Week 10 - Homework

STAT 420, Fall 2024, Banghao Chi

11/03/2024

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) {
  resids = residuals(model)
  fitted_vals = fitted(model)

  if(testit) {
    sw_test = shapiro.test(resids)
    output = list(
      p_val = sw_test$p.value,
      decision = ifelse(sw_test$p.value < alpha, "Reject", "Fail to Reject")
    )
  }

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

```
    plot(fitted_vals, resids,
         col = pcol,
         xlab = "Fitted",
         ylab = "Residuals",
         main = "Residuals vs Fitted Values")
    abline(h = 0, col = lcol, lwd = 2)

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

  if(testit) {
    return(output)
  }
}
```

(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)
```

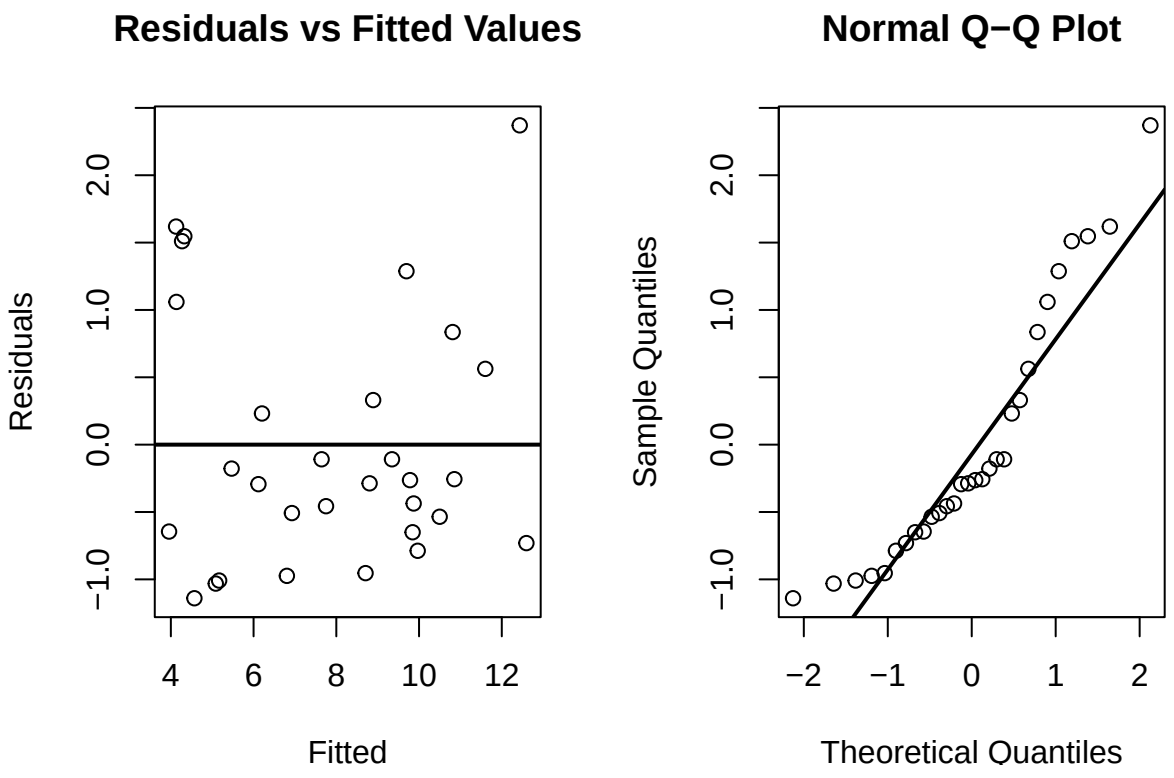
```
diagnostics(fit_1, plotit = FALSE)$p_val
```

```
## [1] 0.005613
```

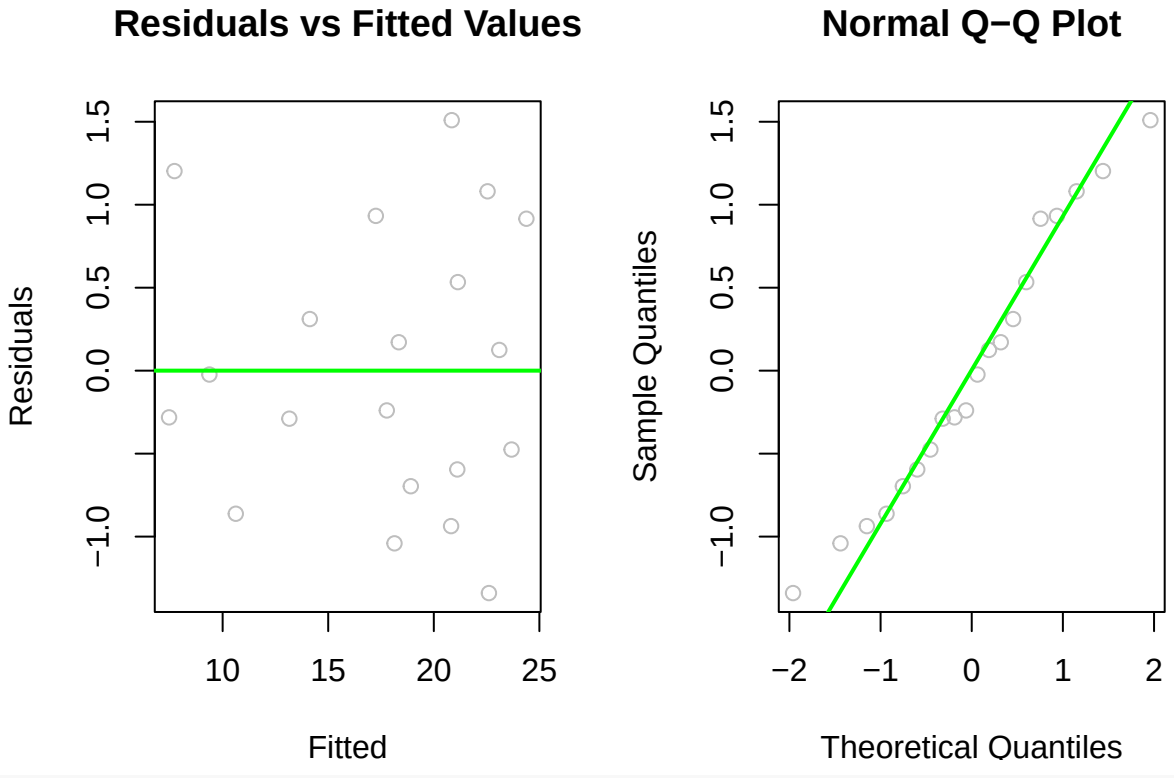
```
diagnostics(fit_2, plotit = FALSE)$decision
```

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

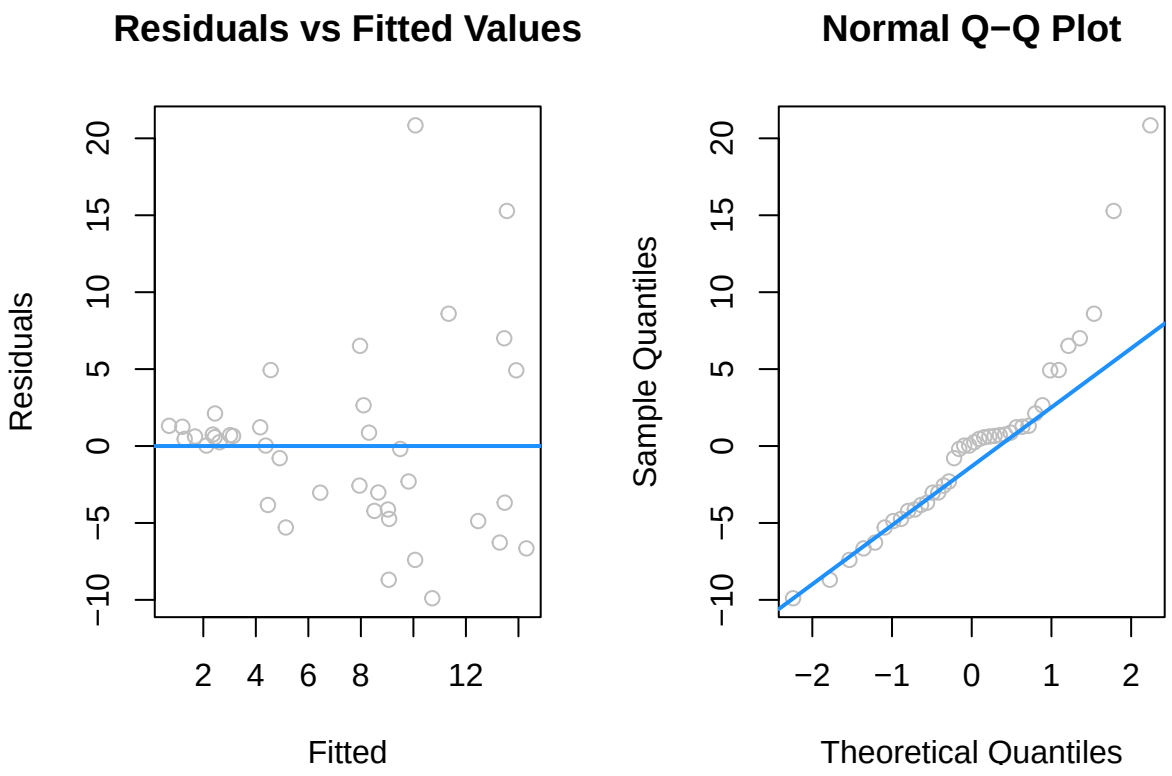
```
diagnostics(fit_1, testit = FALSE, pcol = "black", lcol = "black")
```




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




