

## A Primer on Using R

R is a free software environment for statistical computing and graphics. To obtain R, one can download it from the following website: <http://mirrors.ibiblio.org/pub/mirrors/CRAN/>. Like SAS IML, R has built-in operators and functions for most standard matrix operations.

### Reading in Data

A matrix can be defined in two ways in R:

- (1) manually enter them using code, and
- (2) define a file containing the matrix which is later called.

Entering Data in a Matrix:

```
a <- 2 #scalar
b <- matrix(c(1, 2, 3, 4),nrow=1, ncol=4,byrow=TRUE) # 1 x 4 row vector
c <- matrix(c(1.0, 0, 0,
              0.2, 1.0, 0,
              0.8, 0.4, 1.0),
            nrow=3, ncol=3, byrow=TRUE) #3x3 (correlation) matrix
```

The function `c()` combines values separated by commas, into a vector or list. The matrix function `matrix()` creates a matrix from a given set of values. The argument `nrow` specifies the desired number of rows, `ncol` specifies the number of columns and `byrow=TRUE` specifies that the matrix is filled by rows. The default is `byrow=FALSE` where the matrix is filled by columns. The pound sign `#` is used to make comments.

Providing a file of elements:

Assume that you have a space delimited file containing elements of a 9 x 9 (correlation) matrix called "holzinger.txt". The following code is used to read in the file and determine a matrix with the elements from the file "matrix.txt".

```
setwd('') #set the working directory where the file is stored
X <-read.table(file("holzinger.txt"))
```

The function `setwd()` is used to set the working directory to where the file is stored. Another way to change the working directory using windows explorer is explained in this website:

<http://research.stowers-institute.org/efg/R/TechNote/WindowsExplorerWorkingDirectory/index.htm>

## Printing out matrices of interest

R can be used interactively – where statements are executed immediately. Once matrices are defined, the `print()` statement causes R to display the result of each assignment statement in the console.

For example, we want to specify a correlation matrix  $A = \begin{bmatrix} 1.0 & & & \\ 0.5 & 1.0 & & \\ 0.6 & 0.7 & 1.0 & \\ 0.2 & 0.1 & 0.8 & 1.0 \end{bmatrix}$  and wanted to

have R display out the elements of A.

R code:

```
A <- matrix(c(1.0, 0.5, 0.6, 0.2,
              0.5, 1.0, 0.7, 0.1,
              0.6, 0.7, 1.0, 0.8,
              0.2, 0.1, 0.8, 1.0),
            nrow=4, ncol=4, byrow=TRUE)
print(A) #print matrix A in R console
```

The R console will display this:

```
> print(A)
      [,1] [,2] [,3] [,4]
[1,] 1.0 0.5 0.6 0.2
[2,] 0.5 1.0 0.7 0.1
[3,] 0.6 0.7 1.0 0.8
[4,] 0.2 0.1 0.8 1.0
>
```

## Functions which create matrices

a. `diag(size)`

Identity matrix where `size` = number of diagonal 1s.

b. `diag(vector)`

Creates a square matrix with the elements of the vector along the main diagonal.

c. `diag(matrix)`

Returns a row vector which retains the main diagonal of the argument matrix.

Example code:

```
a <- diag(6) # 6x6 identity matrix
b <- diag(c(1,2,3,4))
c <- diag(matrix(c(1, 2, 3, 4),
                 nrow=2, ncol=2, byrow=TRUE)) #vector of elements of principle
axis
print(a)
print(b)
print(c)
```

R console:

```
> print(a)
      [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    0    0    0    0    0
[2,]    0    1    0    0    0    0
[3,]    0    0    1    0    0    0
[4,]    0    0    0    1    0    0
[5,]    0    0    0    0    1    0
[6,]    0    0    0    0    0    1
> print(b)
      [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    0    2    0    0
[3,]    0    0    3    0
[4,]    0    0    0    4
> print(c)
[1] 1 4
>
```

## Matrix Operations

- a. Addition operator: +
- b. Subtraction operator: -
- c. Sign reverse operator: -
- d. Absolute function: `abs(matrix)`
- e. Matrix product operator: `%*%`
- f. Square root function: `sqrt(matrix)`
- g. Transpose operator: `t(matrix)`
- h. Determinant: `det(matrix)`
- i. Matrix inverse operator: `inv(matrix)`
- j. Eigen value and vectors: `eigen(matrix)`
  - a. `$val` are the eigenvalues of matrix
  - b. `$vec` are the eigen vectors of matrix
- k. Minimum value: `min(matrix)`
- l. Maximum value: `max(matrix)`

### Example code:

```
X <- matrix(c(1,2,3,4),nrow=2,ncol=2,byrow=TRUE)
Y <- matrix(c(-4,-3,-2,-1),nrow=2,ncol=2,byrow=TRUE)

a <- X+X
b <- X-X
c <- -Y
d <- abs(Y)
e <- X%*%X
f <- sqrt(X)
g <- t(X)
h <- det(X)
i <- solve(X)
j <- eigen(X)
k <- min(X)
l <- max(X)

print(a)
print(b)
print(c)
print(d)
print(e)
print(f)
print(g)
print(h)
print(i)
print(j$val)
print(j$vec)
print(k)
print(l)
```

## R console:

```
> print(a)
      [,1] [,2]
[1,]    2    4
[2,]    6    8

> print(b)
      [,1] [,2]
[1,]    0    0
[2,]    0    0

> print(c)
      [,1] [,2]
[1,]    4    3
[2,]    2    1

> print(d)
      [,1] [,2]
[1,]    4    3
[2,]    2    1

> print(e)
      [,1] [,2]
[1,]    7   10
[2,]   15   22

> print(f)
      [,1]      [,2]
[1,] 1.000000 1.414214
[2,] 1.732051 2.000000
```

```
> print(g)
      [,1] [,2]
[1,]    1    3
[2,]    2    4

> print(h)
[1] -2

> print(i)
      [,1] [,2]
[1,] -2.0  1.0
[2,]  1.5 -0.5

> print(j$val)
[1]  5.3722813 -0.3722813

> print(j$vec)
      [,1]      [,2]
[1,] -0.4159736 -0.8245648
[2,] -0.9093767  0.5657675

> print(k)
[1] 1

> print(l)
[1] 4
>
```