homework4

Stewart Renehan

5/29/2019

1

```r
source('/Users/stewart/projects/stats/527/meatspec-train-test.R')
resubstitution_error = mean((lm(fat~., data = train)$fitted.values - train$fat)^2)
print(paste0('Average squared resubstitution error ', resubstitution_error))
```

```
## [1] "Average squared resubstitution error 0.079895532033752"
```

```r
gcv_estimate = mean((((lm(fat~., data = train)$fitted.values - train$fat) / (1 - (100 / nrow(train))))^2
print(paste0('GCV Estimated average squared resubstitution error ', gcv_estimate))
```

```
## [1] "GCV Estimated average squared resubstitution error 4.6960818273172"
```

```r
X = as.matrix(train[,1:100])
require(MASS)
```

```
## Loading required package: MASS
```

```r
H = X %*% ginv(t(X) %*% X) %*% t(X)
cv_estimate = mean((((lm(fat~., data = train)$fitted.values - train$fat) / (1 - diag(H)))^2)
print(paste0('CV Estimated average squared resubstitution error ', cv_estimate))
```

```
## [1] "CV Estimated average squared resubstitution error 0.08979431426324"
```

```r
test_error = mean((predict(lm(fat~., data = train), test[,1:100]) - test$fat) ^ 2)
print(paste0('Test Error ', test_error))
```

```
## [1] "Test Error 72.8619360736888"
```

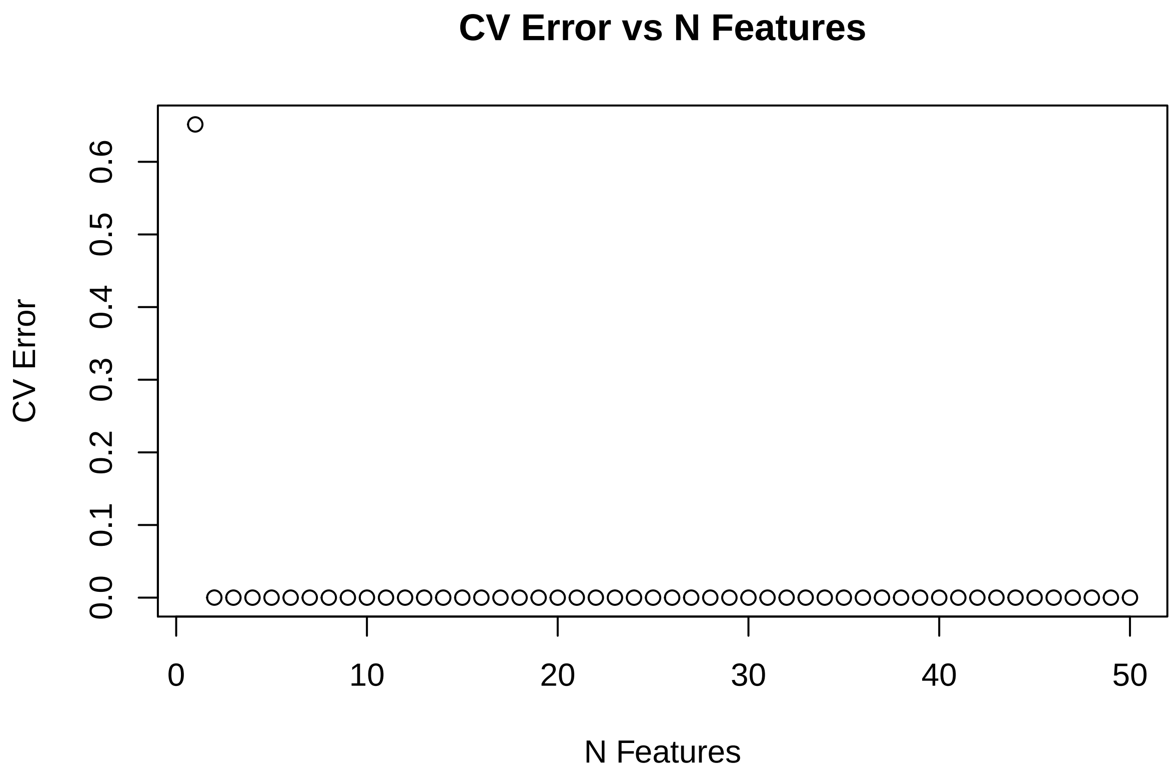
2

a)