```
diagnostics(fit_3)
```




```
## $p_val
## [1] 0.001608
##
## $decision
## [1] "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)
```

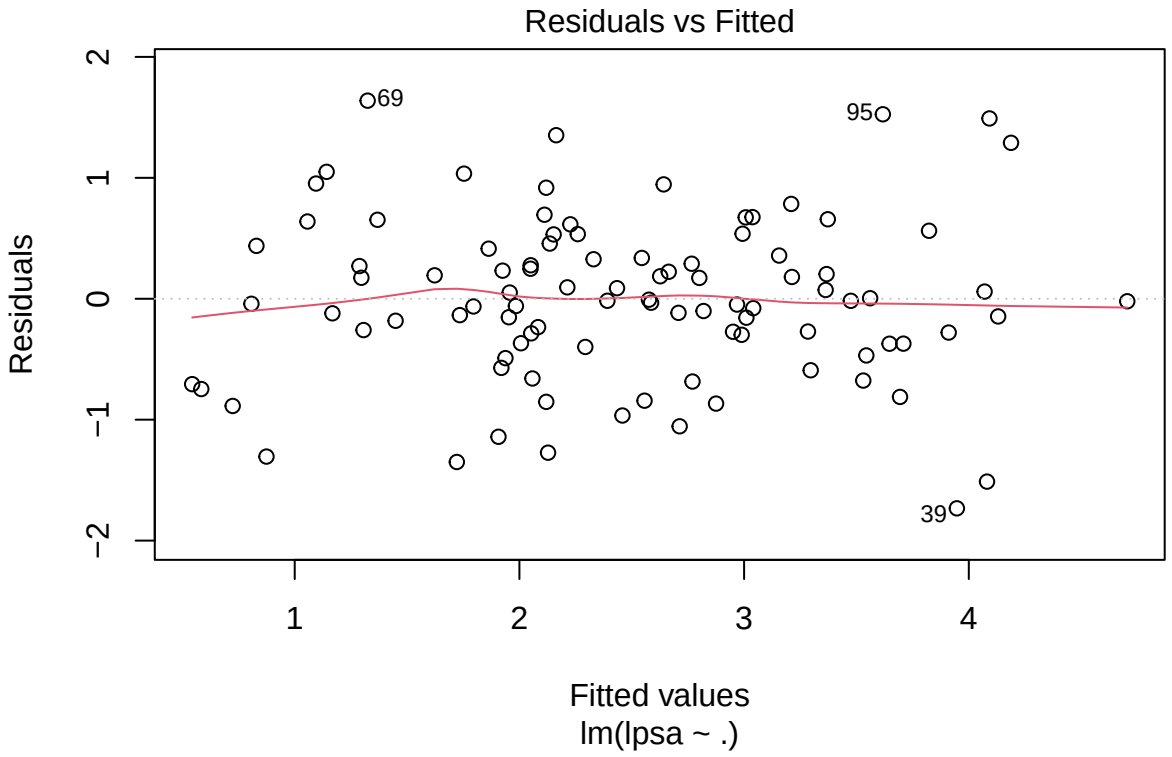
(a) Fit an additive multiple regression model with `lpsa` as the response and the remaining variables in the `prostate` dataset as predictors. Report the R^2 value for this model.

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

The R^2 value for this model is 0.6548.

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

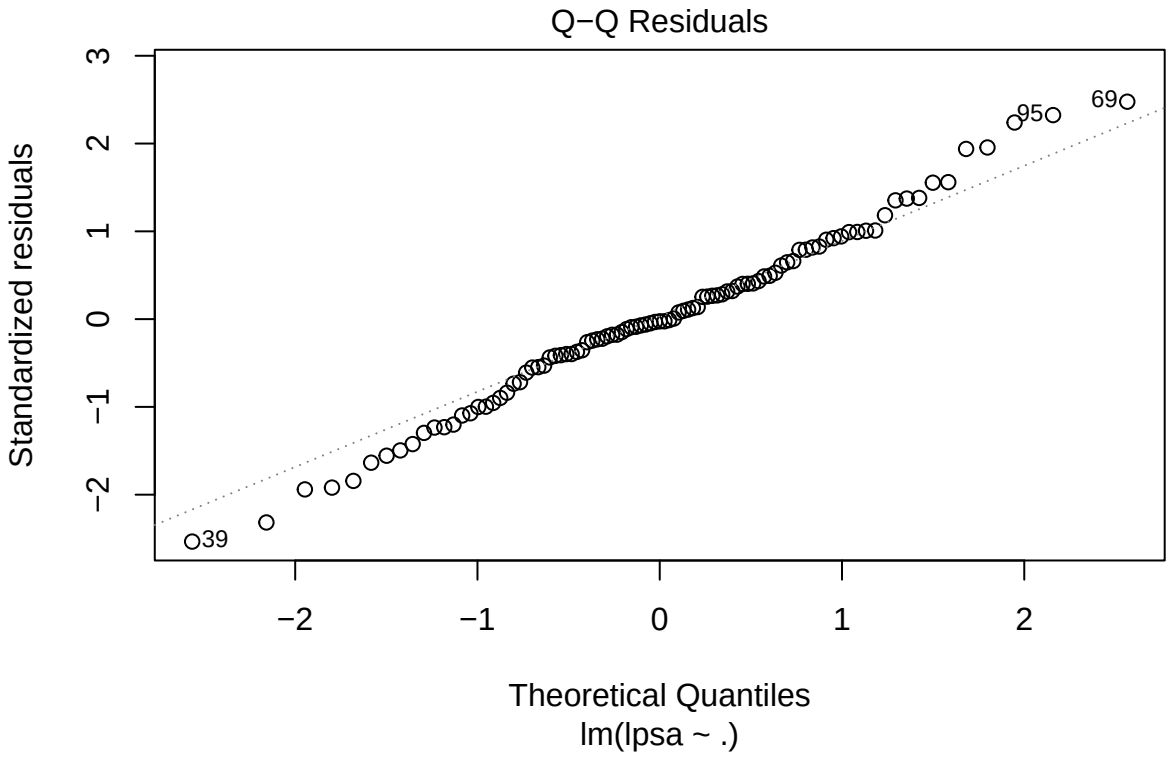
```
plot(model, which = 1)
```



Look at the Fitted vs Residuals plot, I don't think the constant variance assumption for this model is violated, as for any fitted value, the distribution of the residuals seem to be similar (in terms of range and density).

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

