Introduction to R Software

Basics of Calculations

::::

Matrix Operations

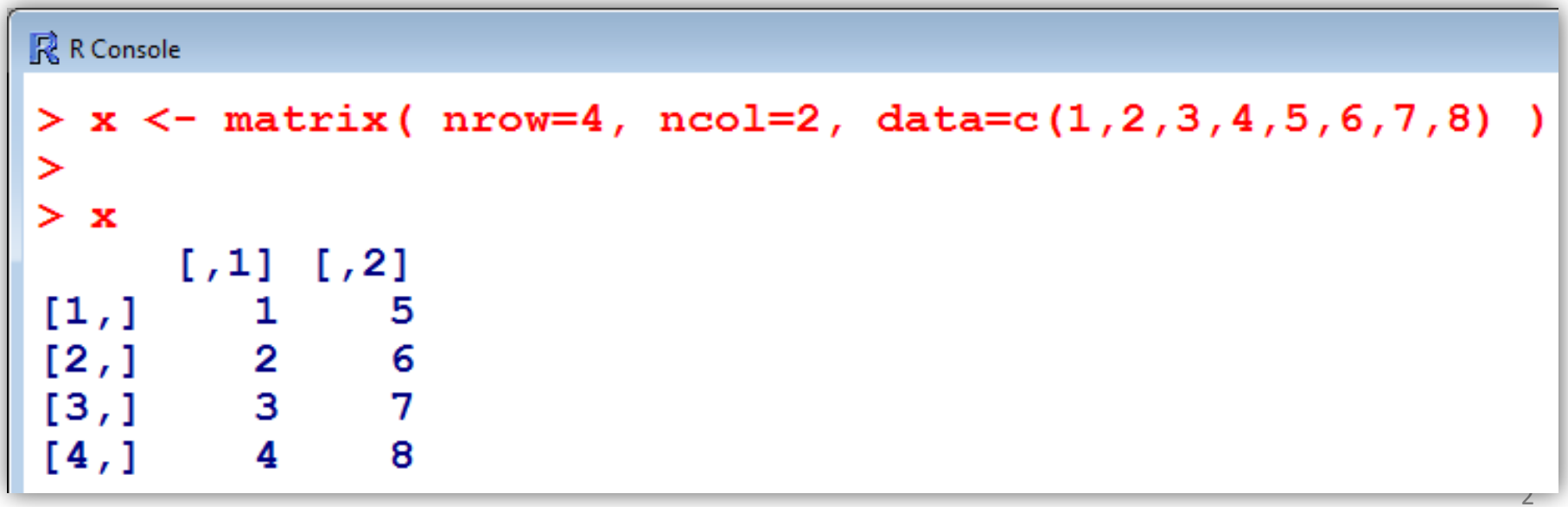
Shalabh

Department of Mathematics and Statistics

Indian Institute of Technology Kanpur

In R, a 4 × 2-matrix *X* can be created with a following command:

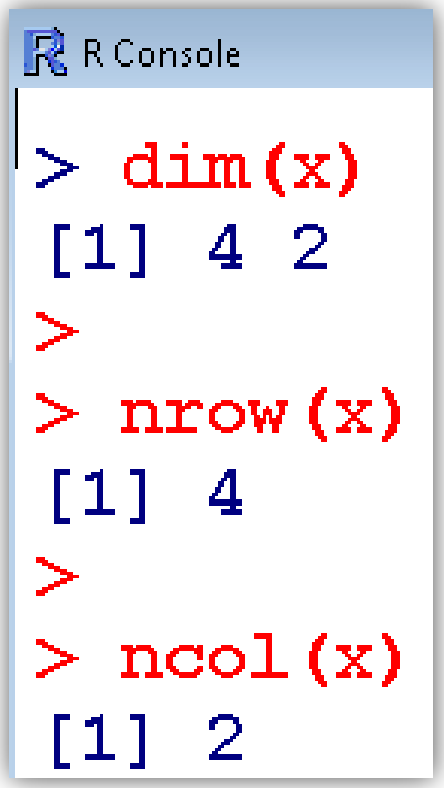
```
> x <- matrix( nrow=4, ncol=2,
                        data=c(1,2,3,4,5,6,7,8) )
> x
     [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    7
[4,]    4    8
```



Properties of a Matrix

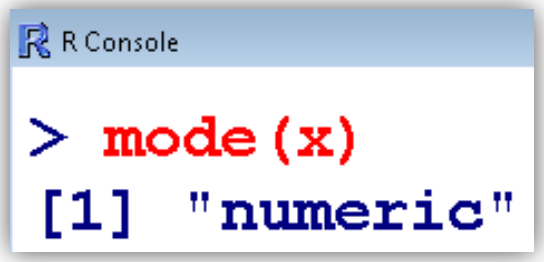
We can get specific *properties* of a matrix:

```
> dim(x)   # tells the
[1] 4 2      dimension of matrix


> nrow(x) # tells
[1] 4        the number of rows


> ncol(x) # tells
[1] 2        the number of columns
```