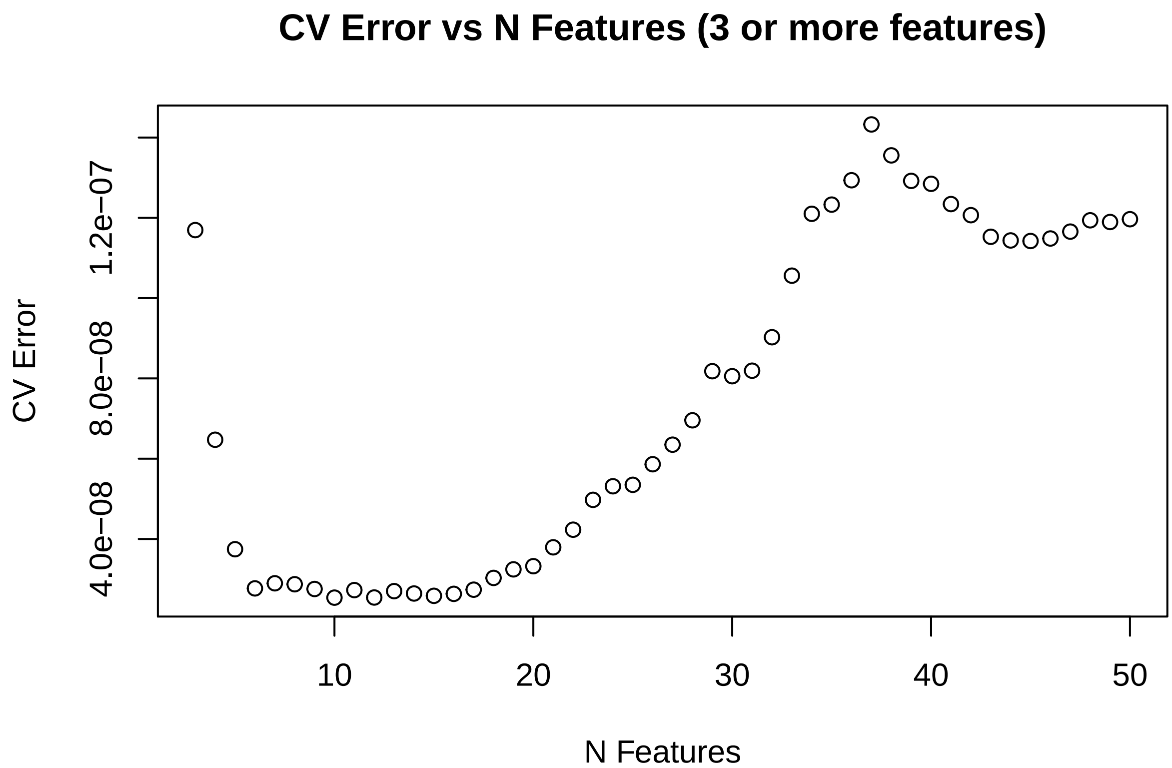
```r
require('leaps')
```

```
## Loading required package: leaps
```

```r
forward.select.features <- function(X, Y, nterm, selection_method="forward") {
  features = summary(
    regsubsets(x=X, y=Y, method=selection_method, nvmax=nterm, intercept = T)
  )$which[nterm,];
  return(names(X)[features[2:51]])
}
train_ = matrix(0, nrow=nrow(train), ncol=51)
test_ = matrix(0, nrow=nrow(test), ncol=51)
cols = ceiling(1:100 / 2 );
for (i in 1:50) {
  train_[, i] = rowMeans(train[, cols==i]);
  test_[, i] = rowMeans(test[, cols==i]);
}
train_[, 51] = train$fat
test_[, 51] = test$fat
train_ = as.data.frame(train_)
test_ = as.data.frame(test_)
names(train_)[51] = 'fat'
names(test_)[51] = 'fat'

folds <- cut(seq(1,nrow(train_)),breaks=5,labels=FALSE)

best_n_features=  0;
best_error = 'inf'
cv_errors = rep(0, 50);
for (n_features in 1:50) {
  errors = rep(0, 5)
  for (fold in unique(folds)) {
    fold_train = train_[folds != fold,];
    fold_test = train_[folds == fold,];
    features = forward.select.features(fold_train[,1:50], fold_train$fat, n_features)
    model <- lm(paste0('fat ~ ', paste(features, collapse='+')), fold_train)
    mse = mean((predict(model, fold_test) - fold_test$fat) ^2)
    errors[fold] = mse
  }
  cv_errors[n_features] = mean(errors)
  if (mean(errors) < best_error) {
    best_error = mean(errors);
    best_n_features = n_features;
  }
}

plot(1:50, cv_errors[1:50], main='CV Error vs N Features', ylab='CV Error', xlab='N Features')
```




```
plot(3:50, cv_errors[3:50], main='CV Error vs N Features (3 or more features)', ylab='CV Error', xlab='
```



