Phoneme and Nonparametric Methods

Rob Leonard (robleonard@tamu.edu (mailto:robleonard@tamu.edu))

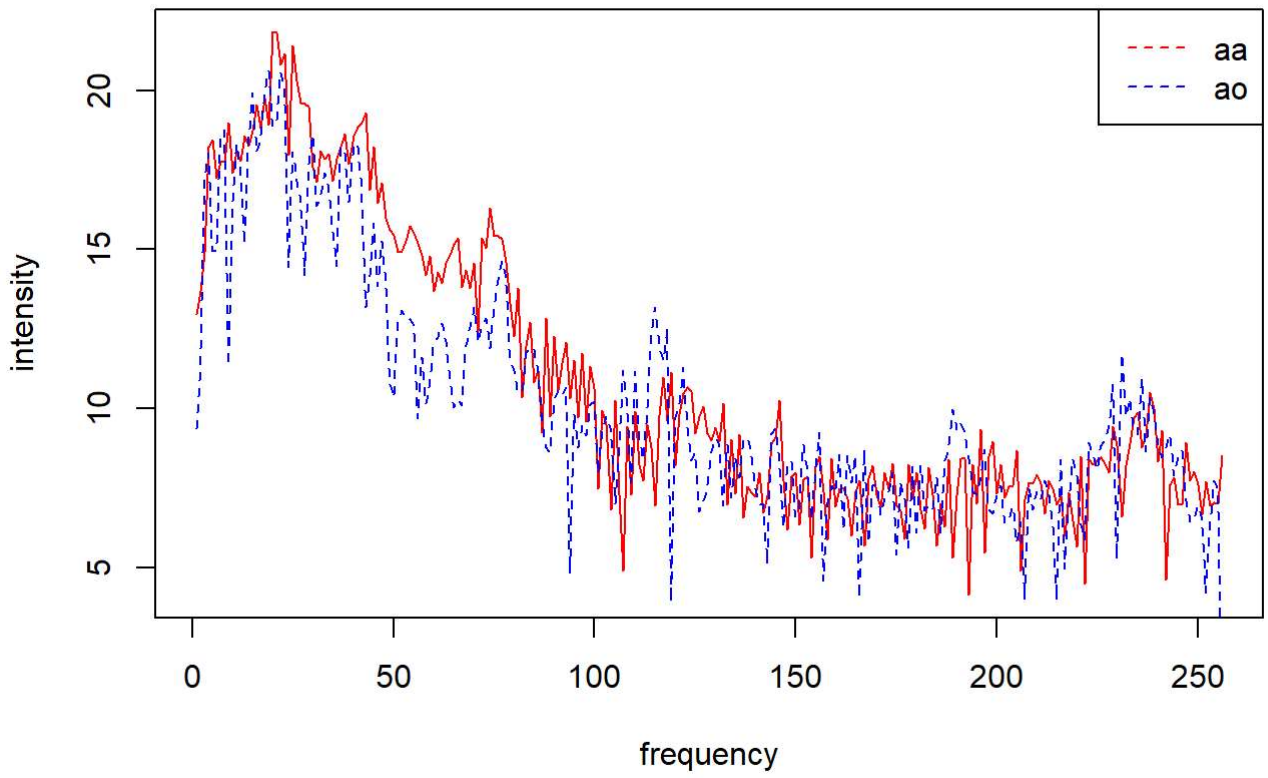
Introduction

There is a major interest in automated translation of audio language into written language. We will look into the most atomic aspect of this problem: phoneme (pronounced ‘fo-neem’) classification. In the American English language, the phonemes ‘aa’ (http://www.speech.cs.cmu.edu/cgi-bin/pronounce#phones) and ‘ao’ (http://www.speech.cs.cmu.edu/cgi-bin/pronounce#phones) are difficult to distinguish. We have a data set of recorded pronunciations of these phonemes along with the label of what the speaker was intending to pronounce. Furthermore, this (analog) audio signal is converted to a digital representation in terms of frequencies. Here are examples of what the data look like:

```
out = read.csv('phoneme.csv',  stringsAsFactors = FALSE)
X   = select(out, -row.names, -g, -speaker)
Y   = select(out, g)

X    = filter(X, Y == 'aa' | Y == 'ao')
Xmat = as.matrix(X)
Y    = as.factor(filter(Y, Y == 'aa' | Y == 'ao') %>% unlist)

# The 'min' parts are just to make sure I'm getting one 'aa' curve and one 'ao' curve
phonemeExample1 = as.numeric(X[min(which(Y == 'aa')),])
phonemeExample2 = as.numeric(X[min(which(Y == 'ao')),])
plot(1:256, phonemeExample1, col = 'red', type= 'l',
     xlab = 'frequency', ylab = 'intensity')
lines(1:256, phonemeExample2, col = 'blue', lty = 2)
legend('topright', legend = c('aa', 'ao'), col = c('red','blue'), lty=2)
```