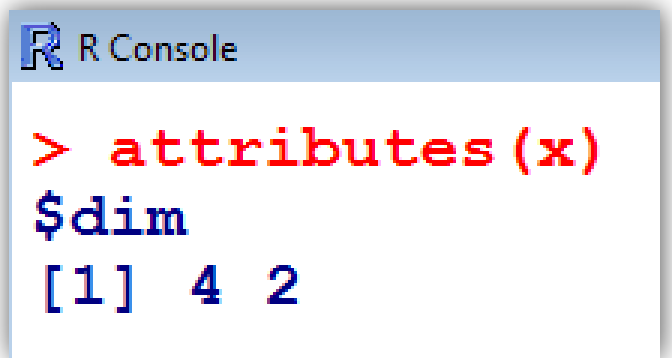
Properties of a Matrix

```
> mode(x)          # Informs the type or storage
                     mode of an object, e.g.,
                     numerical, logical etc.
[1] "numeric"
```



`attributes` **provides all the attributes of an object**

```
> attributes(x) #Informs the dimension of matrix
$dim [1] 4 2
```



Help on the Object "Matrix"

To know more about these important objects, we use R–help on "matrix".

```
> help("matrix")
matrix      package:base      R Documentation

Matrices

Description:

'matrix' creates a matrix from the given set of
values.

'as.matrix' attempts to turn its argument into a
matrix.

'is.matrix' tests if its argument is a (strict)
matrix. It is generic: you can write methods to
handle specific classes of objects, see Internal
Methods.
```

Then we get an overview on how a matrix can be created and what parameters are available:

Usage:

```
matrix(data [= NA,nrow = 1,ncol = 1,byrow = FALSE,
                                    dimnames = NULL)
as.matrix(x)
is.matrix(x)
```

Arguments:

```
data: an optional data vector.
nrow: the desired number of rows
ncol: the desired number of columns
byrow: logical. If 'FALSE'(the default) the matrix
  is filled by columns, otherwise the matrix is
  filled by rows.

dimnames: A 'dimnames' attribute for the matrix: a
  'list' of length 2.

  x: an R object.
```

Then, the meaning of each parameter is explained:

Details:

If either of 'nrow' or 'ncol' is not given, an attempt is made to infer it from the length of 'data' and the other parameter.

If there are too few elements in 'data' to fill the array, then the elements in 'data' are recycled. If 'data' has length zero, 'NA' of an appropriate type is used for atomic vectors and 'NULL' for lists.

'is.matrix' returns 'TRUE' if 'x' is a matrix (i.e., it is _not_ a 'data.frame' and has a 'dim' attribute of length 2) and 'FALSE' otherwise.

'as.matrix' is a generic function. The method for data frames will convert any non-numeric/complex column into a character vector using 'format' and so return a character matrix, except that all-logical data frames will be coerced to a logical matrix.

Finally, references and cross-references are displayed...

```
References:
   Becker, R. A., Chambers, J. M. and Wilks, A.
   R. (1988) _The New S Language_. Wadsworth &
   Brooks/Cole.

See Also:
   'data.matrix', which attempts to convert to
   a numeric matrix.
```

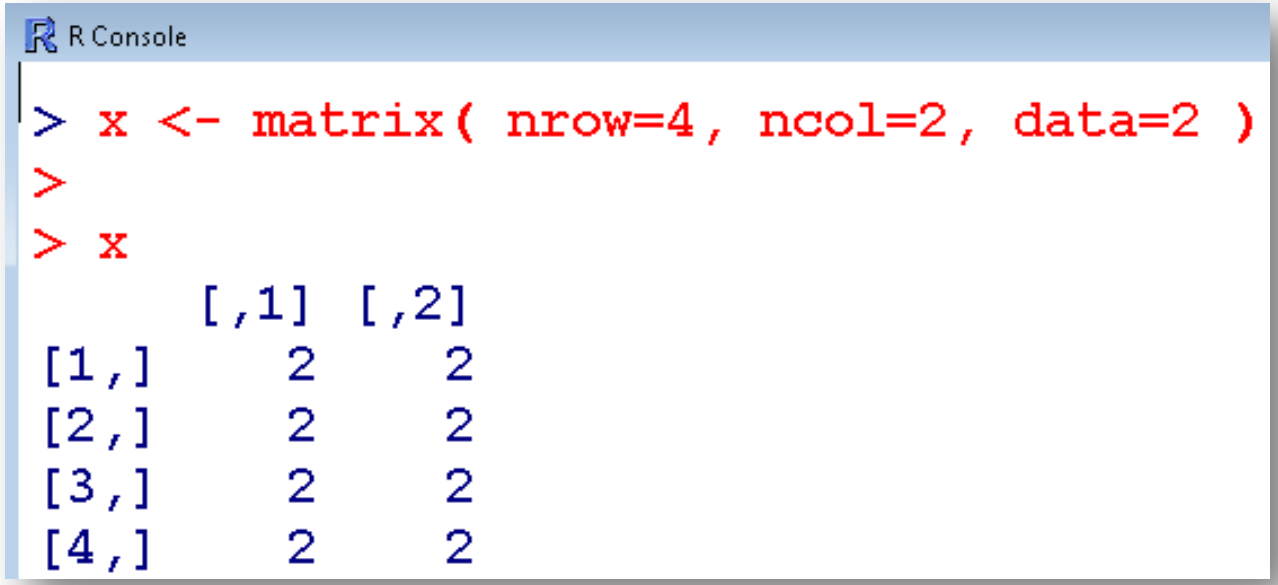
.. as well as an example:

```
Examples:
   is.matrix(as.matrix(1:10))
   data(warpbreaks)
   !is.matrix(warpbreaks)# data.frame, NOT matrix!
   warpbreaks[1:10,]
   as.matrix(warpbreaks[1:10,]) #using
     as.matrix.data.frame(.) method
```

