Working With Matrices in R

To create a vector in R, there are several approaches. Let's suppose we want to name the vector `vec`.
We can use the concatenate function to make an ordered list of numbers, e.g.,

```
> vec <- c( 1.9, -2.8, 5.6, 0, 3.4, -4.2 )
```

Then the vector looks like:

```
> vec
[1]  1.9 -2.8  5.6  0.0  3.4 -4.2
```

We can use the colon operator to make a sequence of integers, e.g.,

```
> vec <- -2:5
```

Then the vector looks like:

```
> vec
[1] -2 -1  0  1  2  3  4  5
```

Or we can make the sequence run the other way, e.g.,

```
> vec <- 5:-2
```

Then we get:

```
> vec
[1]  5  4  3  2  1  0 -1 -2
```

We can use the sequence function. We have to give a starting value, and ending value, and an increment value, e.g.,

```
> vec <- seq( -3.1, 2.2, by=0.1 )
```

Then we get:

```
> vec
 [1] -3.1 -3.0 -2.9 -2.8 -2.7 -2.6 -2.5 -2.4 -2.3 -2.2 -2.1 -2.0 -1.9 -1.8 -1.7
[16] -1.6 -1.5 -1.4 -1.3 -1.2 -1.1 -1.0 -0.9 -0.8 -0.7 -0.6 -0.5 -0.4 -0.3 -0.2
[31] -0.1  0.0  0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  1.0  1.1  1.2  1.3
[46]  1.4  1.5  1.6  1.7  1.8  1.9  2.0  2.1  2.2
```

Note that R wraps the vector around to four rows, and at the beginning of each row gives you the position in the vector of the first element of the row. We can run a sequence backward, from a larger value to a smaller value, but we must use a negative increment value.
We can use the replicate function to a get a vector whose elements are all the same, e.g.,

```
> vec <- rep( 1, 8 )
```

Then we get:

```
> vec
[1] 1 1 1 1 1 1 1 1
```

That is, we get 1 replicated 8 times.
We can concatenate two or more vectors to make a larger vector, e.g.,

```
> vec <- c( rep(2,4), 1:3, c(8,1,9) )
```

Then we get

```
> vec
[1] 2 2 2 2 1 2 3 8 1 9
```

We can take one of the columns from a data table and make it a vector, e.g.,

```
> vec <- Data$ACT
```

This will result in a vector consisting of the 120 ACT scores from the dataset `Data` we used for Problem 1.19 of the text.
If we want to know how many elements are in the vector, we use the length function, e.g.,

```
> length(vec)
```

If we want to reference an element in the vector, we use square brackets, e.g.,

```
> vec[5]
```

This will give us the fifth element of the vector `vec`.
If we want to reference a consecutive subset of the vector, we use the colon operator within the square brackets, e.g.,

```
> vec[2:7]
```

This will give us elements two through seven of the vector `vec`.

To create a matrix in R, we may use the matrix function. We need to provide a vector containing the elements of the matrix, and specify either the number of rows or the number of columns of the matrix. This number should divide evenly into the length of the vector, or we will get a warning. For example, to make a 2 x 3 matrix named `M` consisting of the integers 1 through 6, we can do this:

```
> M <- matrix( 1:6, nrow=2 )
```

or this:

```
> M <- matrix( 1:6, ncol=3 )
```

We get:

```
> M
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6
```

Note that R places the row numbers on the left and the column numbers on top. Also note that R filled in the matrix column-by-column. If we prefer to fill in the matrix row-by-row, we must activate the `byrow` setting, e.g.,

```
> M <- matrix( 1:6, ncol=3, byrow=TRUE )
```

Then we get:

```
> M
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

In place of the vector `1:6` you would place any vector containing the desired elements of the matrix.
To obtain the transpose of a matrix, we use the transpose function, e.g.,

```
> t(M)
```

This gives us the transpose of the matrix `M` created above:

```
> t(M)
     [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
```

To add or subtract two matrices (or vectors) which have the same number of rows and columns, we use the plus and minus symbols, e.g.,

```
> A + B
> A - B
```

To multiply two matrices with compatible dimensions (i.e., the number of columns of the first matrix equals the number of columns of the second matrix), we use the matrix multiplication operator `%*%`. For example,

```
> A %*% B
```
If we just use the multiplication operator `*`, R will multiply the corresponding elements of the two matrices, provided they have the same dimensions. But this is not the way to multiply matrices.
Likewise, to multiply two vectors to get their scalar (inner) product, we use the same operator, e.g.,
```
> a %*% b
```
Technically, we should use the transpose of `a`. But R will transpose `a` for us rather than giving us an error message.
To create the identity matrix for a desired dimension, we use the diagonal function, e.g.,
```
> I <- diag(5)
```
This gives us the 5 x 5 identity matrix:
```
> 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
```
To find the determinant of a square matrix `N`, use the determinant function, e.g.,
```
> det( N )
```
To obtain the inverse N^{-1} of an invertible square matrix `N`, we use the solve function, e.g.,
```
> solve( N )
```
If the matrix is singular (not invertible), or almost singular, we get an error message.

In linear regression, we must evaluate the equation $\mathbf{b} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}$ to obtain the estimated regression parameters. Here **Y** is the vector of the values of the response variable, and **X** is the design matrix consisting of a column of ones and a column for the values of each predictor variable. If **X** and **Y** have already been created in R, then we can evaluate the equation in R like so:
```
> b <- solve( t(X) %*% X ) %*% t(X) %*% Y
```
Then the vector **b** will contain the estimated regression coefficients.