Week 8 - Homework

STAT 420, Summer 2023, D. 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.

```
diagnostics = function(model, pcol = "grey", lcol = "dodgerblue", alpha = 0.05,
                       plotit = TRUE, testit = TRUE){
  if(testit){
    p_val = shapiro.test(resid(model))$p.value
    if(p_val < alpha){
      decision = "Reject"
    }else{
      decision = "Fail to Reject"
    }
    result = list(p_val = p_val, decision = decision)
  }else{
    result = NULL
```

```
  }
  if(plotit){
    par(mfrow = c(1,2))
    plot(fitted(model), resid(model), pch = 20, col = pcol,
         xlab = "Fitted",
         ylab = "Residual",
         main = "Fitted vs. Residual")
    abline(h = 0, col = lcol, lwd = 2)

    qqnorm(resid(model), main = "Normal Q-Q plot", pch = 20, col = pcol)
    qqline(resid(model), col = lcol, lwd = 2)
  }
  return(result)
}
```

(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)
```

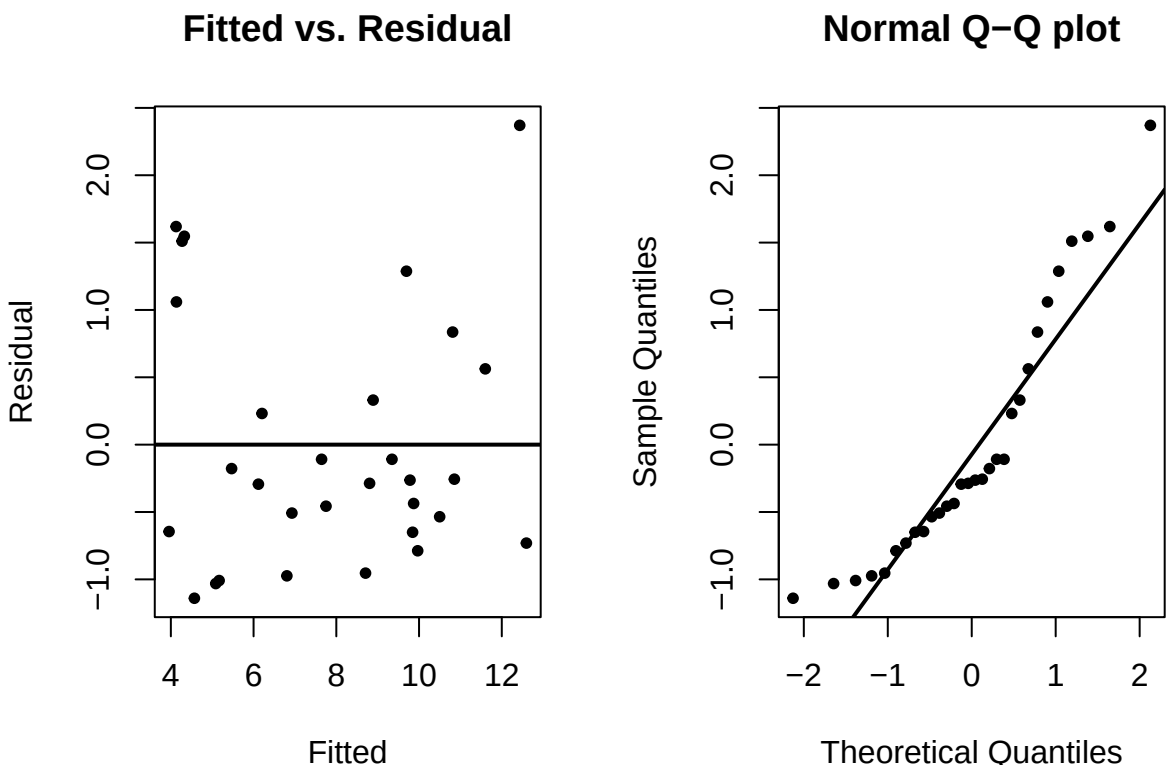
```
diagnostics(fit_1, plotit = FALSE)$p_val
```

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

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

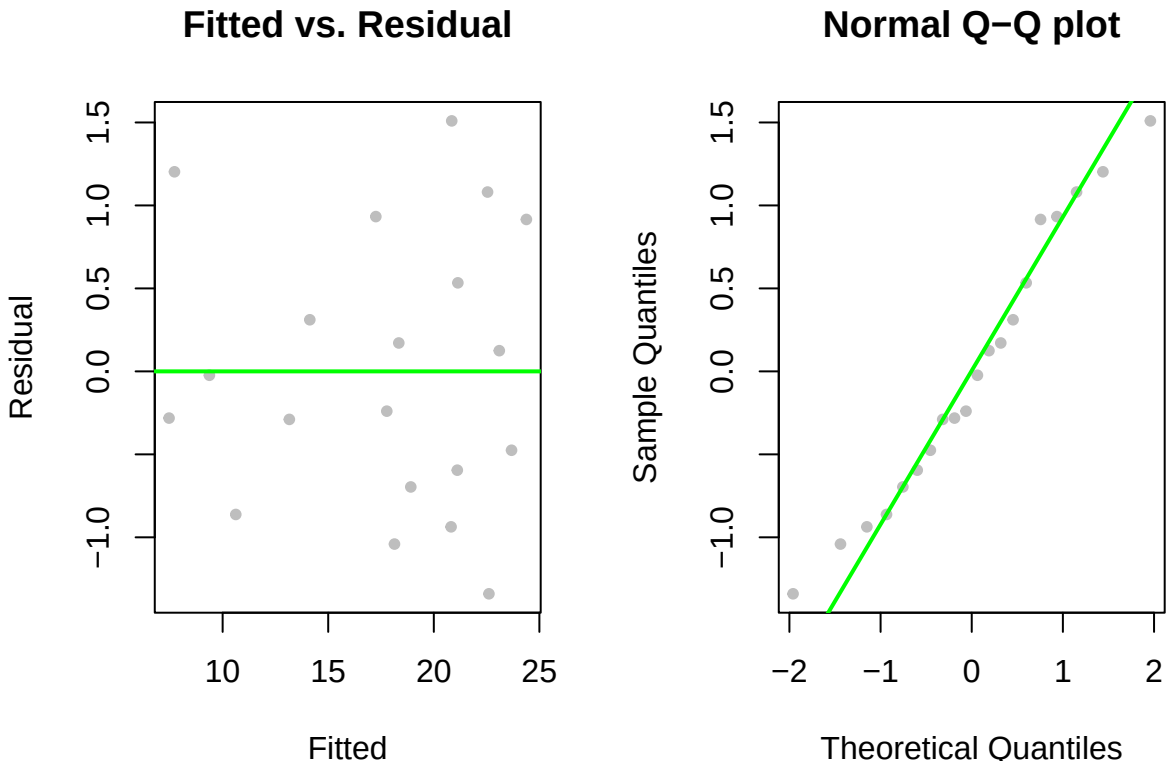
```
## [1] "Fail to Reject"
```

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



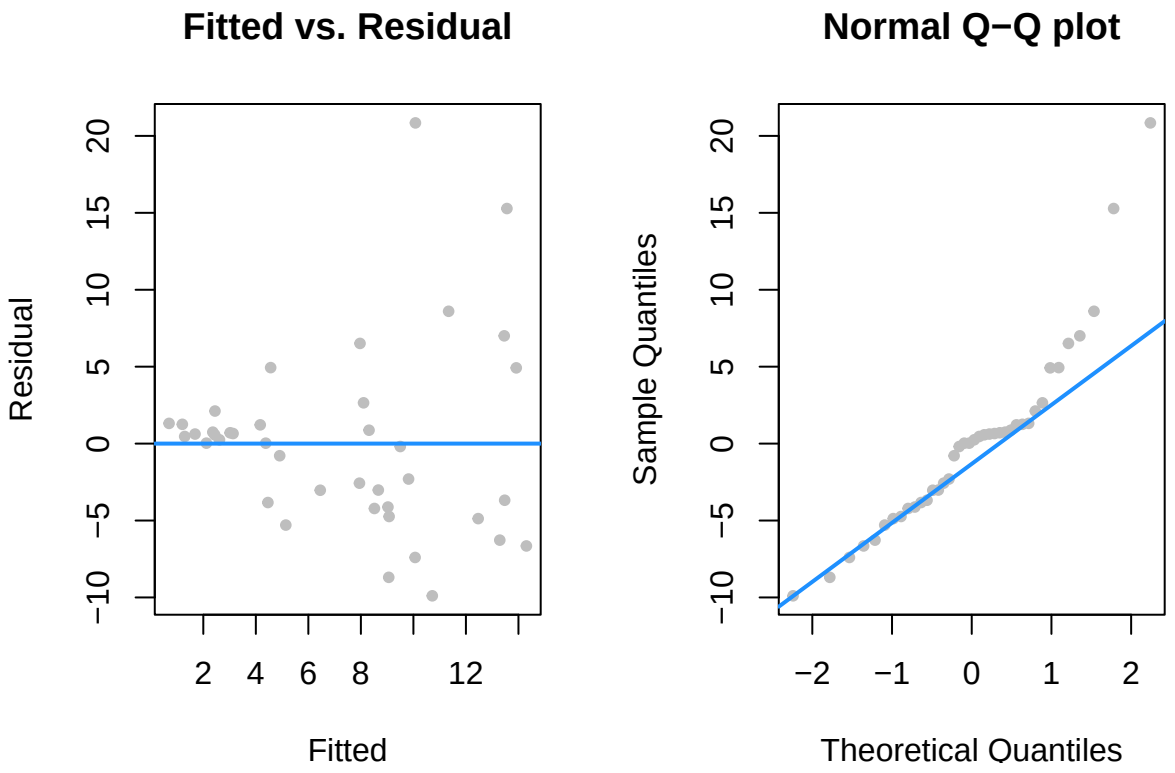