Matrix Operations

- **Assigning a specified number to all matrix elements:**

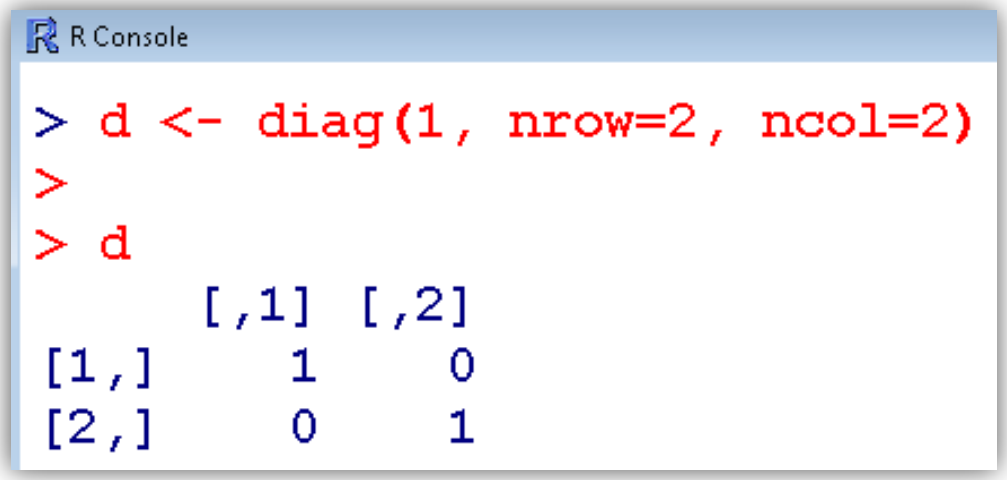
```
> x <- matrix( nrow=4, ncol=2, data=2 )
> x
     [,1] [,2]
[1,]    2    2
[2,]    2    2
[3,]    2    2
[4,]    2    2
```



Matrix Operations

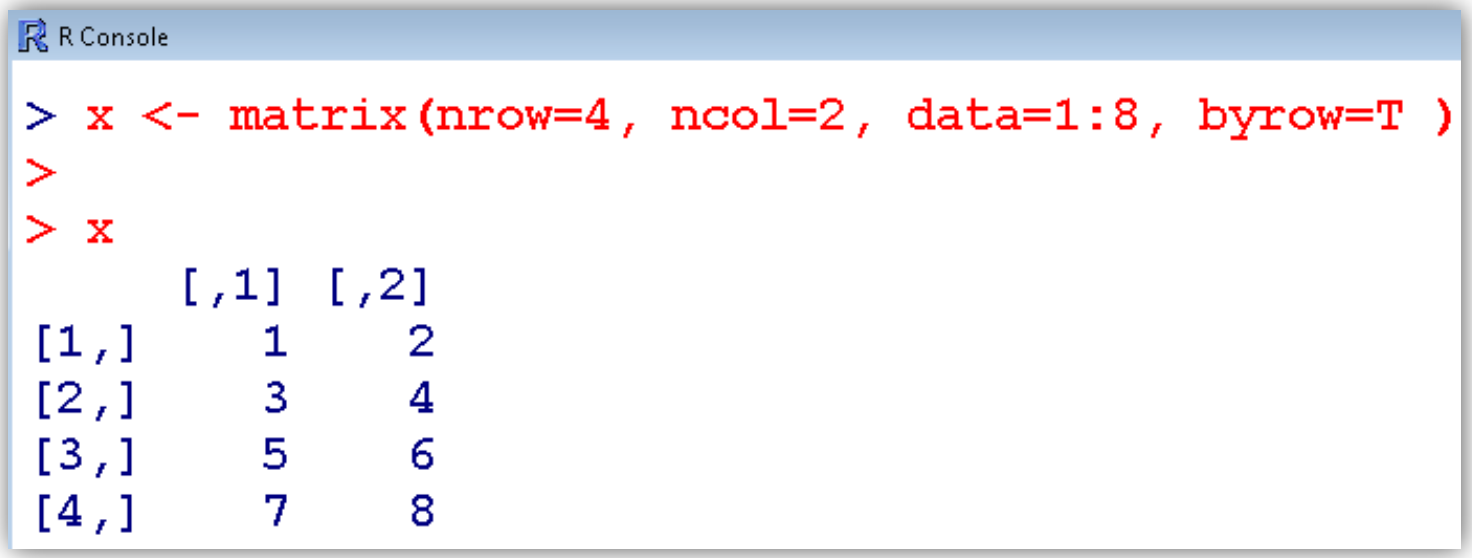
- **Construction of a diagonal matrix, here the identity matrix of a dimension 2:**

```
> d <- diag(1, nrow=2, ncol=2)

> d
     [,1] [,2]
[1,]    1    0
[2,]    0    1
```



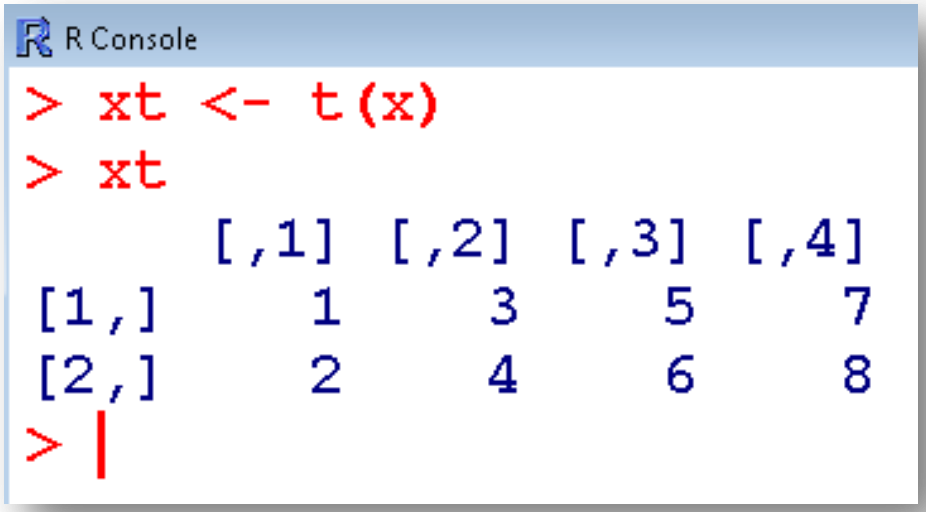
- **Transpose of a matrix *X*: *X'***

```
> x <- matrix(nrow=4, ncol=2, data=1:8, byrow=T )
> x
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
[4,]    7    8
```



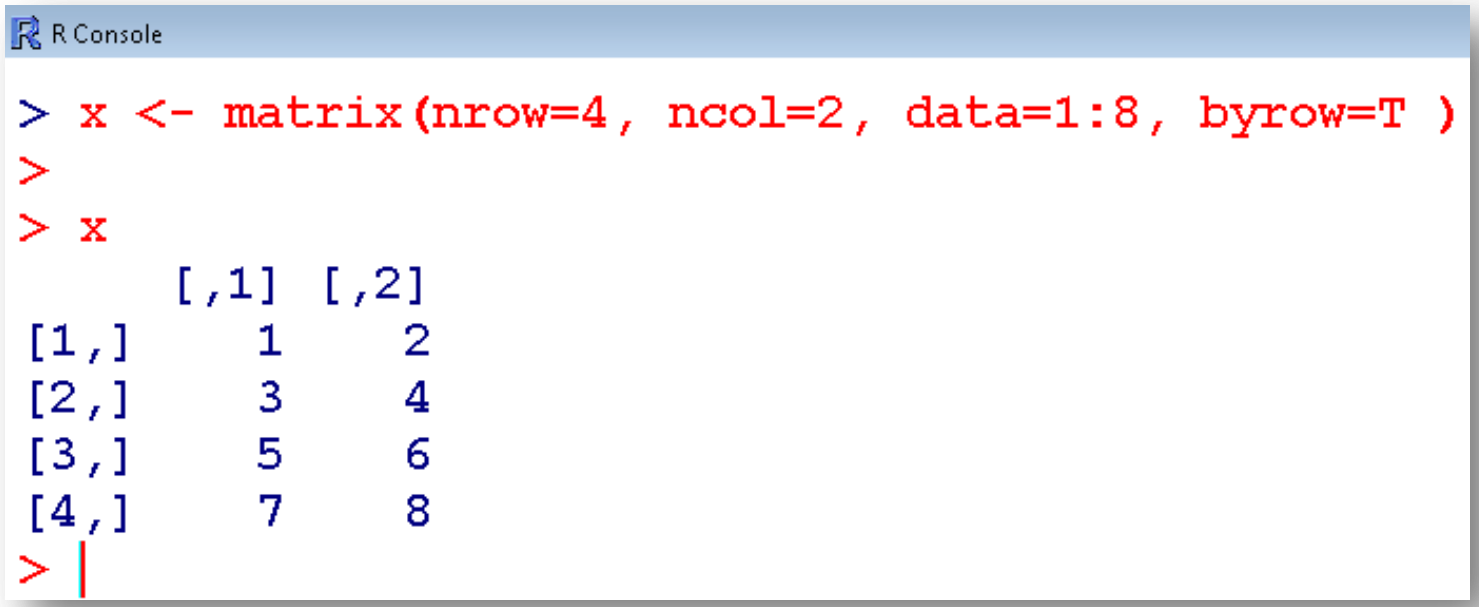
- **Transpose of a matrix *X*: *X'***

```
> xt <- t(x)

> xt
     [,1] [,2] [,3] [,4]
[1,]    1    3    5    7
[2,]    2    4    6    8
```



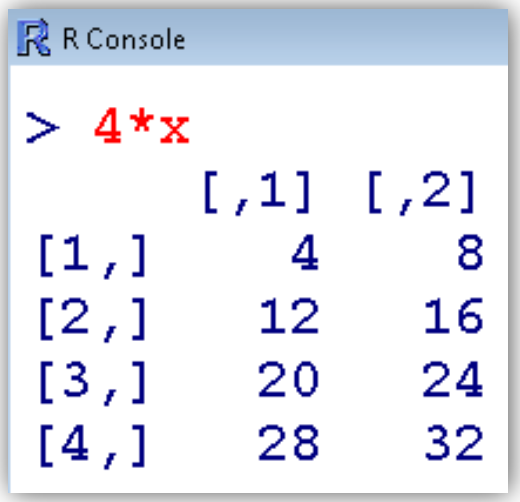
- **Multiplication of a matrix with a constant**

```
> x <- matrix(nrow=4, ncol=2, data=1:8, byrow=T )

> x
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
[4,]    7    8
```



- **Multiplication of a matrix with a constant**

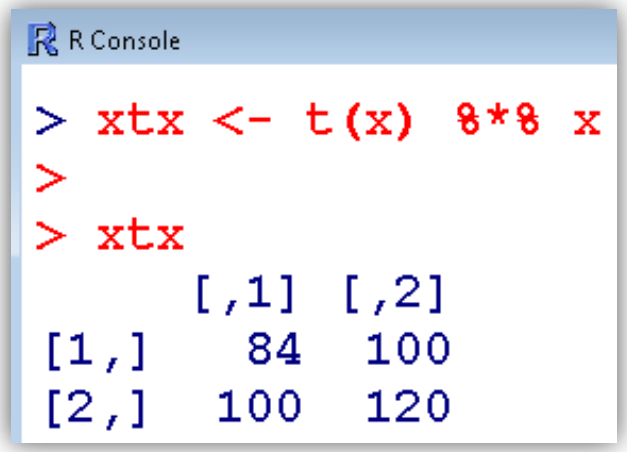
```
> 4*x
     [,1] [,2]
[1,]    4    8
[2,]   12   16
[3,]   20   24
[4,]   28   32
```



- **Matrix multiplication: operator %*%**

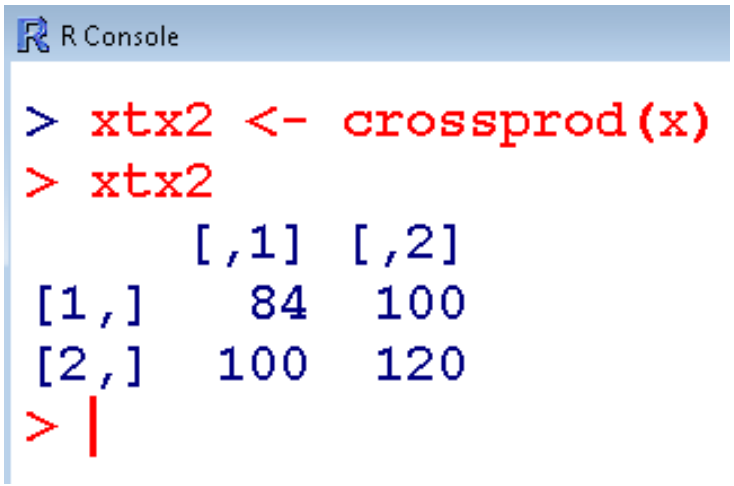
Consider the multiplication of *X'* with *X*

```
> xtx <- t(x) %*% x

> xtx
     [,1]   [,2]
[1,]   84    100
[2,]  100    120
```



