



The Statistical Language *R* - Matrices

This document is designed to introduce matrix calculations useful for statistics. It assumes some familiarity with *R*. Those meeting *R* for the first time should go through guides such as “*Using R in Windows XP on the ISS network*” and “*The R Language - Basics*”.

A. Variables - vectors and matrices

1. Vectors are the key variables within *R*. They are most easily constructed using the “combine” `c()` function; e.g. `x = c(1,2,4,5)` creates a vector `x` of length 4 with elements 1,2,4,5. Matrices are variables represented as 2-way arrays of numbers. They are most easily created from vectors using the `matrix()` function:

$$x = \text{matrix}(c(1,2,3,4,5,6), \text{nrow} = 3) \quad \text{yields} \quad \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

$$x = \text{matrix}(c(1,2,3,4,5,6), \text{nrow} = 3, \text{byrow} = \text{T}) \quad \text{yields} \quad \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}$$

In each case a vector of length 6 is turned into a 3×2 matrix. You can specify `nrow=3` or `ncol=2` or both; all three choices are equivalent. The vector fills up the matrix in successive columns unless “`byrow = T`” is specified, in which case the matrix is filled up by successive rows.

2. `rbind` and `cbind` are functions which can combine matrices columnwise or rowwise. If `x` and `y` contain the matrices

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix},$$

then `cbind(x,y)` and `rbind(x,y)` contain the matrices

$$\begin{bmatrix} 1 & 2 & 5 & 6 \\ 3 & 4 & 7 & 8 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \end{bmatrix},$$

respectively. If `x` and/or `y` is a vector, it is interpreted as a column in `cbind` and as a row in `rbind`.

3. Subsets. Elements or blocks of elements of vectors and matrices can be picked out using square brackets. If \mathbf{x} is a vector and \mathbf{y} is a matrix, then

<code>x[3]</code>	element 3 of \mathbf{x}
<code>x[1:3]</code>	a subvector of \mathbf{x} containing the first 3 elements
<code>y[5,6]</code>	the element of the matrix \mathbf{y} from row 5, column 6
<code>y[c(1,2), c(2,3)]</code>	a 2×2 submatrix of \mathbf{y} containing rows 1,2 and columns 2,3
<code>z = y[c(1,2), c(2,3)]</code>	save this submatrix as variable \mathbf{z}
<code>y[c(1,2), c(2,3)] = matrix(c(5, 9, 2.5, 3), nrow=2)</code>	

Assign new values to a submatrix of \mathbf{y}

B. Basic matrix calculations

- In general, operations on vectors and matrices act elementwise. If \mathbf{x} and \mathbf{y} are $n \times m$ matrices, then $\mathbf{x} + \mathbf{y}$ and $\mathbf{x} * \mathbf{y}$ are $n \times m$ matrices with elements $x[i, j] + y[i, j]$, and $x[i, j] * y[i, j]$, respectively. See below for “proper” matrix multiplication. If a is a scalar, $\mathbf{a} * \mathbf{x}$ has components $\mathbf{a} * x[i, j]$.
- Matrix addition. If \mathbf{x} and \mathbf{y} are $n \times m$ matrices, then the matrices $\mathbf{x} + \mathbf{y}$ and $\mathbf{x} - \mathbf{y}$ make sense and are evaluated elementwise. If \mathbf{a} is a scalar, $\mathbf{a} + \mathbf{x}$ and $\mathbf{a} * \mathbf{x}$ are $n \times m$ matrices with elements $\mathbf{a} + x[i, j]$ and $\mathbf{a} * x[i, j]$.
- Matrix multiplication. If \mathbf{x} is an $n \times m$ matrix and \mathbf{y} is $m \times p$, then $\mathbf{x} \%*\% \mathbf{y}$ represents the matrix product with elements $\sum_{j=1}^m x[i, j] * y[j, k]$. There are also some special cases involving vectors. In an expression involving a matrix and a vector, the vector is interpreted as either a row vector or a column vector, whichever makes the multiplication legitimate.
 - If \mathbf{x} is a vector of length m and \mathbf{y} is an $m \times p$ matrix, $\mathbf{x} \%*\% \mathbf{y}$ is a vector of length p .
 - If \mathbf{x} is an $n \times m$ matrix and \mathbf{y} is a vector of length m , $\mathbf{x} \%*\% \mathbf{y}$ is a vector of length n .

- If \mathbf{x} and \mathbf{y} are vectors of length m , `x%*%y` is a scalar (i.e. vector of length 1) representing the inner product $\sum_{i=1}^m \mathbf{x}[i]*\mathbf{y}[i]$.
4. Matrix inverse. If \mathbf{a} is a square non-singular $n \times n$ matrix, `solve(a)` is an $n \times n$ matrix giving the inverse of \mathbf{a} . If \mathbf{b} is a vector of length n , `solve(a,b)` gives the matrix product of the inverse of \mathbf{a} times \mathbf{b} .

Note. In *R*, `a^(-1)` or `1/a` is an $n \times n$ matrix with elements `1/a[i,j]` – the element-wise inverse of \mathbf{a} . This is very different from the matrix inverse of \mathbf{a} !

5. `t(a)` gives the transpose of a rectangular matrix \mathbf{a} .
6. Diagonals. The function `diag(a)` has 3 distinct interpretations
- If \mathbf{a} is a positive integer scalar (e.g. 5) then `diag(a)` is a 5×5 matrix with ones down the diagonal and zeros elsewhere.
 - If \mathbf{a} is a vector of length n , `diag(a)` is an $n \times n$ diagonal matrix with $(i, i)^{\text{th}}$ element `a[i]`.
 - If \mathbf{a} is an $n \times n$ matrix, `diag(a)` is a vector containing the diagonal elements `a[i,i]`, $i = 1, \dots, n$.

C. Plotting

If \mathbf{x} is an $n \times p$ data matrix, then

```
matplot(x, col=1)
```

plots each variable in an $n \times p$ data matrix vs 1:n. The i th variable in the plot is labelled i , $i = 1, \dots, p$. Remove the option `col=1` to see each variable in a different colour.

When the matrix represents image or spatial data then

```
image(x,col=gray((0:32)/32)) # draws a grayscale image
```

```
contour(x, add = TRUE, drawlabels = FALSE) # overlays a contour plot
```

D. The main structures in *R*

There are 3 main structures in *R*: *vectors*, *matrices* and *data frames*. Recall that data frames are typically obtained from external data files using the `read.table` command. Let `av`, `am`, `adf` be a vector, a matrix, and a data frame, respectively. Here are some commands to manipulate them.

1. If `a` is an object of uncertain or unknown structure, type `attributes(a)` to find out.
 - If `$dim` is defined and contains *one* number (the length), then `a` is a vector. The function `length(av)` gives the length of the vector `av`.
 - If `$dim` is defined and contains *two* numbers (`nrow` and `ncol`), then `a` is a matrix. The functions `nrow(am)`, `ncol(am)` give the numbers of rows and columns of the matrix `am`.
 - If `$class` is defined and contains the value `'data.frame'`, then `a` is a data frame.
2. `matrix(av)` or `matrix(av, ncol=1)` converts the vector `av` to a matrix with one column. Similarly, `matrix(av, nrow=1)` or the transpose `t(av)` converts `av` to a matrix with one row.
3. `as.data.frame(am)` converts a matrix to a data frame
4. `as.matrix(adf)` converts a data frame to a matrix.
5. If the matrix `am` has only one column or one row, then `as.vector(am)` turns it into a vector.

This document was produced by R.G. Aykroyd but was previously the second part of the notes *The Statistical Language R*, by J.T. Kent.