HW6

Ashish Kumar // ashishk2@illinois.edu (mailto:ashishk2@illinois.edu)

1. Linear regression

```
rm(list = ls())
library(stats)
library(gdata)
library(caret)
library(caTools)
library(glmnet)
library(plotmo)
library(MASS)
library(ModelMetrics)
library(knitr)
set.seed(2018)
origData = read.table("~/cs498aml/music/default_plus_chromatic_features_1059_tracks.t
xt", header = FALSE, sep = ",")
```

Build a straight forward linear regression

```
reg1.lm = lm(V117 ~ . - V118, data=origData)
reg1.mse = mean((origData$V117 - reg1.lm$fitted.values)^2)
reg2.lm = lm(V118 ~ . - V117, data=origData)
reg2.mse = mean((origData$V118 - reg2.lm$fitted.values)^2)
rsq_table = data.frame(Model=c("No transformation "),
                       LatAIC = c(AIC(reg1.lm)),
                       LatBIC = c(BIC(reg1.lm)),
                       LongAIC = c(AIC(reg2.lm)),
                       LongBIC = c(BIC(reg2.lm)))
```

R-Squared values

```
print(paste("Adjusted R-Squared (Lat.)",summary(reg1.lm)$adj.r.squared))
```

```
## [1] "Adjusted R-Squared (Lat.) 0.241168493013819"
```

```
print(paste("Adjusted R-Squared (Long.)",summary(reg2.lm)$adj.r.squared))
```

```
## [1] "Adjusted R-Squared (Long.) 0.318176624480052"
```

```
print(paste("R-Squared (Lat.)",summary(reg1.lm)$r.squared))
```

```
## [1] "R-Squared (Lat.) 0.292809200483578"
```

```
print(paste("R-Squared (Long.)",summary(reg2.lm)$r.squared))
```

```
## [1] "R-Squared (Long.) 0.364576702965342"
```

```
print(paste("MSE (Lat.)",reg1.mse))
```

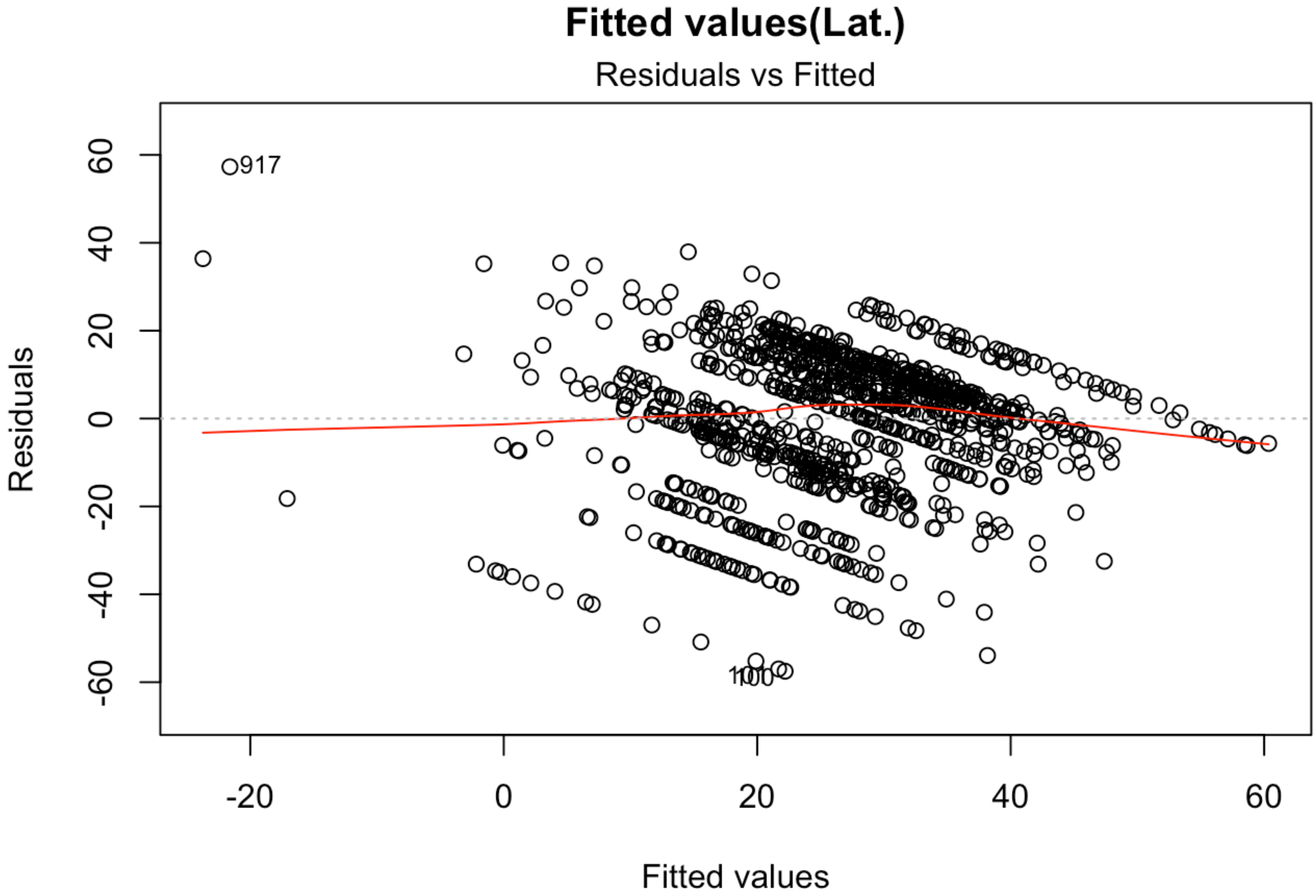
```
## [1] "MSE (Lat.) 240.748148858786"
```

```
print(paste("MSE (Long.)",reg2.mse))
```

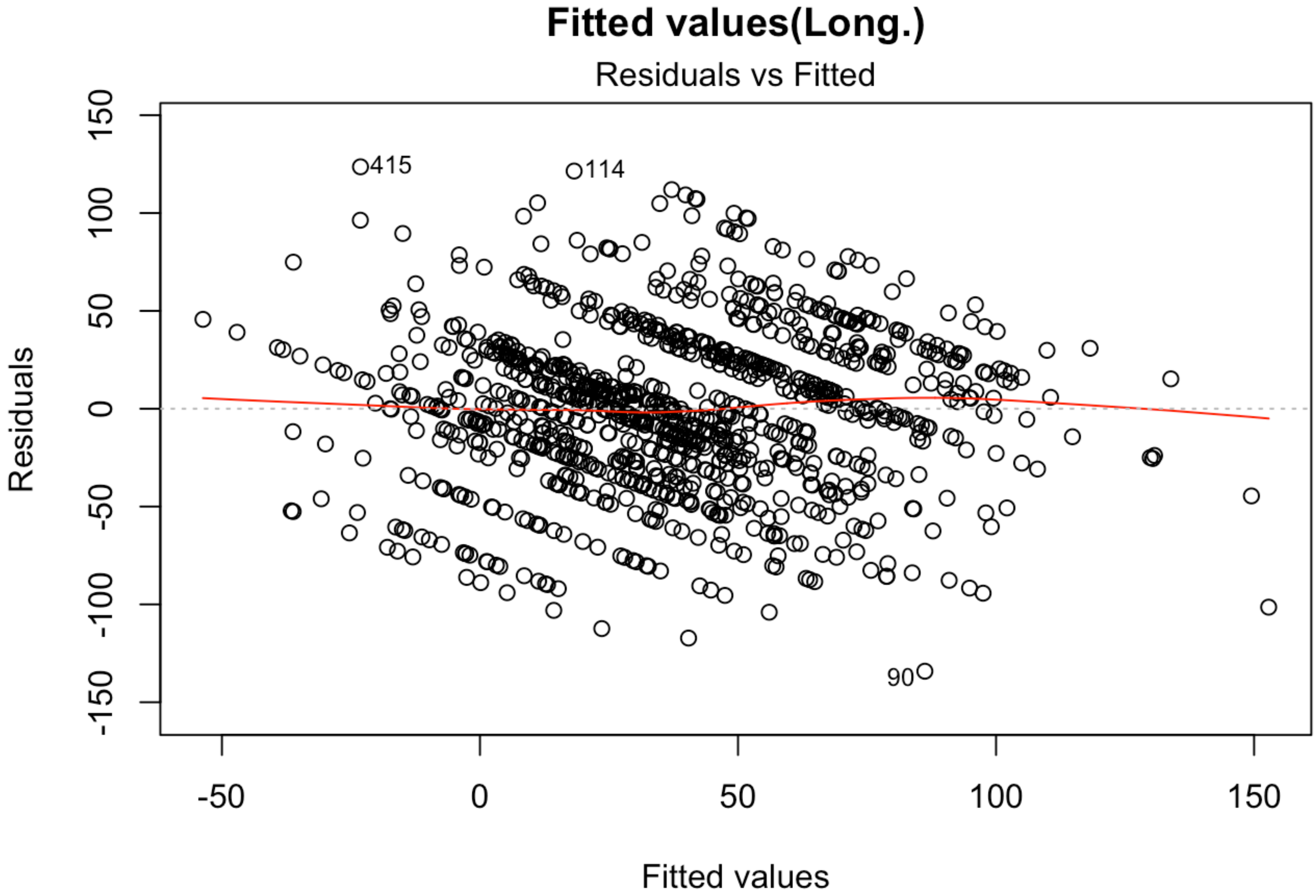
```
## [1] "MSE (Long.) 1613.81929356418"
```

