**Overview of Data Structures in R**

When data mining, it is especially important to understand the different ways we can format our data so that we can further refine it and ultimately come to conclusions. For the purposes of this course, we will be focusing on four data structures in particular. These structures include vectors, data frames, and matrices/arrays. In this exercise, we will overview each one of these structures and the data that can be stored in them.

**Vectors:**

When it comes to the R language, vectors are the most fundamental data type. Vectors, in layman’s terms, are simple data sets that are stored in a variable. Let’s look at the example provided below.

> x <- c(1,2,3,4,5)

In this example, we concatenate the values 1, 2, 3, 4, and 5 into variable x. The data set contains all integer values which is important because any R vector must contain values of the same mode or type. You can determine what type or mode a vector is by performing the typeof() command. In this case…

> typeof(x)

[1] “double”

If elements are of different mode, it can change the structure of the vector and thus change how it can be utilized. For example…

> y <- c(1,2,3,4,5,”Cat”)

>typeof(y)

[1] “character”

As you can see, inserting the value “Cat” into the set of numbers listed above changes the identity of the vector from it containing all double values to now containing “character” values. Since the data set is now identified as having character values, we can no long perform arithmetic functions on it. As you may be able to see, vectors can hold all sorts of values such as, but not limited to, integers, floating-point values, characters, logical functions, etc.

**Matrices and Arrays:**

One way you can look at matrices in R is that they look like multiple vectors piled on top of each other. Another way to look at a matrix in R is to look at it as if you were in math class; they are practically identical as you might think. Unlike vectors, matrices have two extra attributes: the number of rows and columns. Lets take a look at the example below…

> x <- rbind(c(1,2),c(3,4))

> x

[,1] [,2]

[1,] 1 2

[2,] 3 4

You might think that the notation seen in the example above might look similar to that of what we used to create a vector. In reality, as mentioned above, matrices in R are quite literally vectors piled on top of each other. In order to combine the two vectors into a matrix, we use the rbind (row bind) command. Once again, as with vectors, it is important to make sure that all elements of a matrix are of the same mode. We will be using matrices a fair amount in this class (in a more complex form of course).

When it comes to the R language, an array is very similar to a matrix, except it has more than 2 dimensions. In fact, for our purposes, an array has 3 dimensions. They include the number of rows and columns in an array as well as the number of elements contained within the array. Let’s look at the example below…

> x<-array(sample(1:24), dim=c(3,4,3))

> x

, , 1

[,1] [,2] [,3] [,4]

[1,] 22 4 13 8

[2,] 1 3 21 18

[3,] 9 6 5 12

, , 2

[,1] [,2] [,3] [,4]

[1,] 15 2 24 11

[2,] 17 7 20 10

[3,] 16 14 19 23

, , 3

[,1] [,2] [,3] [,4]

[1,] 22 4 13 8

[2,] 1 3 21 18

[3,] 9 6 5 12

In the example above, we have effectively created three 3x4 arrays and have filled them with random numbers between 1 and 24. Another way to interpret this would be to say that this particular array contains three elements (matrices), with each element having a 3x4 structure. As with matrices, arrays must contain data of the same mode.

**Data Frames:**

In essence, data frames are essentially like matrices with one key exception: the elements of a data frame may be different compared to one another. Essentially, this means a data frame can contain different modes. Besides this exception, data frames and matrices look virtually identical. Let’s take a look at an example…

> jobs <- c(“Manager”, “Cashier”, “Bagger”)

> pay <- c(25.00, 10.00, 8.00)

> x <- data.frame(jobs,pay)

> x

jobs pay

1 Manager 25

2 Cashier 10

3 Bagger 8

Once again, the notation used to create the data frame above is the same notation used to create vectors. To create the data frame, we simply use the data.frame command and specify the vectors that will make up the contents of the frame. It should be noted that data frames can be treated as matrices (to an extent), therefore you can use some commands that you would use with matrices (for example, str()).