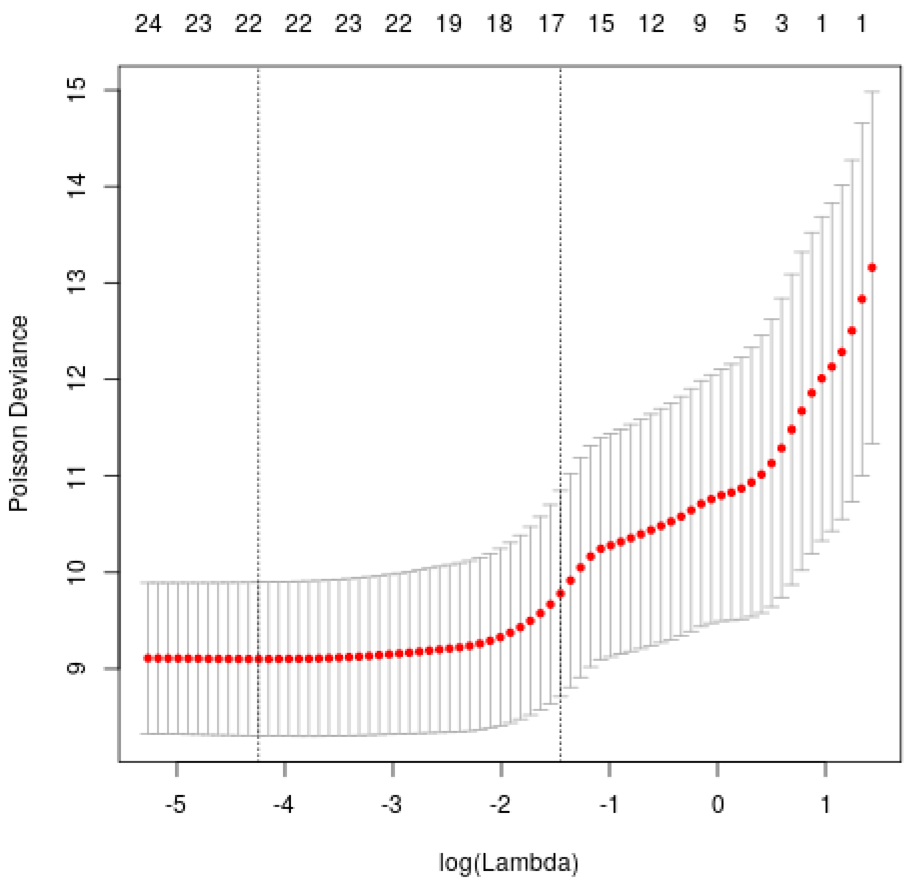
Homework 7

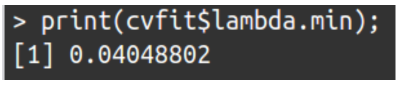
12.3 As the dataset is too large, we choose a subset of the training data according to Piazza.
We choose to draw each test.csv separately and choose four plots for this report.

1. **Show your plot of the cross-validated deviance of the model against the regularization variable.**

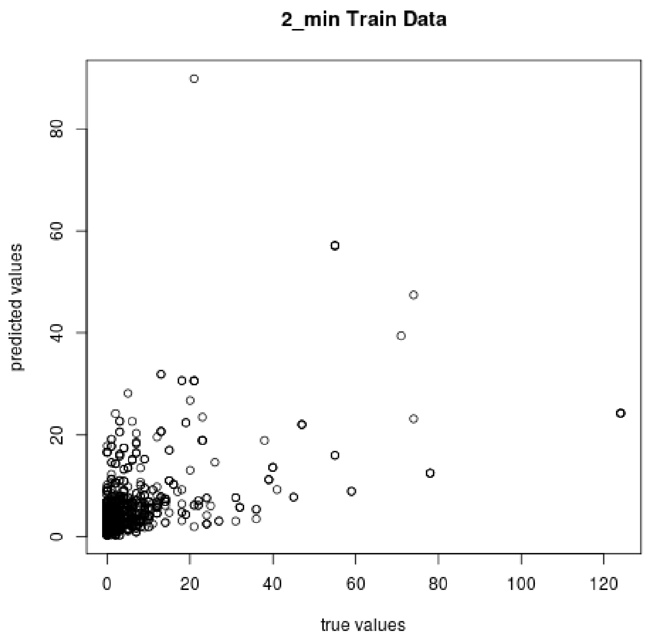


2. **Indicate the value of regularization constant that you choose, and show the scatter plot of true values vs predicted values for your training data:**
(We choose to draw each test.csv separately and choose four plots for this report.)

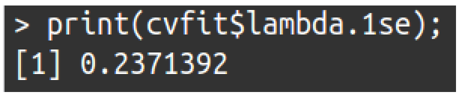
1)



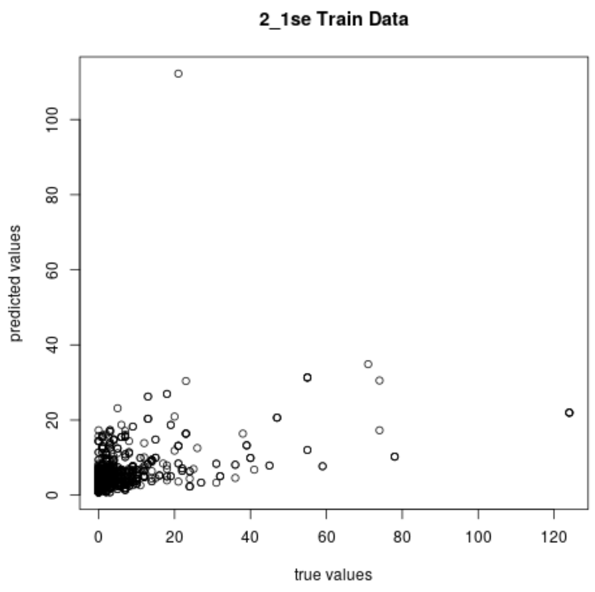
```
> print(cvfit$lambda.min);
[1] 0.04048802
```



2)

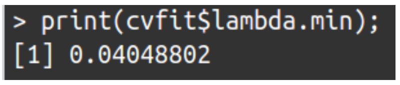


```
> print(cvfit$lambda.1se);
[1] 0.2371392
```

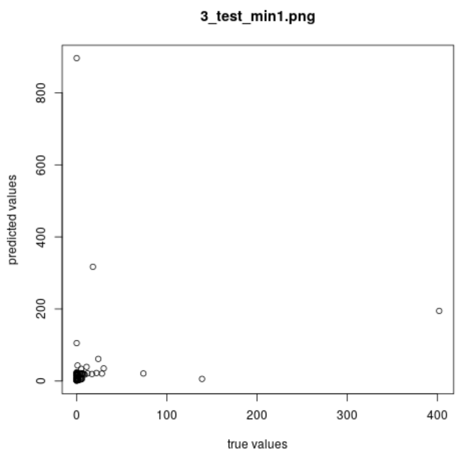


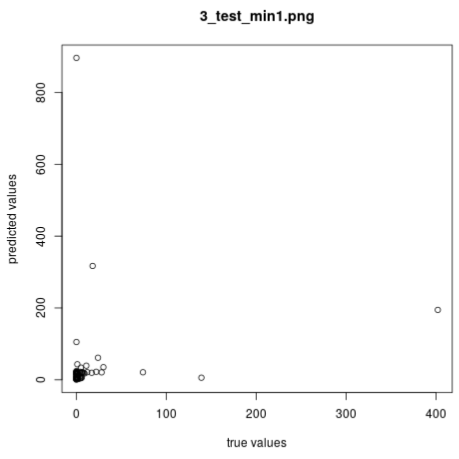
3. Indicate the value of regularization constant that you choose, and show the scatter plot of true values vs predicted values for your testing data

1)

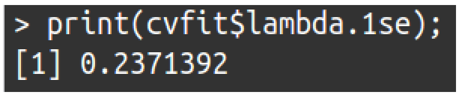


```
> print(cvfit$lambda.min);
[1] 0.04048802
```

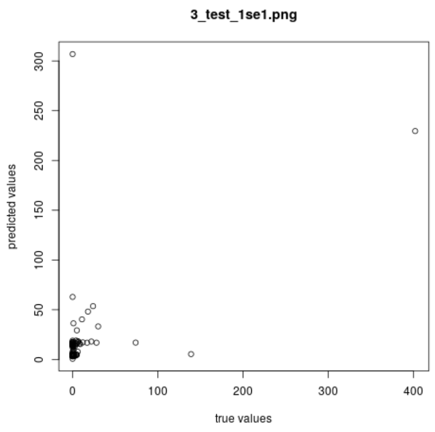


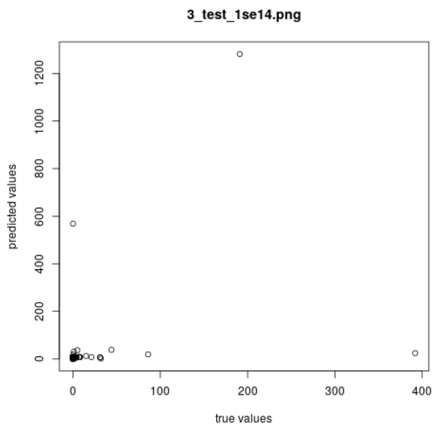


2)



```
> print(cvfit$lambda.1se);
[1] 0.2371392
```





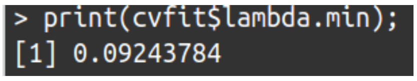
4. Compare the two plots and comment on the performance of the model. Provide comment on why this regression is difficult.

As showed in the two plots, most samples locate near [0,0]. Since there are too many 0 in training data as feature vector, this kind of sparse dataset is hard for regression to extract meaningful features and predict the real value.

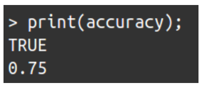
5. **Show the plot of the classification error of the model against the regularization variable. Indicate the value of regularization constant of your choice and provide comment on the model performance compared with the baseline. Remember to include the classification accuracy in**

the comparison.

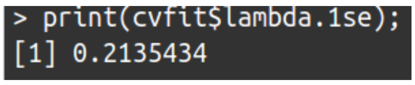
Our choice of regularization constant:



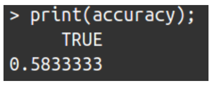
```
> print(cvfit$lambda.min);
[1] 0.09243784
```



```
> print(accuracy);
TRUE
0.75
```

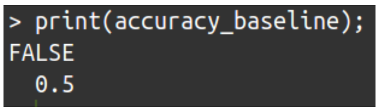


```
> print(cvfit$lambda.1se);
[1] 0.2135434
```

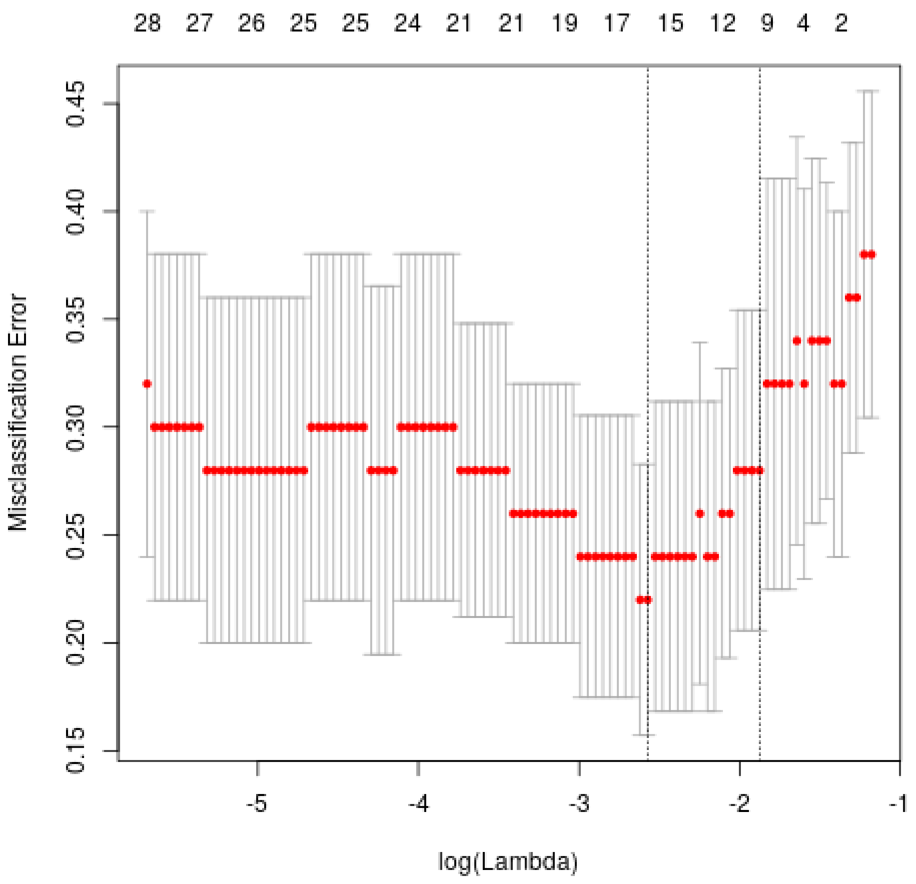


```
> print(accuracy);
     TRUE
0.5833333
```

