Homework 6

CS 498, Spring 2018, Xiaoming Ji

Problem 1

Linear regression with various regularizers The UCI Machine Learning dataset repository hosts a dataset giving features of music, and the latitude and longitude from which that music originates here. Investigate methods to predict latitude and longitude from these features, as below. There are actually two versions of this dataset. We will use the one with more independent variables. We will ignore outliers. We also regard latitude and longitude as entirely independent.

```r
library(readr)

ALL_DF = read_csv("./Geographical Original of Music/default_plus_chromatic_features_1059_tracks.txt",
                  col_names = FALSE)
COL_NUM = dim(ALL_DF)[2]
colnames(ALL_DF)[1:(COL_NUM - 2)] = sapply(1:(COL_NUM - 2), FUN =
                                             function(x) {paste("F", x, sep = "")})
colnames(ALL_DF)[COL_NUM - 1] = "Lat"
colnames(ALL_DF)[COL_NUM] = "Long"

#Change origin in order to eliminate negative value
ALL_DF$Lat = 180 + ALL_DF$Lat
ALL_DF$Long =  180 + ALL_DF$Long

Lat_DF = ALL_DF[, -COL_NUM]
Long_DF = ALL_DF[, -(COL_NUM - 1)]
```

1.1

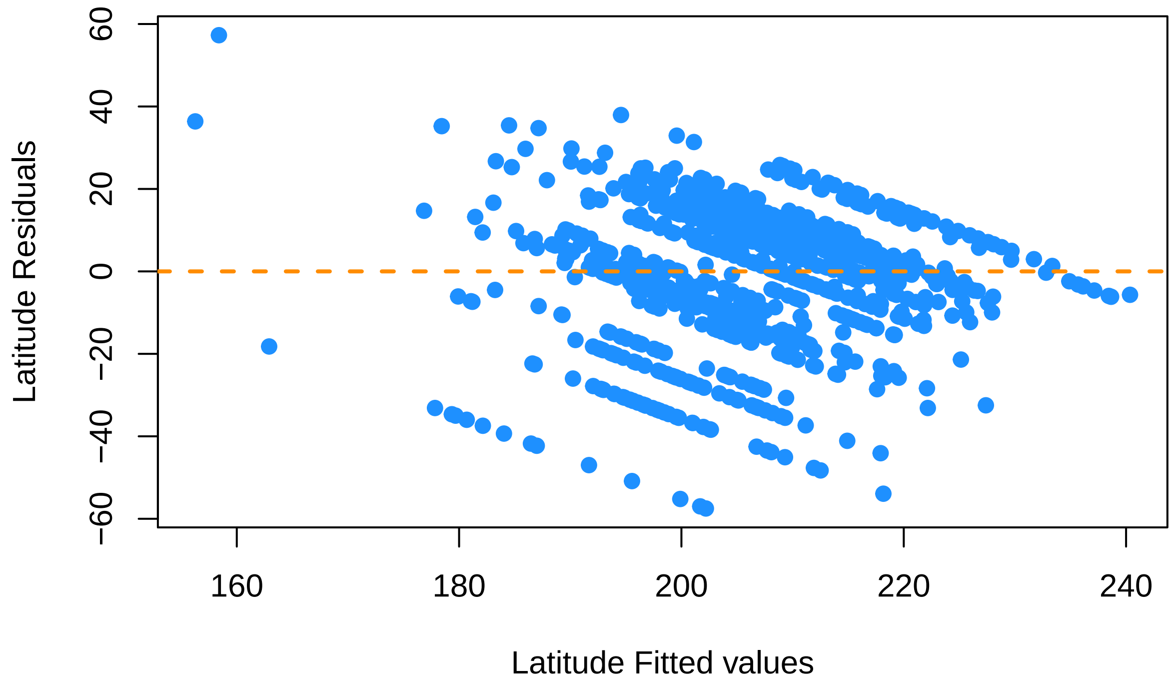
First, build a straightforward linear regression of latitude (resp. longitude) against features. What is the R-squared? Plot a graph evaluating each regression.