```
## NULL
```

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




```
## NULL
```

```
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)
```

```
library(lmtest)
```

(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.

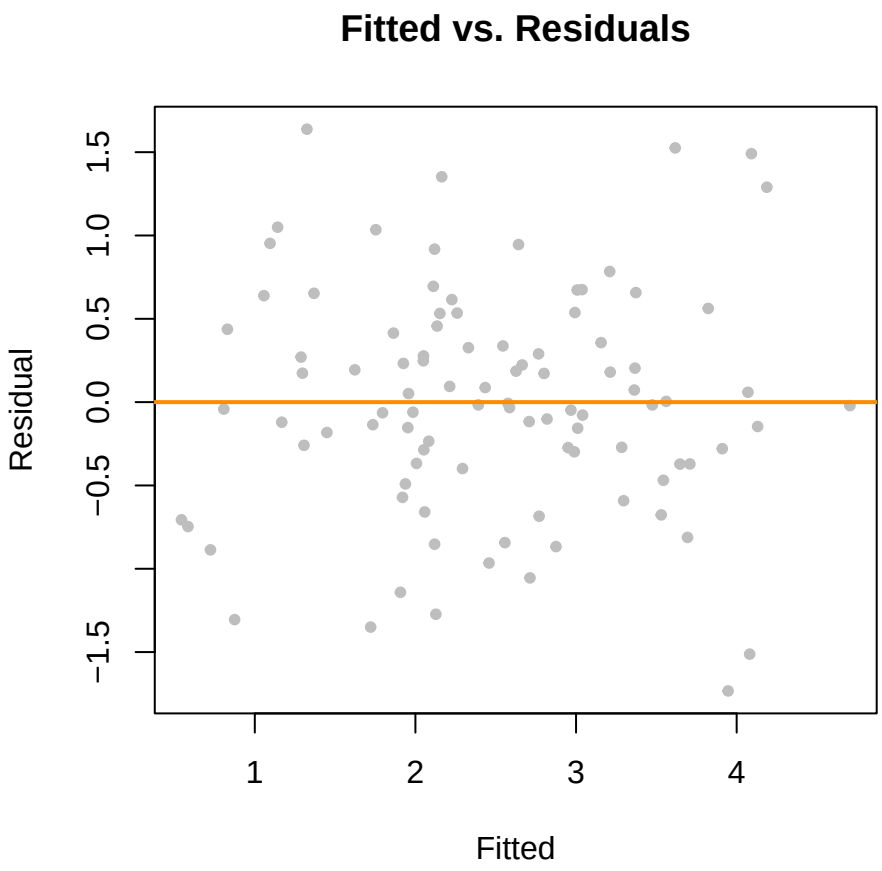
```
prostate_add = lm(lpsa ~ ., data = prostate)
summary(prostate_add)$r.squared
```

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

Solution: R^2 value: 0.6548

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

```
plot(fitted(prostate_add), resid(prostate_add), col = "grey", pch = 20,
     xlab = "Fitted", ylab = "Residual",
     main = "Fitted vs. Residuals")
abline(h = 0, col = "darkorange", lwd = 2)
```



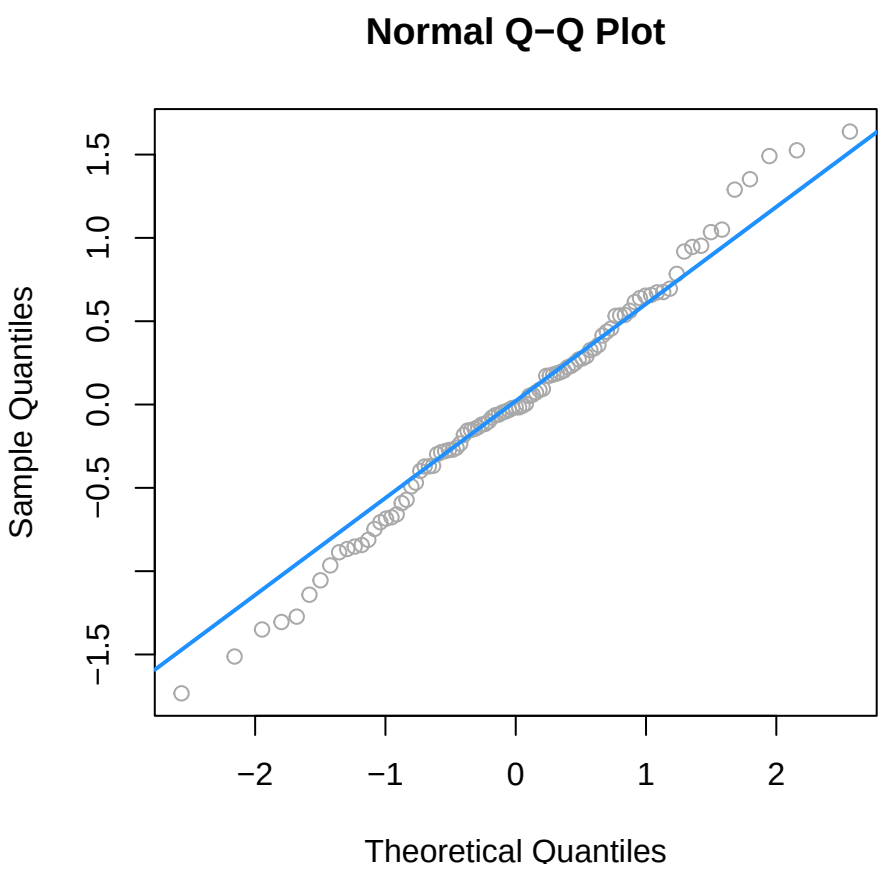

