1. See Github repository, David\_Manahan\_BISC481
2. a) Two *in-vitro* genomic analyses are SELEX-seq (Systematic Evolution of Ligand by Exponential enrichment) and PBM (Protein Binding Microarray).

In PBM, a microarray (i.e. an assay plate with up to thousands of wells) is set up to have a different double-stranded DNA sequence (e.g. from a genome) in each well. A transcription factor is added to each well and then antibodies are added that target the transcription factor’s epitope. Luminescent signal is indicative of the transcription factor’s binding affinity to the DNA sequence and these affinities are read out.

In SELEX-seq, a pool of synthesized DNA oligonucleotides are sequenced in Round 0. These oligos are ran on a gel with target DNA-binding protein, and based on mobility, DNA-protein interactions can be gauged. The interacting sequences are kept in subsequent rounds and are put through more affinity testing to attain affinities of said protein to varying DNA sequences.

b) In CHIP-seq, an *in vivo* method, cellular, genomic DNA is bound by DNA-binding proteins of interest. The DNA is digested with the proteins still bound, and then the proteins are precipitated out using antibodies. The DNA strands that are precipitated out along with the protein are then sequenced to identify DNA-protein binding affinities.

c) As an in-vivo experiment, ChIP-seq has the advantage of being in the natural environment for the DNA and binding proteins, which could give more reliable results. However, compared to PBM and SELEX-seq, ChIP-seq is less quantitative (i.e. in terms of binding affinity).

PBM and SELEX-seq are both more quantitative, with SELEX-seq having more stringent and longer pipeline than PBM, hence having the drawback of taking longer and more resources to complete. PBM however can screen hundreds (thousands?) of sequences and proteins at a time, with enough automation.

1. MLR package was used (downloaded from provided link).

To access *Mad*, *Max*, and *Myc* fasta files, the working directory set was as follows:

C:\Users\David\Desktop\BISC481\BISC481-master\gcPBM.

This was inserted at the highlighted yellow part of the downloaded code, which was then run using RStudio:

######################################

# 12.10.2016

# Multiple Linear Regression (MLR) example

# BISC 481

######################################

## Install packages

# Bioconductor

source("https://bioconductor.org/biocLite.R")

biocLite()

a# DNAshapeR

biocLite("DNAshapeR")

# Caret

install.packages("caret")

## Initialization

library(DNAshapeR)

library(caret)

workingPath <- "C:\\Users\\David\\Desktop\\BISC481\\BISC481-master\\gcPBM\\"

## Predict DNA shapes

fn\_fasta <- paste0(workingPath, "Mad.txt.fa")

pred <- getShape(fn\_fasta)

## Encode feature vectors

featureType <- c("1-mer")

featureVector <- encodeSeqShape(fn\_fasta, pred, featureType)

head(featureVector)

## Build MLR model by using Caret

# Data preparation

fn\_exp <- paste0(workingPath, "Mad.txt")

exp\_data <- read.table(fn\_exp)

df <- data.frame(affinity=exp\_data$V2, featureVector)

# Arguments setting for Caret

trainControl <- trainControl(method = "cv", number = 10, savePredictions = TRUE)

# Prediction without L2-regularized

model <- train (affinity~ ., data = df, trControl=trainControl,

method = "lm", preProcess=NULL)

summary(model)

# Prediction with L2-regularized

model2 <- train(affinity~., data = df, trControl=trainControl,

method = "glmnet", tuneGrid = data.frame(alpha = 0, lambda = c(2^c(-15:15))))

model2

result <- model2$results$Rsquared[1]

1. Refer to scripted code in 3). Highlighted in RED were the inputs varied between Mad, Max, and Myc in combination with “1-mer” and “1-mer”, “shape” in BLUE. This line of code gave R2-values, which were then compiled in the following table:

|  |  |  |
| --- | --- | --- |
| **R2-values table** | 1-mer | 1-mer+shape |
| Mad | 0.775 | 0.863 |
| Max | 0.786 | 0.864 |
| Myc | 0.778 | 0.779 |

1. Below is the code used to generate the plot of “1-mer” vs “1-mer+shape.” Note that data1 corresponds to “1-mer+shape” R2-values, while data2 refers to “1-mer” R2-values. The highlighted portion is where the calculated R2-values from 4) were inserted.

######################################

# 11.10.2016

# Multiple Linear Regression (MLR) example

# BISC 481

######################################

## Install and initialize packages

install.packages("ggplot2")

install.packages("grid")

library(ggplot2)

library(grid)

## Theme

my.theme <- theme(

plot.margin = unit(c(0.1, 0.5, 0.1, 0.1), "cm"),

axis.text = element\_text(colour="black", size=12),

axis.title.x = element\_text(colour="black", size=12),

axis.title.y = element\_text(colour="black", size=12),

panel.grid.major = element\_blank(),

panel.grid.minor = element\_blank(),

panel.background = element\_blank(),

axis.line = element\_line(colour = "black"),

axis.text = element\_text(colour ="black"),

axis.ticks = element\_line(colour = "black")

)

## Data preparation #data1X #data2Y

data1 <- c(0.862902,0.864174, 0.7786731)

data2 <- c(0.77482, 0.778076, 0.785727)

## Ploting

ggplot() +

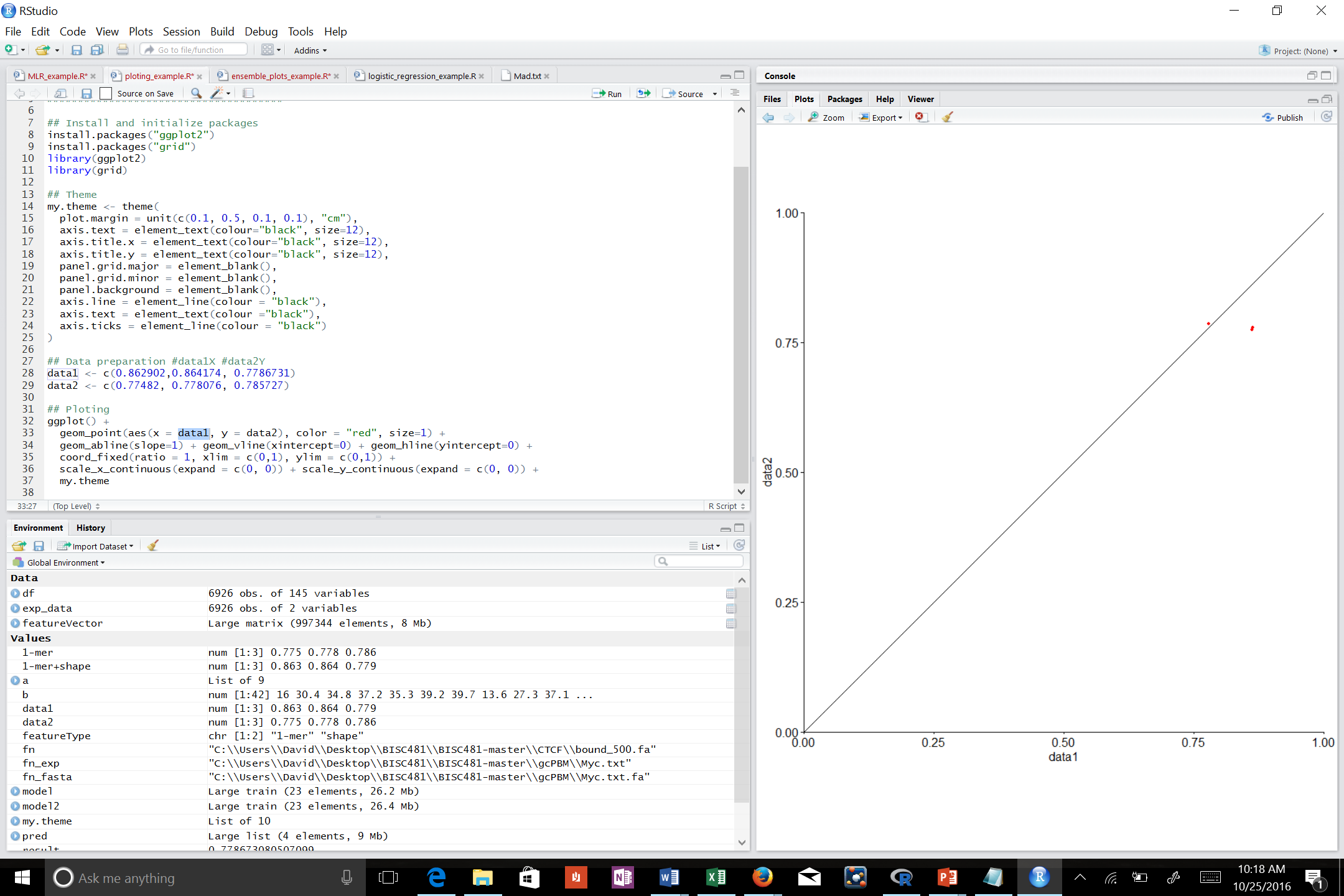
geom\_point(aes(x = data1, y = data2), color = "red", size=1) +

geom\_abline(slope=1) + geom\_vline(xintercept=0) + geom\_hline(yintercept=0) +

coord\_fixed(ratio = 1, xlim = c(0,1), ylim = c(0,1)) +

scale\_x\_continuous(expand = c(0, 0)) + scale\_y\_continuous(expand = c(0, 0)) +

my.theme



This plot, 1-mer vs 1-mer+shape, is a comparison of performance for the two models derived in question 5. Two red data points are seen on the plot but in fact these are 6 data point (Mad, Myc, and Max proteins with the 1-mer and 1-mer+shape model for each). The closer a data point is to the linear regression model line, the less of a difference there is between the viability of the two different methods of modelling for that protein. Therefore, the data point on the line indicates that Myc (with R2 values of 0.778 and 0.779) performs equally well regardless of method chosen. The off-line cluster indicates that Mad and Max perform better with the 1-mer+shape method instead of the 1-mer method.