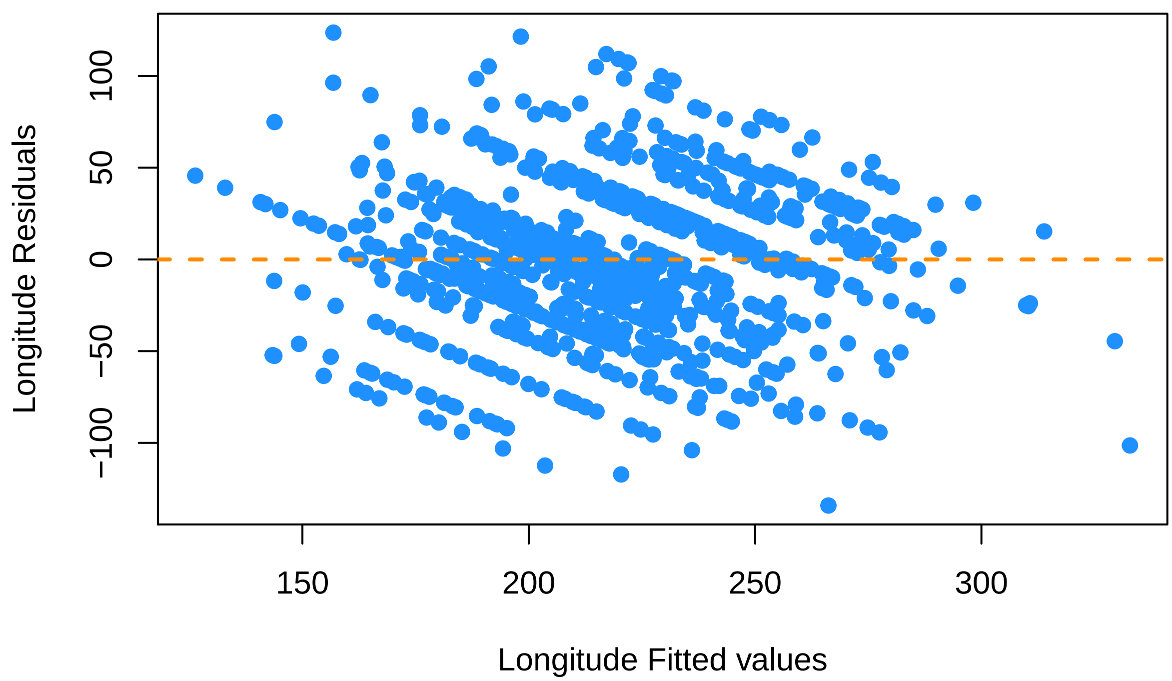
```r
lat_model = lm(Lat ~ ., data = Lat_DF)
long_model = lm(Long ~ ., data = Long_DF)
```

We get R-squared for each model as,

- Latitude: 0.2928092
- Longitude: 0.3645767

Plots of Residual vs Fitted Values as,





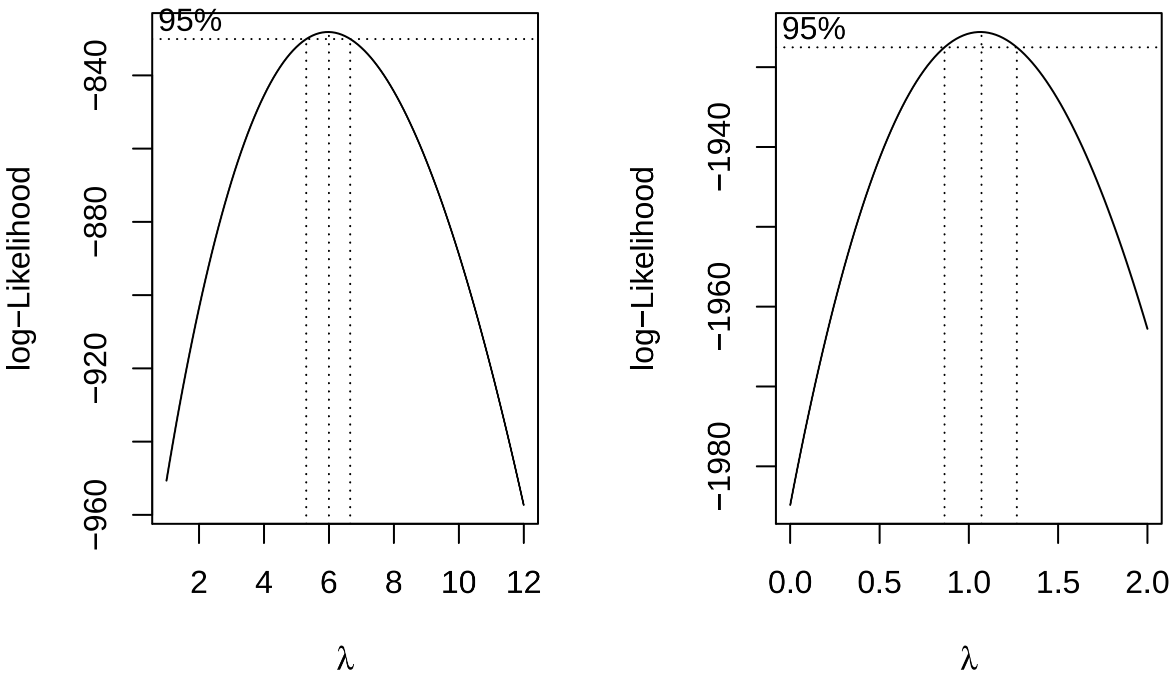
From these plots, we see the residuals are quite large and the residual of Longitude model is even larger. They also don't follow the linearity very well.

1.2

Does a Box-Cox transformation improve the regressions?

We make Box-Cox plot to determine λ.

```r
library(MASS)
par(mfrow=c(1, 2))
boxcox(lat_model, lambda = seq(1, 12, 0.1), plotit = TRUE)
boxcox(long_model, lambda = seq(0, 2, 0.1), plotit = TRUE)
```



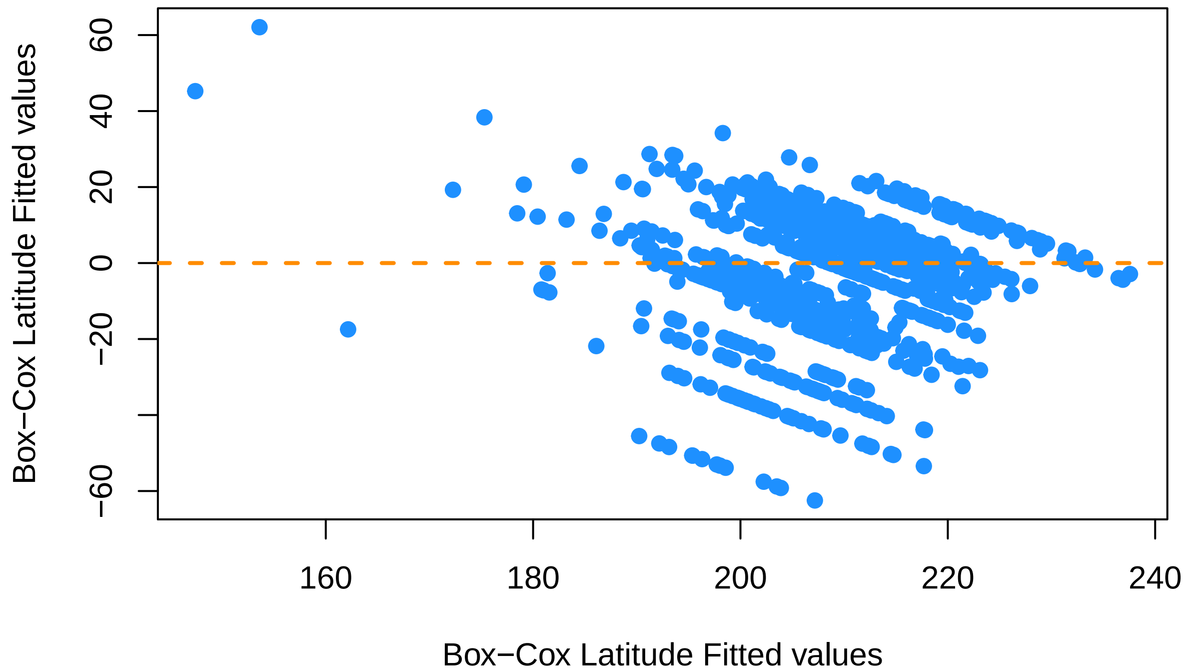
From the above plots, we see the best λ,

- Latitude: 6
- Longitude: 1, which means no need to do transformation.

We evaluate whether Box-Cox transformation for $\lambda = 6$ can give us better Latitude regression model.

