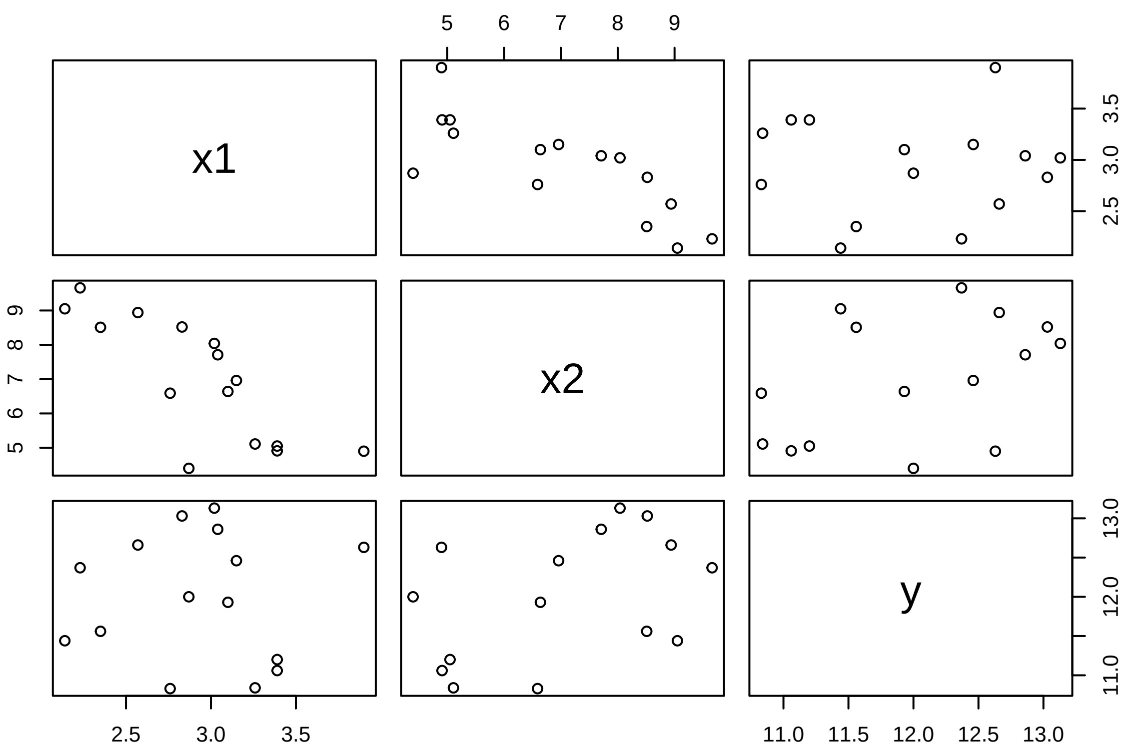
hw5

2022-11-03

##Problem 1

```r
dat = data.frame(
     x1=c(2.23,2.57,2.87,3.1,3.39,2.83,3.02,2.14,3.04,3.26,3.39,2.35,
       2.76,3.9,3.15),
     x2=c(9.66,8.94,4.4,6.64,4.91,8.52,8.04,9.05,7.71,5.11,5.05,8.51,
       6.59,4.9,6.96),
     y=c(12.37,12.66,12,11.93,11.06,13.03,13.13,11.44,12.86,10.84,
       11.2,11.56,10.83,12.63,12.46))
#(a)
plot(dat)
```



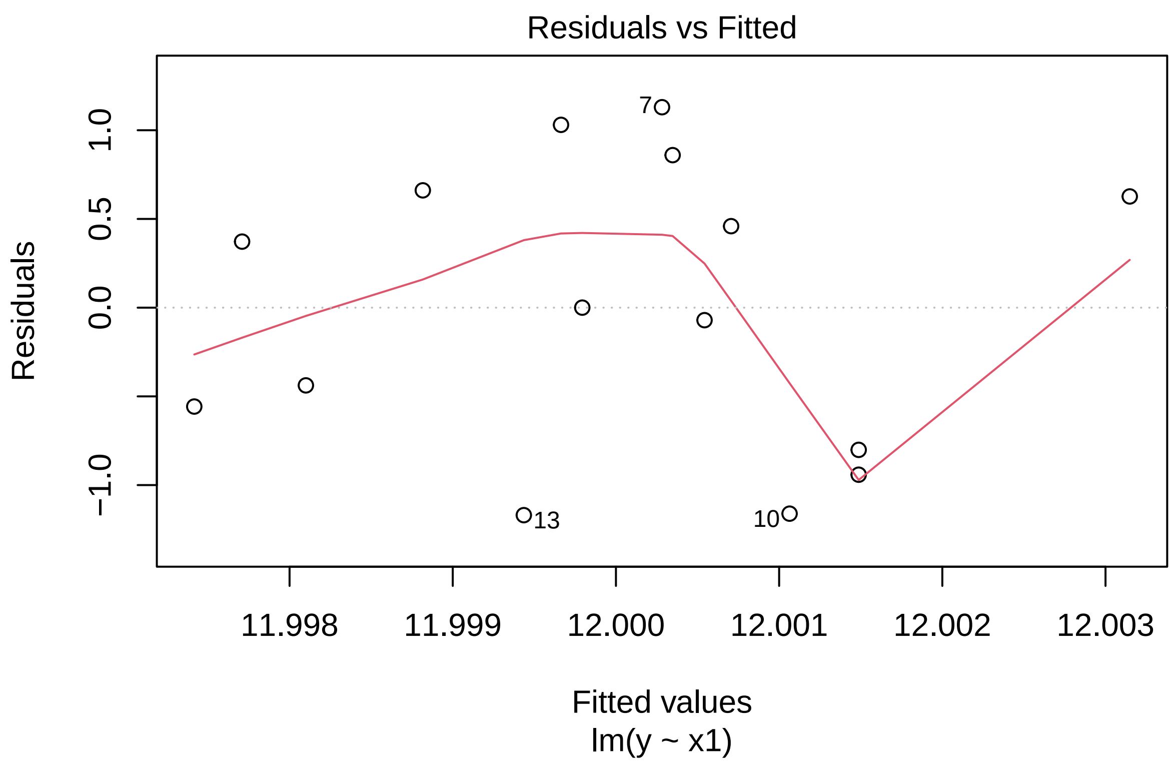
From the scatter plot, I don't see any clear association between y and x1 and between y and x2. And there is a negative association between x1 and x2.

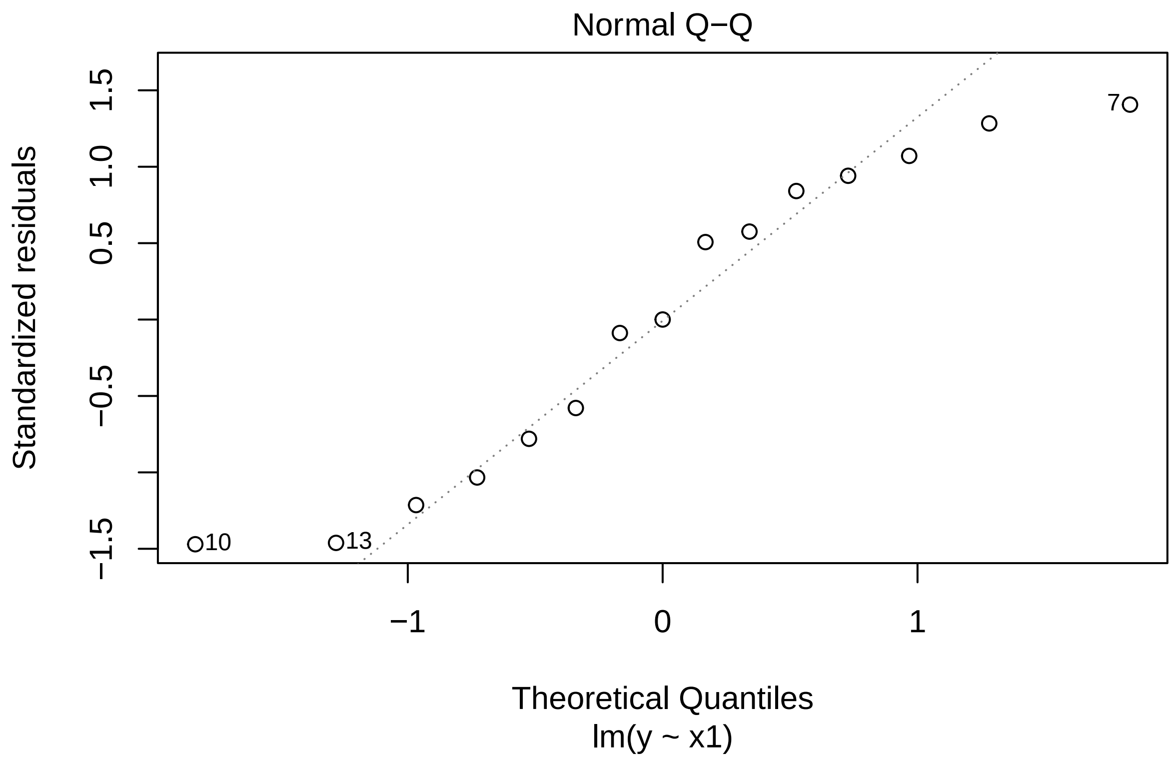
```r
#(b)
fit1 <- lm(y~x1,data = dat)
summary(fit1)
```

```
##
## Call:
## lm(formula = y ~ x1, data = dat)
##
```

