FIELD PAPERS

SOMALIA

Outline for a Workshop on Advanced Computer and Statistical Topics

April 1987

IEES

Improving the Efficiency of Educational Systems

Florida State University
Howard University
Institute for International Research
State University of New York at Albany

United States Agency for International Development
Bureau for Science and Technology
Office of Education
Contract No. DPE-5283-C-00-4013-00

Improving the Efficiency of Educational Systems (IEES) is an initiative funded in 1984 by the Agency for International Development (AID), Bureau for Science and Technology, Office of Education. The principal goals of the IEES project are to help developing countries improve the performance of their educational systems and strengthen their capabilities for educational planning, management, and research. To achieve these goals, a consortium of U.S. institutions has been formed to work collaboratively with selected host governments and USAID Missions for ten years. The consortium consists of The Florida State University (prime contractor), Howard University, the Institute for International Research, and the State University of New York at Albany.

There are seven countries working with the IEES initiative to improve educational efficiency: Botswana, Haiti, Indonesia, Liberia, Nepal, Somalia, and Yemen Arab Republic.

Documents published by IEES are produced to promote improved educational practice, planning, and research within these countries. All publications generated by project activities are held in the IEES Educational Efficiency Clearinghouse at The Florida State University. Requests for project documents should be addressed to:

IEES
Educational Efficiency Clearinghouse
Learning Systems Institute
204 Dodd Hall
The Florida State University
Tallahassee, Florida 32306
USA
(904) 644-5442

Agency for International Development
Bureau for Science and Technology
Office of Education
Contract No. DPE-5823-C-00-4013-00
Project No. 936-5823

SOMALIA

Outline for a Workshop on Advanced Computer and Statistical Topics

Michael S. Green

April 1987

IEES Field Papers are the products of IEES assistance in collaborating countries. They are papers, outlines, guides, or summaries produced by consultants or staff as part of the long-term IEES strategy which is collaboratively developed with each country for improving educational efficiency. Some of these papers are products of USAID-funded and IEES-administered field projects now underway. These papers are copied and disseminated in their original form because of their potential usefulness to technical staff, planners, and decisionmakers in developing nations and to agencies assisting those nations in the improvement of their educational systems.

A complete list of **IEES Field Papers** may be obtained from:

Improving the Efficiency of Educational Systems
204 Dodd Hall
Florida State University
Tallahassee, FL 32306 USA

CONTENTS

Measures of Central Tendency & Dispersion 1

Forecasting 3

Cohort Analysis 4

OLS Regression 6

Factor Analysis 8

Cycle Cost & Efficiency 10

BASIC Programming 12

IEES--Ministry of Education, Somalia

Workshop

Advanced Computer and Statistical Topics
March 29 - April 13, 1987

Dr. Michael S. Green
State University of New York at Albany

Measures of Central Tendency & Dispersion

Central Tendency

Summary statistics such as measures of central tendency are used to describe a sample or population in an attempt to characterize a large amount of data with one or two "average" scores. Each measure of central tendency is an actual score within the distribution. The three most commonly used statistics are:

-mode
-median
-mean

Mode

The mode is the most frequently occurring score in the distribution. It is quite possible to have more than one mode in a particular distribution (in fact, in a distribution in which all scores occur only once, each could be considered to be a mode). When the data being considered in only nominal or categorical in nature (such as sex, race, religion, etc.), it is the only measure of central tendency that can be applied.

Median

The median is the score at which 50% of the distribution lies at or below, in other words, the 50% percentile. It is the score that divides the distribution in half. To calculate the median, arrange the data from lowest to highest and take the middle score. For example,

3, 5, 7, 8, 3, 27, 33 median=8
11, 12, 14, 17, 19, 20 median=15.5

It is an appropriate measure of central tendency to employ when the data is at least ordinal (signifying ranks) in nature.

It is also resistant to extreme scores as the median is defined as the 50% percentile, thus what takes place at either end of the distribution has very little, if any, effect.

Mean

The mean is what is commonly thought of as the "average" score, the sum of the scores divided by the number of scores or in statistical terms

$$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n}$$

Ex. 65,50,77,90,82,93,88,73,91,85

$\bar{X} = 794/10 = 79.4$

It is the balance point of the distribution and is the appropriate measure to employ when the data constitutes an interval or ratio scale.

As the mean takes into account every score in the distribution, it is highly influenced by extreme scores, especially when the sample size is small. For example, if to the list of scores above we add the score of 10, the mean goes from 79.4 to 73.09. However, if in a distribution of 100 score where $\bar{X} = 7940/10 = 79.4$ and we add the same scores of 10, $\bar{X} = 78.7$.

Knowledge of all three measures of central tendency can tell us the shape of the data distribution. If the mean, median, and mode are equal to one another, the distribution is symmetric. If the mean is smaller than the median which is smaller than the mode, the distribution is negatively skewed. If the reverse is true, i.e., the mode is smaller than the median which is smaller than the mean, the distribution is positively skewed.

Dispersion

Once one has an idea about the "average" performance within a distribution of scores, the next question usually asked is what is the spread of those scores. How far apart or close together are they? Some sort of measure of dispersion is needed to put the central tendency measure into perspective.

Range

The range is defined as the distance the lowest to the highest score plus one.

rng = High - Low + 1

It is the simplest of the dispersion measure to calculate but it is not very representative of the distribution as it only takes into account two points--the two ends.

