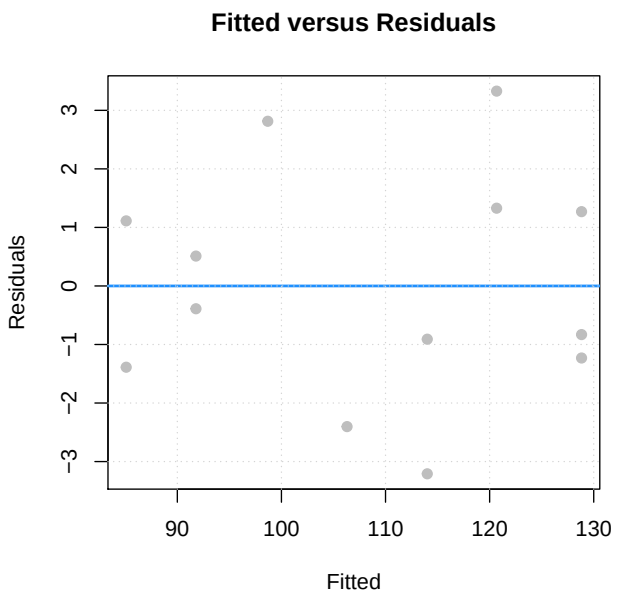
(b) Fit higher order polynomial models of degree 2, 3, and 4. For each, plot a fitted versus residuals plot and comment on the constant variance assumption. Based on those plots, which of these three models do you think are acceptable? Use a statistical test(s) to compare the models you just chose. Based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

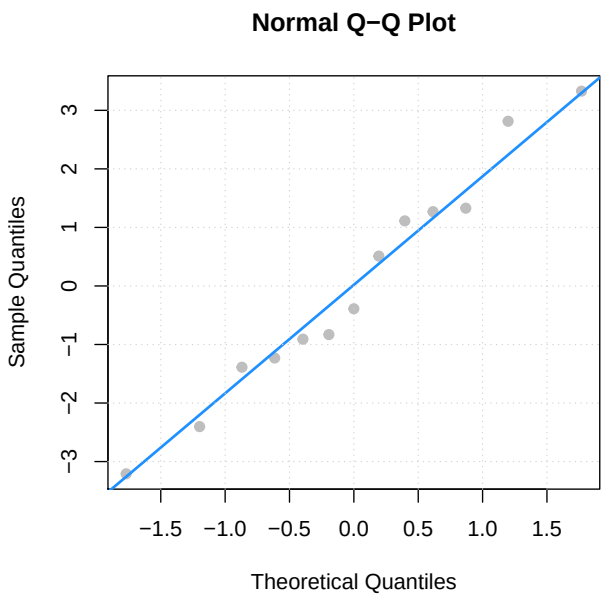
Solution:

```
corr_mod_2 = lm(loss ~ poly(Fe, 2), data = corrosion)
diagnostics(corr_mod_2, testit = FALSE)
```