Plots

```
plot(reg1.lm, which=1, main="Fitted values(Lat.)", sub = "")
```

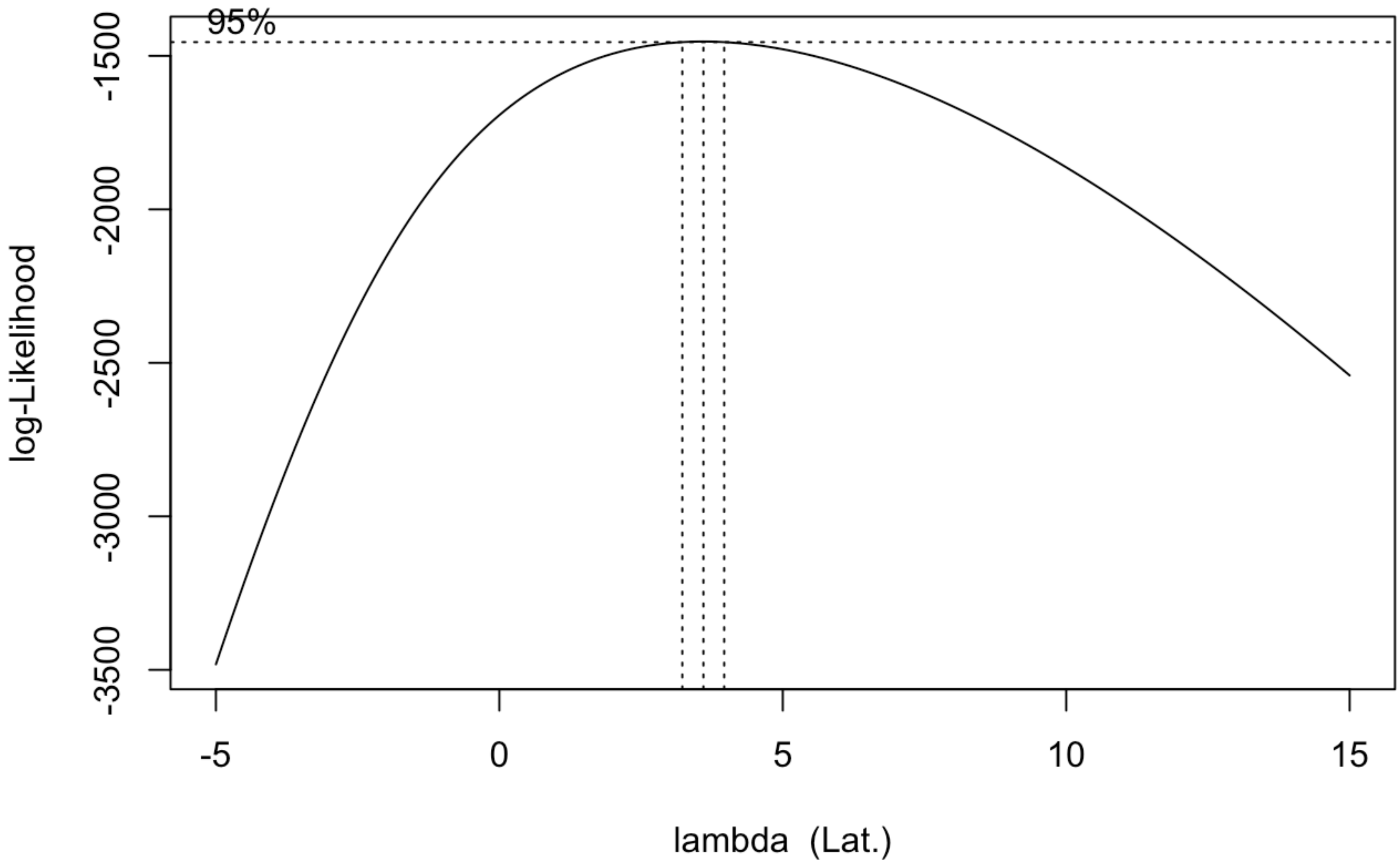



```
plot(reg2.lm, which=1, main="Fitted values(Long.)", sub = "")
```

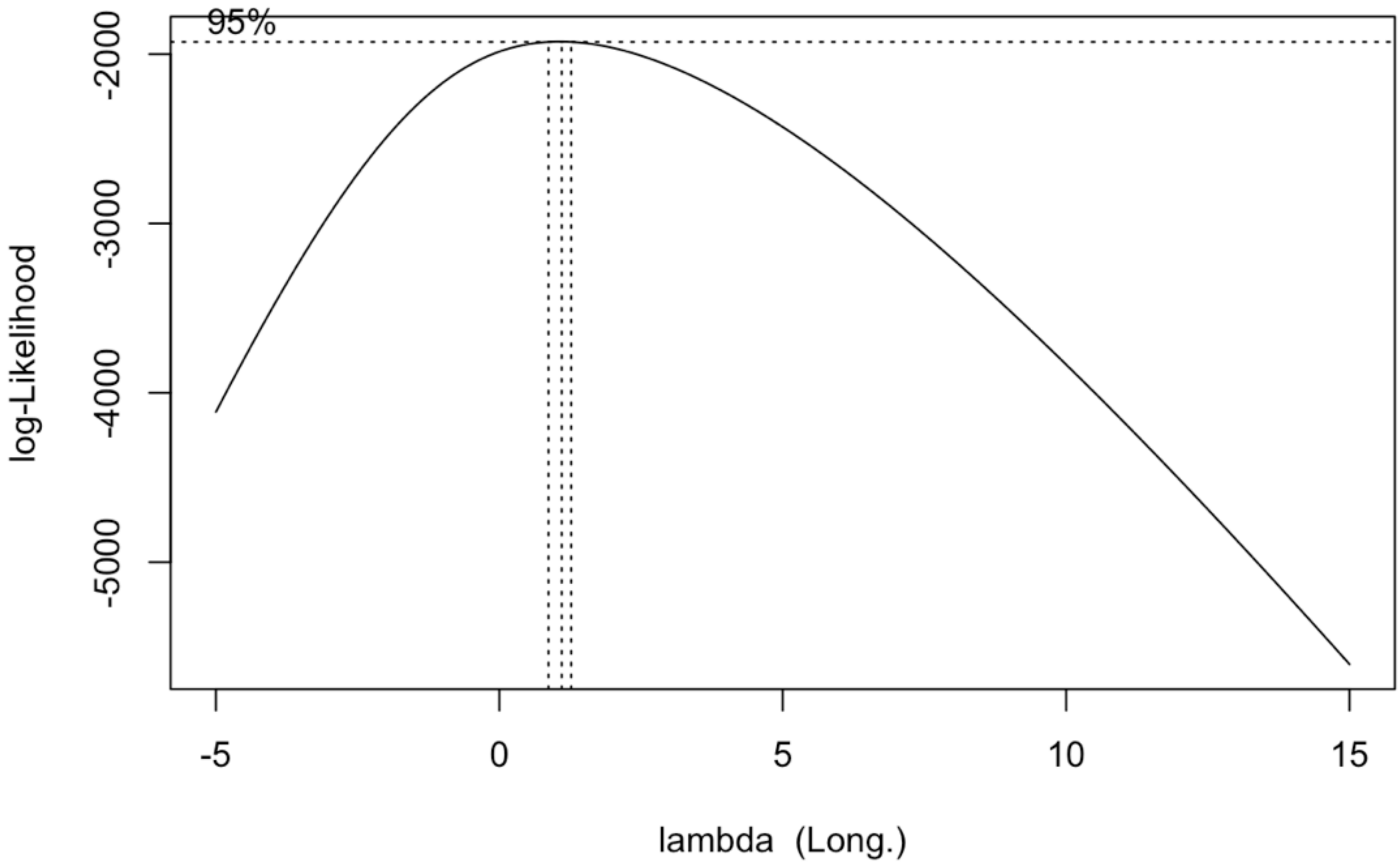


Perform Box-Cox transformation

```
data = read.table("~/cs498aml/music/default_plus_chromatic_features_1059_tracks.txt",
header = FALSE, sep = ",")
data$V117 = data$V117 + 90
data$V118 = data$V118 + 180

reg3.lm = lm(V117 ~ . - V118, data=data)
reg4.lm = lm(V118 ~ . - V117, data=data)

bxcxLat = MASS::boxcox(reg3.lm, lambda = seq(-5, 15, 1/10),
                   plotit = TRUE,
                   xlab = paste(expression(lambda), " (Lat.)"))
```




```
bxcxLong = MASS::boxcox(reg4.lm,
                    lambda = seq(-5, 15, 1/10),
                    plotit = TRUE,
                    xlab = paste(expression(lambda), " (Long.)"))
```



Choose best box cox lambda value

```
lambdaLat = bxcxLat$x[which(bxcxLat$y == max(bxcxLat$y))]
lambdaLong = bxcxLong$x[which(bxcxLong$y == max(bxcxLong$y))]

print(paste("Lat : The best value of lamba is ", lambdaLat, " with high log likelihoo
d of ", max(bxcxLat$y)))
```

```
## [1] "Lat : The best value of lamba is  3.6  with high log likelihood of  -1453.223
00489803"
```

```
print(paste("Long : The best value of lamba is ", lambdaLong, " with high log likelih
ood of ", max(bxcxLong$y)))
```

```
## [1] "Long : The best value of lamba is  1.1  with high log likelihood of  -1925.65
7954218"
```

Choose best model

```
data$V117 = (data$V117^lambdaLat - 1)/lambdaLat
data$V118 = (data$V118^lambdaLong - 1)/lambdaLong
bxcxLat.lm = lm(V117 ~ . -V118, data=data)
bxcxLong.lm = lm(V118 ~ . -V117, data=data)

rsq_table = rbind(rsq_table,
                  data.frame(Model=c("Box Cox Transformation "),
                             LatAIC = c(AIC(bxcxLat.lm)),
                             LatBIC = c(BIC(bxcxLat.lm)),
                             LongAIC = c(AIC(bxcxLong.lm)),
                             LongBIC = c(BIC(bxcxLong.lm))))
```

Smaller the AIC or BIC, the better is the model

```
knitr::kable(rsq_table, format="markdown", digits=6, align=c('c','c','c','c','c'), pa
dding=20, caption="Model Comparison")
```

Model	LatAIC	LatBIC	LongAIC	LongBIC
No transformation	8960.605	9328.02	10975.47	11342.88
Box Cox Transformation	34859.710	35227.13	12109.86	12477.27

> Untransformed model is better because it produced the smallest AIC and BIC for both latitude and longitude. I will use untransformed data for the rest of this exercise

Unregularized

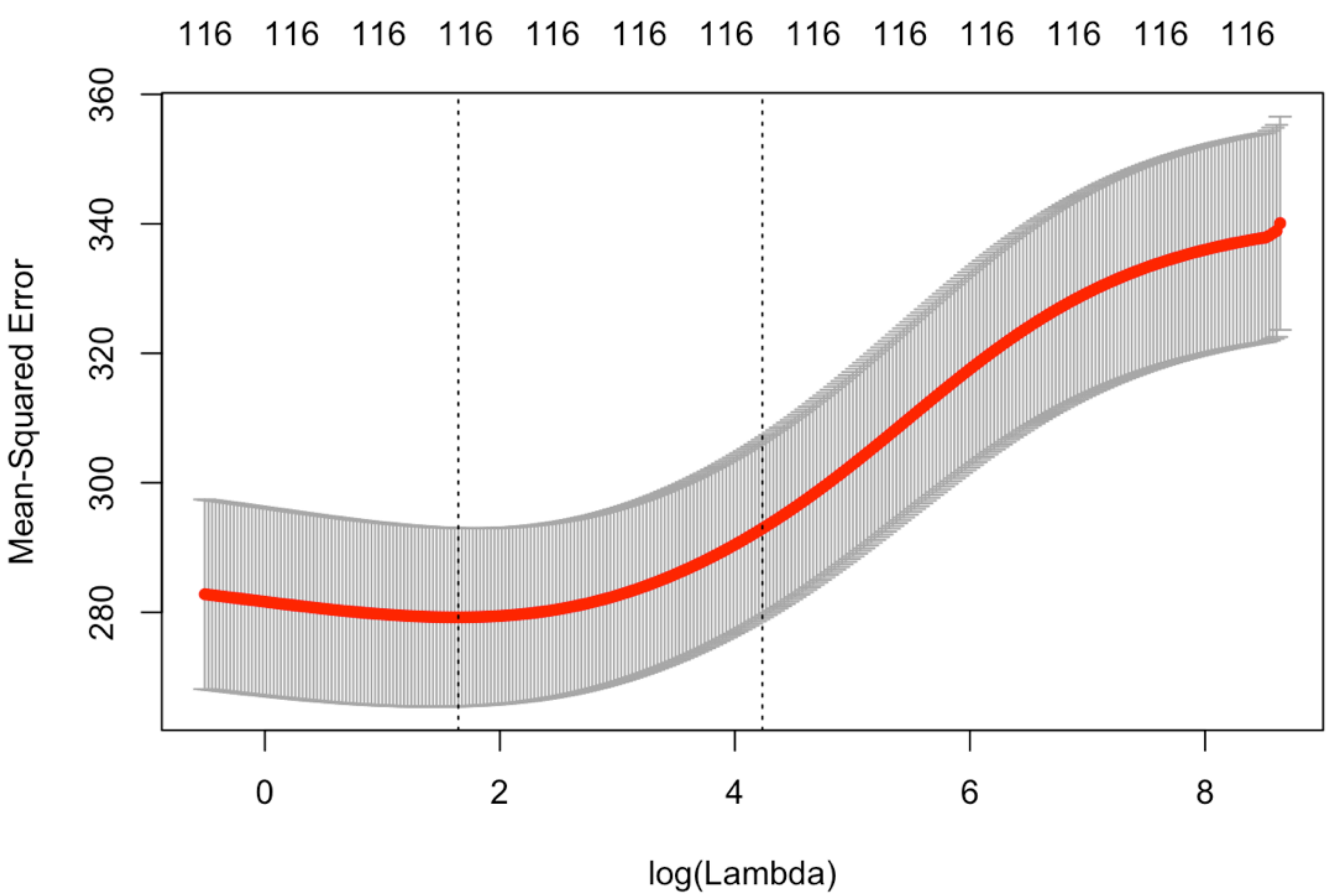
```
data = origData
noreg1.model = cv.glmnet(as.matrix(data[,-c(117,118)]), data$V117, family="gaussian",
lambda = seq(0,0.000001, 0.000001), alpha=0)
noreg2.model = cv.glmnet(as.matrix(data[,-c(117,118)]), data$V118, family="gaussian",
lambda = seq(0,0.000001, 0.000001), alpha=0)
cv_rslt_table = data.frame(Items=c("Unregularized "),
                           CVErrLat = c(noreg1.model$cvm[2]),
                           CVErrLong = c(noreg2.model$cvm[2]),
                           LambdaLat = c(NA),
                           LambdaLong = c(NA))
```

L2 Regularization

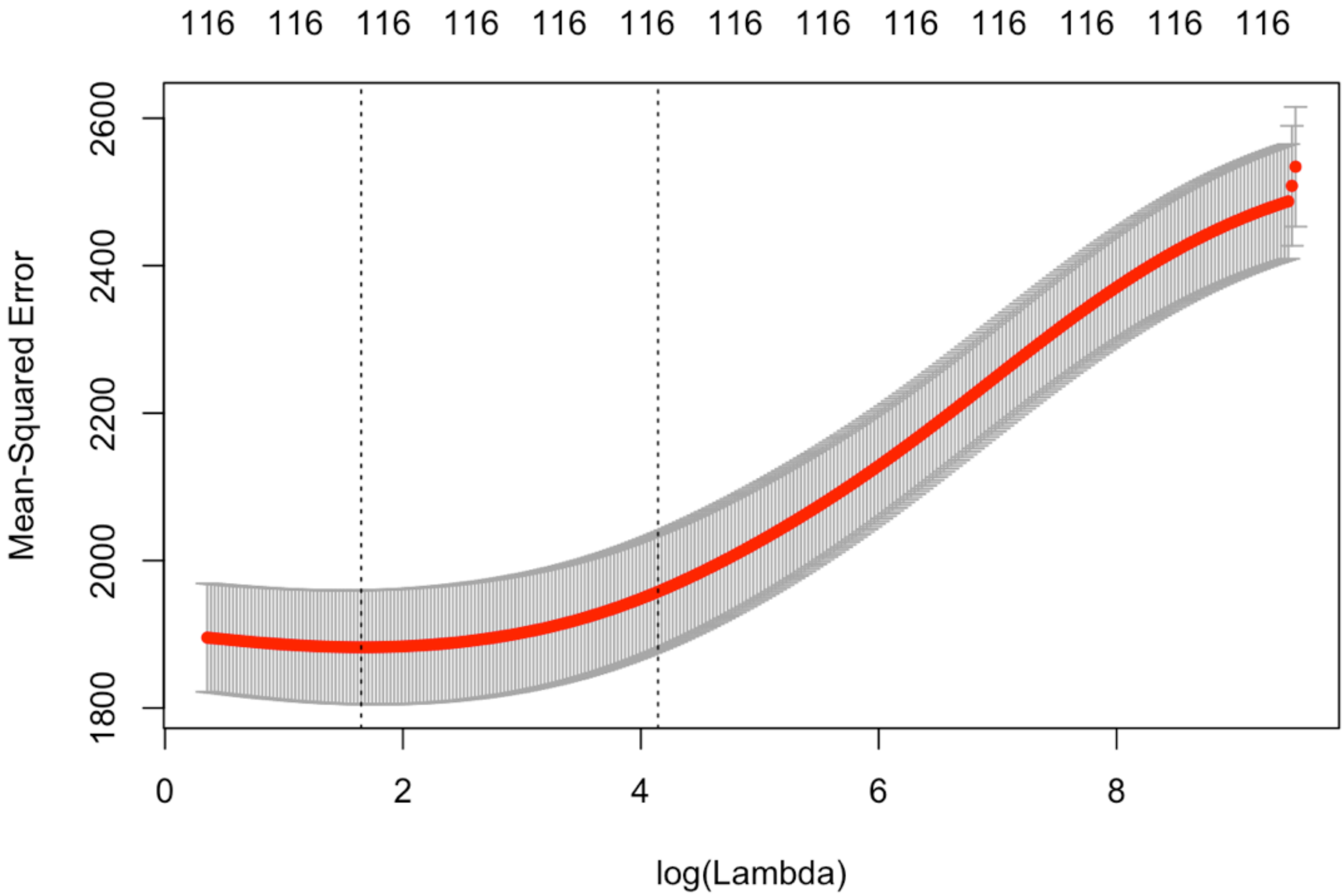
```
ridge1.model = cv.glmnet(as.matrix(as.matrix(data[,-c(117,118)])), data$V117,  family
="gaussian", nlambda = 300, alpha=0)
idx1 = which(ridge1.model$cvm == min(ridge1.model$cvm))
ridge2.model = cv.glmnet(as.matrix(as.matrix(data[,-c(117,118)])), data$V118,  family
="gaussian", nlambda = 300, alpha=0)
idx2 = which(ridge2.model$cvm == min(ridge2.model$cvm))
```

Plots

```
plotres(ridge1.model, info=TRUE, which = 1, caption = "Ridge (L2) Lat.")
```




```
plotres(ridge2.model, info=TRUE, which = 1, caption = "Ridge (L2) Long.")
```



Best regularization values

```
print(paste("L2 Lamba Lat. ", ridge1.model$lambda[idx1]))
```

```
## [1] "L2 Lamba Lat.  5.1866448037629"
```

```
print(paste("L2 Lamba Long. ", ridge2.model$lambda[idx2]))
```

```
## [1] "L2 Lamba Long.  5.20322382700935"
```

```
cv_rslt_table = rbind(cv_rslt_table,
                     data.frame(Items=c("Ridge"),
                                CVErrLat = c(min(ridge1.model$cvm)),
                                CVErrLong = c(min(ridge2.model$cvm)),
                                LambdaLat = c(ridge1.model$lambda[idx1]),
                                LambdaLong = c(ridge2.model$lambda[idx2])))
```

```
knitr::kable(cv_rslt_table, format="markdown", digits=6, align=c('c','c','c','c','c')
, padding=20, caption="CVErr, Lambda Comparison")
```

Items	CVErrLat	CVErrLong	LambdaLat	LambdaLong
Unregularized	289.7296	1929.891	NA	NA
Ridge	279.2253	1882.386	5.186645	5.203224