```
bptest(prostate_add)
```

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

Solution: The fitted versus residuals plot looks good. We don't see any obvious pattern, and the variance looks roughly constant. So we check the bptest and check the p-value. p-value = 0.3. It is pretty large p-value and we fail to reject null hypothesis. Constant variance assumption is not suspect.

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

```r
qqnorm(resid(prostate_add), col = "darkgrey")
qqline(resid(prostate_add), col = "dodgerblue", lwd = 2)
```




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

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

Solution:
Normal Q-Q Plot looks not bad. And we confirm with p-value of shapiro-wilk test(p-value = 0.8) We fail to reject the null hypothesis and normality is not suspect.

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

```r
which.max(hatvalues(prostate_add))
```

```
## 32
## 32
```

```r
sum(hatvalues(prostate_add) > 2 * mean(hatvalues(prostate_add)))
```

```
## [1] 5
```

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

```
##    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
```

Solution:
There are 5 observations that have large leverage. Observation 32 has highest leverage.

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

```
out_liers= rstandard(prostate_add)[abs(rstandard(prostate_add)) > 2]

influential_idx = which(cooks.distance(prostate_add) > 4 / nrow(prostate))
#prostate[influential_idx,]

intersect(as.numeric(names(out_liers)), as.numeric(influential_idx))
```

```
## [1] 39 47 69 95 97
```

```
(influential = prostate[names(out_liers), ])
```

```
##     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
```

Solution: Based on the previous test for high leverage and large residual, 5 observations have both high leverage and large residual and thus considered to be 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.

```
prostate_add_cd = lm(lpsa ~ ., data = prostate,
                     subset = cooks.distance(prostate_add) <= 4 / nrow(prostate))
coef(prostate_add_cd)
```

```
## (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
```

```
coef(prostate_add)
```

```
## (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(prostate_add_cd) - coef(prostate_add)
```

```
## (Intercept)      lcavol     lweight         age        lbph         svi
##   -0.915412   -0.022027    0.089959    0.001074    0.026225   -0.021408
##         lcp     gleason       pgg45
##   -0.049945    0.069583    0.002125
```

Solution:
We can see that coefficients for lcavol, svi, and lcp were being over-estimated from the 7 influential observations.

(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:

```
removed = data.frame(prostate[names(out_liers), ])
predict(prostate_add, newdata = removed) - predict(prostate_add_cd, newdata = removed)
```

```
##       39       47       69       95       97
##  0.04897 -0.20268 -0.01548  0.29059  0.18666
```

Model with influential observations.

```
predict(prostate_add, newdata = removed) - removed$lpsa
```

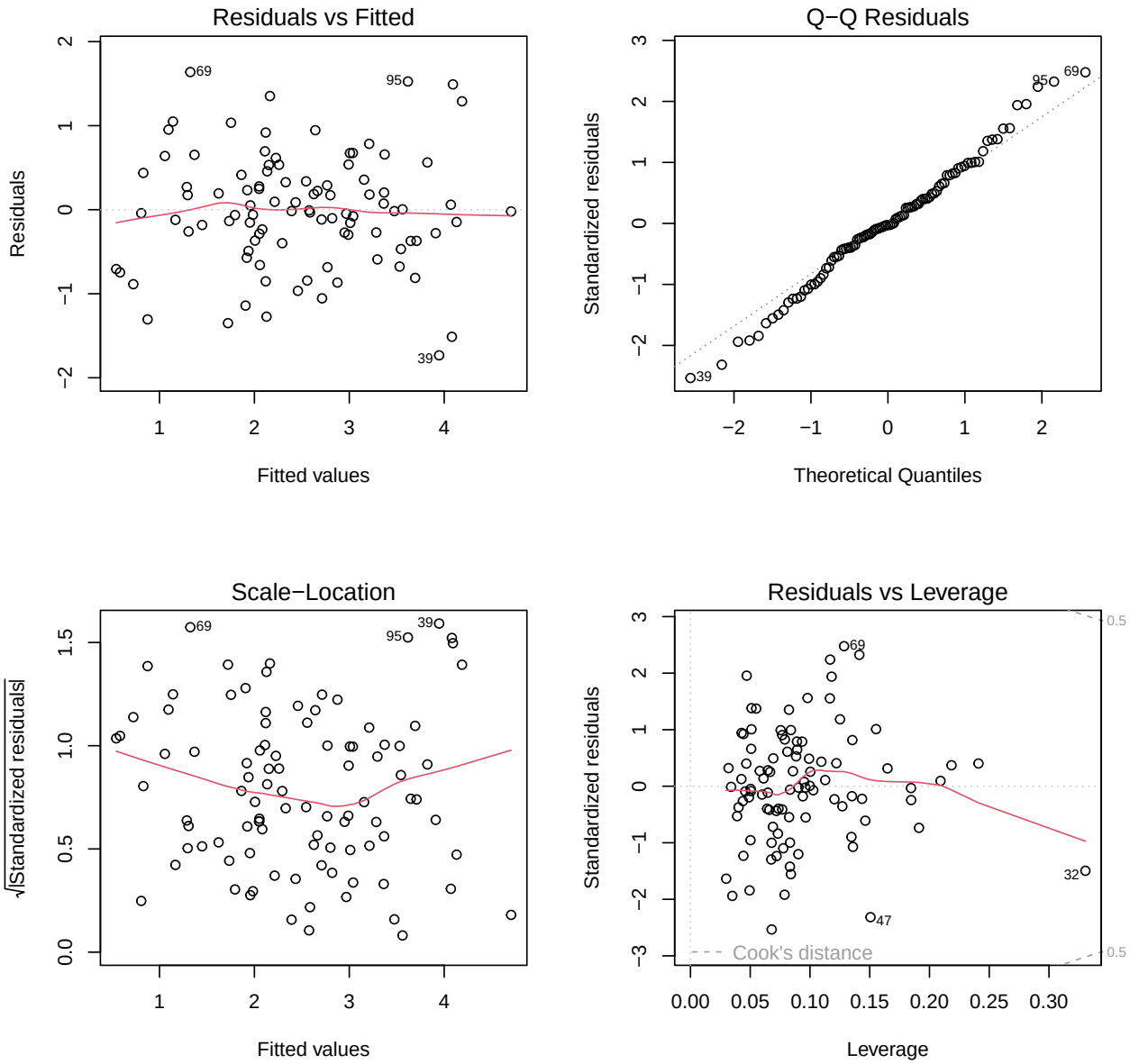
```
##     39     47     69     95     97
##  1.733  1.512 -1.638 -1.526 -1.491
```

Model without influential observations.

