5-1985

Data Analysis Using Experimental Design Model Factorial Analysis of Variance/Covariance (DMAOVC.BAS)

Wesley E. Newton

DATA ANALYSIS USING EXPERIMENTAL DESIGN MODEL FACTORIAL ANALYSIS OF VARIANCE/COVARIANCE (DMAOVC.BAS)

by

Wesley E. Newton

A thesis submitted in partial fullfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Applied Statistics

Approved:

UTAH STATE UNIVERSITY
Logan, Utah

1985

ACKNOWLEDGEMENTS

I would like to thank all of the faculty members in the Department of Applied Statistics for the educational experience they each provided for me. I am especially thankful for the their time and efforts they gave. Their professional example is something I hope to carry with me the most. I wish them all the best.

Special thanks go to my father, Ray, my mother, Kaye, and sisters, Becky and Bobbi, for their love and support. I also wish to thank my grandmother, Maurine Fowler, and a good friend Steve McCaslin for their continued support and encouragement, of which I could not have completed this without.

Finally, to my wife, Robin, for her patience, love and support in completing this assignment.

Wesley E. Newton

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Data Analysis Using Experimental Design Model
Factorial Analysis of Variance/Covariance
(DMAOVC.BAS)

by

Wesley E. Newton, Master of Science
Utah State University, 1985

Major Professor: Dr. Rex L. Hurst
Department: Applied Statistics

DMAOVC.BAS is a computer program written in the compiler version of microsoft basic which performs factorial analysis of variance/covariance with expected mean squares. The program accomodates factorial and other heirarchical experimental designs with balanced sets of data. The program is written for use on most modest sized microprocessors, in which the compiler is available. The program is parameter file driven where the parameter file consists of the response variable structure, the experimental design model expressed in a similar structure as seen in most textbooks, information concerning the factors (i.e. fixed or random, and the number of levels), and necessary information to perform covariance analysis. The results of the analysis are written to separate files in a format that can be used for reporting purposes and further computations if needed.

(108 pages)

CHAPTER I

HISTORY AND PURPOSE

A computer program was developed by Dr. Rex L. Hurst, Utah State University, Logan, Utah, for microprocessors called Factorial Analysis of Variance/Covariance (FCT.BAS). FCT.BAS is written in the compiler version of microsoft basic and can be used on any microprocessor where the compiler is avaiable. FCT.BAS computes a factorial analysis of variance/covariance for balanced sets of data. FCT.BAS requires that an extensive, rigidly structured job parameter file be constructed as a "driver" for execution. Although this is acceptable for less complicated and smaller experimental designs, a great deal of effort and time is required in constructing this parameter file. For more information and description of FCT.BAS (Hurst, n.d.), the Department of Applied Statistics, Utah State University, Logan, Utah, has documentation available, called FCT.DOC (Hurst, .n.d).

Dr. David Turner, also of Utah State University, suggested simplification of the parameter file required by FCT.BAS allowing FCT.BAS to be more "user friendly" and easier to execute. Thus, a masters project was developed to simplify this "driver" parameter file and add some other

statistical capabilities to the program. The remainder of this document discusses and should serve as a users' guide to this new program, called DMAOVC.BAS (DESIGN MODEL ANALYSIS OF VARIANCE/COVARIANCE). It is assumed the reader and user has had some background in experimental design, experimental design models, and the "traditional" computational approach to factorial analysis of variance/covariance for balanced sets of data. If not, a review of an experimental design text would be useful.

CHAPTER II

PROGRAM DESCRIPTION

Introduction

DMAOVC.BAS will compute factorial anlaysis of variance with expected values of mean squares for balanced sets of data based on an input experimental design model. DMAOVC.BAS will also compute analysis of covariance for any number of models based on the same input experimental design model. DMAOVC.BAS requires that the input design model be on a parameter file, along with other descriptive specifications (see part 3). By using this input parameter file, DMAOVC.BAS will derive and output the analysis of variance table, expected values of mean squares, analysis of covariance results (if required) and store intermediate computational values in separate data files (see part 3).

DMAOVC.BAS requirements and parameter specifications are the same as FCT.DOC (Hurst, n.d.) and reproduced below.

1. The program is written in the compiler version of Microsoft MBASIC.
2. The program should run on any microprocessor for which the compiler is available.
3. The dimension statements may have to be adjusted for the smaller 8 bit machines.
4. The program will handle up to 20 response variables. That is, the sum of the independent and dependent variables must equal 20.
5. It will handle up to 8 factors, including replications.

6. The program is designed to accommodate the accumulation of up to 2000 totals. This means that the product of the number of variables by the number of levels per computation needed, must not be greater than 2000. As an example a 3*4*2*5 factorial with 3 variables would require 3*(3*4*5) = 180 totals to obtain the "abd" set of totals.
7. This program can be made to handle any size problem within the capability of the machine by adjusting the dimension statements. See Part 5.4 for making additions, deletions or changes.
8. The user must set up a parameter file which describes the problem to be processed. Methods for constructing this parameter file will be taken up in Part 3.3.
9. DMAOVC.BAS does not check to see if your design is balanced. If you use DMAOV.BAS for unbalanced data, you must assume responsiblity for the validity of the output.
10. The program requires eight data files to be open at one time, therefore the microprocessor must have this capability.

Design Models

As a review, experimental design models, also termed analysis of variance models, are additive models used to describe the structure of an experiment being analyzed. In a given experiment, a response variable(s) is measured under various experimental conditions identified by classification variables or factors. The variation in the response variable could be due to or "explained" by effects in the classification with random error, if any, accounting for the remaining variation. For purposes of discussion, a two-factor analysis of variance will be illustrated. Each response or observation can be thought of in terms of the experimental design model;

$$Y_{ijk} = U + A_i + B_j + AB_{ij} + E_{k(ij)}$$

where: $i = 1,...a$ (number of levels of factor A)

j = 1,...b (number of levels of factor B)
k = 1,...n (number of levels of factor C, or in this case replications)
U = overall population mean for this particular experiment
A_i = the ith level effect of factor A; where the factor can be fixed or random
B_j = the jth level effect of factor B; where the factor can be fixed or random
AB_{ij} = the interaction or crossed effect between ith level of factor A and jth level of factor B
$E_{k(ij)}$ = random error or "unexplained" variation nested within the factor levels

note: a. subscript or index "i" indicates a main effect
b. subscript or index "j" indicates a main effect
c. subscripts or indices ij (without parantheses) indicates a crossed or interaction effect
d. subscripts or indices k(ij) indicates k is nested within ij

By examination of the above model, information can be communicated to DMAOVC.BAS as to the relationship among the factors by examining the indices; i.e. is a factor nested (or crossed) and if it is nested (or crossed) with which factor(s) is it nested within (or crossed with). Thus, DMAOVC.BAS can then compute factorial analysis of variance/covariance and derive expected mean squares by exploiting the ideas above for balanced data. That is, the program works by knowing or describing the experiment to be analyzed with an input experimental design model from a parameter file. The parameter file will have to define the response variable, the model, which factors are fixed or random, the number of levels for each factor (including replications), and their respective relationships. The exact procedures for this are described in part 3.

Computing Methods

Based on the information provided by the input experimental design model as described in part 2.2, DMAOVC.BAS computes the appropriate analysis of variance by using the traditional, or "classical" approach for balanced sets of data. The traditional approach is to lay out the analysis of variance in symbolic form. This is done by first defining the sources of variation and using the symbolic form of the degrees-of-freedom as a key to the needed sums-of-squares. A large number of statistical textbooks, namely experimental design texts, describe this approach to the anlysis of variance. It is assumed the reader and user are acquainted with this approach. If not, some texts are given as references for review (Dowdy and Wearden, 1983; Gill, 1978; Kirk, 1982; Steel and Torrie, 1980; Winer, 1971).

DMAOVC.BAS first parse's the input experimental design model into respective design model components or terms, which are really the sources of variation. The program then examines the indices for each design model component and derives their respective linear combinations of binary code.[1] For or each unique binary code orstring,DMAOVC.BAS computes the uncorrected sums-of-squares. Then by taking the linear combinations, computes the sums-of-squares for

[1]Appendix A contains the algorithm used to derive these linear combinations of binary code based on the indices for each individual design model component.

the respective sources of variation. As an example, let's look at the two-way factorial model described in part 2.2. The model;

$$Y_{ijk} = U + A_i + B_j + AB_{ij} + E_{k(ij)}$$

is first parsed into the sources of variation: A_i, B_j, AB_{ij}, and $E_{k(ij)}$. By examining the indices, the linear combinations of binary code are derived, with the following results;

$$\begin{aligned} A_i &= 100 - 000 \\ B_j &= 010 - 000 \\ AB_{ij} &= 110 - 100 - 010 + 000 \\ E_{k(ij)} &= 111 - 110 \end{aligned}$$

where each binary code represents uncorrected sums-of-squares; i.e. 001 = Y(..k)/ab, 110 = Y(ij.)/c, etc. Then, from this information, the appropriate analysis of variance is developed.

Using the same design model components, indices, and knowing whether a factor is fixed or random, DMAOVC.BAS uses the rules as prescribed in Kirk (1982) to derive the appropriate expected mean squares.

The covariance analysis also uses the traditional computational approach as described above for balanced sets of data. However more information must be given than just the experimental design model to compute the covariance analysis, as will be seen in part 3.

CHAPTER III

PROGRAM USAGE

Specifying the Design Model

The easiest way to learn how to use DMAOVC.BAS is to work through the following experiment.

> Because of energy shortages, oil companies are considering secondary and even tertiary recovery methods for obtaining more petroleum from exhausted oil wells. These methods attempt to free the oil from porous rock so that it can be pumped from the ground. To compare three such methods, an oil company takes a random sample of four exhausted oil fields and tries each method on two wells at each field. The results (in barrels of oil per day) are given in figure 1.

		Oil Field (Factor B)			
		1	2	3	4
Method (Factor A)	Mechanical fracture	2	4	3	1
		1	2	1	1
	Carbon dioxide	4	3	6	6
		5	3	7	5
	Pressurized steam	6	8	7	5
		4	8	8	6

Figure 1
Three by Four Factorial AOV (Dowdy, 1983 page 335).

