STAT 40001/STAT 50001 Statistical Computing

Lecture 3

Department of Mathematics and Statistics

Outline

The fundamental data type in R is the vector. In this lecture we will discuss

- **Creating a Matrix**
- **Matrix Operations**
- **Transpose and Concatenation**
- **Solve a System of Equations**
- **Data Frame**
- **Merging, Sorting data frames**

A **matrix** is a vector with two additional attributes: the number of rows and the number of columns. A column is a vertical representation of data, while a row is a horizontal representation of data. Since matrices are vectors, they also have modes, such as numeric and character.

Matrices are special cases of a more general R type of object: **arrays.** Arrays can be multidimensional. For example, a three-dimensional array would consist of rows, columns, and layers, not just rows and columns as in the matrix case. Each value of the matrix can be located by its row and column numbers.

The basic R command to define a matrix requires a list of elements (c(.,.,., .,)) and the number of rows **nrow** in the matrix.

We create $\begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix}$ using the following steps in R

Creating a matrix

```
> C <- matrix(c(1,2,3,4,5,6,7,8,9),nrow=3, ncol=3)
> C
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

R uses the **nrow** command to set the dimension of the matrix. We can instruct R to enter rows first by adding the command **byrow=T**

```
> D <- matrix(c(1,2,3,4,5,6,7,8,9),nrow=3,byrow=T)
> D
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

Naming Matrix row and column

```
> M <- matrix(c(1,2,3,4,5,6,7,8,9),nrow=3,byrow=T)
> M
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
> rownames(M)<-c("A","B","C")
> colnames(M)<-c("D","E","F")
> M
  D E F
A 1 2 3
B 4 5 6
C 7 8 9
> dimnames(M)
[[1]]
[1] "A" "B" "C"
```

Dimension of matrix

The dimension of a matrix can be checked using **dim()**or **attributes()**
We can add optional character "dimnames" giving the row and column names respectively, list names will be used as names for the dimensions

```
> x <- matrix(c(5,0,6,1,3,5,9,5,7,1,5,3), nrow=3, ncol=4, byrow=TRUE,
 dimnames=list(rows=c("r.1", "r.2", "r.3"), cols=c("c.1", "c.2", "c.3", "c.4")))
> x
     cols
rows  c.1 c.2 c.3 c.4
  r.1   5   0   6   1
  r.2   3   5   9   5
  r.3   7   1   5   3

> dim(x)
[1] 3 4
> attributes(x)
$dim
[1] 3 4

$dimnames
$dimnames$rows
[1] "r.1" "r.2" "r.3"

$dimnames$cols
[1] "c.1" "c.2" "c.3" "c.4"
```

Matrix-Example

```
> x <- matrix(c(5,0,6,1,3,5,9,5,7,1,5,3), nrow=3, ncol=4, byrow=TRUE)
> x
     [,1] [,2] [,3] [,4]
[1,]    5    0    6    1
[2,]    3    5    9    5
[3,]    7    1    5    3
> x[2,3] # Row 2, Column 3
[1] 9
> x[1,] # Row 1
[1] 5 0 6 1
> x[,2] # Column 2
[1] 0 5 1

> x[c(1,3),] # Rows 1 and 3, all Columns
    [,1] [,2] [,3] [,4]
[1,] 5   0  6   1
[2,] 7   1  5   3

> x[3,] # Row 3 in the form of a vector
[1] 7 1 5 3

> x[3,,drop=F] # Row 3 in the form of a matrix
     [,1] [,2] [,3] [,4]
[1,]    7    1    5    3

> x[-1,] # matrix x without its first row
     [,1] [,2] [,3] [,4]
[1,]    3    5    9    5
[2,]    7    1    5    3
```

Matrix

```
> x <- matrix(c(5,0,6,1,3,5,9,5,7,1,5,3), nrow=3, ncol=4, byrow=TRUE)
> x
     [,1] [,2] [,3] [,4]
[1,]    5    0    6    1
[2,]    3    5    9    5
[3,]    7    1    5    3

>x[,x[1,]>4]# Choose columns of matrix x with the value in 1st line greater than 4
     [,1] [,2]
[1,]    5    6
[2,]    3    9
[3,]    7    5

>x[x>4]
[1] 5 7 5 6 9 5 5

> x[x>3]<-NA # Replacing with NA
> x
     [,1] [,2] [,3] [,4]
[1,]   NA    0   NA    1
[2,]    3   NA   NA   NA
[3,]   NA    1   NA    3
```

Lower and Upper Triangular Part of a Matrix

```
> M <- matrix(c(5,0,6,1,3,5,9,5,7), nrow=3, byrow=TRUE)
> M
     [,1] [,2] [,3]
[1,]    5    0    6
[2,]    1    3    5
[3,]    9    5    7
> M_lower<- M[lower.tri(M, diag=TRUE)]
> M_lower
[1] 5 1 9 3 5 7
> M_upper<- M[upper.tri(M, diag=TRUE)]
> M_upper
[1] 5 0 3 6 5 7
```

Common Vectors

Unit Vector

```
> U <- matrix(1,3,1)
> U
     [,1]
[1,]    1
[2,]    1
[3,]    1
```

Zero Vector

```
> V <- matrix(0,3,1)
> V
     [,1]
[1,]    0
[2,]    0
[3,]    0
```

Common Matrices

Unit Matrix

```
> U <- matrix(1,3,2)
> U
     [,1] [,2]
[1,]    1    1
[2,]    1    1
[3,]    1    1
```

Zero Matrix

```
> Z <- matrix(0,3,2)
> Z
     [,1] [,2]
[1,]    0    0
[2,]    0    0
[3,]    0    0
```

Common Matrices

Identity Matrix

```
> I <- diag(c(1,1,1))
> I
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
```

Diagonal Matrix

```
> S <- matrix(c(2,3,-2,1,2,2,4,2,3),3,3)
> S
     [,1] [,2] [,3]
[1,]    2    1    4
[2,]    3    2    2
[3,]   -2    2    3
> D <- diag(S)
> D
```

Matrix Operations

```
> X
     [,1] [,2] [,3] [,4] [,5]
[1,]    1       0    2      5      3
[2,]    1      1    3      1      3
[3,]    3      1    0      2      2
[4,]    1      0    2      1      0
> mean(X[,5])
[1] 2
>var(X[4,])
[1] 0.7
>rowSums(X) # Note the uppercase S
[1] 11 9 8 4
>colSums(X)
[1] 6 2 7 9 8
>rowMeans(X)# Note the uppercase M
[1] 2.2 1.8 1.6 0.8
```

Concatenation of Matrices

Horizontal Concatenation

```
> A<-matrix(c(2,3,-2,1,2,2), 3,2)
> A
     [,1] [,2]
[1,]    2    1
[2,]    3    2
[3,]   -2    2

> B <- matrix(c(1,3,2,1,4,2),3,2)
> B
     [,1] [,2]
[1,]    1    1
[2,]    3    4
[3,]    2    2
> C <- cbind(A,B)
> C
     [,1] [,2] [,3] [,4]
[1,]    2    1    1    1
[2,]    3    2    3    4
[3,]   -2    2    2    2
```

Vertical Concatenation(Appending) of Matrices

```
> A<-matrix(c(2,3,-2,1,2,2), 3,2)
> A
     [,1] [,2]
[1,]    2    1
[2,]    3    2
[3,]   -2    2
> B <- matrix(c(1,3,2,1,4,2),3,2)
> B
     [,1] [,2]
[1,]    1    1
[2,]    3    4
[3,]    2    2
> C <- rbind(A,B)
> C
     [,1] [,2]
[1,]    2    1
[2,]    3    2
[3,]   -2    2
[4,]    1    1
[5,]    3    4
[6,]    2    2
```

Matrix Operations

The **apply** function is used for applying functions to the rows or columns of matrices. In a matrix **margin 1 refers to rows** and **margin 2 refers to the column**

```
> X=matrix(1:24, nrow=4)
> X
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    5    9   13   17   21
[2,]    2    6   10   14   18   22
[3,]    3    7   11   15   19   23
[4,]    4    8   12   16   20   24

> rowSums(X)
[1] 66 72 78 84
> colSums(X)
[1] 10 26 42 58 74 90
> apply(X,1,sum)
[1] 66 72 78 84
> apply(X,2, sum)
[1] 10 26 42 58 74 90
```

Matrix Operations

Function	Description
A+B	Addition of matrices
A-B	Subtraction of matrices
$A\%*\%B$	Product of matrices
t(A)	Transposition of a matrix
diag(5)	Identity matrix of order 5
diag(A)	Vector with the values of the diagonal elements
crossprod(A,B)	Cross product ($t(A)\%*\%B$)
det(A)	Determinant of matrix A
svd(A)	Singular value decomposition
eigen(A)	Matrix diagonalisation
solve(A)	Matrix inversion
solve(A,b)	Solving linear systems
chol(A)	Cholesky decomposition
qr(A)	QR decomposition

