**R final Report Research**

I use stepwise, backward and forward regression as my model.

**I. Read data**

library(readxl)

my\_data <- read\_excel("data-tableB3.xlsx")

data <- as.matrix(my\_data)

head(data)

dim(data)

#remove NA

df <- na.omit(data)

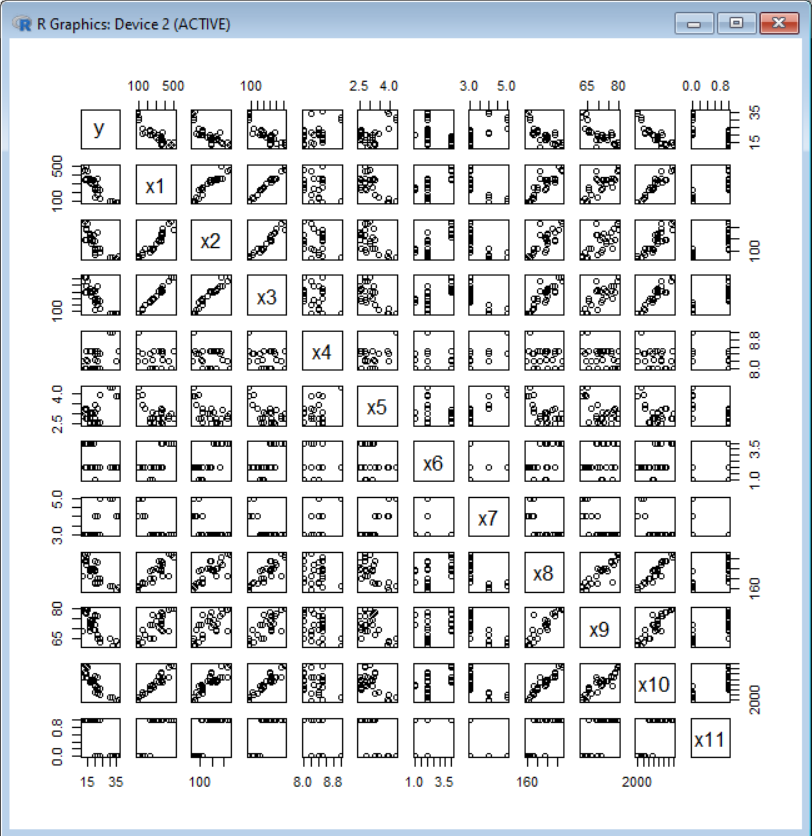
dim(df)

head(df)

y <- df[,1]

x <- df[,2:12]

pairs(y~ ., data = df)



**II. Modeling & Result:**

**1.Both (Forward & backward) direction regression**

(1) **y ~ original data**

**#AIC = 68.29 x5,x8,x10 Ra=0.7808**

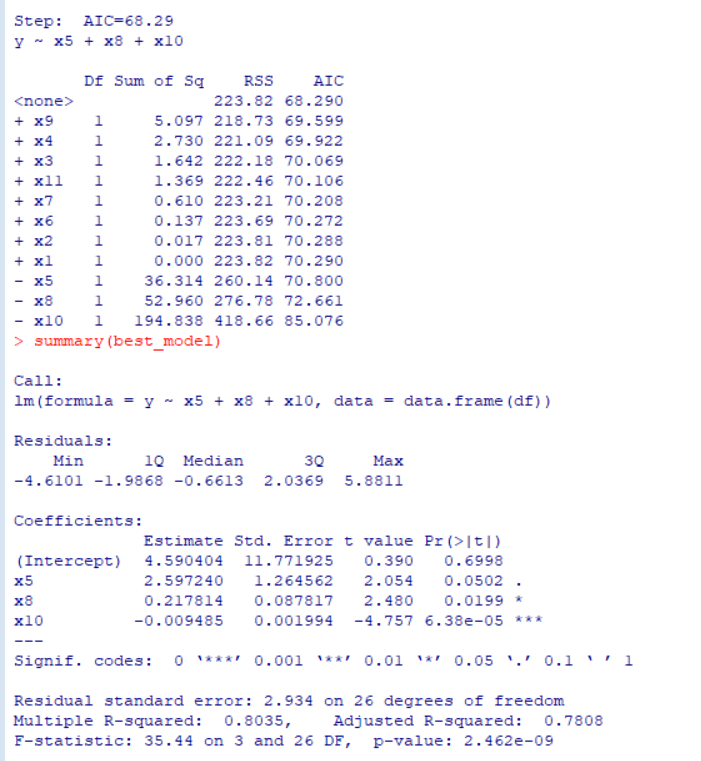
init\_model <- lm(y ~ ., data = data.frame(df))

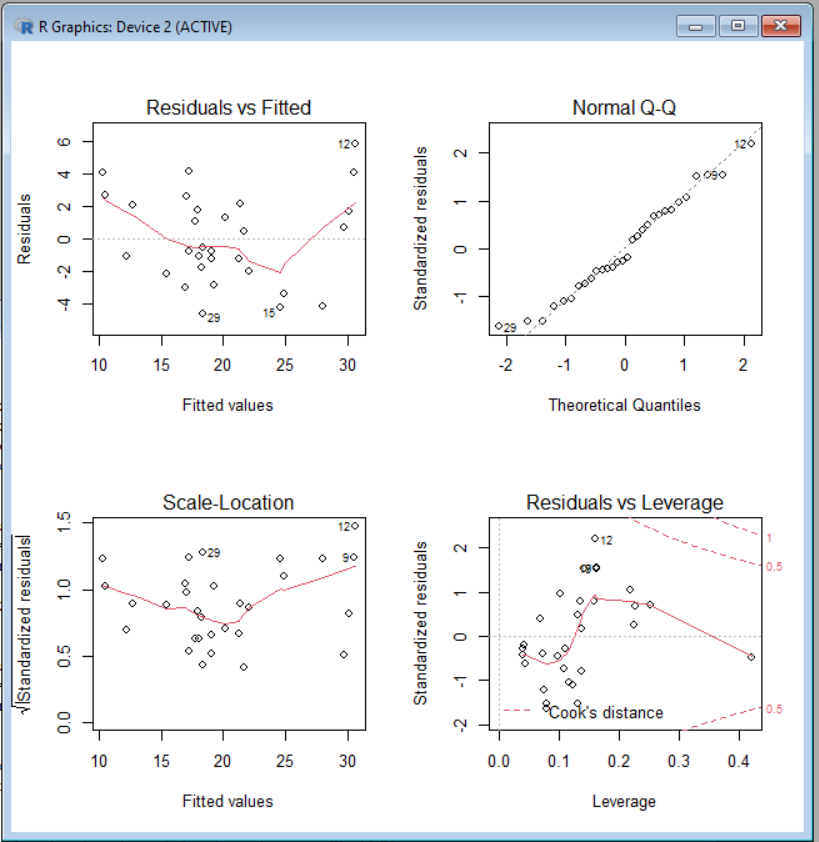
best\_model <- step(init\_model, direction = "both")

summary(best\_model)

par(mfrow = c(2, 2))

plot(best\_model)





As above, we know that x5,x8,x10 will get the better model

Thus, try x5^3, x8^3, x10^3 in following.

