MS&E 226: Mini-project part 2

Recording from the Brains of Mice in Virtual Reality to Understand the Effects of Ketamine on Spatial Memory and Navigation

Erin Brown & Francis Kei Masuda

December 6, 2019

Part 2

Brief Re-Introduction

Since the 1970s, ketamine has been commonly used in the clinics as a rapid-acting dissociative anesthetic drug. Yet, despite frequent use and much scientific and clinical attention, ketamine's mechanism of action on neurological circuitry remains poorly understood. Our dataset examines how ketamine affects spatial memory and navigation. It is known that the region of the brain known as the medial entorhinal cortex formation is responsible for spatial memory and navigation. Our dataset includes information about the electrical activity from individual neurons in the medial entorhinal cortex of mice running in virtual reality hallways in the presence and absence of ketamine. Additionally, there is a hypothesis that ketamine is disrupting neural activity by acting on a specific pace-making ion-channel known as HCN1. The shape of our data matrix shape is 5000 trials x 20 covariates. Each trial is one run down the 400cm long VR hallway. We took 100 trials from each of the 50 recording sessions to form 5000 rows of trials. The covariate columns are: (1) Animal Name (2) Session Date (3) Trial Number (4) Total Number of Cells Recorded in the Session (5) Gender of Mouse (6) Genotype – Wild Type or HCN1 knockout animal (7) Weight of the animal in grams (8) Number of days the animal has been exposed to ketamine. (9) Correlation score (10) Lick Accuracy (11) Lick Number (12) Average Firing Rate (13) Average Single Cell Variance (14) Variance Firing Rate (15) Average Trial Speed (17) Variance Speed (18) Median Cell Depth (19) Time Since Ketamine Injection (20) Ketamine Administered. We choose to use 'Time Since Ketamine Injection' as our continuous response variable for the regression task. This allows us to ask the question — *given information about the neural activity and animal behavior held within our covariates, can we predict how long it has been in seconds since ketamine was administered to the animal?* We chose to use the bool 'Ketamine Administered' as our binary response variable. This allows us to ask a slightly different question — *given information about neural activity and animal behavior held within our covariates, can we classify the trial as a trial under the influence of ketamine or a normal trial*?

Changes from Part I

Classification model selection

For our classification model in Part I, we considered only logistic regression with L2 regularization over the set of covariates including interaction terms, while in this updated search we considered both lasso and ridge regression via *glmnet* over the same set of covariates, as well as logistic regression without any regularization. We split the training set in half, using the built-in cross validation function of *glmnet* on one half to determine the optimal regularization parameter λ, which we then fixed by selecting the λ that minimized binomial deviance. We then used the other half of the training set to obtain model coefficients as well as an estimate of test error using 10-fold cross validation. We found that the best lasso (with $\lambda = 0.0003$) outperformed the best ridge (with $\lambda = 0.0299$) regression, achieving zero-one losses of 0.092 and 0.126 (or accuracies of 0.908 and 0.874) respectively. Logistic regression without regularization achieved a zero-one loss of 0.091 (accuracy of 0.909). As logistic regression without regularization reaches the lowest zero-one loss and is in some ways simpler to work with for inference than the comparably performing lasso model, we select that model as our best model.

1. Prediction on the Test Set

The best regression model from Part I was a standard linear regression over the full covariate set augmented with interaction terms, which outperformed both ridge and lasso regression on the same covariates. From cross-validation on the training set, we estimated the test RMSE and scaled RMSE as 1511.8 and 0.1139 respectively. Now, evaluating on the test set held out yields actual test RMSE of

1544.0 with scaled RMSE of 0.1163, so our estimate from the training set was an underestimate of the actual test error but still quite close.

The selected prediction model as detailed above was logistic regression on the augmented covariate set including interaction terms with no regularization. Test error was estimated by 10-fold cross-validation on the training set as a zero-one loss of 0.09 (accuracy of 0.909). Evaluating on the test set yields a zero-one loss of 0.135 (accuracy of 0.865), suggesting that the model without regularization is prone to overfitting.

2. Inference

(a)Coefficient Statistical Significance

On the training set, out of 62 covariates plus an intercept term, 19 were found to be significant at the 0.001 level, 8 at the 0.01 level, and 4 at the 0.05 level. The p-values provide the probability of observing a test statistic at least as extreme as the one observed if the null hypothesis (that the coefficient is equal to zero, having no effect on the response variable) were true. This means that for a given significance level, we expect to incorrectly reject the null hypothesis in no more than the fraction of our tests equal to the p-value on average. and for those covariates, we reject the hypothesis that it has no correlation with ketamine administration. Some of these found to be significant, even at the 0.001 level, such as *medianCellDepthxAvgTrialSpeed*, we do not expect to be significant based on our understanding of the experiment, while others, such as *avgSingleCellVariance* we do expect. We find in particular that many of the interaction terms with *genotype* are significant, which lends some credence to one of our experimental hypotheses that the two genotypes considered are affected by ketamine differently. However, there are enough covariates found significant that we find it likely that artifacts in the training data set have caused us to spuriously find some covariates significant that may not be in actuality. Examining coefficient significance on the test data set can give us a better understanding.

(b)Test Data Performance

Indeed, we find fewer statistically significant covariates on the test data set: 6 at the 0.001 level, 2 at the 0.01 level, and 9 at the 0.05 level. In this case, though, many covariates that we think should be significant are not found to be, such as *avgFR* and *varianceSpeed*. While it is possible that these covariates genuinely are uncorrelated with ketamine administration within the context of this model with all of the other terms, it is also possible that there is enough noise in our dataset that we are unable to properly identify all significant terms. The less than excellent performance (under 90%) of our classification model on the test set suggests that this is certainly a possibility to consider.

Note that to compensate for our fairly large number of covariates, we could confine our consideration of “statistically significant” covariates to those found to be significant at the 0.001 level or better (comparable to using the Bonferroni correction), but for our experimental purposes, we wish to err on the side of considering more covariates statistically significant rather than fewer, as we can take statistical significance in this model to be a sign that the effect of ketamine on that covariate is worth investigating further.

(c)Bootstrapping

We plotted coefficient estimate and confidence intervals for each of the covariate coefficients plus intercept term. These were generated in one case by the standard regression output (Figure 1) and in the other by bootstrapping (Figure 2). The intervals shown in these plots are for 95% confidence and are bounded by the coefficient estimate plus or minus 1.96 times the standard error. Note that the intervals are much smaller for the bootstrap case, as expected. Interestingly, in some cases, the coefficient estimate from the standard regression output falls outside of the 95% bootstrap confidence interval. As there are more of these cases (>5%) than we might expect statistically, this could indicate that there is bias in our training set that is being compounded by the bootstrap process.

(d)Covariate Selection

What covariates we chose definitely had a large effect on which coefficients were considered significant. The different covariates definitely affected the model performance: in our simplest logistic regression model (model 2) without interaction terms the model had a 10-fold cross-validation average Zero-One Loss of 0.14 and an 10-fold cross-validation average accuracy of 0.86 on the test set; in contrast, our Logistic Regression Model with Interaction Terms (model 1) had a 10-fold cross-validation average Zero-One Loss of 0.09 and an 10-fold cross-validation average accuracy of 0.91 on the test set.

Additionally, not only did the different covariate sets affect the performance of the model, but also the different covariates notably affected the which coefficients where found to be significant. One of the most interesting changes revolves around the mouse's genotype covariate. On its own, a mouse's genotype is not a significant predictor of whether or not the mouse had been administered ketamine. However, when combined with interaction terms, the coefficient for genotype becomes very (***) significant. This makes intuitive sense because on its own the genetic make-up of a mouse has no relationship with whether or not the mouse has been administered ketamine — yet, the genetic make-up of a mouse is intricately linked to how ketamine affects all of the other major covariates such as average mouse speed, it's lick accuracy on the spatial navigation task, the firing rate variance of neurons. Another interesting case study of how adding covariates affect coefficient significance was the total cell number covariate. Similarly, to genotype, on its own the gender of a mouse has no significant relationship with ketamine administration. Yet, when paired with interaction terms, the gender of the mouse recorded affects how the model interprets values like the average firing rate of neurons and the average variance of single cells firing rate. Both of these differences make sense because these categorical covariates are definitely subsets of the population that should respond to the administration of ketamine slightly differently.

(e)Potential problems with the analysis

There are several problems with the analysis that have to be kept in mind. For example, we have to be wary that multiple hypothesis testing might be affecting model interpretation. Large numbers of covariates increases the chances that coefficients are spuriously found to be significant. Since our models use anywhere between 14 and 63 covariates, this is definitely a problem. A Bonferroni correction would definitely be valuable in this case because that is a correction that tries to lower false significance positives by lowering the alpha values when many statistical tests are being performed simultaneously. We are also potentially concerned about the fact that we may be missing a crucial covariate in training our model. For example we initially decided to drop 'mouse name', which could potentially be a crucial covariate that improves the model performance in a similar why how grouping data by users helped Netflix improve its models. We are also very concerned about the effect that collinearity is having on our models. Collinearity, or when covariates have a linear relationship, is definitely visible in our data set based on the facet grid comparison from part 1 of this project. This could be having the unfortunate effect of increasing the variance of a subset of regression coefficients and even flipping the sign of regression coefficients. Finally, we are not particularly worried about post-selection inference since our model performance on the test set and the estimated performance from the training set were similar.