The first step is to specify or describe the experiment with an experimental design model using the notation discussed in part 2. The experimental design model for this

example is the same as that described in part 2 and rewritten here in terms of the example problem;

$$Y_{ijk} = U + A_i + B_j + AB_{ij} + E_{k(ij)}$$

where:
- U = overall population mean for this particular experiment
- A_i = ith level of Method (Factor A), indexed by i
- B_j = jth level of Oil Field (Factor B), indexed by j
- AB_{ij} = the interaction effect of Factor A and Factor B, indexed by ij
- $E_{k(ij)}$ = random replication error nested within Factor A and Factor B, indexed by k(ij)

The second step is to designate which factors are fixed and which factors are random, and also their respective number of levels. For the above example the folling designations are made;

- i = 1 to 3 levels of Factor A (Method), where Factor A is a fixed factor
- j = 1 to 4 levels of Factor B (Oil Field), where Factor B is a random factor
- k = 1 to 2 levels of Fctor C (Replications), where Factor C is a random factor

Once these two steps are defined, we are ready to construct the job parameter file used by DMAOVC.BAS to conduct the appropriate analysis of variance with expected mean squares. This will be taken up in part 3.

Constructing the Job Parameter File

After specifying the experimental design model, as we did for the example in part 3, we are ready to construct the job parameter file required by DMAOVC.BAS. There are six basic steps in construction of the parameter file. Again, the easiest way to do this is by illustration with

the example in part 3. There we saw the experimental design model as;

$$Y_{ijk} = U + A_i + B_j + AB_{ij} + E_{k(ij)}.$$

The first step in constructing the job parameter file is to write the response variable, Yijk, in a notation communicatable to DMAOVC.BAS. To do this simply enclose the indices in parantheses without spaces between them, and also between the "(" and "Y". This is illustrated as follows;

Response Variable: Y_{ijk} ==> Y(IJK) or Y(ijk)

We could also write the response varaible in a more descriptive way, relating it to the problem being analyzed, in this case as follows;

Response Variable Y_{ijk} ==> BARRELS(IJK) or Barrels(ijk)

However, the total length of the Response Variable description should not exceed 20 characters total; e.g. BARRELS(IJK) has twelve characters.

The second step in constructing the job parameter file is to write the rest of the experimental design model in notation communicatable to DMAOVC.BAS. To do this, again simply enclose the indices in parentheses without spaces between them. If a factor is nested within other factors, this will be denoted by separation with a "/". This is illustrated for figure 1 as follows;

The Design Model;

$$U + A_i + B_j + AB_{ij} + E_{k(ij)}$$

would be written as;

A(I)+B(J)+AB(IJ)+E(K/IJ)

for DMAOVC.BAS. This could also be written as;

A(i)+B(j)+AB(ij)+E(k/ij)

depending on whether we used upper or lower case one character alphabetic names for the indices in the response variable description. One thing to notice is U is assumed and is left out of the DMAOVC.BAS model. We can also be more descriptive of the factors by writing the model as;

METHOD(I)+FIELD(J)+M*F(IJ)+E(K/IJ).

The model can be shortened by writing the E(K/IJ) term as RESIDUAL, Residual, or residual, as follows;

A(I)+B(J)+AB(IJ)+RESIDUAL

or can be left off completely as follows;

A(I)+B(J)+AB(IJ)

and will be computed as a residual.

Four conditions of the model are:

1. The total length of each term in the model cannot exceed 20 characters; e.g. METHOD(I) has nine characters.
2. Each term in the model must be separated by a "+".
3. There can be spaces between the design model terms, but not within, e.g. A(I) + AB(IJ) is allowable where as A(I) + A B(I J) is not allowed.
4. The indices in the model terms must ALWAYS be in the same order as that defined in the response variable. Examine the following examples carefully.

   If the Response Variable is Y(IJKL) then the following design model terms are acceptable:

   ABD(IJL) ABCD(IJKL) D(J/IKL) CD(KL/IJ).

   and the following design model terms are NOT acceptable:

   ABD(JIL) ABCD(KILI) D(J/KIL) CD(LK/JI).

note: The user is responsible for validating that the design model term indices remain in the same order as that defined in the response variable. This is true on both sides of the "/".

The third step in constructing the job parameter file is to specify in some way or via a command to let DMAOVC.BAS know what the response variable is and what the design model is. This is done via DMAOVC.BAS five character commands that allows DMAOVC.BAS to know what follows. Again this is best illustrated with an example.

To specify the response variable, simply precede it with the command "REVAR" (upper case letters only) and a space as shown in the following example.

```
REVAR Y(IJK)    or
REVAR Y(ijk)    or
REVAR BARRELS(IJK)
```

The format, or location of the response variable in the data file, must also be specified with the REVAR command as the following illustrates;

```
REVAR Y(IJK),c,w,d;
```

where: c = the beginning column in a sequential data file
w = the width of the response variable field, decimal point included if appropriate
d = the number of numerical values beyond the decimal point

note: ";" should always follow the number of numerical values beyond the decimal point, this indicates to DMAOVC.BAS the end of the command

To specify the design model to DMAOVC.BAS, simply precede it with the "MODEL" command (upper case letters only) and a space as shown in the following for figure 3.1.

```
REVAR Y(IJK),5,1,0;
MODEL A(I)+B(J)+AB(IJ)+E(K/IJ);
```

note: ";" should always follow the last term in the model, this indicates to DMAOVC.BAS the end of the command

If more than one line is needed for the design model, multiple MODEL commands can be used as the following illustrates.

If the design model is;

$$Y_{ijkl} = U + A_i + B_j + C_k + AB_{ij} + AC_{ik} + AC_{ik} + AD_{il} + BC_{jk} + BD_{jl} + CD_{kl} + E_{ijkl}$$

this is set up in the parameter file as;

```
REVAR Y(ijkl),c,w,d;
MODEL A(i) + B(j) + C(k) + D(l);
MODEL AB(ij) + AC(ik) + AD(il);
MODEL BC(jk) + BD(jl) + CD(kl) + RESIDUAL;
```

note: each MODEL command must end with a ";" after the last design model term

The fourth step is to specify the factors that are random or fixed, and also their respective number of levels. This is done by either using the "FIXED" (fixed factor command) or the "RANDM" (random factor command). This is illustrated for figure 1 as follows;

```
FIXED I,3;
RANDM J,4;
RANDM K,2;
```

where: I is a fixed factor with 3 levels
J is a random factor with 4 levels
K is a random factor with 2 levels

note:
a. always put a single space between the command and the factor level
b. separate index and number of levels with a ","
c. ";" should always follow the number of levels, indicates to DMAOVC.BAS the end of the command
d. again the indices must be either upper-case or lower-case depending on how they are defined in the REVAR command

The fifth step for constructing the parameter file is to designate to DMAOVC.BAS the name of the data file and the disk drive location. This is accomplished with the command

"DATFN", and illustrated as follows;

```
DATFN B:OIL.DAT;
```

where: B: indicates the disk drive location
OIL is the filename
.DAT is the filename extension

note: ";" should always follow the the filename and extension, indicates to DMAOVC.BAS the end of the command

The sixth step in constructing the parameter file is to load it on a floppy diskette, usually the same one as that which contains the data file. The output will go to the same drive as that which the parameter file is on. The parameter file for example 3.1, with all the commands pulled together, was placed on a floppy diskette using Wordstar in non-document mode, as follows;

Parameter File B:OIL.PAR

```
REVAR Y(IJK),5,1,0;
MODEL A(I)+B(J)+AB(IJ)+E(K/IJ);
FIXED I,3;
RANDM J,4;
RANDM K,2;
DATFN B:OIL.DAT;
```

note: the only restriction on the order of the DMAOVC.BAS commands is that the REVAR command must be specified before the FIXED or RANDM commands.

Before executing DMAOVC.BAS, a discussion of the data file structure needs to be examined. This is done in the next section, part 3.

Data File Structure

The data records must be sorted into lexographic order on the factor indices as defined by the response variable,

REVAR, structure. Using figure 1 to illustrate, the response variable appeared as Y(IJK), since the index "I" is first, "J" is second, and "K" is third, the the data must be sorted as "K" within "J" within "I". Below is a portion of the data of example 3.1 to illustrate the sorting order.

```
Data Filename  B:OIL.DAT
-------------------------

111 2
112 1
121 4
122 2
... .
... .
... .

341 5
342 6
```

note: the levels for each factor's indices (e.g. 111,112,...) are not used by DMAOVC.BAS, these are included in the data file for easier interpretation by the user

Executing DMAOVC.BAS

The execution of DMAOVC.BAS requires the compiled version of DMAOVC.BAS. Once this has been accomplished, and the job parameter file and data file have been constructed on disk files, DMAOVC.BAS is ready for execution. To execute DMAOVC.BAS, simply type the disk drive location and DMAOVC as follows;

```
A> A:DMAOVC <RET>
```

After the program title, DMAOVC.BAS prompts the user for the name and disk drive location of the job parameter file. Simply answer this prompt as follows;

ENTER DISK DRIVE AND JOB PARAMETER FILE NAME
SEPARATE WITH A COLON (e.g. B:FILENAME.EXT)? B:OIL.PAR

DMAOVC.BAS will then display the job parameter file and prompt for a CONTINUE(Y or N)?. This allows the user to double check the parameter file one more time for typing errors, input accuracy, etc. After responding to this prompt, DMAOVC.BAS will then display the first five observations of the data file and give the same prompt to continue or not. This allows the user to review the data and input accuracy. After responding to this prompt, DMAOVC.BAS will then display what is currently being processed to inform the user as to the execution progress.

After execution, DMAOVC.BAS will then display the various files that are created, some of these are output files for reporting purposes, others are just work files used by DMAOVC.BAS. The output files created are described more fully in part 3.

Execution Results

There are six files created during execution for an analysis of variance with expected mean squares. These are listed below and described more fully in the following sections.

	Files Created	Short Description
1.	***AOV.OUT	Contains the Analysis of Variance Table
2.	***EMS.OUT	Contains the Expected Mean Squares
3.	***AVG.OUT	Contains the Overall and Cell Means
4.	***TOT.OUT	Contains the Totals Used
5.	***USS.OUT	Contains the Uncorrected Sums-of-Squares
6.	***COP.DAT	Contains a Copy of the Data File

note: *** = the first three characters of the parameter file name

Analysis of Variance Table

The analysis of varaince table is contained on ***AOV.OUT. The analysis of varaince table contains the variable (VAR), the sources of variation (SOURCE), the degrees of freedom (DF), the sums-of-squares (SS), and mean squares (MS). Below is a print out of the file OILAOV.OUT for figure 1.

