**Week 2 – Data Structures**

This week we will focus on the following:

1. Vectors
2. Matrices
3. Lists
4. DataFrames

**Vectors**

Vectors are the simplest data structures in R. They are sequences of elements of the same basic type. These types can be numeric, integer, complex, character, and logical. In R, the more complicated data structures are made with vectors as building-blocks. Vectors contain what are called indices which can be thought of as the position in which a value is located within the vector.

So how do you define a vector in R?   
1) Use c():

EX: number\_vector <- c(1,2,3,5,8,13,21)

2) Using an operator:

EX: number\_vector <- 1:10

3) Taking a slice from a data frame or matrix:

EX1: price\_vector <- boston\_housing\_df[,boston\_housing\_df$price]

EX2: number\_vector <- number\_mat[,1]

More on this later….

**Vector Types & Operations**

Vectors can contain any type of data in R. This means that there are special operations you can perform on the different types. Here are some that will be important for you to learn.

1. Slicing: Slicing is a way for you to subset your vectors to extract elements out of it. There are a few ways to do it in R:
   1. Vectors are ordered by their **indices**. Since R is a 1 based indexed language the first element in a vector will be 1. You can extract specific indices using the bracket notation:

> fib\_vec[1] #will slice the first element

[1] 1

> fib\_vec[1:10] #will slice the first 10 elements

[1] 1 1 2 3 5 8 13 21 34 55

> fib\_vec[c(1,5,10)] #will slice the 1st, 5th, and 10th elements

[1] 1 5 55

1. Sometimes you will want to know the index position a specific number is located at. Or you may want to know where the specific min or max is located. To do this we will use the function **which().**

> which(fib\_vec == 8)

[1] 6

> which.max(fib\_vec)

[1] 10

> which.min(fib\_vec)

[1] 1

1. To remove an element from a vector simply take the inverse of the indices.

> fib\_vec[-8] #remove the 8th element

[1] 1 1 2 3 5 8 13 34 55

> fib\_vec[-1:-10] #remove the first through 10th element

numeric(0)

> fib\_vec[c(-1,-5,-10)] #remove the first, 5th, and 10th element

[1] 1 2 3 8 13 21 34

1. You can also add, subtract, multiply, or divide vectors. Its important to note when you do this each element is added/subtracted/multiplied/divided by the corresponding element in the second vector. If the second vector is of unequal length, the smaller vector will be recycled to compensate for the different length.

> fib\_vec + catalan\_vec#vector addition

[1] 2 3 7 17 47 140 442 1451 4896 16851

> catalan\_vec - fib\_vec #vector subtraction

[1] 0 1 3 11 37 124 416 1409 4828 16741

> catalan\_vec \* fib\_vec#vector multiplication

[1] 1 2 10 42 210 1056 5577 30030 165308 923780

> catalan\_vec/fib\_vec#vector division

[1] 1.000000 2.000000 2.500000 4.666667 8.400000 16.500000 33.000000

[8] 68.095238 143.000000 305.381818

1. Other useful functions with vectors:
   1. any()
   2. all()
   3. sort()
   4. length()
   5. rep()
   6. seq()

> any(catalan\_vec==1) #any of the values equals 1

[1] TRUE

> all(catalan\_vec==1) #all of the values equals 1

[1] FALSE

> sort(catalan\_vec,decreasing=TRUE)#vector sorting

[1] 16796 4862 1430 429 132 42 14 5 2 1

> length(catalan\_vec) #find the length of the vector

[1] 10

> rep(catalan\_vec,2) #repeat the vector 2 times

[1] 1 2 5 14 42 132 429 1430 4862 16796 1 2 5

[14] 14 42 132 429 1430 4862 16796

> seq(from=1,to=10,by=2) #create a vector from 1 to 10 by 2

[1] 1 3 5 7 9

**Matrices**

Matrices are the R objects in which the elements are arranged in a two-dimensional rectangular layout. They contain elements of the same atomic types. Though we can create a matrix containing only characters or only logical values, they are not of much use. We use matrices containing numeric elements to be used in mathematical calculations.

A lot of times we will be using high level libraries to model our data and will not be doing matrix calculations directly. However, it is still good to understand how to access matrix components and perform calculations against them. Much like vectors matrix elements can be accessed using the bracket notation with the difference being that they are multi-dimensional. The first element within the brackets is the row and the second is the column. So to access elements you can apply the same logic you did for vectors just by slicing either the rows or columns. Full examples can be found in the Ex2\_matrices.R file in /Examples.

**Data Frames**

Data frames are the most widely used data structure in data science. They can contain multiple columns of differing data types and can be thought of as a collection of rows and columns much like what you would find in an Excel workbook. The reason we covered vectors first is because each data frame column is simply a vector. Further, elements of a data frame can be accessed just as they were with matrices. There are rules that data frame objects must follow:

* The column names should be non-empty.
* The row names should be unique.
* The data stored in a data frame can be of numeric, factor or character type.
* Each column should contain same number of data items.

There are several ways in which you can create data frames, either by using the R function data.frame() directly, or by importing data into R from an outside source.

1. CSV files

> sales\_csv <- read.csv("Week\_2/Data/sales.csv"

+ ,stringsAsFactors=FALSE

+ )

1. Tab or any other delimiter

> sales\_tab\_delim <- read.delim("Week\_2/Data/sales.txt"

+ ,stringsAsFactors=FALSE

+ ,sep = "\t"

+ )

1. Excel files

> sales\_excel <- readxl::read\_excel("Week\_2/Data/sales.xlsx"

+ ,sheet = "sales"

+ )

**Data Frame Data Types**

Data frames typically contain varying data types. Its important to understand what each is and how they will be used. By now you should be familiar with integers and numeric vectors so we will skip those data types and focus on dates, characters, factors, and NA values (not an actual data type).

**Characters** (strings)

A character in R is any combination of numbers, letters, or other characters (text) wrapped in quotes. Characters typically hold descriptions, keys, raw dates, names, or any other descriptive data.

**Dates**

When you import data into R dates are usually represented as character strings.