CS498AML Applied Machine Learning Homework 5

Huamin Zhang, Rongzi Wang

Mar 20, 2017

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. Either one is OK by me, but I think you'll find the one with more independent variables more interesting. You should ignore outliers (by this I mean you should ignore the whole question; do not try to deal with them). You should regard latitude and longitude as entirely independent.

1.1

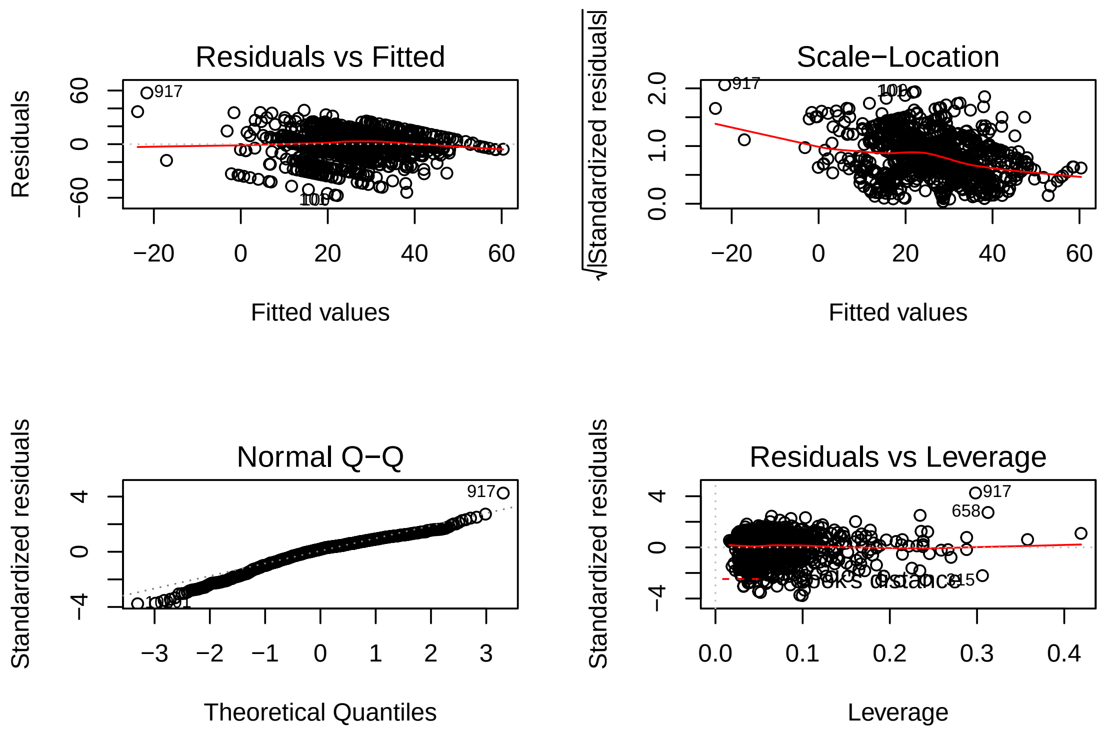
First, build a straightforward linear regression of latitude (resp. longitude) against features. What is the R-squared? Plot a graph evaluating each regression.

Answer:

```r
set.seed(1)
library(MASS)
library(glmnet)
```

```
## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:base':
##
##     crossprod, tcrossprod

## Loading required package: foreach

## Loaded glmnet 2.0-5
```

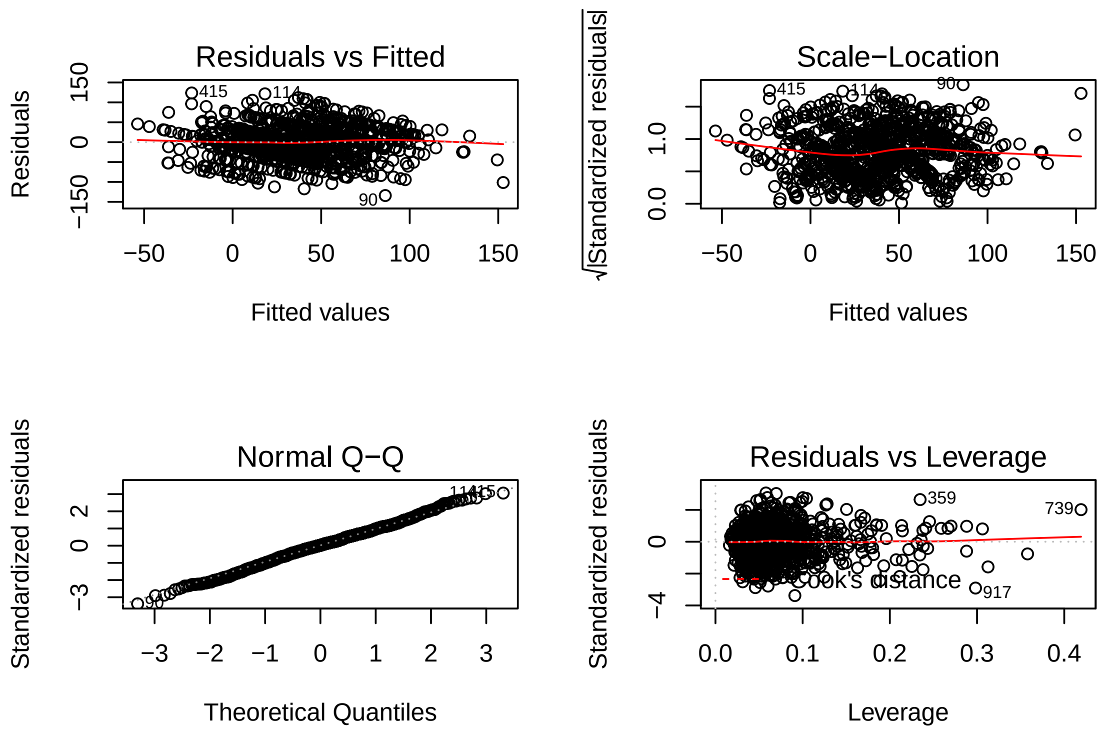
```r
setwd("C:/Users/98302/Desktop/hw5")
read.csv("default_plus_chromatic_features_1059_tracks.txt",header=F)->data
latitude = as.matrix(data[,dim(data)[2]-1])
longitude = as.matrix(data[,dim(data)[2]])
data = as.matrix(data[,-c(dim(data)[2]-1,dim(data)[2])])
latitude_lm = lm(latitude~data)
mat<-matrix(1:4,2,2)
layout(mat)
plot(latitude_lm)
```



```
latitude_r2 = summary(latitude_lm)$adj.r.squared
latitude_r2
```

```
## [1] 0.2411685
```

```
longitude_lm = lm(longitude~data)
layout(mat)
plot(longitude_lm)
```



```r
longitude_r2 = summary(longitude_lm)$adj.r.squared
longitude_r2
```

```
## [1] 0.3181766
```

So for the straightforward linear regression, the R^2 was 0.2411685 for latitude and 0.3181766 for longitude.

1.2

Does a Box-Cox transformation improve the regressions? Notice that the dependent variable has some negative values, which Box-Cox doesn't like. You can deal with this by remembering that these are angles, so you get to choose the origin. why do you say so? For the rest of the exercise, use the transformation if it does improve things, otherwise, use the raw data.

Answer:

```r
summary(latitude)
```

```
##         V1
##  Min.   :-35.30
##  1st Qu.: 14.66
##  Median : 33.66
##  Mean   : 26.65
##  3rd Qu.: 39.91
##  Max.   : 54.68
```

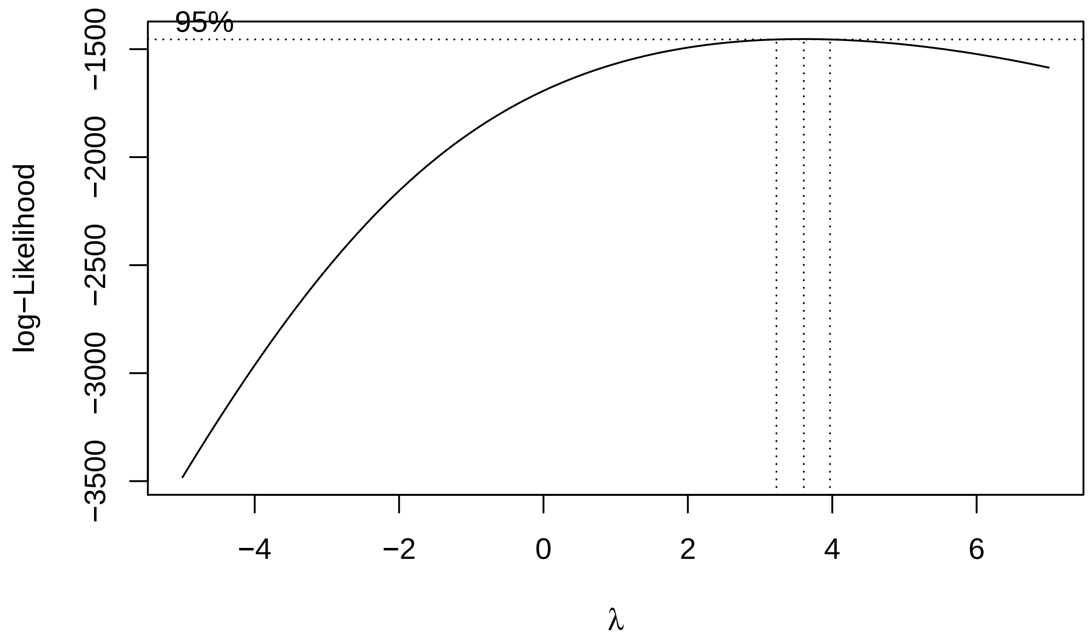
```r
summary(longitude)
```

```
##          V1
##  Min.   :-88.76
##  1st Qu.:  3.21
##  Median : 32.83
##  Mean   : 38.41
##  3rd Qu.: 74.60
##  Max.   :149.12
```

```
latitude_orginal <- latitude
longitude_orginal <- longitude
```

To do a Box-Cox transformation, the dependent variable should be positive values. Since the negative value in longitude means it is in Western Hemisphere and the negative value in latitude means it is in Southern Hemisphere. So we plus 90 to all the latitude and 180 to all the longitude.

