Appendix Part I

Vectors and Matrices

In this first part of the appendix, we provide a brief survey, "sketchy" overview of essential matrix and linear algebra that is useful for understanding parts of the book at a more technical level, what some would also call a "deeper" level. We proceed quickly in demonstrating these operations using R software. Once the student overcomes the initial learning curve (it can be quite steep at times) for performing such operations in R, she will experience a tremendous wealth of computing power at her hands, unlike what could ever be experienced using packages such as SPSS. Working with R, the user becomes the **body of knowledge**, the "stimuli," for which the computer is the response. The user can input even a single request and immediately receive a response. This is a very powerful way to learn univariate and multivariate analysis along with its numerous matrix operations and computations.

The contents of this appendix are by no means designated to be formal, thorough, or even "complete" in any sense whatsoever. The area of **linear algebra** is one of the pillars of modern mathematics and has a life of its own. Our overview is merely a shorthand collection of notes on select topics in matrix operations useful for understanding statistical methods in their broadest generality. There exist a plethora of excellent texts on the mathematics of matrix algebra, and the reader is strongly encouraged to consult such texts to accompany his or her study of this book and multivariate analysis more generally if a more technical (again, some would say "deeper") understanding of statistical methods is desired. One of the most readable sources is that of Searle (1982). Gill (2006) also contains chapters devoted to essentials of linear algebra and matrices. Carroll and Green (1997) provide a good overview of the mathematics of multivariate analysis. Strang (1993), Anton and Rorres (2000), and Harville (1997) are matrix texts for the mathematical sciences, the latter on the technical side. Roman (2008) is an advanced text, but also features a review of much of the "basics" of the area.

A.1. Matrices

A **matrix** is simply an ordered array of numbers, square or rectangular, denoted by n rows and m columns. For instance, the following matrix $\mathbf{A}$ is a 3-row by 2-column matrix (i.e., it is of **order** 3 x 2):

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix}$$

The contents of the matrix consist of elements a_{11}, a_{12}, a_{21}, and so on. A matrix is **square** if the number of n rows equals the number of m columns. For example, the matrix $\mathbf{B}$ is a square matrix:

$$\mathbf{B} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

A.2. Building Matrices in R

We can generate a matrix in R in many different ways. As an example, suppose we wished to produce a matrix containing a sequence of numbers from 1 to 8 and having four rows and two columns. We say that the **dimension** of the matrix is 4 x 2. We can use the function `matrix` to create the matrix. Let us call this matrix **S**:

```
> S <- matrix(1:8, 4, 2)
> S
     [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    7
[4,]    4    8
```

where `S` is the name we are giving to our matrix, `1:8` designates the matrix to have entries 1-8, and `4,2` designates the matrix to have four rows and two columns.

We could expand on the above code and generate a wide variety of matrices of various dimensions. We could also change the contents of the matrix. For instance, suppose that instead of inputting entries from 1 to 8, we chose instead to input entries from 20 to 27. We would have:

```
> S <- matrix(20:27, 4, 2)
> S
     [,1] [,2]
[1,]   20   24
[2,]   21   25
[3,]   22   26
[4,]   23   27
```

Suppose we now wanted to add two new elements to the matrix, elements 28 and 29. To accommodate these new elements, we need to expand on the number of rows in the matrix. Instead of having four rows, we now have five rows:

```
> S <- matrix(20:29, 5, 2)
> S
     [,1] [,2]
[1,]   20   25
[2,]   21   26
[3,]   22   27
[4,]   23   28
[5,]   24   29
```

where `(20:29, 5, 2)` requests to build a 5 x 2 matrix with entries 20 to 29. Notice that if we specified a range of entries the dimension of the matrix could not handle neatly, R will still compute a matrix, but will either shrink or expand the entries smaller or greater than the range

20-29 as needed. For instance, suppose we again requested entries 20-29, but instead a 2 x 2 matrix:

```
> S <- matrix(20:29, 2, 2)
> S
     [,1] [,2]
[1,]   20   22
[2,]   21   23
```

Notice that because we asked for a 2 x 2 matrix, R simply generated the first four entries in the range of 20-29. On the other hand, if we requested a "too large" of a matrix given the range, R will simply start repeating values. For example, suppose we specified the same range of 20-29, but asked for a 10 x 5 matrix:

```
> S <- matrix(20:29, 10, 5)
> S
      [,1] [,2] [,3] [,4] [,5]
 [1,]   20   20   20   20   20
 [2,]   21   21   21   21   21
 [3,]   22   22   22   22   22
 [4,]   23   23   23   23   23
 [5,]   24   24   24   24   24
 [6,]   25   25   25   25   25
 [7,]   26   26   26   26   26
 [8,]   27   27   27   27   27
 [9,]   28   28   28   28   28
[10,]   29   29   29   29   29
```

Notice that R accommodated the 10 x 5 matrix by replicating values.

A.3. Dimension of a Matrix

Recall the dimension of a matrix is defined by the number of rows and columns in the matrix. For instance, consider the matrix **S** just computed. Matrix **S** has 10 rows and 5 columns. We can easily verify the dimension of a matrix by requesting `dim`:

```
> dim(S)
[1] 10 5
```

where "10" is the number of rows and "5" is the number of columns. Two matrices are considered **equal** only if they are of the same dimension and have the same elements in the same places (i.e., the corresponding elements must be the same). For example, matrices $\mathbf{A}$ and $\mathbf{B}$ are equal, whereas each is unequal to matrix $\mathbf{C}$:

$$\mathbf{A} = \begin{bmatrix} 2 & 7 \\ 4 & 9 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 2 & 7 \\ 4 & 9 \end{bmatrix} \quad \mathbf{C} = \begin{bmatrix} 4 & 9 \\ 2 & 7 \end{bmatrix}$$

A.4. Constructing a Covariance or Correlation Matrix

Though there are many ways to build matrices in R, we demonstrate a simple way of producing a matrix that is used in the book. Consider the following correlation matrix (lower triangular shown only):

```
1.00000
.343      1.00000
.505       .203   1.00000
.308       .400    .398  1.00000
.693       .187    .303   .205   1.00000
.208       .108    .277   .487    .200  1.00000
.400       .386    .286   .385    .311   .432  1.00000
.455       .385    .167   .465    .485   .310   .365 1.00000
```

We could generate the above matrix as a text file and read it directly from that file using a function such as `read.table`, but we could also quite easily build the matrix in R. We build the matrix by first "concatenating" the row vectors. By defining each row as `r1`, `r2`, `r3`, etc., and by specifying `c` to concatenate each vector, we build the vectors for the matrix:

```
> r1 <- c(1.000, 0.343, 0.505, 0.308, 0.693, 0.208, 0.400, 0.455)
> r2 <- c(0.343, 1.000, 0.203, 0.400, 0.187, 0.108, 0.386, 0.385)
> r3 <- c(0.505, 0.203, 1.000, 0.398, 0.303, 0.277, 0.286, 0.167)
> r4 <- c(0.308, 0.400, 0.398, 1.000, 0.205, 0.487, 0.385, 0.465)
> r5 <- c(0.693, 0.187, 0.303, 0.205, 1.000, 0.200, 0.311, 0.485)
> r6 <- c(0.208, 0.108, 0.277, 0.487, 0.200, 1.000, 0.432, 0.310)
> r7 <- c(0.400, 0.386, 0.286, 0.385, 0.311, 0.432, 1.000, 0.365)
> r8 <- c(0.455, 0.385, 0.167, 0.465, 0.485, 0.310, 0.365, 1.000)
```

Recall that since in a correlation or covariance matrix the lower triangular of the matrix is identical to the upper triangular, it was a simple matter to insert the correct off-diagonal correlations in producing the above vectors (i.e., notice that the upper triangular part of the matrix mirrors that of the lower). We then use the `rbind` function (i.e., "bind the rows together") to join the rows 1, 2, 3, etc. of the above vectors, identifying the new matrix as `cormatrix`:

```
> cormatrix <- rbind(r1, r2, r3, r4, r5, r6, r7, r8)
```

To verify that we built the matrix correctly, we request it by its name:

```
> cormatrix
    [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]
c1 1.000 0.343 0.505 0.308 0.693 0.208 0.400 0.455
c2 0.343 1.000 0.203 0.400 0.187 0.108 0.386 0.385
c3 0.505 0.203 1.000 0.398 0.303 0.277 0.286 0.167
c4 0.308 0.400 0.398 1.000 0.205 0.487 0.385 0.465
c5 0.693 0.187 0.303 0.205 1.000 0.200 0.311 0.485
c6 0.208 0.108 0.277 0.487 0.200 1.000 0.432 0.310
c7 0.400 0.386 0.286 0.385 0.311 0.432 1.000 0.365
c8 0.455 0.385 0.167 0.465 0.485 0.310 0.365 1.000
```

We can see that the generated `cormatrix` was correctly built, and is now ready for analyses that require a correlation matrix as input. `Cormatrix` is featured in Chapters 13 and 14.

A.5. Building the Matrix by Concatenating Columns Instead of Rows

We could have just as easily built `cormatrix` by concatenating columns instead of rows. To proceed this way, we build each vector by their respective columns. To make things easier, we name each column vector 1, 2, 3, etc., by c1, c2, c3, and so on for columns 1, 2, 3 . . . instead of r1, r2, r3, which recall was used to designate rows:

```
> c1 <- c(1.000, 0.343, 0.505, 0.308, 0.693, 0.208, 0.400, 0.455)
> c2 <- c(0.343, 1.000, 0.203, 0.400, 0.187, 0.108, 0.386, 0.385)
> c3 <- c(0.505, 0.203, 1.000, 0.398, 0.303, 0.277, 0.286, 0.167)
> c4 <- c(0.308, 0.400, 0.398, 1.000, 0.205, 0.487, 0.385, 0.465)
> c5 <- c(0.693, 0.187, 0.303, 0.205, 1.000, 0.200, 0.311, 0.485)
> c6 <- c(0.208, 0.108, 0.277, 0.487, 0.200, 1.000, 0.432, 0.310)
> c7 <- c(0.400, 0.386, 0.286, 0.385, 0.311, 0.432, 1.000, 0.365)
> c8 <- c(0.455, 0.385, 0.167, 0.465, 0.485, 0.310, 0.365, 1.000)
```

We then use the `cbind` function instead of the `rbind` function to join the column vectors, assigning the matrix once again the name `cormatrix`:

```
> cormatrix <- cbind(c1, c2, c3, c4, c5, c6, c7, c8)
> cormatrix
       c1    c2    c3    c4    c5    c6    c7    c8
[1,] 1.000 0.343 0.505 0.308 0.693 0.208 0.400 0.455
[2,] 0.343 1.000 0.203 0.400 0.187 0.108 0.386 0.385
[3,] 0.505 0.203 1.000 0.398 0.303 0.277 0.286 0.167
[4,] 0.308 0.400 0.398 1.000 0.205 0.487 0.385 0.465
[5,] 0.693 0.187 0.303 0.205 1.000 0.200 0.311 0.485
[6,] 0.208 0.108 0.277 0.487 0.200 1.000 0.432 0.310
[7,] 0.400 0.386 0.286 0.385 0.311 0.432 1.000 0.365
[8,] 0.455 0.385 0.167 0.465 0.485 0.310 0.365 1.000
```

A covariance or correlation matrix is a **symmetric** matrix, meaning that the upper and lower triangulars are mirror images of each other. Generating it by concatenating rows or columns results in the same matrix.

A.6. Operations on Matrices

Matrix addition and subtraction are defined only for matrices of the same dimension. For example, we can add matrices $\mathbf{A}$ and $\mathbf{B}$:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{32} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{32} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$

such that

$$\mathbf{A} + \mathbf{B} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} a_{11}+b_{11} & a_{12}+b_{12} & a_{13}+b_{13} \\ a_{21}+b_{21} & a_{22}+b_{22} & a_{23}+b_{23} \\ a_{31}+b_{31} & a_{32}+b_{32} & a_{33}+b_{33} \end{bmatrix}$$

Addition of matrices **A** and **C**, however, is undefined since **A** and **C** are not of the same dimension:

$$\mathbf{A}+\mathbf{C} \neq \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} + \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \\ c_{31} & c_{32} \end{bmatrix}$$

To demonstrate in R, let matrix $\mathbf{A} = \begin{bmatrix} 5 & 9 \\ 3 & 4 \end{bmatrix}$ and matrix $\mathbf{B} = \begin{bmatrix} 1 & 3 \\ 7 & 2 \end{bmatrix}$. $\mathbf{A}+\mathbf{B}$ is therefore:

```
> A + B
     a1 a2
[1,]  6 12
[2,] 10  6
```

The following properties hold for the addition of matrices. Many of these properties you might also recognize from the general properties of real numbers.

Matrix addition is **commutative**, which means:

$$\mathbf{A}+\mathbf{B}=\mathbf{B}+\mathbf{A}$$

That is, whether you add matrix $\mathbf{B}$ to matrix $\mathbf{A}$ or matrix $\mathbf{A}$ to matrix $\mathbf{B}$, the same answer will be obtained.

Matrix addition is **associative**, which means:

$$\mathbf{A}+(\mathbf{B}+\mathbf{C}) = (\mathbf{A}+\mathbf{B})+\mathbf{C}$$

That is, whether you add $\mathbf{A}$ to the sum of $\mathbf{B}+\mathbf{C}$ or add the sum of $\mathbf{A}+\mathbf{B}$ to $\mathbf{C}$, you will obtain the same answer. Matrix multiplication also obeys the associative law, which states:

$$(\mathbf{AB})\mathbf{C} = \mathbf{A}(\mathbf{BC})$$

When multiplying matrices $\mathbf{A}$ and $\mathbf{C}$, the product $\mathbf{AC}$ is defined only for matrices for which the **number of columns in $\mathbf{A}$ is equal to the number of rows in $\mathbf{C}$**. When the number of columns in the first matrix is equal to the number of rows in the second matrix, we say the matrices are **conformable for multiplication**. For example, let matrices $\mathbf{A}$ and $\mathbf{C}$ be defined as:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \quad \mathbf{C} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \\ c_{31} & c_{32} \end{bmatrix}.$$

Notice that the number of columns in $\mathbf{A}$ is equal to the number of rows in $\mathbf{C}$. That is, there are three columns in $\mathbf{A}$ and three rows in $\mathbf{C}$. They are conformable for multiplication so long as we wish to generate the product $\mathbf{AC}$. Notice that the product $\mathbf{CA}$ is not defined, since the number of columns in $\mathbf{C}$ (equal to 2) is not equal to the number of rows in $\mathbf{A}$ (equal to 3).

To get the product $\mathbf{AC}$, we carry on with multiplying each element in respective rows of $\mathbf{A}$ against each element in respective columns of $\mathbf{C}$:

$$\mathbf{AC} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \\ c_{31} & c_{32} \end{bmatrix} = \begin{bmatrix} a_{11}c_{11} + a_{12}c_{21} + a_{13}c_{31} & a_{11}c_{12} + a_{12}c_{22} + a_{13}c_{32} \\ a_{21}c_{11} + a_{22}c_{21} + a_{23}c_{31} & a_{21}c_{12} + a_{22}c_{22} + a_{23}c_{32} \\ a_{31}c_{11} + a_{32}c_{21} + a_{33}c_{31} & a_{31}c_{12} + a_{32}c_{22} + a_{33}c_{32} \end{bmatrix}$$

so that the product $\mathbf{AC}$ has the *n* rows of $\mathbf{A}$ and the *m* columns of $\mathbf{C}$.

Multiplying matrices in R is easy. To get the product $\mathbf{A} = \begin{bmatrix} 5 & 9 \\ 3 & 4 \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} 1 & 3 \\ 7 & 2 \end{bmatrix}$ we compute

```
> r1 <- c(5, 9)
> r2 <- c(3, 4)
> A <- rbind(r1, r2)
> A
   [,1] [,2]
r1    5    9
r2    3    4

> c1 <- c(1, 7)
> c2 <- c(3, 2)
> B <- cbind(c1, c2)
> B
     c1 c2
[1,]  1  3
[2,]  7  2
> A%*%B
     b1 b2
[1,] 68 33
[2,] 31 17
```

As mentioned, since matrix multiplication is not commutative, $\mathbf{BA}$ will yield a different result than $\mathbf{AB}$:

```
> B%*%A
     a1 a2
[1,] 14 21
[2,] 41 71
```

Note that the above matrix product is not equal to that of $\mathbf{AB}$ computed earlier.

Finally, be sure when multiplying matrices that you use "%" to enclose "*" and not "*" alone to get the product. If you use "*", as might at first seem like the obvious thing to do, you will not get a matrix product as computed above. What you will get is a product **elementwise**, meaning each respective element in each position in the matrix is multiplied by each other.

For example, in multiplying $\mathbf{AB}$ using only "*" instead of "%*%", we get

```
> A*B
   [,1] [,2]
r1    5   27
r2   21    8
```

As can be seen, it generated the matrix product $\mathbf{AB}$ by multiplying corresponding elements in each position. It generated a product, but not the product we typically want when multiplying matrices.

For a given matrix $\mathbf{A}$ that is symmetric and a $\mathbf{y}$ vector, the product $\mathbf{y}'\mathbf{Ay}$ is known as a **quadratic form**. In many multivariate procedures, it is important that the product $\mathbf{y}'\mathbf{Ay}$ be positive (i.e., $\mathbf{y}'\mathbf{Ay} > 0$). Such quadratic forms are called **positive definite**. Positive **semidefinite** quadratic forms are those for which $\mathbf{y}'\mathbf{Ay} \geq 0$. Matrices that are positive definite have **eigenvalues** greater than zero while matrices that are positive semidefinite have eigenvalues of zero or greater. Matrices that are not positive definite or positive semidefinite have negative eigenvalues. Matrices that are of full rank (see Section A.18) will have all eigenvalues positive.

A.7. Transpose & Trace

The **transpose** of a matrix is the matrix generated when one interchanges the rows and columns of a matrix. For instance, we denote the transpose of matrix $\mathbf{B} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$ as $\mathbf{B}'$:

$$\mathbf{B}' = \begin{bmatrix} b_{11} & b_{21} \\ b_{12} & b_{22} \end{bmatrix}$$

The **trace** of a square matrix is the sum of elements along the main diagonal. For example, the trace of the 3 x 3 matrix:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

is equal to $a_{11} + a_{22} + a_{33}$. The trace of non-square matrices is not defined. Because elements that make up the trace of a covariance matrix represent variances, the computation of the trace is very common in multivariate analysis. For instance, in the derivation of **principal components** (Chapter 13), one of the properties of principal components is that the sum of their variances, which are called **eigenvalues**, is equal to the sum of the variances of the original variables. In a covariance matrix, that sum of the individual variances of each variable is found along the main diagonal. Hence, we say that a feature of principal components analysis is that the sum of eigenvalues is equal to the trace of the original covariance matrix:

$$\sum_{i=1}^{p} \lambda_i = s_1^2 + s_2^2 + ... + s_p^2$$

where the sample variances $s_1^2 + s_2^2 + ... + s_p^2$ are found along the main diagonal of the covariance matrix:

$$\mathbf{S} = \mathbf{ASA}' = \begin{pmatrix} s_1^2 & 0 & ... & 0 \\ 0 & s_2^2 & ... & 0 \\ 0 & 0 & s_3^2 & 0 \\ 0 & 0 & ... & s_p^2 \end{pmatrix}$$

We can easily demonstrate in R some useful results in matrix algebra involving transposes and traces:

1. $\mathbf{tr(A + B) = tr(A) + tr(B)}$ - In words, **the trace of a sum is equal to the sum of the traces**. In R, we request the trace of a matrix by `tr`. We first request the trace of the sum $\mathbf{A+B}$:

```
> library(psych)
> tr(A + B)
[1] 12
```

The above represents the left-hand side of the result. Computing the right-hand side, we find that the above sum is equal to

```
> tr(A) + tr(B)
[1] 12
```

which demonstrates the equivalence.

2. $\mathbf{(A + B)' = A' + B'}$ - In words, **the transpose of a sum is equal to the sum of transposes**. We first generate the transpose of the sum (i.e., $\mathbf{(A + B)'}$):

```
> t(A+B)
   [,1] [,2]
a1    6   10
a2   12    6
```

We then request the sum of the transposes (i.e., $\mathbf{A' + B'}$):

```
> t(A) + t(B)
   [,1] [,2]
a1    6   10
a2   12    6
```

which demonstrates the equivalence.

3. $(\mathbf{AB})' = \mathbf{B}'\mathbf{A}'$ - In words, **the transpose of a product is equal to the product of transposes in reverse order**. Recall that to multiply matrices in R, we use the operator `%*%`. Thus, for the expression on the left-hand side, we compute the transpose of the matrix product $\mathbf{AB}$:

```
> t(A%*%B)
   [,1] [,2]
b1   68   31
b2   33   17
```

The expression on the right-hand side is computed as the transpose of $\mathbf{B}$ multiplied by the transpose of $\mathbf{A}$:

```
> t(B)%*%t(A)
   [,1] [,2]
b1   68   31
b2   33   17
```

which demonstrates the equivalence.

A.8. Identity Matrices

An **identity matrix** is defined as a matrix having zeros everywhere except the main diagonal which has elements equal to 1. A property of identity matrices is that for any matrix $\mathbf{A}$, it is true that $\mathbf{IA} = \mathbf{AI} = \mathbf{A}$. For example, a 3 x 3 identity matrix is given as:

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

We could construct the matrix simply via `diag(3)`. We can also generate a 3 x 3 identity matrix `I` in R by:

```
> I <- matrix(0, nrow = 3, ncol = 3)
> I[1, 1] <- 1
> I[2, 2] <- 1
> I[3, 3] <- 1
> I
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    1    0
[3,]    0    0    1
```

where `I` is the name of our newly created matrix, the line `matrix(0, nrow = 3, ncol = 3)` requests a matrix of zeros with three rows and three columns, `I[1, 1] <- 1` requests R to put a "1" in row 1, column 1, `I[2, 2] <- 1` requests R to put a "1" in row 2, column 2, and `I[3, 3] <- 1` requests R to put a "1" in row 3, column 3. Had we not requested these last three lines of code, R would have simply generated a 3 x 3 zero matrix:

```
> I <- matrix(0, nrow = 3, ncol = 3)
> I
     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    0    0
[3,]    0    0    0
```

If we desired a larger identity matrix, suppose 5 x 5, we could likewise code:

```
> I <- matrix(0, nrow = 5, ncol = 5)
> I[1, 1] <- 1
> I[2, 2] <- 1
> I[3, 3] <- 1
> I[4, 4] <- 1
> I[5, 5] <- 1
> I

     [,1] [,2] [,3] [,4] [,5]
[1,]    1    0    0    0    0
[2,]    0    1    0    0    0
[3,]    0    0    1    0    0
[4,]    0    0    0    1    0
[5,]    0    0    0    0    1
```

We could also alter the general `matrix` code, and easily generate a matrix with values of say, 5 everywhere, except for values of 2 along the main diagonal:

```
> M <- matrix(5, nrow = 3, ncol = 3)
> M[1, 1] <- 2
> M[2, 2] <- 2
> M[3, 3] <- 2
> M
     [,1] [,2] [,3]
[1,]    2    5    5
[2,]    5    2    5
[3,]    5    5    2
```

We can also verify the main diagonal of an identity matrix or any other matrix for that matter, by the function `diag`. For example, for the matrix `M`, we request the diagonal:

```
> diag(M)
[1] 2  2  2
```

which also serves to confirm the matrix to be a 3 x 3 matrix (since there are three matching values of 2).

Even more generally, we can use the matrix function to generate any matrix with specific values in each position. For instance, if we wanted to generate the matrix $\mathbf{A} = \begin{bmatrix} 5 & 9 \\ 3 & 4 \end{bmatrix}$, we would enter the following code:

```
> A <- matrix(nrow = 2, ncol = 2)
> A[1, 1] <- 5
> A[1, 2] <- 9
> A[2, 1] <- 3
> A[2, 2] <- 4
> A
     [,1] [,2]
[1,]    5    9
[2,]    3    4
```

Of course, it is usually much easier to generate such matrices by simply binding column or row vectors, as we did earlier, especially for large matrices. Even more efficiently, we could have also constructed $\mathbf{A}$ by:

```
> A <- matrix(c(5, 3, 9, 4), nrow = 2)
> A
     [,1] [,2]
[1,]    5    9
[2,]    3    4
```

A.9. Vectors

In physical applications, a **vector** is a quantity that has **magnitude** and **direction**. In computer science fields, it can be considered simply a list of numbers. It can also be likened to a matrix that consists of a single row or a single column. For example, vector $\mathbf{c}$ is a 3 x 1 matrix (i.e., 3 rows, 1 column):

$$\mathbf{c} = \begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \end{bmatrix}$$

whereas $\mathbf{c}'$ is a 1 x 3 row matrix:

$$\mathbf{c}' = \begin{bmatrix} c_{11} & c_{12} & c_{13} \end{bmatrix}$$

As we can see, $\mathbf{c}'$ is the transpose of $\mathbf{c}$.

Let us now assign real numbers to vector **c**. Let vector **c** equal:

$$\mathbf{c} = \begin{bmatrix} 5 \\ 2 \\ 9 \end{bmatrix}$$

To generate this vector in R, we compute

```
> c <- c(5, 2, 9)
> c
[1] 5 2 9
```

A.10. Inner Product or Dot Product

The **inner product** or dot product (Strang, 1993, p. 10) is the product generated when two vectors are multiplied by one another, where $\mathbf{a}'\mathbf{a}$ denotes a sum of squared elements. Two vectors are **orthogonal** if their inner product is equal to 0. If we take the square root of $\mathbf{a}'\mathbf{a}$, we obtain the length of $\mathbf{a}$.

The inner product of vectors $\mathbf{c}$ and $\mathbf{d}$ is equal to

$$\mathbf{c}'\mathbf{d} = \begin{bmatrix} 5 & 2 & 9 \end{bmatrix} \begin{bmatrix} 6 \\ 2 \\ 5 \end{bmatrix} = 30 + 4 + 45 = 79$$

To demonstrate in R, we first generate the vector $\mathbf{d}$:

```
> d <- c(6, 2, 5)
> d
[1] 6 2 5
```

The inner product of vectors $\mathbf{c}$ and $\mathbf{d}$ is therefore

```
> c%*%d
     [,1]
[1,]   79
```

which we note is equal to that computed manually.

Notice that in computing the product $\mathbf{c}'\mathbf{d}$ in R, we did not have to specify vector $\mathbf{c}$ as its transpose. R makes the adjustment itself and multiplies the vectors. Had R not made this adjustment, we would be multiplying two column vectors, which would not be conformable for multiplication. We could have also specified the transpose directly and obtained the same inner product:

```
> t(c)%*%d
     [,1]
[1,]   79
```

Multiplication of $\mathbf{c}$ by a scalar a proceeds by multiplying each element of $\mathbf{c}$ by that scalar. Thus, the product $a\mathbf{c}$ is given by

$$a\mathbf{c} = a \begin{bmatrix} c_{11} \\ c_{21} \\ c_{31} \end{bmatrix} = \begin{bmatrix} a(c_{11}) \\ a(c_{21}) \\ a(c_{31}) \end{bmatrix}.$$

For example, suppose we wish to multiply matrix $\mathbf{c} = \begin{bmatrix} 5 \\ 2 \\ 9 \end{bmatrix}$ by the scalar $a = 10$. The result is

```
> 10*c
[1] 50 20 90
```

As we can see, R multiplied each element of the vector by the scalar 10. Multiplying vectors by scalars is a key operation in multivariate analysis. For instance, when obtaining principal components, each component is nothing more than a vector (called an **eigenvector**) associated with a unique scalar (called an **eigenvalue**). The eigenvector determines the **direction** of the component, while the eigenvalue determines the magnitude (or **length**) of the vector. For details, see Chapter 13.

A.11. Zero Matrix

In a zero matrix (or "null" matrix), every element is equal to 0. We can easily produce a zero matrix in R. For instance, suppose we want a 3 x 3 zero matrix:

```
> M <- matrix(0, nrow = 3, ncol = 3)
> M
     [,1] [,2] [,3]
[1,]    0    0    0
[2,]    0    0    0
[3,]    0    0    0
```

It is easy to see that the zero matrix is the matrix equivalent of zero in scalar algebra, such that pre- or post-multiplication of a matrix by the zero matrix always generates another zero matrix. For example, for a non-zero matrix $\mathbf{A}$, the products $\mathbf{A0}$ and $\mathbf{0A}$ are the same. In both cases, the solution is $\mathbf{0}$.

A.12. Inverse of a Matrix

Inverses in matrix algebra are very important and play a significant role in multivariate analysis. Recall that in scalar algebra, the inverse of an ordinary scalar *a* is defined as a^{-1} so that

$$a \cdot a^{-1} = \frac{a}{1} \cdot \frac{1}{a} = \frac{a}{a} = 1$$

is true, assuming $a \neq 0$ (if $a = 0$ then while $\frac{a}{1}$ is defined, $\frac{1}{a}$ is not).

In matrix algebra, the analogous statement of the scalar relation for matrices $\mathbf{A}$ and $\mathbf{B}$ is:

$$\mathbf{AB} = \mathbf{BA} = \mathbf{I}$$

That is, if $\mathbf{A}$ and $\mathbf{B}$ are square matrices and a matrix can be found such that $\mathbf{AB}=\mathbf{BA}=\mathbf{I}$, then the matrix $\mathbf{A}$ is regarded as an invertible matrix, and matrix $\mathbf{B}$ is regarded as the inverse of matrix $\mathbf{A}$.

Generating matrix inverses in R is easy. Oftentimes in multivariate analysis, we are required to invert either covariance or correlation matrices, usually not explicitly, but rather as a part of one or more of the statistical tests we run (Raveh, 1985). For instance, consider once more the previously constructed correlation matrix:

```
> cormatrix
         a     b     c     d     e     f     g     h
[1,] 1.000 0.343 0.505 0.308 0.693 0.208 0.400 0.455
[2,] 0.343 1.000 0.203 0.400 0.187 0.108 0.386 0.385
[3,] 0.505 0.203 1.000 0.398 0.303 0.277 0.286 0.167
[4,] 0.308 0.400 0.398 1.000 0.205 0.487 0.385 0.465
[5,] 0.693 0.187 0.303 0.205 1.000 0.200 0.311 0.485
[6,] 0.208 0.108 0.277 0.487 0.200 1.000 0.432 0.310
[7,] 0.400 0.386 0.286 0.385 0.311 0.432 1.000 0.365
[8,] 0.455 0.385 0.167 0.465 0.485 0.310 0.365 1.000
```

We can request R to compute the inverse of `cormatrix` by `solve(cormatrix)`. We name the new matrix by the name $\mathbf{D}$:

```
> D <- solve(cormatrix)

          [,1]        [,2]        [,3]        [,4]        [,5]        [,6]        [,7]        [,8]
a  2.62391779 -0.32529402 -0.79748547  0.07143344 -1.34883744  0.14277980 -0.27074036 -0.25993622
b -0.32529402  1.45211520  0.03592592 -0.40955548  0.22539180  0.31041745 -0.38009280 -0.29342246
c -0.79748547  0.03592592  1.56495953 -0.48016535  0.04087805 -0.13719304 -0.03786275  0.34747695
d  0.07143344 -0.40955548 -0.48016535  1.85491453  0.19261836 -0.58606745 -0.05349807 -0.54938320
e -1.34883744  0.22539180  0.04087805  0.19261836  2.14973569 -0.07659356 -0.06236400 -0.56556392
f  0.14277980  0.31041745 -0.13719304 -0.58606745 -0.07659356  1.53618210 -0.49852528 -0.14614977
g -0.27074036 -0.38009280 -0.03786275 -0.05349807 -0.06236400 -0.49852528  1.55055676 -0.08044157
h -0.25993622 -0.29342246  0.34747695 -0.54938320 -0.56556392 -0.14614977 -0.08044157  1.77763927
```

To verify that the above matrix is indeed the inverse of `cormatrix`, it can easily be demonstrated that their product is equal to the identity matrix, hence establishing the relation $\mathbf{AB}=\mathbf{BA}=\mathbf{I}$.

We now demonstrate some of the more common properties of matrix inverses.

1. $(\mathbf{A}^{-1})' = (\mathbf{A}')^{-1}$ - In words, **the transpose of the inverse of a matrix is equal to the inverse of its transpose**. We demonstrate with matrix $\mathbf{A}=\begin{bmatrix}5 & 9\\ 3 & 4\end{bmatrix}$. We first request the left-hand side of the property, that of the transpose of the inverse (recall that `t` in R is the function for **transpose** whereas `tr` is the function for **trace**):

```
> t(solve(A))
           a1         a2
[1,] -0.5714286  0.4285714
[2,]  1.2857143 -0.7142857
```

We then compute the inverse of the transpose, and note that it is equal to the transpose of the inverse computed above:

```
> solve(t(A))
             a1         a2
[1,] -0.5714286  0.4285714
[2,]  1.2857143 -0.7142857
```

2. $(\mathbf{AB})^{-1} = (\mathbf{B})^{-1}(\mathbf{A})^{-1}$ - In words, **the inverse of the product of matrices is equal to the product of inverses in reverse order**. We demonstrate this property by first requesting the inverse of the product (left-hand side):

```
> solve(A%*%B)
          [,1]       [,2]
b1  0.1278195 -0.2481203
b2 -0.2330827  0.5112782
```

We then request the right-hand side, the product of inverses in reverse order:

```
> solve(B) %*% solve(A)
          [,1]       [,2]
b1  0.1278195 -0.2481203
b2 -0.2330827  0.5112782
```

Note that the resulting matrix is identical to that produced above.

3. $(\mathbf{A}^{-1})^{-1} = \mathbf{A}$ - In words, **the inverse of the inverse of a matrix is equal to the original matrix**. This is also easily demonstrated in R. The left-hand side, $(\mathbf{A}^{-1})^{-1}$, is computed as

```
> solve(solve(A))
     a1 a2
[1,]  5  9
[2,]  3  4
```

which is easily shown to be equal to the original matrix $\mathbf{A}$:

```
> A
     a1 a2
[1,]  5  9
[2,]  3  4
```

A.13. Symmetric Matrices and Inverses

An *n* x *n* matrix $\mathbf{A}$ is symmetric if it is equal to its transpose, that is, $\mathbf{A} = \mathbf{A}'$. For instance, consider the matrix $\mathbf{S}$:

$$\mathbf{S} = \begin{bmatrix} 6 & 4 & 8 \\ 4 & 9 & 3 \\ 8 & 3 & 7 \end{bmatrix}$$

The matrix is symmetric since the upper triangular is a mirror reflection of the lower triangular such that $\mathbf{S} = \mathbf{S}'$, as we can easily demonstrate in R. First, we construct the matrix $\mathbf{S}$:

```
> s1 <- c(6, 4, 8)
> s2 <- c(4, 9, 3)
> s3 <- c(8, 3, 7)
> S <- cbind (s1, s2, s3)
> S
     s1 s2 s3
[1,]  6  4  8
[2,]  4  9  3
[3,]  8  3  7
```

When taking the transpose, we find

```
> t(S)
   [,1] [,2] [,3]
s1    6    4    8
s2    4    9    3
s3    8    3    7
```

which we note is equivalent to the original matrix $\mathbf{S}$. Hence, matrix $\mathbf{S}$ is symmetric. If a matrix is symmetric and invertible, then its inverse is also symmetric. For instance, computing the inverse of $\mathbf{S}$, we obtain

```
> solve(S)
          [,1]        [,2]        [,3]
s1 -0.31395349  0.02325581  0.34883721
s2  0.02325581  0.12790698 -0.08139535
s3  0.34883721 -0.08139535 -0.22093023
```

We note that the matrix $\mathbf{S}^{-1}$ is symmetric, as was the original symmetric matrix $\mathbf{S}$.

A.14. Determinants

The determinant of a matrix is a unique number associated with a matrix. Determinants are defined only for **square** matrices. Though the computation of determinants for matrices of higher dimensions can quickly become unwieldy and is best left to software, it is a simple matter to demonstrate the computation for simple lower-dimension matrices. For 2 x 2 matrix $\mathbf{A}$, the determinant is computed as

$$|\mathbf{A}| = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}.$$

For a 3 x 3 matrix, the computation of the determinant is computed:

$$|\mathbf{A}| = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix}$$

$$= a_{11}\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix}(-1)^2 + a_{12}\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix}(-1)^3 + a_{13}\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}(-1)^4$$

$$= a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{21}a_{32}a_{13} - a_{31}a_{22}a_{13} - a_{21}a_{12}a_{33} - a_{32}a_{23}a_{11}.$$

We can obtain the determinant of a matrix easily in R. For example, the determinant of `cormatrix` aforementioned is computed as:

```
> det(cormatrix)
[1] 0.06620581
```

Determinants play a significant role in statistical analyses. When computing, for instance, $\mathbf{b} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}$ in Chapter 7, to solve for $\mathbf{b}$, we are required to take the inverse of $(\mathbf{X}'\mathbf{X})$. The inverse, however, does not exist for a matrix whose determinant is equal to 0. What this translates into statistically is that if the data matrix $\mathbf{X}$ exhibits any kind of problems, such as being of less than full rank (which translates to columns and rows being **dependent** instead of **independent** – see Section A.15), this means $\mathbf{X}'\mathbf{X}$ will not be invertible, and hence a solution for $\mathbf{b}$ in $\mathbf{b} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}$ is not obtainable. The geometric interpretation of a determinant for two vectors is that of a quadrilateral for which if the two vectors are perfectly correlated, the quadrilateral collapses into a line, which simultaneously generates a determinant of zero (Lattin, Carroll and Green, 2003). Some properties and results of determinants include the following:

1. $|\mathbf{A}'| = |\mathbf{A}|$ - In words, **the determinant of the transpose of a matrix is equal to the determinant of the matrix**. We demonstrate in R using `cormatrix`. We first compute the determinant of the transpose:

```
> det(t(cormatrix))
[1] 0.06620581
```

This is equal to the original determinant of `cormatrix`, since:

```
> det(cormatrix)
[1] 0.06620581
```

2. $|\mathbf{AB}| = |\mathbf{A}||\mathbf{B}|$ - In words, **the determinant of a product of matrices (if defined, which means if $\mathbf{A}$ and $\mathbf{B}$ are conformable for multiplication, and since we are taking the determinant, implies $\mathbf{AB}$ is square) is equal to the product of determinants**. This is easily demonstrated. Recall matrices $\mathbf{A} = \begin{bmatrix} 5 & 9 \\ 3 & 4 \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} 1 & 3 \\ 7 & 2 \end{bmatrix}$. We compute the determinant of $\mathbf{AB}$:

```
> det(A%*%B)
[1] 133
```

and equate this to the product of determinants:

```
> det(A) %*% det(B)

     [,1]
[1,]  133
```

which demonstrates the equivalence.

3. $|k\mathbf{A}| = k^n |\mathbf{A}|$ - In words, **if *k* is a constant, then for a square matrix with n rows, the determinant of the product of the constant and the matrix is equal to the product of the constant raised to the power of n and the determinant of the matrix**. This is also easily demonstrated in R. Suppose we let *k* = 3. Then the determinant of $3\mathbf{A}$ is found:

```
> det(3*A)
[1] -63
```

and is shown equivalent to

```
> 3^2%*%det(A)
     [,1]
[1,]  -63
```

A.15. Linear Independence, Determinants, and Matrix Invertibility

The concepts of **linear independence** and **linear dependence** are at the core of linear and matrix algebra, and since these are the structural foundations of statistics, they are likewise important concepts in applied analysis.

We begin by first defining linear dependence. Given a set of two vectors, if we can write one or more vectors as a **linear combination** of one or more vectors in the set, then the set of vectors is deemed **linearly dependent**. If we cannot write one or more vectors as a linear combination of one or more vectors in the set, then the set of vectors is **linearly independent**.

More formally, consider the following linear combination:

$$a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \ldots + a_n\mathbf{v}_n = \mathbf{0} \qquad \text{(A-1)}$$

If equality (A-1) holds true for possible scalars a_1, a_2, . . . other than 0 (i.e., the trivial case), then the set of vectors is a set of linearly **dependent** vectors. If the above equality holds for 0 only, then the set of vectors is a set of linearly **independent** vectors.

For instance, consider the matrix $\mathbf{D}$:

$$\mathbf{D} = \begin{bmatrix} 2 & 5 & 7 \\ 3 & 1 & 8 \\ 4 & 10 & 14 \end{bmatrix}$$

Is the set linearly dependent? Yes, because notice that row 3 (i.e., 4, 10, 14) is a scalar multiple of row 1 (i.e., 2, 5, 7), since if we multiply row 1 by a scalar of 2, we get row 3. Hence, there is linear dependence among these two rows, and when there is linear dependence, this also implies that the determinant of the matrix will equal 0 and the matrix will not be invertible. Notice that linear dependence or independence refers to the **particular set of vectors you are considering**. Speaking of the linear independence or dependence of a single vector **without reference to its set** has no meaning. Strictly speaking, a single nonzero vector is considered by mathematicians to be linearly independent, but the point is that **without reference to its set**, one cannot determine the linear dependence or independence of a vector.

We demonstrate how to establish linear dependence of matrix $\mathbf{D}$ in R. First we construct the matrix:

```
> c1 <- c(2, 3, 4)
> c2 <- c(5, 1, 10)
> c3 <- c(7, 8, 14)
> D <- cbind(c1, c2, c3)
> D
     c1 c2 c3
[1,]  2  5  7
[2,]  3  1  8
[3,]  4 10 14
```

We compute the determinant of $\mathbf{D}$, and find it equal to 0:

```
> det(D)
[1] 0
```

Given that $\mathbf{D}$ has a determinant equal to 0, we should also expect to not be able to compute its inverse. When we attempt to solve, we obtain:

```
> solve(D)
Error in solve.default(D) :
  Lapack routine dgesv: system is exactly singular
```

R reports that "system is exactly singular." A matrix is regarded as **singular** when it cannot be inverted. Analogously, it implies that its determinant is equal to 0. Another equivalent statement is to say the matrix is not of full rank (see Section A.18). Nonsingular matrices have determinants that are not equal to 0 and are of full rank.

A set of linearly independent vectors in a vector space that span the space is generally referred to as a **basis** for the space. As we will discuss shortly, the number of linearly independent vectors in that space is the **rank** of the vector space. If the pairs of vectors in that space are pairwise mutually orthogonal, then that basis is regarded as an **orthogonal basis**. Furthermore, if each of the vectors in the space is of unit length (i.e., $\mathbf{aa}' = 1$), then the basis is called an

orthonormal basis. Orthonormal bases can be generated using what is known as the **Gram-Schmidt orthogonalization process**. If every vector in a vector space is expressible as a linear combination of particular vectors, then these vectors are said to **span the vector space**.

A.17. Orthogonality

A square matrix $\mathbf{A}$ is said to be an orthogonal matrix if the following holds:

$$\mathbf{AA'} = \mathbf{A'A} = \mathbf{I}$$

In words, **a matrix post-multiplied by its transpose is equal to the matrix pre-multiplied by its transpose, and both of these products are equal to the identity matrix**. If this condition exists, matrix $\mathbf{A}$ is defined as **orthogonal**. Analogously, $\mathbf{AA'} = \mathbf{A'A} = \mathbf{I}$ also implies that the columns or rows of $\mathbf{A}$ form an orthonormal basis.

Substantive areas of multivariate analysis in which orthogonal matrices play a very important role are in **principal components analysis** and **factor analysis**. More specifically, these concepts play a role in the orthogonal rotation of estimated component or factor loadings. For example, as discussed in Chapter 14, to rotate a factor solution to a new coordinate system, loadings are multiplied by the transformation matrix $\mathbf{T}$:

$$\mathbf{T} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}.$$

A few other properties of orthogonal matrices (which we do not demonstrate):

- If $\mathbf{A}$ is an orthogonal matrix, then its inverse, $\mathbf{A}^{-1}$ is also an orthogonal matrix.
- If $\mathbf{A}$ is an orthogonal matrix, and $\mathbf{B}$ is an orthogonal matrix, then $\mathbf{AB}$ is an orthogonal matrix.
- The determinant of an orthogonal matrix is equal to either 1 or -1.

A.17. Eigenvalues & Eigenvectors

The importance and relevance of the concepts of eigenvalues and eigenvectors in both univariate and multivariate applied statistics cannot be overstated. Quite simply, much of so-called advanced statistical analysis boils down to the extraction of eigenvalues and eigenvectors of covariance or correlation matrices.

Though there are numerous substantive reasons for applying eigenvalue and eigenvector decomposition (EVD) to problems in the social and natural sciences, our consideration here is to simply survey how they are computed. Throughout the book, especially later chapters, their use will become apparent.

Let $\mathbf{A}$ be a $n \cdot n$ square matrix. The eigenvalue problem is to find a scalar λ and vector $\mathbf{x}$ (other than zero) so that the following equality holds:

$$\mathbf{Ax} = \boldsymbol{\lambda}\mathbf{x} \qquad \text{(A-2)}$$

The scalar λ is called an **eigenvalue** of the matrix $\mathbf{A}$ and the vector $\mathbf{x}$ is called an **eigenvector** associated with λ. To solve for λ and $\mathbf{x}$, we can re-write (A-2) as

$$\mathbf{Ax} - \lambda\mathbf{x} = \mathbf{0}$$
$$(\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = \mathbf{0}$$

It stands that if $|\mathbf{A} - \lambda\mathbf{I}| \neq 0$, then this implies that $(\mathbf{A} - \lambda\mathbf{I})$ has an inverse, which means that $\mathbf{x} = \mathbf{0}$ is the only solution. This is referred to as the **trivial solution**. To obtain nontrivial solutions, we deliberately set $|\mathbf{A} - \lambda\mathbf{I}| = 0$ and find values of λ that can be substituted into $(\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = \mathbf{0}$ to then provide a solution for $\mathbf{x}$.

The equation $|\mathbf{A} - \lambda\mathbf{I}| = 0$ is called the **characteristic equation**. For a matrix $\mathbf{A}$ that is $n \cdot n$ (i.e., square, with n rows and n columns), the characteristic equation will have n roots, that is, n eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$, not all necessarily different from one another and not all nonzero.

As a simple example of EVD, consider the following matrix:

$$\mathbf{A} = \begin{bmatrix} 2 & 2 \\ 0 & 6 \end{bmatrix}$$

The vector $\mathbf{x} = \begin{bmatrix} 0.4472136 \\ 0.8944272 \end{bmatrix}$ is an eigenvector of $\mathbf{A}$ that corresponds to the eigenvalue 6, because

$$\begin{bmatrix} 2 & 2 \\ 0 & 6 \end{bmatrix} \begin{bmatrix} 0.4472136 \\ 0.8944272 \end{bmatrix} = 6 \begin{bmatrix} 0.4472136 \\ 0.8944272 \end{bmatrix}$$

We can easily demonstrate in R that the above holds. We first construct the matrix $\mathbf{A}$:

```
> a1 <- c(2, 0)
> a2 <- c(2, 6)
> A <- cbind(a1, a2)
> A
     a1 a2
[1,]  2  2
[2,]  0  6

> x <- c(0.4472136, 0.8944272)
> x
[1] 0.4472136 0.8944272
```

The product of **Ax** is equal to:

```
> A%*%x
         [,1]
[1,] 2.683282
[2,] 5.366563
```

which is equal to the product of

```
> 6*x
[1] 2.683282 5.366563
```

and hence we have demonstrated that $\mathbf{x} = \begin{bmatrix} 0.4472136 \\ 0.8944272 \end{bmatrix}$ is an eigenvector of $\mathbf{A} = \begin{bmatrix} 2 & 2 \\ 0 & 6 \end{bmatrix}$

corresponding to an eigenvalue of 6.

Computing eigenvalues and eigenvectors for matrices of small dimension provides little computational difficulty. However, for the majority of times in multivariate analysis we are required to compute EVD for matrices of much larger dimension, and computed manually "by hand," the computations quickly become unwieldy. Software is consequently required. For instance, consider once more `cormatrix`. We can request the eigenvalues and eigenvectors of this matrix quite easily in R by using the `eigen` function:

```
> eigen(cormatrix)

$values
[1] 3.4470520 1.1572358 0.9436513 0.8189869 0.6580753 0.3898612 0.3360577
[8] 0.2490798

$vectors
           [,1]        [,2]         [,3]        [,4]         [,5]       [,6]
[1,] -0.4125682 -0.45771854 -0.098873465  0.08795002  0.066755981 -0.1352881
[2,] -0.3034726  0.11443307  0.637320040  0.47138991  0.088175982 -0.4697628
[3,] -0.3180940 -0.06869755 -0.546391282  0.58136533 -0.121757436  0.1921707
[4,] -0.3730602  0.43006317 -0.001725853  0.11149001 -0.471416291  0.1757057
[5,] -0.3572744 -0.54341592 -0.067983885 -0.31379342  0.005351703 -0.2192800
[6,] -0.3008318  0.49347718 -0.380799221 -0.40057953  0.065440460 -0.5409419
[7,] -0.3664721  0.21605328  0.060830978 -0.06123129  0.775010839  0.4437028
[8,] -0.3806410 -0.04717545  0.363552650 -0.39618322 -0.381782069  0.3945174
           [,7]        [,8]
[1,]  0.0911831  0.75610613
[2,] -0.1315217 -0.14378501
[3,] -0.3684690 -0.26465820
[4,]  0.6393505  0.03954700
[5,]  0.3112839 -0.57354688
[6,] -0.2453502  0.05840491
[7,]  0.1046301 -0.05574971
[8,] -0.5116712  0.02337248
```

The first eigenvalue extracted is equal to 3.4470520 with associated eigenvector

$$\begin{bmatrix} -0.4125682 \\ -0.3034726 \\ -0.3180940 \\ -0.3730602 \\ -0.3572744 \\ -0.3008318 \\ -0.3664721 \\ -0.3806410 \end{bmatrix}.$$

In line with our earlier discussion of EVD, then it should be true that the product of `cormatrix` and the eigenvector is equal to the eigenvalue multiplied by the eigenvector. We verify this in R by first producing the eigenvector, then generating the two products:

```
> cormatrix%*%eigenvec
        [,1]
c1 -1.422144
c2 -1.046086
c3 -1.096486
c4 -1.285958
c5 -1.231543
c6 -1.036983
c7 -1.263249
c8 -1.312089

> 3.4470520*eigenvec
[1] -1.422144 -1.046086 -1.096487 -1.285958 -1.231543 -1.036983 -1.263248
[8] -1.312089
```

We note that both vectors, that of `cormatrix*eigenvec` and `eigenvalue*eigenvec`, are equal, confirming that the stated eigenvector is indeed an eigenvector of the eigenvalue 3.4470520.

An especially important and relevant property of eigenvalues is the following:

$\prod_{i=1}^{n} \lambda_i = |\mathbf{A}|$ - In words, **the product of eigenvalues for matrix A is equal to the determinant of matrix A**. We can verify this for `cormatrix`. We first produce the vector of extracted eigenvalues and then request the product (`prod`) of these eigenvalues :

```
> eigenvalues <- c(3.4470520, 1.1572358, 0.9436513, 0.8189869, 0.6580753, 0.3898612, 0.3360577,
0.2490798)
> prod(eigenvalues)
[1] 0.06620582
```

We note that the value of 0.06620582 matches that value of the determinant of `cormatrix` computed earlier [i.e., `det(cormatrix)` = 0.06620581].

Another important feature of eigenvalues that is relevant to multivariate analysis is that the trace of a square matrix is equal to the sum of eigenvalues for that matrix. That is, for matrix $\mathbf{A}$, the following holds:

$$\text{tr}(\mathbf{A}) = \sum_{i=1}^{n} \lambda_i$$

For a covariance matrix, the trace of the matrix is the sum of variances along the main diagonal. What the above is saying is that once we solve for the given eigenvalues of the matrix, we can re-express the trace of the original matrix through summing eigenvalues. This simple equality is the basis on which the method of **principal components analysis** rests. More formally, PCA operates by **diagonalizing** the covariance (or correlation) matrix such that all entries in the matrix other than in the trace are equal to 0. In the spirit of factor analysis and structural equation modeling, eigenvalues are also sometimes referred to as **latent values** and their associated eigenvectors as **latent vectors** (Bollen, 1989).

A.18. Rank

A central problem of linear algebra, and hence one that also makes its way into multivariate analysis, is to obtain a solution for $\mathbf{x}$ in the fundamental equation:

$$\mathbf{Ax} = \mathbf{b}$$

For a solution to $\mathbf{x}$, we can solve $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$. **The rank of a matrix is defined as the number of linearly independent rows and columns of a matrix**. If the number of columns or rows in a matrix is equal to the number of linearly independent columns or rows, the matrix is said to be of **full rank**, and $\mathbf{A}^{-1}$ exists. If the number of columns or rows in a matrix is unequal to the number of linearly independent columns or rows, the matrix is said to be not of full rank, or equivalently, **reduced rank**. The objective in principal components analysis, for example, is to obtain a low-rank approximation of a given covariance or correlation matrix. The lower rank approximation constitutes the eigenvectors of the analyzed matrix. For instance, in a 10-variable problem, we will extract 10 components (there are typically as many components as there are variables), but we choose to retain only a few of these components if they account for most of the variance in the initial variables. Hence, through retaining only a few "meaningful" components, we obtain a lower rank approximation without, hopefully, losing too much information in the process.

Checking whether rows or columns of a matrix are linearly independent or dependent can be a lengthy ordeal. Fortunately, if linear independence does hold for a set of vectors, then it is a fact of matrix theory that the determinant of a matrix will not equal 0. Hence, as a quick check to ensure a matrix is of full rank, one simply needs to compute the determinant of the matrix in question and ensure it is not equal to 0.

For example, recall matrix $\mathbf{A}$:

$$\mathbf{A} = \begin{bmatrix} 2 & 2 \\ 0 & 6 \end{bmatrix}$$

The determinant of this matrix is equal to 12. Hence, we know the matrix is of full rank because the determinant is not equal to 0. Equivalently, the rows and columns in $\mathbf{A}$ are linearly independent. If the rows are linearly independent, this implies the columns are linearly independent as well.

In R, we can obtain the rank of a matrix by the function `qr`, which stands for the **QR factorization** of a matrix. Because the rows and columns of matrix $\mathbf{A}$ are linearly independent, we should expect the rank to be equal to 2. We confirm this in R (the function `qr` will produce more output than we need, we only show the output for matrix rank below):

```
> qr(A)

$rank
[1] 2
```

Indeed, as R confirms, the rank of matrix $\mathbf{A}$ is equal to 2. One can also use a variety of other functions in R to compute rank. For example, `rank.condition()` in the package `corpcor` (Schafer et al., 2014) will return the rank of a matrix.