BISC 577 Unit 3 HW 2

Liana Engie

May 9, 2016

My github repository for this assignment can be found at `https://github.com/lengie/577Spring16`. This document was created using R markdown and has the code and results imbedded directly into the pdf. First, the necessary packages are loaded into R. I am using R version 3.3.0.

```
library(ggplot2)
library(DNAshapeR)
library(caret)
library(AnnotationHub)
library(rtracklayer)
library(BSgenome.Mmusculus.UCSC.mm10)
library(e1071)
library(ROCR)
library(pROC)
```

High Throughput Binding Assays

Systematic evolution of ligands by exponential enrichment with next-gen sequencing, or SELEX-seq, is an iterative *in vitro* method that identifies DNA sequences that bind to the desired protein or peptide. A large DNA library is generated and the protein is introduced. Those that bind the protein are separated from the unbound DNA and these are sequenced and used as the new DNA library. Cycles are repeated with different selection criteria. SELEX-seq selects moderate to highly selective binding sites, identifies more sites than traditional SELEX, and requires fewer iterations.

Protein binding microarrays, or PBMs, is a procedure that involves a surface where tens of thousands of double stranded DNA sequences are bound. Multiple proteins can be applied and the amount of bound protein is quantified using fluorescence.

Chromatin immuno precipitation sequencing, also known as ChIP-seq, identifies DNA binding sites *in vivo*. Cells are lysed and their DNA sonicated. Antibodies targeted to the desired protein are added and the proteins, with the DNA it's bound to, are pulled out via beads that are attached to the antibodies. The proteins are unlinked from the DNA pieces and these segments are sequenced.

Building prediction models for *in vitro* data

In vitro data from gcPBM is used in the following section. We will compare prediction models using sequence alone ("1-mer" sequence model) versus sequence and shape features together ("1-mer+shape" model). The `caret` package allows us to run simple machine learning algorithms on the data, splitting the data into training and test data. We will look at predicting DNA binding using shape and sequence or sequence alone, then compare the coefficients of determination. These, the R^2 values, will help us compare the two methods. Code results are hidden so outputted code and warnings are not displayed. A visual comparison follows in the next block of code.

```
#shape prediction
md <- "Mad.txt.fa"
predMd <- getShape(md)
mx <- "Max.txt.fa"
```

```
predMx <- getShape(mx)
myc <- "Myc.txt.fa"
predMyc <- getShape(myc)

#Feature vectors for each data set
#It's going to be a bit messy with a lot of repeats; could code a function instead
featureType <- "1-mer"
featureTypeB <- c("1-mer", "1-shape")
featVectMd <- encodeSeqShape(md,predMd,featureType)
featVectMx <- encodeSeqShape(mx,predMx,featureType)
featVectMyc <- encodeSeqShape(myc,predMyc,featureType)
featVectMd2 <- encodeSeqShape(md,predMd,featureTypeB)
featVectMx2 <- encodeSeqShape(mx,predMx,featureTypeB)
featVectMyc2 <- encodeSeqShape(myc,predMyc,featureTypeB)

md_data <- read.table("Mad.txt")
mx_data <- read.table("Max.txt")
myc_data <- read.table("Myc.txt")
#Could put all into one data frame but it might be more convenient to keep them separate
dfMd <- data.frame(affinity=md_data$V2, featVectMd)
dfMx <- data.frame(affinity=mx_data$V2, featVectMx)
dfMyc <- data.frame(affinity=myc_data$V2, featVectMyc)
dfMd2 <- data.frame(affinity=md_data$V2, featVectMd2)
dfMx2 <- data.frame(affinity=mx_data$V2, featVectMx2)
dfMyc2 <- data.frame(affinity=myc_data$V2, featVectMyc2)

#Settings for caret package, 10fold cross validation for LR
trainControl <- trainControl(method = "cv", number = 10, savePredictions = TRUE)

#Prediction without (then with) L2-regularized
modelMd <- train(affinity~., data = dfMd, trControl=trainControl,
                 method = "glmnet", tuneGrid = data.frame(alpha = 0, lambda = c(2^c(-15:15))))
modelMd2 <- train(affinity~., data = dfMd2, trControl=trainControl,
                 method = "glmnet", tuneGrid = data.frame(alpha = 0, lambda = c(2^c(-15:15))))
modelMx <- train(affinity~., data = dfMx, trControl=trainControl,
                 method = "glmnet", tuneGrid = data.frame(alpha = 0, lambda = c(2^c(-15:15))))
modelMx2 <- train(affinity~., data = dfMx2, trControl=trainControl,
                 method = "glmnet", tuneGrid = data.frame(alpha = 0, lambda = c(2^c(-15:15))))
modelMyc <- train(affinity~., data = dfMyc, trControl=trainControl,
                  method = "glmnet", tuneGrid = data.frame(alpha = 0, lambda = c(2^c(-15:15))))
modelMyc2 <- train(affinity~., data = dfMyc2, trControl=trainControl,
                  method = "glmnet", tuneGrid = data.frame(alpha = 0, lambda = c(2^c(-15:15))))
```

So now we have the models of all three experiments, with either sequence alone or sequence and shape as a feature. We look at the mean R-sequared values, as well as the maximum R^2 per dataset.

```
mean(modelMd$results$Rsquared,na.rm=TRUE)
```

```
## [1] 0.7170044
```

```
mean(modelMx$results$Rsquared,na.rm=TRUE)
```

```
## [1] 0.7024444
```

```
mean(modelMyc$results$Rsquared,na.rm=TRUE)
```

```
## [1] 0.7209758
```

```
MdSeq <- max(modelMd$results$Rsquared,na.rm=TRUE)
MxSeq <- max(modelMx$results$Rsquared,na.rm=TRUE)
MycSeq <- max(modelMyc$results$Rsquared,na.rm=TRUE)

mean(modelMd2$results$Rsquared,na.rm=TRUE)
```

```
## [1] 0.7331589
```

```
mean(modelMx2$results$Rsquared,na.rm=TRUE)
```

```
## [1] 0.7201405
```

```
mean(modelMyc2$results$Rsquared,na.rm=TRUE)
```

