Carolyn Cotterman
Stat 243
December 2, 2010

Assignment 3

1 Introduction

For this assignment, I implemented some exciting methods in statistical analysis. Specifically, I implemented the Cholesky Decomposition, one-way ANOVA, and linear regression using Gram-Schmidt orthogonalization. In doing so, I learned a good deal both about the functionality of C, and about statistical methods and their properties. My source code and output are contained in the following documents:

- HW3_main.c (source code)
- HW3_functions.c (source code)
- hw3_out.txt (output)

The linear regression works with user-supplied data in the form of text files. It may be tested using the following data:

- X_data.txt
- Y_data.txt

2 Cholesky Decomposition

Cholesky decomposition can be used to construct an upper triangular matrix, U, for a given positive semi-definite matrix, A, such that $A = U'U$. Or, equivalently, Cholesky decomposition can be used to construct a lower triangular matrix, L, for a given semi-definition matrix, A, such that $A = LL'$. To construct the desired matrix, L, Cholesky decomposition is implemented through the following algorithm:

$$L_{ji} = \frac{1}{u_{ii}} \left(a_{ji} - \sum_{k=1}^{i-1} L_{ik} L_{jk} \right) \tag{1}$$

$$L_{ii} = \left(a_{ii} - \sum_{k=1}^{i-1} L_{ik}^2 \right)^{1/2} \tag{2}$$

I tested my implementation of this algorithm by demonstrating that the resulting matrix, L, multiplied by its transpose, L', gives the original matrix, A.

3 Simulation

One-way ANOVA is a method used to test the null hypothesis that the means of several groups of observations are all equal to one another. We can let x_{ij} represent the jth observation in the ith group, and let n_i represent the number of observations in group i. Now, if k is the total number of groups, then the corresponding test statistic for the described null hypothesis is:

$$F = \frac{\sum_{i=1}^{k} n_i(\bar{x}_i - \bar{x})^2/(k-1)}{\sum_{i=1}^{k}\sum_{j=1}^{n_i}(x_{ij} - \bar{x}_i)^2/(N-k)} \tag{3}$$

where

$$\bar{x}_i = (1/n_i)\sum_{j=1}^{n_i} x_{ij}, \qquad \bar{x} = (1/N)\sum_{i=1}^{k}\sum_{j=1}^{n_i} x_{ij}, \qquad and \qquad N = \sum_{i=1}^{k} n_i \tag{4}$$

If the null is true (i.e., the means of all groups are equal), then the statistic F follows the F-distribution with k-1 and N-k degrees of freedom.

I developed a function in C that will calculate the F-statistic, above, for any set of data. I tested my function by using it to calculate the F-statistic for 20,000 different datasets. Each dataset consisted of 6 observations for 5 different groups. Half of the datasets were generated using output from the N(0,1) random number generator that I developed for the previous assignment. By constructing these datasets in such a fashion, I ensured that there would be no correlation, either between groups, or within groups. The other half of the datasets were created using my uncorrelated datasets, but only after imposing a correlation of .7 between the observations belonging to the same group. I imposed this correlation using the Cholesky Decomposition method (please see code for details.)

I found that the distribution of my F-statistic when using non-correlated data closely resembles the distribution expected under the null hypothesis that the means of all of the groups are equal. Specifically, my 90th, 95th, and 99th percentile values are:

- 90th percentile, uncorrelated data: 2.184615
- 95th percentile, uncorrelated data: 2.770003
- 99th percentile, uncorrelated data: 4.193593

By comparison, the corresponding values of the true F-distribution are:

- 90th percentile, true null distribution: 2.18424
- 90th percentile, true null distribution: 2.75871
- 90th percentile, true null distribution: 4.17742

So, with uncorrelated data, we would correctly fail to reject the null that the means of the groups are equal at the rate predicted by our chosen level of significance. Correlated data, however, drastically inflates the value of my F-statistics, thus giving a misleading statistical diagnosis. With correlated data, the percentile values of my F-statistics are:

- 90th percentile, correlated data: 14.992662
- 90th percentile, correlated data: 19.465061
- 90th percentile, correlated data: 29.979936

By construction, I know that the means of my groups are still equal for the correlated data, so we should not reject the null. Thus, with correlated data, we are more likely to commit a Type I error (reject the null even though it is true) than would naively be predicted based on the one-way ANOVA test's level of significance. This is why one of the criteria for proper use of one-way ANOVA is that the responses for a given group are independent and identically distributed (i.e., not correlated!).

4 Gram-Schmidt Orthogonalization

In order to understand the Gram-Schmidt orthogonalization process, I dusted off my trusty linear algebra book and refreshed my memory on what it means to create an orthogonal projection of a vector onto a subspace. I then coded up the Gram-Schmidt process and, to my delight, found that if I provided an n x p input matrix, X, then I was returned an orthonormal matrix, Q, as well as an upper triangular matrix, R, such that the following equation holds:

$$X = QR \tag{5}$$

The Gram-Schmidt process allows us to create an orthonormal basis, which I will denote as $\vec{w_1}, \vec{w_2}, ...\vec{w_p}$, from the column vectors of our design matrix, which I will denote as $\vec{v_1}, \vec{v_2}, ...\vec{v_p}$. To do this, we transform each column of the design matrix, X, iteratively, starting with the first column, $\vec{v_1}$. The first column of our orthonormal matrix Q is simply $\vec{v_1}$ divided by the "vector length" of $\vec{v_1}$, which is $\sqrt{\vec{v_1} \cdot \vec{v_1}}$. The second column of Q, $\vec{w_2}$, must be orthogonal to $\vec{w_1}$. This means that it will be in the "direction" of $\vec{v_2} - (\vec{w_1} \cdot \vec{v_2})\vec{w_1}$, where $\vec{v_2} - (\vec{w_1} \cdot \vec{v_2})\vec{w_1}$ is interpreted as the projection of $\vec{v_2}$ onto the subspace spanned by $\vec{v_1}$. Since we want $\vec{w_2}$ to be of unit length, we then divide this resulting vector by its length which, again, is its inner product. The third column of Q, $\vec{w_3}$, must be orthogonal to both $\vec{w_1}$ and $\vec{w_2}$. Such a vector will be equal to $\vec{v_3} - ((\vec{w_1} \cdot \vec{v_2})\vec{w_1} + (\vec{w_2} \cdot \vec{v_3})\vec{w_2})$, where $((\vec{w_1} \cdot \vec{v_2})\vec{w_1} + (\vec{w_2} \cdot \vec{v_3})\vec{w_2})$ can be interpreted as the projection of $\vec{v_3}$ onto the subspace spanned by $\vec{v_1}$ and $\vec{v_2}$. Again, we normalize the resulting vector by dividing it by its length. This process continues until all columns of matrix X have been transformed into orthonormal vectors. I created the upper triangular matrix, R, using the results of the Gram-Schmidt orthogonalization process.

I tested my Gram-Schmidt orthogonalization process on a number of matrices. Results were verified by both showing that equation 5 holds, and by demonstrating that $Q^T Q$ yields the identity matrix (a property of orthonormal matrices.)

5 Regression using Gram-Schmidt Orthogonalization

Getting a function to perform the Gram-Schmidt orthogonalization was exciting, but even more exciting was getting to use this function to perform a linear regression. Doing so provided me with a new perspective on linear regression and, in particular, the nature of the resulting residuals (e.g., residuals can be thought of as the component of the Y vector that is orthogonal to the orthogonalized X matrix!).

Given the classical linear regression model

$$Y = X\beta + \epsilon \tag{6}$$

we may be inclined to solve for $\hat{\beta}$ by rearranging the above equation and finding

$$\hat{\beta} = (X'X)^{-}1X'Y \tag{7}$$

However, finding $(X'X)^{-1}$ is difficult computationally, and there is a neat alternative that gives us a more favorable time-scaler.

We can use the Gram-Schmidt orthogonalization technique to solve for the orthonormal matrix Q, and the upper triangular matrix, R, where $X = QR$ (as described in the previous section). Substituting equation 5 into equation 7 and rearranging terms gives us

$$R\hat{\beta} = Q'Y \tag{8}$$

which we can solve using backwards substitution.

Now comes the fun part: a method for finding the residuals. We can create an augmented design matrix, call it $X^{(A)}$, that consists of the X matrix with an additional column added on the right-hand side. This additional column contains the Y values. Now, we can proceed with creating orthonormal vectors $\vec{w}_1, \vec{w}_2, ...\vec{w}_p$ from the first p columns of $X^{(A)}$ which, following our previous notation, we will call $\vec{v}_1, \vec{v}_2, ...\vec{v}_p$. If we then create a vector, $\vec{w}^*_{p+1}$ that is the component of $\vec{v}_{p+1}$ that is orthogonal to all of vectors $\vec{w}_1, \vec{w}_2, ...\vec{w}_p$, but do not normalize it, then this vector, $\vec{w}^*_{p+1}$, will contain our residual values. In other words, if we apply Gram-Schmidt orthogonalization to $X^{(A)}$ but do not noramlize the last column, then the last column of the resulting Q matrix will contain our residual values.

Using the residual values and our matrix R from the Gram-Scmidt process, we can find the standard errors of our regression coefficents ($\hat{\beta}$) using a nifty result of matrix manipulations. Namely, we see that

$$(X'X)^{-1} = ((QR)'QR)^{-1} = (R'Q'QR)^{-1} = (R'R)^{-1} \tag{9}$$

In other words, if we let $S = R^{-1}$ then

$$(X'X)^{-1} = SS' \tag{10}$$

Now, our usual equation for the covariance matrix of β can be written as

$$\Sigma_\beta = SS'\sigma^2 \tag{11}$$

where σ^2 is the mean square of our residuals, discussed above, and the diagonals of Σ_β contain our estimates of the standard errors of $\hat{\beta}$. Note that while we normally want to avoid finding the inverse of matrices, we are fortunate in that the above equations mandate only that we find the inverse of matrix R, which is an upper triangular matrix, and is therefore much more easily inverted.

My C program for this assignment uses the above equations and methods to determine parameter estimates and their standard errors, the estimated mean square error, and the corresponding residual values for a linear regression model. I tested my program on a variety of datasets, which can either be entered as matrices into the source code itself, or can be read from a text file. I am providing along with this report a dataset containing X values and a dataset containing Y values that can be used to demonstrate my program's functionalities.

6 Conclusion

This assignment augmented both my understanding of statstics, and my programming skills. For example, it required that I make careful use of "two-dimensional" matrices. The truth is, C does not store matrices in two dimensions like a layperson may think. In fact, if you define a matrix Z as, for example, "int Z[5][3]," then you will not actually produce a two-dimensional matrix with cells that are contiguous in memory. Instead, you will create 5 different arrays, each containing 3 elements, and there is no guarantee that these arrays will be located near each other in the computer's memory (and if they are not near each other in memory, then your program could run much more slowly). Furthermore, multi-dimensional arrays in C are incompatible with R, fortran, etc. becuase these other programs assume that the elements of a matrix are stored contiguously in memory. I therefore made sure to work exclusivley with one-dimensional arrays, which I interpreted as two-dimentional matrices to facilitate the execution of various statistical procedures.