Homework 7 Help: Vowel Formant Space

Lew Harvey

14 March 2016

Part 1

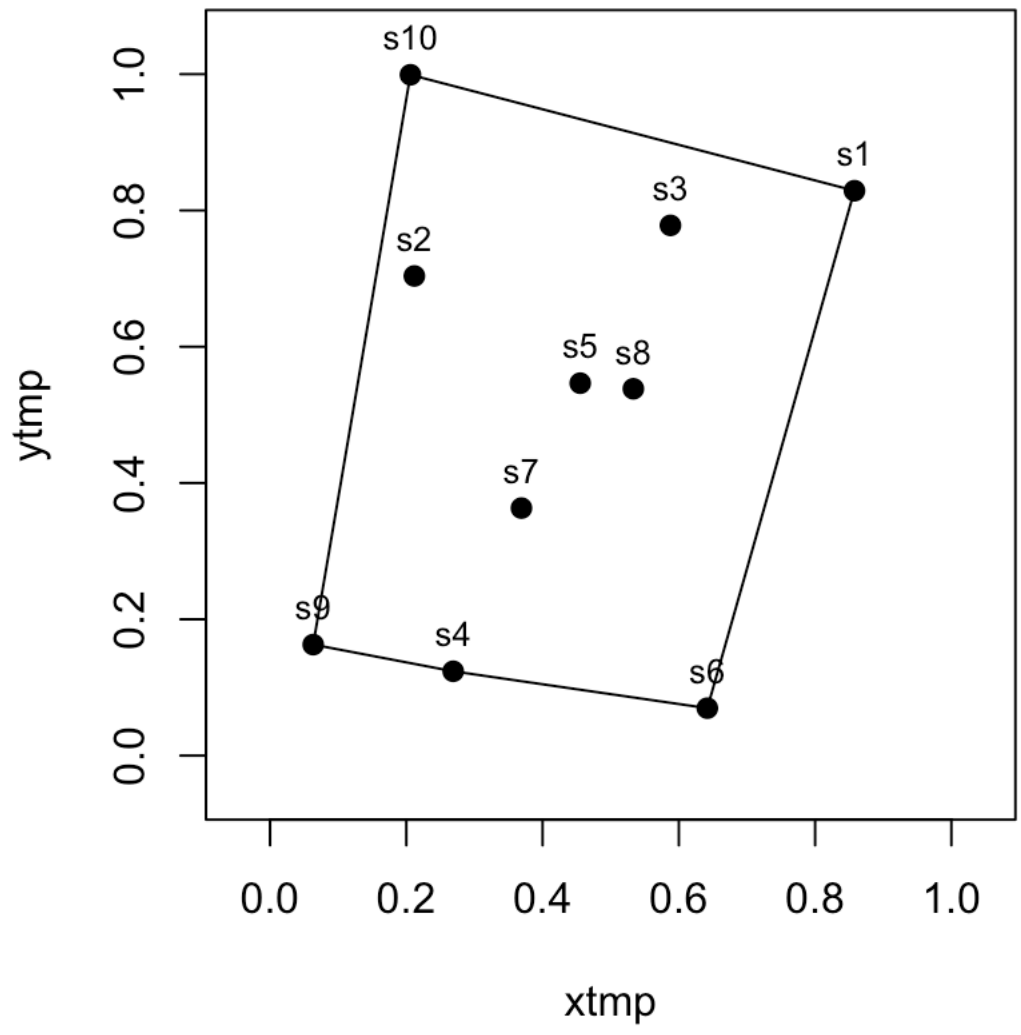
One way to organize the formant data given in the “Average Vowel Formants” table of the homework assignment is to create three vectors containing the character symbol, the frequency of the first formant, and the frequency of the second formant:

1. `vo <- c("i", "y", "e", "ø", "ɛ", "œ", "a", "æ", "a*", "b", "^", "ɔ", "ɤ", "o", "ɯ", "u" )`
2. `f1 <- c(240, the rest of the first formant frequencies go here)`
3. `f2 <- c(2400, the rest of the second formant frequencies go here)`

The symbols given in `vo` above are the correct IPA symbols. You should be able to copy and paste that line. Or you can go to the Wikipedia web page from which the data were obtained: `http://en.wikipedia.org/wiki/Formant` and copy them from there.