b)

```
print(paste0('Estimate for optimal number of features: ', best_n_features));
```

```
## [1] "Estimate for optimal number of features: 10"
```

```
features = forward.select.features(train_[,1:50], train_$fat, best_n_features)
mse_train = mean((lm(paste0('fat ~ ', paste(features, collapse='+')), train_)$fitted.values - train_$fa
print(paste0('Resubstitution Error for optimal number of features: ', mse_train))
```

```
## [1] "Resubstitution Error for optimal number of features: 1.78959478801327e-08"
```

```
mse_test = mean((predict(lm(paste0('fat ~ ', paste(features, collapse='+')), train_), test_) - test_$fa
print(paste0('Test Error for optimal number of features: ', mse_test))
```

```
## [1] "Test Error for optimal number of features: 2.26146629130484e-08"
```

3

a)

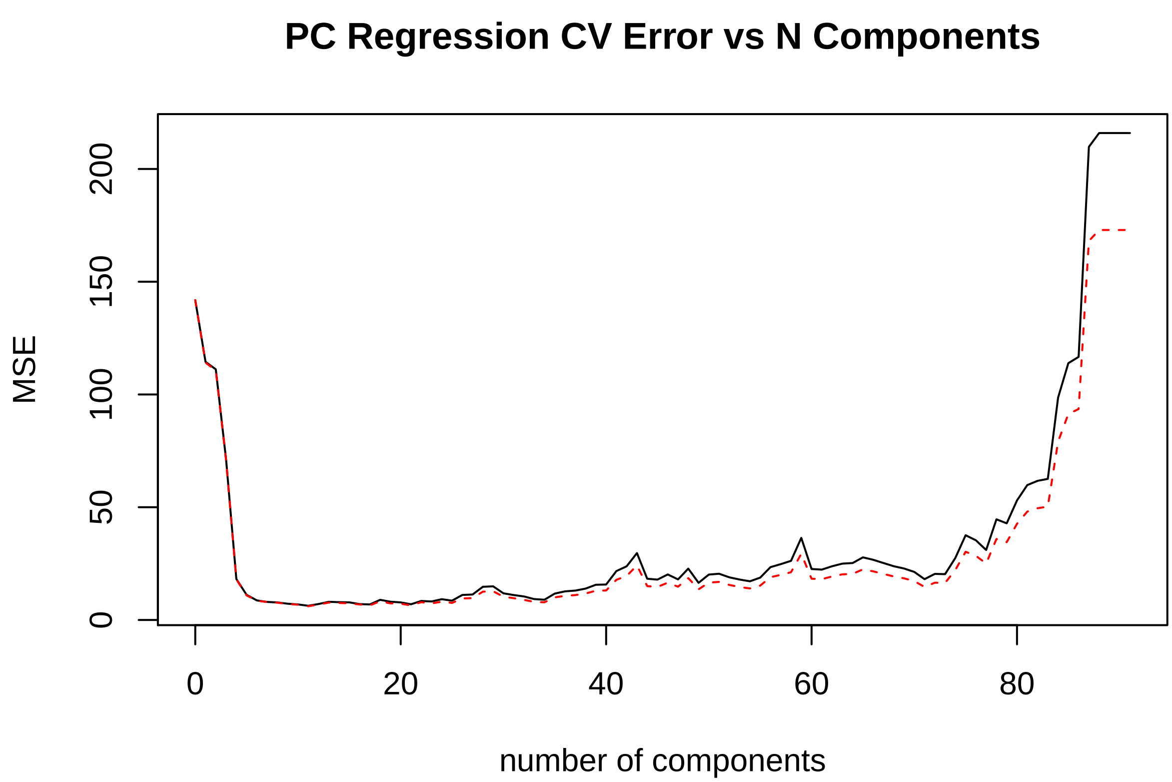
```r
# install.packages('pls')
require(pls)
```

```
## Loading required package: pls

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

##
## Attaching package: 'pls'

## The following object is masked from 'package:stats':
##
##     loadings
```

```r
pc_regression = pcr(fat~., 100, data=train, validation = 'CV', segments=5, )
```

```
## Warning in pls::mvr(fat ~ ., 100, data = train, validation = "CV", segments
## = 5, : `ncomp' reduced to 91 due to cross-validation
```

```r
validationplot(pc_regression, val.type='MSEP', ylab='MSE', main='PC Regression CV Error vs N Components
```



b)

```r
opt.n.features = which.min(pc_regression$validation$PRESS)
print(paste0('Optimal Number of components kopt: ', opt.n.features,  ' components'))
```

```
## [1] "Optimal Number of components kopt: 11 components"
```

```r
train_preds = pc_regression$fitted.values[1:nrow(train), 1, opt.n.features]
test_preds = predict(pc_regression, newdata = test)[1:100, 1, opt.n.features]
test_mse = mean((test_preds - test$fat) ^2)
train_mse = mean((train_preds - train$fat) ^2)
print(paste0("Test MSE: ", test_mse))
```

```
## [1] "Test MSE: 8.59791693302082"
```

```r
print(paste0("Train MSE: ", train_mse))
```

```
## [1] "Train MSE: 5.23447802282281"
```

4

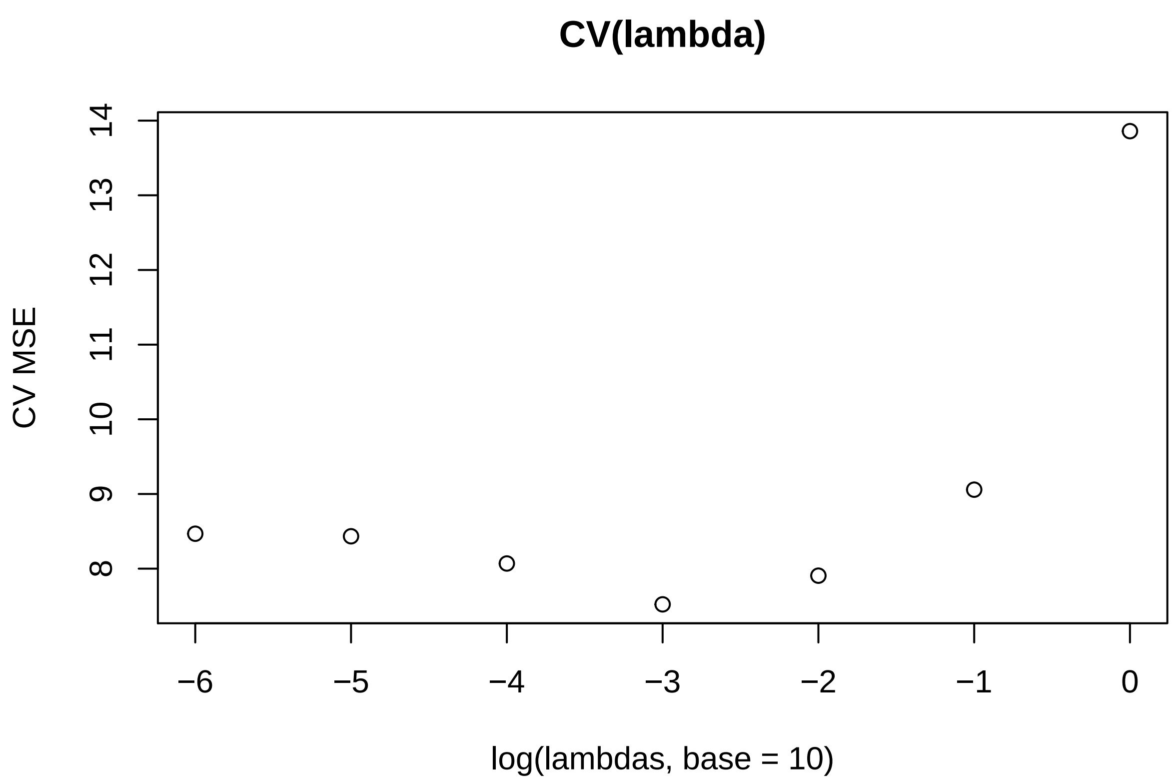
a)

```r
# install.packages('matrixStats')
library(matrixStats)
```

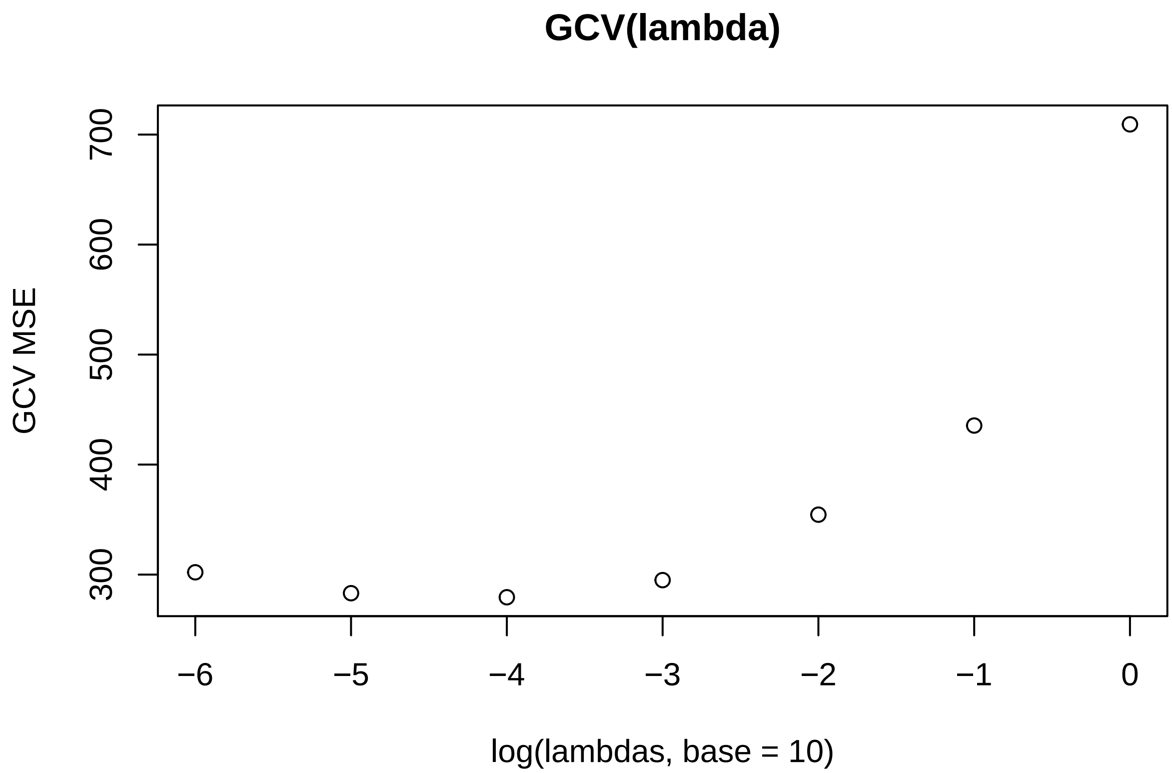
```
## Warning: package 'matrixStats' was built under R version 3.4.4
```

```r
colmeans = as.vector(colMeans(train[,1:100]));
colstdevs = colSds(as.matrix(train[,1:100]));
normalize <- function(row) {
  return ((row - colmeans) / colstdevs);
}
train.normalized = t(apply(as.matrix(train[,1:100]), 1, normalize))
train.normalized = cbind(as.vector(rep(1, nrow(train.normalized))), train.normalized);
I = diag(ncol(train.normalized))
# don't regularize the intercept
I[1,1] = 0

lambdas = c(0, 0.000001, 0.00001, 0.0001, 0.001, 0.01, .1, 1);
cv_errors = c();
best_cv_lambda = 0;
best_cv_error = 'inf'
best_gcv_lambda = 0;
best_gcv_error = 'inf'
gcv_errors = c();
X = train.normalized;
y = train$fat
for (lambda in lambdas) {
  H = X %*% ginv((t(X) %*% X) + lambda * I) %*% t(X);
  fitted.values = H %*% y;
  cv_estimate = mean(((fitted.values - train$fat) / (1 - diag(H)))^2)
```

```
  gcv_estimate = mean(((fitted.values - train$fat) / (1 - (100 / nrow(X))))^2);
  if (cv_estimate < best_cv_error) {
    best_cv_error = cv_estimate;
    best_cv_lambda = lambda;
  }
  if (gcv_estimate < best_gcv_error) {
    best_gcv_error = gcv_estimate;
    best_gcv_lambda = lambda;
  }
  cv_errors=c(cv_errors, cv_estimate);
  gcv_errors=c(gcv_errors, gcv_estimate);
}

plot(log(lambdas, base=10), cv_errors, ylab='CV MSE', main='CV(lambda)');
```




```
plot(log(lambdas, base=10), gcv_errors, ylab='GCV MSE', main='GCV(lambda)');
```




```
print(paste0('Best CV Lambda: ', best_cv_lambda));
```

```
## [1] "Best CV Lambda: 0.001"
```

```
print(paste0('Best GCV Lambda: ', best_gcv_lambda));
```

```
## [1] "Best GCV Lambda: 1e-04"
```