We firstly make Residual vs Fitted Values plot.

```r
#We make another dataframe and do the transform on the data.
#This is to benefit the later regularization comparison
BoxCox_Lat_DF = Lat_DF
BoxCox_Lat_DF$Lat = (BoxCox_Lat_DF$Lat ^ 6 - 1) / 6
boxcox_lat_model = lm(Lat ~ ., data = BoxCox_Lat_DF)

v = (6 * boxcox_lat_model$fitted.values + 1) ^ (1/6)
r = Lat_DF$Lat - v

plot_residule(v, r, "Box-Cox Latitude Fitted values",
              "Box-Cox Latitude Fitted values")
```



This plot doesn't seem to have much different with the regular one. We then check the R Squared, R Adjusted Squared and RMSE value of both models.

```r
regular_rmse = sqrt(mean(resid(lat_model) ^ 2))
boxcox_rmse = sqrt(mean(r ^ 2))
```

Model	R.Squared	R.Adjusted.Squared	RMSE
Regular	0.2928092	0.2411685	15.516061
Box-Cox	0.3301234	0.2812074	16.0806264

Above info shows Box-Cox Latitude regression model has better R Squared (12.7% gain) and R Adjusted Squared (16.6% gain) value than the regular one. Although it has a bigger RMSE, the incremental is relatively small (3.6% bigger). Thus, we believe the Box-Cox model is better.

To conclude:

- Box-Cox transformation gives better Latitude regression model.
- No need to do any transformation for Longitude model.

1.3

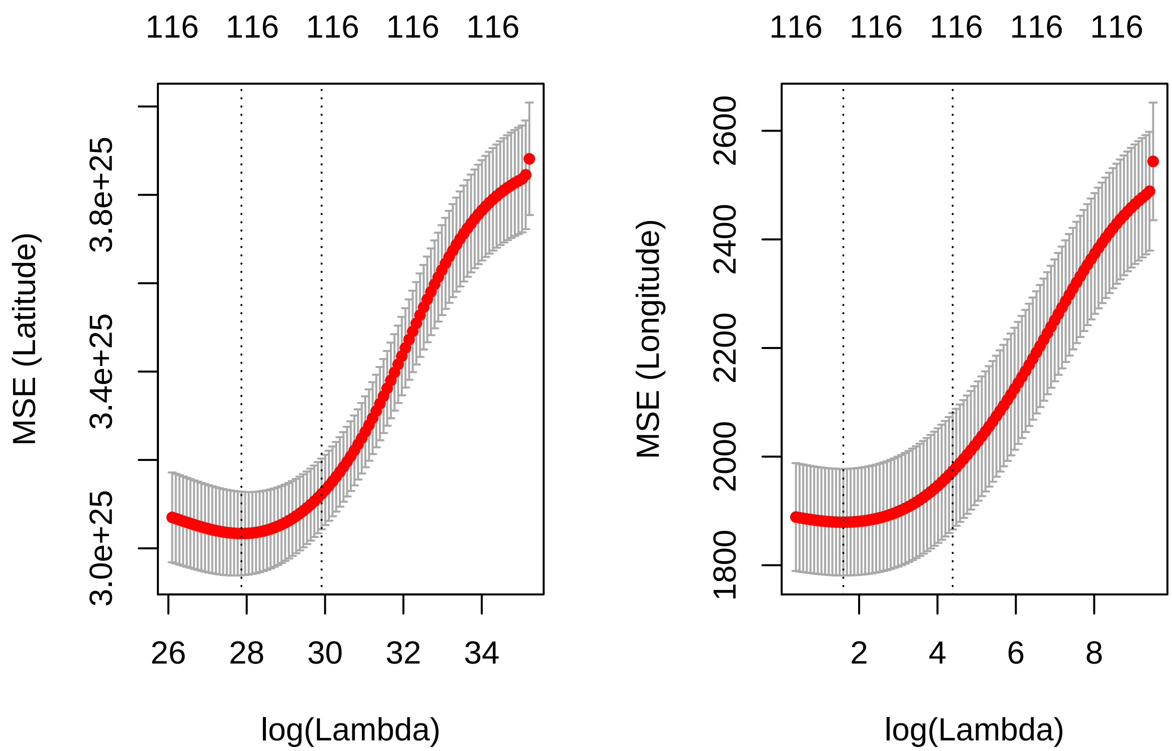
Use glmnet to produce:

1.3.1

A regression regularized by L2 (equivalently, a ridge regression). We will estimate the regularization coefficient that produces the minimum error. Is the regularized regression better than the unregularized regression?