- **Cross product of a matrix *X*, *X'X*, with a function `crossprod`**

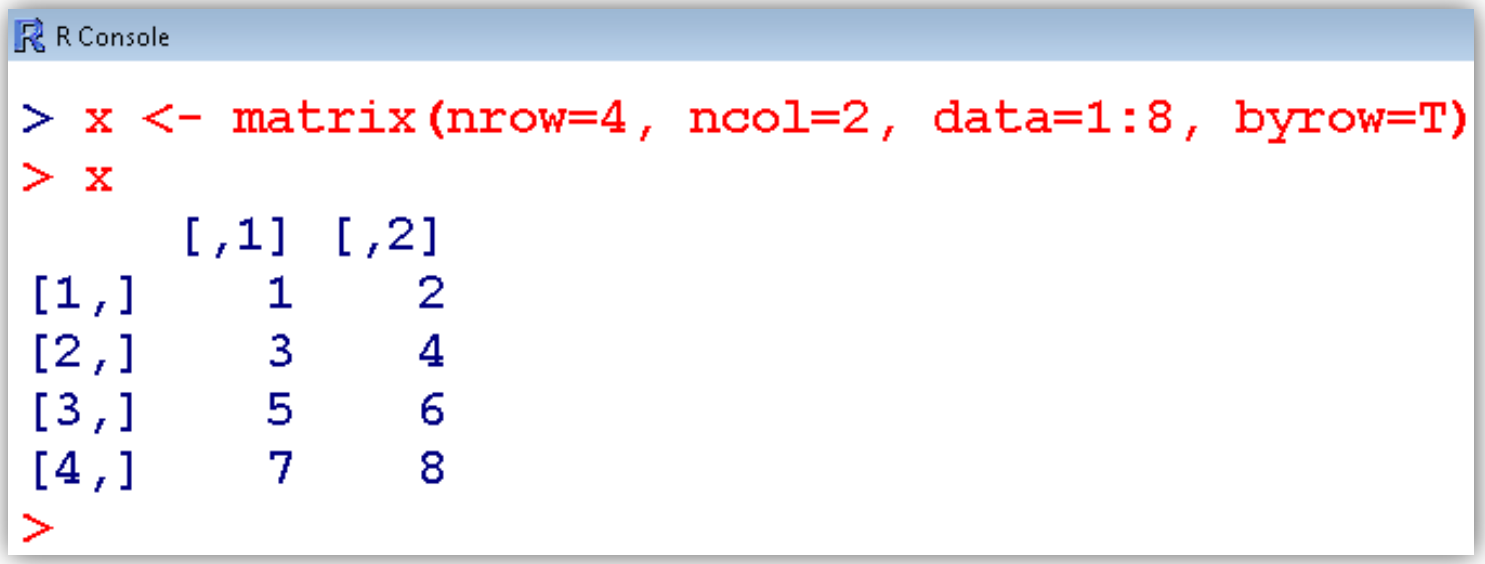
```
> xtx2 <- crossprod(x)

> xtx2
     [,1]   [,2]
[1,]   84    100
[2,]  100    120
```



Note: Command `crossprod()` executes the multiplication faster than the conventional method with `t(x) %*% x`

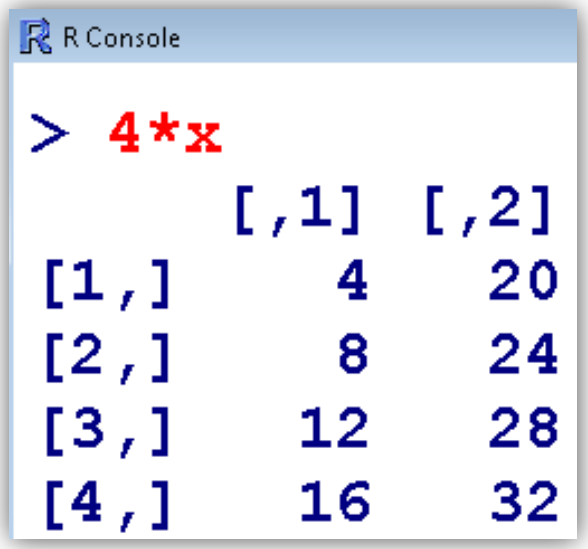
- **Addition and subtraction of matrices (of same dimensions) can be executed with the usual operators + and -**

```
> x <- matrix(nrow=4, ncol=2, data=1:8, byrow=T)

> x
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
[4,]    7    8
```



