STAT425_HW5_Jinran Yang

Jinran Yang

11/9/2018

1.

```r
library(faraway)
library(MASS)
attach(sat)
names(sat)
```

```
## [1] "expend" "ratio"  "salary" "takers" "verbal" "math"   "total"
```

```r
lm<-lm(total~takers+ratio+salary,data = sat)
summary(lm)
```

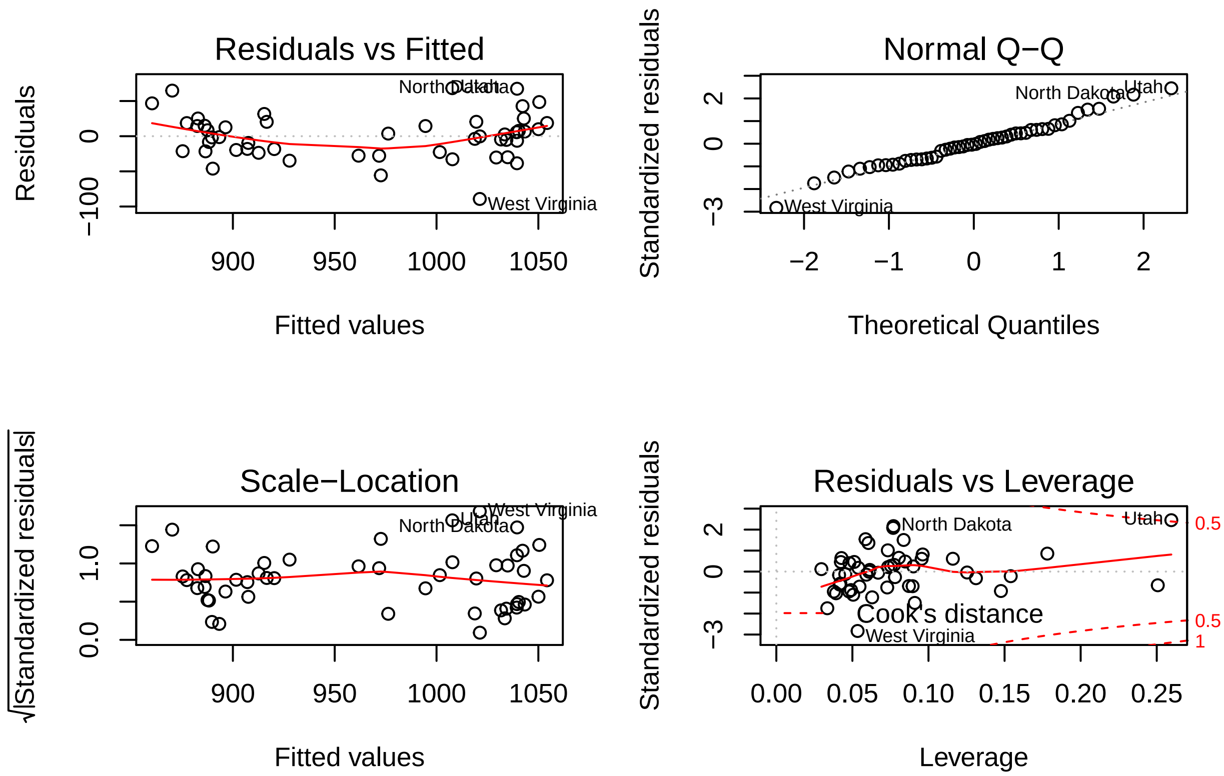
```
##
## Call:
## lm(formula = total ~ takers + ratio + salary, data = sat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -89.244 -21.485  -0.798  17.685  68.262
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1057.8982    44.3287  23.865   <2e-16 ***
## takers        -2.9134     0.2282 -12.764   <2e-16 ***
## ratio         -4.6394     2.1215  -2.187   0.0339 *
## salary         2.5525     1.0045   2.541   0.0145 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 32.41 on 46 degrees of freedom
## Multiple R-squared:  0.8239, Adjusted R-squared:  0.8124
## F-statistic: 71.72 on 3 and 46 DF,  p-value: < 2.2e-16
```

```r
rlm<-rlm(total~takers+ratio+salary,data = sat)
summary(rlm)
```

```
##
## Call: rlm(formula = total ~ takers + ratio + salary, data = sat)
## Residuals:
##       Min        1Q    Median        3Q       Max
## -91.25763 -18.04967   0.06361  15.80909  79.48664
##
## Coefficients:
##             Value     Std. Error t value
## (Intercept) 1070.5209   42.4863    25.1968
## takers        -2.9773     0.2188   -13.6101
## ratio         -6.0178     2.0333    -2.9596
## salary         2.8930     0.9628     3.0049
```

```
##
## Residual standard error: 26.72 on 46 degrees of freedom
```

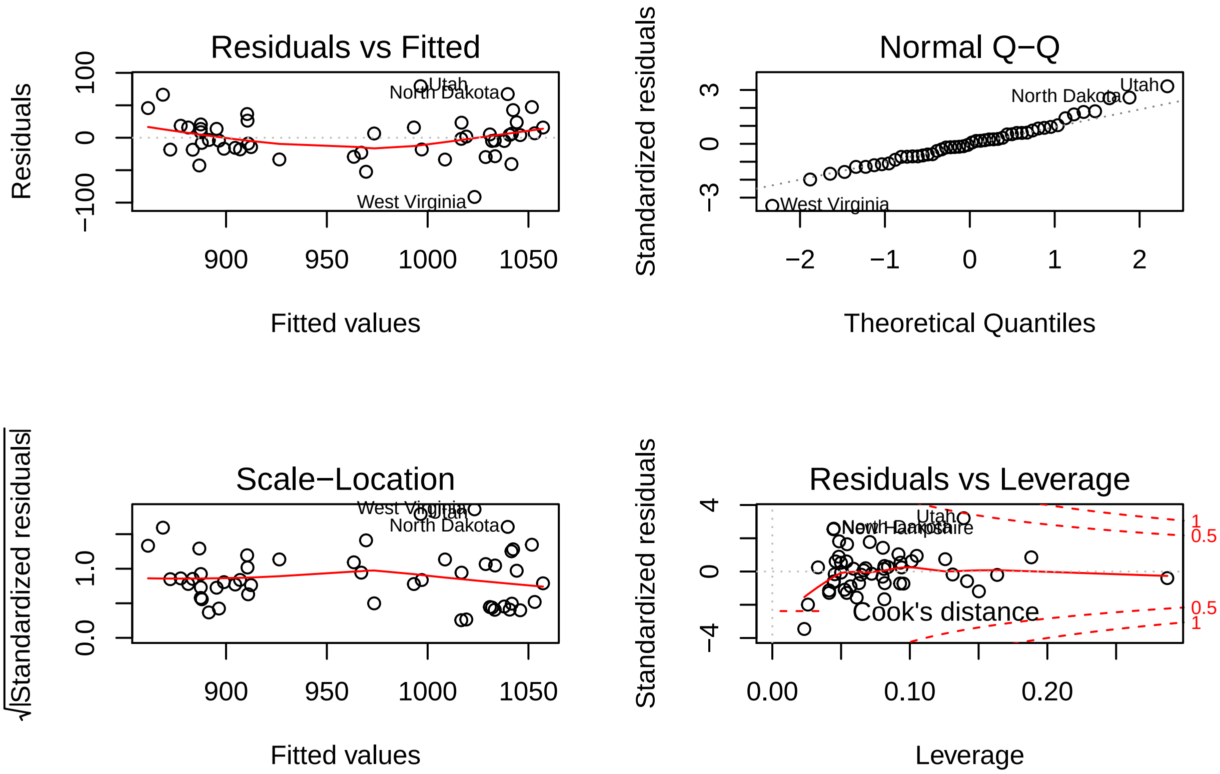
```r
par(mfrow=c(2,2))
plot(lm)
```




```r
which(cooks.distance(lm)>=1)
```

```
## named integer(0)
```

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




```r
which(cooks.distance(rlm)>=1)
```

```
## named integer(0)
```

As we can see, the result of two method are very close to each other. The coefficients estimated by two models are quite similiar to each other,

```r
#Huber's robust regression, takers
pt(13.6101,46,lower.tail=F)
```

```
## [1] 4.823055e-18
```

```r
#Huber's robust regression, ratio
pt(2.9596,46,lower.tail=F)
```

```
## [1] 0.002427596
```

```r
#Huber's robust regression, salary
pt(3.0049,46,lower.tail=F)
```

```
## [1] 0.002145188
```

As we can see from above, `takers`, `ratio` and `salary` are all significent (their P-value are all smaller than 0.05).

```r
rss <- sum((total - rlm$fitted.values) ^ 2)  # residual sum of squares
tss <- sum((total - mean(total)) ^ 2)  # total sum of squares
rsq <- 1 - rss/tss
rsq
```

```
## [1] 0.8218491
```

The R^2 of Huber's robust regression result is 0.8218491 which is quite close to that of the ordinary least squares (0.8239); Based on the cook's distance, there is no influential point in both model. Overall, the results obtained by two models are similar.

2.

```r
#(a)
set.seed(1)
I<-diag(5)
J<-matrix(1,5,5)
Sigma<-0.8*I+0.2*J
Sigma
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]  1.0  0.2  0.2  0.2  0.2
## [2,]  0.2  1.0  0.2  0.2  0.2
## [3,]  0.2  0.2  1.0  0.2  0.2
## [4,]  0.2  0.2  0.2  1.0  0.2
## [5,]  0.2  0.2  0.2  0.2  1.0
```

```r
X<-mvrnorm(n=100,rep(0,5),Sigma)
dim(X)
```

```
## [1] 100   5
```

```r
#(b)
A_element<-vector(length = 100)
for (i in 1:100){
 A_element[i]<-sqrt(i)
}
A<-diag(x = A_element,nrow = 100,ncol = 100)

set.seed(1)
error<-mvrnorm(n=1,mu=rep(0,100),A )

beta<-matrix(c(1,-1,1,-1,1),ncol = 1)

Y=1+X %*% beta +error

#least square
beta_LS<-solve(t(X)%*%X)%*%t(X)%*%Y
beta_GLS<-solve(t(X)%*%solve(A)%*%X)%*%t(X)%*%solve(A)%*%Y

MSE<-function(x,beta){
  MSE1<-0
  for (i in 1:length(x)){
    MSE1<-MSE1+(x[i]-beta[i])^2
  }
  return(MSE1*(1/length(x)))
}

MSE(beta_LS,beta)#least square
```

```
## [1] 0.01717338
```

```r
MSE(beta_GLS,beta)#generalized least square
```

```
## [1] 0.02370031
```