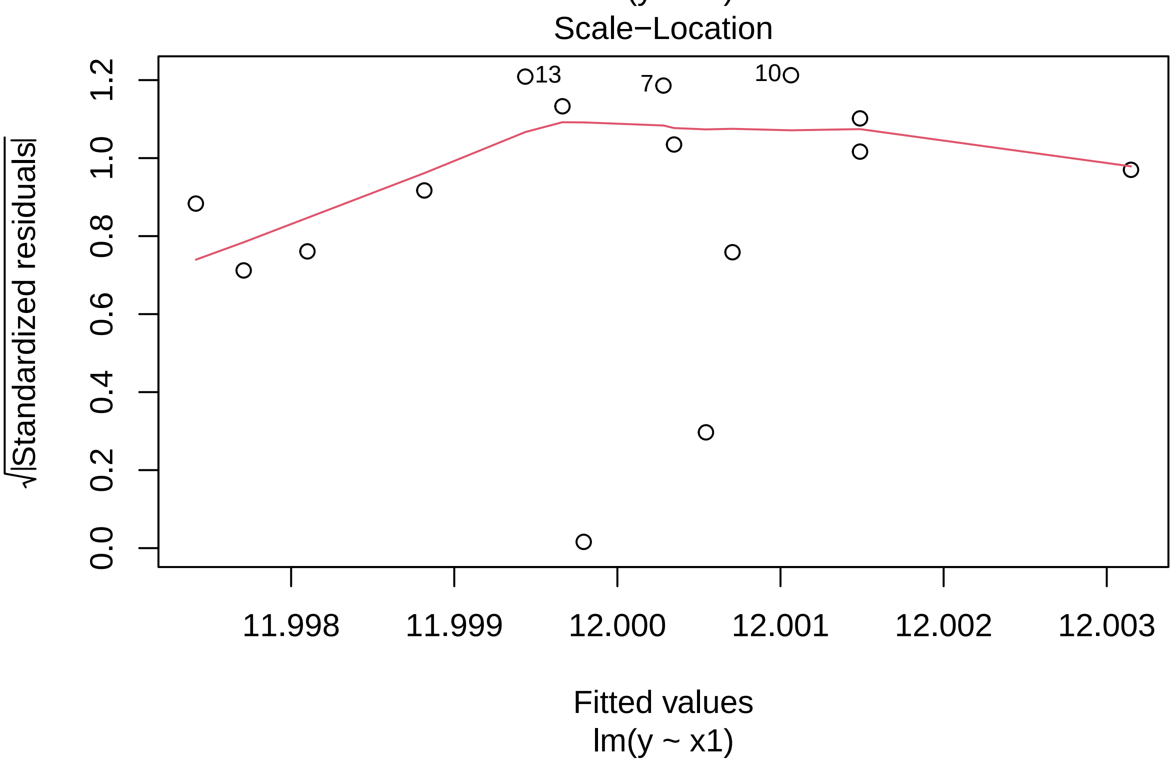
```
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -1.16944 -0.67945  0.00021  0.64402  1.12972 
## 
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 11.990446   1.383341   8.668  9.2e-07 ***
## x1           0.003257   0.465866   0.007    0.995    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.8324 on 13 degrees of freedom
## Multiple R-squared:  3.76e-06,   Adjusted R-squared:  -0.07692 
## F-statistic: 4.888e-05 on 1 and 13 DF,  p-value: 0.9945
```

```
plot(fit1)
```

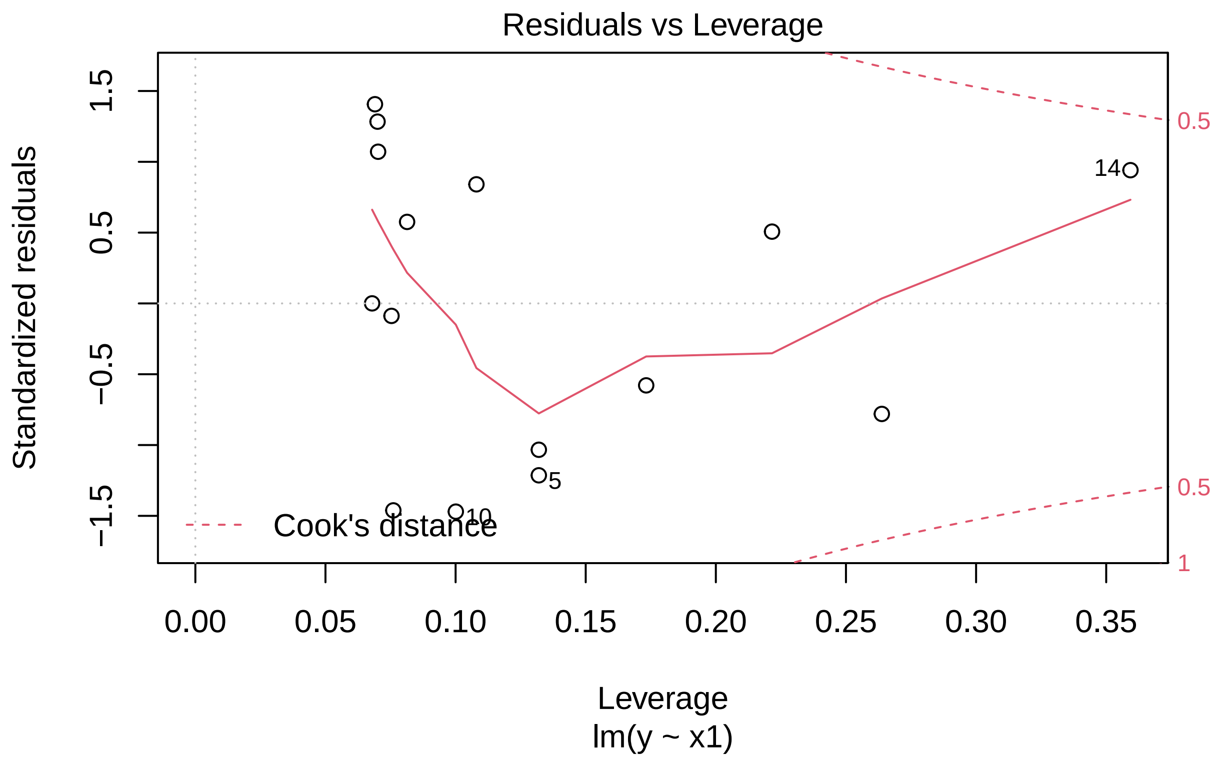




Normal Q−Q
Standardized residuals
−1.5
−0.5
0.5
1.0
1.5
−1
0
1
10
13
7
Theoretical Quantiles
lm(y ~ x1)



Scale−Location
√|Standardized residuals|
0.0
0.2
0.4
0.6
0.8
1.0
1.2
13
7
10
11.998
11.999
12.000
12.001
12.002
12.003
Fitted values
lm(y ~ x1)

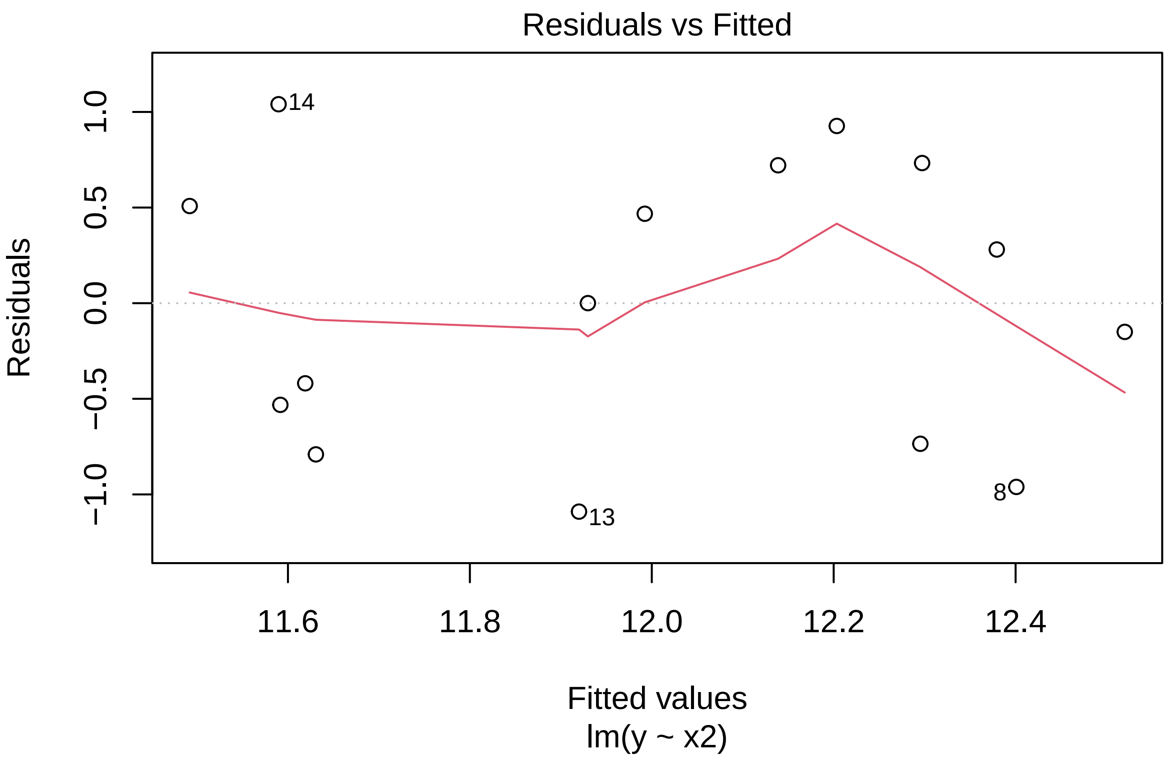


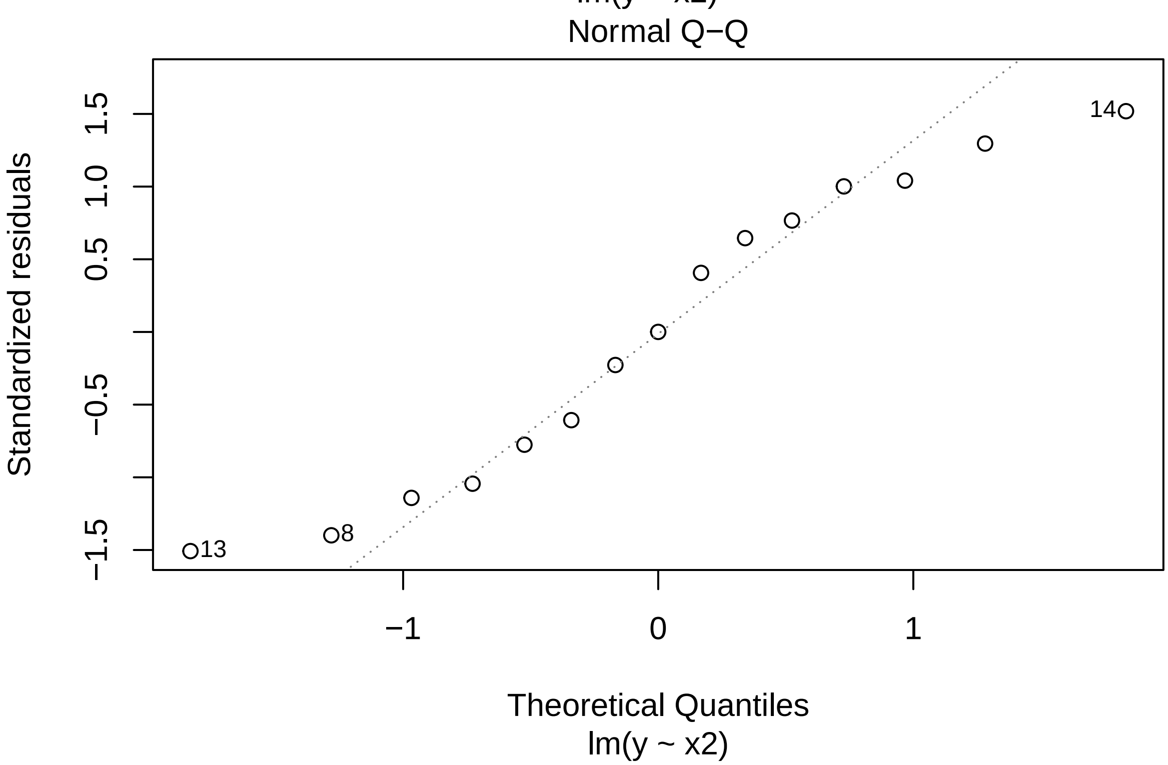
The overall model is not significant. Residuals do not distribute randomly above and below residuals = 0 line. The residual plot shows that there's no linear relationship between y and x1. And there exist heteroscedasticity. From the qq plot, we see that data is not normally distributed.

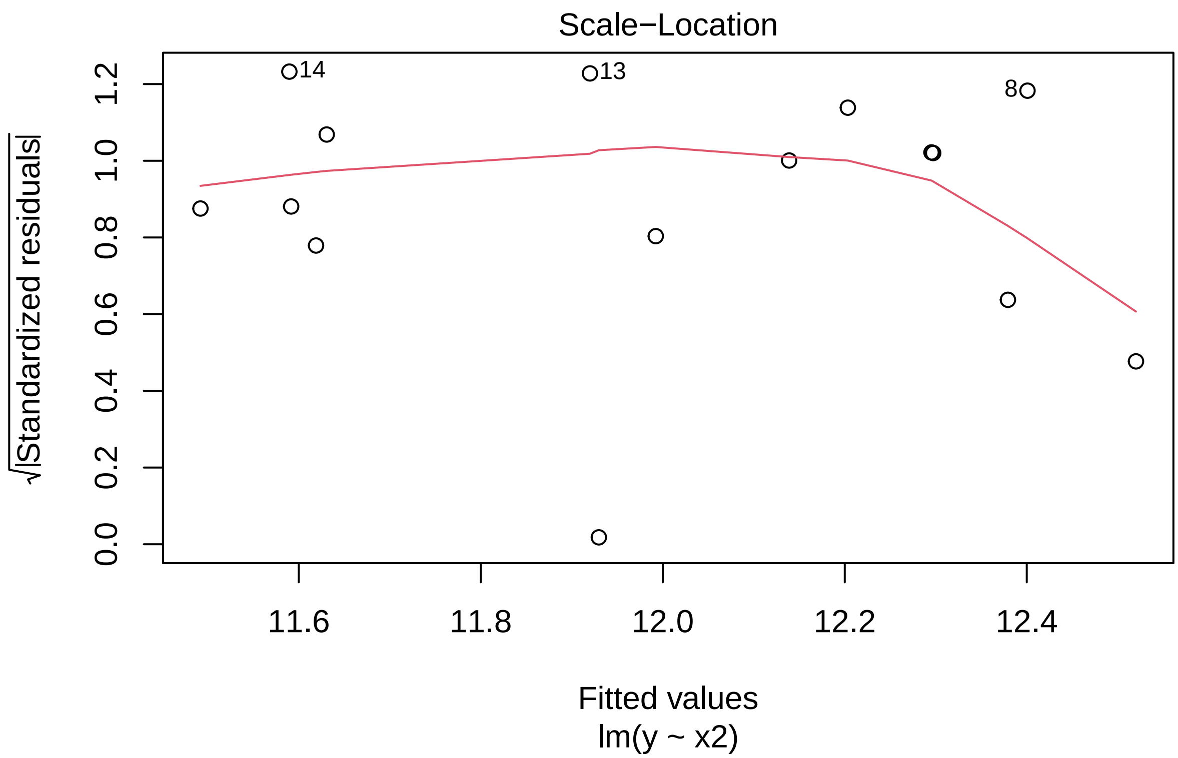
```
#(c)
fit2 = lm(y~x2, data = dat)
summary(fit2)
```

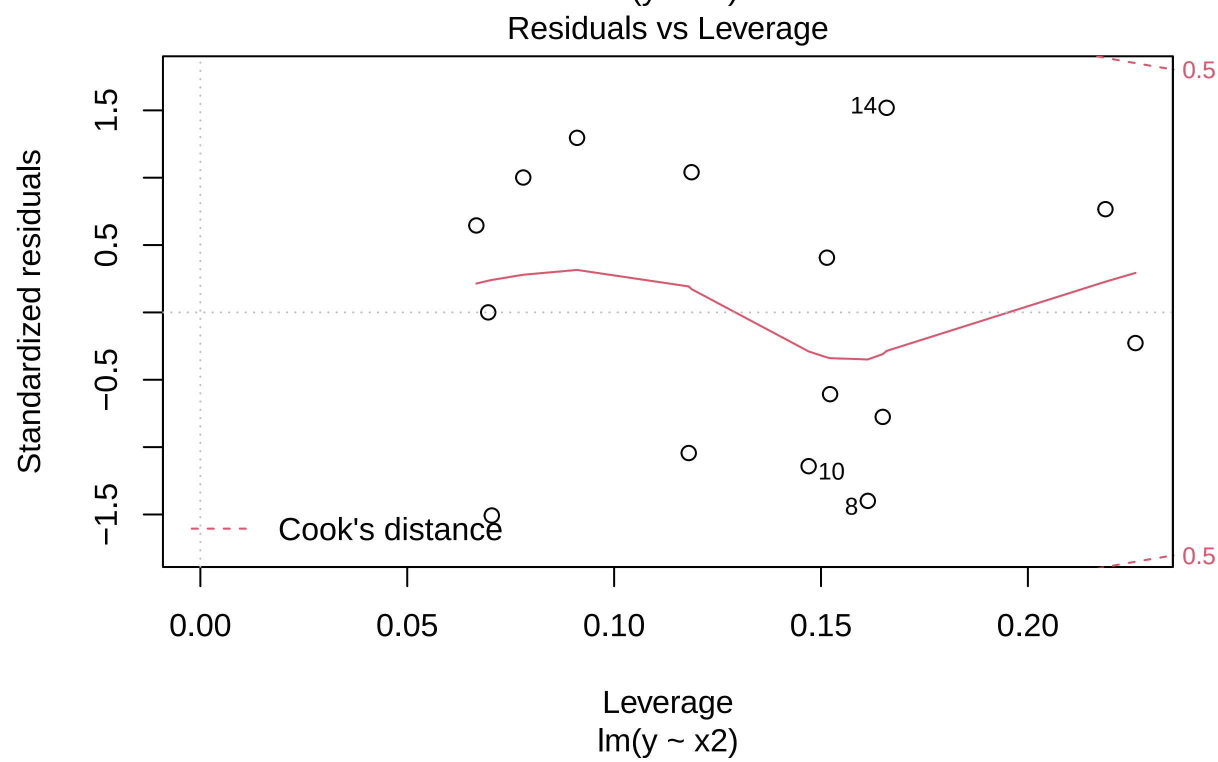
```
##
## Call:
## lm(formula = y ~ x2, data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.08999 -0.63345  0.00023  0.61458  1.04033
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  10.6319     0.8109  13.111 7.18e-09 ***
## x2            0.1955     0.1125   1.737    0.106
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7499 on 13 degrees of freedom
## Multiple R-squared:  0.1884, Adjusted R-squared:  0.126
## F-statistic: 3.018 on 1 and 13 DF,  p-value: 0.106
```

```
plot(fit2)
```


Residuals vs Fitted
Residuals
Fitted values
lm(y ~ x2)
14
13
8
1.0
0.5
0.0
−0.5
−1.0
11.6
11.8
12.0
12.2
12.4


Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(y ~ x2)
14
13
8
1.5
1.0
0.5
−0.5
−1.5
−1
0
1



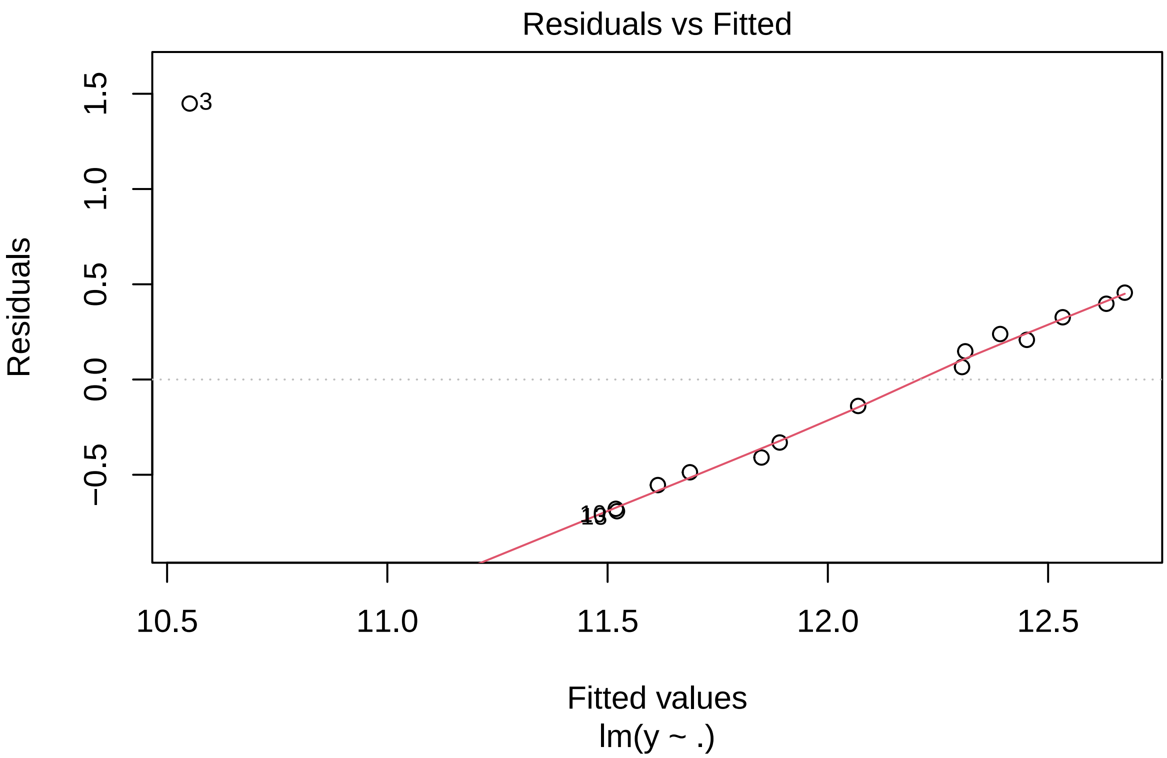


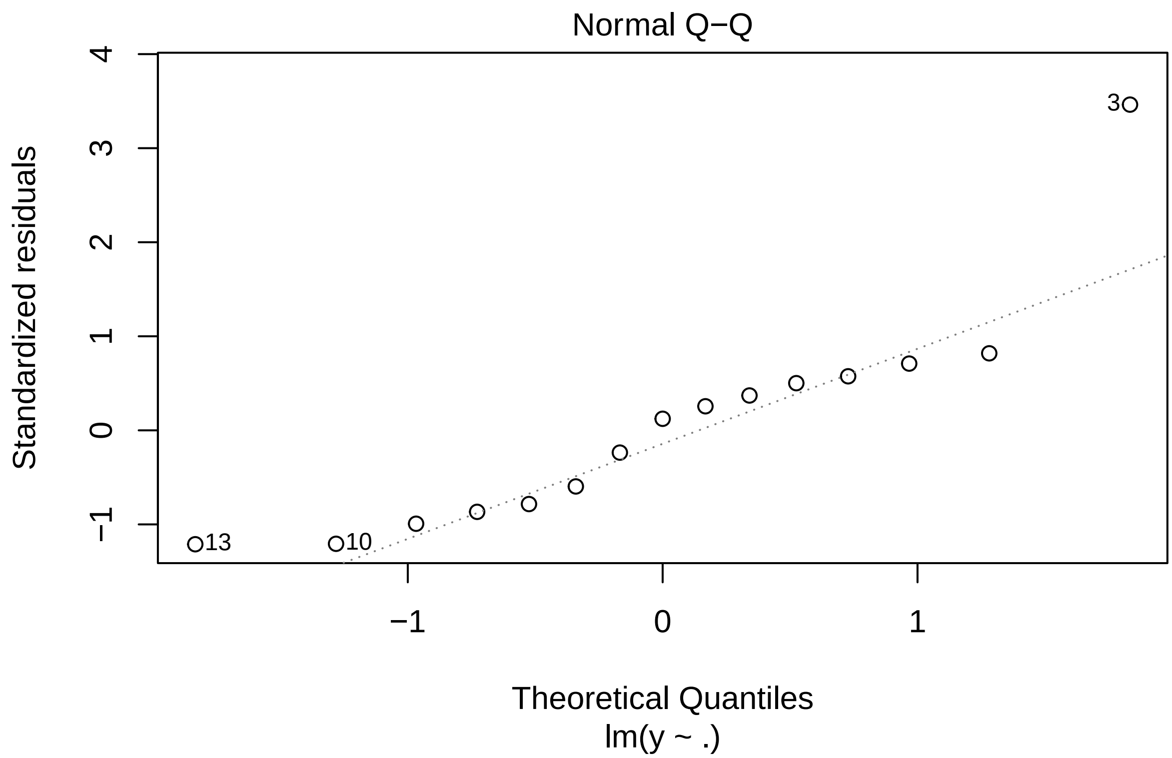
The overall model is not significant. Similarly, residuals do not distribute randomly above and below residuals = 0 line. The residual plot shows that there's no linear relationship between y and x2. Similarly, there exist heteroscedacsticity and the data is not normally distributed.

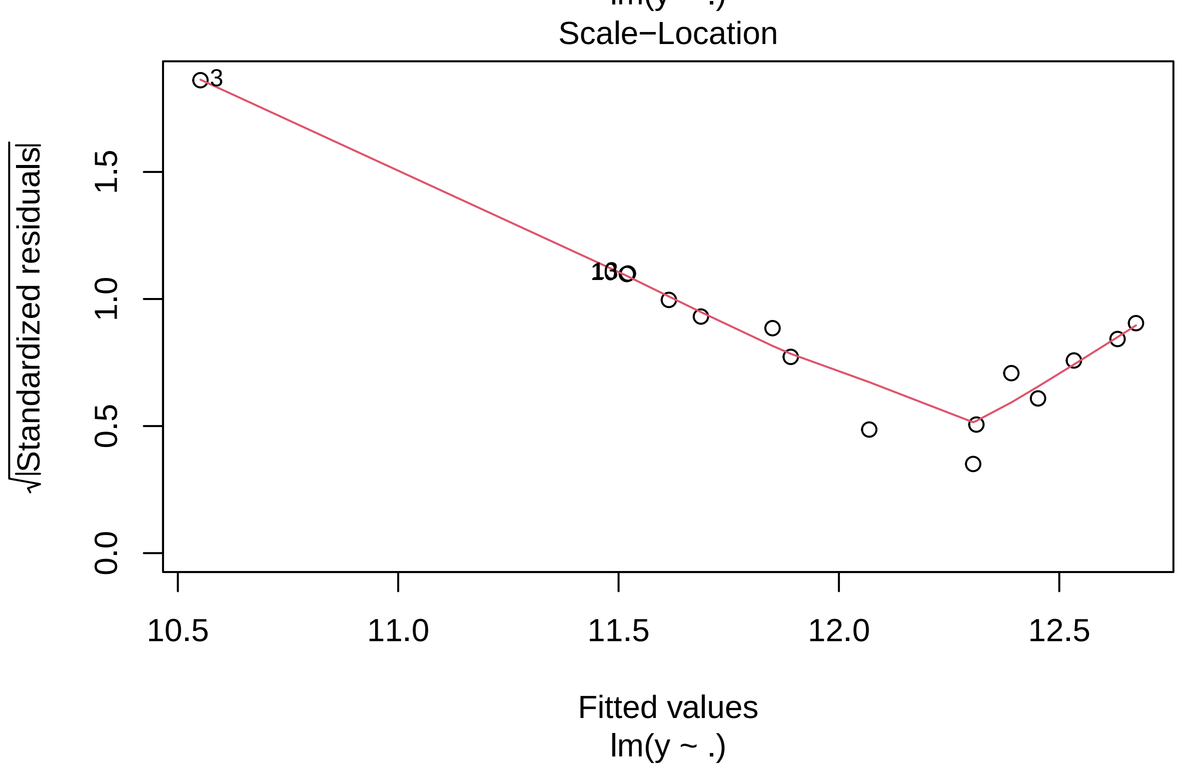
```
#(d)
fit3 <- lm(y~., data = dat)
summary(fit3)
```

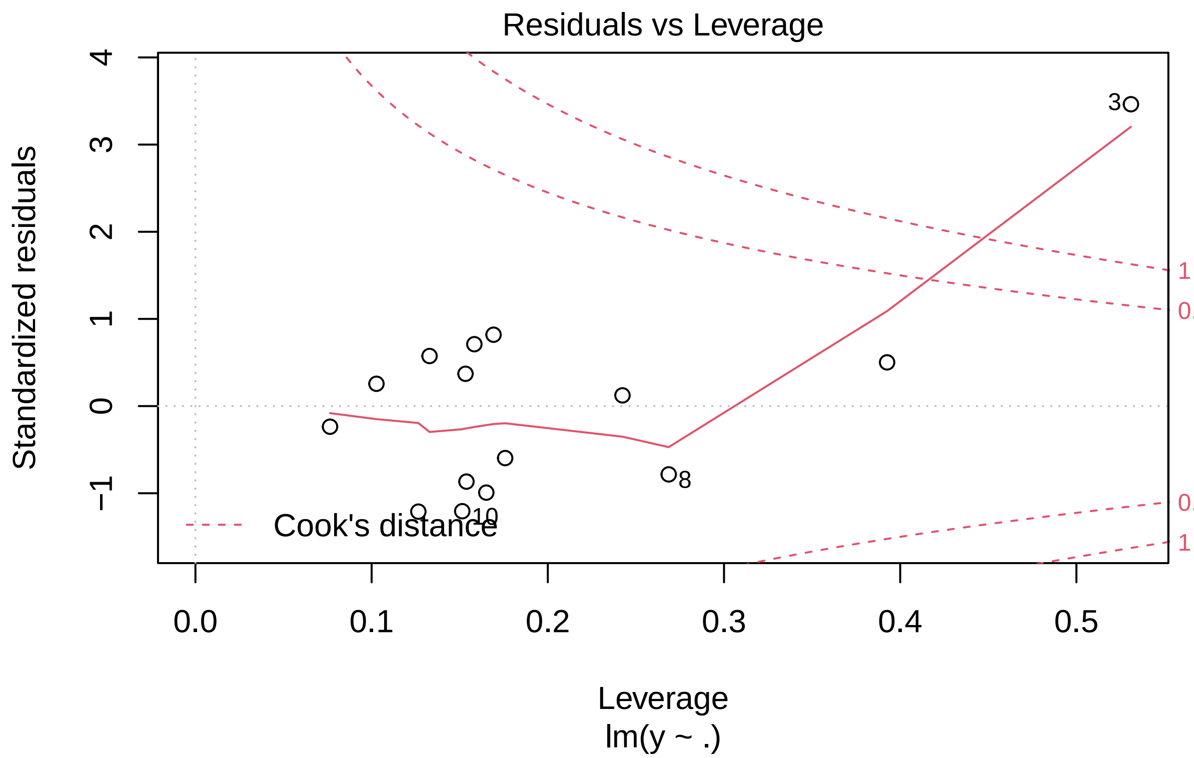
```
##
## Call:
## lm(formula = y ~ ., data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.69127 -0.44813  0.06541  0.28281  1.44873
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.8610     2.5440   1.518   0.1550
## x1            1.5339     0.5566   2.756   0.0174 *
## x2            0.5200     0.1492   3.485   0.0045 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6108 on 12 degrees of freedom
## Multiple R-squared:  0.503,  Adjusted R-squared:  0.4202
## F-statistic: 6.073 on 2 and 12 DF,  p-value: 0.01507
```

```
plot(fit3)
```




Normal Q−Q
Standardized residuals
−1
0
1
2
3
4
13
10
3
−1
0
1
Theoretical Quantiles
lm(y ~ .)


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
3
10
13
10.5
11.0
11.5
12.0
12.5
Fitted values
lm(y ~ .)



The overall model is significant since the p value of f test is 0.01507 < 0.05. Although the f test shows the overall model is significant, the residuals plot does not show constant variance or linear relationship. The residuals lie on a straight line and do not distribute randomly above and below residuals = 0 line. There's a clear pattern in the residual plot.

##(e) The problem tells me that forward variable selection and backward variable selection do not necessarily generate the same result. They may not select the same variables. If we implement forward variable selection technique, no variable will be selected because neither x1 nor x2 is significant to y, as illustrated in (a) and (b). However, if we implement backward variable selection, both x1 and x2 will be selected and included in the model because they both have p-value < 0.5 in (d). And no variable will be dropped.

Problem 2:

a. Ridge Regression: $\hat{\beta}_\lambda = \underset{\beta}{\operatorname{argmin}} \left[\sum_{i=1}^{n} (y_i - \hat{\beta}_i)^2 + \lambda \sum_{j=1}^{n} \hat{\beta}_j^2 \right]$

$\frac{\partial \text{Loss}}{\partial \beta_i} = -2(y_i - \hat{\beta}_i) + 2\lambda \hat{\beta}_i = 0. \Leftrightarrow \hat{\beta}_i(2+2\lambda) = 2y_i \rightarrow \hat{\beta}_i = \frac{y_i}{1+\lambda} \Rightarrow \hat{\beta}_i^R(\lambda) = \frac{\hat{\beta}_i}{1+\lambda}$

b. Lasso Regression: $\hat{\beta}_\lambda = \underset{\beta}{\operatorname{argmin}} \left[\sum_{i=1}^{n} (y_i - \hat{\beta}_i)^2 + \lambda \sum_{j=1}^{n} |\beta_j| \right]^2.$

Considering $\frac{\partial |\beta_i|}{\partial \beta_i} = S_i$.
$\begin{cases} S_i = \text{sign}(\beta_i) \text{ when } \beta_i \neq 0. \\ S_i \in [-1,1] \text{ if } \beta_j = 0. \end{cases}$

① when $\beta_i > 0$

$\frac{\partial \text{Loss}}{\partial \beta_i} = -2(y_i - \beta_i) + \lambda = 0 \Rightarrow \beta_i = y_i - \frac{\lambda}{2}$

$y_i - \frac{\lambda}{2} > 0 \Rightarrow y_i > \frac{\lambda}{2}$

② when $\beta_i < 0$.

$\frac{\partial \text{Loss}}{\partial \beta_i} = -2(y_i - \beta_i) - \lambda = 0 \Rightarrow \beta_i = y_i + \frac{\lambda}{2}$

$y_i + \frac{\lambda}{2} \cancel{> 0} < 0 \Rightarrow y_i < -\frac{\lambda}{2}$

③ when $\beta_i = 0$

$-2(y_i - \cancel{\beta_i}^{0}) + S_i\lambda = 0.$

$-2y_i + \lambda S_i = 0.$

$y_i = \frac{\lambda S_i}{2}$

$\Leftarrow \because S_i \in [-1,1]$

$|y_i| \le \frac{\lambda}{2}$

So $\hat{\beta}_i^L(\lambda) = \begin{cases} y_i - \frac{\lambda}{2} & \text{if } y_i > \frac{\lambda}{2} \\ y_i + \frac{\lambda}{2} & \text{if } y_i < -\frac{\lambda}{2} \\ 0 & \text{if } |y_i| \le \frac{\lambda}{2} \end{cases}$

c. When shrinking LS estimator, ridge does not tend to shrink LS estimator to zeros expect when λ is very large. Ridge tends to shrink LS estimator to smaller value. But the lasso tends to force some coefficients to equal zero. For example, in the example above, when $|y_i| = |\hat{\beta}_i^{OLS}| \le \frac{\lambda}{2}$, lasso will shrink them to be 0.

PA_HW5

Group10

2022-11-08

Problem 3

3(a)

```
library(MASS)
# dim(Boston)
Boston$logcrim = log(Boston$crim) # create log transform of crim
# summary(Boston)
set.seed(12345)
train = runif(nrow(Boston))<.5 # pick train/test split
print("Frequency Distribution")
```

```
## [1] "Frequency Distribution"
```

```
table(train) # frequency distribution
```

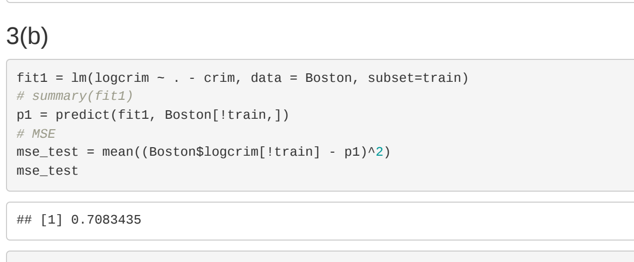
```
## train
## FALSE  TRUE
##   282   224
```

3(b)

```
fit1 = lm(logcrim ~ . - crim, data = Boston, subset=train)
# summary(fit1)
p1 = predict(fit1, Boston[!train,])
# MSE
mse_test = mean((Boston$logcrim[!train] - p1)^2)
mse_test
```

```
## [1] 0.7083435
```

```
# residual plot
plot(fit1, which = 1, pch = 16, cex = .5)
```



The test set MSE is 0.7083435. From the residual plot,we learn that the variance are not very constant.

3(c)

```
fit2 = step(fit1, trace = F)
p2 = predict(fit2, Boston[!train,])
mean((Boston$logcrim[!train] - p2)^2)
```

```
## [1] 0.7033381
```

The test set MSE is 0.7033381.

3(d)

```
library('glmnet')
```

```
## Warning: package 'glmnet' was built under R version 4.2.2
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1-4
```

```
# fit ridge
x = model.matrix(logcrim ~ . - crim -1,Boston)
fit_ridge = glmnet(x[train,], Boston$logcrim[train], alpha = 0)

# CV
set.seed(1234)
ridge_cv = cv.glmnet(x[train,], Boston$logcrim[train], alpha = 0) # find optimal lambda
l1 = ridge_cv$lambda.min
l1
```

```
## [1] 0.1866301
```

```
# test set mse
p3 = predict(fit_ridge, s=l1, newx=x[!train,])
mean((Boston$logcrim[!train] - p3)^2)
```

```
## [1] 0.7767352
```

The test set MSE is 0.7767352.

3(e)

```
# fit lasso
x = model.matrix(logcrim ~ . - crim -1,Boston)
fit_lasso = glmnet(x[train,], Boston$logcrim[train], alpha = 1)

# CV
set.seed(1234)
lasso_cv = cv.glmnet(x[train,], Boston$logcrim[train], alpha = 1) # find optimal lambda
l2 = lasso_cv$lambda.min
l2
```

```
## [1] 0.009288624
```

```
# test set mse
p4 = predict(fit_lasso, s=l2, newx=x[!train,])
mean((Boston$logcrim[!train] - p4)^2)
```

```
## [1] 0.7023993
```

The test set MSE is 0.7023993.