(f)Causal Interpretation?

We cannot infer any causal relationships from our model due to its very formulation as all of the covariates are either non-predictive metadata on their own or measurements made at a later time than the event of ketamine administration that we are trying to predict.

3. Discussion

(a) Model Practicality

We created our classification model asking one basic question: *given information about neural activity and animal behavior held within our covariates, can we classify the trial as a trial under the influence of ketamine or a normal trial*? Our models perform admirably—especially, considering that the model can make this classification on the recorded activity of thousands of neurons out of the billions of neurons that exist in the brain.

There are two levels at which to think about the practicality of this model. On one hand, at a surface level, this classification model cannot be used in a real-world setting. Rarely, if ever, will a mouse present us with its neural activity and ask us to tell it if it is under the influence of ketamine or not. While it is slightly more believable that we could one day in the future be given an electrical brain scan of a human patient and be asked by the DEA to predict if that patient is high on ketamine, it is impossible to use a model built on mouse data to predict the drug status of a human.

On the other hand, this model is of practical use for scientists trying to understand how ketamine impacts neural activity in the brain. It was not immediately apparent from the outset that the medial entorhinal cortex (the structure from which the neural activity was recorded) would contain information about whether or not the animal has been dosed with ketamine. Thus, the fact that we were able to create a model that is given activity from neurons in the medial entorhinal cortex and behavioral data and is successfully able to classify with greater than 90% accuracy whether or not an animal is under the influence of ketamine is practical information. Additionally, understanding which coefficients are significantly impacting the model's ability to correctly classify is practical information. Thus, this model is best used for inference.

That being said, there are important caveats to interpreting this model. With the design of our model, it is very difficult to make any causal claims. This is due to the fact that we are trying to predict whether or not an animal has been exposed to ketamine based on neural activity and behavioral data. It would be illogical to infer that the neural/behavioral coefficients caused ketamine exposure. The experimental manipulation was the intraperitoneal administration of the drug, so in fact a more logical chain of events is that ketamine caused the changes in neural and behavioral activity, not the other way around.

(b) Model Performance Over Time

We believe that this model's performance over time should remain relatively stable as long as the all of the parameters of the experiment remain stable. Essentially, as long as we're sampling from C57/Black 6 mice running in the same 400cm virtual reality linear track, we believe that our ketamine classification model will perform reasonably well. If any of the parameters changed (different species, different strain of mouse, different virtual reality environment, different task, different drug, different drug dose, etc.) then the model will probably have to be refit onto new data.

(c) Model Assumptions/Data Analysis Choices

While the project requests that we comment on what we might want to tell a manager or a client, there is a clear analogous situation in the world of neuroscience — publication in a scientific journal. The idealized purpose of a research paper is to not only convey a finding or a result, but to also enable any reasonable scientist to reproduce the steps necessary to confirm your finding. In order to make sense of and reproduce the findings necessary for this model, many assumptions and choices will need to be conveyed to the reader about both the experimental set up as well as the data analysis. Of course, the reader should know that the models are dependent on the fact that the mice are head-fixed, that they are running on a wheel in a one-dimensional virtual environment, that we are recording when they lick for a water reward, and that we are recording from neurons in the part of the brain known as the medial entorhinal cortex. The models also assume that the mice are receiving 25mg/kg of ketamine delivered intraperitoneally.

With regards to the data preprocessing, it is crucial to know that the classification model and the regression model use different subsets of the data. The regression model is trained on 100 trials after ketamine was injected into the mouse so that we could predict the time since ketamine injection. The classification model was trained on the 100 trials before and after ketamine was administered. Also it is important to know that our models often performed best with many interaction terms added. Covariates such as genotype (whether or not the animal has been genetically modified or not) do not significantly impact model performance on their own, but when combined with other covariate terms, it becomes highly significant. Based on our performance on the test set, our classification model does not vulnerable to overfitting within the context of this experiment.

In the context of inference, multiple hypothesis testing is definitely affecting our interpretation of the model. We have to remain aware of that fact that with 14 independent covariates or up to 63 covariates including interaction terms, the chances that we will spuriously find significant coefficients is significantly increased. In the future, we may want to be more stringent in how we define significance by using bonferroni or sth. We are not very concerned with post-selection inference since our model performance on the test set and the estimated performance from the training set were similar. We are also potentially concerned about the fact that we may be missing a crucial covariate in training our model. Since we do not know how the neural circuitry in brain works, and we are recording randomly sampled neurons within a structure, there could be an essential covariate missing that is explanatory. If this is the case it is possible that we could use imputation to correct for this, which takes advantage of the idea that we should be able to estimate an important missing feature from existing data. Additionally, collinearity is definitely influencing the models that we have constructed. Since we are interested in inference, collinearity has the effect of inflating the variance of a subset of regression coefficients and even flipping the sign of regression coefficients. A more rigorous removal of colinear coefficients could help with this.

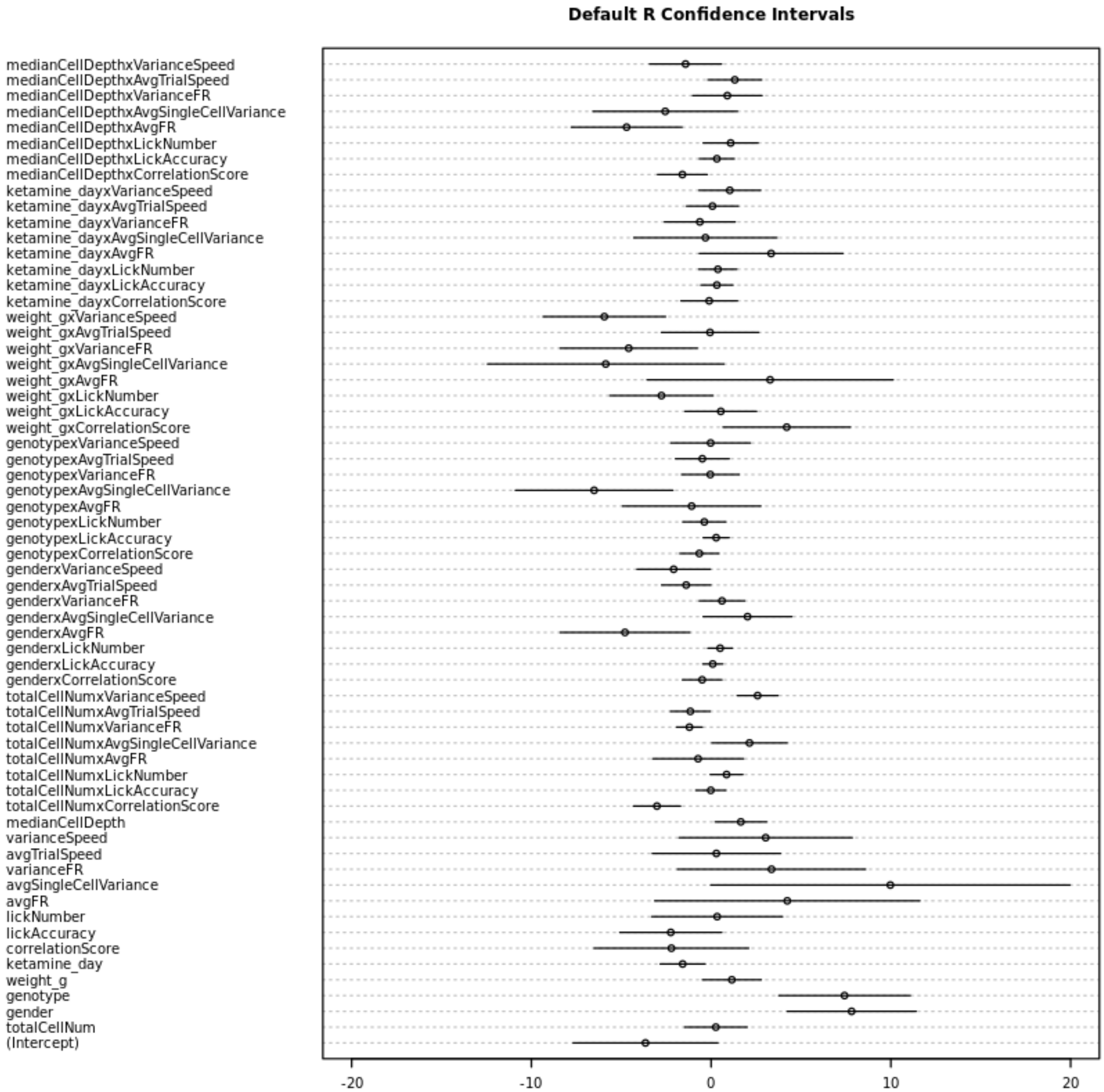
(d)How would you change the data collection process? What covariates are you missing?