As we can see, the mean square error of beta estimated by generalized least squares is larger than that of beta estimated by the least squares.

```
set.seed(1)
X_10<-list()
for (i in 1:10){
  X_10[[i]]<-mvrnorm(n=100,rep(0,5),Sigma,empirical = FALSE)
}

error_10<-list()
for (i in 1:10){
  error_10[[i]]<-mvrnorm(n=1,mu=rep(0,100),A )
}

Y_10<-list()
for (i in 1:10){
  Y_10[[i]]<-1+ X_10[[i]] %*% beta +error_10[[i]]
}


beta_LS_10<-list()
for (i in 1:10){
  beta_LS_10[[i]]<-solve(t(X_10[[i]])%*%X_10[[i]])%*%t(X_10[[i]])%*%Y_10[[i]]
}

beta_GLS_10<-list()
for (i in 1:10){
  beta_GLS_10[[i]]<-solve(t(X_10[[i]])%*%solve(A)%*%X_10[[i]])%*%t(X_10[[i]])%*%solve(A)%*%Y_10[[i]]
}

MSE_LS<-vector(length = 10)
for (i in 1:10){
  MSE_LS[i]<-MSE(beta_LS_10[[i]],beta)
}

AMSE_LS<-mean(MSE_LS)
AMSE_LS#least square
```

```
## [1] 0.08150759
```

```
MSE_GLS<-vector(length = 10)
for (i in 1:10){
  MSE_GLS[i]<-MSE(beta_GLS_10[[i]],beta)
}

AMSE_GLS<-mean(MSE_GLS)
AMSE_GLS#generalized least square
```

```
## [1] 0.06889253
```

As we can see from the result above, the average mean squared error (over ten synthetic dataset) of beta estimated by generalized least squares is smaller than that of the beta estimated by least squares. Therefore, the generalized least squares performs better when it comes to the variance of error is not equal and the errors are uncorrelated.

```r
J1<-matrix(1,100,100)
B<-A+0.1*J1

set.seed(1)
error1<-mvrnorm(n=1,mu=rep(0,100),B )

Y1=1+ X %*% beta +error1

#least square
beta_LS1<-solve(t(X)%*%X)%*%t(X)%*%Y1
beta_GLS1<-solve(t(X)%*%solve(B)%*%X)%*%t(X)%*%solve(B)%*%Y1

MSE(beta_LS1,beta)#least square
```

```
## [1] 0.1191465
```

```r
MSE(beta_GLS1,beta)#generalized least square
```

```
## [1] 0.06654401
```

As we can see, the mean square error of beta estimated by generalized least squares is much more smaller than that of beta estimated by the least squares.

```r
set.seed(1)

error_10_1<-list()
for (i in 1:10){
  error_10_1[[i]]<-mvrnorm(n=1,mu=rep(0,100),B)
}

Y_10_1<-list()
for (i in 1:10){
  Y_10_1[[i]]<-1+ X_10[[i]] %*% beta +error_10_1[[i]]
}



beta_LS_10_1<-list()
for (i in 1:10){
  beta_LS_10_1[[i]]<-solve(t(X_10[[i]])%*%X_10[[i]])%*%t(X_10[[i]])%*%Y_10_1[[i]]
}

beta_GLS_10_1<-list()
for (i in 1:10){
  beta_GLS_10_1[[i]]<-solve(t(X_10[[i]])%*%solve(B)%*%X_10[[i]])%*%t(X_10[[i]])%*%solve(B)%*%Y_10_1[[i]
}

MSE_LS1<-vector(length = 10)
for (i in 1:10){
  MSE_LS1[i]<-MSE(beta_LS_10_1[[i]],beta)
}

AMSE_LS1<-mean(MSE_LS1)
AMSE_LS1##least square
```

```
## [1] 0.08732846
```

```r
MSE_GLS1<-vector(length = 10)
for (i in 1:10){
  MSE_GLS1[i]<-MSE(beta_GLS_10_1[[i]],beta)
}

AMSE_GLS1<-mean(MSE_GLS1)
AMSE_GLS1#generalized least square
```

```
## [1] 0.0578809
```

The average mean squared error (over ten synthetic dataset) of beta estimated by generalized least squares is smaller than that of beta estimated by least squares.Therefore, the generalized least squares performs better when it comes to the variance of error is not equal and the error are correlated.