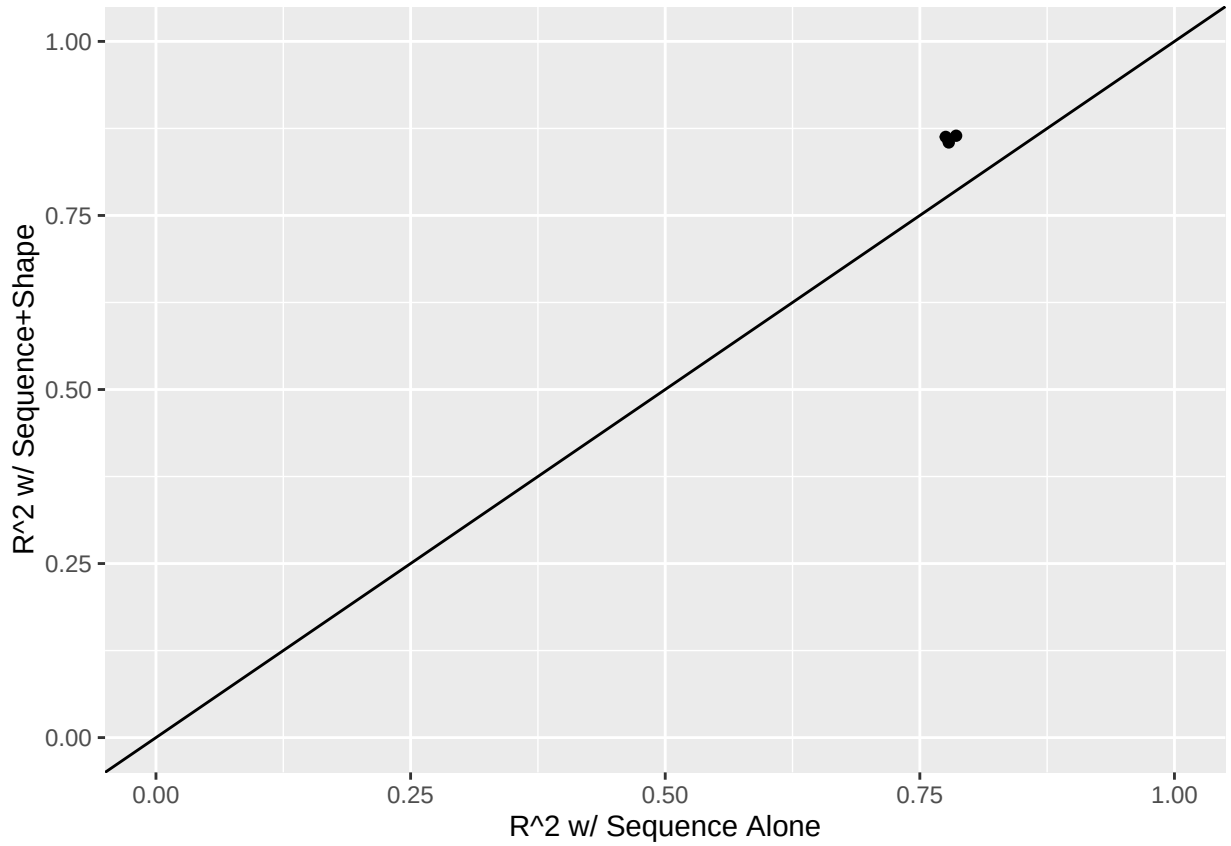
```
## [1] 0.7300687
```

```
MdShpSeq <- max(modelMd2$results$Rsquared,na.rm=TRUE)
MxShpSeq <- max(modelMx2$results$Rsquared,na.rm=TRUE)
MycShpSeq <- max(modelMyc2$results$Rsquared,na.rm=TRUE)
```

High throughput *in vitro* data analysis

From this data we can create a graph that compares the performance of the sequence alone versus sequence plus shape. The average values are actually not too far from each other. The R^2 values for the sequence+shape are .01 to .02 better than that of sequence alone, which is difficult to see on a graph. So, the following plot takes the maximum values for each and plots them against each other.

```
rsq <- data.frame(c(MdSeq,MxSeq,MycSeq),c(MdShpSeq,MxShpSeq,MycShpSeq))
colnames(rsq) = c("Seq","ShpSeq")
ggplot(rsq,aes(x=Seq,y=ShpSeq)) + xlim(c(0,1)) + ylim(c(0,1)) +
     xlab("R^2 w/ Sequence Alone") +
     ylab("R^2 w/ Sequence+Shape") +
     geom_point() + geom_abline(intercept=0,slope=1)
```



From the graph it is possibe to see that sequence and shape together did a better job of predicting DNA binding. Stronger evidence would be to show that the p-value indicates that this difference is significant. We have two samples that are not independent. I will use the Wilcox rank sum test to compare two related samples.

```
wilcox.test(c(MdSeq,MxSeq,MycSeq),c(MdShpSeq,MxShpSeq,MycShpSeq),paired = TRUE)
```

```
##
##  Wilcoxon signed rank test
##
## data:  c(MdSeq, MxSeq, MycSeq) and c(MdShpSeq, MxShpSeq, MycShpSeq)
## V = 0, p-value = 0.25
## alternative hypothesis: true location shift is not equal to 0
```

PBM data, as stated above, shows the binding of proteins to DNA. The results show that using sequence to predict the binding of said proteins is not as strong as using both the sequence and the DNA shape for those sequences (which takes more local information into account).

High throughput *in vivo* data analysis

First ChIP-seq data for the CTCF transcription factor of *Mus musculus* is saved as a fasta file, prepped via the Annotation Hub package.

```
ah <- AnnotationHub()
```

```
## snapshotDate(): 2016-05-12
```

```
#unique(ah$dataprovider) #suppressed because it was so many pages
#unique(ah$species)

ah <- subset(ah, species == "Mus musculus")
ah
```

```
## AnnotationHub with 1398 records
## # snapshotDate(): 2016-05-12
## # $dataprovider: Haemcode, UCSC, Ensembl, Gencode, Inparanoid8, ftp://f...
## # $species: Mus musculus
## # $rdataclass: GRanges, BigWigFile, FaFile, ChainFile, TwoBitFile, Inpa...
## # additional mcols(): taxonomyid, genome, description, tags,
## #   sourceurl, sourcetype
## # retrieve records with, e.g., 'object[["AH187"]]'
##
##             title
##   AH187   | Mus_musculus.GRCm38.69.cdna.all.fa
##   AH188   | Mus_musculus.GRCm38.69.dna.toplevel.fa
##   AH189   | Mus_musculus.GRCm38.69.dna_rm.toplevel.fa
##   AH190   | Mus_musculus.GRCm38.69.dna_sm.toplevel.fa
##   AH191   | Mus_musculus.GRCm38.69.ncrna.fa
##   ...       ...
##   AH50608 | Mus_musculus.GRCm38.cdna.all.2bit
##   AH50609 | Mus_musculus.GRCm38.dna.primary_assembly.2bit
##   AH50610 | Mus_musculus.GRCm38.dna_rm.primary_assembly.2bit
##   AH50611 | Mus_musculus.GRCm38.dna_sm.primary_assembly.2bit
##   AH50612 | Mus_musculus.GRCm38.ncrna.2bit
```

From this we see a list of all data available for *Mus musculus*, which, honestly, wasn't really necessary because Tsu-Pei has already provided the ID for the data that we need:

```
CTCF <- ah[["AH28451"]]
```

```
## loading from cache 'C:/Users/Liana/Documents/AppData/.AnnotationHub/33891'
```

```
seqlevelsStyle(CTCF) <- "UCSC"
getFasta(CTCF, BSgenome.Mmusculus.UCSC.mm10, width = 150, filename = "CTCF.fa")
```