Semi-interquartile range

This measure is defined as the distance between the 75% percentile (Q3) and the 25% percentile (Q1) divided by two or

$$Q = \frac{Q3 - Q1}{2}$$

It, like the range only takes into account two points of the distribution. It does provide a little more information than the range as to calculate it you need Q3 and Q1. If one has Q2, the median, as well, one can tell the shape of the distribution. If the distance from Q1 to Q2 is equal to the distance from Q2 to Q3, the distribution is symmetric. If the distance from Q1 to Q2 is greater than that of Q2 to Q3, the distribution is negatively skewed. If the distance from Q1 to Q2 is less than the distance from Q2 to Q3, the distribution is positively skewed.

Variance & Standard Deviation

The variance and the standard deviation take into account every data point in the distribution. The variance is equal to the sum of the squared deviations from the mean divided by the number of deviations or

$$s_x^2 = \frac{\sum_{i=1}^{n} (X - \bar{X})^2}{n} = \frac{\sum_{i=1}^{n} X_i^2}{n} - \bar{X}^2 = \frac{1}{n}\left(\sum_{i=1}^{n} X_i^2 - \frac{\left(\sum_{i=1}^{n} X\right)^2}{n}\right)$$

and the standard deviation is the square root of the variance

$$s_x = \sqrt{s_x^2}$$

Note that as the variance uses squared deviations, it can best be thought of as a measure of area, rather than one of dispersion. That is why the standard deviation, which is the square root of the variance, is used to provide a measure of dispersion.

As the variance and the standard deviation take into account every score in the distribution, they are greatly influenced by extreme, especially when n is small. If one calculates the variance and standard deviation of the set of scores used in the example for the mean, $s_x^2 = 166.24$ and $s_x = 12.89$. If the score of 10 is added to the distribution $s_x^2 = 549.17$ and $s_x = 23.43$.

Forecasting

Forecasting is the process of making a prediction, an educated guess, about the state of affairs in the future given data from the past and conditions today. All forecasting models make assumptions regarding the data undergoing analysis, the current and past environment, other variables influencing the data, and future directions. Given its nature, it is as much a science as an art, a process which employs rigorous mathematical models or one that goes on "gut feelings."

All forecasting methods require the following:

1. The data to be used must be organized in a systematic fashion.

2. The method employed must be consistent over time.
3. The assumptions underlying the forecast must be clearly stated and understood.
4. The reasons for the choice of method must be clearly stated and understood.

Note, too, that all forecasting methods require the user to make some assumptions regarding conditions in the future and what events are likely to take place that might influence his or her predictions. These may include such things as:

1. birth rate
2. death rate
3. economic conditions
4. political and social conditions
5. administrative policies
6. "acts of God"

There exists a hierarchy of methods, increasing in the complexity of their models and the amount of data used as a basis for prediction. They include:

1. "eyeballing the data"
2. percentages changes (increase or decrease) from year to year
3. cohort survival technique
4. regression techniques
5. economic modeling
6. census based forecasting

Very often the nature and amount of data available will determine the choice of forecasting method. As one moves up the hierarchy more and more data is required. If one has very little on which to base a projection, one may only be able to employ a percent change approach or use own best judgement ("eyeballing the data").

Regardless of the method employed, forecasting is only as good as the data on which the forecast is based, the appropriateness of the assumptions made, and the completeness of the assumptions made.

Cohort Analysis

Cohort analysis is used to forecast enrollments for schools or for any uniform group (cohort) that progresses through a series of grades or uniform steps. Basically, the technique calculates the average change in enrollment (survival rate) and uses it to make a linear projection from the last data point. The steps involved are as follows:

1. Organize the data by grade by year

Yr/Grade	I	II	III	IV	V
80-81	59809	40212	40838	31135	18290
81-82	47507	37922	22118	33882	26314
82-83	37415	33153	29809	28232	31113
83-84	42840	31275	28579	26843	27247
84-85	34223	32012	27110	24423	25526

2. For each year calculate the survival rate of the student cohort moving from one grade to the next. This is done by dividing the number of students in the higher grade in the second year by the number of students in the lower grade the previous year.

37922/59809	22118/40212	33882/40838	26314/31135
33153/47507	29809/37922	28232/22118	31113/33882
31275/37415	28579/33153	26843/29809	27247/28232
32012/42840	27110/31275	24423/28579	25526/26843

Yr/Grade	I	II	III	IV	V
2		6341	.5500	.8297	.8452
3		6979	.7861	1.2764	.9183
4		.8359	.8620	.9005	.9651
5		.7472	.8668	.8546	.9509

3. Calculate the average survival for each grade by summing the survival rates for each grade across the years and dividing by the number of years.

Yr/Grade	I	II	III	IV	V
		.7288	.7662	.9653	.9199

4. To make a forecast, multiply the average survival rate for the grade by the number of people in the previous grade in the year preceeding the one to be forecast.

Yr/Grade	I	II	III	IV	V
80-81	59809	40212	40838	31135	18290
81-82	47507	37922	22118	33882	26314
82-83	37415	33153	29809	28232	31113
83-84	42840	31275	28579	26843	27247
84-85	34223	32012	27110	24423	25526
85-86*		24940	24529	26169	22466
		34223	32012	27110	24423
		* .7288	* .7662	* .9653	* .9199

Note that the cohort analysis technique assumes that the trends present in past years will continue into the years of the forecast. If this is not the case, adjustments must be made to the forecast to reflect the anticipated conditions. Note, too, that the greater the number of years on which the forecast, the more faith one can place in the projections made assuming the past data is both reliable and valid. If the past data is not reliable and valid, any projections made using it as a base are spurious at best.

