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))
test <- subset(test, select = -c(sessionDate, trialNum, timeSinceKetamine))

#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))
test1 <- subset(test1, select = -c(sessionDate, trialNum, timeSinceKetamine))

#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 ~ . + animalName:correlationScore
                    + animalName:lickAccuracy
                    + animalName:lickNumber
                    + animalName:avgFR
                    + animalName:avgSingleCellVariance
                    + animalName:varianceFR
                    + animalName:avgTrialSpeed
                    + animalName:varianceSpeed, 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 ~ . + animalName:correlationScore
                    + animalName:lickAccuracy
                    + animalName:lickNumber
                    + animalName:avgFR
                    + animalName:avgSingleCellVariance
                    + animalName:varianceFR
                    + animalName:avgTrialSpeed
                    + animalName:varianceSpeed, 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")
```

```
Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”Warning message in predict.lm(object, newdata, se.fit, scale
= 1, type = if (type == :
“prediction from a rank-deficient fit may be misleading”Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”Warning message in predict.lm(object, newdata, se.fit, scale
= 1, type = if (type == :
“prediction from a rank-deficient fit may be misleading”Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”Warning message in predict.lm(object, newdata, se.fit, scale
= 1, type = if (type == :
“prediction from a rank-deficient fit may be misleading”Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”Warning message in predict.lm(object, newdata, se.fit, scale
= 1, type = if (type == :
“prediction from a rank-deficient fit may be misleading”Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”Warning message in predict.lm(object, newdata, se.fit, scale
= 1, type = if (type == :
“prediction from a rank-deficient fit may be misleading”Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”Warning message in predict.lm(object, newdata, se.fit, scale
= 1, type = if (type == :
“prediction from a rank-deficient fit may be misleading”Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”Warning message in predict.lm(object, newdata, se.fit, scale
= 1, type = if (type == :
“prediction from a rank-deficient fit may be misleading”Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”Warning message in predict.lm(object, newdata, se.fit, scale
= 1, type = if (type == :
“prediction from a rank-deficient fit may be misleading”Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”Warning message in predict.lm(object, newdata, se.fit, scale
= 1, type = if (type == :
“prediction from a rank-deficient fit may be misleading”Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”Warning message in predict.lm(object, newdata, se.fit, scale
= 1, type = if (type == :
“prediction from a rank-deficient fit may be misleading”

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

Zero-One Loss (10-fold Cross-Validation Average): 0.05053967
Accuracy (10-fold Cross-Validation Average): 0.9494603
=====================================================================
```

In [6]:
```
summary(model_cv)
```

```
Call:
glm(formula = ketBool ~ . + animalName:correlationScore + animalName:lickAccuracy +
    animalName:lickNumber + animalName:avgFR + animalName:avgSingleCellVariance +
    animalName:varianceFR + animalName:avgTrialSpeed + animalName:varianceSpeed,
    family = "binomial", data = df_train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.5384  -0.1027   0.0000   0.0712   5.5159