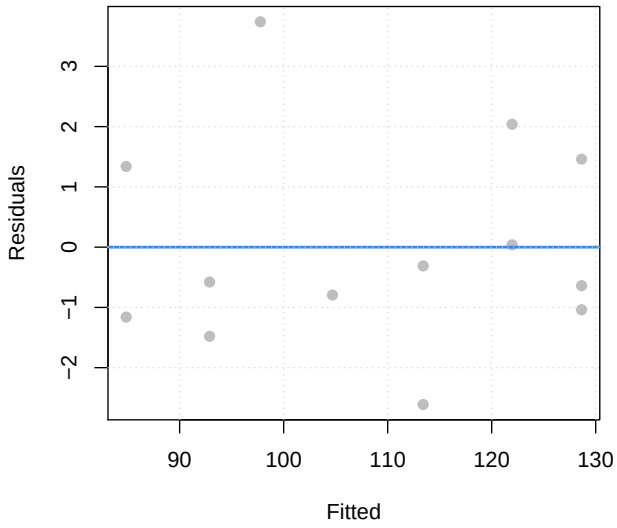


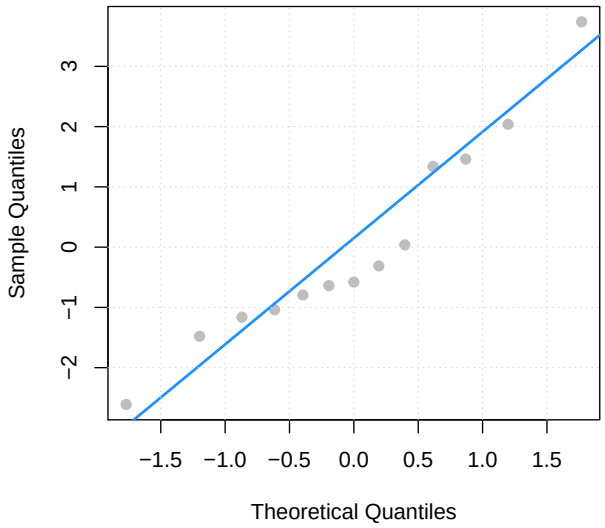

```
corr_mod_3 = lm(loss ~ poly(Fe, 3), data = corrosion)
diagnostics(corr_mod_3, testit = FALSE)
```






```
corr_mod_4 = lm(loss ~ poly(Fe, 4), data = corrosion)
diagnostics(corr_mod_4, testit = FALSE)
```






```
anova(corr_mod_3, corr_mod_4)
```

```
## Analysis of Variance Table
##
## Model 1: loss ~ poly(Fe, 3)
## Model 2: loss ~ poly(Fe, 4)
##   Res.Df  RSS Df Sum of Sq   F Pr(>F)
## 1      9 45.1
## 2      8 35.0  1      10.1 2.3   0.17
```

```
shapiro.test(resid(corr_mod_3))
```

```
##
```

```
##  Shapiro-Wilk normality test
##
## data:  resid(corr_mod_3)
## W = 0.97, p-value = 0.9
```

```
cook_corr_mod_3 = cooks.distance(corr_mod_3)
cook_corr_mod_3[which(cook_corr_mod_3 > 4 / length(cook_corr_mod_3))]
```

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

Based on the plots, the models of degree 3 and 4 are acceptable. Based on the test, we prefer the model of degree 3. The Q-Q plot and the Shapiro-Wilk test suggest that there is no issue with the normality assumption. No observations are identified as influential.

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.

Solution:

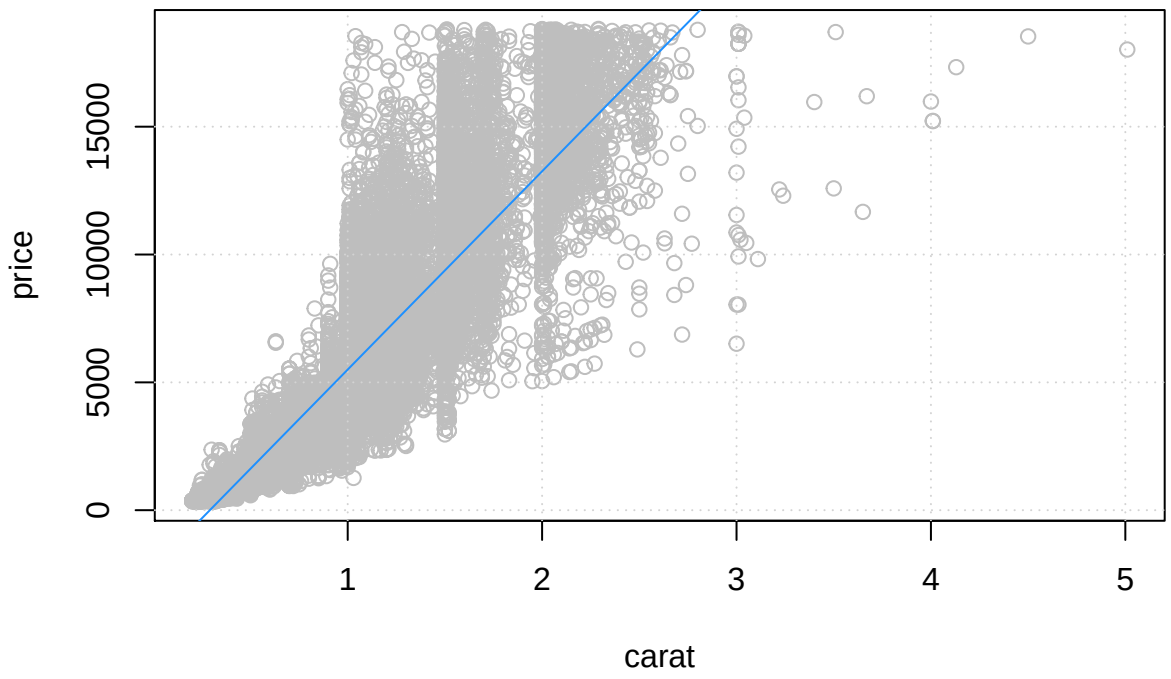
```
diamond_mod = lm(price ~ carat, data = diamonds)
summary(diamond_mod)
```

```
##
## 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
```

