521 HW 3

William Tirone

The Honor Code

❗ Important

(a) Please state the names of people who you worked with for this homework. You can also provide your comments about the homework here.
Eli Gnesin, Natalie Smith, Tommy Misikoff, Alonso Guererro
(b) Please type/write the following sentences yourself and sign at the end. We want to make it extra clear that nobody cheats even unintentionally.
I hereby state that all of my solutions were entirely in my words and were written by me. I have not looked at another student's solutions and I have fairly credited all external sources in this write up.

Q1

1.1

TRUE. It's easier to shrink the small ones, this is also seen on p.37 of lecture 9.

1.2

FALSE. We don't necessarily know what will happen to test error.

1.3

FALSE. We can specify a lack of knowledge with a non-informative prior.

1.4

FALSE. Bayesian intervals can be used to make probabilistic statements and confidence intervals cannot, only under repeated experiments.

1.5

FALSE. Bias will increase as lambda increases to reduce variance.

1.6

FALSE. At some values of lambda, can have negative coefficients. Also see Lecture 10 page 11.

1.7

TRUE. If there are two collinear variables, for example, LASSO may return different solutions.

1.8

FALSE. We should scale if predictors are not on the same scale and center if we don't have an intercept term.

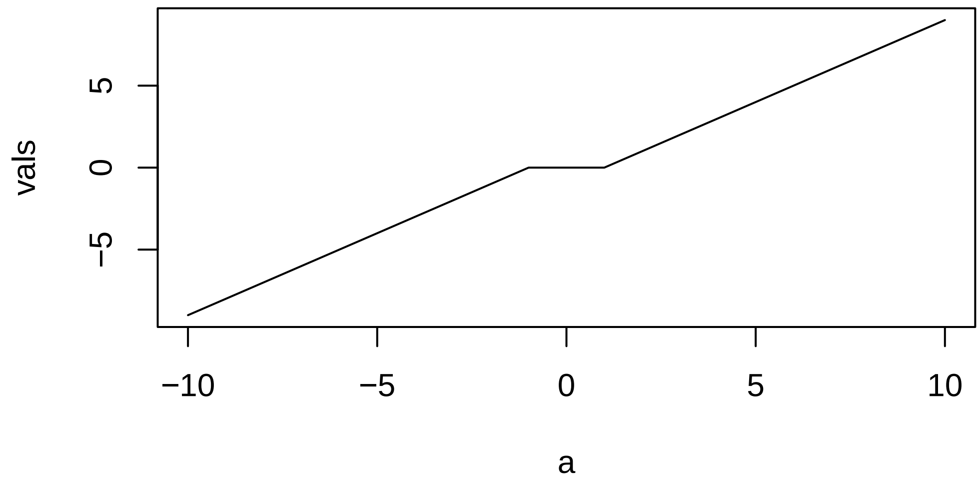
Q2

2.1

$$soft(a,1) = sign(a)(|a| - 1)_+$$

The soft-thresholding function is non decreasing on the whole domain.

```
a = -10:10
vals = sign(a) * (abs(a) - 1)

plot(a,vals,type='l')
```


vals
5
0
−5
−10
−5
0
5
10
a

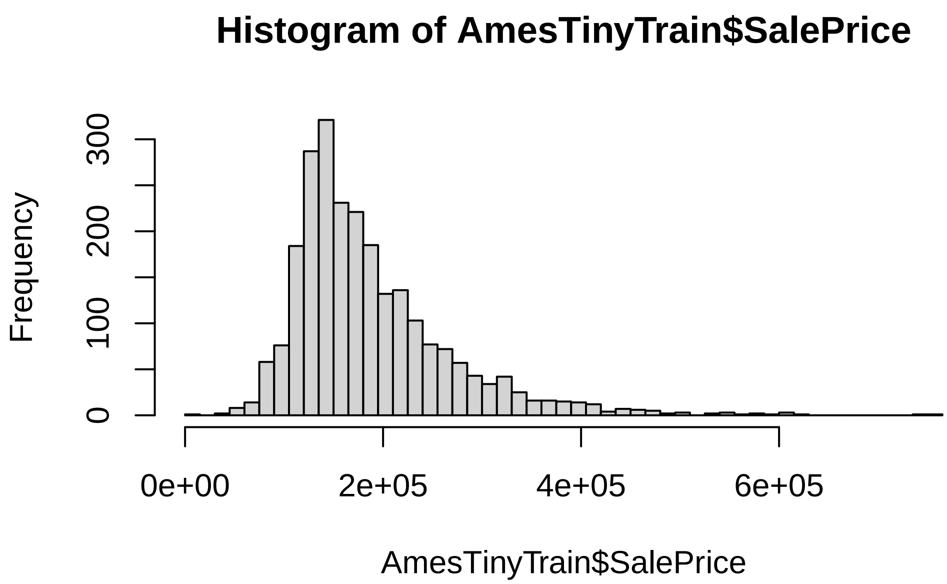
Q3

3.1

3.1.1

Since the data is skewed, we might want a log transform to make it more unimodal.

```
hist(AmesTinyTrain$SalePrice, breaks= seq(0, 770000, 15000))
```



3.1.2

Checking for NAs. Both checks return FALSE, so no NAs present.

```
any(is.na(AmesTinyTrain))
```

```
[1] FALSE
```

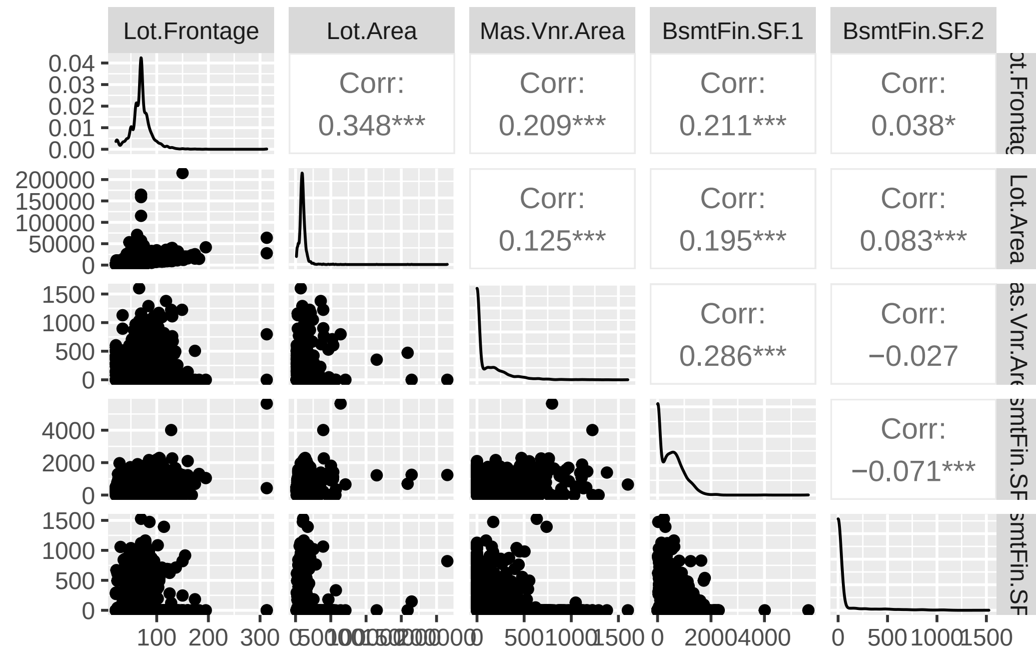
```
any(is.na(AmesTinyTest))
```

```
[1] FALSE
```

3.1.3

It does not look like there is collinearity.

```
v = continuousVar[1:5]
ggpairs(AmesTiny[,v], progress = FALSE)
```



3.2

Fitting the lm:

```
models = c()

for (i in 1:11) {
    fit <- lm(reformulate(Xnames[1:Xname_stops[i]],
                response='log(SalePrice + 1)'), data = AmesTinyTrain)

    models = c(models, list(fit))
}
```

3.2.1

Function for MSE:

```
MSE = function(y,X,B) {
  n = dim(X)[1]
  (1/n) * (norm(y- X %*% B,type = "2")^2)
}
```

3.2.2

Function for R2:

```
R2 = function(y,X,B) {
  cor(y,X %*% B)^2
}
```

3.2.3

The R implementation and my implementation match exactly, so it worked. The final model, with the most features, has the lowest training MSE.

```
# initializing variables to store values in loop
mse_train = c()
R2_train = c()
mse_RImp = c()
R2_RImp = c()
mse_test = c()
num_predictors = c()

# loop to calculate values of MSE / R2
for (i in seq(1,11)){

  y.i = log(AmesTinyTrain$SalePrice + 1)
  X.i = model.matrix(models[[i]])
  B.i = matrix(models[[i]]$coefficients)

  mse_i = MSE(y = y.i,
              X = X.i,
              B = B.i)
```

```
  r2_i = R2(y = y.i,
            X = X.i,
            B = B.i)

  num_predictors = c(num_predictors, length(models[[i]]$coefficients))

  mse_train = c(mse_train, mse_i)
  R2_train = c(R2_train, r2_i)

  mse_RImp = c(mse_RImp, mean(models[[i]]$residuals^2))
  R2_RImp = c(R2_RImp, summary(models[[i]])$r.squared)

  mse_test = c(mse_test,
               mean((log(AmesTinyTest$SalePrice +1) -
                       predict(models[[i]], AmesTinyTest))^2))
}

#my implementation of MSE and R Squared
modelQuality = data.frame(num_predictors, mse_train, R2_train)

# R implementation of MSE and R Squared
modelQualityRImp = data.frame(num_predictors, mse_RImp, R2_RImp)

# R imp of test set
modelQualityTest = data.frame(num_predictors, mse_test)

print(modelQuality)
```

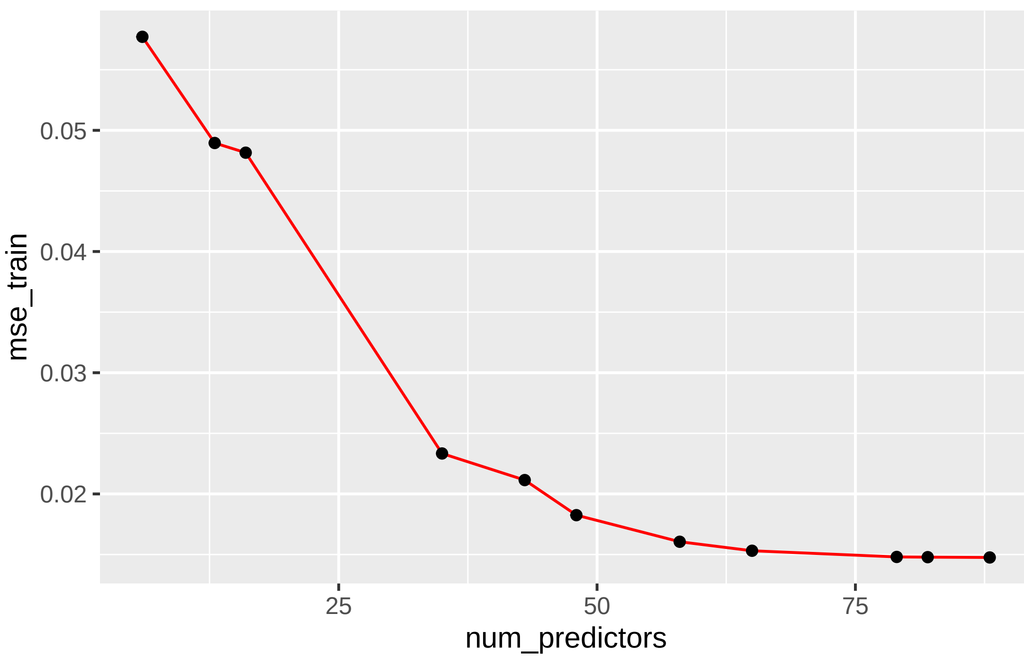
```
   num_predictors  mse_train  R2_train
1               6 0.05771607 0.6147249
2              13 0.04895374 0.6732166
3              16 0.04815538 0.6785459
4              35 0.02333795 0.8442110
5              43 0.02114386 0.8588573
6              48 0.01825005 0.8781745
7              58 0.01605491 0.8928278
8              65 0.01531166 0.8977893
9              79 0.01480165 0.9011938
10             82 0.01478316 0.9013172
11             88 0.01475757 0.9014880
```

```r
print(modelQualityRImp)
```

```
   num_predictors   mse_RImp   R2_RImp
1               6 0.05771607 0.6147249
2              13 0.04895374 0.6732166
3              16 0.04815538 0.6785459
4              35 0.02333795 0.8442110
5              43 0.02114386 0.8588573
6              48 0.01825005 0.8781745
7              58 0.01605491 0.8928278
8              65 0.01531166 0.8977893
9              79 0.01480165 0.9011938
10             82 0.01478316 0.9013172
11             88 0.01475757 0.9014880
```

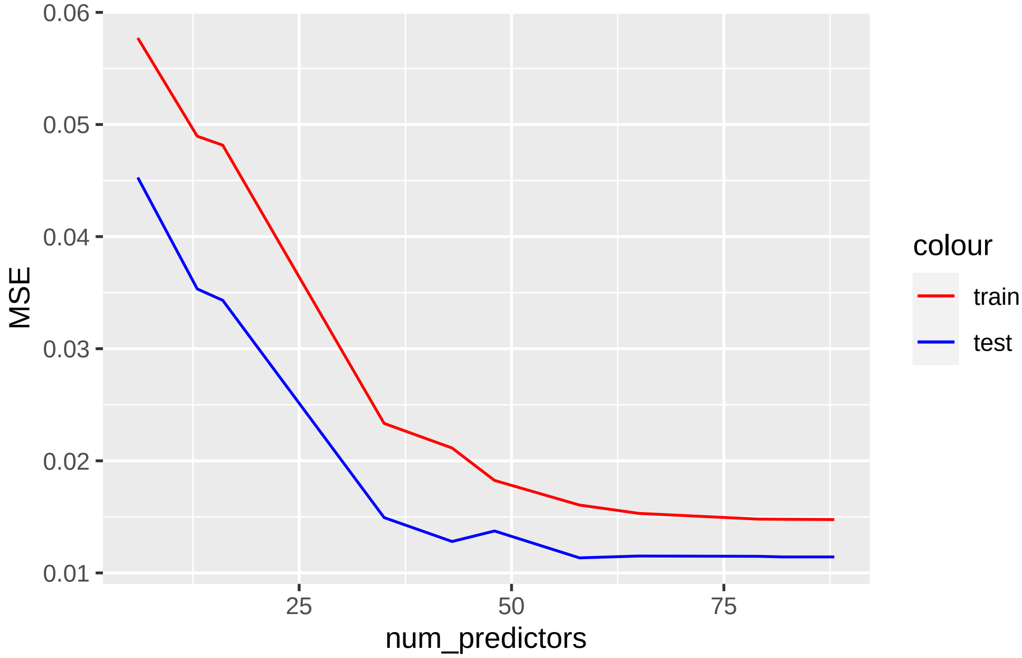
3.2.4

As model complexity is increased, train MSE declines but levels out around 60 predictors.



3.2.5

Test MSE is always decreasing, and the model with the most predictors, 88, has the lowest test MSE.




```
   num_predictors   mse_test
11             88 0.01142879
```

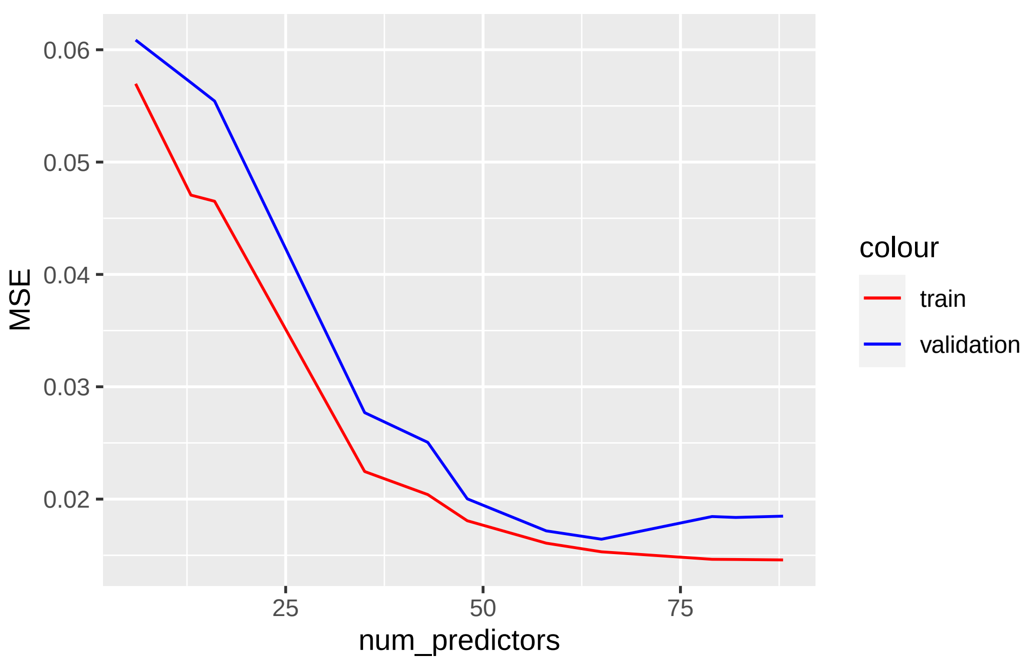
3.3

3.3.2

The lowest validation MSE happens with 65 predictors in the model.

```
train_mse = c()
validation_mse = c()

for (i in 1:11) {
    fit <- lm(reformulate(Xnames[1:Xname_stops[i]],
              response='log(SalePrice + 1)'),
              data = AmesTinyActTrain)
```

```
    actual_vals = log(AmesTinyActTrain$SalePrice + 1)
    val_vals = log(AmesTinyActVal$SalePrice + 1)

    train_mse = c(train_mse,
                  mean((actual_vals - predict(fit, AmesTinyActTrain))^2))

    validation_mse = c(validation_mse,
                       mean((val_vals - predict(fit, AmesTinyActVal))^2))
}

modelQualitySingleVal = data.frame(num_predictors, train_mse, validation_mse)

colors=c("train" = "red", "validation" = "blue")

ggplot() +
  geom_line(data=modelQualitySingleVal, aes(x=num_predictors,y=train_mse,color="train")) +
  geom_line(data=modelQualitySingleVal, aes(x=num_predictors,y=validation_mse, color="vali
  ylab("MSE") +
  scale_color_manual(values=colors)
```



3.4

3.4.1

```r
set.seed(10)
folds <- createFolds(AmesTinyTrain$SalePrice, k = 5)

cv_frame = data.frame(matrix(nrow=11,ncol=5))
colnames(cv_frame) = c("Fold1","Fold2","Fold3","Fold4","Fold5")

for (i in 1:5){

  set = AmesTinyTrain[folds[[i]],]
  y = log(set$SalePrice+1)

  mse_cv = c()

  for (j in 1:11) {
    fit <- lm(reformulate(Xnames[1:Xname_stops[j]],
                          response='log(SalePrice + 1)'),
              data = set)

    mse_cv = c(mse_cv, mean((y - predict(fit, set))^2))
  }

  cv_frame[,i] = mse_cv
}
```

3.4.2

```r
# column sums to get mse_cv
cv_frame |> mutate(mse_cv = rowSums(cv_frame)/5)
```

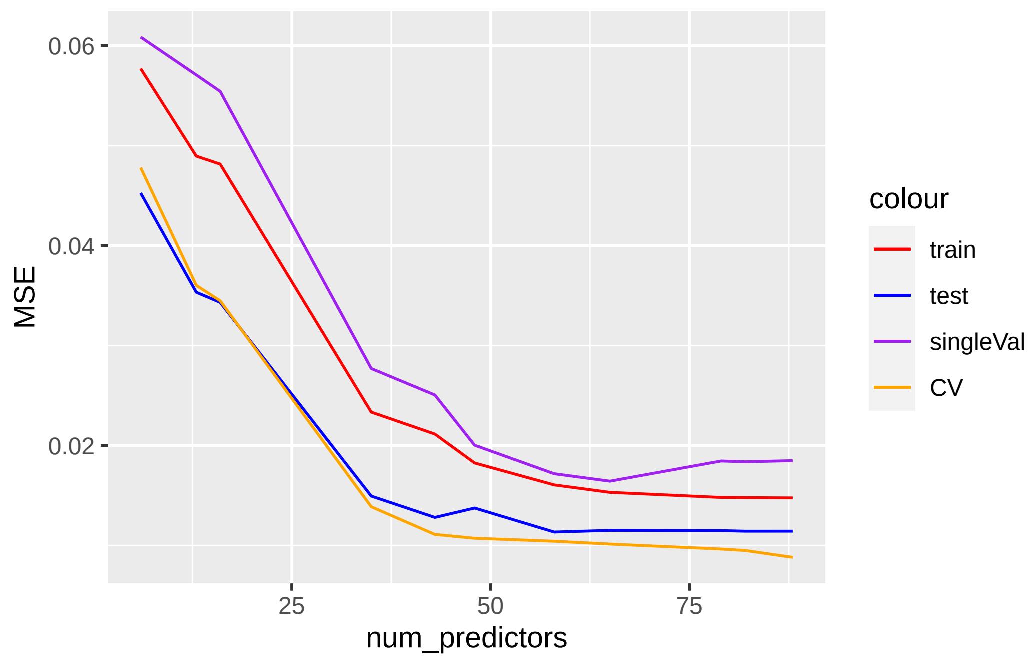
```
       Fold1      Fold2      Fold3      Fold4       Fold5     mse_cv
1 0.04291269 0.06322655 0.04541869 0.08156591 0.047811399 0.05618705
2 0.03263735 0.05495737 0.03424959 0.07214557 0.036018962 0.04600177
3 0.03085771 0.05380221 0.03402883 0.07146872 0.034478084 0.04492711
4 0.01608269 0.02572567 0.01482178 0.02902741 0.013871251 0.01990576
5 0.01399665 0.02283743 0.01324116 0.02691107 0.011102238 0.01761771
6 0.01373550 0.01834089 0.01266668 0.01816463 0.010719974 0.01472553
```

```
7  0.01315626 0.01507087 0.01217719 0.01350229 0.010424715 0.01286626
8  0.01270876 0.01441715 0.01169204 0.01301549 0.010138797 0.01239445
9  0.01259211 0.01327357 0.01065529 0.01257786 0.009642392 0.01174824
10 0.01246576 0.01318445 0.01062252 0.01250784 0.009498545 0.01165582
11 0.01240341 0.01275796 0.01031678 0.01229198 0.008808233 0.01131567
```

```r
cv_frame$num_predictors = num_predictors
```

Plotting the MSEs. Again, the model with the most features gave the lowest CV-MSE = 0.008808233.

```r
colors=c("train" = "red",
         "test" = "blue",
         "singleVal" = "purple",
         "CV" = "orange")

ggplot() +
  geom_line(data=modelQuality,
            aes(x=num_predictors,y=mse_train,color="train")) +
  geom_line(data=modelQualityTest,
            aes(x=num_predictors,y=mse_test, color="test")) +
  geom_line(data=modelQualitySingleVal,
            aes(x=num_predictors,y=validation_mse, color="singleVal")) +
  geom_line(data=cv_frame,
            aes(x=num_predictors,y=mse_cv, color="CV")) +
  ylab("MSE") +
  scale_color_manual(values=colors)
```



3.5

3.5.1

```
# referenced code here: https://www.statology.org/ridge-regression-in-r/
# and referenced Lecture 8 p. 26 for dummy variable creation

# making dummy columns for ridge
AmesTiny = dummy_cols(AmesTiny,
                      select_columns = c("Overall.Qual","Exter.Qual",
                                         "Bsmt.Qual", "Kitchen.Qual",
                                         "Garage.Qual", "Heating.QC",
                                         "Foundation"),
                      remove_selected_columns = TRUE,
                      remove_first_dummy = TRUE)

# fitting the ridge model
ridge_model <- glmnet(x=subset(AmesTiny, select=-c(SalePrice)),
              y=log(AmesTiny$SalePrice+1),
              family="gaussian",
              standardize = TRUE,
              alpha=0,
```

```
                lambda = 1)
```

3.5.2

The model that achieves the smallest MSE in this case, for both CV-MSE and trainMSE, has the smallest lambda value = 0.1.

```
# setting up vars to use later
lambdas = seq(0.1,1000,length.out = 12)

# Code to create CV-MSE for Ridge
set.seed(12)
folds <- createFolds(AmesTiny$SalePrice, k = 5)
ridge_cv_frame = data.frame(matrix(nrow=12,ncol=5))
colnames(ridge_cv_frame) = c("Fold1","Fold2","Fold3","Fold4","Fold5")

# resuing code from previous CV
for (i in 1:5){

  set = AmesTiny[folds[[i]],]
  y_set = log(set$SalePrice+1)

  ridge_mse_cv = c()

  for (j in lambdas) {
    fit <- glmnet(x=subset(set, select=-c(SalePrice)),
                  y=y_set,
                  family="gaussian",
                  standardize = TRUE,
                  alpha=0,
                  lambda = j)

    ridge_mse_cv = c(ridge_mse_cv,
                     mean((y_set - predict(fit,
                                           as.matrix(subset(set,select=-c(SalePrice)))))^2
  }

  ridge_cv_frame[,i] = ridge_mse_cv
}
```

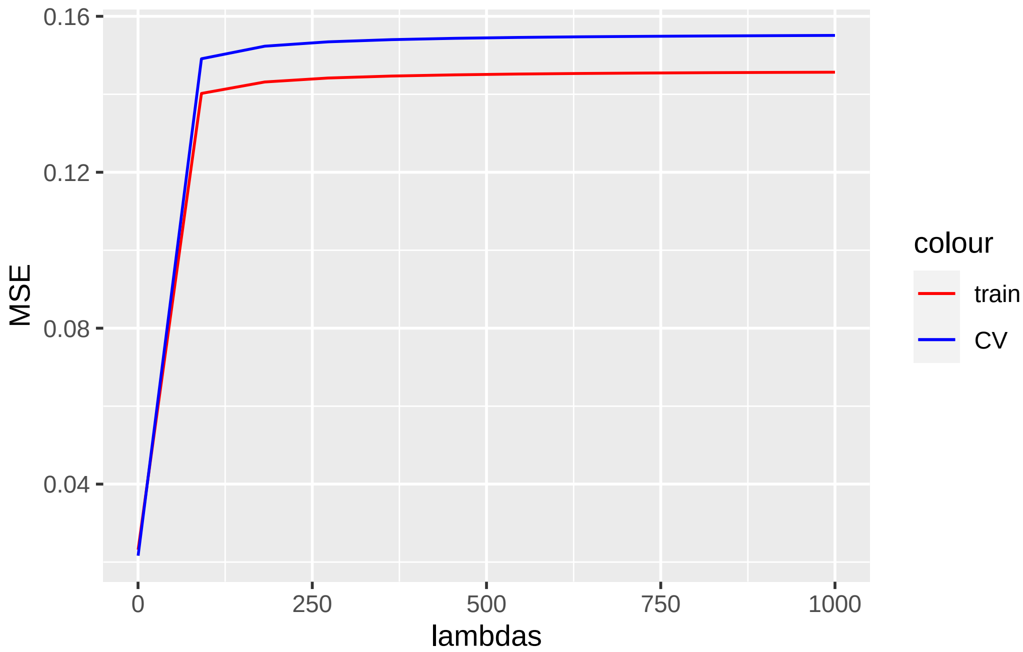
```r
# this column will be plotted as CV-MSE-Ridge
ridge_cv_frame |> mutate(ridge_mse_cv = rowSums(ridge_cv_frame)/5)
```

```
        Fold1     Fold2      Fold3      Fold4      Fold5 ridge_mse_cv
1  0.02967101 0.0140351 0.01242689 0.02314245 0.02163299   0.02018169
2  0.14721200 0.1412582 0.12388157 0.13899693 0.14910913   0.14009156
3  0.15010172 0.1445384 0.12656465 0.14184914 0.15234424   0.14307964
4  0.15109392 0.1456650 0.12748447 0.14282743 0.15345555   0.14410528
5  0.15159563 0.1462348 0.12794930 0.14332190 0.15401759   0.14462384
6  0.15189846 0.1465787 0.12822979 0.14362031 0.15435687   0.14493682
7  0.15210111 0.1468088 0.12841745 0.14381997 0.15458393   0.14514626
8  0.15224624 0.1469737 0.12855182 0.14396294 0.15474654   0.14529624
9  0.15235529 0.1470975 0.12865278 0.14407036 0.15486873   0.14540893
10 0.15244023 0.1471940 0.12873141 0.14415403 0.15496391   0.14549671
11 0.15250748 0.1472704 0.12879370 0.14422030 0.15503926   0.14556622
12 0.15256333 0.1473338 0.12884539 0.14427530 0.15510184   0.14562393
```

```r
ridge_cv_frame$lambdas = lambdas
```

```r
# fitting the models to calc. training MSE

ridge_mse = c()

for (i in lambdas) {
    fit <- glmnet(x=subset(AmesTiny, select=-c(SalePrice)),
                  y=log(AmesTiny$SalePrice+1),
                  family="gaussian",
                  standardize = TRUE,
                  alpha=0,
                  lambda = i)

    ridge_mse = c(ridge_mse, mean((log(AmesTiny$SalePrice+1) - predict(fit, as.matrix(subs
}

#putting in df to plot
ridge_mse = data.frame(lambdas, ridge_mse)

colors=c("train" = "red", "CV" = "blue")

# plotting train vs. CV-MSE
```

```r
ggplot() +
  geom_line(data=ridge_mse, aes(x=lambdas,y=ridge_mse,color="train")) +
  geom_line(data=ridge_cv_frame, aes(x=lambdas,y=ridge_mse_cv, color="CV")) +
  ylab("MSE") +
  scale_color_manual(values=colors)
```



3.6

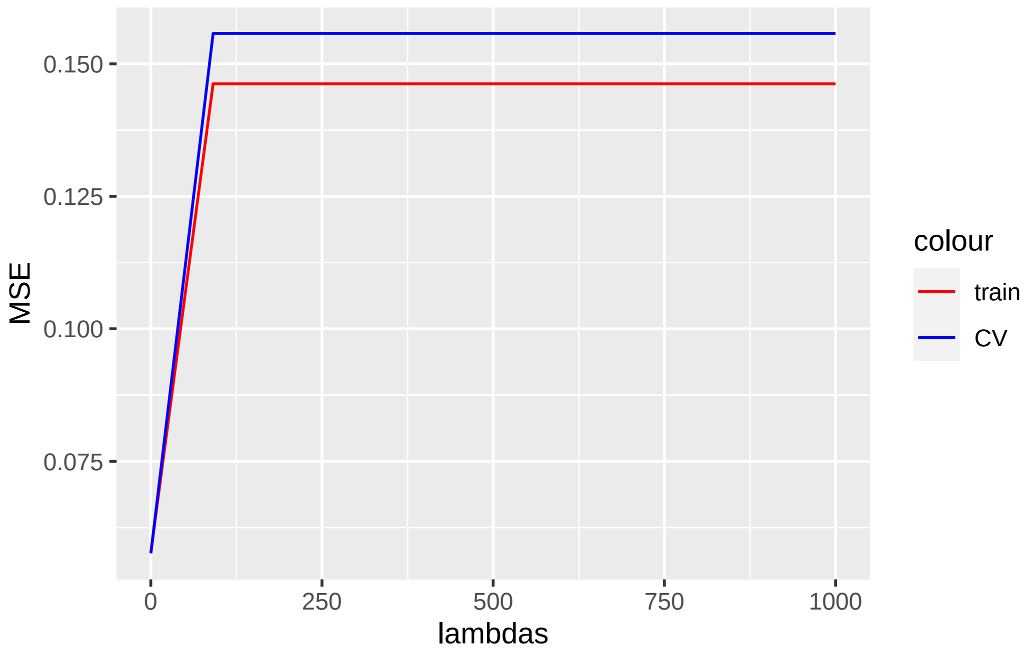
```r
# setting up vars to use later
lambdas = seq(0.1,1000,length.out = 12)

# Code to create CV-MSE for lasso
set.seed(12)
folds <- createFolds(AmesTiny$SalePrice, k = 5)
lasso_cv_frame = data.frame(matrix(nrow=12,ncol=5))
colnames(lasso_cv_frame) = c("Fold1","Fold2","Fold3","Fold4","Fold5")

# resuing code from previous CV
for (i in 1:5){

  set = AmesTiny[folds[[i]],]
```

```r
  y_set = log(set$SalePrice+1)

  lasso_mse_cv = c()

  for (j in lambdas) {
    fit <- glmnet(x=subset(set, select=-c(SalePrice)),
                  y=y_set,
                  family="gaussian",
                  standardize = TRUE,
                  alpha=1,
                  lambda = j)

    lasso_mse_cv = c(lasso_mse_cv,
                     mean((y_set - predict(fit,
                                           as.matrix(subset(set,select=-c(SalePrice)))))^2
  }

  lasso_cv_frame[,i] = lasso_mse_cv
}

# this column will be plotted as CV-MSE-lasso
lasso_cv_frame |> mutate(lasso_mse_cv = rowSums(lasso_cv_frame)/5)
```

```
        Fold1      Fold2      Fold3      Fold4      Fold5 lasso_mse_cv
1  0.07131213 0.05145498 0.04535241 0.05877826 0.05763469    0.0569065
2  0.15312361 0.14797018 0.12936385 0.14482703 0.15572974    0.1462029
3  0.15312361 0.14797018 0.12936385 0.14482703 0.15572974    0.1462029
4  0.15312361 0.14797018 0.12936385 0.14482703 0.15572974    0.1462029
5  0.15312361 0.14797018 0.12936385 0.14482703 0.15572974    0.1462029
6  0.15312361 0.14797018 0.12936385 0.14482703 0.15572974    0.1462029
7  0.15312361 0.14797018 0.12936385 0.14482703 0.15572974    0.1462029
8  0.15312361 0.14797018 0.12936385 0.14482703 0.15572974    0.1462029
9  0.15312361 0.14797018 0.12936385 0.14482703 0.15572974    0.1462029
10 0.15312361 0.14797018 0.12936385 0.14482703 0.15572974    0.1462029
11 0.15312361 0.14797018 0.12936385 0.14482703 0.15572974    0.1462029
12 0.15312361 0.14797018 0.12936385 0.14482703 0.15572974    0.1462029
```

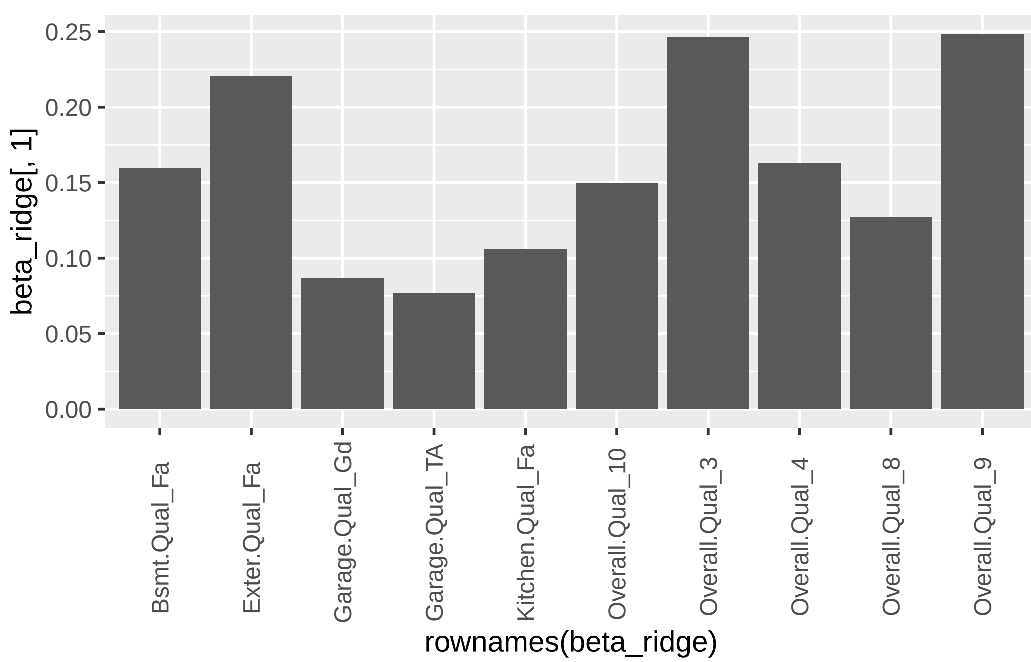
```r
lasso_cv_frame$lambdas = lambdas

lasso_mse = c()
```

```r
for (i in lambdas) {
    fit <- glmnet(x=subset(AmesTiny, select=-c(SalePrice)),
                  y=log(AmesTiny$SalePrice+1),
                  family="gaussian",
                  standardize = TRUE,
                  alpha=1,
                  lambda = i)

    lasso_mse = c(lasso_mse, mean((log(AmesTiny$SalePrice+1) - predict(fit, as.matrix(subs
}

#putting in df to plot
lasso_mse = data.frame(lambdas, lasso_mse)

colors=c("train" = "red", "CV" = "blue")

# plotting train vs. CV-MSE
ggplot() +
  geom_line(data=lasso_mse, aes(x=lambdas,y=lasso_mse,color="train")) +
  geom_line(data=lasso_cv_frame, aes(x=lambdas,y=lasso_mse_cv, color="CV")) +
  ylab("MSE") +
  scale_color_manual(values=colors)
```



3.7

```
fit <- glmnet(x=subset(AmesTiny, select=-c(SalePrice)),
              y=log(AmesTiny$SalePrice+1),
              family="gaussian",
              standardize = TRUE,
              alpha=0, # ridge = 0
              lambda = 0.1)

beta_mat_ridge = as.matrix(abs(fit$beta)) # taking absolute values

beta_mat_ridge = beta_mat_ridge[order(beta_mat_ridge[,1],decreasing=TRUE),]
beta_ridge = data.frame(beta_mat_ridge[0:10])

ggplot() +
  geom_col(data=beta_ridge, aes(x=rownames(beta_ridge), y=beta_ridge[,1])) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=.3))
```




```
fit <- glmnet(x=subset(AmesTiny, select=-c(SalePrice)),
              y=log(AmesTiny$SalePrice+1),
              family="gaussian",
```

```
                 standardize = TRUE,
                 alpha=1,
                 lambda = 0.1)

beta_mat_lasso = as.matrix(abs(fit$beta)) # taking absolute values

beta_mat_lasso = beta_mat_lasso[order(beta_mat_lasso[,1],decreasing=TRUE),]
beta_lasso = data.frame(beta_mat_lasso[0:10])

ggplot() +
  geom_col(data=beta_lasso, aes(x=rownames(beta_lasso), y=beta_lasso[,1])) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=.2))
```