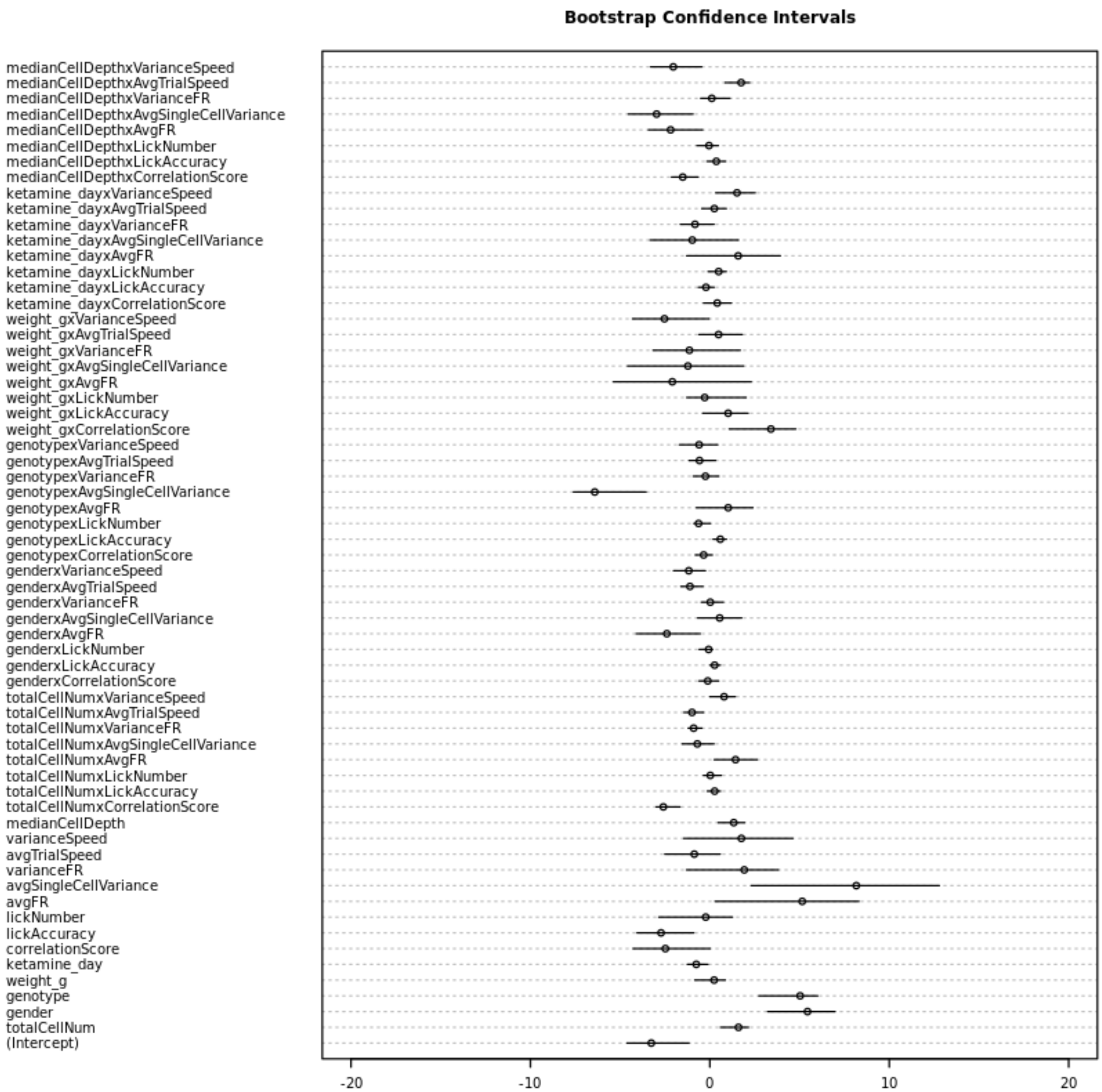
If we could change the data collection process, we would try to add more potentially relevant covariates. While it is experimentally difficult, it would be very useful if we could get a covariate with cell type information about the neural data. The neurons in the brain are not homogenous — there exist different morphological subtypes (pyramidal neurons, basket neurons, interneurons etc.), different activity subtypes (excitatory neurons, inhibitory neurons), and different functional subtypes (grid cells, head direction cells, speed cells, border cells, etc). How ketamine affects these different neural subtypes would definitely be both informative and affect how the model performs. Additionally, another covariate that would be good to add to the dataset is mouse sensitivity to ketamine. Based on each individual mouse's metabolism, every mouse responds to ketamine differently. A covariate that measured this would be very important and add lots of good information to the model.

(e)How would you attack the dataset differently?

The main aspect of this whole analysis process that we would do differently if we could rewind time would be to restructure the question we were asking before building our models. Our current model asks the question is it possible to predict whether or not an animal is under the influence of ketamine based on the neural and behavioral data. However, this prevents us from making any causal inferences. If we had structured the question such that we are predicting a property of neural firing such as average firing rate from with administration of ketamine as a covariate, then it would be possible to make the claim that ketamine causes a change in behavioral data or in neural data. Another thing that we could have done differently is to try lasso instead because it might have helped get better covariate significances.

Appendix


Default R Confidence Intervals
medianCellDepthxVarianceSpeed
medianCellDepthxAvgTrialSpeed
medianCellDepthxVarianceFR
medianCellDepthxAvgSingleCellVariance
medianCellDepthxAvgFR
medianCellDepthxLickNumber
medianCellDepthxLickAccuracy
medianCellDepthxCorrelationScore
ketamine_dayxVarianceSpeed
ketamine_dayxAvgTrialSpeed
ketamine_dayxVarianceFR
ketamine_dayxAvgSingleCellVariance
ketamine_dayxAvgFR
ketamine_dayxLickNumber
ketamine_dayxLickAccuracy
ketamine_dayxCorrelationScore
weight_gxVarianceSpeed
weight_gxAvgTrialSpeed
weight_gxVarianceFR
weight_gxAvgSingleCellVariance
weight_gxAvgFR
weight_gxLickNumber
weight_gxLickAccuracy
weight_gxCorrelationScore
genotypexVarianceSpeed
genotypexAvgTrialSpeed
genotypexVarianceFR
genotypexAvgSingleCellVariance
genotypexAvgFR
genotypexLickNumber
genotypexLickAccuracy
genotypexCorrelationScore
genderxVarianceSpeed
genderxAvgTrialSpeed
genderxVarianceFR
genderxAvgSingleCellVariance
genderxAvgFR
genderxLickNumber
genderxLickAccuracy
genderxCorrelationScore
totalCellNumxVarianceSpeed
totalCellNumxAvgTrialSpeed
totalCellNumxVarianceFR
totalCellNumxAvgSingleCellVariance
totalCellNumxAvgFR
totalCellNumxLickNumber
totalCellNumxLickAccuracy
totalCellNumxCorrelationScore
medianCellDepth
varianceSpeed
avgTrialSpeed
varianceFR
avgSingleCellVariance
avgFR
lickNumber
lickAccuracy
correlationScore
ketamine_day
weight_g
genotype
gender
totalCellNum
(Intercept)
-20
-10
0
10
20


Bootstrap Confidence Intervals
medianCellDepthxVarianceSpeed
medianCellDepthxAvgTrialSpeed
medianCellDepthxVarianceFR
medianCellDepthxAvgSingleCellVariance
medianCellDepthxAvgFR
medianCellDepthxLickNumber
medianCellDepthxLickAccuracy
medianCellDepthxCorrelationScore
ketamine_dayxVarianceSpeed
ketamine_dayxAvgTrialSpeed
ketamine_dayxVarianceFR
ketamine_dayxAvgSingleCellVariance
ketamine_dayxAvgFR
ketamine_dayxLickNumber
ketamine_dayxLickAccuracy
ketamine_dayxCorrelationScore
weight_gxVarianceSpeed
weight_gxAvgTrialSpeed
weight_gxVarianceFR
weight_gxAvgSingleCellVariance
weight_gxAvgFR
weight_gxLickNumber
weight_gxLickAccuracy
weight_gxCorrelationScore
genotypexVarianceSpeed
genotypexAvgTrialSpeed
genotypexVarianceFR
genotypexAvgSingleCellVariance
genotypexAvgFR
genotypexLickNumber
genotypexLickAccuracy
genotypexCorrelationScore
genderxVarianceSpeed
genderxAvgTrialSpeed
genderxVarianceFR
genderxAvgSingleCellVariance
genderxAvgFR
genderxLickNumber
genderxLickAccuracy
genderxCorrelationScore
totalCellNumxVarianceSpeed
totalCellNumxAvgTrialSpeed
totalCellNumxVarianceFR
totalCellNumxAvgSingleCellVariance
totalCellNumxAvgFR
totalCellNumxLickNumber
totalCellNumxLickAccuracy
totalCellNumxCorrelationScore
medianCellDepth
varianceSpeed
avgTrialSpeed
varianceFR
avgSingleCellVariance
avgFR
lickNumber
lickAccuracy
correlationScore
ketamine_day
weight_g
genotype
gender
totalCellNum
(Intercept)
-20
-10
0
10
20

Linear Regression

Predicting time since ketamine administration

In [1]:
```
# packages
import os
import pandas as pd
import math
from scipy import io
import numpy as np
from numpy import squeeze
from sklearn import linear_model
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import KFold
from sklearn.metrics import zero_one_loss
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
from matplotlib import style
style.use('ggplot')
```

Load in data and perform checks

In [2]:
```
allData = pd.read_csv('postKetamineTable.csv')
```

In [3]:
```
allData.keys()
```

Out[3]:
```
Index(['animalName', 'sessionDate', 'trialNum', 'totalCellNum', 'gender',
       'genotype', 'weight_g', 'ketamine_day', 'correlationScore',
       'lickAccuracy', 'lickNumber', 'avgFR', 'avgSingleCellVariance',
       'varianceFR', 'avgTrialSpeed', 'varianceSpeed', 'medianCellDepth',
       'timeSinceKetamine', 'ketamineAdministered'],
      dtype='object')
```

In [4]:
```
# Check size information
print("num_cols =",len(allData.keys()))
print("num_rows =",len(allData))

# Check for duplicate rows
print("num_dup =",np.sum(pd.DataFrame.duplicated(allData)))
```

```
num_cols = 19
num_rows = 5000
num_dup = 0
```

In [5]:
```
# Check for NaNs and see where they are coming from
np.sum(pd.isna(allData))
```

Out[5]:
```
animalName               0
sessionDate              0
trialNum                 0
totalCellNum             0
gender                   0
genotype                 0
weight_g                 0
ketamine_day             0
correlationScore         0
lickAccuracy             0
lickNumber               0
avgFR                    0
avgSingleCellVariance    0
varianceFR               5
avgTrialSpeed            0
varianceSpeed            0
medianCellDepth          0
timeSinceKetamine        0
ketamineAdministered     0
dtype: int64
```

In [6]:
```
# Remove any rows with nans
allDataNN = pd.DataFrame.dropna(allData,'index')
print("After Drop NaN")
print("num_rows =",len(allDataNN))
```

```
After Drop NaN
num_rows = 4995
```

In [7]:
```
ketBool = allDataNN['ketamineAdministered']
timeSinceKetamine = allDataNN['timeSinceKetamine']
sessionDate = allDataNN['sessionDate']
trialNum = allDataNN['trialNum']
neuralData = allDataNN[['animalName', 'totalCellNum',
       'gender', 'genotype', 'weight_g',
       'ketamine_day', 'correlationScore', 'lickAccuracy',
       'lickNumber', 'avgFR', 'avgSingleCellVariance',
       'varianceFR', 'avgTrialSpeed', 'varianceSpeed',
       'medianCellDepth']]
```

In [8]:
```
# Convert categorical columns
le = LabelEncoder()
neuralData_LE = neuralData.copy()
neuralData_LE['animalName'] = le.fit_transform(neuralData_LE['animalName'])
neuralData_LE['gender'] = le.fit_transform(neuralData_LE['gender'])
neuralData_LE['genotype'] = le.fit_transform(neuralData_LE['genotype'])
features = list(neuralData_LE.keys())
```

In [9]:
```
# Standardize data
stdNeuralData = StandardScaler().fit_transform(neuralData_LE)
```

```
/home/browne/anaconda3/lib/python3.7/site-packages/sklearn/preprocessing/data.py:625: DataConversionWarning: Data with
input dtype int64, float64 were all converted to float64 by StandardScaler.
  return self.partial_fit(X, y)
/home/browne/anaconda3/lib/python3.7/site-packages/sklearn/base.py:462: DataConversionWarning: Data with input dtype in
t64, float64 were all converted to float64 by StandardScaler.
  return self.fit(X, **fit_params).transform(X)
```

In [10]:
```
# Split off test set for later
X, X_ho, y, y_ho = train_test_split(stdNeuralData,timeSinceKetamine.values.ravel(), test_size=0.2, random_state = 2019)
```

In [11]:
```
# Split for cross validation, use 10 folds
num_folds = 10
XA = np.array(X)
yA = np.array(y)
X_train = []
X_test = []
y_train = []
y_test = []
kf = KFold(n_splits=num_folds)
for train_index, test_index in kf.split(XA, yA):
    X_train.append(XA[train_index])
    X_test.append(XA[test_index])
    y_train.append(yA[train_index])
    y_test.append(yA[test_index])
```

In [12]:
```
# Run basic linreg model on full train set, check performance against train
model = linear_model.LinearRegression(fit_intercept=True,normalize=False,copy_X=True,n_jobs=None).fit(X,y)
```

In [13]:
```
print("Intercept: ",model.intercept_)
print(features,model.coef_)
#print(model.coef_)
y_pred = model.predict(X)
rmse = np.sqrt(mean_squared_error(y,y_pred))
r2 = r2_score(y,y_pred)
print("RMSE: ",rmse)
print("R2:",r2)
```

```
Intercept:  1881.3040918494605
['animalName', 'totalCellNum', 'gender', 'genotype', 'weight_g', 'ketamine_day', 'correlationScore', 'lickAccuracy', 'l
ickNumber', 'avgFR', 'avgSingleCellVariance', 'varianceFR', 'avgTrialSpeed', 'varianceSpeed', 'medianCellDepth'] [  89.
37637717  -48.31423738 -388.77287316  233.85776474  580.47450308
 -334.13598052  261.54986522  304.54847766   24.61109336 -597.46522638
  262.84684555  244.33634822 -395.38514629  314.91671046 -387.04573926]
RMSE:  1879.9189130710613
R2: 0.2643830933222706
```

In [14]:
```
scaled_RMSE = rmse/(max(y)-min(y))
print(scaled_RMSE)
```

```
0.14169716149281847
```

In [15]:
```
rmse_cv = []
for i in range(0,num_folds):
    model = linear_model.LinearRegression(fit_intercept=True,normalize=False,copy_X=True,n_jobs=None).fit(X_train[i],y_
train[i])
    y_pred = model.predict(X_test[i])
    rmse_cv.append(np.sqrt(mean_squared_error(y_test[i],y_pred)))
print("Average RMSE across Folds:",np.mean(rmse_cv))
print("Average Scaled RMSE across Folds:",np.mean(rmse_cv)/(max(y)-min(y)))
```

```
Average RMSE across Folds: 1883.6861818752764
Average Scaled RMSE across Folds: 0.1419811159189515
```

Now let's try with some second order interaction terms

In [16]:
```
AugData = neuralData_LE.copy()
AugData.keys()
```

Out[16]:
```
Index(['animalName', 'totalCellNum', 'gender', 'genotype', 'weight_g',
       'ketamine_day', 'correlationScore', 'lickAccuracy', 'lickNumber',
       'avgFR', 'avgSingleCellVariance', 'varianceFR', 'avgTrialSpeed',
       'varianceSpeed', 'medianCellDepth'],
      dtype='object')
```

In [17]:
```
primaryF = ['correlationScore', 'lickAccuracy', 'lickNumber',
       'avgFR', 'avgSingleCellVariance', 'varianceFR', 'avgTrialSpeed',
       'varianceSpeed']
secondaryF = ['animalName', 'totalCellNum', 'gender', 'genotype', 'weight_g',
       'ketamine_day','medianCellDepth']
```

In [18]:
```
AugData['animalNamexCorrelationScore'] = AugData['animalName']*AugData['correlationScore']
AugData['animalNamexLickAccuracy'] = AugData['animalName']*AugData['lickAccuracy']
AugData['animalNamexLickNumber'] = AugData['animalName']*AugData['lickNumber']
AugData['animalNamexAvgFR'] = AugData['animalName']*AugData['avgFR']
AugData['animalNamexAvgSingleCellVariance'] = AugData['animalName']*AugData['avgSingleCellVariance']
AugData['animalNamexVarianceFR'] = AugData['animalName']*AugData['varianceFR']
AugData['animalNamexAvgTrialSpeed'] = AugData['animalName']*AugData['avgTrialSpeed']
AugData['animalNamexVarianceSpeed'] = AugData['animalName']*AugData['varianceSpeed']
```

In [19]:
```
AugData['totalCellNumxCorrelationScore'] = AugData['totalCellNum']*AugData['correlationScore']
AugData['totalCellNumxLickAccuracy'] = AugData['totalCellNum']*AugData['lickAccuracy']
AugData['totalCellNumxLickNumber'] = AugData['totalCellNum']*AugData['lickNumber']
AugData['totalCellNumxAvgFR'] = AugData['totalCellNum']*AugData['avgFR']
AugData['totalCellNumxAvgSingleCellVariance'] = AugData['totalCellNum']*AugData['avgSingleCellVariance']
AugData['totalCellNumxVarianceFR'] = AugData['totalCellNum']*AugData['varianceFR']
AugData['totalCellNumxAvgTrialSpeed'] = AugData['totalCellNum']*AugData['avgTrialSpeed']
AugData['totalCellNumxVarianceSpeed'] = AugData['totalCellNum']*AugData['varianceSpeed']
```

In [20]:
```
AugData['genderxCorrelationScore'] = AugData['gender']*AugData['correlationScore']
AugData['genderxLickAccuracy'] = AugData['gender']*AugData['lickAccuracy']
AugData['genderxLickNumber'] = AugData['gender']*AugData['lickNumber']
AugData['genderxAvgFR'] = AugData['gender']*AugData['avgFR']
AugData['genderxAvgSingleCellVariance'] = AugData['gender']*AugData['avgSingleCellVariance']
AugData['genderxVarianceFR'] = AugData['gender']*AugData['varianceFR']
AugData['genderxAvgTrialSpeed'] = AugData['gender']*AugData['avgTrialSpeed']
AugData['genderxVarianceSpeed'] = AugData['gender']*AugData['varianceSpeed']
```

In [21]:
```
AugData['genotypexCorrelationScore'] = AugData['genotype']*AugData['correlationScore']
AugData['genotypexLickAccuracy'] = AugData['genotype']*AugData['lickAccuracy']
AugData['genotypexLickNumber'] = AugData['genotype']*AugData['lickNumber']
AugData['genotypexAvgFR'] = AugData['genotype']*AugData['avgFR']
AugData['genotypexAvgSingleCellVariance'] = AugData['genotype']*AugData['avgSingleCellVariance']
AugData['genotypexVarianceFR'] = AugData['genotype']*AugData['varianceFR']
AugData['genotypexAvgTrialSpeed'] = AugData['genotype']*AugData['avgTrialSpeed']
AugData['genotypexVarianceSpeed'] = AugData['genotype']*AugData['varianceSpeed']
```

In [22]:
```
AugData['weight_gxCorrelationScore'] = AugData['weight_g']*AugData['correlationScore']
AugData['weight_gxLickAccuracy'] = AugData['weight_g']*AugData['lickAccuracy']
AugData['weight_gxLickNumber'] = AugData['weight_g']*AugData['lickNumber']
AugData['weight_gxAvgFR'] = AugData['weight_g']*AugData['avgFR']
AugData['weight_gxAvgSingleCellVariance'] = AugData['weight_g']*AugData['avgSingleCellVariance']
AugData['weight_gxVarianceFR'] = AugData['weight_g']*AugData['varianceFR']
AugData['weight_gxAvgTrialSpeed'] = AugData['weight_g']*AugData['avgTrialSpeed']
AugData['weight_gxVarianceSpeed'] = AugData['weight_g']*AugData['varianceSpeed']
```

In [23]:
```
AugData['ketamine_dayxCorrelationScore'] = AugData['ketamine_day']*AugData['correlationScore']
AugData['ketamine_dayxLickAccuracy'] = AugData['ketamine_day']*AugData['lickAccuracy']
AugData['ketamine_dayxLickNumber'] = AugData['ketamine_day']*AugData['lickNumber']
AugData['ketamine_dayxAvgFR'] = AugData['ketamine_day']*AugData['avgFR']
AugData['ketamine_dayxAvgSingleCellVariance'] = AugData['ketamine_day']*AugData['avgSingleCellVariance']
AugData['ketamine_dayxVarianceFR'] = AugData['ketamine_day']*AugData['varianceFR']
AugData['ketamine_dayxAvgTrialSpeed'] = AugData['ketamine_day']*AugData['avgTrialSpeed']
AugData['ketamine_dayxVarianceSpeed'] = AugData['ketamine_day']*AugData['varianceSpeed']
```

In [24]:
```
AugData['medianCellDepthxCorrelationScore'] = AugData['medianCellDepth']*AugData['correlationScore']
AugData['medianCellDepthxLickAccuracy'] = AugData['medianCellDepth']*AugData['lickAccuracy']
AugData['medianCellDepthxLickNumber'] = AugData['medianCellDepth']*AugData['lickNumber']
AugData['medianCellDepthxAvgFR'] = AugData['medianCellDepth']*AugData['avgFR']
AugData['medianCellDepthxAvgSingleCellVariance'] = AugData['medianCellDepth']*AugData['avgSingleCellVariance']
AugData['medianCellDepthxVarianceFR'] = AugData['medianCellDepth']*AugData['varianceFR']
AugData['medianCellDepthxAvgTrialSpeed'] = AugData['medianCellDepth']*AugData['avgTrialSpeed']
AugData['medianCellDepthxVarianceSpeed'] = AugData['medianCellDepth']*AugData['varianceSpeed']
```

In [25]:
```
# Standardize data
stdNeuralDataAug = StandardScaler().fit_transform(AugData)
```

```
/home/browne/anaconda3/lib/python3.7/site-packages/sklearn/preprocessing/data.py:625: DataConversionWarning: Data with
input dtype int64, float64 were all converted to float64 by StandardScaler.
  return self.partial_fit(X, y)
/home/browne/anaconda3/lib/python3.7/site-packages/sklearn/base.py:462: DataConversionWarning: Data with input dtype in
t64, float64 were all converted to float64 by StandardScaler.
  return self.fit(X, **fit_params).transform(X)
```

In [26]:
```
# Split off test set for later
X, X_ho, y, y_ho = train_test_split(stdNeuralDataAug,timeSinceKetamine.values.ravel(), test_size=0.2, random_state=2019
)
```

In [27]:
```
# Split for cross validation, use 10 folds
num_folds = 10
XA = np.array(X)
yA = np.array(y)
X_train = []
X_test = []
y_train = []
y_test = []
kf = KFold(n_splits=num_folds)
for train_index, test_index in kf.split(XA, yA):
    X_train.append(XA[train_index])
    X_test.append(XA[test_index])
    y_train.append(yA[train_index])
    y_test.append(yA[test_index])
```

In [28]:
```
# Run basic linreg model on full train set, check performance against train
model = linear_model.LinearRegression(fit_intercept=True,normalize=False,copy_X=True,n_jobs=None).fit(X,y)

print("Intercept: ",model.intercept_)
print(features,model.coef_)
#print(model.coef_)
y_pred = model.predict(X)
rmse = np.sqrt(mean_squared_error(y,y_pred))
r2 = r2_score(y,y_pred)
print("RMSE: ",rmse)
print("R2:",r2)

scaled_RMSE = rmse/(max(y)-min(y))
print(scaled_RMSE)
```

```
Intercept:  1882.209448651285
['animalName', 'totalCellNum', 'gender', 'genotype', 'weight_g', 'ketamine_day', 'correlationScore', 'lickAccuracy', 'l
ickNumber', 'avgFR', 'avgSingleCellVariance', 'varianceFR', 'avgTrialSpeed', 'varianceSpeed', 'medianCellDepth'] [ 1.32
202873e+03 -8.91679131e+02 -2.06782434e+03  6.59351804e+02
  7.64377410e+02 -2.08632125e+03 -5.32902729e+02  1.08121105e+03
 -1.41105840e+01 -3.37318052e+03  1.03249755e+03  1.13663558e+03
 -1.89808003e+03  7.79734372e+01 -1.59342695e+03  1.83861714e+02
  1.52829870e+02 -1.45411245e+02 -7.72091371e+02 -1.79495421e+02
  2.49282418e+02 -9.68636013e+02 -4.68015392e+02  3.69227271e+02
 -3.75314462e+02 -3.27108339e+02  3.96505917e+02  1.92277904e+02
  4.20441858e+01  3.53520147e+02  7.28198961e+01 -8.83274314e+02
  3.05970931e+02  3.55601020e+02 -1.86602325e+03  3.77074704e+03
 -6.18993573e+02  1.38501951e+03 -3.65147648e+02 -9.43761107e+01
  1.54665444e+02  4.59137890e+01 -1.69314230e+03  2.34941560e+03
 -7.38502618e+02  1.29824145e+02 -3.17202563e+02  8.41440984e+02
 -6.30467051e+02  3.52670997e+00  1.74618314e+03 -5.32068004e+03
  1.12276741e+03  7.83081844e+02  9.69140010e+02  3.00707198e+02
 -1.99411562e+02  6.39908126e+01  2.68099617e+03 -5.74977686e+02
 -5.57100624e+02  8.75358993e+02 -5.86782274e+02 -4.18856382e+01
 -2.18915645e+02  2.33213773e+02  1.21191515e+03  2.21624222e+03
 -6.93099690e+02  1.13823488e+02  4.27475283e+02]
RMSE:  1513.3468605647722
R2: 0.5232942086644587
0.11406712970709422
```

In [29]:
```
rmse_cv = []
for i in range(0,num_folds):
    model = linear_model.LinearRegression(fit_intercept=True,normalize=False,copy_X=True,n_jobs=None).fit(X,y)
    y_pred = model.predict(X_test[i])
    rmse_cv.append(np.sqrt(mean_squared_error(y_test[i],y_pred)))
print("Average RMSE across Folds:",np.mean(rmse_cv))
print("Average Scaled RMSE across Folds:",np.mean(rmse_cv)/(max(y)-min(y)))
```

```
Average RMSE across Folds: 1511.7573978732548
Average Scaled RMSE across Folds: 0.11394732541653638
```

In [31]:
```
rmse_cv = []
for i in range(0,num_folds):
    model = linear_model.LinearRegression(fit_intercept=True,normalize=False,copy_X=True,n_jobs=None).fit(X,y)
    y_pred = model.predict(X_test[i])
    rmse_cv.append(np.sqrt(mean_squared_error(y_test[i],y_pred)))
print("Average RMSE across Folds:",np.mean(rmse_cv))
print("Average Scaled RMSE across Folds:",np.mean(rmse_cv)/(max(y)-min(y)))
```

```
Average RMSE across Folds: 1511.7573978732548
Average Scaled RMSE across Folds: 0.11394732541653638
```

In [33]:
```
# Check performance on test set
model = linear_model.LinearRegression(fit_intercept=True,normalize=False,copy_X=True,n_jobs=None).fit(X,y)
y_pred = model.predict(X_ho)
rmse = np.sqrt(mean_squared_error(y_ho,y_pred))
r2 = r2_score(y_ho,y_pred)
print("RMSE: ",rmse)
print("R2:",r2)

scaled_RMSE = rmse/(max(y)-min(y))
print(scaled_RMSE)
```

```
RMSE:  1544.0262848099287
R2: 0.5359175043695535
0.11637956313257161
```

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 [ ]:

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 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))
```

Model Generation and Test Error Estimation

Basic Logistic Regression Model with Interaction Terms

In [4]:

```
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")

# Train now on entire training set to get model for prediction
model1 <- glm(ketBool ~ ., data=train1, family='binomial')
```

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

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

Basic Logistic Regression without Interaction Terms

In [5]:
```
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")

# Train now on entire training set to get model for prediction
model2 <- glm(ketBool ~ ., data=train2, family='binomial')
```

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

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

Look at Coefficients on TRAIN

Model 1 (including interaction terms) Summary

In [6]:
```
summary(model1)
```

```
Call:
glm(formula = ketBool ~ ., family = "binomial", data = train1)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.9532  -0.2677   0.0100   0.2355   5.0333

Coefficients:
                                      Estimate Std. Error z value Pr(>|z|)
(Intercept)                           -3.07543    0.84032  -3.660 0.000252 ***
totalCellNum                           1.48468    0.36237   4.097 4.18e-05 ***
gender                                 5.27077    0.80163   6.575 4.86e-11 ***
genotype                               4.70327    0.73513   6.398 1.58e-10 ***
weight_g                               0.12584    0.35738   0.352 0.724760
ketamine_day                          -0.71834    0.25786  -2.786 0.005340 **
correlationScore                      -2.30386    0.96069  -2.398 0.016479 *
lickAccuracy                          -2.59838    0.62801  -4.137 3.51e-05 ***
lickNumber                            -0.51051    0.76272  -0.669 0.503289
avgFR                                  4.72545    1.78614   2.646 0.008154 **
avgSingleCellVariance                  7.85496    2.25356   3.486 0.000491 ***
varianceFR                             1.58919    1.02274   1.554 0.120216
avgTrialSpeed                         -0.91611    0.73195  -1.252 0.210715
varianceSpeed                          1.67179    1.06811   1.565 0.117539
medianCellDepth                        1.26773    0.30867   4.107 4.01e-05 ***
totalCellNumxCorrelationScore         -2.45892    0.29758  -8.263  < 2e-16 ***
totalCellNumxLickAccuracy              0.24686    0.16550   1.492 0.135803
totalCellNumxLickNumber                0.08498    0.19849   0.428 0.668571
totalCellNumxAvgFR                     1.43740    0.56874   2.527 0.011493 *
totalCellNumxAvgSingleCellVariance    -0.67645    0.40254  -1.680 0.092871 .
totalCellNumxVarianceFR               -0.86265    0.16684  -5.171 2.33e-07 ***
totalCellNumxAvgTrialSpeed            -0.94312    0.24154  -3.905 9.44e-05 ***
totalCellNumxVarianceSpeed             0.75321    0.28168   2.674 0.007495 **
genderxCorrelationScore               -0.08490    0.24236  -0.350 0.726094
genderxLickAccuracy                    0.28144    0.12668   2.222 0.026304 *
genderxLickNumber                     -0.14458    0.10581  -1.366 0.171828
genderxAvgFR                          -2.35801    0.80157  -2.942 0.003264 **
genderxAvgSingleCellVariance           0.54791    0.54906   0.998 0.318326
genderxVarianceFR                      0.08090    0.30108   0.269 0.788160
genderxAvgTrialSpeed                  -1.05481    0.29317  -3.598 0.000321 ***
genderxVarianceSpeed                  -1.15447    0.38986  -2.961 0.003064 **
genotypexCorrelationScore             -0.33733    0.22293  -1.513 0.130233
genotypexLickAccuracy                  0.56838    0.16717   3.400 0.000674 ***
genotypexLickNumber                   -0.52894    0.14448  -3.661 0.000251 ***
genotypexAvgFR                         0.92673    0.72838   1.272 0.203262
genotypexAvgSingleCellVariance        -5.99671    0.85149  -7.043 1.89e-12 ***
genotypexVarianceFR                   -0.22840    0.33708  -0.678 0.498033
genotypexAvgTrialSpeed                -0.49684    0.32787  -1.515 0.129686
genotypexVarianceSpeed                -0.61546    0.44262  -1.391 0.164377
weight_gxCorrelationScore              3.16988    0.79975   3.964 7.38e-05 ***
weight_gxLickAccuracy                  0.94502    0.50304   1.879 0.060297 .
weight_gxLickNumber                    0.04100    0.56794   0.072 0.942451
weight_gxAvgFR                        -1.81136    1.68183  -1.077 0.281473
weight_gxAvgSingleCellVariance        -1.28619    1.45427  -0.884 0.376470
weight_gxVarianceFR                   -0.93719    0.97722  -0.959 0.337537
weight_gxAvgTrialSpeed                 0.54390    0.57514   0.946 0.344310
weight_gxVarianceSpeed                -2.35374    0.78367  -3.003 0.002669 **
ketamine_dayxCorrelationScore          0.41956    0.33708   1.245 0.213237
ketamine_dayxLickAccuracy             -0.20682    0.20144  -1.027 0.304539
ketamine_dayxLickNumber                0.45125    0.22458   2.009 0.044502 *
ketamine_dayxAvgFR                     1.44940    0.96651   1.500 0.133715
ketamine_dayxAvgSingleCellVariance    -0.92388    0.94597  -0.977 0.328747
ketamine_dayxVarianceFR               -0.76557    0.45440  -1.685 0.092027 .
ketamine_dayxAvgTrialSpeed             0.24416    0.32816   0.744 0.456857
ketamine_dayxVarianceSpeed             1.47173    0.38611   3.812 0.000138 ***
medianCellDepthxCorrelationScore      -1.44980    0.32376  -4.478 7.53e-06 ***
medianCellDepthxLickAccuracy           0.36188    0.22533   1.606 0.108280
medianCellDepthxLickNumber            -0.08501    0.26385  -0.322 0.747295
medianCellDepthxAvgFR                 -2.05059    0.75163  -2.728 0.006368 **
medianCellDepthxAvgSingleCellVariance -2.85366    0.94725  -3.013 0.002590 **
medianCellDepthxVarianceFR             0.20696    0.27064   0.765 0.444430
medianCellDepthxAvgTrialSpeed          1.63901    0.31478   5.207 1.92e-07 ***
medianCellDepthxVarianceSpeed         -1.95645    0.45961  -4.257 2.07e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5538.9  on 3996  degrees of freedom
Residual deviance: 1794.6  on 3934  degrees of freedom
AIC: 1920.6
```

```
Number of Fisher Scoring iterations: 7
```

Model 2 (not including interaction terms) Summary

In [7]:
```
summary(model2)
```

```
Call:
glm(formula = ketBool ~ ., family = "binomial", data = train2)

Deviance Residuals: 
    Min       1Q   Median       3Q      Max  
-3.1024  -0.4780   0.0845   0.4688   3.8002  