Table 1
AOV Table for Example Data

ANALYSIS OF VARIANCE TABLE

VAR	SOURCE	DF	SS	MS
1	TOTAL	23	0.12783333D+03	0.55579710D+01
1	A(I)	2	0.88083333D+02	0.44041667D+02
1	B(J)	3	0.98333333D+01	0.32777778D+01
1	AB(IJ)	6	0.20916667D+02	0.34861111D+01
1	E(K/IJ)	12	0.90000000D+01	0.75000000D+00

NOTE:
VAR 1 = Y(IJK)

Expected Mean Squares

The expected mean squares are contained on ***EMS.OUT. Below is a print out of OILEMS.OUT for example 3.1 containing the format of and the expected mean squares.

SOURCE	EXPECTED VALUES OF MEAN SQUARES
A(I)	(V)K/IJ + K(V)IJ + JK(V)I
B(J)	(V)K/IJ + IK(V)J
AB(IJ)	(V)K/IJ + K(V)IJ
E(K/IJ)	(V)K/IJ

where: (V) = sigma squared
indices to the left of (V) are the coefficients
indices to the right of (V) are the subscripts

note: if the there is any residual sums-of-squares, the notationfor this would be "(V)RES" which represents sigma squared sub error (see part 4 for examples)

Overall and Cell Averages

The overall and cell averages are contained on ***AVG.OUT. Below is a print out of OILAVG.OUT for figure 1 containing the format of the various averages.

```
AVERAGES FOR 111
------------
DIV =  24
LEVEL = OVERALL
 4.416666666666667

AVERAGES FOR 100
------------
DIV =  8
LEVEL 1
 1.875
LEVEL 2
 4.875
LEVEL 3
 6.5

AVERAGES FOR 010
------------
DIV =  6
LEVEL 1
 3.666666666666667
LEVEL 2
 4.666666666666667
LEVEL 3
 5.333333333333333
LEVEL 4
 4
```

```
AVERAGES FOR 010
------------
DIV = 2
LEVEL 1  1
 1.5
LEVEL 1  2
 3
LEVEL 1  3
 2
LEVEL 1  4
 1
LEVEL 2  1
 4.5
LEVEL 2  2
 3
LEVEL 2  3
 6.5
LEVEL 2  4
 5.5
LEVEL 3  1
 5
LEVEL 3  2
 8
LEVEL 3  3
 7.5
LEVEL 3  4
 5.5
```

note: if there is more than one response variable, the averages will be print on the same line for each response variable

Totals Used

***TOT.OUT is an unformatted work file used by DMAOVC.BAS to compute various values (e.g. averages, adjusted averages, etc). Below is a print out of OILTOT.OUT for figure 1. This file is saved on the disk, but does not have much use as far as reporting purposes and can be deleted if desired. The reason it is saved is that at a later date some additional computations might be desired, therefore it is still available for use.

```
 1  24     ==>          1 = indicates the number of totals
 106                  24 = the number of observations
 3  8                    to obtain the total
 15                  106 = indicates the total for response
 39                       variable one
 .                     .
 .                     .
 .                     .
 etc.                 etc.
```

Uncorrected Sums
Of Squares Used

***USS.OUT contains the uncorrected sums-of-squares used to compute the analysis of variance table. Below is a print out of OILUSS.OUT for figure 1. This file is an unformatted work file and is saved for some of the same reasons as described in above and can be deleted if desired.

```
111         ==>          111 = the uncorrected sums-of-squares
 24                           binary code
596                      24 = the degrees of freedom for 111
000                     596 = the uncorrected sums-of-squares
 1                           for response variable one
 468.1666666666667         .
100                        .
 3                         .
 556.25                  etc.
010
 4
```

```
 478
110
12
 587
```

Copy of the Data File

***COP.DAT is a copy of the raw data. DMAOVC.BAS makes a copy of the data file and then uses it as a work file in a format more easily used by DMAOVC.DAT. The file is saved for the same reasons as described above and can be deleted if desired.

Options Available

There are three options availble in DMAOVC.BAS. These are described in the following sections.

The TITLE Command

A title can be placed at the top of the analysis of variance table, the expected mean squares file, and the covariance analysis. This is done for the purpose of labeling the output for reporting purposes. DMAOVC.BAS allows up to twenty lines of title. The following illustrates how the title command is specified in the parameter file.

```
TITLE                    OIL RECOVERY METHODS
TITLE                      A TWO-FACTOR CRD
TITLE                  DOWDY AND WEARDON PAGE 335
REVAR Y(IJK),5,1,0;
MODEL A(I)+B(J)+AB(IJ)+E(K/IJ);
FIXED I,3;
RANDM J,4;
RANDM K,2;
DATFN B:OIL.DAT;
```

note: the TITLE can be placed anywhere in the parameter file, but it is best to place it at the top,this also allows a description of the parameter file.

***AOV.OUT would then look similar to the following for figure 1.

Table 2
AOV Table for Example Data with a Title

OIL RECOVERY METHODS
TWO-FACTOR CRD
DOWDY AND WEARDON PAGE 335

ANALYSIS OF VARIANCE TABLE

VAR	SOURCE	DF	SS	MS
1	TOTAL	23	0.12783333D+03	0.55579710D+01
1	A(I)	2	0.88083333D+02	0.44041667D+02
1	B(J)	3	0.98333333D+01	0.32777778D+01
1	AB(IJ)	6	0.20916667D+02	0.34861111D+01
1	E(K/IJ)	12	0.90000000D+01	0.75000000D+00

NOTE:
VAR 1 = Y(IJK)

and ***EMS.OUT would look similar to the following for figure 1.

Table 3
Expected Mean Squares for Example Data

OIL RECOVERY METHODS
TWO-FACTOR CRD
DOWDY AND WEARDON PAGE 335

SOURCE	EXPECTED VALUES OF MEAN SQUARES
A(I)	(V)K/IJ + K(V)IJ + JK(V)I
B(J)	(V)K/IJ + IK(V)J
AB(IJ)	(V)K/IJ + K(V)IJ
E(K/IJ)	(V)K/IJ

Expected Mean Squares Only

Sometimes it is necessary to determine the expected

mean squares for a particular design model without actually processing a set of data. DMAOVC.BAS can do this by using the EMSQR command in the parameter file. The parameter file for example 3.1 would then be specified as follows;

```
TITLE                    OIL RECOVERY METHODS
TITLE                       TWO-FACTOR CRD
TITLE                  DOWDY AND WEARDON PAGE 335
REVAR Y(IJK);
MODEL A(I)+B(J)+AB(IJ)+E(K/IJ);
FIXED I;
RANDM J;
RANDM K;
EMSQR
```

by leaving out the DATFN command, leaving off the response variable format in the REVAR command, leaving off the number of levels in the FIXED and RANDM commands, leaving the MODEL command as is, and completing the parameter file with the EMSQR command.

The parameter file is now ready to be processed by DMAOVC.BAS. ***EMS.OUT, in our example OILEMS.OUT, is the only file to be created upon execution.

Covariance Analysis

DMAOVC.BAS will perform analysis of covariance for any number of models with up to twenty response variables, sum of independent and dependent. In order for DMAOVC.BAS to perform analysis of covariance, the parameter file has to be more specific in its structure and ability to communicate the required information to DMAOVC.BAS. The easiest way to learn the appropriate parameter file structure is to look at the following example.

Let us suppose in figure 1 that in addition to the number of barrels of oil a day being recovered, another variable, or covariate was also measured (e.g. time), denoted as "X", and thought to be linearly related to barrels of oil per day, denoted as "Y". One other assumption that we need to make is that Factor B (Oil Field) is fixed. To have DMAOVC.BAS conduct this analysis of covariance, five new commands are used in addition to those already used. Let's look at the parameter file constructed thus far below, without the TITLE option, and changing RANDM J,4; TO FIXED J,4;.

```
REVAR Y(IJK),5,1,0;
MODEL A(I)+B(J)+AB(IJ)+E(K/IJ);
FIXED I,3;
FIXED J,4;
RANDM K,2;
DATFN B:OIL.DAT;
```

So far the parameter file will perform analysis of variance for only the response varaible, barrels of oil per day. When there are more than one response variable, we have to specify this in the parameter file by adding another REVAR command as follows;

```
REVAR Y(IJK),5,1,0;
REVAR X(IJK),7,2,1;
MODEL A(I)+B(J)+AB(IJ)+E(K/IJ);
FIXED I,3;
FIXED J,4;
RANDM K,2;
DATFN B:OIL.DAT;
```

where in the REVAR command we also have to specify the beginning column, width, and the number numerical values following the decimal point for X. The parameter will

specify to DMAOVC.BAS to conduct separate analysis of variance for each response variable.

The command COVAR will now be used in the parameter file to specify to DMAOVC.BAS that covariance anlaysis is desired as follows;

```
REVAR Y(IJK),5,1,0;
REVAR X(IJK),7,2,1;
MODEL A(I)+B(J)+AB(IJ)+E(K/IJ);
FIXED I,3;
FIXED J,4;
RANDM K,2;
DATFN B:OIL.DAT;
COVAR 1;
```

where the "1" specifies that one covariance model or anlaysis is desired.

DMAOVC.BAS now needs to know which response variable(s) is the independent variable(s) and which response variable is the dependent variable. This is done by using the "INDEP" and "DEPEN" commands as follows;

```
REVAR Y(IJK),5,1,0;
MODEL A(I)+B(J)+AB(IJ)+E(K/IJ);
FIXED I,3;
FIXED J,4;
RANDM K,2;
DATFN B:OIL.DAT;
COVAR 1;
INDEP 2;
DEPEN 1;
```

where "INDEP 2;" indicates to DMAOVC.BAS that the independent variable is number 2 in the order given in the parameter file and "DEPEN 1;" indicates to DMAOVC.BAS that the dependent variable is number 1 in the order given in the parameter file. The DEPEN command must always follow the INDEP command and both must follow the COVAR command.