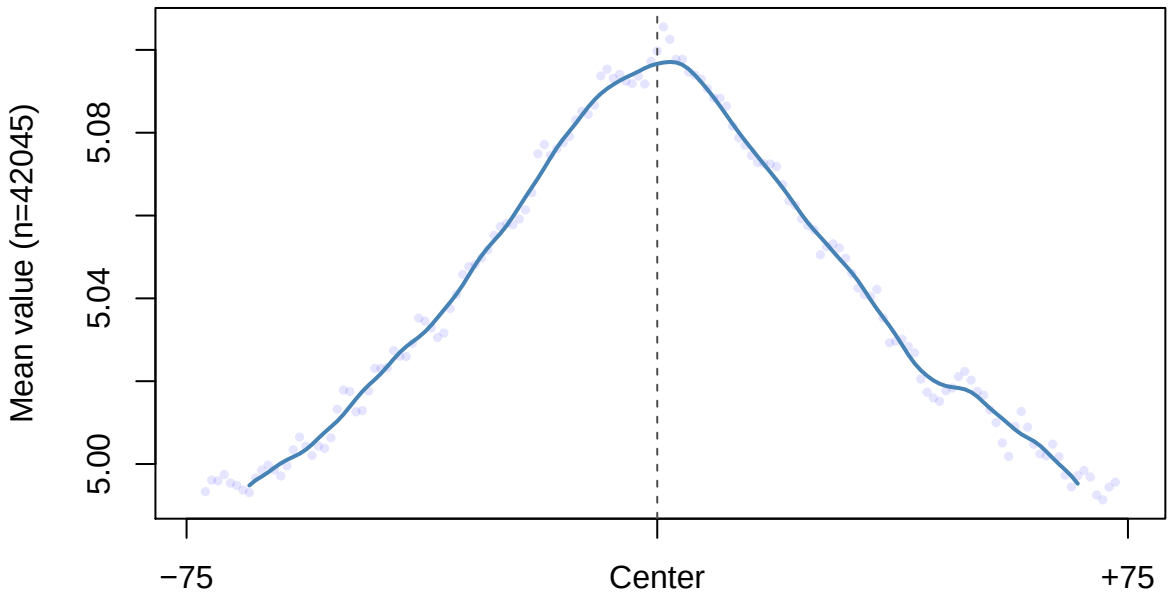
```
## Removed sequences #: integer(0)
```

Now, DNAshapeR is used to generate emsemble plots for: minor groove width, propeller twist, roll, helix twist. I felt that the metaprofiles were more clear than the heat maps, so I am using those in this report. The following graphs plot a sequence of means for consecutive instances of the selected variable.

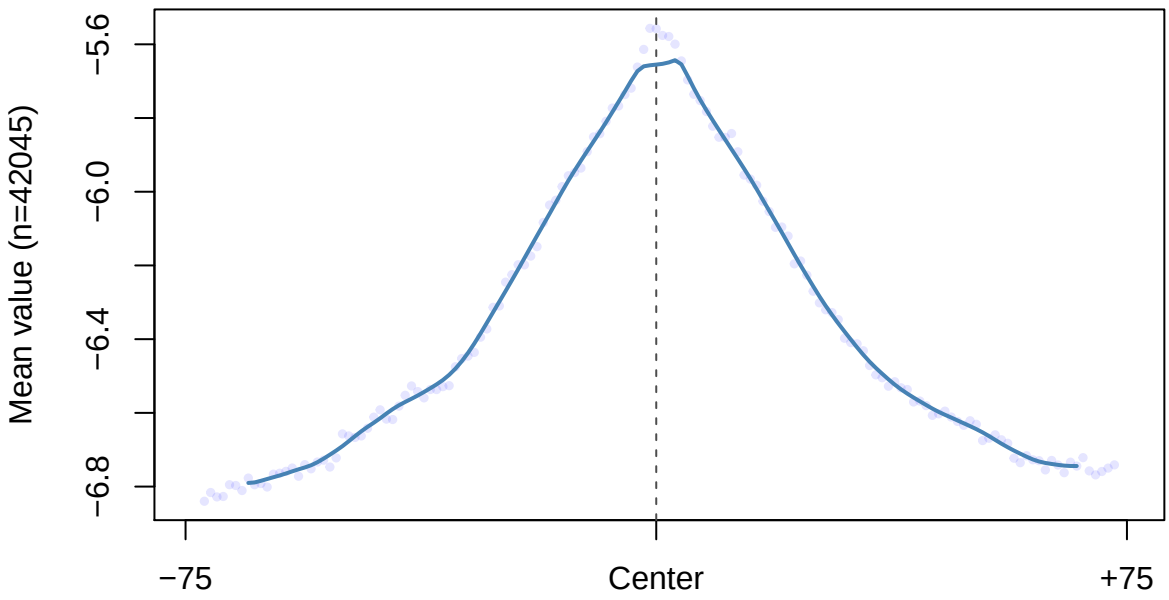
```
predMus <- getShape("CTCF.fa")
```

```
## Reading the input sequence......
## Reading the input sequence......
## Reading the input sequence......
## Reading the input sequence......

## Parsing files......

## Record length: 150

## Record length: 149

## Record length: 150

## Record length: 149

## Done
```

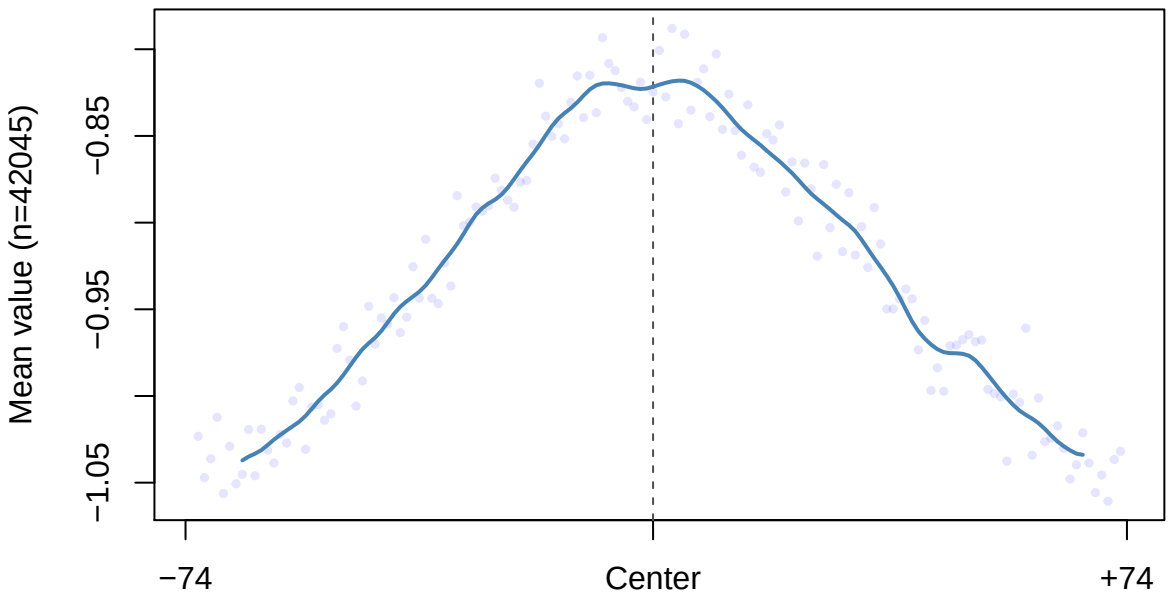
```r
#Minor groove width
plotShape(predMus$MGW)
```



