Lab 10

STAT 131

Welcome to the lab 10! Today, we will use linear regression to predict the red wine quality using physicochemical tests scores such as citric acid, pH, etc.

But first, a demonstration of using the predict() function.

```
x1 = rnorm(100)
x2 = rnorm(100)
y= 2*x1 + x2 + rnorm(100)
lm_out = lm(y~x1 + x2)
summary(lm_out)
```

```
##
## Call:
## lm(formula = y ~ x1 + x2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.69140 -0.57704 -0.03997  0.60543  2.18549
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.16273    0.09415  -1.728   0.0871 .
## x1           1.78170    0.10679  16.684   <2e-16 ***
## x2           1.07747    0.09766  11.032   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9357 on 97 degrees of freedom
## Multiple R-squared:  0.7935, Adjusted R-squared:  0.7892
## F-statistic: 186.3 on 2 and 97 DF,  p-value: < 2.2e-16
```

Calculate prediction for y when x1 is 1 and x2 is 0.5.

```
lm_out$coefficients[1] + lm_out$coefficients[2]* 1 + lm_out$coefficients[3]* 0.5
```

```
## (Intercept)
##    2.157709
```

Another way to do this.

```
predict(lm_out, newdata = data.frame(x1= 1, x2= 0.5))
```

```
##        1
## 2.157709
```

Can do several at once.

```
predict(lm_out, newdata = data.frame(x1= c(1, 2), x2= c(0.5, -1) ))
```

```
##        1        2
## 2.157709 2.323212
```

We can find intervals for confidence of average or prediction interval for an individual outcome.

```r
predict(lm_out, newdata = data.frame(x1= c(1, 2), x2= c(0.5, -1) ), interval = "confidence")
```

```
##        fit      lwr      upr
## 1 2.157709 1.846496 2.468922
## 2 2.323212 1.816374 2.830050
```

```r
predict(lm_out, newdata = data.frame(x1= c(1, 2), x2= c(0.5, -1) ), interval = "prediction")
```

```
##        fit       lwr      upr
## 1 2.157709 0.2747662 4.040652
## 2 2.323212 0.3982431 4.248181
```

A few other notes on regression.

1) If you try to predict one variable and include a perfectly correlated variable in the prediction set, then that variable will be perfectly fit to the outcome to the exclusion of all others.

```r
perf_cor = y/4
summary(lm( y ~ perf_cor + x1 + x2 ))
```

```
## Warning in summary.lm(lm(y ~ perf_cor + x1 + x2)): essentially perfect fit:
## summary may be unreliable
```

```
##
## Call:
## lm(formula = y ~ perf_cor + x1 + x2)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -4.695e-16 -1.857e-17 -5.080e-18  6.950e-18  4.418e-16
##
## Coefficients:
##              Estimate Std. Error   t value Pr(>|t|)
## (Intercept) 0.000e+00  1.015e-17 0.000e+00    1.000
## perf_cor    4.000e+00  4.311e-17 9.279e+16   <2e-16 ***
## x1          2.789e-17  2.230e-17 1.251e+00    0.214
## x2          2.149e-17  1.557e-17 1.380e+00    0.171
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.931e-17 on 96 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 1.389e+34 on 3 and 96 DF,  p-value: < 2.2e-16
```

2) If there are more variables used for prediction than there are observations, lm will only keep the first n-1 variables.

```r
x3= rnorm(100)
x4= rnorm(100)
x5= rnorm(100)

all_x = data.frame(x1,x2,x3,x4,x5, y)
lm(y~ . ,data= all_x[1:4,])
```

```
##
## Call:
## lm(formula = y ~ ., data = all_x[1:4, ])
##
## Coefficients:
## (Intercept)           x1           x2           x3           x4           x5
##     -0.1978       3.0568       1.3830      -0.2328           NA           NA
```

Wine data

The wine dataset is related to red variants of the Portuguese "Vinho Verde" wine. There are 1599 samples available in the dataset. Due to privacy and logistic issues, only physicochemical (inputs) and sensory (the output) variables are available (e.g. there is no data about grape types, wine brand, wine selling price, etc.).