Baseline:



```
> print(accuracy_baseline);
FALSE
  0.5
```

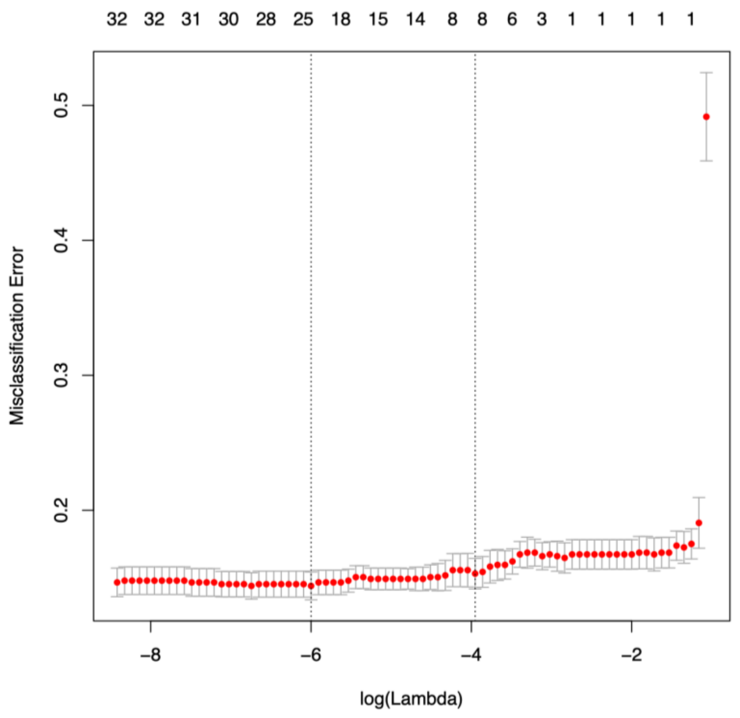


6. Predict gender with the features. Show the plot of the classification error of the model against the regularization variable. Indicate the value of regularization constant of your choice and provide comment on the model performance compared with the baseline. Remember to include

the classification accuracy in the comparison.

Our choice of regularization constant is 0.0101.

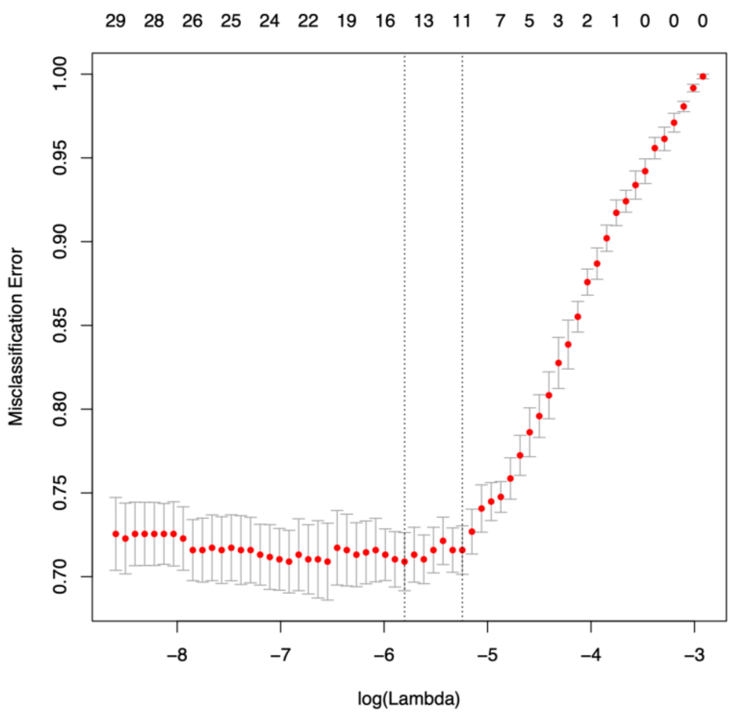
Baseline is 0.511, but accuracy of logistic regression is 0.822, better than baseline.



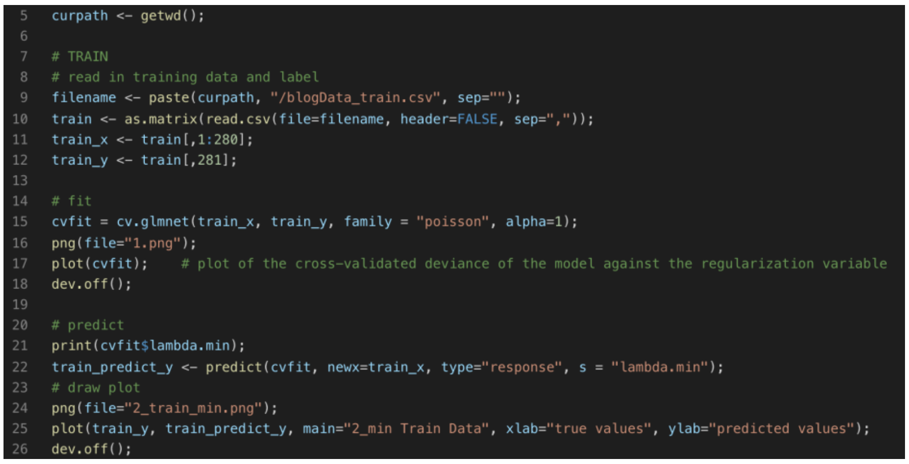
7. Predict the strain of a mouse with the features. Show the plot of the classification error of the model against the regularization variable. Indicate the value of regularization constant of your choice and provide comment on the model performance compared with the baseline. Remember to include the classification accuracy in the comparison.

Our choice of regularization constant is 0.00115.

Baseline is 0.0244, but accuracy of multinomial logistic regression is 0.309, better than baseline.

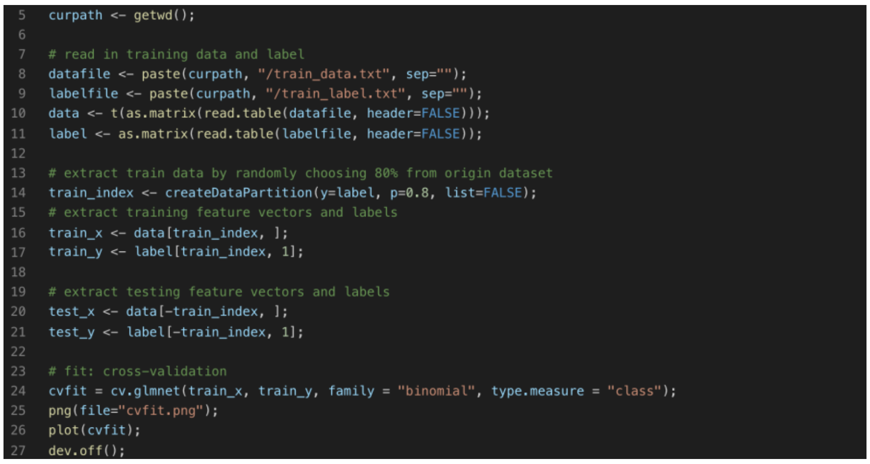


8. 1 page Code screenshot



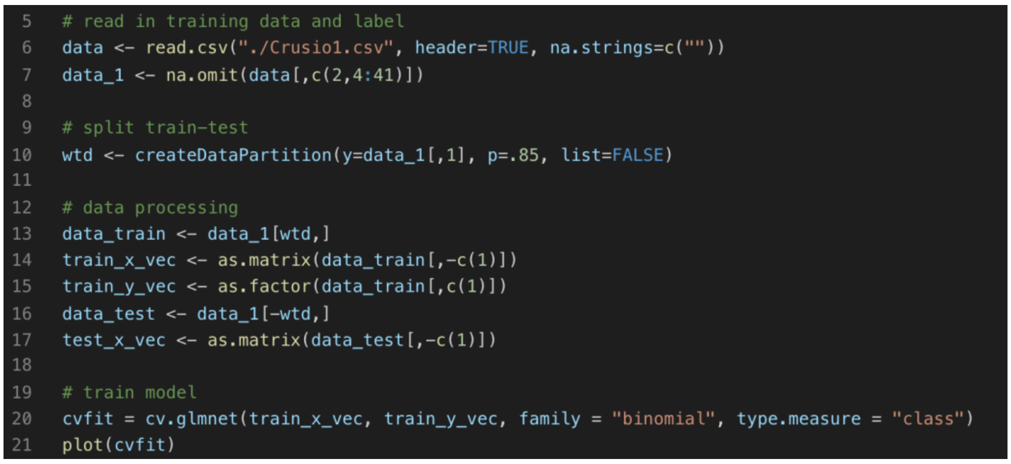
```r
curpath <- getwd();

# TRAIN
# read in training data and label
filename <- paste(curpath, "/blogData_train.csv", sep="");
train <- as.matrix(read.csv(file=filename, header=FALSE, sep=","));
train_x <- train[,1:280];
train_y <- train[,281];

# fit
cvfit = cv.glmnet(train_x, train_y, family = "poisson", alpha=1);
png(file="1.png");
plot(cvfit);    # plot of the cross-validated deviance of the model against the regularization variable
dev.off();

# predict
print(cvfit$lambda.min);
train_predict_y <- predict(cvfit, newx=train_x, type="response", s = "lambda.min");
# draw plot
png(file="2_train_min.png");
plot(train_y, train_predict_y, main="2_min Train Data", xlab="true values", ylab="predicted values");
dev.off();
```

Screen shot of 12.3



```r
curpath <- getwd();

# read in training data and label
datafile <- paste(curpath, "/train_data.txt", sep="");
labelfile <- paste(curpath, "/train_label.txt", sep="");
data <- t(as.matrix(read.table(datafile, header=FALSE)));
label <- as.matrix(read.table(labelfile, header=FALSE));

# extract train data by randomly choosing 80% from origin dataset
train_index <- createDataPartition(y=label, p=0.8, list=FALSE);
# extract training feature vectors and labels
train_x <- data[train_index, ];
train_y <- label[train_index, 1];

# extract testing feature vectors and labels
test_x <- data[-train_index, ];
test_y <- label[-train_index, 1];

# fit: cross-validation
cvfit = cv.glmnet(train_x, train_y, family = "binomial", type.measure = "class");
png(file="cvfit.png");
plot(cvfit);
dev.off();
```

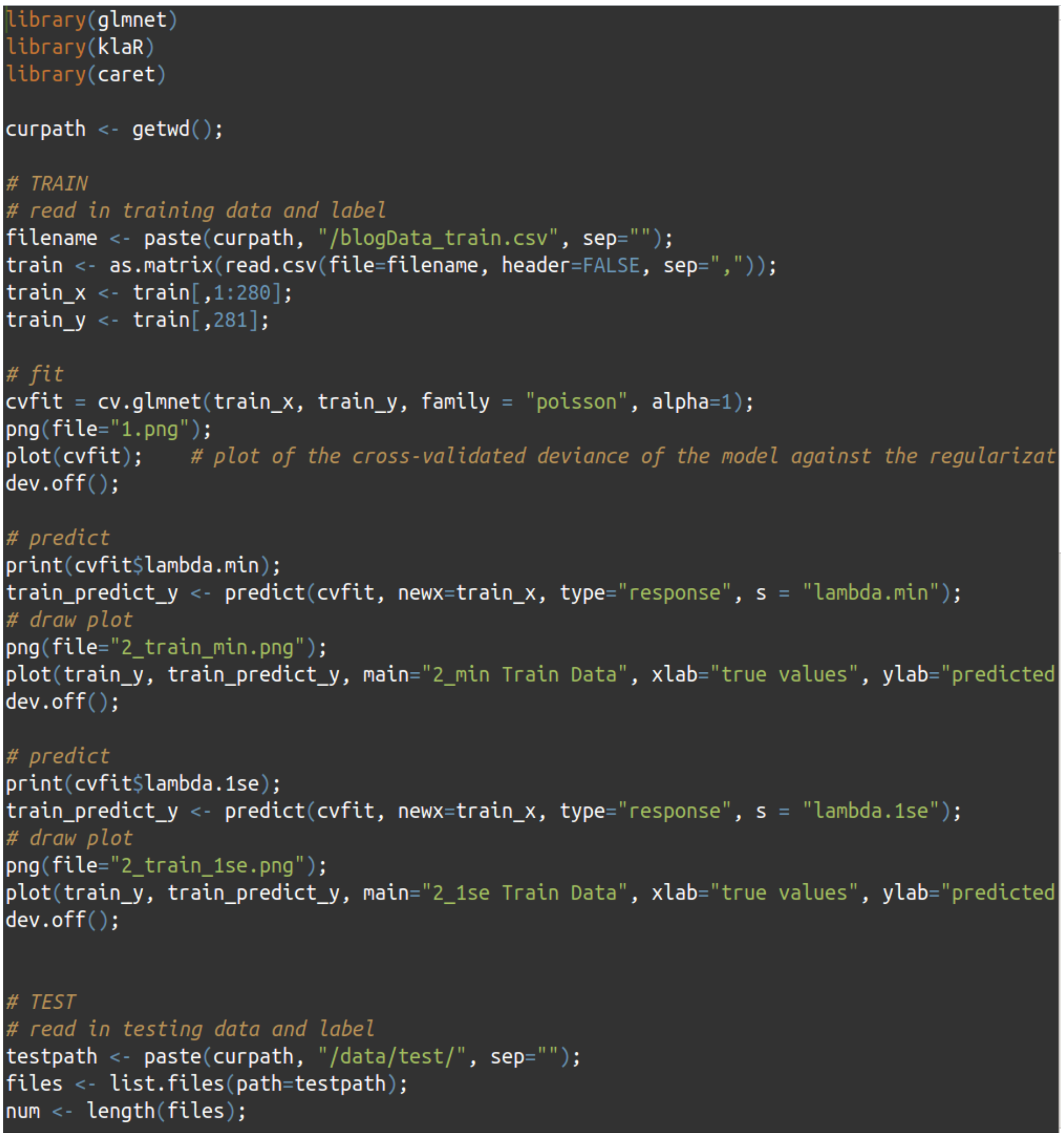
Screen shot of 12.4



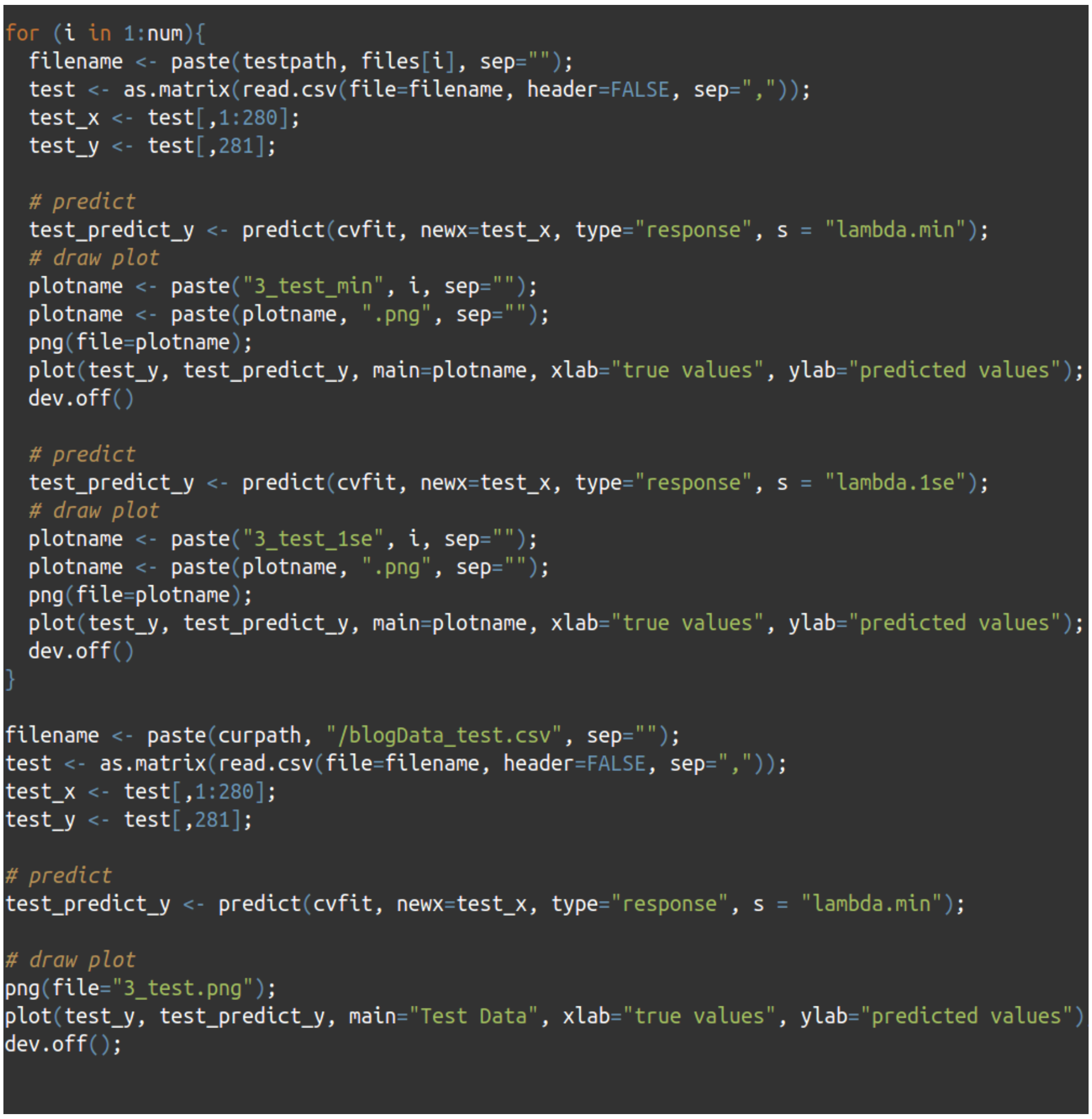
```r
# read in training data and label
data <- read.csv("./Crusio1.csv", header=TRUE, na.strings=c(""))
data_1 <- na.omit(data[,c(2,4:41)])

# split train-test
wtd <- createDataPartition(y=data_1[,1], p=.85, list=FALSE)

# data processing
data_train <- data_1[wtd,]
train_x_vec <- as.matrix(data_train[,-c(1)])
train_y_vec <- as.factor(data_train[,c(1)])
data_test <- data_1[-wtd,]
test_x_vec <- as.matrix(data_test[,-c(1)])

# train model
cvfit = cv.glmnet(train_x_vec, train_y_vec, family = "binomial", type.measure = "class")
plot(cvfit)
```

Screen shot of 12.5

12.3

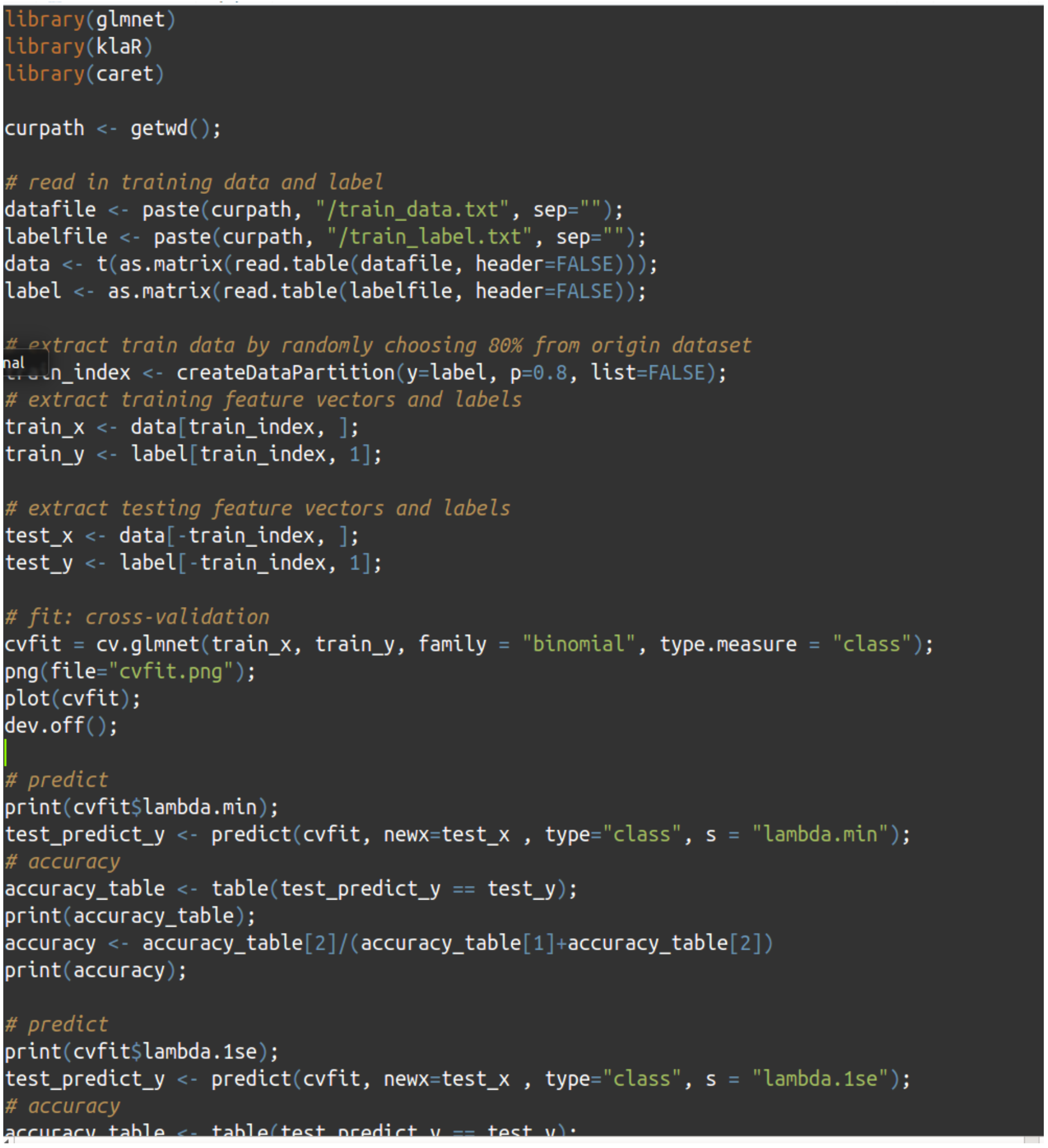


```r
library(glmnet)
library(klaR)
library(caret)

curpath <- getwd();

# TRAIN
# read in training data and label
filename <- paste(curpath, "/blogData_train.csv", sep="");
train <- as.matrix(read.csv(file=filename, header=FALSE, sep=","));
train_x <- train[,1:280];
train_y <- train[,281];

# fit
cvfit = cv.glmnet(train_x, train_y, family = "poisson", alpha=1);
png(file="1.png");
plot(cvfit);     # plot of the cross-validated deviance of the model against the regularizat
dev.off();

# predict
print(cvfit$lambda.min);
train_predict_y <- predict(cvfit, newx=train_x, type="response", s = "lambda.min");
# draw plot
png(file="2_train_min.png");
plot(train_y, train_predict_y, main="2_min Train Data", xlab="true values", ylab="predicted
dev.off();

# predict
print(cvfit$lambda.1se);
train_predict_y <- predict(cvfit, newx=train_x, type="response", s = "lambda.1se");
# draw plot
png(file="2_train_1se.png");
plot(train_y, train_predict_y, main="2_1se Train Data", xlab="true values", ylab="predicted
dev.off();


# TEST
# read in testing data and label
testpath <- paste(curpath, "/data/test/", sep="");
files <- list.files(path=testpath);
num <- length(files);
```

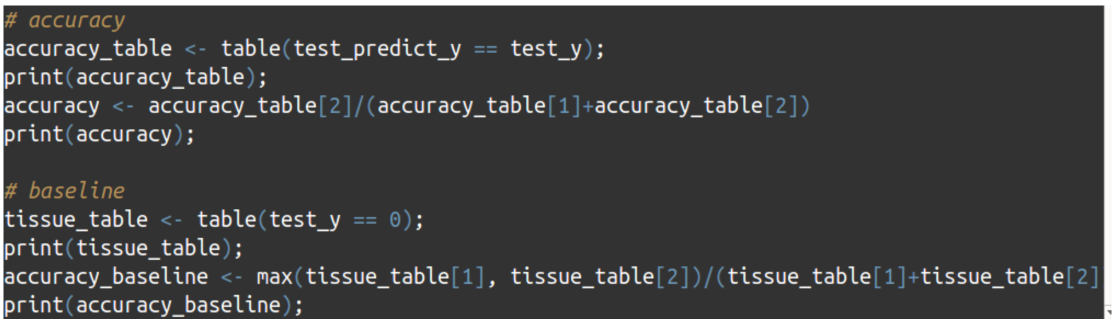


```r
for (i in 1:num){
  filename <- paste(testpath, files[i], sep="");
  test <- as.matrix(read.csv(file=filename, header=FALSE, sep=","));
  test_x <- test[,1:280];
  test_y <- test[,281];

  # predict
  test_predict_y <- predict(cvfit, newx=test_x, type="response", s = "lambda.min");
  # draw plot
  plotname <- paste("3_test_min", i, sep="");
  plotname <- paste(plotname, ".png", sep="");
  png(file=plotname);
  plot(test_y, test_predict_y, main=plotname, xlab="true values", ylab="predicted values");
  dev.off()

  # predict
  test_predict_y <- predict(cvfit, newx=test_x, type="response", s = "lambda.1se");
  # draw plot
  plotname <- paste("3_test_1se", i, sep="");
  plotname <- paste(plotname, ".png", sep="");
  png(file=plotname);
  plot(test_y, test_predict_y, main=plotname, xlab="true values", ylab="predicted values");
  dev.off()
}

filename <- paste(curpath, "/blogData_test.csv", sep="");
test <- as.matrix(read.csv(file=filename, header=FALSE, sep=","));
test_x <- test[,1:280];
test_y <- test[,281];

# predict
test_predict_y <- predict(cvfit, newx=test_x, type="response", s = "lambda.min");

# draw plot
png(file="3_test.png");
plot(test_y, test_predict_y, main="Test Data", xlab="true values", ylab="predicted values")
dev.off();
```

12.4

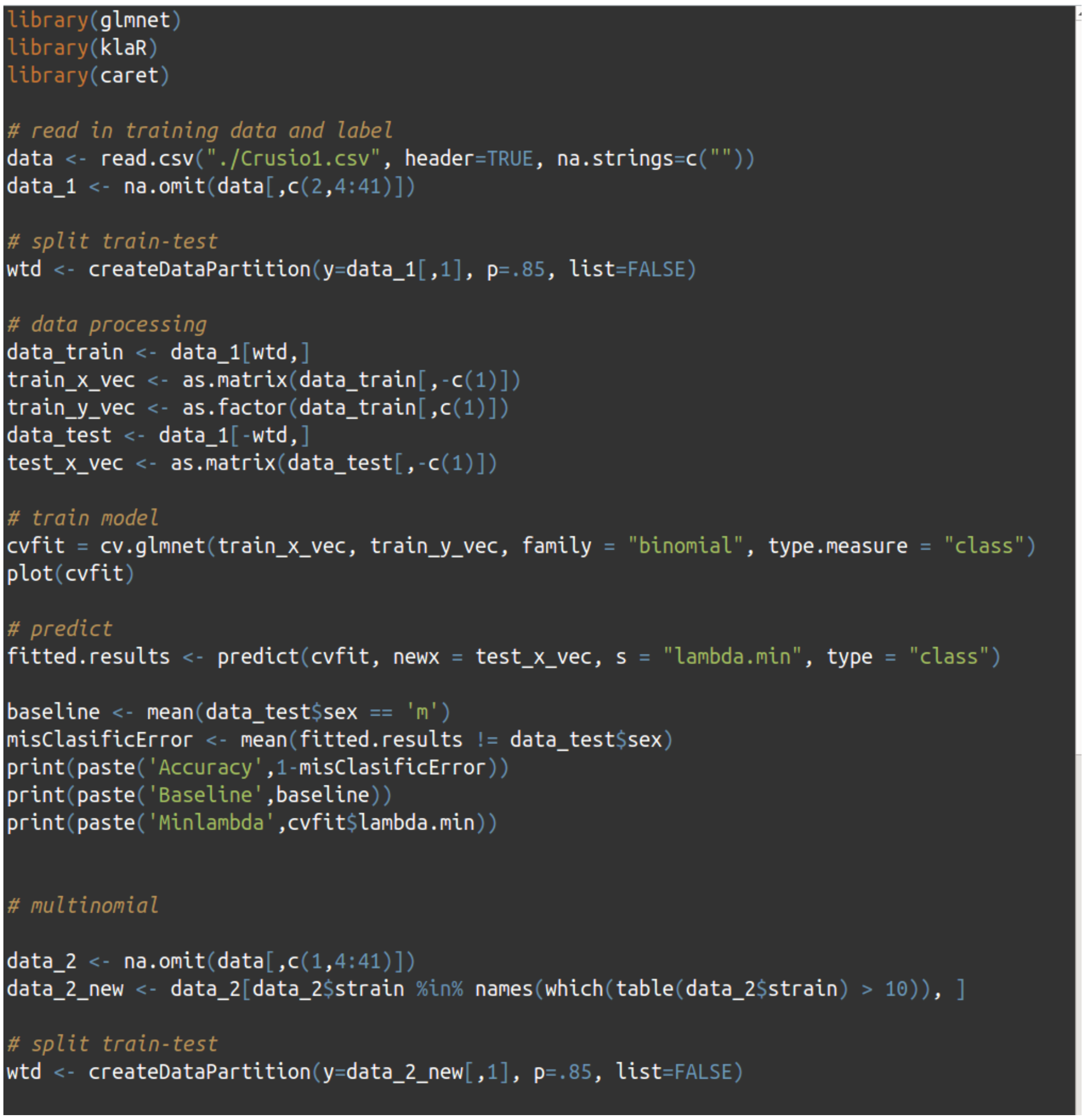


```
library(glmnet)
library(klaR)
library(caret)

curpath <- getwd();

# read in training data and label
datafile <- paste(curpath, "/train_data.txt", sep="");
labelfile <- paste(curpath, "/train_label.txt", sep="");
data <- t(as.matrix(read.table(datafile, header=FALSE)));
label <- as.matrix(read.table(labelfile, header=FALSE));

# extract train data by randomly choosing 80% from origin dataset
train_index <- createDataPartition(y=label, p=0.8, list=FALSE);
# extract training feature vectors and labels
train_x <- data[train_index, ];
train_y <- label[train_index, 1];

# extract testing feature vectors and labels
test_x <- data[-train_index, ];
test_y <- label[-train_index, 1];

# fit: cross-validation
cvfit = cv.glmnet(train_x, train_y, family = "binomial", type.measure = "class");
png(file="cvfit.png");
plot(cvfit);
dev.off();

# predict
print(cvfit$lambda.min);
test_predict_y <- predict(cvfit, newx=test_x , type="class", s = "lambda.min");
# accuracy
accuracy_table <- table(test_predict_y == test_y);
print(accuracy_table);
accuracy <- accuracy_table[2]/(accuracy_table[1]+accuracy_table[2])
print(accuracy);

# predict
print(cvfit$lambda.1se);
test_predict_y <- predict(cvfit, newx=test_x , type="class", s = "lambda.1se");
# accuracy
accuracy_table <- table(test_predict_y == test_y);
```

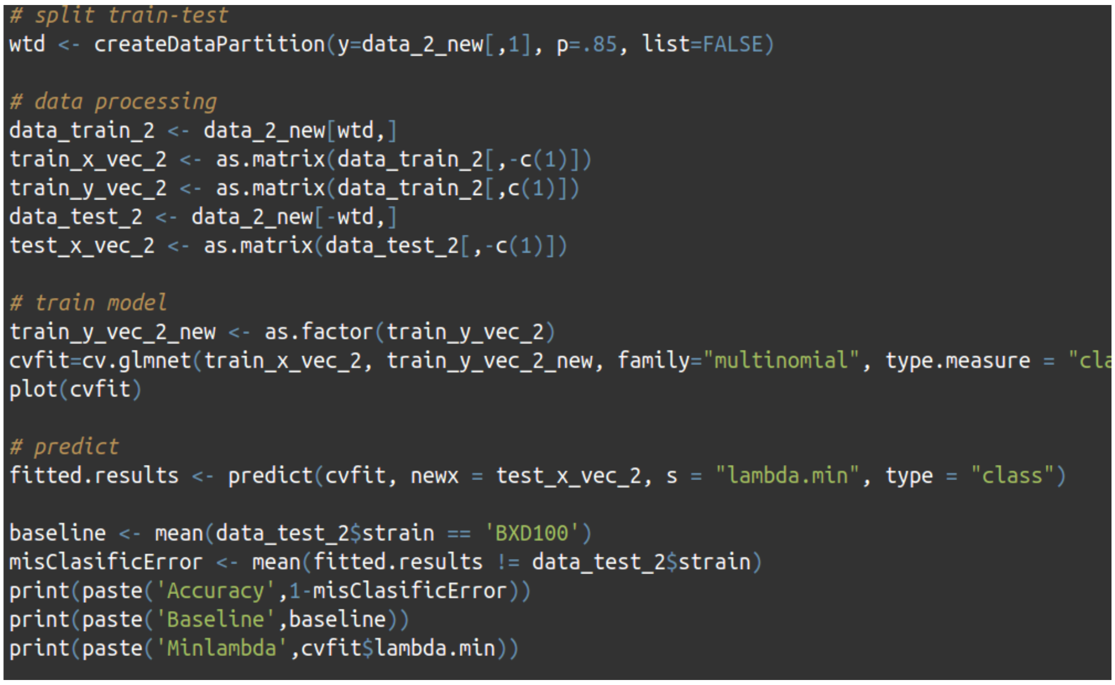


```
# accuracy
accuracy_table <- table(test_predict_y == test_y);
print(accuracy_table);
accuracy <- accuracy_table[2]/(accuracy_table[1]+accuracy_table[2])
print(accuracy);

# baseline
tissue_table <- table(test_y == 0);
print(tissue_table);
accuracy_baseline <- max(tissue_table[1], tissue_table[2])/(tissue_table[1]+tissue_table[2]
print(accuracy_baseline);
```

12.5



```r
library(glmnet)
library(klaR)
library(caret)

# read in training data and label
data <- read.csv("./Crusio1.csv", header=TRUE, na.strings=c(""))
data_1 <- na.omit(data[,c(2,4:41)])

# split train-test
wtd <- createDataPartition(y=data_1[,1], p=.85, list=FALSE)

# data processing
data_train <- data_1[wtd,]
train_x_vec <- as.matrix(data_train[,-c(1)])
train_y_vec <- as.factor(data_train[,c(1)])
data_test <- data_1[-wtd,]
test_x_vec <- as.matrix(data_test[,-c(1)])

# train model
cvfit = cv.glmnet(train_x_vec, train_y_vec, family = "binomial", type.measure = "class")
plot(cvfit)

# predict
fitted.results <- predict(cvfit, newx = test_x_vec, s = "lambda.min", type = "class")

baseline <- mean(data_test$sex == 'm')
misClasificError <- mean(fitted.results != data_test$sex)
print(paste('Accuracy',1-misClasificError))
print(paste('Baseline',baseline))
print(paste('Minlambda',cvfit$lambda.min))


# multinomial

data_2 <- na.omit(data[,c(1,4:41)])
data_2_new <- data_2[data_2$strain %in% names(which(table(data_2$strain) > 10)), ]

# split train-test
wtd <- createDataPartition(y=data_2_new[,1], p=.85, list=FALSE)
```



```
# split train-test
wtd <- createDataPartition(y=data_2_new[,1], p=.85, list=FALSE)

# data processing
data_train_2 <- data_2_new[wtd,]
train_x_vec_2 <- as.matrix(data_train_2[,-c(1)])
train_y_vec_2 <- as.matrix(data_train_2[,c(1)])
data_test_2 <- data_2_new[-wtd,]
test_x_vec_2 <- as.matrix(data_test_2[,-c(1)])

# train model
train_y_vec_2_new <- as.factor(train_y_vec_2)
cvfit=cv.glmnet(train_x_vec_2, train_y_vec_2_new, family="multinomial", type.measure = "cla
plot(cvfit)

# predict
fitted.results <- predict(cvfit, newx = test_x_vec_2, s = "lambda.min", type = "class")

baseline <- mean(data_test_2$strain == 'BXD100')
misClasificError <- mean(fitted.results != data_test_2$strain)
print(paste('Accuracy',1-misClasificError))
print(paste('Baseline',baseline))
print(paste('Minlambda',cvfit$lambda.min))
```