```
predict(prostate_add_cd, newdata = removed) - removed$lpsa
```

```
##     39     47     69     95     97
##  1.684  1.715 -1.623 -1.816 -1.678
```

```
par(mfrow = c(2, 2))
plot(prostate_add)
```

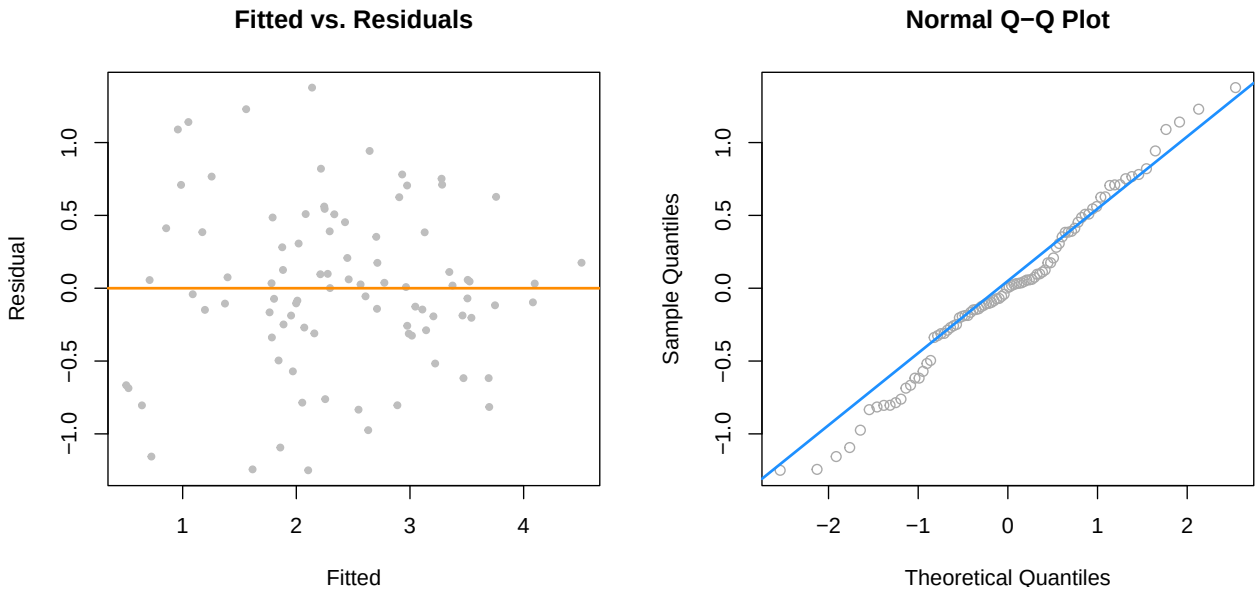



```
par(mfrow=c(1,2))
plot(fitted(prostate_add_cd), resid(prostate_add_cd), col = "grey", pch = 20,
     xlab = "Fitted", ylab = "Residual",
     main = "Fitted vs. Residuals")
abline(h = 0, col = "darkorange", lwd = 2)

bptest(prostate_add_cd)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  prostate_add_cd
## BP = 18, df = 8, p-value = 0.02
```

```r
qqnorm(resid(prostate_add_cd), col = "darkgrey")
qqline(resid(prostate_add_cd), col = "dodgerblue", lwd = 2)
```




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

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(prostate_add_cd)
## W = 0.99, p-value = 0.4
```

For some observations, the residuals slightly increased for the model without influential observations while for some, the residuals are decreased slightly. Overall, the estimated coefficients do not appear substantially different between the models. The influential values might not have significant influence on the model's overall performance. It is possible that rather than removing these found unusual observations, transformation or using different models might be more helpful.

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.

```r
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 = 19870503
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:

```
set.seed(birthday)
for (i in 1:num_sims) {
  y_1 = 4 + 1 * x_1 + 0 * x_2 + rnorm(n = 50, 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] = bptest(fit_1)$p.value
  p_val_2[i] = bptest(fit_2)$p.value
}
print(mean(p_val_1))
```

```
## [1] 0.4855
```

```
print(mean(p_val_2))
```

```
## [1] 0.428
```

(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:

```
result_1 = c(sum(p_val_1 < 0.01) / num_sims,
             sum(p_val_1 < 0.05) / num_sims,
             sum(p_val_1 < 0.10) / num_sims
            )
```

```
result_2 = c(sum(p_val_2 < 0.01) / num_sims,
             sum(p_val_2 < 0.05) / num_sims,
             sum(p_val_2 < 0.10) / num_sims
            )
```

```
result = data.frame(Levels = c("< 0.01", "< 0.05", "< 0.10"),
                    p_val_1 = result_1,
                    p_val_2 = result_2)
knitr::kable(result, align = "c")
```

Levels	p_val_1	p_val_2
< 0.01	0.0052	0.0116
< 0.05	0.0444	0.0688
< 0.10	0.1040	0.1476

From the table of results, we can see that as the level of threshold increase, the proportion that is smaller than the level increases. It is clear that the model fit_1 with constant variance has very small proportion

that is smaller than level 0.01. In comparison, larger proportion of of p_val_2 are smaller than each levels compared to p_val_1. As expected the y_2 has higher probability that it will violate constant variance assumption as the data was simulated using non-constant variance.

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.

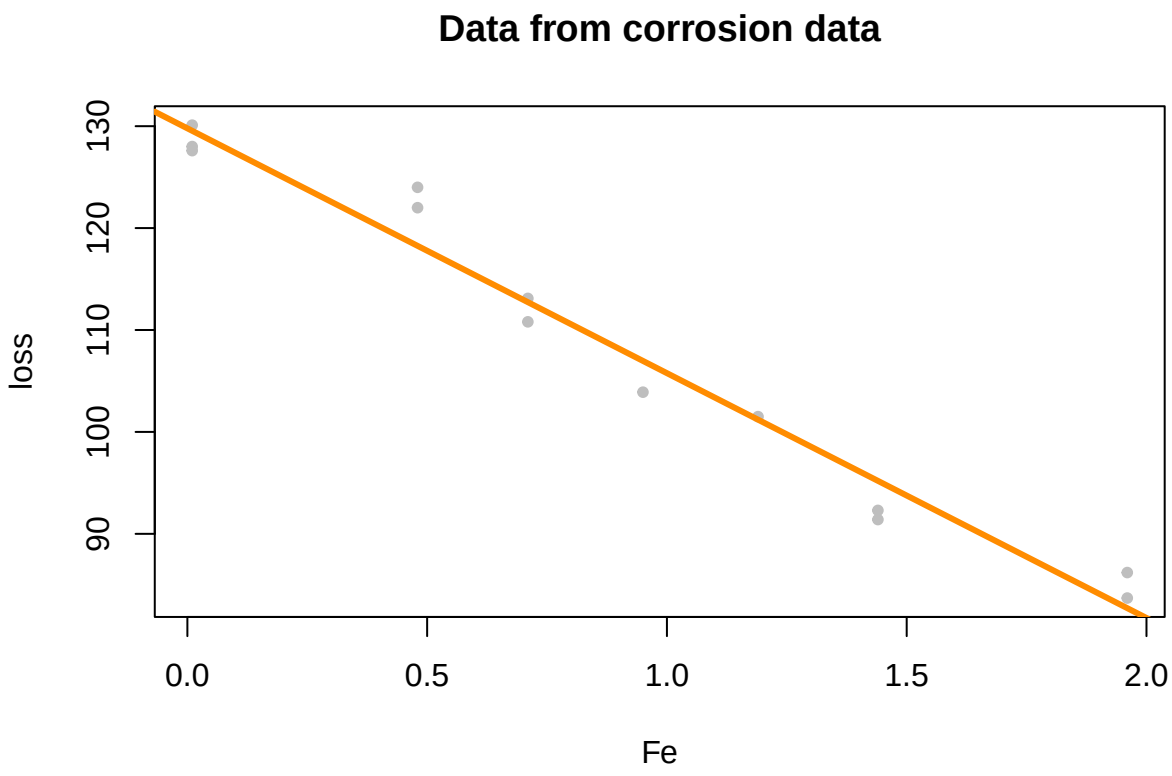
Solution:

We fit a simple linear regression model.

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

We plot a scatterplot and add the fitted line above.

```
plot(loss ~ Fe, data = corrosion, col = "grey", pch = 20,
     main = "Data from corrosion data")
abline(corrosion_model, col = "darkorange", lwd = 3)
```



Now we check the model assumptions are not violated.

1. Linearity and constant Variance

```
bptest(corrosion_model)
```

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

- We see a large p-value, so we fail to reject the null hypothesis and constant variance is not suspect.

