```
## Warning: package 'glmnet' was built under R version 3.4.4

## Warning: package 'Matrix' was built under R version 3.4.4

## Warning: package 'ggplot2' was built under R version 3.4.4

## Warning: package 'caret' was built under R version 3.4.4
```

#5.4

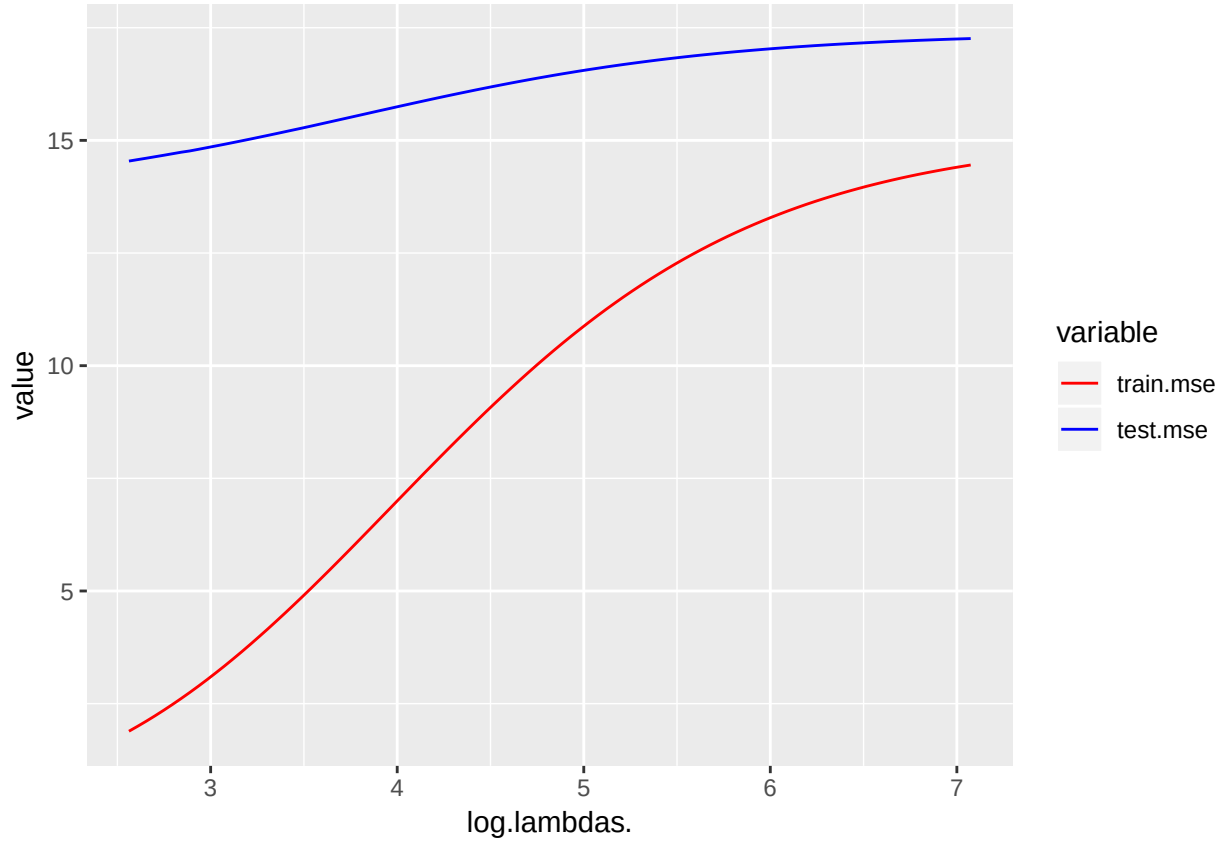
```
n=1000
d=5000
x = matrix(rnorm(5000*1000, mean = 0, sd = 1),nrow = 1000)
err = rnorm(1000,0,sqrt(0.25))
beta = c(rep(1,15),rep(0,4985))
y = x %*% beta + err
```

#5.5

```
training = x[1:800,]
training.label = y[1:800]
test = x[801:1000,]
test.label = y[801:1000]
```