4.

```r
#(a)
model<-lm(happy~.,data=happy)
summary(model)
```

```
##
## Call:
## lm(formula = happy ~ ., data = happy)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.7186 -0.5779 -0.1172  0.6340  2.0651
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.072081   0.852543  -0.085   0.9331
## money        0.009578   0.005213   1.837   0.0749 .
## sex         -0.149008   0.418525  -0.356   0.7240
## love         1.919279   0.295451   6.496 1.97e-07 ***
## work         0.476079   0.199389   2.388   0.0227 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.058 on 34 degrees of freedom
## Multiple R-squared:  0.7102, Adjusted R-squared:  0.6761
## F-statistic: 20.83 on 4 and 34 DF,  p-value: 9.364e-09
```

sex is not significant, since the p-value is larger than 0.05. So let's remove it and refit the model.

```r
model<-lm(happy~money+love+work,data=happy)
summary(model)
```

```
##
## Call:
## lm(formula = happy ~ money + love + work, data = happy)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.62468 -0.57099 -0.08903  0.58675  2.14389
##
```

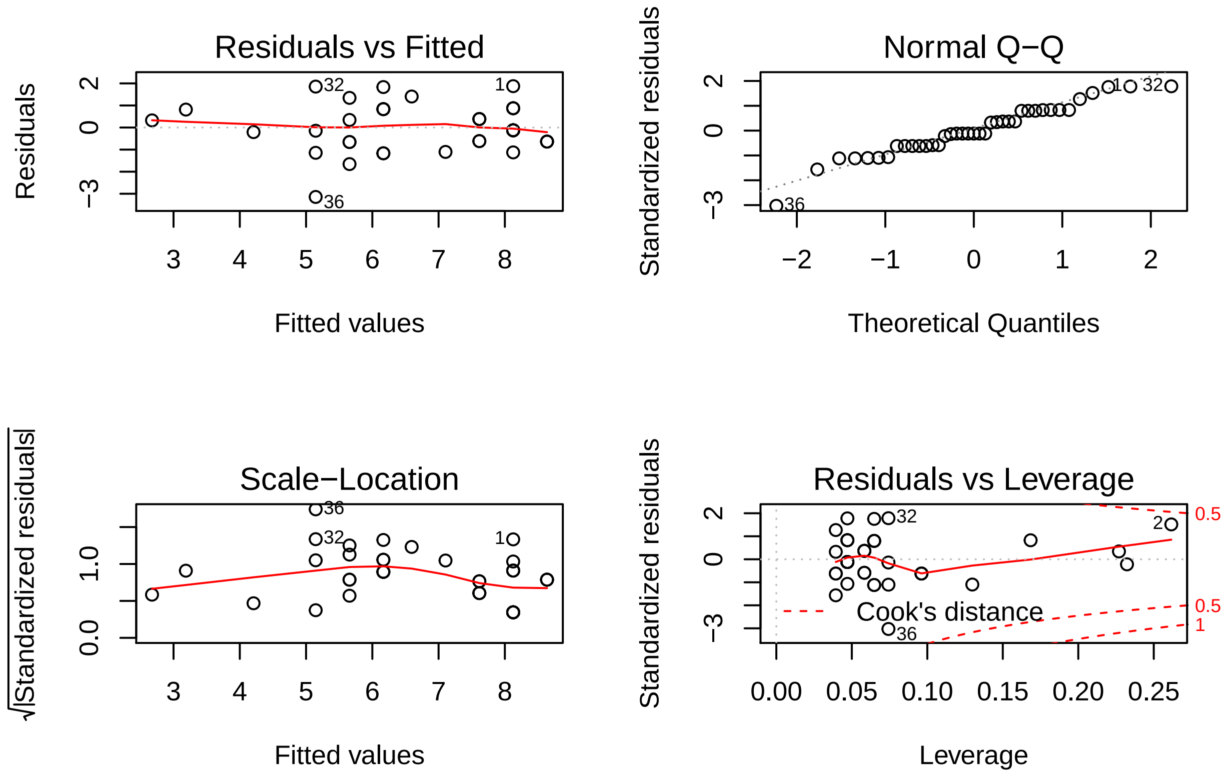
```
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.185936   0.780372  -0.238   0.8131
## money        0.008959   0.004852   1.846   0.0733 .
## love         1.901709   0.287644   6.611 1.22e-07 ***
## work         0.503602   0.181486   2.775   0.0088 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.045 on 35 degrees of freedom
## Multiple R-squared:  0.7091, Adjusted R-squared:  0.6842
## F-statistic: 28.44 on 3 and 35 DF,  p-value: 1.689e-09
```

`money` is not significant, since the p-value is larger than 0.05. So let's remove it and refit the model.

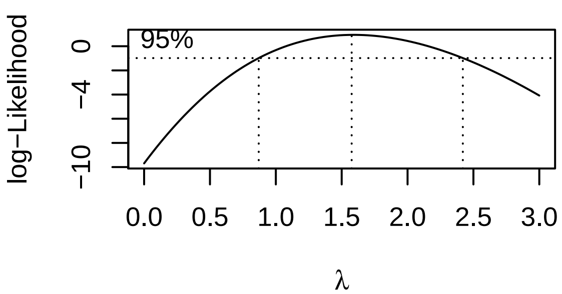
```r
model<-lm(happy~love+work,data=happy)
summary(model)
```

```
##
## Call:
## lm(formula = happy ~ love + work, data = happy)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.1454 -0.6365 -0.1259  0.8333  1.8741
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.2057     0.7757   0.265  0.79241
## love          1.9592     0.2954   6.633 9.99e-08 ***
## work          0.5106     0.1874   2.725  0.00987 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.08 on 36 degrees of freedom
## Multiple R-squared:  0.6808, Adjusted R-squared:  0.6631
## F-statistic: 38.39 on 2 and 36 DF,  p-value: 1.182e-09
```