```
shapiro.test(resid(corrosion_model))
```

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

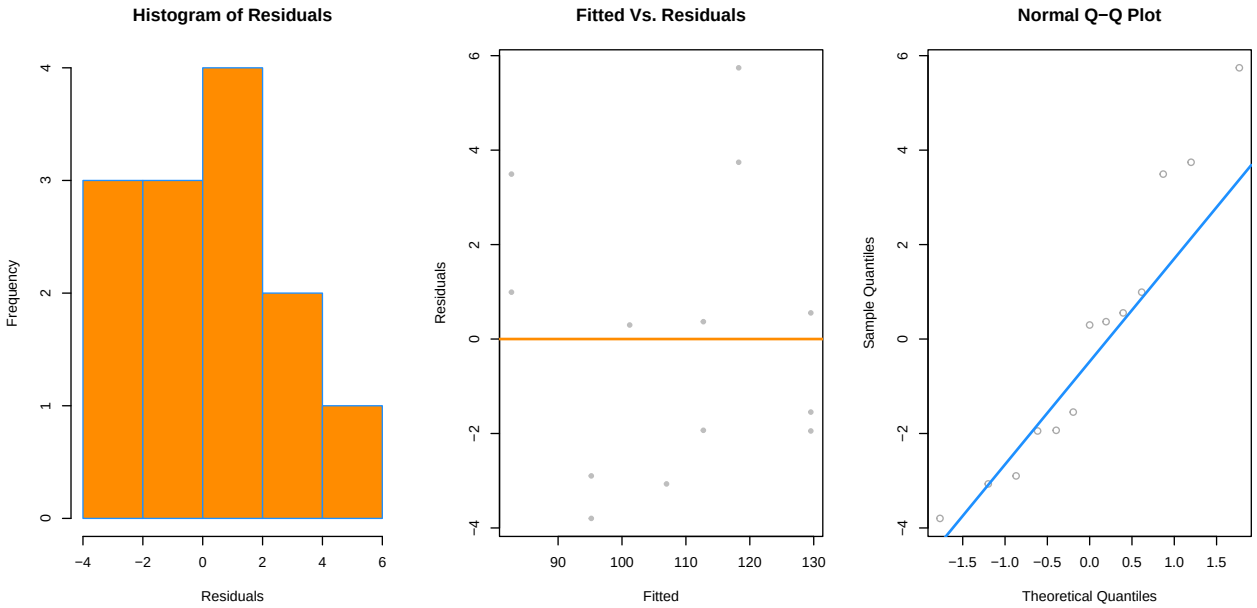
- We see a large p-value for shapiro-wilk test. We fail to reject the null hypothesis and normality is not suspect.

```
par(mfrow = c(1, 3))

hist(resid(corrosion_model),
     xlab   = "Residuals",
     main   = "Histogram of Residuals",
     col    = "darkorange",
     border = "dodgerblue",
     breaks = 6)

plot(fitted(corrosion_model), resid(corrosion_model), col = "grey", pch = 20,
     xlab = "Fitted", ylab = "Residuals", main = "Fitted Vs. Residuals")
abline(h = 0, col = "darkorange", lwd = 2)

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



- From the histogram, it is not very clear that the errors are normally distributed. - Looking at fitted vs. residuals plot, we can see that for any fitted value, the residuals seem roughly centered at 0 and no clear pattern is visible. Therefore, linearity is not violated.

(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:

```
fit_poly_2 = lm(loss ~ poly(Fe, degree = 2, raw = TRUE), data = corrosion)
fit_poly_3 = lm(loss ~ poly(Fe, degree = 3, raw = TRUE), data = corrosion)
fit_poly_4 = lm(loss ~ poly(Fe, degree = 4, raw = TRUE), data = corrosion)
```

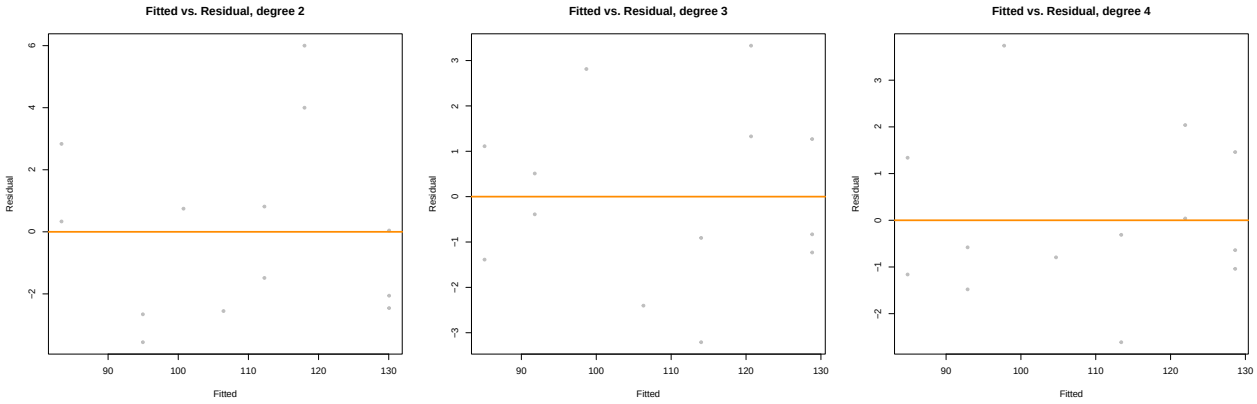
```
par(mfrow = c(1,3))

plot(fitted(fit_poly_2), resid(fit_poly_2), col = "grey", pch = 20,
     xlab = "Fitted",
     ylab = "Residual",
     main = "Fitted vs. Residual, degree 2"
     )
abline(h = 0, col = "darkorange", lwd = 2)

plot(fitted(fit_poly_3), resid(fit_poly_3), col = "grey", pch = 20,
     xlab = "Fitted",
     ylab = "Residual",
     main = "Fitted vs. Residual, degree 3"
     )
abline(h = 0, col = "darkorange", lwd = 2)

plot(fitted(fit_poly_4), resid(fit_poly_4), col = "grey", pch = 20,
     xlab = "Fitted",
     ylab = "Residual",
     main = "Fitted vs. Residual, degree 4")
```

```r
abline(h = 0, col = "darkorange", lwd = 2)
```



Based on the plots, out of above three, degree 3 and 4 polynomial are acceptable. Degree 2 possibly violating normality assumption.

- Statistical test(s) to compare the models.

```r
anova(fit_poly_3, fit_poly_4)
```

```
## Analysis of Variance Table
##
## Model 1: loss ~ poly(Fe, degree = 3, raw = TRUE)
## Model 2: loss ~ poly(Fe, degree = 4, raw = TRUE)
##   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
```

Here we see large p-value. We prefer fit_poly_3 model.

```r
shapiro_3 = shapiro.test(resid(fit_poly_3))
print(shapiro_3)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(fit_poly_3)
## W = 0.97, p-value = 0.9
```

The p-value from bptest is high enough to reject H0 and constant variance is not suspect.