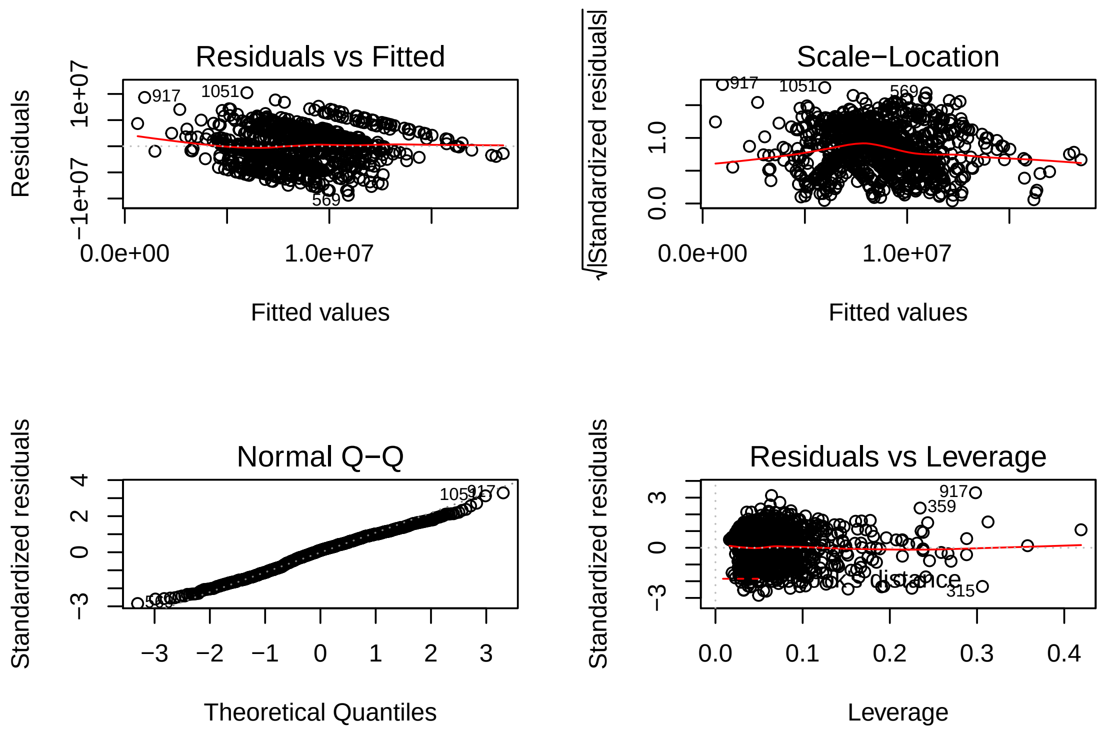
```
latitude_new <- latitude + 90
longitude_new <- longitude + 180

#boxcox
layout(1)
la_boxcox_new = boxcox(lm(latitude_new~data),lambda = seq(-5, 7, length = 100))
```



```
la_lambda_new = la_boxcox_new$x[which.max(la_boxcox_new$y)]
latitude_new_trans=(latitude_new^la_lambda_new-1)/la_lambda_new
latitude_new_trans.lm = lm(latitude_new_trans~data)
latitude_new_trans.r2 = summary(latitude_new_trans.lm)$adj.r.squared
latitude_new_trans.r2
```

```
## [1] 0.2782052
```

```r
layout(mat)
plot(latitude_new_trans.lm)
```



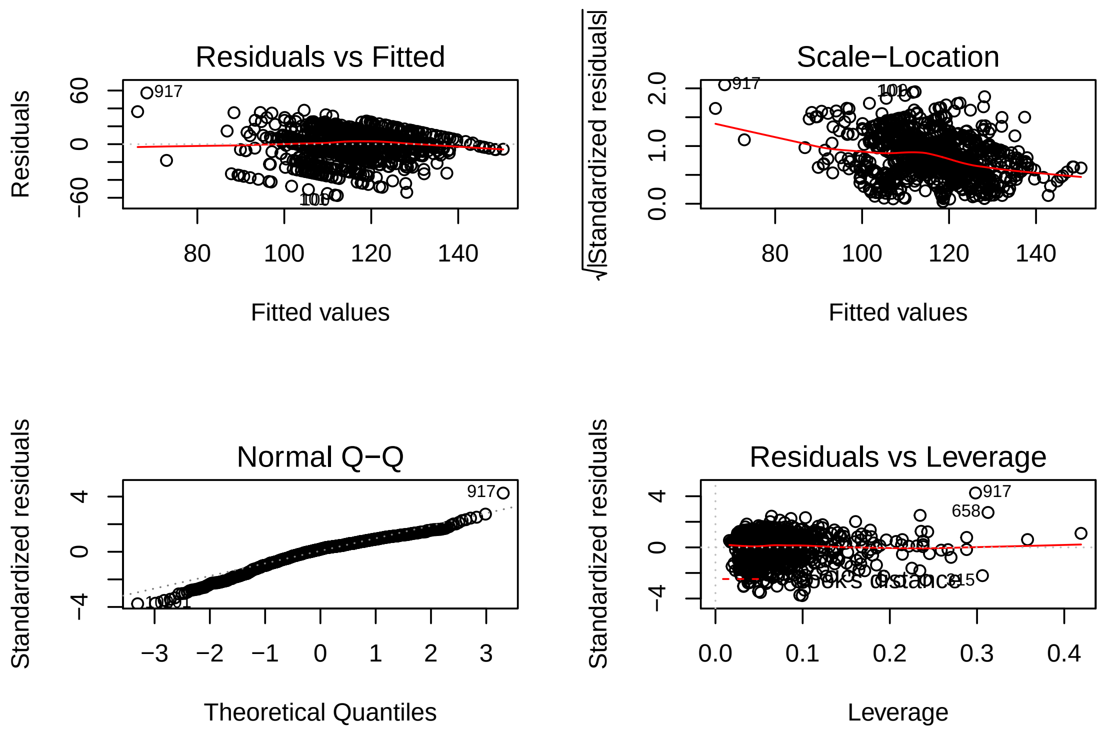
```r
latitude_new_trans.mse = mean((latitude_new_trans.lm$fitted.values - latitude_new_trans)^2)
latitude_new_trans.mse
```

```
## [1] 1.06593e+13
```

```r
latitude_new_without_trans.lm = lm(latitude_new~data)
latitude_new_without_trans.r2 = summary(latitude_new_without_trans.lm)$adj.r.squared
latitude_new_without_trans.r2
```

```
## [1] 0.2411685
```

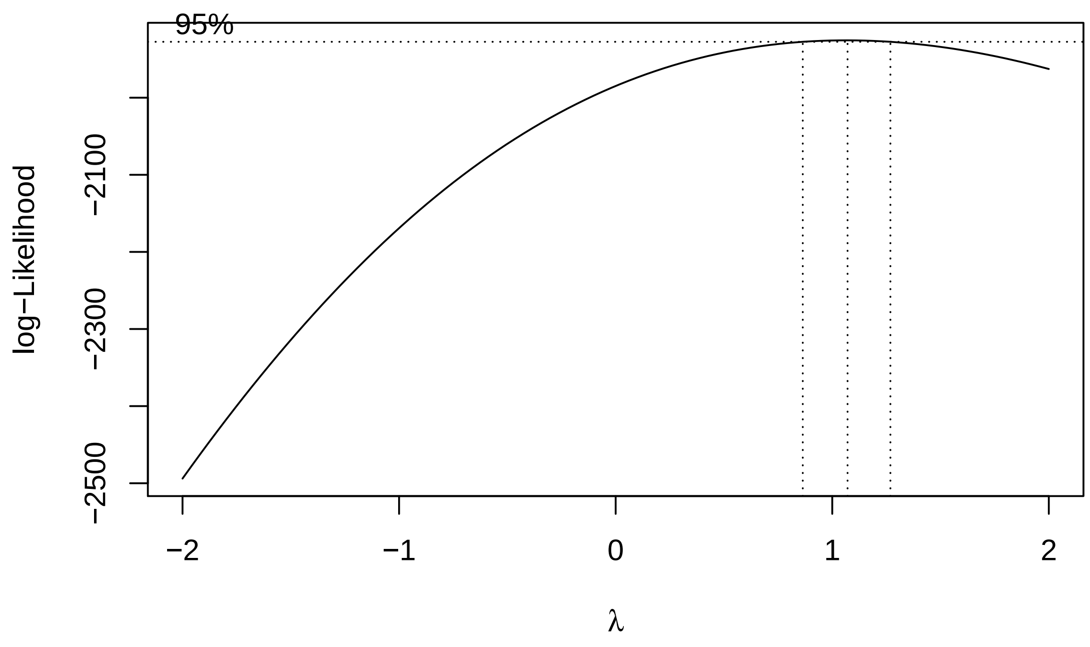
```r
plot(latitude_new_without_trans.lm)
```



```
latitude_new_without_trans.pred = (latitude_new_without_trans.lm$fitted.values^
                                    la_lambda_new-1)/la_lambda_new
latitude_new_without_trans.mse = mean((latitude_new_without_trans.pred - latitude_new_trans)^2)
latitude_new_without_trans.mse
```

```
## [1] 1.123119e+13
```

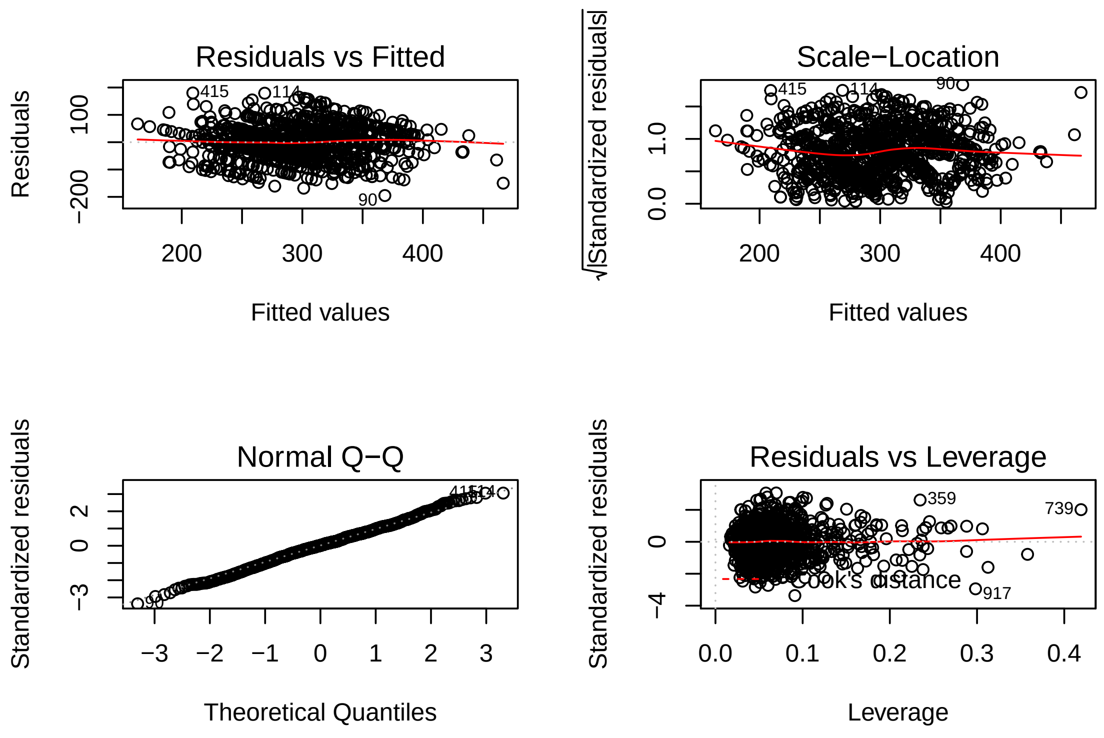
```
layout(1)
lo_boxcox_new = boxcox(lm(longitude_new~data))
```



```
lo_lambda_new = lo_boxcox_new$x[which.max(lo_boxcox_new$y)]
longitude_new_trans=(longitude_new^lo_lambda_new-1)/lo_lambda_new
longitude_new_trans.lm = lm(longitude_new_trans~data)
longitude_new_trans.r2 = summary(longitude_new_trans.lm)$adj.r.squared
longitude_new_trans.r2
```

```
## [1] 0.3187052
```

```
layout(mat)
plot(longitude_new_trans.lm)
```