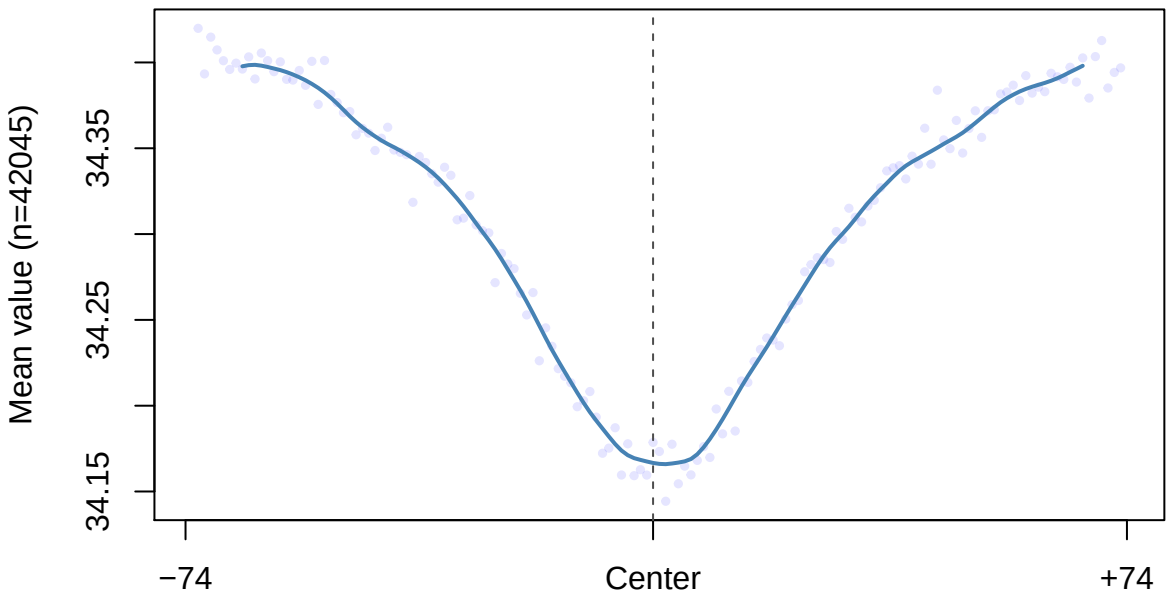

```r
#Propeller twist
plotShape(predMus$ProT)
```




```
#Roll
plotShape(predMus$Roll)
```




```
#Helix twist
plotShape(predMus$HelT)
```



B-DNA is known to have a helix twist of about 36 degrees, while A-DNA has a twist of 32.7 degrees. We can see that the helix twist of the data is sort of in between, holding a range between 34-35 degrees. B-DNA also has a smaller minor groove width than A-DNA. The minor groove width throughout this data is more indicative of B-DNA. Propeller twist of both A and B-DNA, however, is 0, as the two base pairs that align with each other tend to sit flat. In this base, the bases are rotated in relation to each other. There is also a slight roll, where the stacked base pairs are at a slight angle in relation to each other. These distortions would seem to be caused by the binding of the CTCF transcription factor to the DNA, or, perhaps, the distortions in the data allow for the binding of the TF.

Prediction models for *in vitro* data

For the last portion of this report, I redownload the CTCF data to make a file of only 30 basepairs. We want to be able to generate sequences from random genomic regions which do not overlap with genomic regions from the ChIP-seq data. A function to create this data follows, differentiating between the two categories of "bound" and "unbound" data.

```
sampleSize <- 1000

# Reducing the sequence length of bound ChIP-seq data
getFasta(CTCF, BSgenome.Mmusculus.UCSC.mm10, width = 30, filename = "bound.fa" )

# Unbound
chrName <- names(Mmusculus)[1:22]
chrLength <- seqlengths(Mmusculus)[1:22]
```

```
#Initialize
randomGr <- GRanges()
while ( length(randomGr) < sampleSize ) {
     #take a random chromosome, without replacement
     tmpChrName <- sample(chrName, 1)
     tmpChrLength <- chrLength[tmpChrName]
     #Keep the size of the sequence the same as the bound data
     tmpGRanges <- GRanges(seqnames = tmpChrName, strand = "+",
                           IRanges(start = sample(1:(tmpChrLength-30),1),
                                   width = 30))
     #check to make sure there are no overlaps with the bound data
     if( length(findOverlaps(CTCF, tmpGRanges)) == 0 ){
          randomGr <- c( randomGr, tmpGRanges)
          print(length(randomGr))
     }else{
          print(paste(length(randomGr), "There is overlap with bound"))
     }
}
# Overlap checking
findOverlaps(CTCF, randomGr)
# Fasta file generation
getFasta(randomGr, BSgenome.Mmusculus.UCSC.mm10, width = 30, filename = "unbound.fa")


## Merge bound and unbound data
# Combine two datasets and generate one file for linear regression
boundFasta <- readBStringSet("bound.fa")
# Only randomly choose fixed size of data for sampling
boundFasta <- sample(boundFasta, sampleSize)
unboundFasta <- readBStringSet("unbound.fa")
names(unboundFasta) <- paste0( names(unboundFasta), "_unbound")
writeXStringSet( c(boundFasta, unboundFasta), "AllCTCF.fa" )

# Generate binding classfication file
boundTxt <- cbind( sapply(1:length(boundFasta),
                          function(x) as.character(boundFasta[[x]])),
                   matrix(1, length(boundFasta), 1))
unboundTxt <- cbind( sapply(1:length(unboundFasta),
                            function(x) as.character(unboundFasta[[x]])),
                     matrix(0, length(unboundFasta), 1))
write.table(rbind(boundTxt, unboundTxt), "AllCTCF.txt",
            quote = FALSE, col.names = FALSE, row.names = FALSE)
```

Logistic regression models for "1-mer" and "1-mer+shape" features, as performed earlier, are done on this data.

```
## DNAshapeR prediction
BUnbPred <- getShape("AllCTCF.fa")
```

```
## Reading the input sequence......
## Reading the input sequence......
## Reading the input sequence......
## Reading the input sequence......
```

```
## Encode feature vectors
#featureType and featureTypeBoth were generated earlier in the file and is the same
featureVector <- encodeSeqShape("AllCTCF.fa", BUnbPred, featureType)
featureVector2 <- encodeSeqShape("AllCTCF.fa", BUnbPred, featureTypeB)

## Perform logistic regression using caret
exp_data <- read.table("AllCTCF.txt")
# Prepare data
exp_data$V2 <- ifelse(exp_data$V2 == 1 , "Y", "N")
df1 <- data.frame(isBound = exp_data$V2, featureVector)
df2 <- data.frame(isBound = exp_data$V2, featureVector2)
# Set parameters for Caret with 2-fold cross validation
trainControl <- trainControl(method = "cv", number = 2,
                             savePredictions = TRUE, classProbs = TRUE)
# Perform prediction
model1 <- train(isBound~ ., data = df1, trControl = trainControl,
                method = "glm", family = binomial, metric ="ROC")
model2 <- train(isBound~ ., data = df2, trControl = trainControl,
                method = "glm", family = binomial, metric ="ROC")
```

