Homework 1

R Homework 1

Part 1

1.Create an RMarkDown project

2.Section 1 Univariate Regression Data

a. Write f(x)= sin(x)/x in [-5pi,5pi] mathematically f(x) = sin(x)/x in [-5pi, 5pi]

b. Set n as sample size to some value, set 200

```
n <-200
```

c. Generate equally spaced value of x in [-5pi,5pi] and store in vector x=(x1,x2,x3,. . . .xn)^T

```
x <- seq(-5*pi,5*pi, length.out = n)
```

d. Write an R function that returns f(x)=sin(x)/x, Note be mindful of what happens of x=0.

```
f <- function(x){
  return(ifelse(x == 0,1,sin(x)/x))
}
```

e. Generate a vector of n noise value from Normal(o, sigma^2)

```
sigma <-0.05
y.noise <- rnorm(n,0,sigma)
```

f. Create the n-dimension vector Y=(y1,y2,. . . yn)^T transpose where Yi=f(xi)+ei

```
Y <- f(x) + y.noise
Y <- matrix(Y)
```

g. Create matrix X=[1,x1] (2 x n)

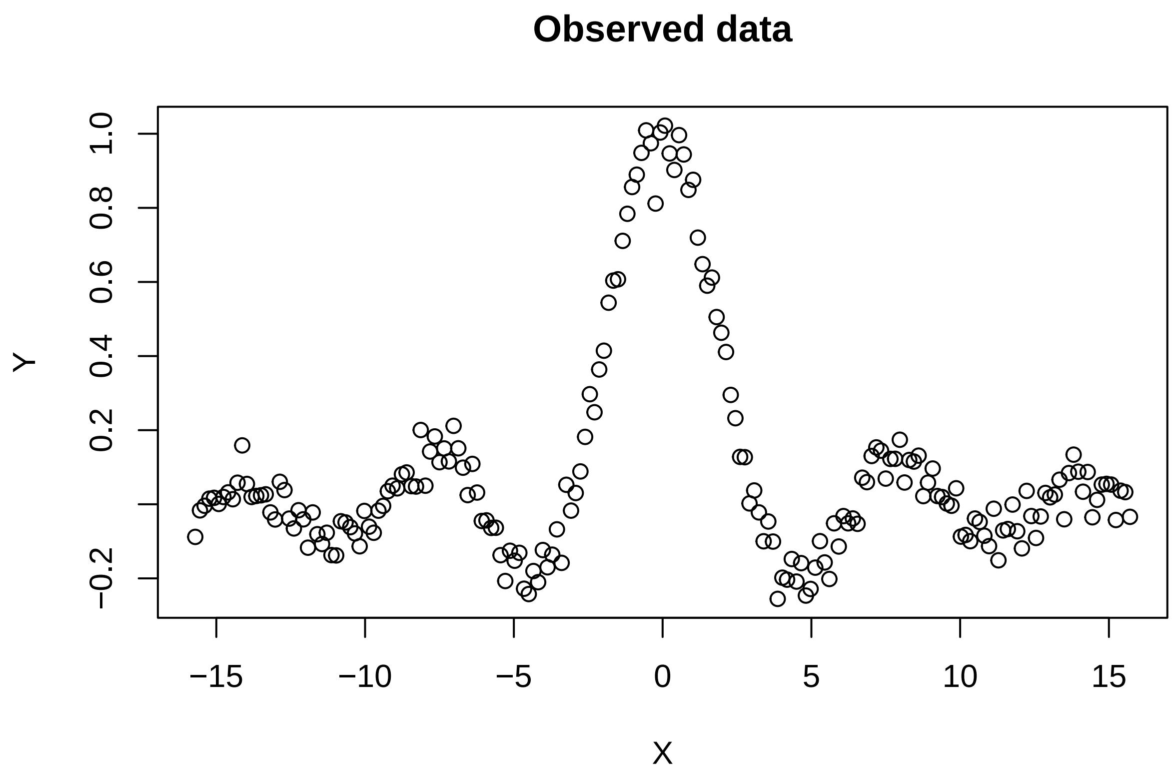
```
X <- matrix(x)
X <- cbind(1, X)
```

h. Create data frame of two columns X and Y

```
df <- data.frame(X = x, Y = Y)
```

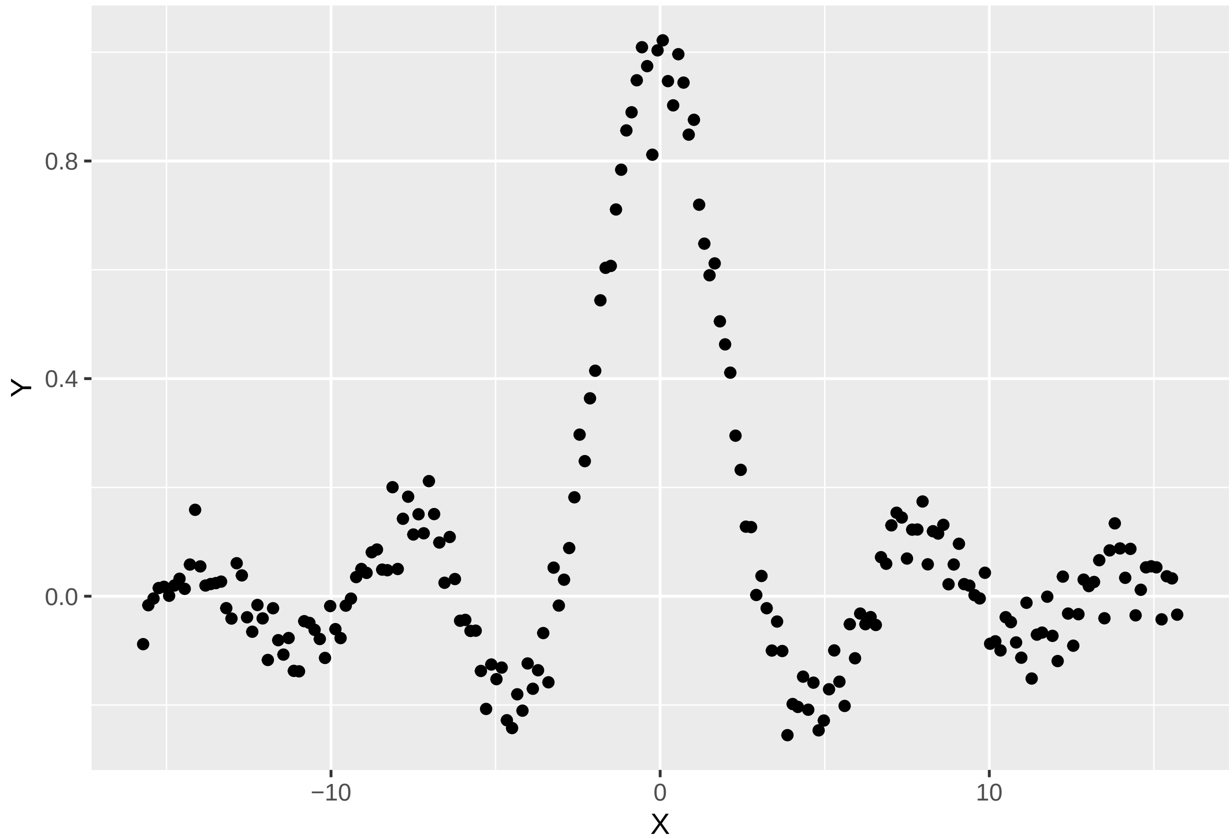
i. Using classical plotting plot data points and superpose the tree function, and add the suitable legend

```
plot(df$X,df$Y,xlab = "X",ylab = "Y",main = 'Observed data' )
```



j. Redo (i) using ggplot2

```
library(ggplot2)
ggplot(data = df,mapping = aes(x = X, y = Y)) + geom_point()
```

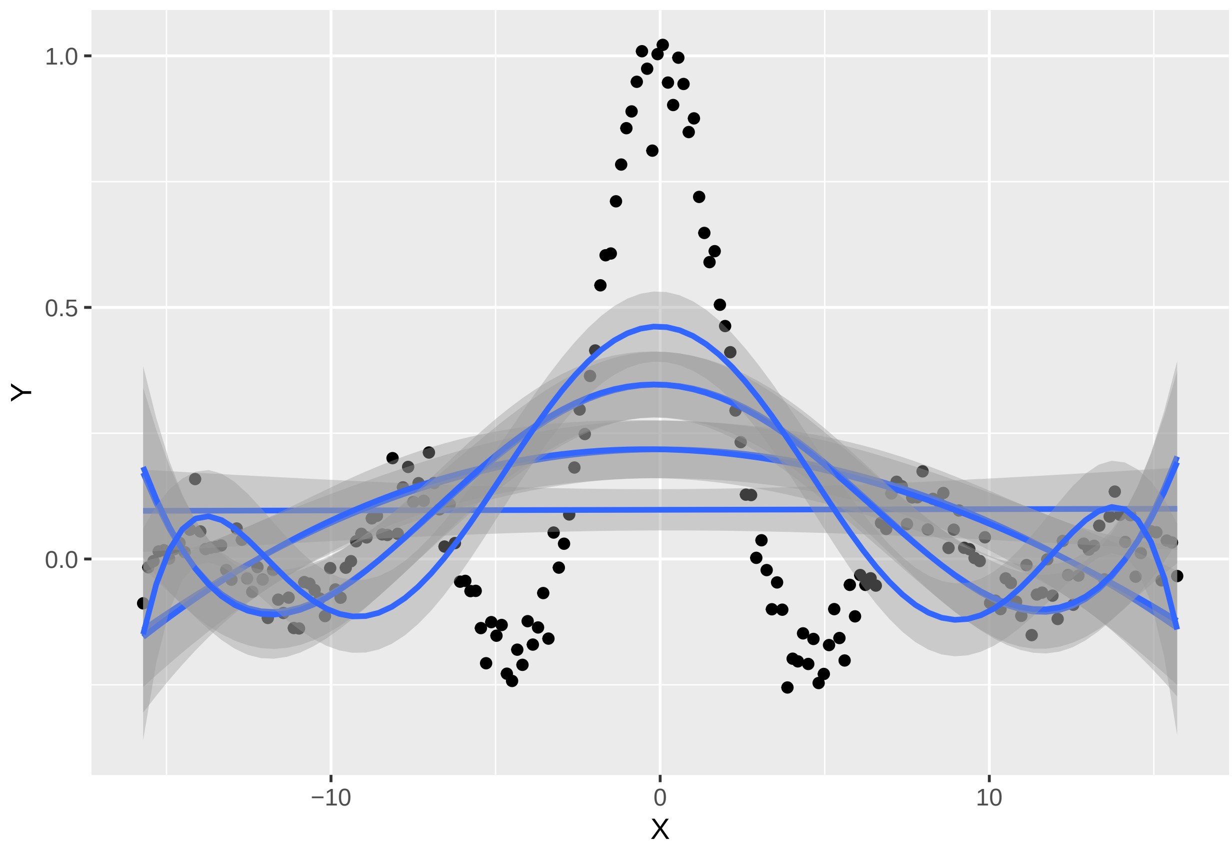


k. Fit a polynomial regression learning machine to the data, exploring degree 1,2,3,4,5,6

```
Degree <- 6
poly1 <- Y ~ poly(x = x ,degree =  1)
poly2 <- Y ~ poly(x = x ,degree =  2)
poly3 <- Y ~ poly(x = x ,degree =  3)
poly4 <- Y ~ poly(x = x ,degree =  4)
poly5 <- Y ~ poly(x = x ,degree =  5)
poly6 <- Y ~ poly(x = x ,degree =  6)
```

l. Redo(i) with the added smooth curves of (k)

```
library(ggplot2)
ggplot(data = df,mapping = aes(x = X,y = Y)) +
  geom_point() +
  stat_smooth(method='lm', formula = poly1, size = 1) +
  stat_smooth(method='lm', formula = poly2, size = 1) +
  stat_smooth(method='lm', formula = poly3, size = 1) +
  stat_smooth(method='lm', formula = poly4, size = 1) +
  stat_smooth(method='lm', formula = poly5, size = 1) +
  stat_smooth(method='lm', formula = poly6, size = 1) +
  xlab('X') +
  ylab('Y')
```



Part 2

a. Write a function that takes a frame or the indices there of and return a stochastic Hold out split into training set and test set. tao 1/3 from (0,1) is proportion of data in test set and 1-tao 2/3 is proportion in train set sho <- function(n,tau) nte <-round(n * tau) ntr <-n-nte id.tr <-sample(n)[1:ntr] id.tr <- set diff(1:n,id.ty) return()

```
tau <- 1/3

sho <- function(n, tau){
  ntr <- NULL
  nte <- NULL
  nte <- round(n * tau)
  ntr <- n - nte
  id.tr <- sample(n)[1 : ntr]
  id.te <- setdiff(1:n, id.tr)
  return(list(tr = ntr,ts = nte,id.tr = id.tr, id.te =id.te))
}
```

b. Set S as number of replications (S=100)

```
S <- 100
```

c. Set the seed 787369

```
set.seed(787369)
```

d. Create two matrices of errors, one for training error, one for testing error, initialized to 0

```
Train<- matrix(0, S ,6)
Test <- matrix(0, S ,6)
```

e. Repeat S time, Split with SHO function, fit each 6 learners with training data,
f. compute prediction for each with test data.

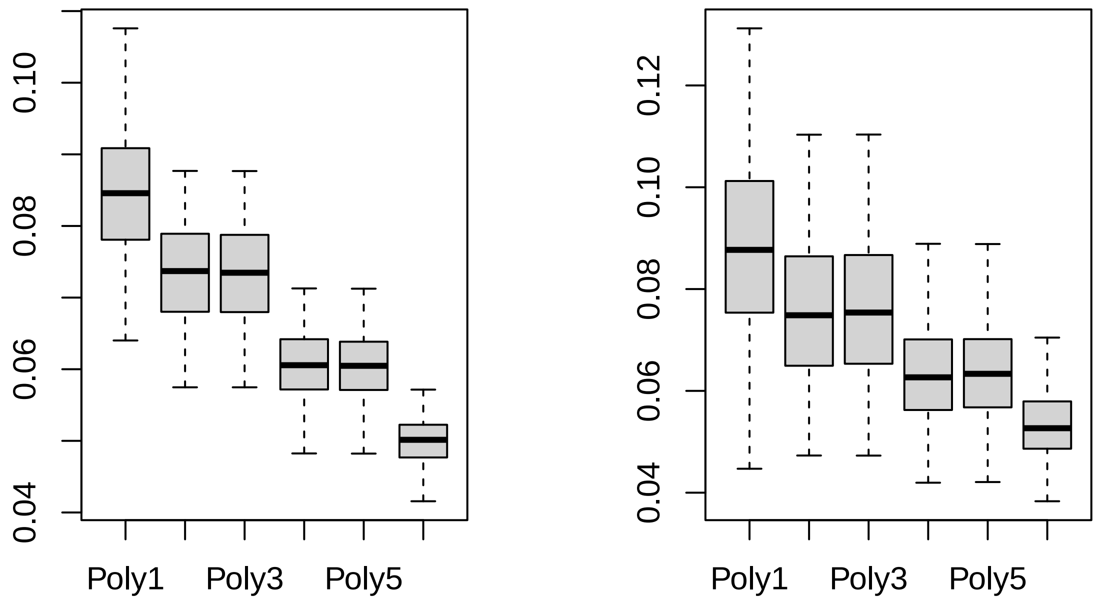
```
for (i in 1 : S) {
  split <- sho(n,tau = 1/3)
  train_index <- split$id.tr
  test_index <- split$id.te
  train_data <- df[train_index, ]
  test_data <- df[test_index, ]

  for (j in 1 : Degree) {
    train <- lm(Y~poly(X,j), data = train_data)
    y_train <- predict(train)
    train_error <- mean((train_data$Y - y_train)^2)
    y_test <-  predict(train, test_data)
    test_error <- mean((test_data$Y - y_test)^2)
    Train[i,j] <- train_error
    Test[i,j] <- test_error
  }
}
```

```
colnames(Train) <- c("Poly1","Poly2","Poly3","Poly4","Poly5","Poly6")
colnames(Test) <- c("Poly1","Poly2","Poly3","Poly4","Poly5","Poly6")
```

f. Box plot(properly labelled with learning machine names). matrix of training error. matrix of text errors.

```
par(mfrow = c(1,2))
boxplot(Train)
boxplot(Test)
```



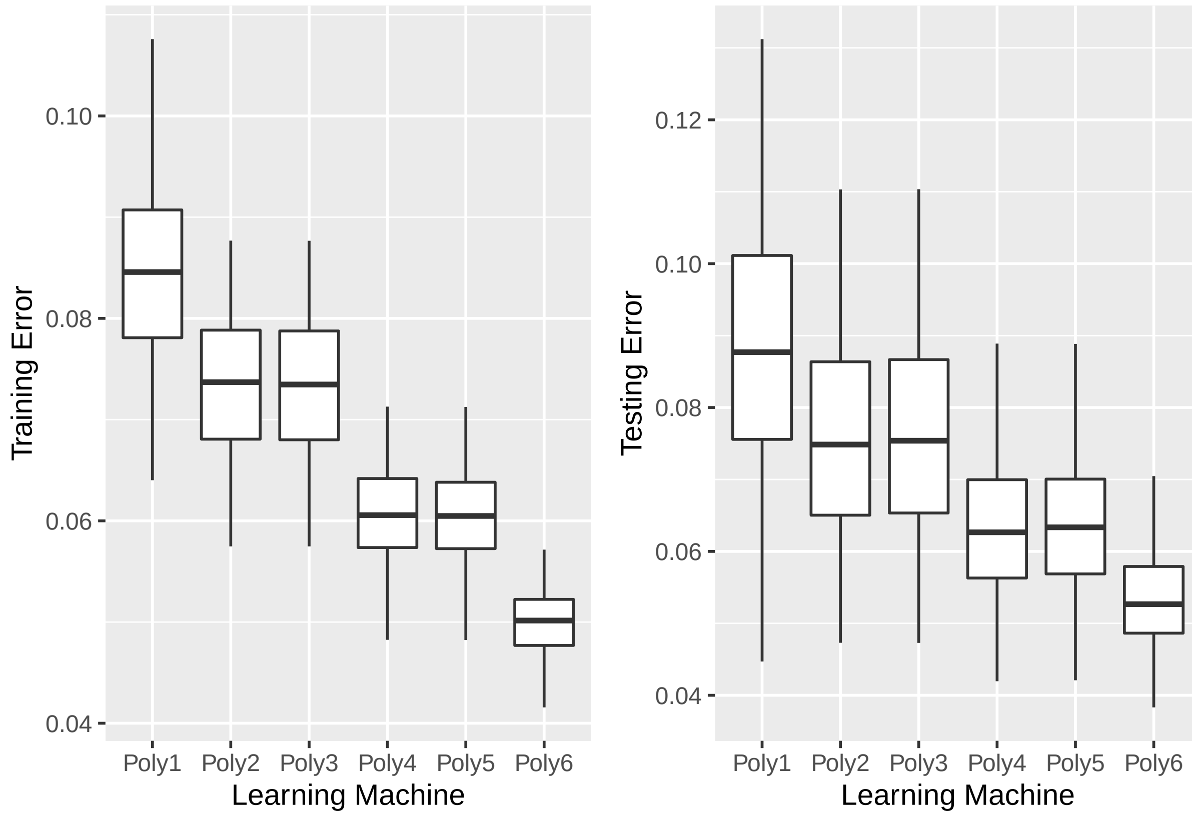
g. Redo this with ggplot2.

```
library(patchwork)
Train <-as.data.frame(Train)
Test <- as.data.frame(Test)

train_plot <- ggplot(stack(Train), aes(x = ind, y = values)) +
  geom_boxplot() +
  xlab('Learning Machine') +
  ylab('Training Error')

test_plot <- ggplot(stack(Test), aes(x = ind, y = values)) +
  geom_boxplot() +
  xlab('Learning Machine') +
  ylab('Testing Error')

train_plot + test_plot
```



h. Perform ANOVA on test error.

```
aov_test_1 <- aov(Test$Poly1~Test$Poly2+Test$Poly3+Test$Poly4+Test$Poly5+Test$Poly6,Test)
summary(aov_test_1)
```

```
##              Df   Sum Sq  Mean Sq  F value Pr(>F)
## Test$Poly2    1 0.031091 0.031091 3441.103 <2e-16 ***
## Test$Poly3    1 0.000034 0.000034    3.711 0.0571 .
## Test$Poly4    1 0.001201 0.001201  132.925 <2e-16 ***
## Test$Poly5    1 0.000002 0.000002    0.222 0.6387
## Test$Poly6    1 0.000038 0.000038    4.214 0.0429 *
## Residuals    94 0.000849 0.000009
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
aov_test_2 <- aov(Test$Poly2~Test$Poly1+Test$Poly3+Test$Poly4+Test$Poly5+Test$Poly6,Test)
summary(aov_test_2)
```

```
##              Df   Sum Sq  Mean Sq   F value   Pr(>F)
## Test$Poly1    1 0.018047 0.018047 1.130e+05  < 2e-16 ***
## Test$Poly3    1 0.001207 0.001207 7.560e+03  < 2e-16 ***
## Test$Poly4    1 0.000001 0.000001 9.157e+00   0.0032 **
## Test$Poly5    1 0.000009 0.000009 5.757e+01 2.32e-11 ***
## Test$Poly6    1 0.000000 0.000000 7.310e-01   0.3947
```

```
## Residuals   94 0.000015 0.000000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
aov_test_3 <-aov(Test$Poly3~Test$Poly2+Test$Poly1+Test$Poly4+Test$Poly5+Test$Poly6,Test)
summary(aov_test_3)
```

```
##             Df   Sum Sq  Mean Sq   F value   Pr(>F)
## Test$Poly2   1 0.019190 0.019190 1.121e+05  < 2e-16 ***
## Test$Poly1   1 0.000000 0.000000 2.547e+00    0.114
## Test$Poly4   1 0.000000 0.000000 1.983e+00    0.162
## Test$Poly5   1 0.000011 0.000011 6.274e+01 4.68e-12 ***
## Test$Poly6   1 0.000000 0.000000 4.600e-02    0.830
## Residuals   94 0.000016 0.000000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
aov_test_4 <- aov(Test$Poly4~Test$Poly2+Test$Poly3+Test$Poly1+Test$Poly5+Test$Poly6,Test)
summary(aov_test_4)
```

```
##             Df   Sum Sq  Mean Sq  F value   Pr(>F)
## Test$Poly2   1 0.009022 0.009022 23639.34  < 2e-16 ***
## Test$Poly3   1 0.000023 0.000023    59.06 1.46e-11 ***
## Test$Poly1   1 0.000448 0.000448  1174.85  < 2e-16 ***
## Test$Poly5   1 0.000291 0.000291   763.35  < 2e-16 ***
## Test$Poly6   1 0.000005 0.000005    12.67 0.000586 ***
## Residuals   94 0.000036 0.000000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
aov_test_5 <-aov(Test$Poly5~Test$Poly2+Test$Poly3+Test$Poly4+Test$Poly1+Test$Poly6,Test)
summary(aov_test_5)
```

```
##             Df   Sum Sq  Mean Sq   F value Pr(>F)
## Test$Poly2   1 0.008860 0.008860 17274.030 <2e-16 ***
## Test$Poly3   1 0.000114 0.000114   221.434 <2e-16 ***
## Test$Poly4   1 0.000833 0.000833  1624.286 <2e-16 ***
## Test$Poly1   1 0.000000 0.000000     0.212  0.646
## Test$Poly6   1 0.000000 0.000000     0.017  0.896
## Residuals   94 0.000048 0.000001
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

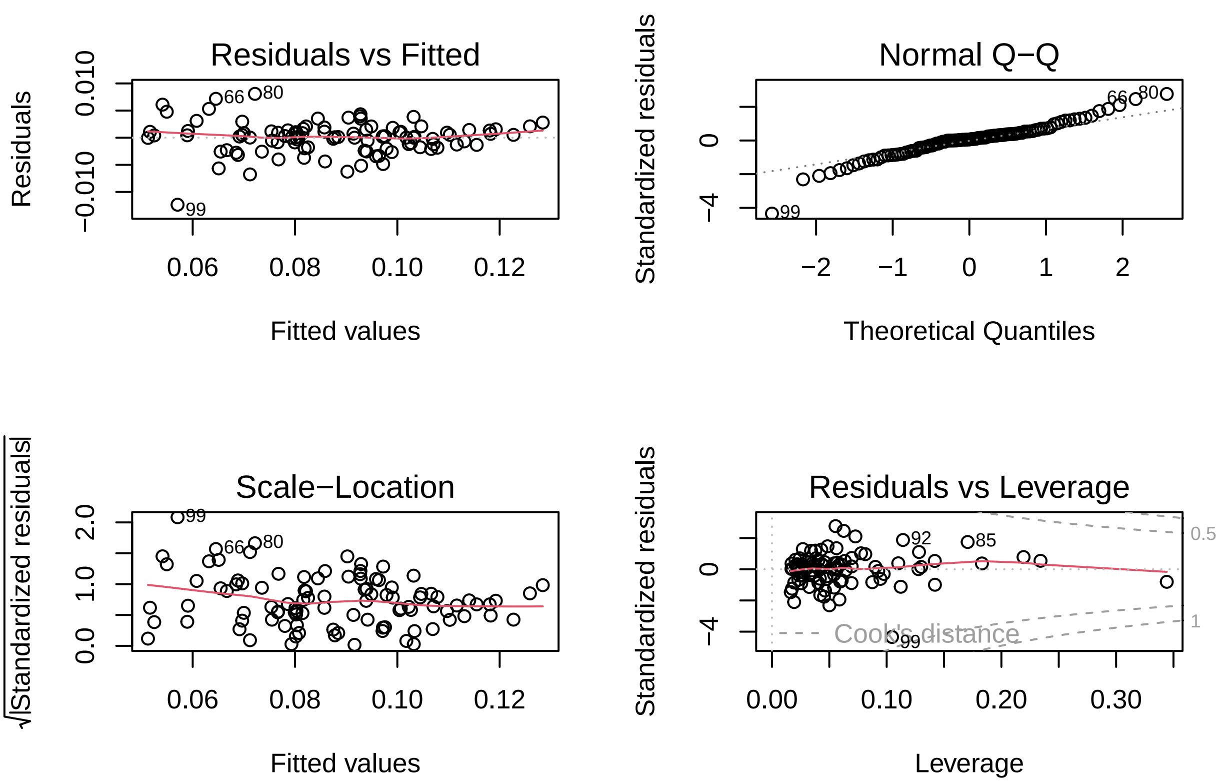
```
aov_test_6 <- aov(Test$Poly6~Test$Poly2+Test$Poly3+Test$Poly4+Test$Poly5+Test$Poly1,Test)
summary(aov_test_6)
```

```
##             Df   Sum Sq  Mean Sq F value Pr(>F)
## Test$Poly2   1 0.003464 0.003464 645.160 <2e-16 ***
## Test$Poly3   1 0.000017 0.000017   3.121 0.0805 .
## Test$Poly4   1 0.000859 0.000859 160.053 <2e-16 ***
## Test$Poly5   1 0.000000 0.000000   0.052 0.8199
```

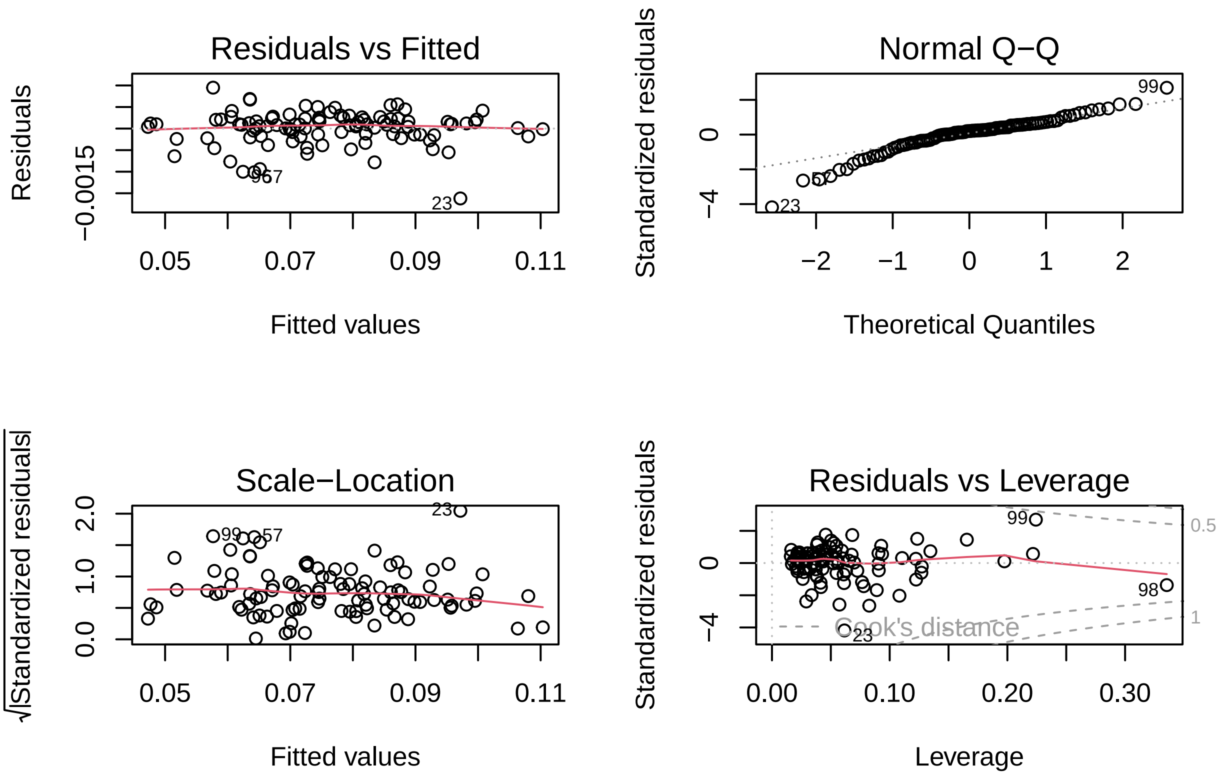
```
## Test$Poly1    1 0.000023 0.000023   4.214 0.0429 *
## Residuals    94 0.000505 0.000005
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

i. Plot multiple comparisons.

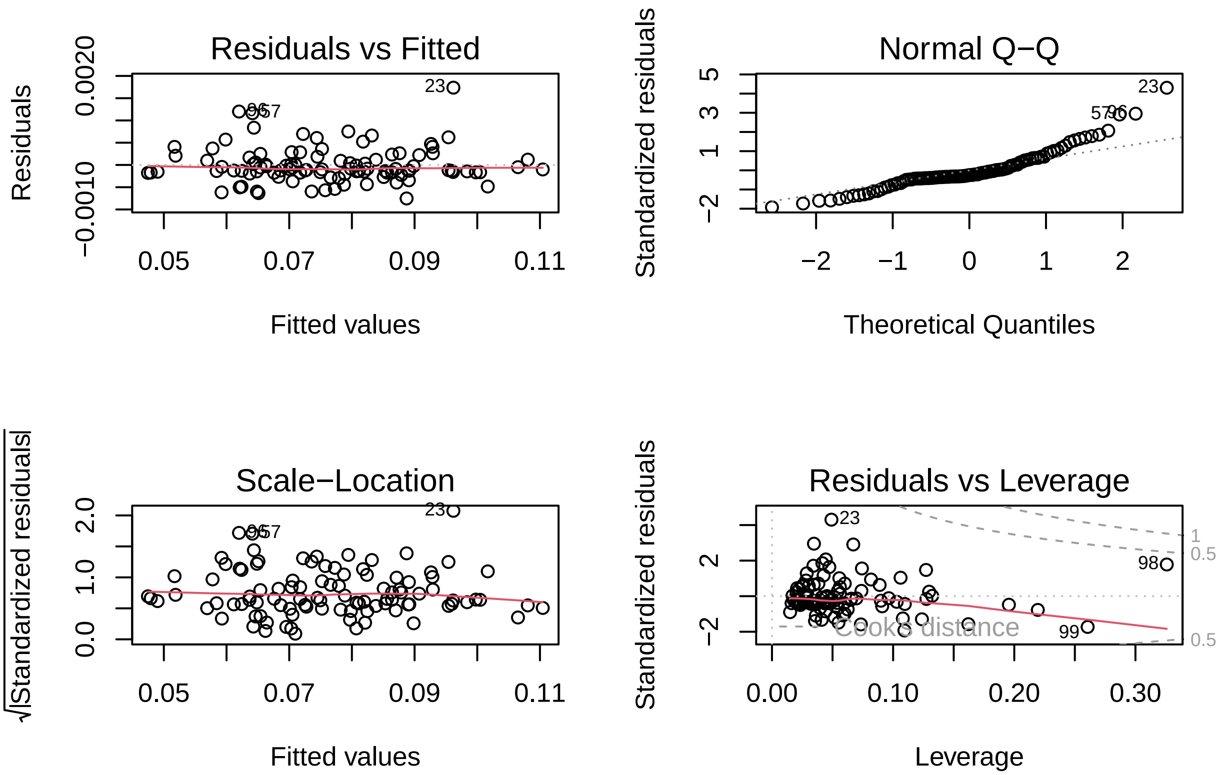
```
par(mfrow=c(2,2))
plot(aov_test_1)
```



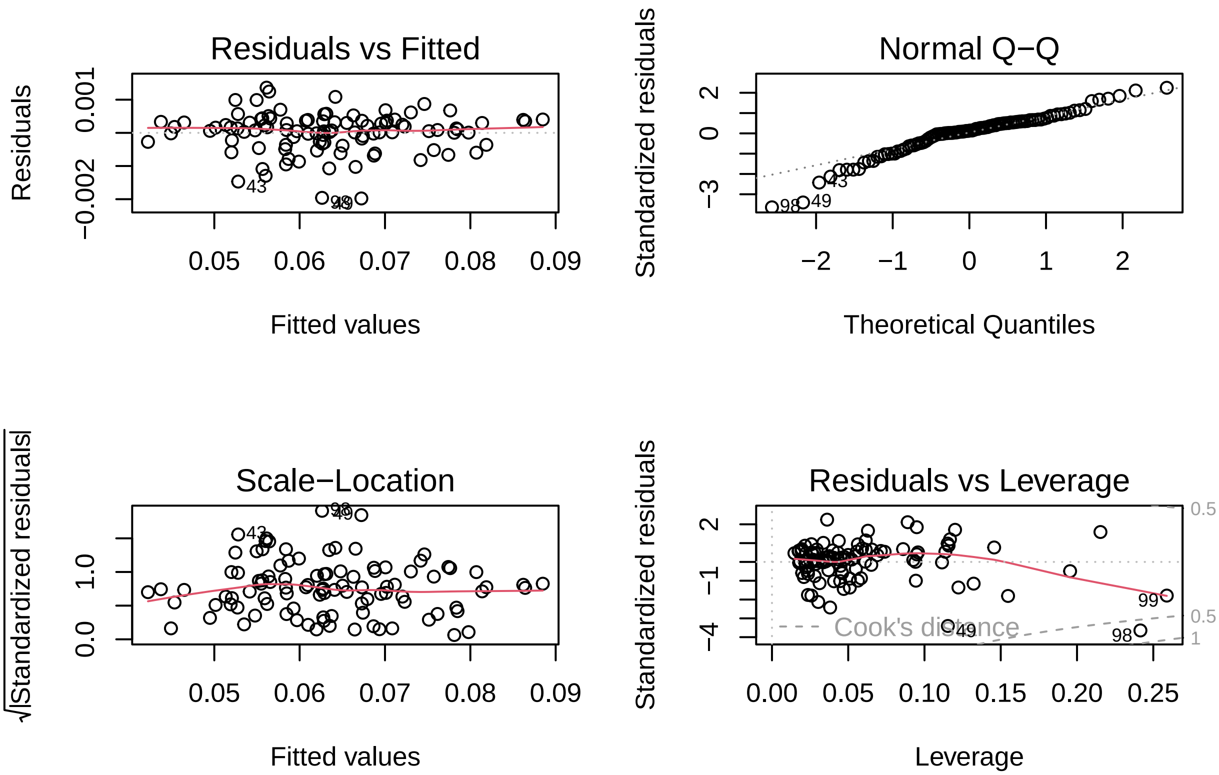

```
par(mfrow=c(2,2))
plot(aov_test_2)
```



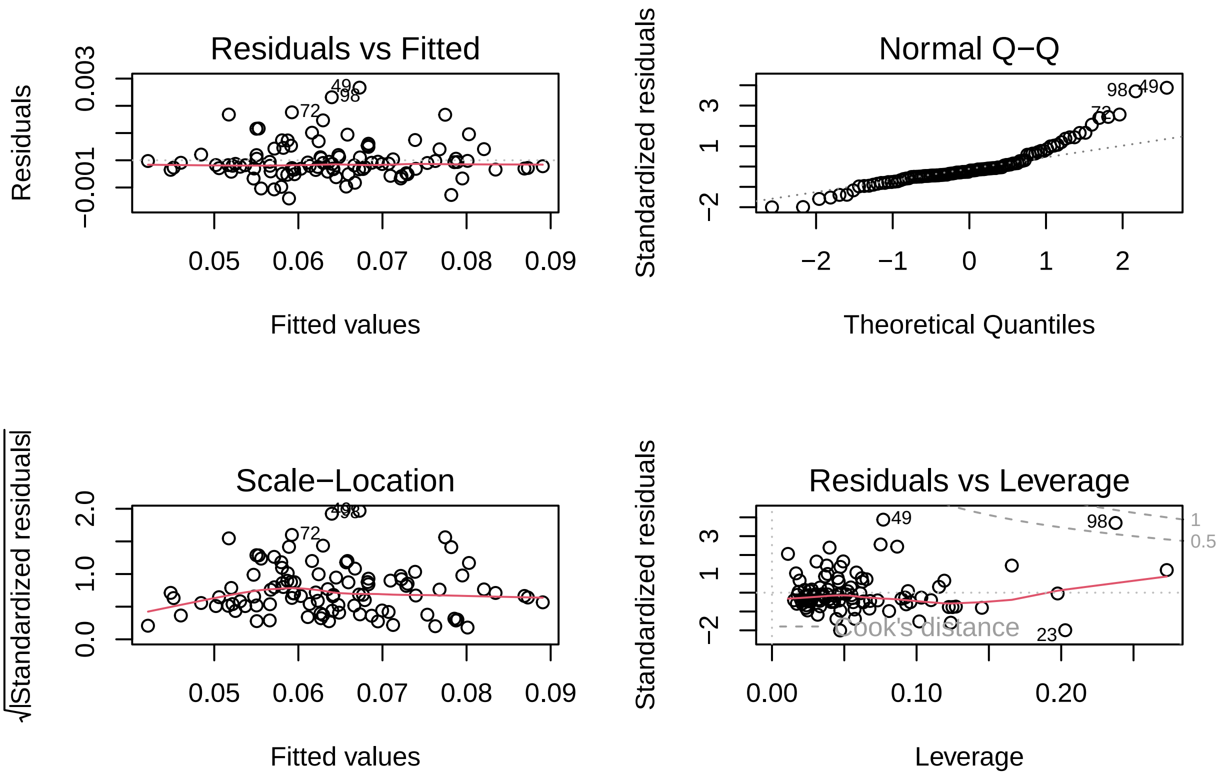

```
par(mfrow=c(2,2))
plot(aov_test_3)
```



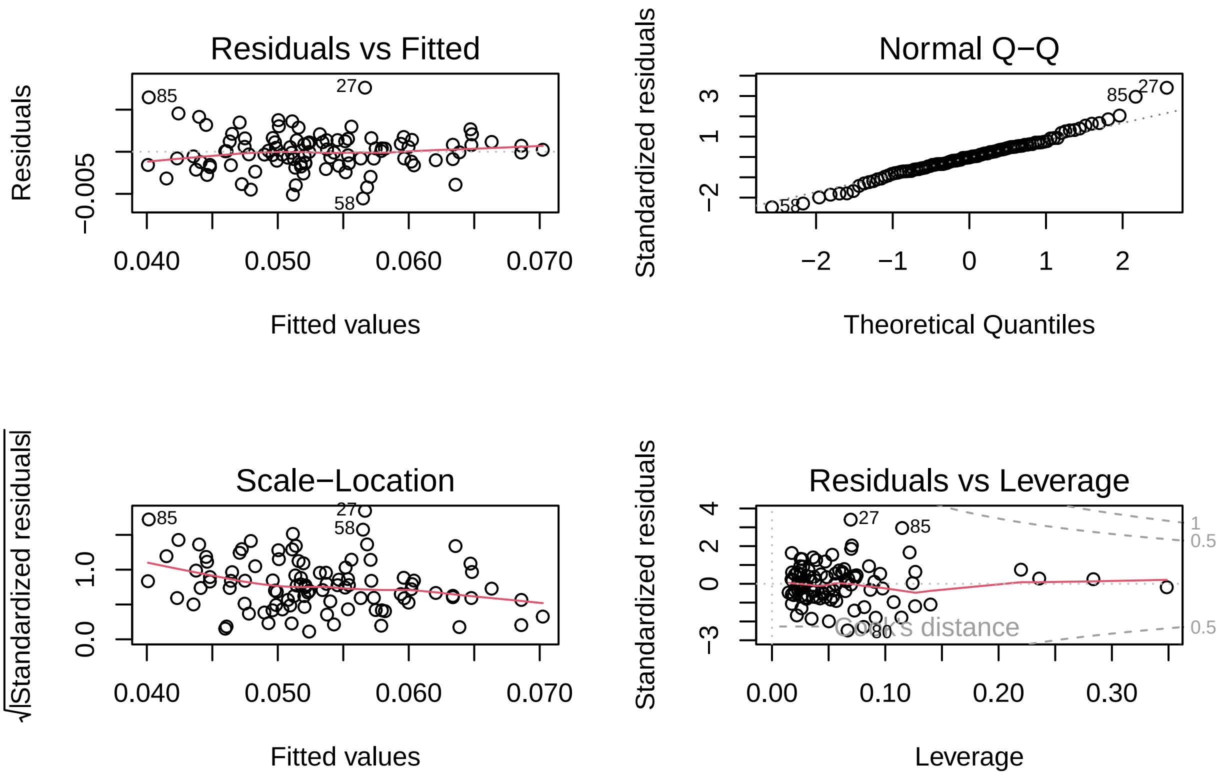

```
par(mfrow=c(2,2))
plot(aov_test_4)
```




```
par(mfrow=c(2,2))
plot(aov_test_5)
```




```r
par(mfrow=c(2,2))
plot(aov_test_6)
```


Residuals vs Fitted
Residuals
Fitted values
Normal Q−Q
Standardized residuals
Theoretical Quantiles
Scale−Location
√|Standardized residuals|
Fitted values
Residuals vs Leverage
Standardized residuals
Leverage
Cook's distance