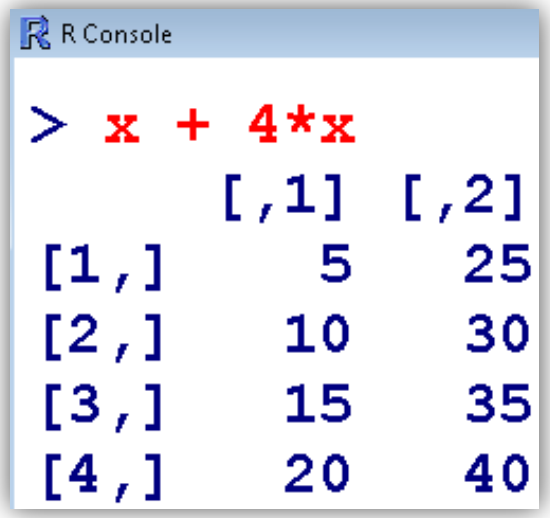
- **Addition and subtraction of matrices (of same dimensions!) can be executed with the usual operators + and -**

```
> 4*x
     [,1] [,2]
[1,]    4    8
[2,]   12   16
[3,]   20   24
[4,]   28   32
```

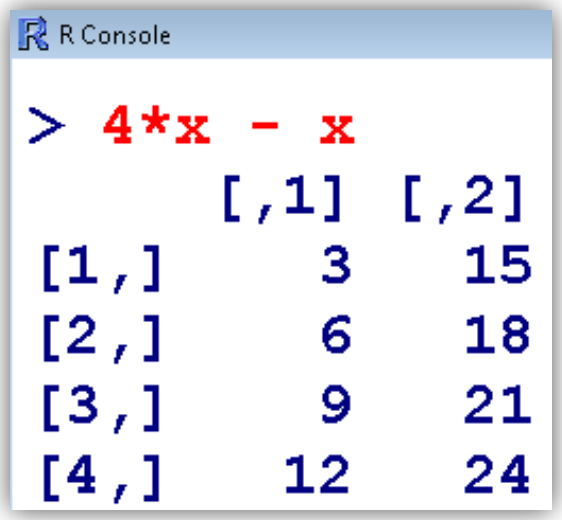


- **Addition and subtraction of matrices (of same dimensions!) can be executed with the usual operators + and -**

```
> x + 4*x
     [,1] [,2]
[1,]    5   10
[2,]   15   20
[3,]   25   30
[4,]   35   40
```

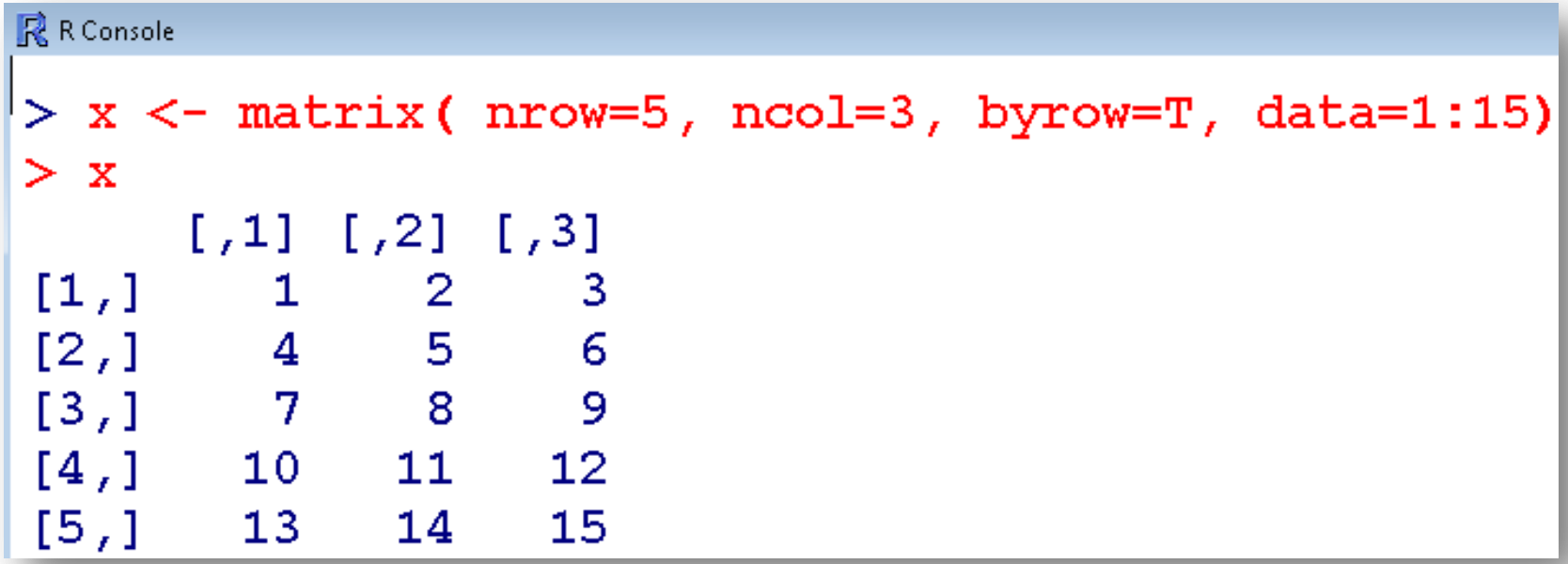



```
> 4*x - x
     [,1] [,2]
[1,]    3    6
[2,]    9   12
[3,]   15   18
[4,]   21   24
```



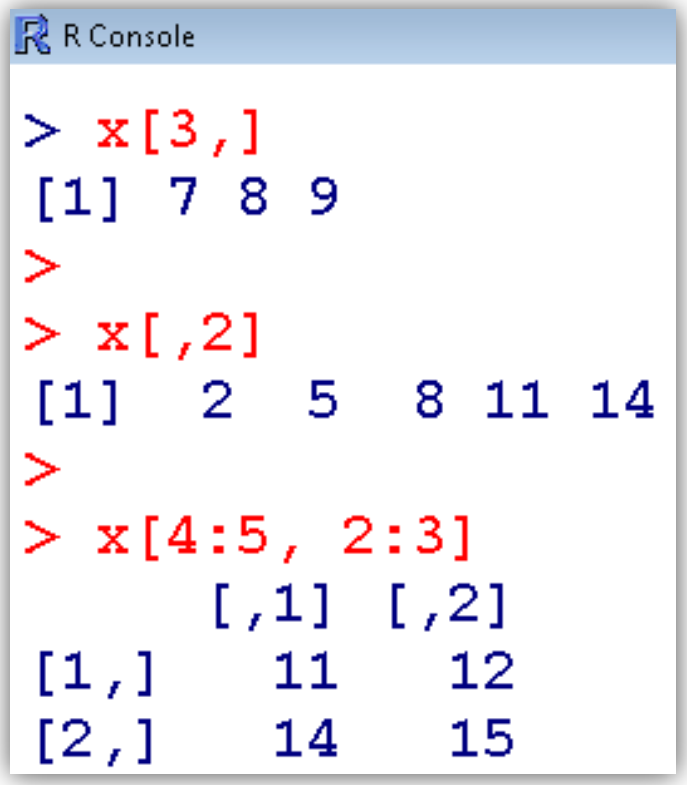
- **Access to rows, columns or submatrices:**

```
> x <- matrix( nrow=5, ncol=3, byrow=T, data=1:15)
> x
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
[4,]   10   11   12
[5,]   13   14   15
```



- **Access to rows, columns or submatrices:**

```
> x[3,]
[1]  7  8  9

> x[,2]
[1]  2  5  8 11 14

> x[4:5, 2:3]
     [,1] [,2]
[1,]   11   12
[2,]   14   15
```

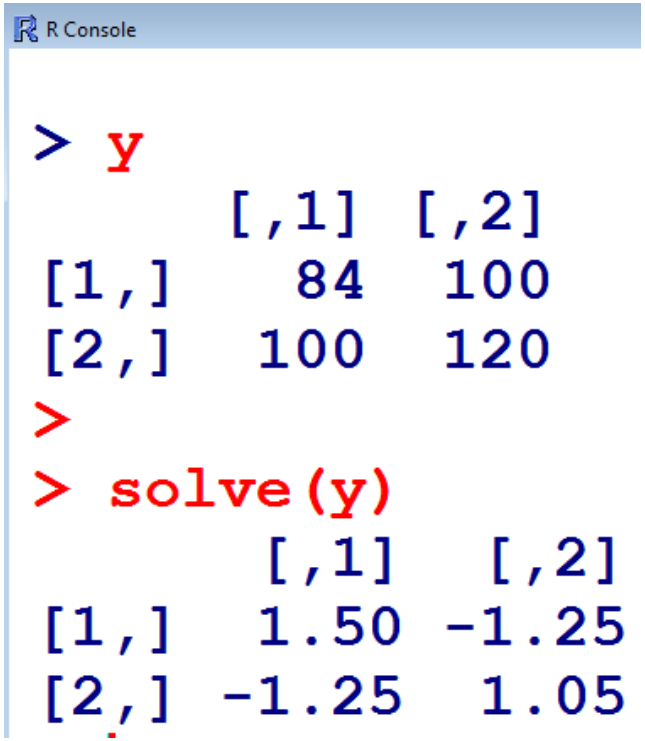


- **Inverse of a matrix:**

`solve()` finds the inverse of a positive definite matrix

Example:

```
> y<- matrix( nrow=2, ncol=2, byrow=T,
data=c(84,100,100,120))

> y
     [,1] [,2]
[1,]   84  100
[2,]  100  120

> solve(y)
       [,1]   [,2]
[1,]   1.50 -1.25
[2,] -1.25   1.05
```



- **Eigen Values and Eigen Vectors:**

`eigen()` **finds the eigen values and eigen vectors of a positive definite matrix**

Example:

```
> y
     [,1] [,2]
[1,]   84  100
[2,]  100  120

> eigen(y)
$values
[1] 203.6070864   0.3929136

$vectors
          [,1]       [,2]
[1,] 0.6414230 -0.7671874
[2,] 0.7671874  0.6414230
```