The explanatory variables are all continuous variables based on physicochemical tests:

- **fixed acidity**
- **volatile acidity**
- **citric acid**
- **residual sugar**
- **chlorides**
- **free sulfur dioxide**
- **total sulfur dioxide**
- **density**
- **pH**
- **sulphates**
- **alcohol**

The response variable is the **quality** score between 0 and 10 (based on sensory data).

Read data. We randomly split the data into two parts-the `wine` dataset with 1199 samples and the `wine.test` dataset with 400 samples. Splitting the dataset is a common technique when we want to evaluate the model performance. There are training set, validation set, and test set. The validation set is used for model selection. That is, to estimate the performance of the different model in order to choose the best one. The test set is used for estimating the performance of our final model.

```
set.seed("20170413")
wine.dataset <- read.csv("winequality-red.csv", sep = ";")
test.samples <- sample(1:nrow(wine.dataset), 400)
wine <- wine.dataset[-test.samples, ]
wine.test <- wine.dataset[test.samples, ]
```

To check the correlation between explanatory variables:

```
cor(wine[, -1])
```

```
##                      volatile.acidity citric.acid residual.sugar   chlorides
## volatile.acidity           1.00000000 -0.55181854    0.018252483  0.04518050
## citric.acid               -0.55181854  1.00000000    0.137994773  0.21684547
## residual.sugar             0.01825248  0.13799477    1.000000000  0.08275991
## chlorides                  0.04518050  0.21684547    0.082759914  1.00000000
## free.sulfur.dioxide       -0.01273824 -0.06423634    0.202975741  0.01270462
## total.sulfur.dioxide       0.07489677  0.02600556    0.205758724  0.05036798
## density                    0.03058276  0.35934671    0.370353372  0.20674145
## pH                         0.23621677 -0.54074018   -0.079922023 -0.26315313
## sulphates                 -0.26854815  0.31640987    0.014083620  0.38019243
## alcohol                   -0.20741181  0.11530555    0.012547909 -0.21053550
## quality                   -0.39251884  0.19969389    0.005793697 -0.13975190
```

```
##                       free.sulfur.dioxide total.sulfur.dioxide      density
## volatile.acidity              -0.012738244           0.07489677  0.030582758
## citric.acid                   -0.064236336           0.02600556  0.359346705
## residual.sugar                 0.202975741           0.20575872  0.370353372
## chlorides                      0.012704623           0.05036798  0.206741454
## free.sulfur.dioxide            1.000000000           0.65558561 -0.005227165
## total.sulfur.dioxide           0.655585607           1.00000000  0.084272651
## density                       -0.005227165           0.08427265  1.000000000
## pH                             0.082969156          -0.04574626 -0.327139624
## sulphates                      0.068308113           0.04302256  0.135721724
## alcohol                       -0.080517911          -0.22446333 -0.498422073
## quality                       -0.027710526          -0.19077256 -0.194995723
##                               pH   sulphates     alcohol      quality
## volatile.acidity      0.23621677 -0.26854815 -0.20741181 -0.392518837
## citric.acid          -0.54074018  0.31640987  0.11530555  0.199693888
## residual.sugar       -0.07992202  0.01408362  0.01254791  0.005793697
## chlorides            -0.26315313  0.38019243 -0.21053550 -0.139751896
## free.sulfur.dioxide   0.08296916  0.06830811 -0.08051791 -0.027710526
## total.sulfur.dioxide -0.04574626  0.04302256 -0.22446333 -0.190772562
## density              -0.32713962  0.13572172 -0.49842207 -0.194995723
## pH                    1.00000000 -0.19627991  0.19515875 -0.045387331
## sulphates            -0.19627991  1.00000000  0.11609469  0.270797056
## alcohol               0.19515875  0.11609469  1.00000000  0.498409123
## quality              -0.04538733  0.27079706  0.49840912  1.000000000
```