We do cross-validation to compare the regularized and unregularized models.

```r
library(glmnet)

#Compare the regularized model with the unregularized one
compare_model = function(df, alpha, family = "gaussian") {
  col_num = dim(df)[2]
  x = as.matrix(df[, -col_num])
  y = as.matrix(df[,col_num])

  reg_model = cv.glmnet(x, y, alpha = alpha, family = family)
  reg_lambda = reg_model$lambda.min

  com_model = cv.glmnet(x, y, alpha = alpha, lambda = c(reg_lambda, 0),
                        family = family)

  return (list(reg_model = reg_model,
               cvm = com_model$cvm))
}

lat_result = compare_model(BoxCox_Lat_DF, 0)
long_result = compare_model(Long_DF, 0)
```



Model	Error (Regularized)	Error (UnRegularized)
Latitude	3.0235564×10^{25}	3.117634×10^{25}
Longitude	1862.3411713	1905.6846661

- From the comparison above, we see both Latitude and Longitude regularized L2 models have smaller cross-validation error (MSE) than the unregulized ones. Thus are better.

Lets also check the λ:

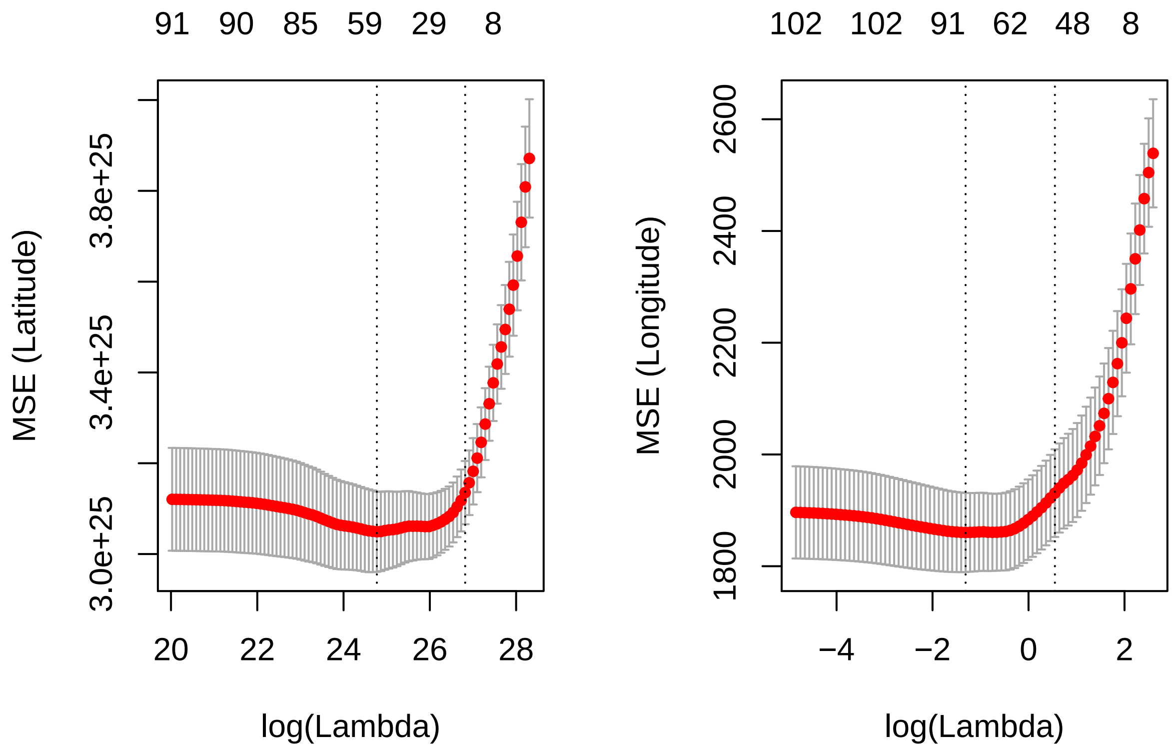
Model	lambda.min	lambda.1se
Latitude	1.2638832×10^{12}	9.7857885×10^{12}
Longitude	4.9351495	80.4306398

Note: Since we only need to draw conclusion of which model is better, we don't need to change the MSE to the original coordinates.

1.3.2

A regression regularized by L1 (equivalently, a lasso regression). We will estimate the regularization coefficient that produces the minimum error. How many variables are used by this regression? Is the regularized regression better than the unregularized regression?

```
lat_result = compare_model(BoxCox_Lat_DF, 1)
long_result = compare_model(Long_DF, 1)

lat_para_num = lat_result$reg_model$glmnet.fit$df[which.min(lat_result$reg_model$cvm)]
long_para_num = long_result$reg_model$glmnet.fit$df[which.min(long_result$reg_model$cvm)]
```



- See table below to find the number of variables used by each regression.