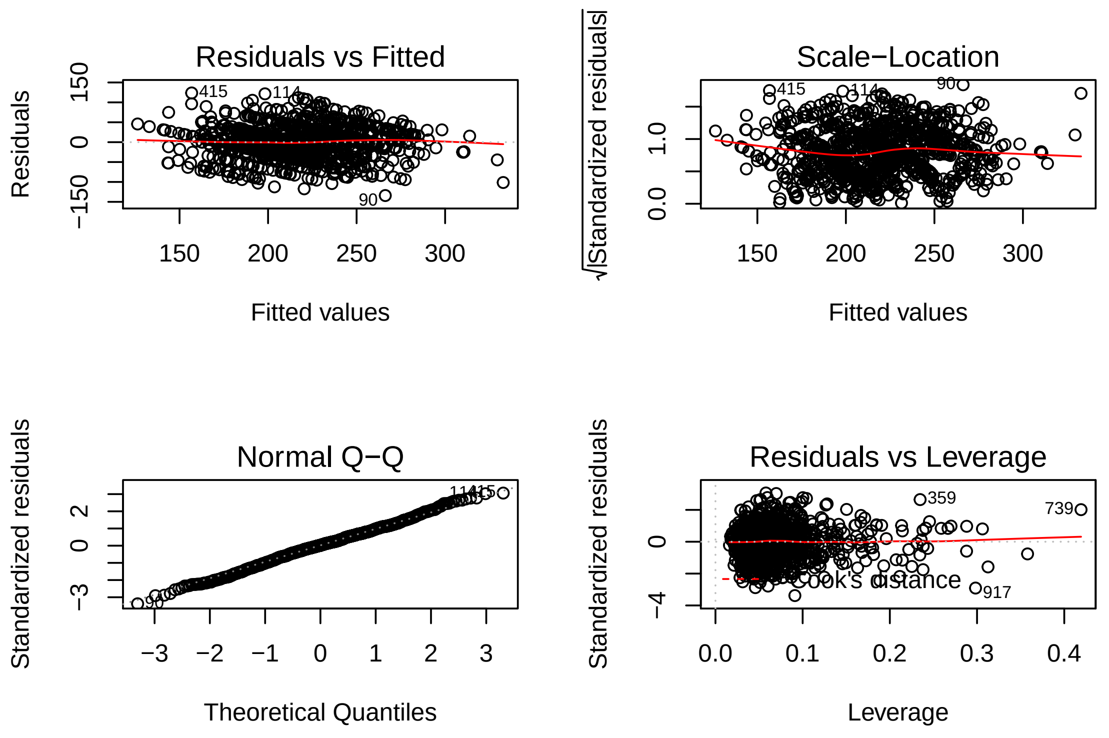
```
longitude_new_trans.mse = mean((longitude_new_trans.lm$fitted.values - longitude_new_trans)^2)
longitude_new_trans.mse
```

```
## [1] 3441.23
```

```
longitude_new_without_trans.lm = lm(longitude_new~data)
longitude_new_without_trans.r2 = summary(longitude_new_without_trans.lm)$adj.r.squared
longitude_new_without_trans.r2
```

```
## [1] 0.3181766
```

```
plot(longitude_new_without_trans.lm)
```



```
longitude_new_without_trans.pred = (longitude_new_without_trans.lm$fitted.values^
                                     lo_lambda_new-1)/lo_lambda_new
longitude_new_without_trans.mse = mean((longitude_new_without_trans.pred - longitude_new_trans)^2)
longitude_new_without_trans.mse
```

```
## [1] 3442.134
```

So the R^2 of latitude before Box-Cox transformation is 0.2411685, and the R^2 after Box-Cox transformation is 0.2782052.The mean square error of latitude before Box-Cox transformation is `1.123119e+13`, and the MSE after Box-Cox transformation is `1.06593e+13`.

Thus, for latitude prediction, using the transformation improves the result.

The R^2 of longitude before Box-Cox transformation is 0.3181766, and the R^2 after Box-Cox transformation is 0.3187052.The mean square error of longitude before Box-Cox transformation is 3442.1343336, and the MSE after Box-Cox transformation is 3441.2295086.

Thus, for longitude prediction, using the transformation improves the result.

1.3a ridge

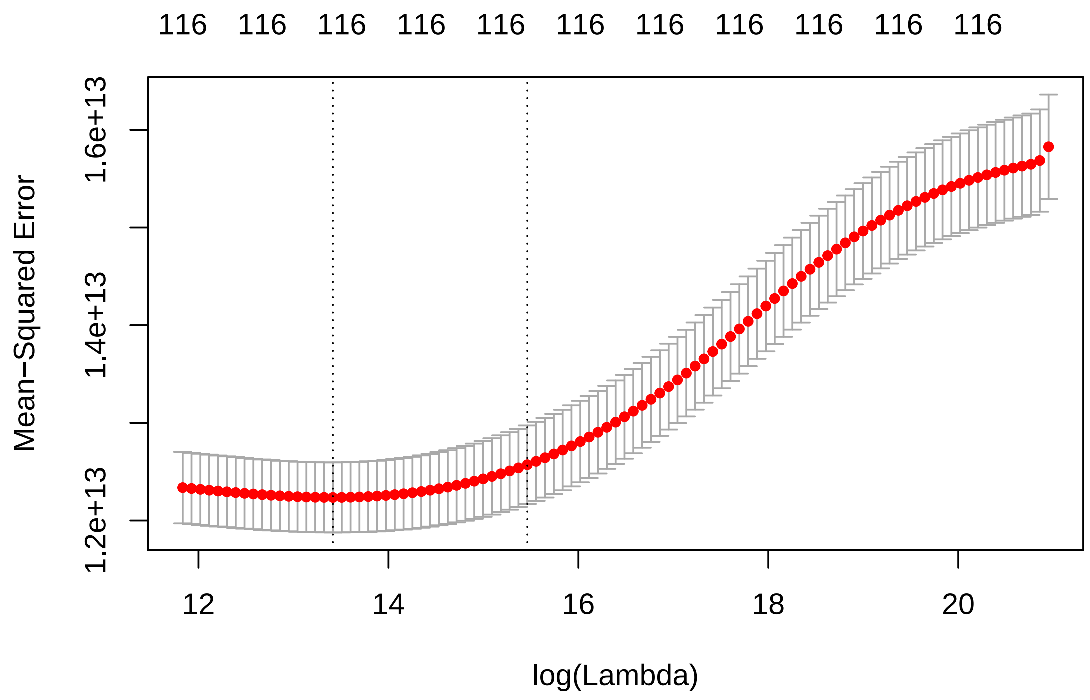
Use glmnet to produce:

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

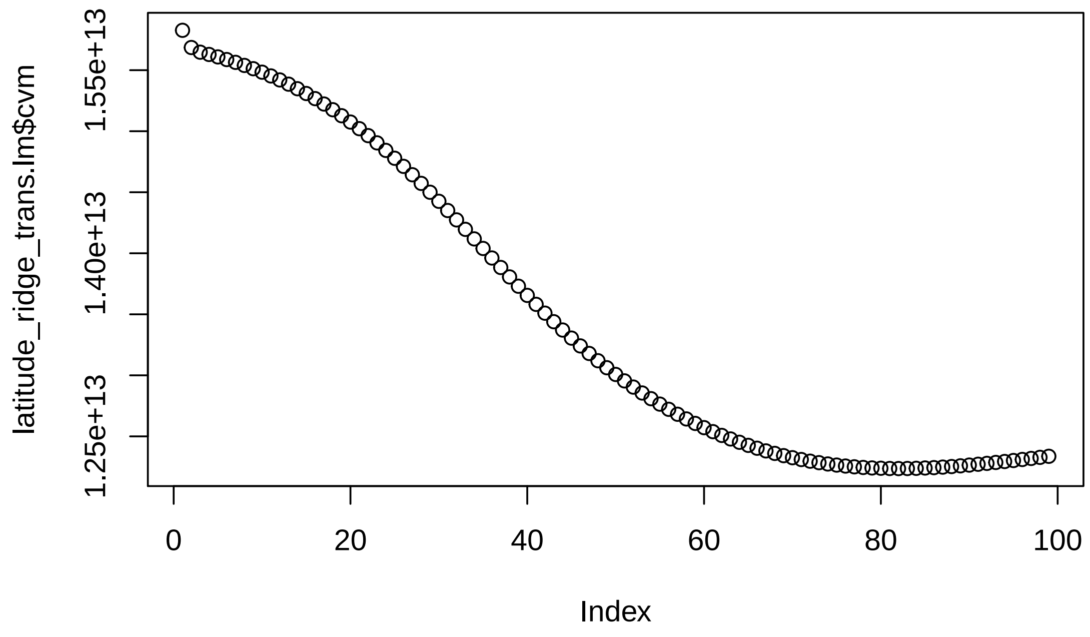
Answer:

latitude

```
layout(1)
latitude_ridge_trans.lm = cv.glmnet(x=data,y=latitude_new_trans,alpha=0,
                                    nfold = 10,family = "gaussian")
plot(latitude_ridge_trans.lm)
```



```
plot(latitude_ridge_trans.lm$cvm)
```



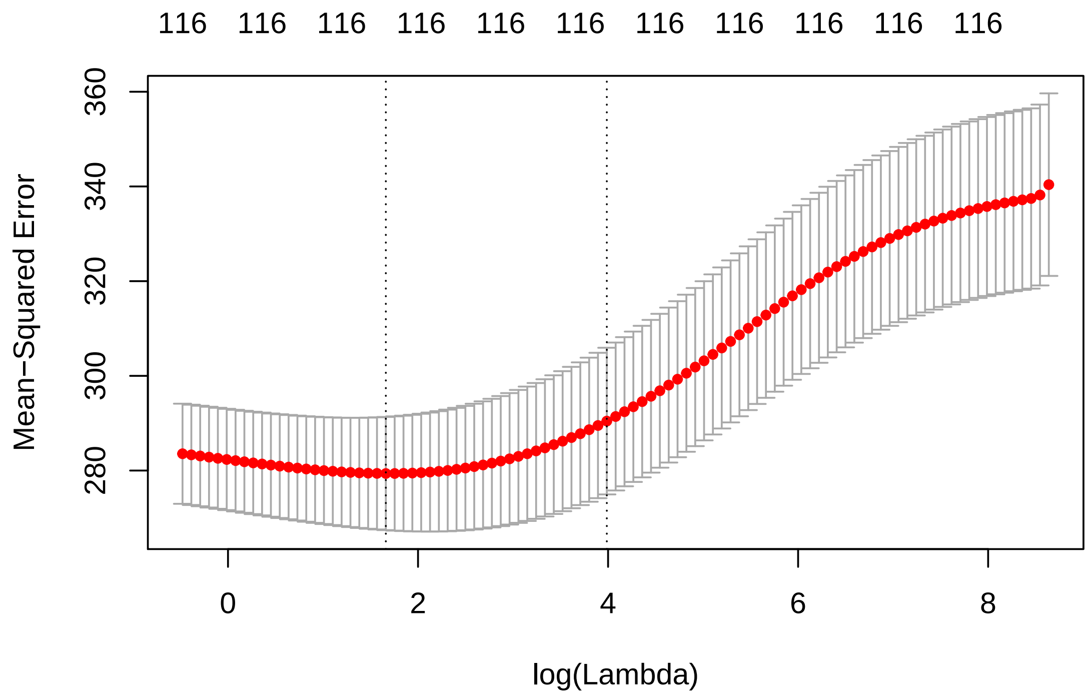
```
latitude_ridge_trans.pred <- predict(latitude_ridge_trans.lm, s =
                                    latitude_ridge_trans.lm$lambda.min, newx = data)
latitude_ridge_trans.R2 = var(latitude_ridge_trans.pred)/var(latitude_new_trans)
latitude_ridge_trans.R2
```

```
##             1
## 1 0.2409245
```