(2) **y ~origin data + x5^3, x8^3, x10^3**

**# AIC=40.95 ,y ~ x2 + x3 + x5 + x7 + x8 + x10 + V13 + V14,Ra = 0.9218**

df3=cbind(df,,df[,6]^3,df[,9]^3,df[,11]^3)

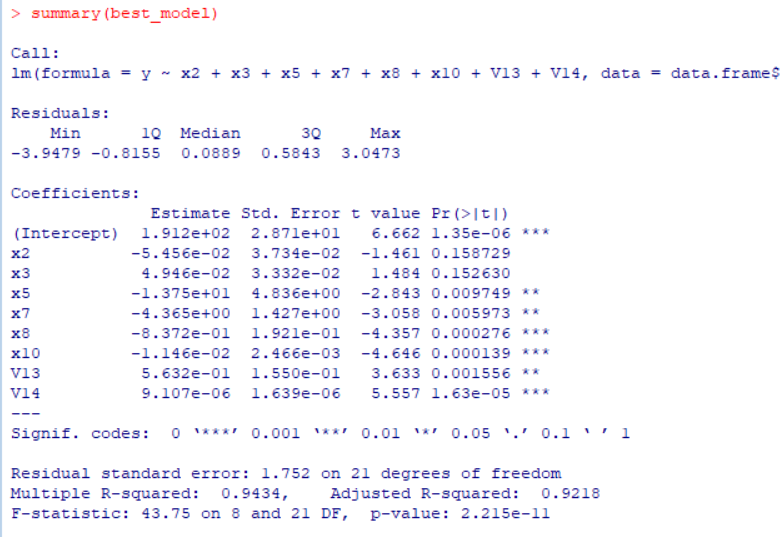
init\_model <- lm(y ~ ., data = data.frame(df3))

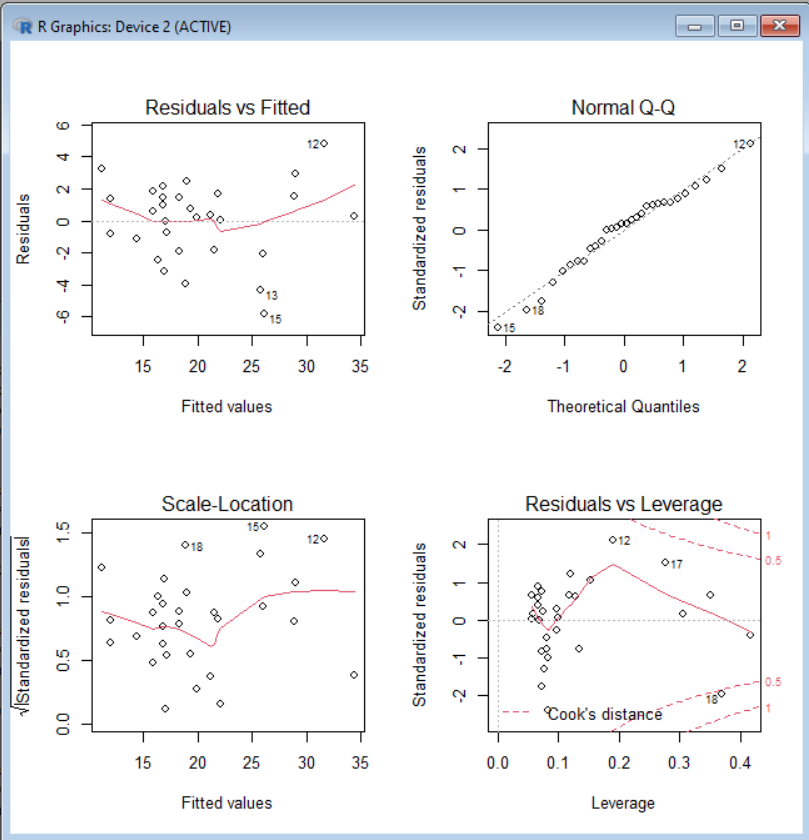
best\_model <- step(init\_model, direction = "both")

summary(best\_model)

par(mfrow = c(2, 2))

plot(best\_model)





However, it gets a little overfitting.

(3) **y ~ origin data + x5,x8,x10 Add weight**

**# AIC=68.29, y ~x5 + x8 + x10, Ra = 0.7808**

df3=cbind(df,3\*df[,6],7\*df[,9],10\*df[,11])

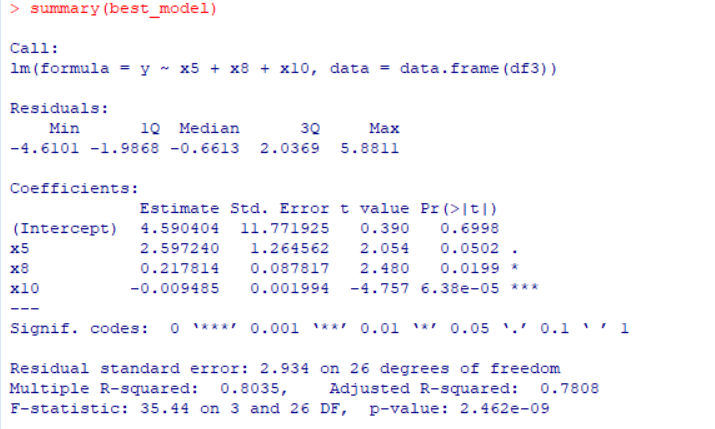
init\_model <- lm(y ~ ., data = data.frame(df3))

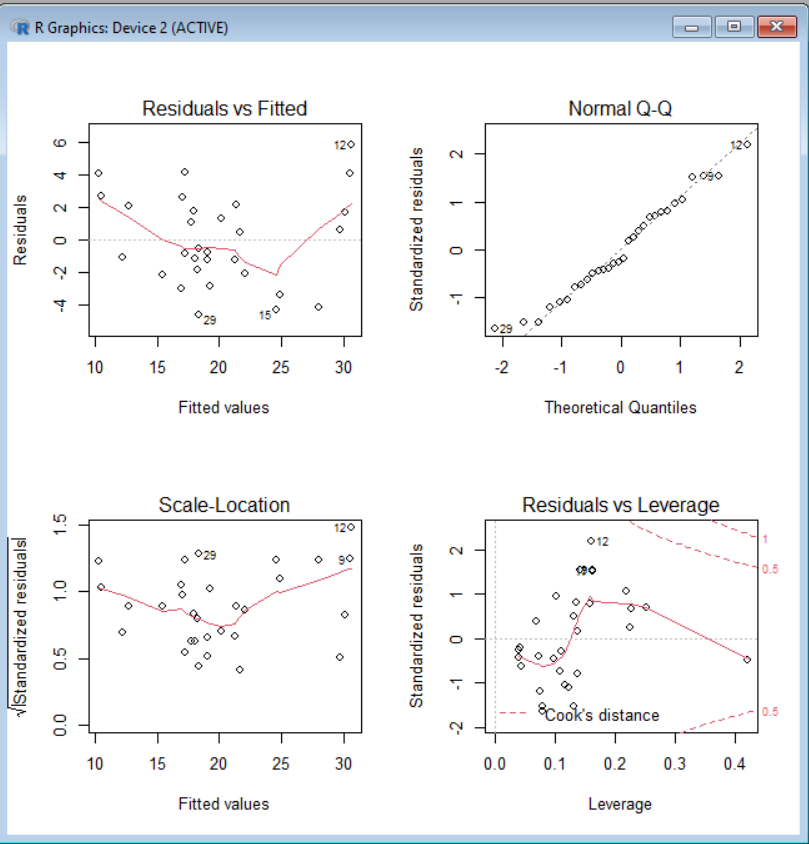
best\_model <- step(init\_model, direction = "both")

summary(best\_model)

par(mfrow = c(2, 2))

plot(best\_model)





Adjust-R square doesn’t change whether x5,x8,x10 has added weight or not

1. **log**

**(1) log(y) ~ log(origin data[,2；11])**

**# AIC=-130.39, get log(y) ~ x1 + x5 + x8 + x10,Ra=0.8661**

df2=log(df[,2:11])

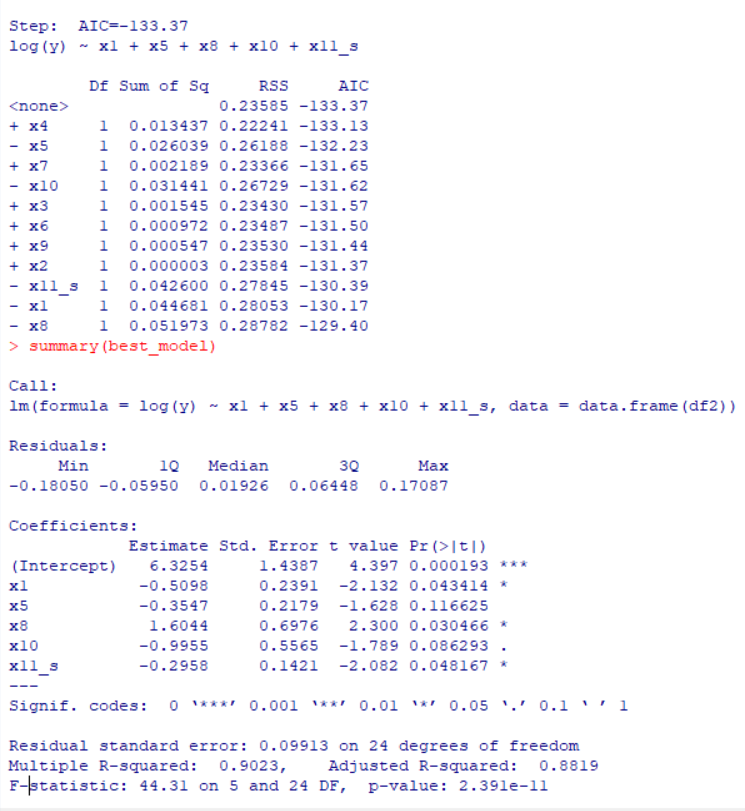
init\_model <- lm(log(y) ~ ., data = data.frame(df2))

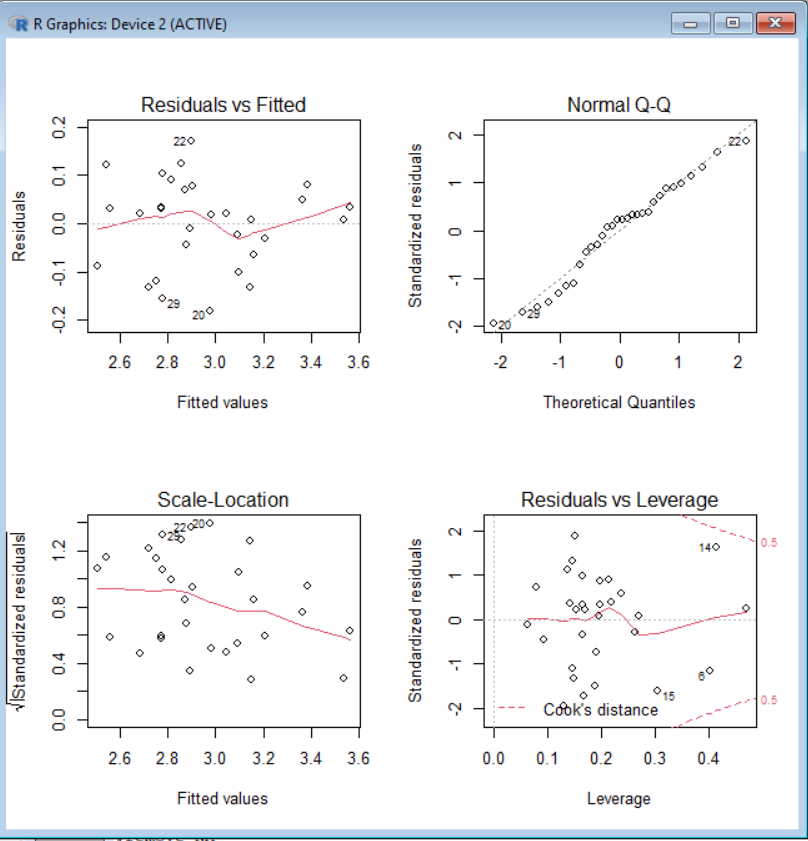
best\_model <- step(init\_model, direction = "both")

summary(best\_model)

par(mfrow = c(2, 2))

plot(best\_model)





(2) **log(y) ~ log(df[,2]) + log(df[,9])^4 + log(df[,11])^4 + df[,12]^3**

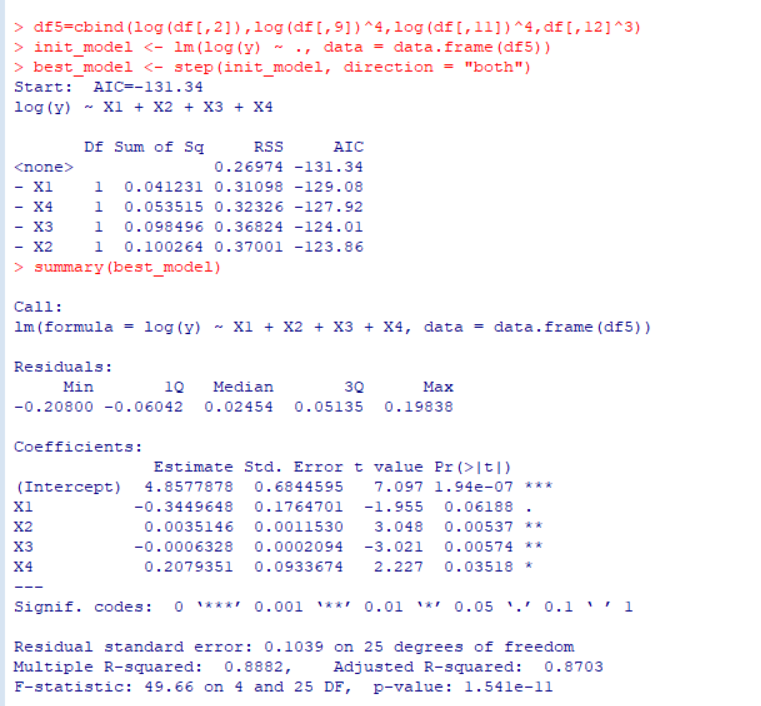
**# AIC=-131.34 log(y) ~ X1 + X2 + X3 + X4 Ra=0.875**

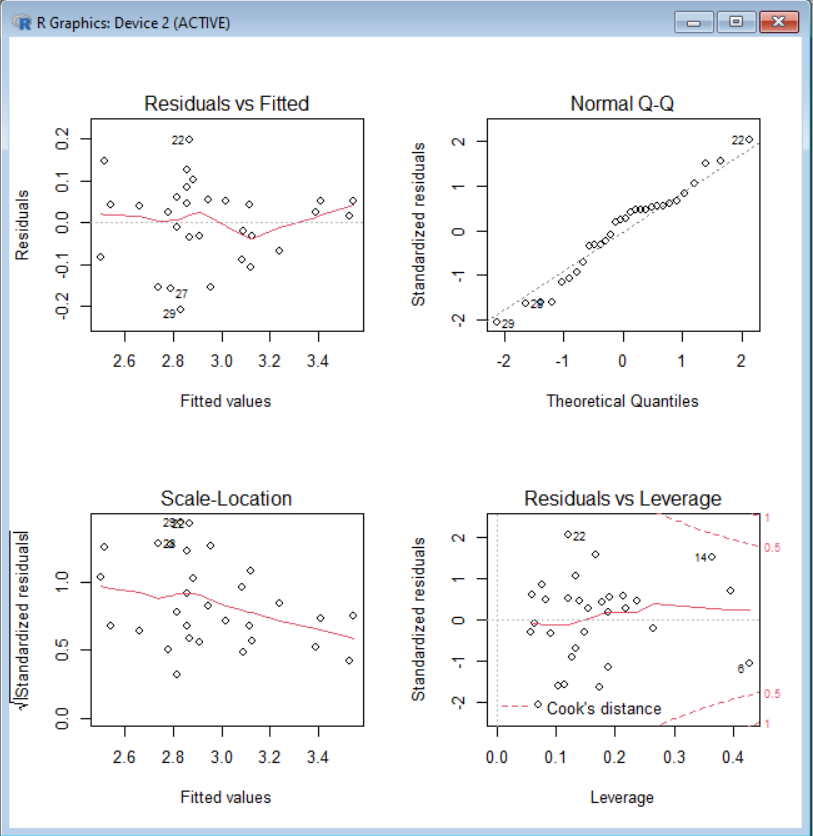
df5=cbind(log(df[,2]),log(df[,9])^4,log(df[,11])^4,df[,12]^3)

init\_model <- lm(log(y) ~ ., data = data.frame(df5))

best\_model <- step(init\_model, direction = "both")

summary(best\_model)





**3. sigmoid**

**(1) y ~ log(df[,2:11]) + sigmoid(x11)**

**AIC=-133.37 log(y) ~ x1 + x5 + x8 + x10 + x11\_s Ra=0.8819**

x11\_s=1/1+exp(-df[,12])

df2=cbind(log(df[,2:11]),x11\_s)

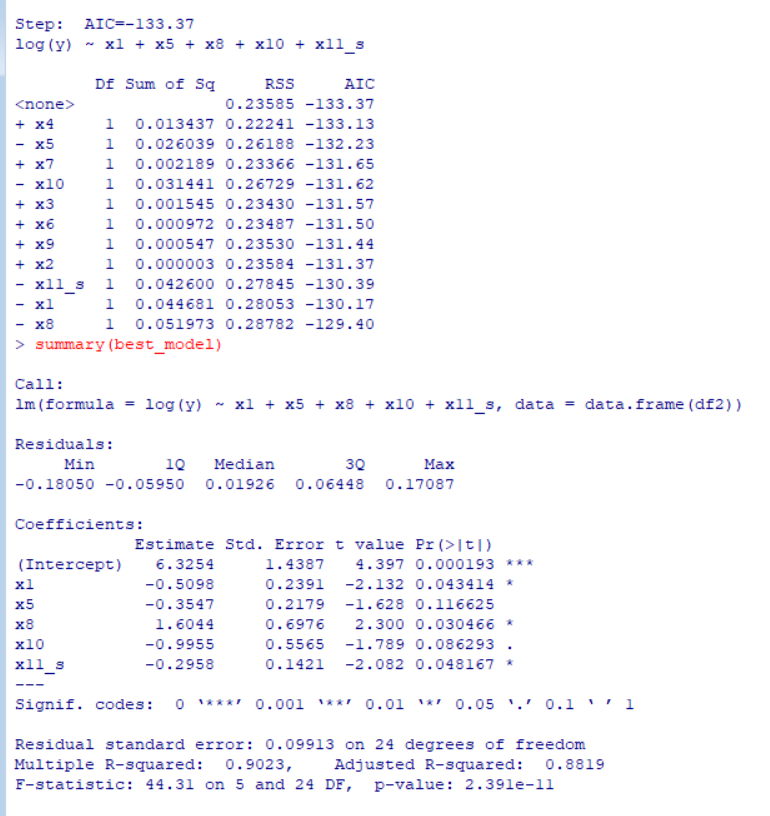
init\_model <- lm(log(y) ~ ., data = data.frame(df2))

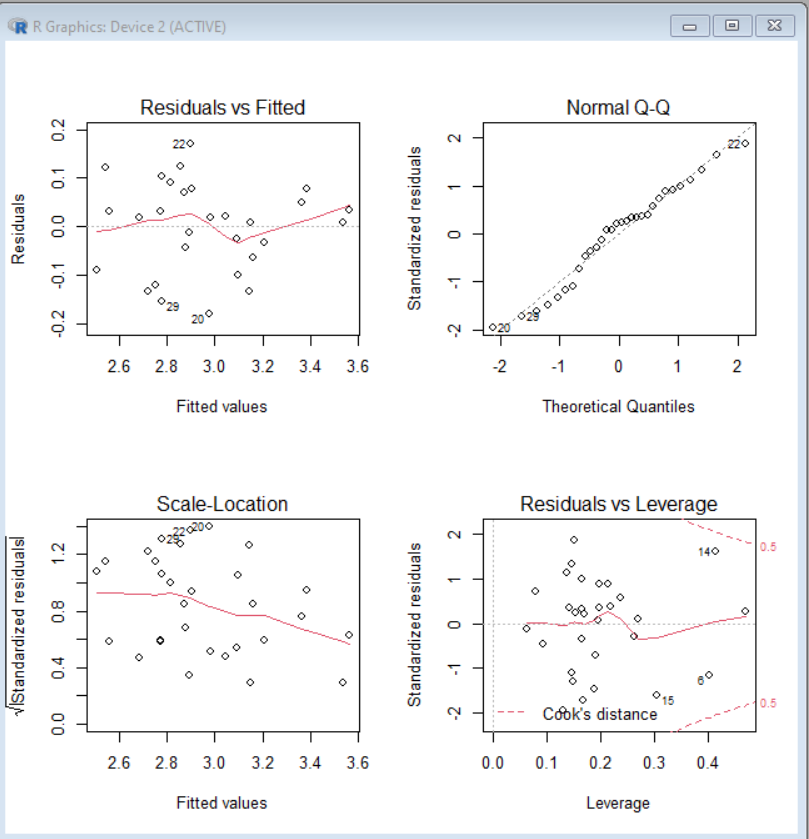
best\_model <- step(init\_model, direction = "both")

summary(best\_model)

par(mfrow = c(2, 2))

plot(best\_model)





**(2) origin data +1/exp(x6,x8,x10)**

**AIC=45.25 y ~ x4 + x6 + x8 + x10 + x11 + x10\_s Ra=0.9059**

x6\_s=1/exp(-df[,7])

x8\_s=1/exp(-df[,9])

x10\_s=1/exp(-df[,11]/sum(df[,11]))

df2=cbind(df,x6\_s,x8\_s,x10\_s)

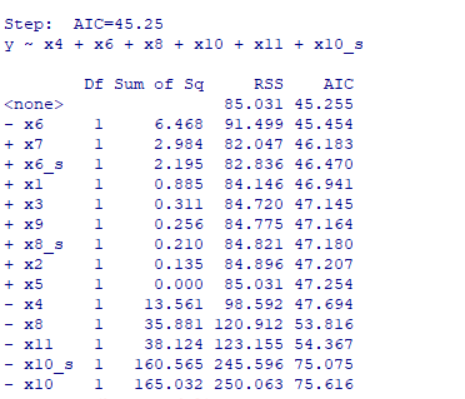
init\_model <- lm(y ~ ., data = data.frame(df2))

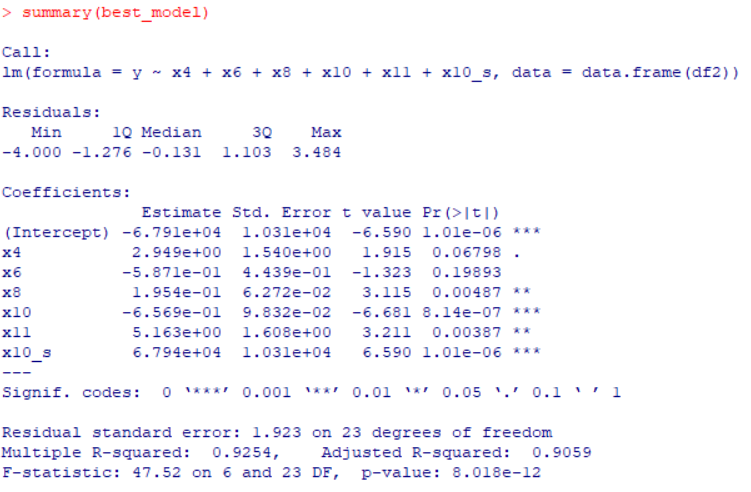
best\_model <- step(init\_model, direction = "both")

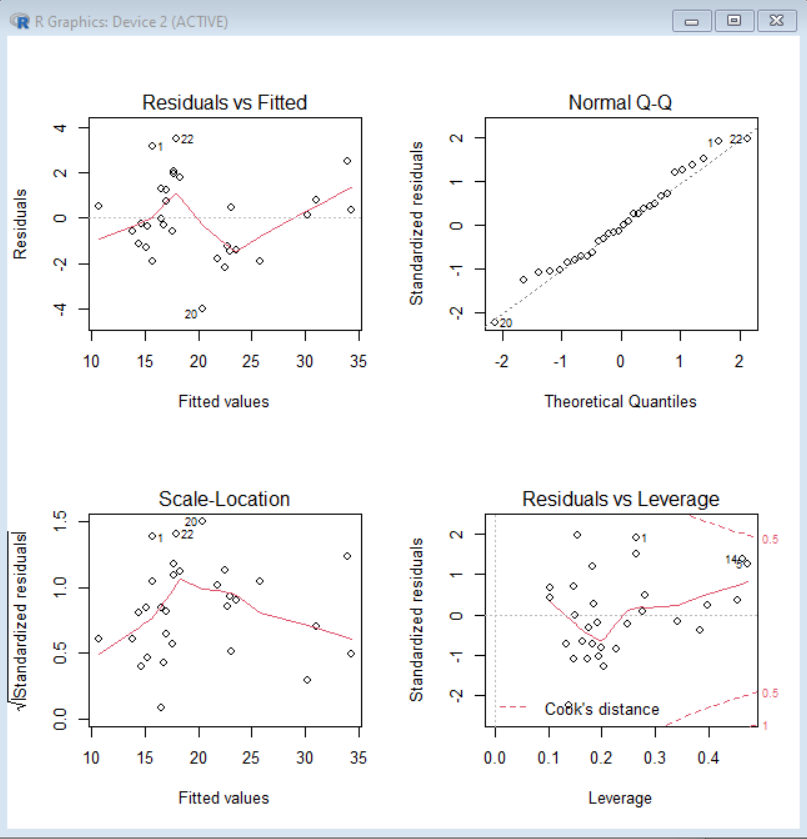
summary(best\_model)

par(mfrow = c(2, 2))

plot(best\_model)







**(3) origin data + sigmoid(x6,x8,x10)**

**#AIC=44.25 y ~ x4 + x8 + x10 + x11 + x6\_s + x10\_s Ra=0.9059**

x6\_s=1/1+exp(-df[,7])

x8\_s=1/1+exp(-df[,9])

x10\_s=1/1+exp(-df[,11]/sum(df[,11]))

df3=cbind(df,x6\_s,x8\_s,x10\_s)

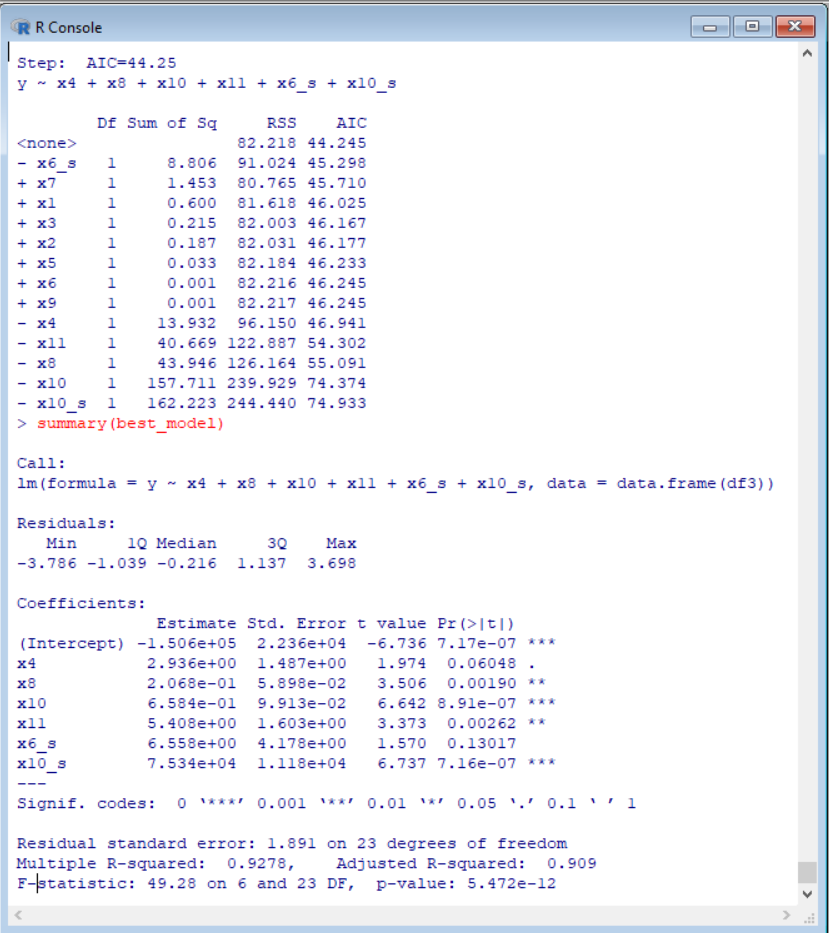
init\_model <- lm(y ~ ., data = data.frame(df3))

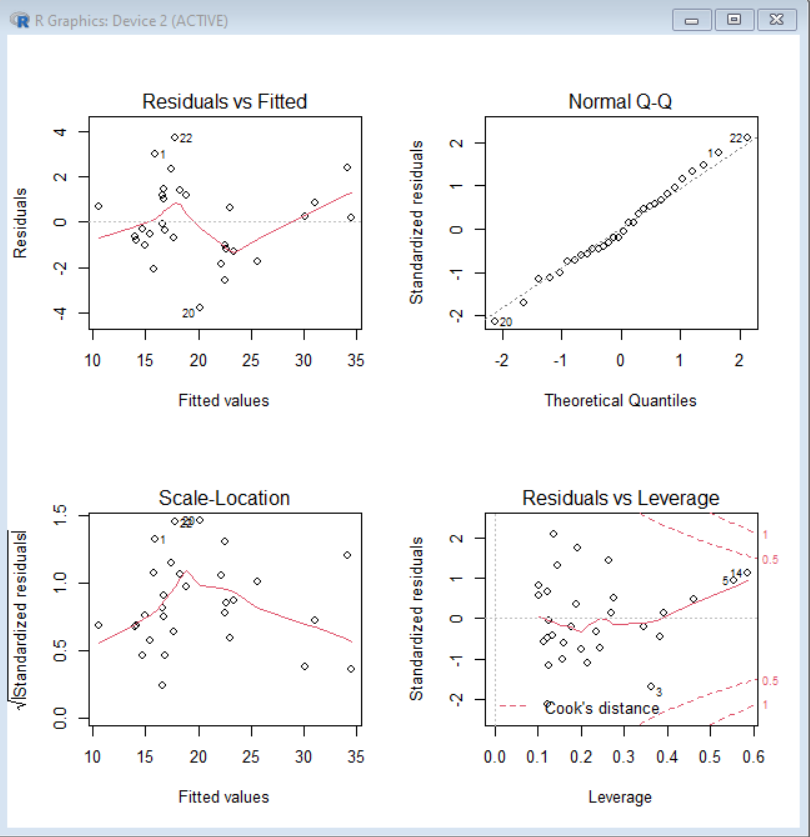
best\_model <- step(init\_model, direction = "both")

summary(best\_model)

par(mfrow = c(2, 2))

plot(best\_model)





**4.cross**

**(1) Let each column data multiple with other column of data**

best\_Ra=0

df2=0

for(i in 2:12){

for(j in 2:12){

df22=(df[,i]\*df[,j])/max(df[,i]\*df[,j])

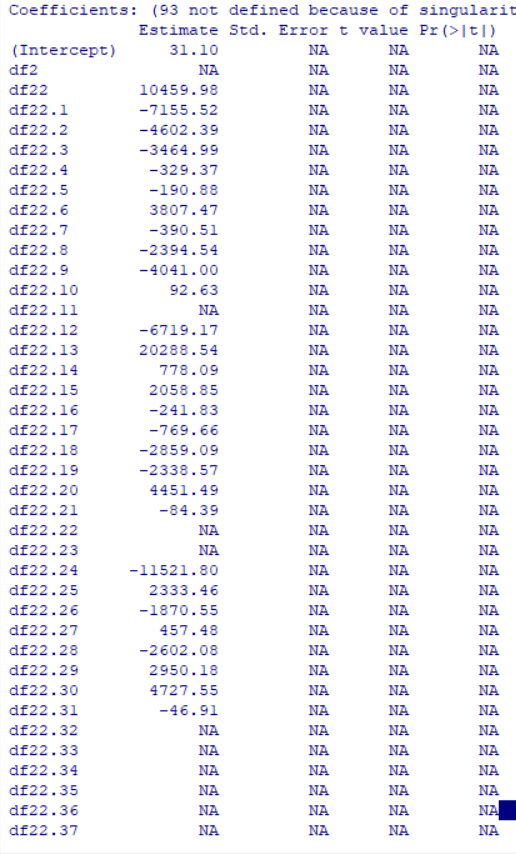
df2=cbind(df2,df22)

}

}

init\_model <- lm(y ~ ., data = data.frame(df2),na.action=na.omit)

however, number of features >> number of data, so it will predict NAs



**(2) Let every12 columns of these result data be input data to fit the lm**

**# Ra = 0.873**

Get the best model of these crossed data

df2=0

for(i in 2:12){

for(j in 2:12){

df22=(df[,i]\*df[,j])/max(df[,i]\*df[,j])

df2=cbind(df2,df22)

}

}

df3=df2[,-1]

best\_Ra=0

for(i in 0:9){

idx1=12\*i+1

idx2=12\*(i+1)

df2=df3[,idx1:idx2]

init\_model <- lm(y ~ ., data = data.frame(df2),na.action=na.omit)

best\_model <- step(init\_model, direction = "both")

Ra=summary(best\_model)$adj.r.squared

if (Ra>best\_Ra){

best\_Ra = Ra

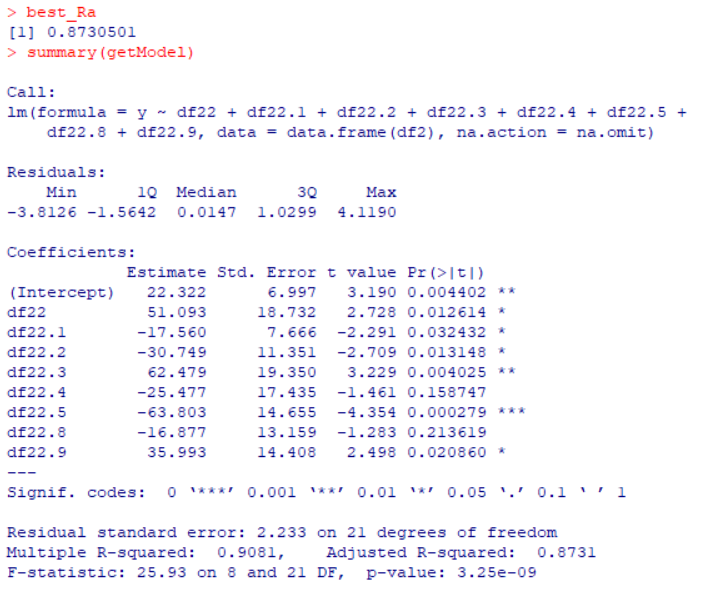
getModel = best\_model}}

best\_Ra

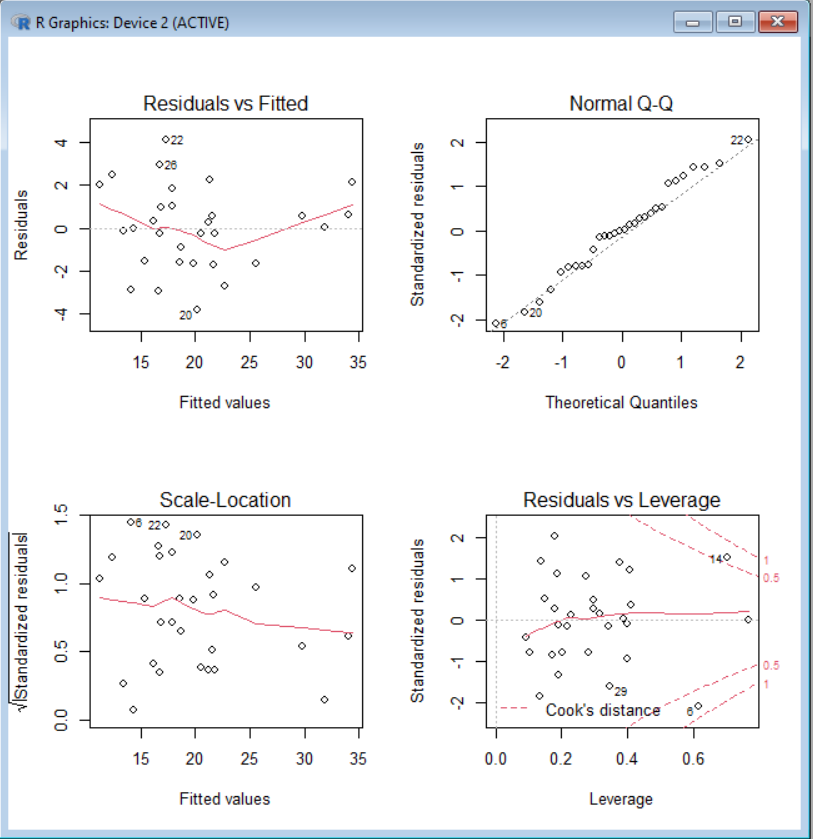
summary(getModel)

par(mfrow = c(2, 2))

plot(getModel)



However, it is a little overfitting.



**5. Mixture of the method**

(1) **sigmoid + cross, Ra = 0.6192753**

df2=0

for(i in 2:12){

for(j in 2:12){

df22=1/1+exp(-df[,i]\*df[,j])

df2=cbind(df2,df22)

}

}

df3=df2[,-1]

best\_Ra=0

for(i in 0:9){

idx1=12\*i+1

idx2=12\*(i+1)

df2=df3[,idx1:idx2]

init\_model <- lm(y ~ ., data = data.frame(df2),na.action=na.omit)

best\_model <- step(init\_model, direction = "both")

Ra=summary(best\_model)$adj.r.squared

if (Ra>best\_Ra){

best\_Ra = Ra

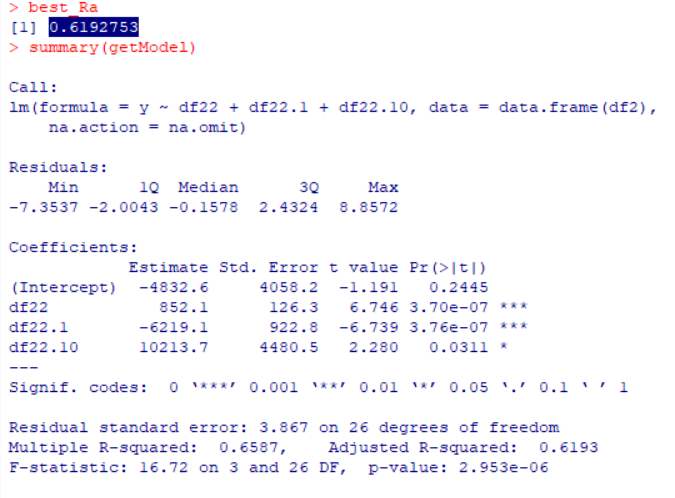
getModel = best\_model}}

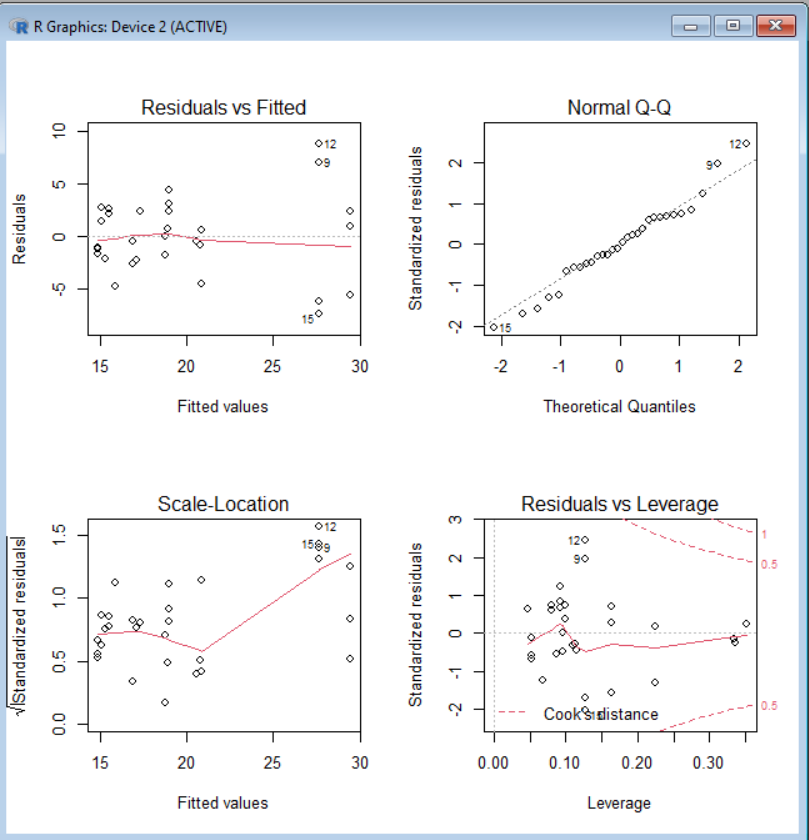
best\_Ra

summary(getModel)

par(mfrow = c(2, 2))

plot(getModel)





**(2) log + cross, Ra = 0.879**

df2=0

for(i in 2:11){

for(j in 2:11){

df22=log(df[,i]\*df[,j])

df2=cbind(df2,df22)

}

}

df3=df2[,-1]

best\_Ra=0

for(i in 0:9){

idx1=10\*i+1

idx2=10\*(i+1)

df2=df3[,idx1:idx2]

init\_model <- lm(y ~ ., data = data.frame(df2),na.action=na.omit)

best\_model <- step(init\_model, direction = "both")

Ra=summary(best\_model)$adj.r.squared

if (Ra>best\_Ra){

best\_Ra = Ra

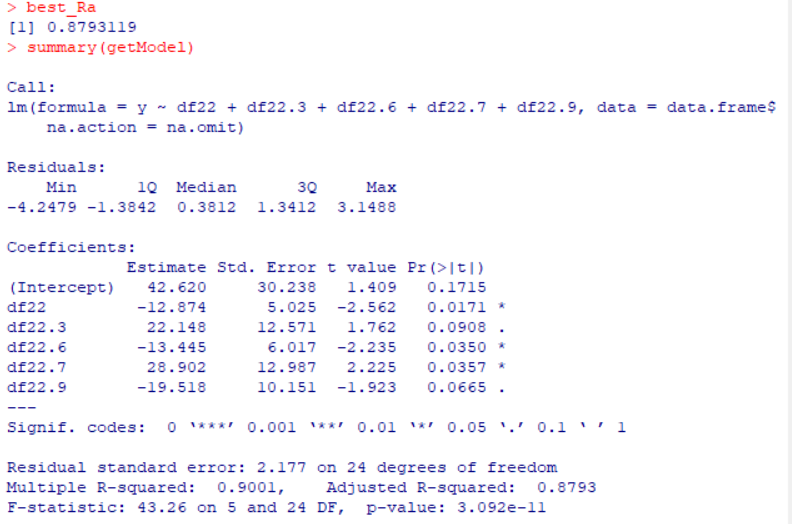
getModel = best\_model}}

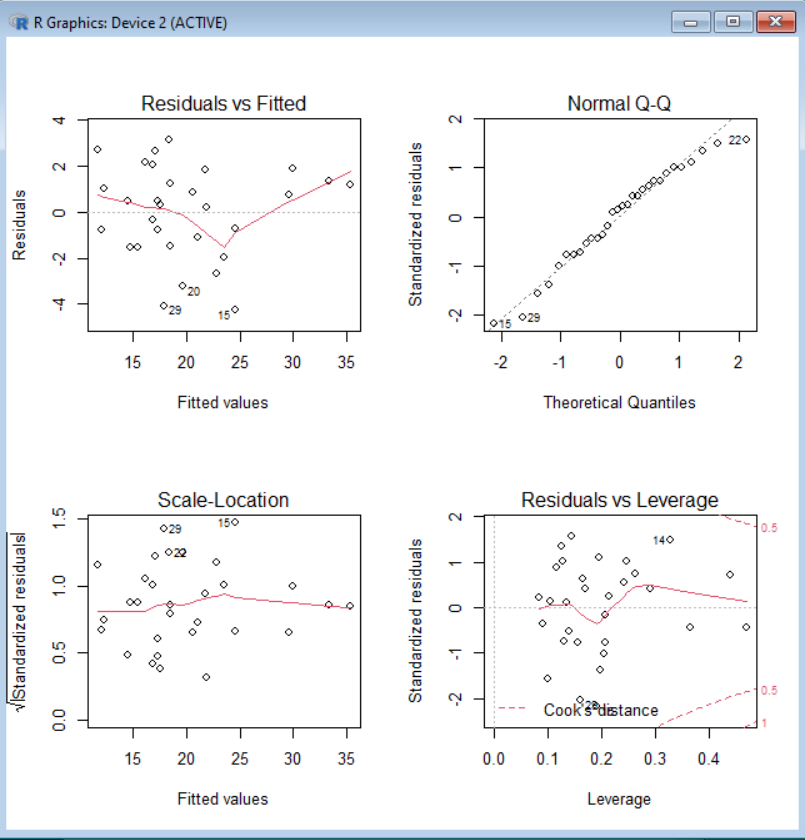
best\_Ra

summary(getModel)

par(mfrow = c(2, 2))

plot(getModel)





**III. Conclusion**

1. In these model, p-value is small, so cannot reject H0

2. In these model, the points on Normal QQ plot is almost a line. Thus, we can say that residual ~ normal distribution

3. The best model is y ~origin data + x5^3, x8^3, x10^3

Its AIC=40.95

Its result is y ~ x2 + x3 + x5 + x7 + x8 + x10 + V13 + V14

Its adjusted R\_square is 0.9218

However, I think it’s a little overfitting for eight variable in the model.

Thus, I consider the first best model as y ~ origin data + sigmoid(x6,x8,x10)

Its AIC=44.25

Its result is y ~ x4 + x8 + x10 + x11 + x6\_s + x10\_s

Its adjusted R\_square is 0.9059

the second best model as y ~ origin data +1/exp(x6,x8,x10)

Its AIC=45.25

It result is y ~ x4 + x6 + x8 + x10 + x11 + x10\_s

Its adjusted R\_square is 0.9059