CM 2062 - Statistical Computing with R
Lab Sheet 2

List

Lists are the R objects which contain elements of different types like - numbers, strings, vectors and another list inside it. List is created using **list()** function.

```
> list_data <- list("Red", c(21,32,11), TRUE, 51.23)
> list_data
[[1]]
[1] "Red"

[[2]]
[1] 21 32 11

[[3]]
[1] TRUE

[[4]]
[1] 51.23
```

Give names to the elements in the list

```
> names(list_data) <- c("Colour", "Vector", "Logical", "Number")
> list_data
$Colour
[1] "Red"

$Vector
[1] 21 32 11

$Logical
[1] TRUE

$Number
[1] 51.23
```

Accessing the elements in the list

```
> list_data[1]
$Colour
[1] "Red"

> list_data[2]
$Vector
[1] 21 32 11

#The $ operator returns a named element of a list.
> list_data$Vector
[1] 21 32 11
```

Add element at the end of the list.

```
> list_data[5] <- "New element"
> list_data
```

Remove the last element.

```
> list_data[4] <- NULL
> list_data
```

Merging Lists

```
> merged.list <- c(list1,list2)
> merged.list
```

Convert the lists to vectors

```
> list1 <- list(1:5)
> list2 <-list(10:14)
> v1 <- unlist(list1)
> v2 <- unlist(list2)
> result <- v1+v2
> result
[1] 11 13 15 17 19
```

Arrays

An array can be considered as a multiply subscripted collection of data entries, for example numeric. R allows simple facilities for creating and handling arrays. You can use array() function to create an array.

```
> a1 <- array(1:5,dim = c(1,5))
> a1
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    2    3    4    5

> a2 <- array(1:3,dim = c(3,1))
> a2
     [,1]
[1,]    1
[2,]    2
[3,]    3

> m1 <- array(1:15,dim = c(3,5))
> m1
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10   13
[2,]    2    5    8   11   14
[3,]    3    6    9   12   15

# Multidimensional Array

Example 1
> arr <- array(2:13, dim = c(2, 3, 2))
> arr
, , 1

     [,1] [,2] [,3]
[1,]    2    4    6
[2,]    3    5    7

, , 2

     [,1] [,2] [,3]
[1,]    8   10   12
[2,]    9   11   13

Example 2
array1 <-  c (5, 10, 15, 20)
array2 <-  c (25, 30, 35, 40, 45, 50, 55, 60)

final <- array (c (array1, array2),dim =c(4,4,3))

> final <- array (c (array1, array2),dim =c(4,4,3))
> final
```

```
, , 1

     [,1] [,2] [,3] [,4]
[1,]    5   25   45    5
[2,]   10   30   50   10
[3,]   15   35   55   15
[4,]   20   40   60   20

, , 2

     [,1] [,2] [,3] [,4]
[1,]   25   45    5   25
[2,]   30   50   10   30
[3,]   35   55   15   35
[4,]   40   60   20   40



, , 3

     [,1] [,2] [,3] [,4]
[1,]   45    5   25   45
[2,]   50   10   30   50
[3,]   55   15   35   55
[4,]   60   20   40   60


> final <- array (c (array1, array2),dim =c(4,3,3))
> final
, , 1

     [,1] [,2] [,3]
[1,]    5   25   45
[2,]   10   30   50
[3,]   15   35   55
[4,]   20   40   60

, , 2

     [,1] [,2] [,3]
[1,]    5   25   45
[2,]   10   30   50
[3,]   15   35   55
[4,]   20   40   60
```

```
, , 3

     [,1] [,2] [,3]
[1,]    5   25   45
[2,]   10   30   50
[3,]   15   35   55
[4,]   20   40   60

> final <- array (c (array1, array2),dim =c(3,3,3))
> final
, , 1

     [,1] [,2] [,3]
[1,]    5   20   35
[2,]   10   25   40
[3,]   15   30   45

, , 2

     [,1] [,2] [,3]
[1,]   50    5   20
[2,]   55   10   25
[3,]   60   15   30

, , 3

     [,1] [,2] [,3]
[1,]   35   50    5
[2,]   40   55   10
[3,]   45   60   15
```

Let's rename our array to "Arr1" and "Arr2" by using "matrix.names". Also,the rows name changed to ("row1","row2") and column names will be changed to ("column1","column2","column3") respectively. The dimension of the matrix is 2 rows and 3 columns.

```
> array1 <-  c (9 , 18 )
>  array2 <-  c (27, 36)
>  r.names = c ("row1","row2")
>  c.names = c ("column1","column2","column3")
>  m.names = c ("Arr1", "Arr2")
>  final <- array (c (array1,array2), dim=c (2,3,2),
                dimnames=list (r.names, c.names, m.names))
> final
```

```
, , Arr1

     column1 column2 column3
row1       9      27       9
row2      18      36      18

, , Arr2

     column1 column2 column3
row1      27       9      27
row2      36      18      36
```

Exercise
Create an array with the name "arr" and dim=c (3,3,1), with the below two vectors.
a1= c (1,2,3,4)
a2= c (5,6,7,8,9)
Rename the column names as (c1,c2,c3) and row names as (r1,r2,r3) and the matrix name as "first".

Indexing in an array

In the previous Exercise array is,

```
, , first

   c1 c2 c3
r1  1  4  7
r2  2  5  8
r3  3  6  9
```

Let's see how the elements in the array can be extracted with the following example

1. Let's extract number '7' from the above array 'arr'.

```
> arr[1,3,1]
[1] 7
```

2. To access multiple values at once, you need to specify the range you want.

```
> arr[1:2,1:2,1]
   c1 c2
r1  1  4
r2  2  5
```

3. You can access the entire array 'arr' with the following syntax where 'arr[, ,1]' specifies to include all rows and columns each separated by commas, which are indicated by space. The 1 specifies the array 'arr' to be extracted.

```
> arr[ , ,1]
   c1 c2 c3
r1  1  4  7
r2  2  5  8
r3  3  6  9
```

4. You can get the entire second row by following code where arr[2, ,1] gets the second row with space, and 1 is the 'arr' to be extracted.

```
> arr[2, ,1]
c1 c2 c3
 2  5  8
```

5. You can get the entire second column by following code where arr[,2,1] space with 2 is the second column, and 1 is the 'arr' to be extracted.

```
> arr[,2,1]
r1 r2 r3
 4  5  6
```

5. You can change the value of an element in the array.

```
> arr[1,3,1] <- 5
> arr
, , first

   c1 c2 c3
r1  1  4  5
r2  2  5  8
r3  3  6  9
```

Matrices

A matrix is a two dimensional array. A matrix can be generated by the array(...) function with the dim argument set to be a vector of length 2. Alternatively, we can use the matrix(...) command or provide a dim attribute for a vector.

```
> m1 <- array(1:15,dim = c(3,5))
> m1
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    4    7   10   13
[2,]    2    5    8   11   14
[3,]    3    6    9   12   15

> m2 <- matrix(1:15,3,5)
> m2

> m2 <- matrix(1:15,nrow=3,ncol=3)
> m2
> m3 <- 1:15
> dim(m3) <- c(3,5)
> m3
```

By default the matrix function reorders a vector into columns, but we can also tell R to use rows instead.

```
m3 <- matrix(1:15, nrow=3, ncol=3, byrow = TRUE)
m3
```

Define the column and row names

```
> rownames = c("row1", "row2", "row3")
> colnames = c("col1", "col2", "col3")
> m3 <- matrix(1:15,nrow=3,ncol=3, dimnames = list(rownames, colnames))
> m3
     col1 col2 col3
row1    1    4    7
row2    2    5    8
row3    3    6    9
```

If their sizes match, vectors can be combined to form matrices, and matrices can be combined with vectors or matrices to form other matrices. The functions that do this are cbind and rbind.

cbind binds together columns of two objects. One thing it can do is put vectors together to form a matrix:

```
> m4 <- cbind(1:3,4:6,5:7)
> m4
     [,1] [,2] [,3]
[1,]    1    4    5
[2,]    2    5    6
[3,]    3    6    7

> m5 <- rbind(1:3,4:6)
> m5
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

The **diag()** function can be used create diagonal Matrix.

```
> diag(3)
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
```

```
> diag(1:3)
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    2    0
[3,]    0    0    3
```

Index and Sub-set in Matrices

```
> m3[1,2]
[1] 2
```

Here we accessed the element in the first row and the second column. We could also subset an entire row or column.

```
> m3[1,]
[1] 1 2 3

> m3[ ,2]
[1] 2 5 8
```

We can also use vectors to subset more than one row or column at a time. Here we subset to the first and third column of the second row.

```
> m3[2 ,c(1, 3)]
[1] 4 6

We can also subset the diagonal elements.
> diag(m3)
[1] 1 5 9
```

Arithmetic Operations in Matrices

You can perform arithmetic operations element by element in matrices using arithmetic operators.

```
> rowSums(m1)
[1] 35 40 45
> colSums(m1)
[1]  6 15 24 33 42

> m1+m1
     [,1] [,2] [,3] [,4] [,5]
[1,]    2    8   14   20   26
[2,]    4   10   16   22   28
[3,]    6   12   18   24   30
```

```
> m1-m2
     [,1] [,2] [,3] [,4] [,5]
[1,]    0    0    0    0    0
[2,]    0    0    0    0    0
[3,]    0    0    0    0    0

> 3*m1
     [,1] [,2] [,3] [,4] [,5]
[1,]    3   12   21   30   39
[2,]    6   15   24   33   42
[3,]    9   18   27   36   45

> m1*m2
     [,1] [,2] [,3] [,4] [,5]
[1,]    1   16   49  100  169
[2,]    4   25   64  121  196
[3,]    9   36   81  144  225

# Notice that m1*m2 does not result the standard matrix multiplication. It is eleme

> t(m1)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
[4,]   10   11   12
[5,]   13   14   15

# Transpose of m1
```

If two matrices are conformal (that is, matrices for which the number of columns of the first is equal to the number of rows of the other), then can be multiplied using the operator %*% to result the matrix multiplication.

```
> m3 <- matrix(1:6,3,2)
> m3
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6

> m4<- matrix(1:6,2,3)
> m4
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

```
> m3%*%m4
        [ ,1]  [ ,2]  [ ,3]
[1 ,]       9     19     29
[2 ,]      12     26     40
[3 ,]      15     33     51
```

Matrix inversion and (equivalent) solution of linear systems of equations can be performed using the function solve(...). For example, to solve the system $ax = b$, we would use solve(a, b). If a single matrix argument is passed to solve(...), it will return the inverse of the matrix.

Example

Consider the following system of equations.

$$x_1 + 2x_2 = 1$$
$$3x_1 + x_2 = 0$$

In matrix from, this becomes $ax = b$ where

$$a = \begin{pmatrix} 1 & 2 \\ 3 & 1 \end{pmatrix}$$

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

$$b = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

```
> m6 <- array(c(1,3,2,1),c(2,2))
> m6
        [ ,1]  [ ,2]
[1 ,]       1      2
[2 ,]       3      1

>  v1 <- array(c(1,0),  c(2,1))
>  v1
        [ ,1]
[1 ,]       1
[2 ,]       0

> solve(m6,v1)
        [ ,1]
[1 ,]  -0.2
[2 ,]   0.6

> solve(m6)
        [ ,1]  [ ,2]
[1 ,]  -0.2    0.4
```

```
[2,]    0.6  -0.2

> solve(m6) %*% v1 # does the same as solve(m6,v1)
     [,1]
[1,] -0.2
[2,]  0.6
```