1. Working directory set to “C:\\Users\\David\\Desktop\\BISC481\\BISC481-master\\CTCF\\” to access bound\_500.fa and unbound\_500.fa files.
2. The following lines of code as accessed and ran in RStudio:

######################################

# 01.10.2016

# Emsemble plots example

# BISC 481

######################################

# Initialization

library(DNAshapeR)

# Extract sample sequences

fn <- "C:\\Users\\David\\Desktop\\BISC481\\BISC481-master\\CTCF\\bound\_500.fa"

# Predict DNA shapes

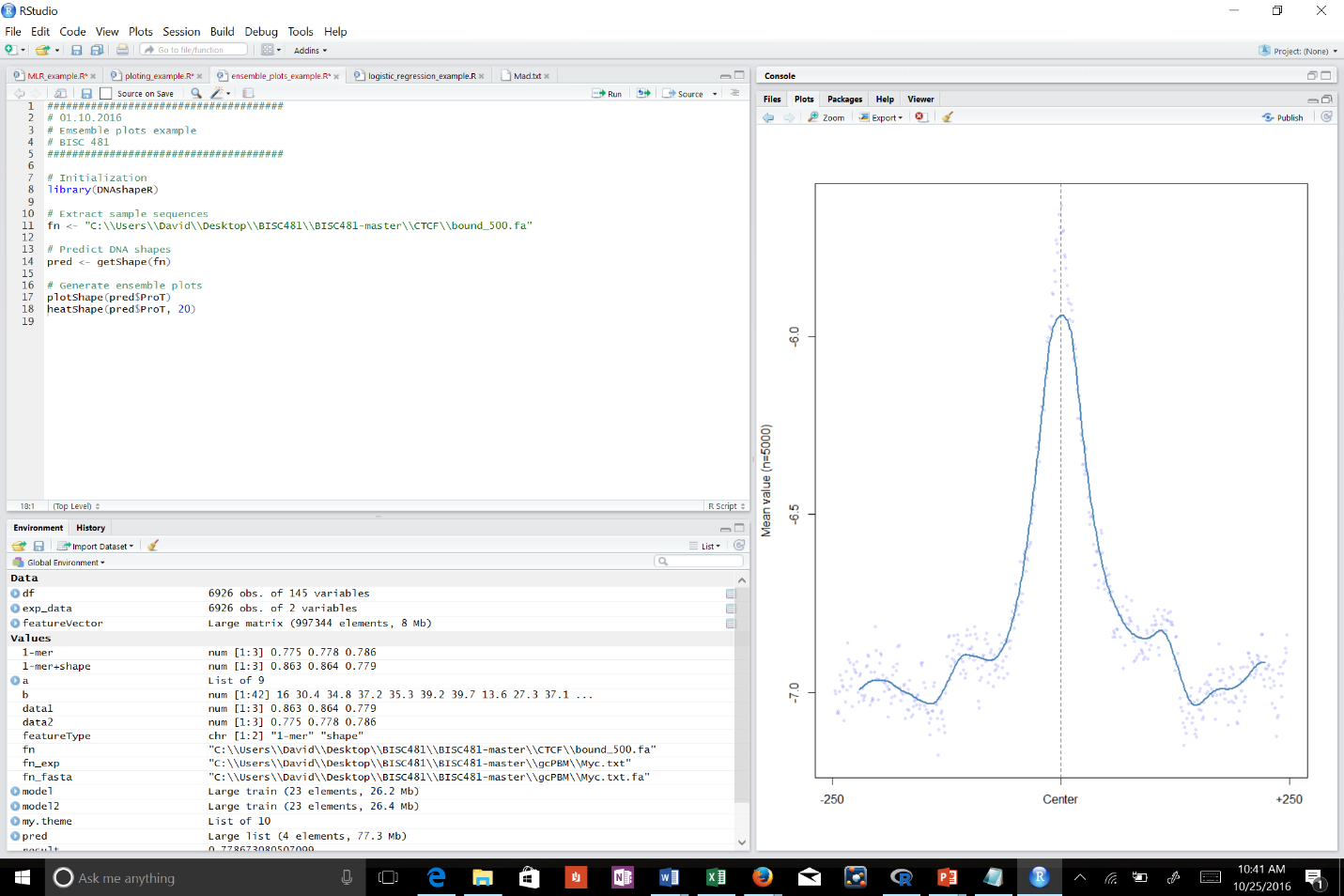
