EXPERIMENT 6

Problem Statement: To implement a program using 'apply' function to create and apply normalization function on each of the numeric variables/columns of iris dataset to transform them into

i.) 0 to 1 range with min-max normalization.
ii.) a value around 0 with z-score normalization.

Theoretical Background

apply library in R language

The apply() function is the most basic of all collection as it contains sapply(), lapply() and tapply().The apply in R function can be feed with many functions to perform redundant application on a collection of object (data frame, list, vector, etc.). The purpose of apply() is primarily to avoid explicit uses of loop constructs. They can be used for an input list, matrix or array and apply a function. Any function can be passed into apply().

apply() function:

apply() takes Data frame or matrix as an input and gives output in vector, list or array. Apply function in R is primarily used to avoid explicit uses of loop constructs. It is the most basic of all collections can be used over a matrice.

```
apply(X, MARGIN, FUN)
```

```
-x: an array or matrix
-MARGIN:  take a value or range between 1 and 2 to define where to apply
the function:
-MARGIN=1: the manipulation is performed on rows
-MARGIN=2: the manipulation is performed on columns
-MARGIN=c(1,2)` the manipulation is performed on rows and columns
-FUN: tells which function to apply. Built functions like mean, median,
sum, min, max and even user-defined functions can be applied>
```

sapply() function

The sapply() function helps us in applying functions on a list, vector, or data frame and returns an array or matrix object of the same length. The sapply() function in the R Language takes a list, vector, or data frame as input and gives output in the form of an array or matrix object. Since the sapply() function applies a certain operation to all the elements of the object it doesn't need a MARGIN. It is the same as lapply() with the only difference being the type of return object.
Syntax: sapply(x, fun)
Parameters:
x: determines the input vector or an object.
fun: determines the function that is to be applied to input data.

lapply() function:

lapply() function is useful for performing operations on list objects and returns a list object of same length of original set. lappy() returns a list of the similar length as input list object, each element of which is the result of applying FUN to the corresponding element of list. Lapply in R takes list, vector or data frame as input and gives output in list.

```
lapply(X, FUN)
Arguments:
-X: A vector or an object
-FUN: Function applied to each element of x
```

l in lapply() stands for list. The difference between lapply() and apply() lies between the output return. The output of lapply() is a list. lapply() can be used for other objects like data frames and lists.

Program Code

#Min max normalization

```
head(iris)

#define Min-Max normalization function

min_max_norm<- function(x) {

  (x - min(x)) / (max(x) - min(x))

}

#apply Min-Max normalization to first four columns in iris dataset

iris_norm<- as.data.frame(lapply(iris[1:4], min_max_norm))

#view first six rows of normalized iris dataset

head(iris_norm)

#add back Species column

iris_norm$Species<- iris$Species

#view first six rows of iris_norm

head(iris_norm)
```

#Z-Score Normalization

```
#standardize first four columns of iris dataset
iris_standardize<- as.data.frame(lapply(iris[1:4],scale))
#view first six rows of standardized dataset
head(iris_standardize)
#add back Species column
iris_standardize$Species<- iris$Species
#view first six rows of iris_standardize
head(iris_standardize)
```

Program Output:

#Min max normalization output:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
1	0.22222222	0.6250000	0.06779661	0.04166667
2	0.16666667	0.4166667	0.06779661	0.04166667
3	0.11111111	0.5000000	0.05084746	0.04166667
4	0.08333333	0.4583333	0.08474576	0.04166667

5	0.19444444	0.6666667	0.06779661	0.04166667
6	0.30555556	0.7916667	0.11864407	0.12500000

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	0.22222222	0.6250000	0.06779661	0.04166667	setosa
2	0.16666667	0.4166667	0.06779661	0.04166667	setosa
3	0.11111111	0.5000000	0.05084746	0.04166667	setosa
4	0.08333333	0.4583333	0.08474576	0.04166667	setosa
5	0.19444444	0.6666667	0.06779661	0.04166667	setosa
6	0.30555556	0.7916667	0.11864407	0.12500000	setosa

#Z-Score Normalization output:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
1	0.22222222	0.6250000	0.06779661	0.04166667
2	0.16666667	0.4166667	0.06779661	0.04166667
3	0.11111111	0.5000000	0.05084746	0.04166667
4	0.08333333	0.4583333	0.08474576	0.04166667
5	0.19444444	0.6666667	0.06779661	0.04166667
6	0.30555556	0.7916667	0.11864407	0.12500000

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	0.22222222	0.6250000	0.06779661	0.04166667	setosa
2	0.16666667	0.4166667	0.06779661	0.04166667	setosa

3	0.11111111	0.5000000	0.05084746	0.04166667	setosa
4	0.08333333	0.4583333	0.08474576	0.04166667	setosa
5	0.19444444	0.6666667	0.06779661	0.04166667	setosa
6	0.30555556	0.7916667	0.11864407	0.12500000	setosa

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
1	-0.8976739	1.01560199	-1.335752	-1.311052
2	-1.1392005	-0.13153881	-1.335752	-1.311052
3	-1.3807271	0.32731751	-1.392399	-1.311052
4	-1.5014904	0.09788935	-1.279104	-1.311052
5	-1.0184372	1.24503015	-1.335752	-1.311052
6	-0.5353840	1.93331463	-1.165809	-1.048667

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	-0.8976739	1.01560199	-1.335752	-1.311052	setosa
2	-1.1392005	-0.13153881	-1.335752	-1.311052	setosa
3	-1.3807271	0.32731751	-1.392399	-1.311052	setosa
4	-1.5014904	0.09788935	-1.279104	-1.311052	setosa
5	-1.0184372	1.24503015	-1.335752	-1.311052	setosa
6	-0.5353840	1.93331463	-1.165809	-1.048667	setosa