b)

```
gcv.coef = ginv((t(X) %*% X) + best_gcv_lambda * I) %*% t(X) %*% y;
cv.coef = ginv((t(X) %*% X) + best_cv_lambda * I) %*% t(X) %*% y;

X.test = t(apply(as.matrix(test[,1:100]), 1, normalize))
X.test = cbind(as.vector(rep(1, nrow(X.test))), X.test);
gcv.resub = mean((X%*% gcv.coef - train$fat)^2);
cv.resub = mean((X%*% cv.coef - train$fat)^2);

gcv.test.error = mean((X.test%*% gcv.coef - test$fat)^2);
cv.test.error = mean((X.test%*% cv.coef - test$fat)^2);

print(paste0('CV Optmial Lambda resubstitution error ', cv.resub));
```

```
## [1] "CV Optmial Lambda resubstitution error 5.01848356801186"
```

```r
print(paste0('GCV Optmial Lambda resubstitution error ', gcv.resub));
```

```
## [1] "GCV Optmial Lambda resubstitution error 4.75379745555223"
```

```r
gcv.cv.error = cv_errors[which.min(gcv_errors)];
cv.cv.error = min(cv_errors)
print(paste0('CV Estimate of Prediction Error for CV Optimal Lambda: ', cv.cv.error));
```

```
## [1] "CV Estimate of Prediction Error for CV Optimal Lambda: 7.52198187047969"
```

```r
print(paste0('GCV Estimate of Prediction Error for CV Optimal Lambda: ', gcv.cv.error));
```

```
## [1] "GCV Estimate of Prediction Error for CV Optimal Lambda: 8.0689709496705"
```

```r
print(paste0('CV Optmial Lambda test error ', cv.test.error));
```

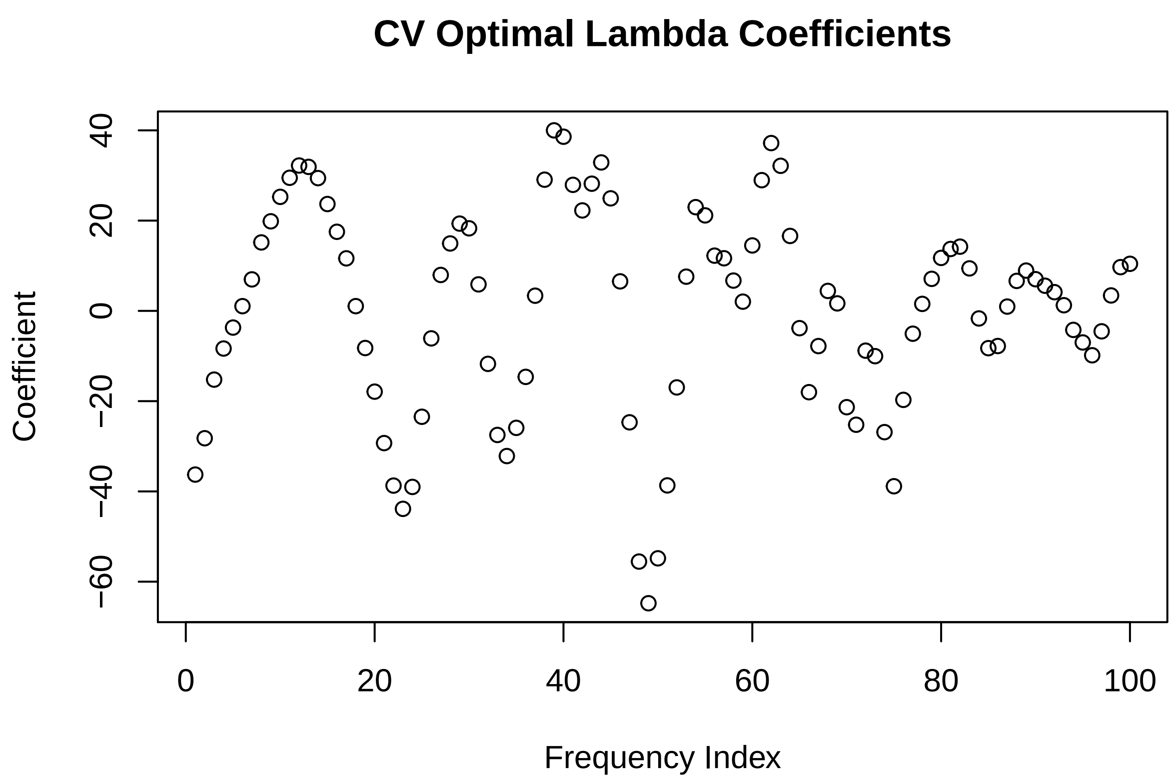
```
## [1] "CV Optmial Lambda test error 8.35427929290468"
```

```r
print(paste0('GCV Optmial Lambda test error ', gcv.test.error));
```