From this plot, we can see the difficulty in distinguishing these sounds: they are very similar, especially at higher frequencies. (Note that this plot is technically called a 'log-periodogram' if you're interested in delving into this topic further)

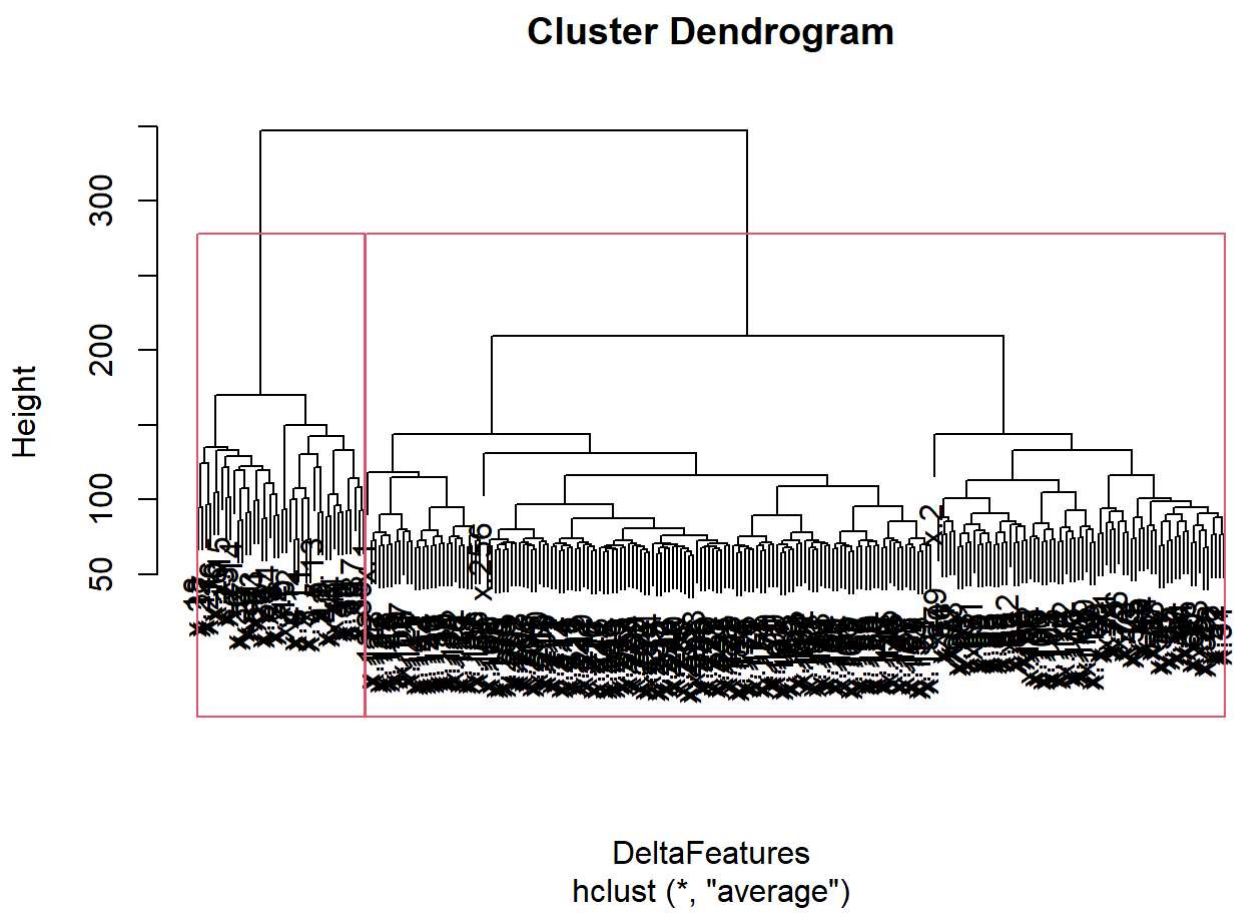
Clustering

Let's look into clustering.

Clustering the features (which are frequencies)

To help with clustering the observations, let's look at clustering the features. The reason for this is it appears, based on the first plot we made, that some of the frequencies are better at discriminating between the phonemes. Let's use hierarchical clustering and average linkage to find two groups of features (reminder: we can cluster features by applying clustering procedures to the transpose of the feature matrix).