Now we look at the data that emerged, and plot the area under the curve graphs.

```
summary(model1)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.1981  -0.7525   0.0476   0.7565   3.0532
##
## Coefficients: (29 not defined because of singularities)
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -16.566068 394.482722  -0.042 0.966503
## X1           10.639945 394.483010   0.027 0.978482
## X2           10.899977 394.483013   0.028 0.977956
## X3           10.823977 394.483019   0.027 0.978110
## X4           10.850644 394.482995   0.028 0.978056
## X5           -0.063683   0.163370  -0.390 0.696680
## X6            0.500472   0.169778   2.948 0.003200 **
## X7            0.321364   0.170771   1.882 0.059857 .
## X8                  NA         NA      NA       NA
## X9           -0.106125   0.166556  -0.637 0.524013
## X10           0.070783   0.168444   0.420 0.674328
## X11           0.178612   0.168259   1.062 0.288450
## X12                 NA         NA      NA       NA
## X13          -0.408898   0.165572  -2.470 0.013526 *
## X14           0.010793   0.167395   0.064 0.948593
## X15           0.047136   0.169829   0.278 0.781358
## X16                 NA         NA      NA       NA
## X17          -0.306728   0.165000  -1.859 0.063033 .
## X18           0.255931   0.167700   1.526 0.126979
```

```
## X19          0.266578   0.171227   1.557 0.119501
## X20                NA         NA      NA       NA
## X21         -0.375114   0.170037  -2.206 0.027378 *
## X22          0.292143   0.167585   1.743 0.081290 .
## X23          0.338269   0.171532   1.972 0.048605 *
## X24                NA         NA      NA       NA
## X25         -0.071896   0.161197  -0.446 0.655585
## X26          0.038626   0.170222   0.227 0.820491
## X27          0.065368   0.173376   0.377 0.706150
## X28                NA         NA      NA       NA
## X29         -0.303759   0.167111  -1.818 0.069109 .
## X30          0.442175   0.167306   2.643 0.008220 **
## X31          0.314565   0.168266   1.869 0.061560 .
## X32                NA         NA      NA       NA
## X33         -0.024256   0.168452  -0.144 0.885504
## X34          0.590760   0.170940   3.456 0.000548 ***
## X35          0.524074   0.169454   3.093 0.001983 **
## X36                NA         NA      NA       NA
## X37         -0.063412   0.167817  -0.378 0.705532
## X38          0.300156   0.172091   1.744 0.081130 .
## X39          0.592272   0.172161   3.440 0.000581 ***
## X40                NA         NA      NA       NA
## X41         -0.162835   0.168549  -0.966 0.333993
## X42          0.517491   0.170427   3.036 0.002394 **
## X43          0.465120   0.171445   2.713 0.006669 **
## X44                NA         NA      NA       NA
## X45          0.017956   0.170424   0.105 0.916090
## X46          0.171761   0.171846   1.000 0.317551
## X47          0.606486   0.176106   3.444 0.000573 ***
## X48                NA         NA      NA       NA
## X49         -0.024962   0.169739  -0.147 0.883083
## X50          0.217401   0.169894   1.280 0.200678
## X51          0.227213   0.172438   1.318 0.187620
## X52                NA         NA      NA       NA
## X53         -0.265472   0.169480  -1.566 0.117256
## X54          0.453564   0.172859   2.624 0.008693 **
## X55          0.587895   0.169609   3.466 0.000528 ***
## X56                NA         NA      NA       NA
## X57          0.011895   0.167145   0.071 0.943266
## X58          0.479035   0.173731   2.757 0.005828 **
## X59          0.600866   0.170988   3.514 0.000441 ***
## X60                NA         NA      NA       NA
## X61          0.063849   0.170237   0.375 0.707614
## X62          0.813343   0.173059   4.700 2.60e-06 ***
## X63          0.605228   0.174385   3.471 0.000519 ***
## X64                NA         NA      NA       NA
## X65         -0.132047   0.166947  -0.791 0.428973
## X66          0.729997   0.172929   4.221 2.43e-05 ***
## X67          0.577164   0.170467   3.386 0.000710 ***
## X68                NA         NA      NA       NA
## X69          0.027100   0.168712   0.161 0.872386
## X70          0.544812   0.171925   3.169 0.001530 **
## X71          0.668972   0.171898   3.892 9.95e-05 ***
## X72                NA         NA      NA       NA
```

```
## X73          -0.362708   0.169621  -2.138 0.032488 *
## X74           0.224912   0.170943   1.316 0.188270
## X75           0.185640   0.170958   1.086 0.277532
## X76                 NA         NA      NA       NA
## X77           0.115872   0.167632   0.691 0.489423
## X78           0.501926   0.173310   2.896 0.003778 **
## X79           0.524091   0.171944   3.048 0.002303 **
## X80                 NA         NA      NA       NA
## X81           0.137015   0.168545   0.813 0.416259
## X82           0.854755   0.175409   4.873 1.10e-06 ***
## X83           0.790574   0.175991   4.492 7.05e-06 ***
## X84                 NA         NA      NA       NA
## X85          -0.126280   0.169123  -0.747 0.455261
## X86           0.407512   0.170519   2.390 0.016856 *
## X87           0.667291   0.173398   3.848 0.000119 ***
## X88                 NA         NA      NA       NA
## X89          -0.068502   0.167493  -0.409 0.682554
## X90           0.384758   0.172125   2.235 0.025395 *
## X91           0.315219   0.173373   1.818 0.069041 .
## X92                 NA         NA      NA       NA
## X93           0.165101   0.166411   0.992 0.321133
## X94           0.551880   0.173359   3.183 0.001455 **
## X95           0.388392   0.174082   2.231 0.025675 *
## X96                 NA         NA      NA       NA
## X97           0.009834   0.165414   0.059 0.952591
## X98           0.332495   0.172335   1.929 0.053688 .
## X99           0.423654   0.172295   2.459 0.013937 *
## X100                NA         NA      NA       NA
## X101          0.322684   0.167959   1.921 0.054705 .
## X102          0.643699   0.178003   3.616 0.000299 ***
## X103          0.358612   0.176109   2.036 0.041719 *
## X104                NA         NA      NA       NA
## X105         -0.089960   0.167748  -0.536 0.591764
## X106          0.300595   0.171850   1.749 0.080261 .
## X107          0.348163   0.173060   2.012 0.044240 *
## X108                NA         NA      NA       NA
## X109          0.052736   0.166636   0.316 0.751644
## X110          0.308302   0.173631   1.776 0.075796 .
## X111          0.472808   0.175919   2.688 0.007196 **
## X112                NA         NA      NA       NA
## X113          0.145578   0.164404   0.885 0.375891
## X114          0.414593   0.171963   2.411 0.015911 *
## X115          0.487133   0.176430   2.761 0.005761 **
## X116                NA         NA      NA       NA
## X117          0.467473   0.167712   2.787 0.005314 **
## X118          0.562162   0.170212   3.303 0.000958 ***
## X119          0.717826   0.169869   4.226 2.38e-05 ***
## X120                NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2772.6  on 1999  degrees of freedom
```

```
## Residual deviance: 1873.0  on 1908  degrees of freedom
## AIC: 2057
##
## Number of Fisher Scoring iterations: 15
```

```
summary(model2)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -3.2875  -0.6564   0.0310   0.6744   2.5571
##
## Coefficients: (29 not defined because of singularities)
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  128.06081  395.25465   0.324 0.745941
## X1          -171.70853  395.60599  -0.434 0.664260
## X2          -171.41794  395.60510  -0.433 0.664793
## X3          -171.49095  395.60585  -0.433 0.664659
## X4          -171.45026  395.60567  -0.433 0.664734
## X5            -0.10172    0.36298  -0.280 0.779298
## X6             0.40648    0.30042   1.353 0.176047
## X7             0.33619    0.24605   1.366 0.171842
## X8                  NA         NA      NA       NA
## X9             0.23839    0.45693   0.522 0.601860
## X10            0.76990    0.46520   1.655 0.097927 .
## X11            0.70917    0.50235   1.412 0.158038
## X12                 NA         NA      NA       NA
## X13            0.03219    0.49541   0.065 0.948199
## X14            0.42524    0.50502   0.842 0.399779
## X15            0.48497    0.49375   0.982 0.325998
## X16                 NA         NA      NA       NA
## X17           -0.09337    0.48643  -0.192 0.847777
## X18            0.25355    0.50799   0.499 0.617686
## X19           -0.37457    0.51171  -0.732 0.464162
## X20                 NA         NA      NA       NA
## X21            0.10100    0.49979   0.202 0.839854
## X22           -0.29125    0.51468  -0.566 0.571470
## X23           -0.19304    0.50513  -0.382 0.702340
## X24                 NA         NA      NA       NA
## X25           -0.37396    0.48986  -0.763 0.445231
## X26           -0.37794    0.50267  -0.752 0.452134
## X27           -0.62450    0.51751  -1.207 0.227538
## X28                 NA         NA      NA       NA
## X29           -0.28401    0.50785  -0.559 0.575995
## X30           -0.47237    0.50026  -0.944 0.345033
## X31            0.05114    0.49888   0.103 0.918346
## X32                 NA         NA      NA       NA
## X33           -0.78404    0.48730  -1.609 0.107631
## X34            0.27892    0.50656   0.551 0.581901
## X35            0.08124    0.49408   0.164 0.869402
## X36                 NA         NA      NA       NA
```