#5.6

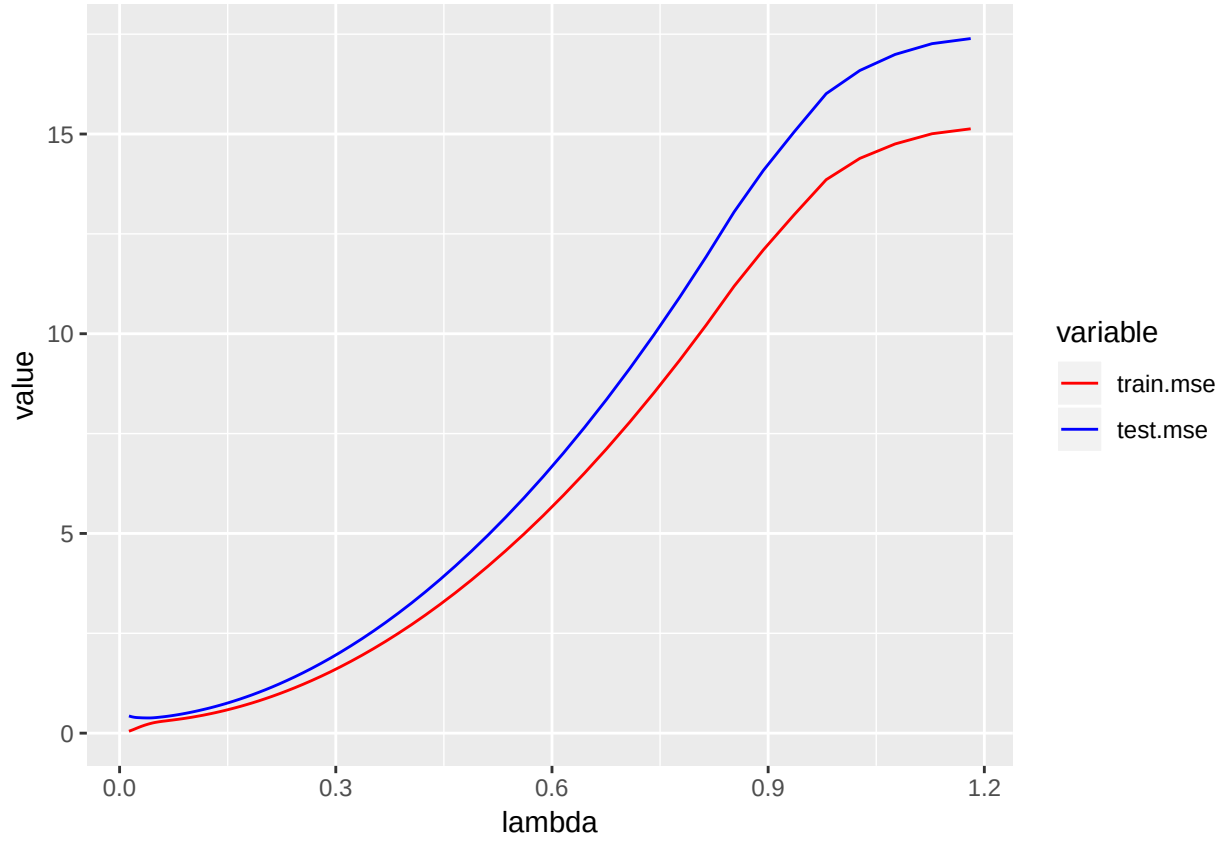
```
mod = cv.glmnet(training, training.label, alpha=0, type.measure = "mse")

lambdas = mod$lambda
training.error = rep(0,length(lambdas))
test.error = rep(0,length(lambdas))

for (i in 1:length(lambdas)) {
  fit.ridge= glmnet(training, training.label, alpha=0, lambda = lambdas[i])
  training.error[i] = mean((training.label - predict(fit.ridge, training))^2)
  test.error[i] = mean((test.label - predict(fit.ridge, test))^2)
}

dat <- data.frame("log(lambdas)" = log(lambdas), train.mse = training.error, test.mse = test.error)
dat.m <- melt(dat, id.vars = "log.lambdas.")
ggplot(dat.m, aes(log.lambdas., value, colour = variable)) +
  geom_line() +
  scale_colour_manual(values = c("red", "blue"))
```




```
ridge.mse = test.error[which(test.error == min(test.error))]
ridge.lambda.min = lambdas[which(test.error == min(test.error))]
```

#5.7

```
mod = cv.glmnet(training, training.label, alpha=1, type.measure = "mse")

lambdas = mod$lambda
training.error = rep(0,length(lambdas))
test.error = rep(0,length(lambdas))

for (i in 1:length(lambdas)) {
  fit.lasso= glmnet(training, training.label, alpha=1, lambda = lambdas[i])
  training.error[i] = mean((training.label - predict(fit.lasso, training))^2)
  test.error[i] = mean((test.label - predict(fit.lasso, test))^2)
}

dat <- data.frame(lambda = lambdas, train.mse = training.error, test.mse = test.error)
dat.m <- melt(dat, id.vars = "lambda")
ggplot(dat.m, aes(lambda, value, colour = variable)) +
  geom_line() +
  scale_colour_manual(values = c("red", "blue"))
```




```
lasso.mse = test.error[which(test.error == min(test.error))]
lasso.lambda.min = lambdas[which(test.error == min(test.error))]
```

#5.8

```
ridge.lambda.min
```

```
## [1] 12.96238
```

```
ridge.mse
```

```
## [1] 14.54114
```

```
lasso.lambda.min
```

```
## [1] 0.03958523
```

```
lasso.mse
```

```
## [1] 0.3805818
```

#6

#6.1-6.2

```
MSE = function(y, x, beta){
  sum((y-(x %*% beta))^2)/length(y)
}
R2 = function(y, x, beta){
  cor(y, x %*% beta)^2
}
```

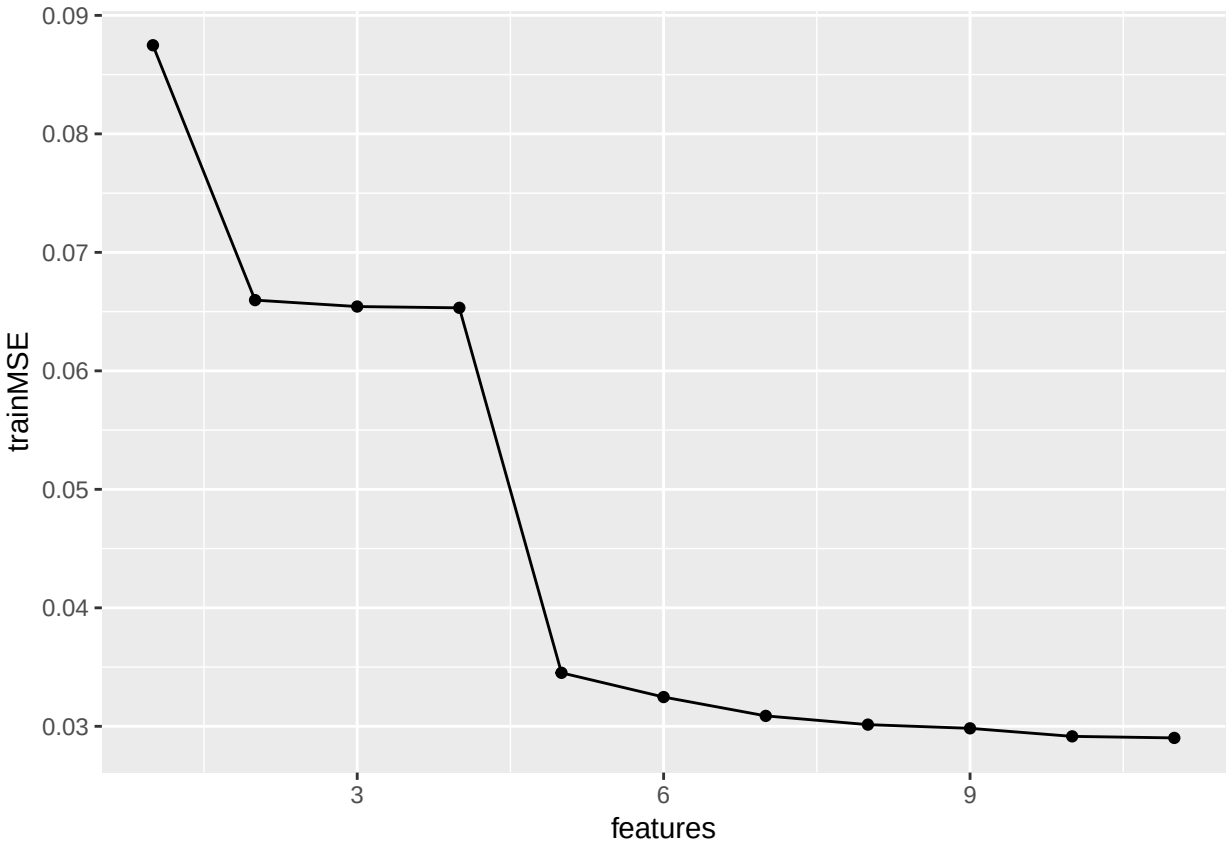
#6.3

```
test.mse= rep(0, 11)
modelQuality = matrix(rep(0,22),nrow=11)
modelQualityRImp =matrix(rep(0,22),nrow=11)
for (i in 1:length(Xnames)) {
  formula = as.formula(paste("log(SalePrice + 1) ~ ", paste(Xnames[1:i], collapse= "+")))
  lmod = lm(formula, data = AmesTinyTrain)
  training.label = log(AmesTinyTrain$SalePrice+1)
  mse = MSE(training.label, model.matrix(lmod),lmod$coefficients)
  r2 = R2(training.label,model.matrix(lmod),lmod$coefficients)
  modelQuality[i,1] = mse
  modelQuality[i,2] = r2
  r.out = summary(lmod)
  modelQualityRImp[i,1] = sum(r.out$residuals^2)/nrow(model.matrix(lmod))
  modelQualityRImp[i,2] = r.out$r.squared

  test.label = log(AmesTinyTest$SalePrice+1)
  test.mse[i] = sum((test.label-predict(lmod, AmesTinyTest))^2)/nrow(AmesTinyTest)
}

modelQuality = as.data.frame(modelQuality)
modelQualityRImp = as.data.frame(modelQualityRImp)
colnames(modelQuality) <- c("MSE", "R2")
colnames(modelQualityRImp) <- c("MSE", "R2")
modelQuality
```

```
##           MSE        R2
## 1  0.08748009 0.4745942
## 2  0.06596899 0.6037900
## 3  0.06542675 0.6070466
## 4  0.06531829 0.6076980
## 5  0.03451592 0.7926972
## 6  0.03247054 0.8049818
## 7  0.03087920 0.8145394
## 8  0.03014235 0.8189649
## 9  0.02982983 0.8208419
## 10 0.02915284 0.8249079
## 11 0.02902088 0.8257005
```

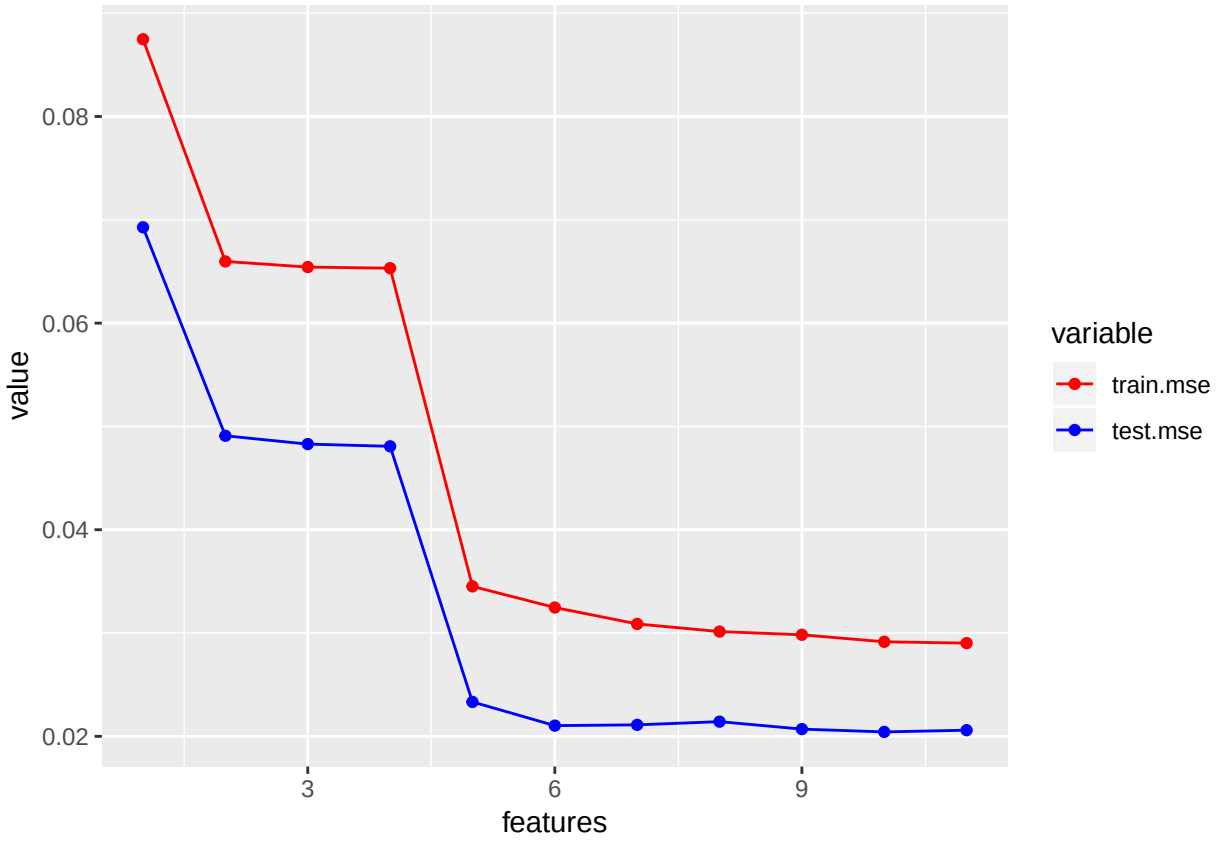
```
modelQualityRImp
```

```
##           MSE        R2
## 1  0.08748009 0.4745942
## 2  0.06596899 0.6037900
## 3  0.06542675 0.6070466
## 4  0.06531829 0.6076980
## 5  0.03451592 0.7926972
## 6  0.03247054 0.8049818
## 7  0.03087920 0.8145394
## 8  0.03014235 0.8189649
## 9  0.02982983 0.8208419
## 10 0.02915284 0.8249079
## 11 0.02902088 0.8257005
```



#6.4

the training MSE is decreasing as the number of predictors increase

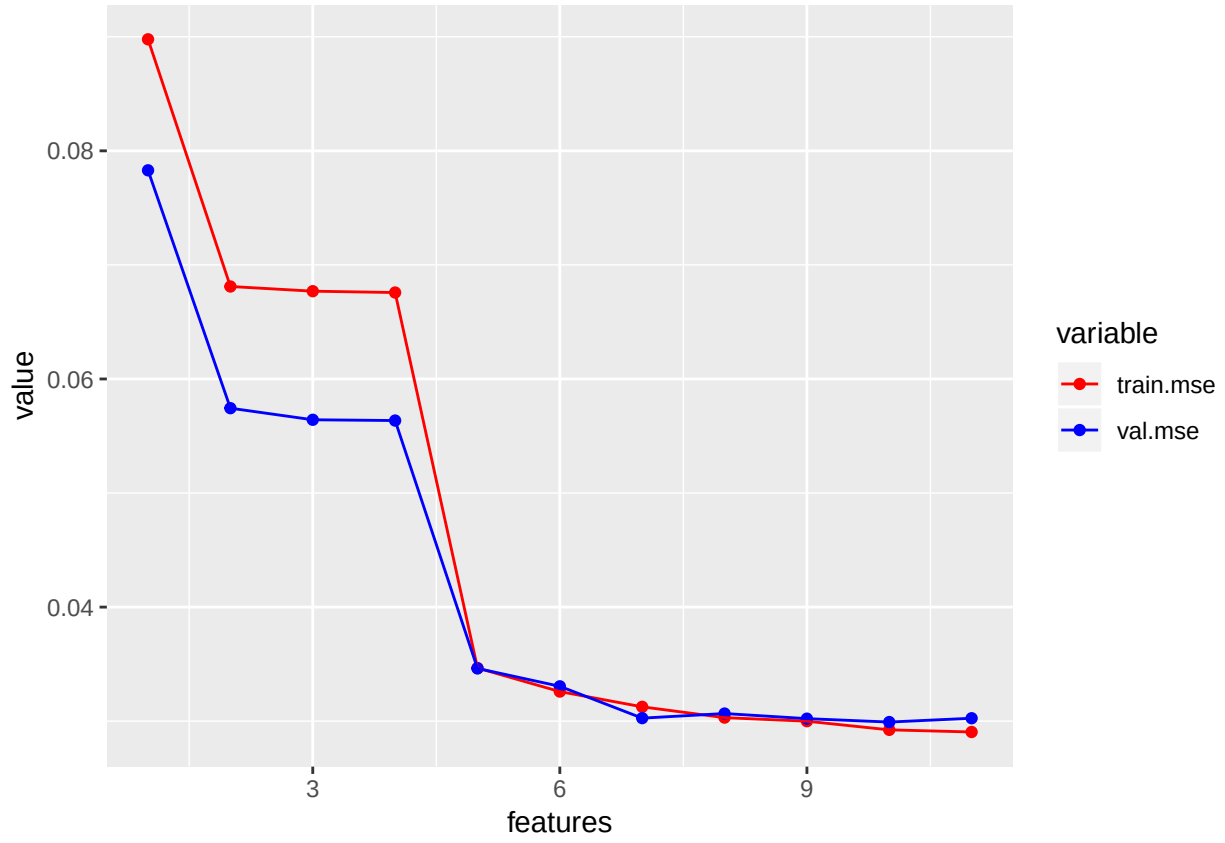


#6.5

test MSE is decreseing in [1:10] and starts increasing at 11 and the 10th model gives the lowest test MSE

#6.2.2

```
modelQualitySingleVal = matrix(rep(0,22),nrow=11)
for (i in 1:length(Xnames)) {
  formula = as.formula(paste("log(SalePrice + 1) ~ ", paste(Xnames[1:i], collapse= "+")))
  lmod = lm(formula, data = AmesTinyActTrain)
  training.label = log(AmesTinyActTrain$SalePrice+1)
  mse = MSE(training.label, model.matrix(lmod),lmod$coefficients)
  modelQualitySingleVal[i,1] = mse

  val.label = log(AmesTinyActVal$SalePrice+1)
  modelQualitySingleVal[i, 2] = sum((val.label-predict(lmod, AmesTinyActVal))^2)/nrow(AmesTinyActVal)
}

dat <- data.frame(features = 1:length(Xnames), train.mse = modelQualitySingleVal[,1], val.mse = modelQua
dat.m <- melt(dat, id.vars = "features")
ggplot(dat.m, aes(features, value, colour = variable)) +
  geom_point() + geom_line(aes(features, value, colour = variable))+
  scale_colour_manual(values = c("red", "blue"))
```




```
which(modelQualitySingleVal[,2] == min(modelQualitySingleVal[,2]))
```

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

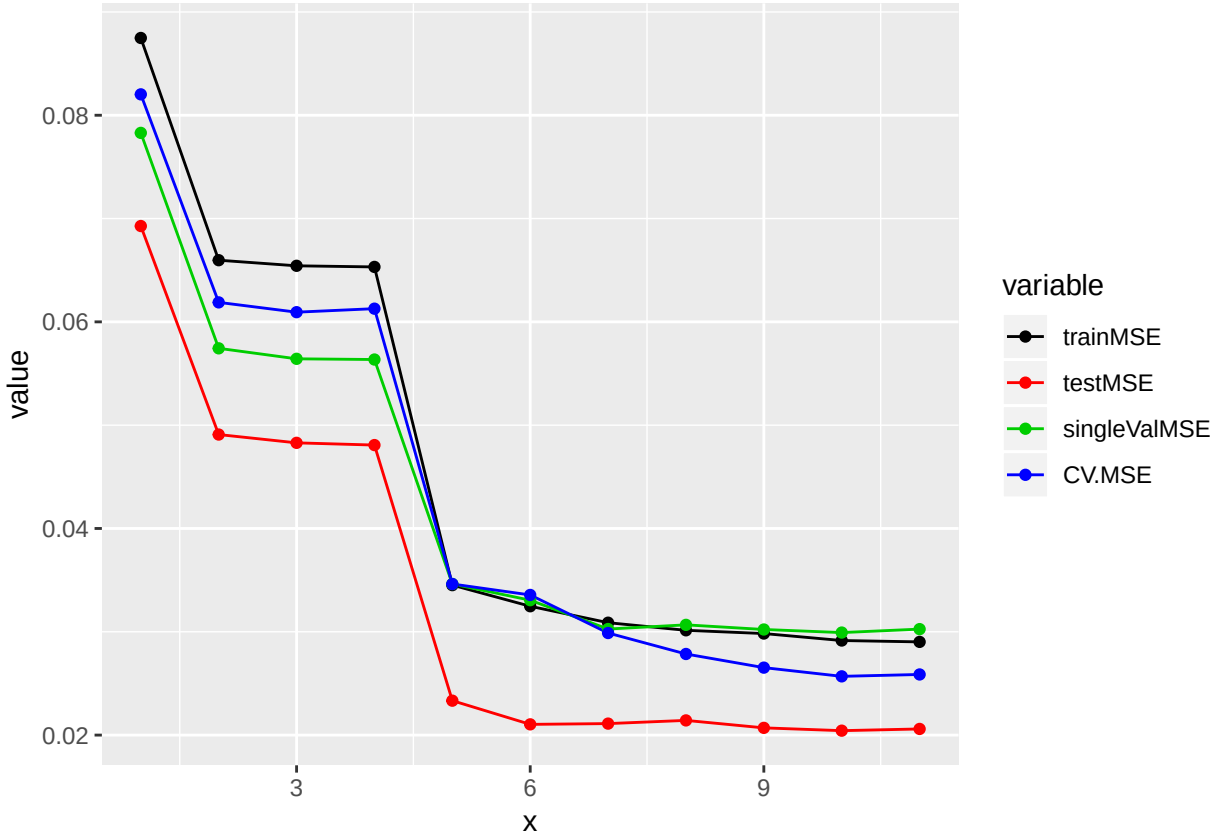
the 10th model gives the lowest validation MSE

#6.3.1

```
set.seed(123)
folds <- createFolds(AmesTinyTrain$SalePrice, k = 5)
cvMSE = matrix(rep(0,55),nrow=11)

for (i in 1:length(Xnames)) {
  formula = as.formula(paste("log(SalePrice + 1) ~ ", paste(Xnames[1:i], collapse= "+")))
  for (f in 1:5) {
    train = AmesTinyTrain[-folds[[f]], ]
    lmod = lm(formula, data = train)
    test = AmesTinyTrain[folds[[1]], ]
    pred<- predict(lmod, test)
    true_y<- log(test$SalePrice + 1)
    mse1 = 1/length(folds[[1]]) * sum((pred-true_y)^2)
    cvMSE[i,f] = mse1
  }
}
```

#6.3.2

```
dat <- data.frame(x = 1:length(Xnames), trainMSE = modelQuality$MSE, testMSE = test.mse, singleValMSE =
dat.m <- melt(dat, id.vars = "x")
ggplot(dat.m, aes(x, value, colour = variable)) +
  geom_line() + geom_point(aes(x, value, colour = variable))+
  scale_colour_manual(values = 1:5)
```




```
#10th model
# which(dat$CV.MSE == min(dat$CV.MSE))
```

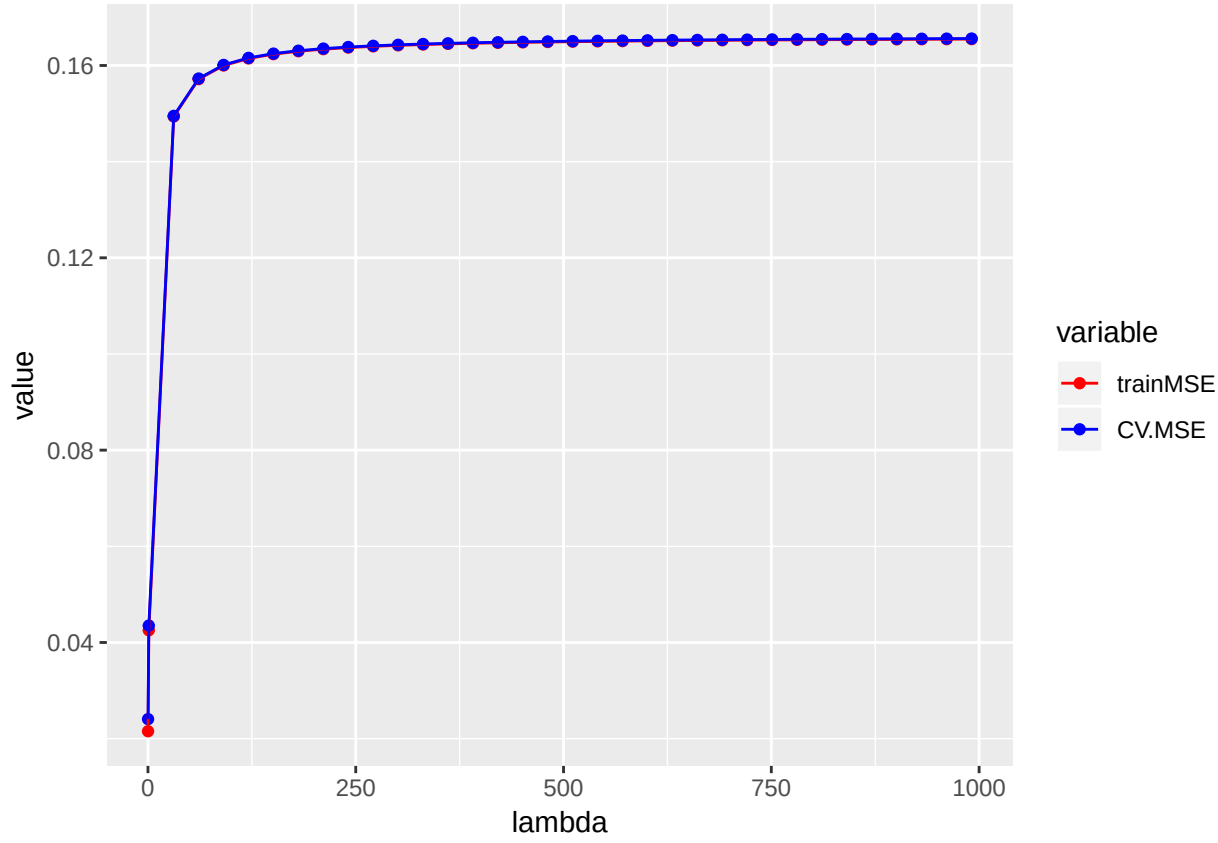
the 10th model gives the lowest CV-MSE

#6.4.1

```
train = AmesTiny[setdiff(names(AmesTiny), c("SalePrice"))]

x_train <- model.matrix( ~ .-1, train)
train.label = log(AmesTiny$SalePrice + 1)
mod.ridge = glmnet(x_train, log(AmesTiny$SalePrice + 1), alpha = 0, lambda = 1)
```

#6.4.2

```
lambdas = c(0.1, seq(1,1000,30))

training.error = rep(0,length(lambdas))
test.error = rep(0,length(lambdas))

for (i in 1:length(lambdas)) {
  fit.ridge= glmnet(x_train, train.label, alpha=0, lambda = lambdas[i])
  training.error[i] = mean((train.label - predict(fit.ridge, x_train))^2)
}

cvmod = cv.glmnet(x_train, train.label, alpha=0, lambda = lambdas, type.measure = 'mse',nfolds = 5)

dat <- data.frame(lambda = lambdas, trainMSE = training.error, CV.MSE = rev(cvmod$cvm))
dat.m <- melt(dat, id.vars = "lambda")
ggplot(dat.m, aes(lambda, value, colour = variable)) +
  geom_point() + geom_line(aes(lambda, value, colour = variable))+
  scale_colour_manual(values = c("red", "blue"))
```



when $\lambda = 0.1$, we have both minimum training MSE and CV-MSE