3(f)

```
# ridge model
x = model.matrix(logcrim ~zn
        +indus+I(indus^2)
        +chas
        +nox+I(nox^2)
        +rm+log(rm)
        +age+log(age)
        +dis+log(dis)
        +rad+log(rad)
        +tax+log(tax)
        +ptratio+log(ptratio)
        +black+log(black)
        +lstat+log(lstat)
        +medv+log(medv), Boston)
fit_r = glmnet(x[train,], Boston$logcrim[train], alpha=0)

fit_cv = cv.glmnet(x[train,], Boston$logcrim[train], alpha=0)
l_r = fit_cv$lambda.min

yhat = predict(fit_r, s=fit_cv$lambda.min, newx=x[!train,])
mean((Boston$logcrim[!train] - yhat)^2)
```

```
## [1] 0.6510401
```

```
# lasso model
fit_l = glmnet(x[train,], Boston$logcrim[train], alpha=1)

fit_cv2 = cv.glmnet(x[train,], Boston$logcrim[train], alpha=1)
l_l = fit_cv2$lambda.min

yhat2 = predict(fit_l, s=fit_cv2$lambda.min, newx=x[!train,])
mean((Boston$logcrim[!train] - yhat2)^2)
```

```
## [1] 0.5612213
```

```
# step-wise
fit4= lm(logcrim ~ 1, Boston, subset=train)
fit_s = step(fit4, scope=~zn
        +indus+I(indus^2)
        +chas
        +nox+I(nox^2)
        +rm+log(rm)
        +age+log(age)
        +dis+log(dis)
        +rad+log(rad)
        +tax+log(tax)
        +ptratio+log(ptratio)
        +black+log(black)
        +lstat+log(lstat)
        +medv+log(medv), trace = FALSE)
yhat3 = predict(fit_s, Boston[!train,])
mean((Boston$logcrim[!train] - yhat3)^2)
```

```
## [1] 0.5694736
```

Log transformations and square transformations of predictors are important. Because these transformations help reduce MSE in all three models.

Problem 4

4(a)

```
sigma = matrix(0.9, nrow=4, ncol=4) + .1*diag(4)
A = chol(sigma)
t(A) %*% A
```

```
##      [,1] [,2] [,3] [,4]
## [1,]  1.0  0.9  0.9  0.9
## [2,]  0.9  1.0  0.9  0.9
## [3,]  0.9  0.9  1.0  0.9
## [4,]  0.9  0.9  0.9  1.0
```

4(b)

```
Z = matrix(rnorm(4000), nrow=1000)
X = Z %*% A
var(X)
```

```
##           [,1]      [,2]      [,3]      [,4]
## [1,] 0.9319924 0.8492134 0.8528907 0.8503184
## [2,] 0.8492134 0.9769819 0.8667180 0.8691371
## [3,] 0.8528907 0.8667180 0.9625422 0.8652925
## [4,] 0.8503184 0.8691371 0.8652925 0.9634800
```

It approximately equal $ $.

4(c)

```
set.seed(12345)
# generate a new Z, A and X
n = 10100
ntrain=100
ntest=n-ntrain
Z <- matrix(rnorm(n*15), nrow=n)

sigma <- matrix(0.9, nrow=15, ncol=15) + diag(rep(1-0.9, 15))
A <- chol(sigma)
X <- Z %*% A

beta = c(1,-1,1.5,0.5,-0.5,rep(0,10))
e = rnorm(10100)*3
y = 3 + X %*% beta + e

train <- data.frame(X[1:ntrain,], y=y[1:ntrain])
test <- data.frame(X[(ntrain+1):n,], y=y[(ntrain+1):n])
```

4(d)

```
dat = data.frame(X)
dat$y <- y

fit = lm(y~X1+X2+X3+X4+X5,data=train)
summary(fit)
```

```
##
## Call:
## lm(formula = y ~ X1 + X2 + X3 + X4 + X5, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.8436 -2.0442  0.2997  1.8333  6.9526
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.0256     0.3295   9.183    1e-14 ***
## X1            0.9439     0.8875   1.064  0.29026
## X2           -1.6256     1.0049  -1.618  0.10906
## X3            2.7879     0.8924   3.124  0.00237 **
## X4           -0.3034     1.0439  -0.291  0.77200
## X5           -0.3711     0.8164  -0.455  0.65048
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.2 on 94 degrees of freedom
## Multiple R-squared:  0.2407, Adjusted R-squared:  0.2003
## F-statistic: 5.961 on 5 and 94 DF,  p-value: 7.843e-05
```

```
mean((test$y-predict(fit, test))^2)
```

```
## [1] 9.449501
```

```
confint(fit)
```

```
##                  2.5 %    97.5 %
## (Intercept)  2.3714294 3.6797538
## X1          -0.8182307 2.7059553
## X2          -3.6208313 0.3695436
## X3           1.0160970 4.5597365
## X4          -2.3759925 1.7692921
## X5          -1.9919717 1.2498126
```

The residual standard error is 3.2. Slopes for xi(i=1,2,3,4,5) are 0.9439, -1.6256, 2.7879, -0.3034, -0.3711. R-squared is 0.2407. Slope for x1 roughly equals the true parameter and the other slops aren't. Slope for x4 has the wrong sign, other slopes have the right signs. In terms of statistical significance, only slope for x3 is significant at 0.01 level. All the 95% confidence interval covers the true value.

4(e)

```
mean((test$y-predict(fit, test))^2)
```

```
## [1] 9.449501
```

The value of MSE is 9.449501.

4(f)

```
fit_4f = lm(y ~ ., data = train)
summary_4f = summary(fit_4f)
coefs = summary_4f$coefficients

betas_4f = coefs[2:16]
se = coefs[18:32]
ci_l = c()
ci_u = c()
for (idx in 1:length(betas_4f)) {
  ci_l = append(ci_l, (betas_4f[idx] - 2 * se[idx]))
  ci_u = append(ci_u, (betas_4f[idx] + 2 * se[idx]))
}

is_in = c()
for (idx in 1:length(betas_4f)) {
  if ((beta[idx] > ci_l[idx]) & (beta[idx] < ci_u[idx])) {
    is_in = append(is_in, 1)
  } else {
    is_in = append(is_in, 0)
  }
}
vars_in = sum(is_in)
print(summary(fit_4f))
```

```
##
## Call:
## lm(formula = y ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.7768 -1.8727  0.0985  1.8531  6.4236
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.09711    0.34378   9.009 5.69e-14 ***
## X1           1.64535    1.01410   1.622  0.10845
## X2          -1.27455    1.12632  -1.132  0.26102
## X3           3.04446    0.99629   3.056  0.00301 **
## X4           0.17894    1.16865   0.153  0.87867
## X5           0.12057    0.95410   0.126  0.89974
## X6           0.42167    1.04928   0.402  0.68880
## X7          -0.05058    1.16496  -0.043  0.96547
## X8          -1.48874    1.18517  -1.256  0.21255
## X9           1.02701    1.03928   0.988  0.32589
## X10         -0.83981    1.13596  -0.739  0.46179
## X11          0.68516    1.02798   0.667  0.50691
## X12         -0.55163    1.07908  -0.511  0.61055
## X13         -1.25600    1.22391  -1.026  0.30773
## X14          0.52319    1.01348   0.516  0.60705
## X15         -0.73817    1.23259  -0.599  0.55086
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.258 on 84 degrees of freedom
## Multiple R-squared:  0.2964, Adjusted R-squared:  0.1708
## F-statistic: 2.359 on 15 and 84 DF,  p-value: 0.007036
```

- 15 of the 15 are in the confidence intervals
- The only feature that's significant is x_3 orβ_3. None of the other xs (1 through 5) are significant.
- All of the signs are not correct for beta 1 through 5, β_5 should be positive.

4(g)

```
mse_4g = mean((test$y-predict(fit_4f, test))^2)
```

The MSE is 10.2347653 for the full model.

4(h)

```
step_4h = stepAIC(lm(y ~., data = train),
                  direction = "both",
                  trace = F)
summary(step_4h)
```

```
##
## Call:
## lm(formula = y ~ X1 + X3 + X8 + X13, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.2310 -1.8975  0.2254  1.6861  7.4489
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.0673     0.3217   9.533 1.64e-15 ***
## X1            1.3978     0.8835   1.582 0.116921
## X3            3.0285     0.8466   3.577 0.000548 ***
## X8           -1.5716     0.9797  -1.604 0.112011
## X13          -1.4510     1.0226  -1.419 0.159185
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.14 on 95 degrees of freedom
## Multiple R-squared:  0.2609, Adjusted R-squared:  0.2297
## F-statistic: 8.382 on 4 and 95 DF,  p-value: 7.787e-06
```

The “right” variables did not come into the model.

4(i)

```
mse_4i = mean((test$y-predict(step_4h, test))^2)
```

The MSE is 10.0442626 for the model found using stepAIC.

4(j)

```
x_4j = model.matrix(y ~ . -1, data = train)

set.seed(1234)
ridge_cv_4j = cv.glmnet(x_4j, train$y, alpha = 0, nfolds = 5)
plot(ridge_cv_4j, xvar = 'lambda', label = T)
```

```
## Warning in plot.window(...): "xvar" is not a graphical parameter
```

```
## Warning in plot.window(...): "label" is not a graphical parameter
```

```
## Warning in plot.xy(xy, type, ...): "xvar" is not a graphical parameter
```

```
## Warning in plot.xy(xy, type, ...): "label" is not a graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "xvar" is not a
## graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "label" is not a
## graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "xvar" is not a
## graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "label" is not a
## graphical parameter
```

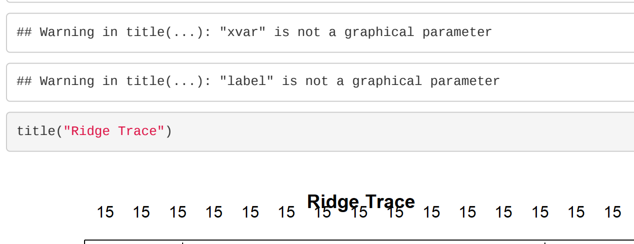
```
## Warning in box(...): "xvar" is not a graphical parameter
```

```
## Warning in box(...): "label" is not a graphical parameter
```

```
## Warning in title(...): "xvar" is not a graphical parameter
```

```
## Warning in title(...): "label" is not a graphical parameter
```

```
title("Ridge Trace")
```



4(k)

```
x_4k = model.matrix(y ~ . - 1, data = test)
mse_4k = mean((test$y - predict(ridge_cv_4j, s = ridge_cv_4j$lambda.min, x_4k))^2)
```

The MSE is 9.3658854 for the Ridge model.

4(l)

```
x_4l = model.matrix(y ~ . -1, data = train)

set.seed(1234)
lasso_cv_4l = cv.glmnet(x_4l, train$y, alpha = 1, nfolds = 5)
plot(lasso_cv_4l, xvar = 'lambda', label = T)
```

```
## Warning in plot.window(...): "xvar" is not a graphical parameter
```

```
## Warning in plot.window(...): "label" is not a graphical parameter
```

```
## Warning in plot.xy(xy, type, ...): "xvar" is not a graphical parameter
```

```
## Warning in plot.xy(xy, type, ...): "label" is not a graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "xvar" is not a
## graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "label" is not a
## graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "xvar" is not a
## graphical parameter
```

```
## Warning in axis(side = side, at = at, labels = labels, ...): "label" is not a
## graphical parameter
```

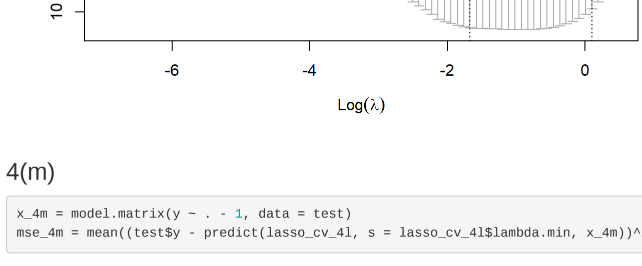
```
## Warning in box(...): "xvar" is not a graphical parameter
```

```
## Warning in box(...): "label" is not a graphical parameter
```

```
## Warning in title(...): "xvar" is not a graphical parameter
```

```
## Warning in title(...): "label" is not a graphical parameter
```

```
title("Lasso Trace")
```



4(m)

```
x_4m = model.matrix(y ~ . - 1, data = test)
mse_4m = mean((test$y - predict(lasso_cv_4l, s = lasso_cv_4l$lambda.min, x_4m))^2)
```

The MSE is 9.4173197 for the Ridge model.

4(n)

```
source("hw5.R")
rho = c(0.9,0.5,0.1)
sigma = c(5,3,1)

for (r in rho) {
  for (s in sigma) {
    hw5(rho = r, sigmae = s)
    cat("Rho value: ", r, " ", "Sigma value: ", s, "\n")
  }
}
```

```
## ------------------------------------------
## correlation between x:  0.9
## Error variance:  25
## OLS x1-x5: 26.24862
## OLS x1-x15: 28.4299
## backward ( 2 ): 26.96192
## forward ( 2 ): 26.96192
## ridge: 25.89624
## lasso: 25.72874
## Rho value:  0.9   Sigma value:  5
## ------------------------------------------
## correlation between x:  0.9
## Error variance:  9
## OLS x1-x5: 9.449501
## OLS x1-x15: 10.23477
## backward ( 4 ): 10.04426
## forward ( 2 ): 9.879715
## ridge: 9.365885
## lasso: 9.41732
## Rho value:  0.9   Sigma value:  3
## ------------------------------------------
## correlation between x:  0.9
## Error variance:  1
## OLS x1-x5: 1.049945
## OLS x1-x15: 1.137196
## backward ( 4 ): 1.14975
## forward ( 4 ): 1.14975
## ridge: 1.282693
## lasso: 1.269259
## Rho value:  0.9   Sigma value:  1
## ------------------------------------------
## correlation between x:  0.5
## Error variance:  25
## OLS x1-x5: 26.24862
## OLS x1-x15: 28.4299
## backward ( 4 ): 27.28365
## forward ( 2 ): 26.96062
## ridge: 27.19028
## lasso: 26.70358
## Rho value:  0.5   Sigma value:  5
## ------------------------------------------
## correlation between x:  0.5
## Error variance:  9
## OLS x1-x5: 9.449501
## OLS x1-x15: 10.23477
## backward ( 4 ): 10.06145
## forward ( 4 ): 10.06145
## ridge: 9.699253
## lasso: 9.695756
## Rho value:  0.5   Sigma value:  3
## ------------------------------------------
## correlation between x:  0.5
## Error variance:  1
## OLS x1-x5: 1.049945
## OLS x1-x15: 1.137196
## backward ( 6 ): 1.093831
## forward ( 6 ): 1.093831
## ridge: 1.324893
## lasso: 1.382698
## Rho value:  0.5   Sigma value:  1
## ------------------------------------------
## correlation between x:  0.1
## Error variance:  25
## OLS x1-x5: 26.24862
## OLS x1-x15: 28.4299
## backward ( 4 ): 27.49461
## forward ( 4 ): 27.49461
## ridge: 27.69429
## lasso: 27.17986
## Rho value:  0.1   Sigma value:  5
## ------------------------------------------
## correlation between x:  0.1
## Error variance:  9
## OLS x1-x5: 9.449501
## OLS x1-x15: 10.23477
## backward ( 4 ): 10.27753
## forward ( 4 ): 10.27753
## ridge: 9.917381
## lasso: 9.858322
## Rho value:  0.1   Sigma value:  3
## ------------------------------------------
## correlation between x:  0.1
## Error variance:  1
## OLS x1-x5: 1.049945
## OLS x1-x15: 1.137196
## backward ( 6 ): 1.095551
## forward ( 6 ): 1.095551
## ridge: 1.31475
## lasso: 1.264195
## Rho value:  0.1   Sigma value:  1
```

When there is moderate to low rho/multicollinearity, OLS generally preforms better than ridge or lasso regression. Ridge or lasso regression work better when there is high multicollinearity and moderate to high noise. When using lasso or ridge we will use lasso when noise is higher. In this test they preform best when rho is 0.9 and sigma is greater than or equal to 5.