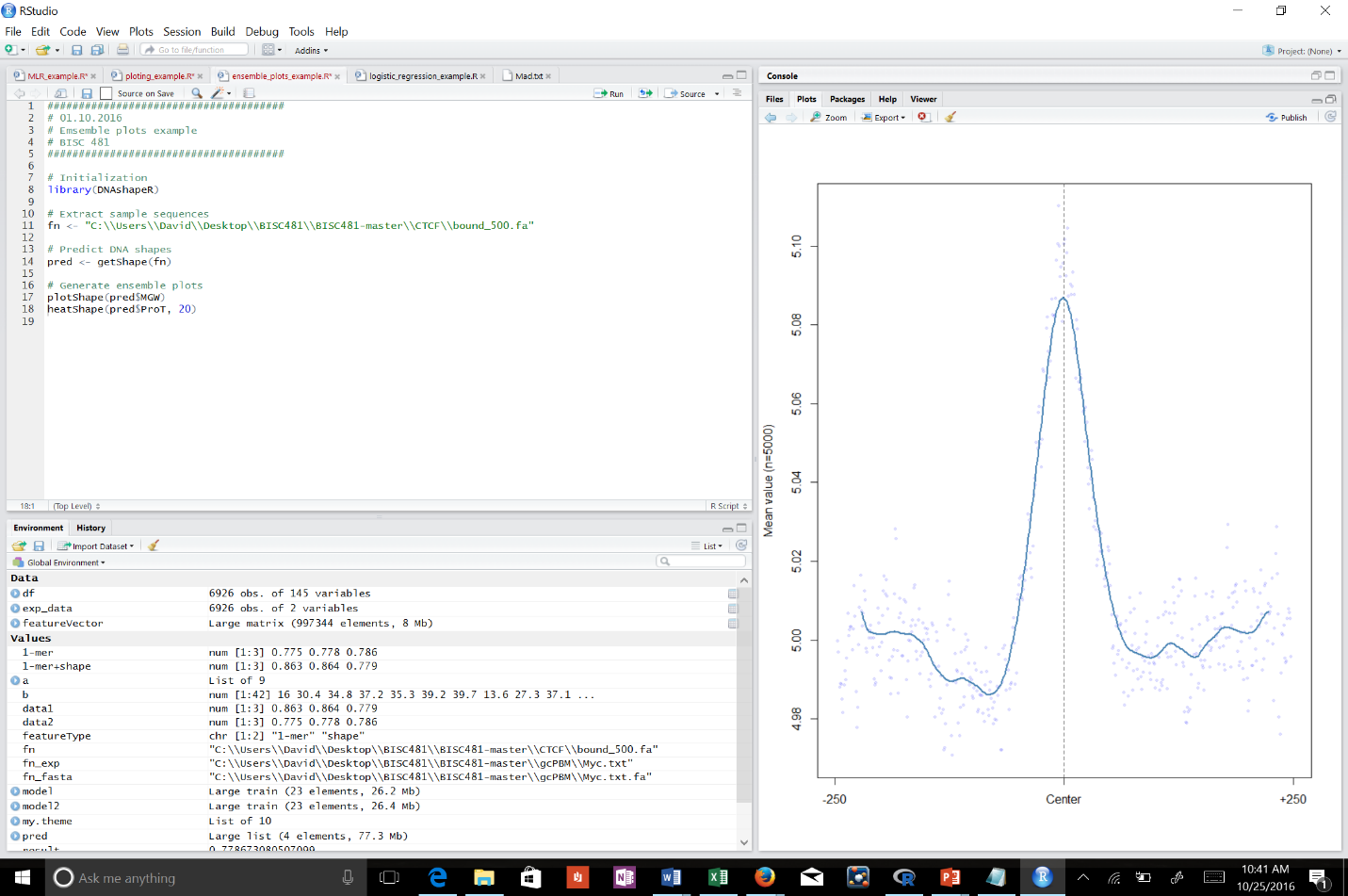
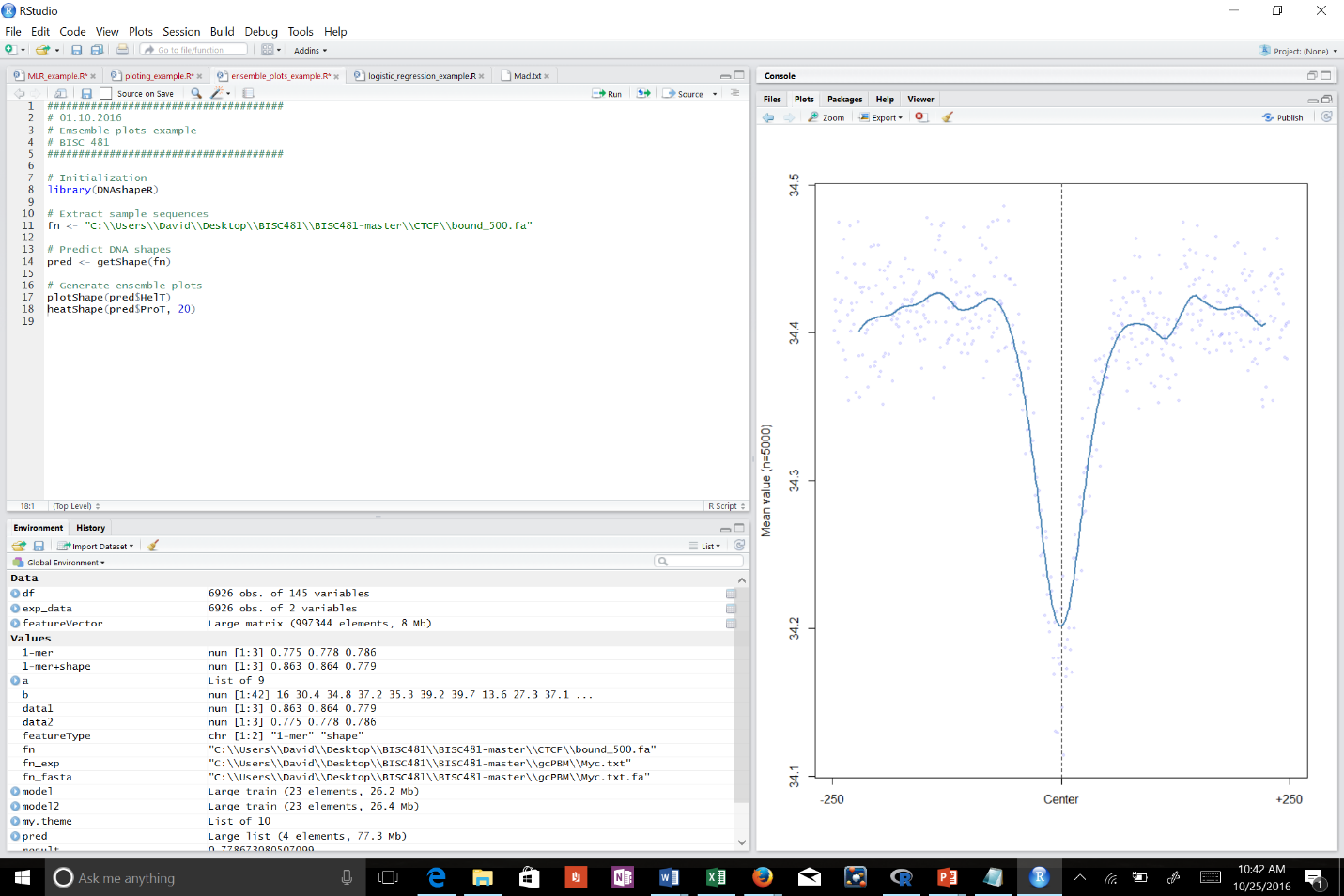
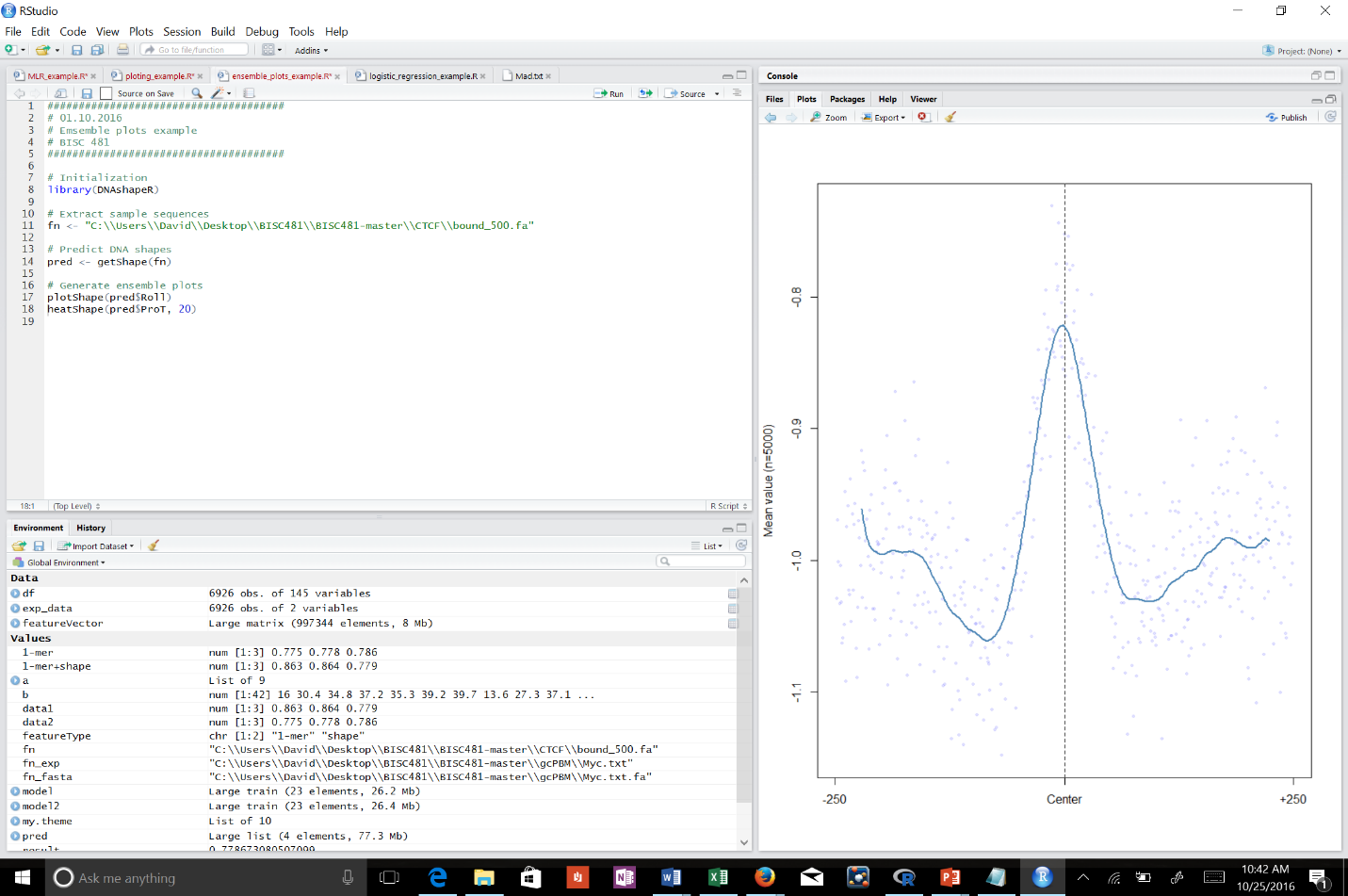
pred <- getShape(fn)

# Generate ensemble plots

plotShape(pred$HelT)

heatShape(pred$ProT, 20)

Highlighted in GREEN are the two functions ran (separately) to generate the Plots and Heat Maps. The shape parameters, i.e. MGW (minor groove width), Roll, HelT (Helix Twist) and propeller twist (ProT) were alternated in the spots highlighted in BLUE. Highlighted in YELLOW is where to input either “bound” or “unbound” data.

