Appendix E
Appendix: A Review of Matrices

Although first used by the Babylonians, matrices were not introduced into psychological research until Thurstone first used the word matrix in 1933 (Bock, 2007). Until then, data and even correlations were organized into "tables". Vectors, matrices and arrays are merely convenient ways to organize objects (usually numbers) and with the introduction of matrix notation, the power of matrix algebra was unleashed for psychometrics. Much of psychometrics in particular, and psychological data analysis in general consists of operations on vectors and matrices. In many commercial software applications, some of the functionality of matrices is seen in the use of "spreadsheets". Many commercial statistical packages do the analysis in terms of matrices but shield the user from this fact. This is unfortunate, because it is (with some practice) easier to understand the similarity of many algorithms when they are expressed in matrix form.

This appendix offers a quick review of matrix algebra with a particular emphasis upon how to do matrix operations in R. The later part of the appendix shows how some fairly complex psychometrics concepts are done easily in terms of matrices.

E.1 Vectors

A *vector* is a one dimensional array of n elements where the most frequently used elements are integers, reals (numeric), characters, or logical. Basic operations on a vector are addition, subtraction and multiplication. Although addition and subtraction are straightforward, multiplication is somewhat more complicated, for the order in which two vectors are multiplied changes the result. That is $\mathbf{ab} \neq \mathbf{ba}$. (In an attempt at consistent notation, vectors will be **bold** faced lower case letters.)

Consider **v1** = the first 6 integers, and **v2** = the next 6 integers:

```
> v1 <- seq(1, 6)
> v2 <- seq(7, 12)
```

```
> v1
[1] 1 2 3 4 5 6
> v2
[1]  7  8  9 10 11 12
```

We can add a constant to each element in a vector, add each element of the first vector to the corresponding element of the second vector, multiply each element by a *scaler*, or multiply each element in the first by the corresponding element in the second:

```
> v3 <- v1 + 20
> v4 <- v1 + v2
> v5 <- v1 * 3
> v6 <- v1 * v2
```

```
> v3
[1] 21 22 23 24 25 26
> v4
[1]  8 10 12 14 16 18
> v5
[1]  3  6  9 12 15 18
> v6
[1]  7 16 27 40 55 72
```

E.1.1 Vector multiplication

Strangely enough, a vector in R is dimensionless, but it has a `length`. There are three types of multiplication of vectors in R. Simple multiplication (each term in one vector is multiplied by its corresponding term in the other vector (e.g., **v6** <- **v1** $*$ **v2**), as well as the *inner* and *outer* products of two vectors. The *inner product* is a very powerful operation for it combines both multiplication and addition. That is, for two vectors of the same length, the inner product of **v1** and **v2** is found by the matrix multiply operator `%*%`

$$\begin{pmatrix} 1 \; 2 \; 3 \; 4 \; 5 \; 6 \end{pmatrix} \% * \% \begin{pmatrix} 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ 12 \end{pmatrix} = \sum_{i=1}^{n} v1_i v2_i = \sum_{i=1}^{n} v6 = 217 \tag{E.1}$$

In the previous example, because of the way R handles vectors, because **v1** and **v2** were of the same length, it was not necessary to worry about rows and columns and $\mathbf{v2}\% * \%\mathbf{v1} = \mathbf{v1}\% * \%\mathbf{v2}$. In general, however, the multiplication of two vectors will yield different results depending upon the order. A row vector times a column vector of the same length produces a single element which is equal to the sum of the products of the respective elements. But a column vector of length c times a row vector of length r times results in the c x r *outer product* matrix of products. To see this, consider the vector **v7** = seq(1,4) and the results of $\mathbf{v1}\% * \%\mathbf{v7}$ versus $\mathbf{v7}\% * \%\mathbf{v1}$. Unless otherwise specfied, all vectors may be thought of as column vectors. To force **v7** to be a row vector, use the transpose function `t`. To transpose a vector changes a column vector into a row vector and a row vector into a column vector. It is shown with the superscript T or sometimes with the superscript '.

Then $\underset{(6x1)}{\mathbf{v1}} \% * \% \underset{(1x4)}{\mathbf{v7}'} = \underset{(6x4)}{\mathbf{V8}}$ and $\underset{(4x1)}{\mathbf{v7}} \% * \% \underset{(1x6)}{\mathbf{v1}'} = \underset{(4x6)}{\mathbf{V9}}$. To clarify this notation, note that the first subscript of each vector refers to the number of rows and the second to the number of columns in a matrix. Matrices are written in **bold** face upper case letters. For a vector, of course, either the number of columns or rows is 1. Note also that for the multiplication to be done, the inner subscripts (e.g., 1 and 1 in this case) must correspond, but that the outer subscripts (e.g., 6 and 4) do not.

$$\underset{(6x1)}{\mathbf{v1}} \,\%*\%\, \underset{(1x4)}{\mathbf{v7}'} = \begin{pmatrix}1\\2\\3\\4\\5\\6\end{pmatrix} \%*\% \begin{pmatrix}1 & 2 & 3 & 4\end{pmatrix} = \begin{pmatrix}1 & 2 & 3 & 4\\2 & 4 & 6 & 8\\3 & 6 & 9 & 12\\4 & 8 & 12 & 16\\5 & 10 & 15 & 20\\6 & 12 & 18 & 24\end{pmatrix} = \underset{(6x4)}{\mathbf{V8}} \tag{E.2}$$

but

$$\underset{(4x1)}{\mathbf{v7}} \,\%*\%\, \underset{(1x6)}{\mathbf{v1}'} = \begin{pmatrix}1\\2\\3\\4\end{pmatrix} \%*\% \begin{pmatrix}1 & 2 & 3 & 4 & 5 & 6\end{pmatrix} = \begin{pmatrix}1 & 2 & 3 & 4 & 5 & 6\\2 & 4 & 6 & 8 & 10 & 12\\3 & 6 & 9 & 12 & 15 & 18\\4 & 8 & 12 & 16 & 20 & 24\end{pmatrix} = \underset{(4x6)}{\mathbf{V9}} \tag{E.3}$$

That is, in R

```
> v7 <- seq(1,4)
> V8  <-  v1 %*% t(v7)
> V9 <-  v7 %*% t(v1)
```

```
v1 %*% t(v7)
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    2    4    6    8
[3,]    3    6    9   12
[4,]    4    8   12   16
[5,]    5   10   15   20
[6,]    6   12   18   24

v7 %*% t(v1)
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    2    3    4    5    6
[2,]    2    4    6    8   10   12
[3,]    3    6    9   12   15   18
[4,]    4    8   12   16   20   24
```

and $\underset{(4x1)}{\mathbf{v7}} \,\%*\%\, \underset{(1x6)}{\mathbf{v1}} = \underset{(4x6)}{\mathbf{V9}} \neq \underset{(6x4)}{\mathbf{V8}}$.

E.1.2 Simple statistics using vectors

Although there are built in functions in R to do most of our statistics, it is useful to understand how these operations can be done using vector and matrix operations. Here we consider how to find the mean of a vector, remove it from all the numbers, and then find the average squared deviation from the mean (the variance).

