In [1]:
```
library(boot)
library(glmnet)
```

```
Loading required package: Matrix
Loading required package: foreach
Loaded glmnet 2.0-16
```

In [2]:
```
set.seed(2019)
```

In [3]:
```
train <- read.csv("trainC.csv")
test <- read.csv("testC.csv")
train <- subset(train, select = -c(sessionDate, trialNum, timeSinceKetamine, animalName))
test <- subset(test, select = -c(sessionDate, trialNum, timeSinceKetamine, animalName))

#TRAIN 1: ALL COVARIATES PLUS INTERACTION TERMS
train1 <- read.csv("trainC.csv")
test1 <- read.csv("testC.csv")
train1 <- subset(train1, select = -c(sessionDate, trialNum, timeSinceKetamine, animalName))
test1 <- subset(test1, select = -c(sessionDate, trialNum, timeSinceKetamine, animalName))

#TRAIN 2: ALL COVARIATES NO INTERACTION TERMS
train2 <- subset(train1, select = c(totalCellNum,gender,genotype,weight_g,ketamine_day,
                                    correlationScore,lickAccuracy,lickNumber,avgFR,
                                    avgSingleCellVariance,varianceFR,avgTrialSpeed,
                                    varianceSpeed,medianCellDepth,ketBool))
test2 <- subset(test1, select = c(totalCellNum,gender,genotype,weight_g,ketamine_day,
                                    correlationScore,lickAccuracy,lickNumber,avgFR,
                                    avgSingleCellVariance,varianceFR,avgTrialSpeed,
                                    varianceSpeed,medianCellDepth,ketBool))
```

In [4]:
```
# First, let's do a 50% split on the training data to determine the best lambda
n = length(train[,1])
n50 = round(n/2)
train50A = train[1:n50,]
train50B = train[(n50+1):n,]
```

Basic Logistic Regression Model with Interaction Terms

Estimate test error

In [5]:
```
k = 10
n = length(train1[,1])
fsize = round(n/k)
rmse = rep(0,k)
zoloss = rep(0,k)
for (i in 1:(k-1)){
    # Get train and validation sets
    df_train <- train1[-(((i-1)*fsize+1):(i*fsize)),]
    df_val <- train1[((i-1)*fsize+1):(i*fsize),]
    # Fit model on training and make predictions on validation
    model_cv <- glm(ketBool ~ ., data=df_train, family='binomial')
    lr_pred_lo <- predict(model_cv,df_val) # lo : log odds
    num_val = length(df_val$ketBool)
    lr_pred = rep(0,num_val)
    actual = rep(0,num_val)
    for (j in 1:num_val){
        if (lr_pred_lo[j]>0){
            lr_pred[j]=1
        }
    actual[j] = df_val$ketBool[j]
    }
    # Compute 0-1 loss for each observation
    lr_loss = abs(lr_pred-actual) # loss is 0 if NB_pred=actual, 1 otherwise
    # Compute mean 0-1 loss on the val set
    zoloss[i] = mean(lr_loss)
}
df_train <- train1[-(((k-1)*fsize+1):n),]
df_val <- train1[((k-1)*fsize+1):n,]
# Fit model on training and make predictions on validation
model_cv <- glm(ketBool ~ ., data=df_train, family='binomial')
lr_pred_lo <- predict(model_cv,df_val) # lo : log odds
num_val = length(df_val$ketBool)
lr_pred = rep(0,num_val)
actual = rep(0,num_val)
for (j in 1:num_val){
    if (lr_pred_lo[j]>0){
        lr_pred[j]=1
    }
    actual[j] = df_val$ketBool[j]
}
lr_loss = abs(lr_pred-actual)
zoloss[k] = mean(lr_loss)
test_error_est = mean(zoloss)

cat("=====================================================================\n")
cat("Logistic Regression Model with Interaction Terms\n\n")
cat("Zero-One Loss (10-fold Cross-Validation Average):",test_error_est,"\n")
cat("Accuracy (10-fold Cross-Validation Average):",1-test_error_est,"\n")
cat("=====================================================================\n")
```

```
=====================================================================
Logistic Regression Model with Interaction Terms

Zero-One Loss (10-fold Cross-Validation Average): 0.09182746
Accuracy (10-fold Cross-Validation Average): 0.9081725
=====================================================================
```

Reduced dataset to match Lasso and Ridge

In [6]:

```
k = 10
n = length(train50B[,1])
fsize = round(n/k)
rmse = rep(0,k)
zoloss = rep(0,k)
for (i in 1:(k-1)){
    # Get train and validation sets
    df_train <- train50B[-(((i-1)*fsize+1):(i*fsize)),]
    df_val <- train50B[((i-1)*fsize+1):(i*fsize),]
    # Fit model on training and make predictions on validation
    model_cv <- glm(ketBool ~ ., data=df_train, family='binomial')
    lr_pred_lo <- predict(model_cv,df_val) # lo : log odds
    num_val = length(df_val$ketBool)
    lr_pred = rep(0,num_val)
    actual = rep(0,num_val)
    for (j in 1:num_val){
        if (lr_pred_lo[j]>0){
            lr_pred[j]=1
        }
    actual[j] = df_val$ketBool[j]
    }
    # Compute 0-1 loss for each observation
    lr_loss = abs(lr_pred-actual) # loss is 0 if NB_pred=actual, 1 otherwise
    # Compute mean 0-1 loss on the val set
    zoloss[i] = mean(lr_loss)
}
df_train <- train50B[-(((k-1)*fsize+1):n),]
df_val <- train50B[((k-1)*fsize+1):n,]
# Fit model on training and make predictions on validation
model_cv <- glm(ketBool ~ ., data=df_train, family='binomial')
lr_pred_lo <- predict(model_cv,df_val) # lo : log odds
num_val = length(df_val$ketBool)
lr_pred = rep(0,num_val)
actual = rep(0,num_val)
for (j in 1:num_val){
    if (lr_pred_lo[j]>0){
        lr_pred[j]=1
    }
    actual[j] = df_val$ketBool[j]
}
lr_loss = abs(lr_pred-actual)
zoloss[k] = mean(lr_loss)
test_error_est = mean(zoloss)

cat("======================================================================\n")
cat("Logistic Regression Model with Interaction Terms\n\n")
cat("Zero-One Loss (10-fold Cross-Validation Average):",test_error_est,"\n")
cat("Accuracy (10-fold Cross-Validation Average):",1-test_error_est,"\n")
cat("======================================================================\n")
```

```
======================================================================
Logistic Regression Model with Interaction Terms

Zero-One Loss (10-fold Cross-Validation Average): 0.09505025
Accuracy (10-fold Cross-Validation Average): 0.9049497
======================================================================
```

Basic Logistic Regression without Interaction Terms

In [7]:
```
k = 10
n = length(train2[,1])
fsize = round(n/k)
rmse = rep(0,k)
zoloss = rep(0,k)
for (i in 1:(k-1)){
    # Get train and validation sets
    df_train <- train2[-(((i-1)*fsize+1):(i*fsize)),]
    df_val <- train2[((i-1)*fsize+1):(i*fsize),]
    # Fit model on training and make predictions on validation
    model_cv <- glm(ketBool ~ ., data=df_train, family='binomial')
    lr_pred_lo <- predict(model_cv,df_val) # lo : log odds
    num_val = length(df_val$ketBool)
    lr_pred = rep(0,num_val)
    actual = rep(0,num_val)
    for (j in 1:num_val){
        if (lr_pred_lo[j]>0){
            lr_pred[j]=1
        }
    actual[j] = df_val$ketBool[j]
    }
    # Compute 0-1 loss for each observation
    lr_loss = abs(lr_pred-actual) # loss is 0 if NB_pred=actual, 1 otherwise
    # Compute mean 0-1 loss on the val set
    zoloss[i] = mean(lr_loss)
}
df_train <- train2[-(((k-1)*fsize+1):n),]
df_val <- train2[((k-1)*fsize+1):n,]
# Fit model on training and make predictions on validation
model_cv <- glm(ketBool ~ ., data=df_train, family='binomial')
lr_pred_lo <- predict(model_cv,df_val) # lo : log odds
num_val = length(df_val$ketBool)
lr_pred = rep(0,num_val)
actual = rep(0,num_val)
for (j in 1:num_val){
    if (lr_pred_lo[j]>0){
        lr_pred[j]=1
    }
    actual[j] = df_val$ketBool[j]
}
lr_loss = abs(lr_pred-actual)
zoloss[k] = mean(lr_loss)
test_error_est = mean(zoloss)

cat("======================================================================\n")
cat("Logistic Regression Model without Interaction Terms\n\n")
cat("Zero-One Loss (10-fold Cross-Validation Average):",test_error_est,"\n")
cat("Accuracy (10-fold Cross-Validation Average):",1-test_error_est,"\n")
cat("======================================================================\n")
```

```
======================================================================
Logistic Regression Model without Interaction Terms

Zero-One Loss (10-fold Cross-Validation Average): 0.1413709
Accuracy (10-fold Cross-Validation Average): 0.8586291
======================================================================
```

GLMNET

In [8]:
```
# First, let's do a 50% split on the training data to determine the best lambda
n = length(train[,1])
n50 = round(n/2)
train50A = train[1:n50,]
train50B = train[(n50+1):n,]

xA = as.matrix(train50A[,-length(train50A)])
yA = as.matrix(train50A$ketBool)
xB = as.matrix(train50B[,-length(train50B)])
yB = as.matrix(train50B$ketBool)
```

Lasso

In [9]:
```
# Select regularization parameter over trainA (50% of training data)
model_lasso <- cv.glmnet(xA, yA, family='binomial',alpha=1)
lambda_min = model_lasso$lambda.min
lambda_1se = model_lasso$lambda.1se
```

In [10]:
```
k = 10
n = length(train50B[,1])
fsize = round(n/k)
rmse = rep(0,k)
zoloss = rep(0,k)
for (i in 1:(k-1)){
    # Get train and validation sets
    xB_train = xB[-(((i-1)*fsize+1):(i*fsize)),]
    yB_train = yB[-(((i-1)*fsize+1):(i*fsize)),]
    xB_val = xB[((i-1)*fsize+1):(i*fsize),]
    yB_val = yB[((i-1)*fsize+1):(i*fsize),]
    # Fit model on training and make predictions on validation
    model_cv <- glmnet(xB_train, yB_train, family='binomial',alpha=1,lambda=lambda_min)
    pred_lo = predict(model_cv, newx = xB_val)
    num_val = length(yB_val)
    lr_pred = rep(0,num_val)
    actual = rep(0,num_val)
    for (j in 1:num_val){
        if (pred_lo[j]>0){
            lr_pred[j]=1
        }
        actual[j] = yB_val[j]
    }
    # Compute 0-1 loss for each observation
    lr_loss = abs(lr_pred-actual) # loss is 0 if NB_pred=actual, 1 otherwise
    # Compute mean 0-1 loss on the val set
    zoloss[i] = mean(lr_loss)
}
xB_train = xB[-(((k-1)*fsize+1):(length(yB))),]
yB_train = yB[-(((k-1)*fsize+1):(length(yB))),]
xB_val = xB[((k-1)*fsize+1):(length(yB)),]
yB_val = yB[((k-1)*fsize+1):(length(yB)),]
# Fit model on training and make predictions on validation
model_cv <- glmnet(xB_train, yB_train, family='binomial',alpha=1,lambda=lambda_min)
pred_lo = predict(model_cv, newx = xB_val)
num_val = length(yB_val)
lr_pred = rep(0,num_val)
actual = rep(0,num_val)
for (j in 1:num_val){
    if (pred_lo[j]>0){
        lr_pred[j]=1
    }
    actual[j] = yB_val[j]
}
# Compute 0-1 loss for each observation
lr_loss = abs(lr_pred-actual) # loss is 0 if NB_pred=actual, 1 otherwise
# Compute mean 0-1 loss on the val set
zoloss[k] = mean(lr_loss)
test_error_est = mean(zoloss)

cat("=====================================================================\n")
cat("GLMNET Lasso Logistic Regression Model with lambda.min\n\n")
cat("Zero-One Loss (10-fold Cross-Validation Average):",test_error_est,"\n")
cat("Accuracy (10-fold Cross-Validation Average):",1-test_error_est,"\n")
cat("=====================================================================\n")
```

```
=====================================================================
GLMNET Lasso Logistic Regression Model with lambda.min

Zero-One Loss (10-fold Cross-Validation Average): 0.09205025
Accuracy (10-fold Cross-Validation Average): 0.9079497
=====================================================================
```