OLS Regression

In OLS (Ordinary Least Squares) regression forecasting an attempt is made to construct an equation where one set of variables called predictors or independent variables are used to predict performance on some criterion or dependent variable. The simplest case is when there is one predictor and one criterion.

In regression one uses the relationship between predictor and criterion to aid in the construction of the equation. The linear relationship between two variables can be measured by the correlation betwen those two variables.

$$r_{xy} = \frac{\sum_{i=1}^{n} x_i y_i}{n} - (\bar{X})(\bar{Y})$$

The correlation (in this case a Pearson Product Moment correlation) can range from +1.00 to -1.00. The closer r is to either +1 or -1, the greater the linear relationship between the two variables (let us call them X & Y for illustrative purposes). The closer r is to 0, the less the relationship. If r is positive there is said to be a direct relationship between X & Y (as one variable increases, the other increases), if the correlation is negative it is said to indicate an inverse relationship (as one variables goes up, the other goes down). One can think of correlation as the overlap between two variables, the higher the correlation, the more they share in common.

Regression tries to construct a line of "best fit" through the data points. The question is, however, what defines "best fit?" OLS regression defines it as a line that minimizes the sum of the squared errors of prediction (the difference between the predicted Y value and the actual Y value). This criterion is the basis for OLS regression and is called the Least Squares Criterion.

If one goes through the mathematics, the line of best fit, Y=bX+a, is produced when

$$b = r_{xy} \left(\frac{S_Y}{S_X}\right) \qquad a = \bar{Y} - b\bar{X}$$

To calculate the predicted value, merely substitute the desired value of X in the equation and solve for Y.

The effectiveness of the regression can be investigated by examining r^2, the coefficient of determination. This measure indicates the amount of

variance in one variable that can be explained by, or attributed to, the other variable. In this case, it is the amount of variation in Y (the dependent variable) that can be attributed to X (the independent variable). The greater the value of r^2, the better the regression equation and the better the predictions. This measure can be assessed through a statistical test as follows.

$$F = \frac{r^2}{(1-r^2/(n-2)} \qquad df=(1,n-2)$$

If the calculated F-statistic is greater than the tabled F-value given the degrees of freedom and chosen level of significance, the regression equation is said to account for a significant amount of the variation in Y, thereby providing confidence in any predictions made.

As an example, let us predict the 1985/86 Grade II total enrollment using the enrollments from 1980/81 through 1984/85 to develop the regression equation. Here, then, x would be years coded from 1 to 5 and Y would be the total enrollments for each of those years.

$X = 1 + 2 + 3 + 4 + 5 = 15$

$X = 1^2 + 2^2 + 3^2 + 4^2 + 5^2 = 55$

$Y = 40212 + 37922 + 33153 + 31275 + 32012 = 174574$

$Y = (40212)^2 + (37922)^2 + (33153)^2 + (31275)^2 + (32012)^2$
$= 6157098206$

$XY = (1)\ (40212) + (2)\ (37922) + (3)\ (33153) + (4)\ (31275) + (5)\ (32012)$
$= 500675$

$\overline{X} = 15/5 = 3.0$

$s = (55/5) - (3.0) = 1.41$

$\overline{Y} = 174574/5 = 34914.8$

$s = (6157098206/5) - (34914.8) = 3518.01$

$r = ((500675/5)-(3.0)\ (34914.8)) / ((1.41)\ (3518.01)) = -.9265$

$r^2 = .8564$

$b = (-.9265)\ (3518.01)/1.41 = -2304.7$

$a = 34914.8 - (-2304.7)\ (3.0) = 41828.9$

$F = .8584/ ((1-.8584) / (5-2)) = 18.1792 \qquad df=(1,3)$

1985/86* $\hat{Y} = bX + a = (-2304.7)\ (6) + 41828.9 = 28001$

Very often the regression equation will appear not to work, i.e., a low r^2 will be produced. One possible explanation for this is that the relationship between X and Y is nonlinear. Whenever possible, one should graph the data points to see if such a relationship is present.

If the relationship does appear to be nonlinear then a transformation of the X variable can be attempted to try to convert the relationship into a linear one. Some of the more common transformations include X^2, $\sqrt{X}$, $1/X$, e^x, etc. Very often a number of different transformations will have to be tried before the one producing the best fit can be found.

Up to this point only simple regression (1 independent, 1 dependent) have been discussed. All that has been covered can be extended to the multiple regression (more than one independent variable) case. In multiple regression a b term or regression coefficient is calculated for each independent variable. In this case, however, rather than just having to deal with the correlation between that variable and Y, one must also take into account the correlations with that variable and all the other independent variables. If these intercorrelations are high (in the neighborhood of .80 or above) a condition called multicolinearity is said to be present in your regression equation and steps should be taken to reduce it. Such steps might include such things as dropping variables, combining highly correlated variables to form composites, or if a sufficient number of variables and data points are available, subjecting the set of independent variables to a factor analysis.

A multiple regression equation has the form

$$Y = b_o + b_1 X_1 + D_2 X_2 + \ldots b_p X_p$$

and the F-test used to test such a model is

$$F = \frac{R^2/p}{(1-R^2/(n-p)}$$

df = (p,n-p)

p = number of X variables

R^2 = corr. between Y & $\hat{Y}$

Factor Analysis

Factor analysis is a method for taking a large number of variables and reducing them down to a smaller number of factors, which are linear combinations or composites of the original variables. The process accomplished through a statistical examination of the matrix of intercorrelations among these variables in an attempt to find sets of variables, while correlating highly among each other, do not correlate well with the other variables in the matrix.

There are a number of different methods that can be used to arrive at these factors. They include principal components, principal factors, and image analysis, to name a few (the most commonly employed are principal

components and principal factors). Regardless of method, the correlation matrix is subjected to analysis, resulting in the calculation of a set of eigenvalues, one for each factor. These eigenvalues indicate the relative importance of each factor. The first eigenvalue is associated with the first factor, which is the one that accounts for the largest portion of variance, and so on. There is a rule of thumb that states that a factor is considered important if its eigenvalue is equal to or greater than one.

The result of this initial factor analysis is a matrix of correlations between each of the variables and the factors derived from the variables. This matrix is called a structure matrix and the correlations within it are called loadings. The higher the loading, the more the variable correlates with the factor.

One very useful property of the factors that are produced at this point is that they are uncorrelated with each other. Each accounts for its own unique portion of the variance. This uncorrelatedness is called orthogonality and the factors are said to be orthogonal to each other.

Very often this initial structure matrix is difficult to interpret as all the variables have high loading with the first factors and low loadings elsewhere. In an attempt to simplify the structure and make it more readily interpretable, very often this structure matrix is subjected to a rotation. There are a number of possible rotation techniques that can be used such as varimax, quartimax, and equimax. All of these make the loadings easier to interpret by spreading the variation out across the rows or columns, causing the first factor to no longer have all the high loadings. These three rotation techniques also have the added property of preserving the orthogonality of the derived factor system. Even after rotation, then, the factors will still continue to be uncorrelated with each other. There is a fourth rotation technique that is sometimes used called an oblique rotation. Like the other three it is designed to make the structure matrix easier to interpret. Unlike the others, however, it does not leave the factors uncorrelated. In fact, it introduces correlation to the system. This method might be the one to employ when the theory behind the research being done would lead one to hypothesize factors that are non-orthogonal (self-esteem and self-concept, for example).

Once rotation has been completed, a second structure matrix is produced, It is this matrix that is used from this point. To arrive at names for the factors produced, an arbitrary loading is chosen (say .50 to begin) and all loadings that are as large or larger than this value are identified. Then, within each factor, the researcher looks at the variables that meet this criterion and tries to decide what it is that this subset of variables share in common. That shared trait or construct then becomes the name of the factor. Very often one will have to either increase or decrease the criterion value in order to arrive at a readily interpretable set of factors. The best of all possible worlds is when each item loads on one and only one factor. It is not unusual, however, to find a variable that loads on two or more factors. This is not a major problem, it just means that the item is tapping more than one construct.

Once the factors have been named, factor scores must be calculated. There are a number of approaches to this that can be taken. The first is just to assign weights of +1 or -1 (depending on whether the loading is positive or negative) and then add up the subject's scores on the high loading variables (I designate this the factor scale method). Another method is to use the loadings like regression coefficients and multiply the scores on the high loading variables and add up the results (mixed method). The third approach is to say that every variable can be considered to be part of a factor except some are more important than others and the relative importance of each variable is indicated by its loading. Therefore, the best factor score must be one which uses every variable, multiplied by its respective loading, and summed to produce a total for the factor (factor score method). Of the three methods, it is the third, the factor score method, that maintains the orthogonality of the system the best.

As can be seen from this very brief description, factor analysis is a highly subjective statistical technique with choices left to the researcher at every point of the process. It would not be unheard of to give two different researchers the same set of data and have them produce two completely different factor structures.

Cycle Cost & Efficiency

It is useful to follow a cohort of students through the educational process in order to get some idea regarding the efficiency of the system and what the cost is of producing a graduate. The table below tracks a cohort of students through an educational system containing six grades and requiring the passing of an examination to graduate. Let us assume that the cohort starts with 100 students in Grade 1, Year 1. As we go from Grade 1 to Grade 2, 80 students successfully make the transition while 5 are required to repeat Grade 1 and 5 drop out (the numbers enclosed within circles are the dropouts). Moving from grade to grade a steady decrease can be observed and assuming that no new students join this first cohort (due to transfers, for example), 68 of the original 100 make it through the full six years and of those 68, 60 pass the exit examination on the first try.

Given the information in the above table, a measure of the cycle cost can be calculated. Cycle cost can be defined as how many years, on the average, does it take to produce a graduate.

$$cc = \frac{\#\ \text{student years}}{\#\ \text{graduates}}$$

In this example, the cycle cost is

$$cc = \frac{117 + 94 + 94 + 83 + 80 + 86}{69} = \frac{554}{69} = 8.03 \text{ years}$$

Tracking Table

Gr / Yr	1	2	3	4	5	6	GRAD
1	100	5					
2	15	80	1				
3	2	14	75				
4		0	18	70	1		
5			1	13	68	0	
6				0	12	68	60
7						18	9
TOT	117	94	94	83	80	86	69

On the average, then, it takes 8.03 years to produce a graduate.

The efficiency of the system can be assessed by dividing the cycle cost by the number of years it is supposed to take to produce a graduate. For this example, the efficiency would be 8.03/6.0 or 1.34. The closer this ratio is to one, the more efficient the system. The greater the ratio, the less efficient the system.

The cycle cost and efficiency indices can be used for comparative purposes to identify district or regional differences. For example, it might be interesting to see if urban districts are more or less efficient than rural districts.

It should be noted that to utilize the approach to its fullest, very extensive and reliable data must be available. The more questionable the data, the more questionable the resulting indices.

BASIC Programming

All computers must be given lists of instructions that tell them how to perform the tasks requested of them. These lists of instructions are called programs. These programs are written in programming languages which are artificial languages with rules of grammar and syntax that the computer can be made to understand. There are many different languages (Fortran, COBOL, PL/I, APL, Pascal, C, Lisp, etc.) but the language that is standard with most, if not all, microcomputers is BASIC.

What one does when writing any program is to take a task, let us say the calculation of the average of three numbers, and split it into its component parts. In our example, these would include (in order): get the three data points, add the three data points, divided the resulting sum by three, and display the answer. Once the subtasks have been specified they are converted into instructions to the computer using the programming language.

What follows is a description of some of the commands available to the programmer when using BASIC. This is not a complete list and additional features can be found in the Wang Professional Computer Series BASIC Language Guide, but it is enough to give the reader an introduction to BASIC programming and to begin to write one's own programs.

READ & DATA Statements

In all programs, one must provide the computer with the data on which it is to operate. One way to do this is with the READ and DATA statements.

```
READ var list
DATA values

Ex. READ X,Y
    DATA 1,41433
```

In the above example X and Y are variable names. A variables name can be up to 40 characters long, must start with a letter, and not include any spaces. Both X and Y are treated as variables representing numeric data. If the data is to be character data (numbers and letters) or strings as such data is called in BASIC, the variable name must end with a $. For example, NAME$.

There is a one to one correspondence between the READ statement and the DATA statement. The READ statement above tells the computer to expect to read two variables, X and Y, and to assign the first value read to X and the second to Y. In this example, then, X=1, and Y=41433. Note, also, that every READ must have a DATA statement to go along with it somewhere in the program.

LET Statements

LET statements or assignment statements allow us to manipulate the data within the program. They are the statements used to perform mathematical calculations.

```
LET var = expression

Ex.  LET C=A+B
     LET X=1
     LET Y=A*B+2
     LET N=N+1
```

Note that in all cases the new value is the one on the left side of the equal sign and the old value is on the right.

There is a hierarchy of arithmetic operations that is employed in BASIC. Powers [↑] are done first, then multiplication [*] and division [/], and then addition [+] and subtraction [-]. To override this order, one must enclose the desired portion of the calculation within parentheses [()]. Any calculation enclosed within parentheses is always done first.

PRINT and LPRINT Statements

Once the data has been manipulated in a suitable fashion, a way is needed to display the results. This is done with a PRINT or a LPRINT statement. The only difference between the two is that the PRINT statement directs the results (output) to the screen while the LPRINT directs it to the printer.

```
PRINT var list

Ex. PRINT X,Y
    PRINT "A = ";A
```

In the first example, the two variables are separated by commas. In BASIC on the Wang, a line on the screen or printer is split into print zones of 14 columns each. The comma here means to print this value in the next

print zone. In the second example, one portion of the command is enclosed within double quotes ("). This is called a literal and the computer will print whatever is enclosed within quotes exactly as it is written. The semicolon between the " and the A tell the computer to print the value of A starting in the column directly after where it had just finished printing.

END Statement

All that is needed in order to be able to write a simple program is a way of telling the computer that it has reached the end of a program. This is done through the use of an END statement. The last line of a program is always an END statement.

Let us take the task, then, of finding the average of three numbers and write a program to do it.

10 READ ENR1,ENR2,ENR3	read 3 vars
20 DATA 40212,37922,33153	specify values
30 LET TOTAL=ENR1+ENR2+ENR3	add 3 vars to compute TOTAL
40 LET MEAN=TOTAL/3	divide TOTAL by 3 to get MEAN
50 PRINT "MEAN = ";MEAN	print value of MEAN
60 END	

Note that each line of BASIC begins with a line number. It is this line number that tells the computer the order in which to do the statements. The computer will execute the statements in order from lowest to highest.

BASIC Instructions

RUN	Tells computer to execute program currently in memory
SAVE"name"	Tells computer to save program in memory onto specified disk under specified name
LOAD"name"	Reads specified file from disk and places it into memory
LIST	Requests computer to display listing of program on screen
LLIST	Requests computer to print listing of program
NEW	Instructs computer to clear (erase) memory
SYSTEM	Instructs computer to leave BASIC and return to DOS

REM Statement

When writing programs it is important to place comments within them so that if you come back to the program at a later date you will know what the program does and how to use it. Such comments also make the program easier for someone else to understand if you are no longer available to provide assistance. These comments are called documentation and can be inserted into the program through the use of REM statements.

REM text

Ex. REM THIS PROGRAM CALCULATES THE MEAN OF 3 NUMBERS

A program may have as many REM statements as the programmer would like to have. They are non-executable statements and are ignored by the computer when the program is running. They are there only for the user's benefit.

INPUT Statement

A little earlier the READ and DATA statement combination was presented as a way to get data into a program. The only program with this approach is that it is somewhat limiting. As these statements are part of the program, if a user wishes to run a program using a different set of data, he or she must go in and change the appropriate DATA statements. One can, instead, employ a statement that will request the user to insert the data, when needed, via the keyboard. This statement is called the INPUT statement.

INPUT var list

Ex. INPUT ENR1, ENR2, ENR3

When the program reaches this statement, it will stop and prompt the user with a question mark (?), signifying that it is waiting for data. It will continue to wait until the requested data has been entered and then the program will continue.

The statement can also be written in such a way as to print out a message to the user reminding them of the data been requested.

Ex. INPUT "ENTER ENROLLMENTS FOR YEARS 1 THRU 3";ENR1,ENR2,ENR3

IF-THEN Statements

Very often a programmer will want a step or series of steps to be executed only if a certain condition is met. This can be accomplished through the use of the IF-THEN statement.

IF condition THEN statement

Ex. IF A=1 THEN C=B * .90

The = sign is what is called a relational operator and there are six of them.

= equal to	< less than	<= less than or equal to
<> not equal to	> greater than	>= greater than or equal to

A number of conditions can be strung together to form a complex task by using the logical operators AND and OR.

IF A=1 AND B=2 THEN C=C+1	both conditions must be true before the statement C=C+1 will be executed
IF A=1 OR B=2 THEN C=C+1	either one or the other condition must be true before the statement C=C+1 will be executed

GOTO Statement

The GOTO statement allows the programmer to transfer execution from one part of the program to another. This is very useful, especially when combined with an IF-THEN statement, in forcing the program to execute a series of statements more than once.

GOTO line

Ex. GOTO 50 transfers execution to line 50 of the program

A program can now be written using the statements that have been discussed to this point that will calculate the average of any number of points.

```
10  REM PROGRAM TO CALCULATE AVERAGE

20  INPUT "ENTER NUMBER OF CASES";N

30  C=1: T=0                        C is the number of the data pt
                                    T is sum of the data pts
40  IF C>N THEN GOTO 90             Test to see if all data pts
                                    have been processed
50  INPUT "ENTER DATA PT";X

60  T=T+X                           Accumulates sum of data pts

70  C=C+1                           C increased by 1 as have
                                    processed data pt
80  GOTO 40                         Send program back to process
                                    next data pt
90  AVE=T/N                         Calculate average

100 PRINT "N=";N,"AVERAGE=";AVE     Print N and average

110 END
```

Ex. Program to calculate correlation

```
10   REM PROGRAM TO CALCULATE CORRELATION
20   INPUT "ENTER NUMBER OF CASES";N
30   SX=0: SY=0: SX2=0: SY2=0: SXY=0     Set sums to 0
40   C=1
50   IF C>N THEN GOTO 140
60   INPUT "ENTER X,Y";X,Y
70   SX=SX+X
80   SX2=SX2+(X↑2)
90   SY=SY+Y
100  SY2=SY2+(Y↑2)
110  SXY=SXY+(X*Y)
120  C=C+1
130  GOTO 50                            Go back to process next X,Y
140  MX=MX/N: MY=MY/N                   Calculate means
150  SDX=((SDX/N)-(MX↑2))↑0.5           Calculate std dev of X
160  SDY=((SDY/N)-(MY↑2))↑0.5           Calculate std dev of Y
170  R=((SXY/N)-(MX*MY))/(SDX*SDY)      Calculate correlation
180  PRINT "MEAN X=";MX,"MEAN Y=";MY    Print means
190  PRINT "STD DEV X=";SDX,"STD DEV Y=";SDY   Print std devs
200  PRINT "R=";R                       Print correlation
210  END
```

FOR-NEXT (loop) Statements

Very often a programmer will want the computer to repeat a series of steps a certain number of times. This could be done using the IF-THEN and GOTO combination used in the correlation program, above, but there is a

simpler way. A sequence of statements executed more than once is called a loop. One way to create such a loop is with FOR-NEXT statements.

```
FOR counter=start value TO end value
   .
   .   BASIC statements to be repeated
   .
NEXT counter

EX. FOR I=1 TO 10
    I2=I*I
    ISQ=I↑0.5
    PRINT I,I2,ISQ
    NEXT I
```

In a FOR-NEXT loop, the counter variable, I in the above example, is incremented by one each time the computer reaches the NEXT I statement and is then sent back to the beginning of the loop (the FOR line). Then a test is performed to see if I is now greater than the end value of the loop. If it is, the loop ends and the program continues with the line directly following the NEXT I. If not, the loop repeats.

A loop may contain any number of statements between the FOR and NEXT lines. It is also possible to increment the counter in steps other than one by making a small change to the FOR line as follows:

```
FOR I=1 TO 10 STEP 2
```

This would cause I to increase by two each time the loop executed (in this case I=1,3,5,7,9). A counter can even be set to go backwards by specifying a negative STEP value.

```
FOR I=10 TO 1 STEP -1
```

Functions

Functions are defined sets of operations that are built into BASIC and are referenced by name. They include (to name only a few):

SQR(X)	calculates square root of X
INT(X)	returns closest integer<X
FIX(X)	returns integer portion of X
RND (X)	returns random number between 0-1

- In order to make sure that the numbers produced by the RND function are indeed random, the command RANDOMIZE should be placed at the beginning of the program. This will request the user to specify a starting point (a seed) that unless you use the same seed again, each time the program is run a different set of random numbers will result.

TAB(X)

- TAB is used in a PRINT of LPRINT statement to force the computer to move to a certain column before printing (similar to a TAB key on a typewriter).

Ex. Program to perform OLS regression forecasting

```
10  REM PROGRAM TO DO REGRESSION FORECASTING FOR ENROLLMENTS
20  INPUT "ENTER NUMBER OF YEARS";NY
30  SX=0: SY=0: SX2=0: SY2=0: SXY=0
40  FOR I=1 TO NY                              Loop to enter & sum data
50  PRINT "ENTER ENROLLMENT FOR YEAR";I
60  INPUT ENRL
70  SX=SX+I:  SX2=SX2+(I*I)
80  SY=SY+ENRL: SY2=SY2+(ENRL↑2)
90  SXY=SXY+(I*ENRL)
100  NEXT I
110  REM CALCULATE MEAN, STD DEV, R, R2
120  SX=SX/NY: SY=SY/N
130  SX2=SQR((SX2/NY)-(SX↑2))            Calculate std dev years
140  SY2=SQR((SY2/NY)-(SY↑2))            Calculate std dev enroll
150  R=((SXY/NY)-(SX*SY))/(SX2*SY2): R2=R*R
160  PRINT "MEAN YEARS=";SX;TAB(30);"STD DEV YEARS=";SX2: PRINT
170  PRINT "MEAN ENROLLMENT=";SY;TAB(30);"STD DEV ENROLLMENT=";SY2: PRINT
180  PRINT "R=";R;TAB(15);"R-SQUARE=;R2
190  PRINT
200  REM CALCULATE REGRESSION LINE
210  B=R*SY2/SX2                            Calculate b term
220  A=SY-(B*SX)                            Calculate a term
```

```
230  PRINT "REGRESSION EQUATION:": PRINT

240  PRINT TAB(10);"Y = ";B;" * X + ";A: PRINT

250  F=R2/((1-R2)/(NY-2))          Perform F-test

260  PRINT "F=";F;TAB(15);"DF=1,";NY-2: PRINT: PRINT

270  INPUT "DO YOU WISH TO MAKE A PREDICTION (Y,N)";C$

280  IF C$="N" THEN GOTO 330              No prediction requested

290  INPUT "ENTER YEAR TO BE PREDICTED";YR

300  ENRL=(B*YR)+A                        Predicted enrollment

310  PRINT "PREDICTED ENROLLMENT FOR YEAR";YR;" = ";ENRL: PRINT

320  GOTO 270                     Request another predict

330  INPUT "DO YOU WISH TO DO ANOTHER REGRESSION (Y,N)";C$

340  IF C$="Y" THEN GOTO 20       Do another regression

350  END
```

Arrays & Matrices

An array is a group or set of values of the same data type (all number or all character) that are referenced by the same variable name. Each element within the array is called an element.

A=10,17,2,5,8 A is an array & 10 is an element of array A

To refer to a specific element within an array a subscript must be specified. This subscript refers to the position of the data element within the array. For example, in array A the number 10 is in the first position, therefore A(1)=10 and A(4)=5.

When using arrays in a program one must first tell the computer the maximum size of each array to be used. This is done through a DIM statement.

DIM array

Ex. DIM X(15),Y(15),N$(15)

One is not required to use the entire array but an upper limit must be specified.

In the example below the computer is given the task of sorting a set of numbers contained within an array. This particular approach to sorting is

called a bubble sort because the lowest number "bubbles up" to the first position of the array.

```
10  REM PROGRAM TO SORT DATA IN ASCENDING ORDER

20  DIM X(50)                  Array can have up to 50 elements

30  INPUT  "ENTER NUMBER OF VALUES";N

40  FOR I=1 TO N

50  PRINT "ENTER VALUE";I      Note loops are used to get data into arrays

60  INPUT X(I)

70  NEXT I

80  REM SORTING ROUTINE

85  SW=0                       Indicates whether had to switch position of
                               elements in array
90  FOR I=1 TO N-1

100 IF X(I) < X(I+1) THEN GOTO 150  Two elements in proper order

110 T=X(I+1)                   T is temporary variable

120 X(I+1)=X(I)                Lines 110-139 switch element positions

130 X(I)=T

140 SW=1                       Switch was made

150 NEXT I

160 IF SW=1 THEN GOTO 85       Repeat process to check array

170 PRINT "SORTED ARRAY:"

180 FOR I=1 TO N

190 PRINT X(I)

200 NEXT I

210 END
```

The arrays discussed to this point have one dimension, they can be thought of as either a row or column of data. Very often it is useful to store data in two dimensions (rows and columns) as is the case in a Lotus 1-2-3 worksheet. This can be done in BASIC as well through the use of matrices. A matrix is nothing more than an array of more than one dimension. As is the

case for arrays, matrices must be dimensioned through the use of a DIM statement before they can be used in a program.

Ex. DIM B(10,3)

In the two-dimensional case, the first number refers to the number of rows and the second to the number of columns. B, then, is a 10 by 3 matrix containing 10 rows and 3 columns. To refer to a specific element of a matrix, one must specify the appropriate subscripts just like in Lotus where one can access a cell by specifying its row and column location.

The following is an example of a program which uses both arrays and matrices to calculate semester averages and assigns letter grades for classroom teachers.

```
10  REM PROGRAM TO CALCULATE SEMESTER GRADES FOR TEACHERS

20  DIM TEST(25,5),AVTEST(25),N$(25),GRADE$(25)  Class can have up to
                                                 25 students & 5 tests
30  INPUT "ENTER NUMBER OF STUDENTS";N

40  INPUT "ENTER NUMBER OF TESTS";T

50  FOR I=1 TO N                    I (row) represents student

60  INPUT "ENTER STUDENT NAME";N$(I)

70  FOR J=1 TO T                    J (col) represents test

80  PRINT "ENTER TEST";J

90  INPUT TEST(I,J)                 Note that it takes loop within
                                    loop to get data into matrix
100 AVTEST(I)=AVTEST(I)+TEST(I,J)   Sum test scores

110 NEXT J

120 AVTEST(I)=AVTEST(I)/T           Calculate average test score

130 NEXT I

140 REM ASSIGN GRADES & PRINT RESULTS

150 PRINT "NAME";TAB(20);"AVERAGE";TAB(30);"GRADE"  Prints column
                                                    headings
160 FOR I=1 TO 25

170 IF AVTEST (I) >= 90 THEN GRADE$(I)="A"     Lines 170-210 assign letter
                                               grades based on average
180 IF AVTEST (I) < 90 THEN GRADE$(I)="B"      test score

190 IF AVTEST (I) < 80 THEN GRADE$(I)="C"
```

```
200 IF AVTEST (I) < 70 THEN GRADE$(I)="D"

210 IF AVTEST (I) < 65 THEN GRADE$(I)="F"

220 AVTEST (I) = INT ((AVTEST(I)*10)+0.5)/10     rounds average to
                                                 nearest tenth
230 PRINT N$(I);TAB(21);AVTEST(I);TAB(32);GRADE$(I)

240 NEXT I

250 END
```

Data Files

The programs discussed up to this point have used either READ/DATA or INPUT statements to get data into a program. Both approaches have their limitation in that for READ/DATA statements the data must be part of every program that uses the data and for INPUT statements the data must be re-entered each time the program is run. Neither works well if the task is to use the same data in a number of different programs.

If one has a case where the same set of data is going to be used by a number of programs, probably the most efficient way to handle the data is to put it in a data file. Each program, then, will access that data file and that way the data will only have to be entered into the computer once.

There are two types of data files: sequential, which require that the data be read in sequence from first to last, and random, which allow data to be read or accessed in any order. This discussion will concern itself with only sequential files for they are easier to create and are accessible by Lotus 1-2-3.

Data files are usually created by programs. Within these programs are statements that inform the computer of the presence of files, read from files, write to files, and instruct the computer that the program is finished with the files. They are as follows:

OPEN "mode",#file number,"file name"	Tells computer to open a file for use
OPEN "O",#1,"GRADES.DAT"	Open a file for output ("O"), refer to it as file #1 in the program, and store it on the disk under the name GRADES.DAT (note that both file number & name are arbitrary)
OPEN "I",#2,"TEST.DATA"	Open a file for input ("I"), refer to it as file #2 in the program, and look for it on the disk under the same TEST.DAT

Note that output means that you are writing out to the file (creating it) and input means you are reading from the file into the program (reading the data).

WRITE#___,var list	Tells computer to write specified data to file.
WRITE#1,A,B,C	Write A,B,C to file #1
INPUT#_,var list	Tells computer to read data from file and assign to variables.
INPUT#2,A,B,C	Read data from file #2 and assign to variables A,B,C
CLOSE #_	Tells computer program finished file. Whenever a file has been OPENed it must be CLOSEd.
CLOSE #1	Close file #1

The following two programs create and then access a sequential data file which contains student enrollments.

```
10 REM PROGRAM TO SEQUENTIAL DATA FILE FOR ENROLL DATA

20 DIM ENR(5)

30 OPEN "O",#1,"ENROLL.PRN"        Opens for output file #1 to be
                                   saved under name ENROLL.PRN
40 INPUT "ENTER NUMBER OF YEARS";NY

50 FOR I=1 TO NY

60 INPUT "ENTER ENROLLMENT YEAR";YR$

70 FOR J=1 TO 5

80 PRINT "ENTER ENROLLMENT FOR YEAR ";YR$;" GRADE";J

90 INPUT ENR(J)

100 NEXT J

110 WRITE#1,YR$,ENR(1),ENR(2),ENR(3),ENR(4),ENR(5)  Write data to file#1

120 NEXT I

130 CLOSE #1     Close file #1

140 END

10 REM PROGRAM TO READ SEQUENTIAL DATA FILE - ENROLLMENT DATA

20 DIM ENROLL(5)
```

```
30 OPEN "I",#1,"ENROLL.PRN"        Open file ENROLL.PRN for reading
                                   (input) as file #1
40 IF EOF(1) THEN GOTO 130         When the end of file #1 reached go to
                                   line 130 (EOF-end of file)
50  INPUT#1,YR$                    Read from #1 data & call it YR$

60  FOR J=1 TO 5

70  INPUT#1,ENROLL(J)              Read enrollment data from file #1

80  TOTAL=TOTAL+ENROLL(J)          Sum enrollments for year

90  NEXT J

100 PRINT "YEAR ";YR$;TAB(15);"TOTAL ENROLLMENT="; TOTAL

110 TOTAL=0                        Set total back to 0

120 GOTO 40                        Go back and repeat process

130 CLOSE #1

140 END
```

Lotus 1-2-3 and BASIC data files

As was mentioned earlier, Lotus 1-2-3 can access BASIC sequential data file. This is useful if you want to get the results from a BASIC program into Lotus. For example, Lotus cannot do factor analysis. A program could be written in BASIC that could do factor analysis, computer factor scores, and write the factor scores to an output file. This file could then be copied into Lotus for further analysis.

The process to copy a data file into Lotus is relatively easy. The only constraint that is placed on the user is that the data file to be copied must have as its file extension PRN (for example, ENROLL.PRN). If the file does not already have this extension, the name must be changed before the copy can take place.

Once the name is correct the process to copy the data is to place the cursor where you want the data to begin and

/File Import Numbers

This will cause each row of data in the file to be treated as a row in a Lotus worksheet and each piece of data in the row to occupy a separate column with all data enclosed within double quotes (") in the file to be treated as a left-aligned label and all numbers treated as numbers.