HW4.

1. T/F.

(a) T.

$\hat{\theta} = (X^T X)^{-1} X^T Y.$

(b). F. $var(\theta^{RR}) = w_\lambda \sigma^2 (X^T X)^{-1} w_\lambda^T$ $\quad w_\lambda \downarrow$ as $\lambda \uparrow$

(c). T. Lasso bias increased relative to OLS

(d) F. Lasso will regularize some of the $\hat{\beta}$s to 0.

? (e). T

(f). T $M\bar{v} = \lambda \bar{v} = M\lambda \bar{v}$

(g) True

Symmetric $HH^T = H^T H.$ $\quad X(X^TX)^{-1}X^T [X(X^TX)^{-1}X^T]^T = (X(X^TX)^{-1}X^T X^T((X^TX)^{-1})^T X$

idempotent $H^2 = H.$

PSD. $H^T A H \geq 0.$ easy to show

(h) F. $trace(Q) \leq (n-d)$

2. OLS theoretical properties

2.1. $y = X \cdot \hat{\beta} + e \quad e = y - X \cdot \hat{\beta}$

$e^T e = (y - X\hat{\beta})^T (y - X\hat{\beta}) = y^T y - \beta^T X^T y - y^T X \hat{\beta} + \hat{\beta}^T X^T X \hat{\beta}$

$= y^T y - 2\beta' X' y + \hat{\beta}^T X^T X \hat{\beta}.$

$\Rightarrow (X^T X)\hat{\beta} = X^T y$

By definition. $(X^T X)^{-1}(X^T X) = I.$

$(X^T X)^{-1}(X^T X)\hat{\beta} = (X^T X)^{-1} X^T y$

$\Rightarrow \hat{\beta} = (X^T X)^{-1} X^T y.$

$\hat{\beta} = (X^T X)^{-1} X^T (X\hat{\beta} + e)$

$\hat{\beta} = (X^T X)^{-1} X^T (X\hat{\beta} + e) = (X^T X)^{-1}(X^T X)\beta + (X^T X)^{-1} X^T e$

$= \beta + (X^T X)^{-1} X^T e$

since $E(X^T e) = 0$. and $E(e) = 0$.

$E(\hat{\beta}) = \beta + 0 = \beta$. So OLS estimator is unbiased.

since $\hat{\beta} - E(\hat{\beta}) = (X^TX)^{-1}X^T e$.

$$var(\hat{\beta}) = E(\hat{\beta} - E(\hat{\beta}))^2$$
$$= E[((X^TX)^{-1}X^Te)^T (X^TX)^{-1}X^Te] \;??$$
$$= E[(X^TX)^{-1}X^Te e^T X(X^TX)^{-1}$$
$$= \sigma^2(X^TX)^{-1}.$$

thus $\hat{\beta} \sim N(\beta, \sigma^2(X^TX)^{-1})$ is proved. unbiasedness has been proved in previous derivation.

2.2.

$$RSS = S_{yy} - \frac{S_{xy}^2}{S_{xx}}$$
and $\hat{\beta} = \frac{S_{xy}}{S_{xx}}$.
$$\Rightarrow RSS = S_{yy} - \frac{S_{xy}^2}{S_{xx}} = S_{yy} - \left(\frac{S_{xy}^2}{S_{xx}^2}\right)S_{xx} = S_{yy} - \hat{\beta}^2 S_{xx}$$
$$E(\hat{\beta}^2 S_{xx}) = (var(\hat{\beta}) + E(\hat{\beta})^2)S_{xx}$$
$$= \left(\frac{\sigma^2}{S_{xx}} + \beta^2\right)S_{xx} = \sigma^2 + \beta^2 S_{xx}$$
$$y_i - \bar{y} = (\alpha + \beta x_i + \epsilon_i) - (\alpha + \beta\bar{x} + \bar{\epsilon})$$
$$= \beta(x_i - \bar{x}) + (\epsilon_i - \bar{\epsilon})$$
$$(y_i - \bar{y})^2 = \beta^2(x_i - \bar{x})^2 + 2\beta(x_i - \bar{x})(\epsilon_i - \bar{\epsilon}) + \sum_{i=1}^{n}(\epsilon_i - \bar{\epsilon})^2.$$
$$E(S_{yy}) = \beta^2 \cdot S_{xx} + E\left(\sum_{i=1}^{n}(\epsilon_i - \bar{\epsilon})^2\right)$$
$$= \beta^2 \cdot S_{xx} + (n-1)\sigma^2.$$
$$\Rightarrow E(RSS) = (n-2)\sigma^2.$$

2.3.

$$\text{let } \hat{\beta} = (X^TX)^{-1}X^Ty + \Delta$$
$$E(\hat{\beta}) = E((X^TX)^{-1}X^T(X\beta^* + \epsilon) + \Delta)$$
$$= E(\beta^* + \Delta)$$
$$= \beta^* + E(\Delta)$$

For $\hat{\beta}$ to be unbiased

$$var(\hat{\beta}) = E(\hat{\beta}^2) - E(\tilde{\beta})^2$$
$$= E((X^TX)^{-1}X^Ty)^2 + \Delta^2 + (X^TX)^{-1}X^Ty\Delta) - E(\beta^* + \Delta)^2$$
$$= E(\hat{\beta}^2) - E(\beta^*)^2 + E(\Delta^2 + (X^TX)^{-1}X^Ty\Delta).$$

where $E(\Delta^2 + (X^TX)^{-1}X^Ty\Delta) = E(\Delta^2) + (X^TX)^{-1}X^TyE(\Delta) = E(\Delta^2) \geq 0$

2.4.

3. Theory of Ridge Regression.

3.1 objective function: $f(\beta) = (y - X\beta)^T (y - X\beta) + \lambda \beta^T \beta$. for non-negative λ.

since $\lambda \geq 0$, it has positive square root $\nu^2 = \lambda$.

$\tilde{X} = \begin{pmatrix} X \\ \nu I \end{pmatrix}$. $\tilde{y} = \begin{pmatrix} y \\ 0_p \end{pmatrix}$.

Normal Equations become. $(\tilde{X}^T \tilde{X}) \beta = \tilde{X}^T \tilde{y}$

$\Rightarrow (X^T X + \nu^2 I) \beta = X^T \cdot y$

$\Rightarrow (X^T X + \lambda I) \beta = X^T \cdot y$.

$\Rightarrow \hat{\beta}_{rr} = (X^T X + \lambda I)^{-1} X^T y$.

3.2. Show Ridge Regression Distribution.

Define $W = (X^T X + \lambda I)^{-1}$.

$E(\hat{\beta}_{rr}) = E[(X^T X + \lambda I)^{-1} X^T y] = W_\lambda \beta$

$var(\hat{\beta}_{rr}) = var(W_\lambda \hat{\beta})$

$= W_\lambda \sigma^2 (X^T X)^{-1} W_\lambda^T$

$= \sigma^2 (X^T X + \lambda I_d)^{-1} X^T X (X^T X)^{-1} X^T X ((X^T X + \lambda I_d)^{-1})^T$

$= \sigma^2 (X^T X + \lambda I_d)^{-1} X^T X ((X^T X + \lambda I_d)^T)^{-1}$

$= \sigma^2 W_\lambda (X^T X + \lambda I_d)^{-1}$

Scanned by CamScanner

3.3. $X^TX = UDU^T$. $D_{ii} = d_i$.

(a). Inverting (X^TX) can be computationally expensive as $O(p^3)$

Rather, we use SVD.

where $X = \cancel{UDV^T}$ VDU^T — $V = (v_1, v_2 \dots v_p)$ is $n\times p$ orthogonal matrix. $U^T = \begin{pmatrix} u_1^T \\ u_2^T \\ u_p^T \end{pmatrix}$ is $p\times p$ orthogonal matrix.

thus $X^TX = (VDU^T)^T VDU^T = UD^2U^T$

$XX^T = (VDU^T)(VDU^T)^T = VD\underline{U^T U}D^TV^T = VD^2V^T$

$$\hat{\beta}_{RR} = (X^TX + \lambda I)^{-1} \cdot X^Ty$$
$$= (UD^2U^T + \lambda)^{-1} \cdot UDV^Ty$$
$$= (UD^2U^T + \lambda U^TU)^{-1} UDV^Ty$$
$$= (U(D^2+\lambda)U^T))^{-1} \cdot UDV^Ty \quad ?$$
$$= U(D^2+\lambda)^{-1} \underline{U^TU}DV^Ty$$
$$= U(D^2+\lambda)^{-1} DV^Ty$$

$$E(\hat{\beta}_{RR}) - \beta = E\left(U(D^2+\lambda)^{-1}DV^Ty\right) - \beta$$
$$= E(U(D^2+\lambda)^{-1}DV^T \cdot X \cdot \hat{\beta}) - \beta$$
$$= E(U(D^2+\lambda)^{-1}DV^T \cdot VDU^T \cdot \hat{\beta}) - \beta.$$
$$= E(U(D^2+\lambda)^{-1}D^2U^T \hat{\beta}) - \beta$$
$$= [U(D^2+\lambda)^{-1}D^2 - 1]U^T \cdot \beta$$
$$= -\lambda(U(D^2+\lambda)^{-1}U^T\beta.$$

since for OLS estimator $E(\hat{\beta}) = 0$.

thus squared bias of $\hat{\beta}_{RR}$ is derived as:

$$\|E(\hat{\beta}_{rr}) - \beta\|_2^2 = \sum_{i=1}^{d} \frac{\lambda^2}{(d_i+\lambda)^2}(v_i)^2.$$

thus proved.

(b). variance of $\hat{\beta}_{RR}$ is expressed as,

$$E\|\hat{\beta}_{rr} - E(\hat{\beta}_{rr})\|_2^2$$

suppose $X = VDU^T$. $\Rightarrow X\cdot U = VDU^T \cdot U = VD$.

suppose new design matrix $\tilde{X} = X\cdot U$.

since $\hat{\beta}_{rr} = (\tilde{X}^T\tilde{X} + \lambda I)^{-1}\tilde{X}^Ty$

$$= ((X\cdot U)^T(X\cdot U) + \lambda I)^{-1} (X\cdot U)^T \cdot y$$
$$= [(VD)^T V\cdot D + \lambda \cdot I)^{-1} (V\cdot D)^T \cdot y]$$
$$= (D^2 + \lambda I)^{-1} \cdot D^T V^T \cdot y$$
$$\cancel{= d_{ii}}$$

$$\Rightarrow E\|\hat{\beta}_{rr} - E(\hat{\beta}_{rr})\|_2^2 = \sigma^2 \sum_{i=1}^{d} \frac{d_{ii}^2}{(d_{ii}^2+\lambda)^2}.$$

(c) when $\lambda = 0$. $\begin{cases} \text{square bias} = 0. \\ \text{variance} = \sigma^2 \end{cases}$

when $\lambda \to \infty$ $\begin{cases} \text{square bias} \to \infty \\ \text{variance} \to 0. \end{cases}$

As λ increase, square bias is increasing and variance is decreasing.

3.5. show ~~tha~~ $E[\|\hat{\theta}_\lambda - \theta^*\|_2^2] = \text{trace}(M(\lambda))$ where $M(\lambda) = E[(\hat{\theta}_\lambda - \theta^*)(\hat{\theta}_\lambda - \theta^*)^T]$

also show $MSE(\hat{\theta}_\lambda) =$ squared bias + scalar variance.

$\text{trace}(M(\lambda)) = \text{trace}\, E[(\hat{\theta}_\lambda - \theta^*)(\hat{\theta}_\lambda - \theta^*)^T]$

$= E(tr(\hat{\theta}_\lambda - \theta^*)(\hat{\theta}_\lambda - \theta^*)^T)$

$= E(\Sigma(\hat{\theta}_{\lambda i} - \theta^*_i)^2)$

$MSE(\hat{\theta}_\lambda) = E(\|\hat{\theta}_\lambda - \theta^*\|_2^2)$

$= E(\Sigma(\hat{\theta}_{\lambda i} - \theta^*_i)^2)$.

$var(x) = E(x^2) - [E(x)]^2$

$MSE(\hat{\theta}_\lambda) -$ squared bias $=$ scalar-variance

thus proved

3.6. Show ~~that~~. $MSE(\theta^{ols}) = \text{trace}(M(0))$.

and $MSE(\hat{\theta}_\lambda) < MSE(\theta^{ols})$. $\exists \lambda > 0$.

$\lambda = 0$, ridge $\Leftrightarrow$ OLS.

$MSE(\theta^{ols}) = \text{trace}(M(0))$

$-MSE(\hat{\theta}_\lambda) + MSE(\theta^{ols}) > 0$

$= \lambda^2 (X^TX + \lambda I)^{-1} \theta^* \theta^{*T} (X^TX + \lambda I)^{-1} > 0$.

$\Rightarrow 2\sigma^2 I - \lambda^2 \theta^* \theta^{*T} > 0$

$\Rightarrow 0 < \lambda < \sqrt{\dfrac{2\sigma^2 I}{\theta^* \theta^{*T}}}$

4.

4.1 $\nabla^2 f(x) = \nabla^2 \frac{L}{2} x^2$

$= L \geq 0$ PSD

~~$x_1 = x^{(0)}$~~

$x_1 = x^{(0)} - \gamma \nabla f(x_1) = x^{(0)} - \gamma L x^{(0)}$

4.2. $\gamma \in \{\frac{1}{L}, \frac{2}{L}\}$. $x^{(1)} \in \{0, -x^{(0)}\}$.

if $\gamma \neq \frac{2}{L}$, update becomes smaller in absolute values.

if $\gamma = \frac{2}{L}$. update fluctuates around $x^{(0)}$.

if $\gamma \in [0, \frac{2}{L})$, updates becomes smaller in absolute values.

$|x^{(0)}| \cdot ((\gamma - \frac{1}{L}) \cdot L)^t \leq \varepsilon$

$\Rightarrow t \leq \log | \frac{\varepsilon}{|x^{(0)}|} - (\gamma - \frac{1}{L}) \cdot L)$

4.3. $|x^{(0)}| \to |x^{(0)}| - k|$.

everything else remain same.

4.4 $\nabla x_1^2 f(x) = L$

$\nabla x_2^2 f(x) = m$

$\nabla x_1 x_2 = 0$

$\begin{pmatrix} a \\ b \end{pmatrix} = \gamma \begin{pmatrix} L & 0 \\ 0 & m \end{pmatrix}$.

5.

Scanned by CamScanner

stat154_hw4

5.High dimensional regression

5.1.

```
#if XTX matrix is not invertible(when X is not full rank), the objective function is no longer strictly
#Residuals of ols solutions is orthogonal of all columns of design matrix.
```

5.2

```
#Yes.(XTX+LAMBDA*I)^(-1) always exists.Thus ridge regression guarantees unique solution.
```

5.3

5.4

```
n=1000
d=5000

x=matrix(rnorm(5000000,0,1),nrow=n,ncol=d)
e=as.matrix(rnorm(n,0,0.25))
beta=as.matrix(c(rep(1,15),rep(0,d-15)))
dim(beta)
```

```
## [1] 5000    1
```

```
dim(x)
```

```
## [1] 1000 5000
```

```
dim(e)
```

```
## [1] 1000    1
```

```
y=x%*%beta+e
dim(y)
```

```
## [1] 1000    1
```

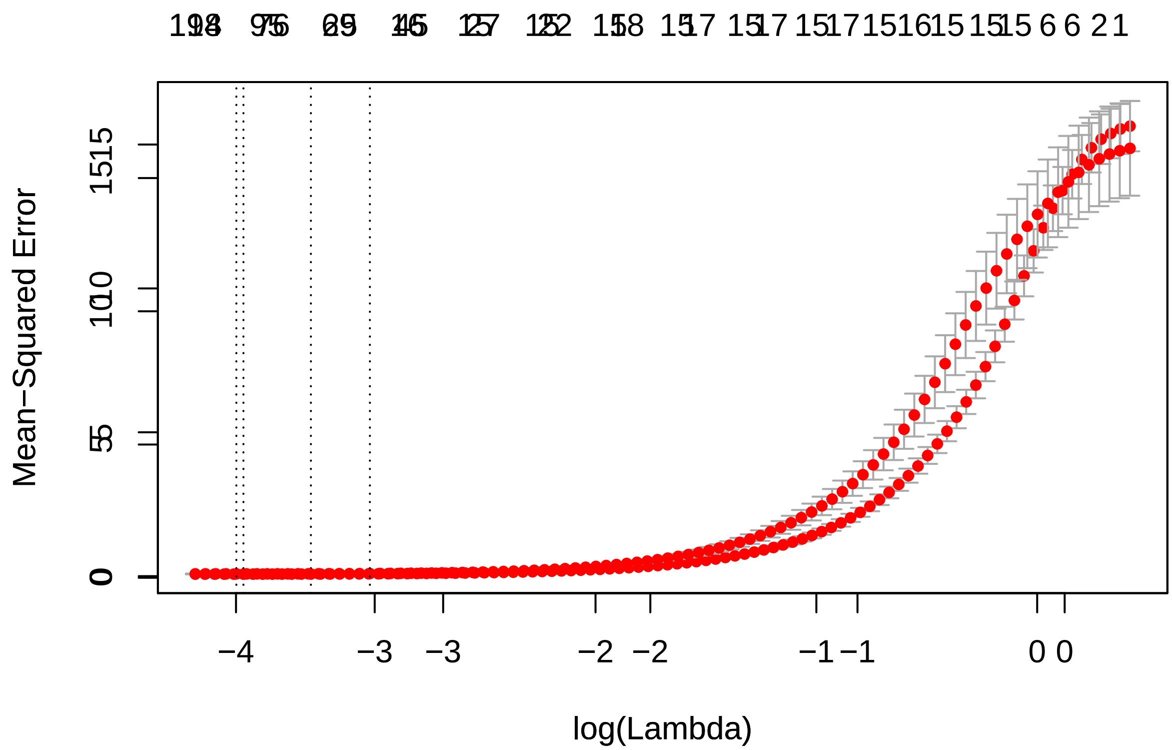
5.5

```
train_id=sample(nrow(x),size=0.8*nrow(x))
x.train=x[train_id,]
x.test=x[-train_id,]
```

```
y.train=y[train_id,]
y.test=y[-train_id,]
```

5.6

```
lm.rr=glmnet(x.train,y.train,alpha=0)
#on training data
cvfit=cv.glmnet(x.train,y.train)
plot(cvfit)
opt_lambda <- cvfit$lambda.min
#on test data
cvfit2=cv.glmnet(x.test,y.test)
par(new=T)
plot(cvfit2)
```




```
opt_lambda2 <- cvfit2$lambda.min
opt_lambda
```

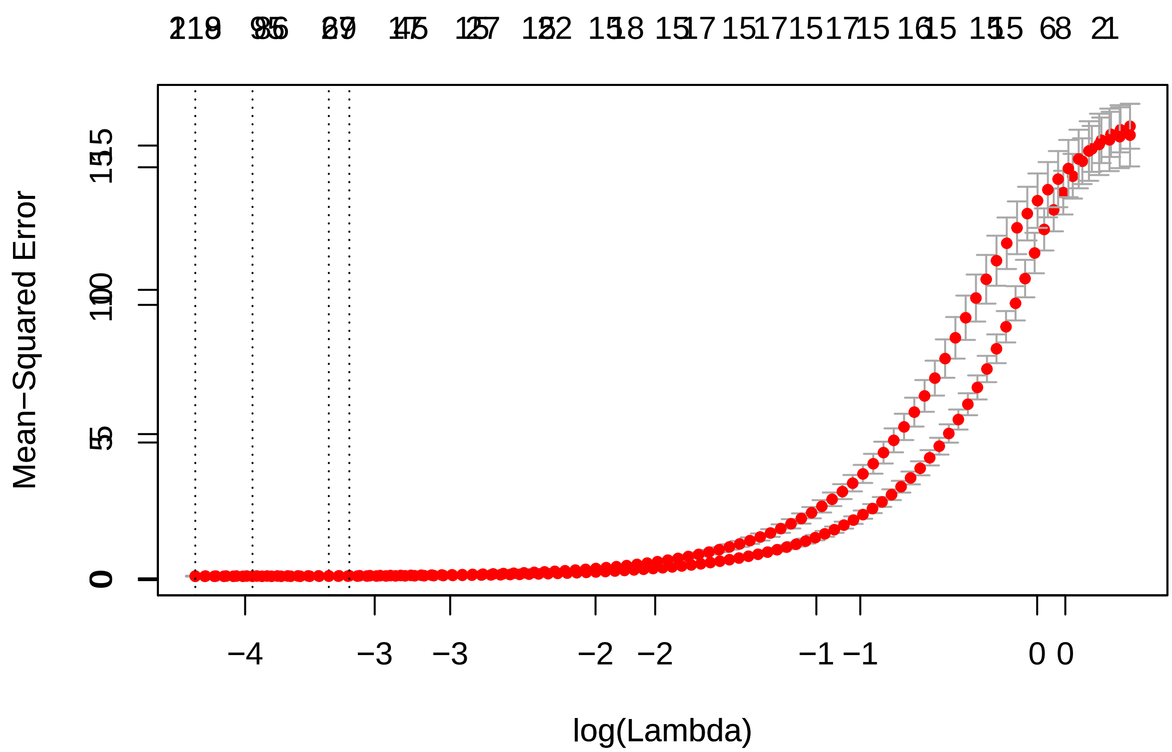
```
## [1] 0.01898677
```

```
opt_lambda2
```

```
## [1] 0.02661034
```

5.7

```r
lm.lasso=glmnet(x.train,y.train,alpha=1)
#on training data
cvfit=cv.glmnet(x.train,y.train)
plot(cvfit)
opt_lambda <- cvfit$lambda.min
#on test data
cvfit2=cv.glmnet(x.test,y.test)
par(new=T)
plot(cvfit2)
```




```r
opt_lambda2 <- cvfit2$lambda.min
opt_lambda
```

```
## [1] 0.01898677
```

```r
opt_lambda2
```

```
## [1] 0.02209237
```

5.8

```r
#Ridge seems to be better algorithm.
#optimal lambda which minimize MSE occurs at around 0.01-0.03.
```

6.Regression analysis on Ames data

```
Ames <- read.delim("AmesHousing.txt", header = TRUE, sep = "\t", dec = ".")
```

```
continuousVar <- colnames(Ames)[grep("Frontage|SF|Area|Porch",
colnames(Ames))]

AmesTiny <- Ames[, c(continuousVar,
c("Overall.Qual",
"Overall.Cond","Neighborhood",
"SalePrice"))]
# check NA
colSums(is.na(AmesTiny))
```

```
##    Lot.Frontage        Lot.Area    Mas.Vnr.Area    BsmtFin.SF.1
##             490               0              23               1
##    BsmtFin.SF.2     Bsmt.Unf.SF   Total.Bsmt.SF     X1st.Flr.SF
##               1               1               1               0
##     X2nd.Flr.SF Low.Qual.Fin.SF     Gr.Liv.Area     Garage.Area
##               0               0               0               1
##     Wood.Deck.SF   Open.Porch.SF  Enclosed.Porch     X3Ssn.Porch
##               0               0               0               0
##    Screen.Porch       Pool.Area    Overall.Qual    Overall.Cond
##               0               0               0               0
##    Neighborhood       SalePrice
##               0               0
```

```
AmesTiny$Garage.Area[is.na(AmesTiny$Garage.Area)] = 0
# change factor variable to actual factor in the data frame
AmesTiny$Overall.Qual <- factor(AmesTiny$Overall.Qual)
AmesTiny$Overall.Cond <- factor(AmesTiny$Overall.Cond)
# fill the continuous variable with column mean
for(i in 1:ncol(AmesTiny)){
AmesTiny[is.na(AmesTiny[,i]), i] <- mean(AmesTiny[,i], na.rm = TRUE)
}
```

```
## Warning in mean.default(AmesTiny[, i], na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(AmesTiny[, i], na.rm = TRUE): argument is not
## numeric or logical: returning NA

## Warning in mean.default(AmesTiny[, i], na.rm = TRUE): argument is not
## numeric or logical: returning NA
```

```
# divide the data into training and test datasets
testSize <- floor(nrow(AmesTiny)*0.1)
testIndex <- sample(seq_len(nrow(AmesTiny)), size = testSize)
AmesTinyTrain <- AmesTiny[-testIndex, ]
AmesTinyTest <- AmesTiny[testIndex, ]
```

6.1

```r
#1.MSE function
mse=function(beta,x,y){
  tmp=t(y-x%*%beta)%*%(y-x%*%beta)/dim(x)[1]
  return(tmp)
}
```

```r
#2. R^2 function
r2=function(beta,x,y){
  tmp=(cor(y,x%*%beta))^2
  return(tmp)
}
```

```r
#3.

y=I(log(AmesTinyTrain$SalePrice+1))
x1=cbind(rep(1,2637),AmesTinyTrain$Gr.Liv.Area)
lm1=lm(log(SalePrice+1) ~ Gr.Liv.Area,data=AmesTinyTrain)
beta1=lm1$coefficients

dumQ=dummy(AmesTinyTrain$Overall.Qual,drop=F)
dumC=dummy(AmesTinyTrain$Overall.Cond,drop=F)
dim(dumQ)
```

```
## [1] 2637   10
```

```r
dim(dumC)
```

```
## [1] 2637    9
```

```r
dumQ=dumQ[,-1]
dumC=dumC[,-1]

x2=cbind(x1,AmesTinyTrain$Garage.Area)
lm2=lm(log(SalePrice+1) ~ Gr.Liv.Area+Garage.Area,data=AmesTinyTrain)
beta2=lm2$coefficients

x3=cbind(x2,AmesTinyTrain$Open.Porch.SF)
lm3=lm(log(SalePrice+1) ~ Gr.Liv.Area+Garage.Area+Open.Porch.SF,data=AmesTinyTrain)
beta3=lm3$coefficients

x4=cbind(x3,AmesTinyTrain$Lot.Area)
lm4=lm(log(SalePrice+1) ~ Gr.Liv.Area+Garage.Area+Open.Porch.SF+Lot.Area,data=AmesTinyTrain)
beta4=lm4$coefficients

x5=cbind(x4,dumQ)
lm5=lm(log(SalePrice+1) ~ Gr.Liv.Area+Garage.Area+Open.Porch.SF+Lot.Area+dumQ,data=AmesTinyTrain)
beta5=lm5$coefficients

x6=cbind(x5,dumC)
lm6=lm(log(SalePrice+1) ~ Gr.Liv.Area+Garage.Area+Open.Porch.SF+Lot.Area+dumQ+dumC,data=AmesTinyTrain)
beta6=lm6$coefficients

x7=cbind(x6,log(AmesTinyTrain$Gr.Liv.Area+1))
lm7=lm(log(SalePrice+1) ~ Gr.Liv.Area+Garage.Area+Open.Porch.SF+Lot.Area+dumQ+dumC+I(log(Gr.Liv.Area+1))
```

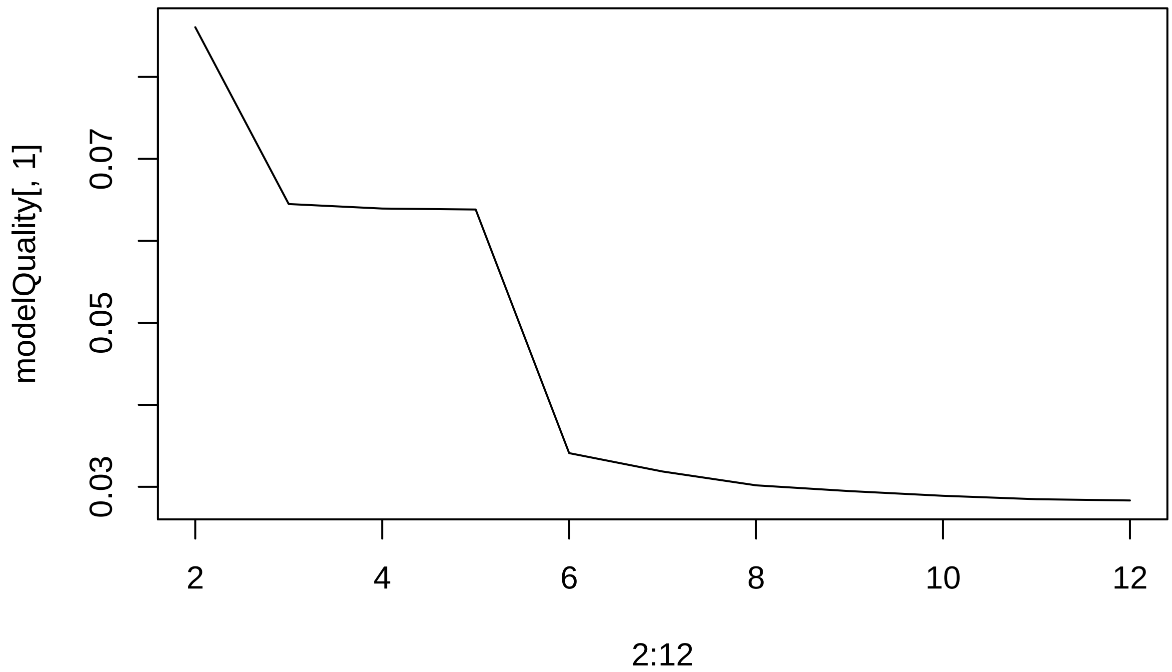
```r
beta7=lm7$coefficients

x8=cbind(x7,log(AmesTinyTrain$Gr.Liv.Area+1)^2)
lm8=lm(log(SalePrice+1) ~ Gr.Liv.Area+Garage.Area+Open.Porch.SF+Lot.Area+dumQ+dumC+I(log(Gr.Liv.Area+1)
beta8=lm8$coefficients

x9=cbind(x8,log(AmesTinyTrain$Gr.Liv.Area+1)^3)
lm9=lm(log(SalePrice+1) ~ Gr.Liv.Area+Garage.Area+Open.Porch.SF+Lot.Area+dumQ+dumC+I(log(Gr.Liv.Area+1)
beta9=lm9$coefficients

x10=cbind(x9,log(AmesTinyTrain$Gr.Liv.Area+1)^4)
lm10=lm(log(SalePrice+1) ~ Gr.Liv.Area+Garage.Area+Open.Porch.SF+Lot.Area+dumQ+dumC+I(log(Gr.Liv.Area+1
beta10=lm10$coefficients

x11=cbind(x10,log(AmesTinyTrain$Gr.Liv.Area+1)^5)
lm11=lm(log(SalePrice+1) ~ Gr.Liv.Area+Garage.Area+Open.Porch.SF+Lot.Area+dumQ+dumC+I(log(Gr.Liv.Area+1
beta11=lm11$coefficients


x=list(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11)
beta=list(beta1,beta2,beta3,beta4,beta5,beta6,beta7,beta8,beta9,beta10,beta11)
modelQuality=matrix(0,nrow=11,ncol=2)
for(i in 1:11){
  modelQuality[i,1]=mse(beta[[i]],x[[i]],y)
  modelQuality[i,2]=r2(beta[[i]],x[[i]],y)
}
modelQuality
```

```
##             [,1]      [,2]
##  [1,] 0.08606239 0.4812453
##  [2,] 0.06449069 0.6112721
##  [3,] 0.06393848 0.6146007
##  [4,] 0.06382338 0.6152944
##  [5,] 0.03410639 0.7944183
##  [6,] 0.03186574 0.8079242
##  [7,] 0.03017804 0.8180971
##  [8,] 0.02947377 0.8223422
##  [9,] 0.02890086 0.8257955
## [10,] 0.02848414 0.8283073
## [11,] 0.02833350 0.8292153
```

```r
#which model has smallest training MSE
#model11 has the smallest MSE
```

6.1.4

```r
plot(x=2:12,y=modelQuality[,1],type='l')
```




```
#trend shows that as number of predictor increases, training model MSE decreases.
```

6.1.5

```
dumQ=dummy(AmesTinyTest$Overall.Qual,drop=F)
dumC=dummy(AmesTinyTest$Overall.Cond,drop=F)
dim(dumQ)
```

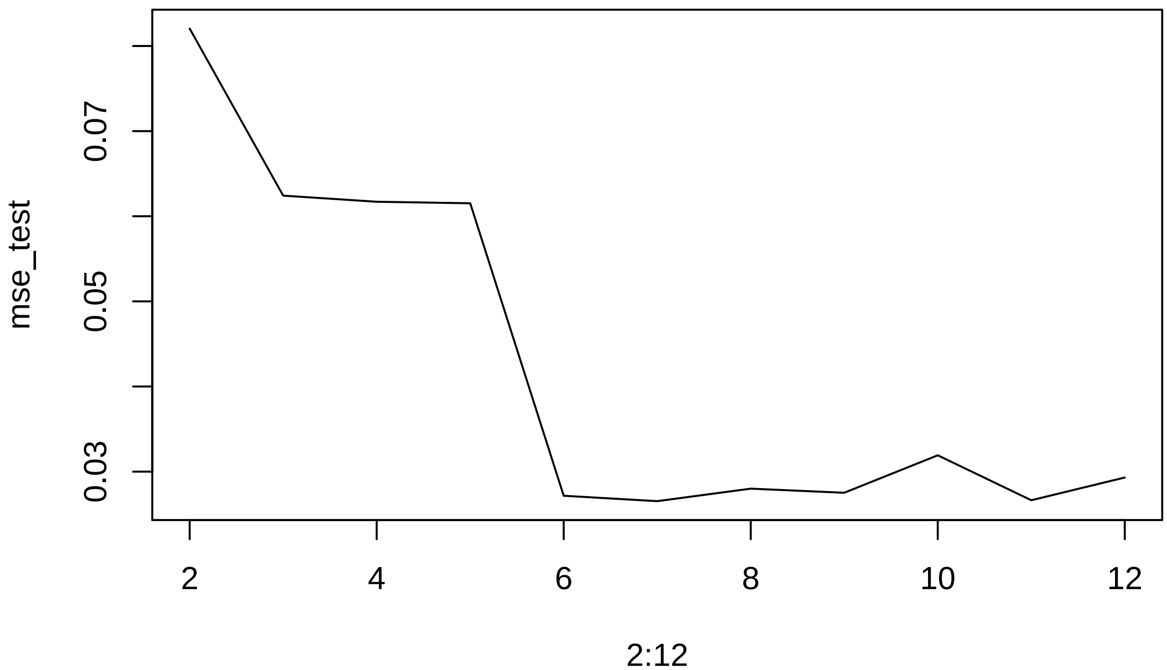
```
## [1] 293  10
```

```
dim(dumC)
```

```
## [1] 293   9
```

```
dumQ=dumQ[,-1]
dumC=dumC[,-1]

y_hat1=predict.lm(lm1,AmesTinyTest,type='response')
y_hat2=predict.lm(lm2,AmesTinyTest,type='response')
y_hat3=predict.lm(lm3,AmesTinyTest,type='response')
y_hat4=predict.lm(lm4,AmesTinyTest,type='response')
y_hat5=predict.lm(lm5,AmesTinyTest,type='response')
y_hat6=predict.lm(lm6,AmesTinyTest,type='response')
y_hat7=predict.lm(lm7,AmesTinyTest,type='response')
y_hat8=predict.lm(lm8,AmesTinyTest,type='response')
y_hat9=predict.lm(lm9,AmesTinyTest,type='response')
y_hat10=predict.lm(lm10,AmesTinyTest,type='response')
y_hat11=predict.lm(lm11,AmesTinyTest,type='response')
y=log(AmesTinyTest$SalePrice+1)
mse1=t(y-y_hat1)%*%(y-y_hat1)/293
mse2=t(y-y_hat2)%*%(y-y_hat2)/293
mse3=t(y-y_hat3)%*%(y-y_hat3)/293
```

```r
mse4=t(y-y_hat4)%*%(y-y_hat4)/293
mse5=t(y-y_hat5)%*%(y-y_hat5)/293
mse6=t(y-y_hat6)%*%(y-y_hat6)/293
mse7=t(y-y_hat7)%*%(y-y_hat7)/293
mse8=t(y-y_hat8)%*%(y-y_hat8)/293
mse9=t(y-y_hat9)%*%(y-y_hat9)/293
mse10=t(y-y_hat10)%*%(y-y_hat10)/293
mse11=t(y-y_hat11)%*%(y-y_hat11)/293
mse_test=c(mse1,mse2,mse3,mse4,mse5,mse6,mse7,mse8,mse9,mse10,mse11)
plot(mse_test,x=2:12,type='l')
```




```r
#MSE does not always decrease in test data, model 9 has the lowest MSE.
```

6.2

```r
set.seed(123456)
valSize <- floor(nrow(AmesTinyTrain)*0.2)
valIndex <- sample(seq_len(nrow(AmesTinyTrain)), size = valSize)
# actual training data
AmesTinyActTrain <- AmesTinyTrain[-valIndex, ]
AmesTinyActVal <- AmesTinyTrain[valIndex, ]
```

6.2.2

```r
y=I(log(AmesTinyActTrain$SalePrice+1))
x1=cbind(rep(1,2637),AmesTinyActTrain$Gr.Liv.Area)
```

```
## Warning in cbind(rep(1, 2637), AmesTinyActTrain$Gr.Liv.Area): number of
```

```
## rows of result is not a multiple of vector length (arg 2)
```

```
lm1=lm(log(SalePrice+1) ~ Gr.Liv.Area,data=AmesTinyActTrain)
beta1=lm1$coefficients

dumQ=dummy(factor(AmesTinyActTrain$Overall.Qual))
dumC=dummy(factor(AmesTinyActTrain$Overall.Cond))
dumQ=dumQ[,-1]
dumC=dumC[,-1]
dim(dumQ)
```

```
## [1] 2110    9
```

```
dim(dumC)
```

```
## [1] 2110    8
```

```
x2=cbind(x1,AmesTinyActTrain$Garage.Area)
```

```
## Warning in cbind(x1, AmesTinyActTrain$Garage.Area): number of rows of
## result is not a multiple of vector length (arg 2)
```

```
lm2=lm(log(SalePrice+1) ~ Gr.Liv.Area+Garage.Area,data=AmesTinyActTrain)
beta2=lm2$coefficients

x3=cbind(x2,AmesTinyActTrain$Open.Porch.SF)
```

```
## Warning in cbind(x2, AmesTinyActTrain$Open.Porch.SF): number of rows of
## result is not a multiple of vector length (arg 2)
```

```
lm3=lm(log(SalePrice+1) ~ Gr.Liv.Area+Garage.Area+Open.Porch.SF,data=AmesTinyActTrain)
beta3=lm3$coefficients

x4=cbind(x3,AmesTinyActTrain$Lot.Area)
```

```
## Warning in cbind(x3, AmesTinyActTrain$Lot.Area): number of rows of result
## is not a multiple of vector length (arg 2)
```

```
lm4=lm(log(SalePrice+1) ~ Gr.Liv.Area+Garage.Area+Open.Porch.SF+Lot.Area,data=AmesTinyActTrain)
beta4=lm4$coefficients

#x=list(x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11)
#beta=list(beta1,beta2,beta3,beta4,beta5,beta6,beta7,beta8,beta9,beta10,beta11)
#trainmse=matrix(0,nrow=11,ncol=2)
#for(i in 1:11){
# trainmse[i,1]=mse(beta[[i]],x[[i]],y)
#}
```

6.3

```
folds <- createFolds(log(AmesTinyTrain$SalePrice+1), k = 5)
set.seed(123)
```

6.4.1

```r
levels(AmesTiny[,"Neighborhood"]) <- c(1:28)
AmesTiny$Neighborhood <- as.numeric(AmesTiny$Neighborhood)
AmesTiny$Overall.Cond <- as.numeric(AmesTiny$Overall.Cond)
AmesTiny$Overall.Qual <- as.numeric(AmesTiny$Overall.Qual)

model = glmnet(x = as.matrix(AmesTiny[,-22]), y = log(AmesTiny[,22]+1), alpha = 0, lambda = 1)

coef(model)
```

```
## 22 x 1 sparse Matrix of class "dgCMatrix"
##                            s0
## (Intercept)      1.117286e+01
## Lot.Frontage     6.705277e-04
## Lot.Area         1.447722e-06
## Mas.Vnr.Area     1.213241e-04
## BsmtFin.SF.1     5.270285e-05
## BsmtFin.SF.2     1.171623e-05
## Bsmt.Unf.SF      1.970148e-05
## Total.Bsmt.SF    7.768222e-05
## X1st.Flr.SF      7.957522e-05
## X2nd.Flr.SF      5.095720e-05
## Low.Qual.Fin.SF -8.158883e-05
## Gr.Liv.Area      8.373209e-05
## Garage.Area      1.879323e-04
## Wood.Deck.SF     1.580119e-04
## Open.Porch.SF    2.363803e-04
## Enclosed.Porch  -1.503185e-04
## X3Ssn.Porch      1.133202e-04
## Screen.Porch     1.279524e-04
## Pool.Area       -1.793611e-05
## Overall.Qual     4.426085e-02
## Overall.Cond     4.134911e-03
## Neighborhood     1.101701e-03
```

6.4.2