Consider the mean of all numbers in a vector. To find this we just need to add up the numbers (the inner product of the vector with a vector of 1's) and then divide by n (multiply by the scaler 1/n). First we create a vector, **v** and then a second vector **one** of 1s by using the repeat operation.

```
> v <- seq(1, 7)
> one <- rep(1,length(v))
> sum.v <- t(one) %*% v
> sum.v

       [,1]
[1,]   28

> mean.v <- sum.v * (1/length(v))

     [,1]
[1,]  4

> mean.v <- t(one) %*% v * (1/length(v))

> v
[1] 1 2 3 4 5 6 7
> one
[1] 1 1 1 1 1 1 1
> sum.v
     [,1]
[1,]   28
```

The mean may be calculated in three different ways, all of which are equivalent.

```
> mean.v <- t(one) %*% v/length(v)

>  sum.v * (1/length(v))
     [,1]
[1,] 4
>  t(one) %*% v * (1/length(v))
     [,1]
[1,] 4
> t(one) %*% v/length(v)
     [,1]
[1,]  4
```

As vectors, this was

$$\sum_1^n v_i/n = \mathbf{1}^T\mathbf{v} * \frac{1}{n} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{pmatrix} * \frac{1}{7} = 4 \tag{E.4}$$

The variance is the average squared deviation from the mean. To find the variance, we first find deviation scores by subtracting the mean from each value of the vector. Then, to find the sum of the squared deviations take the inner product of the result with itself. This Sum of Squares becomes a variance if we divide by the degrees of freedom (n-1) to get an unbiased estimate of the population variance. First we find the mean centered vector:

```
> v - mean.v
```

```
[1] -3 -2 -1  0  1  2  3
```

And then we find the variance as the mean square by taking the inner product:

```
> Var.v <- t(v - mean.v) %*% (v - mean.v) * (1/(length(v) - 1))
```

```
Var.v
         [,1]
[1,] 4.666667
```

Compare these results with the more typical `scale`, `mean` and `var` operations:

```
> scale(v, scale = FALSE)
```

```
     [,1]
[1,]   -3
[2,]   -2
[3,]   -1
[4,]    0
[5,]    1
[6,]    2
[7,]    3
attr(,"scaled:center")
[1] 4
```

```
> mean(v)
[1] 4
> var(v)
[1] 4.666667
```

E.1.3 Combining vectors with `cbind` *and* `rbind`

To combine two or more vectors with the result being a vector, use the `c` function.

```
> x <- c(v1, v2,v3)
```

```
[1]  1  2  3  4  5  6  7  8  9 10 11 12 21 22 23 24 25 26
```

We can form more complex data structures than vectors by combining the vectors, either by columns (`cbind`) or by rows (`rbind`). The resulting data structure is a matrix with the number of rows and columns depending upon the number of vectors combined, and the number of elements in each vector.

```
> Xc <- cbind(v1, v2, v3)
```

```
     V1 V2 V3
[1,]  1  7 21
[2,]  2  8 22
[3,]  3  9 23
[4,]  4 10 24
[5,]  5 11 25
[6,]  6 12 26
```

```
> Xr <- rbind(v1, v2, v3)
```

```
   [,1] [,2] [,3] [,4] [,5] [,6]
V1    1    2    3    4    5    6
V2    7    8    9   10   11   12
V3   21   22   23   24   25   26
```

```
> dim(Xc)
```

```
[1] 6 3
```

```
> dim(Xr)
```

```
[1]  3 10
```

E.2 Matrices

A *matrix* is just a two dimensional (rectangular) organization of numbers. It is a vector of vectors. For data analysis, the typical data matrix is organized with rows containing the responses of a particular subject and the columns representing different variables. Thus, a 6 x 4 data matrix (6 rows, 4 columns) would contain the data of 6 subjects on 4 different variables. In the example below the matrix operation has taken the numbers 1 through 24 and organized them column wise. That is, a matrix is just a way (and a very convenient one at that) of organizing a data vector in a way that highlights the correspondence of multiple observations for the same individual.

R provides numeric row and column names (e.g., [1,] is the first row, [,4] is the fourth column, but it is useful to label the rows and columns to make the rows (subjects) and columns (variables) distinction more obvious. We do this using the `rownames` and `colnames` functions, combined with the `paste` and `seq` functions.

```
> Xij <- matrix(seq(1:24), ncol = 4)
> rownames(Xij) <- paste("S", seq(1, dim(Xij)[1]), sep = "")
> colnames(Xij) <- paste("V", seq(1, dim(Xij)[2]), sep = "")
> Xij
```

```
  V1 V2 V3 V4
S1  1  7 13 19
S2  2  8 14 20
S3  3  9 15 21
S4  4 10 16 22
S5  5 11 17 23
S6  6 12 18 24
```

Just as the *transpose of a vector* makes a column vector into a row vector, so does the *transpose of a matrix* swap the rows for the columns. Applying the `t` function to the matrix **Xij** produces **Xij**′. Note that now the subjects are columns and the variables are the rows.

```
> t(Xij)

   S1 S2 S3 S4 S5 S6
V1  1  2  3  4  5  6
V2  7  8  9 10 11 12
V3 13 14 15 16 17 18
V4 19 20 21 22 23 24
```

E.2.1 Adding or multiplying a vector and a Matrix

Just as we could with vectors, we can add, subtract, muliply or divide the matrix by a *scalar* (a number without a dimension).

```
> Xij + 4

   V1 V2 V3 V4
S1  5 11 17 23
S2  6 12 18 24
S3  7 13 19 25
S4  8 14 20 26
S5  9 15 21 27
S6 10 16 22 28
```

```
> round((Xij + 4)/3, 2)

     V1   V2   V3   V4
S1 1.67 3.67 5.67 7.67
S2 2.00 4.00 6.00 8.00
S3 2.33 4.33 6.33 8.33
S4 2.67 4.67 6.67 8.67
S5 3.00 5.00 7.00 9.00
S6 3.33 5.33 7.33 9.33
```

We can also add or multiply each row (or column, depending upon order) by a vector. This is more complicated that it would appear, for R does the operations columnwise. This is best seen in an example:

```
> v <- 1:4

[1] 1 2 3 4
```

```
> Xij + v

   V1 V2 V3 V4
S1  2 10 14 22
S2  4 12 16 24
```

```
S3   6 10 18 22
S4   8 12 20 24
S5   6 14 18 26
S6   8 16 20 28
```

```
> Xij * v
```

```
   V1 V2 V3 V4
S1  1 21 13 57
S2  4 32 28 80
S3  9  9 45 21
S4 16 20 64 44
S5  5 33 17 69
S6 12 48 36 96
```

These are not the expected results if the intent was to add or multiply a different number to each column! R operates on the columns and wraps around to the next column to complete the operation. To add the n elements of **v** to the n columns of **Xij**, use the `t` function to transpose **Xij** and then transpose the result back to the original order:

```
>  t(t(Xij) + v)
```

```
   V1 V2 V3 V4
S1  2  9 16 23
S2  3 10 17 24
S3  4 11 18 25
S4  5 12 19 26
S5  6 13 20 27
S6  7 14 21 28
```

```
> V10 <-  t(t(Xij) * v)
> V10
```

```
   V1 V2 V3 V4
S1  1 14 39 76
S2  2 16 42 80
S3  3 18 45 84
S4  4 20 48 88
S5  5 22 51 92
S6  6 24 54 96
```

To find a matrix of deviation scores, just subtract the means vector from each cell. The `scale` function does this with the option scale=FALSE. The default for `scale` is to convert a matrix to standard scores.

```
> scale(V10,scale=FALSE)
```

```
> scale(V10,scale=FALSE)
     V1 V2   V3  V4
S1 -2.5 -5 -7.5 -10
S2 -1.5 -3 -4.5  -6
S3 -0.5 -1 -1.5  -2
```

```
S4  0.5  1  1.5   2
S5  1.5  3  4.5   6
S6  2.5  5  7.5  10
attr(,"scaled:center")
  V1   V2   V3   V4
 3.5 19.0 46.5 86.0
```

E.2.2 Matrix multiplication

Matrix multiplication is a combination of multiplication and addition and is one of the most used and useful matrix operations. For a matrix $\underset{(rxp)}{\mathbf{X}}$ of dimensions r*p and $\underset{(pxc)}{\mathbf{Y}}$ of dimension p * c, the product, $\underset{(rxp)(pxc)}{\mathbf{X}\ \mathbf{Y}}$, is a r * c matrix where each element is the sum of the products of the rows of the first and the columns of the second. That is, the matrix $\underset{(rxc)}{\mathbf{XY}}$ has elements xy_{ij} where each

$$xy_{ij} = \sum_{k=1}^{n} x_{ik} * y_{kj}$$

The resulting x_{ij} cells of the product matrix are sums of the products of the column elements of the first matrix times the row elements of the second. There will be as many cell as there are rows of the first matrix and columns of the second matrix.

$$\underset{(rxc)}{\mathbf{XY}} = \xrightarrow{\begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{14} \\ x_{21} & x_{22} & x_{23} & x_{24} \end{pmatrix}} \downarrow \begin{pmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \\ y_{31} & y_{32} \\ y_{41} & y_{42} \end{pmatrix} = \begin{pmatrix} \sum_i^p x_{1i}y_{i1} & \sum_i^p x_{1i}y_{i2} \\ \sum_i^p x_{2i}y_{i1} & \sum_i^p x_{2i}y_{i2} \end{pmatrix}$$

It should be obvious that matrix multiplication is a very powerful operation, for it represents in one product the r * c summations taken over p observations.

E.2.2.1 Using matrix multiplication to find means and deviation scores

Matrix multiplication can be used with vectors as well as matrices. Consider the product of a vector of ones, **1**, and the matrix $\underset{(rxc)}{\mathbf{Xij}}$ with 6 rows of 4 columns. Call an individual element in this matrix x_{ij}. Then the sum for each column of the matrix is found multiplying the matrix by the "one" vector with **Xij**. Dividing each of these resulting sums by the number of rows (cases) yields the mean for each column. That is, find

$$\mathbf{1'Xij} = \sum_{i=1}^{n} X_{ij}$$

for the c columns, and then divide by the number (n) of rows. Note that the same result is found by the `colMeans(Xij)` function.

$$\mathbf{1}'\mathbf{Xij}\frac{1}{n} = \xrightarrow{\begin{pmatrix}1 & 1 & 1 & 1 & 1 & 1\end{pmatrix}}\downarrow \begin{pmatrix}1 & 7 & 13 & 19\\ 2 & 8 & 14 & 20\\ 3 & 9 & 15 & 21\\ 4 & 10 & 16 & 22\\ 5 & 11 & 17 & 23\\ 6 & 12 & 18 & 24\end{pmatrix}\frac{1}{4} = \begin{pmatrix}21 & 57 & 93 & 129\end{pmatrix}\frac{1}{4} = \begin{pmatrix}3.5 & 9.5 & 15.5 & 21.5\end{pmatrix}$$

We can use the `dim` function to find out how many cases (the number of rows) or the number of variables (number of columns). `dim` has two elements: dim(**Xij**)[1] = number of rows, dim(**Xij**)[2] is the number of columns.

```
> dim(Xij)

[1] 6 4

> one <- rep(1,dim(Xij)[1]) #a vector of 1s
> t(one)  %*% Xij  #find the column sum

     V1 V2 V3  V4
[1,] 21 57 93 129

> X.means <-  t(one) %*% Xij /dim(Xij)[1] #find the column average

  V1   V2   V3   V4
 3.5  9.5 15.5 21.5
```

A built in function to find the means of the columns is `colMeans`. (See `rowMeans` for the equivalent for rows.)

```
> colMeans(Xij)

  V1   V2   V3   V4
 3.5  9.5 15.5 21.5
```

To form a matrix of deviation scores, where the elements of each column are deviations from that column mean, it is necessary to either do the operation on the transpose of the **Xij** matrix, or to create a matrix of means by premultiplying the means vector by a vector of ones and subtracting this from the data matrix.

```
>  X.diff <- Xij - one %*% X.means

> X.diff
     V1   V2   V3   V4
S1 -2.5 -2.5 -2.5 -2.5
S2 -1.5 -1.5 -1.5 -1.5
S3 -0.5 -0.5 -0.5 -0.5
S4  0.5  0.5  0.5  0.5
S5  1.5  1.5  1.5  1.5
S6  2.5  2.5  2.5  2.5
```

This can also be done by using the `scale` function which will mean center each column and (by default) standardize by dividing by the standard deviation of each column.

E.2.2.2 Using matrix multiplication to find variances and covariances

Variances and covariances are measures of dispersion around the mean. We find these by first subtracting the means from all the observations. This means centered matrix is the original matrix minus a vector of means. To make a more interesting data set, randomly order (in this case, `sample` without replacement) from the items in **Xij** and then find the **X**.**means** and **X**.**diff** matrices.

```
> set.seed(42) #set random seed for a repeatable example
> Xij <- matrix(sample(Xij),ncol=4) #random sample from Xij
> rownames(Xij) <- paste("S", seq(1, dim(Xij)[1]), sep = "")
> colnames(Xij) <- paste("V", seq(1, dim(Xij)[2]), sep = "")
> Xij
```

```
   V1 V2 V3 V4
S1 22 14 12 15
S2 24  3 17  6
S3  7 11  5  4
S4 18 16  9 21
S5 13 23  8  2
S6 10 19  1 20
```

```
> X.means <-  t(one) %*% Xij /dim(Xij)[1] #find the column average
> X.diff <- Xij -t(one) %*% X.means
> X.diff
```

```
          V1          V2         V3        V4
S1  6.333333  -0.3333333  3.3333333  3.666667
S2  8.333333 -11.3333333  8.3333333 -5.333333
S3 -8.666667  -3.3333333 -3.6666667 -7.333333
S4  2.333333   1.6666667  0.3333333  9.666667
S5 -2.666667   8.6666667 -0.6666667 -9.333333
S6 -5.666667   4.6666667 -7.6666667  8.666667
```

Compare this result to just using the `scale` function to mean center the data:

`X.cen <- scale(Xij,scale=FALSE).`

To find the variance/covariance matrix, find the the matrix product of the means centered matrix **X**.**diff** with itself and divide by n-1. Compare this result to the result of the `cov` function (the normal way to find covariances). The differences between these two results is the rounding to whole numbers for the first, and to two decimals in the second.

```
> X.cov <- t(X.diff) %*% X.diff /(dim(X.diff)[1]-1)
> round(X.cov)
```

```
    V1  V2  V3  V4
V1  46 -23  34   8
V2 -23  48 -25  12
V3  34 -25  31 -12
V4   8  12 -12  70
```

```
> round(cov(Xij),2)

       V1     V2     V3     V4
V1  45.87 -22.67  33.67   8.13
V2 -22.67  47.87 -24.87  11.87
V3  33.67 -24.87  30.67 -12.47
V4   8.13  11.87 -12.47  70.27
```

E.2.3 Finding and using the diagonal

Some operations need to find just the *diagona*l of the matrix. For instance, the diagonal of the matrix **X.cov** (found above) contains the variances of the items. To extract just the diagonal, or create a matrix with a particular diagonal we use the `diag` command. We can convert the covariance matrix **X.cov** to a correlation matrix **X.cor** by pre and post multiplying the covariance matrix with a diagonal matrix containing the reciprocal of the standard deviations (square roots of the variances). Remember (Chapter 4 that the correlation, r_{xy}, is the ratio of the covariance to the squareroot of the product of the variances:

$$\frac{C_{xy}}{\sqrt{V_x V_y}}.$$

```
> X.var <- diag(X.cov)

      V1       V2       V3       V4
45.86667 47.86667 30.66667 70.26667

> sdi <- diag(1/sqrt(diag(X.cov)))
> rownames(sdi) <- colnames(sdi) <- colnames(X.cov)
> round(sdi, 2)

     V1   V2   V3   V4
V1 0.15 0.00 0.00 0.00
V2 0.00 0.14 0.00 0.00
V3 0.00 0.00 0.18 0.00
V4 0.00 0.00 0.00 0.12

> X.cor <- sdi %*% X.cov %*% sdi #pre and post multiply by 1/sd
> rownames(X.cor) <- colnames(X.cor) <- colnames(X.cov)
> round(X.cor, 2)

      V1    V2    V3    V4
V1  1.00 -0.48  0.90  0.14
V2 -0.48  1.00 -0.65  0.20
V3  0.90 -0.65  1.00 -0.27
V4  0.14  0.20 -0.27  1.00
```

Compare this to the standard command for finding correlations `cor`.

```
> round(cor(Xij), 2)
```

```
      V1    V2    V3    V4
V1  1.00 -0.48  0.90  0.14
V2 -0.48  1.00 -0.65  0.20
V3  0.90 -0.65  1.00 -0.27
V4  0.14  0.20 -0.27  1.00
```

E.2.4 The Identity Matrix

The *identity matrix* is merely that matrix, which when multiplied by another matrix, yields the other matrix. (The equivalent of 1 in normal arithmetic.) It is a diagonal matrix with 1 on the diagonal.

```
> I <- diag(1,nrow=dim(X.cov)[1])
```

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

E.2.5 Matrix Inversion

The *inverse* of a square matrix is the matrix equivalent of dividing by that matrix. That is, either pre or post multiplying a matrix by its inverse yields the identity matrix. The inverse is particularly important in multiple regression, for it allows us to solve for the beta weights.

Given the equation

$$\hat{\mathbf{y}} = \mathbf{bX} + c$$

we can solve for **b** by multiplying both sides of the equation by **X**' to form a square matrix $\mathbf{XX'}$ and then take the inverse of that square matrix:

$$\mathbf{yX'} = \mathbf{bXX'} <=> \mathbf{b} = \mathbf{yX'}(\mathbf{XX'})^{-1}$$

We can find the inverse by using the `solve` function. To show that $\mathbf{XX}^{-1} = \mathbf{X}^{-1}\mathbf{X} = \mathbf{I}$, we do the multiplication.

```
> X.inv <- solve(X.cov)
```

```
            V1          V2          V3          V4
V1  0.19638636  0.01817060  0.06024476 -0.07130491
V2  0.01817060  0.12828756  0.03787166 -0.00924279
V3  0.06024476  0.03787166  0.10707738 -0.03402838
V4 -0.07130491 -0.00924279 -0.03402838  0.11504850
```

```
> round(X.cov %*% X.inv, 2)
```

```
   V1 V2 V3 V4
V1  1  0  0  0
V2  0  1  0  0
V3  0  0  1  0
V4  0  0  0  1
```

```
> round(X.inv %*% X.cov, 2)
```

```
   V1 V2 V3 V4
V1  1  0  0  0
V2  0  1  0  0
V3  0  0  1  0
V4  0  0  0  1
```

There are multiple ways of finding the matrix inverse, `solve` is just one of them. Appendix F goes into more detail about how inverses are used in systems of simultaneous equations. Chapter 5 considers the use of matrix operations in multiple regression.

E.2.6 Eigenvalues and Eigenvectors

The *eigenvectors* of a matrix are said to provide a *basis space* for the matrix. This is a set of orthogonal vectors which when multiplied by the appropriate scaling vector of *eigenvalues* will reproduce the matrix.

Given a $n*n$ matrix $\mathbf{R}$, each eigenvector solves the equation

$$\mathbf{x_i R} = \lambda_i \mathbf{x_i}$$

and the set of n eigenvectors are solutions to the equation

$$\mathbf{RX} = \mathbf{X}\lambda$$

where $\mathbf{X}$ is a matrix of orthogonal eigenvectors and λ is a diagonal matrix of the the eigenvalues, λ_i. Then

$$\mathbf{x_i R} - \lambda_i \mathbf{XI} = 0 <=> \mathbf{x_i}(\mathbf{R} - \lambda_i \mathbf{I}) = 0$$

Finding the eigenvectors and values is computationally tedious, but may be done using the `eigen` function which uses a **QR** *decomposition* of the matrix. That the vectors making up $\mathbf{X}$ are orthogonal means that

$$\mathbf{XX'} = \mathbf{I}$$

and that

$$\mathbf{R} = \mathbf{X}\lambda\mathbf{X'}.$$

That is, it is possible to recreate the correlation matrix $\mathbf{R}$ in terms of an orthogonal set of vectors (the *eigenvectors*) scaled by their associated *eigenvalues*. (See 9.1.1 and Table 9.2 for an example of an eigenvalue decomposition using the `eigen` function.

The sum of the eigenvalues for a correlation matrix is the *rank* of the correlation matrix. The product of the eigenvalues is the *determinant* of the matrix.

E.2.7 Determinants

The *determinant* of an n * n correlation matrix may be thought of as the proportion of the possible n-space spanned by the variable space and is sometimes called the *generalized variance* of the matrix. As such, it can also be considered as the volume of the variable space. If the correlation matrix is thought of a representing vectors within a n dimensional space, then the square roots of the eigenvalues are the lengths of the axes of that space. The product of these, the determinant, is then the volume of the space. It will be a maximum when the axes are all of unit length and be zero if at least one axis is zero. Think of a three dimensional sphere (and then generalize to a n dimensional hypersphere.) If it is squashed in a way that preserves the sum of the lengths of the axes, then volume of the *oblate hyper sphere* will be reduced.

The determinant is an inverse measure of the redundancy of the matrix. The smaller the determinant, the more variables in the matrix are measuring the same thing (are correlated). The determinant of the identity matrix is 1, the determinant of a matrix with at least two perfectly correlated (linearly dependent) rows or columns will be 0. If the matrix is transformed into a lower diagonal matrix, the determinant is the product of the diagonals. The determinant of a n * n square matrix, **R** is also the product of the n *eigenvalues* of that matrix.

$$det(\mathbf{R}) = \|\mathbf{R}\| = \Pi_{i=1}^{n} \lambda_i \tag{E.5}$$

and the *characteristic equation* for a square matrix, **X**, is

$$\|\mathbf{X} - \lambda \mathbf{I}\| = 0$$

where λ is an eigenvalue of **X**.

The determinant may be found by the `det` function. The determinant may be used in estimating the goodness of fit of a particular model to the data, for when the model fits perfectly, then the inverse of the model times the data will be an identity matrix and the determinant will be one (See Chapter 9 for much more detail.)

E.3 Matrix operations for data manipulation

Using the basic matrix operations of addition and multiplication allow for easy manipulation of data. In particular, finding subsets of data, scoring multiple scales for one set of items, or finding correlations and reliabilities of composite scales are all operations that are easy to do with matrix operations.

In the next example we consider 5 extraversion items for 200 subjects collected as part of the Synthetic Aperture Personality Assessment project. The items are taken from the International Personality Item Pool (ipip.ori.org) and are downloaded from a remote server. A larger data set taken from the SAPA project is included as the `bfi` data set in `psych`. We use this remote set to demonstrate the ability to read data from the web. Because the first item is an identification number, we drop the first column

```
> datafilename = "http://personality-project.org/R/datasets/extraversion.items.txt"
> items = read.table(datafilename, header = TRUE)
> items <- items[, -1]
> dim(items)
```

```
[1] 200   5
```

We first use functions from the `psych` package to describe these data both numerically and graphically.

```
> library(psych)
```

```
[1] "psych"     "stats"     "graphics"  "grDevices" "utils"     "datasets"  "methods"   "base"
```

```
> describe(items)
```

```
       var   n mean   sd median trimmed  mad min max range  skew kurtosis   se
q_262    1 200 3.07 1.49      3    3.01 1.48   1   6     5  0.23    -0.90 0.11
q_1480   2 200 2.88 1.38      3    2.83 1.48   0   6     6  0.21    -0.85 0.10
q_819    3 200 4.57 1.23      5    4.71 1.48   0   6     6 -1.00     0.71 0.09
q_1180   4 200 3.29 1.49      4    3.30 1.48   0   6     6 -0.09    -0.90 0.11
q_1742   5 200 4.38 1.44      5    4.54 1.48   0   6     6 -0.72    -0.25 0.10
```

```
> pairs.panels(items)
```

We can form two composite scales, one made up of the first 3 items, the other made up of the last 2 items. Note that the second (q1480) and fourth (q1180) are negatively correlated with the remaining 3 items. This implies that we should reverse these items before scoring.

To form the composite scales, reverse the items, and find the covariances and then correlations between the scales may all be done by matrix operations on either the items or on the covariances between the items. In either case, we want to define a "keys" matrix describing which items to combine on which scale. The correlations are, of course, merely the covariances divided by the square root of the variances.

E.3.1 Matrix operations on the raw data

```
> keys <- matrix(c(1, -1, 1, 0, 0, 0, 0, 0, -1, 1), ncol = 2) #specify
> keys      # and show the keys matrix
> X <- as.matrix(items) #matrix operations require matrices
> X.ij <- X %*% keys  #this results in the scale scores
> n <- dim(X.ij)[1]     # how many subjects?
> one <- rep(1, dim(X.ij)[1])
> X.means <- t(one) %*% X.ij/n
> X.cov <- t(X.ij - one %*% X.means) %*% (X.ij - one %*% X.means)/(n - 1)
> round(X.cov, 2)
```

```
 > keys
     [,1] [,2]
[1,]    1    0
[2,]   -1    0
[3,]    1    0
```

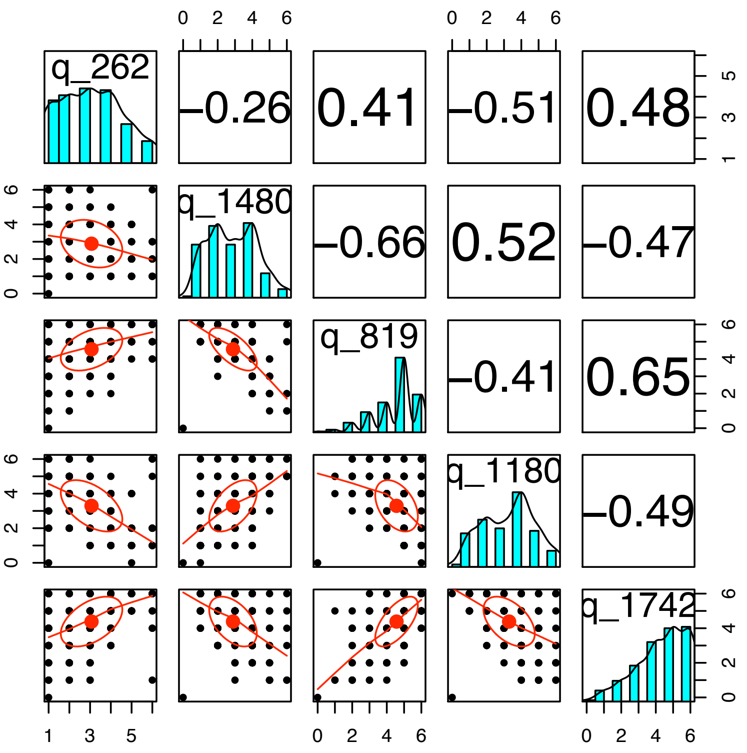

Fig. E.1 Scatter plot matrix (SPLOM) of 5 extraversion items for 200 subjects.

```
[4,]     0   -1
[5,]     0    1

      [,1] [,2]
[1,] 10.45 6.09
[2,]  6.09 6.37
```

```
> X.sd <- diag(1/sqrt(diag(X.cov)))
> X.cor <- t(X.sd) %*% X.cov %*% (X.sd)
> round(X.cor, 2)
```

```
     [,1] [,2]
[1,] 1.00 0.75
[2,] 0.75 1.00
```

E.3.2 Matrix operations on the correlation matrix

The previous example found the correlations and covariances of the scales based upon the raw data. We can also do these operations on the correlation matrix.

```
> keys <- matrix(c(1, -1, 1, 0, 0, 0, 0, 0, -1, 1), ncol = 2)
> X.cor <- cor(X)
> round(X.cor, 2)
```

```
       q_262 q_1480 q_819 q_1180 q_1742
q_262   1.00  -0.26  0.41  -0.51   0.48
q_1480 -0.26   1.00 -0.66   0.52  -0.47
q_819   0.41  -0.66  1.00  -0.41   0.65
q_1180 -0.51   0.52 -0.41   1.00  -0.49
q_1742  0.48  -0.47  0.65  -0.49   1.00
```

```
> X.cov <- t(keys) %*% X.cor %*% keys
> X.sd <- diag(1/sqrt(diag(X.cov)))
> X.cor <- t(X.sd) %*% X.cov %*% (X.sd)
> keys
```

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

```
> round(X.cov, 2)
```

```
     [,1] [,2]
[1,] 5.66 3.05
[2,] 3.05 2.97
```

```
> round(X.cor, 2)
```

```
     [,1] [,2]
[1,] 1.00 0.74
[2,] 0.74 1.00
```

E.3.3 Using matrices to find test reliability

The reliability of a test may be thought of as the correlation of the test with a test just like it. One conventional estimate of reliability, based upon the concepts from domain sampling theory, is *coefficient alpha* (*alpha*). For a test with just one factor, α is an estimate of the amount of the test variance due to that factor. However, if there are multiple factors in the test, α neither estimates how much the variance of the test is due to one, general factor, nor does it estimate the correlation of the test with another test just like it. (See Zinbarg et al. (2005) for a discussion of alternative estimates of reliabiity.)

Given either a covariance or correlation matrix of items, α may be found by simple matrix operations:

1) **V**= the correlation or covariance matrix
2) Let V_t = the total variance = the sum of all the items in the correlation matrix for that scale.
3) Let n = number of items in the scale
4)

$$\alpha = \frac{V_t - diag(V)}{V_t} * \frac{n}{n-1}$$

To demonstrate the use of matrices to find coefficient α, consider the five items measuring extraversion taken from the International Personality Item Pool. Two of the items need to be weighted negatively (reverse scored).

Alpha may be found from either the correlation matrix (standardized alpha) or the covariance matrix (raw alpha). In the case of standardized alpha, the diag(V) is the same as the number of items. Using a key matrix, we can find the reliability of 3 different scales, the first is made up of the first 3 items, the second of the last 2, and the third is made up of all the items.

```
> datafilename <- "http://personality-project.org/R/datasets/extraversion.items.txt"
> items = read.table(datafilename, header = TRUE)
> items <- items[, -1]
> key <- matrix(c(1, -1, 1, 0, 0, 0, 0, 0, -1, 1, 1, -1, 1, -1, 1), ncol = 3)
 > colnames(key) <- c("V1-3", "V4-5", "V1-5")
 > rownames(key) <- colnames(items)

> key

      V1-3 V4-5 V1-5
q_262      1    0    1
q_1480    -1    0   -1
q_819      1    0    1
q_1180     0   -1   -1
q_1742     0    1    1


> raw.r <- cor(items)   #find the correlations -- could have been done with matrix operations
> V <- t(key) %*% raw.r %*% key
> rownames(V) <- colnames(V) <- c("V1-3", "V4-5", "V1-5")
> round(V, 2)

     V1-3 V4-5  V1-5
V1-3 5.66 3.05  8.72
V4-5 3.05 2.97  6.03
V1-5 8.72 6.03 14.75

> n <- diag(t(key) %*% key)
> alpha <- (diag(V) - n)/(diag(V)) * (n/(n - 1))
> round(alpha, 2)

V1-3 V4-5 V1-5
0.71 0.66 0.83
```

As would be expected, there are multiple functions in R to score scales and find coefficient alpha this way. In `psych` the `score.items` function will work on raw data, and `cluster.cor` function for correlations matrices.

E.4 Multiple correlation

Given a set of n predictors of a criterion variable, what is the optimal weighting of the n predictors? This is, of course, the problem of multiple correlation or multiple regression. Although we would normally use the **linear model (lm)** function to solve this problem, we can also do it from the raw data or from a matrix of covariances or correlations by using matrix operations and the `solve` function.

Consider the data set, **X**, created in section E.2.1. If we want to predict **V4** as a function of the first three variables, we can do so three different ways, using the raw data, using deviation scores of the raw data, or with the correlation matrix of the data.

For simplicity, lets relable V_4 to be **Y** and V_1 ... V_3 to be X_1 ...X_3 and then define **X** as the first three columns and **Y** as the last column:

```
    X1 X2 X3
S1   9  4  9
S2   9  7  1
S3   2  9  9
S4   8  2  9
S5   6  4  0
S6   5  9  5
S7   7  9  3
S8   1  1  9
S9   6  4  4
S10  7  5  8

 S1  S2  S3  S4  S5  S6  S7  S8  S9 S10
  7   8   3   6   0   8   0   2   9   6
```

E.4.1 Data level analyses

At the data level, we can work with the raw data matrix **X**, or convert these to deviation scores (**X.dev**) by subtracting the means from all elements of **X**. At the raw data level we have

$$_m\hat{Y}_1 =_m X_{nn}\beta_1 +_m \varepsilon_1 \tag{E.6}$$

and we can solve for $\mathbf{_n\beta_1}$ by pre-multiplying by $\mathbf{_nX'_m}$ (thus making the matrix on the right side of the equation into a square matrix so that we can multiply through by an inverse. See section E.2.5)

$$_nX'_{mm}\hat{Y}_1 =_n X'_{mm}X_{nn}\beta_1 +_m \varepsilon_1 \tag{E.7}$$

and then solving for beta by pre-multiplying both sides of the equation by $(\mathbf{XX'})^{\mathbf{-1}}$

$$\beta = (XX')^{-1}X'Y \tag{E.8}$$

These beta weights will be the weights with no intercept. Compare this solution to the one using the `lm` function with the intercept removed:

```
> beta <- solve(t(X) %*% X) %*% (t(X) %*% Y)
> round(beta, 2)

   [,1]
X1 0.56
X2 0.03
X3 0.25

> lm(Y ~ -1 + X)

Call:
lm(formula = Y ~ -1 + X)

Coefficients:
    XX1      XX2      XX3
0.56002  0.03248  0.24723
```

If we want to find the intercept as well, we can add a column of 1's to the **X** matrix. This matches the normal `lm` result.

```
> one <- rep(1, dim(X)[1])
> X <- cbind(one, X)
> print(X)

    one X1 X2 X3
S1    1  9  4  9
S2    1  9  7  1
S3    1  2  9  9
S4    1  8  2  9
S5    1  6  4  0
S6    1  5  9  5
S7    1  7  9  3
S8    1  1  1  9
S9    1  6  4  4
S10   1  7  5  8

> beta <- solve(t(X) %*% X) %*% (t(X) %*% Y)
> round(beta, 2)

     [,1]
one -0.94
X1   0.62
X2   0.08
X3   0.30

> lm(Y ~ X)
```

```
Call:
lm(formula = Y ~ X)

Coefficients:
(Intercept)         Xone          XX1          XX2          XX3
   -0.93843           NA      0.61978      0.08034      0.29577
```

We can do the same analysis with deviation scores. Let X.dev be a matrix of deviation scores, then can write the equation

$$\hat{Y} = X\beta + \varepsilon \tag{E.9}$$

and solve for

$$\beta = (X.devX.dev')^{-1}X.dev'Y. \tag{E.10}$$

(We don't need to worry about the sample size here because n cancels out of the equation).

At the structure level, the covariance matrix = $\mathbf{XX'}$/(n-1) and $\mathbf{X'Y}$/(n-1) may be replaced by correlation matrices by pre and post multiplying by a diagonal matrix of 1/sds) with r_{xy} and we then solve the equation

$$\beta = R^{-1}r_{xy} \tag{E.11}$$

Consider the set of 3 variables with intercorrelations (R)

```
     x1   x2   x3
x1 1.00 0.56 0.48
x2 0.56 1.00 0.42
x3 0.48 0.42 1.00
```

and correlations of x with y (r_{xy})

```
   x1   x2  x3
y 0.4 0.35 0.3
```

From the correlation matrix, we can use the solve function to find the optimal beta weights.

```
> R <- matrix(c(1, 0.56, 0.48, 0.56, 1, 0.42, 0.48, 0.42, 1), ncol = 3)
> rxy <- matrix(c(0.4, 0.35, 0.3), ncol = 1)
> colnames(R) <- rownames(R) <- c("x1", "x2", "x3")
> rownames(rxy) <- c("x1", "x2", "x3")
> colnames(rxy) <- "y"
> beta <- solve(R, rxy)
> round(beta, 2)

      y
x1 0.26
x2 0.16
x3 0.11
```

Using the correlation matrix to do multiple R is particular useful when the correlation or covariance matrix is from a published source, or if, for some reason, the original data are not available. The `mat.regress` function in `psych` finds multiple R this way. Unfortunately, by not having the raw data, many of the error diagnostics are not available.

Appendix F
More on Matrices

F.1 Multiple regression as a system of simultaneous equations

Many problems in data analysis require solving a system of simultaneous equations. For instance, in multiple regression with two predictors and one criterion with a set of correlations of:

$$\left\{ \begin{matrix} r_{x1x1} & r_{x1x2} & r_{x1y} \\ r_{x1x2} & r_{x2x2} & r_{x2y} \\ r_{x1y} & r_{x2y} & r_{yy} \end{matrix} \right\} \tag{F.1}$$

we want to find the find weights, β_i, that when multiplied by x_1 and x_2 maximize the correlations with y. That is, we want to solve the two simultaneous equations

$$\left\{ \begin{matrix} r_{x1x1}\beta_1 + r_{x1x2}\beta_2 = r_{x1y} \\ r_{x1x2}\beta_1 + r_{x2x2}\beta_2 = r_{x2y} \end{matrix} \right\}. \tag{F.2}$$

We can directly solve these two equations by adding and subtracting terms to the two such that we end up with a solution to the first in terms of β_1 and to the second in terms of β_2:

$$\left\{ \begin{matrix} \beta_1 + r_{x1x2}\beta_2/r_{x1x1} = r_{x1y}/r_{x1x1} \\ r_{x1x2}\beta_1/r_{x2x2} + \beta_2 = r_{x2y}/r_{x2x2} \end{matrix} \right\}$$

which becomes

$$\left\{ \begin{matrix} \beta_1 = (r_{x1y} - r_{x1x2}\beta_2)/r_{x1x1} \\ \beta_2 = (r_{x2y} - r_{x1x2}\beta_1)/r_{x2x2} \end{matrix} \right\} \tag{F.3}$$

Substituting the second row of (F.3) into the first row, and vice versa we find

$$\left\{ \begin{matrix} \beta_1 = (r_{x1y} - r_{x1x2}(r_{x2y} - r_{x1x2}\beta_1)/r_{x2x2})/r_{x1x1} \\ \beta_2 = (r_{x2y} - r_{x1x2}(r_{x1y} - r_{x1x2}\beta_2)/r_{x1x1})/r_{x2x2} \end{matrix} \right\}$$

Collecting terms, we find:

$$\left\{ \begin{matrix} \beta_1 r_{x1x1} r_{x2x2} = (r_{x1y} r_{x2x2} - r_{x1x2}(r_{x2y} - r_{x1x2}\beta_1)) \\ \beta_2 r_{x2x2} r_{x1x1} = (r_{x2y} r_{x1x1} - r_{x1x2}(r_{x1y} - r_{x1x2}\beta_2)) \end{matrix} \right\}$$

and rearranging once again:

$$\left\{ \begin{array}{c} \beta_1 r_{x1x1} r_{x2x2} - r^2_{x1x2}\beta_1 = (r_{x1y} r_{x2x2} - r_{x1x2}(r_{x2y}) \\ \beta_2 r_{x1x1} r_{x2x2} - r^2_{x1x2}\beta_2 = (r_{x2y} r_{x1x1} - r_{x1x2}(r_{x1y} \end{array} \right\}$$

Struggling on:

$$\left\{ \begin{array}{c} \beta_1 (r_{x1x1} r_{x2x2} - r^2_{x1x2}) = r_{x1y} r_{x2x2} - r_{x1x2} r_{x2y} \\ \beta_2 (r_{x1x1} r_{x2x2} - r^2_{x1x2}) = r_{x2y} r_{x1x1} - r_{x1x2} r_{x1y} \end{array} \right\}$$

And finally:

$$\left\{ \begin{array}{c} \beta_1 = (r_{x1y} r_{x2x2} - r_{x1x2} r_{x2y})/(r_{x1x1} r_{x2x2} - r^2_{x1x2}) \\ \beta_2 = (r_{x2y} r_{x1x1} - r_{x1x2} r_{x1y})/(r_{x1x1} r_{x2x2} - r^2_{x1x2}) \end{array} \right\}$$

F.2 Matrix representation of simultaneous equation

Alternatively, these two equations (F.2) may be represented as the product of a vector of unknowns (the βs) and a matrix of coefficients of the predictors (the r_{xi}'s) and a matrix of coefficients for the criterion (rx_iy): [1]

$$(\beta_1 \beta_2) \begin{pmatrix} r_{x1x1} & r_{x1x2} \\ r_{x1x2} & r_{x2x2} \end{pmatrix} = (r_{x1y} \quad r_{x2x2}) \tag{F.4}$$

If we let $\beta = (\beta_1 \beta_2)$, R = $\begin{pmatrix} r_{x1x1} & r_{x1x2} \\ r_{x1x2} & r_{x2x2} \end{pmatrix}$ and $r_{xy} = (r_{x1y} \quad r_{x2x2})$ then equation (F.4) becomes

$$\beta R = r_{xy} \tag{F.5}$$

and we can solve (F.5) for β by multiplying both sides by the inverse of R.

$$\beta = \beta R R^{-1} = r_{xy} R^{-1}$$

F.2.1 Finding the inverse of a 2 x 2 matrix

But, how do we find the inverse (R^{-1})? As an example we solve the inverse of a 2 x2 matrix, but the technique may be applied to a matrix of any size. First, define the identity matrix, I, as

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

and then the equation

$$R = IR$$

[1] See Appendix -1 for a detailed discussion of how this is done in practice with some "real" data using the statistical program, R. In R, the inverse of a square matrix, X, is found by the solve function: X.inv <- solve(X)

may be represented as

$$\begin{pmatrix} r_{x1x1} & r_{x1x2} \\ r_{x1x2} & r_{x2x2} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} r_{x1x1} & r_{x1x2} \\ r_{x1x2} & r_{x2x2} \end{pmatrix}$$

Dropping the x subscript (for notational simplicity) we have

$$\begin{pmatrix} r_{11} & r_{12} \\ r_{12} & r_{22} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} \\ r_{12} & r_{22} \end{pmatrix} \tag{F.6}$$

We may multiply both sides of equation (F.6) by a simple transformation matrix (T) without changing the equality. If we do this repeatedly until the left hand side of equation (F.6) is the identity matrix, then the first matrix on the right hand side will be the inverse of R. We do this in several steps to show the process.

Let

$$T_1 = \begin{pmatrix} \frac{1}{r_{11}} & 0 \\ 0 & \frac{1}{r_{22}} \end{pmatrix}$$

then we multiply both sides of equation (F.6) by T_1 in order to make the diagonal elements of the left hand equation = 1 and we have

$$T_1 R = T_1 I R \tag{F.7}$$

$$\begin{pmatrix} 1 & \frac{r_{12}}{r_{11}} \\ \frac{r_{12}}{r_{22}} & 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{r_{11}} & 0 \\ 0 & \frac{1}{r_{22}} \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} \\ r_{12} & r_{22} \end{pmatrix} \tag{F.8}$$

Then, by letting

$$T_2 = \begin{pmatrix} 1 & 0 \\ -\frac{r_{12}}{r_{22}} & 1 \end{pmatrix}$$

and multiplying T_2 times both sides of equation (F.8) we can make the lower off diagonal element = 0. (Functionally, we are subtracting $\frac{r_{12}}{r_{22}}$ times the first row from the second row).

$$\begin{pmatrix} 1 & \frac{r_{12}}{r_{11}} \\ 0 & 1 - \frac{r_{12}^2}{r_{11}r_{22}} \end{pmatrix} = \begin{pmatrix} 1 & \frac{r_{12}}{r_{11}} \\ 0 & \frac{r_{11}r_{22} - r_{12}^2}{r_{11}r_{22}} \end{pmatrix} = \begin{pmatrix} \frac{1}{r_{11}} & 0 \\ -\frac{r_{12}}{r_{11}r_{22}} & \frac{1}{r_{22}} \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} \\ r_{12} & r_{22} \end{pmatrix} \tag{F.9}$$

Then, in order to make the diagonal elements all = 1 , we let

$$T_3 = \begin{pmatrix} 1 & 0 \\ 0 & \frac{r_{11}r_{22}}{r_{11}r_{22} - r_{12}^2} \end{pmatrix}$$

and multiplying T_3 times both sides of equation (F.9) we have

$$\begin{pmatrix} 1 & \frac{r_{12}}{r_{11}} \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{r_{11}} & 0 \\ -\frac{r_{12}}{r_{11}r_{22} - r_{12}^2} & \frac{r_{11}}{r_{11}r_{22} - r_{12}^2} \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} \\ r_{12} & r_{22} \end{pmatrix} \tag{F.10}$$

Then, to make the upper off diagonal element = 0, we let

$$T_4 = \begin{pmatrix} 1 & -\frac{r_{12}}{r_{11}} \\ 0 & 1 \end{pmatrix}$$

and multiplying T_4 times both sides of equation (F.10) we have

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{r_{22}}{r_{11}r_{22}-r_{12}^2} & -\frac{r_{12}}{r_{11}r_{22}-r_{12}^2} \\ -\frac{r_{12}}{r_{11}r_{22}-r_{12}^2} & \frac{r_{11}}{r_{11}r_{22}-r_{12}^2} \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} \\ r_{12} & r_{22} \end{pmatrix}$$

That is, the inverse of our original matrix, R, is

$$R^{-1} = \begin{pmatrix} \frac{r_{22}}{r_{11}r_{22}-r_{12}^2} & -\frac{r_{12}}{r_{11}r_{22}-r_{12}^2} \\ -\frac{r_{12}}{r_{11}r_{22}-r_{12}^2} & \frac{r_{11}}{r_{11}r_{22}-r_{12}^2} \end{pmatrix} \tag{F.11}$$

The previous example was drawn out to be easier to follow, and it would be possible to combine several steps together. The important point is that by successively multiplying equation F.6 by a series of transformation matrices, we have found the inverse of the original matrix.

$$T_4T_3T_2T_1R = T_4T_3T_2T_1IR$$

or, in other words

$$T_4T_3T_2T_1R = I = R^{-1}R$$

$$T_4T_3T_2T_1I = R^{-1} \tag{F.12}$$

F.3 A numerical example of finding the inverse

Consider the following Covariance matrix, C, and set of transform matrices, T1 ... T4, as derived before.

$$C = \begin{pmatrix} 3 & 2 \\ 2 & 4 \end{pmatrix}$$

The first transformation is to change the diagonal elements to 1 by dividing all elements by the reciprocal of the diagonal elements. (This is two operations, the first divides elements of the first row by 3, the second divides elements of the second row by 4).

$$T_1C = \begin{pmatrix} .33 & .00 \\ .00 & .25 \end{pmatrix} \begin{pmatrix} 3 & 2 \\ 2 & 4 \end{pmatrix} = \begin{pmatrix} 1.0 & .667 \\ .5 & 1 \end{pmatrix}$$

The next operation is to make the lower off diagonal element 0 by subtracting .5 times the first row from the second row.

$$T_2T_1C = \begin{pmatrix} 1.0 & 0 \\ -.5 & 1 \end{pmatrix} \begin{pmatrix} .33 & .00 \\ .00 & .25 \end{pmatrix} \begin{pmatrix} 3 & 2 \\ 2 & 4 \end{pmatrix} = \begin{pmatrix} 1.0 & .667 \\ 0 & .667 \end{pmatrix}$$

Then make the make the diagonals 1 again by multiplying elements of the second by 1.5 (this could be combined with the next operation).

$$T_3T_2T_1C = \begin{pmatrix} 1.0 & 0 \\ 0 & 1.5 \end{pmatrix} \begin{pmatrix} 1.0 & 0 \\ -.5 & 1 \end{pmatrix} \begin{pmatrix} .33 & .00 \\ .00 & .25 \end{pmatrix} \begin{pmatrix} 3 & 2 \\ 2 & 4 \end{pmatrix} = \begin{pmatrix} 1.0 & .67 \\ 0 & 1.0 \end{pmatrix}$$

Now multiply the second row by -.67 and add to the first row. The set of products has created the identify matrix.

$$T_4T_3T_2T_1C = \begin{pmatrix} 1 & -.67 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1.0 & 0 \\ 0 & 1.5 \end{pmatrix} \begin{pmatrix} 1.0 & 0 \\ -.5 & 1 \end{pmatrix} \begin{pmatrix} .33 & .00 \\ .00 & .25 \end{pmatrix} \begin{pmatrix} 3 & 2 \\ 2 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

As shown in equation F.12, if apply this same set of transformations to the identity matrix, I, we find the inverse of R

$$T_4T_3T_2T_1I = \begin{pmatrix} 1 & -.67 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1.0 & 0 \\ 0 & 1.5 \end{pmatrix} \begin{pmatrix} 1.0 & 0 \\ -.5 & 1 \end{pmatrix} \begin{pmatrix} .33 & .00 \\ .00 & .25 \end{pmatrix} \begin{pmatrix} 3 & 2 \\ 2 & 4 \end{pmatrix} = \begin{pmatrix} .5 & -.250 \\ -.25 & .375 \end{pmatrix}$$

That is,

$$C^{-1} = \begin{pmatrix} .5 & -.250 \\ -.25 & .375 \end{pmatrix}$$

We confirm this by multiplying

$$CC^{-1} = \begin{pmatrix} 3 & 2 \\ 2 & 4 \end{pmatrix} \begin{pmatrix} .5 & -.250 \\ -.25 & .375 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Of course, a much simpler technique is to simply enter the original matrix into R and use the solve function:

```
>C <- matrix(c(3,2,2,4),byrow=TRUE,nrow=2)
> C
     [,1] [,2]
[1,]    3    2
[2,]    2    4
> solve(C)
      [,1]   [,2]
[1,]  0.50 -0.250
[2,] -0.25  0.375
```

F.4 Examples of inverse matrices

F.4.1 Inverse of an identity matrix

The inverse of the identity matrix is just the identity matrix:

```
I
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    0    1    0    0
[3,]    0    0    1    0
[4,]    0    0    0    1
> solve (I)
```

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

F.4.2 The effect of correlation size on the inverse

As the correlations in the matrix become larger, the elements of the inverse become disproportionally larger. This is shown on the next page for matrices of size 2 and 3 with correlations ranging from 0 to .99.

The effect of multicollinearity is not particularly surprising when we examine equation (F.11) and notice that in the two by two case, the elements are divided by $r_{11}r_{22} - r_{12}^2$. As r_{12}^2 approaches $r_{11}r_{22}$, this ratio will tend towards ∞.

Because the inverse is used in estimation of the linear regression weights, as the correlations between the predictors increases, the elements of the inverse grow very large and small variations in the pattern of predictors will lead to large variations in the beta weights.

```
Original matrix                        Inverse of Matrix
> a                                    > round(solve(a),2)
     [,1] [,2]                               [,1]  [,2]
[1,]  1.0  0.5                         [1,]  1.33 -0.67
[2,]  0.5  1.0                         [2,] -0.67  1.33
> b                                    > round(solve(b),2)
     [,1] [,2]                               [,1]  [,2]
[1,]  1.0  0.8                         [1,]  2.78 -2.22
[2,]  0.8  1.0                         [2,] -2.22  2.78
> c                                    > round(solve(c),2)
     [,1] [,2]                               [,1]  [,2]
[1,]  1.0  0.9                         [1,]  5.26 -4.74
[2,]  0.9  1.0                         [2,] -4.74  5.26

> A                                    > round(solve(A),2)
     [,1] [,2] [,3]                         [,1] [,2] [,3]
[1,]    1    0    0                    [1,]    1    0    0
[2,]    0    1    0                    [2,]    0    1    0
[3,]    0    0    1                    [3,]    0    0    1
> B                                    > round(solve(B),2)
     [,1] [,2] [,3]                          [,1]  [,2]  [,3]
[1,]  1.0  0.0  0.5                    [1,]  1.38  0.23 -0.76
[2,]  0.0  1.0  0.3                    [2,]  0.23  1.14 -0.45
[3,]  0.5  0.3  1.0                    [3,] -0.76 -0.45  1.52
C                                      > round(solve(C),2)
     [,1] [,2] [,3]                         [,1]  [,2]  [,3]
[1,]  1.0  0.8  0.5                    [1,]  3.5 -2.50 -1.00
[2,]  0.8  1.0  0.3                    [2,] -2.5  2.88  0.38
[3,]  0.5  0.3  1.0                    [3,] -1.0  0.38  1.38
> D                                    > round(solve(D),2)
     [,1] [,2] [,3]                          [,1]  [,2]  [,3]
[1,]  1.0  0.9  0.5                    [1,]  7.58 -6.25 -1.92
[2,]  0.9  1.0  0.3                    [2,] -6.25  6.25  1.25
[3,]  0.5  0.3  1.0                    [3,] -1.92  1.25  1.58
> E                                    > round(solve(E),2)
     [,1] [,2] [,3]                           [,1]   [,2]  [,3]
[1,] 1.00 0.95  0.5                    [1,]  21.41 -18.82 -5.06
[2,] 0.95 1.00  0.3                    [2,] -18.82  17.65  4.12
[3,] 0.50 0.30  1.0                    [3,]  -5.06   4.12  2.29
> F                                    > round(solve(F),2)
     [,1] [,2] [,3]                           [,1]   [,2]  [,3]
[1,] 1.00 0.99  0.5                    [1,] -39.39  36.36  8.79
[2,] 0.99 1.00  0.3                    [2,]  36.36 -32.47 -8.44
[3,] 0.50 0.30  1.0                    [3,]   8.79  -8.44 -0.86
```