Great! The correlations are not as high as the diamond dataset we saw in the last lab, which means we do not need to worry too much about heteroscedasticity. We now fit the linear regression using all of the explanatory variables:

```
wine.fit <- lm(quality ~. ,data = na.omit(wine))
summary(wine.fit)
```

```
##
## Call:
## lm(formula = quality ~ ., data = na.omit(wine))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.69755 -0.35429 -0.03872  0.42375  1.99847
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)           2.093e+01  2.416e+01   0.867   0.3864
## fixed.acidity         3.130e-02  2.980e-02   1.050   0.2938
## volatile.acidity     -1.163e+00  1.373e-01  -8.465  < 2e-16 ***
## citric.acid          -3.944e-01  1.649e-01  -2.392   0.0169 *
## residual.sugar        2.289e-02  1.693e-02   1.351   0.1768
## chlorides            -2.191e+00  4.821e-01  -4.544 6.09e-06 ***
## free.sulfur.dioxide   6.202e-03  2.407e-03   2.576   0.0101 *
## total.sulfur.dioxide -3.471e-03  8.193e-04  -4.237 2.44e-05 ***
## density              -1.699e+01  2.468e+01  -0.688   0.4913
## pH                   -4.129e-01  2.176e-01  -1.898   0.0580 .
## sulphates             1.022e+00  1.311e-01   7.802 1.33e-14 ***
## alcohol               2.860e-01  2.991e-02   9.562  < 2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6314 on 1187 degrees of freedom
## Multiple R-squared:  0.3891, Adjusted R-squared:  0.3834
## F-statistic: 68.72 on 11 and 1187 DF,  p-value: < 2.2e-16
```

Exercise 1 Confidence Interval

(a) Calculate the confidence interval for all the coefficients from the regression done above. Which of these factors will positively influence the wine quality?

```
# Insert your code here to calculate the confidence intervals for the regression coefficients.
confint(wine.fit)
```

```
##                              2.5 %       97.5 %
## (Intercept)          -26.462057216 68.327234319
## fixed.acidity         -0.027165117  0.089762530
## volatile.acidity      -1.431958438 -0.893068438
## citric.acid           -0.718019962 -0.070876695
## residual.sugar        -0.010337622  0.056108942
## chlorides             -3.136567966 -1.244719980
## free.sulfur.dioxide    0.001478981  0.010925732
## total.sulfur.dioxide  -0.005078448 -0.001863641
## density              -65.406383126 31.429741775
## pH                    -0.839715970  0.013981287
## sulphates              0.765352411  1.279626998
## alcohol                0.227336816  0.344705115
```

(b) Calculate the confidence intervals for the samples in `wine.test` using the model you just fit. Which confidence interval will you use? Confidence intervals for the average response or the prediction interval?

```
# insert your code here and save your confidence intervals as `wine.confint`
wine.confint <- predict(wine.fit, newdata=data.frame(wine.test),interval="prediction")
head(wine.confint)
```

```
##           fit      lwr      upr
## 112  5.249080 4.002376 6.495785
## 1020 5.582145 4.341077 6.823214
## 1121 6.602847 5.356963 7.848730
## 266  6.081182 4.832368 7.329996
## 1368 5.877727 4.624099 7.131356
## 218  4.920709 3.678067 6.163351
```

(c) What is the percentage that your interval in (b) covers the true **quality** score in `wine.test`? What if you use the other confidence interval? Which one is consistent with your confidence level?

```
# insert your code here and save your percentage as `pct.covered`
subset = wine.test[which(wine.test$quality <= wine.confint[,3] & wine.test$quality >= wine.confint[,2])
pct.covered <- nrow(subset) / nrow(wine.test)
pct.covered
```

```
## [1] 0.9225
```

```
# insert your code here and save your percentage calculated
# using the other confidence interval as `pct.covered.other`
wine.confint.other <- predict(wine.fit, newdata=data.frame(wine.test),interval="confidence")
subset2 = wine.test[which(wine.test$quality <= wine.confint.other[,3]
                          & wine.test$quality >= wine.confint.other[,2]),]
pct.covered.other <- nrow(subset2) / nrow(wine.test)
```

```
pct.covered.other
```

```
## [1] 0.1225
```

Exercise 2 Bootstrap CI

Scale the columns of the dataset using scale() and then make 95% bootstrap confidence intervals for the coefficients for the predictors. Plot these confidence intervals using the plotCI() function in gplots. Code from professor for making bootstrap CI is included. You can use this or write your own code. Use the wine subset as used above.

```
bootstrapLM <- function(y,x, repetitions, confidence.level=0.95){
    # calculate the observed statistics
  stat.obs <- coef(lm(y~., data=x))
  # calculate the bootstrapped statistics
  bootFun<-function(){
          sampled <- sample(1:length(y), size=length(y),replace = TRUE)
          coef(lm(y[sampled]~.,data=x[sampled,])) #small correction here to make it for a matrix x
  }
  stat.boot<-replicate(repetitions,bootFun())
  # nm <-deparse(substitute(x))
  # row.names(stat.boot)[2]<-nm
  level<-1-confidence.level
  confidence.interval <- apply(stat.boot,1,quantile,probs=c(level/2,1-level/2))
    return(list(confidence.interval = cbind("lower"=confidence.interval[1,],"estimate"=stat.obs,"upper"=
}
library(gplots)
```

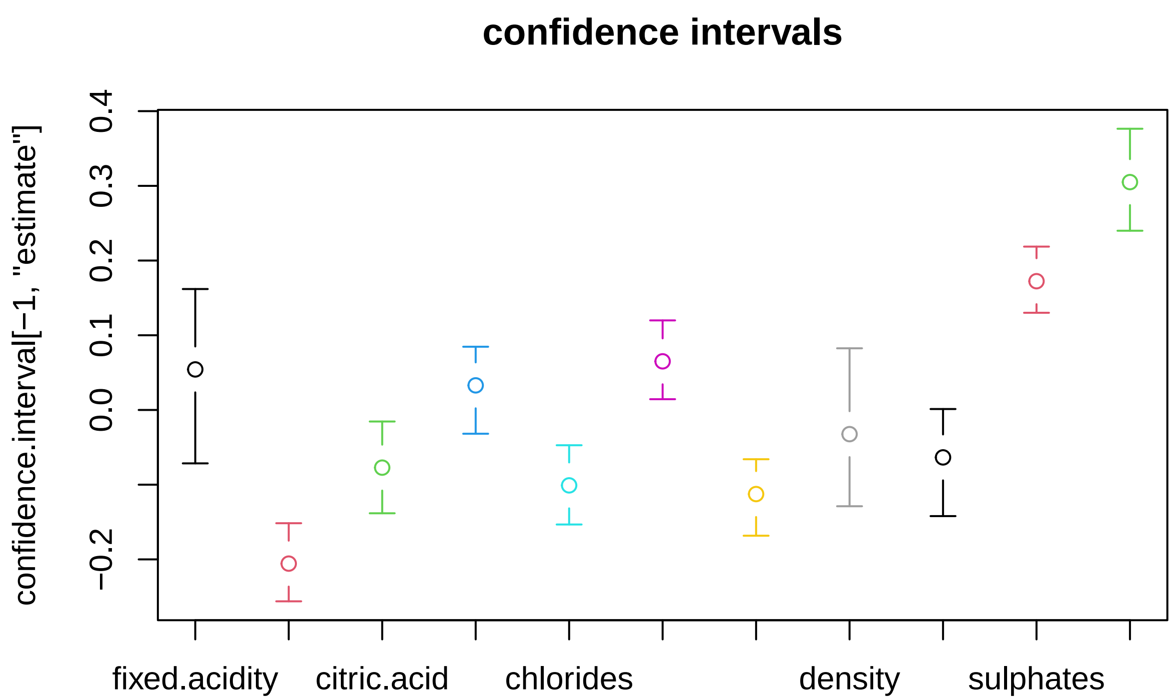
```
##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess
```

```
wineTemp <- wine
wineTemp[,-12]<-scale(wineTemp[,-12])
ftScale = lm(wine$quality ~ . , data = wineTemp)
wineBoot<-with(wineTemp,bootstrapLM(y=wine$quality,x=wineTemp[,-12],repetitions=1000))
wineBoot$conf
```

```
##                            lower    estimate        upper
## (Intercept)           5.60335271  5.63886572  5.672212915
## fixed.acidity        -0.07147370  0.05427201  0.161875561
## volatile.acidity     -0.25611564 -0.20567177 -0.151579068
## citric.acid          -0.13826187 -0.07716114 -0.015429577
## residual.sugar       -0.03182161  0.03294081  0.084650422
## chlorides            -0.15329516 -0.10093701 -0.047225115
## free.sulfur.dioxide   0.01440840  0.06509238  0.119883100
## total.sulfur.dioxide -0.16832647 -0.11255219 -0.065929025
## density              -0.12884727 -0.03235458  0.082565804
## pH                   -0.14204822 -0.06348670  0.001255322
## sulphates             0.13009200  0.17236563  0.218660614
## alcohol               0.23990829  0.30500162  0.376413887
```

```
with(wineBoot,plotCI(confidence.interval[-1,"estimate"],ui=confidence.interval[-1,"upper"],li=confidenc
axis(side=1,at=1:(nrow(wineBoot$conf)-1),rownames(wineBoot$conf)[-1])
```