Ridge

In [11]:
```
# Select regularization parameter over trainA (50% of training data)
model_lasso <- cv.glmnet(xA, yA, family='binomial',alpha=0)
lambda_min = model_lasso$lambda.min
lambda_1se = model_lasso$lambda.1se
```

In [12]:
```
k = 10
n = length(train50B[,1])
fsize = round(n/k)
rmse = rep(0,k)
zoloss = rep(0,k)
for (i in 1:(k-1)){
    # Get train and validation sets
    xB_train = xB[-(((i-1)*fsize+1):(i*fsize)),]
    yB_train = yB[-(((i-1)*fsize+1):(i*fsize)),]
    xB_val = xB[((i-1)*fsize+1):(i*fsize),]
    yB_val = yB[((i-1)*fsize+1):(i*fsize),]
    # Fit model on training and make predictions on validation
    model_cv <- glmnet(xB_train, yB_train, family='binomial',alpha=0,lambda=lambda_min)
    pred_lo = predict(model_cv, newx = xB_val)
    num_val = length(yB_val)
    lr_pred = rep(0,num_val)
    actual = rep(0,num_val)
    for (j in 1:num_val){
        if (pred_lo[j]>0){
            lr_pred[j]=1
        }
        actual[j] = yB_val[j]
    }
    # Compute 0-1 loss for each observation
    lr_loss = abs(lr_pred-actual) # loss is 0 if NB_pred=actual, 1 otherwise
    # Compute mean 0-1 loss on the val set
    zoloss[i] = mean(lr_loss)
}
xB_train = xB[-(((k-1)*fsize+1):(length(yB))),]
yB_train = yB[-(((k-1)*fsize+1):(length(yB))),]
xB_val = xB[((k-1)*fsize+1):(length(yB)),]
yB_val = yB[((k-1)*fsize+1):(length(yB)),]
# Fit model on training and make predictions on validation
model_cv <- glmnet(xB_train, yB_train, family='binomial',alpha=0,lambda=lambda_min)
pred_lo = predict(model_cv, newx = xB_val)
num_val = length(yB_val)
lr_pred = rep(0,num_val)
actual = rep(0,num_val)
for (j in 1:num_val){
    if (pred_lo[j]>0){
        lr_pred[j]=1
    }
    actual[j] = yB_val[j]
}
# Compute 0-1 loss for each observation
lr_loss = abs(lr_pred-actual) # loss is 0 if NB_pred=actual, 1 otherwise
# Compute mean 0-1 loss on the val set
zoloss[k] = mean(lr_loss)
test_error_est = mean(zoloss)

cat("======================================================================\n")
cat("GLMNET Ridge Logistic Regression Model with lambda.min\n\n")
cat("Zero-One Loss (10-fold Cross-Validation Average):",test_error_est,"\n")
cat("Accuracy (10-fold Cross-Validation Average):",1-test_error_est,"\n")
cat("======================================================================\n")
```

```
======================================================================
GLMNET Ridge Logistic Regression Model with lambda.min

Zero-One Loss (10-fold Cross-Validation Average): 0.1260879
Accuracy (10-fold Cross-Validation Average): 0.8739121
======================================================================
```

In [ ]: