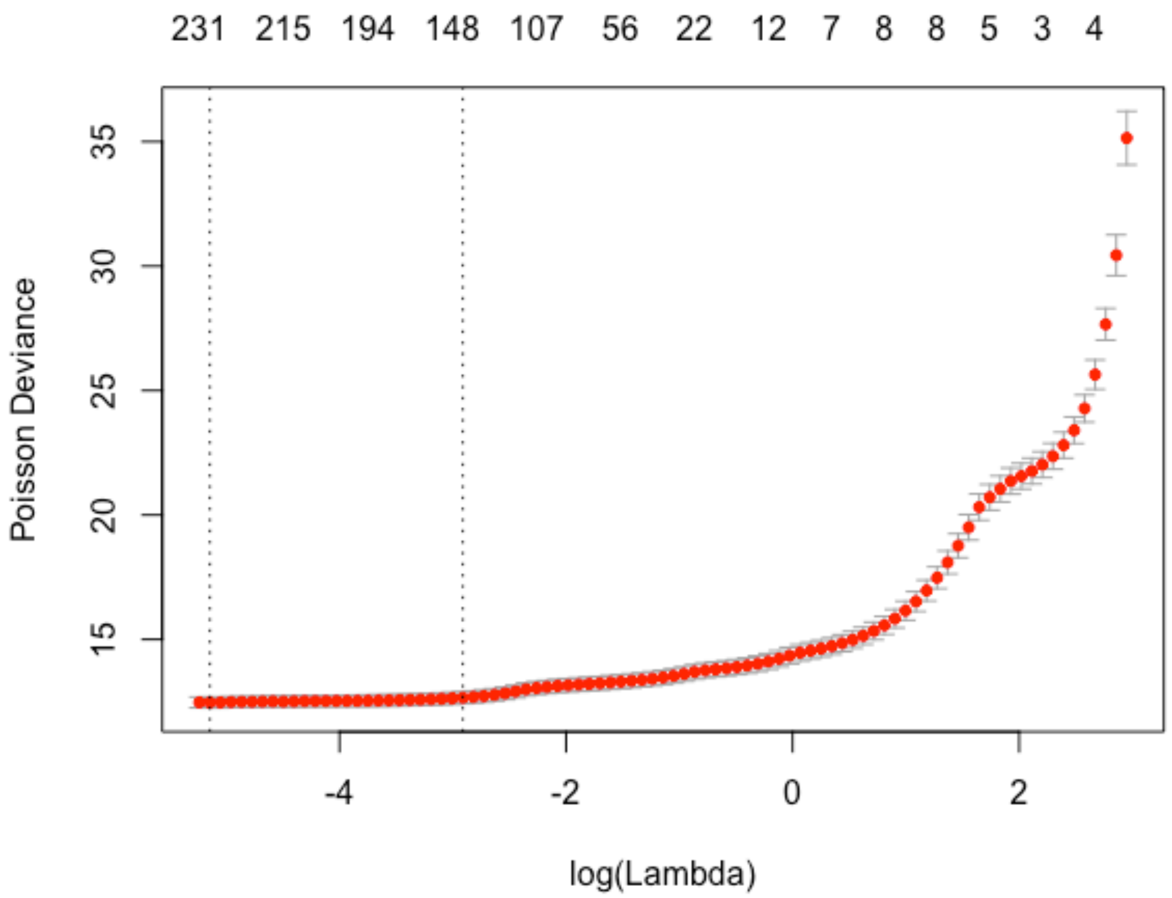
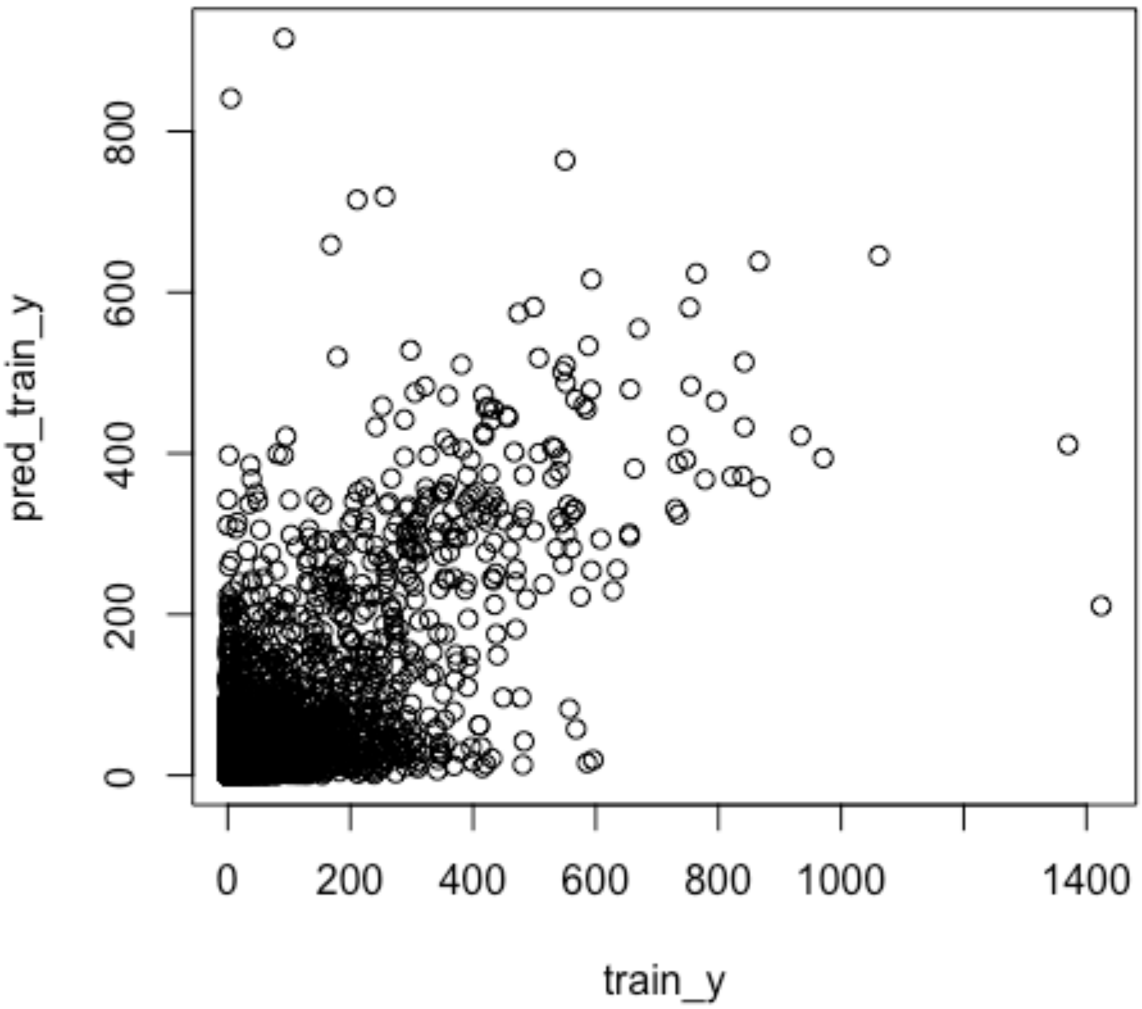
