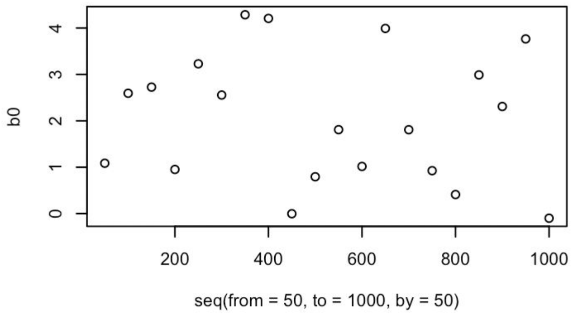
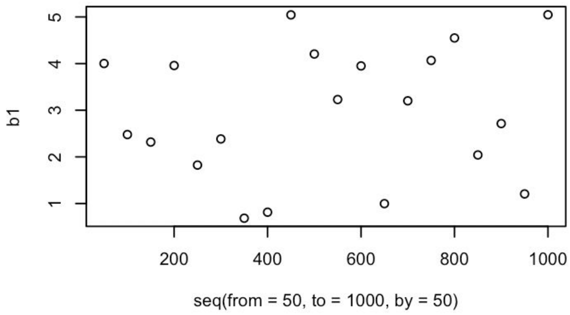
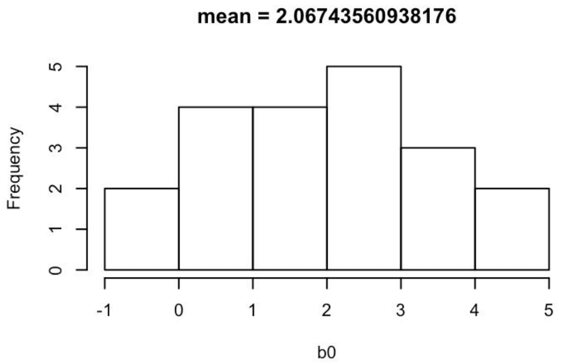
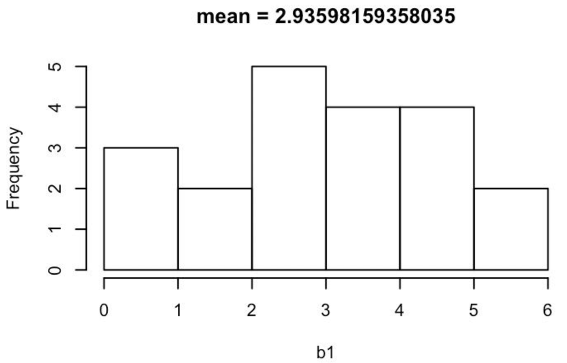
1.
e)

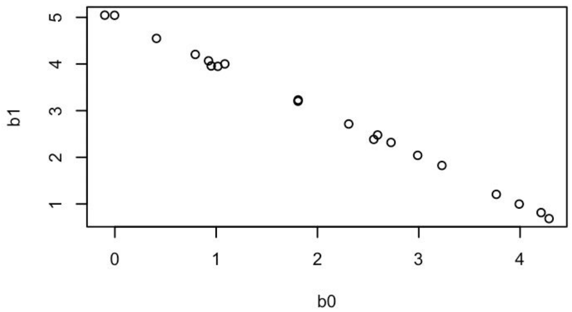
```
#declare weight vectors
b0 <- vector()
b1 <- vector()

for (val in seq(from = 50, to=1000, by = 50)) # for different n, obtain b0 and b1
{
n = val
x1 = matrix(1,n,1) #x1 is n-by-1 matrix
x1 = c(x1, 2) # x1 is now (n+1)-by-1 matrix
e = rnorm(n+1,0,1) # a vector of n+1 normally distributed noise
y = 2+3*x1 + e
simplefit = lm(y~x1)#least squares linear regression
coeff = coef(simplefit)#get coefficients

#store coefficients for the current n
b0 = rbind(b0,coeff[1])
b1 = rbind(b1,coeff[2])
}
#plot weights across n
plot(seq(from = 50, to=1000, by = 50),b0)
plot(seq(from = 50, to=1000, by = 50),b1)
meanb0 = paste("mean =",toString(mean(b0)), sep = " ", collapse = NULL)
meanb1 =  paste("mean =",toString(mean(b1)), sep = " ", collapse = NULL)

#histograms of weights
hist(b0, main = meanb0)
hist(b1, main = meanb1)

Correlation between b0 and b1
correlation = cor(b0,b1)
plot(b0,b1)
```




According to the four graphs in the first two rows, “n” (the number of observations) does not seem to influence weights very much; correlation between b0 and n is -0.1344135, whereas correlation between b1 and n is 0.1230906.  Rather, b0 seems to follow a normal distribution with mean 2, whereas b1 follows a normal distribution with mean 3. Furthermore, as shown in the last graph, b0 and b1 are negatively correlated, with correlation = -0.9994496.

3
a)

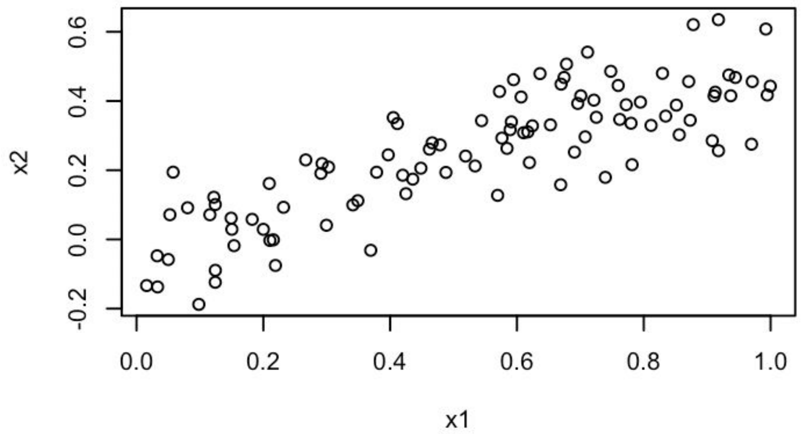
#run commands in the question
set.seed(240)
x1=runif(100)
x2=0.5*x1+rnorm(100)/10
y=2+2*x1+0.3*x2+rnorm(100)

Linear Model

the regression coefficients

B0 = 2. B1 = 2. B2 = 0.3

b)
corr = cor(x1,x2)
plot(x1,x2)




Correlation between x1 and x2 is 0.835556.

c)
multifit = lm(y~x1+x2)
summary(multifit)

```
Residuals:
     Min       1Q   Median       3Q      Max 
-3.05592 -0.70231 -0.02194  0.75459  3.15141 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 1.969709   0.218532   9.013 1.81e-14 ***
x1          2.035884   0.647079   3.146   0.0022 ** 
x2          0.005801   1.017236   0.006   0.9955    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.021 on 97 degrees of freedom
Multiple R-squared:  0.2532,	Adjusted R-squared:  0.2378 
F-statistic: 16.45 on 2 and 97 DF,  p-value: 7.068e-07
```

The model gives B0 = 1.969709, B1 = 2.035884, and B2 = 0.005801.
P-values for the coefficients are under the “Pr(>|t|)”,
Since the p-value for B1 is significant, I can reject the null hypothesis that B1 = 0.
On the other hand, since the p-value for B2 is not significant, I cannot reject the null hypothesis that B2 = 0.

d)
simplefitx1 = lm(y~x1)
summary(simplefitx1)

```
Residuals:
     Min       1Q   Median       3Q      Max 
-3.05494 -0.70239 -0.02164  0.75511  3.15114 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   1.9695     0.2151   9.154 8.28e-15 ***
x1            2.0390     0.3537   5.765 9.49e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.016 on 98 degrees of freedom
Multiple R-squared:  0.2532,	Adjusted R-squared:  0.2456 
F-statistic: 33.23 on 1 and 98 DF,  p-value: 9.492e-08
```

The model gives B0 = 1.9695 and B1 = 2.0390.
Since the p-value for B1 is significant, I can reject the null hypothesis that B1 = 0.

e)
simplefitx2 = lm(y~x2)
summary(simplefitx2)

```
Residuals:
    Min      1Q  Median      3Q     Max 
-3.5851 -0.6310 -0.0088  0.6724  3.0686 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   2.3820     0.1826  13.041  < 2e-16 ***
x2            2.6800     0.5837   4.591 1.31e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.066 on 98 degrees of freedom
Multiple R-squared:  0.177,	Adjusted R-squared:  0.1686 
F-statistic: 21.08 on 1 and 98 DF,  p-value: 1.308e-05
```

The model gives B0 = 2.382 and B2 = 2.68.
Since the p-value for B2 is significant, I can reject the null hypothesis that B2 = 0.

f)

In the the least squares regression to predict y using x1 and x2, I could not reject the null hypothesis that B2 = 0. On the other hand, in the the least squares regression to predict y using only x2, I could reject the null hypothesis that B2 = 0.

However, the above results from the two models don't contradict each other because the seemingly identical null hypothesis (B2 = 0) is not actually identical in the two models. In the multivariate model, the null hypothesis means "B2 = 0 given y = B0+B1*x1+B2*x2". In contrast, the null hypothesis of the univariate model means "B2 = 0 given y = B0 + B2*x2"

g)

```
x1=c(x1,0.1)
x2=c(x2,0.8)
y=c(y,6)

#regressions with the new observation
newsimplefitx1 = lm(y~x1)
summary(newsimplefitx1)
newmultifit = lm(y~x1+x2)
summary(newmultifit)
newsimplefitx2 = lm(y~x2)
summary(newsimplefitx2)
```

```
Call:
lm(formula = y ~ x1 + x2)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.3985 -0.7134 -0.0901  0.6590  3.1700 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   2.1154     0.2209   9.577    1e-15 ***
x1            0.9522     0.5510   1.728   0.0871 .  
x2            1.8120     0.8397   2.158   0.0334 *  
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.06 on 98 degrees of freedom
Multiple R-squared:  0.2354,	Adjusted R-squared:  0.2198 
F-statistic: 15.09 on 2 and 98 DF,  p-value: 1.939e-06
```

The above is the summary of the new model predicting y using x1 and x2. The new observation showed two noticeable changes: B1 from 2.036 to 0.9522 and B2 from 0.006 to 1.812. The above changes in B1 and B2 are almost 2 standard error away from the original model. The new observation is an outlier in the sense that it produced significant changes. However, although the amplitudes of the changes are significant and unlikely, the amplitudes of the changes are still expected to occur rarely (2 standard error away). Thus, the new observation is a high leverage point.

```
Call:
lm(formula = y ~ x1)

Residuals:
    Min      1Q  Median      3Q     Max
-3.1177 -0.7514 -0.0038  0.7910  3.7041

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.1116     0.2249   9.389 2.36e-15 ***
x1            1.8431     0.3715   4.961 2.92e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.079 on 99 degrees of freedom
Multiple R-squared:  0.1991,	Adjusted R-squared:  0.191
F-statistic: 24.61 on 1 and 99 DF,  p-value: 2.917e-06
```

The above is the summary of the new model predicting y using only x1. The new observation changes B0 from 1.9695 to 2.1116 and B1 from 2.0390 to 1.8431. The amplitudes of the changes are less than 1 standard error, so the differences in the coefficients are likely due to chance. Thus, the new observation is not a high leverage point. Also, since the new observation does not change the model significantly, the new observation is not an outlier.

```
Call:
lm(formula = y ~ x2)

Residuals:
    Min      1Q  Median      3Q     Max
-3.6200 -0.6158 -0.0234  0.6339  3.1045

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.3397     0.1805  12.964  < 2e-16 ***
x2            2.8993     0.5616   5.163 1.26e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.07 on 99 degrees of freedom
Multiple R-squared:  0.2121,	Adjusted R-squared:  0.2042
F-statistic: 26.65 on 1 and 99 DF,  p-value: 1.258e-06
```

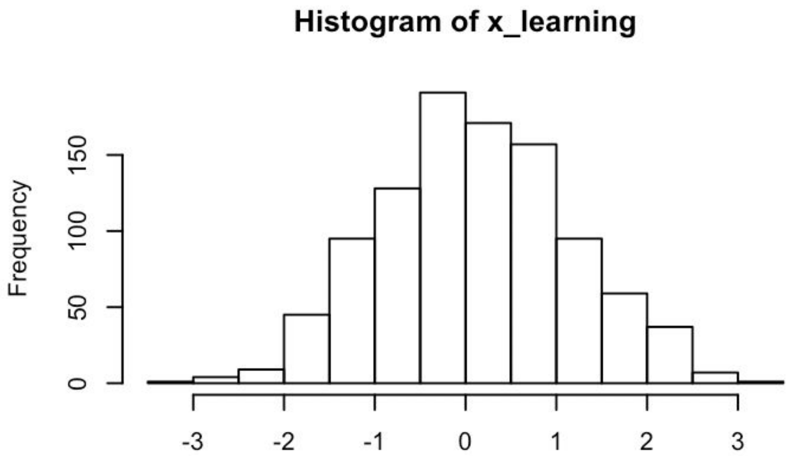
The above is the summary of the new model predicting y using only x2.  The new observation changes B0 from 2.382 to 2.3397 and B2 from 2.68 to 2.8993. The amplitudes of the changes are less than 1 standard error, so the differences in the coefficients are likely due to chance. Thus, the new observation is not a high leverage point. Also, since the new observation does not change the model significantly, the new observation is not an outlier.

5. (collaborated with Ingrid)

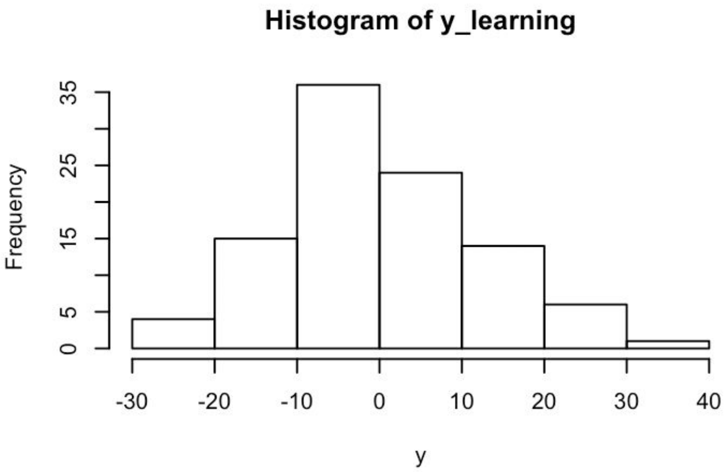
a)

```
library(MASS)
library(glmnet)
zeros_j = c(0,0,0,0,
        0,0,0,0,0,0)
betas = t(t(c(-4,-3,-2,-1,0,0,1,2,3,4)))

#covariates with autocorrection
gamma_vals = c(1, 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, 1/128, 1/256, 1/512,
            1/2, 1, 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, 1/128, 1/256,
            1/4, 1/2, 1, 1/2, 1/4, 1/8, 1/16, 1/32, 1/64, 1/128,
            1/8, 1/4, 1/2, 1, 1/2, 1/4, 1/8, 1/16, 1/32, 1/64,
            1/16, 1/8, 1/4, 1/2, 1, 1/2, 1/4, 1/8, 1/16, 1/32,
            1/32, 1/16, 1/8, 1/4, 1/2, 1, 1/2, 1/4, 1/8, 1/16,
            1/64, 1/32, 1/16, 1/8, 1/4, 1/2, 1, 1/2, 1/4, 1/8,
            1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2, 1, 1/2, 1/4,
            1/256, 1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2, 1, 1/2,
            1/512, 1/256, 1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2, 1)
#gamma cov matrix
Gamma = matrix(gamma_vals, nrow = 10, ncol = 10)

x_learning= mvrnorm(n = 100, mu = zeros_j, Sigma = Gamma)#x learning set
y_learning= vector()
for (n in 1:100)
{
  yval = rnorm(1, x_learning[n,]%*%betas, 4) #obtain y for each row in x learning set
  y_learning = c(y_learning,yval) #construct y learning set
}

x_testing <- mvrnorm(n = 1000, mu = zeros_j, Sigma = Gamma)# x testin set
y_testing = vector()
for (n in 1:1000)
{
  yval = rnorm(1, x_testing[n,]%*%betas, 4) #obtain y for each row in x testing set
  y_testing = c(y_testing,yval) #construct y testing set
}

hist(x_learning,xlab = "x")
hist(y_learning,xlab = "y")
hist(x_testing, xlab = "x")
hist(y_testing, xlab = "y")
```

**X-learning set**

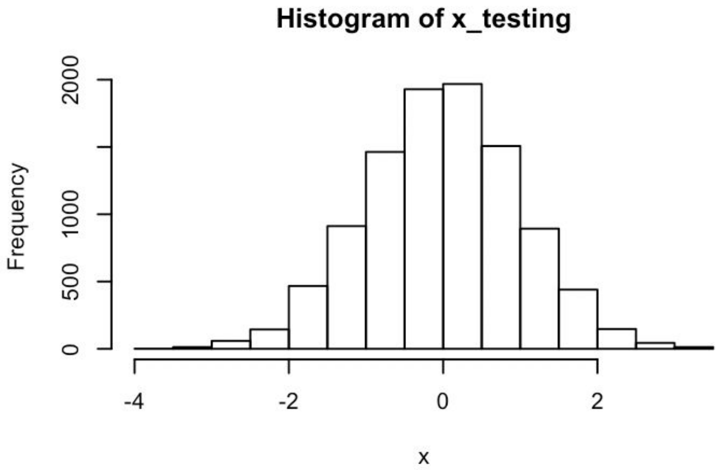
- Mean = 0.1209188
- Standard deviation = 1.051537
- 


**Y-learning set**

- Mean =  -0.1352824
- Standard deviation =  12.2593
- 

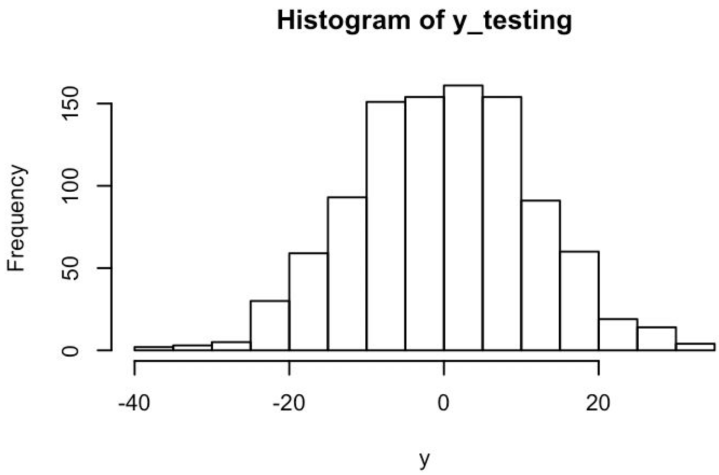

**X-testing set**

- Mean = -0.005178464
- Standard deviation = 0.9929037




- 

**Y-learning set**
- Mean = 0.0122951
- Standard deviation = 11.54817




- 

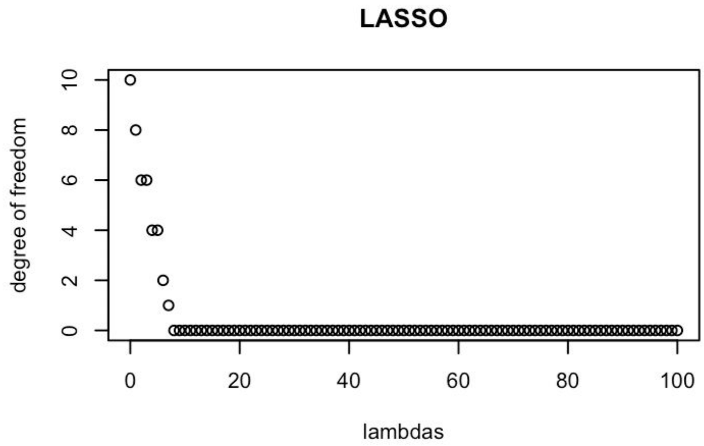
b)

lambdas <- seq(0,100)#lambdas from 0 to 100
#perform the three regression on x learning set and y learning set, with the lambda set

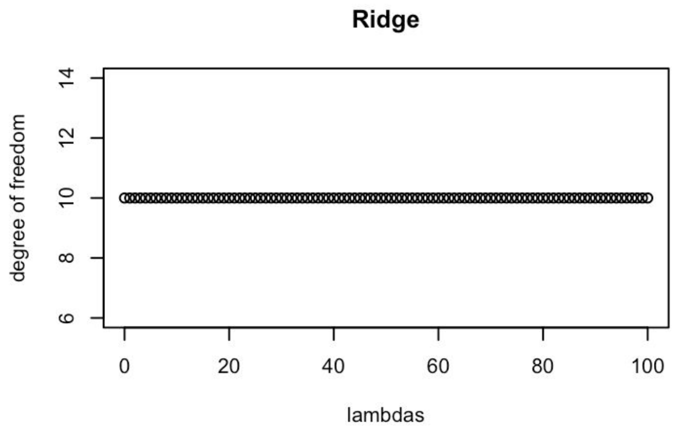
fit.lasso <- glmnet(x_learning, y_learning,family="gaussian", lambda=lambdas, alpha=1)
fit.ridge <- glmnet(x_learning, y_learning,family="gaussian", lambda=lambdas, alpha=0)
fit.elnet <- glmnet(x_learning, y_learning,family="gaussian", lambda=lambdas, alpha=0.5)

**Plot the effective degrees of freedom versus the shrinkage parameter**
- 

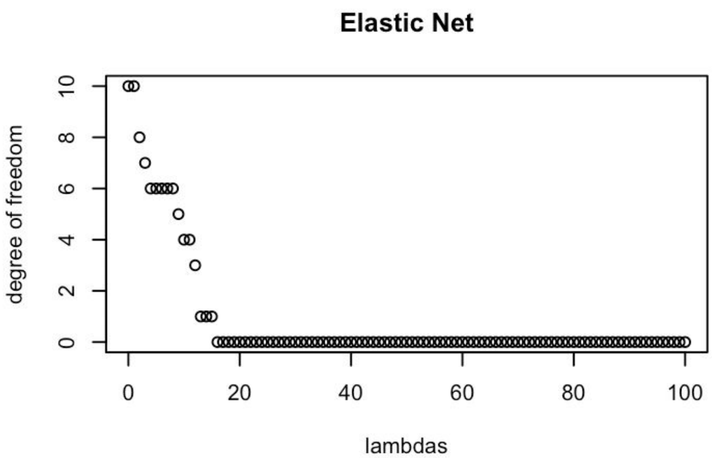
- plot(as.numeric(unlist(fit.lasso[5])),as.numeric(unlist(fit.lasso[3])), ylab = "degree of freedom",main = "LASSO")




  - LASSO selects “useful”, significant predictors, so the degree of freedom decreases as lambda increases.
- plot(as.numeric(unlist(fit.ridge[5])),as.numeric(unlist(fit.ridge[3])), ylab = "degree of freedom",main = "Ridge")



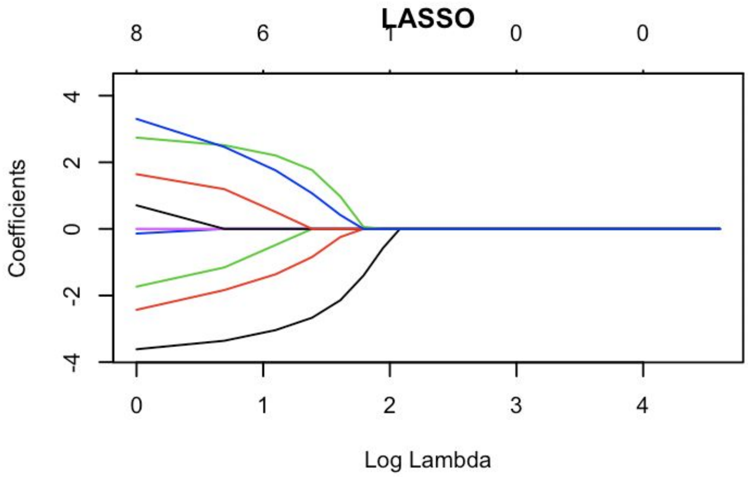

  - Ridge shrinks the amplitude of the predictors, rather than selecting significant predictors, so the degree of freedom remains constant because no predictor is thrown out.
- plot(as.numeric(unlist(fit.elnet[5])),as.numeric(unlist(fit.elnet[3])), ylab = "degree of freedom",main = "Elastic Net")



- Elastic net shrinks and selects predictors. As a combination of Ridge and LASSO, the degree of freedom decreases more slowly in Elastic net than it decreases in LASSO.

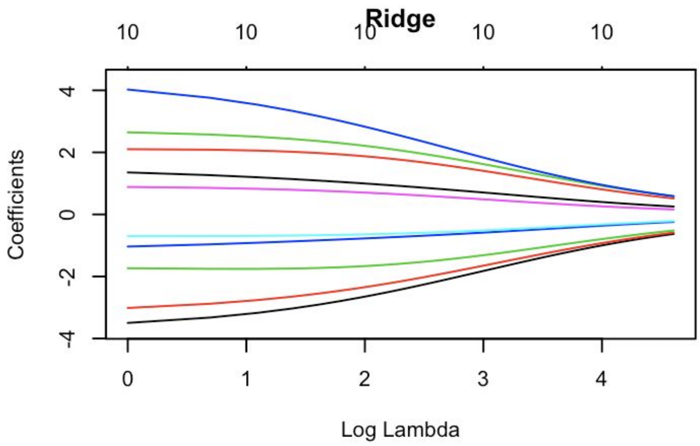
**Plot the effective degrees of freedom versus the shrinkage parameter**

- plot(fit.lasso, xvar="lambda", main = “LASSO”)



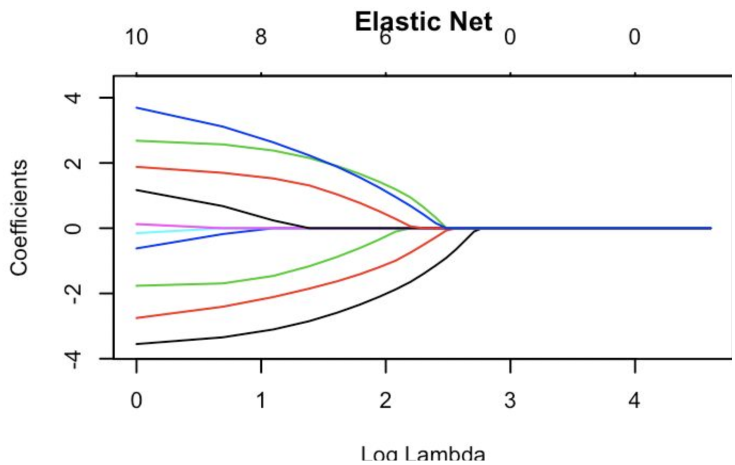

  - 

- plot(fit.ridge, xvar="lambda", main="Ridge")




  - 

- plot(fit.elnet, xvar="lambda", main="Elastic Net")

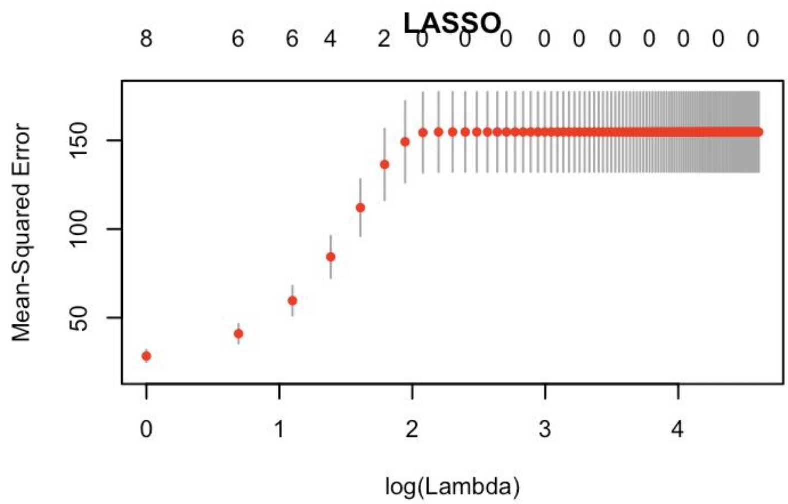



○

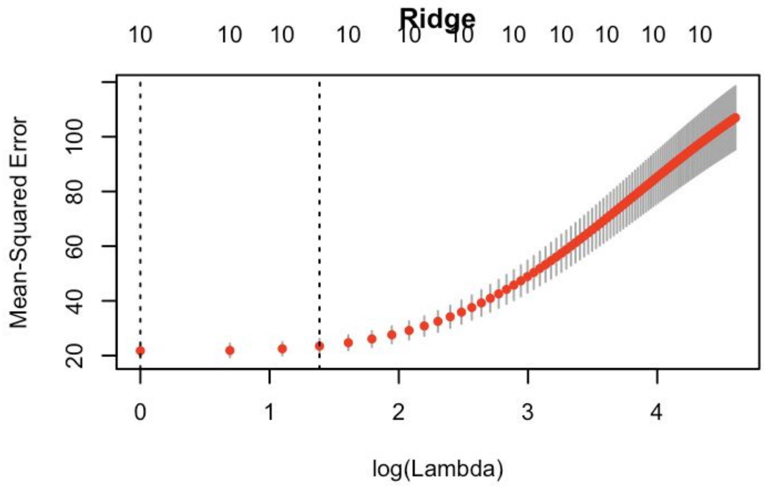
**Plot MSE versus the shrinkage parameter**

fit.lasso.mse <- cv.glmnet(x_learning, y_learning, type.measure = "mse",family="gaussian", lambda=lambdas, alpha=1)
fit.ridge.mse <- cv.glmnet(x_learning, y_learning, type.measure = "mse",family="gaussian", lambda=lambdas, alpha=0)
fit.elnet.mse <- cv.glmnet(x_learning, y_learning, type.measure = "mse", lambda=lambdas, alpha=0.5,family="gaussian")

- plot(fit.lasso.mse, main ="LASSO")

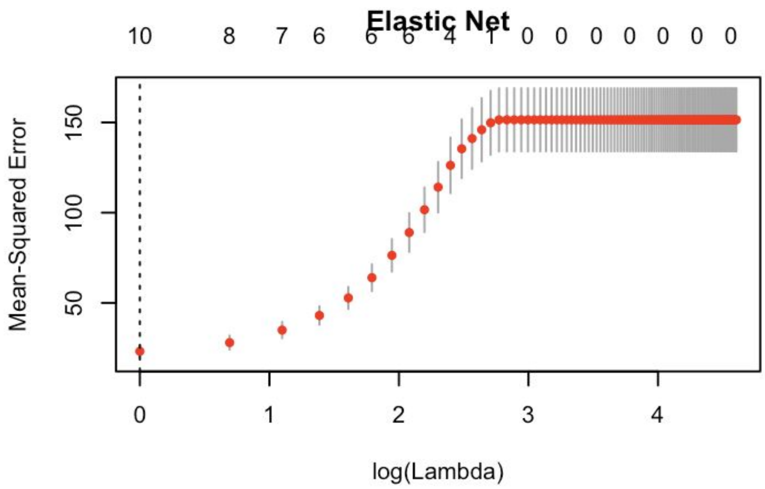



  ○
  ○ The above plot is the cross-validation curve (red dotted line), and upper and lower standard deviation curves along the lambda sequence (error bars). MSE value for lambda = 0 is not on the plot above (log(0) = -inf). But “fit.lasso.mse$lambda.min” provides a lambda value that minimizes risk. The “fit.lasso.mse$lambda.min” gives lambda = 0 as the shrinkage parameter that minimizes risk.
- plot(fit.ridge.mse, main="Ridge")




- ○
- ○ The above plot is the cross-validation curve (red dotted line), and upper and lower standard deviation curves along the lamba sequence (error bars). Two selected lambdas are indicated by the vertical dotted lines. Since the MSE is lowest when log(lambda) = 0, the MSE is lowest when lambda = 1. Thus, lambda =1 is the parameter that minimizes risk.

- plot(fit.elnet.mse, main="Elastic Net")



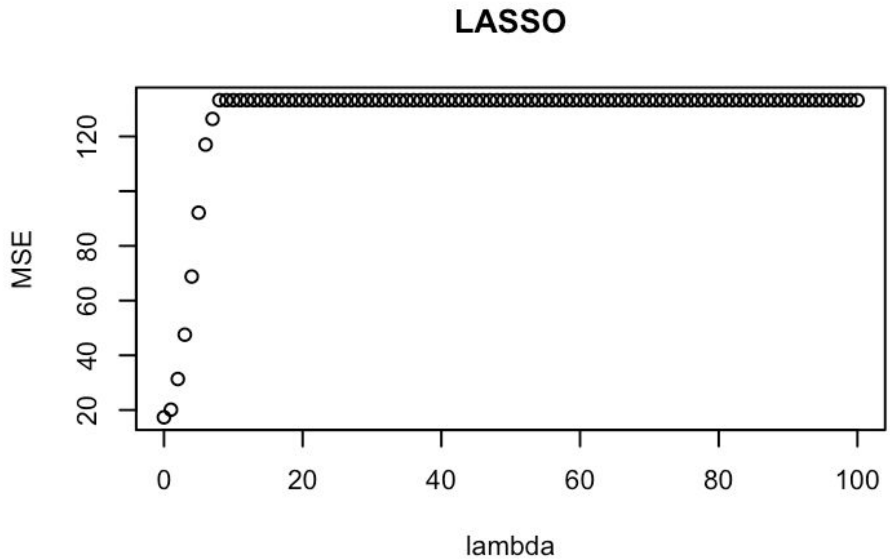

- ○
- ○ The above plot is the cross-validation curve (red dotted line), and upper and lower standard deviation curves along the lamba sequence (error bars). One selected lambdas are indicated by the vertical dotted line. Since the MSE is lowest when log(lambda) = 0, the MSE is lowest when lambda = 1. Thus, lambda =1 is the parameter that minimizes risk.

c)
```
#instantiate mse vectors to store mse values
mse0= vector()
mse1= vector()
mse2= vector()

#obtain predictions y for each lambda
for (n in lambdas)
{
```

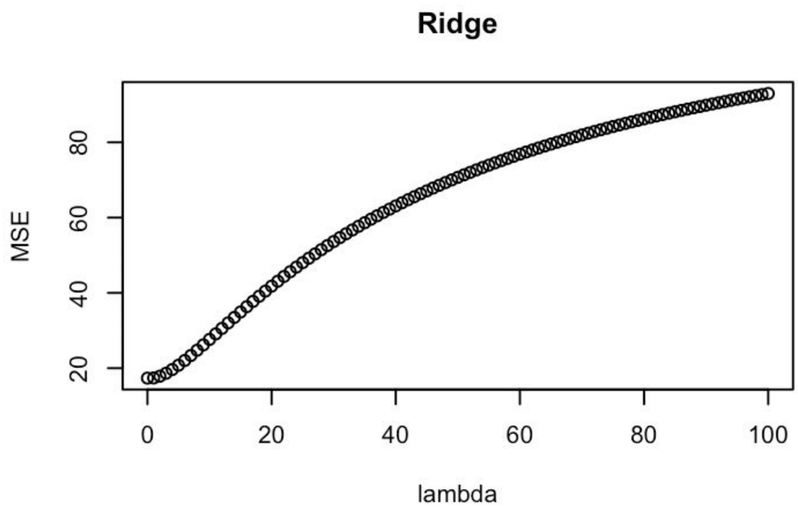
```
yhat0 <- predict.cv.glmnet(fit.lasso.mse, s=n, newx=x_testing)
yhat1 <- predict.cv.glmnet(fit.ridge.mse, s=n, newx=x_testing)
yhat2 <- predict.cv.glmnet(fit.elnet.mse, s=n, newx=x_testing)

#store mse for each lambda
mse0 = c(mse0, mean((y_testing - yhat0)^2))
mse1 <- c(mse1, mean((y_testing - yhat1)^2))
mse2 <- c(mse2, mean((y_testing - yhat2)^2))
}
```

**Plot risk versus the shrinkage parameter**

- plot(lambdas,mse0,xlab="lambda",ylab="MSE", main="LASSO")

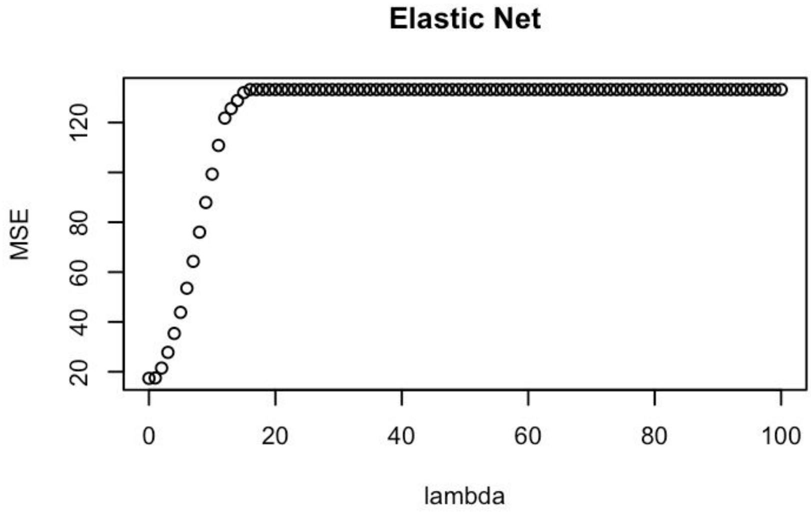



  - The above graph plots MSE for different lambdas. MSE increases rapidly and quickly converges as lambda increases
- plot(lambdas,mse1,xlab="lambda",ylab="MSE", main="Ridge")

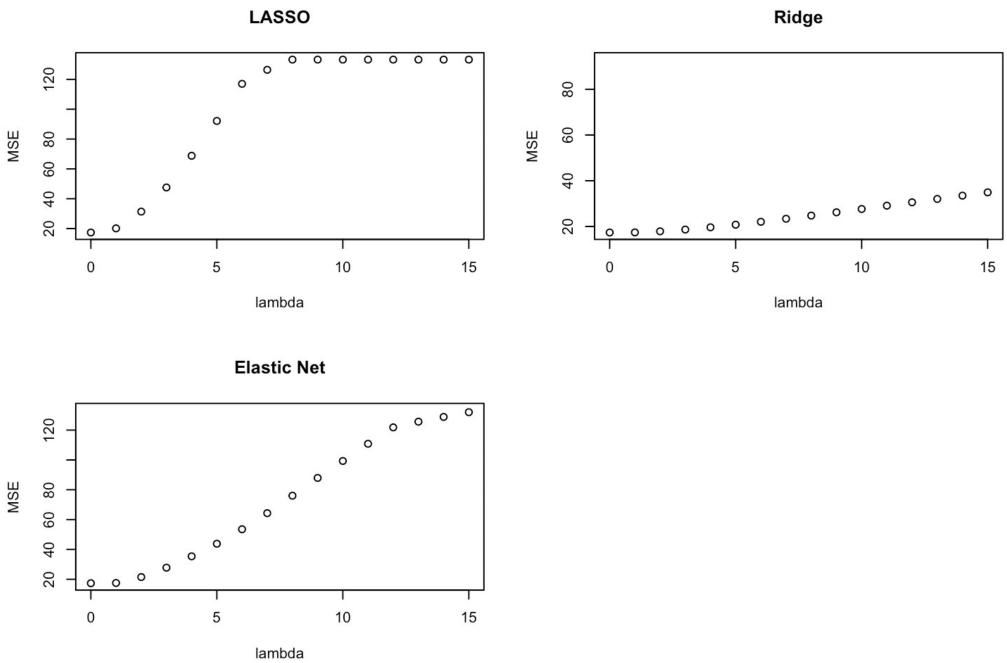



  - The above graph plots MSE for different lambdas. Unlike in LASSO regression, MSE in Ridge increases gradually as lambda increases

- plot(lambdas,mse2,xlab="lambda",ylab="MSE", main="Elastic Net")




    - 
    - The above graph plots MSE for different lambdas. As in LASSO, MSE rapidly increases and converges as lambda increases. However, the amplitude of increase is not greater than that of LASSO.




- 
    - The above three graphs show MSE for lambdas from 0 to 15. MSE values are lowest when lambda = 0 for all three model. So this suggests that “the optimal” seems to be simply least square regression (lambda = 0)