Model	Error (Regularized)	Error (UnRegularized)	# of Regularized Variables
Latitude	3.0633416×10^{25}	3.1424591×10^{25}	58
Longitude	1885.0105489	1917.0918323	85

- From the comparison above, we see both Latitude and Longitude regularized L1 models have smaller cross-validation error (MSE) than the unregulized ones. Thus are better.

Lets also check the λ:

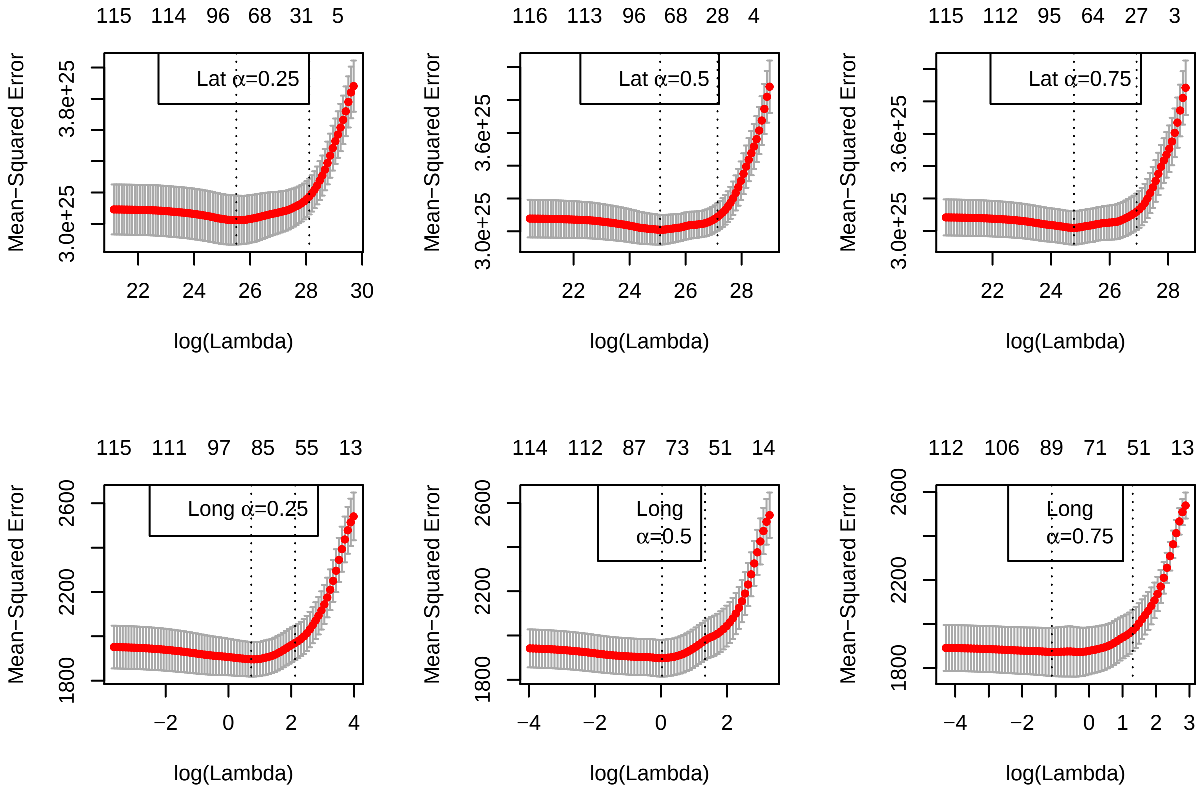
Model	lambda.min	lambda.1se
Latitude	5.7315567×10^{10}	4.4377362×10^{11}
Longitude	0.2695715	1.7328256

Note: The # of parameters for unregularized models are always the # of columns: 116.

1.3.3

A regression regularized by elastic net (equivalently, a regression regularized by a convex combination of L1 and L2). Try three values of alpha, the weight setting how big L1 and L2 are. We will estimate the regularization coefficient that produces the minimum error. How many variables are used by this regression? Is the regularized regression better than the unregularized regression?

```r
lat_result_25 = compare_model(BoxCox_Lat_DF, 0.25)
lat_para_num_25 = lat_result_25$reg_model$glmnet.fit$df[which.min(lat_result_25$reg_model$cvm)]
long_result_25 = compare_model(Long_DF, 0.25)
long_para_num_25 = long_result_25$reg_model$glmnet.fit$df[which.min(long_result_25$reg_model$cvm)]

lat_result_50 = compare_model(BoxCox_Lat_DF, 0.5)
lat_para_num_50 = lat_result_50$reg_model$glmnet.fit$df[which.min(lat_result_50$reg_model$cvm)]
long_result_50 = compare_model(Long_DF, 0.5)
long_para_num_50 = long_result_50$reg_model$glmnet.fit$df[which.min(long_result_50$reg_model$cvm)]

lat_result_75 = compare_model(BoxCox_Lat_DF, 0.75)
lat_para_num_75 = lat_result_75$reg_model$glmnet.fit$df[which.min(lat_result_75$reg_model$cvm)]
long_result_75 = compare_model(Long_DF, 0.75)
long_para_num_75 = long_result_75$reg_model$glmnet.fit$df[which.min(long_result_75$reg_model$cvm)]
```



- See table below to find the number of variables used by each regression.

Model	Error (Regularized)	Error (UnRegularized)	# of Regularized Variables
Latitude (α=0.25)	3.0103298×10^{25}	3.0850364×10^{25}	91
Latitude (α=0.5)	2.9955088×10^{25}	3.0791036×10^{25}	83
Latitude (α=0.75)	3.0406426×10^{25}	3.1019722×10^{25}	81
Longitude (α=0.25)	1860.0390264	1903.6778435	79
Longitude (α=0.5)	1896.0275724	1945.341521	77
Longitude (α=0.75)	1837.6766186	1891.0435277	89

- From the comparison above, we see all Latitude and Longitude regularized L1 models have smaller cross-validation error (MSE) than the unregulized ones. Thus are better.

Lets also check the λ:

Model	lambda.min	lambda.1se
Latitude (α=0.25)	1.1953754×10^{11}	1.6174×10^{12}
Latitude (α=0.5)	7.9010794×10^{10}	6.1175188×10^{11}
Latitude (α=0.75)	5.7809499×10^{10}	4.912382×10^{11}
Longitude (α=0.25)	2.0680561	8.3487783
Longitude (α=0.5)	1.0340281	3.8035479
Longitude (α=0.75)	0.3274981	3.6788645

Problem 2

Logistic regression. The UCI Machine Learning dataset repository hosts a dataset giving whether a Taiwanese credit card user defaults against a variety of features here. Use logistic regression to predict whether the user defaults. We will ignore outliers, but try the various regularization schemes we have discussed.

```r
set.seed(19720816)
library(readxl)
library(glmnet)
CreditDF = read_excel("default of credit card clients.xls", skip = 1)

CreditDF = CreditDF[, -1] #Remove the ID column
RecordNum = dim(CreditDF)[1]
ColumnNum = dim(CreditDF)[2]

#Make 80%-20% training-test data split
TrainIndex = sample(1:RecordNum, RecordNum * 0.8)
TestIndex = (1:RecordNum)[-TrainIndex]
TrainCreditDF = CreditDF[TrainIndex,]
TestCreditDF = CreditDF[TestIndex,]

TrainCreditX = as.matrix(TrainCreditDF[,1:(dim(TrainCreditDF)[2] - 1)])
TrainCreditY = as.matrix(TrainCreditDF[,dim(TrainCreditDF)[2]])
```