DMAOVC.BAS now needs to know the different error lines and sources of variation desired in the covariance analysis. These are specified by the following commands illustrated as follows;

```
ERRLN E(K/IJ);
SORCE A(I)+E(K/IJ);
SORCE B(J)+E(K/IJ);
SORCE AB(IJ)+E(K/IJ);
```

where we see the ERRLN command is listed first, followed by the term E(K/IJ); inidicating to DMAOVC.BAS the error line is this model term. The SORCE commands are the various sources of variation associated with the analysis of covariance for this particular experiment. Two items to make note of is: (1) the ERRLN command must be listed first then the corresponding SORCE commands and (2) like all other DMAOVC.BAS commands, each one ends with a ";".

Using this information, the total parameter file would now look like the following;

```
REVAR Y(IJK),5,1,0;
REVAR X(IJK),7,4,1;
MODEL A(I)+B(J)+AB(IJ)+E(K/IJ);
FIXED I,3;
FIXED J,4;
RANDM K,2;
DATFN B:OIL.DAT;
COVAR 1;
INDEP 2;
DEPEN 1;
ERRLN E(K/IJ);
SORCE A(I)+E(K/IJ);
SORCE B(J)+E(K/IJ);
SORCE AB(IJ)+E(K/IJ);
```

and is ready to be processed by DMAOVC.BAS.

In addition to those output files created as described in part 3.5, three additional files are created. The first,

called ***COV.OUT, contains the analysis of covariance results, from which needed values can be pulled off for reporting purposes. The second file created is called ***CSS.OUT, and contains the corrected sums-of-squares, sums-of-cross-products matrix. The third file created is called ***REG.OUT, and contains the regression coefficients. Both of these last two are unformatted work files used by DMAOVC.BAS and can be deleted if desired.

Below are the output files associated with covariance analysis as described above for figure 1.

1. OILCOV.OUT

```
            ANALYSIS OF COVARIANCE RESULTS
            ------------------------------

                        MODEL 1

 INVERSE MATRIX
 --------------
  2  9.447331128956141D-02

 REGRESSION COEFFICIENTS
 -----------------------
  2  .8927727916863543

 SSSP/MSMP DUE TO REGRESSION, DF=  1
 ---------------------------
  2  2  8.436702881436039  8.436702881436039

 RESIDUAL SSSP/MSMP, DF=  11
 ------------------
  2  2  .5632971185639615  5.120882896036013D-02

 A(I) + E(K/IJ)
 ----------------------------------------
 TREATMENT(ADJ) SSSP/MSMP, DF=  2
  2  2  .223686663064111  .1118433315320555

 TREATMENT DEVIATIONS, ADJ DEVIATIONS, STD ERROR
   1    1    8 -.25416667D+01 -.34321367D+00 0.18904359D+00
   2    1    8 0.45833333D+00 0.12354354D+00 0.84151226D-01
   3    1    8 0.20833333D+01 0.21967013D+00 0.16577958D+00
```

```
B(J) + E(K/IJ)
----------------------------------------
TREATMENT(ADJ) SSSP/MSMP, DF=  3
 2  2  .3409132616199164  .1136377538733055

TREATMENT DEVIATIONS, ADJ DEVIATIONS, STD ERROR
  1    1   6 -.75000000D+00 -.23027870D-02 0.10921591D+00
  2    1   6 0.25000000D+00 0.14956306D+00 0.92714801D-01
  3    1   6 0.91666667D+00 0.42493308D-01 0.11477454D+00
  4    1   6 -.41666667D+00 -.18975358D+00 0.94060272D-01

AB(IJ) + E(K/IJ)
----------------------------------------
TREATMENT(ADJ) SSSP/MSMP, DF=  6
 2  2  .5804298075183976  9.673830125306626D-02

TREATMENT DEVIATIONS, ADJ DEVIATIONS, STD ERROR
 1    1   2 -.29166667D+01 -.70953393D-01 0.27341892D+00
 2    1   2 -.14166667D+01 -.26722170D+00 0.18336831D+00
 3    1   2 -.24166667D+01 -.37444891D+00 0.22565308D+00
 4    1   2 -.34166667D+01 -.66023067D+00 0.26780990D+00
 5    1   2 0.83333333D-01 0.11681231D+00 0.16003505D+00
 6    1   2 -.14166667D+01 0.89887419D-01 0.19844646D+00
 7    1   2 0.20833333D+01 0.10807353D+00 0.22200570D+00
 8    1   2 0.10833333D+01 0.17940088D+00 0.17482558D+00
 9    1   2 0.58333333D+00 -.52767281D-01 0.16751235D+00
10    1   2 0.35833333D+01 0.62602346D+00 0.28051501D+00
11    1   2 0.30833333D+01 0.39385530D+00 0.26364525D+00
12    1   2 0.10833333D+01 -.88430956D-01 0.18422377D+00
```

2. OILCSS.OUT

```
E(K/IJ)                ==>      E(K/IJ) = error line
 -1  12  0                      -1 = matrix sign, error line
  9  9.449999999999989           12 = degrees of freedom
 10.58499999999992               0 = total location in
A(I) + E(K/IJ)                          ***TOT.OUT
 1  14  3                            .
 97.0833333333333  95.6875          .
 95.08249999999987                  .
B(J) + E(K/IJ)
 1  15  4
 18.8333333333333  19.40833333333299
 21.0095833333329
AB(IJ) + E(K/IJ)
 1  18  5
 29.9166666666667  29.82916666666699
 30.92416666666702
```

CHAPTER IV

EXAMPLE PARAMETER FILES

In the following sections are various example experimental designs, experimental design models and the appropriate parameter file structure needed to drive DMAOVC.BAS. It is of intentions to include a wide variety of factorial designs for user reference. It is suggested the user read through the different designs and examine how the parameter file is constructed for each situation. Notes and comments are scattered through out for a better understanding. The numeric values used are only for demonstration purposes.

One-Factor Completely Randomized Design

Experimental Design Model

$$Y_{ij} = U + A_i + E_{j(i)}$$

Job Parameter File Structure

```
TITLE ONE-FACTOR CRD
REVAR Y(ij),5,3,1;
MODEL A(i)+E(j/i);
FIXED i,3;
RANDM j,4;
DATFN B:NAME.DAT;
```

Job Parameter File for
Expected Mean Squares Only

```
TITLE ONE-FACTOR CRD
REVAR Y(ij);
MODEL A(i)+E(j/i);
FIXED i;
RANDM j;
EMSQR
```

note: The parameter file for expected mean squares only is structured the same except for (1) the format in the REVAR command is left off, (2) the number of levels for the FIXED and RANDM commands are also left off and (3) the DATFN command is replaced with the EMSQR command. This will be the case for all parameter files with expected mean squares only.

One-Factor Completely Randomized Design with Subsampling

Experimental Design Model

$$Y_{ijk} = U + A_i + E_{j(i)} + S_{k(ij)}$$

Job Parameter File Structure

```
TITLE ONE-FACTOR CRD WITH SAMPLING
REVAR Y(ijk),3,4,1;
MODEL A(i)+E(j/i)+S(k/ij);
FIXED i,3;
RANDM j,3;
RANDM k,3;
DATFN B:NAME.DAT;
```

One-Factor Randomized Complete Block Design

Experimental Design Model

$$Y_{ij} = U + B_j + A_i + E_{ij}$$

Job Parameter File Structure

```
TITLE ONE-FACTOR RBD
REVAR Y(ij),3,2,0;
```

```
MODEL BLKS(j)+TRT(i)+E(ij);
FIXED i,5;
RANDM j,8;
DATFN B:NAME.DAT;
```

Two-Factor Completely Randomzied Design with Subsampling

Experimental Design Model

$$Y_{ijkl} = U + A_i + B_j + AB_{ij} + E_{k(ij)} + S_{l(ijk)}$$

Job Parameter File Structure

```
TITLE TWO-FACTOR CRD W/SAMPLING
REVAR Y(ijkl),4,3,2;
MODEL A(i)+B(j)+AB(ij)+E(k/ij)+S(l/ijk);
FIXED i,3;
FIXED j,4;
RANDM k,2;
RANDM l,2;
DATFN B:NAME.DAT
```

Two-Factor Completely Randomized Block Design

Experimental Design Model

$$Y_{ijk} = U + R_k + A_i + B_j + AB_{ij} + E_{ijk}$$

Job Parameter File Structure

```
TITLE TWO-FACTOR CRBD
REVAR Y(ijk),4,2,0;
MODEL REPS(k)+A(i)+B(j)+AB(ij)+RESIDUAL;
FIXED i,3;
FIXED j,2;
RANDM k,2;
DATFN B:NAME.DAT
```

note: the model could be written as;

```
MODEL REPS(k)+A(i)+B(j)+AB(ij);
```

by leaving off the RESIDUAL

Three-Factor Completely Randomized Design

Experimental Design Model

$$Y_{ijkl} = A_i + B_j + C_k + AB_{ij} + AC_{ik} + BC_{jk} + ABC_{ijk} + E_{l(ijk)}$$

Job Parameter File Structure

```
TITLE THREE-FACTOR CRD
REVAR Y(ijkl),5,2,1;
MODEL A(i) + B(j) + C(k);
MODEL AB(ij) + AC(ik) + BC(jk);
MODEL ABC(ijk) + E(l/ijk);
FIXED i,3;
FIXED j,4;
RANDM k,2;
RANDM l,3;
DATFN B:NAME.DAT;
```

note: a. notice how the MODEL command is continued, each MODEL command still ends with a ";"
b. notice also the spacing between the model terms and with no spacing within
c. notice also how the indices remain as lower-case characters throughout all the commands and they always appear in the same order as with the REVAR command

Two-Factor Split Plot Design

Experimental Design Model

$$Y_{ijk} = U + T_i + R_{k(i)} + P_j + TP_{ij} + E_{ijk}$$

Job Parameter File Structure

```
TITLE TWO-FACTOR SPLIT PLOT DESIGN
REVAR Y(IJK),4,2,0;
MODEL T(I)+R(K/I)+P(J)+TP(IJ)+RESIDUAL;
FIXED I,2;
FIXED J,4;
RANDM K,6;
DATFN B:NAME.DAT;
```

note: notice how this example use upper-case letters to describe the indices, and how they remain upper-case throughout all the other commands

Two-Factor Split Plot Randomized Block Design

Experimental Design Model

$$Y_{ijk} = R_k + A_i + E_{ij} + B_j + AB_{ij} + E_{ijk}$$

Job Parameter File Structure

```
TITLE TWO-FACTOR SPLIT PLOT RBD
TITLE SEED EVALUATION STUDY
REVAR Y(IJK),4,3,2;
MODEL BLK(K)+LOTS(I)+BLK*LOTS(IK);
MODEL TRT(J)+LOTS*TRT(IJ)+RESIDUAL;
FIXED I,4;
FIXED J,4;
RANDM K,4;
DATFN B:NAME.DAT
```

One-Factor Completely Randomized Design with One Covariate

Experimental Design Model

$$Y_{ij} = U + A_i + E_{j(k)}$$

Job Parameter File Structure

```
TITLE TWO-FACTOR CRD WITH ONE COVARIATE
REVAR Y(ij),3,2,0;
REVAR X(ij),6,2,1;
MODEL A(i)+E(j/i);
FIXED i,3;
RANDM j,5;
DATFN B:NAME.DAT
COVAR 1;
INDEP 2;
DEPEN 1;
ERRLN E(j/i);
SORCE A(i)+E(j/i);
```

One-Factor Completely Randomized Design with Two Covariates

Experimental Design Model

$$Y_{ij} = U + A_i + Ej/i$$

Job parameter File Structure

```
TITLE ONE-FACTOR CRD WITH TWO COVARIATES
REVAR Y(IJ),4,2,1;
REVAR X(IJ),8,3,2;
REVAR Z(IJ),14,2,0;
MODEL A(I) + E(J/I);
FIXED I,3;
RANDM J,5;
DATFN B:NAME.DAT;
COVAR 1;
INDEP 2,3;
DEPEN 1;
ERRLN E(J/I);
SORCE A(I)+E(J/I);
```

note: The example above had one covariance model as specified in the COVAR command, two or more covariance models can be specified in the following manner;

```
COVAR 2;
INDEP 2;
DEPEN 1;
INDEP 3;
DEPEN 1;
```

where for each model there has to be a "pair" of INDEP and DEPEN commands specifying the indpendent variable(s) and dependent variable by the order given in the REVAR commands.

Two-Factor Randomized Block Design with Two Covariates

Experimental Design Model

$$Y_{ijk} = U + R_k + A_i + B_j + AB_{ij} + E_{ijk}$$

Job Parameter File Structure

```
TITLE TWO-FACTOR RBD WITH TWO COVARIATES
```

```
REVAR X1(IJK),4,2,0;
REVAR X2(IJK),7,2,0;
REVAR X3(IJK),10,2,0;
MODEL BLK(K) + SOIL(I) + FERT(J);
MODEL S*F(IJ) + RESIDUAL;
FIXED I,4;
FIXED J,2;
RANDM K,3;
DATFN B:NAME.DAT;
COVAR 2;
INDEP 1,3;
DEPEN 2;
INDEP 1;
DEPEN 3;
ERRLN RESIDUAL;
SORCE SOIL(I)+RESIDUAL;
SORCE FERT(J)+RESIDUAL;
SORCE S*F(IJ)+RESIDUAL;
```

note: This parameter file requests for 2 covariance models, the first with response variable 1 and 3 as the independent variables versus 2 as the dependent varialble.
The second with response variable 1 as the independent variable versus 3 as the dependent variable.

Two by Two Factorial Design with Repeated Measures and One Covariate

Experimental Design Model

$$Y_{ijk} = U + A_i + C_{k(i)} + B_j + AB_{ij} + E_{ijk}$$

Job Parameter File Structure

```
TITLE TWO BY TWO FACTORIAL WITH REPEATED
TITLE MEASURES AND ONE COVARIATE
REVAR X(IJK),5,2,0;
REVAR Y(IJK),8,2,0;
MODEL A(I)+C(J/I)+B(J)+A*B(IJ)+RESIDUAL;
FIXED I,2;
FIXED J,2;
RANDM K,4;
DATFN B:NAME.DAT;
COVAR 1;
INDEP 1;
DEPEN 2;
ERRLN C(K/I);
SORCE A(I)+C(K/I);
```

```
ERRLN RESIDAUL;
SORCE B(J)+RESIDUAL;
SORCE A*B(IJ)+RESIDAUL;
```

note: On this example, notice the two ERRLN commands. When there are two or more error lines in the covariance analysis, then the sources of variation pertaining to a particular error line must follow immediately after the ERRLN command, then comes the next ERRLN and the respective SORCE commands, etc., as the above example illustrates. ERRLN must always precede any SORCE command.

CHAPTER V

SUMMARY OF COMMANDS

In the description of the commands to follow, the following notational conventions are used:

[ ] - brackets enclose all required text
A - is to be replaced by alphanumeric information
i - is to be replaced by factor indices or subscripts
... - etc.
c - beginning column in a data file
w - width of a data field, including decimal points
d - number of numerical values to the right of a decimal point
nl - numeric value of the number of levels
dr - disk drive
fn - filename
nm - numeric value of the number of models
id - numeric value of the location of a variable in the order given

The TITLE Command

Form

TITLE [A]

Function

Specifies a title name or other note information to be printed at the top of the anlaysis of variance table, expected mean squares, and the covariance analysis.

Restrictions

Can fall anywhere in the job parameter file. It is best to be placed as the first commands in the job parameter file. No more than 20 lines of title allowed.

Example

TITLE Two-Factor RBD

The REVAR Command

Form

REVAR [A(i...),c,w,d;]

Function

Specifies to DMAOVC.BAS the response variable structure, and format in the data file.

Restrictions

"A" to ")" can be no longer than 20 total characters. Maximum of 8 indices between the parantheses. Must be specified in the parameter file before the FIXED and RANDM commands. Must end with a ";". The indices must be enclosed in parantheses.

Options

If the EMSQR option is in the parameter, then the c,w,d is left off.

Example

REVAR Y(ijk),5,2,1;

The MODEL Command

Form

MODEL [A(i...)+A(i...)+...;]

Function

Specifies the form of the experimental design model to be used in the analysis.

Restrictions

"A" to ")" can be no longer than 20 total characters. Each design model term must be separated with a "+". There can be spaces between the design model terms, but not within. The order of the indices must be the same as in the REVAR command. Multiple model commands can be used, each ending with a ";". The indices must be enclosed in parantheses. The indices must be either in upper-case alphabetic characters or lower alphabetic characters, but must remain the same as specified in the REVAR command.
"/" is used with the parantheses to separate the indices that are crossed from those that are nested.

Options

In some models the error term can be specified as RESIDUAL, Residual, or residual, or can be left off and will automatically be computed as a residual.

Example

If the response variable command is:

```
REVAR Y(IJK),c,w,d;
```

then the model command is:

```
MODEL A(I) + B(J) + AB(IJ) + E(K/IJ)
```

The FIXED Command

Form

```
FIXED [i,nl;]
```

Function

Specifies which factor (index) is fixed and the number of levels.

Restrictions

The index must be either an upper-case or lower-case alphabetic character, depending on how they were specified in the REVAR command. Must follow the REVAR command in the parameter file. Must end with a ";" after the "nl".

Options

If the EMSQR option is the parameter file, then the "nl" is left off.

Example

If the response variable command is:

```
REVAR Y(IJK),c,w,d;
```

then the fixed command is:

```
FIXED J,nl;
```

The RANDM Command

Form

```
RANDM [i,nl;]
```

Function

Specifies which factor (index) is random and the number of levels.

Restrictions

The index must be either an upper-case or lower-case alphabetic character, depending on how they were specified in the REVAR command. Must follow the REVAR command in the parameter file. Must end with a ";" after the "nl".

Options

If the EMSQR option is in the parameter file then the "nl" is left off.

Example

If the response variable command is:

```
REVAR Y(IJK),c,w,d;
```

Then the random command is:

```
RANDM I,nl;
```

The DATFN Command

Form

```
DATFN [dr:fn;]
```

Function

Specifies the raw data file name and disk drive location.

Restrictions

Must end with a ";".

Options

If the EMSQR option is in the parameter file then the DATFN is left completely from the parameter file.

Example

```
DATFN B:NAME.EXT;
```

The EMSQR Command

Form

```
EMSQR
```

Function

Specifies to only process the expected mean squares.

Restrictions

Must be at the end of the parameter file. No covariance commands will follow. See the above commands for their restrictions.

The COVAR Command

Form

COVAR [nm;]

Function

Specifies that covariance analysis is to be conducted for "nm" models.

Restrictions

Must precede any of the other covariance commands below (i.e. INDEP, DEPEN, ERRLN, SORCE). Must have ";" after "nm".

Example

COVAR nm;

The INDEP and DEPEN Commands

Form

INDEP [id,...;] and DEPEN [id,...;]

Function

Specifies the independent and dependent response variables for each model based on the order of input in the parameter file for covariance analysis.

Restrictions

Must follow the COVAR command. The last independent variable must end with a ";". The dependent variable must end witha ";". The DEPEN command must always follow the INDEP command. For each covariance model there must be a pair of INDEP and DEPEN commands.

Example

If the response variables are:

```
REVAR X1(IJK),c,w,d;
```

```
REVAR X2(IJK),c,w,d;
REVAR X3(IJK),c,w,d;
REVAR X4(IJK),c,w,d;
```

and the Covariance commands are the following for three covariance models:

```
COVAR 3;
INDEP 1,3;
DEPEN 4;
INDEP 1,2,4;
DEPEN 3;
INDEP 1;
DEPEN 2;
```

The ERRLN Command

Form

ERRLN [A;]

Function

Specifies the error line(s) in the covariance anlaysis.

Restricitons

Must follow the COVAR command. Must be one of the design model terms. Must end with ";". Must precede any SORCE commands.

Options

The ERRLN command and the SORCE command(s) must go together (i.e. for every SORCE sequence of SORCE commands, they have to be preceded by an ERRLN command).

Example

See the SORCE command below.

The SORCE Command

Form

SORCE [A+A+...;]

Function

Specifies the corrected sources of variation in the covariance analysis.

Restrictions

Must follow an ERRLN or other SORCE commands. The SORCE command is used in conjunction with the ERRLN command. For every ERRLN command

there must be at least one SORCE command following it. Must end with a ";".

Example

If there two error lines in a covariance analysis with one adjusted source of variation for one and two for the other then the sequence of commands would be as follows:

```
ERRLN AB(ij);
SORCE A(i)+AB(ij);
ERRLN RESIDUAL;
SORCE B(j)+RESIDUAL;
SORCE BC(jk)+RESIDUAL;
```

REFERENCES

Dowdy, S. and Wearden, S. (1983). Statistics for Research. John Wiley and Sons, New York.

