OS 4118 **MDS Example** Fall AY 2020

**Introduction and Data Handling:** This question uses the “splice” dataset from the U.C. Irvine Repository; I’ve put it in the folder under the name splice.data.txt. The splice.names file gives more details (Windows users, open in Wordpad, not Notepad). Read the data into R. The predictors are DNA letters A, C, T, or G. You want 60 separate columns of predictors, not one long predictor that’s 60 characters long.

Because this is a **fixed-width** file, you will want to use the read.fwf() function to read the data in. You will have to specify the lengths of the fields. For example, if you knew that your fields had length 3, 11, and 8, you would use read.fwf() with widths = c(3, 11, 8). In this case you will want a few things up at the front, plus 60 fields of one character each. The rep() function may be useful here.

The first column serves as the response; discard the stuff between the response and the first predictor. Once you’ve read the data in, do these things:

(1) Delete any rows with predictors other than A, C, T or G. Here’s one way to do that: start by identifying which rows those are, maybe with a command like

bad <- apply (splice[,-1], 1, function (x)

any (! x %in% c("A", "C", "T", "G")))

This uses apply() to run a function on every row of the data frame (the 1 in the second argument specifies rows rather than columns). The function returns TRUE is a row is “bad,” that is, contains anything other than an A, C, T, or G. In my example I used splice[,-1]to exclude the response column. You will need to do something similar.

The result of this could should be a vector containing all of the rows to be deleted. Use that to remove those bad rows. If you don’t know how to do that, ask me.

(2) Select a random sample of size 1000 (using sample()), and then remove and save the class column.

At the end of these operations you should have a data set that is 1000 x 60, and also a vector of the true classes that has length 1000.

**Computing the Distances:** Use daisy() (from the cluster library) to compute the inter-point distances in the Gower style. In this example the Manhattan distance measure sthe number of sites at which two rows disagree, whereas the Euclidean distance measures the square root of that. Does one make more sense that the other? Now install the treeClust() library. Its treeClust.dist() function acts as a competitor to daisy() in computing the inter-point distances. Compute those distances.

(c) It’s difficult to visualize 60-dimensional space. Use cmdscale() to convert the output of daisy() and, separately, treeClust.dist() into a two-dimensional matrix. (For treeClust.dist(), set d.num = 4). Then draw a pretty picture of those 1,000 points with colors given by the true class membership. Now do the same for those distances with Rtsne() from the Rtsne library. (See the help to figure out the name of the item of the result that contains the new coordinates.) Do the colors appear to be in separate places on the picture?