**Principal Components Analysis (PCA)**

Given a table of two or more variables, PCA generates a new table with the same number of variables, called the **principal components**. Each principal component is a linear transformation of the entire original data set. The coefficients of the principal components are calculated so that the **first principal component contains the maximum variance** (which we may tentatively think of as the "**maximum information**"). The second principal component is calculated to have the second most variance, and, importantly, is uncorrelated (in a linear sense) with the first principal component. Further principal components, if there are any, exhibit decreasing variance and are uncorrelated with all other principal components.

PCA is completely reversible (the original data may be recovered exactly from the principal components), making it a versatile tool, useful for data reduction, noise rejection, visualization and data compression among other things.

**Performing Principal Components Analysis**

Performing PCA will be illustrated using the following data set, which consists of 3 measurements taken of a particular subject over time:

>> A = [269.8 38.9 50.5

272.4 39.5 50.0

270.0 38.9 50.5

272.0 39.3 50.2

269.8 38.9 50.5

269.8 38.9 50.5

268.2 38.6 50.2

268.2 38.6 50.8

267.0 38.2 51.1

267.8 38.4 51.0

273.6 39.6 50.0

271.2 39.1 50.4

269.8 38.9 50.5

270.0 38.9 50.5

270.0 38.9 50.5 ];

To summarize the data, we calculate the **sample mean** vector and the sample **standard deviation** vector:

>> AMean = mean(A)  
  
AMean =  
  
269.9733 38.9067 50.4800  
  
>> AStd = std(A)  
  
AStd =  
  
1.7854 0.3751 0.3144

Most often, the first step in PCA is to standardize the data. Here, "standardization" means subtracting the sample mean from each observation, then dividing by the sample standard deviation. This centers and scales the data. Sometimes there are good reasons for modifying or not performing this step, but I will recommend that you standardize unless you have a good reason not to. This is easy to perform, as follows:

>> B = (A - repmat(AMean,[n 1])) ./ repmat(AStd,[n 1])  
  
B =  
  
-0.0971 -0.0178 0.0636  
1.3591 1.5820 -1.5266  
0.0149 -0.0178 0.0636  
1.1351 1.0487 -0.8905  
-0.0971 -0.0178 0.0636  
-0.0971 -0.0178 0.0636  
-0.9932 -0.8177 -0.8905  
-0.9932 -0.8177 1.0178  
-1.6653 -1.8842 1.9719  
-1.2173 -1.3509 1.6539  
2.0312 1.8486 -1.5266  
0.6870 0.5155 -0.2544  
-0.0971 -0.0178 0.0636  
0.0149 -0.0178 0.0636  
0.0149 -0.0178 0.0636  
  
  
This calculation can also be carried out using the *zscore* function from the Statistics Toolbox:  
  
  
>> B = zscore(A)  
  
B =  
  
-0.0971 -0.0178 0.0636  
1.3591 1.5820 -1.5266  
0.0149 -0.0178 0.0636  
1.1351 1.0487 -0.8905  
-0.0971 -0.0178 0.0636  
-0.0971 -0.0178 0.0636  
-0.9932 -0.8177 -0.8905  
-0.9932 -0.8177 1.0178  
-1.6653 -1.8842 1.9719  
-1.2173 -1.3509 1.6539  
2.0312 1.8486 -1.5266  
0.6870 0.5155 -0.2544  
-0.0971 -0.0178 0.0636  
0.0149 -0.0178 0.0636  
0.0149 -0.0178 0.0636

Calculating the coefficients of the principal components and their respective variances is done by finding the eigenfunctions of the sample covariance matrix:  
  
>> [V D] = eig(cov(B))  
V =  
0.6505 0.4874 -0.5825  
-0.7507 0.2963 -0.5904  
-0.1152 0.8213 0.5587  
  
D =  
0.0066 0 0  
0 0.1809 0  
0 0 2.8125

The matrix V contains the coefficients for the principal components. The diagonal elements of D store the variance of the respective principal components. We can extract the diagonal like this:  
  
>> diag(D)  
ans =  
0.0066  
0.1809  
2.8125

The coefficients and respective variances of the principal components could also be found using the *princomp* function from the Statistics Toolbox:  
  
>> [COEFF SCORE LATENT] = princomp(B)  
  
COEFF =  
0.5825 -0.4874 0.6505  
0.5904 -0.2963 -0.7507  
-0.5587 -0.8213 -0.1152  
  
SCORE =  
-0.1026 0.0003 -0.0571  
2.5786 0.1226 -0.1277  
-0.0373 -0.0543 0.0157  
1.7779 -0.1326 0.0536  
-0.1026 0.0003 -0.0571  
-0.1026 0.0003 -0.0571  
-0.5637 1.4579 0.0704  
-1.6299 -0.1095 -0.1495  
-3.1841 -0.2496 0.1041  
-2.4306 -0.3647 0.0319  
3.1275 -0.2840 0.1093  
0.8467 -0.2787 0.0892  
-0.1026 0.0003 -0.0571  
-0.0373 -0.0543 0.0157  
-0.0373 -0.0543 0.0157  
  
LATENT =  
2.8125  
0.1809  
0.0066

Note three important things about the above:  
  
1. The order of the principal components from *princomp* is opposite of that from *eig(cov(B))*. *princomp* orders the principal components so that the first one appears in column 1, whereas *eig(cov(B))* stores it in the last column.  
2. Some of the coefficients from each method have the opposite sign. This is fine: There is no "natural" orientation for principal components, so you can expect different software to produce different mixes of signs.  
3. SCORE contains the actual principal components, as calculated by *princomp*.

To calculate the principal components without *princomp*, simply multiply the standardized data by the principal component coefficients:  
  
  
>> B \* COEFF  
ans =  
-0.1026 0.0003 -0.0571  
2.5786 0.1226 -0.1277  
-0.0373 -0.0543 0.0157  
1.7779 -0.1326 0.0536  
-0.1026 0.0003 -0.0571  
-0.1026 0.0003 -0.0571  
-0.5637 1.4579 0.0704  
-1.6299 -0.1095 -0.1495  
-3.1841 -0.2496 0.1041  
-2.4306 -0.3647 0.0319  
3.1275 -0.2840 0.1093  
0.8467 -0.2787 0.0892  
-0.1026 0.0003 -0.0571  
-0.0373 -0.0543 0.0157  
-0.0373 -0.0543 0.0157

Let's consider what we've gained by making the trip to the principal component coordinate system. First, more variance has indeed been squeezed in the first principal component, which we can see by taking the sample variance of principal components:  
  
>> var(SCORE)  
ans =  
2.8125 0.1809 0.0066  
  
The cumulative variance contained in the first so many principal components can be easily calculated thus:  
>> cumsum(var(SCORE)) / sum(var(SCORE))  
ans =  
0.9375 0.9978 1.0000

Interestingly in this case, the first principal component contains nearly 94% of the variance of the original table. A lossy data compression scheme which discarded the second and third principal components would compress 3 variables into 1, while losing only 6% of the variance.

**Discussion**  
  
PCA "squeezes" as much information (as measured by variance) as possible into the first principal components. In some cases the number of principal components needed to store the vast majority of variance is shockingly small: a tremendous feat of data manipulation. This transformation can be performed quickly on contemporary hardware and is invertible, permitting any number of useful applications.  
  
For the most part, PCA really is as wonderful as it seems. There are a few caveats, however:  
  
1. PCA doesn't always work well, in terms of compressing the variance. Sometimes variables just aren't related in a way which is easily exploited by PCA. This means that all or nearly all of the principal components will be needed to capture the multivariate variance in the data, making the use of PCA moot.  
  
2. Variance may not be what we want condensed into a few variables. For example, if we are using PCA to reduce data for predictive model construction, then it is not necessarily the case that the first principal components yield a better model than the last principal components (though it often works out more or less that way).  
  
3. PCA is built from components, such as the sample covariance, which are not statistically robust. This means that PCA may be thrown off by outliers and other data pathologies. How seriously this affects the result is specific to the data and application.  
  
4. Though PCA can cram much of the variance in a data set into fewer variables, it still requires all of the variables to generate the principal components of future observations. Note that this is true, regardless of how many principal components are retained for the application. PCA is not a subset selection procedure, and this may have important logistical implications.