Matrix Algebra in R

Dr. Noah Mutai

1 Matrices

A matrix is a collection of data elements of the same type arranged in a two-dimensional rectangle

To create a matrix we must indicate the elements, as well as the number of rows (nrow) and columns (ncol)

To declare a matrix in R use the function mat () and a name. As a rule of thumb, matrix names are capitalized. However, R takes even lower case.

```
A <- matrix(1:9, nrow = 3, ncol = 3)
A
```

To check the components of the mat () function use;

```
# ?matrix
```

By default, any matrix is created column-wise. To change that we set an additional argumnet byrow = TRUE.

```
A <- matrix(1:9, nrow = 3, ncol = 3, byrow = TRUE)
A
```

It is not necessary to specify both the number of rows and columns We can only indicate one of them. The number of elements must be a multiple of the number of rows or columns

```
A <- matrix(1:9, nrow = 3)
A


B <- matrix(1:9, ncol = 3)
B
```

To get the class use the fucntion class()

```
class(A)
```

Other functions include

dim() dimesnion of matrix nrow total rows ncol total columns

example;

```
dim(A)
nrow(A)
ncol(A)
```

To check is somethin is a matrix use;

```
is.matrix(A)
```

1.0.1 rbind() AND cbind()

rbind() and cbind() allow us to bind vectors in order to create a matrix. The vectors must have the same length

Example;

Declare three vectors.

```
a <- c(1,2,3,4)
b <- c(10,11,12,13)
c <- c(20,30,40,50)
```

If we use rbind(), our vectors will be rows

```
e <- rbind(a, b, c)
e
```

The result is a matrix.

```
class(e)
```

The order does not matter.

```
e <- rbind(c, a, b)
e
```

Vectors can be repeated.

```
e <- rbind(a, b, c, a)
e
```

It is not necessary to create the vectors first. We can enter them directly in the rbind() function.

```
e <- rbind(c(1,2,3), c(7,8,9), c(2,3,4))
e
```

If we use cbind() the vectors will be columns.

```
e <- cbind(a, b, c)
e
```

1.0.2 Naming the rows and columns of a matrix

Naming with dimnames()

```
e <- matrix(c(1,2,3,4,5,6), nrow = 2,
            dimnames = list(c("row1", "row2"), c("col1", "col2", "col3")))
e
```

Using the functions rownames() and colnames()

```
e <- matrix(c(1,2,3,4,5,6), nrow = 2)
rownames(e) <- c("row1", "row2")
colnames(e) <- c("col1", "col2", "col3")
e
```

Remove row and column names, assign NULL.

```
rownames(e) <- NULL
colnames(e) <- NULL
e
```

1.0.3 Indexing matrices

Indexing means accessing one or several matrix elements. Indices must be put between square brackets We must use two indices: one for the row and the other one for the column

```
e <- matrix(1:16, nrow = 4, byrow = TRUE)
e
```

Access the element on row 3, column 2

```
e[3,2]
```

Access the element on row 4, column 1

```
e[4,1]
```

1.1 Operations on matrices

1.1.1 Matrix multiplication

```

s <- 10
s*A
```

1.1.1.1 Scalar multiplication

```

A%*%B
```

1.1.1.2 Matrix multiplication

```
A + B
A - B
```

1.1.1.3 Matrix addition and substraction

```
t(A)
t(B)
```

1.1.1.4 Transpose

1.1.1.5 Inverse If it exist;

```
A <- matrix(c(2,4,6,-1,2,-1,10,11,12), ncol = 3, nrow = 3)
solve(A)
```

```
det(A)
```

1.1.1.5.1 Determinant

1.1.2 Other operations

Combining matrices;

Rowise combination.

```
rbind(A,B)
```

Columnwise combination.

```
cbind(A,B)
```

Row and column sums.

```
colSums(A)
rowSums(A)
```

Row and column means.

```
colMeans(A)
rowMeans(A)
```

Some types of matrices.

Identity matrix

```
C <- diag(3)
C
```

Unity matrix

```
U <- matrix(1, 3, 2)
U
```

1.2 Sytem of linear equations

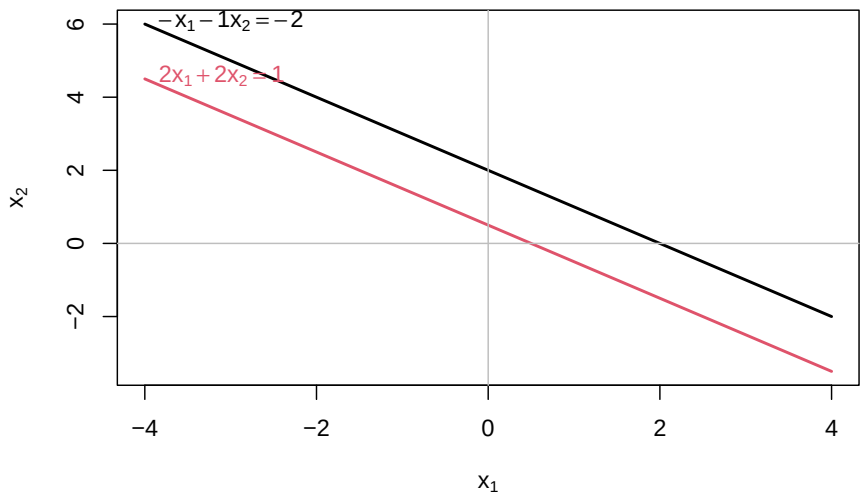
1.2.1 Use R package matlib

Install and load matlib R package.

```
# install.packages("matlib")
library(matlib)
```

Example

```
A <- matrix(c(-1, 2, -1, 2), 2, 2)
b <- c(-2,1)
showEqn(A, b)
plotEqn(A,b)
```




```
Solve(A, b, fractions = TRUE)
```

Example;

Solve the equation;

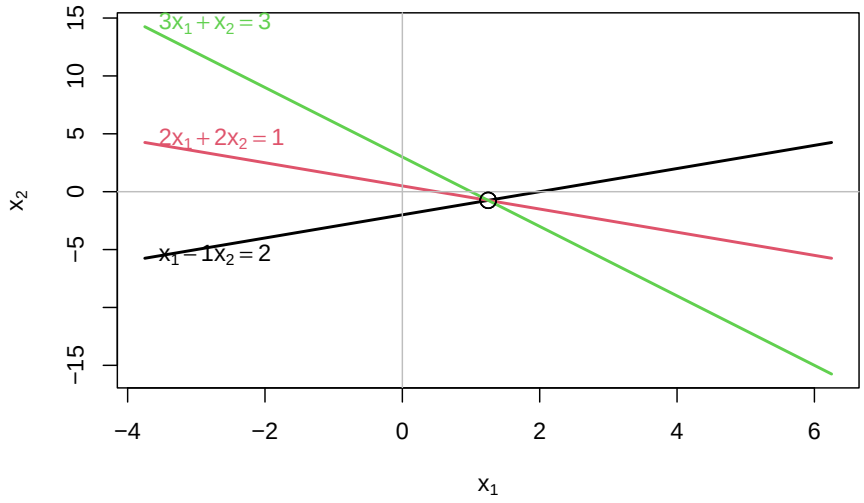
$$\begin{array}{rcrcrcr} 4x & - & 3y & + & z & = & -10 \\ 2x & + & y & + & 3z & = & 0 \\ -x & + & 2y & - & 5z & = & 17 \end{array}$$

```
A <- matrix(c(4,-3,1,2,1,3,-1,2,-5), nrow = 3, ncol = 3)
b <- c(-10,0,17)
showEqn(A,b)
Solve(A, b, fractions = TRUE)
plotEqn3d(A,b)
```

```
A <- matrix(c(1,-2,4,-5,2,3,6,2,4), nrow = 3, ncol = 3)
b <- c(23,45,32)
# plotEqn3d (A,b)
Solve(A, b, fractions = TRUE)
```

Another example;

```
A <- matrix(c(1,2,3, -1, 2, 1), 3, 2)
b <- c(2,1,3)
showEqn(A, b)
plotEqn(A,b)
```




```
Solve(A, b, fractions=TRUE)
```

1.2.2 Applying functions to matrices

To perform operations on the matrix rows and columns we can use the apply() function

Create a matrix.

```r
A <- matrix(10:25, nrow = 4)
A
```

Compute the sum of the elements on each row and column, respectively

```r
apply(A, 1, sum)
apply(A, 2, sum)
```

Compute the product of the elements on each row and column, respectively.

```r
apply(A, 1, prod)
apply(A, 2, prod)
```

Compute the mean for each row and column, respectively

```r
apply(A, 1, mean)
apply(A, 2, mean)
```

Compute the standard deviation for each row and column, respectively

```r
apply(A, 1, sd)
apply(A, 2, sd)
```

Compute the cumulative sums for the data values in each row

```r
apply(A, 1, cumsum)
```

The cumulative sums are computed by row, BUT the matrix is built column-wise (the default way in R)

Create a mtrix row-wise with byrow=TRUE

```r
B <- matrix(apply(A, 1, cumsum), nrow = 4, byrow = TRUE)
B
```

1.2.3 compute the cumulative sums for each column

```r
apply(A, 2, cumsum)
```