Prediction of Adar Editing Levels from bpRNA Structure

load the feature matrix and split into training and test sets

```
set.seed(1)

#load the feature matrix
data=read.table("rf_features.txt",header=TRUE,sep='\t')
#remove values with no NaN in editing level
data=data[is.nan(data$ave_editing_level)==FALSE,]
head(data)
```

```
##   ave_editing_level edit_feat prev_feat next_feat mp1 mref1 malt1   mtype1
## 1              0.58         I         S         S  44     G     A mismatch
## 2              0.46         I         S         S  45     G     A mismatch
## 3              0.49         B         S         S  46     G     A mismatch
## 4              0.48         I         S         S  47     G     A mismatch
## 5              0.59         I         S         S  48     C     A mismatch
## 6              0.39         I         S         S  52     G     A mismatch
##   mfeat1 mfeat1_prev mfeat1_next adist1  mp2 mref2 malt2 mtype2 mfeat2
## 1      S           H           I      6 None  None  None   None   None
## 2      S           H           I      5 None  None  None   None   None
## 3      S           H           B      4 None  None  None   None   None
## 4      S           H           I      3 None  None  None   None   None
## 5      S           H           I      2 None  None  None   None   None
## 6      S           I           I     -2 None  None  None   None   None
##   mfeat2_prev mfeat2_next adist2
## 1        None        None   None
## 2        None        None   None
## 3        None        None   None
## 4        None        None   None
## 5        None        None   None
## 6        None        None   None
```

The feature values are: * edit_feat – Single character value of structure type for the edited A at position 50. One of S,H,M,I,B,X,E * prev_feat – The 5' structural feature upstream of edited A * next_feat – The 3' structural feature downstream of edited A

- mp1 – Firt mutated position along the RNA sequence
- mref1 – Reference allele at first mutated position
- mtype1 – One of "mismatch","indel","wt"
- mfeat1 – Single character value of structure type for mutated base. One of S,H,M,I,B,X,E
- mfeat1_prev – The 5' structural feature upstream of mutated base.
- mfeat1_next – The 3' structural feature downstream of mutated base.
- adist1 – The distance (in sequence space) of the mutated base from the edited A.

Repeat for possible second mutation ('None' if only 1 mutation present in sequence): * mp2

- mref2
- mtype2
- mfeat2

- mfeat2_prev
- mfeat2_next
- adist2

```
#get training and test splits; use 80% train, 20% test split
train_indices=sample(nrow(data),0.8*nrow(data),replace=FALSE)
train_split=data[train_indices,]
test_split=data[-train_indices,]
```

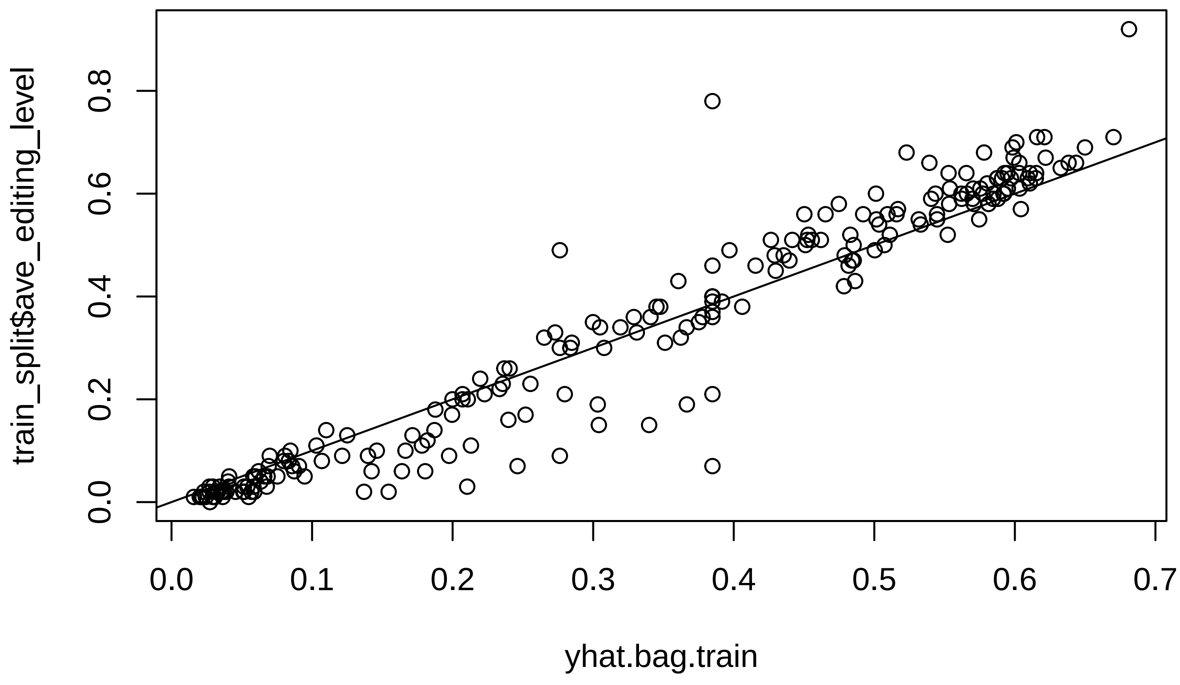
Bagging

```
#train rf
bag.data=randomForest(ave_editing_level~.,data=data,subset=train_indices,mtry=19,importance=TRUE)
print(bag.data)
```

```
##
## Call:
##  randomForest(formula = ave_editing_level ~ ., data = data, mtry = 19,      importance = TRUE, subse
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 19
##
##           Mean of squared residuals: 0.02046805
##                     % Var explained: 66.02
```

Predictions on training split

```
#get predictions on training & test data
yhat.bag.train=predict(bag.data,newdata=train_split)
yhat.bag.test=predict(bag.data,newdata=test_split)
plot(yhat.bag.train,train_split$ave_editing_level)
abline(0,1)
```

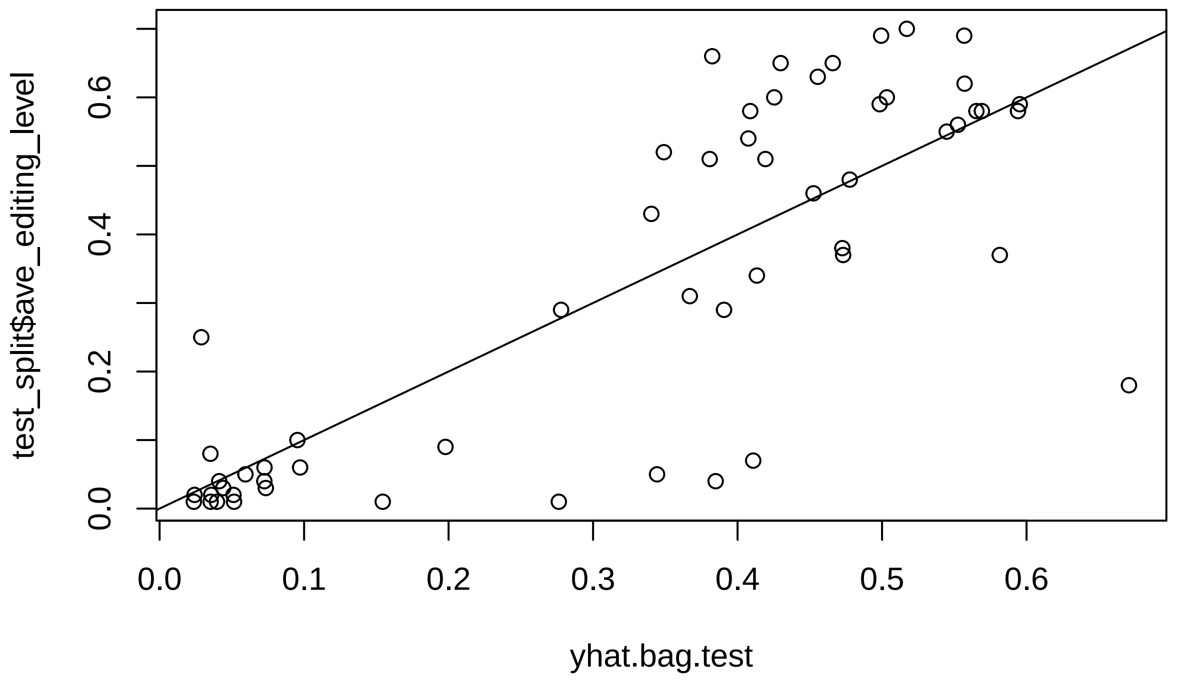


```
print(mean((yhat.bag.train-train_split$ave_editing_level)^2))
```

```
## [1] 0.004677599
```

Predictions on test split

```
plot(yhat.bag.test,test_split$ave_editing_level)
abline(0,1)
```



```
print(mean((yhat.bag.test-test_split$ave_editing_level)^2))
```

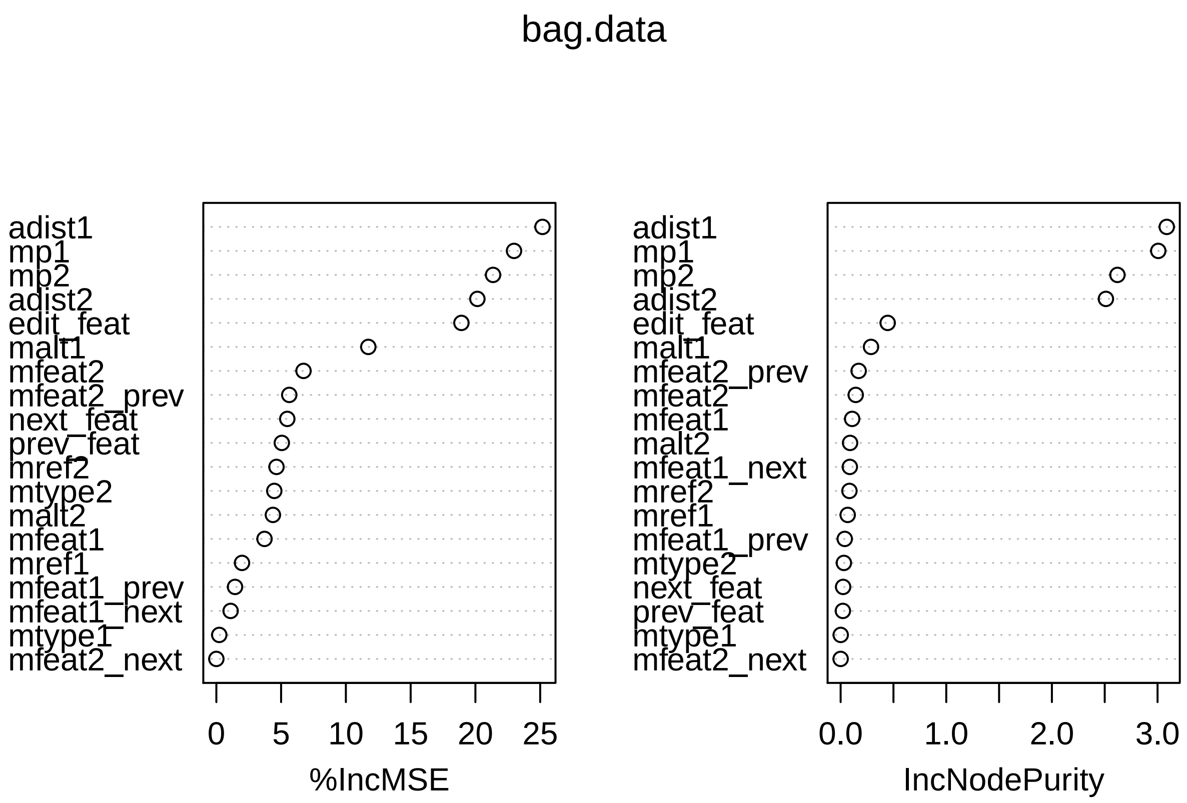
```
## [1] 0.022297
```

Feature Importance

```
#get the feature importance
importance (bag.data )
```

```
##                %IncMSE IncNodePurity
## edit_feat   18.9209983  0.4455900061
## prev_feat    5.0558625  0.0212787339
## next_feat    5.4774264  0.0235704694
## mp1         22.9818109  3.0067065220
## mref1        1.9819797  0.0669387909
## malt1       11.7334443  0.2878079870
## mtype1       0.2259686  0.0009146741
## mfeat1       3.7181708  0.1091955264
## mfeat1_prev  1.4418862  0.0391103544
## mfeat1_next  1.1022915  0.0872495660
## adist1      25.1776183  3.0870761409
## mp2         21.3626764  2.6203176944
## mref2        4.6431558  0.0835320087
## malt2        4.3680359  0.0895073848
## mtype2       4.4722928  0.0308120723
## mfeat2       6.7302069  0.1431442206
## mfeat2_prev  5.6299865  0.1709225453
## mfeat2_next  0.0000000  0.0000000000
## adist2      20.1530985  2.5110449696
```

```
varImpPlot(bag.data)
```



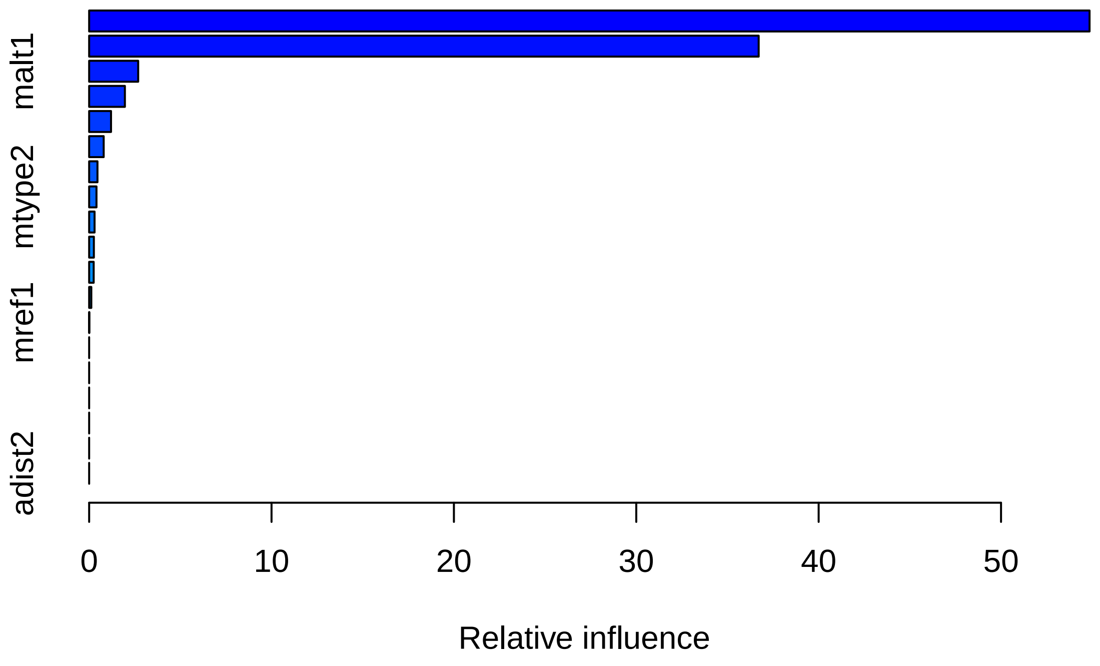
The features of highest importance are distance from editing site, position of the first and second mutations.

Boosting

```
boost.data=gbm(ave_editing_level~.,data=train_split,distribution="gaussian",n.trees=5000,interaction.dep
```

```
## Warning in gbm.fit(x, y, offset = offset, distribution = distribution, w =
## w, : variable 18: mfeat2_next has no variation.
```

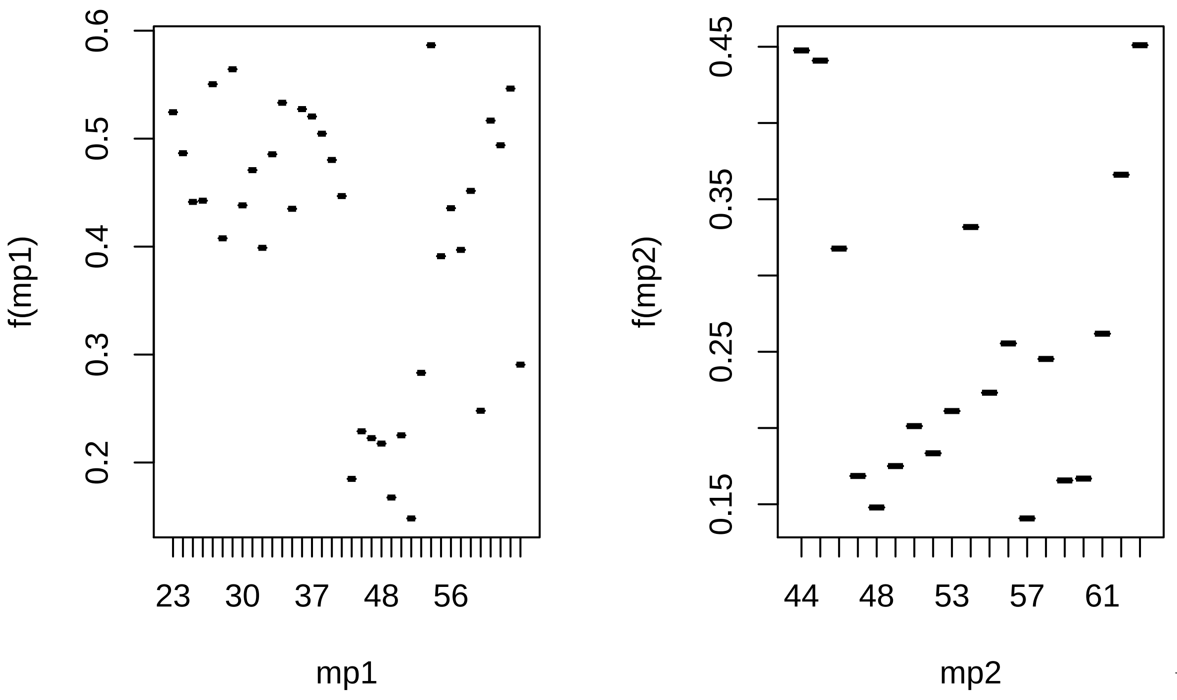
```
summary(boost.data)
```



```
##                     var      rel.inf
## mp1                 mp1 54.854106865
## mp2                 mp2 36.710661070
## malt1             malt1  2.688685945
## edit_feat     edit_feat  1.959715484
## mfeat2_prev mfeat2_prev  1.199563084
## mfeat1_next mfeat1_next  0.797053226
## mfeat1           mfeat1  0.456482574
## mtype2           mtype2  0.402604340
## mfeat2           mfeat2  0.296840920
## mref2             mref2  0.258297885
## malt2             malt2  0.249275037
## mfeat1_prev mfeat1_prev  0.118063219
## mref1             mref1  0.008650351
## prev_feat     prev_feat  0.000000000
## next_feat     next_feat  0.000000000
## mtype1           mtype1  0.000000000
## adist1           adist1  0.000000000
## mfeat2_next mfeat2_next  0.000000000
## adist2           adist2  0.000000000
```

The “mp1” and “mp2” features are the most important variables (these are the positions along the sequence of mutation 1 and 2)

```
par(mfrow=c(1,2))
plot(boost.data,i="mp1")
plot(boost.data,i="mp2")
```



We use the boosted model to predict on the test data:

```
yhat.boost=predict(boost.data,newdata=test_split,n.trees=5000)
mean((yhat.boost-test_split$ave_editing_level)^2)
```

```
## [1] 0.02035235
```

Experiment with the boosting shrinkage parameter (increase to 0.2 from default of 0.001)

```
for(shrinkage_val in c(0.005, 0.01, 0.05, 0.1, 0.2))
{
print(shrinkage_val)
boost.data=gbm(ave_editing_level~.,data=train_split,distribution="gaussian",n.trees=5000,interaction.dep
yhat.boost=predict(boost.data,newdata=test_split,n.trees=5000)
print(mean((yhat.boost-test_split$ave_editing_level)^2))
}
```

```
## [1] 0.005

## Warning in gbm.fit(x, y, offset = offset, distribution = distribution, w =
## w, : variable 18: mfeat2_next has no variation.

## [1] 0.02283084
## [1] 0.01

## Warning in gbm.fit(x, y, offset = offset, distribution = distribution, w =
## w, : variable 18: mfeat2_next has no variation.

## [1] 0.02650651
## [1] 0.05

## Warning in gbm.fit(x, y, offset = offset, distribution = distribution, w =
## w, : variable 18: mfeat2_next has no variation.

## [1] 0.03181452
## [1] 0.1

## Warning in gbm.fit(x, y, offset = offset, distribution = distribution, w =
## w, : variable 18: mfeat2_next has no variation.

## [1] 0.03275378
## [1] 0.2

## Warning in gbm.fit(x, y, offset = offset, distribution = distribution, w =
## w, : variable 18: mfeat2_next has no variation.

## [1] 0.04258782
```