Gill, J. L. (1978). Design and Analysis of Experiments in the Animal and Medical Sciences. Iowa State University Press, Amex, Iowa.

Hurst, R. L. (n.d.). FCT.BAS (Factorial Analysis of Variance/Covariance) Computer program sourcecode. Department of Applied Statistics, Utah State University, Logan, Utah.

Hurst, R. L. (n.d.). FCT.DOC (Factorial Analysis of Variance Covariance) Unpublished documentation for FCT.BAS. Department of Applied Statistics, Utah State University, Logan, Utah.

Kirk, R. E. (1982). Experimental Design: Procedures for the Behavioral Sciences. 2nd edition. Brook/Cole Publishing Company, Belmont, California.

Steel, R. G. and Torrie, J. H. (1980). Principles and Procedures of Statistics: A Biometrical Approach. 2nd edition. McGraw-Hill Book Company, New York.

Winer, B. J. (1971). Statistical Principles in Experimental Design. 2nd edition. McGraw-Hill Book Company, New York.

APPENDICES

Appendix A: An Algorithm for Binary Codes

In this appendix, an algorithm is presented that derives the linear combinations of binary code used as a key for computing the appropriate degrees-of-freedom and sums-of-squares for a balanced factorial analysis of variance. These linear combinations of binary code are derived by recognizing the structure of the factor indices from input experimental design model components. The algorithm is presented in two parts. The first part is a method to derive the linear combinations of binary codes by hand. The second method is a method that can be directly programmed on the computer and in fact, is the method used by DMAOVC.BAS.

The easiest way to demonstrate the steps is to work through an example. Suppose we have a response variable with the following structure;

$$Y_{ijklmn}$$

which has six factors as indicated by the subscripts or indices. Suppose we want to determine the linear combinations of binary code for the following experimental design model component;

$$CDE_{klm(ij)}$$

where klm are crossed and nested within ij.

Step 1

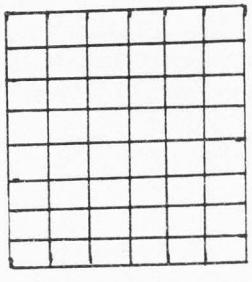
Construct a two-way table.

a. The number of columns equal to the number of factors defined in the response variable.

b. The column headings consisting of the indices used in the response variable.
c. The number of rows equal to the value two raised to the power of the number of crossed indices, excluding those that the crossed ones are nested within.

For our example, this would appear as;

i j k l m n



where there are six factor indices for the columns and two raised to the power of three which equals eight for the number of rows since there are three crossed factor indices, excluding nested ones in the model component.

Step 2

Take all possible combinations of the number of crossed factor indices, excluding those they are nested within. Begin by taking all of then at a time and work down to and include zero at a time. Code these into "1's" and "0's" as shown in our example below.

i j k l m n

		1	1	1	
		1	1	0	
		1	0	1	
		0	1	1	
		1	0	0	
		0	1	0	
		0	0	1	
		0	0	0	

Step 3

Fill in the rest of the columns in the table with either "1's" or "0's". Use "0's" if the factor indices are not included in the model component. Use "1's" if the factor index is in the model component. For our example the table now appears as follows;

i	j	k	l	m	n
1	1	1	1	1	0
1	1	1	1	0	0
1	1	1	0	1	0
1	1	0	1	1	0
1	1	1	0	0	0
1	1	0	1	0	0
1	1	0	0	1	0
1	1	0	0	0	0

Step 4

Sum accross the rows and record these sums in anew column as shown below for our example.

i	j	k	l	m	n	
1	1	1	1	1	0	5
1	1	1	1	0	0	4
1	1	1	0	1	0	4
1	1	0	1	1	0	4
1	1	1	0	0	0	3
1	1	0	1	0	0	3
1	1	0	0	1	0	3
1	1	0	0	0	0	2

Step 5

Add one more column to the table and fill this in with one's as show below for our example.

i	j	k	l	m	n		
1	1	1	1	1	0	5	1
1	1	1	1	0	0	4	1
1	1	1	0	1	0	4	1
1	1	0	1	1	0	4	1
1	1	1	0	0	0	3	1
1	1	0	1	0	0	3	1
1	1	0	0	1	0	3	1
1	1	0	0	0	0	2	1

Step 6

Beginning with the top row sum and the top value of positive one in the new column of one's, work through these and when the sum changes, "toggle" the ones from positive to negative to positive, etc. until you get to the end of the rows. This is illustrated below for our example.

i	j	k	l	m	n		
1	1	1	1	1	0	5	+1
1	1	1	1	0	0	4	-1
1	1	1	0	1	0	4	-1
1	1	0	1	1	0	4	-1
1	1	1	0	0	0	3	+1
1	1	0	1	0	0	3	+1
1	1	0	0	1	0	3	+1
1	1	0	0	0	0	2	-1

Step 7

The final step is to use the rows as the binary code in determining the uncorrected sums-of-squares needed. Then, by using the "+1's" and "-1's" column as a determination of whether to add or subtract a particular sums-of-squares for the particular design model component. This same method will also be used for determining the degrees-of-freedom associated with each uncorrected sums-of-squares.

Although the above algorithm might seem more trouble than it is worth for determining the sums-of-squares needed for a particular design model component, it can be and was programmed quite nicely. Below describes the steps needed to program this algorithm.

Step 1

Determine and set up an array with the columns equal to the total number of factors in the response variable.

Step 2

Determine the number of crossed factors in a particular design model component. Exclude those that the crossed factors are nested within. Raise two to the power of this number and use as the number of rows.

Step 3

Use binary arithmetic to derive the various possible combinations as described in step 2 of method 1. These will have to be sorted later.

Step 4

Conduct step 3 and step 4 in method 1 above.

Step 5

Sort the sums of binary codes from highest to lowest, simultaneously sorting the binary codes themselves.

Step 6

Complete steps 5, 6, and 7 for method 1 above.

Step 7

Repeat as needed for each term in the design model.

Appendix B. DMAOVC.BAS Pseudo-Code

Main Program

The main program is mostly a general processing block that calls individual subroutines. Some processing takes place in between the subroutine calls, or gosubs, but this is kept to a minimum.

1. Define all variables as integer, character, or double precision.
2. Dimensionalize all arrays.
3. Print to terminal program title page.
4. Call subroutine Input Parameter File.
5. Open expected mean squares matrix work file.
6. Open averages output file.
7. Call subroutine Data Input.
8. Open Analysis of Varaince Table output file.
9. Write title and heading to AOV table file if opted for.
10. Write total degrees of freedom and sums-of-squares to AOV table file.
11. Begin loop for processing each individual model term which are really sources of variation.
12. Call subroutine Count Subscripts.
13. If expected mean squares only wanted then go to Subroutine Combine Subscript Binary Codes.
14. Call subroutine Binary Arithmetic.
15. Call subroutine Quick Sort.
16. Resort binary codes and linear combinations in descending order.
17. Call subroutine Combine Subscript Binary Codes.
18. Write expected matrix for current design model term being processed.
19. Call subroutine Linear Combinations.
20. Call subroutine Uncorrected Sums of Squares.
21. Call subroutine ANOVA table.
22. Goto 11. and begin processing next design model term.
23. If expected means square only wanted, close expected mean squares matrix work file and Call Expected Mean Squares Subroutine.
24. Check to see if the accumulated degrees of freedom is less than the total degrees of freedom, if they are then determine the residual degrees of freedom and sums of squares and write this to the ANOVA table.
25. Close all work files that are created in the above various subroutines.
26. Call subroutine Expected Mean Squares.
27. If expected mean squares only wanted then print the name of the expected mean squares file to the terminal

and end.
28. If covariance analysis is requested then call subroutine Corrected SS/SP Matrices Block.
29. If covariance analysis is requested then call subroutine Covariance Block.
30. Print to terminal the output files that were created during processing.
31. End program.

Input Parameter File

This subroutine prompts for the job parameter file and reads in one command line at a time. For each job parameter file command there is a computed on-go-to statement that processes each command separately. This subroutine prepares the appropriate arrays, commands, etc. for processing throughout the remainder of the program.

1. Set up the command array with the commands for comparison by using the data statement.
2. Prompt for the input job parameter file name, disk drive, and open the job parameter file.
3. Read one command line at a time and print to terminal.
4. Begin loop for comparing input command with the array containing the commands to determine which part to goto in the subroutine for processing that specific command.
5. TITLE command: Store the titles in an array, if any.
6. REVAR command: Parser the line into the response variable, store the indices into an array, count the number of indices which will equal the number of factors, store the beginning data column, width and decimal values into their respective arrays.
7. MODEL command: Parser the model into the respective model terms, count the number design model components, check if there is a residual term and store these terms in an array.
8. FIXED and RANDM command: Parser these commands, identify the number of levels for each index in the response variable and store in an array which index is fixed and which is random.
9. DATFN command: Store the data filename.
10. EMSQR command: Toggle on or off by using "0" = off and "1" = on.
11. COVAR command: Toggle on or off by using "0" = off and "1" = on, and store the number covariance models wanted. Also open the corrected sums of squares/cross

products work files.
12. INDEP command: Determine which response variables are the independent variables for each covariance model.
13. DEPEN command: Determine which response variable is the dependent variable for each covariance model and write the indpendent and dependent information to a work file.
14. ERRLN command: Determine and write to the ss/sp work file the error line(s).
15. SORCE command: Determine and write to the ss/sp work file the sources of variation for covariance analysis.
16. Prompt for a continuation after printing the parameter file as to continue with the execution or not.
17. Return to main program.

Data Input

This subroutine opens and reads in the raw data file based on the input format from the job parameter file. It first reformats the data coming and writes it out to an unformatted work file. Prints the first five observations to the terminal with a continuation of execution. As it reads the raw data in, it computes the overall average, the uncorrected total sums-of-squares and the correction term, and writes these to the uncorrected sums-of-squares work file. Parts of this subroutine was extracted from FCT.BAS (Hurst, n.d.).