Coefficients: (18 not defined because of singularities)
                                    Estimate Std. Error z value Pr(>|z|)
(Intercept)                         -27.2669     6.0252  -4.526 6.03e-06 ***
animalNameG2                          0.4197     3.3163   0.127 0.899281
animalNameG3                         -0.9364     3.5589  -0.263 0.792468
animalNameG4                          1.4332     3.8333   0.374 0.708496
animalNameG5                         -8.9931     4.1376  -2.173 0.029743 *
animalNameHCN1                      -14.0175    10.5442  -1.329 0.183715
animalNameHCNb2                       8.4330     4.4879   1.879 0.060239 .
animalNameHCNb4                      -4.7334    23.8994  -0.198 0.843002
animalNameHCNd1                     104.4537    16.9813   6.151 7.69e-10 ***
animalNameHCNd2                      68.3925    13.2492   5.162 2.44e-07 ***
animalNameHCNe1                      57.9201    13.7486   4.213 2.52e-05 ***
animalNameHCNe2                     102.2463    16.6521   6.140 8.25e-10 ***
animalNameHCNe3                      66.1861    12.7951   5.173 2.31e-07 ***
animalNamenpI1                       -7.5109     6.2653  -1.199 0.230603
totalCellNum                          1.4313     1.2711   1.126 0.260152
gender                                    NA         NA      NA       NA
genotype                                  NA         NA      NA       NA
weight_g                            -42.4023     7.1213  -5.954 2.61e-09 ***
ketamine_day                          0.8366     0.7416   1.128 0.259265
correlationScore                      0.4354     7.9683   0.055 0.956425
lickAccuracy                         -9.0325     5.2510  -1.720 0.085409 .
lickNumber                           16.5052     8.4302   1.958 0.050246 .
avgFR                              -116.8217    22.1238  -5.280 1.29e-07 ***
avgSingleCellVariance               118.9793    28.1016   4.234 2.30e-05 ***
varianceFR                          -11.7773     9.6821  -1.216 0.223831
avgTrialSpeed                        -9.3312     5.5556  -1.680 0.093036 .
varianceSpeed                        17.2301     6.8436   2.518 0.011813 *
medianCellDepth                       4.3650     0.8865   4.924 8.49e-07 ***
totalCellNumxCorrelationScore        -3.1603     1.0375  -3.046 0.002319 **
totalCellNumxLickAccuracy             0.8750     0.5696   1.536 0.124463
totalCellNumxLickNumber               1.6250     0.7325   2.218 0.026530 *
totalCellNumxAvgFR                    0.8384     1.7622   0.476 0.634260
totalCellNumxAvgSingleCellVariance    2.1842     1.5135   1.443 0.148980
totalCellNumxVarianceFR              -0.6762     0.5323  -1.270 0.203957
totalCellNumxAvgTrialSpeed           -0.7475     0.9554  -0.782 0.433959
totalCellNumxVarianceSpeed            1.2069     0.6701   1.801 0.071709 .
genderxCorrelationScore               4.7113     4.3436   1.085 0.278070
genderxLickAccuracy                  -3.1125     2.0324  -1.531 0.125657
genderxLickNumber                     4.7001     2.6640   1.764 0.077679 .
genderxAvgFR                        -87.0050    14.6140  -5.954 2.62e-09 ***
genderxAvgSingleCellVariance         40.7485    11.0054   3.703 0.000213 ***
genderxVarianceFR                    -4.6695     5.1769  -0.902 0.367069
genderxAvgTrialSpeed                 -9.4424     3.8540  -2.450 0.014284 *
genderxVarianceSpeed                  4.5012     3.7641   1.196 0.231762
genotypexCorrelationScore             3.0093     2.0973   1.435 0.151338
genotypexLickAccuracy                -0.2219     1.1406  -0.195 0.845758
genotypexLickNumber                   4.0348     1.9529   2.066 0.038822 *
genotypexAvgFR                      -42.7736     8.1363  -5.257 1.46e-07 ***
genotypexAvgSingleCellVariance       37.2330     8.5842   4.337 1.44e-05 ***
genotypexVarianceFR                  -2.6952     2.6845  -1.004 0.315386
genotypexAvgTrialSpeed               -1.6597     2.0224  -0.821 0.411833
genotypexVarianceSpeed               -0.2610     1.7360  -0.150 0.880487
weight_gxCorrelationScore            -8.5846    10.1501  -0.846 0.397685
weight_gxLickAccuracy                10.7017     7.2068   1.485 0.137560
weight_gxLickNumber                 -26.7826    11.5589  -2.317 0.020501 *
weight_gxAvgFR                      193.5887    31.4637   6.153 7.61e-10 ***
weight_gxAvgSingleCellVariance     -154.5028    30.8680  -5.005 5.58e-07 ***
weight_gxVarianceFR                  18.4687    14.8475   1.244 0.213540
weight_gxAvgTrialSpeed               13.0744     6.7444   1.939 0.052555 .
weight_gxVarianceSpeed              -17.5478     8.0375  -2.183 0.029019 *
ketamine_dayxCorrelationScore         2.0090     0.6098   3.295 0.000985 ***
ketamine_dayxLickAccuracy            -0.2291     0.4973  -0.461 0.645009
ketamine_dayxLickNumber               0.6505     0.5670   1.147 0.251278
ketamine_dayxAvgFR                   -7.5420     2.7782  -2.715 0.006633 **
ketamine_dayxAvgSingleCellVariance    0.7418     2.6459   0.280 0.779199
ketamine_dayxVarianceFR              -0.9275     1.0399  -0.892 0.372456
ketamine_dayxAvgTrialSpeed            1.3230     0.7267   1.820 0.068686 .
ketamine_dayxVarianceSpeed            0.1314     0.8028   0.164 0.869944
medianCellDepthxCorrelationScore     -1.4019     0.5466  -2.565 0.010321 *
```

```
medianCellDepthxLickAccuracy            -0.3065     0.4351  -0.704 0.481208
medianCellDepthxLickNumber               0.4651     0.7274   0.639 0.522605
medianCellDepthxAvgFR                  -14.1645     2.7873  -5.082 3.74e-07 ***
medianCellDepthxAvgSingleCellVariance    8.6487     4.3498   1.988 0.046779 *
medianCellDepthxVarianceFR               0.6166     0.7940   0.777 0.437441
medianCellDepthxAvgTrialSpeed            1.5695     0.7607   2.063 0.039087 *
medianCellDepthxVarianceSpeed           -2.1309     0.9237  -2.307 0.021060 *
animalNameG2:correlationScore            1.5773     1.9963   0.790 0.429456
animalNameG3:correlationScore            1.3056     2.4116   0.541 0.588254
animalNameG4:correlationScore            0.8286     2.3078   0.359 0.719573
animalNameG5:correlationScore            1.3590     2.7886   0.487 0.626014
animalNameHCN1:correlationScore          7.7178     4.1183   1.874 0.060928 .
animalNameHCNb2:correlationScore         5.6795     3.0742   1.847 0.064681 .
animalNameHCNb4:correlationScore        14.4818     6.8450   2.116 0.034373 *
animalNameHCNd1:correlationScore         0.9176     2.0826   0.441 0.659487
animalNameHCNd2:correlationScore         4.9989     1.7497   2.857 0.004276 **
animalNameHCNe1:correlationScore         1.6307     3.1039   0.525 0.599324
animalNameHCNe2:correlationScore             NA         NA      NA       NA
animalNameHCNe3:correlationScore             NA         NA      NA       NA
animalNamenpI1:correlationScore          7.3425     2.8767   2.552 0.010697 *
animalNameG2:lickAccuracy               -1.7099     1.1916  -1.435 0.151303
animalNameG3:lickAccuracy                0.4135     0.9017   0.459 0.646531
animalNameG4:lickAccuracy               -0.1212     1.1478  -0.106 0.915904
animalNameG5:lickAccuracy                2.3511     1.7690   1.329 0.183839
animalNameHCN1:lickAccuracy             -2.1156     2.6306  -0.804 0.421274
animalNameHCNb2:lickAccuracy            -3.7346     1.6714  -2.234 0.025455 *
animalNameHCNb4:lickAccuracy            -5.7505     3.6852  -1.560 0.118654
animalNameHCNd1:lickAccuracy             0.5928     1.1011   0.538 0.590326
animalNameHCNd2:lickAccuracy            -3.2750     1.8848  -1.738 0.082279 .
animalNameHCNe1:lickAccuracy             2.7730     2.1648   1.281 0.200209
animalNameHCNe2:lickAccuracy                 NA         NA      NA       NA
animalNameHCNe3:lickAccuracy                 NA         NA      NA       NA
animalNamenpI1:lickAccuracy             -2.8775     1.1676  -2.464 0.013723 *
animalNameG2:lickNumber                 -0.1500     1.0227  -0.147 0.883379
animalNameG3:lickNumber                 -1.8930     1.2920  -1.465 0.142882
animalNameG4:lickNumber                 -3.7432     1.6083  -2.327 0.019944 *
animalNameG5:lickNumber                 -9.2212     2.5582  -3.605 0.000313 ***
animalNameHCN1:lickNumber                9.9468     4.2924   2.317 0.020488 *
animalNameHCNb2:lickNumber               2.1215     3.2220   0.658 0.510251
animalNameHCNb4:lickNumber               4.2728     8.6890   0.492 0.622893
animalNameHCNd1:lickNumber               1.6771     1.2682   1.322 0.186031
animalNameHCNd2:lickNumber               0.9823     1.4131   0.695 0.486990
animalNameHCNe1:lickNumber              -8.0085     3.3984  -2.357 0.018446 *
animalNameHCNe2:lickNumber                   NA         NA      NA       NA
animalNameHCNe3:lickNumber                   NA         NA      NA       NA
animalNamenpI1:lickNumber               -4.2673     3.6055  -1.184 0.236592
animalNameG2:avgFR                       7.4748     3.6790   2.032 0.042177 *
animalNameG3:avgFR                      23.6143     5.1125   4.619 3.86e-06 ***
animalNameG4:avgFR                      31.2290     4.7689   6.548 5.81e-11 ***
animalNameG5:avgFR                      39.7199     6.5348   6.078 1.22e-09 ***
animalNameHCN1:avgFR                   -45.6980    12.3054  -3.714 0.000204 ***
animalNameHCNb2:avgFR                  -35.6108     7.8888  -4.514 6.36e-06 ***
animalNameHCNb4:avgFR                   -9.3360    62.1295  -0.150 0.880554
animalNameHCNd1:avgFR                    9.1863     3.9552   2.323 0.020202 *
animalNameHCNd2:avgFR                  -13.7740     6.7154  -2.051 0.040256 *
animalNameHCNe1:avgFR                   56.3254     8.8355   6.375 1.83e-10 ***
animalNameHCNe2:avgFR                       NA         NA      NA       NA
animalNameHCNe3:avgFR                       NA         NA      NA       NA
animalNamenpI1:avgFR                    -6.9399     8.7446  -0.794 0.427414
animalNameG2:avgSingleCellVariance      -5.8451     7.1199  -0.821 0.411667
animalNameG3:avgSingleCellVariance     -21.4194     8.6198  -2.485 0.012958 *
animalNameG4:avgSingleCellVariance     -21.1395     8.1704  -2.587 0.009673 **
animalNameG5:avgSingleCellVariance     -43.8020     9.4607  -4.630 3.66e-06 ***
animalNameHCN1:avgSingleCellVariance    57.2169    15.6053   3.667 0.000246 ***
animalNameHCNb2:avgSingleCellVariance   26.6612    11.7217   2.275 0.022935 *
animalNameHCNb4:avgSingleCellVariance   53.1048    30.8837   1.720 0.085522 .
animalNameHCNd1:avgSingleCellVariance   -0.8131     6.4811  -0.125 0.900156
animalNameHCNd2:avgSingleCellVariance   30.6919    12.4083   2.473 0.013380 *
animalNameHCNe1:avgSingleCellVariance  -38.3291    10.3293  -3.711 0.000207 ***
animalNameHCNe2:avgSingleCellVariance       NA         NA      NA       NA
animalNameHCNe3:avgSingleCellVariance       NA         NA      NA       NA
animalNamenpI1:avgSingleCellVariance   -12.8571    16.3104  -0.788 0.430536
animalNameG2:varianceFR                  0.7852     1.8165   0.432 0.665556
animalNameG3:varianceFR                  1.3876     2.6141   0.531 0.595547
animalNameG4:varianceFR                  2.3089     2.5615   0.901 0.367378
animalNameG5:varianceFR                  4.1829     3.0171   1.386 0.165618
animalNameHCN1:varianceFR               -3.2583     4.8433  -0.673 0.501115
animalNameHCNb2:varianceFR              -3.6846     3.1546  -1.168 0.242803
animalNameHCNb4:varianceFR             -25.8202    31.0535  -0.831 0.405705
animalNameHCNd1:varianceFR               0.1037     1.9293   0.054 0.957136
animalNameHCNd2:varianceFR              -2.8652     2.7017  -1.061 0.288911
animalNameHCNe1:varianceFR               4.8279     4.0581   1.190 0.234157
```

```
animalNameHCNe2:varianceFR                  NA         NA      NA       NA
animalNameHCNe3:varianceFR                  NA         NA      NA       NA
animalNamenpI1:varianceFR                 4.5827     6.4964   0.705 0.480550
animalNameG2:avgTrialSpeed               -2.4850     1.7024  -1.460 0.144372
animalNameG3:avgTrialSpeed               -1.8267     1.9955  -0.915 0.359969
animalNameG4:avgTrialSpeed               -1.1678     1.9046  -0.613 0.539752
animalNameG5:avgTrialSpeed               -0.8649     2.2433  -0.386 0.699814
animalNameHCN1:avgTrialSpeed             -7.7469     3.0655  -2.527 0.011499 *
animalNameHCNb2:avgTrialSpeed            -6.3337     2.3588  -2.685 0.007251 **
animalNameHCNb4:avgTrialSpeed            -6.3992     4.3391  -1.475 0.140272
animalNameHCNd1:avgTrialSpeed            -0.6755     1.3358  -0.506 0.613098
animalNameHCNd2:avgTrialSpeed             0.7792     1.2862   0.606 0.544644
animalNameHCNe1:avgTrialSpeed             2.9309     2.3740   1.235 0.216977
animalNameHCNe2:avgTrialSpeed                NA         NA      NA       NA
animalNameHCNe3:avgTrialSpeed                NA         NA      NA       NA
animalNamenpI1:avgTrialSpeed             -4.7091     2.0490  -2.298 0.021549 *
animalNameG2:varianceSpeed                0.9403     2.2028   0.427 0.669485
animalNameG3:varianceSpeed               -0.8958     2.6235  -0.341 0.732764
animalNameG4:varianceSpeed               -3.0501     2.4720  -1.234 0.217244
animalNameG5:varianceSpeed               -3.5242     2.8684  -1.229 0.219219
animalNameHCN1:varianceSpeed              2.9468     4.3016   0.685 0.493319
animalNameHCNb2:varianceSpeed             3.0695     3.3371   0.920 0.357673
animalNameHCNb4:varianceSpeed           -20.5634    18.9175  -1.087 0.277034
animalNameHCNd1:varianceSpeed            -1.3491     1.9970  -0.676 0.499317
animalNameHCNd2:varianceSpeed            -4.5265     2.2934  -1.974 0.048410 *
animalNameHCNe1:varianceSpeed            -5.4418     2.5092  -2.169 0.030103 *
animalNameHCNe2:varianceSpeed                NA         NA      NA       NA
animalNameHCNe3:varianceSpeed                NA         NA      NA       NA
animalNamenpI1:varianceSpeed             -3.9996     2.8047  -1.426 0.153858
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 4989.22  on 3599  degrees of freedom
Residual deviance:  876.32  on 3438  degrees of freedom
AIC: 1200.3