The `chull()` function returns the indices of the points of the convex enclosing polygon. These indices may be used to plot the polygon. Here is an example of how to use `chull()` and how to label each point. Note that the x- and y-axes are made to run from 0 to 1 because the `runif()` returns a random number sampled from a uniform distribution of numbers ranging from 0 to 1. Actually I made the limits run from -0.05 to 1.05 to give a little more space for the labels. Since this graph has identical ordinate and abscissa axes that are the same size, it makes sense to make the plot square (`par(pty = "s")`). Note that I have embedded the plot commands inside a function called `plot0()` so it can be evoked again later on.

```
# generate 10 random x and y points and plot them
xtmp <- runif(10)
ytmp <- runif(10)
# make a vector of 10 character strings
lab0 <- c("s1", "s2", "s3", "s4", "s5", "s6", "s7", "s8", "s9", "s10")
# define a function to make the plot for convience
plot0 <- function() {
    op <- par(pty = "s") # make plot square
    plot(xtmp, ytmp, pch = 19, xlim = c(-0.05, 1.05), ylim = c(-0.05, 1.05))
    # find the points that form the convex hull
    i <- chull(xtmp, ytmp)
    # draw the convex hull in the plot
    polygon(xtmp[i], ytmp[i])
    # add the labels
    text(xtmp, ytmp, lab0, pos = 3, cex = 0.8)
    par(op)
}
plot0()
```



Write your own code to generate the vowel plot using the ideas above. Set the limits of the x-axis to be 200 to 900 and the limits of the y-axis to be from 500 to 2600. First create the graph, then add the labels, and finally draw the convex hull using the `polygon()` function.

```
# put your code here
```

Part 2

Computing the distance among all possible pairs of points can be extremely tedious if done pairwise by hand, but R makes it easy with the `dist()` function. It is easiest if you put the coordinates of the points into a data frame. Here is an example using the random numbers previously generated above and that were stored in vectors `xtmp` and `ytmp` and using the arbtrary labels stored in `lab0` :

```
# assemble the coordinates into a data frame
df0 <- data.frame(x = xtmp, y = ytmp)
# name the rows with the symbols
row.names(df0) <- lab0
# compute the distances among all pairs
d0 <- dist(df0)
print(round(d0, 2))
```

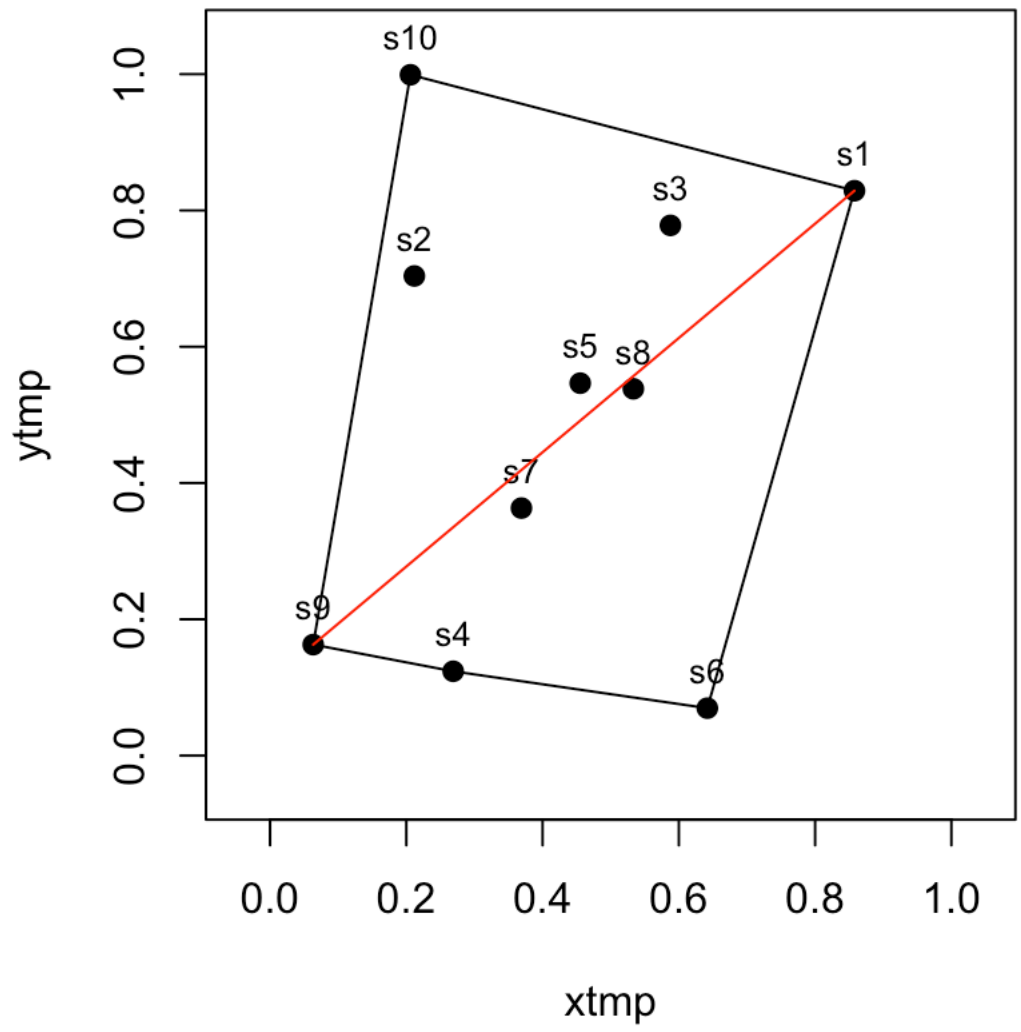
```
##       s1   s2   s3   s4   s5   s6   s7   s8   s9
## s2  0.66
## s3  0.27 0.38
## s4  0.92 0.58 0.73
## s5  0.49 0.29 0.27 0.46
## s6  0.79 0.77 0.71 0.38 0.51
## s7  0.68 0.38 0.47 0.26 0.20 0.40
## s8  0.44 0.36 0.25 0.49 0.08 0.48 0.24
## s9  1.04 0.56 0.81 0.21 0.55 0.59 0.37 0.60
## s10 0.67 0.30 0.44 0.88 0.52 1.03 0.66 0.57 0.85
```

You can find the largest distance by visual inspection but of course R has an easier way using the `which()` function that returns the row and column index of the cell that meets your specified criterion (i.e., max(d1)). You should convert the distance matrix, which is lower triangular, into a full matrix using the `as.matrix()` function so that you can recover the row and column of the maximum, as shown in the example below. The `which()` function returns the name of the symbols in the row names of the 2x2 matrix holding the indices of the max (there are two because the distance matrix is symmetrical; it does not matter which you use)

```
dd <- which(as.matrix(d0) == max(d0), arr.ind = TRUE)
nn <- row.names(dd)
print(paste("greatest distance, ", round(max(d0), 2), ", is between ", nn[1], " and ", nn[
2], sep = ""))
```

```
## [1] "greatest distance, 1.04, is between s9 and s1"
```

```
# plot the random points again and connect the two points with the greatest distance
plot0()
lines(df0[dd[, 1], "x"], df0[dd[, 1], "y"], col = "red")
```


s10
s1
s3
s2
s5
s8
s7
s9
s4
s6
ytmp
xtmp
0.0
0.2
0.4
0.6
0.8
1.0