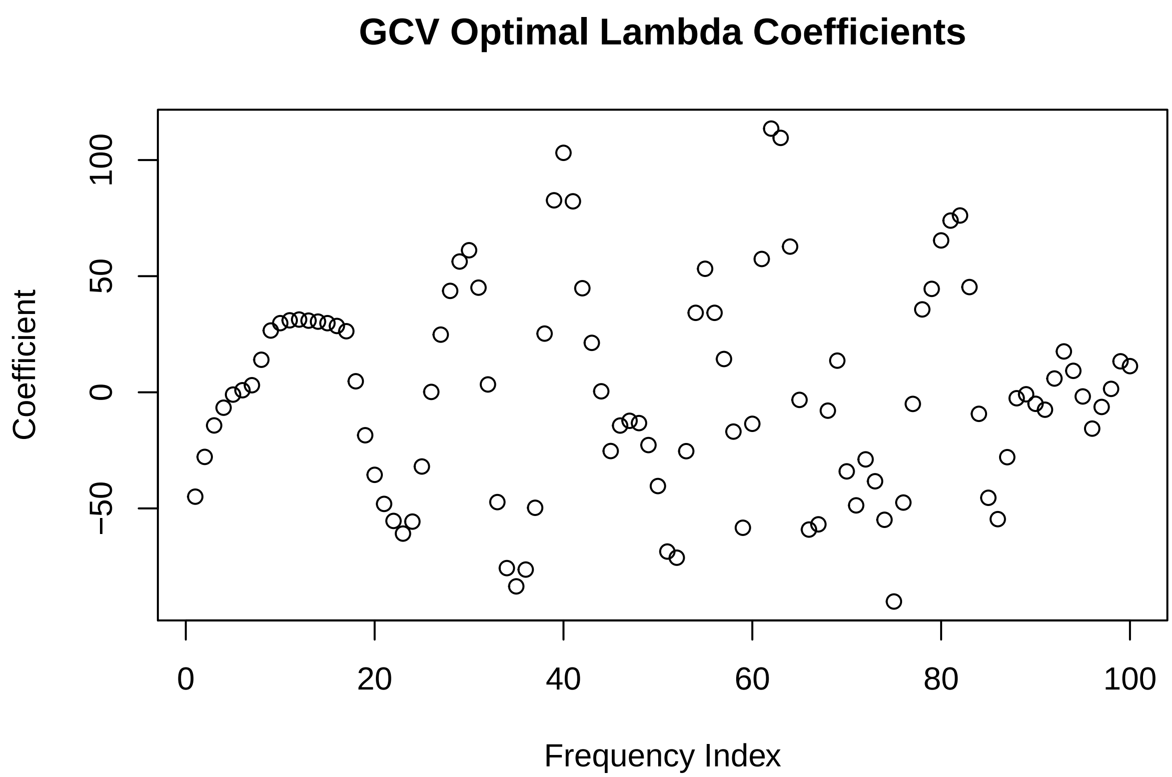
```
## [1] "GCV Optmial Lambda test error 8.08651973727821"
```

c)

```r
plot(1:100, cv.coef[2:101], xlab='Frequency Index', ylab='Coefficient', main='CV Optimal Lambda Coeffic
```



```
plot(1:100, gcv.coef[2:101], xlab='Frequency Index', ylab='Coefficient', main='GCV Optimal Lambda Coeffi
```