Regression dianosis

Red wine dataset

Read data.

```
wine<- read.csv("winequality-red.csv", sep = ";")
wine$quality <- wine$quality + rnorm(length(wine$quality))
```

Fit the model.

```
wine.fit <- lm(quality~volatile.acidity+chlorides+free.sulfur.dioxide+total.sulfur.dioxide+pH+sulphates+
summary(wine.fit)
```

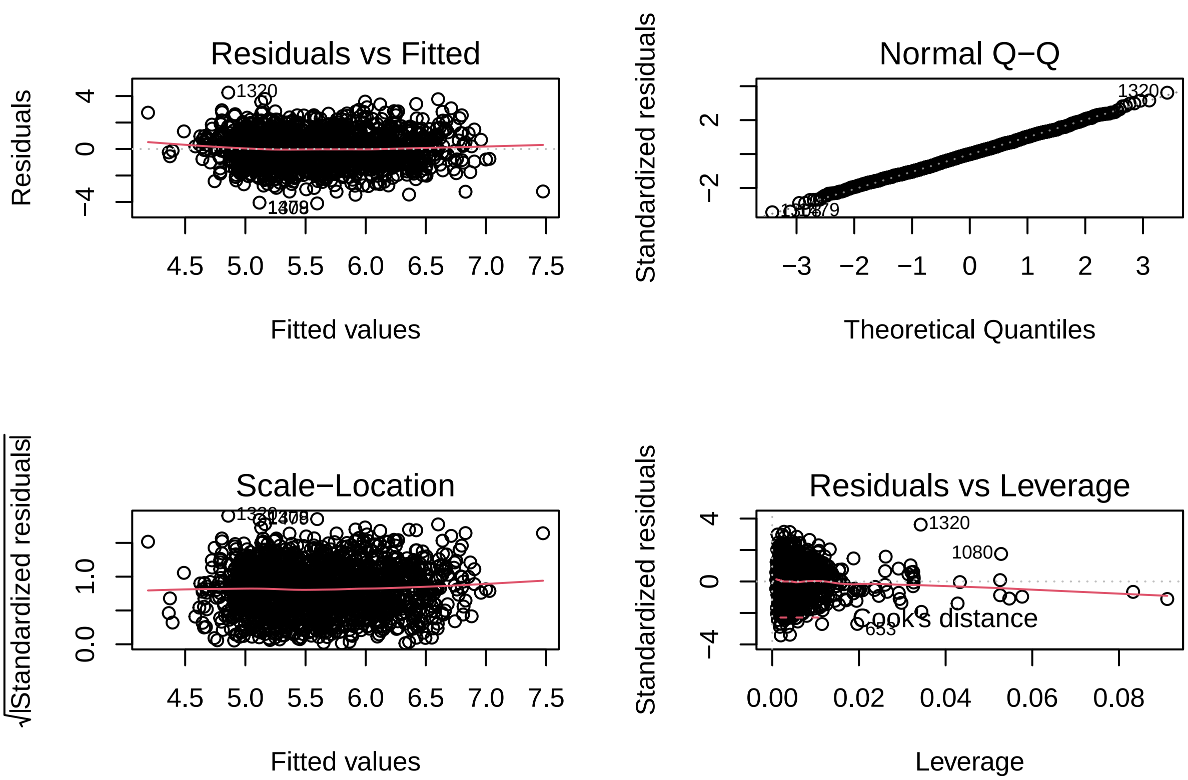
```
##
## Call:
## lm(formula = quality ~ volatile.acidity + chlorides + free.sulfur.dioxide +
##     total.sulfur.dioxide + pH + sulphates + alcohol, data = wine)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.1079 -0.8500  0.0007  0.7979  4.2605
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)           4.989896   0.746097   6.688 3.12e-11 ***
## volatile.acidity     -0.970431   0.186735  -5.197 2.29e-07 ***
```

```
## chlorides            -2.651956   0.736144  -3.602 0.000325 ***
## free.sulfur.dioxide   0.001675   0.003936   0.426 0.670502
## total.sulfur.dioxide -0.002988   0.001272  -2.350 0.018907 *
## pH                   -0.706068   0.217687  -3.244 0.001205 **
## sulphates             0.775964   0.203522   3.813 0.000143 ***
## alcohol               0.316487   0.031101  10.176  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.199 on 1591 degrees of freedom
## Multiple R-squared:  0.1515, Adjusted R-squared:  0.1477
## F-statistic: 40.57 on 7 and 1591 DF,  p-value: < 2.2e-16
```

Exercise 3

(a) Do regression diagnostics using the `plot` function.

```
# insert your code here to do regression diagnostics.
par(mfrow = c(2, 2))
plot(wine.fit)
```



(b) Answer the following TRUE/FALSE questions based on the diagnostics plot. Uncomment your answer.

```
### I. The plot indicates heteroscedasticity.
TRUE
```

```
## [1] TRUE
```

```
# FALSE
### II. There are non-linearity between the explanatory variable and response variable.
# TRUE
FALSE
```

```
## [1] FALSE
```

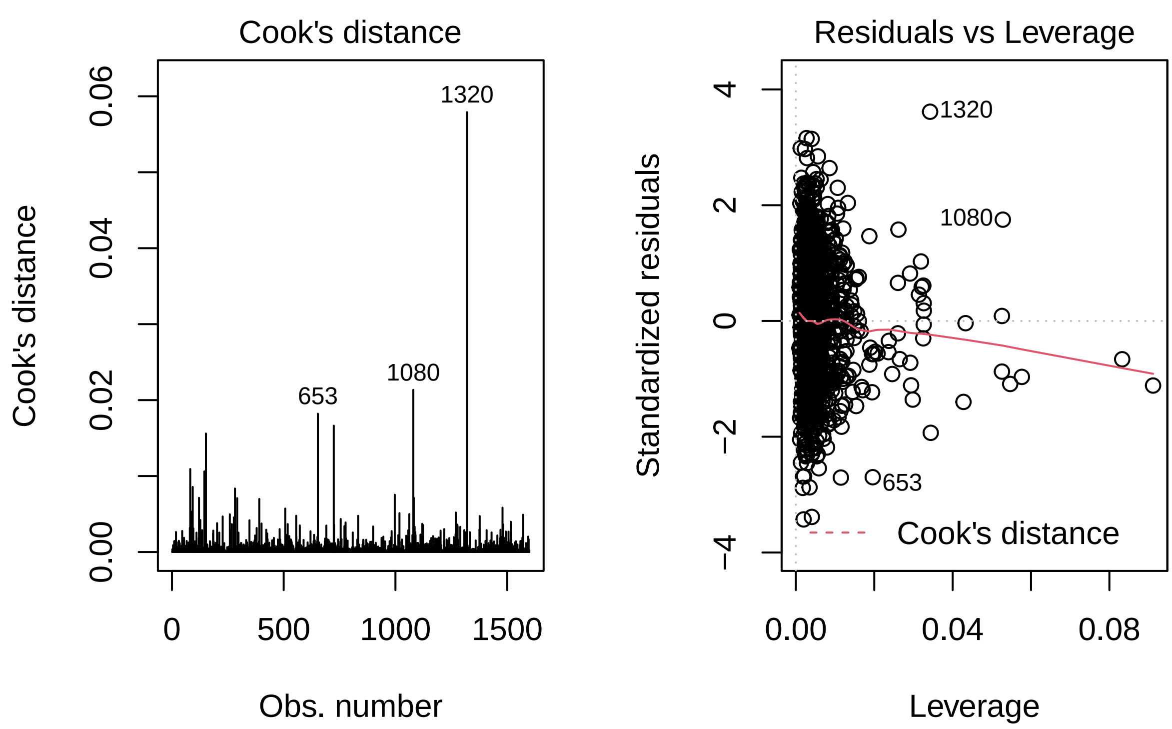
```
### III. The normal assumption holds for this model.
TRUE
```

```
## [1] TRUE
```

```
# FALSE
```

(c) Identify at least two outliers from the data.

```
par(mfrow=c(1,2))
plot(wine.fit,which=c(4:5))
```



I think the sample 1320 and 1080 are outliers.

Diamond dataset

Read the data.

```
diamonds <- read.csv("diamonds.csv")
diamonds <- diamonds[sample(1:nrow(diamonds), 1000), ]
head(diamonds)
```

```
##       carat       cut color clarity depth table price length.in.mm width.of.mm
```

```
## 20453  1.10     Ideal     F     VS1  61.5    59  8796         6.63        6.58
## 5623   0.91 Very Good     E     SI2  62.8    59  3876         6.10        6.16
## 32068  0.33   Premium     D     SI1  61.5    60   780         4.44        4.41
## 42306  0.55 Very Good     D     SI2  63.3    56  1295         5.22        5.24
## 28967  0.31 Very Good     H     SI1  62.4    57   435         4.31        4.35
## 9247   1.01 Very Good     H     VS2  63.3    57  4559         6.32        6.27
##       depth.in.mm
## 20453        4.06
## 5623         3.85
## 32068        2.72
## 42306        3.31
## 28967        2.70
## 9247         3.99
```

Fit a linear regression.

```
diamond.fit <- lm(price ~ carat + cut + color + clarity + depth + table, data = diamonds)
summary(diamond.fit)
```

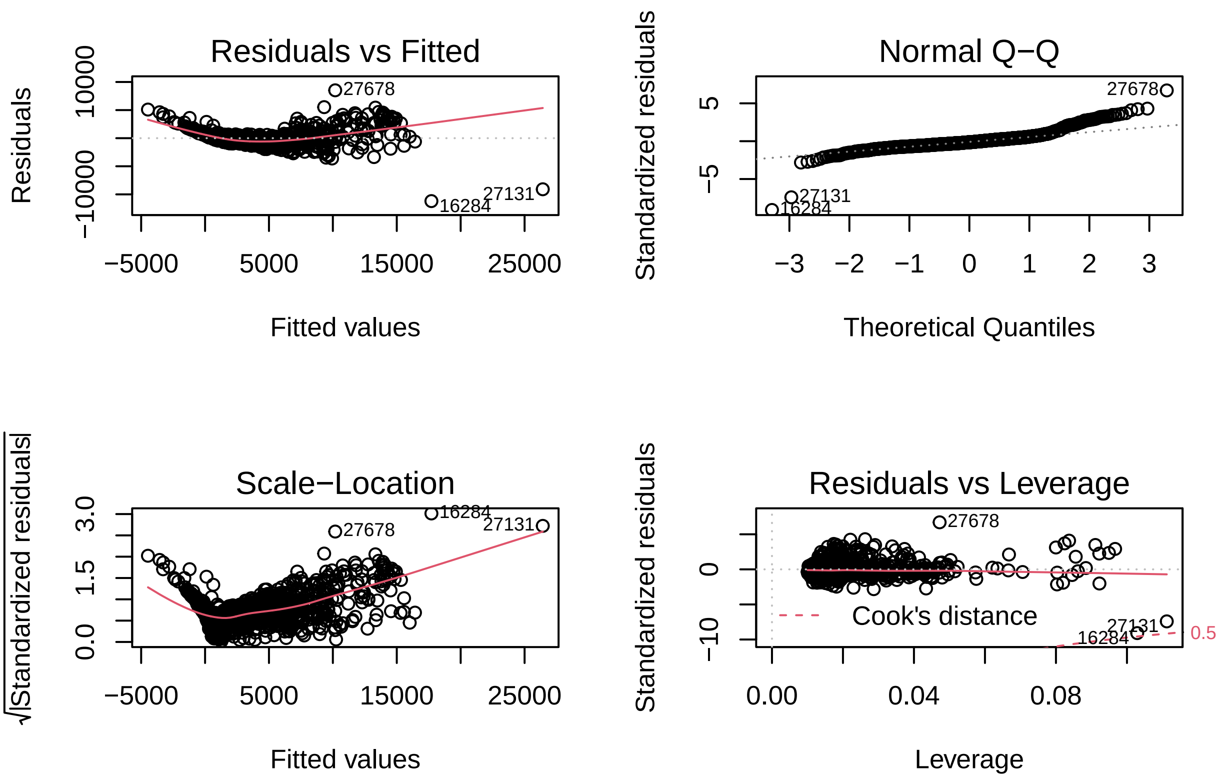
```
##
## Call:
## lm(formula = price ~ carat + cut + color + clarity + depth +
##     table, data = diamonds)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11198.4   -675.7   -176.0    441.4   8517.8
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -5768.684   3207.117  -1.799 0.072372 .
## carat         8797.403     98.722  89.113  < 2e-16 ***
## cutGood        954.297    278.673   3.424 0.000642 ***
## cutIdeal      1024.089    281.940   3.632 0.000295 ***
## cutPremium    1210.205    271.485   4.458 9.24e-06 ***
## cutVery Good   926.977    269.127   3.444 0.000597 ***
## colorE        -170.755    151.825  -1.125 0.260999
## colorF        -108.752    158.375  -0.687 0.492453
## colorG        -397.542    152.837  -2.601 0.009433 **
## colorH        -960.381    161.576  -5.944 3.87e-09 ***
## colorI       -1255.133    185.107  -6.781 2.07e-11 ***
## colorJ       -2160.127    235.827  -9.160  < 2e-16 ***
## clarityIF     6809.361    455.055  14.964  < 2e-16 ***
## claritySI1    4776.702    377.042  12.669  < 2e-16 ***
## claritySI2    4020.381    378.912  10.610  < 2e-16 ***
## clarityVS1    5778.042    387.502  14.911  < 2e-16 ***
## clarityVS2    5376.663    380.428  14.133  < 2e-16 ***
## clarityVVS1   6290.624    415.121  15.154  < 2e-16 ***
## clarityVVS2   6305.024    398.125  15.837  < 2e-16 ***
## depth           -2.539     33.988  -0.075 0.940466
## table          -49.445     25.154  -1.966 0.049618 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1301 on 979 degrees of freedom
```

```
## Multiple R-squared:    0.9,  Adjusted R-squared:  0.8979
## F-statistic: 440.3 on 20 and 979 DF,  p-value: < 2.2e-16
```

Exercise 4

(a) Do regression diagnostics using the `plot` function.

```
# insert your code here to do regression diagnostics.
par(mfrow = c(2, 2))
plot(diamond.fit)
```



(b) Answer the following TRUE/FALSE questions based on the diagnostics plot. Uncomment your answer.

```
### I. The plot indicates heteroscedasticity.
# TRUE
FALSE
```

```
## [1] FALSE
```

```
### II. There are non-linearity between the explantory variable and response variable.
# TRUE
FALSE
```

```
## [1] FALSE
```

```
### III. The normal assumtion holds for this model.
# TRUE
FALSE
```

```
## [1] FALSE
```