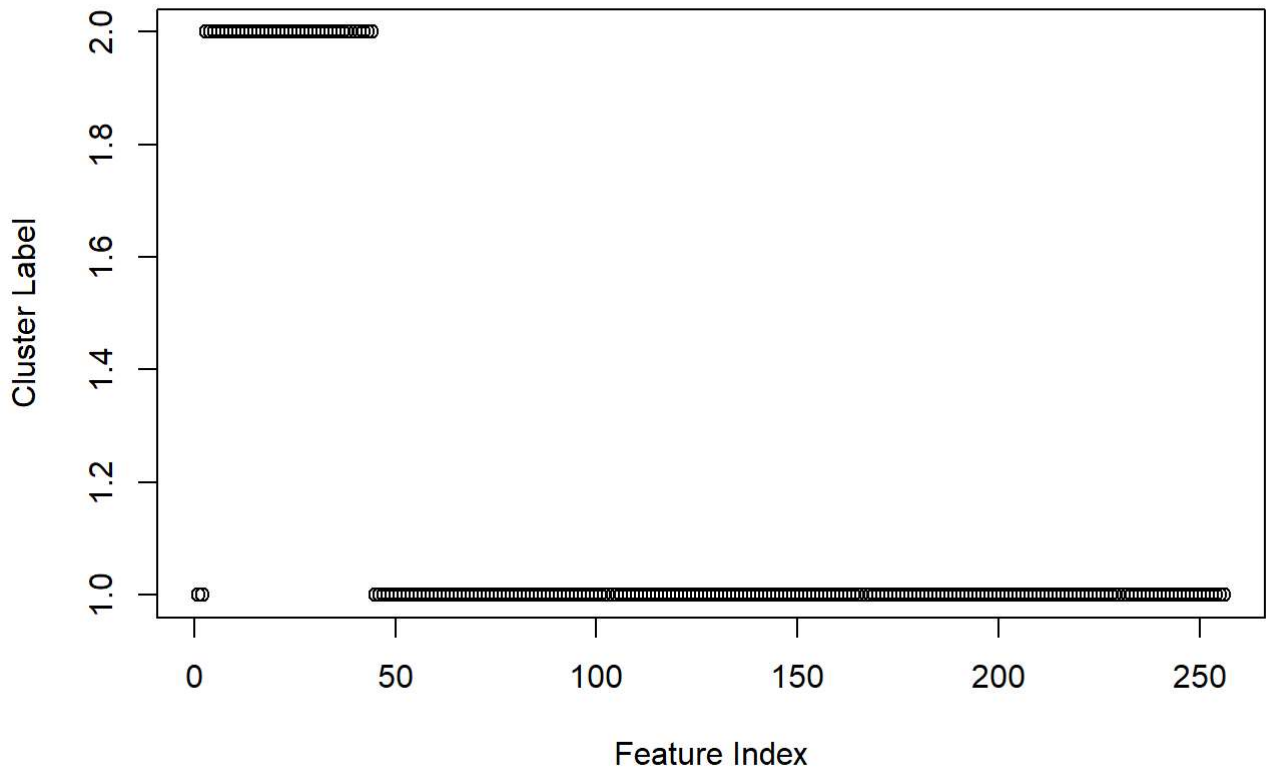
```
DeltaFeatures   = dist(t(X))
out.average = hclust(DeltaFeatures, method = 'average') #get the hierarchical clustering solutio
n
plot(out.average) #Plot the dendrogram
rect.hclust(out.average, k=2)  # highlight the two clusters via rect.clust,
```




```
clust1 = cutree(out.average, k=2) #get the two cluster solution
```

Now, we can plot the cluster solution in terms of the indices of the features (reminder, the cluster label is arbitrary)

```
plot(   as.numeric(substr(names(X),3,5)),unname(clust1), xlab = "Feature Index", ylab = "Cluster Label") # Make a scatter plot with the feature index on the horizontal axis and the cluster label on the vertical axis
```



Which frequencies appear to be related to each other via this clustering solution?