To identify influential observations, we carry out cooks distance test

```r
(influential_pt_3 = cooks.distance(fit_poly_3) > 4 / length(cooks.distance(fit_poly_3)))
```

```
##     1     2     3     4     5     6     7     8     9    10    11    12    13
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

Based on the cooks distance values, we do not see influential observations of this model.

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)
```

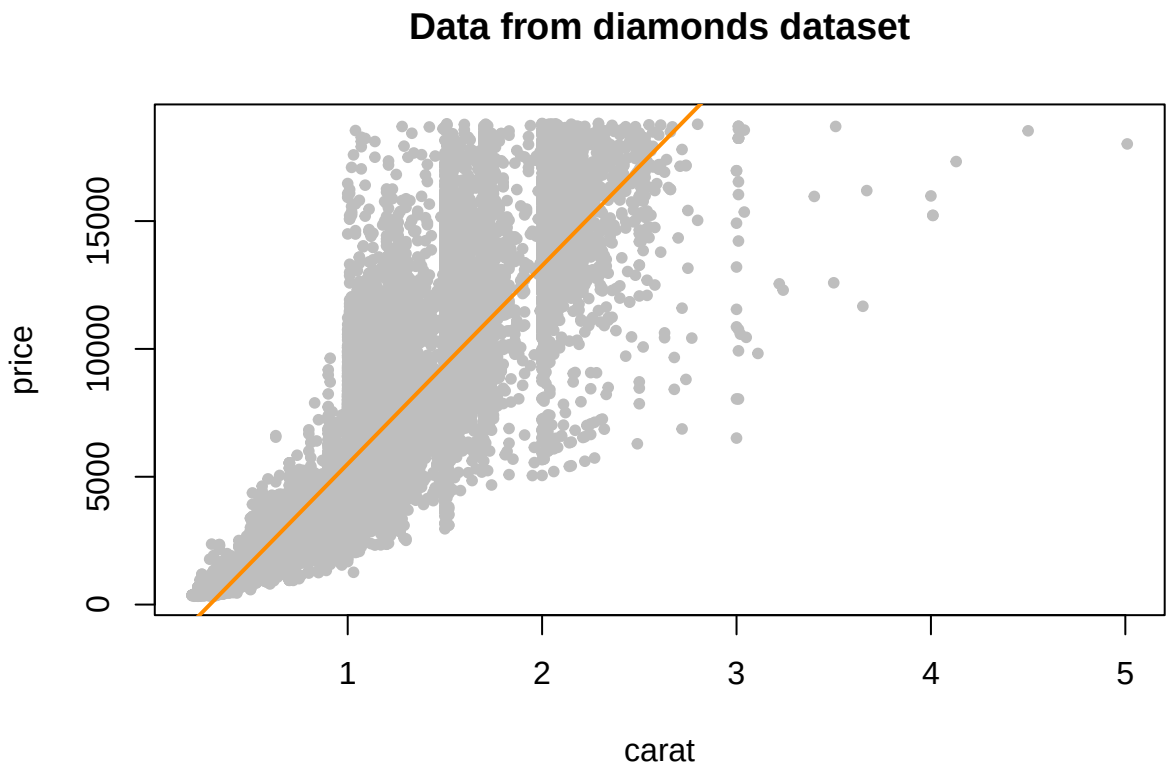
(a) Fit a linear model with `price` as the response variable y, and `carat` as the predictor x. Return the summary information of this model.

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

```
##
## 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
```

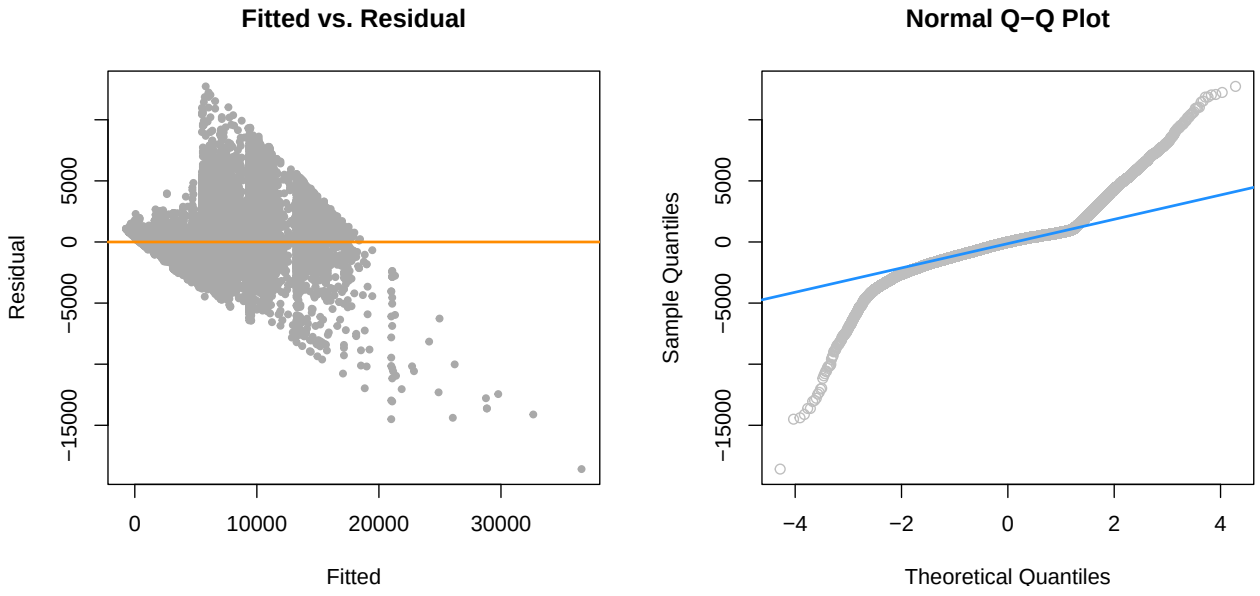
(b) Plot a scatterplot of price versus carat and add the line for the fitted model in part **(a)**. Using a fitted versus residuals plot and/or a Q-Q plot, comment on the diagnostics.

```
plot(price ~ carat, data = diamonds, col = "grey", pch = 20,
     main = "Data from diamonds dataset")
abline(dia_model, col = "darkorange", lwd = 2)
```




```
par(mfrow = c(1,2))
plot(fitted(dia_model), resid(dia_model), col = "darkgrey", pch = 20,
     xlab = "Fitted",
     ylab = "Residual",
     main = "Fitted vs. Residual")
abline(h = 0, col = "darkorange", lwd = 2)

qqnorm(resid(dia_model), col = "grey")
qqline(resid(dia_model), col = "dodgerblue", lwd = 2)
```



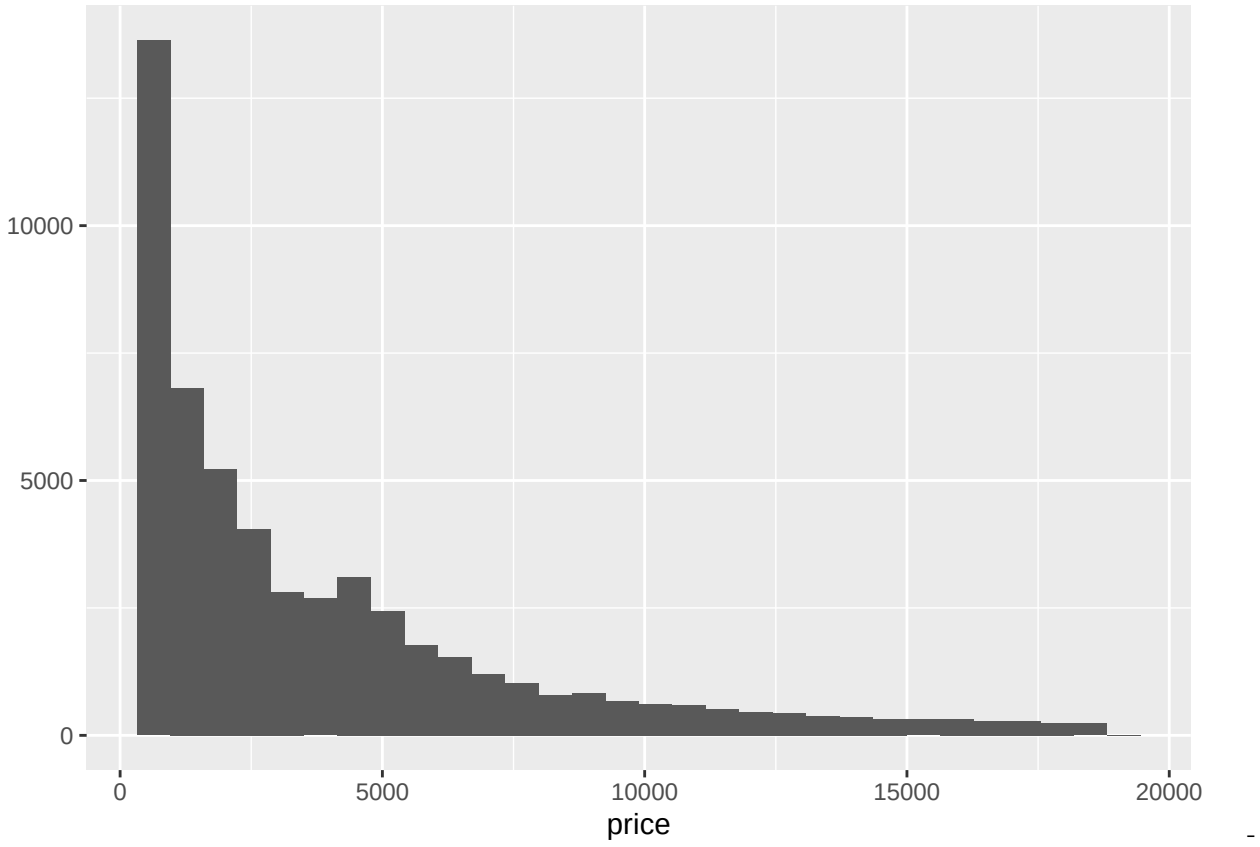
From the fitted vs. residual plot, we see that for each fitted point, residual is not centered around 0 and