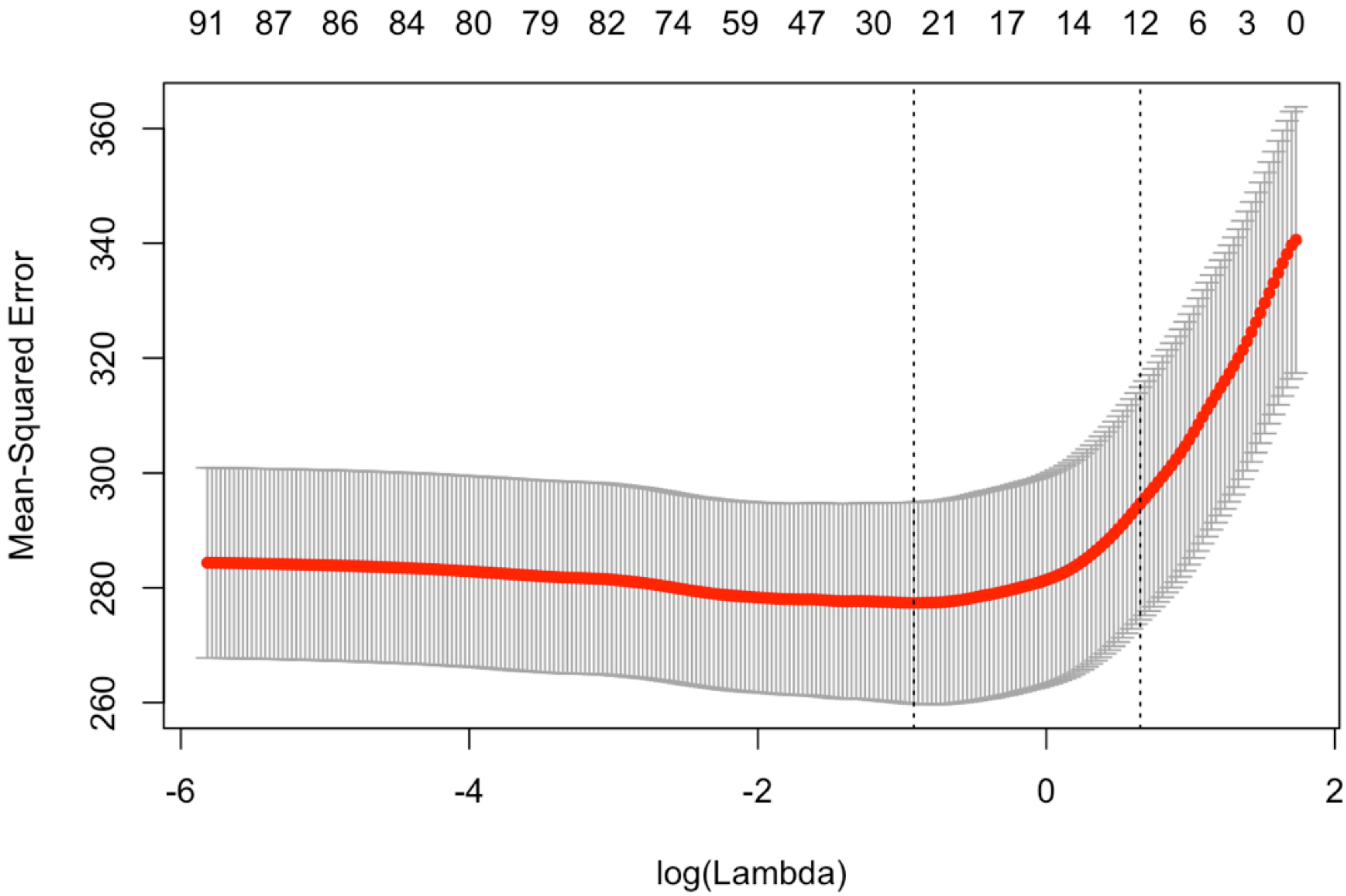
> Ridge (L2) Model is better than Unregularized version because it has lower CV Error.

L1 Regularization

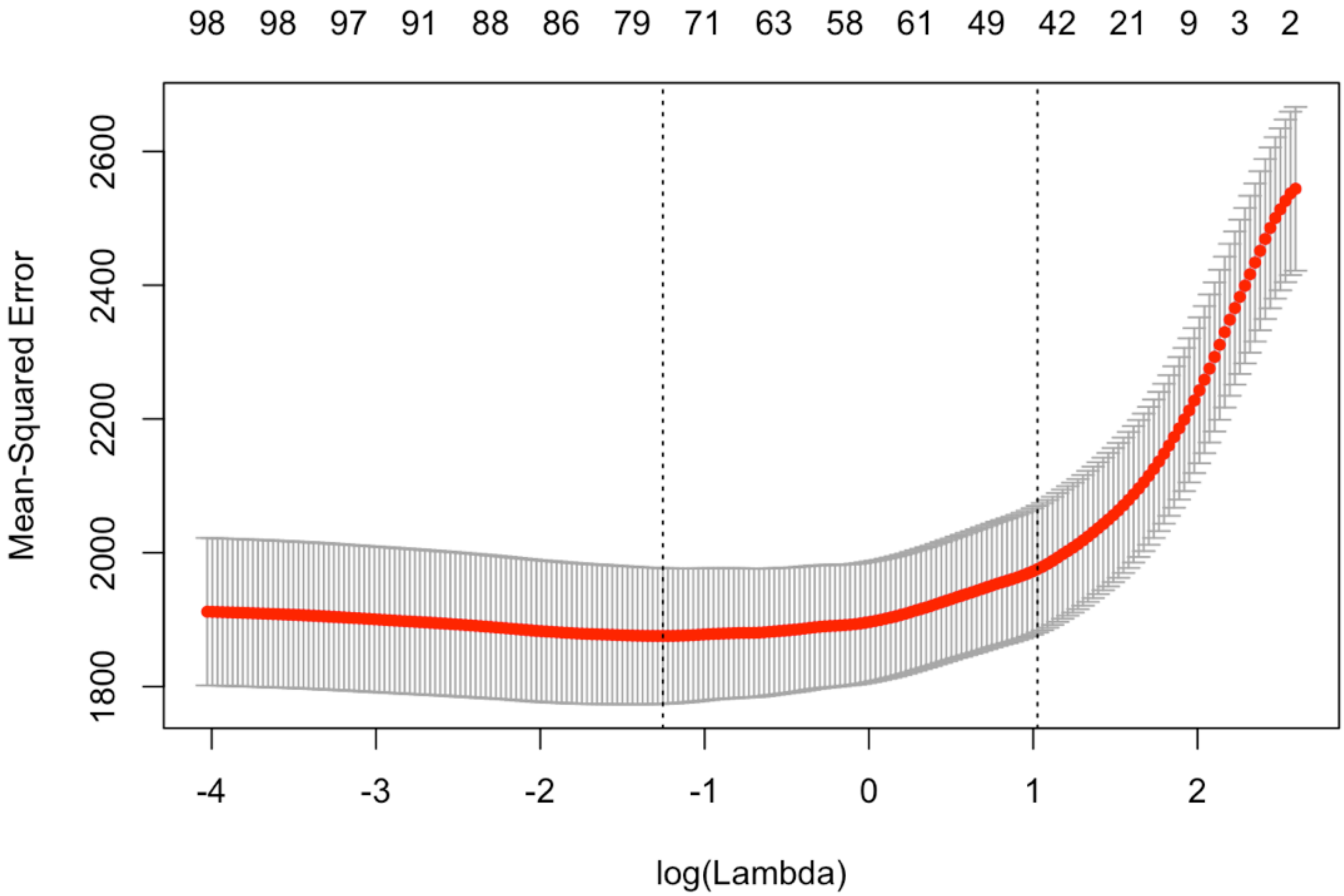
```
lasso1.model = cv.glmnet(as.matrix(as.matrix(data[,-c(117,118)])),
                         data$V117, family="gaussian", nlambda = 300, alpha=1.0)
lasso2.model = cv.glmnet(as.matrix(as.matrix(data[,-c(117,118)])),
                         data$V118, family="gaussian", nlambda = 300, alpha=1.0)

idx1 = which(lasso1.model$cvm == min(lasso1.model$cvm))
idx2 = which(lasso2.model$cvm == min(lasso2.model$cvm))
```

Plots

```
plotres(lasso1.model, info=TRUE, which = 1, caption="Lasso (L1) Lat.")
```




```
plotres(lasso2.model, info=TRUE, which = 1, caption="Lasso (L1) Long.")
```



Best regularization values

```
print(paste("L1 Lamba Lat. ", lasso1.model$lambda[idx1]))
```

```
## [1] "L1 Lamba Lat.  0.399184925618332"
```

```
print(paste("L1 Lamba Long. ", lasso2.model$lambda[idx2]))
```

```
## [1] "L1 Lamba Long.  0.285366470061178"
```

```
cv_rslt_table2 = data.frame(Items=c("Unregularized"),
                           CVErrLat = c(noreg1.model$cvm[2]),
                           CVErrLong = c(noreg2.model$cvm[2]),
                           LambdaLat = c(NA),
                           LambdaLong = c(NA),
                           ParamLat = c(noreg1.model$nzero[2]),
                           ParamLong = c(noreg2.model$nzero[2]))


cv_rslt_table2 = rbind(cv_rslt_table2,
                     data.frame(Items=c("Lasso"),
                                CVErrLat = c(lasso1.model$cvm[idx1]),
                                CVErrLong = c(lasso2.model$cvm[idx2]),
                                LambdaLat = c(lasso1.model$lambda[idx1]),
                                LambdaLong = c(lasso2.model$lambda[idx2]),
                                ParamLat = c(lasso1.model$nzero[idx1]),
                                ParamLong = c(lasso2.model$nzero[idx2])))
```

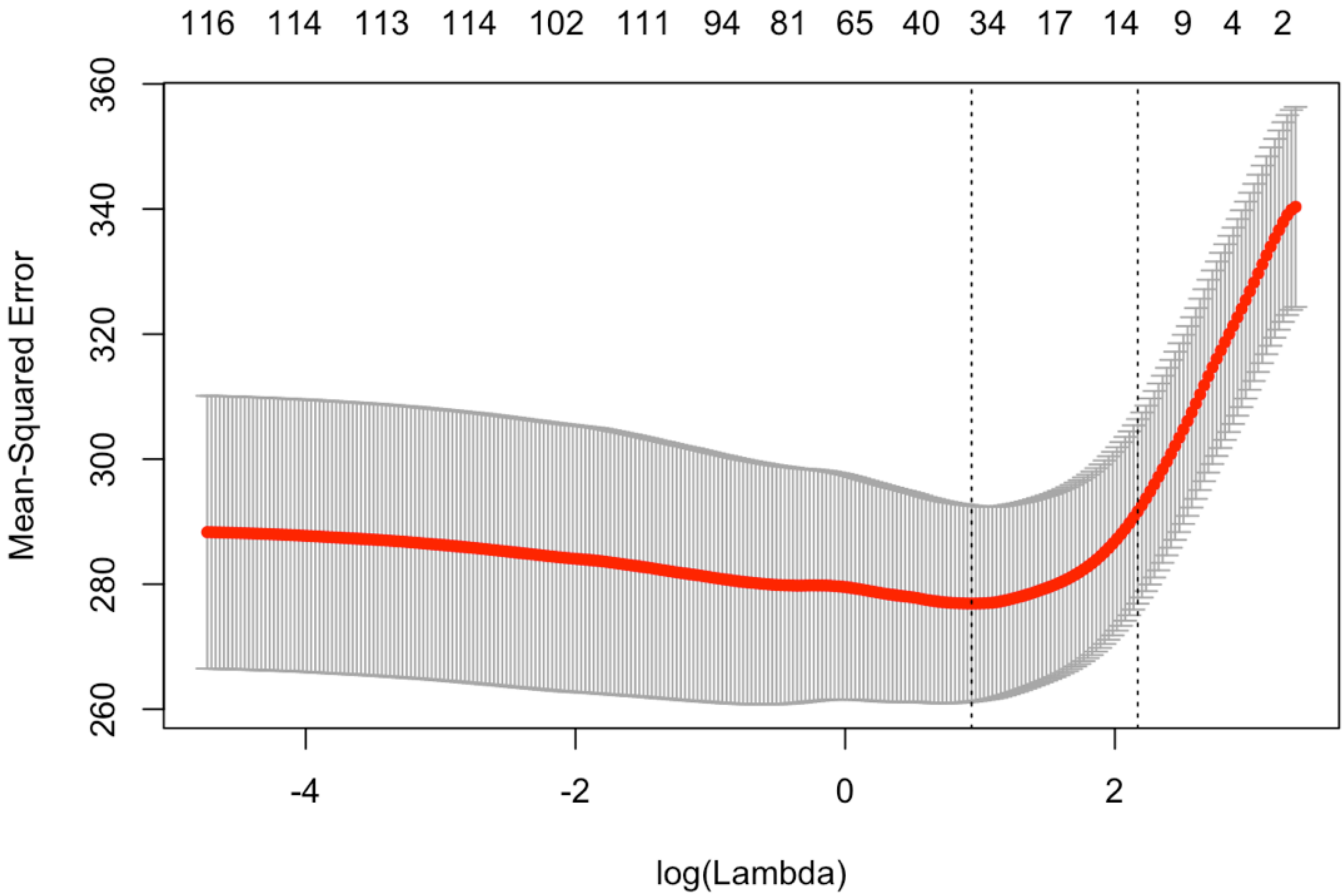
```
knitr::kable(cv_rslt_table2, format="markdown", digits=6, align=c('c','c','c','c','c'
,'c','c'), padding=20, caption="CVErr, Lambda, Param Comparison")
```

	Items	CVErrLat	CVErrLong	LambdaLat	LambdaLong	ParamLat	ParamLong
s1	Unregularized	289.7296	1929.891	NA	NA	116	116
s86	Lasso	277.3477	1875.666	0.399185	0.285366	24	77

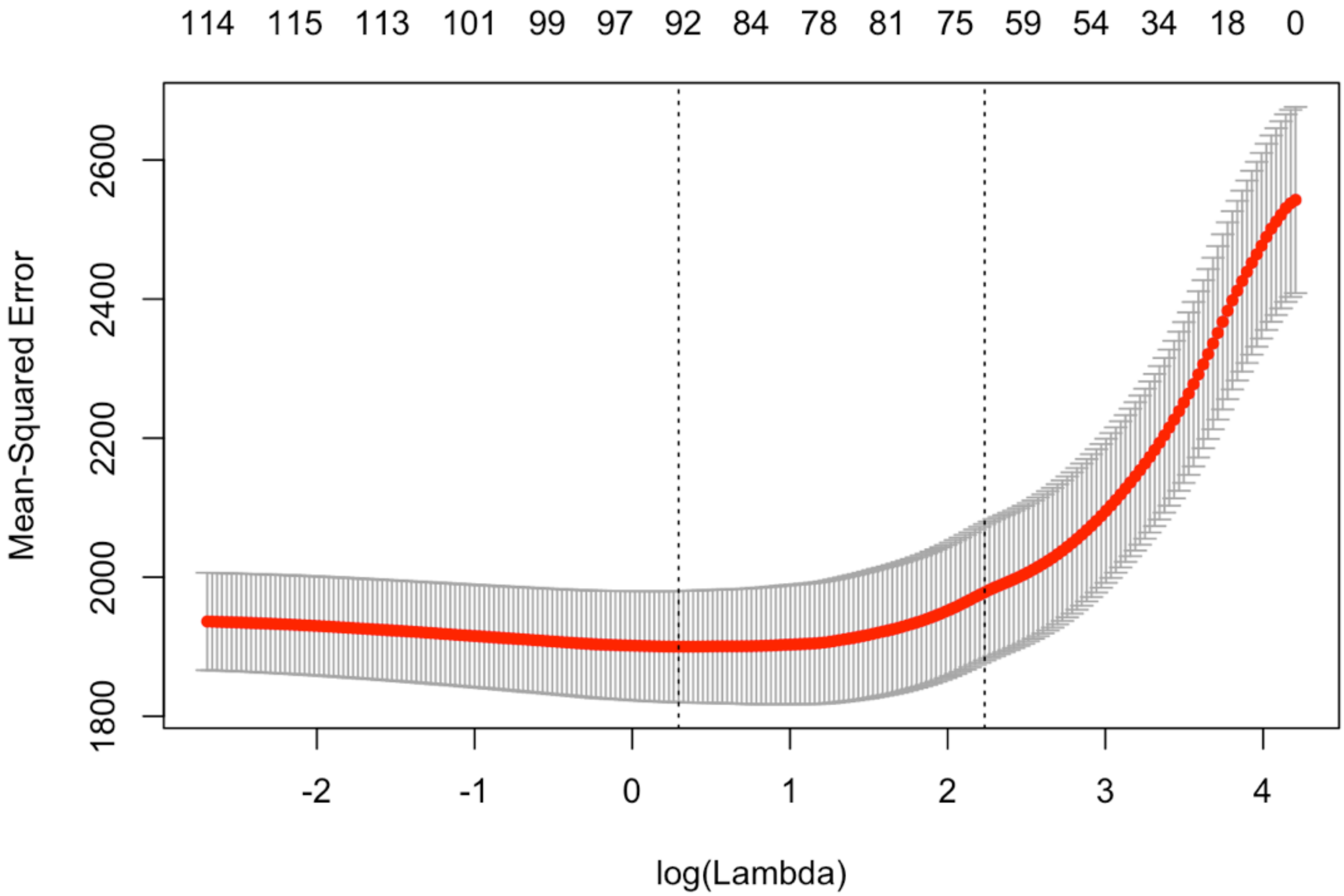
> Lasso (L1) Model is better than Unregularized version because it has lower CV Error.

ElasticNet

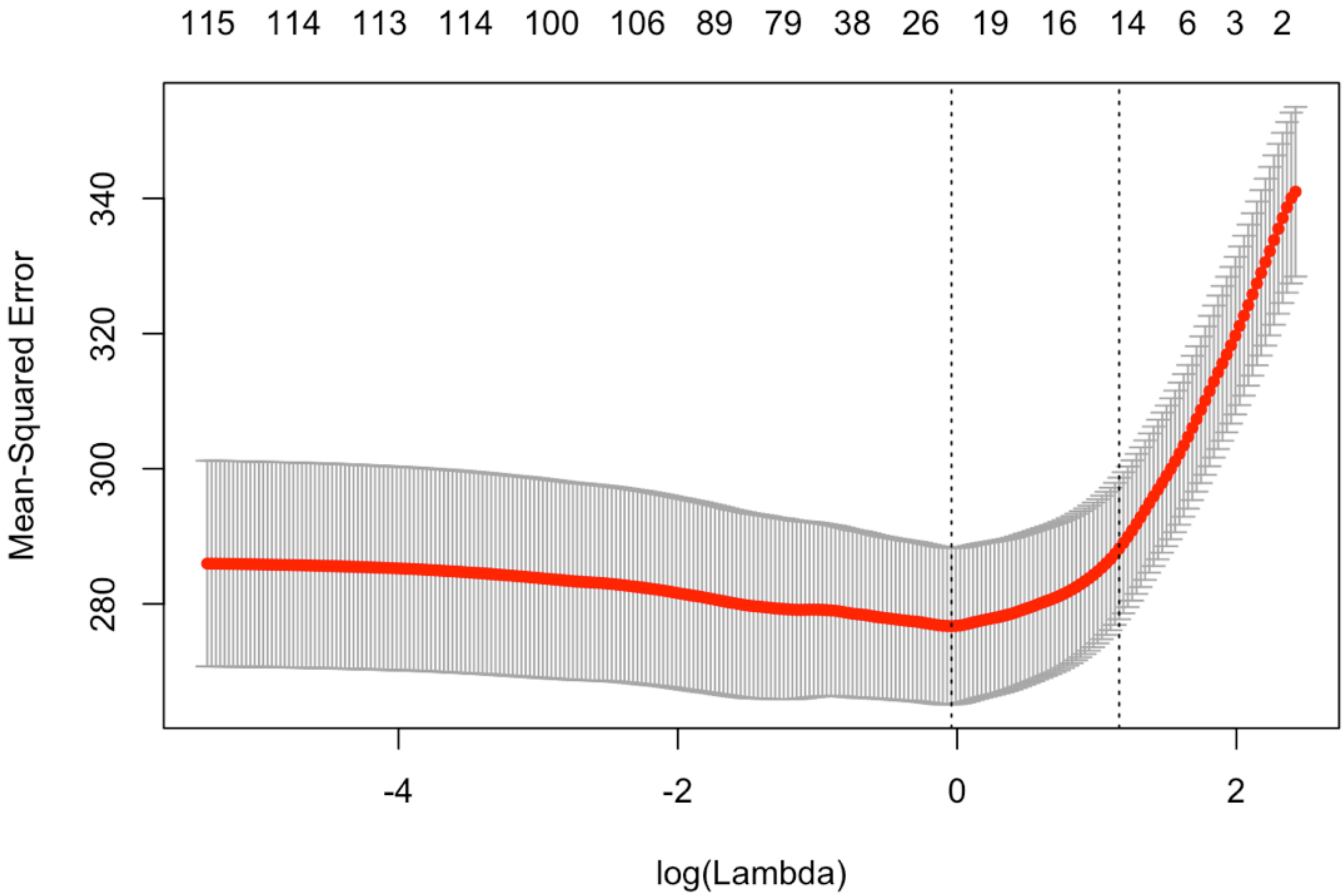
```r
elnetlat_tbl = data.frame(Items=c("Unregularized"),
                            CVErrLat = c(noreg1.model$cvm[2]),
                            CVErrLong = c(noreg2.model$cvm[2]),
                            LambdaLat = c(NA),
                            LambdaLong = c(NA),
                            ParamLat = c(noreg1.model$nzero[2]),
                            ParamLong = c(noreg2.model$nzero[2]))

for (l in c(0.2,0.5,0.8)) {
  elnet1.model = cv.glmnet(as.matrix(as.matrix(data[,-c(117,118)])), data$V117,
                           family="gaussian", nlambda = 300, alpha=l)
  idx1 = which(elnet1.model$cvm == min(elnet1.model$cvm))
  elnet2.model = cv.glmnet(as.matrix(as.matrix(data[,-c(117,118)])), data$V118,
                           family="gaussian", nlambda = 300, alpha=l)
  idx2 = which(elnet2.model$cvm == min(elnet2.model$cvm))
  elnetlat_tbl = rbind(elnetlat_tbl,
                       data.frame(Items=c(paste("ElasticNet ", l)),
                                  CVErrLat = c(elnet1.model$cvm[idx1]),
                                  CVErrLong = c(elnet2.model$cvm[idx2]),
                                  LambdaLat = c(elnet1.model$lambda[idx1]),
                                  LambdaLong = c(elnet2.model$lambda[idx2]),
                                  ParamLat = c(elnet1.model$nzero[idx1]),
                                  ParamLong = c(elnet2.model$nzero[idx2])))

  ###  part(a) Plots, Best regularization values
  plotres(elnet1.model, info=TRUE, which = 1, caption=paste("Alpha = ",l))
  print(paste("L1 Lamba Lat. ", elnet1.model$lambda[idx1]))

  plotres(elnet2.model, info=TRUE, which = 1,caption=paste("Alpha = ",l))
  print(paste("L1 Lamba Long. ", elnet2.model$lambda[idx2]))

}
```



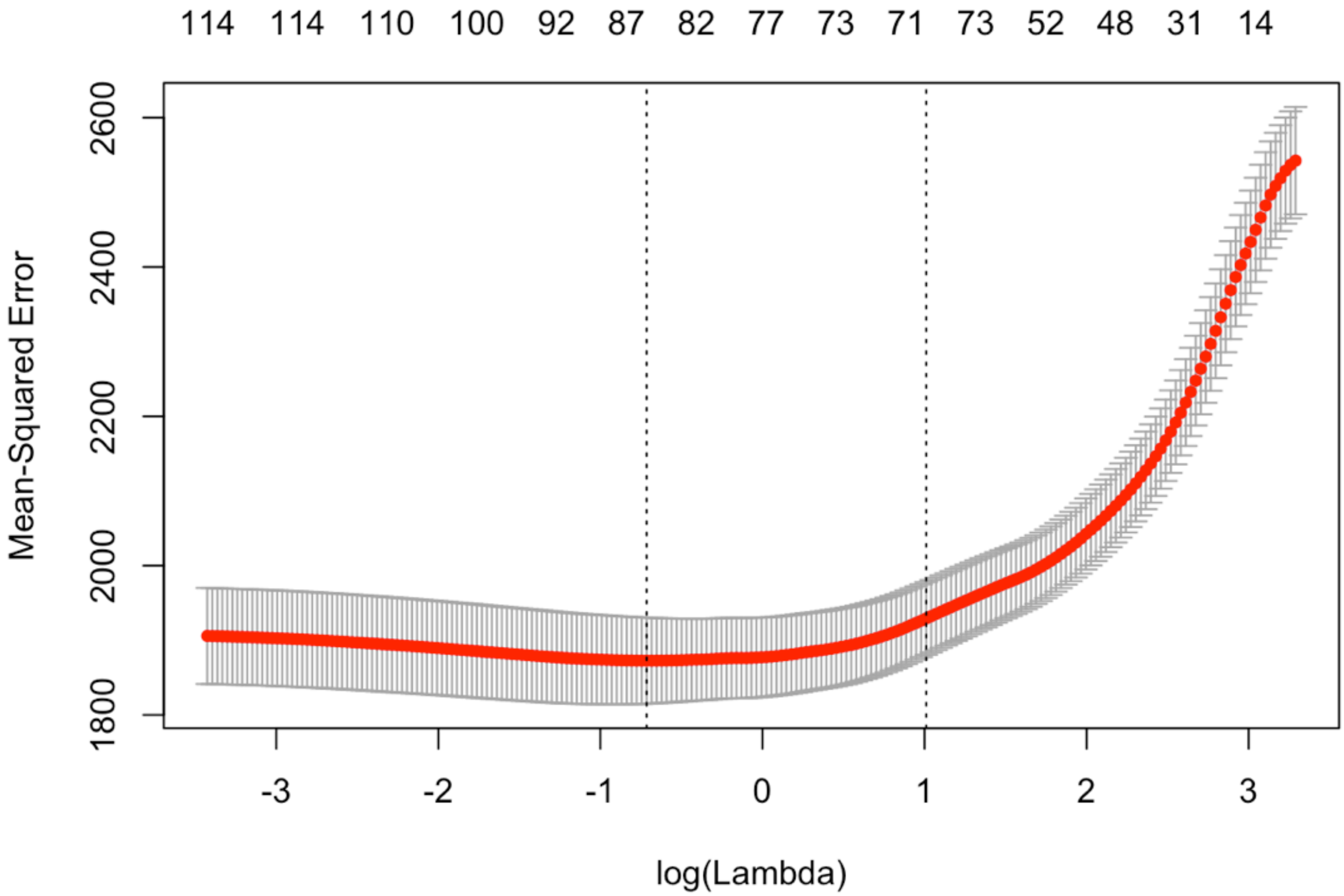

```
## [1] "L1 Lamba Lat.  2.55368631071419"
```



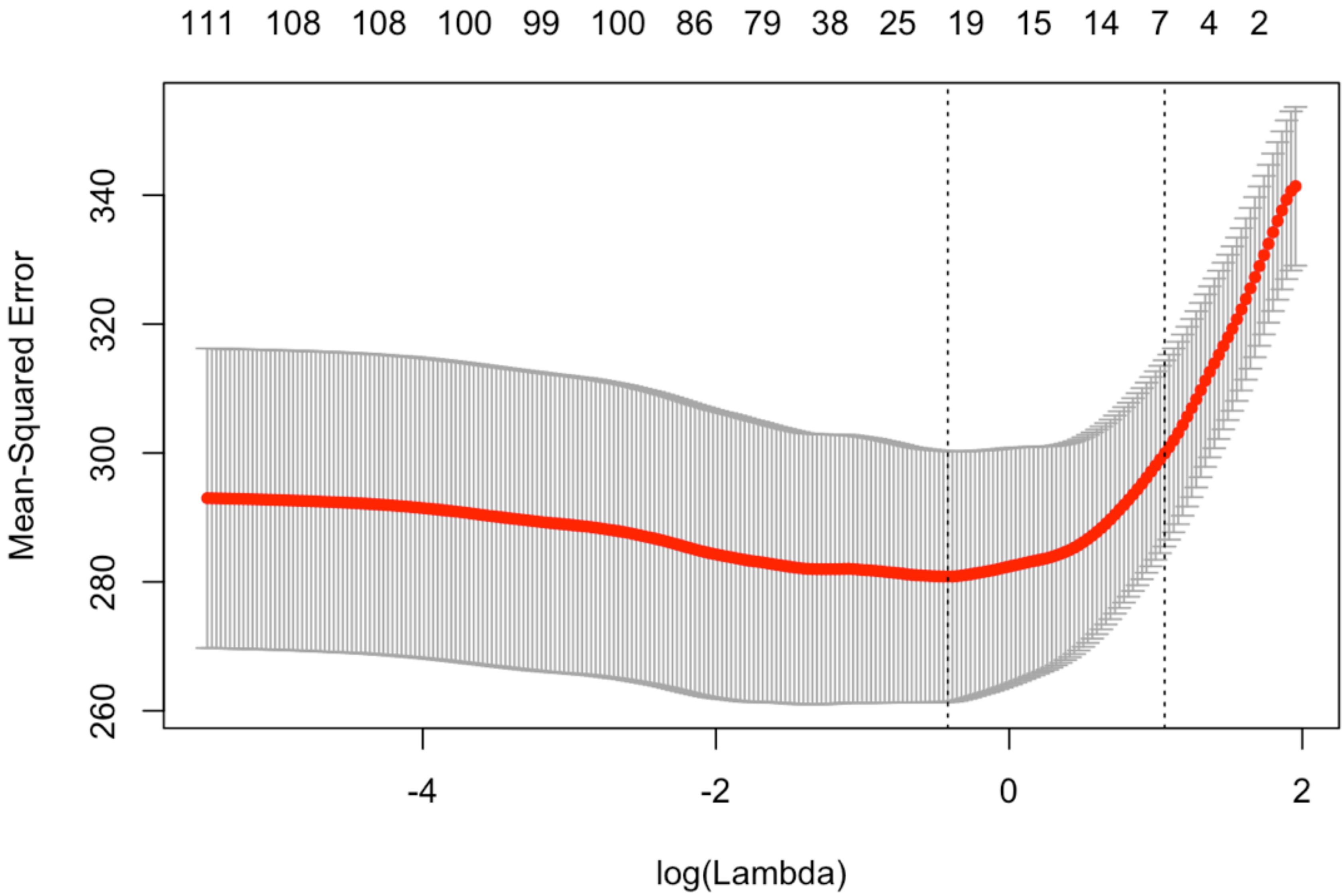

```
## [1] "L1 Lamba Long.  1.34158160429683"
```



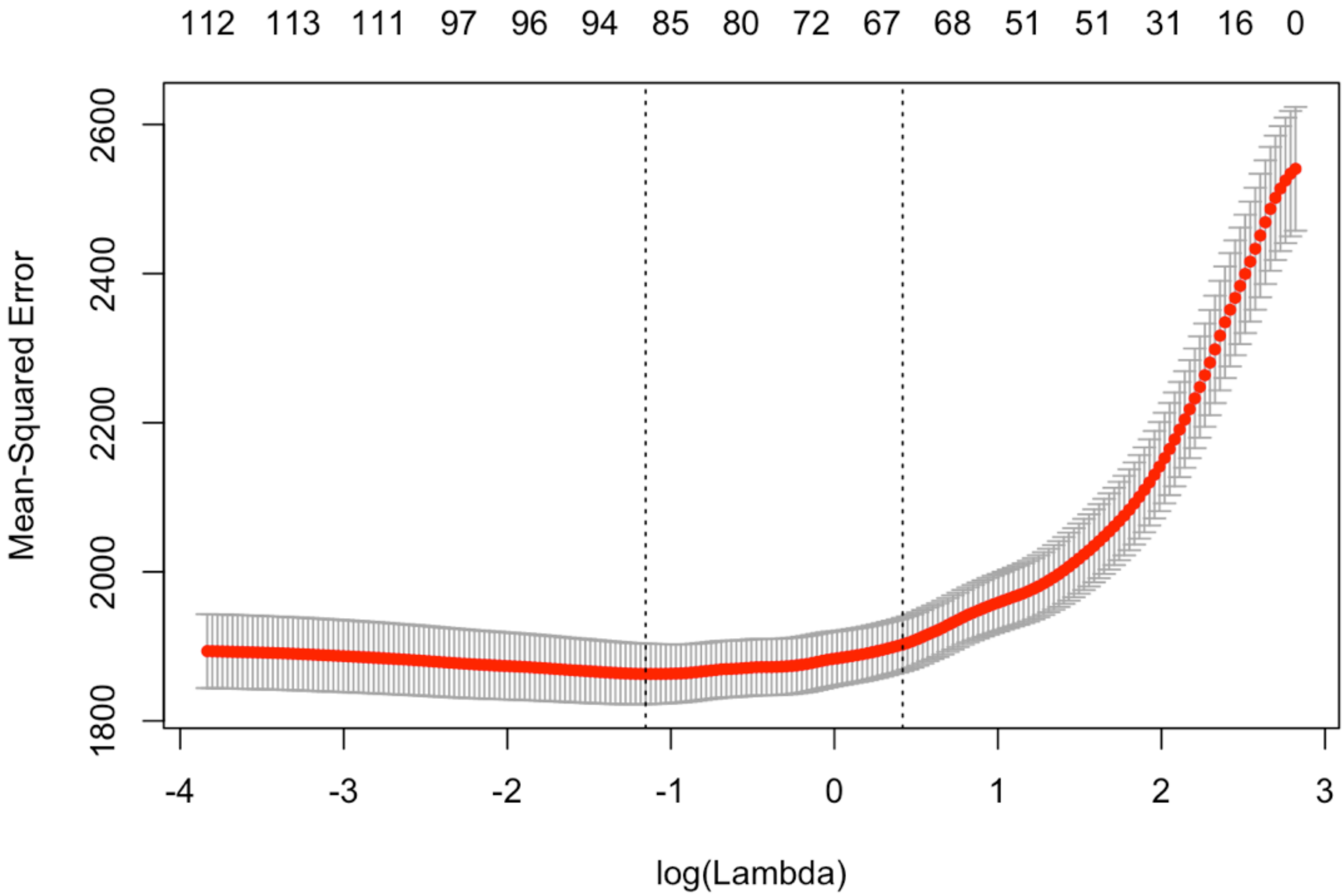

```
## [1] "L1 Lamba Lat.  0.960443201856009"
```




```
## [1] "L1 Lamba Long.  0.48926405079683"
```




```
## [1] "L1 Lamba Lat.  0.658393422469656"
```




```
## [1] "L1 Lamba Long.  0.315356110443145"
```

```
knitr::kable(elnetlat_tbl, format="markdown", digits=6, align=c('c','c','c','c','c','c','c'), padding=20, caption="CVErr, Lambda, Param Comparison")
```

	Items	CVErrLat	CVErrLong	LambdaLat	LambdaLong	ParamLat	ParamLong
s1	Unregularized	289.7296	1929.891	NA	NA	116	116
s78	ElasticNet 0.2	276.8958	1899.993	2.553686	1.341582	34	92
s80	ElasticNet 0.5	276.7314	1872.617	0.960443	0.489264	22	86
s77	ElasticNet 0.8	280.8278	1863.131	0.658393	0.315356	21	88

> ElasticNet with alpha of 0.5 produced smallest CV Error for Latitude. However for longitude parameter, the smallest CV Error is produced by alpha of 0.8. The values are also the best values amongs all models compared in this exercise.

2. Logistic regression

```
options(scipen = 0)
rm(list = ls())
library(stats)
library(ggplot2)
library(gdata)
library(caret)
library(caTools)
library(glmnet)
library(plotmo)
library(ggplot2)
set.seed(2018)
```

```
webLink = "http://archive.ics.uci.edu/ml/machine-learning-databases/00350/default%20o
f%20credit%20card%20clients.xls"
localLink = "~/cs498aml/default of credit card clients.xls"
data2 = read.xls(localLink, header = TRUE, sheet = 1)
data2[,3] = as.factor(data2[,3]) #gender
data2[,4] = as.factor(data2[,4]) #education
data2[,5] = as.factor(data2[,5]) # marriage
data = data2[,-c(1)] #index column adds no value, ditch it
idx = createDataPartition(y=data$Y, p = 0.8, list=FALSE)
trainData = data[idx,]
testData = data[-idx,]
```

General Regularized Model

Plot Residual plot for generalized linear model

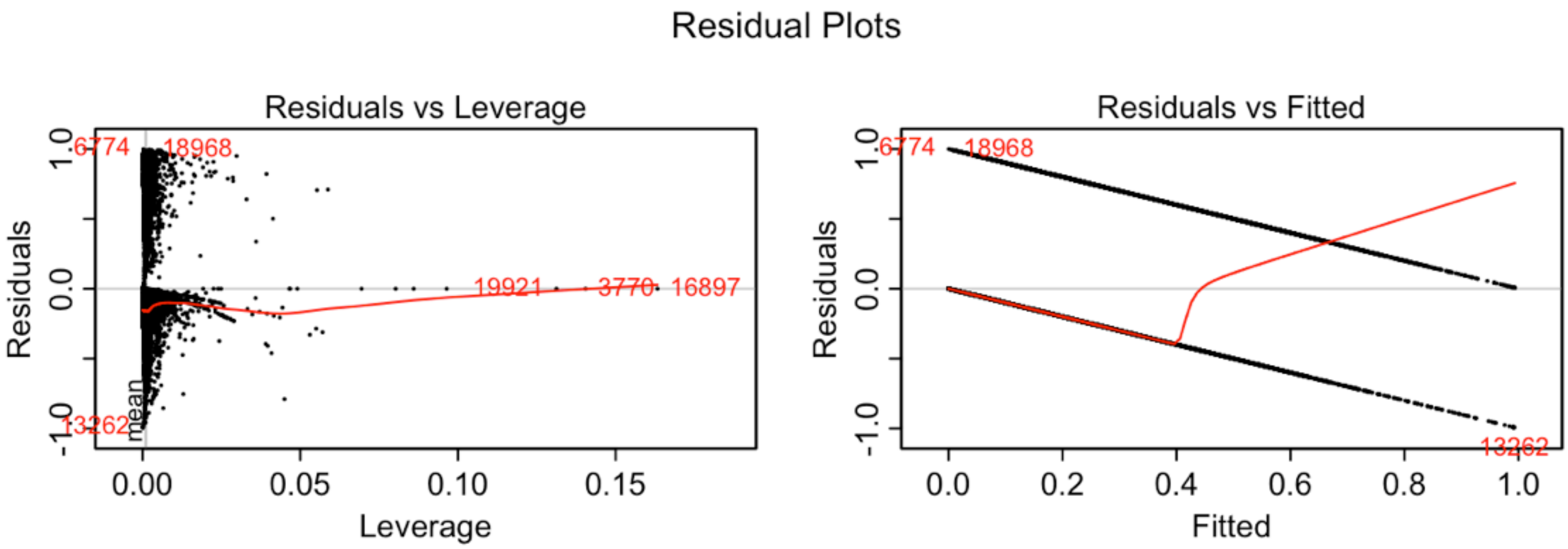
```
#glm.model = glm(data.matrix(data[,-c(24)]), as.factor(data$Y), family="binomial")

glm.model = glm(Y ~ ., data = data, family="binomial")
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
p = c(1,nrow(data))
for (i in 1:length(glm.model$fitted.values)) {
  if (glm.model$fitted.values[i] < 0.5) {
    p[i] = 0
  } else {
      p[i] = 1}
}
print(paste("GLM Accuracy =", mean(p == data$Y)))
```

```
## [1] "GLM Accuracy = 0.811233333333333"
```

```
plotres(glm.model, which = c(1,3), npoints=40000, caption="Residual Plots")
```



UnRegularized Model

```
unreg.model = cv.glmnet(data.matrix(trainData[,-c(24)]), as.factor(trainData$Y), fami
ly="binomial",
                        type.measure="class", lambda=c(0,1e-7), alpha=0)
yhat0 = predict(unreg.model, s = unreg.model$lambda[2],
                type="class", newx = data.matrix(testData[,-24]))

print(paste("Unreg. Model Accuracy =",  mean(as.factor(yhat0) == testData$Y) ))
```

```
## [1] "Unreg. Model Accuracy = 0.813666666666667"
```

Identify Optimal regularization values

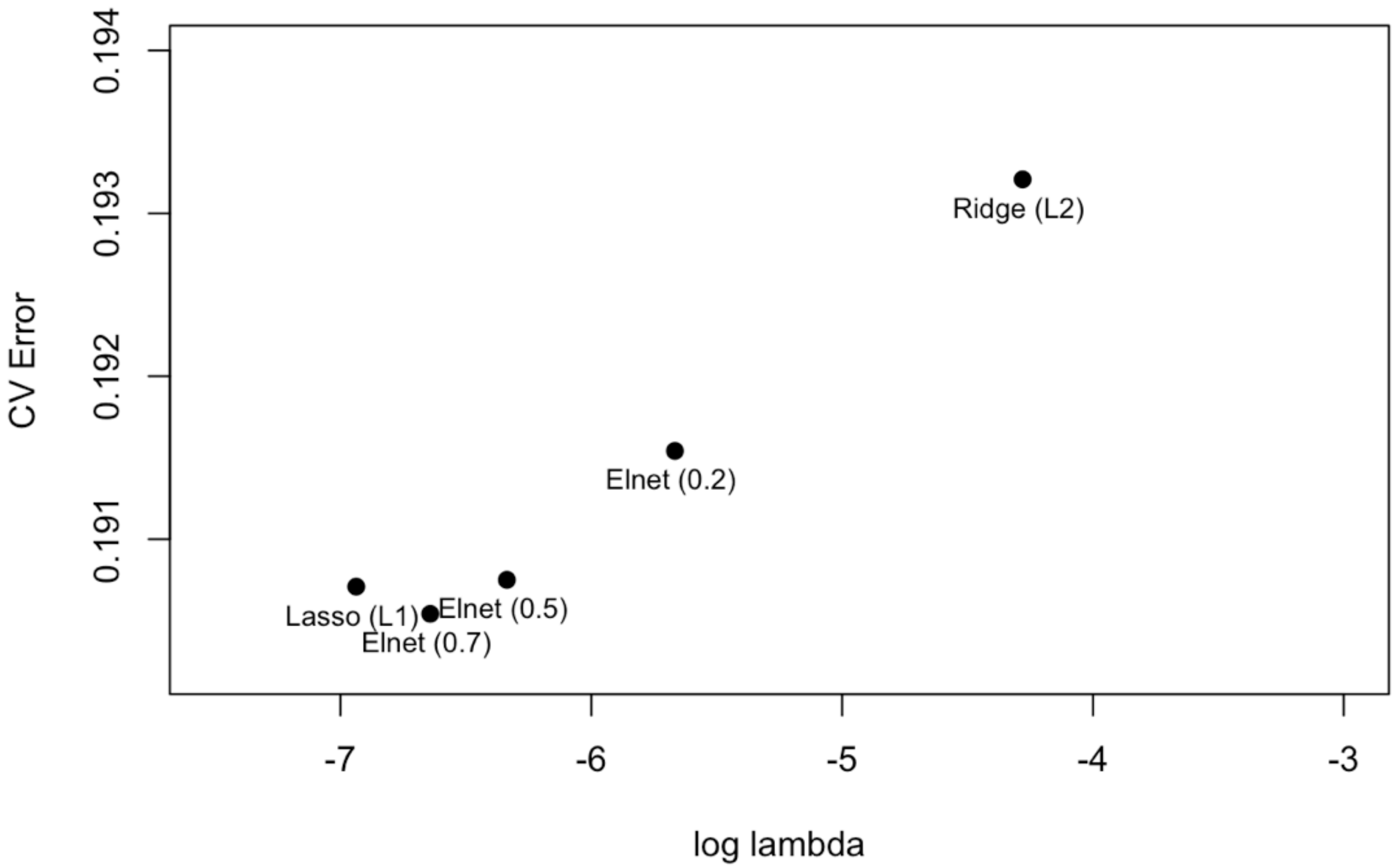
L1,L2 and three Alphas in Elastic Net Regression

```
lasso.model = cv.glmnet(data.matrix(trainData[,-c(24)]), as.factor(trainData$Y), fami
ly="binomial",
                        type.measure="class", nlambda = 300, alpha=1.0)
lasso.minIdx = which.min(lasso.model$cvm)
ridge.model = cv.glmnet(data.matrix(trainData[,-c(24)]), as.factor(trainData$Y), fami
ly="binomial",
                        type.measure="class", nlambda = 300, alpha=0)
ridge.minIdx = which.min(ridge.model$cvm)
for (l in c(0.2,0.5,0.7)) {
  assign(paste("elnet.model", l*10, sep=""), cv.glmnet(data.matrix(trainData[,-c(24)]
), as.factor(trainData$Y),
                                                           family="binomial", type.measur
e="class", nlambda = 300, alpha=l))
}

elnet.model2.minIdx = which.min(elnet.model2$cvm)
elnet.model5.minIdx = which.min(elnet.model5$cvm)
elnet.model7.minIdx = which.min(elnet.model7$cvm)
```

```
result_table = data.frame(Model=c("Unregularized "),
                       NumParams = c(as.vector(unreg.model$nzero[2])),
                       Lambda= c(NA),
                       CrossVal.Err = c(unreg.model$cvm[2]))

result_table = rbind(result_table,
                      data.frame(Model=c("Lasso (L1) "),
                       NumParams = c(as.vector(lasso.model$nzero[lasso.minIdx])),
                       Lambda = c(lasso.model$lambda[lasso.minIdx]),
                       CrossVal.Err = c(lasso.model$cvm[lasso.minIdx])))

result_table = rbind(result_table,
                      data.frame(Model=c("Ridge (L2) "),
                       NumParams = c(as.vector(ridge.model$nzero[ridge.minIdx])),
                       Lambda = c(ridge.model$lambda[ridge.minIdx]),
                       CrossVal.Err = c(ridge.model$cvm[ridge.minIdx])))

result_table = rbind(result_table,
                      data.frame(Model=c("Elnet (0.2) "),
                       NumParams = c(as.vector(elnet.model2$nzero[elnet.model2.minIdx
])),
                       Lambda = c(elnet.model2$lambda[elnet.model2.minIdx]),
                       CrossVal.Err = c(elnet.model2$cvm[elnet.model2.minIdx])))

result_table = rbind(result_table,
                      data.frame(Model=c("Elnet (0.5) "),
                       NumParams = c(as.vector(elnet.model5$nzero[elnet.model5.minIdx
])),
                       Lambda = c(elnet.model5$lambda[elnet.model5.minIdx]),
                       CrossVal.Err = c(elnet.model5$cvm[elnet.model5.minIdx])))

result_table = rbind(result_table,
                      data.frame(Model=c("Elnet (0.7) "),
                       NumParams = c(as.vector(elnet.model7$nzero[elnet.model7.minIdx
])),
                       Lambda = c(elnet.model7$lambda[elnet.model7.minIdx]),
                       CrossVal.Err = c(elnet.model7$cvm[elnet.model7.minIdx])))
```

CV Error vs Log Lambda plot

```
plot(log(result_table$Lambda[2:6]),
         result_table$CrossVal.Err[2:6], type="p",
         ylim=c(0.1902,0.1940), xlim=c(-7.5,-3), pch=19,
         xlab="log lambda", ylab="CV Error")
text(log(result_table$Lambda[2:6]),
     result_table$CrossVal.Err[2:6],
     labels = result_table$Model[2:6],
     cex =0.8,
     pos=1)
```



Model Comparison table

```
knitr::kable(result_table, format="markdown", digits=6, align=c('c','c','c','c'), pad
ding=20, caption="Model Comparison")
```

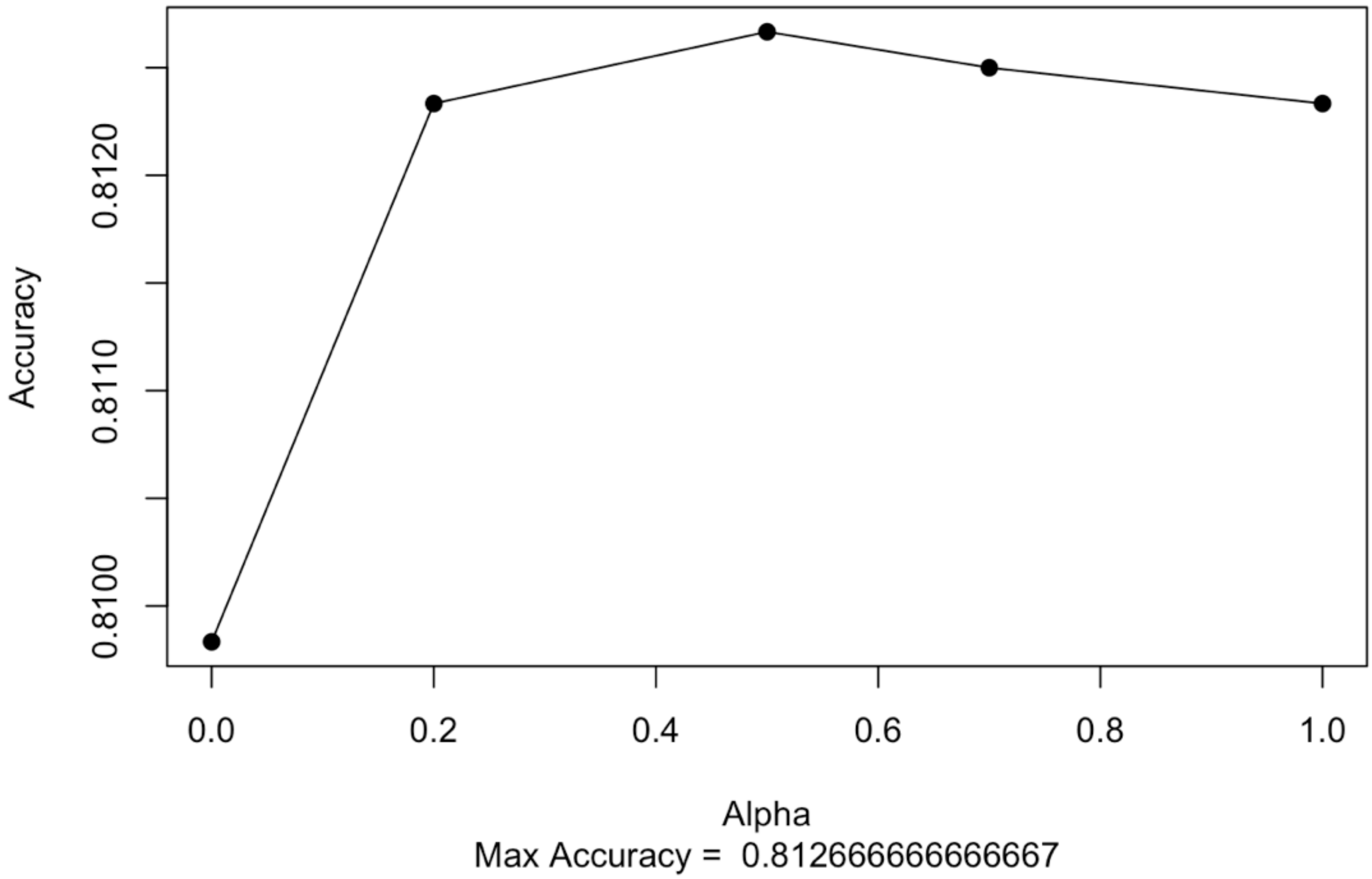
Model	NumParams	Lambda	CrossVal.Err
Unregularized	23	NA	0.189625
Lasso (L1)	20	0.000971	0.190708
Ridge (L2)	23	0.013838	0.193208

Elnet (0.2)	21	0.003460	0.191542
Elnet (0.5)	20	0.001771	0.190750
Elnet (0.7)	20	0.001304	0.190542

> The best Cross validated error value among regularized models is obtained through elnet model with alpha value of 0.7.

Predict values and calculate accuracy

```
yhat1 = predict(lasso.model, s = lasso.model$lambda[lasso.minIdx],
                type="class", newx = data.matrix(testData[,-24]))

yhat2 = predict(ridge.model, s = ridge.model$lambda[ridge.minIdx],
                type="class", newx = data.matrix(testData[,-24]))

yhat3 = predict(elnet.model2, s = elnet.model2$lambda[elnet.model2.minIdx],
                type="class", newx = data.matrix(testData[,-24]))

yhat4 = predict(elnet.model5, s = elnet.model5$lambda[elnet.model5.minIdx],
                type="class", newx = data.matrix(testData[,-24]))

yhat5 = predict(elnet.model7, s = elnet.model7$lambda[elnet.model7.minIdx],
                type="class", newx = data.matrix(testData[,-24]))


acc= c(1:5)

acc[1] = mean(as.factor(yhat2) == testData$Y)
acc[2] = mean(as.factor(yhat3) == testData$Y)
acc[3] = mean(as.factor(yhat4) == testData$Y)
acc[4] = mean(as.factor(yhat5) == testData$Y)
acc[5] = mean(as.factor(yhat1) == testData$Y)


plot(c(0,0.2,0.5,0.7,1.0), acc, type="o", xlab="Alpha", pch=19, ylab="Accuracy", sub=
paste("Max Accuracy = ",max(acc)))
```



Conclusion

> The best accuray amongst regularized model is obtained with elasticnet model (alpha = 0.5). The best overall accuracy is achieved with unregularized model.