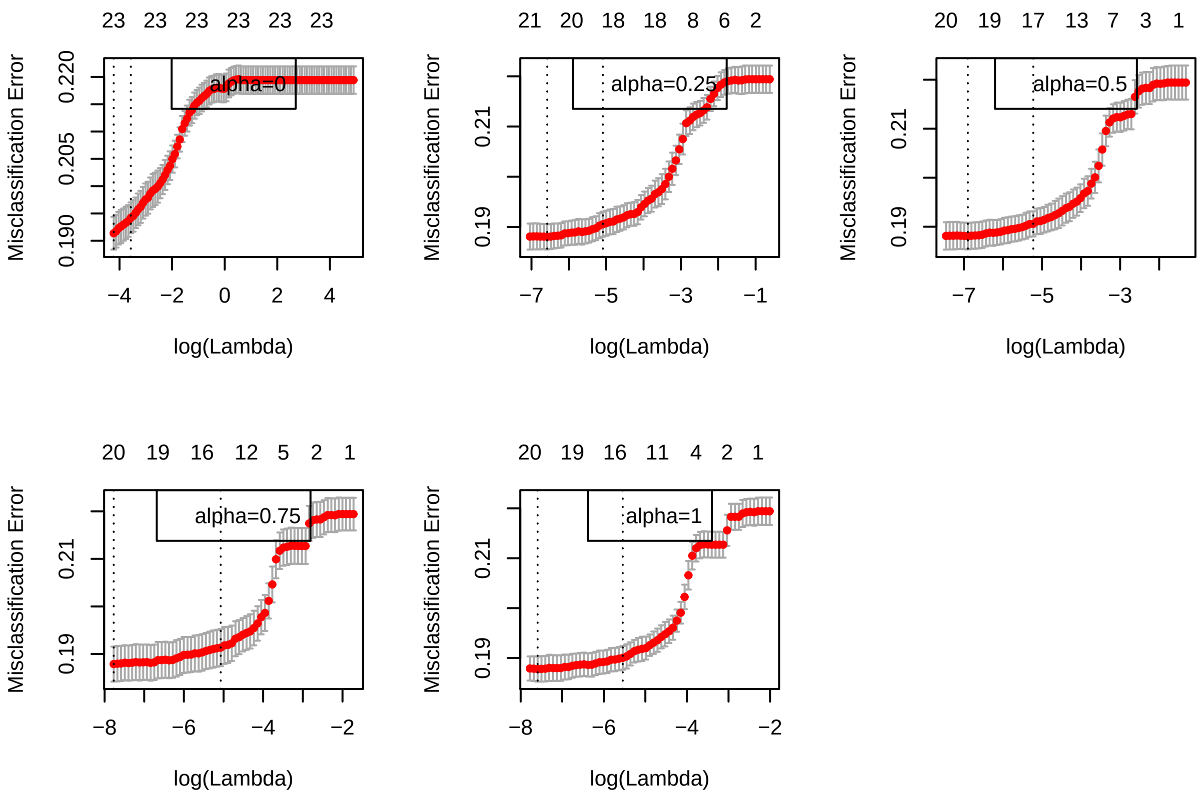
- We try different alpha and do the default 10-fold cross-validation.

```r
Alphas = c(0, 0.25, 0.5, 0.75, 1)

col_num = dim(TrainCreditDF)[2]
train_x = as.matrix(TrainCreditDF[, -col_num])
train_y = as.matrix(TrainCreditDF[,col_num])
test_x = as.matrix(TestCreditDF[,-col_num])
test_y = as.matrix(TestCreditDF[,col_num])
num_test = length(test_y)

models = list()
accuracies = rep(0, length(Alphas))
test_accuracies = rep(0, length(Alphas))

for (i in 1:length(Alphas)) {
  models[[i]] = cv.glmnet(train_x, train_y, family="binomial",
                          alpha = Alphas[i], type.measure = "class")

  accuracies[i] = 1 - min(models[[i]]$cvm)

  print(paste("Acccuracy :", format(accuracies[i],nsmall = 3),
              "; lamda.min :", format(models[[i]]$lambda.min, nsmall = 3),
              "; # of Variables :",
              models[[i]]$glmnet.fit$df[which.min(models[[i]]$cvm)]))
}
```

```
## [1] "Acccuracy : 0.808625 ; lamda.min : 0.01470606 ; # of Variables : 23"
## [1] "Acccuracy : 0.8119583 ; lamda.min : 0.001390893 ; # of Variables : 20"
## [1] "Acccuracy : 0.8119167 ; lamda.min : 0.001008974 ; # of Variables : 20"
## [1] "Acccuracy : 0.812125 ; lamda.min : 0.0004224433 ; # of Variables : 20"
```

```
## [1] "Acccuracy : 0.8122083 ; lamda.min : 0.0005044869 ; # of Variables : 19"
```

```r
par(mfrow=c(2, 3))
for (i in 1:length(Alphas)) {
  plot(models[[i]])
  legend("top", legend = paste("alpha=", Alphas[i], sep = ""))
}
```



- We choose the alpha that gives us the best accuracy.

```r
best_index = which.max(accuracies)
Alphas[best_index]
```

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

```r
accuracies[best_index]
```

```
## [1] 0.8122083
```

- We evaluate the best regularized model against the test data and calculate the accuracy.

```r
sum(predict(models[[best_index]], newx = test_x, s = "lambda.min", type = "class") == test_y) / num_tes
```

```
## [1] 0.802
```

Finally, we make a unregularized model and do cross-validation against the best regularized model.

```r
m = cv.glmnet(train_x, train_y, family="binomial",
          lambda = c(models[[best_index]]$lambda.min, 0),
          alpha = Alphas[best_index], type.measure = "class")
1 - m$cvm
```

```
## [1] 0.8117917 0.8120000
```

- Interesting enough, the unregularized model has slightly better cross-validation accuracy than the best regularized mode.