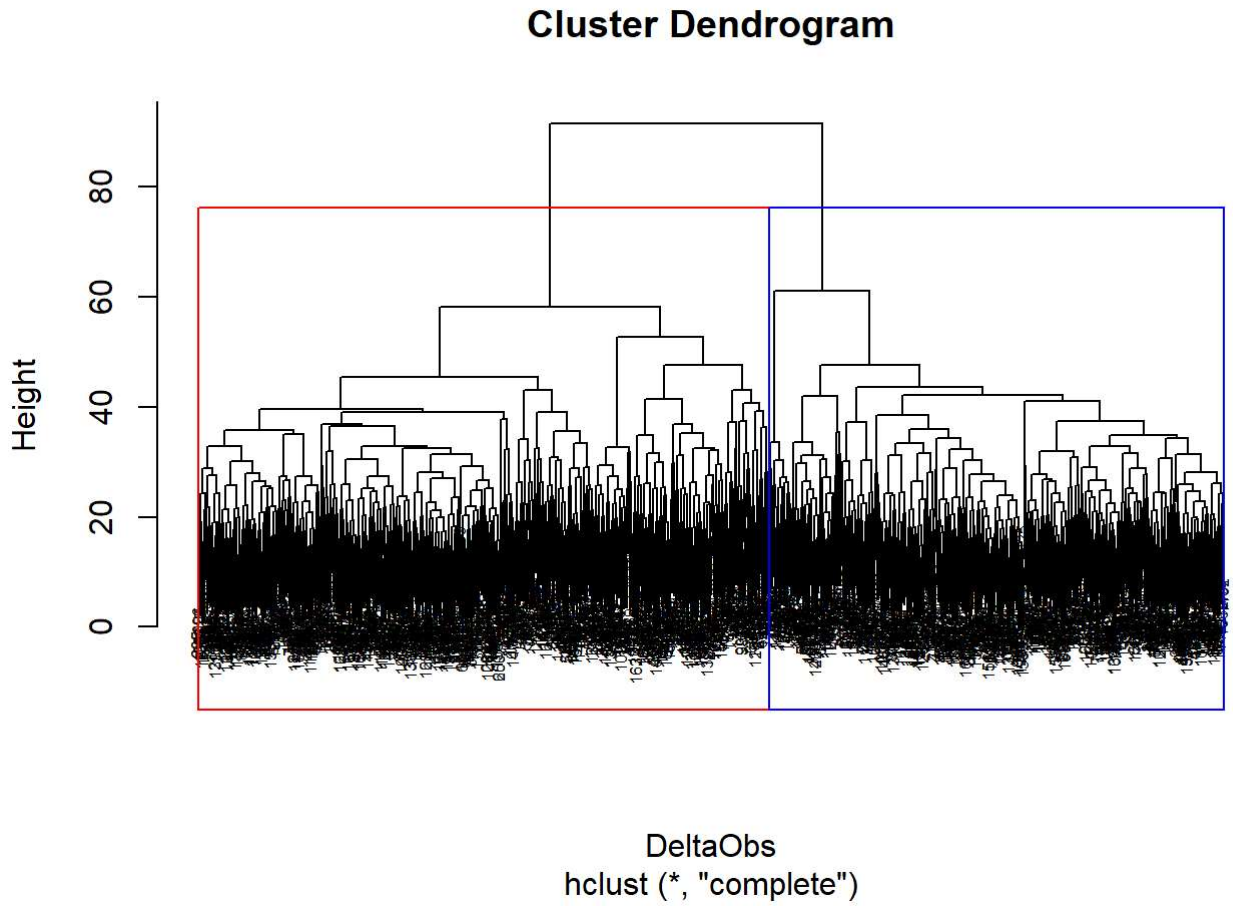
Generally, the frequencies below about 40 are related to each other. Yes, those between say 3 and 40 are a different cluster.

Clustering observations (that is, audio recordings)

For clustering the observations, we will use the information from the previous investigation. We want to make sure we are using at least up to the 40th frequency. Just to make sure, we will use up to the first 75 frequencies. Operationally, this means we will use a modified Euclidean distance: the Euclidean distance only using the frequencies with indices between 1 and 75.

Plot the dendrogram and identify the two clusters with *rect.hclust*.

```
DeltaObs   = dist(X[,1:75])
hclustObs  = hclust(DeltaObs, method = "complete")
plot(hclustObs, cex = 0.5)
rect.hclust(hclustObs, k=2, border=c('red','blue'))
```



Compare this two cluster solution to the known labels via a confusion matrix. Note that the cluster labels are arbitrary; you'll need to decide which cluster best corresponds to each label.

```
hclustObs2 = cutree(hclustObs, k=2)
hclustObs2[hclustObs2 == 2] = "ao"
hclustObs2[hclustObs2 == 1] = "aa"
table(hclustObs2,Y)
```

```
##           Y
## hclustObs2  aa  ao
##         aa 585 371
##         ao 110 651
```