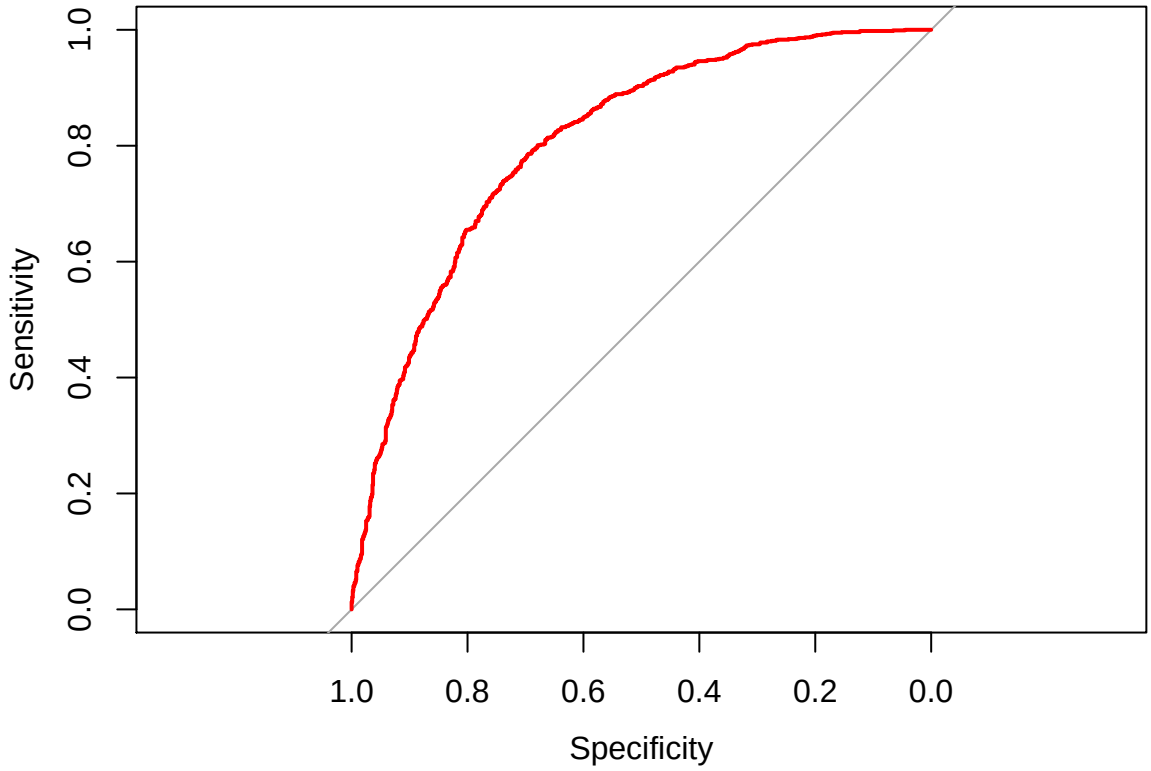
```
## X37           0.44452    0.47712   0.932 0.351505
## X38           0.84790    0.52526   1.614 0.106470
## X39           1.12172    0.48989   2.290 0.022037 *
## X40                NA         NA      NA       NA
## X41          -0.12046    0.48093  -0.250 0.802228
## X42           0.54159    0.51173   1.058 0.289896
## X43           0.49806    0.50202   0.992 0.321145
## X44                NA         NA      NA       NA
## X45           0.14008    0.49458   0.283 0.776998
## X46          -0.14251    0.51186  -0.278 0.780698
## X47           0.64609    0.50279   1.285 0.198790
## X48                NA         NA      NA       NA
## X49          -0.68870    0.49750  -1.384 0.166260
## X50           0.26620    0.51914   0.513 0.608107
## X51          -0.23819    0.48966  -0.486 0.626657
## X52                NA         NA      NA       NA
## X53           0.30874    0.49754   0.621 0.534914
## X54           1.23833    0.50971   2.429 0.015120 *
## X55           1.18576    0.49256   2.407 0.016069 *
## X56                NA         NA      NA       NA
## X57          -0.54300    0.49113  -1.106 0.268894
## X58          -0.82991    0.52732  -1.574 0.115524
## X59          -0.29375    0.49242  -0.597 0.550807
## X60                NA         NA      NA       NA
## X61          -0.23904    0.49840  -0.480 0.631500
## X62           0.05189    0.51350   0.101 0.919507
## X63           0.11086    0.49265   0.225 0.821957
## X64                NA         NA      NA       NA
## X65           0.17899    0.48387   0.370 0.711456
## X66           0.39495    0.50654   0.780 0.435563
## X67           0.47845    0.48971   0.977 0.328565
## X68                NA         NA      NA       NA
## X69           0.12235    0.46633   0.262 0.793036
## X70           0.47403    0.51077   0.928 0.353372
## X71           0.19315    0.48299   0.400 0.689226
## X72                NA         NA      NA       NA
## X73          -0.62573    0.47459  -1.318 0.187346
## X74          -0.23327    0.49290  -0.473 0.636035
## X75          -0.10882    0.48600  -0.224 0.822834
## X76                NA         NA      NA       NA
## X77          -0.52851    0.50095  -1.055 0.291414
## X78          -0.46529    0.51078  -0.911 0.362323
## X79          -0.15571    0.50025  -0.311 0.755603
## X80                NA         NA      NA       NA
## X81           0.36163    0.48103   0.752 0.452177
## X82           1.07770    0.52307   2.060 0.039367 *
## X83           1.15705    0.48790   2.371 0.017718 *
## X84                NA         NA      NA       NA
## X85           0.72282    0.47565   1.520 0.128598
## X86           1.48152    0.50156   2.954 0.003139 **
## X87           1.18664    0.48988   2.422 0.015422 *
## X88                NA         NA      NA       NA
## X89          -0.82126    0.48624  -1.689 0.091221 .
## X90          -0.28394    0.49213  -0.577 0.563965
```

```
## X91          -0.02443    0.50695  -0.048 0.961567
## X92                NA         NA      NA       NA
## X93           0.69092    0.49315   1.401 0.161203
## X94           0.33811    0.51328   0.659 0.510072
## X95          -0.02817    0.49736  -0.057 0.954827
## X96                NA         NA      NA       NA
## X97          -0.34248    0.50560  -0.677 0.498159
## X98          -0.25573    0.50758  -0.504 0.614389
## X99          -0.19003    0.50353  -0.377 0.705884
## X100               NA         NA      NA       NA
## X101          0.53372    0.50192   1.063 0.287628
## X102          0.96298    0.52295   1.841 0.065558 .
## X103          0.65675    0.50752   1.294 0.195656
## X104               NA         NA      NA       NA
## X105         -0.21406    0.48419  -0.442 0.658414
## X106         -0.04597    0.48779  -0.094 0.924925
## X107         -0.33653    0.46969  -0.716 0.473683
## X108               NA         NA      NA       NA
## X109         -0.53149    0.46856  -1.134 0.256659
## X110         -0.81722    0.48971  -1.669 0.095164 .