```
plot(model, which = 2)
```



Look at the Normal Q-Q plot, I think the normality assumption for this model is violated, there seems to be fat tails within the plot. But if we check the p-value of the SW test, we can see a large value, where we won't reject the null hypothesis, meaning that the normality assumption is not violated.

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

```
leverage = hatvalues(model)
high_leverage_cutoff = 2 * mean(leverage)
high_leverage = which(leverage > high_leverage_cutoff)
prostate[high_leverage, ]
```

```
##    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
```

The above is the reported observations that have high leverage.

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

```
cd = cooks.distance(model)
n = length(cd)
influential_cutoff = 4/n
influential_points = which(cd > influential_cutoff)
prostate[influential_points, ]
```

```
##     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
```

The above is the reported observations that are influential.

(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_cleaned = lm(lpsa ~ ., data = prostate[-influential_points,])

coef_comparison = data.frame(
    Original = coef(model),
    Without_Influential = coef(model_cleaned),
    Difference = coef(model_cleaned) - coef(model)
)
round(coef_comparison, 4)
```

```
##             Original Without_Influential Difference
## (Intercept)   0.6693             -0.2461    -0.9154
## lcavol        0.5870              0.5650    -0.0220
## lweight       0.4545              0.5444     0.0900
## age          -0.0196             -0.0186     0.0011
## lbph          0.1071              0.1333     0.0262
## svi           0.7662              0.7447    -0.0214
## lcp          -0.1055             -0.1554    -0.0499
## gleason       0.0451              0.1147     0.0696
## pgg45         0.0045              0.0066     0.0021
```

(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.

```
removed_data = prostate[influential_points,]
pred_original = predict(model, newdata = removed_data)
pred_cleaned = predict(model_cleaned, newdata = removed_data)

prediction_comparison = data.frame(
    Observation = influential_points,
    Original_Pred = pred_original,
    Clean_Model_Pred = pred_cleaned,
    Difference = pred_cleaned - pred_original
)
round(prediction_comparison, 4)
```

```
##    Observation Original_Pred Clean_Model_Pred Difference
## 32          32         2.875            3.107     0.2313
## 39          39         3.947            3.898    -0.0490
## 47          47         4.081            4.284     0.2027
## 69          69         1.325            1.340     0.0155
## 95          95         3.618            3.327    -0.2906
## 96          96         4.188            4.220     0.0323
## 97          97         4.092            3.905    -0.1867
```

When we remove influential points and refit the model, our predictions for those influential points change noticeably. This change in predictions demonstrates how these points were indeed influential - removing them affected the model's parameters enough to create different predictions.

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)
```

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

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

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

```
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 = 20021023
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) {
  x_1 = runif(n, 0, 5)
  x_2 = runif(n, -2, 2)

  y_1 = 4 + 1 * x_1 + 0 * x_2 + rnorm(n = n, mean = 0, sd = 1)
  y_2 = 4 + 1 * x_1 + 0 * x_2 + rnorm(n = n, mean = 0, sd = abs(x_2))

  fit_1 = lm(y_1 ~ x_1 + 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.

```
prop_01_1 = mean(p_val_1 < 0.01)
prop_01_2 = mean(p_val_2 < 0.01)

prop_05_1 = mean(p_val_1 < 0.05)
prop_05_2 = mean(p_val_2 < 0.05)

prop_10_1 = mean(p_val_1 < 0.10)
prop_10_2 = mean(p_val_2 < 0.10)

results = data.frame(
  Alpha = c(0.01, 0.05, 0.10),
  Model1 = c(prop_01_1, prop_05_1, prop_10_1),
  Model2 = c(prop_01_2, prop_05_2, prop_10_2)
)
results
```

```
##   Alpha Model1 Model2
## 1  0.01 0.0092 0.0584
## 2  0.05 0.0504 0.1568
```

```
## 3  0.10 0.1056 0.2284
```

The proportion that we see in the above table is rates of the probability of rejecting $H_0 : \beta_2 = 0$ when it is actually true. For model 1, the rates should be close to the alpha level (which is exactly the case as we see in above table), since it's variance of the noise is indeed a constant; for model 2, since the noise is not a constant, the rates should slightly deviate from the value of alpha due to the violation of the homoscedasticity assumption.

Exercise 4 (Corrosion Data)

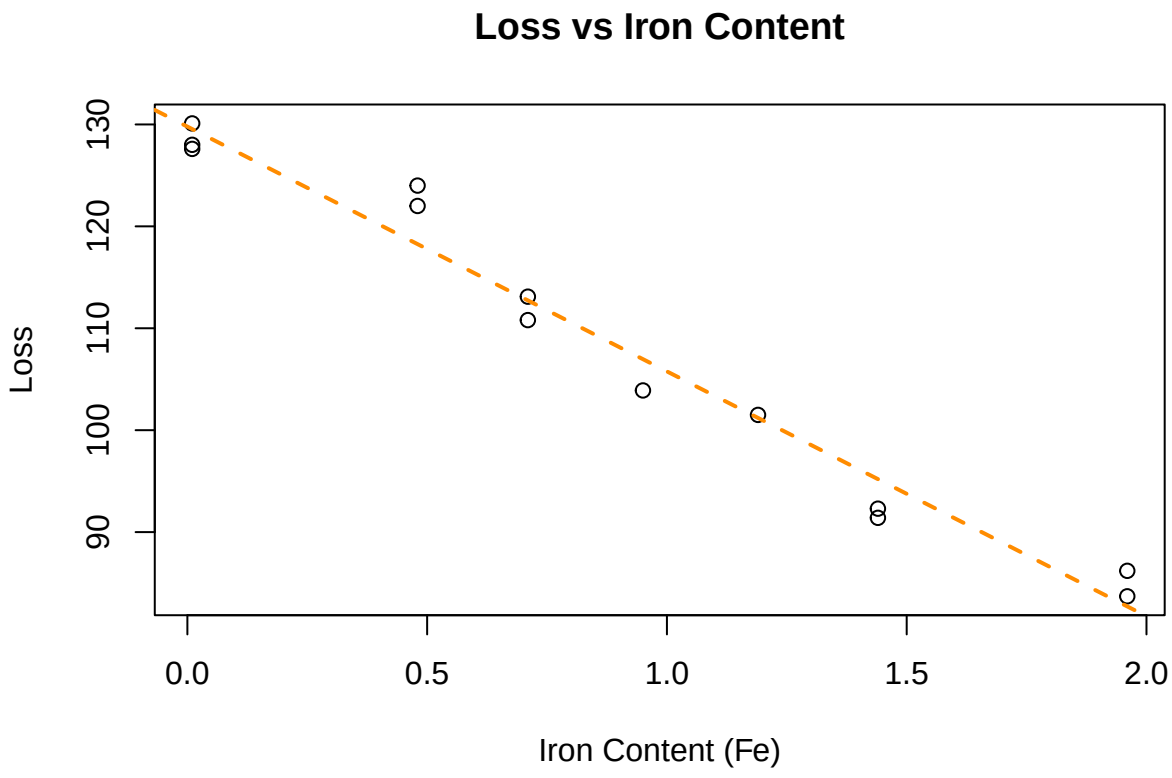
For this exercise, we will use the `corrosion` data, which can be found in the `faraway` package. After loading the `faraway` package, use `?corrosion` to learn about this dataset.

```
library(faraway)
```

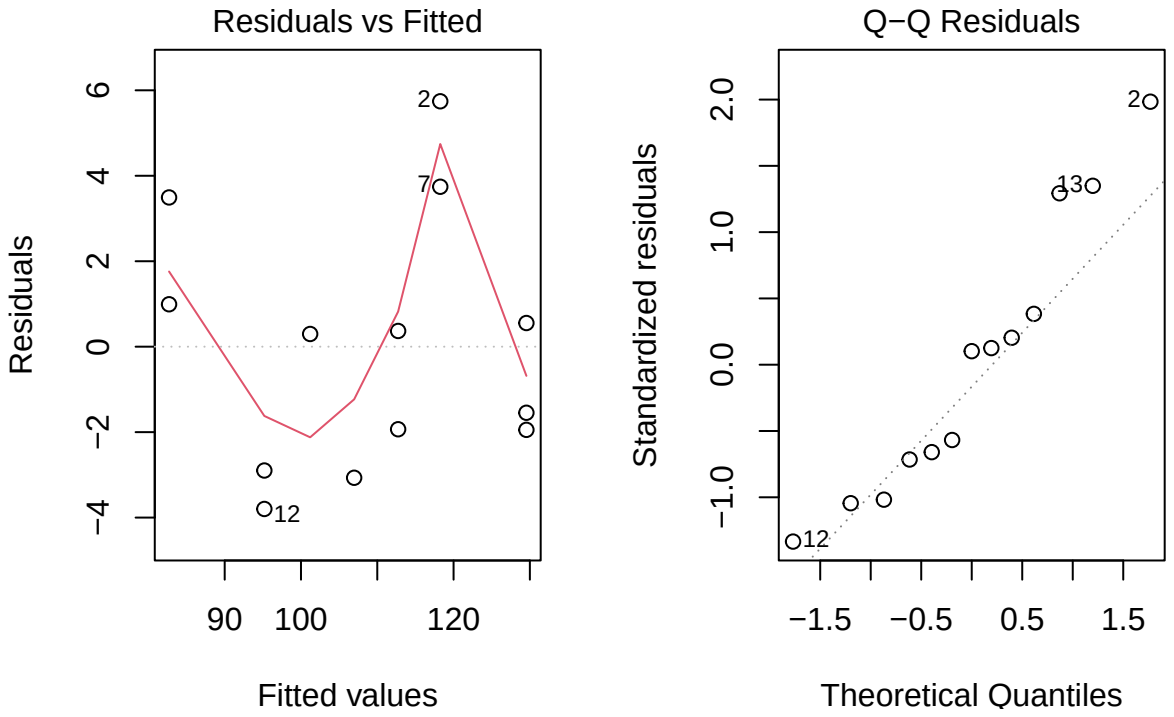
(a) Fit a simple linear regression with `loss` as the response and `Fe` as the predictor. Plot a scatterplot and add the fitted line. Check the assumptions of this model.

```
model1 = lm(loss ~ Fe, data=corrosion)

plot(loss ~ Fe, data=corrosion,
     main="Loss vs Iron Content",
     xlab="Iron Content (Fe)",
     ylab="Loss")
abline(model1, col="darkorange", lwd = 2, lty = 2)
```




```
par(mfrow=c(1,2))
plot(model1, which=1)
plot(model1, which=2)
```




```
bptest(model1)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  model1
## BP = 0.025, df = 1, p-value = 0.9
```

```
shapiro.test(resid(model1))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(model1)
## W = 0.93, p-value = 0.4
```

We check the assumption using statistical method and also **art** (visual) method.

- **Statistically**:
  - For BP test, since the p-value of the test is 0.9, which is greater than most of the alpha value, we fail to reject the null hypothesis. This suggests that the assumption of constant variance is satisfied.
  - For SW test, since the p-value of the test is 0.4, which is greater than most of the alpha value, we fail to reject the null hypothesis. This suggests that the assumption of residuals are normally distributed.
- **Visually**:
  - For Fitted vs Residuals plot, we see that the distribution of the residuals vary according to the value of the fitted values, which suggests that the equal variance assumption is violated.
  - For the Nornal Q-Q plot, we can see a slight fat tail on the right of the plot, which I suspect the assumption of 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.

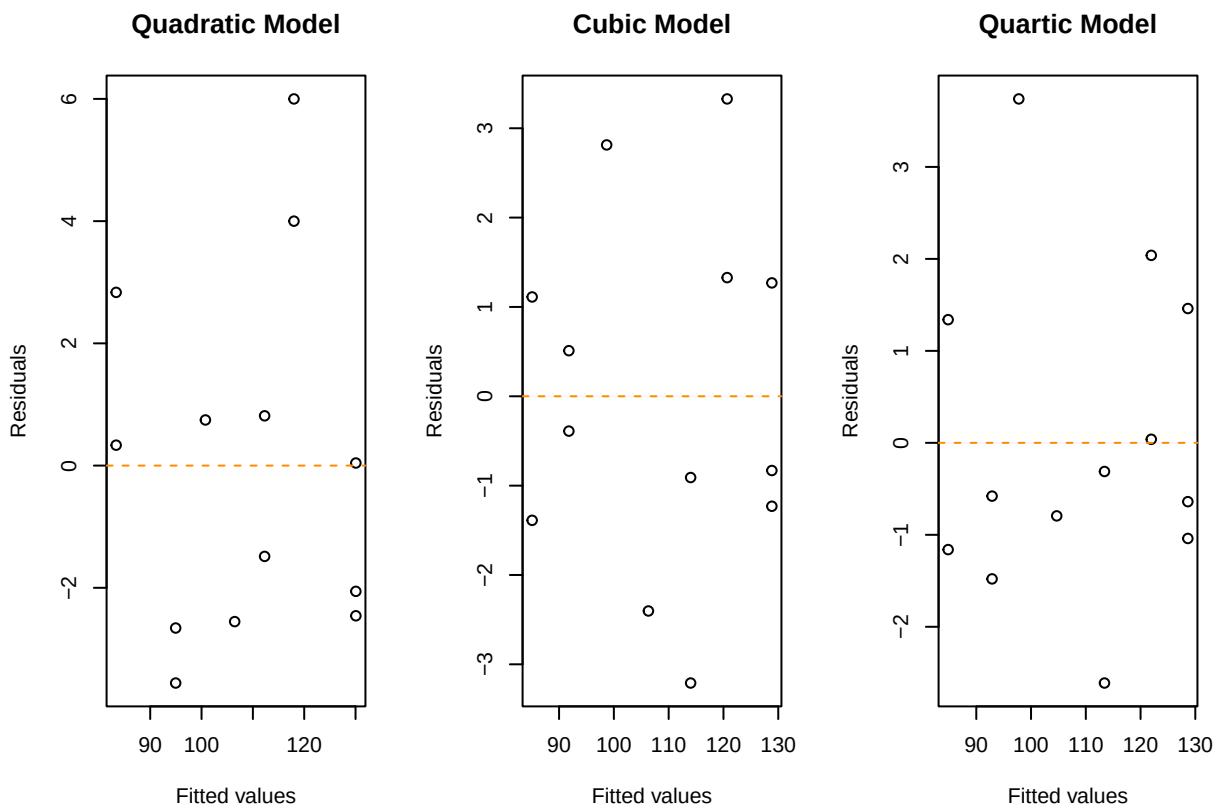
```
model2 = lm(loss ~ Fe + I(Fe^2), data=corrosion)
model3 = lm(loss ~ Fe + I(Fe^2) + I(Fe^3), data=corrosion)
model4 = lm(loss ~ Fe + I(Fe^2) + I(Fe^3) + I(Fe^4), data=corrosion)

par(mfrow=c(1,3))

plot(fitted(model2), resid(model2),
     main="Quadratic Model",
     xlab="Fitted values",
     ylab="Residuals")
abline(h=0, col="darkorange", lty=2)

plot(fitted(model3), resid(model3),
     main="Cubic Model",
     xlab="Fitted values",
     ylab="Residuals")
abline(h=0, col="darkorange", lty=2)

plot(fitted(model4), resid(model4),
     main="Quartic Model",
     xlab="Fitted values",
     ylab="Residuals")
abline(h=0, col="darkorange", lty=2)
```



```r
anova(model2, model3, model4)
```

```
## Analysis of Variance Table
##
## Model 1: loss ~ Fe + I(Fe^2)
## Model 2: loss ~ Fe + I(Fe^2) + I(Fe^3)
## Model 3: loss ~ Fe + I(Fe^2) + I(Fe^3) + I(Fe^4)
##   Res.Df   RSS Df Sum of Sq    F Pr(>F)
## 1     10 100.1
## 2      9  45.1  1      55.0 12.6 0.0075 **
## 3      8  35.0  1      10.1  2.3 0.1675
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
shapiro.test(resid(model3))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(model3)
## W = 0.97, p-value = 0.9
```

```r
cd = cooks.distance(model3)
cd_high_cutoff = 4 / length(cd)
index = which(cd > cd_high_cutoff)
corrosion[index, ]
```

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

- Based on the plots, the Cubic model seems to be the best in my opinion, as the distribution of the residuals seem to be scattered equally no matter the value of fitter value is.
- Based on anova table:
  - Comparing model2 (quadratic) to model3 (cubic):
    - The p-value is 0.0075 (< 0.05), indicating that adding the cubic term significantly improves the model
    - The RSS decreases substantially from 100.1 to 45.1
  - Comparing model3 (cubic) to model4 (quartic):
    - The p-value is 0.1675 (> 0.05), indicating that adding the quartic term does not significantly improve the model
    - The RSS only decreases slightly from 45.1 to 35.0
  - Therefore, Cubic model is the best choice according to he anova table.
- The p-value of the SW test for this model(Cubic Model) is 0.9, which is greater than most of the common alpha values. Therefore, the normality assumption for this model is satisfied.
- There's no influential points based on the threshold of $\frac{4}{n}$.

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.

```r
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.

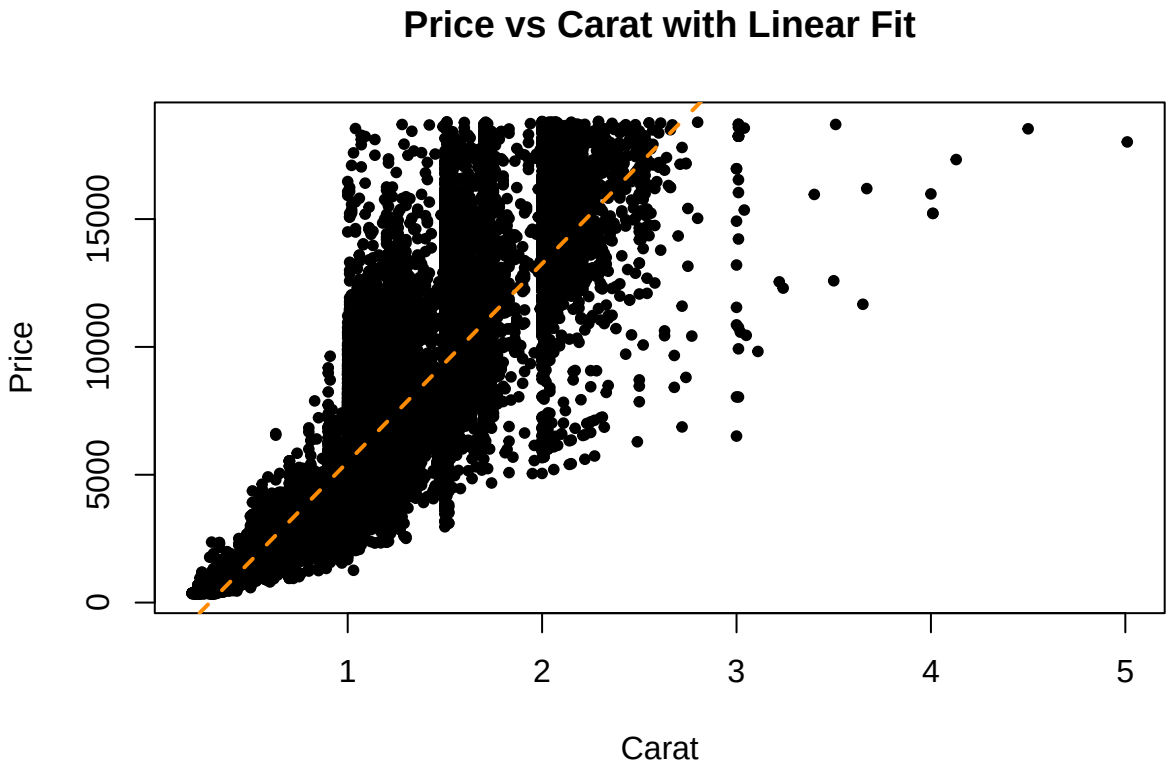
```
model_linear = lm(price ~ carat, data = diamonds)
summary(model_linear)
```

```
##
## 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
```

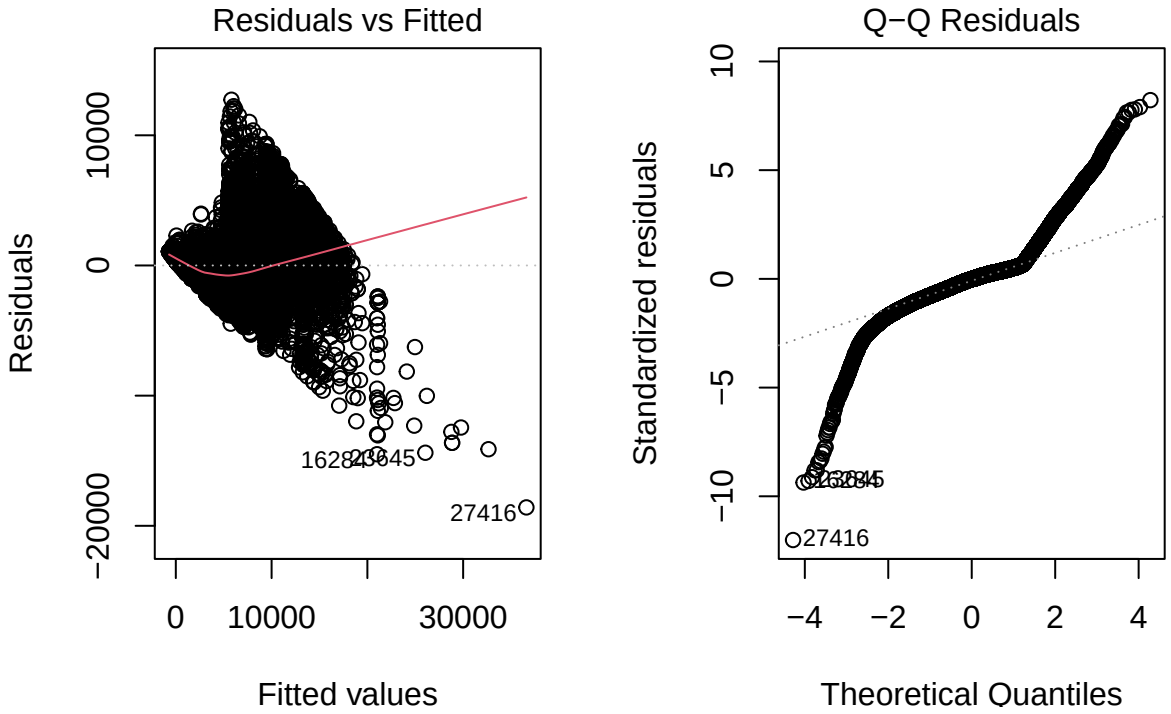
(b) Plot a scatterplot of price versus carat and add the line for the fitted model in part **(a)**. Using a fitted versus residuals plot and/or a Q-Q plot, comment on the diagnostics.

```
plot(diamonds$carat, diamonds$price,
     pch = 20,
     xlab = "Carat",
     ylab = "Price",
     main = "Price vs Carat with Linear Fit")

abline(model_linear, col = "darkorange", lwd = 2, lty = 2)
```




```
par(mfrow=c(1,2))
plot(model_linear, which=1)
plot(model_linear, which=2)
```



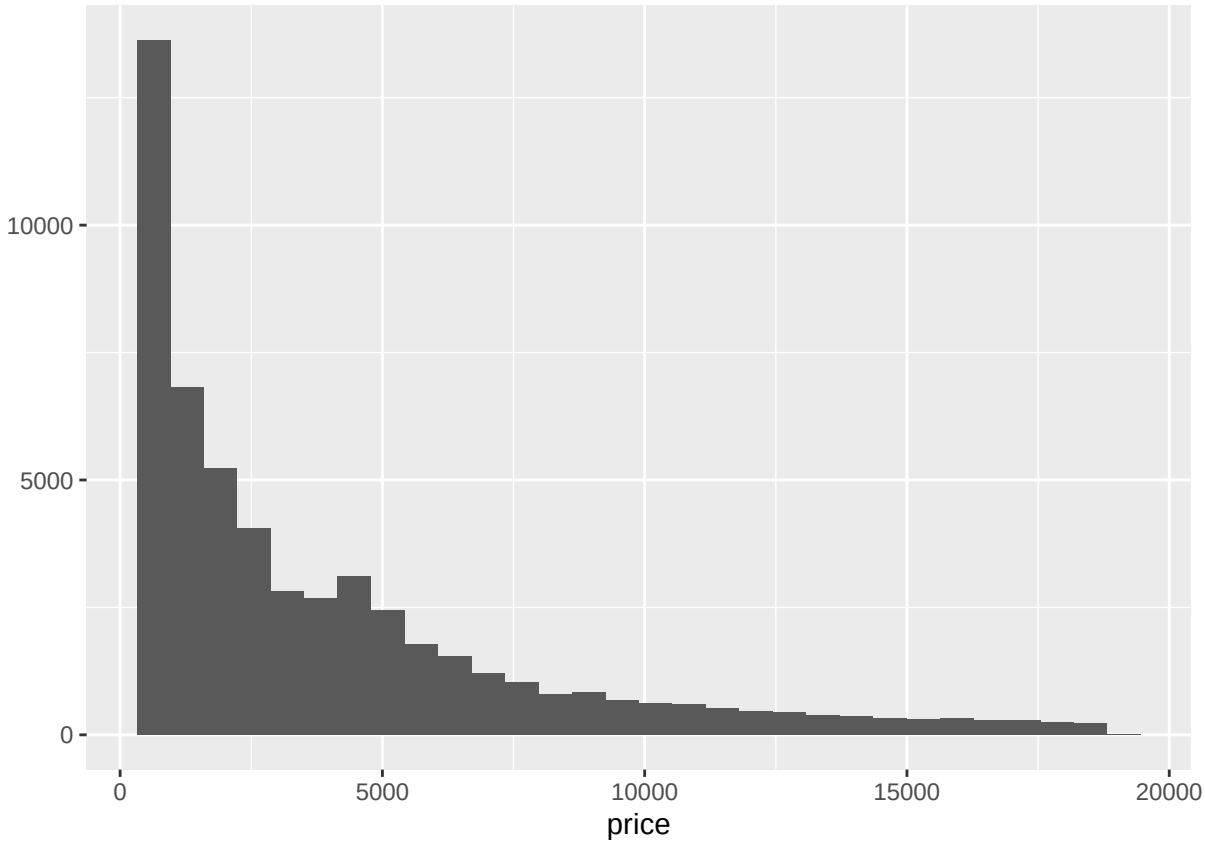
- For linearity: According to the scattered plot, we can see that the relationship is something other than linear, as the regression doesn't fit so well.
- For equal variance assumption: The distribution of the residuals seem to have a relationship with fitted

value, therefore the variance of the distribution of the noise may not be equal.
- For normality assumption: Flat tails are observed within the Normal Q-Q plot, suggesting that the noise may not follow a normal distribution.

(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)
```

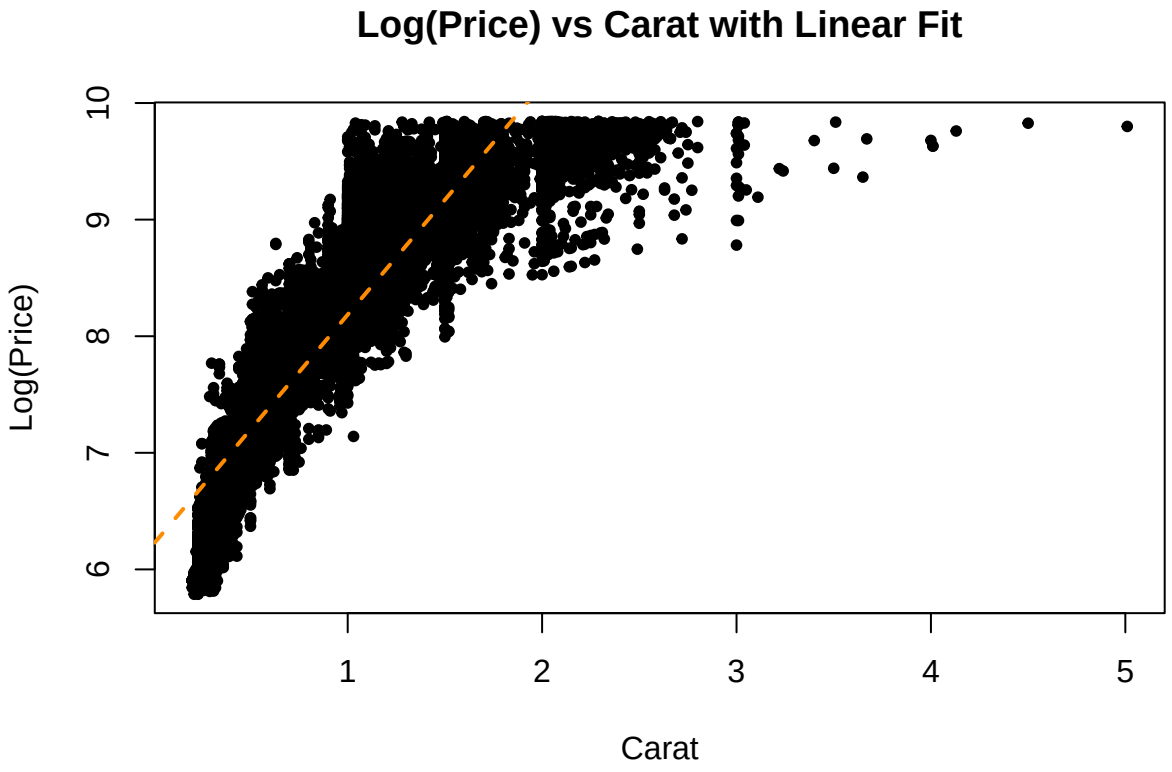
```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



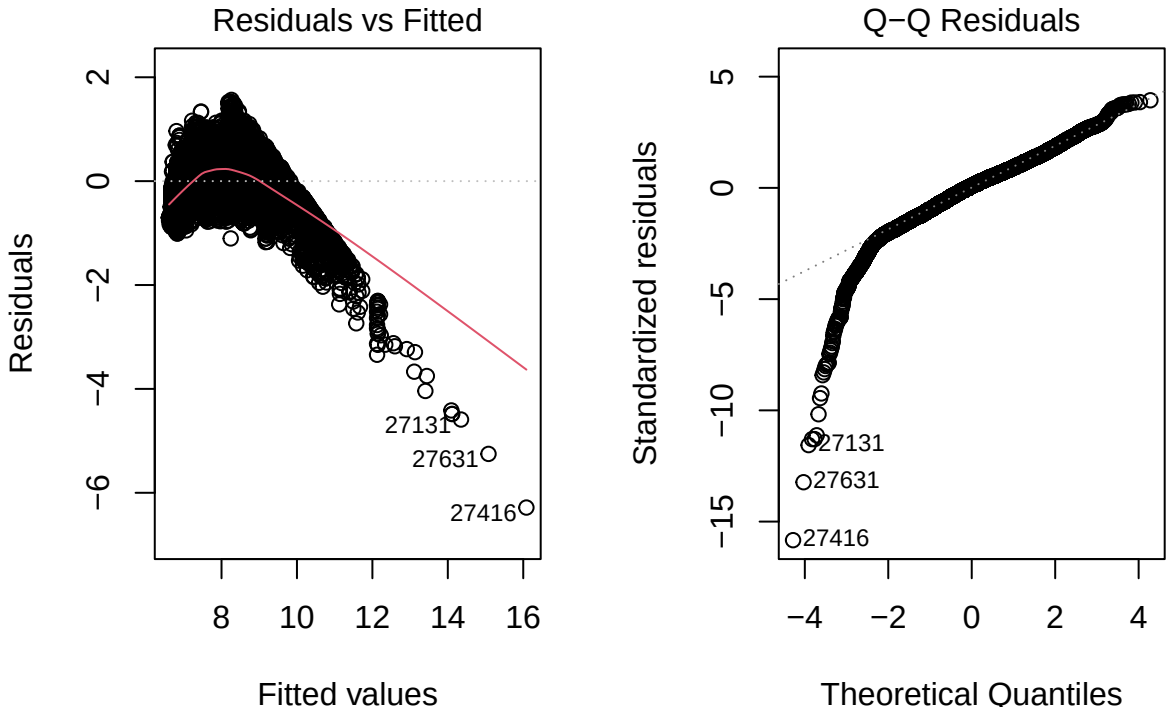

```
model_log_y = lm(log(price) ~ carat, data = diamonds)

plot(diamonds$carat, log(diamonds$price),
     pch = 20,
     xlab = "Carat",
     ylab = "Log(Price)",
     main = "Log(Price) vs Carat with Linear Fit")

abline(model_log_y, col = "darkorange", lwd = 2, lty = 2)
```




```
par(mfrow=c(1,2))
plot(model_log_y, which=1)
plot(model_log_y, which=2)
```



- For linearity: According to the scattered plot, we can see that the relationship is something other than linear, as the regression doesn't fit so well.
- For equal variance assumption: The distribution of the residuals seem to have a relationship with fitted

value, therefore the variance of the distribution of the noise may not be equal.

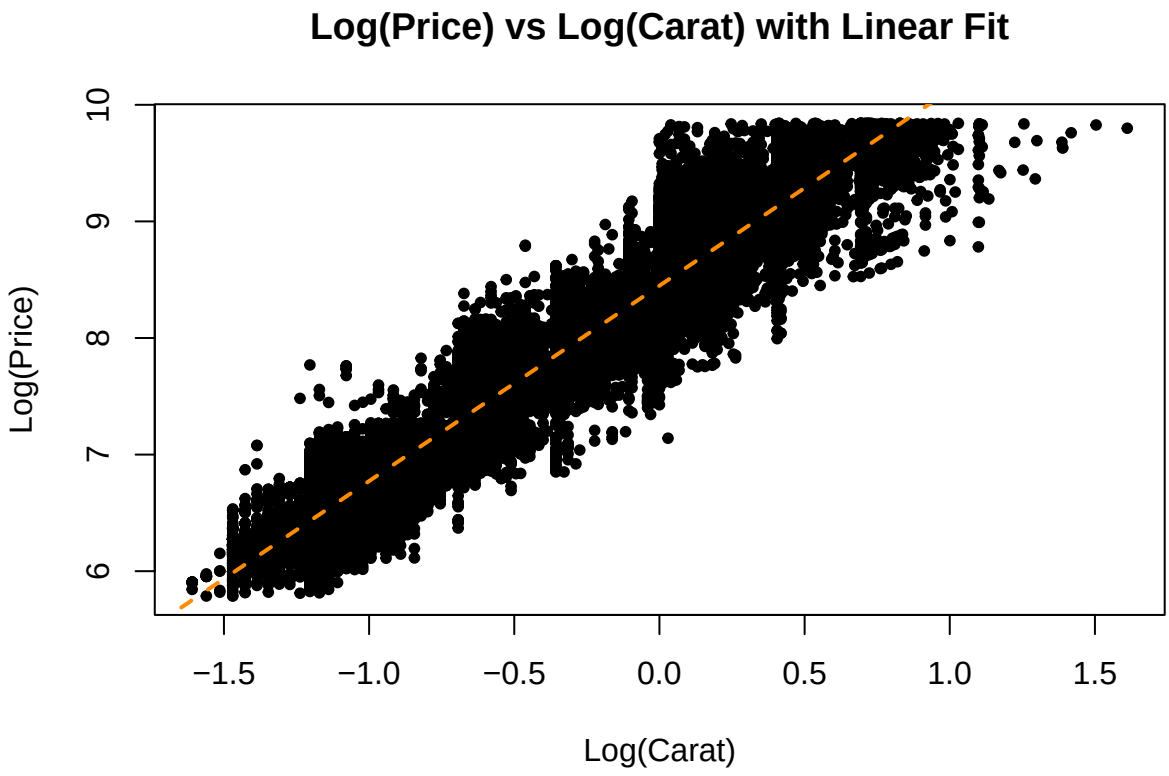
- For normality assumption: Flat tail is observed within the Normal Q-Q plot, suggesting that the noise may not follow a normal distribution.

(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.

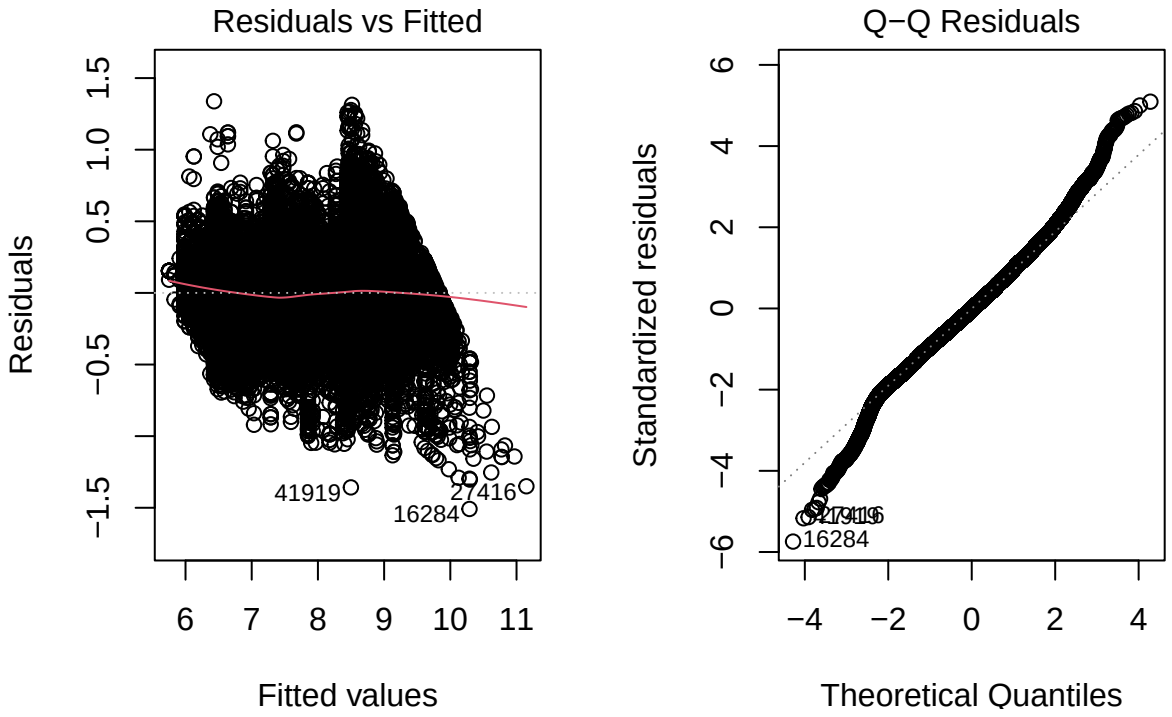
```
model_log_both = lm(log(price) ~ log(carat), data = diamonds)

plot(log(diamonds$carat), log(diamonds$price),
     pch = 20,
     xlab = "Log(Carat)",
     ylab = "Log(Price)",
     main = "Log(Price) vs Log(Carat) with Linear Fit")

abline(model_log_both, col = "darkorange", lwd = 2, lty = 2)
```




```
par(mfrow=c(1,2))
plot(model_log_both, which=1)
plot(model_log_both, which=2)
```



- For linearity: According to the scattered plot, we can see that the relationship is indeed linear, as the regression fits well.
- For equal variance assumption: The distribution of the residuals seem to have a relationship with fitted value, therefore the variance of the distribution of the noise may not be equal.
- For normality assumption: Flat tail is observed within the Normal Q-Q plot, suggesting that the noise may not follow a normal distribution.

(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).

```
new_data = data.frame(carat = 3)
log_pred = predict(model_log_both,
                   newdata = data.frame(carat = log(3)),
                   interval = "prediction",
                   level = 0.99)
exp(log_pred)
```

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