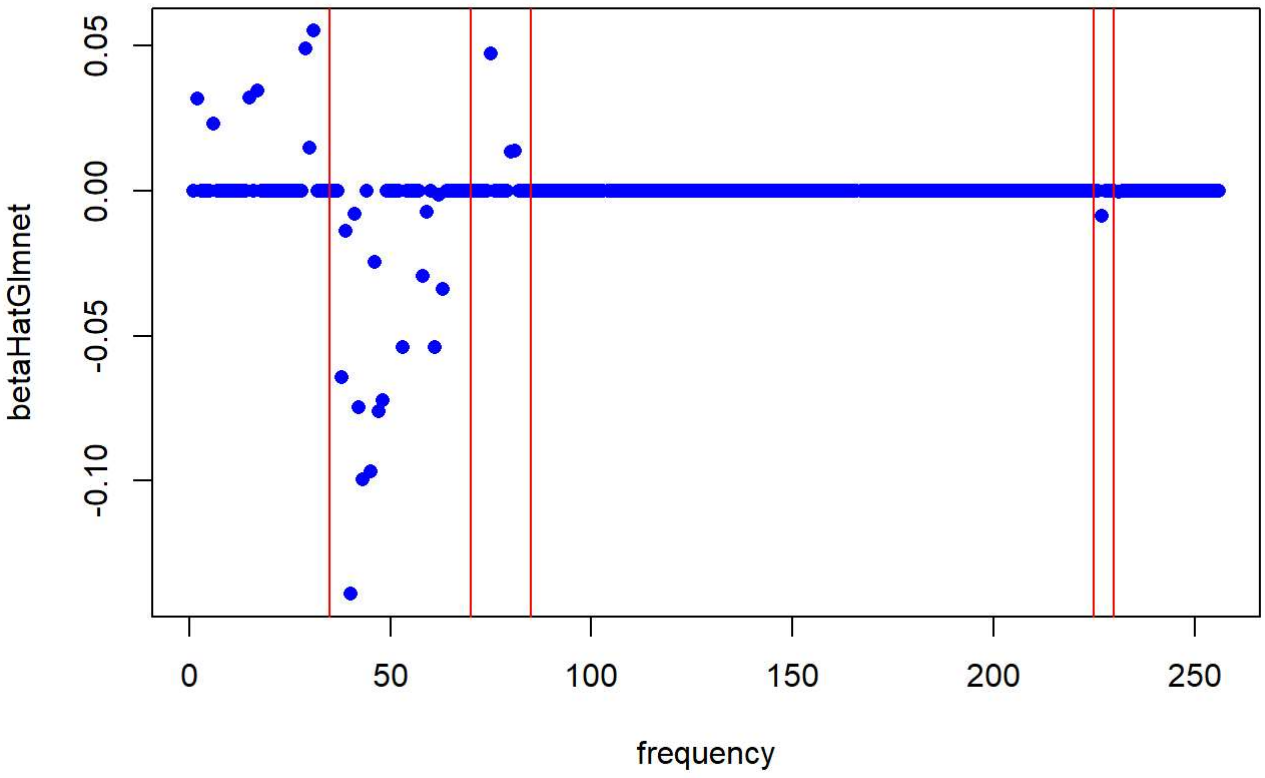
The features of which phoneme seem to be more difficult to cluster? AO seems to be more difficult to cluster. We get worse matching to true values when looking at AO as only approximately 64% of the ao values were matched correctly (compared to 84% of aa correctly matched).

Problem 2 Logistic elastic net

Now, let's apply the logistic elastic net to the problem. In particular, let's estimate which frequencies are associated with each phoneme.

Make a plot of the estimated coefficients selected by the lasso at lambda.min.

```
set.seed(1)
glmnetOut      = cv.glmnet(Xmat,Y, family = 'binomial', alpha = 1)
betaHatGlmnet  = as.numeric(coef(glmnetOut)[-1], s = "lambda.min")
plot(betaHatGlmnet, xlab='frequency', pch = 16, col = 'blue')
abline(v = 35, col = 'red')
abline(v = 70, col = 'red')
abline(v = 85, col = 'red')
abline(v = 225, col = 'red')
abline(v = 230, col = 'red') #Add 4 additional red, vertical lines.  See the next two paragraphs for details
```



Describe in general terms which frequency ranges seem to associated with 'aa' and which are associated with 'ao' (if there are some coefficients of the opposite sign, that is ok. Just look for ranges in which nearly all the estimated coefficients have the same sign as in the example range 1 to 35). Put a vertical red line in the above plot at the end points of each frequency range.

