Matrices and matrix algebra

Introduction

In very simple terms, a matrix is simply a rectangular array of numbers, for example,

$$\begin{pmatrix} 4 & 2 & 10 \\ 2 & 4 & 1 \\ 2 & 8 & 3 \\ 5 & 2 & 11 \end{pmatrix}$$

The dimensions of a matrix are its number of rows and columns. An $r \times s$ matrix will have r rows and s columns. If $r = s$ then the matrix is a square matrix. Vectors can be viewed as a special case of matrices: A row vector is a $1 \times s$ matrix while a column vector is a $r \times 1$ matrix.

Matrices are used in mathematics to represent linear operators from one vector space to another vector space. In addition, matrices turn out to be very useful for representing data — for example, the columns of a matrix could represent different variables and the rows different individuals.

In statistics, these two representations turn out to be complementary, particularly in linear regression and multivariate analysis. For example, least squares estimates of regression parameters can be expressed very compactly in terms of matrix operations.

Addition and multiplication

Matrices can be added and multiplied provided their dimensions are appropriate for these operations. For example, in order for matrices to be added, they must the same number of rows and columns.

Let A and B be $s \times t$ matrices:

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1t} \\ \vdots & \ddots & \vdots \\ a_{s1} & \cdots & a_{st} \end{pmatrix} \quad B = \begin{pmatrix} b_{11} & \cdots & b_{1t} \\ \vdots & \ddots & \vdots \\ b_{s1} & \cdots & b_{st} \end{pmatrix}$$

Then $A + B$ is simply a $s \times t$ matrix whose components are the sums of the components of A and B:

$$A + B = \begin{pmatrix} a_{11} + b_{11} & \cdots & a_{1t} + b_{1t} \\ \vdots & \ddots & \vdots \\ a_{s1} + b_{s1} & \cdots & a_{st} + b_{st} \end{pmatrix}$$

Note that $A + B = B + A$.

Multiplication is somewhat more complicated. Let A be an $r \times s$ matrix and B an $s \times t$ matrix. Then AB will be an $r \times t$ matrix with

$$AB = \begin{pmatrix} c_{11} & \cdots & c_{1t} \\ \vdots & \ddots & \vdots \\ c_{r1} & \cdots & c_{rt} \end{pmatrix}$$

where

$$c_{ij} = \sum_{k=1}^{s} a_{ik} b_{kj}.$$

In general, $AB \neq BA$ even if AB has the same dimensions as BA.

Example. Suppose that

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \quad B = \begin{pmatrix} 4 & 3 \\ 2 & 1 \end{pmatrix}$$

Then

$$AB = \begin{pmatrix} 8 & 5 \\ 20 & 13 \end{pmatrix}$$

while

$$BA = \begin{pmatrix} 13 & 10 \\ 5 & 8 \end{pmatrix}.$$

Clearly, $AB \neq BA$.

Matrix multiplication is useful for expressing a system of linear equations.

Example. Suppose we want to solve the system of equations

$$\begin{aligned} 3x_1 + 4x_2 + x_3 &= 0 \\ x_1 + 3x_2 + 2x_3 &= 1 \\ 5x_1 + x_2 + 4x_3 &= 4 \end{aligned}$$

This can be written as the matrix equation

$$\begin{pmatrix} 3 & 4 & 1 \\ 1 & 3 & 2 \\ 5 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 4 \end{pmatrix}$$

In the next section, we will show how to solve the equations for x_1, x_2, x_3.

The identity matrix is a square matrix with 1's down the "main" diagonal and 0's elsewhere; for example,

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

is a 3×3 identity matrix. In general, an identity matrix is denoted by I.

Suppose that A is a $r \times r$ matrix and I is the $r \times r$ identity matrix matrix. Then it is easy to verify that $IA = AI = A$.

Inverses

In the previous section, we defined the identity matrix I to be a square matrix with 1's on the diagonal and 0's elsewhere. Suppose that A is a square matrix. Then the inverse of A, denoted by A^{-1}, is defined as the matrix which satisfies the equation

$$A^{-1}A = I$$

provided that such a matrix exists. If A^{-1} does not exist then A is said to be singular.

Computing A^{-1} (if it exists) by hand is quite time consuming (except for 2×2 matrices) but can be done easily using a computer.

Example. Suppose that

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}.$$

In this case, A^{-1} does not exist.

Example. Suppose that

$$A = \begin{pmatrix} 2 & 1 & 2 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Then the inverse of A is

$$A^{-1} = \begin{pmatrix} 1/3 & -2/3 & 1/3 \\ -1/3 & 2/3 & 2/3 \\ 1/3 & 1/3 & -2/3 \end{pmatrix}$$

It is easy to verify that $A^{-1}A = I$ and $AA^{-1} = I$.

Example. Suppose we have the matrix equation

$$\begin{pmatrix} 3 & 4 & 1 \\ 1 & 3 & 2 \\ 5 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 4 \end{pmatrix}$$

Then we can solve for x_1, x_2, x_3 by finding the inverse of the 3×3 matrix. We have

$$\begin{aligned} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} &= \begin{pmatrix} 3 & 4 & 1 \\ 1 & 3 & 2 \\ 5 & 1 & 1 \end{pmatrix}^{-1} \begin{pmatrix} 0 \\ 1 \\ 4 \end{pmatrix} \\ &= \begin{pmatrix} 0.04 & -0.12 & 0.20 \\ 0.36 & -0.08 & 0.20 \\ -0.56 & 0.68 & 0.20 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 4 \end{pmatrix} \\ &= \begin{pmatrix} 0.68 \\ -0.88 \\ 1.48 \end{pmatrix} \end{aligned}$$

Hence $x_1 = 0.68$, $x_2 = -0.88$ and $x_3 = 1.48$.

There is a simple formula for computing the inverse of a 2×2 matrix. Suppose that

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

A is invertible provided that $a_{11}a_{22} - a_{12}a_{21} \neq 0$; in this case,

$$A^{-1} = \frac{1}{a_{11}a_{22} - a_{12}a_{21}} \begin{pmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{pmatrix}$$

Unfortunately, no similarly simple formula exists for higher dimensional matrices.

Example. Suppose that

$$A = \begin{pmatrix} 8 & 5 \\ 20 & 13 \end{pmatrix}$$

Then $a_{11}a_{22} - a_{12}a_{21} = 8 \times 13 - 20 \times 5 = 4$. Thus

$$A^{-1} = \frac{1}{4} \begin{pmatrix} 13 & -5 \\ -20 & 8 \end{pmatrix} = \begin{pmatrix} 3.25 & -1.25 \\ -5 & 2 \end{pmatrix}.$$

It is easy to verify that $AA^{-1} = I$.

Matrix algebra in R

R has a number of built-in functions and operators for matrix algebra, allowing us to, for example, add and multiply matrices, solve matrix equations, and compute eigenvalues and eigenvectors.

We can define matrices using the `matrix` function. For example,

```
> x <- c(1:20)
> y <- matrix(x,ncol=4,byrow=T)
```

takes the vector `x` and creates a matrix `y` consisting of 4 columns (`ncol=4`) and 5 rows. The option `byrow=T` indicates that the data from `x` will be put into `y` row-by-row as opposed to column-by-column (`byrow=F`).

Matrices can be added using the `+` operator and multiplied using the `%*%` operator. Note that the `*` operator multiplies element-by-element and does **not** perform matrix multiplication; for example,

```
> a <- matrix(c(1,2,3,4),ncol=2,byrow=T)
> b <- matrix(c(4,3,2,1),ncol=2,byrow=T)
```

```
> a%*%b
     [,1] [,2]
[1,]    8    5
[2,]   20   13
> a*b
     [,1] [,2]
[1,]    4    6
[2,]    6    4
```

The `t` function computes the transpose of a matrix. For example,

```
> a <- c(0,2,4,2,6,3,1,3)
> b <- c(1,4,2,4,2,9,3,1,2,1,7,4)
> a <- matrix(a,ncol=4,byrow=T)
> b <- matrix(b,ncol=3,byrow=T)
> a
     [,1] [,2] [,3] [,4]
[1,]    0    2    4    2
[2,]    6    3    1    3
> b
     [,1] [,2] [,3]
[1,]    1    4    2
[2,]    4    2    9
[3,]    3    1    2
[4,]    1    7    4
> a%*%b
     [,1] [,2] [,3]
[1,]   22   22   34
[2,]   24   52   53
> t(a%*%b)
     [,1] [,2]
[1,]   22   24
[2,]   22   52
[3,]   34   53
```

The inverse of a matrix can be computed using the `solve` function. Let `a` and `b` be as defined above. Then

```
> solve(a%*%t(a))
            [,1]        [,2]
[1,]  0.05169173 -0.01503759
```

```
[2,] -0.01503759  0.02255639
> solve(t(b)%*%b)
            [,1]        [,2]        [,3]
[1,]  0.21122807  0.01894737 -0.10666667
[2,]  0.01894737  0.02661654 -0.02285714
[3,] -0.10666667 -0.02285714  0.07047619
```

If a matrix is not invertible, you will be given an error message to this effect. (For example, `solve(t(a)%*%a)` returns the error message which indicates that the matrix to be inverted is singular.)

More generally, the function `solve` can be used to solve the matrix equation $Ax = b$ for x where A is an $s \times s$ matrix with x and b columns vectors of length s. For example, suppose we want to solve

$$\begin{pmatrix} 2 & 1 & 2 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 1 \end{pmatrix}$$

for x_1, x_2, x_3. In R, this becomes

```
> a <- matrix(c(2,1,2,0,1,1,1,1,0),ncol=3,byrow=T)
> b <- c(1,2,1)
> x <- solve(a,b)
> x
[1] -0.6666667  1.6666667  0.3333333
```

Thus $x_1 = -2/3$, $x_2 = 5/3$ and $x_3 = 1/3$.

It should be noted that numerical computation of matrix inverses is sometimes "unstable" and should be avoided if possible. In fact, computational matrix algebra is surprisingly tricky due to the fact that computers do not represent numbers exactly but rather in "floating point" form; therefore, computational pitfalls (such as accumulation of round-off error) can pose real problems. Fortunately, the algorithms used by most statistical software packages avoid these pitfalls.

Covariance and correlation matrices

Let $X_1, \cdots, X_p$ be random variables and define $\boldsymbol{X} = (X_1, \cdots, X_p)^T$ be a random vector. The covariance (or variance-covariance) matrix of $\boldsymbol{X}$ is defined to be

$$C = \begin{pmatrix} \mathrm{Var}(X_1) & \mathrm{Cov}(X_1, X_2) & \cdots & \mathrm{Cov}(X_1, X_p) \\ \mathrm{Cov}(X_1, X_2) & \mathrm{Var}(X_2) & \cdots & \mathrm{Cov}(X_2, X_p) \\ \vdots & \vdots & \ddots & \vdots \\ \mathrm{Cov}(X_1, X_p) & \mathrm{Cov}(X_2, X_p) & \cdots & \mathrm{Var}(X_p) \end{pmatrix}$$

A special case of the covariance matrix is the correlation matrix of $\boldsymbol{X}$ which is the covariance matrix of the "standardized" random vector $\boldsymbol{X'} = (X'_1, \cdots, X'_p)^T$ where

$$X'_k = \frac{X_k - E(X_k)}{\sqrt{\text{Var}(X_k)}}.$$

The correlation matrix of $\boldsymbol{X}$ is

$$R = \begin{pmatrix} 1 & \text{Corr}(X_1, X_2 & \cdots & \text{Corr}(X_1, X_p) \\ \text{Corr}(X_1, X_2) & 1 & \cdots & \text{Corr}(X_2, X_p) \\ \vdots & \vdots & \ddots & \vdots \\ \text{Corr}(X_1, X_p) & \text{Corr}(X_2, X_p) & \cdots & 1 \end{pmatrix}$$

where $\text{Corr}(X_i, X_j) = \text{Cov}(X'_i, X'_j)$ is the correlation between X_i and X_j. Note that the covariance and correlation matrices, C and R, are both symmetric matrices.

(Typically, the covariance (or correlation matrix) is unknown and must be estimated from the multivariate observations $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n$. From here on, we will C to be either the exact covariance matrix or an estimate of it.)

The covariance matrix C can be used, for example, to compute the variance of $a_1X_1 + a_2X_2 + \cdots + a_pX_p$ for some constants $a_1, \cdots, a_p$; if we define the vector $\boldsymbol{a} = (a_1, \cdots, a_p)$, we have

$$\text{Var}(\boldsymbol{a}^T\boldsymbol{X}) = \text{Var}(a_1X_1 + a_2X_2 + \cdots + a_pX_p) = \boldsymbol{a}^TC\boldsymbol{a}.$$

From a geometrical perspective, the random variable $\boldsymbol{a}^T\boldsymbol{X}$ is actually a one dimensional projection of the random vector $\boldsymbol{X}$ in the direction $\boldsymbol{a}$. Thus we can think of $\text{Var}(\boldsymbol{a}^T\boldsymbol{X})$ as the variance of $\boldsymbol{X}$ in the direction $\boldsymbol{a}$.

The covariance matrix C is also a non-negative definite matrix: For each non-zero vector $\boldsymbol{a}$, $\boldsymbol{a}^TC\boldsymbol{a} \geq 0$ (which follows from the fact that $\boldsymbol{a}^TC\boldsymbol{a} = \text{Var}(\boldsymbol{a}^T\boldsymbol{X}) \geq 0$). If $\boldsymbol{a}^TC\boldsymbol{a} > 0$ for each non-zero $\boldsymbol{a}$ then C is a positive definite matrix. The correlation matrix R is also non-negative definite and positive definite if C is positive definite.

In a variety of problems in multivariate statistics, it is of interest to know in which direction $\boldsymbol{X}$ has the greatest (or least) variance. To be more precise, consider all vectors $\boldsymbol{a}$ such that $\boldsymbol{a}^T\boldsymbol{a} = 1$; then we would like to find the vector which either maximizes or minimizes $\text{Var}(\boldsymbol{a}^T\boldsymbol{X}) = \boldsymbol{a}^TC\boldsymbol{a}$ since these $\{\boldsymbol{a}\}$ can be quite informative about the structure of the multivariate data. These vectors $\boldsymbol{a}$ turn out to be **eigenvectors** of the covariance matrix C.

Eigenvalues and eigenvectors

Let A be any $p \times p$ matrix. Then λ and $\boldsymbol{v} \neq \boldsymbol{0}$ are called (respectively) an eigenvalue and eigenvector of A if

$$A\boldsymbol{v} = \lambda\boldsymbol{v}.$$

Note that for a given eigenvalue λ, $\boldsymbol{v}$ is not uniquely determined; that is, if $\boldsymbol{v}$ is an eigenvector then $a\boldsymbol{v}$ is also an eigenvector for any $a \neq 0$. The eigenvalues of A are the solutions of the equation

$$\det(A - \lambda I) = 0,$$

which leads to an equation of the form

$$b_0 + b_1\lambda + b_2\lambda^2 + \cdots + b_p\lambda^p = 0$$

and so there are at most p distinct eigenvalues. In fact, if we allow for multiple eigenvalues, we have exactly p eigenvalues (although not necessarily p distinct eigenvalues). Eigenvalues are also not necessarily real numbers; the eigenvalues (and eigenvectors) of a matrix can be complex-valued. However, if A is a symmetric matrix then the eigenvalues are real-valued.

Suppose that C is a covariance matrix of a random vector $\boldsymbol{X}$. Then all the all the eigenvalues of C are non-negative. Furthermore, if $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_p$ are the eigenvectors of C then we have

$$\boldsymbol{v}_i^T \boldsymbol{v}_j = 0 \quad \text{for } i \neq j$$

(We say that the eigenvectors are **orthogonal**.) If $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$ are the eigenvalues of $C = \text{Cov}(\boldsymbol{X})$ then

$$\lambda_1 + \lambda_2 + \cdots + \lambda_p = \text{Var}(X_1) + \text{Var}(X_2) + \cdots + \text{Var}(X_p).$$

We want to find a vector $\boldsymbol{a}$ with $\boldsymbol{a}^T\boldsymbol{a} = 1$ which either maximizes or minimizes $\text{Var}(\boldsymbol{a}^T\boldsymbol{X}) = \boldsymbol{a}^T C \boldsymbol{a}$. Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_1$ be the eigenvalues of C. Then

$$\begin{aligned} \max_{a^T a = 1} \boldsymbol{a}^T C \boldsymbol{a} &= \lambda_1 \\ \min_{a^T a = 1} \boldsymbol{a}^T C \boldsymbol{a} &= \lambda_p \end{aligned}$$

and

$$\begin{aligned} \boldsymbol{a}_1^T C \boldsymbol{a}_1 &= \lambda_1 \\ \boldsymbol{a}_p^T C \boldsymbol{a}_p &= \lambda_p \end{aligned}$$

where $\boldsymbol{a}_1$ and $\boldsymbol{a}_p$ are the eigenvectors of C corresponding to the eigenvalues λ_1 and λ_p respectively (where $\boldsymbol{a}_1^T\boldsymbol{a}_1 = \boldsymbol{a}_p^T\boldsymbol{a}_p = 1$).

The eigenvectors $\boldsymbol{a}_1, \cdots, \boldsymbol{a}_p$ (corresponding to $\lambda_1, \cdots, \lambda_p$) also play a special role. If we define $Y_k = \boldsymbol{a}_k^T\boldsymbol{X}$ for $k = 1, \cdots, p$ then $Y_1, \cdots, Y_p$ are called the **principal components** of $\boldsymbol{X}$. The principal components are uncorrelated random variables, that is, $\text{Cov}(Y_i, Y_j) = 0$ for $i \neq j$. In many situations, the first few principal components explain close to 100% of the overall variability of $\boldsymbol{X}$ and thus can give a very good summary of the structure of the data. (With real data, we usually compute principal components using the correlation matrix of the data unless there is a compelling reason to do otherwise; in any event, it is usually a good idea to compute principal components on "dimensionless" data, that is, where each variable has no units of measurement. For example, variables that are positive can be made dimensionless by taking their logarithms.)

Computing eigenvalues and principal components in R

The function `eigen` can be used to compute eigenvalues and eigenvectors in R. For example, if `a` is a square matrix then

```
> evals <- eigen(a)
```

will compute the eigenvalues and eigenvectors of `a`. The output of `eigen` (in this case, `evals`) consists of two components containing the eigenvalues and eigenvectors:

```
> evals$vectors
           [,1]       [,2]        [,3]
[1,]  0.4726920  0.2872900  0.83308270
[2,] -0.6800200 -0.4823537  0.55218451
[3,]  0.5604776 -0.8275261 -0.03264179
> evals$values
[1] 1.0995174 0.8336224 0.7315822
```

The columns of `evals$vectors` are the eigenvectors.

In the case where the matrix is symmetric (for example, covariance and correlation matrices), we can facilitate the computation by telling `eigen` that we have a symmetric matrix. For example, suppose we generate 3 correlated vectors of normal random variables as follows:

```
> x1 <- rnorm(100)
> x2 <- rnorm(100)
> x3 <- rnorm(100)
> x2 <- x1 + x2
> x3 <- x2 + x3
> y <- cbind(x1,x2,x3)
```

The `var` function can be used to compute the estimate of the covariance matrix of the vectors `x1`, `x2`, and `x3`, which make up the matrix `y`.

```
> covar <- var(y) # compute estimated covariance matrix of (x1,x2,x3)
> covar
         x1       x2       x3
x1 1.009215 1.003731 1.061714
x2 1.003731 1.940069 2.052687
x3 1.061714 2.052687 2.938241
> corr <- cor(y) # compute correlation matrix
> corr
          x1        x2        x3
x1 1.0000000 0.7173265 0.6165555
```

```
x2 0.7173265 1.0000001 0.8597463
x3 0.6165555 0.8597463 1.0000000
> r1 <- eigen(covar,symmetric=T) # specify covar to be symmetric
> r1$values # eigenvalues of covariance matrix
[1] 5.0730602 0.5563905 0.2580739
> r2 <- eigen(corr,symmetric=T)
> r2$values # eigenvalues of correlation matrix
[1] 2.4670504 0.4067742 0.1261753
> sum(c(var(x1),var(x2),var(x3)))
[1] 5.887525
> sum(r1$values)
[1] 5.887525
> # sum of variances = sum of eigenvalues of covariance matrix
> sum(r2$values)
[1] 3
> # sum of eigenvalues of correlation matrix = number of variables
```

Principal components can be computed using either the `princomp` or `prcomp` functions. (We will use `princomp` here.) In the following example, we have three data vectors (variables) in `y1`, `y2` and `y3`. We will first compute the correlation matrix of the data and then the principal components (using the correlation matrix).

```
> cor(cbind(y1,y2,y3))
           y1         y2        y3
y1  1.0000000 -0.4848768 0.2905132
y2 -0.4848768  1.0000000 0.2197556
y3  0.2905132  0.2197556 1.0000000
> r <- princomp(~y1 + y2 + y3, cor=T) # cor=T indicates we are using the
> # correlation matrix; alternatively we could compute the principal
> # components using the command
> # r <- princomp(cbind(y1,y2,y3), cor=T)
> names(r) # names of the different components of r
 [1] "sdev"   "loadings"     "correlations" "scores"  "center"
 [6] "scale"  "n.obs"        "terms"        "call"    "factor.sdev"
[11] "coef"
> r$loadings # loadings of principal components
   Comp. 1 Comp. 2 Comp. 3
y1  0.735  -0.228  -0.638
y2 -0.663  -0.434  -0.609
y3  0.138  -0.871   0.471
> plot(r$scores[,1],r$scores[,2]) # plot first two principal components
```

```
> r$sdev^2 # variances of principal components
  Comp. 1  Comp. 2   Comp. 3
 1.491858 1.185757 0.3223852
```

We can also check that we're doing the right thing by comparing the output of `princomp` to the eigenvalues and eigenvectors of the correlation matrix:

```
> eigen(cor(cbind(y1,y2,y3)))
$values:
[1] 1.4918582 1.1857566 0.3223851
$vectors:
           [,1]       [,2]       [,3]
[1,]  0.7354627 -0.2284037 -0.6379078
[2,] -0.6633597 -0.4344956 -0.6092352
[3,]  0.1380166 -0.8712321  0.4710691
```

Note that the "loadings" (component `r$loadings`) are the same as the eigenvectors of the correlation matrix and the eigenvalues of the correlation matrix are the same as the squares of `r$sdev`.

The Cholesky decomposition

If a symmetric matrix A is positive definite (that is, $\boldsymbol{a}^T A\boldsymbol{a} > 0$ for all non-zero $\boldsymbol{a}$) then A can be factored as

$$A = LL^T$$

where L is a lower triangular matrix:

$$L = \begin{pmatrix} \ell_{11} & 0 & 0 & \cdots & 0 \\ \ell_{21} & \ell_{22} & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \cdots & \vdots \\ \ell_{p1} & \ell_{p2} & \ell_{p3} & \cdots & \ell_{pp} \end{pmatrix}.$$

To make the decomposition (called the Cholesky decomposition) unique, we require that the diagonal elements $\ell_{11}, \cdots, \ell_{pp}$ be positive. If A is non-negative definite, we can also write $A = LL^T$ where L is lower triangular although L need no longer be unique.

The Cholesky decomposition gives us a convenient way to define a "square root" of a positive definite matrix. For example, the Cholesky decomposition can be used to generate observations from a multivariate normal distribution with positive definite covariance matrix C: If $\boldsymbol{X}$ is multivariate normal with covariance matrix I (that is, the elements of $\boldsymbol{X}$ are independent $\mathcal{N}(0,1)$ random variables) and $C = LL^T$ then

$$\boldsymbol{Y} = \boldsymbol{\mu} + L\boldsymbol{X}$$

has a multivariate normal distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix C. The following R code illustrates how to do this; note that the R function `chol` (which computes the Cholesky decomposition) returns L^T (an upper triangular matrix) rather than L.

```
> C <- matrix(c(1,0.7,0.7,0.7,0.7,1,0.7,0.7,0.7,0.7,1,0.7,0.7,0.7,0.7,1),
+ ncol=4)
> C
     [,1] [,2] [,3] [,4]
[1,]  1.0  0.7  0.7  0.7
[2,]  0.7  1.0  0.7  0.7
[3,]  0.7  0.7  1.0  0.7
[4,]  0.7  0.7  0.7  1.0
> L <- chol(C)
> L
     [,1]      [,2]      [,3]      [,4]
[1,]    1 0.7000000 0.7000000 0.7000000
[2,]    0 0.7141428 0.2940588 0.2940588
[3,]    0 0.0000000 0.6507914 0.1898142
[4,]    0 0.0000000 0.0000000 0.6224950
> # Now generate 100 observations from a 4-variate normal with mean 0
> # and covariance matrix C
> y <- NULL
> for (i in 1:100) {
+   x <- rnorm(4) # generate 4 independent N(0,1) random variables
+   y <- rbind(y,as.vector(t(L)%*%x)) # each row of y is an observation
+ }
> var(y)  # estimate of covariance matrix
          [,1]      [,2]      [,3]      [,4]
[1,] 1.1237560 0.7690821 0.7280457 0.7360255
[2,] 0.7690821 0.9723985 0.7704144 0.6986990
[3,] 0.7280457 0.7704144 1.0691694 0.6981735
[4,] 0.7360255 0.6986990 0.6981735 0.9871060
```