Coefficients:
                      Estimate Std. Error z value Pr(>|z|)    
(Intercept)           -0.64251    0.15630  -4.111 3.94e-05 ***
totalCellNum          -0.03728    0.06442  -0.579 0.562759    
gender                 0.46578    0.13882   3.355 0.000793 ***
genotype               0.05546    0.11283   0.492 0.623009    
weight_g              -0.46373    0.06503  -7.131 9.94e-13 ***
ketamine_day           0.16121    0.04486   3.594 0.000326 ***
correlationScore      -1.50123    0.07377 -20.351  < 2e-16 ***
lickAccuracy          -0.80960    0.05888 -13.750  < 2e-16 ***
lickNumber            -0.60947    0.06549  -9.306  < 2e-16 ***
avgFR                  1.77153    0.12885  13.749  < 2e-16 ***
avgSingleCellVariance -1.32595    0.11463 -11.567  < 2e-16 ***
varianceFR            -0.23810    0.05966  -3.991 6.58e-05 ***
avgTrialSpeed         -0.18518    0.05614  -3.298 0.000973 ***
varianceSpeed         -0.99420    0.07504 -13.249  < 2e-16 ***
medianCellDepth       -0.04057    0.05383  -0.754 0.451108    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 5538.9  on 3996  degrees of freedom
Residual deviance: 2741.0  on 3982  degrees of freedom
AIC: 2771

Number of Fisher Scoring iterations: 6
```

Test Performance

In [8]:
```
lr_pred_lo <- predict(model1,test1) # lo : log odds
num_val = length(test1$ketBool)
lr_pred = rep(0,num_val)
actual = rep(0,num_val)
for (j in 1:num_val){
    if (lr_pred_lo[j]>0){
        lr_pred[j]=1
    }
    actual[j] = test1$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 (Test Set):",test_error_est,"\n")
cat("Accuracy (Test Set):",1-test_error_est,"\n")
cat("=====================================================================\n")
```

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

Zero-One Loss (Test Set): 0.13475 
Accuracy (Test Set): 0.86525 
=====================================================================
```

In [9]:
```
lr_pred_lo <- predict(model2,test2) # lo : log odds
num_val = length(test2$ketBool)
lr_pred = rep(0,num_val)
actual = rep(0,num_val)
for (j in 1:num_val){
    if (lr_pred_lo[j]>0){
        lr_pred[j]=1
    }
    actual[j] = test2$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 (Test Set):",test_error_est,"\n")
cat("Accuracy (Test Set):",1-test_error_est,"\n")
cat("======================================================================\n")
```

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

Zero-One Loss (Test Set): 0.14175
Accuracy (Test Set): 0.85825
======================================================================
```

Look at Coefficients on TEST

With Interaction Terms

In [10]:

```
model1_test <- glm(ketBool ~ ., data=test1, family='binomial')
summary(model1_test)
```

```
Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”
```

```
Call:
glm(formula = ketBool ~ ., family = "binomial", data = test1)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.0289  -0.1986   0.0016   0.1849   2.9694

Coefficients:
                                      Estimate Std. Error z value Pr(>|z|)
(Intercept)                           -3.65227    2.05234  -1.780 0.075147 .
totalCellNum                           0.26300    0.88535   0.297 0.766426
gender                                 7.81598    1.83003   4.271 1.95e-05 ***
genotype                               7.42112    1.86461   3.980 6.89e-05 ***
weight_g                               1.15238    0.83398   1.382 0.167037
ketamine_day                          -1.58289    0.63604  -2.489 0.012823 *
correlationScore                      -2.20880    2.19214  -1.008 0.313647
lickAccuracy                          -2.24522    1.44600  -1.553 0.120494
lickNumber                             0.33636    1.85504   0.181 0.856112
avgFR                                  4.23584    3.76048   1.126 0.259992
avgSingleCellVariance                  9.97080    5.09635   1.956 0.050411 .
varianceFR                             3.36081    2.66836   1.260 0.207848
avgTrialSpeed                          0.29209    1.81798   0.161 0.872356
varianceSpeed                          3.03594    2.45716   1.236 0.216627
medianCellDepth                        1.65360    0.72813   2.271 0.023146 *
totalCellNumxCorrelationScore         -3.01461    0.66719  -4.518 6.23e-06 ***
totalCellNumxLickAccuracy             -0.01844    0.42563  -0.043 0.965452
totalCellNumxLickNumber                0.85883    0.46080   1.864 0.062355 .
totalCellNumxAvgFR                    -0.72389    1.28868  -0.562 0.574299
totalCellNumxAvgSingleCellVariance     2.13417    1.06527   2.003 0.045134 *
totalCellNumxVarianceFR               -1.20751    0.36359  -3.321 0.000897 ***
totalCellNumxAvgTrialSpeed            -1.15633    0.56101  -2.061 0.039289 *
totalCellNumxVarianceSpeed             2.58590    0.57912   4.465 8.00e-06 ***
genderxCorrelationScore               -0.49791    0.55891  -0.891 0.373005
genderxLickAccuracy                    0.08964    0.28202   0.318 0.750598
genderxLickNumber                      0.50079    0.34721   1.442 0.149218
genderxAvgFR                          -4.79261    1.83800  -2.608 0.009120 **
genderxAvgSingleCellVariance           2.02599    1.25764   1.611 0.107191
genderxVarianceFR                      0.60497    0.64171   0.943 0.345811
genderxAvgTrialSpeed                  -1.38870    0.69665  -1.993 0.046219 *
genderxVarianceSpeed                  -2.08394    1.04008  -2.004 0.045109 *
genotypexCorrelationScore             -0.65555    0.55225  -1.187 0.235209
genotypexLickAccuracy                  0.28092    0.36612   0.767 0.442899
genotypexLickNumber                   -0.37848    0.60850  -0.622 0.533948
genotypexAvgFR                        -1.08513    1.96100  -0.553 0.580020
genotypexAvgSingleCellVariance        -6.50830    2.23454  -2.913 0.003585 **
genotypexVarianceFR                   -0.04429    0.81545  -0.054 0.956689
genotypexAvgTrialSpeed                -0.48904    0.76063  -0.643 0.520263
genotypexVarianceSpeed                -0.02729    1.12641  -0.024 0.980672
weight_gxCorrelationScore              4.20517    1.80688   2.327 0.019949 *
weight_gxLickAccuracy                  0.53872    1.01541   0.531 0.595730
weight_gxLickNumber                   -2.76258    1.46563  -1.885 0.059443 .
weight_gxAvgFR                         3.28156    3.49283   0.940 0.347467
weight_gxAvgSingleCellVariance        -5.86363    3.35396  -1.748 0.080417 .
weight_gxVarianceFR                   -4.58480    1.94465  -2.358 0.018391 *
weight_gxAvgTrialSpeed                -0.06083    1.38059  -0.044 0.964858
weight_gxVarianceSpeed                -5.94149    1.73747  -3.420 0.000627 ***
ketamine_dayxCorrelationScore         -0.10723    0.80114  -0.134 0.893524
ketamine_dayxLickAccuracy              0.31969    0.45310   0.706 0.480469
ketamine_dayxLickNumber                0.37617    0.53723   0.700 0.483798
ketamine_dayxAvgFR                     3.33903    2.03621   1.640 0.101042
ketamine_dayxAvgSingleCellVariance    -0.31757    2.03221  -0.156 0.875822
ketamine_dayxVarianceFR               -0.62787    1.00364  -0.626 0.531581
ketamine_dayxAvgTrialSpeed             0.07342    0.73297   0.100 0.920208
ketamine_dayxVarianceSpeed             1.03936    0.87252   1.191 0.233567
medianCellDepthxCorrelationScore      -1.59720    0.70336  -2.271 0.023158 *
medianCellDepthxLickAccuracy           0.32039    0.49388   0.649 0.516519
medianCellDepthxLickNumber             1.08690    0.77763   1.398 0.162200
medianCellDepthxAvgFR                 -4.70330    1.56983  -2.996 0.002735 **
medianCellDepthxAvgSingleCellVariance -2.55433    2.04819  -1.247 0.212353
medianCellDepthxVarianceFR             0.90155    0.98434   0.916 0.359726
medianCellDepthxAvgTrialSpeed          1.31766    0.76205   1.729 0.083792 .
medianCellDepthxVarianceSpeed         -1.42238    1.01890  -1.396 0.162715
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1386.19  on 999  degrees of freedom
Residual deviance:  399.57  on 937  degrees of freedom
AIC: 525.57
```

```
Number of Fisher Scoring iterations: 8
```

Without Interaction Terms

In [11]:
```
model2_test <- glm(ketBool ~ ., data=test2, family='binomial')
summary(model2_test)
```

```
Call:
glm(formula = ketBool ~ ., family = "binomial", data = test2)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.52524  -0.51562   0.05817   0.49094   3.03239

