Week 8 - Homework

STAT 420, Summer 2023, D. Unger

Scott Girten
NetID: sgirten2

Directions

Students are encouraged to work together on homework. However, sharing, copying or providing any part of a homework solution or code is an infraction of the University's rules on Academic Integrity. Any violation will be punished as severely as possible.

- Be sure to remove this section if you use this `.Rmd` file as a template.
- You may leave the questions in your final document.

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:

```
library(tidyverse)
library(patchwork)
# #
# model = fit_1
# pcol = 'grey'
# lcol = 'dodgerblue'
# alpha = 0.05
# plotit = TRUE
# testit = TRUE

diagnostics = function(model, pcol = 'grey', lcol = 'dodgerblue', alpha = 0.05, plotit = TRUE, testit =

  # Create individual diagnostic components

  # Fitted vs Residuals plot
  p1 = ggplot(model, aes(x = .fitted, y = .resid)) +
    geom_point(color = pcol) +
    geom_hline(yintercept = 0, color = lcol) +
    theme_minimal() +
    labs(x = 'Fitted',
         y = 'Residuals',
         title = 'Fitted vs. Residuals Plot')

  # Normal Q-Q Plot
  p2 = ggplot(model, aes(sample = .resid)) +
    geom_qq(color = pcol) +
    geom_qq_line(color = lcol) +
    theme_minimal() +
    labs(x = 'Theoretical Quartiles',
         y = 'Sample Quartiles',
         title = 'Normal Q-Q Plot')

  # Combine plots side by side
  p_final = p1 + p2


  # p-value and decision
  p_val = shapiro.test(model$residuals)$p.value
  decision = if_else(p_val < alpha, 'Reject', 'Fail to Reject')


  # Logic for which diagnostics should be returned by function
  output = list('p_val' = p_val, 'decision' = decision, 'plots' = p_final)

  if(plotit == TRUE & testit == TRUE){
    return(output)
  }
  else if(plotit == TRUE & testit == FALSE){
    return(output[3])
  }
```

```
  else if(plotit == FALSE & testit == TRUE){
    return(output[1:2])
  }

}
```

(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
```

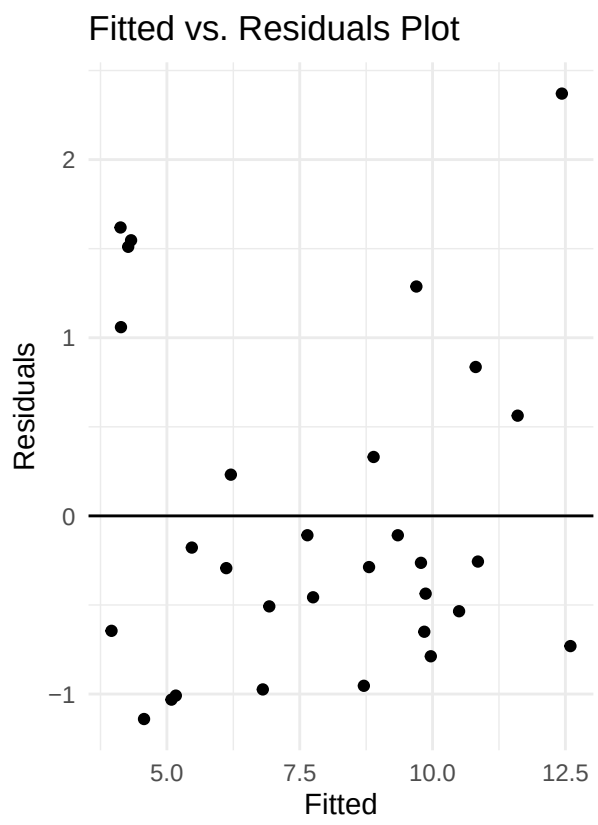
```
## [1] 0.005613
```

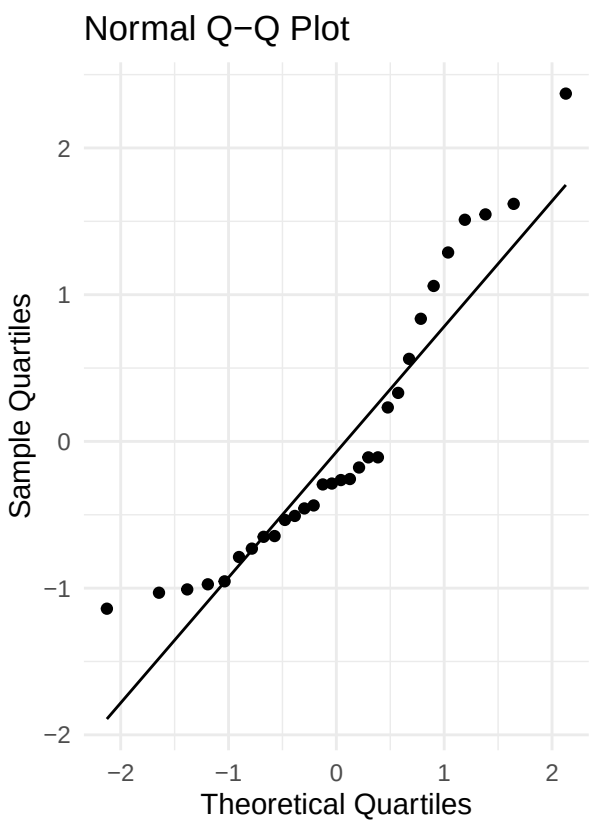
```
diagnostics(fit_2, plotit = FALSE)$decision
```

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

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

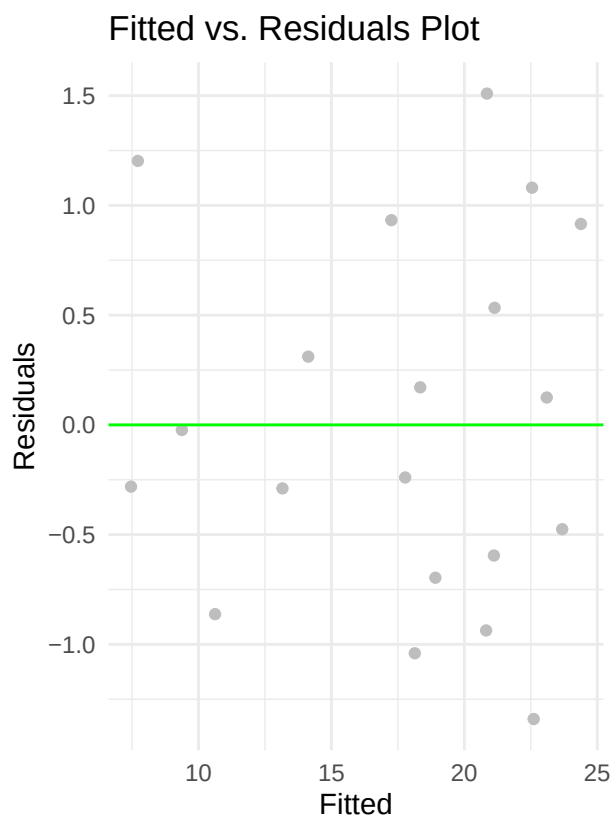
```
## $plots
```

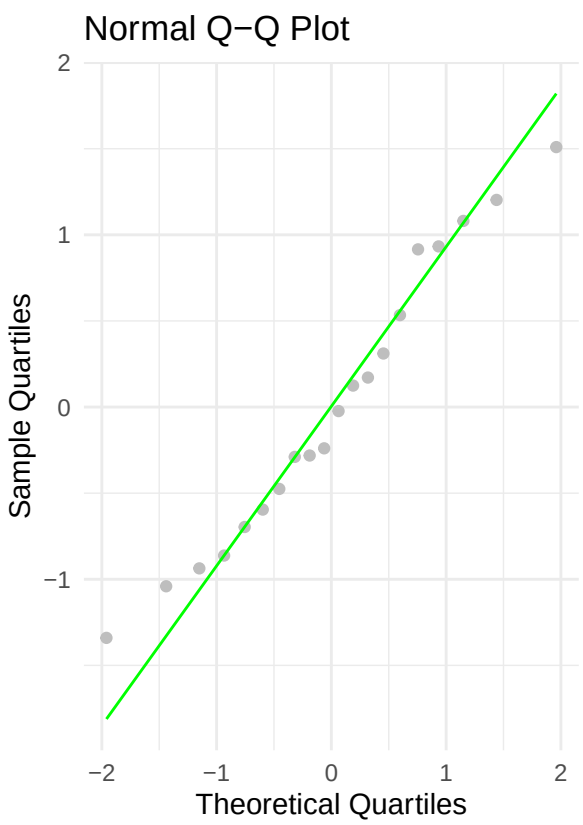





```
diagnostics(fit_2, testit = FALSE, pcol = "grey", lcol = "green")
```

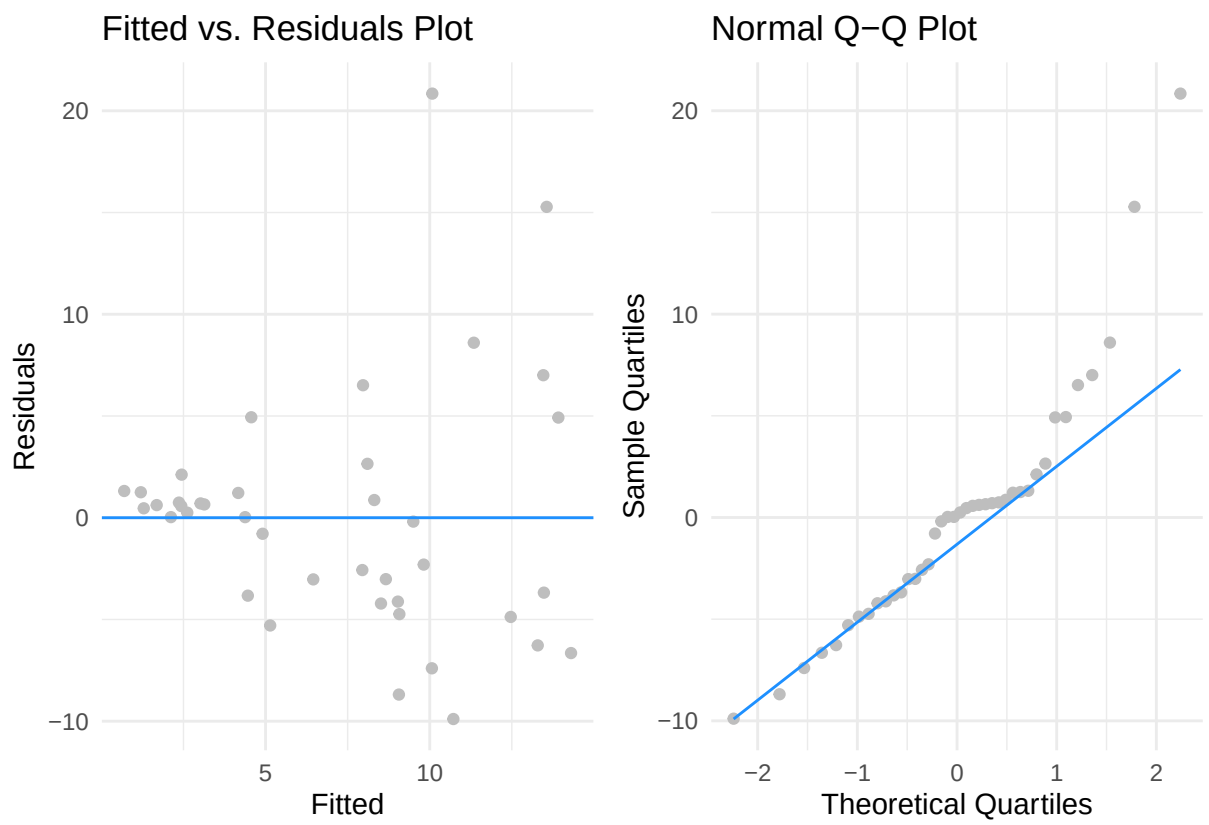
```
## $plots
```






```
diagnostics(fit_3)
```

```
## $p_val
## [1] 0.001608
##
## $decision
## [1] "Reject"
##
## $plots
```



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)
#?prostate
```

(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:

```
mod = lm(lpsa ~ ., data = prostate)
r_2 = summary(mod)$r.squared
```

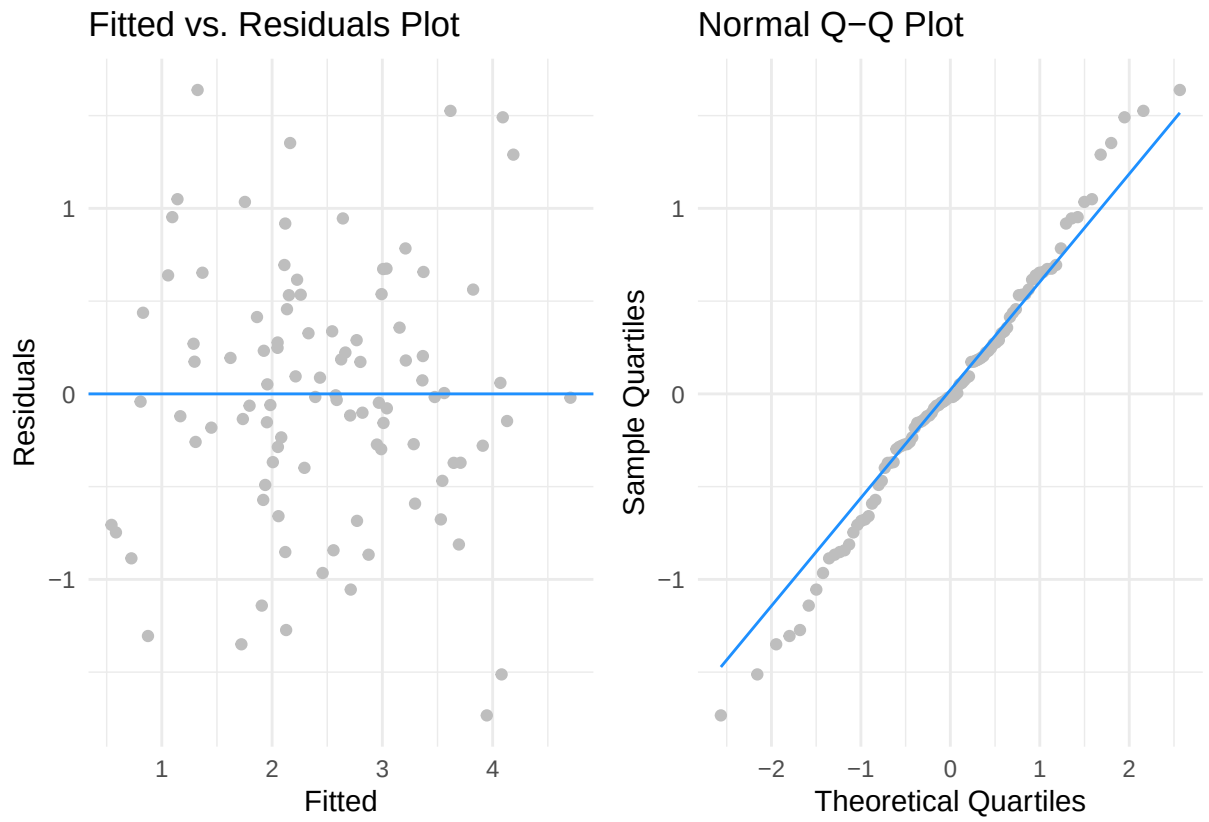
The R^2 for this additive regression model is 0.6548.

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

Solution:

```
diagnostics(model = mod, testit = FALSE)
```

```
## $plots
```



I do not feel that the constant variance assumption has been violated for this model since the Fitted vs. Residuals plot does not show a pattern in the residuals and the residuals seem to show a roughly constant variance for the fitted values.

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

Solution:

```
pval = diagnostics(model = mod, plotit = FALSE)$p_val
decision = diagnostics(model = mod, plotit = FALSE)$decision
```

Since the p-value for the Shapiro-Wilk Test of Normality is 0.7721, I would fail to reject the null hypothesis for this test (that the data is from a normal distribution) and assume that the normality assumption is correct for this model.

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

Solution:

```
library(kableExtra)

# Identify large leverage observations
```

Table 1: High Leverage Observations

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

```
large_lev = hatvalues(mod) > 2 * mean(hatvalues(mod))

# Add leverage boolean column to original data set to filter out observations with high leverage
large_lev_obs = prostate %>%
  bind_cols(large_lev) %>%
  rename(large_lev_obs = 10) %>%
  filter(large_lev_obs == TRUE)

# Table display ouput
large_lev_obs %>%
  kbl(caption = 'High Leverage Observations') %>%
  kable_styling()
```

The 5 observations in the table above are high leverage observations.

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

Solution:

```
# Get list of large standardized residuals
large_rstandard = rstandard(mod)[abs(rstandard(mod)) > 2]
large_rstandard_index = as.numeric(names(large_rstandard))

# Determine if the observations with large residuals are influential
influential_obs = cooks.distance(mod)[large_rstandard_index] > 4 / length(cooks.distance(mod))

# All of the observations are influential
influential = prostate %>%
  filter(row_number() %in% as.numeric(names(influential_obs)))

# Table display output
influential %>%
  kbl(caption = 'Influential Observations') %>%
  kable_styling()
```

The 5 observations in the table above are influential observations in the prostate data set.

(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:

```
# DF with influential observations removed
prostate_cleaned = prostate %>%
```

Table 2: Influential Observations

	lcavol	lweight	age	lbph	svi	lcp	gleason	pgg45	lpsa
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
97	3.4720	3.975	68	0.4383	1	2.904	7	20	5.583

Table 3: Influential Observations

Predictor	Coefficients with All Observations	Coefficients with Influential Observations Removed
(Intercept)	0.6693	0.5873
lcavol	0.5870	0.6001
lweight	0.4545	0.3698
age	-0.0196	-0.0180
lbph	0.1071	0.1612
svi	0.7662	0.8106
lcp	-0.1055	-0.1682
gleason	0.0451	0.0648
pgg45	0.0045	0.0084

```
  filter(!row_number() %in% as.numeric(names(influential_obs)))

# Refit model with influential observations removed
mod2 = lm(lpsa ~ ., data = prostate_cleaned)


mod_coef = tibble(Predictor = names(coef(mod)),
                  `Coefficients with All Observations` = coef(mod),
                  `Coefficients with Influential Observations Removed` = coef(mod2))

# Table display output
mod_coef %>%
  kbl(caption = 'Influential Observations') %>%
  kable_styling()
```

(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:

```
# Use the influential observation dataframe from part e

# Fit influential data points to each model
predict_mod = predict(mod, newdata = influential)
predict_mod2 = predict(mod2, newdata = influential)

# Create table for display
predict_table = tibble(`Model Prediction (All Observations)` = predict_mod,
                       `Model Prediction (Influential Observations Removed)` = predict_mod2)
```

Table 4: Model Predictions

Model Prediction (All Observations)	Model Prediction (Influential Observations Removed)
3.947	3.939
4.081	4.363
1.325	1.104
3.618	3.360
4.092	3.928

```
# Display Table output
predict_table %>%
  kbl(caption = 'Model Predictions') %>%
  kable_styling()
```

The predictions for the model with influential observations removed has a wider range of predictions versus the original model which was fit with all observations.

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 = 19790221
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){

  # Model 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)
  p_val_1[i] = summary(fit_1)$coefficients[3,4]

  # Model 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_2[i] = summary(fit_2)$coefficients[3,4]
}
```

Table 5: Model p-values

Model	Less than 0.01	Less than 0.05	Less than 0.1
Model 1	1.00%	4.84%	9.68%
Model 2	3.40%	10.52%	18.12%

(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:

```
library(scales)

# Create dataframes for each model's p-value to display
df_p1 = tibble(`Model` = 'Model 1',
               `p-value` = p_val_1)

df_p2 = tibble(Model = 'Model 2',
               `p-value` = p_val_2)

df = df_p1 %>%
  bind_rows(df_p2) %>%
  mutate(`Less than 0.01` = if_else(`p-value` < 0.01, 1, 0),
         `Less than 0.05` = if_else(`p-value` < 0.05, 1, 0),
         `Less than 0.1` = if_else(`p-value` < 0.1, 1, 0))

df_table = df %>%
  group_by(Model) %>%
  summarise(`Less than 0.01` = percent(mean(`Less than 0.01`), 0.01),
            `Less than 0.05` = percent(mean(`Less than 0.05`), 0.01),
            `Less than 0.1` =  percent(mean(`Less than 0.1`), 0.01))

# Display Table output
df_table %>%
  kbl(caption = 'Model p-values') %>%
  kable_styling()
```

Since model 1 is fit with data that does not violate assumptions about the normality of the error term, we see that the percent of p-values less than a specified threshold is roughly the value of that threshold and would allow for a proper conclusion of whether $\beta_0 = 0$ at a given level of confidence (α). However, model 2 is fit with data that does violate the assumption of normality for the error term and because of that assumption not being correct, it would cause an incorrect rejection of the *Null* hypothesis that $\beta_2 = 0$ (at a given level of α) for a larger proportion of observations and would erroneously allow for the conclusion that $\beta_2 \neq 0$ when in fact $\beta_2 = 0$.

Exercise 4 (Corrosion Data)

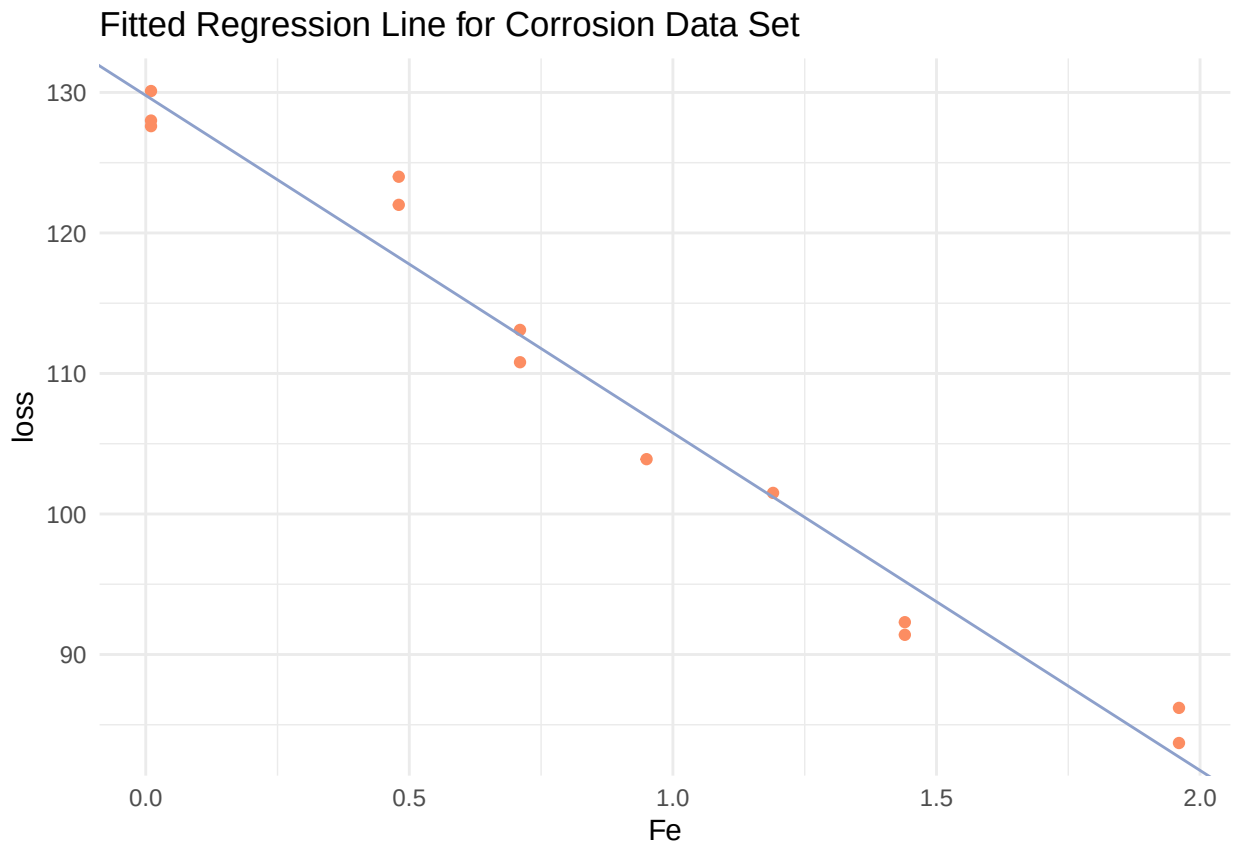
For this exercise, we will use the `corrosion` data, which can be found in the `faraway` package. After loading the `faraway` package, use `?corrosion` to learn about this dataset.

```
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.

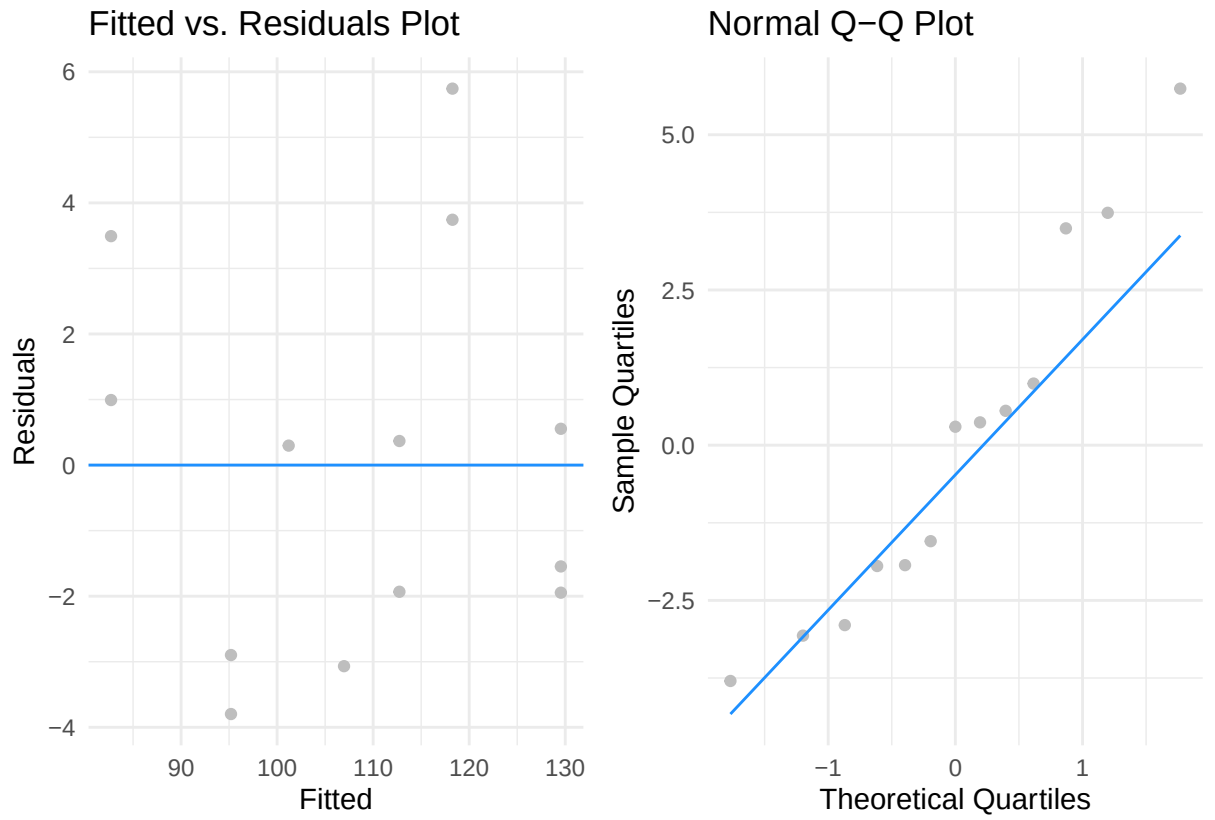
Solution:

```
#?corrosion

# Fit model
mod = lm(loss ~ Fe, data = corrosion)

intercept = coef(mod)[1]
slope = coef(mod)[2]

# Scatter plot and fitted line from regression
p = ggplot(corrosion, aes(x = Fe, y = loss)) +
  geom_point(color = '#fc8d62') +
  geom_abline(intercept = intercept, slope = slope, color = '#8da0cb') +
  theme_minimal() +
  labs(title = 'Fitted Regression Line for Corrosion Data Set')

p
```



```
corrosion_diag = diagnostics(mod)
corrosion_bptest = bptest(mod)$p.value

corrosion_diag[3]
```

```
## $plots
```



p-value for Shapiro-Wilk Test: 0.373283274510585
p-value for Breusch-Pagan Test: 0.8755

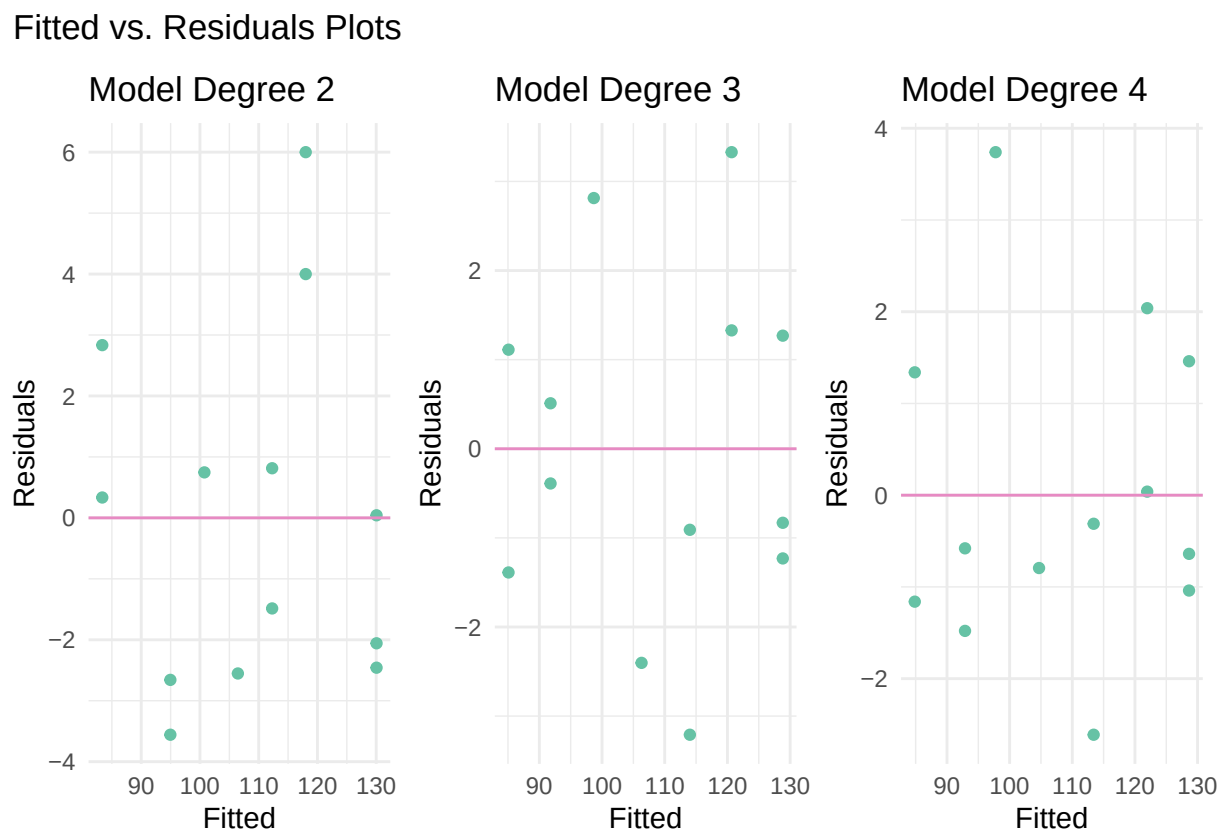
The p-values for both the Shapiro-Wilk & BP test support the null hypothesis in both tests which gives support to the assumptions that the model residuals are normally distributed and the data set has a constant variance.

(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:

```
library(patchwork)

# function for fitted vs. residuals plot
fit_resid = function(model, pcol, lcol, title = 'Fitted vs. Residuals Plot'){
```

```
  p = ggplot(model, aes(x = .fitted, y = .resid)) +
  geom_point(color = pcol) +
  geom_hline(yintercept = 0, color = lcol) +
  theme_minimal() +
  labs(x = 'Fitted',
       y = 'Residuals',
       title = title)

  return(p)
}


# Fitted models of higher degree
mod2 = lm(loss ~ poly(Fe, 2), raw = TRUE, data = corrosion)
mod3 = lm(loss ~ poly(Fe, 3), raw = TRUE, data = corrosion)
mod4 = lm(loss ~ poly(Fe, 4), raw = TRUE, data = corrosion)

mod2_p = fit_resid(mod2, pcol = '#66c2a5', lcol = '#e78ac3', 'Model Degree 2')
mod3_p = fit_resid(mod3, pcol = '#66c2a5', lcol = '#e78ac3', 'Model Degree 3')
mod4_p = fit_resid(mod4, pcol = '#66c2a5', lcol = '#e78ac3', 'Model Degree 4')

plt = mod2_p + mod3_p + mod4_p
plt + plot_annotation(title = 'Fitted vs. Residuals Plots')
```



Based on the fitted vs. residual plots for each model, the assumption of normality for the residuals seems

true for all 3 models. Given that the variance of the fitted vs residual plot seems to be roughly the same for all 3 plots, I would choose the model with the 2nd order polynomial.

```
#  test models 2 & 3
anova(mod2, mod3)
```

```
## Analysis of Variance Table
##
## Model 1: loss ~ poly(Fe, 2)
## Model 2: loss ~ poly(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
```

```
# test models 3 & 4
anova(mod3, mod4)
```

```
## 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
```

Comparing models of degree 2 & 3, the model of degree 3 is significant for the ANOVA F-test. Performing an ANOVA test comparing models of degree 3 & 4 shows no difference in model performance for degree 3 & 4. Using the results of the ANOVA test, I would select model of polynomial degree 3 to be the preferred model.

```
#normality test for all 3 models
shaprio_test = c(shapiro.test(mod2$residuals)$p.value,
                 shapiro.test(mod3$residuals)$p.value,
                 shapiro.test(mod4$residuals)$p.value)

shapiro_table = tibble(`Model Polynomial Order` = c('Polynomial Order 2', 'Polynomial Order 3', 'Polyno
                       `Shapiro Test p-value` = shaprio_test)

# Display Table output
shapiro_table %>%
  kbl(caption = 'Higher Order Models') %>%
  kable_styling()
```

The Shapiro-Wilk test supports the assumption that all 3 of the model residuals are from normal distributions.

```
# Get list of large standardized residuals
large_rstandard = rstandard(mod2)[abs(rstandard(mod2)) > 2]
large_rstandard_index = as.numeric(names(large_rstandard))
```

Table 6: Higher Order Models

Model Polynomial Order	Shapiro Test p-value
Polynomial Order 2	0.2565
Polynomial Order 3	0.9085
Polynomial Order 4	0.4022

```
# Determine if the observations with large residuals are influential
influential_obs = cooks.distance(mod2)[large_rstandard_index] > 4 / length(cooks.distance(mod2))
influential_obs
```

```
##     2
## FALSE
```

None of the observations in the data set appear to be 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.

Solution:

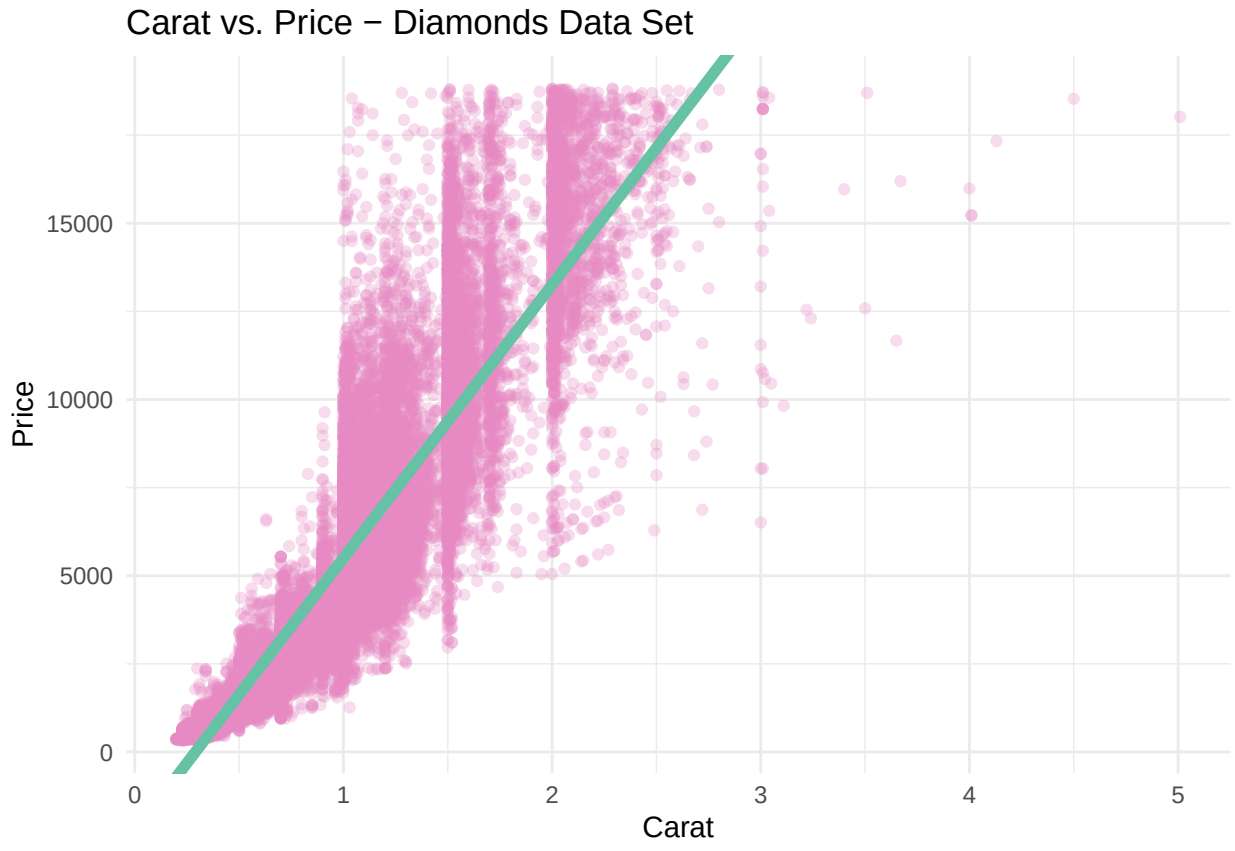
```
mod = lm(price ~ carat, data = diamonds)
summary(mod)
```

```
##
## 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.

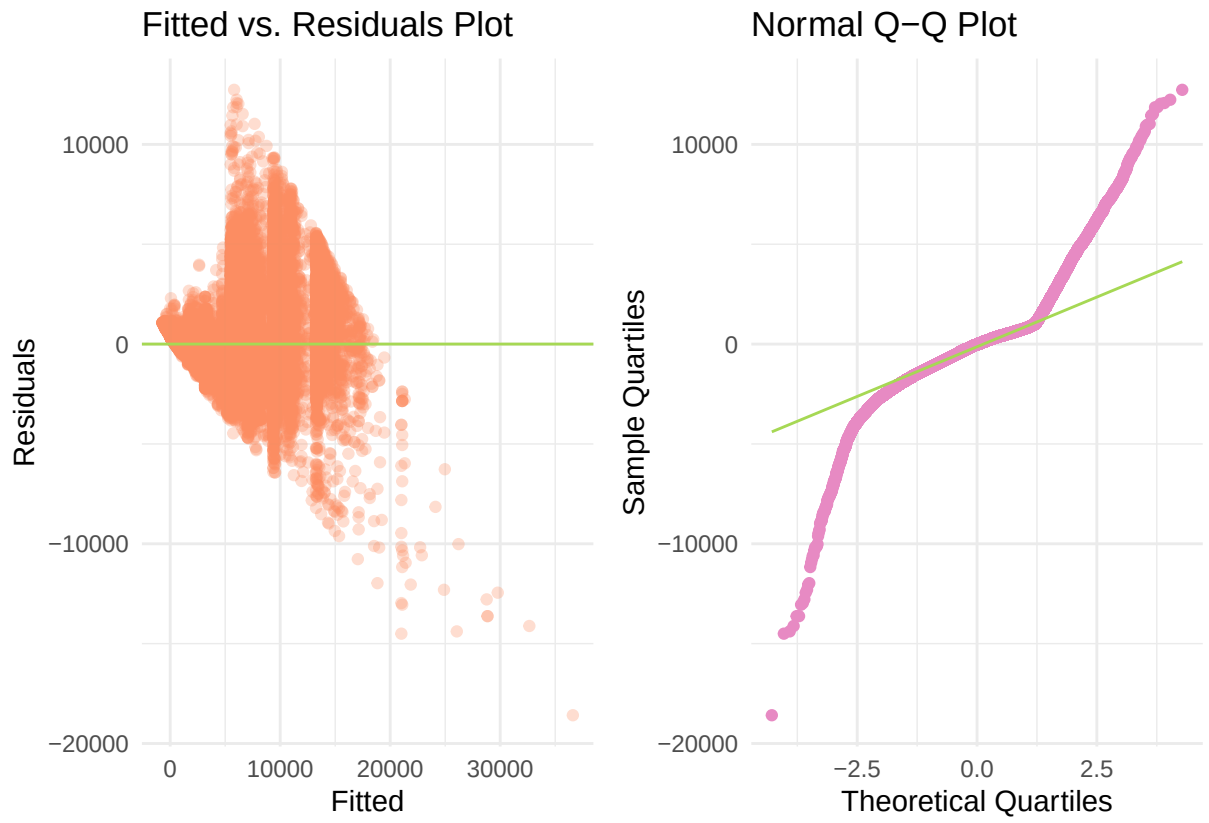
Solution:

```
intercept = coef(mod)[1]
slope = coef(mod)[2]

# Scatter plot
p = ggplot(diamonds, aes(x = carat, y = price)) +
  geom_point(color = '#e78ac3', alpha = 0.3) +
  geom_abline(intercept = intercept, slope = slope, color = '#66c2a5', size = 2) +
  theme_minimal() +
  labs(x = 'Carat',
       y = 'Price',
       title = 'Carat vs. Price - Diamonds Data Set')

p
```




```
  # Fitted vs Residuals plot
  p1 = ggplot(mod, aes(x = .fitted, y = .resid)) +
    geom_point(color = '#fc8d62', alpha = 0.3) +
    geom_hline(yintercept = 0, color = '#a6d854') +
    theme_minimal() +
    labs(x = 'Fitted',
         y = 'Residuals',
```

```
        title = 'Fitted vs. Residuals Plot')

# Normal Q-Q Plot
p2 = ggplot(mod, aes(sample = .resid)) +
  geom_qq(color = '#e78ac3') +
  geom_qq_line(color = '#a6d854') +
  theme_minimal() +
  labs(x = 'Theoretical Quartiles',
       y = 'Sample Quartiles',
       title = 'Normal Q-Q Plot')

# Combine plots side by side
p_final = p1 + p2
p_final
```

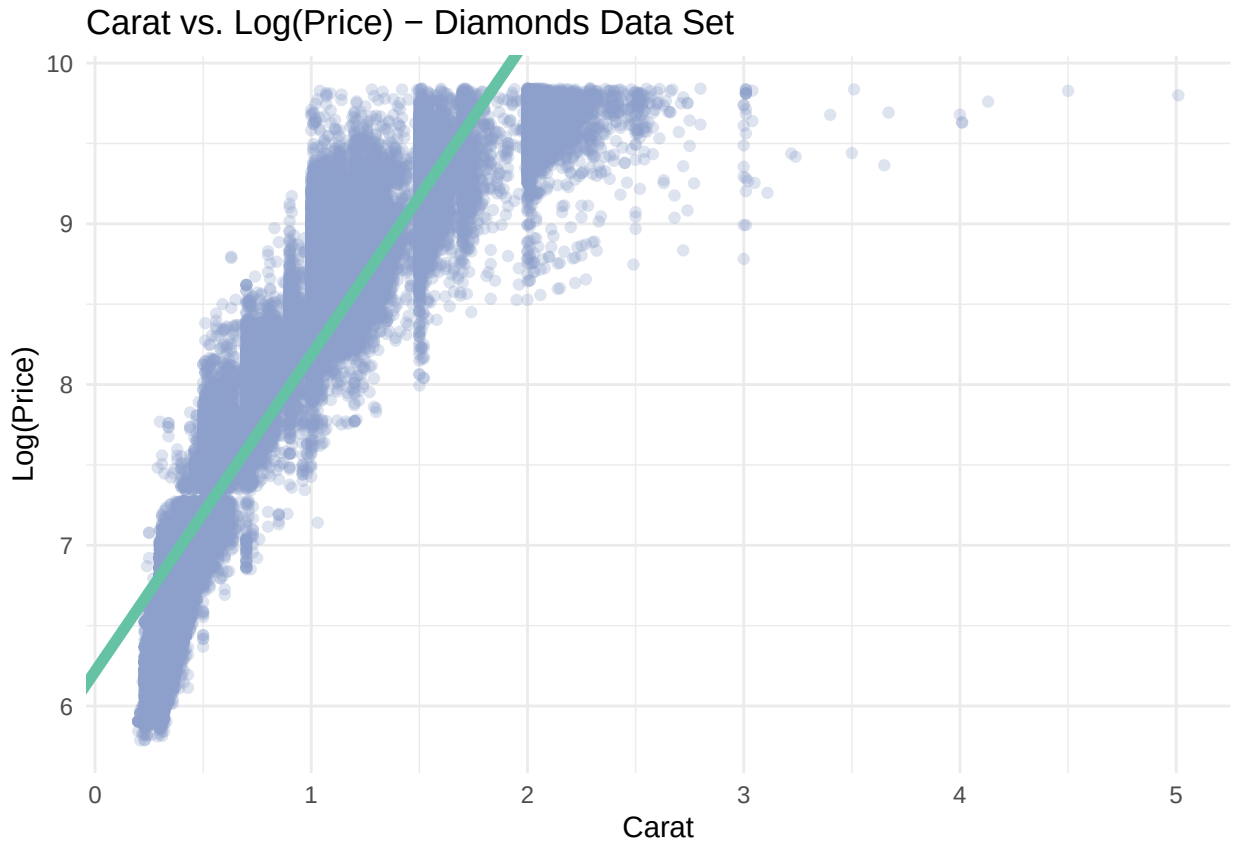


Both the fitted vs. residuals and the Normal Q-Q plot suggest a non-constant variance and non-normality for the fitted model.

(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.

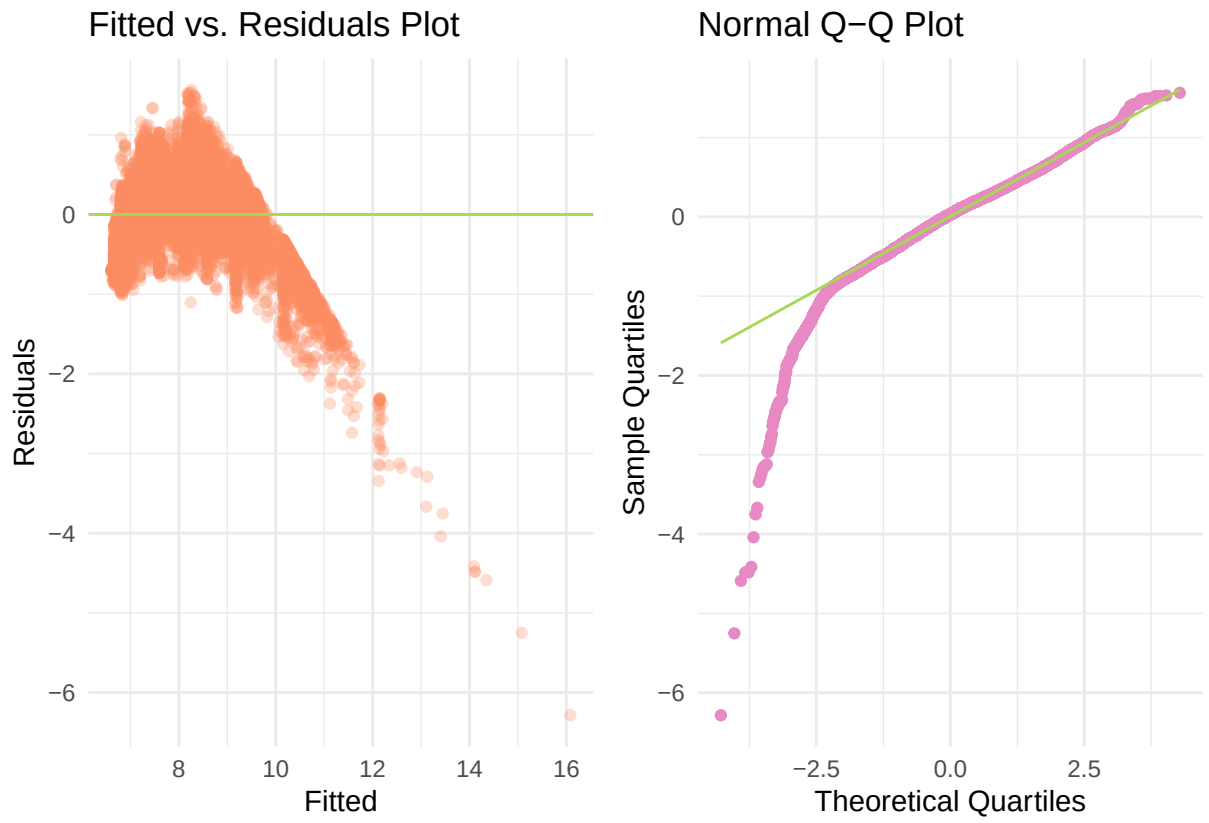
Solution:

```r
# fit model with response log transformation
mod_log_resp = lm(log(price) ~ carat, data = diamonds)

# Slope and intercept for regression line
intcpt = coef(mod_log_resp)[1]
slp = coef(mod_log_resp)[2]

# Scatter plot
p = ggplot(diamonds, aes(x = carat, y = log(price))) +
  geom_point(color = '#8da0cb', alpha = 0.3) +
  geom_abline(intercept = intcpt, slope = slp, color = '#66c2a5', size = 2) +
  theme_minimal() +
  labs(x = 'Carat',
       y = 'Log(Price)',
       title = 'Carat vs. Log(Price) - Diamonds Data Set')

p
```




```r
  # Fitted vs Residuals plot
  p1 = ggplot(mod_log_resp, aes(x = .fitted, y = .resid)) +
    geom_point(color = '#fc8d62', alpha = 0.3) +
    geom_hline(yintercept = 0, color = '#a6d854') +
    theme_minimal() +
    labs(x = 'Fitted',
         y = 'Residuals',
```

```
        title = 'Fitted vs. Residuals Plot')

# Normal Q-Q Plot
p2 = ggplot(mod_log_resp, aes(sample = .resid)) +
  geom_qq(color = '#e78ac3') +
  geom_qq_line(color = '#a6d854') +
  theme_minimal() +
  labs(x = 'Theoretical Quartiles',
       y = 'Sample Quartiles',
       title = 'Normal Q-Q Plot')

# Combine plots side by side
p_final = p1 + p2
p_final
```

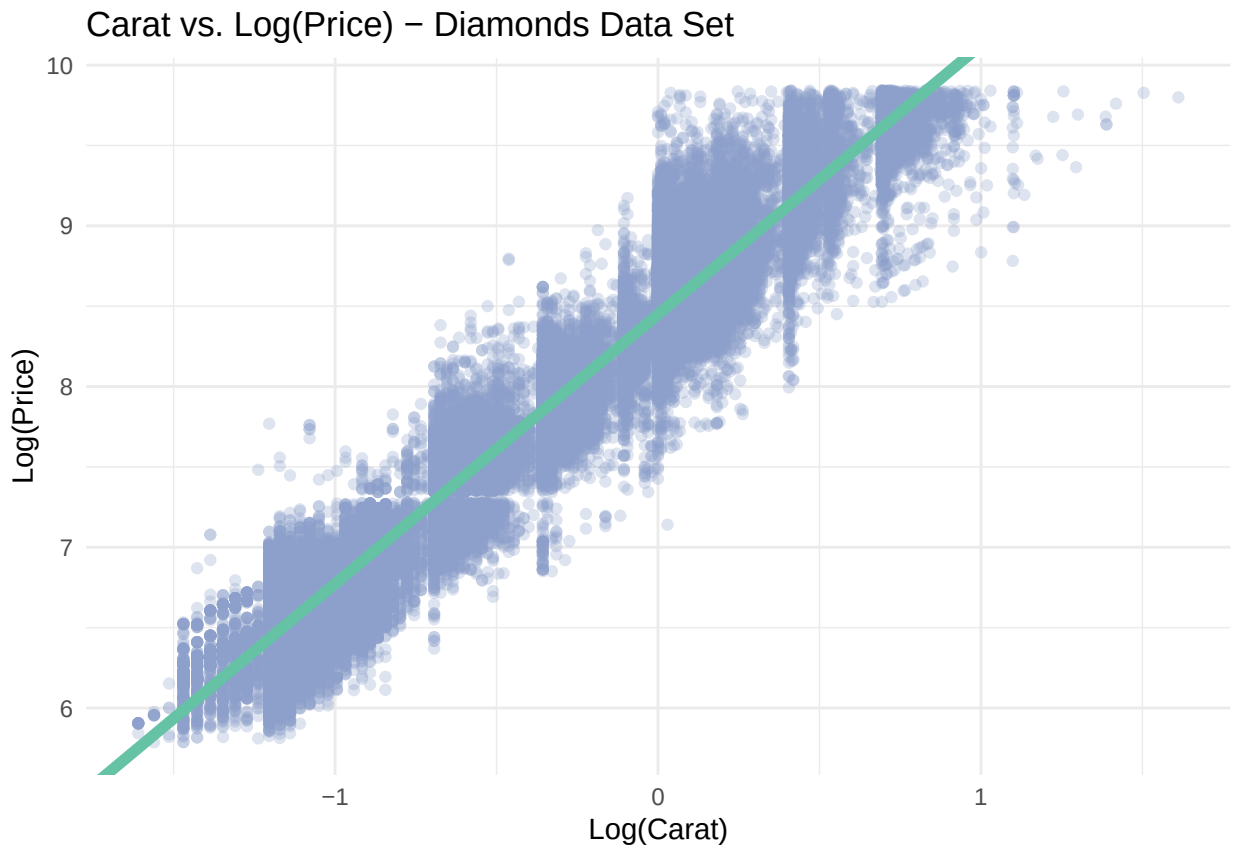


The log transformation of the response variable did seem to somewhat improve the diagnostic plots. The left side of the fitted vs residuals plot seems to start with a constant variance but then develops a tail downward indicating that the variance is changing as the fitted values increase. The Q-Q plot now only has a deviation from the theoretical values on the left side of the plot instead of both sides in the original un-transformed model.

(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.

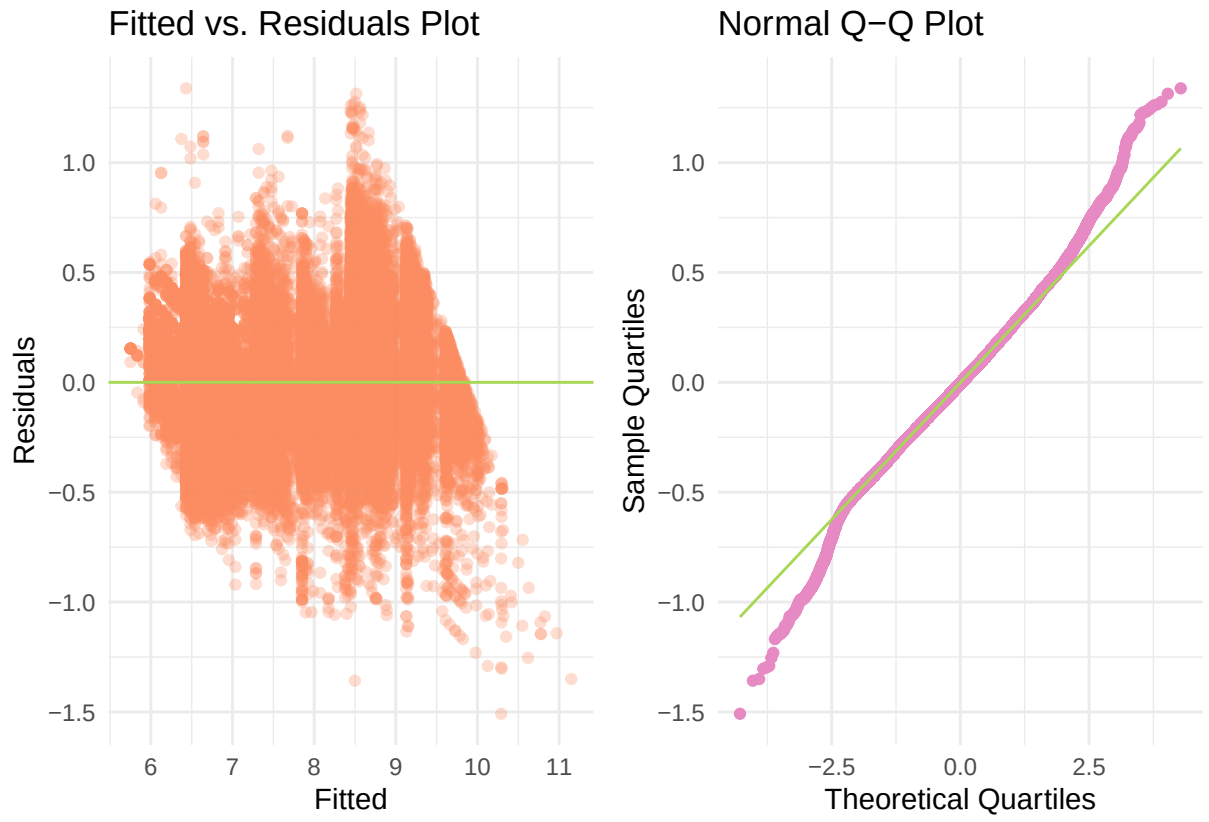
Solution:

```r
# Fit model
mod_log_resp2 = lm(log(price) ~ log(carat), data = diamonds)

# slope and intercept
slp = coef(mod_log_resp2)[2]
int = coef(mod_log_resp2)[1]

# Scatter plot
p = ggplot(diamonds, aes(x = log(carat), y = log(price))) +
  geom_point(color = '#8da0cb', alpha = 0.3) +
  geom_abline(intercept = int, slope = slp, color = '#66c2a5', size = 2) +
  theme_minimal() +
  labs(x = 'Log(Carat)',
       y = 'Log(Price)',
       title = 'Carat vs. Log(Price) - Diamonds Data Set')

p
```




```r
  # Fitted vs Residuals plot
  p1 = ggplot(mod_log_resp2, aes(x = .fitted, y = .resid)) +
    geom_point(color = '#fc8d62', alpha = 0.3) +
    geom_hline(yintercept = 0, color = '#a6d854') +
    theme_minimal() +
    labs(x = 'Fitted',
         y = 'Residuals',
```

```
       title = 'Fitted vs. Residuals Plot')

# Normal Q-Q Plot
p2 = ggplot(mod_log_resp2, aes(sample = .resid)) +
  geom_qq(color = '#e78ac3') +
  geom_qq_line(color = '#a6d854') +
  theme_minimal() +
  labs(x = 'Theoretical Quartiles',
       y = 'Sample Quartiles',
       title = 'Normal Q-Q Plot')

# Combine plots side by side
p_final = p1 + p2
p_final
```



The log transformation of the predictor improved upon the previous model where only the response variable was transformed. The fitted vs. residuals plot does seem to suggest a roughly constant variance for the residuals. The Normal Q-Q plot also seems to suggest a normal distribution for the model 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).

Solution:

```
# Prediction
interval = exp(predict(mod_log_resp2, newdata = data.frame(carat = log(3)), interval = 'prediction', le
```

```r
# Format output
lwr=  formatC(interval[2], format="f", digits=2, big.mark=",")
upr = formatC(interval[3], format="f", digits=2, big.mark=",")
```

The 99% prediction interval for the price of a 3-carat diamond is (2,778.51, 10,752.21)