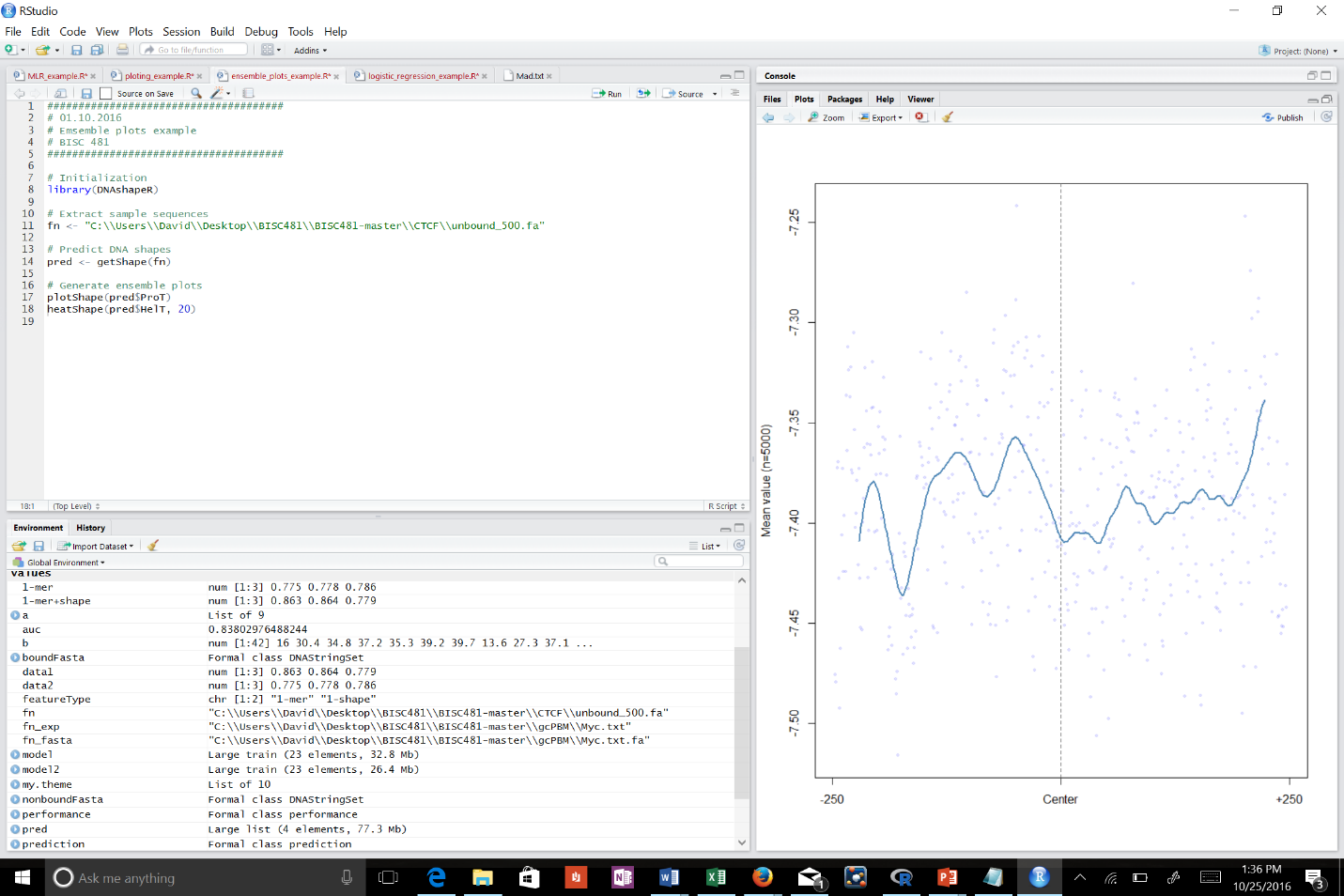
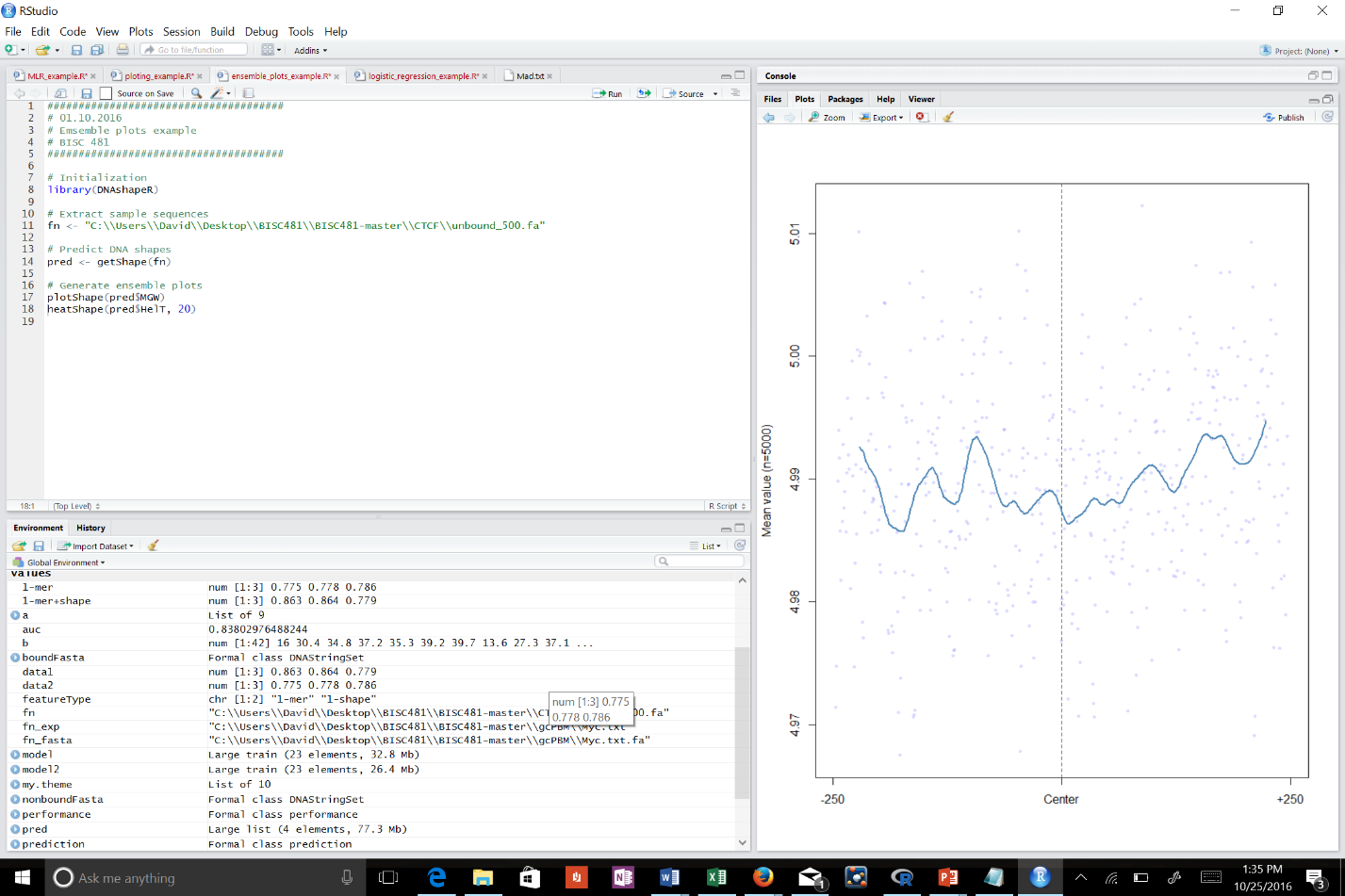
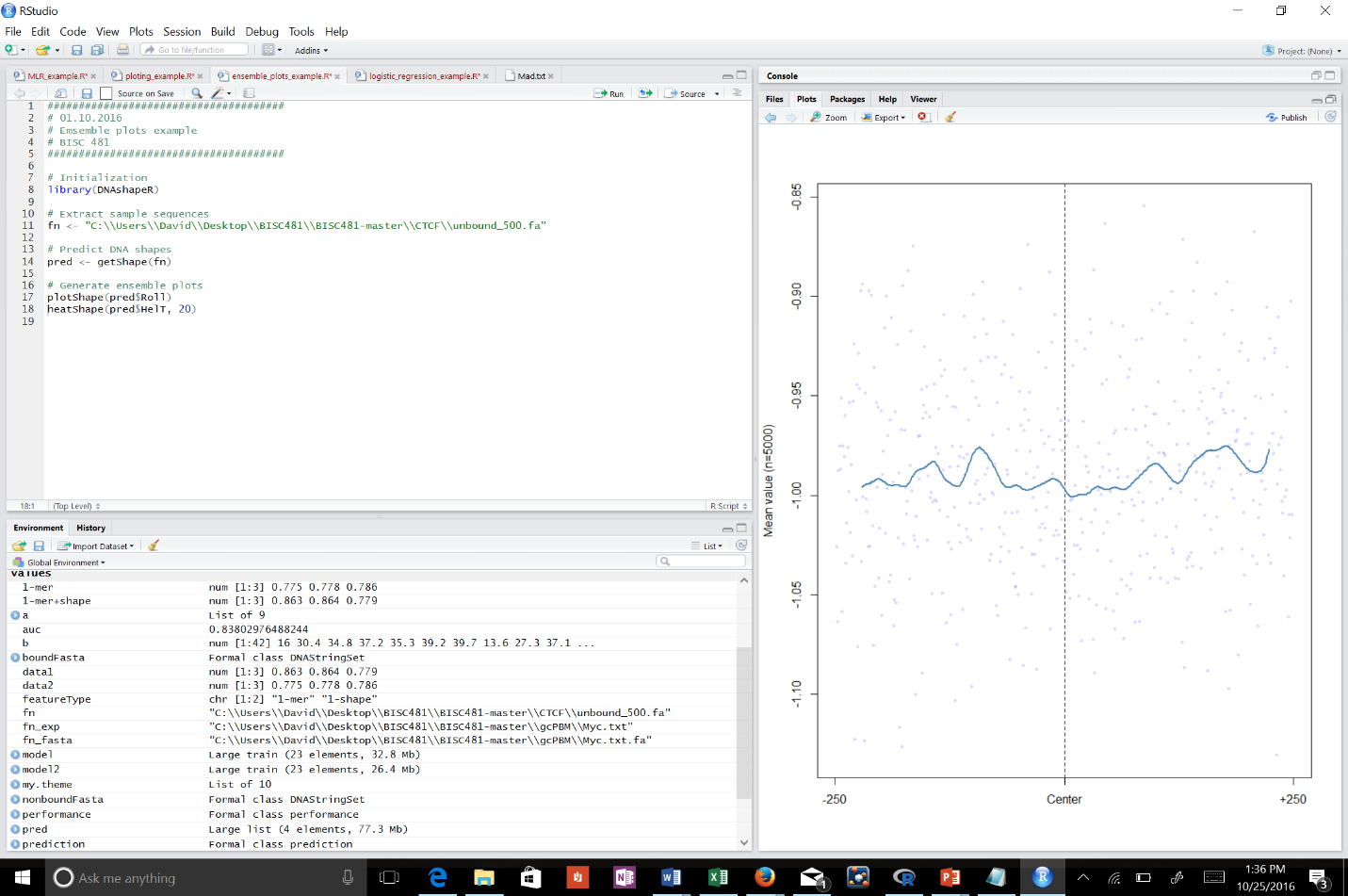
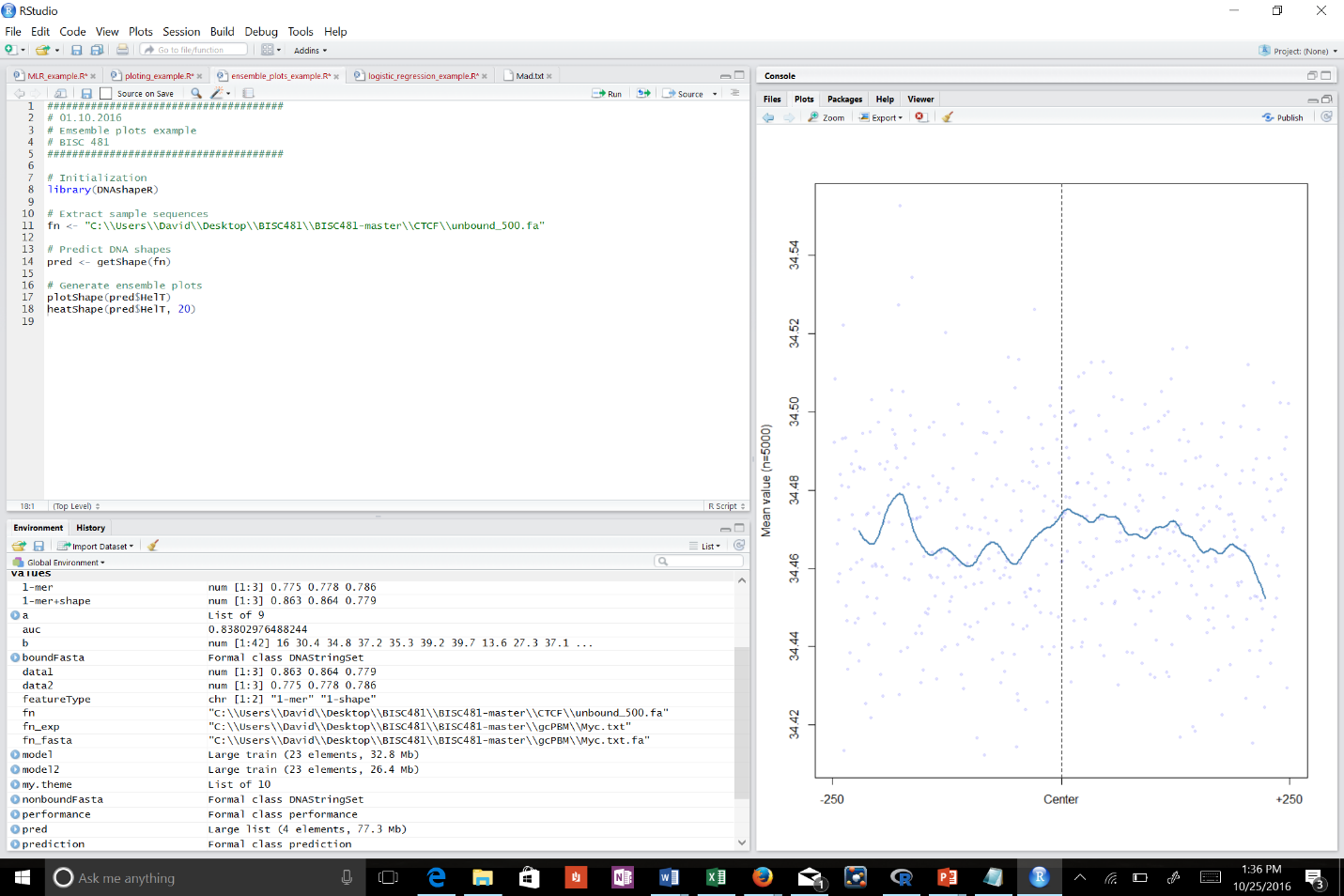


Plot of bound, helix twist

Plot of bound, roll

Plot of bound, propeller twist

Plot of bound, minor groove width

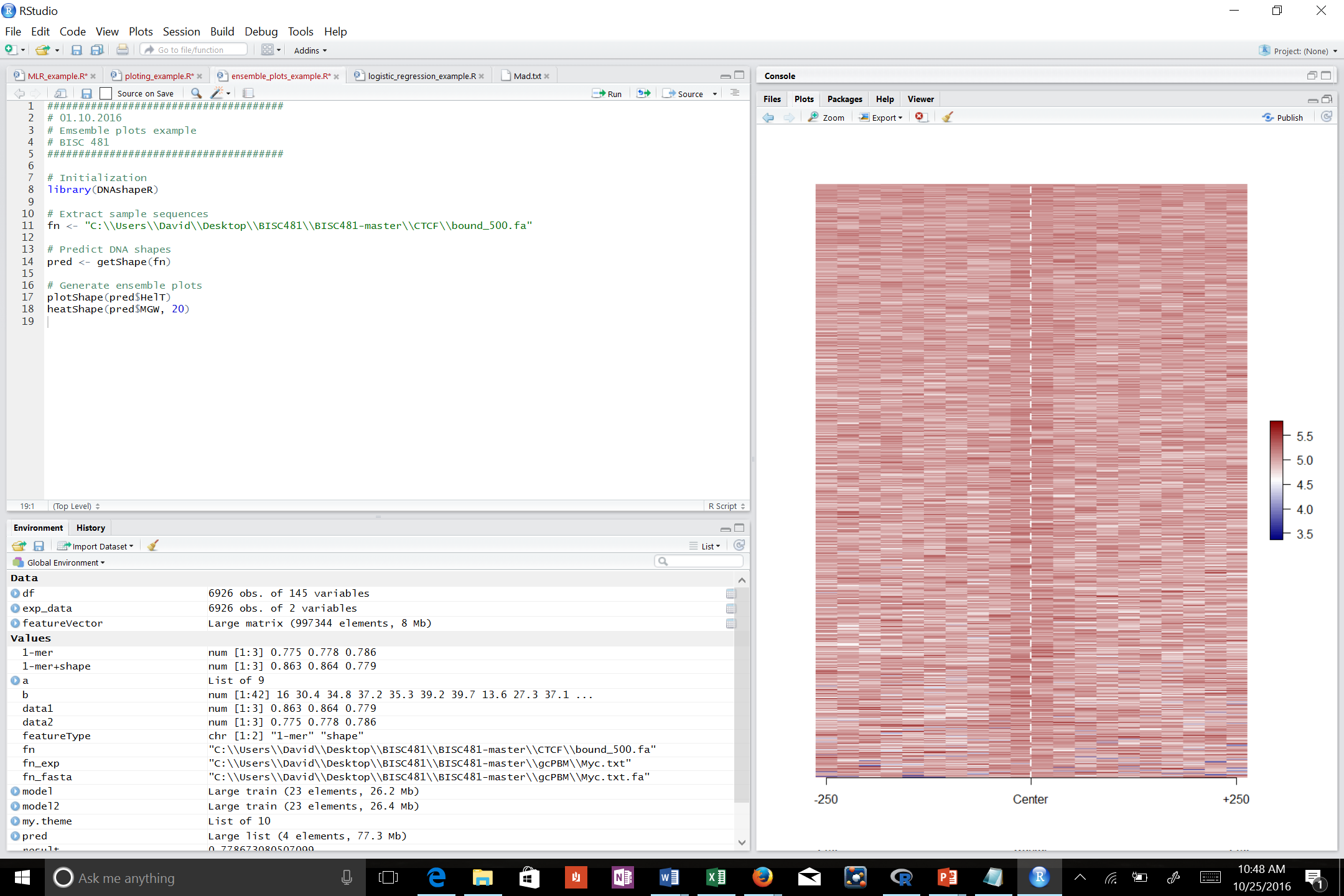


Plot unbound, helix twist

Plot unbound, roll

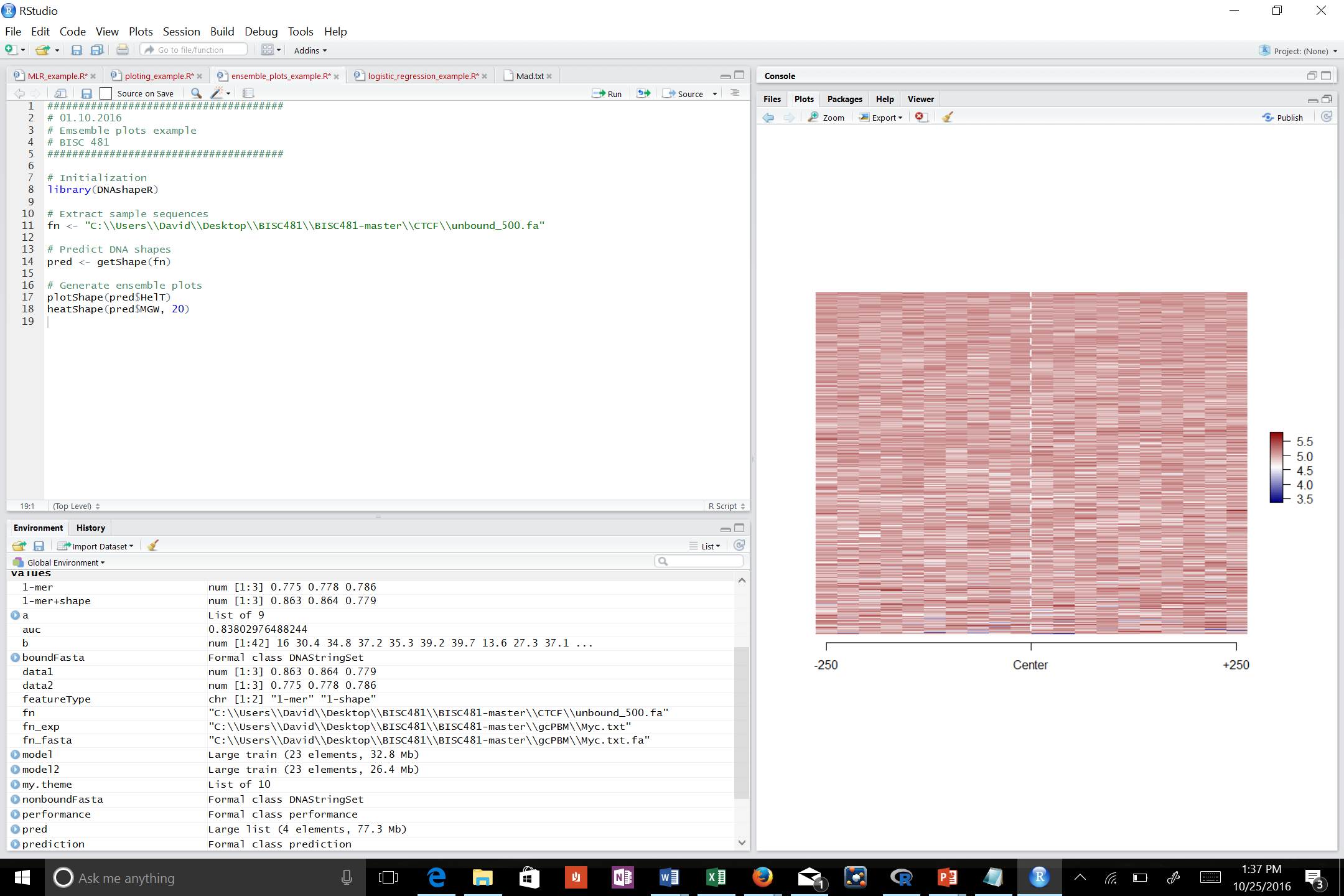
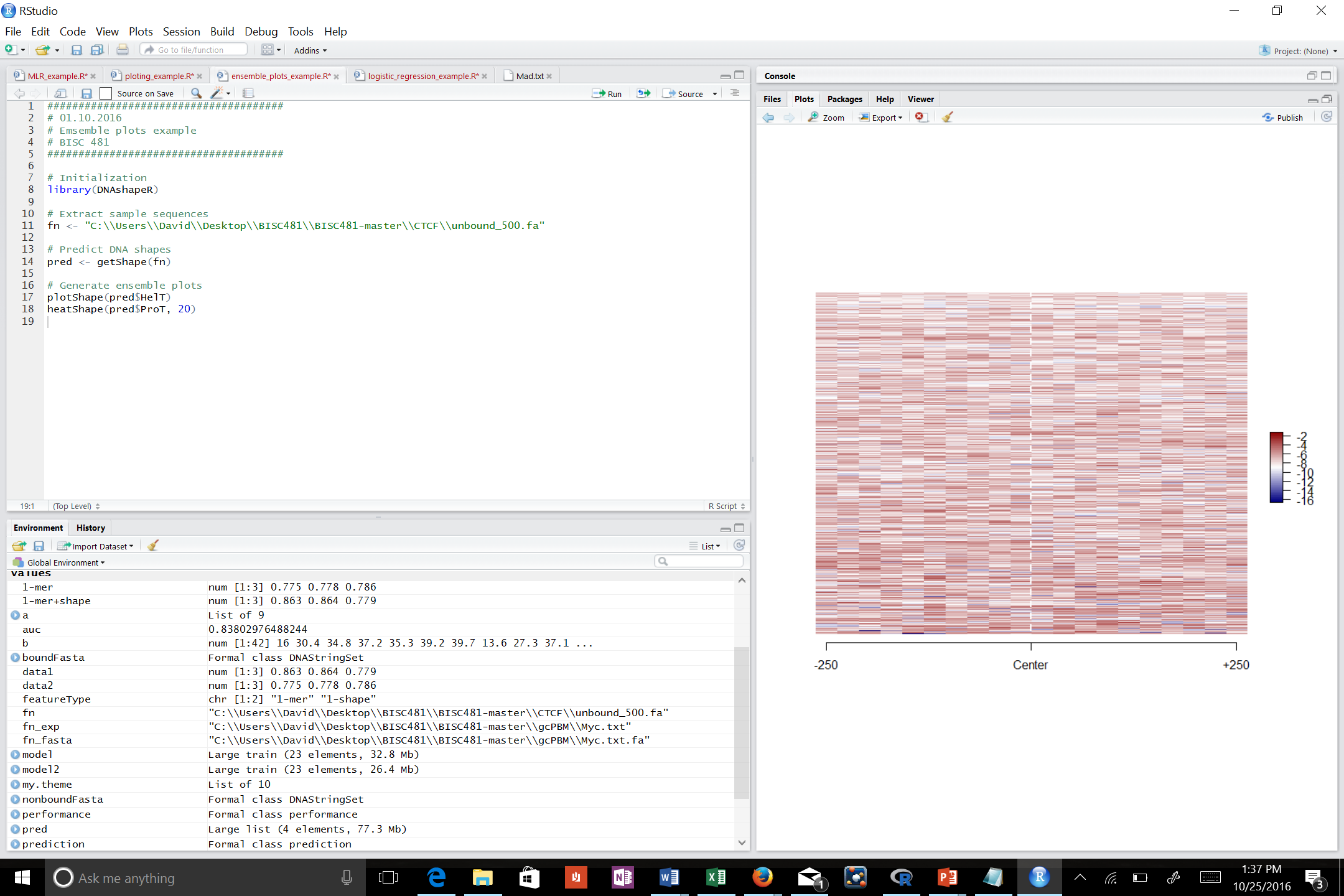
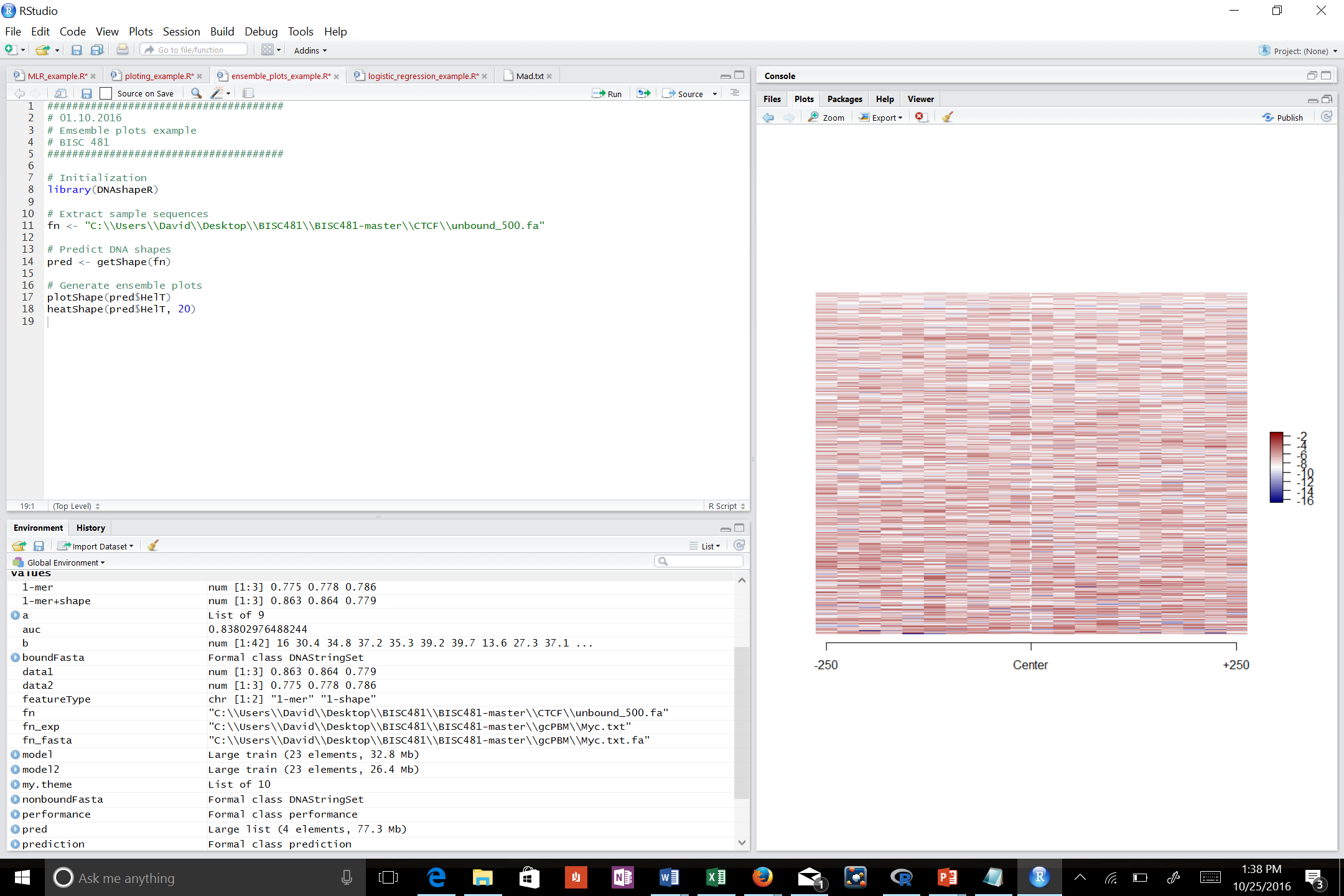
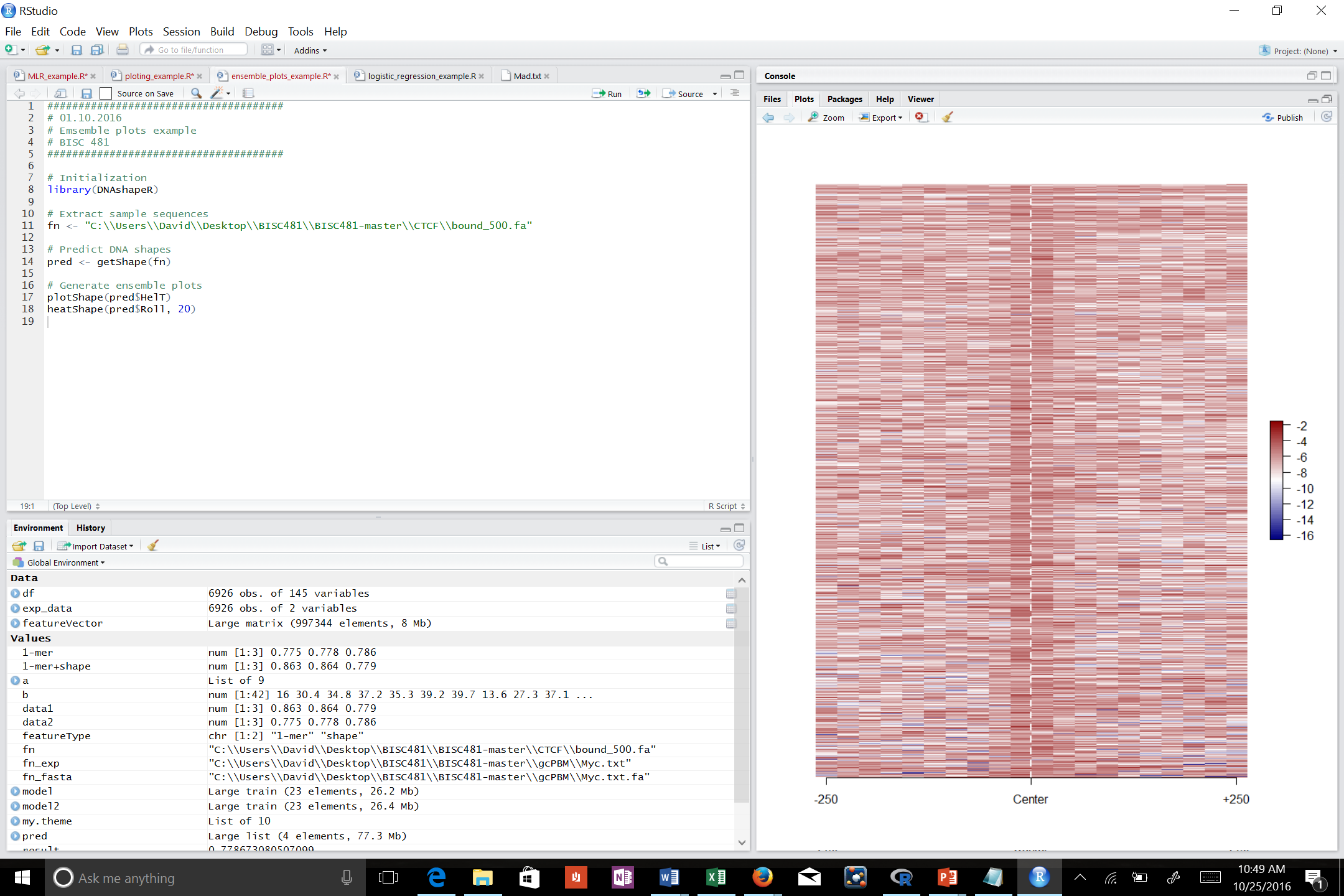
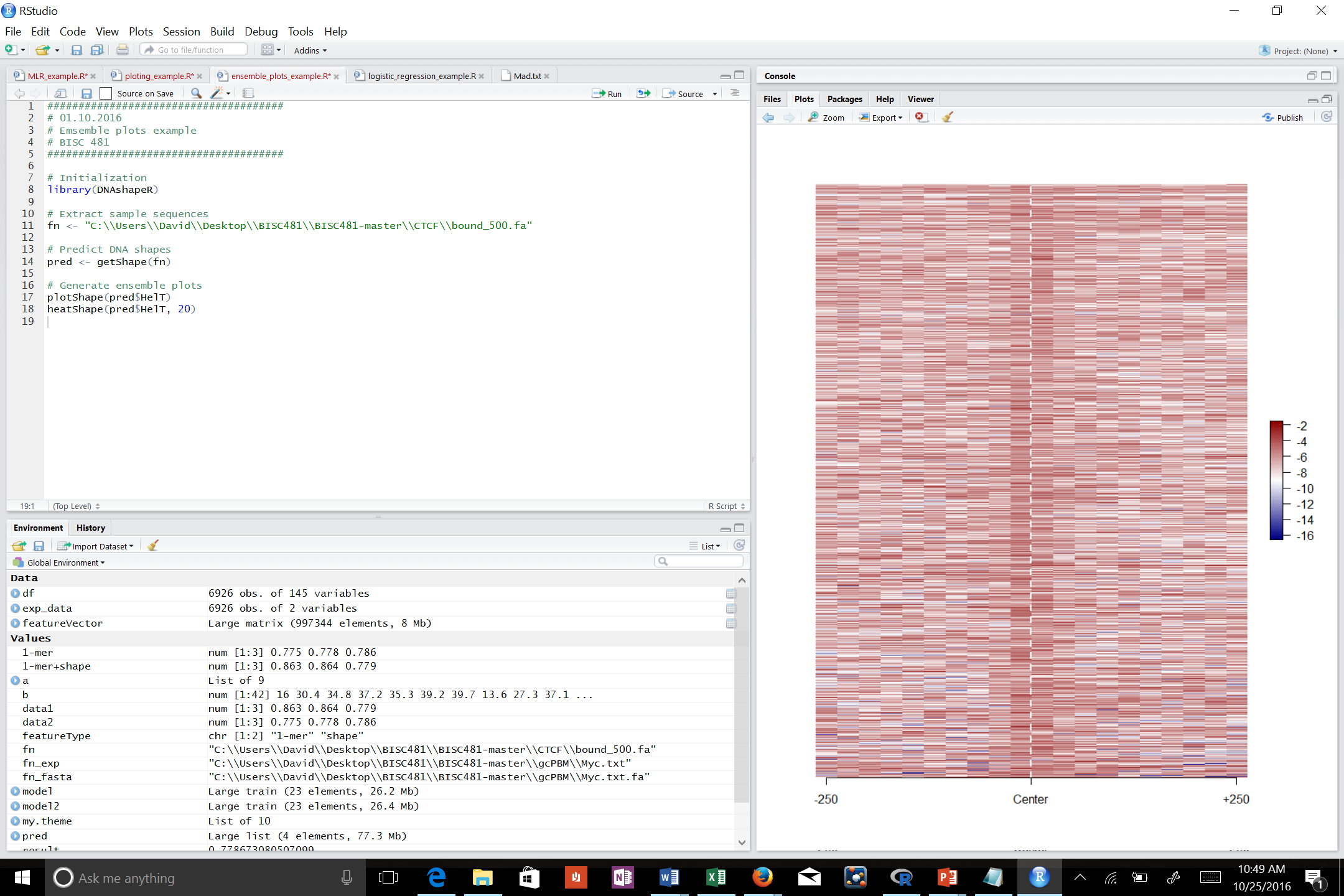
Plot unbound, propeller twist

Plot unbound, minor groove width

]\\]

Heatmap of bound, propeller twist

Heatmap of bound, minor groove width



Heatmap of bound, helix twist

Heatmap of bound, roll

Heatmap unbound, helix twist

Heatmap unbound, roll

Heatmap unbound, minor groove width

Heatmap unbound, propeller twist

These sets of graphs are comparing bound and unbound regions of DNA (i.e. bound or unbound to proteins of interest), thus evaluating how well the written code can distinguish between bound DNA (i.e. DNA sequences of interest, as they interact with protein) and unbound DNA (“junk” DNA between bound regions). Looking at the plots, one can see that bound regions are much more structured, following a notable pattern focused around the center (i.e. no twist or no roll, etc.) as opposed to unbound DNA, which has no discernable pattern present in any of its plots. In addition, looking at heatmaps, a consistent pattern can be seen in bound DNA, dark regions running up through the middle of the plots, indicating a complex, while unbound regions have no discernable pattern once again. Namely, this demonstrates the machine’s ability to distinguish these two types of DNA for experiments.

1. Below is the code used to build and plot the logistic-regression model and calculate its AUC score: (highlighted portion was modified to shown working path)

######################################

# 12.10.2016

# Logistic regression on ChIP-seq data

# BISC 481

######################################

## Install packages

install.packages("caret")

install.packages("e1071")

install.packages("ROCR")

biocLite("Biostrings")

## Initialization

library(DNAshapeR)

library(caret)

library(ROCR)

library(Biostrings)

workingPath <- "C:\\Users\\David\\Desktop\\BISC481\\BISC481-master\\CTCF\\"

## Generate data for the classifcation (assign Y to bound and N to non-bound)

# bound

boundFasta <- readDNAStringSet(paste0(workingPath, "bound\_30.fa"))

sequences <- paste(boundFasta)

boundTxt <- data.frame(seq=sequences, isBound="Y")

# non-bound

nonboundFasta <- readDNAStringSet(paste0(workingPath, "unbound\_30.fa"))

sequences <- paste(nonboundFasta)

nonboundTxt <- data.frame(seq=sequences, isBound="N")

# merge two datasets

writeXStringSet( c(boundFasta, nonboundFasta), paste0(workingPath, "ctcf.fa"))

exp\_data <- rbind(boundTxt, nonboundTxt)

## DNAshapeR prediction

pred <- getShape(paste0(workingPath, "ctcf.fa"))

## Encode feature vectors

featureType <- c("1-mer", "1-shape")

featureVector <- encodeSeqShape(paste0(workingPath, "ctcf.fa"), pred, featureType)

df <- data.frame(isBound = exp\_data$isBound, featureVector)

## Logistic regression

# Set parameters for Caret

trainControl <- trainControl(method = "cv", number = 10,

savePredictions = TRUE, classProbs = TRUE)

# Perform prediction

model <- train(isBound~ ., data = df, trControl = trainControl,

method = "glm", family = binomial, metric ="ROC")

summary(model)

## Plot AUROC

prediction <- prediction( model$pred$Y, model$pred$obs )

performance <- performance( prediction, "tpr", "fpr" )

plot(performance)

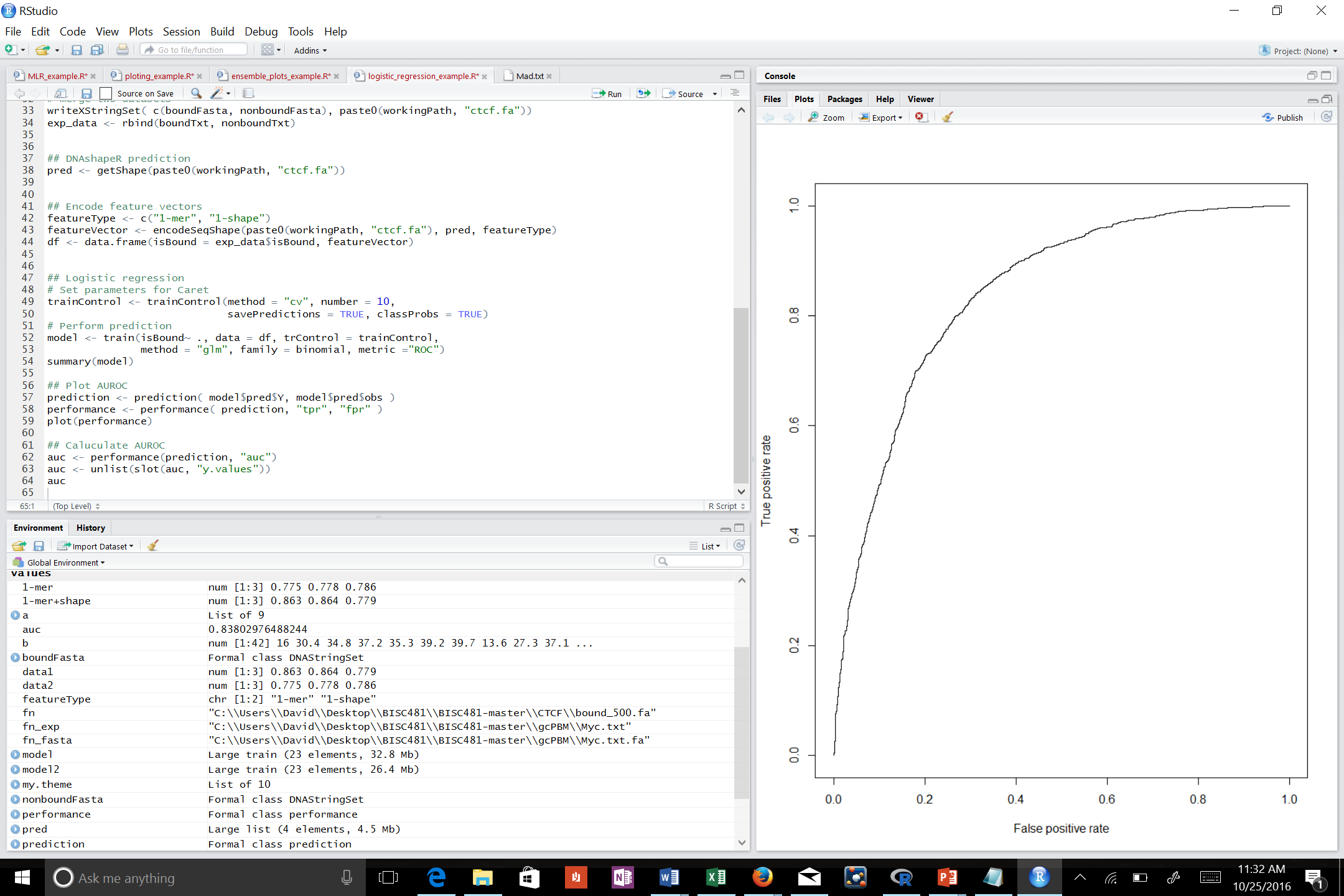
## Caluculate AUROC

auc <- performance(prediction, "auc")

auc <- unlist(slot(auc, "y.values"))

auc

AUC output was 0.8380298.



This plot demonstrates the machine’s likelihood of generating a true positive (i.e. a real result) versus a false positive (i.e. a fake/incorrect result). AUC, or area under the curve, is a classfier for the model being tested: a high AUC means that the rate of true positive results is much higher than the rate of false positives, an important quality for a model. This can be seen in the shape of the graph as well, as the slope (i.e. rise over run, respectively true positive over false positive) is steep, indicating a greater increase in true positives in comparison to false positives. The machine’s ability to distinguish bound vs unbound regions of DNA is rather good.