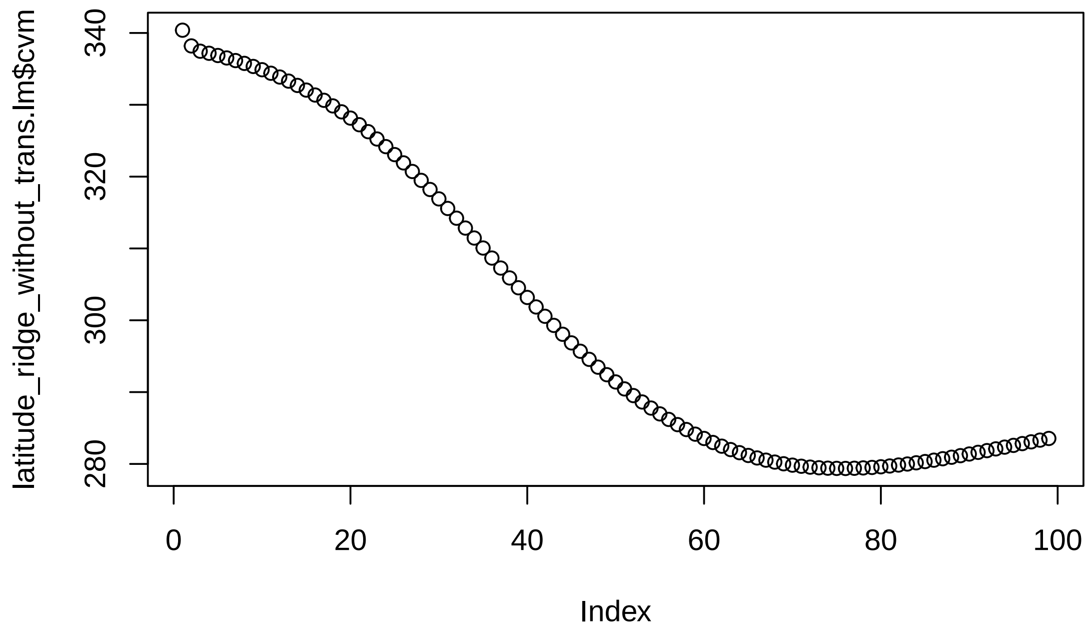
```
latitude_ridge_trans.mse = mean((latitude_ridge_trans.pred - latitude_new_trans)^2)
latitude_ridge_trans.mse
```

```
## [1] 1.1095e+13
```

```
latitude_ridge_without_trans.lm = cv.glmnet(x=data,y=latitude_new,alpha=0,
                                            nfold = 10,family = "gaussian")
plot(latitude_ridge_without_trans.lm )
```



```r
plot(latitude_ridge_without_trans.lm$cvm)
```



```r
latitude_ridge_without_trans.pred <- predict(latitude_ridge_without_trans.lm,
                                    s = latitude_ridge_without_trans.lm$lambda.min,
                                    newx = data)
latitude_ridge_without_trans.pred <- (latitude_ridge_without_trans.pred
                                      ^la_lambda_new-1)/la_lambda_new
latitude_ridge_without_trans.R2 = var(latitude_ridge_without_trans.pred)/var(latitude_new_trans)
latitude_ridge_without_trans.R2
```

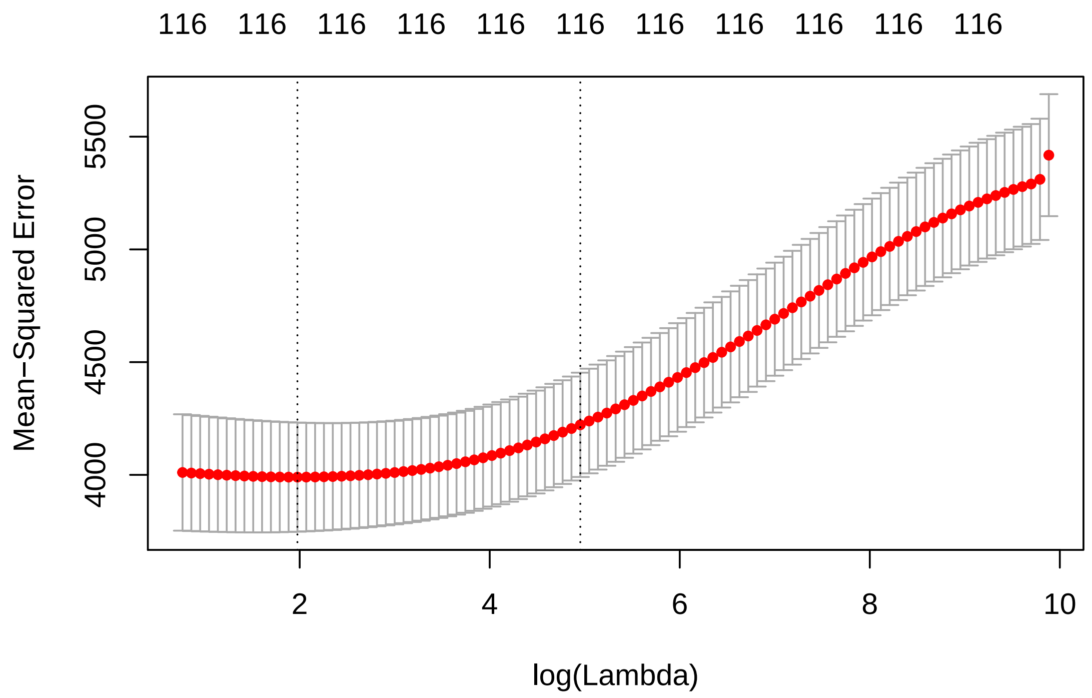
```
##             1
## 1 0.2462596
```

```r
latitude_ridge_without_trans.mse = mean((latitude_ridge_without_trans.pred -
                                           latitude_new_trans)^2)
latitude_ridge_without_trans.mse
```

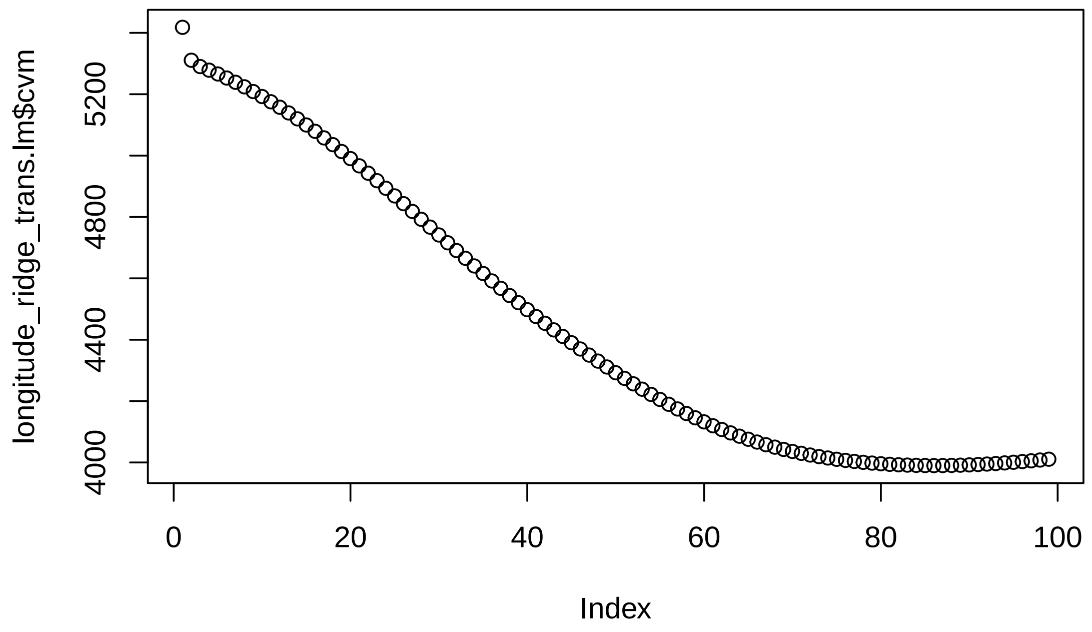
```
## [1] 1.173027e+13
```

longitude

```r
longitude_ridge_trans.lm = cv.glmnet(x=data,y=longitude_new_trans,alpha=0,
                                     nfold = 10,family = "gaussian")
plot(longitude_ridge_trans.lm)
```



```r
plot(longitude_ridge_trans.lm$cvm)
```



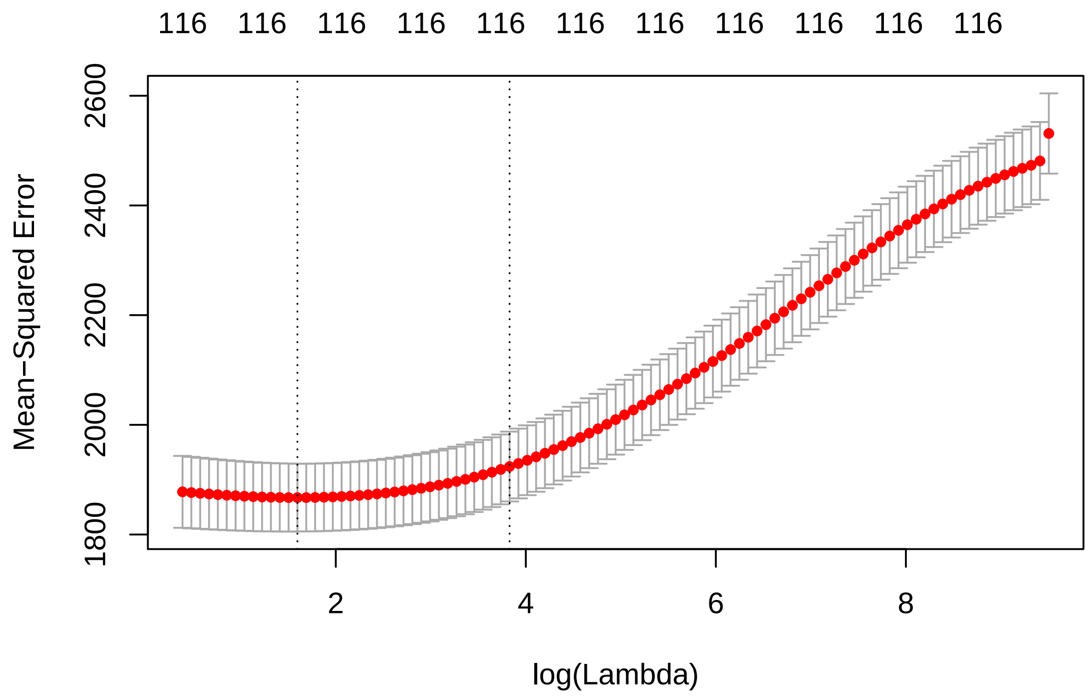
```r
longitude_ridge_trans.pred <- predict(longitude_ridge_trans.lm, s =
                                      longitude_ridge_trans.lm$lambda.min,
                                      newx = data)
longitude_ridge_trans.R2 = var(longitude_ridge_trans.pred)/var(longitude_new_trans)
longitude_ridge_trans.R2
```

```
##           1
## 1 0.2930358
```

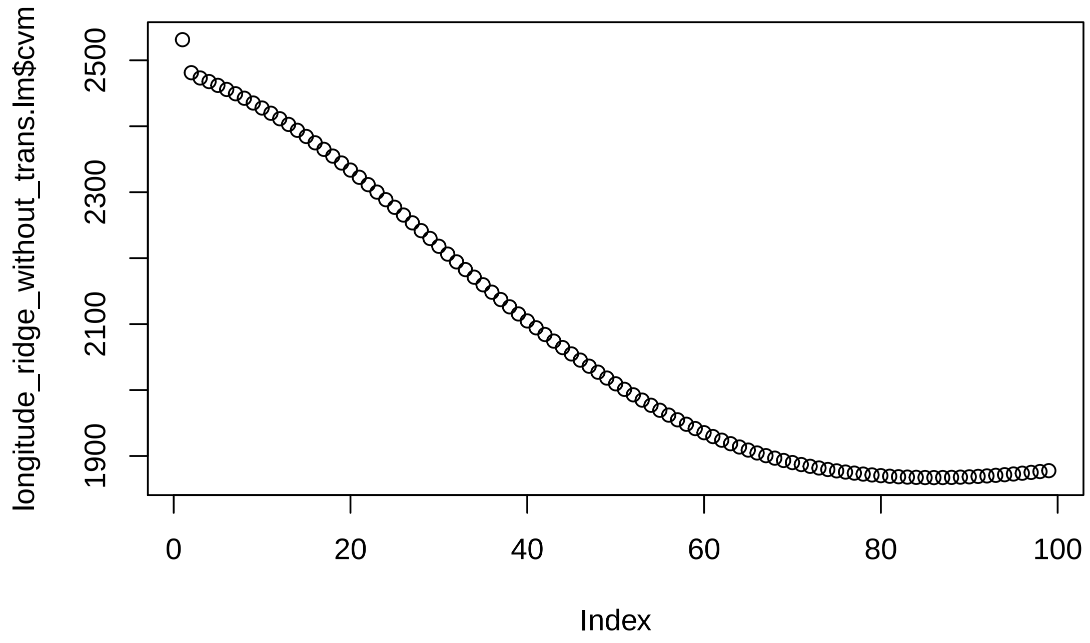
```r
longitude_ridge_trans.mse = mean((longitude_ridge_trans.pred - longitude_new_trans)^2)
longitude_ridge_trans.mse
```

```
## [1] 3533.256
```

```r
longitude_ridge_without_trans.lm = cv.glmnet(x=data,y=longitude_new,alpha=0,
                                             nfold = 10,family = "gaussian")
plot(longitude_ridge_without_trans.lm )
```



```
plot(longitude_ridge_without_trans.lm$cvm)
```



```
longitude_ridge_without_trans.pred <- predict(longitude_ridge_without_trans.lm,
                                              s = longitude_ridge_without_trans.lm$lambda.min,
                                              newx = data)
longitude_ridge_without_trans.pred <- (longitude_ridge_without_trans.pred^
                                         lo_lambda_new-1)/lo_lambda_new
longitude_ridge_without_trans.R2 = var(longitude_ridge_without_trans.pred)/var(longitude_new_trans)
longitude_ridge_without_trans.R2
```

```
##           1
## 1 0.2936222
```

```
longitude_ridge_without_trans.mse = mean((longitude_ridge_without_trans.pred -
                                            longitude_new_trans)^2)
longitude_ridge_without_trans.mse
```

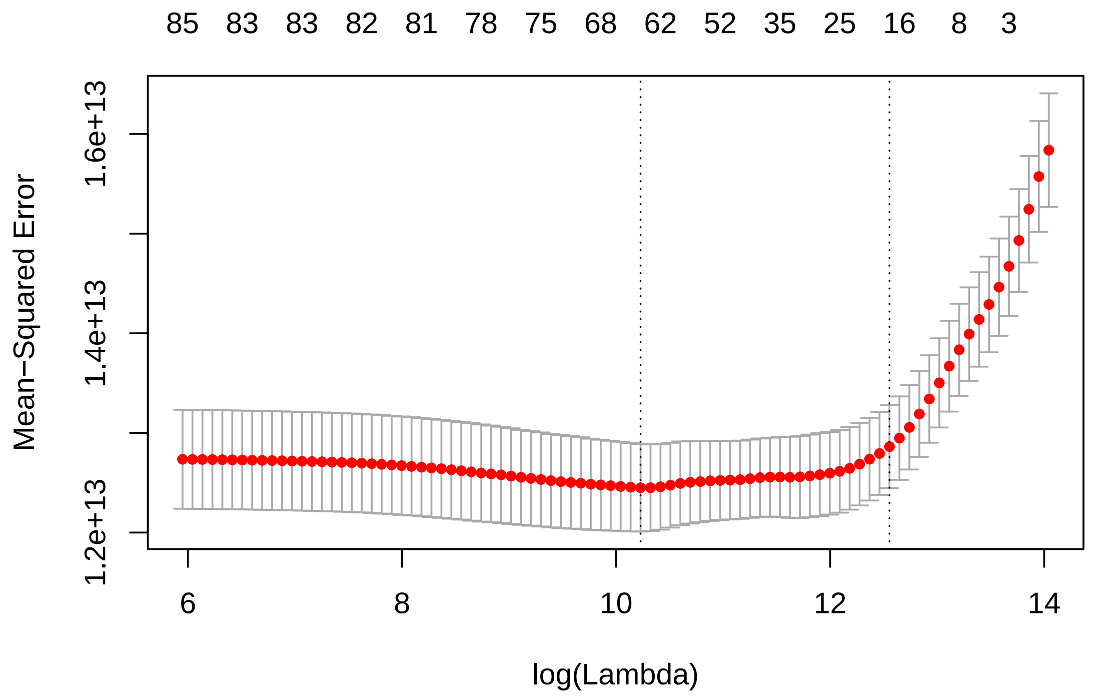
```
## [1] 3532.671
```

1.3b lasso

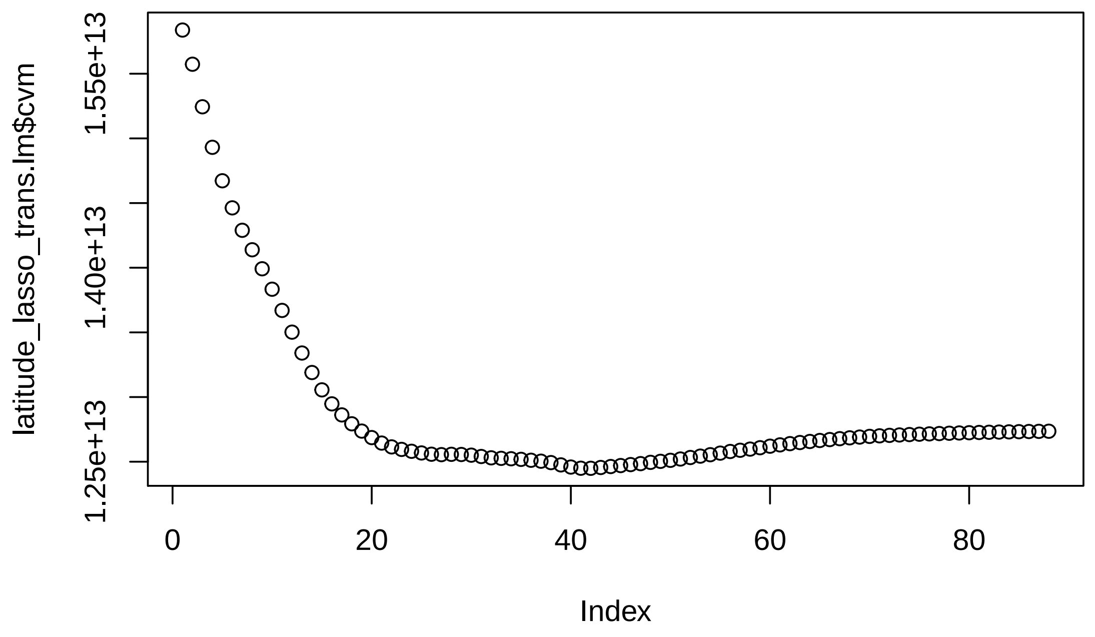
A regression regularized by L1 (equivalently, a lasso regression). You should estimate the regularization coefficient that produces the minimum error. How many variables are used by this regression? Is the regularized regression better than the unregularized regression?

latitude

```
layout(1)
latitude_lasso_trans.lm = cv.glmnet(x=data,y=latitude_new_trans,alpha=1,
                                    nfold = 10,family = "gaussian")
plot(latitude_lasso_trans.lm)
```



```r
plot(latitude_lasso_trans.lm$cvm)
```



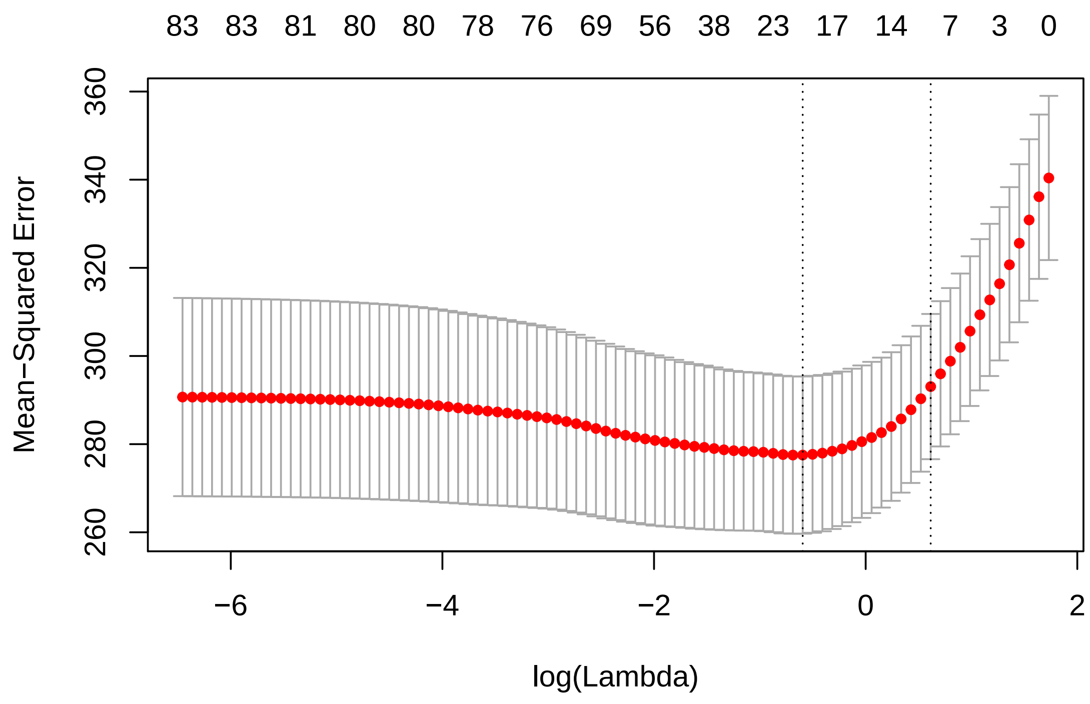
```
latitude_lasso_trans.pred <- predict(latitude_lasso_trans.lm, s =
                                       latitude_lasso_trans.lm$lambda.min, newx = data)
#latitude_lasso_trans.pred.original = (latitude_lasso.pred * la_lambda_new + 1)^(1/la_lambda_new)
latitude_lasso_trans.R2 = var(latitude_lasso_trans.pred)/var(latitude_new_trans)
latitude_lasso_trans.R2
```

```
##           1
## 1 0.2640511
```

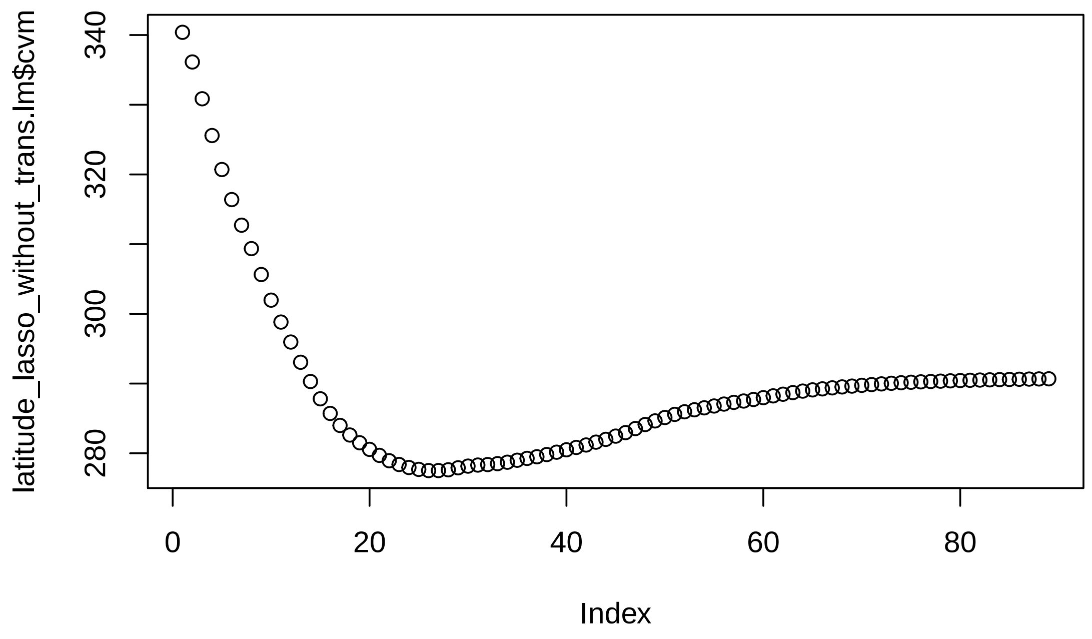
```
latitude_lasso_trans.mse = mean((latitude_lasso_trans.pred - latitude_new_trans)^2)
latitude_lasso_trans.mse
```

```
## [1] 1.098345e+13
```

```
latitude_lasso_without_trans.lm = cv.glmnet(x=data,y=latitude_new,alpha=1,
                                            nfold = 10,family = "gaussian")
plot(latitude_lasso_without_trans.lm )
```



```
plot(latitude_lasso_without_trans.lm$cvm)
```



```
latitude_lasso_without_trans.pred <- predict(latitude_lasso_without_trans.lm,
                                             s = latitude_lasso_without_trans.lm$lambda.min,
                                             newx = data)
#latitude_lasso.pred.original = (latitude_lasso.pred * la_lambda_new + 1)^(1/la_lambda_new)
latitude_lasso_without_trans.pred <- (latitude_lasso_without_trans.pred^
                                        la_lambda_new-1)/la_lambda_new
latitude_lasso_without_trans.R2 = var(latitude_lasso_without_trans.pred)/var(latitude_new_trans)
latitude_lasso_without_trans.R2
```

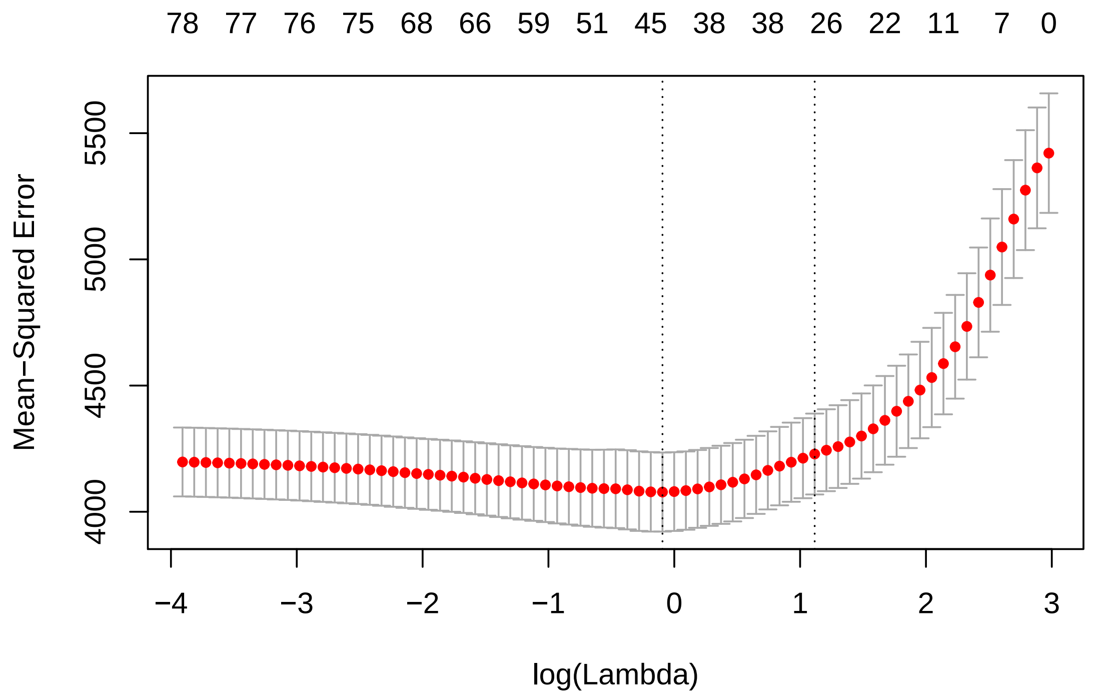
```
##           1
## 1 0.2016096
```

```
latitude_lasso_without_trans.mse = mean((latitude_lasso_without_trans.pred
                                          - latitude_new_trans)^2)
latitude_lasso_without_trans.mse
```

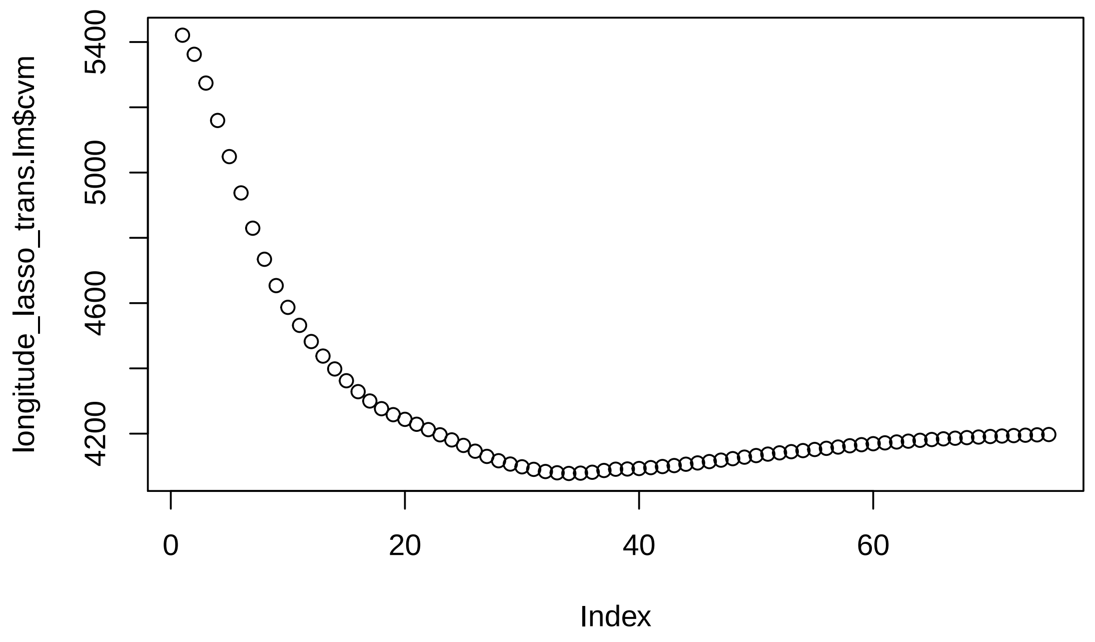
```
## [1] 1.240553e+13
```

longitude

```
longitude_lasso_trans.lm = cv.glmnet(x=data,y=longitude_new_trans,alpha=1,
                                     nfold = 10,family = "gaussian")
plot(longitude_lasso_trans.lm)
```



```
plot(longitude_lasso_trans.lm$cvm)
```



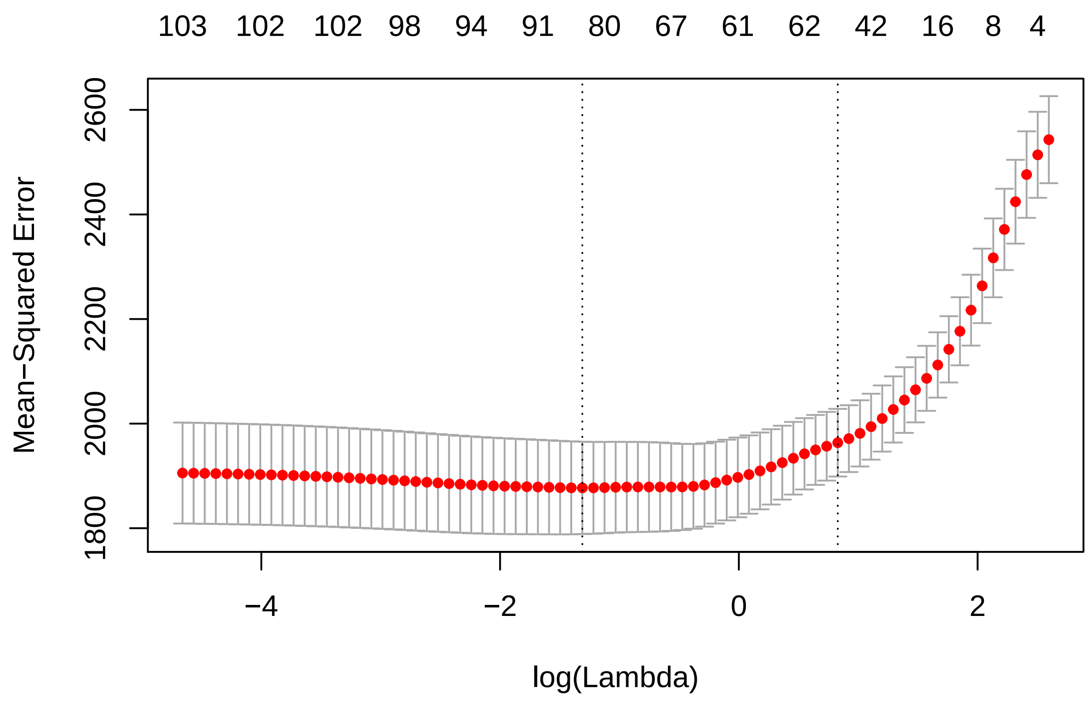
```r
longitude_lasso_trans.pred <- predict(longitude_lasso_trans.lm, s =
                              longitude_lasso_trans.lm$lambda.min, newx = data)
longitude_lasso_trans.R2 = var(longitude_lasso_trans.pred)/var(longitude_new_trans)
longitude_lasso_trans.R2
```

```
##             1
## 1 0.2708683
```

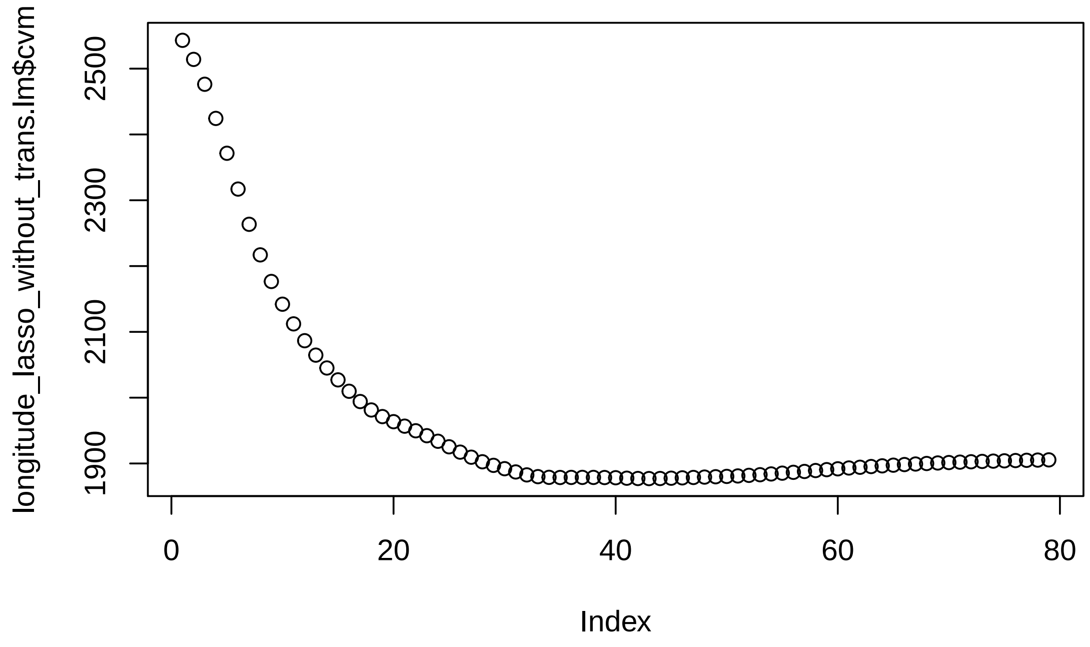
```r
longitude_lasso_trans.mse = mean((longitude_lasso_trans.pred - longitude_new_trans)^2)
longitude_lasso_trans.mse
```

```
## [1] 3618.02
```

```r
longitude_lasso_without_trans.lm = cv.glmnet(x=data,y=longitude_new,alpha=1,
                                             nfold = 10,family = "gaussian")
plot(longitude_lasso_without_trans.lm )
```



```r
plot(longitude_lasso_without_trans.lm$cvm)
```



```
longitude_lasso_without_trans.pred <- predict(longitude_lasso_without_trans.lm,
                                       s = longitude_lasso_without_trans.lm$lambda.min,
                                       newx = data)
longitude_lasso_without_trans.pred <- (longitude_lasso_without_trans.pred^
                                          lo_lambda_new-1)/lo_lambda_new
longitude_lasso_without_trans.R2 = var(longitude_lasso_without_trans.pred)/var(longitude_new_trans)
longitude_lasso_without_trans.R2
```

```
##           1
## 1 0.3147185
```

```
longitude_lasso_without_trans.mse = mean((longitude_lasso_without_trans.pred -
                                             longitude_new_trans)^2)
longitude_lasso_without_trans.mse
```

```
## [1] 3493.799
```

```
model = c("latitude_new_trans.lm","latitude_new_without_trans.lm",
          "latitude_ridge_trans.lm","latitude_ridge_without_trans.lm",
          "latitude_lasso_trans.lm","latitude_lasso_without_trans.lm",
          "longitude_new_trans.lm","longitude_new_without_trans.lm",
          "longitude_ridge_trans.lm","longitude_ridge_without_trans.lm",
          "longitude_lasso_trans.lm","longitude_lasso_without_trans.lm"
          )
R2 = c(latitude_new_trans.r2,latitude_new_without_trans.r2,
      latitude_ridge_trans.R2,latitude_ridge_without_trans.R2,
      latitude_lasso_trans.R2,latitude_lasso_without_trans.R2,
      longitude_new_trans.r2,longitude_new_without_trans.r2,
      longitude_ridge_trans.R2,longitude_ridge_without_trans.R2,
      longitude_lasso_trans.R2,longitude_lasso_without_trans.R2
)
mse = c(latitude_new_trans.mse,latitude_new_without_trans.mse,
      latitude_ridge_trans.mse,latitude_ridge_without_trans.mse,
      latitude_lasso_trans.mse,latitude_lasso_without_trans.mse,
      longitude_new_trans.mse,longitude_new_without_trans.mse,
      longitude_ridge_trans.mse,longitude_ridge_without_trans.mse,
      longitude_lasso_trans.mse,longitude_lasso_without_trans.mse
)
cbind(model,R2,mse)
```

```
##       model                              R2
##  [1,] "latitude_new_trans.lm"            "0.278205162234764"
##  [2,] "latitude_new_without_trans.lm"    "0.24116849301382"
##  [3,] "latitude_ridge_trans.lm"          "0.240924464397298"
##  [4,] "latitude_ridge_without_trans.lm"  "0.246259572258212"
##  [5,] "latitude_lasso_trans.lm"          "0.264051079585654"
##  [6,] "latitude_lasso_without_trans.lm"  "0.201609622712887"
##  [7,] "longitude_new_trans.lm"           "0.318705154418536"
##  [8,] "longitude_new_without_trans.lm"   "0.318176624480052"
##  [9,] "longitude_ridge_trans.lm"         "0.293035773421009"
## [10,] "longitude_ridge_without_trans.lm" "0.293622207449689"
## [11,] "longitude_lasso_trans.lm"         "0.270868290616546"
## [12,] "longitude_lasso_without_trans.lm" "0.314718521104191"
##       mse
##  [1,] "10659299446201.1"
##  [2,] "11231193445874.9"
##  [3,] "11095000365922.8"
##  [4,] "11730270427932.2"
##  [5,] "10983446421506"
##  [6,] "12405533949603"
##  [7,] "3441.22950864091"
##  [8,] "3442.1343335657"
##  [9,] "3533.25648606586"
## [10,] "3532.6713372049"
## [11,] "3618.0203618532"
## [12,] "3493.79880603119"
```

So according to R_2 and MSE in the list, we can find:

For latitude prediction, the model after the transformation without regularization term performs better than both the lasso and ridge regression. The lasso regression preforms better than the ridge regression.

For longitude prediction, the model after the transformation without regularization term performs better than both the lasso and ridge regression. The lasso regression preforms better than the ridge regression.

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. You should ignore outliers, but you should try the various regularization schemes we have discussed.

Answer:

```
read.table("default_of_credit_card_clients.txt",header=T)->data
data[1,]
```

```
##   ID LIMIT_BAL SEX EDUCATION MARRIAGE AGE PAY_0 PAY_2 PAY_3 PAY_4 PAY_5
## 1  1     20000   2         2        1  24     2     2    -1    -1    -2
##   PAY_6 BILL_AMT1 BILL_AMT2 BILL_AMT3 BILL_AMT4 BILL_AMT5 BILL_AMT6
## 1    -2      3913      3102       689         0         0         0
##   PAY_AMT1 PAY_AMT2 PAY_AMT3 PAY_AMT4 PAY_AMT5 PAY_AMT6 default
## 1        0      689        0        0        0        0       1
```

```
dim(data)
```

```
## [1] 30000    25
```

```
x = as.matrix(data[,2:24])
y = data[25]
y <- as.factor(as.matrix(y))
alpha = c(0,0.25,0.5,0.75,1)
acc = NULL
for (i in 1:length(alpha)){
  lm = cv.glmnet(x, y, family = "binomial", alpha = alpha[i], type.measure = "class",
               nfold = 10)
  #plot(lm,main=paste("alpha = ",alpha[i]))
  lm.pred <- predict(lm, x, type = "class", s = "lambda.min")
  lm.acc = sum(lm.pred == y)/length(y)
  acc[i] = lm.acc
}
cbind(alpha,acc)
```

```
##      alpha       acc
## [1,]  0.00 0.8069667
## [2,]  0.25 0.8099667
## [3,]  0.50 0.8105667
## [4,]  0.75 0.8104000
## [5,]  1.00 0.8103000
```

We use the logistic regression to predict whether the user defaults. According to the accuary, we think the model with `alpha = 0.5` performs the best and the accuracy is 0.8105667.

Problem 3 A wide dataset, from cancer genetics

In "Broad patterns of gene expression revealed by clustering of tumor and normal colon tissues probed by oligonucleotide arrays" by U. Alon, N. Barkai, D. A. Notterman, K. Gish, S. Ybarra, D. Mack, and A. J. Levine, Proc. Natl. Acad. Sci. USA, Vol. 96, Issue 12, 6745-6750, June 8, 1999, authors collected data giving gene expressions for tumorous and normal colon tissues. You will find this dataset here. There is a matrix of gene expression levels for 2000 genes (these are the independent variables) for 62 tissue samples. As you can see, there are a lot more independent variables than there are data items. At that website, you will also find a file giving which sample is tumorous and which is normal.

Use a binomial regression model (i.e. logistic regression) with the lasso to predict tumorous/normal. Use cross-validation to assess how accurate your model is. Report both AUC (below) and deviance. How many genes does the best model use?

Answer:

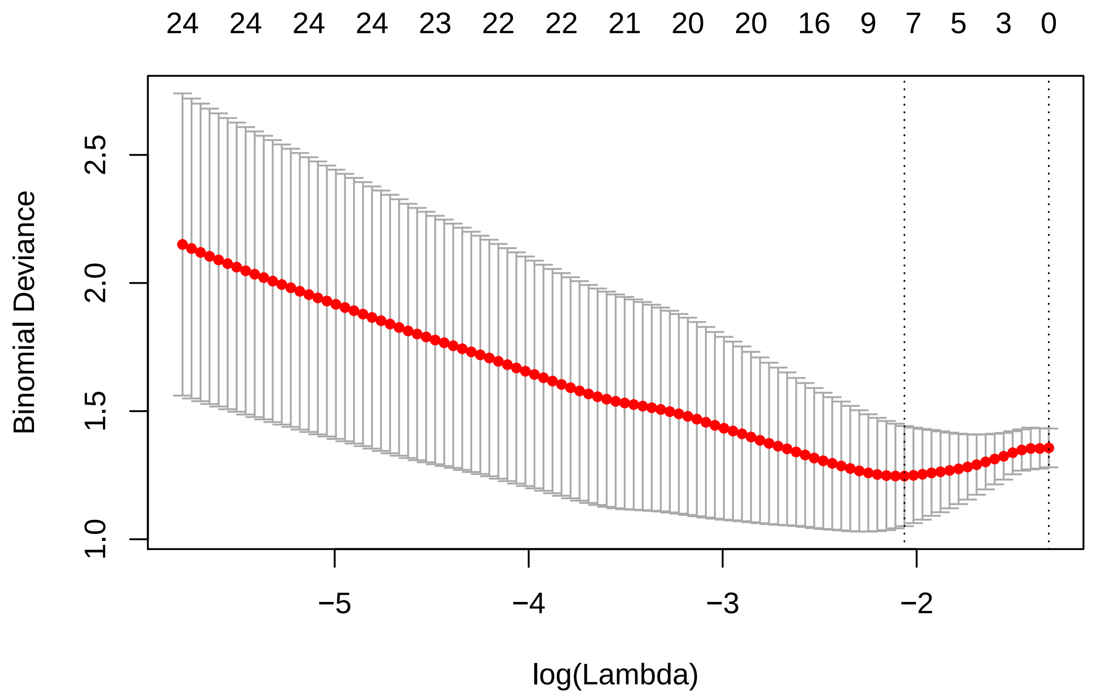
```r
library(caret)
```

```
## Loading required package: lattice

## Loading required package: ggplot2
```

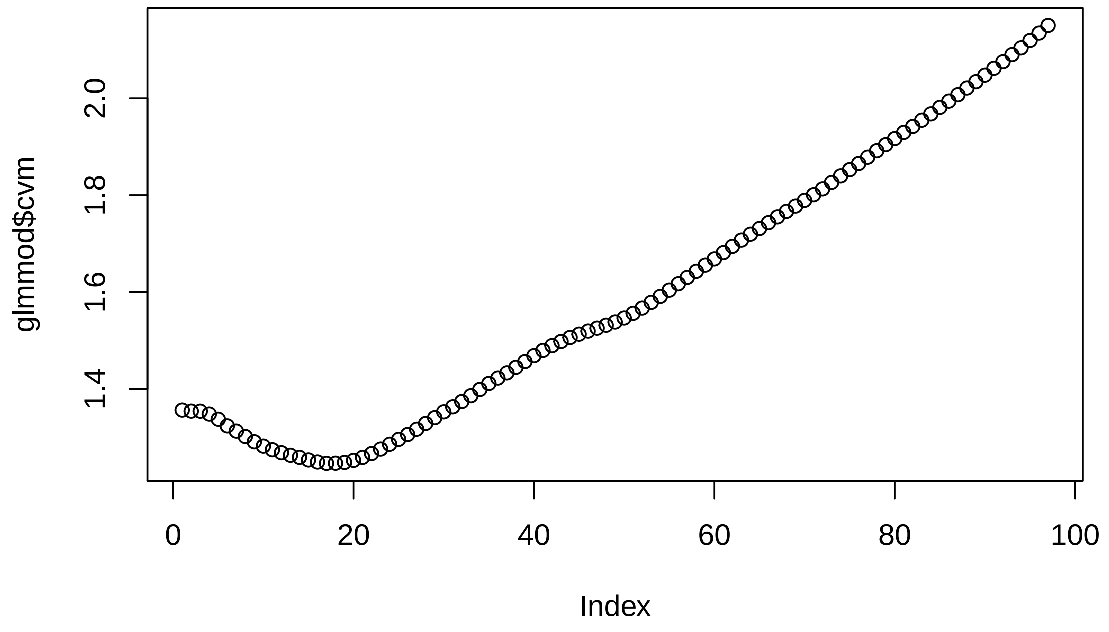
```r
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following object is masked from 'package:glmnet':
##
##     auc

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```r
set.seed(1)
df <- read.table("tissues.txt",sep="\t", header= F, fill = TRUE)
# a positive sign to a normal tissue, and a negative sign to a tumor tissue.
result <- as.vector(df$V1)
result[result>0] <- 1
result[result<=0] <- 0

data <- (matrix(scan("matrix.txt"), nrow = 2000, byrow = T))
data <- t(data)       # row = 62  col = 2000   each row is a sample with 2000 genes
#f1 <- SubLasso(data, result, nfold = 5)

accuracy = 0

cv_partition <- createDataPartition(y = result, p = 0.7, list=FALSE)
train_result <- result[cv_partition]
test_result <- result[-cv_partition]
train_data <- data[cv_partition, ]
test_data <- data[-cv_partition, ]

glmmod <- cv.glmnet(train_data, train_result, alpha=1, family="binomial", type.measure = "deviance", nf

plot(glmmod)
```



```
plot(glmmod$cvm)
```



```r
best_lambda = 0
best_accuracy = 0
for(i in 1:length(glmmod$lambda))
{
  prediction <- predict(glmmod, train_data, s = glmmod$lambda[i], type = "class")    # Give the misclas
  prediction <- as.numeric(prediction)
  accuracy <- sum(prediction == train_result)/length(prediction)
  if(best_accuracy < accuracy)
  {
    best_accuracy = accuracy
    best_lambda = glmmod$lambda[i]
  }
}
best_lambda
```

```
## [1] 0.05762239
```

```r
best_accuracy
```

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

```r
coefficient <- coef(glmmod, s = best_lambda)
sum(coefficient != 0)
```

```
## [1] 21
```

```r
train.pred <- predict(glmmod, train_data, s = best_lambda, type = "class")
train.pred <- as.numeric(train.pred)
train.acc <- sum(train.pred == train_result)/length(train.pred)
train.acc
```

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

```r
train.roc <- roc(response=as.factor(train_result), predictor=train.pred )
auc(train.roc)
```

```
## Area under the curve: 1
```

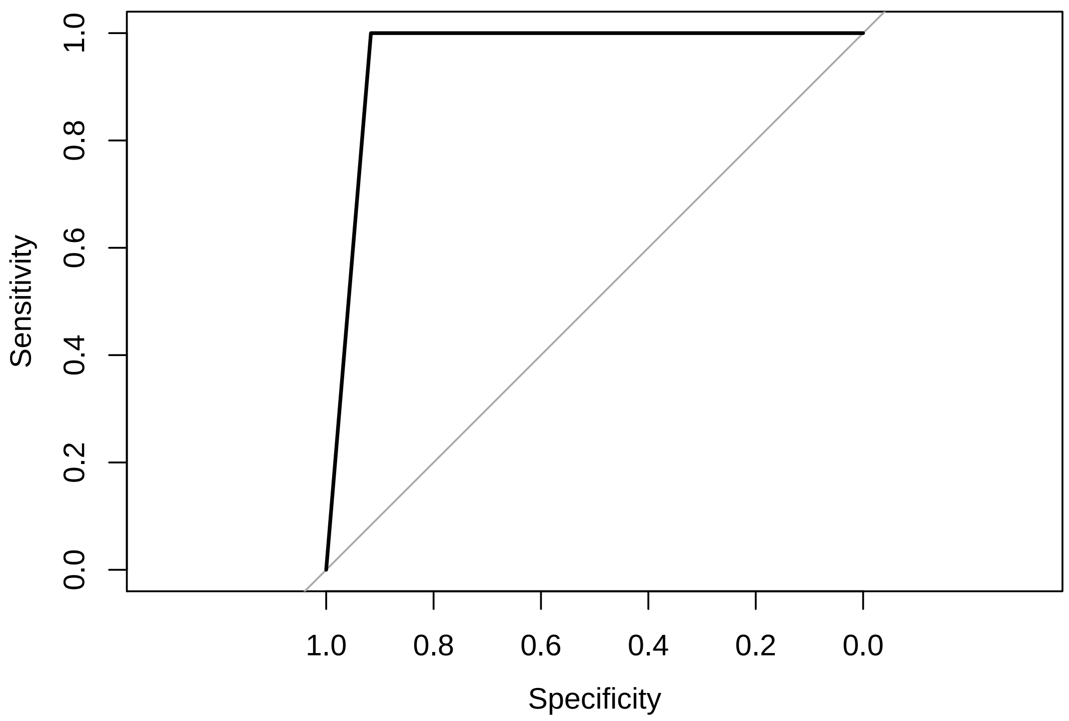
```r
deviance = glmmod$cvm[which(glmmod$lambda == best_lambda)]

test.pred <- predict(glmmod, test_data, s = best_lambda, type = "class")
test.pred <- as.numeric(test.pred)
test.acc <- sum(test.pred == test_result)/length(test.pred)
test.acc
```

```
## [1] 0.9444444
```

```r
test.roc <- roc(response=as.factor(test_result), predictor=test.pred)
auc(test.roc)
```

```
## Area under the curve: 0.9583
```

```r
plot(test.roc)
```



```
##
## Call:
## roc.default(response = as.factor(test_result), predictor = test.pred)
##
## Data: test.pred in 12 controls (as.factor(test_result) 0) < 6 cases (as.factor(test_result) 1).
## Area under the curve: 0.9583
```

So the lasso regression with `lambda` = 0.0576224 performs the best. The model uses 21 genes to predict whether the sample is tumorous or normal. On the training data, the accuracy is 1. The deviance of the model is 1.3990023, and the AUC is 1. On the test data, the accuracy is 0.9444444 and AUC is 0.9583333.