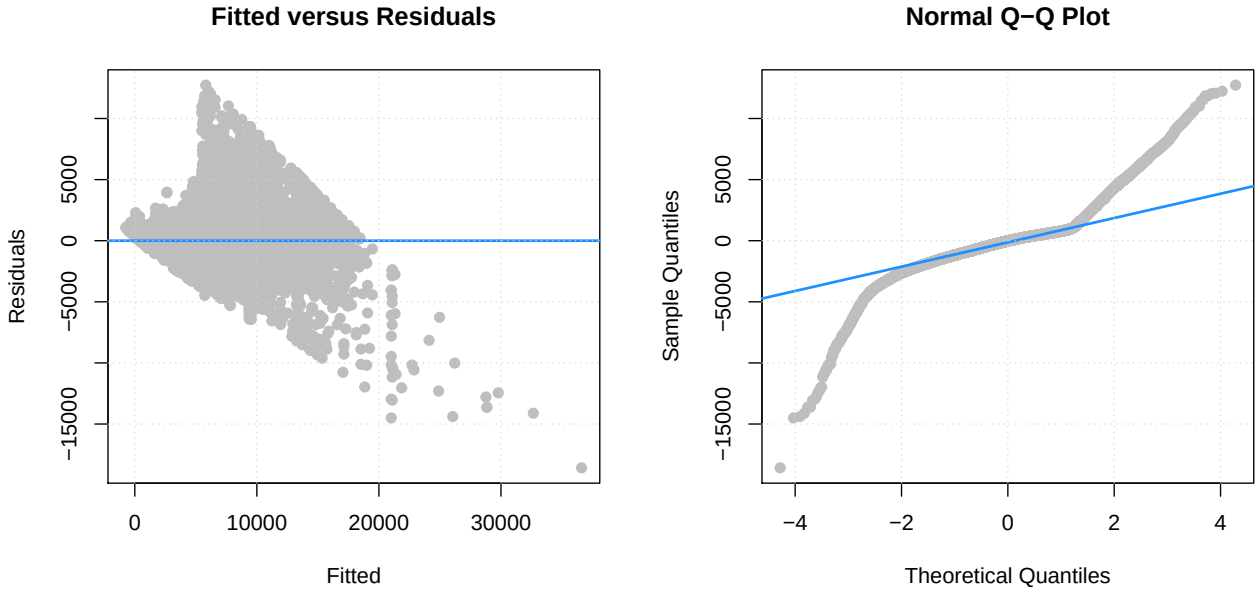
(b) Plot a scatterplot of price versus carat and add the line for the fitted model in part **(a)**. Using a fitted versus residuals plot and/or a Q-Q plot, comment on the diagnostics.

Solution:

```
plot(price ~ carat, data = diamonds, col = "grey")
grid()
abline(diamond_mod, col = "dodgerblue")
```



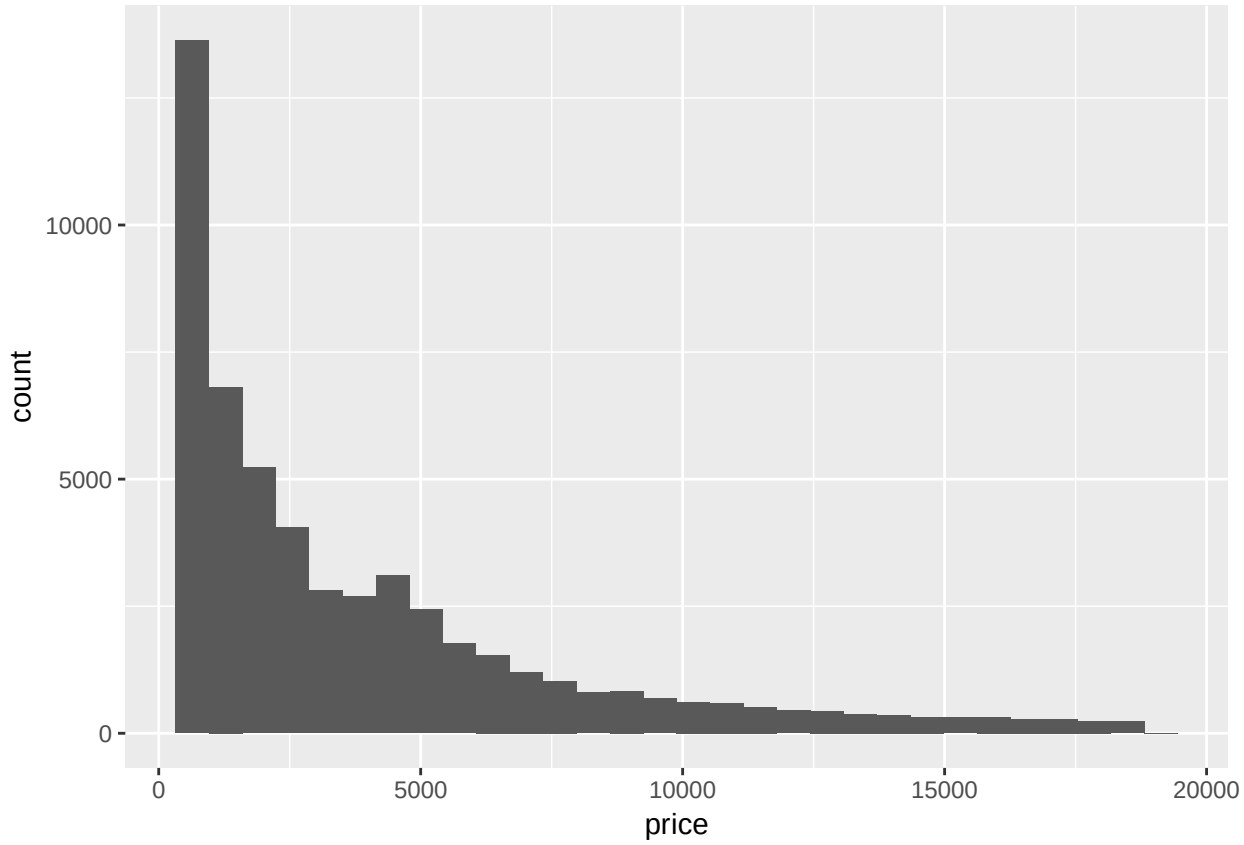

```
diamond_mod = lm(price ~ carat, data = diamonds)
diagnostics(diamond_mod, testit = FALSE)
```



Both plots look simply terrible.

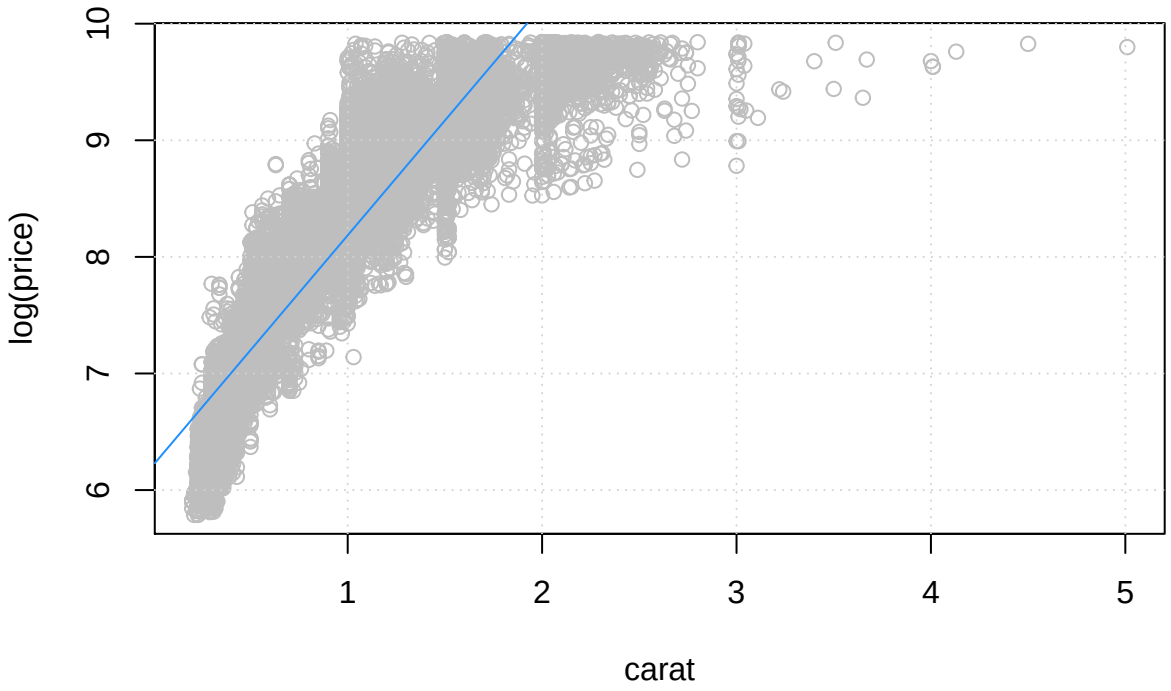
(c) Seeing as the price stretches over several orders of magnitude, it seems reasonable to try a log transformation of the response. Fit a model with a logged response, plot a scatterplot of log-price versus carat and add the line for the fitted model, then use a fitted versus residuals plot and/or a Q-Q plot to comment on the diagnostics of the model.

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

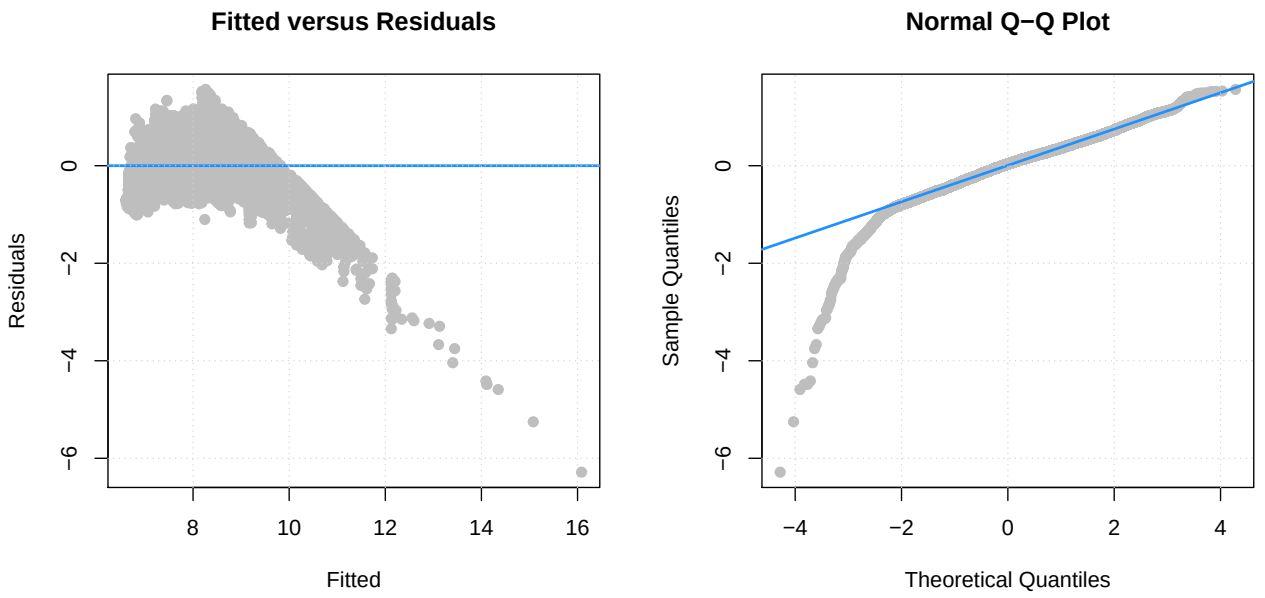


Solution:

```
diamond_mod_log = lm(log(price) ~ carat, data = diamonds)
plot(log(price) ~ carat, data = diamonds, col = "grey")
grid()
abline(diamond_mod_log, col = "dodgerblue")
```




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

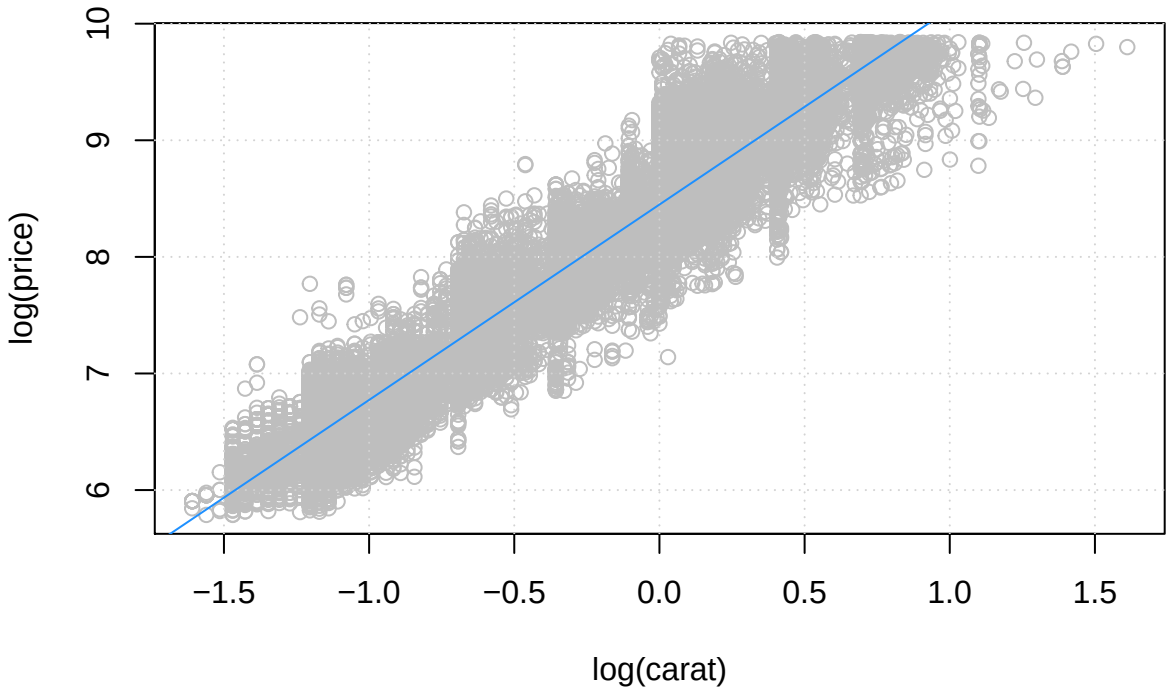


These plots are different, but still not good.

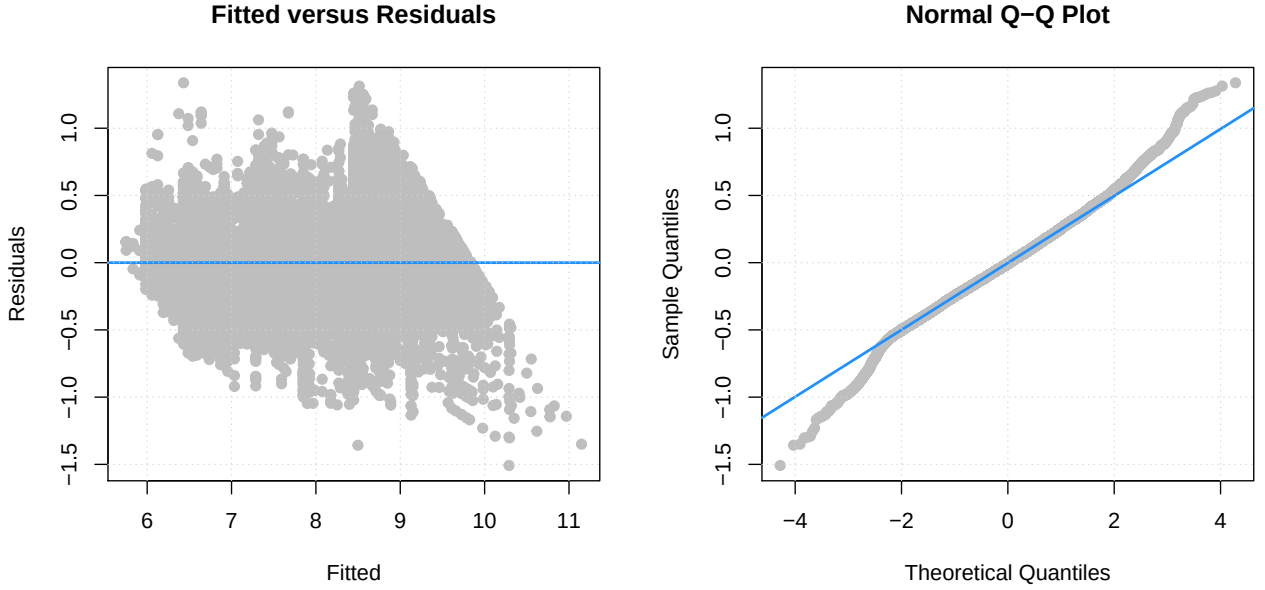
(d) Try adding log transformation of the predictor. Fit a model with a logged response and logged predictor, plot a scatterplot of log-price versus log-carat and add the line for the fitted model, then use a fitted versus residuals plot and/or a Q-Q plot to comment on the diagnostics of the model.

Solution:

```
diamond_mod_log_log = lm(log(price) ~ log(carat), data = diamonds)
plot(log(price) ~ log(carat), data = diamonds, col = "grey")
grid()
abline(diamond_mod_log_log, col = "dodgerblue")
```




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



These plots are much better. Not perfect, but much better.

(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).

Solution:

```
new_diamond = data.frame(carat = 3)
exp(predict.lm(diamond_mod_log_log, new_diamond, interval = c("prediction"), level = 0.99))
```

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

Note, we need to change from a log scale back to the original scale.