Number of Fisher Scoring iterations: 12
```

Reduced dataset to match Lasso and Ridge

In [7]:

```
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")
```

```
Warning message in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :
“prediction from a rank-deficient fit may be misleading”Warning message in predict.lm(object, newdata, se.fit, scale =
1, type = if (type == :
“prediction from a rank-deficient fit may be misleading”Warning message in predict.lm(object, newdata, se.fit, scale =
1, type = if (type == :
“prediction from a rank-deficient fit may be misleading”Warning message in predict.lm(object, newdata, se.fit, scale =
1, type = if (type == :
“prediction from a rank-deficient fit may be misleading”Warning message in predict.lm(object, newdata, se.fit, scale =
1, type = if (type == :
“prediction from a rank-deficient fit may be misleading”Warning message in predict.lm(object, newdata, se.fit, scale =
1, type = if (type == :
“prediction from a rank-deficient fit may be misleading”Warning message in predict.lm(object, newdata, se.fit, scale =
1, type = if (type == :
“prediction from a rank-deficient fit may be misleading”Warning message in predict.lm(object, newdata, se.fit, scale =
1, type = if (type == :
“prediction from a rank-deficient fit may be misleading”Warning message in predict.lm(object, newdata, se.fit, scale =
1, type = if (type == :
“prediction from a rank-deficient fit may be misleading”Warning message in predict.lm(object, newdata, se.fit, scale =
1, type = if (type == :
“prediction from a rank-deficient fit may be misleading”

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

Zero-One Loss (10-fold Cross-Validation Average): 0.07703518
Accuracy (10-fold Cross-Validation Average): 0.9229648
=====================================================================
```

Basic Logistic Regression without Interaction Terms

In [8]:
```
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 [9]:
```
# 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 = model.matrix(ketBool ~ . + animalName:correlationScore
                    + animalName:lickAccuracy
                    + animalName:lickNumber
                    + animalName:avgFR
                    + animalName:avgSingleCellVariance
                    + animalName:varianceFR
                    + animalName:avgTrialSpeed
                    + animalName:varianceSpeed, data = train50A)
yA = train50A$ketBool

xB = model.matrix(ketBool ~ . + animalName:correlationScore
                    + animalName:lickAccuracy
                    + animalName:lickNumber
                    + animalName:avgFR
                    + animalName:avgSingleCellVariance
                    + animalName:varianceFR
                    + animalName:avgTrialSpeed
                    + animalName:varianceSpeed, data = train50B)
yB = train50B$ketBool
```

Lasso

In [10]:
```
# 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
```

lambda.min

In [11]:
```
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): 0.07002764
Accuracy (10-fold Cross-Validation Average): 0.9299724
======================================================================
```

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=1,lambda=lambda_1se)
    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_1se)
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.1se\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.1se

Zero-One Loss (10-fold Cross-Validation Average): 0.07403518
Accuracy (10-fold Cross-Validation Average): 0.9259648
======================================================================
```

Ridge

In [13]:
```
# 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
```

lambda.min

In [16]:
```
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): 0.07603518
Accuracy (10-fold Cross-Validation Average): 0.9239648
======================================================================
```

lambda.1se

In [17]:
```
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_1se)
    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_1se)
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.1se\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.1se

Zero-One Loss (10-fold Cross-Validation Average): 0.07703518
Accuracy (10-fold Cross-Validation Average): 0.9229648
======================================================================
```

In [ ]: