STAT****/STAT****: Linear Models

Assignment 2: Cushion Diamonds Data

****

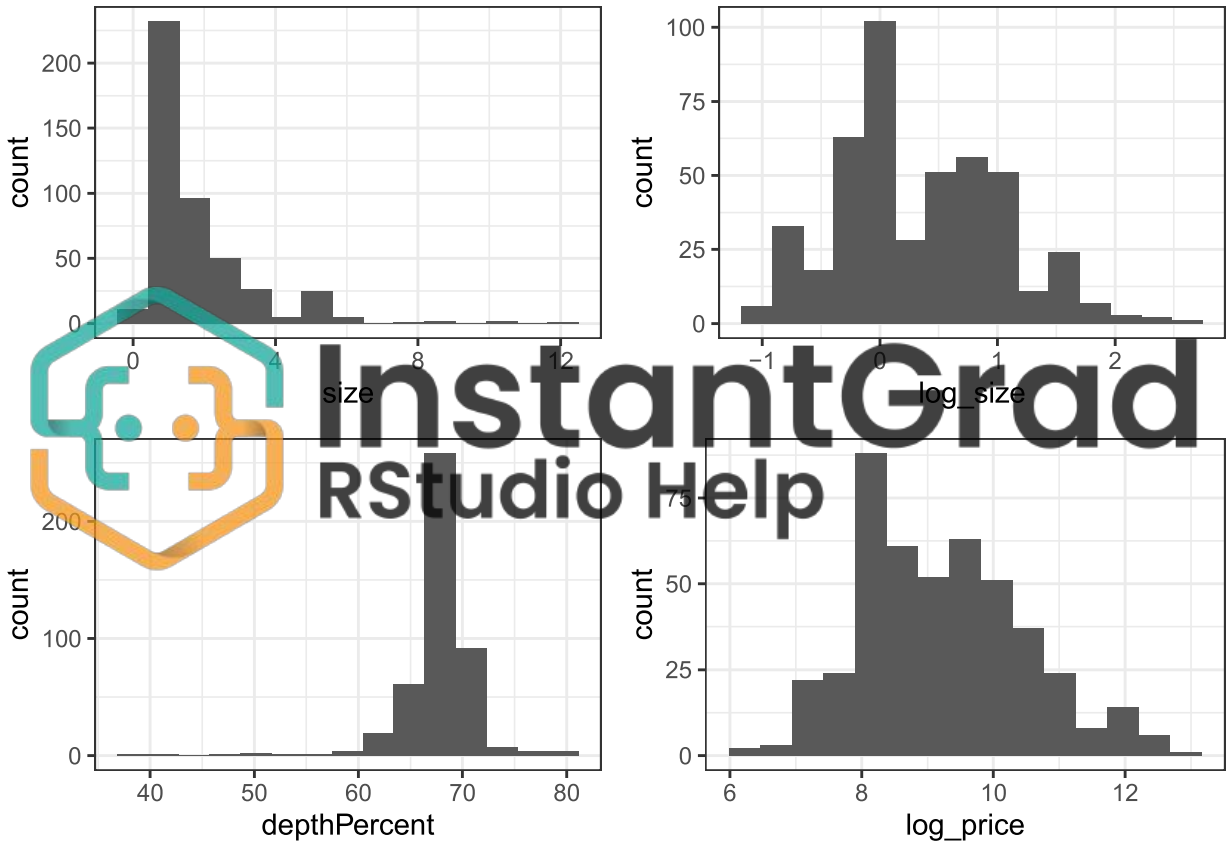
Due date: **-**-****

InstantGrad

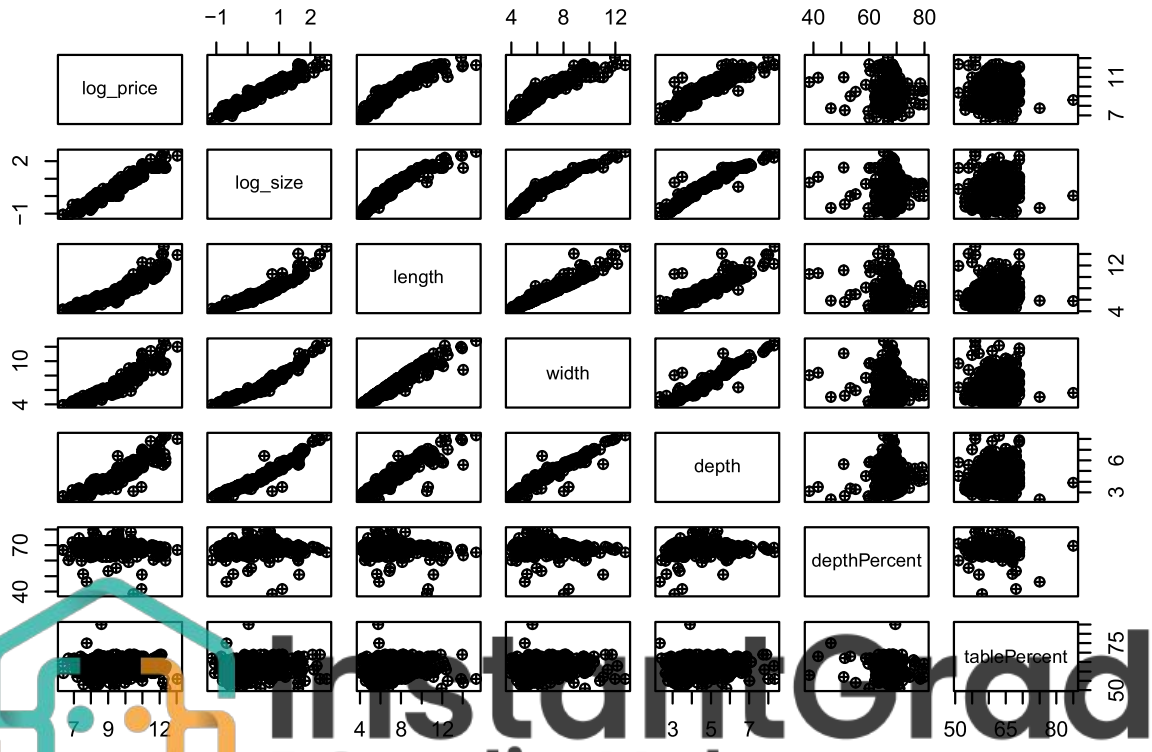
Appendix 2

```
# Create two new variables (same as Appendix 1)
diamonds$log_price = log(diamonds$price)
diamonds$log_size = log(diamonds$size)

# Plot transformation
p8 <- ggplot(diamonds , aes(x=log_price)) + geom_histogram(bins = bins_amount)+theme_bw()
p9 <- ggplot(diamonds , aes(x=log_size)) + geom_histogram(bins = bins_amount)+theme_bw()
grid.arrange(p1,p9,p2,p8,nrow=2)
```

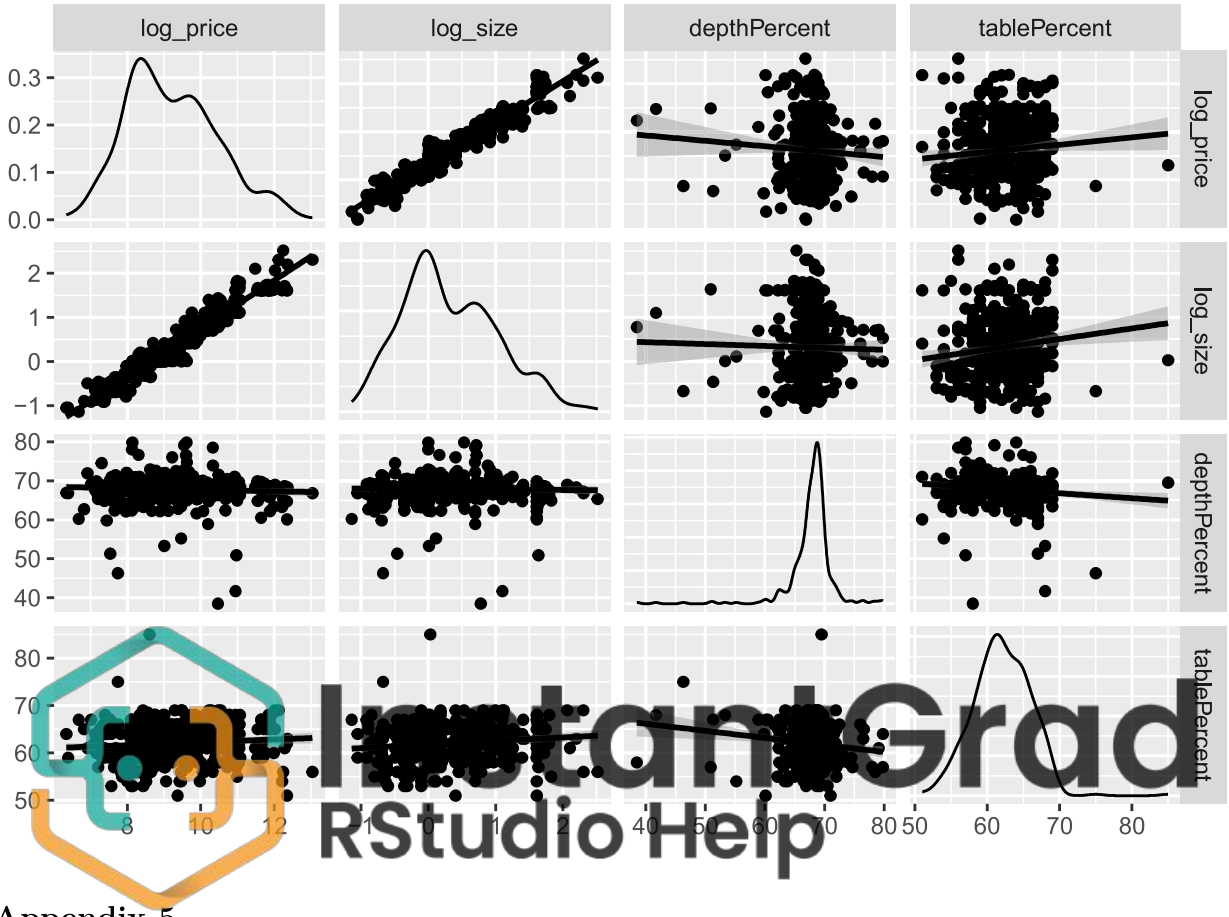



```
# Scatter Matrix
scat_matix <- diamonds %>% select(log_price,log_size,length,width,depth,depthPercent,tablePercent)
pairs(scat_matix, pch = 10)
```



Appendix 4

```
# Scatter Matrix with Loess
ggpairs(scat_matix, columns=c("log_price","log_size", "depthPercent", "tablePercent"),
        upper=list(continuous="smooth"),lower=list(continuous="smooth"))
```



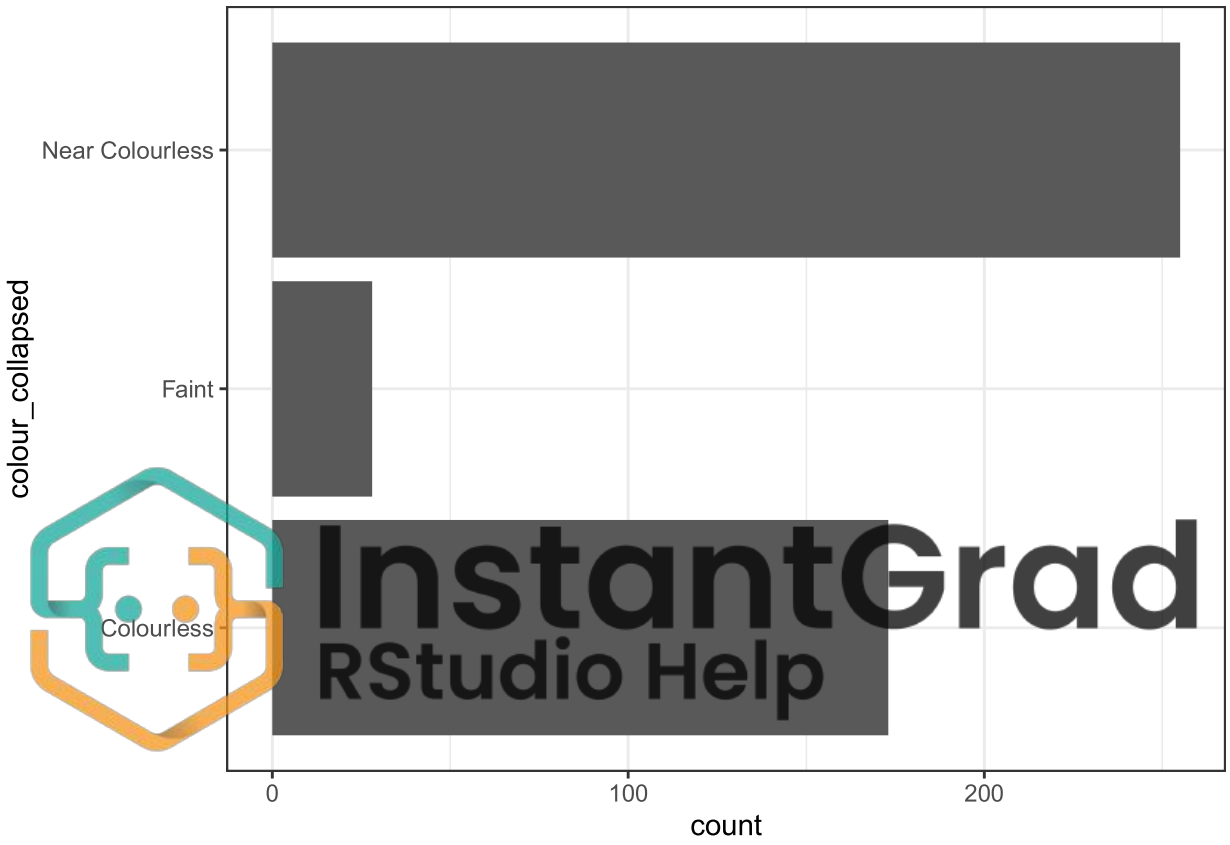
Appendix 5

```
# Creating colour_collapsed
diamonds <- diamonds %>%
  mutate(colour_collapsed = case_when(
    colour == "D" | colour == "E" | colour == "F" ~ "Colourless",
    colour == "G" | colour == "H" | colour == "I" | colour == "J" ~ "Near Colourless",
    colour == "K" | colour == "L" | colour == "M" ~ "Faint",
    colour == "N" | colour == "O" | colour == "P" | colour == "Q" | colour == "R" ~ "Very Light",
    TRUE ~ "Light"))

# Print colour_collapsed
unique(diamonds %>% select(colour, colour_collapsed) %>% arrange(colour))
```

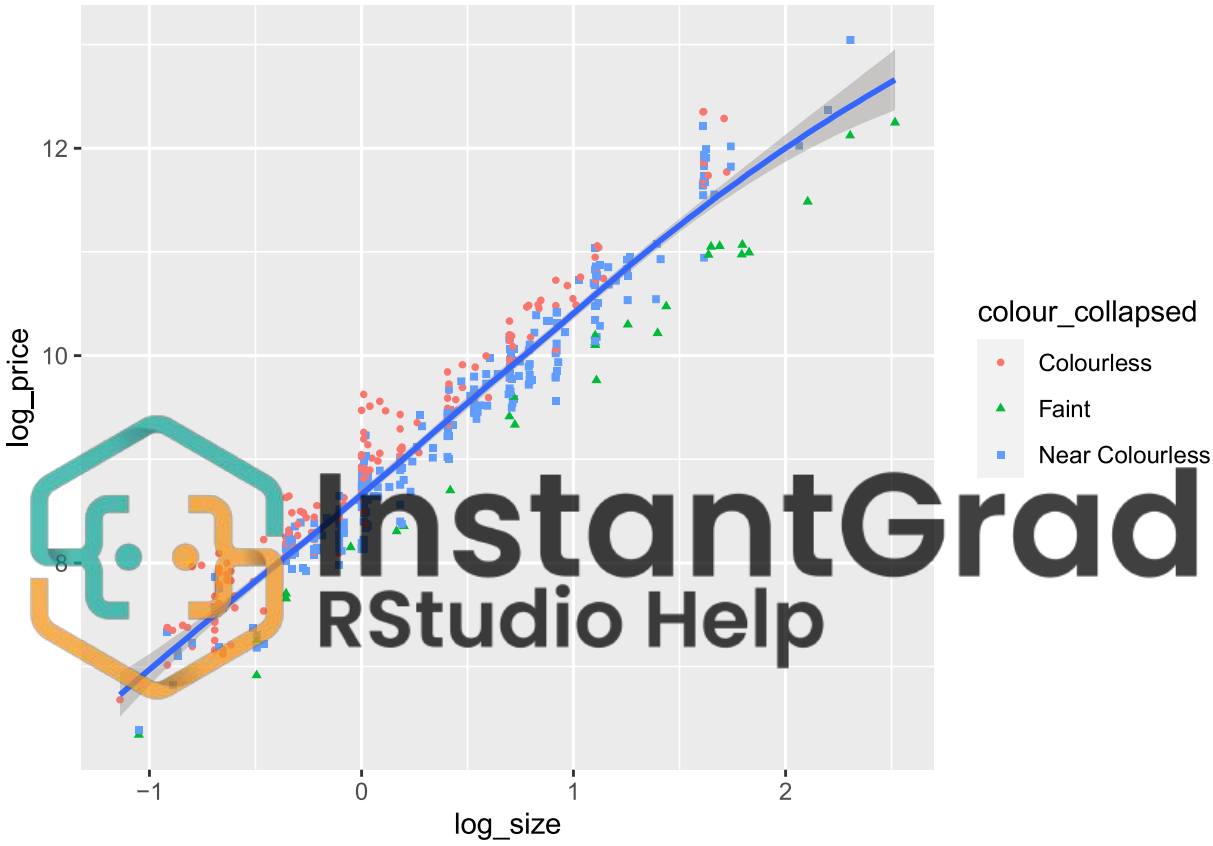
```
##     colour colour_collapsed
## 1        D       Colourless
## 51       E       Colourless
## 111      F       Colourless
## 174      G  Near Colourless
## 254      H  Near Colourless
## 327      I  Near Colourless
## 385      J  Near Colourless
## 429      K            Faint
## 448      L            Faint
## 454      M            Faint
```

```r
# Plot colour_collapsed
ggplot(diamonds %>% group_by(colour_collapsed), aes( y = colour_collapsed)) +
  geom_bar(bins = bins_amount)+
  theme_bw()
```



Appendix 6

```
# Plot log_size and coulour_collapsed
ggplot(diamonds, aes(x=log_size, y=log_price)) +
  geom_point(aes(shape = colour_collapsed, color = colour_collapsed), size = 1)+
  geom_smooth()
```

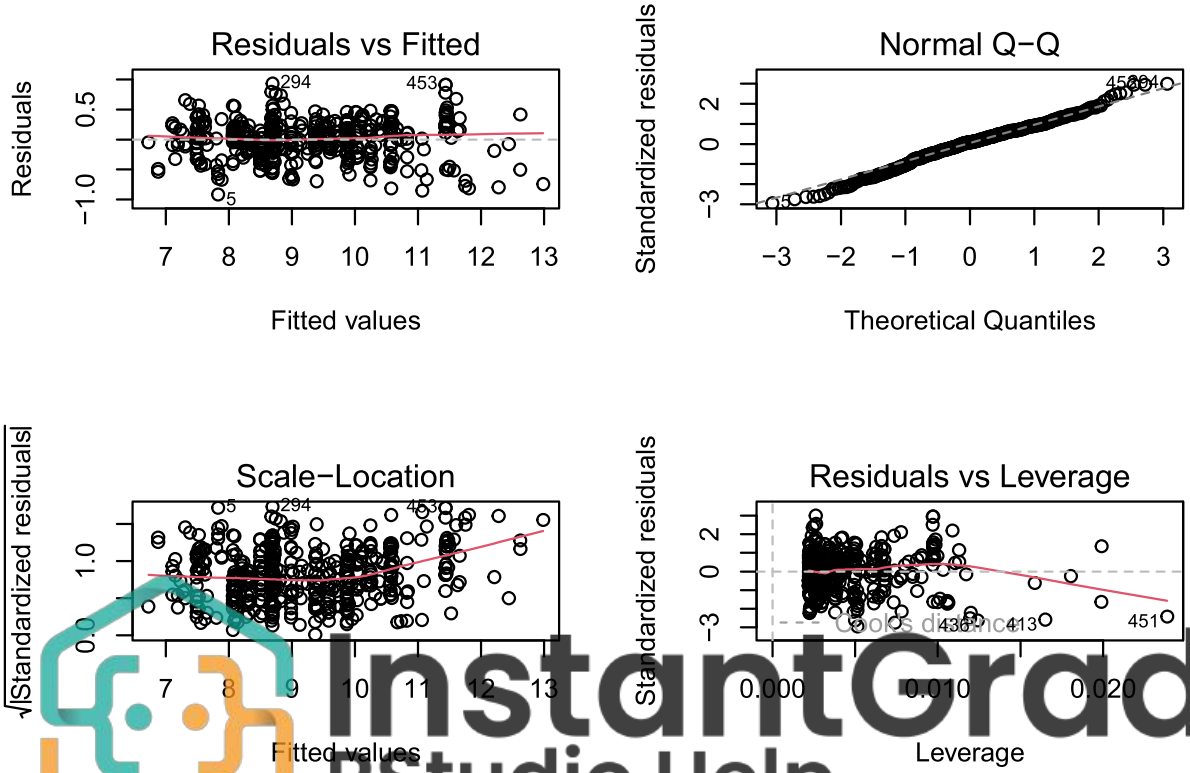


Appendix 7

```
# Run and summary LM
lm1 <- lm(log_price ~ log_size, data = diamonds)
summary(lm1)
```

```
##
## Call:
## lm(formula = log_price ~ log_size, data = diamonds)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.91536 -0.17512  0.02699  0.20391  0.93088
##
## Coefficients:
##             Estimate Std. Error t value            Pr(>|t|)
## (Intercept)  8.67783    0.01599  542.71 <0.0000000000000002 ***
## log_size     1.71303    0.02082   82.29 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3109 on 454 degrees of freedom
## Multiple R-squared:  0.9372, Adjusted R-squared:  0.937
## F-statistic:  6772 on 1 and 454 DF,  p-value: < 0.00000000000000022
```

```
# Plot results
par(mfrow=c(2,2))
plot(lm1)
```




```
# Breusch-Pagan test
bptest(lm1)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  lm1
## BP = 17.56, df = 1, p-value = 0.00002784
```

Appendix 8

```
# Run and summary LM
lm2 <- lm(log_price ~ log_size + colour_collapsed, data = diamonds)
summary(lm2)
```

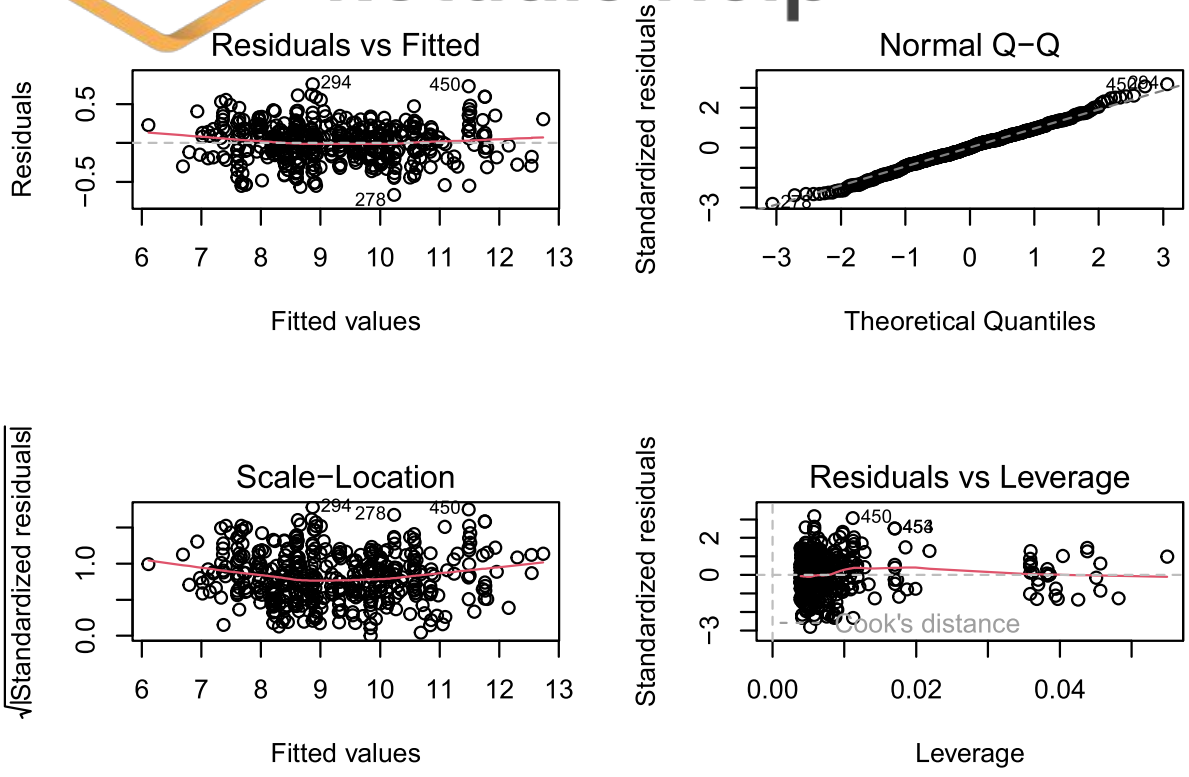
```
##
## Call:
## lm(formula = log_price ~ log_size + colour_collapsed, data = diamonds)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.66864 -0.14969  0.00023  0.15543  0.75596
##
## Coefficients:
##                                Estimate Std. Error t value            Pr(>|t|)
## (Intercept)                     8.85187    0.01824  485.25 <0.0000000000000002
```

```
## log_size                           1.80218    0.01677  107.49 <0.0000000000000002
## colour_collapsedFaint             -0.84720    0.05058  -16.75 <0.0000000000000002
## colour_collapsedNear Colourless -0.26884    0.02404  -11.18 <0.0000000000000002
##
## (Intercept)                       ***
## log_size                          ***
## colour_collapsedFaint             ***
## colour_collapsedNear Colourless ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2389 on 452 degrees of freedom
## Multiple R-squared:  0.9631, Adjusted R-squared:  0.9628
## F-statistic:  3929 on 3 and 452 DF,  p-value: < 0.00000000000000022
```

```
# 95% confidence interval
confint(lm2)
```

```
##                                    2.5 %     97.5 %
## (Intercept)                     8.8160173  8.8877159
## log_size                        1.7692329  1.8351300
## colour_collapsedFaint          -0.9466092 -0.7477951
## colour_collapsedNear Colourless -0.3160928 -0.2215957
```

```
par(mfrow=c(2,2))
plot(lm2)
```

```r
# Breusch-Pagan test
bptest(lm2)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  lm2
## BP = 1.3463, df = 3, p-value = 0.7182
```

Appendix 9

```
# Count rows
n <- nrow(diamonds)
index<-as_tibble(c(1:456))

# Calculate leverage for each observation under model fitted
colnames(index)<-c('Observation')
lev <- as_tibble(hatvalues(lm2))

# Calculate DFBetas for each observation under model fitted
colnames(lev) <- c('Leverage')
dfbetas <- as_tibble(dfbetas(lm2))

# Calculate Cook's distance for each observation under model fitted
colnames(dfbetas) <- c('DFBeta0', 'DFBeta1', 'DFBeta2', 'DFBeta3')
cooksD <- as_tibble(cooks.distance(lm2))
colnames(cooksD)<-c('CooksD')
tibble(lev,dfbetas, cooksD)
```

```
## # A tibble: 456 x 6
##    Leverage  DFBeta0 DFBeta1  DFBeta2 DFBeta3   CooksD
##       <dbl>    <dbl>   <dbl>    <dbl>   <dbl>    <dbl>
##  1  0.0550   0.0132  -0.141   0.210    0.0290 0.0142
##  2  0.0142  -0.0122   0.130  -0.0355  -0.0765 0.00581
##  3  0.0134  -0.0422   0.0439  0.00176  0.0199 0.000850
##  4  0.0121  -0.00547  0.0582 -0.0159  -0.0370 0.00125
##  5  0.0457  -0.00814  0.0866 -0.170   -0.0178 0.00862
##  6  0.0109  -0.0644   0.0559  0.00607  0.0334 0.00167
##  7  0.0118   0.00290 -0.0308  0.00842  0.0199 0.000356
##  8  0.00860 -0.189    0.124   0.0300   0.109  0.0117
##  9  0.0118   0.00454 -0.0484  0.0132   0.0312 0.000877
## 10  0.00784 -0.0127   0.135  -0.0368  -0.112  0.00901
## # ... with 446 more rows
```

```
# Save diagnostics
diagnostics<-tibble(index,lev,dfbetas,cooksD)


# Treshold
thresh <- 2/sqrt(456)
p_plus_1<- 4
thresh_leverage<-2*p_plus_1/456

# Graphs are in order of the questions
# Construct Plot
p1<-ggplot(diagnostics, aes(x = Observation,
                            ymax=pmax(0, Leverage), ymin = pmin(0, Leverage))) +
geom_linerange()+theme_bw() +
geom_hline(yintercept=thresh_leverage, col="red", linetype="dashed") +
labs(y="Leverage")

p2<-ggplot(diagnostics, aes(x = Observation, y=Leverage,
                            ymax=pmax(0, DFBeta0), ymin = pmin(0, DFBeta0))) +
geom_linerange()+theme_bw() +
```

```
geom_hline(yintercept=thresh, col="red", linetype="dashed") +
geom_hline(yintercept=-thresh, col="red", linetype="dashed") +
labs(y="DFBeta0")

p3<- ggplot(diagnostics, aes(x = Observation, ymax=pmax(0, DFBeta1),
                             ymin = pmin(0, DFBeta1))) +
geom_linerange()+theme_bw() +
geom_hline(yintercept=thresh, col="red", linetype="dashed") +
geom_hline(yintercept=-thresh, col="red", linetype="dashed") +
labs(y="DFBeta1")

p4<- ggplot(diagnostics, aes(x = Observation, ymax=pmax(0, DFBeta2),
                             ymin = pmin(0, DFBeta1))) +
geom_linerange()+theme_bw() +
geom_hline(yintercept=thresh, col="red", linetype="dashed") +
geom_hline(yintercept=-thresh, col="red", linetype="dashed") +
labs(y="DFBeta2")

p5<- ggplot(diagnostics, aes(x = Observation, ymax=pmax(0, DFBeta3),
                             ymin = pmin(0, DFBeta1))) +
geom_linerange()+theme_bw() +
geom_hline(yintercept=thresh, col="red", linetype="dashed") +
geom_hline(yintercept=-thresh, col="red", linetype="dashed") +
labs(y="DFBeta3")

p6<-ggplot(diagnostics, aes(x = Observation, ymax=pmax(0, CooksD),
                            ymin = pmin(0, CooksD))) +
geom_linerange()+
geom_hline(yintercept=qf(0.5,p_plus_1,n-p_plus_1),
           col="steelblue", linetype="dashed") +
labs(y="Cook's D")

grid.arrange(p1,p2,p3,p4,p5,p6,nrow=3)
```