Coefficients:
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)           -0.11054    0.29360  -0.377 0.706540
totalCellNum          -0.20103    0.12503  -1.608 0.107873
gender                 0.38001    0.26237   1.448 0.147507
genotype               0.28291    0.21514   1.315 0.188518
weight_g              -0.21089    0.13190  -1.599 0.109851
ketamine_day          -0.07733    0.08284  -0.934 0.350543
correlationScore      -1.27870    0.13062  -9.789  < 2e-16 ***
lickAccuracy          -0.77094    0.10958  -7.035 1.99e-12 ***
lickNumber            -0.69481    0.14704  -4.725 2.30e-06 ***
avgFR                  1.96396    0.25970   7.562 3.96e-14 ***
avgSingleCellVariance -1.21420    0.22392  -5.423 5.88e-08 ***
varianceFR            -0.54385    0.14694  -3.701 0.000215 ***
avgTrialSpeed         -0.22271    0.11054  -2.015 0.043934 *
varianceSpeed         -0.79791    0.14073  -5.670 1.43e-08 ***
medianCellDepth       -0.14587    0.10636  -1.371 0.170241
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1386.19  on 999  degrees of freedom
Residual deviance:  712.67  on 985  degrees of freedom
AIC: 742.67

Number of Fisher Scoring iterations: 6
```

BOOTSTRAP

In [16]:
```
df = train1
coef.boot = function(data, indices) {
fm = glm(data = data[indices,],ketBool ~ ., family='binomial')
return(coef(fm))
}
```

```
Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”
```

5000 iterations for speed

```
In [19]: boot.out5000 = boot(df, coef.boot, 5000)
         boot.out5000
```

```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = df, statistic = coef.boot, R = 5000)


Bootstrap Statistics :
        original       bias    std. error
t1*  -3.07542785 -0.163512021   0.8860557
t2*   1.48468157  0.112356796   0.3982214
t3*   5.27077347  0.159548420   0.9726796
t4*   4.70326862  0.313296250   0.8355723
t5*   0.12583501  0.117775884   0.4392267
t6*  -0.71834083 -0.039675563   0.2893456
t7*  -2.30385934 -0.161958413   1.0846243
t8*  -2.59837684 -0.142450055   0.8149587
t9*  -0.51050792  0.305620053   1.0697160
t10*  4.72544849  0.406416640   2.0435735
t11*  7.85495894  0.347478652   2.6492854
t12*  1.58919458  0.327730785   1.2905950
t13* -0.91610950  0.046432168   0.7747884
t14*  1.67179140  0.069693668   1.5398312
t15*  1.26773118  0.064050586   0.3727106
t16* -2.45892080 -0.136245023   0.3385636
t17*  0.24686052  0.021494625   0.1872190
t18*  0.08497568 -0.062724044   0.2535600
t19*  1.43740112  0.004117731   0.6191387
t20* -0.67644636 -0.031402824   0.4553493
t21* -0.86265381 -0.039073363   0.1912735
t22* -0.94312277 -0.044831936   0.2822165
t23*  0.75320836  0.041489704   0.3633940
t24* -0.08490350 -0.028799448   0.2833826
t25*  0.28144194 -0.014737391   0.1484569
t26* -0.14457774  0.078944184   0.1897451
t27* -2.35801119 -0.041716782   0.9129283
t28*  0.54791051  0.005038970   0.6220143
t29*  0.08089907 -0.056589037   0.3140653
t30* -1.05480872 -0.044183591   0.3112202
t31* -1.15446519 -0.022564232   0.4463352
t32* -0.33733429 -0.007568054   0.2366915
t33*  0.56837745  0.019828235   0.1873987
t34* -0.52894342 -0.097652598   0.2350277
t35*  0.92672514  0.097335353   0.7967583
t36* -5.99671390 -0.421273083   1.0111801
t37* -0.22840376 -0.016616648   0.3633037
t38* -0.49683573 -0.068480186   0.3682878
t39* -0.61546156  0.016337821   0.5350772
t40*  3.16987809  0.223434175   0.9385362
t41*  0.94502026  0.090436758   0.6523454
t42*  0.04099947 -0.343765445   0.8609804
t43* -1.81136124 -0.262571263   1.9759017
t44* -1.28618684  0.044050764   1.6340870
t45* -0.93719327 -0.209127921   1.2355809
t46*  0.54390075 -0.065604833   0.6128631
t47* -2.35374381 -0.158873696   1.0950737
t48*  0.41956403 -0.008785576   0.3909710
t49* -0.20682497 -0.009121904   0.2201788
t50*  0.45125095  0.037531456   0.2569805
t51*  1.44939647  0.129445071   1.3169700
t52* -0.92387593 -0.056870319   1.2501330
t53* -0.76557467 -0.049875559   0.4766562
t54*  0.24416101  0.013699044   0.3461907
t55*  1.47172660  0.036047186   0.5656592
t56* -1.44980365 -0.055473346   0.3799927
t57*  0.36187921  0.005246599   0.2684968
t58* -0.08501437  0.035774747   0.3051450
t59* -2.05058738 -0.137705414   0.7560627
t60* -2.85366214 -0.116652223   0.9145253
t61*  0.20696415 -0.096952072   0.4191914
t62*  1.63900578  0.104593900   0.3455519
t63* -1.95645257 -0.070477153   0.7176047
```

50000 iterations (full)

In [21]:
```
boot.out50000 = boot(df, coef.boot, 50000)
```

```
Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”Warning message:
“glm.fit: fitted probabilities numerically 0 or 1 occurred”
```

In [22]:
```
boot.out50000
```

```
ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = df, statistic = coef.boot, R = 50000)


Bootstrap Statistics :
        original        bias    std. error
t1*  -3.07542785 -1.871698e-01   0.8792577
t2*   1.48468157  1.127601e-01   0.3967608
t3*   5.27077347  1.731006e-01   0.9561716
t4*   4.70326862  3.296524e-01   0.8346279
t5*   0.12583501  1.142588e-01   0.4338361
t6*  -0.71834083 -3.709430e-02   0.2928084
t7*  -2.30385934 -1.744887e-01   1.0945069
t8*  -2.59837684 -1.208257e-01   0.8054944
t9*  -0.51050792  2.811960e-01   1.0476530
t10*  4.72544849  4.244491e-01   2.0446802
t11*  7.85495894  3.091727e-01   2.6725212
t12*  1.58919458  3.292959e-01   1.3055680
t13* -0.91610950  4.970261e-02   0.7872574
t14*  1.67179140  8.618482e-02   1.5495920
t15*  1.26773118  6.642372e-02   0.3741739
t16* -2.45892080 -1.319045e-01   0.3378930
t17*  0.24686052  2.255831e-02   0.1870262
t18*  0.08497568 -5.883696e-02   0.2532629
t19*  1.43740112 -3.469021e-03   0.6104196
t20* -0.67644636 -2.216683e-02   0.4512524
t21* -0.86265381 -3.982182e-02   0.1930281
t22* -0.94312277 -4.960241e-02   0.2765598
t23*  0.75320836  4.067542e-02   0.3610569
t24* -0.08490350 -2.945468e-02   0.2793542
t25*  0.28144194 -1.442663e-02   0.1481560
t26* -0.14457774  7.825786e-02   0.1887436
t27* -2.35801119 -3.853824e-02   0.9099176
t28*  0.54791051  5.621079e-05   0.6256602
t29*  0.08089907 -5.525108e-02   0.3122576
t30* -1.05480872 -5.268672e-02   0.3140202
t31* -1.15446519 -1.871115e-02   0.4537412
t32* -0.33733429 -6.219927e-03   0.2404171
t33*  0.56837745  1.509976e-02   0.1889228
t34* -0.52894342 -9.971533e-02   0.2349638
t35*  0.92672514  9.796381e-02   0.8023645
t36* -5.99671390 -4.181811e-01   1.0309204
t37* -0.22840376 -1.718045e-02   0.3631185
t38* -0.49683573 -7.915667e-02   0.3755853
t39* -0.61546156  1.768040e-02   0.5437601
t40*  3.16987809  2.345893e-01   0.9468469
t41*  0.94502026  7.590176e-02   0.6448561
t42*  0.04099947 -3.273368e-01   0.8450051
t43* -1.81136124 -2.787316e-01   1.9578292
t44* -1.28618684  6.494018e-02   1.6463267
t45* -0.93719327 -2.009340e-01   1.2391201
t46*  0.54390075 -5.728862e-02   0.6157488
t47* -2.35374381 -1.757830e-01   1.0874152
t48*  0.41956403 -6.743100e-03   0.3943418
t49* -0.20682497 -8.665752e-03   0.2227302
t50*  0.45125095  4.128976e-02   0.2580814
t51*  1.44939647  1.304988e-01   1.3308724
t52* -0.92387593 -5.375926e-02   1.2543304
t53* -0.76557467 -5.990636e-02   0.4783992
t54*  0.24416101  8.613222e-03   0.3482670
t55*  1.47172660  4.227326e-02   0.5596560
t56* -1.44980365 -5.973846e-02   0.3787705
t57*  0.36187921 -1.288271e-04   0.2660305
t58* -0.08501437  3.944570e-02   0.3031622
t59* -2.05058738 -1.362395e-01   0.7685607
t60* -2.85366214 -1.131885e-01   0.9198808
t61*  0.20696415 -9.772077e-02   0.4144050
t62*  1.63900578  1.068083e-01   0.3462285
t63* -1.95645257 -8.045814e-02   0.7253150
```

In [ ]: