Chapter 8

Flury Bee

June 9, 2006

1 S-PLUS instructions for Chapter 8

last updated: june 22, 1997

The main computational ingredient of linear principal component analysis is the extraction of eigenvectors and eigenvalues of a symmetric matrix. The numerical methods available for computing eigenvectors and eigenvalues are by themselves highly interesting, but since we are mostly concerned about the statistical aspects, we will simply use the highly efficient procedure available in S-PLUS and not worry about details of the iterative calculations. The particular S-PLUS function we will use is called EIGEN; it computes eigenvalues and associated (normalized) eigenvectors of a symmetric matrix. It would be a good idea to familiarize yourself with EIGEN before starting this tutorial.

We will use the turtle shell example for illustration. First, read the data using

¿ rawdat ¡- matrix(scan("c:/work/book/flury/multidat/tab1$_4$.*dat*"), +*ncol* = 4, *byrow* = *T*)

We want to use the data of the 24 male turtles only, and do principal component analysis on log-transformed data. For numerical convenience, the log-transformed data are also multiplied by 10.

¿ X ¡- 10 * log(rawdat[1:24, 2:4])

Compute the usual sample statistics and, for later use, initialize the values of p (number of variables) and N (number of observations):

¿ xbar ¡- apply(X, 2, mean) ¿ S ¡- var(X) ¿ p ¡- dim(X)[2] ¿ N ¡- dim(X)[1]

Next follows the computation of eigenvectors and eigenvalues:

¿ eig ¡- eigen(S, symm = T) Display eig$values and eig$vectors. In the spectral decomposition theorem, it is convenient to use a diagonal matrix with eigenvalues as diagonal entries; for numerical purposes it is easier to store the eigenvalues in a vector (eig$values). The columns of B are eigenvectors of S in descending order. The following are simple manipulations with data matrices; the only difficulty is that observations are rows of data matrices, while in the theoretical notation observations are column vectors. Make sure you have understood the contents of the software instructions for Section 4.2 before continuing.

By Definition 8.3.1, the transformation from original variables to principal components is achieved by

¿ U ¡- sweep(X, 2, t(xbar))

Using

¿ apply(U, 2, mean) [1] -3.017262e-15 -4.949744e-16 -5.560150e-16 ¿ var(U) [,1] [,2] [,3] [1,] 2.330335e+00 -2.962465e-17 -5.79671e-17 [2,] -2.962465e-17 5.983049e-02 9.93141e-17 [3,] -5.796710e-17 9.931410e-17 3.59836e-02 you may verify that the principal components are centered at zero, and have a diagonal covariance matrix. The eigenvalues in the vector eig$values should appear on the diagonal of the covariance matrix. Consider now computing a two-dimensional principal component approximation. Let A2 be the matrix containing the first two eigenvectors as columns, i.e.,

¿ A2 ¡- eig$vectors[, c(1, 2)]$

By Definition 8.3.1, the two-dimensional principal component approximation of X is

¿ Y2 ¡- sweep(sweep(X, 2, t(xbar)) + FUN = "+") Verify that Y2 has the same mean vector as X. The covariance matrix of Y2 is most easily obtained using the var operator; but you may find it instructive to verify the numerical result using Equation 5 (Property 3) from Section 8.4.

The preceding calculation of the two-dimensional principal component approximation is not really what you would want to do in practical applications with a large number of variables. Rather, you would have only the first two principal components stored, and you would want to construct Y2 using these rather than the original data X. This would work as follows: from the original data, compute the first two principal components by

¿ U2 ¡- sweep(X, 2, t(xbar))

Then, using equation (34) from Section 8.3, the two-dimensional principal component approximation can be computed as

¿ Y2 ¡- sweep(U2

Verify that this gives you the same results as the first method used.

As seen in Section 8.6, we will often be interested in computing standard errors associated with the coefficients of the eigenvectors. Using equation (6), it is not difficult to write a program to do these calculations. Interestingly, it is possible to compute the standard errors for a given eigenvector without using any loops, but this requires some nifty matrix calculations. By equations (2) or (4), we need a matrix of theta-coefficients. First, compute

¿ Ones ¡- matrix(1, p, p) ¿ Lambda ¡- Ones * eig$values > t(Lambda) − Lambda [,1] [,2] [,3] [1,] 0.000000 -2.27050421 -2.29435111 [2,] 2.270504 0.00000000 -0.02384690 [3,] 2.294351 0.02384690 0.00000000

This operation yields a square matrix with value eig$values[i]-eig$values[j] in the [i,j]-th position. We will need the inverse of the square of each entry (except for the diagonal entries which are zero). For any matrix A, the formula $A^(-2)inS-PLUSwillcomputeamatrixofelementwisesquaredinverses.But$

¿ (t(Lambda) - Lambda)^(− 2) [,1] [,2] [,3] [1,] Inf 0.1939793 0.1899679 [2,] 0.1939793 Inf 1758.4753822 [3,] 0.1899679 1758.4753822 Inf will give you a matrix with "Inf" on the diagonal because the diagonal entries are zero. Here is the trick:

¿ Q ¡- (t(Lambda) - Lambda + diag(p))$^{(}$ $-2) - diag(p)$ When you display Q, you will now see a matrix with zeros on the main diagonal, and entries 1 / (eig$values[i] - eig$values[j])^2 else. This calculation will fail if any two entries of eig$values are equal, or if any entry of eig$values is exactly zero.

To get the matrix of theta coefficients, you need to premultiply and post-multiply Q by a diagonal matrix with the eig$values on the diagonal. Since eig$values is a vector rather than a matrix, this is achieved by

¿ Theta1 ¡- sweep(Q, 2, eig$values, FUN = "*") > Theta < -Theta1 * eig$values ¿ Theta [,1] [,2] [,3] [1,] 0.00000000 0.02704558 0.01592954 [2,] 0.02704558 0.00000000 3.78585062 [3,] 0.01592954 3.78585062 0.00000000 Voil!

Let's compute the standard errors for the first eigenvector. One way to do this would be to put the entries in the first column of Theta on the diagonal matrix, say D, and compute `B %*% D %*% t(B)`. A more direct method is

¿ V ¡- eig$vectors This matrix V corresponds to equation (1) or (3) in Section 8.6, with index h=1. The standard errors of the coefficients of the first eigenvector are the square roots of the diagonal entries of V/N:

¿ stdeb1 ¡- sqrt(diag(V/N)) Display the results to verify that you got the same numerical values as those given in Example 8.6.1. Finally, the computation of standard errors of the eigenvalues is very simple, according to equation (7):

¿ stdelam ¡- sqrt(2/N) * eig$values

The above calculation of standard errors of eigenvector coefficients can be done separately for each eigenvector. It will then be more convenient to put these calculations into a loop and store the program as a S-PLUS function, such as function LPC (Linear Principal Components) given below. Call the procedure using

¿ results ¡- lpc(X) and display the results:

¿ cat("eigenvectors:") eigenvectors: ¿ results[1] $eigenvectors[,1][,2][,3][1,]0.6831023 - 0.15947910.7126974[2,]0.5102195 - 0.5940118 - 0.6219534[3,]0.52253920.7884900 - 0.3244015 ¿ cat("eigenvalues:") eigenvalues: ¿ results[2] $eigenvalues[1]2.33033470.05983050.0359836 ¿ cat("standard errors for eigenvector coefficients:") standard errors for eigenvector coefficients: ¿ results[3] $stdeB[,1][,2][,3][1,]0.019125760.28398920.06573967[2,]0.025580750.24761 ¿ cat("standard errors for eigenvalues:") standard errors for eigenvalues: ¿ results[4] $stdelam[1]0.672709690.017271580.01038757

Apply this procedure also to the head dimension data of Example 8.6.2, to verify the numerical results given in Table 8.6.1.