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




```
boxcox(model, lambda = seq(0, 3,0.1))
```



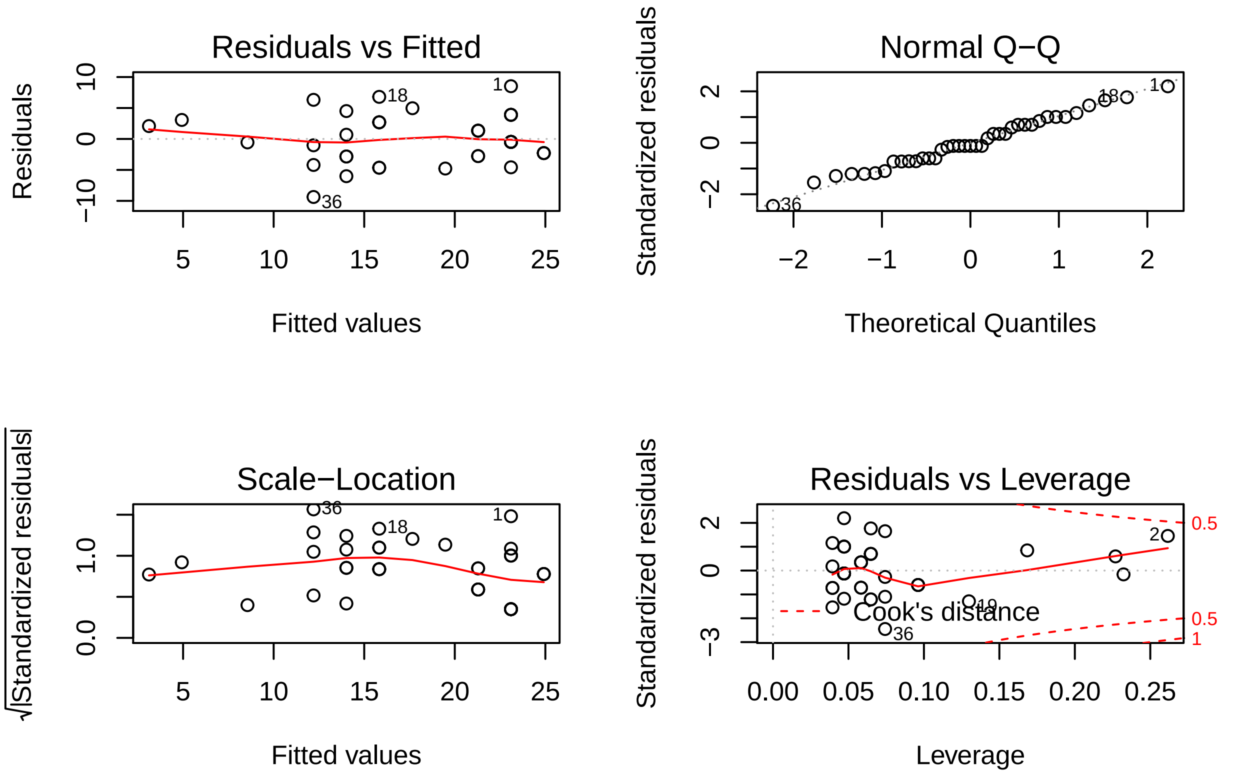
Since 1.5 is inside the 95% CI of lambda, we choose lambda is 1.5 and we fit a new model with happy^1.5 as new dependent variable.

```
newmodel<-lm(happy^1.5~love + work,data=happy)
summary(newmodel)
```

```
##
## Call:
## lm(formula = happy^1.5 ~ love + work, data = happy)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.3739 -2.7684 -0.4739  2.6925  8.5214
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -5.9704     2.8572  -2.090   0.0438 *
## love          7.2736     1.0879   6.686 8.51e-08 ***
## work          1.8127     0.6903   2.626   0.0126 *
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.976 on 36 degrees of freedom
## Multiple R-squared:  0.6801, Adjusted R-squared:  0.6623
## F-statistic: 38.26 on 2 and 36 DF,  p-value: 1.234e-09
```