1. Open data file.
2. Initialize ss/sp array to 0.0.
3. Read in one line of data at a time and based on the format parser into the different response variables.
4. Print the first five observations to the terminal for a continuation or not.
5. While reading the data print the data line by line to a data file work file.
6. While reading the data compute the total, number of observaitons, ss/sp and write these to appropriate files.
7. Close input/output files either for rewinding or for good.
8. Return to main program.

Count Subscripts

This subroutine parse's each design model component or term in their respective subscripts or indices. It counts the number of factors or indices that are crossed or nested and stores these in separate arrays.

1. Initialize number of subscripts and number of nested subscripts to 0.
2. Parser the currently being processed design model term into its respective subscripts counting the number of subscripts and the number of subscripts that are nested.
3. Store the subscripts in arrays as the design model term is parsered.
4. Return to main program.

Binary Arithmetic

This subroutine does binary arithmetic from one to the number of subscripts in the design model component currently being processed to get all possible combinations of binary code for the number of indices.

1. Initialize binary arithmetic array to 0.
2. Loop from one to two raised to the number of subscripts in the design model currently being processed.
3. Do binary aritmetic and store the 0's and 1's in an array.
4. Sum accross the binary numbers and store this in a linear combination array.
5. Return to main program.

Quick Sort

This subroutine sorts the binary codes sum and the binary codes in an ascending order. This subroutine was extracted from a sorting program coded in FORTRAN developed by Dr. Rex L. Hurst and was coded into MBASIC for DMAOVC.BAS.

1. Begin sort process by checking the first term and

replacing it into the next term that is larger.
2. Do this until both the binary code and the sums or linear combinations are sorted into ascending order.
3. Return to main program.

Combine Binary Codes

This subroutine develops the binary code used to compute the uncorrected sums-of-squares. It uses the binary codes from Binary Arithmetic and Quick Sort to "fill in" the remaining subscripts either with a "0" if it is not in the design model term currently being processed or a "1" if it is nested within the design model term. This subroutine writes the appropriate values to the expected mean squares matrix work file, depending on whether the subscripts in the currently being processed design model term are crossed, nested, fixed or random.

1. Initialize variables used to 0 or blank.
2. Process from one to the number of factors in the model.
3. If the factor subscript in the model term currently being processed is equal to the response variables indices, then it gets a "1" if not it gets a "0".
4. It now loops back through the factors and determines which are nested in the design model component currently beingprocessed and it gets a "1" if it is nested.
5. It stores the binary codes in an array ready to be processed for use in computing the uncorrected sums-of squares needed for the design model currently being processed.
6. This part also sets up the matrix on a work file where the rows are equal to the design model component and the columns are the facor indices (including reps). It checks to see if the subscript is in the model component for each subscript; if it is then it gets either a "1" or a "0" dependent if is a fixed or random factor, it it is not it get that subscript which is really a coefficient. If the subscript is nested it then gets a "1".
7. Return to main program.

Linear Combinations

This subroutine uses the sorted binary codes sums to determine the +1.0 or -1.0 linear combinations of the binary codes and thus determining whether to add or subtract uncorrected sums-of-squares.

1. Initialize variables.
2. Loop from one to the number of binary codes for the design model currently being processed.
3. Begin with +1.0 and scan through the binary sums, when the sum changes, toggle the +1.0 *-1.0.
4. Store these +1.0's and -1.0's in a linear combination array.
5. Loop back to 2., until all bianry codes are complete.
6. Return to main program.

Uncorrected Sums of Squares

This subroutine processes the binary code for the currently design model being processed and computes the uncorrected sums-of-squares for each binary string or code. It first checks to see if the particular binary code is computed already, if it has it skips to the next binary code. It it has not been computed, then it computes it and writes these to an uncorrected sums-of-squares work file. Most parts of this subroutine was extracted from FCT.BAS (Hurst, n.d.) with some modifications.

1. Open two files (1) the uncorrected sums-of-squares file and (2) the copy to file for appending new uncorrected sums-of-squares.
2. Begin loop from one to the number of linear combinations of binary code for the design model currently being processed.
3. Read in from the uncorrected ss/sp file and decide if the binary code has been computed, it it has go to 2., it not then compute the necessary ss/sp.
4. Read in the raw data file and compute the sums and sum-of-squares appropriately depending on the binary code.
5. Also compute and write to the averages, totals and

uncorrected ss/sp to their respective files the degrees of freedom, which binary code is being computed, and the appropriate value.
6. Rewind the work files.
7. Loop back to 2., until end of binary codes.
8. Return to main program.

Analysis of Variance Table

This subroutine computes the degrees of freedom, the sums of squares, and the mean squares for the design model currently being processed. It then writes these out to the analysis of variance file.

1. Initialize arrays used.
2. Open uncorrected ss/sp matrices file.
3. Loop from one to the number of binary codes for the design model currently being processed.
4. Read in the uncorrected ss/sp matrices one at a time and check if it is one that is needed in the linear combinat- ions, if it is then multiply it by the linear combiantion of either +1.0 or -1.0 and add to an accumulating sums-of- squares vector for each resonse variable. Also keep track of the degrees of freedom in the same manner.
5. Loop back to 3. until end of binary codes for the design model currently being processed.
6. Write the design model component description, degrees of freedom, sums-of-squares, and mean squares to the analysis of variance file. Also keep a vector of accummulated sums-of-squares and degrees of freedom to determine if there is any residual.
7. Return to main program.

Expected Mean Squares

This subroutine determines the expected mean squares for each design model component by using the developed expected mean squares matrix work file and writes these to a separate output file. It uses the rules for deriving expectedmeansquaresas described by Kirk (1982).

1. Initialize variables.
2. Write title to expected mean squares output file.

3. Open the ems matrix file.
4. By working from the top of the matrix down, covers up subscripts. The ems is the weighted sum of coef. due to all effects containing the particular weight is the product of all entries in the row except the conmponent subscript(s). If the component has nested suscripts then the other effects must contain all subscripts, nested one included.
5. When complete return to main program.

Corrected SS/SP Matrices Block

This subroutines the corrected ss/sp matrices for the analysis of covariance and stores it on a disk file for later use in the covariance processing block.

1. Open work files.
2. Input the sources from the source file.
3. Input the number of degrees of freedom, uncorrected ss/sp identification lines and read in the ss/sp matrix.
4. Check if the design model component is the error line(s).
5. If it is then write it out to the corrected ss/sp work file.
6. Check if the design model component is the one needed, if it is then add it to the error line, along with the degrees of freedom and write it to the corrected ss/sp work file.
7. Loop back to 2. for the next source of variation in the covariance analysis.
8. Close work for rewinding.
9. Return to main program.

Covariance block

This subroutine uses the covariance work files, the uncorrected ss/sp and corrected ss/sp matrices along with the totals file to compute the anlysis of covariance results. This subroutine was extracted from FCT.BAS (Hurst) with some modifications to make it compatible with the rest of DMAOVC.BAS.

1. Open appropriate files.
2. Begin loop for number of covariance models.
3. Input from work file number of independent and dependent

variables needed for current model.
4. Input the uncorrected ss/sp and totals location and values.
5. Compute the approprate "X" matrix.
6. Call DMATIV subroutine and invert "X" matrix.
7. If error line then Call subroutine Error Calculations and compute regression coefficients, sssp due to regression and sss/msmp residual. Write to covariance output file.
8. If not error line then Call subroutine Treatment Error Calculations and compute adjusted treatment sssp/msmp and treatment deviations. Write to covariance output file.
9. Loop back for next source of variation in covariance analysis.

Data matrix inverse

This subroutine inverts the "X" matrix. This subroutine was exrtracted from FCT.BAS (Hurst, n.d.).

Error Calculations

This subroutine computes the regression coeffcients, sssp/msmp due to regression and the residual sssp/msmp. This subroutine was extracted from FCT.BAS (Hurst, n.d.).

Treatment Plus Error Calcualtions

This subroutine computes the adjusted treatment sssp/msmp. This subroutine was extracted from FCT.BAS (Hurst, n.d.).

Appendix C. Definitions of Variables Used

Integer (I - R)

NFL(8) = Number of factor levels
NBINCOD(256,8) = Unsorted binary codes
NBINSUM(256) = Unsorted binary codes sums
NSBNCD(256,8) = Sorted binary code
NSBNSUM(256) = Sorted binary sum
LC(256) = Linear combination (+1 or -1)
NDF(256) = Number of degrees of freedom
NBEG(20) = Response variable beginning column
NWID(20) = Response variable width column
NDEC(20) = Response variable decimal values
RES = Residual toggle
I,J,K,L,M,N,P = Counters
NV = Number of response variables
NT = Number of titles
NDMC = Number of design model components
KNT = Pointer in parameter input file
NLL = Temporary counter of NFL
ID = Decimal point location
PT1...PTk = Pointers
NADF = Number of accumulated degrees of freeedom
Q(20) = Temporary storage in binary arithmetic
R,RR = Counter
NF = Number of factors
NS = Number of subscripts
NNSS = Number of nested subscripts
RUV(16) = Array used in inverting matrix
LV(16) = Array used in inverting matrix
JNL(8) = Array used in inverting matrix
JJNX(8) = Array used in inverting matrix
JNX(8) = Array used in inverting matrix
IX(8) = Array used in inverting matrix
NBC = Counter used to identify uncorrected ss/sp location in work file
NDFT = Number of degrees of freedom total
NOBS = Number of observations
NY = Number of dependent variables
NX = Number of independent variables
JNF = Counter
JLEN = Pointer
NLOC = Location pointer
JLEN = Location pointer
II,JJ,KK,MM = Counters
NDIV = Divisor
NDGF = Number of degrees of freedom
NFLAG = Logical flag
NC1,NC4 = Pointers

Character (C - H, V):

VAR(20) = Response variable (i.e. Y(ijk))