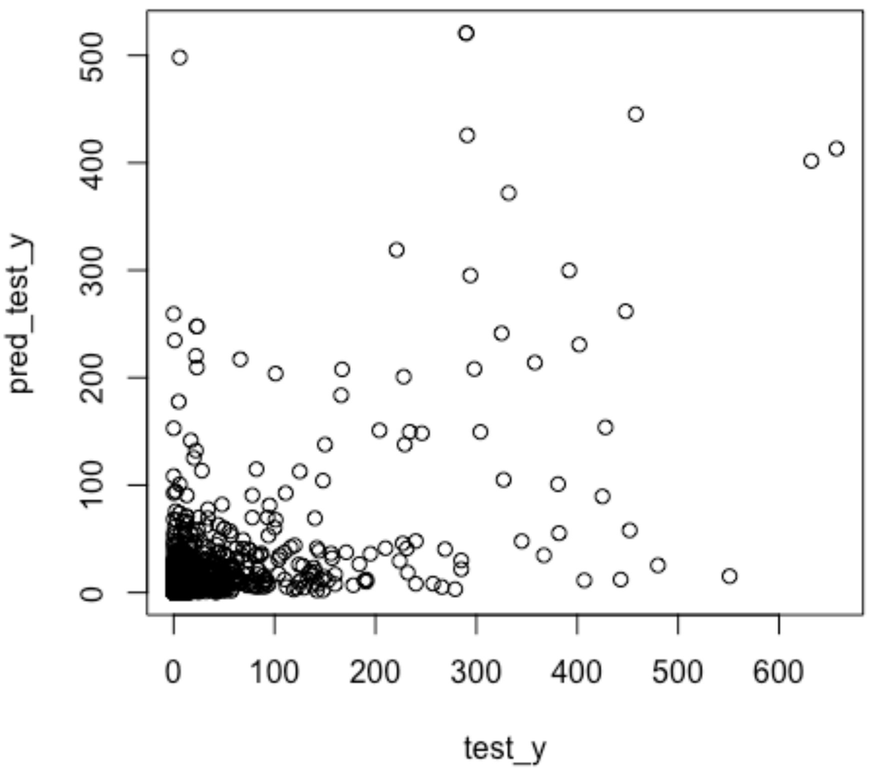
CS498AML_HW7_panz2



Regularization constant: **0.005832828** (lambda.min)



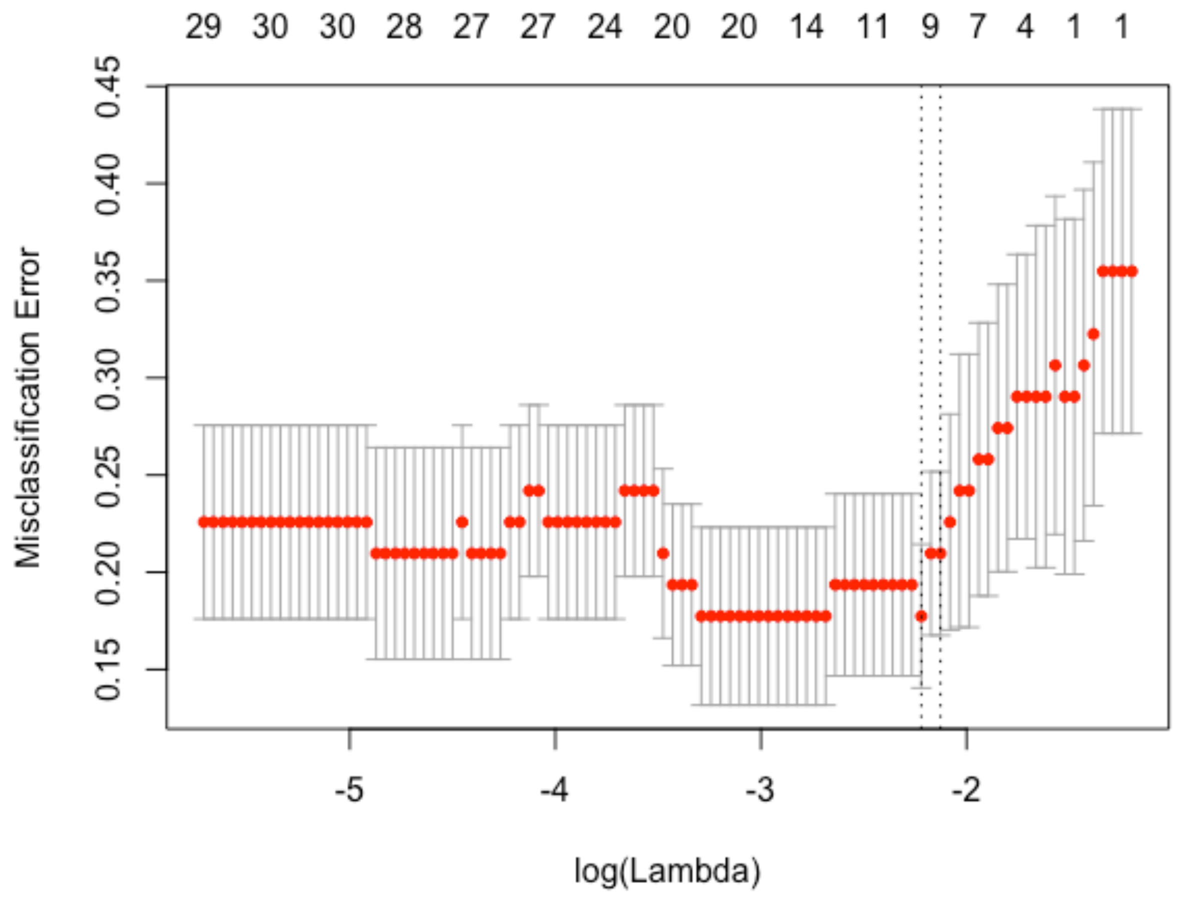
Regularization constant: **0.05439715** (lambda.1se)



Compared these two plot, we can see the plot of train data has trend to gather around y=x, whereas the plot of test data is more random. Two same thing are that most of the points in the plots reside in the lower left corner and there are a lot of outliers in these two plots.

Both two plots indicate the bad performance of this model. What we expected is that all the points distribute close to the diagonal line y=x, which means the true value is nearly equal to the predicted value, but in reality they are not. Besides, the high Poisson Deviance also indicate the bad performance of the model.

This regression is difficult because 1. Many instances are very similar (observations are not independent) and each feature only has several different values (a lot of rows has the same value for each column), which make it hard for lasso to drop dimension, so the model is still very complicated with too much features; 2. The independent variable doesn't have a linear relationship with the dependent variables; 3. The number of features is too large and the data matrix is too sparse, so more training instances is needed to generate a good model; 4. The independent variables are correlate, for example, the correlation coefficient between V1 and V2 is 0.9260629, between V1 and V4 is 0.852126.



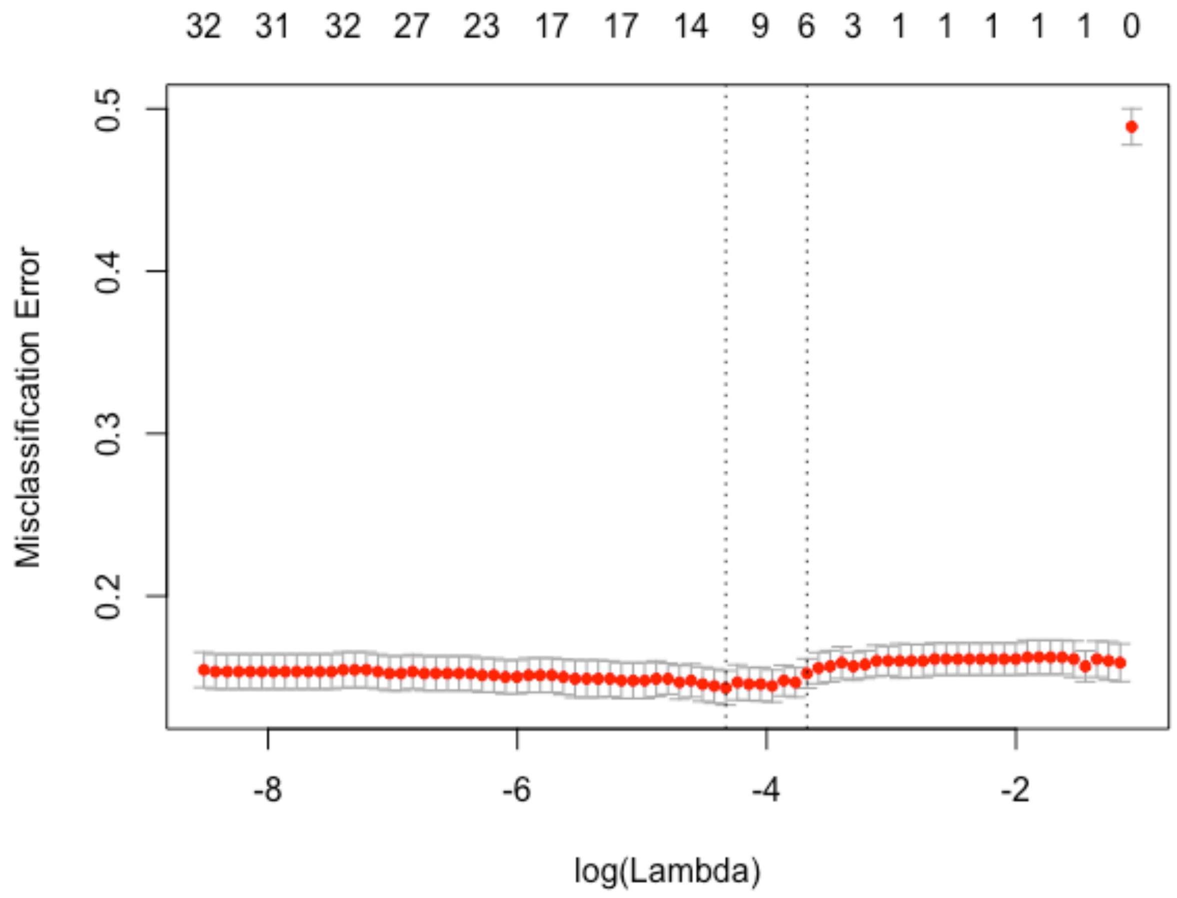
Regularization constant: **0.1191684** (lambda.1se)

Accuracy: **0.8870968**

Baseline: **0.6451613**

This model performs very well, its accuracy is very high (0.887) and misclassification error is low. The accuracy is much higher than the baseline of predicting the most common class (0.645).

Predict gender with the features:



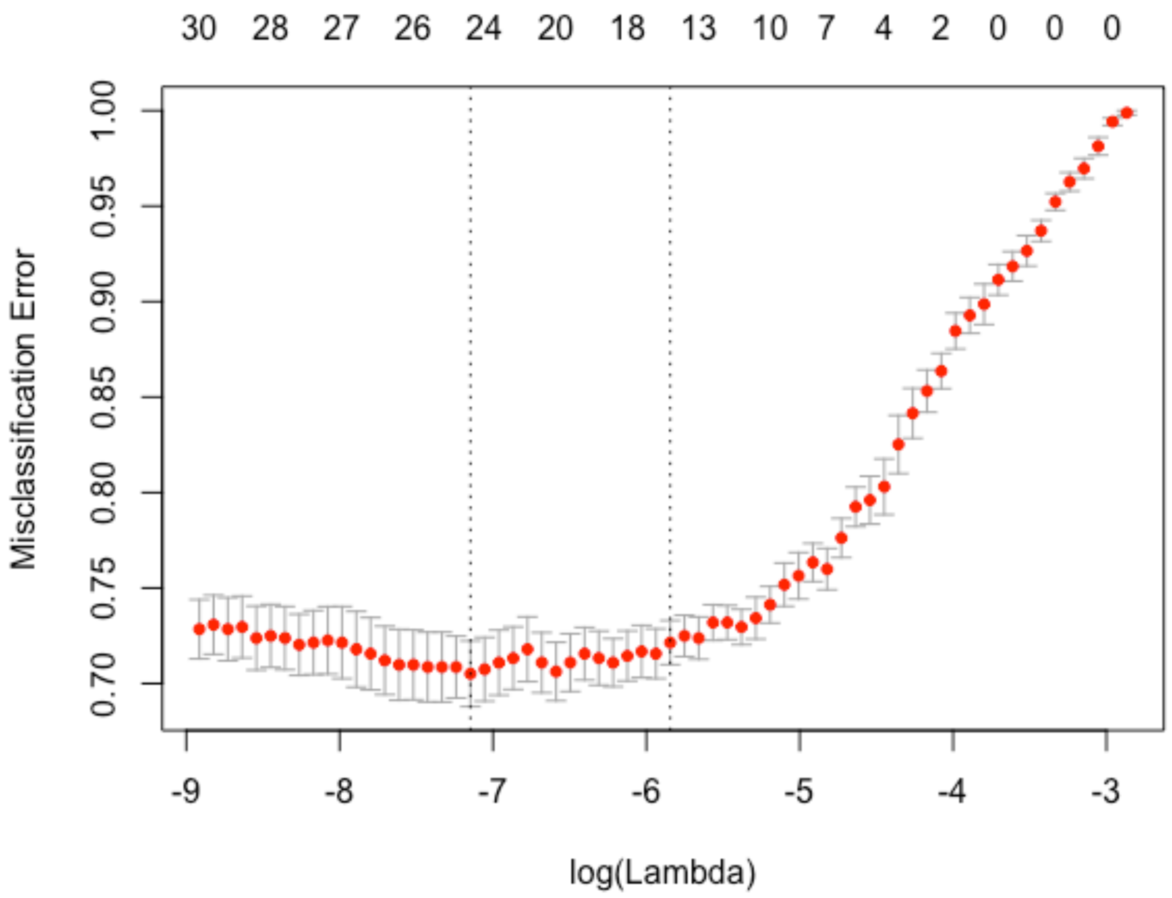
Regularization constant: **0.2843359** (lambda.1se)

Accuracy: **0.8543046**

Baseline: **0.50883**

This model performs very well, its accuracy is very high (0.854) and the misclassification error is very low, and it is much higher than the baseline of predicting the most common class (0.509).

Predict the strain of a mouse with the features:



Regularization constant: **0.002895419** (lambda.1se)

Accuracy: **0.5361305**

Baseline: **0.02255747 (**picks a class at random but follows the class frequency, so random baseline is calculated by adding up the squared probabilities of all the classes.**)**
If the chance for each class is equal, then the baseline is 1/45, **0.02222222.**

This model performs very well. Its accuracy is very high (0.89) compared to the extremely low baseline of predicting a class at random (**0.02255747 or 0.02222222**)

```r
library(glmnet)
#12.3
setwd('/Users/panzhang/Desktop/CS498AML/HW7/BlogFeedback/')
#(a)
train <- read.csv("blogData_train.csv", header=F, sep=",")
train_x <- as.matrix(train[, 1 : ncol(train) - 1])
train_y <- as.matrix(train[,ncol(train)])
print(dim(train_x))
print(dim(train_y))
fit12.3 = cv.glmnet(train_x, train_y, family = "poisson", alpha = 1)
plot(fit12.3)
#(b)
#predict using lambda.min
fit12.3$lambda.min
pred_train_y = predict(fit12.3, train_x, s=fit12.3$lambda.min, type = "response")
plot(train_y,pred_train_y)
#(c)
#combine all the test data
test_name=Sys.glob(file.path("/Users/panzhang/Desktop/CS498AML/HW7/BlogFeedback/","blogData_test*.csv"))
merge.data = read.csv(file = test_name[1],header=F,sep=",")
n=length(test_name)
for (i in 2:n){
  new.data = read.csv(file = test_name[i], header=F, sep=",")
  merge.data = rbind(merge.data,new.data)
}
#predict using lambda.1se
test_x <- as.matrix(merge.data[, 1 : ncol(merge.data) - 1])
test_y <- as.matrix(merge.data[, ncol(merge.data)])
fit12.3$lambda.1se
pred_test_y = predict(fit12.3, test_x, s=fit12.3$lambda.1se, type = "response")
plot(test_y,pred_test_y)

#12.4
setwd('/Users/panzhang/Desktop/CS498AML/HW7')
data <- read.csv("I2000.csv", header=FALSE, sep=",")
tissue <- read.csv("Tissues.csv", header=FALSE, sep=",")
data=as.matrix(t(data))
tissue=as.matrix(tissue/abs(tissue))
##baseline calculating
baseline_tissue=max(table(tissue))/dim(tissue)[1]
baseline_tissue
fit12.4 = cv.glmnet(data, tissue, family = "binomial", type.measure="class", alpha = 1)
plot(fit12.4)
#predict using lambda.1se
fit12.4$lambda.1se
pred_t=predict(fit12.4, data, type='class', s='lambda.1se')
accuracy_tissue= sum (pred_t== tissue )/dim(tissue)[1]
print(accuracy_tissue)

#12.5
crusio  <- read.csv("Crusio1.csv", header=TRUE, sep=",")
#(a)
sex_data=cbind(crusio$sex, crusio[,4:41])
dim(sex_data)
f_data=sex_data[complete.cases(sex_data),]
dim(f_data)
data_x=as.matrix(f_data[,-1])
data_y=as.matrix(f_data[,1])
#baseline calculating
baseline_sex=max(table(data_y))/dim(data_y)[1]
baseline_sex
fit12.5a = cv.glmnet(data_x, data_y, family = "binomial", type.measure="class", alpha = 1)
plot(fit12.5a)
#using lambda.1se to predict and calculate the accuracy
fit12.5a$lambda.1se
pred_sex=predict(fit12.5a, data_x, type='class', s='lambda.1se')
accuracy_sex= sum (pred_sex== data_y )/dim(data_y)[1]
print(accuracy_sex)
#(b)
strain_data = cbind(crusio$strain,crusio[,4:41])
s_data = strain_data[complete.cases(strain_data),]
dim(s_data)
#find the strain with a total count >=10
strain_table = as.data.frame(table(s_data[,1]))
strain_table = strain_table [which(strain_table$Freq>=10),]
y = as.vector(s_data[,1]) %in% as.vector(strain_table$Var1)
filt_data=s_data[y,]
dim(filt_data)
strain_x=as.matrix(filt_data[,-1])
strain_y=as.matrix(filt_data[,1])
#baseline calculating at rondom
ratio_table=table(strain_y)/sum(table(strain_y))
baseline_strain= sum(ratio_table^2)
baseline_strain
#run model
fit12.5b = cv.glmnet(strain_x, strain_y, family = "multinomial", type.measure="class", alpha = 1)
plot(fit12.5b)
fit12.5b$lambda.1se
pred_strain=predict(fit12.5b, strain_x, type='class', s='lambda.1se')
accuracy_strain= sum (pred_strain== strain_y )/dim(strain_y)[1]
print(accuracy_strain)
```