Addition and Subtraction

```
> C <- matrix(c(1,2,3,4,5,6,7,8,9),nrow=3)
> D <- matrix(c(1,2,3,4,5,6,7,8,9),nrow=3,byrow=T)
> C
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> D
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
> C+D
     [,1] [,2] [,3]
[1,]    2    6   10
[2,]    6   10   14
[3,]   10   14   18

> C-D
```

Matrix Multiplication by a scalar

```
> y<-matrix(c(1,2,3,4,5,6), nrow=3)
> y
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> 3*y
     [,1] [,2]
[1,]    3   12
[2,]    6   15
[3,]    9   18
```

Matrix Multiplication

```
> C <- matrix(c(1,2,3,4,5,6,7,8,9),nrow=3)
> D <- matrix(c(1,2,3,4,5,6,7,8,9),nrow=3,byrow=T)

> C%*%D
     [,1] [,2] [,3]
[1,]   66   78   90
[2,]   78   93  108
[3,]   90  108  126

> D%*%C
     [,1] [,2] [,3]
[1,]   14   32   50
[2,]   32   77  122
[3,]   50  122  194
```

Transpose and Concatenation

```
> A <- matrix(c(1,2,3),nrow=3)
> AT<-t(A)
> A
     [,1]
[1,]    1
[2,]    2
[3,]    3
> AT
     [,1] [,2] [,3]
[1,]    1    2    3

> A<-matrix(c(1,4,5,6,7,8, 6,8),nrow=2)
> A
     [,1] [,2] [,3] [,4]
[1,]    1    5    7    6
[2,]    4    6    8    8
> B<-matrix(1:12,nrow=3)
> B
     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> rbind(A,B)
     [,1] [,2] [,3] [,4]
[1,]    1    5    7    6
[2,]    4    6    8    8
[3,]    1    4    7   10
[4,]    2    5    8   11
[5,]    3    6    9   12
```

Inverse of a Matrix

The inverse of a matrix **A** is obtained using the solve command, **solve (A)**.

```
> A <- matrix(c(3,1,4,6),nrow=2)
> A
     [,1] [,2]
[1,]    3    4
[2,]    1    6
> solve(A)
            [,1]       [,2]
[1,]  0.42857143 -0.2857143
[2,] -0.07142857  0.2142857
```

Solve a system of equations

Note that we can express a system of equations in matrix form. Consider a system of equations

$$\begin{aligned} 2x - y &= 4 \\ 2x + 3y &= 12 \end{aligned}$$

which can be written as

$$AX = Y$$

where

$$A = \begin{pmatrix} 2 & -1 \\ 2 & 3 \end{pmatrix}, X = \begin{pmatrix} x \\ y \end{pmatrix}, Y = \begin{pmatrix} 4 \\ 12 \end{pmatrix}$$

hence, the solution is given by

$$X = A^{-1}Y$$

We can use the R code below to solve these system of equations

Solving Equations

```
> A=matrix(c(2,-1,2,3),nrow=2, byrow=T)
> A
     [,1] [,2]
[1,]    2   -1
[2,]    2    3
> Y=matrix(c(4,12), nrow=2)
> Y
     [,1]
[1,]    4
[2,]   12
> X=solve(A)%*%Y
> X
     [,1]
[1,]    3
[2,]    2
```

Data Frames

A data frame is like a matrix, with a two-dimensional rows and columns structure. However, it differs from a matrix in that each column may have a different mode. For instance, one column may consist of numbers, and another column might have character strings.

```
> names<-c("Joe", "Peter","William")
> age<-c(45,78,60)
> data<-data.frame(names,age)
> data
    names age
1     Joe  45
2   Peter  78
3 William  60
```

Accessing Data Frames

We can extract the information in the data frame as in the matrix

```
>  names<-c("Joe", "Peter","William")
>  age<-c(45,78,60)
>  data<-data.frame(names,age)
> data
    names age
1     Joe  45
2   Peter  78
3 William  60

> data[2,2] # extracting the information of the data
[1] 78

> data[[1]]
[1] Joe     Peter   William
Levels: Joe Peter William
> data[[2]]
[1] 45 78 60

> data[1]
    names
1     Joe
2   Peter
3 William

> data[2]
  age
1  45
2  78
3  60
```

Expanding Data Frame

Components can be added easily to a data frame in the natural way.

```
> data$Major<-c("Math", "Biology", "Statistics")
> data
    names age      Major
1     Joe  45       Math
2   Peter  78    Biology
3 William  60 Statistics
```

Suppose we want to add two more entries
Jim 23 Physics
Lucas 54 Math

```
> new=data.frame(names=c("Jim", "Lucas"), age=c(23,54), Major=c("Physics","Math"))
> new
  names age   Major
1   Jim  23 Physics
2 Lucas  54    Math

> newdata<-rbind(data,new)
    names age      Major
1     Joe  45       Math
2   Peter  78    Biology
3 William  60 Statistics
4     Jim  23    Physics
5   Lucas  54       Math
```

Select Rows base variable values

```
> newdata<-rbind(data,new)
    names age      Major
1     Joe  45       Math
2   Peter  78    Biology
3 William  60 Statistics
4     Jim  23    Physics
5   Lucas  54       Math
> attach(newdata)
> old=newdata[age>45,]
> old
    names age      Major
2   Peter  78    Biology
3 William  60 Statistics
5   Lucas  54       Math
```

Sort data frame by selected column

```
> name<-c("S1", "S2", "S3", "S4", "S5")
> gender=c("F","M", "M", "F", "F")
> weight=c(120,134,230, 125, 137)
> age=c(34,27,39,32,55)
> data=data.frame(name,gender, weight, age)
> data
  name gender weight age
1   S1      F    120  34
2   S2      M    134  27
3   S3      M    230  39
4   S4      F    125  32
5   S5      F    137  55
```

See data=data.frame(row.names=name,gender, weight, age)

```
> newdata=data[order(data[,"age"]),]
> newdata
  name gender weight age
2   S2      M    134  27
4   S4      F    125  32
1   S1      F    120  34
3   S3      M    230  39
```

Sort data frame by selected column

rev: reverses the order from increasing to decreasing

```
> name<-c("S1", "S2", "S3", "S4", "S5")
> gender=c("F","M", "M", "F", "F")
> weight=c(120,134,230, 125, 137)
> age=c(34,27,39,32,55)
> data=data.frame(name,gender, weight, age)
> data
  name gender weight age
1   S1      F    120  34
2   S2      M    134  27
3   S3      M    230  39
4   S4      F    125  32
5   S5      F    137  55
> newdata=data[rev(order(data["age"])),]
> newdata
  name gender weight age
5   S5      F    137  55
3   S3      M    230  39
1   S1      F    120  34
4   S4      F    125  32
2   S2      M    134  27
```

unique function

```
data=data.frame(name=c("Jim", "Lucas", "Marcie","Lucas"),
    age=c(23,54,34,54), Major=c("Physics","Math","CS", "Math"))
data
> data
   name age   Major
1    Jim  23 Physics
2  Lucas  54    Math
3 Marcie  34      CS
4  Lucas  54    Math
```

Note that the entry # 4 is exactly same as # 2

```
> unique (data)
> unique(data)
   name age   Major
1    Jim  23 Physics
2  Lucas  54    Math
3 Marcie  34      CS
>
```

Example

Solve the following system of equations

$$\begin{aligned} x_1 + 3x_2 + 2x_3 &= 3 \\ -x_1 + x_2 + 2x_3 &= -2 \\ 2x_1 + 4x_2 + x_3 &= 10 \end{aligned}$$

```
> data=c(1,3,2,-1,1,2,2,4,1)
> A=matrix(data,nrow=3,byrow=T)
> Y=matrix(c(3,-2,10),ncol=1)
> X=solve(A,Y)
> X
      [,1]
[1,] -2.25
[2,]  4.75
[3,] -4.50
```

Example

Solve the following system of equations

$$\begin{aligned} x + y + z + w &= 0 \\ x + y - z - w &= 1 \\ x - y + z + w &= 3 \\ x - y - z + w &= -1 \end{aligned}$$

```
> data=c(1,1,1,1,1,1,-1,-1,1,-1,1,1,1,-1,-1,1)
> A=matrix(data,nrow=4,byrow=T)
> Y=matrix(c(0,1,3,-1),ncol=1)
> X=solve(A,Y)
> X
     [,1]
[1,]  2.0
[2,] -1.5
[3,]  2.0
[4,] -2.5
```