## X111         -0.38556    0.44905  -0.859 0.390552
## X112               NA         NA      NA       NA
## X113         -0.03341    0.36001  -0.093 0.926063
## X114          0.08340    0.31522   0.265 0.791326
## X115          0.60078    0.29003   2.071 0.038316 *
## X116               NA         NA      NA       NA
## X117          0.75032    0.23898   3.140 0.001692 **
## X118          0.80899    0.22749   3.556 0.000376 ***
## X119          0.96163    0.23763   4.047 5.19e-05 ***
## X120               NA         NA      NA       NA
## X121          0.78267    1.50838   0.519 0.603843
## X122          3.98377    1.59601   2.496 0.012557 *
## X123         -0.08123    1.54485  -0.053 0.958063
## X124          2.95870    1.62150   1.825 0.068051 .
## X125          3.19709    1.64271   1.946 0.051627 .
## X126          0.59511    1.67533   0.355 0.722425
## X127          2.13335    1.58614   1.345 0.178628
## X128         -0.34199    1.63962  -0.209 0.834779
## X129          2.07648    1.66207   1.249 0.211542
## X130          0.71679    1.63978   0.437 0.662019
## X131          0.67732    1.63755   0.414 0.679156
## X132          3.13112    1.62564   1.926 0.054094 .
## X133         -0.36937    1.72423  -0.214 0.830376
## X134         -0.44740    1.66174  -0.269 0.787747
## X135          0.09318    1.56184   0.060 0.952426
## X136         -0.21809    1.66596  -0.131 0.895847
## X137          1.90602    1.58772   1.200 0.229954
## X138         -3.24026    1.63085  -1.987 0.046939 *
## X139          1.45027    1.63591   0.887 0.375335
## X140         -0.82108    1.66118  -0.494 0.621111
## X141          1.83873    1.69434   1.085 0.277825
## X142         -2.22700    1.62338  -1.372 0.170119
## X143          0.17032    1.61287   0.106 0.915897
## X144          1.25849    1.64493   0.765 0.444229
```

```
## X145          1.24042    1.58930   0.780 0.435110
## X146         -0.71776    1.45938  -0.492 0.622841
## X147         -1.16582    1.00098  -1.165 0.244149
## X148          0.04173    1.05556   0.040 0.968464
## X149          0.52146    1.09606   0.476 0.634247
## X150          2.46363    1.09101   2.258 0.023939 *
## X151          0.93197    1.07357   0.868 0.385337
## X152          2.31275    1.05593   2.190 0.028506 *
## X153         -0.21629    1.03289  -0.209 0.834131
## X154         -0.25301    1.06253  -0.238 0.811786
## X155          0.13636    1.03165   0.132 0.894843
## X156          1.01651    1.02081   0.996 0.319355
## X157         -0.05255    1.01030  -0.052 0.958521
## X158         -0.55827    1.03512  -0.539 0.589660
## X159          3.11286    1.06498   2.923 0.003468 **
## X160          0.84379    1.04720   0.806 0.420382
## X161          1.06377    1.04713   1.016 0.309679
## X162          1.64501    1.04378   1.576 0.115022
## X163          0.14139    0.99926   0.141 0.887482
## X164          1.64320    1.05667   1.555 0.119930
## X165         -0.32399    1.03090  -0.314 0.753309
## X166         -0.60896    1.01922  -0.597 0.550185
## X167          0.35960    1.07168   0.336 0.737215
## X168          1.62906    1.08062   1.508 0.131678
## X169          1.05646    1.06110   0.996 0.319432
## X170         -0.22237    1.08597  -0.205 0.837756
## X171          1.65969    1.01328   1.638 0.101436
## X172          2.27360    0.99898   2.276 0.022851 *
## X173         -1.31037    1.20262  -1.090 0.275891
## X174         -2.92456    1.41511  -2.067 0.038766 *
## X175         -1.46464    1.43066  -1.024 0.305953
## X176         -1.75763    1.49900  -1.173 0.240982
## X177         -0.46412    1.46232  -0.317 0.750950
## X178         -1.81390    1.49224  -1.216 0.224152
## X179          0.71574    1.51776   0.472 0.637230
## X180         -1.15374    1.45751  -0.792 0.428604
## X181          0.01774    1.52270   0.012 0.990704
## X182          0.19082    1.50808   0.127 0.899309
## X183          0.71177    1.50704   0.472 0.636714
## X184         -2.15653    1.43521  -1.503 0.132944
## X185          0.19196    1.45395   0.132 0.894962
## X186          1.26969    1.49388   0.850 0.395363
## X187          0.75869    1.47935   0.513 0.608052
## X188          1.17366    1.49737   0.784 0.433149
## X189          0.75325    1.52627   0.494 0.621644
## X190          1.47297    1.46364   1.006 0.314233
## X191          0.33378    1.45822   0.229 0.818952
## X192          0.21869    1.46597   0.149 0.881413
## X193          0.61131    1.51066   0.405 0.685727
## X194          1.27893    1.47012   0.870 0.384328
## X195          2.17433    1.49461   1.455 0.145729
## X196         -0.21853    1.48302  -0.147 0.882853
## X197         -0.17237    1.45636  -0.118 0.905786
## X198          0.37063    1.42166   0.261 0.794320
```

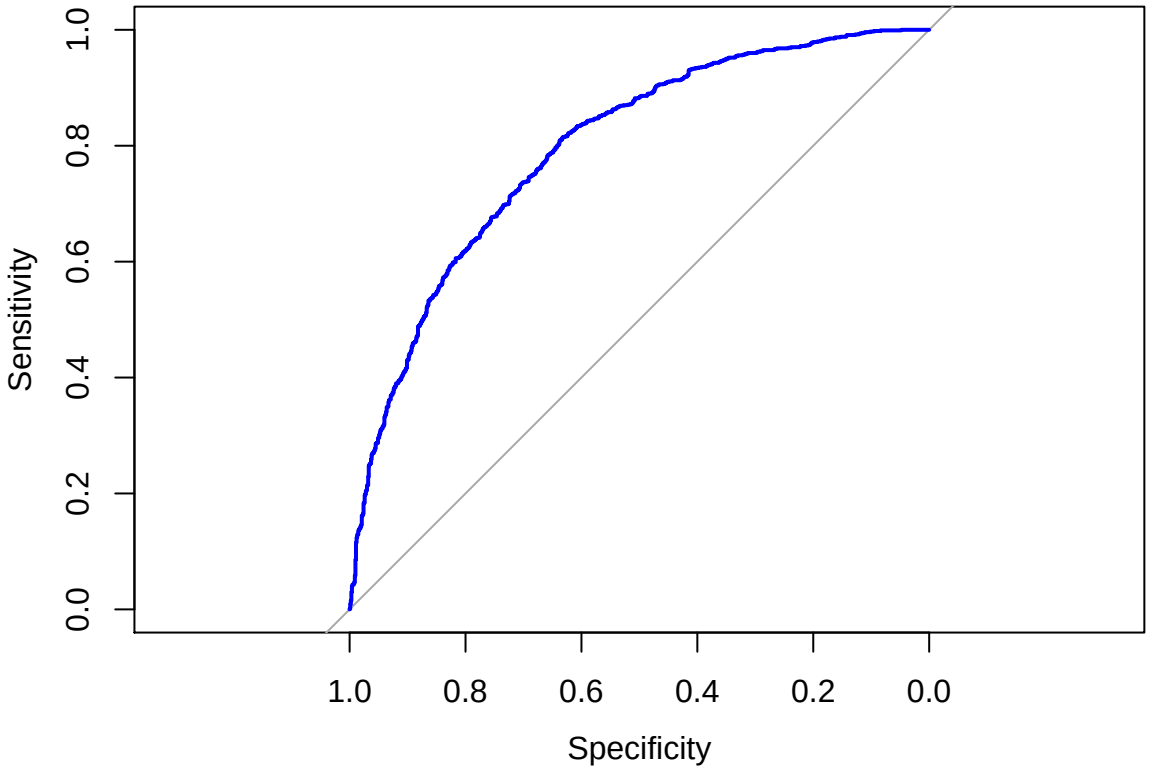
```
## X199           0.39010    1.16088   0.336 0.736842
## X200          -0.65641    0.79528  -0.825 0.409153
## X201           0.07982    0.89738   0.089 0.929120
## X202           0.67254    0.93226   0.721 0.470659
## X203           2.44332    0.97190   2.514 0.011939 *
## X204           3.12833    0.90617   3.452 0.000556 ***
## X205           2.83723    0.95592   2.968 0.002997 **
## X206           1.75703    0.93055   1.888 0.059006 .
## X207           0.59412    0.93984   0.632 0.527286
## X208           1.35980    0.92313   1.473 0.140742
## X209           1.64916    0.92270   1.787 0.073887 .
## X210           0.70228    0.94361   0.744 0.456726
## X211           0.80845    0.93982   0.860 0.389671
## X212           2.87936    0.97801   2.944 0.003239 **
## X213           1.85296    0.91913   2.016 0.043801 *
## X214           1.06825    0.95746   1.116 0.264544
## X215           0.81696    0.92033   0.888 0.374714
## X216           2.02111    0.92436   2.187 0.028779 *
## X217           1.08339    0.94732   1.144 0.252775
## X218          -0.24755    0.94591  -0.262 0.793550
## X219          -0.80934    0.94867  -0.853 0.393586
## X220           1.56318    0.93666   1.669 0.095139 .
## X221           0.38667    0.95225   0.406 0.684700
## X222           1.61826    0.95628   1.692 0.090599 .
## X223           0.96008    0.97177   0.988 0.323165
## X224           1.18648    0.95574   1.241 0.214448
## X225           2.63981    0.90704   2.910 0.003610 **
## X226           1.36590    0.78167   1.747 0.080567 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2772.6  on 1999  degrees of freedom
## Residual deviance: 1717.7  on 1802  degrees of freedom
## AIC: 2113.7
##
## Number of Fisher Scoring iterations: 15
```

```
#model1_2 <- train(isBound~ ., data = df1, trControl=trainControl,
#                  method = "glmnet", tuneGrid = data.frame(alpha = 0, lambda = #c(2^c(-15:15))))
#model2_2 <- train(isBound~ ., data = df1, trControl=trainControl,
#                  method = "glmnet", tuneGrid = data.frame(alpha = 0, lambda = #c(2^c(-15:15))))
#mean(model1_2$results$Rsquared,na.rm=TRUE)
#mean(model2_2$results$Rsquared,na.rm=TRUE)

## Plot the AUROC graph and calculate the AUROC
roc_1 <- roc(model1$pred$obs, model1$pred$Y)
plot(roc_1, col="red")
```




```
##
## Call:
## roc.default(response = model1$pred$obs, predictor = model1$pred$Y)
##
## Data: model1$pred$Y in 1000 controls (model1$pred$obs N) < 1000 cases (model1$pred$obs Y).
## Area under the curve: 0.8074
```

```
roc_2 <- roc(model2$pred$obs, model2$pred$Y)
plot(roc_2, col="blue")
```




```
##
## Call:
## roc.default(response = model2$pred$obs, predictor = model2$pred$Y)
##
## Data: model2$pred$Y in 1000 controls (model2$pred$obs N) < 1000 cases (model2$pred$obs Y).
## Area under the curve: 0.7949
```

Both AUC graphs are much better than 50%. The AUC for the "1-mer" model is 0.8376 and the AUC for "1-mer+shape" is 0.8053. It appears that sequence and shape together do not improve binding prediction for CTCF's DNA binding site. To check these results we should run more simulations, and on a longer sequence rather than 30bp alone. We could use more training sequences to ensure that the model is the most accurate one we can get with the information provided. (in R markdown, the file is rerun every time the pdf is knit, so the random unbound sequence is new for each time I make the PDF. The AUC values therefore might be a bit different than that reported by the above function. I can cache the R blocks but I did so too late for this compiling run.)