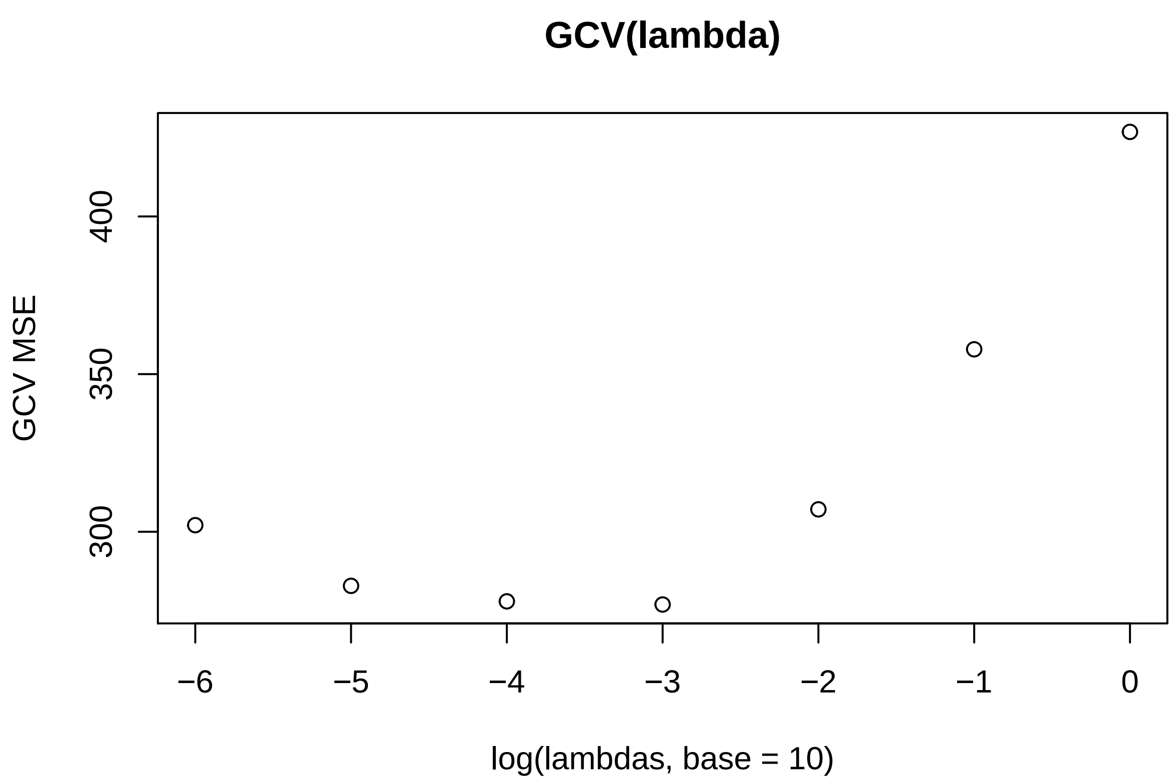


5

a)

```
I_ = diag(ncol(train.normalized));
I_[1, 1] = 0;
I_[nrow(I_), nrow(I_)] = 0;
for (idx in 2:(nrow(I_) - 1)) {
  I_[idx+1, idx] = -1;
}
lambdas = c(0, 0.000001, 0.00001, 0.0001, 0.001, 0.01, .1, 1);
best_gcv_lambda = 0;
best_gcv_error = 'inf'
gcv_errors = c();
X = train.normalized;
y = train$fat
for (lambda in lambdas) {
  H = X %*% ginv((t(X) %*% X) + lambda * (t(I_) %*% I_)) %*% t(X);
  fitted.values = H %*% y;
  gcv_estimate = mean(((fitted.values - train$fat) / (1 - (100 / nrow(X))))^2);
  if (gcv_estimate < best_gcv_error) {
    best_gcv_error = gcv_estimate;
```

```r
    best_gcv_lambda = lambda;
  }
  gcv_errors=c(gcv_errors, gcv_estimate);
}

plot(log(lambdas, base=10), gcv_errors, ylab='GCV MSE', main='GCV(lambda)');
```




```r
print(paste0('Best CV Lambda: ', best_cv_lambda));
```

```
## [1] "Best CV Lambda: 0.001"
```

b)

```r
gcv.coef = ginv((t(X) %*% X) + best_gcv_lambda * (t(I_) %*% I)) %*% t(X) %*% y;
gcv.resub = mean((X%*% gcv.coef - train$fat)^2);
gcv.test.error = mean((X.test%*% gcv.coef - test$fat)^2);
print(paste0('GCV Optmial Lambda resubstitution error ', gcv.resub));
```

```
## [1] "GCV Optmial Lambda resubstitution error 5.01347561938783"
```

```r
gcv.gcv.error = min(gcv_errors);
print(paste0('GCV Estimate of Prediction Error for GCV Optimal Lambda: ', gcv.cv.error));
```

```
## [1] "GCV Estimate of Prediction Error for GCV Optimal Lambda: 8.0689709496705"
```

```
print(paste0('GCV Optmial Lambda test error ', gcv.test.error));
```

```
## [1] "GCV Optmial Lambda test error 8.26175464785656"
```

c)

```
plot(1:100, gcv.coef[2:101], xlab='Frequency Index', ylab='Coefficient', main='GCV With Smoothness Pena
```