also see that there is visible pattern. This suggests the linearity assumption is violated. Also, the form of Q-Q plot suggest that the normality assumption is violated.

(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.
```

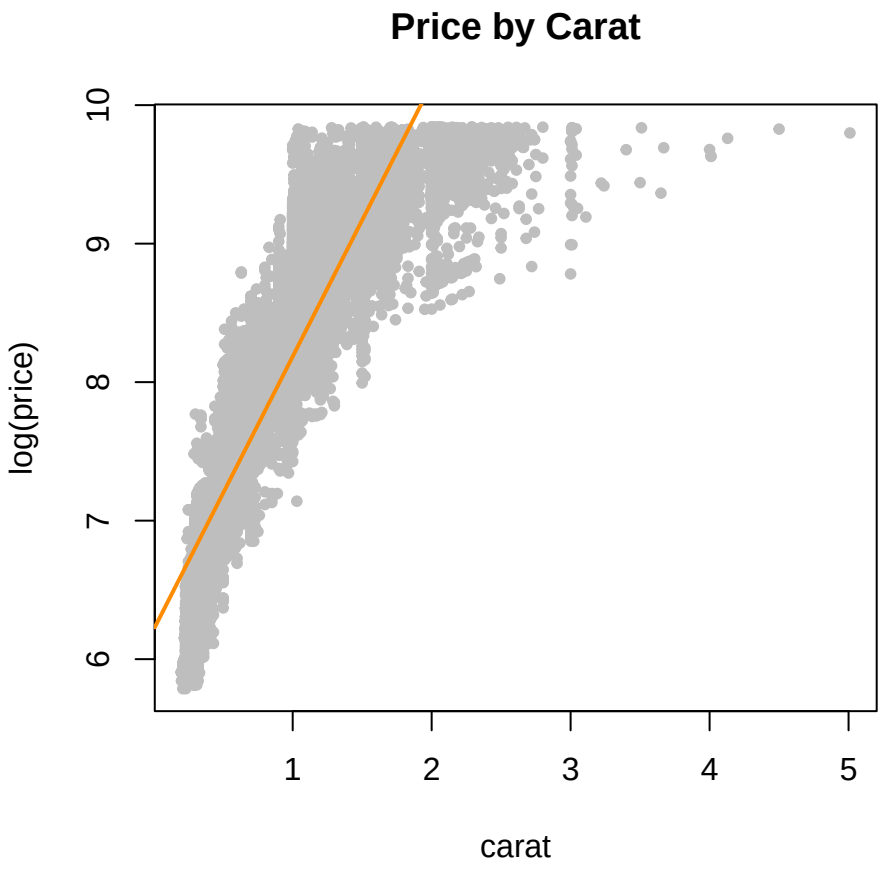


Fit a model with a logged response

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

- scatterplot of log-price versus carat and add the line for the fitted model

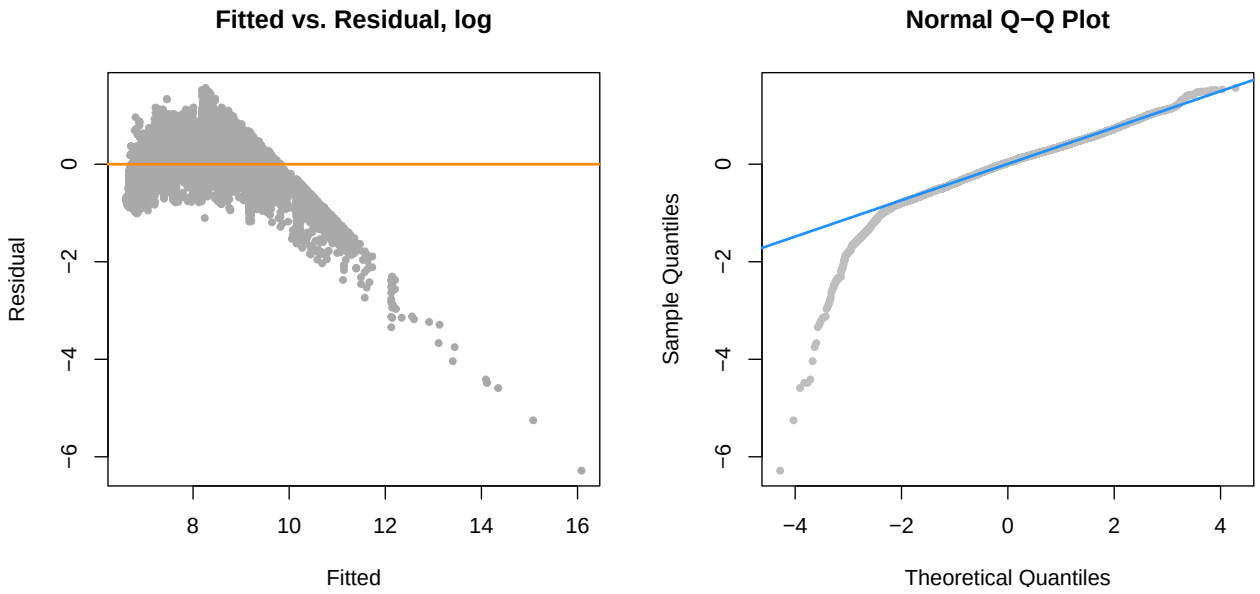
```
plot(log(price) ~ carat, data = diamonds, col = "grey", pch = 20,
     main = "Price by Carat"
     )
abline(dia_model_log, col = "darkorange", lwd = 2)
```



- use a fitted versus residuals plot and/or a Q-Q plot to comment on the diagnostics of the model

```
par(mfrow = c(1,2))
plot(fitted(dia_model_log), resid(dia_model_log), col = "darkgrey", pch = 20,
     xlab = "Fitted",
     ylab = "Residual",
     main = "Fitted vs. Residual, log")
abline(h = 0, col = "darkorange", lwd = 2)

qqnorm(resid(dia_model_log), col = "grey", pch = 20)
qqline(resid(dia_model_log), col = "dodgerblue", lwd = 2)
```



From fitted vs residual plot, we still see that there is a clear form to the plot and non-constant variance issue. Also Q-Q plot also suggests that the normality assumption is violated. These suggest that we need a different 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.
- Fit a model with a logged response and logged predictor

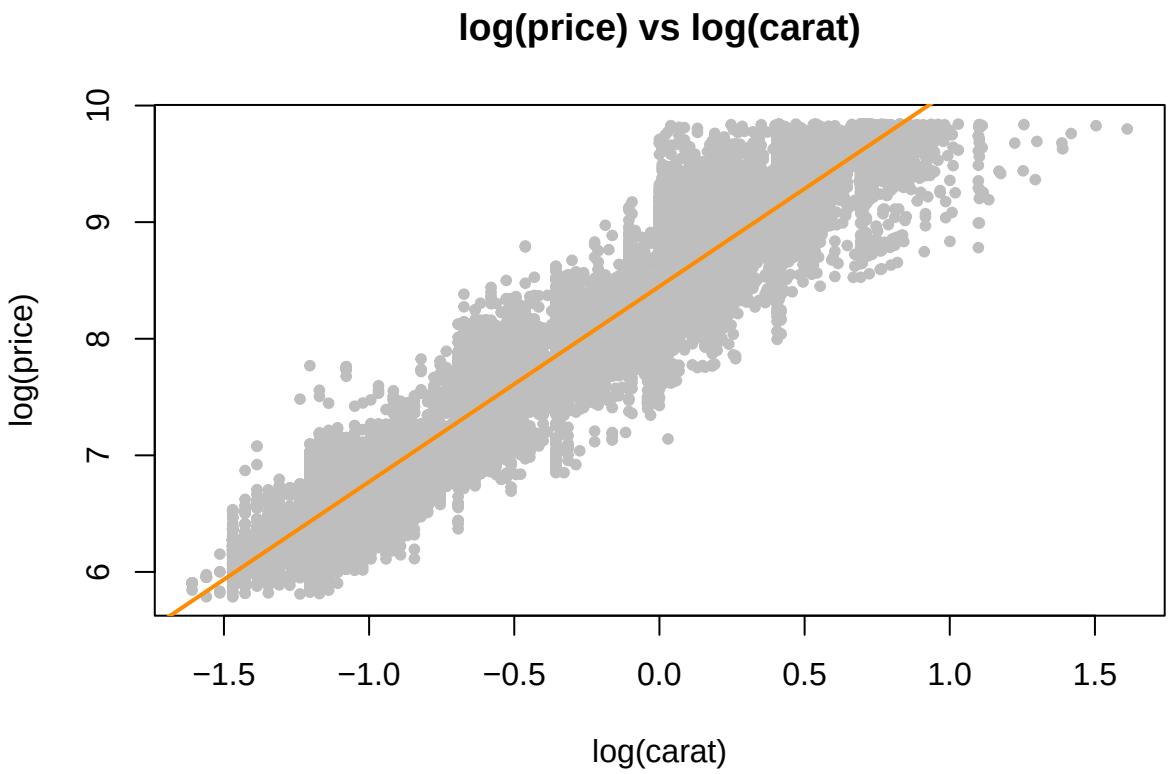
```
dia_model_log_2_s = lm(log(price) ~ log(carat), data = diamonds)
summary(dia_model_log_2_s)
```

```
##
## Call:
## lm(formula = log(price) ~ log(carat), data = diamonds)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5083 -0.1695 -0.0059  0.1664  1.3379
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.44866    0.00136    6191   <2e-16 ***
## log(carat)   1.67582    0.00193     867   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.263 on 53938 degrees of freedom
## Multiple R-squared:  0.933,  Adjusted R-squared:  0.933
## F-statistic: 7.51e+05 on 1 and 53938 DF,  p-value: <2e-16
```

```
dia_model_log_2 = lm(log_price ~ log_carat, data = diamonds)
summary(dia_model_log_2)
```

- plot a scatterplot of log-price versus log-carat and add the line for the fitted model

```r
plot(log(price) ~ log(carat), data = diamonds, col = "grey", pch = 20,
     main = "log(price) vs log(carat)")
abline(dia_model_log_2_s, col = "darkorange", lwd = 2)
```



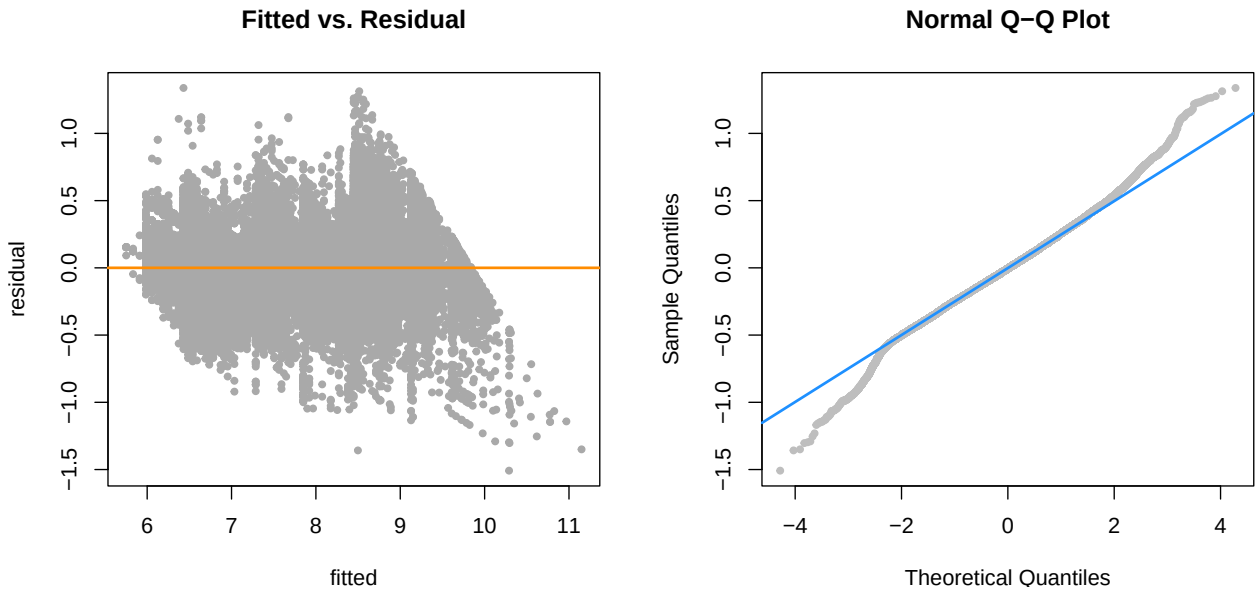
The fitted line seems tom explian the linear relationship between log-price and log-carat well.

- use a fitted versus residuals plot and/or a Q-Q plot

```r
par(mfrow = c(1, 2))

plot(fitted(dia_model_log_2_s), resid(dia_model_log_2_s), col = "darkgrey",
    pch = 20,
    xlab = "fitted",
    ylab = "residual",
    main = "Fitted vs. Residual",
    )
abline(h = 0, col = "darkorange", lwd = 2)

qqnorm(resid(dia_model_log_2_s), col = "grey", pch = 20)
qqline(resid(dia_model_log_2_s), col = "dodgerblue", lwd = 2)
```



These plots look better compared to previous linear or response-log transformed models. However, constant variance seems to be violated looking at fitted vs residual plot especially looking at the right side of the plot starting from fitted value 9. Also, we have suspect Q-Q plot. We probably would not believe the errors 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_carat = 3
newdata = data.frame(carat = new_carat)
prediction =  predict(dia_model_log_2_s, newdata = newdata, interval = "prediction", level = 0.99)
exp(prediction)
```

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

Solution:
99% prediction interval for the price(in dollars): [14959.4438 , 57894.0035]