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

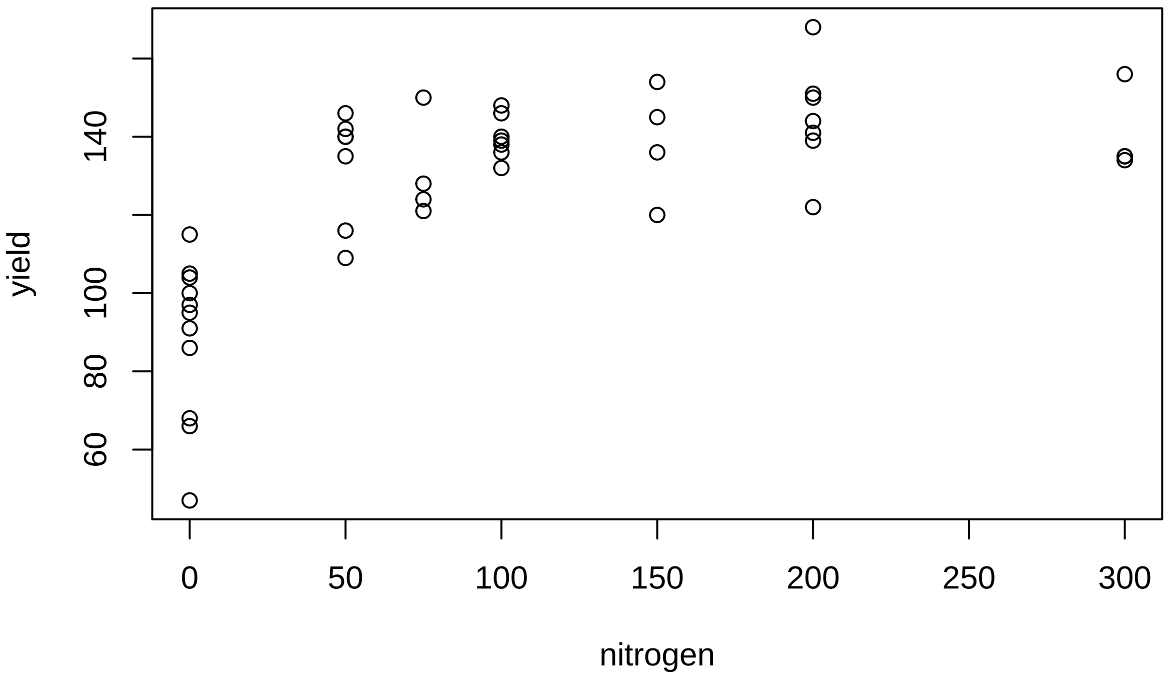


As we can see the Q-Q plot looks more like a straigth line than the previous model. The R^2 is roughly equal to the previous one. And there is no outliers. Therefore, the best model is `happly^1.5 ~ work + love`.

```r
#(b)
attach(cornnit)
names(cornnit)
```

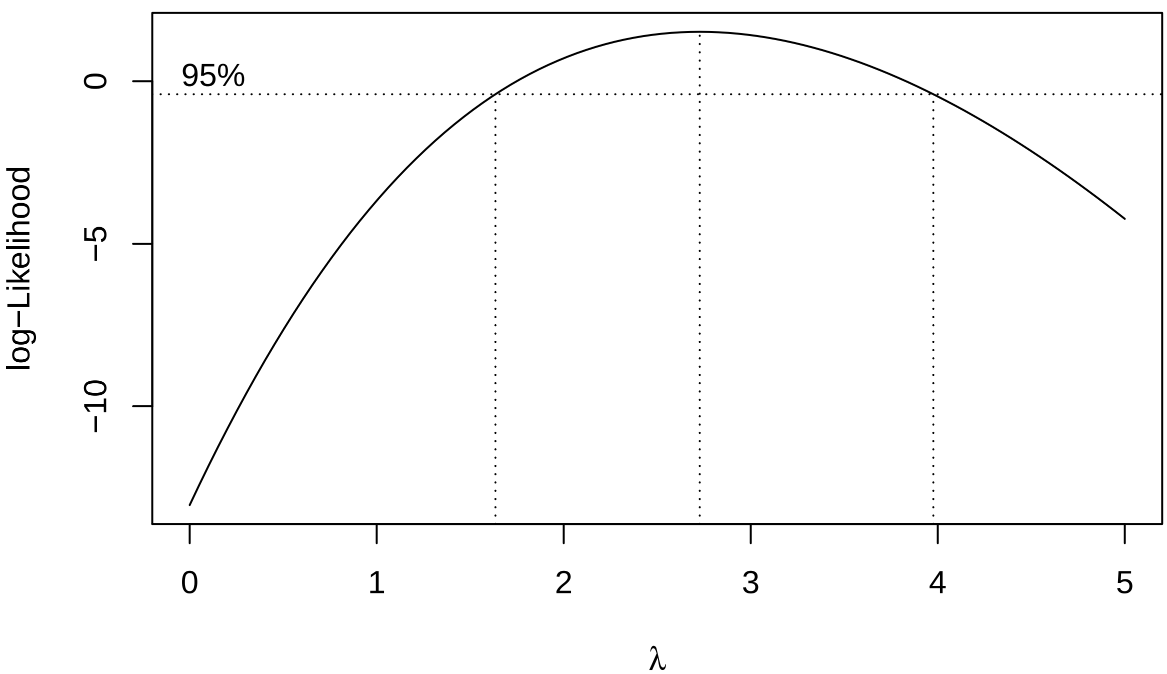
```
## [1] "yield"    "nitrogen"
```

```r
plot(x=nitrogen,y=yield)
```



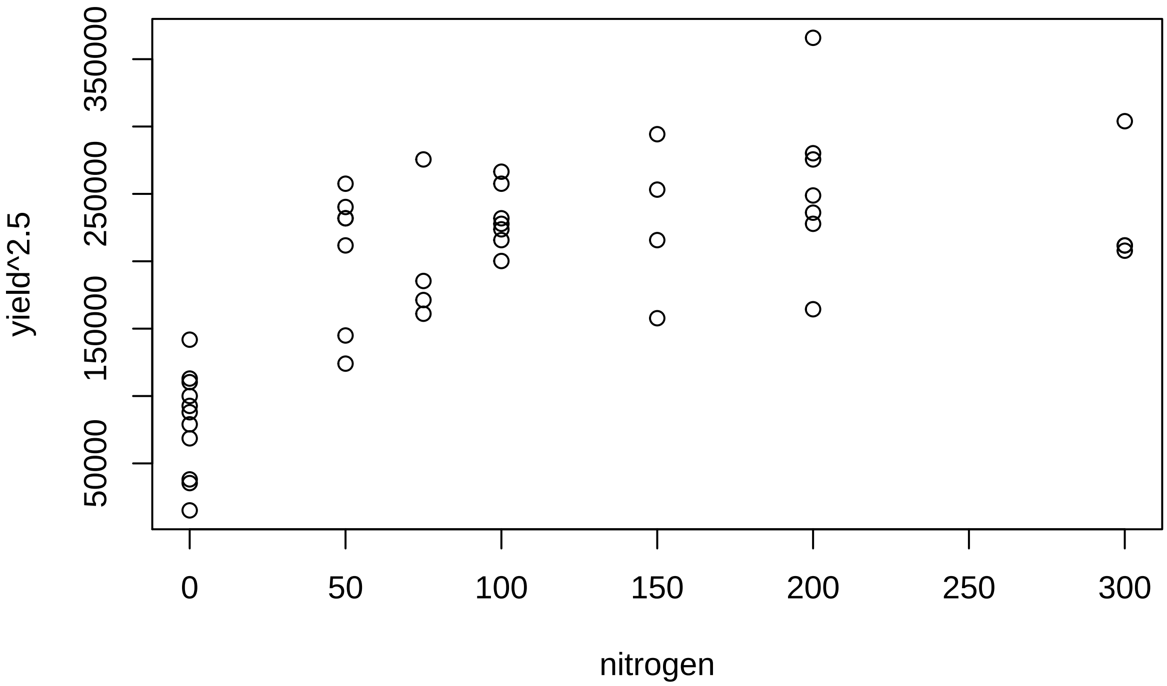
It is not a linear trend so that we try Box-Cox transformation.

```
model_con<-lm(yield~nitrogen)
boxcox(model_con, lambda = seq(0, 5,0.1))
```



Since 2.5 is inside the 95% CI of lambda, we choose lambda is 2.5 and we fit a new model with yield^2.5 as new dependent variable.

```
plot(y=yield^2.5,x=nitrogen)
```

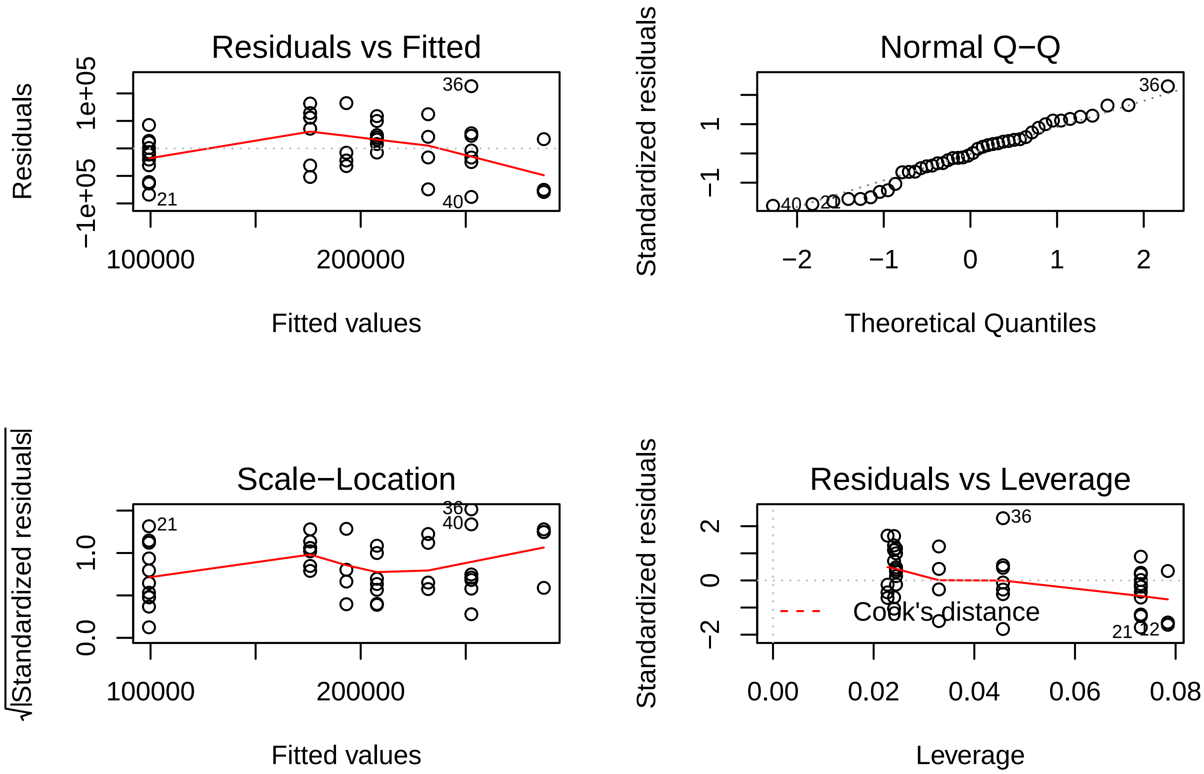


It is still not a linear trend so that we try to transform the predictor.

```
model_sqrt<-lm(yield^2.5~sqrt(nitrogen))
summary(model_sqrt)
```

```
##
## Call:
## lm(formula = yield^2.5 ~ sqrt(nitrogen))
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -88258 -30753  -1537  29595 113167
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)       99250      13647   7.273 5.93e-09 ***
## sqrt(nitrogen)    10848       1342   8.083 4.28e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 50480 on 42 degrees of freedom
## Multiple R-squared:  0.6087, Adjusted R-squared:  0.5994
## F-statistic: 65.34 on 1 and 42 DF,  p-value: 4.282e-10
```

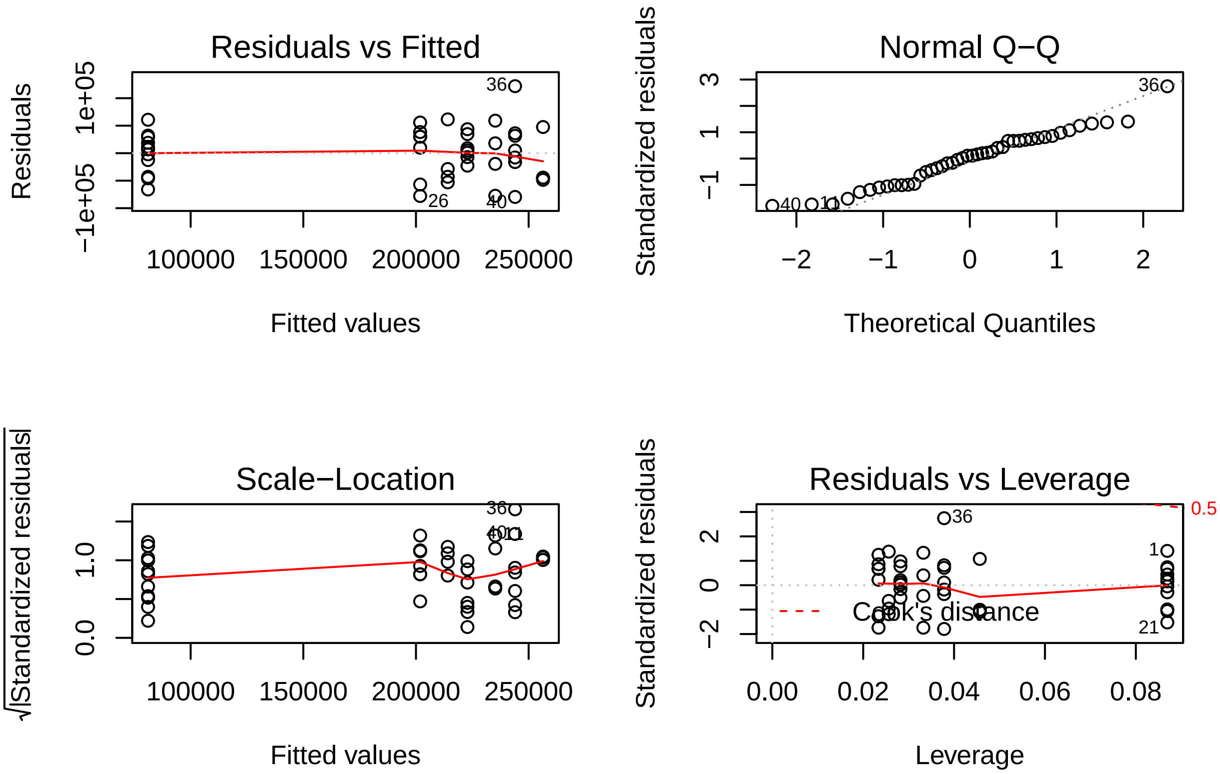
```
par(mfrow=c(2,2))
plot(model_sqrt)
```




```r
model_log<-lm(yield^2.5~log(nitrogen+1))
summary(model_log)
```

```
##
## Call:
## lm(formula = yield^2.5 ~ log(nitrogen + 1))
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -79593 -42913   4889  31652 121833
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)          81089      13341   6.078 3.07e-07 ***
## log(nitrogen + 1)    30717       3210   9.570 4.09e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 45250 on 42 degrees of freedom
## Multiple R-squared:  0.6856, Adjusted R-squared:  0.6781
## F-statistic: 91.59 on 1 and 42 DF,  p-value: 4.095e-12
```

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



According to the result above, the R^2 in `log_model` is 0.6856 which is larger than that of the `sqrt_model`, so with respect to the predicton, model `yield^2.5 ~ log(nitrogen + 1)` might do a better job. As for the diagnostics plot of `log_model`, all plots ,but the Scale-Location seem, look ok.