We have evidence that frequencies 0 to 35 and also 70 to 85 are associated with the phoneme ao. We have evidence that frequencies from 35 to 70 and around 225 to 230 are associated withe phoneme aa.

Problem 3 Nonparametric regression

To get use started, let's just look at getting the nonparametric regression fits of the first phoneme plot we made. This should look like 'smoothed' versions of the raw data. To do this, we need to:

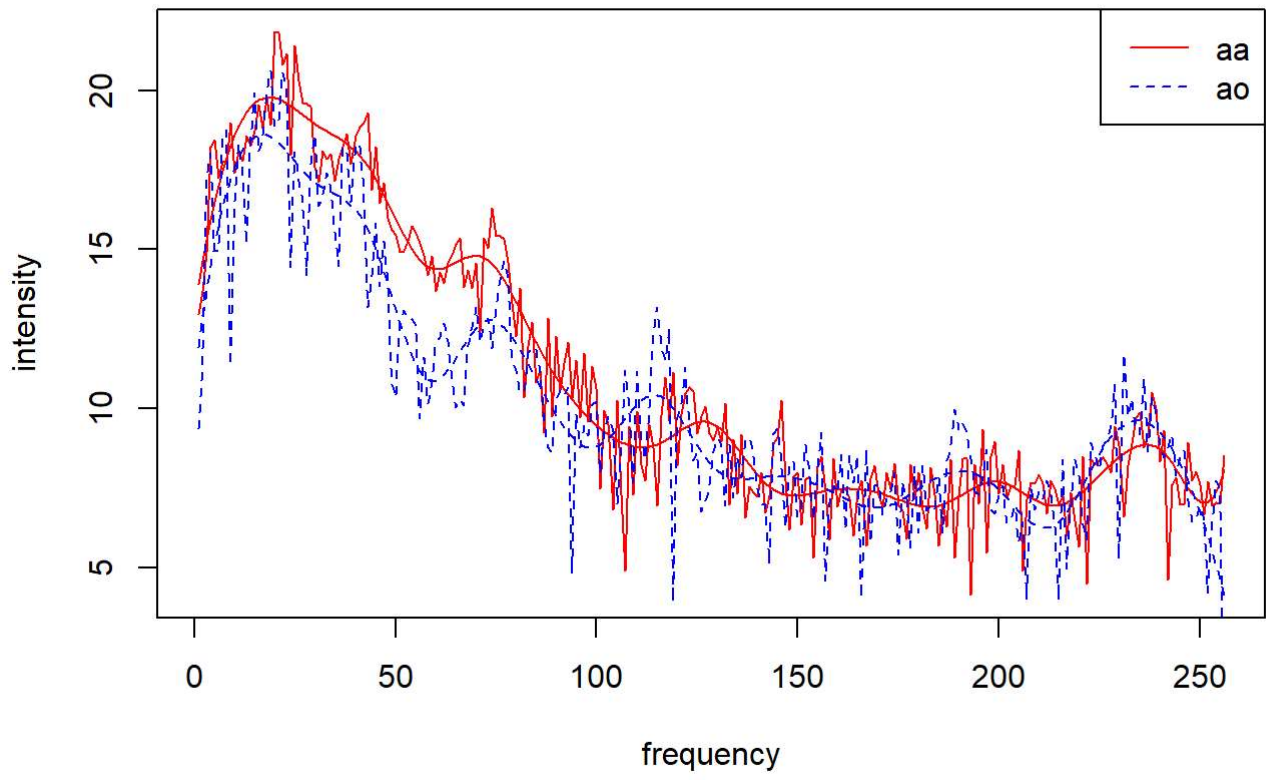
- Get the cubic spline basis evaluated at the frequencies 1, 2, …, 256 (using *bs* to produce Phi). We will just pick a number of basis elements to use here (choose df = 20). In the next problem, we will choose this via a risk estimate (cross-validated accuracy).

```
Phi       = bs(1:256, df = 20)

phonemeExample1smooth = predict(lm(phonemeExample1~., data = Phi))

phonemeExample2smooth = predict(lm(phonemeExample2~., data = Phi))

plot(1:256, phonemeExample1, col = 'red', type= 'l',
     xlab = 'frequency', ylab = 'intensity')
lines(1:256, phonemeExample1smooth, col = 'red')
lines(1:256, phonemeExample2, col = 'blue', lty = 2)
lines(1:256, phonemeExample2smooth, col = 'blue', lty = 2)
legend('topright', legend = c('aa', 'ao'), col = c('red','blue'), lty=c(1,2))
```



Problem 4 Nonparametric classification with splines

Now, we will turn to the classification problem. To apply nonparametric classification, we will do the following steps

- Get the cubic spline basis evaluated at the frequencies 1, 2, …, 256 (using *bs* to produce Phi)
- Get the ‘smoothed’ versions of each observation 1, 2, …, 1717 (making the product X Phi)

- Get the logistic regression fit on the smoothed versions (using X Phi as the feature matrix)

(note: this last step particularly exemplifies the fact that nonparametric methods work by creating feature transformations)

Problem 4.1

We will additionally do this from within a CV loop in order to estimate the test accuracy over a grid (note: we want to maximize the CV accuracy).

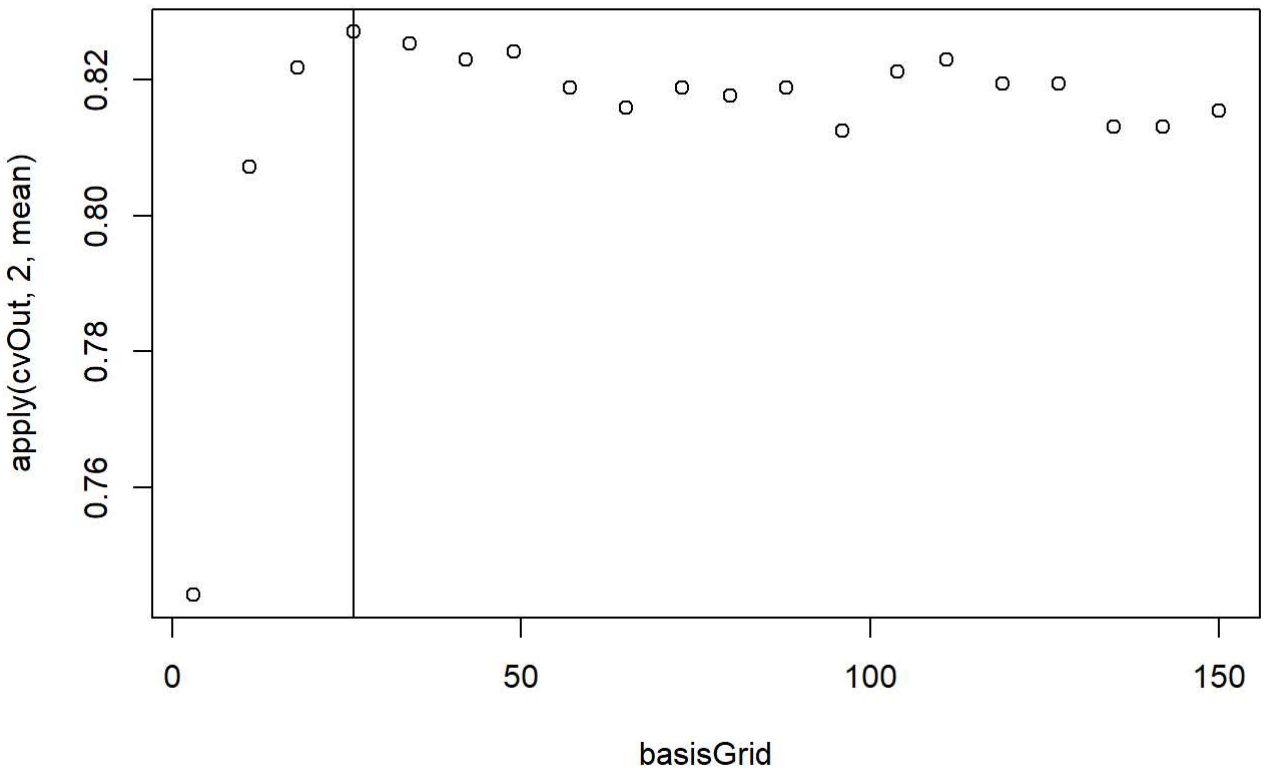
```
set.seed(1)
K         = 10
cvFolds   = createFolds(Y, k = K)
basisGrid = round(seq(3,150, length.out = 20))
cvOut     = matrix(0, nrow = K, ncol = length(basisGrid))

iterBasis = 0
for(b in basisGrid){
  iterBasis = iterBasis + 1
  iterFold  = 0
  Phi       = bs(1:256, df = b)
  smoothX   = as.data.frame(Xmat%*%Phi)
  for(fold in cvFolds){
    iterFold   = iterFold + 1

    glmOut     = glm(Y~., data = smoothX, subset = -fold, family = "binomial")
    logOddsHat = predict(glmOut, smoothX[fold,])
    Yhat       = ifelse(logOddsHat > 0, "ao", "aa")
    cvOut[iterFold, iterBasis] = mean(Yhat == Y[fold])
  }
}

bHat = basisGrid[which.max(apply(cvOut, 2, mean))]

plot(basisGrid, apply(cvOut, 2, mean))
abline(v = bHat)
```



The bHat I found is 26

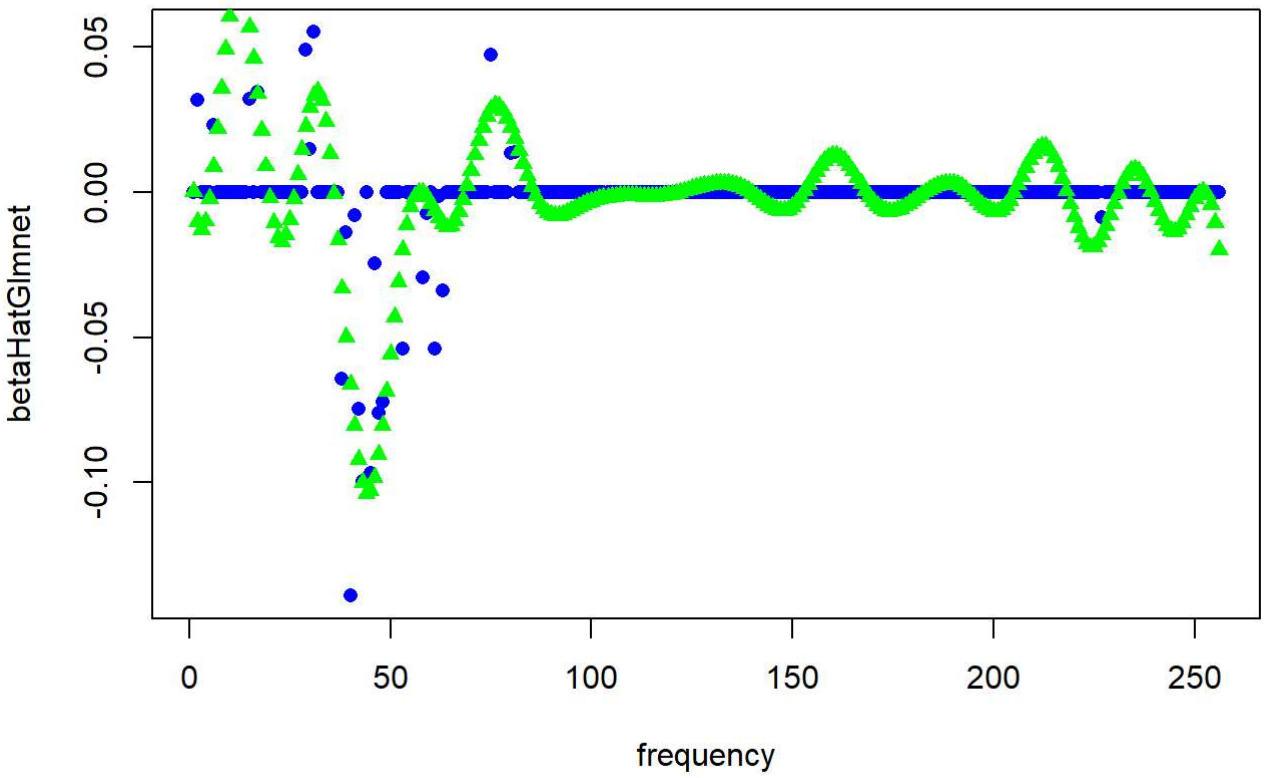
Problem 4.2

Now, let's get the nonparametric logistic regression fit with bHat basis functions (which is the same as saying bHat feature transformations) using all the data

```
Phi        = bs(1:256, df = bHat)
smoothXmat = as.data.frame(Xmat%*%Phi)
smoothX    = as.data.frame(smoothXmat)
glmOut     = glm(Y~., data = smoothX, family = "binomial")
betaHat    = coef(glmOut)[-1]
```

We can directly compare the coefficients we found via the nonparametric method to the ones found via logistic lasso:

```
plot(betaHatGlmnet, xlab='frequency', pch = 16, col = 'blue')
points(Phi %*% betaHat, pch = 17, col = 'green')
```


betaHatGlmnet
0.05
0.00
-0.05
-0.10
0
50
100
150
200
250
frequency