DMC(256) = Design model component
VARS(8) = Response variable subscripts
FS(8) = Factor subscripts
CNSS(8) = Nested subscripts
CROF(8) = Random = "1", fixed = "0"
EMSMAT(8) = Expected mean squares matrix
CTITLE(20) = Title of analysis
COMND(20) = Commands from parameter file
CPFN = Input parameter file name
CLINE = Parameter file input line
FOR1 = Format line
FOR2 = Format line
G = "11111111111111111111"
H = "00000000000000000000"
CSTRG = Data input string
CH = Temporary data input string
CBEG = Character data beginning column
CWID = Character data width
CDEC = Character decimal points
CL = Character level
CANS = Answer Y or N
CE1 = EMS coeficient matrix
CE2 = Factor subscripts
CE3 = Nested subscripts
CFAC = Character string of factors
EMSQR = Logical "1" or "0"
COV = Logical "1" or "0"
FOR4 = Format statement
FOR7 = Format statement
DFLN = Data file name
CBTA = Temporary storage of factor subscripts
CBT1, CBT2 = Temporary storage of factor subscripts and nested subscripts
CSUB = Subscripts
CTSUB = Temporary subscript
EMS = Expected mean squares line in reverse order
FEMS = Expected mean squares line in correct order
COMP1, COMP2 = Input character line for subscripts and nested subscripts
COEF = Expected mean squares coefficient

Double Precision (A, S, X, T)

A(21,20) = Sums of squares/cross products matrix
S(20) = Sums of response variables
X(20) = Individual raw data value
AT(21,20) = Accumulating ss/sp matrix
TOTSS(20) = Total accumulated sums of squares
ASS(20) = Accumulated sums of squares
AT(21) =Temporary storage of uncorrected sums of squares for anova output
TOTSS = Total sums of squares for anova table

TOT(2000) = Totals used
TTOT(20) = Temporary total storage

Appendix D. Input/Output Files

File Unit Number	Name	Description
#1	******.PAR	Input job parameter file
#1	***CSS.TMP	Temporary work file of corrected sums of squares/sums of cross products
#2	***MOD.TMP	Temporary work file used for storing covariance models, independent variables and dependent variables
#2	***AVG.OUT	Contains the overall and cell averages
#3	***EMS.MAT	Temporary work file used for storing expected mean squares matrix
#3	***SRC.OUT	Temporary file containing the error line and sources of variation used to generate the corrected sums of squares
#3	***COV.OUT	Covariance analysis results
#4	***COP.DAT	Work file, copy of the raw data file
#4	***REG.OUT	Work file containing the regression coefficients
#4	***EMS.OUT	Expected mean squares
#5	******.DAT	Raw data file
#5	***AOV.OUT	Analysis of varaince table
#5	***CSS.OUT	Work file of corrected sums of squares/sums of cross products
#6	***TOT.DAT	Contains the overall and cell raw totals and divisor
#7	***USS.OUT	Uncorrected sums of squares sums of cross products
#8	***USS.COP	Temporary work file of uncorrected sums of squares/sums of cross products

note: *** = first three characters of parameter file name

****** = parameter file name and data file name

Appendix E. Making Additions, Deletions or Changes

DMAOVC.BAS has a modular design to it. This will allow the user to make additions, deletions or changes if desired. Below briefly describes how to make some of these changes that might be desired.

Making Additions

To make additions (e.g. other subroutines, etc.) simply add the necessary command statement to the data statement in the subroutine input parameter file, add the gosub statement in the main program and append the subroutine to the end of the program.Some other changes or adjustments might have to be made, for this the user will have to turn to the source code in part 5.6.

Making Changes

If any changes are to be made in the parameter specifications of DMAOVC.BAS (e.g. more than eight factors, more than twenty response variables, etc.) simply change the arrays in the dimension statements (e.g. 8 to the number of factors needed, 20 to the number of response variables needed, etc.). The user needs to keep in mind the capabilities of the microprocessor being used.

Making Deletions

Deletions can be made, but the user should have a good feel for the program flow and the source code to be sure no

other subroutines or processes are effected.

note: after any additions, changes or deletions DMAOVC.BAS will need to recompiled

```
REM*************************************************************************
REM*                                                                       *
REM*          DESIGN MODEL ANALYSIS OF VARIANCE/COVARIANCE                 *
REM*                                                                       *
REM*                          (DMAOVC.BAS)                                 *
REM*                       (COMPILER MBASIC)                               *
REM*                                                                       *
REM*                           APS 697                                     *
REM*                                                                       *
REM*                             by                                        *
REM*                                                                       *
REM*                         WESLEY E. NEWTON                              *
REM*                                                                       *
REM*              A master's project submitted in partial                  *
REM*                fulfillment of the requirements for                    *
REM*                          the degree of                                *
REM*                                                                       *
REM*                        MASTER OF SCIENCE                              *
REM*                                                                       *
REM*                             IN                                        *
REM*                                                                       *
REM*                       APPLIED STATISTICS                              *
REM*                                                                       *
REM*                       UTAH STATE UNIVERSITY                           *
REM*                            Logan, Utah                                *
REM*                              1985                                     *
REM*                                                                       *
REM*************************************************************************
REM MAIN PROGRAM-DMAOVC.BAS-DESIGN MODEL ANLAYSIS OF VARIANCE/COVARIANCE
REM*************************************************************************
      DEFINT I-R
      DEFSTR C-H,V
      DEFDBL A,S,X,T
REM
```

```
      DIM NFL(8),NBINCOD(256,8),NSBNSUM(256),LC(256),NDF(256)
      DIM VAR(20),DMC(256),VARS(8),FS(8),CNSS(8),CROF(8),NBINSUM(256)
      DIM NSBNCD(256,8),Q(20),RUV(16),LV(16),JNL(8),JJNX(8),JNX(8),IX(8)
      DIM EMSMAT(8),CTITLE(20),COMND(20),NBEG(20),NWID(20),NDEC(20)
      DIM A(21,20),S(20),X(20),AT(21,20),TOTSS(21,20),ASS(21,20)
      DIM TOT(2000),TTOT(20),CBTA(8),SS(21,20),IDX(20),IDY(20),ID(20)
REM
      PRINT "" :PRINT "" :PRINT "" :PRINT "" :PRINT "" :PRINT ""
      PRINT TAB(14) "DESIGN MODEL ANALYSIS OF VARIANCE/COVARIANCE"
      PRINT ""
      PRINT TAB(31) "(DMAOV.BAS)" :PRINT TAB(28) "(COMPILER MBASIC)"
      PRINT "" :PRINT TAB(35) "BY" :PRINT ""
      PRINT TAB(28) "WESLEY E. NEWTON" :PRINT "" :PRINT ""
      PRINT TAB(20) "DEPARTMENT OF APPLIED STATISTICS"
      PRINT TAB(25) "UTAH STATE UNIVERSITY"
      PRINT TAB(27) "LOGAN, UTAH  84322"
      PRINT TAB(26) "PHONE:  801-750--2419"
      PRINT "" :PRINT "" :PRINT "" :PRINT "" :PRINT "" :PRINT ""
REM
      GOSUB 1200 :REM SUBROUTINE INPUT PARAMETER FILE
REM
      FT="" :FT=FLN :FT=FT+"EMS.MAT" :OPEN "O",#3,FT
      IF EMSQR="1" THEN GOTO 50
      FT="" :FT=FLN :FT=FT+"AVG.OUT" :OPEN "O",#2,FT
      IF COV="1" THEN FT="" :FT=FLN :FT=FT+"CSS.TMP" :OPEN "O",#1,FT
REM
      GOSUB 4300 :REM SUBROUTINE DATA INPUT
REM
      FT="" :FT=FLN :FT=FT+"AOV.OUT" :OPEN "O",#5,FT
REM
      IF NT > 0 THEN_
        FOR I = 1 TO NT :PRINT #5,CTITLE(I) :NEXT I
      PRINT #5," " :PRINT #5,TAB(20) "ANALYSIS OF VARIANCE TABLE"
      PRINT #5,TAB(20) "--------------------------" :PRINT #5,""
```

```
      PRINT #5,"VAR" TAB(8) "SOURCE" TAB(30) "DF" TAB(39) "SS" TAB(55) "MS"
      PRINT #5,"---" TAB(8) "------" TAB(30) "--" TAB(39) "--" TAB(55) "--"
      FT="" :FT=FLN :FT=FT+"USS.OUT" :OPEN "I",#7,FT
      INPUT #7,CBT1 :INPUT #7,NDFT
      FOR I = 1 TO NV
        IF COV="0" THEN INPUT #7,A(I,I)
        IF COV="1" THEN
          FOR J = I TO NV :INPUT #7, A(I,J) :NEXT J
      NEXT I
      INPUT #7,CBT1 :INPUT #7,NDFT
      FOR I = 1 TO NV
        IF COV="0" THEN INPUT #7,AT(I,I)
        IF COV="1" THEN
          FOR J = I TO NV :INPUT #7, AT(I,J) :NEXT J
      NEXT I
      TOTDF=NDF(1)-NDF(2)
      FOR4=" ## \                          \ ####  #.########^^^^  #.########^^^^"
      FOR I = 1 TO NV
        TOTSS(I,I)=A(I,I)-AT(I,I)
        PRINT #5,USING FOR4; I; "TOTAL"; TOTDF; TOTSS(I,I);_
          TOTSS(I,I)/TOTDF
        IF COV = "1" THEN
          FOR J = I TO NV :TOTSS(I,J)=A(I,J)-AT(I,J) :NEXT J
      NEXT I
      CLOSE #7
REM
      NADF=0
      FOR I = 1 TO NV
        FOR J = I TO NV :ASS(I,J)=0.0 :NEXT J
      NEXT I
      NBC=3
REM
50    FOR R = 1 TO NDMC
        IF EMSQR="1" THEN GOTO 52
```

```
        PRINT " "
        PRINT "PROCESSING" TAB(12) DMC(R) TAB(35) "-- PLEASE WAIT --"
52      FOR L = 1 TO NF :FS(L)=" " :CNSS(L)=" " :NEXT L
REM
        GOSUB 2460  :REM SUBROUTINE COUNT SUBSCRIPTS
REM
        IF EMSQR = "1" THEN GOTO 55
REM
        GOSUB 2810  :REM SUBROUTINE BINARY ARITHMETIC
REM
        GOSUB 3140  :REM SUBROUTINE QUICK SORT
REM
        FOR L = 1 TO 2^NS
          FOR J = 1 TO NF
            NSBNCD(L,J)=0
          NEXT J
        NEXT L
        K=2^NS :N=2^NS
        FOR L = 1 TO N
           FOR J = 1 TO NS
             NSBNCD(L,J)=NBINCOD(K,J)
           NEXT J
           NSBNSUM(L)=NBINSUM(K)
           K=K-1
        NEXT L
REM
55      GOSUB 3650  :REM SUBROUTINE COMBINE SUBSCRIPT BINARY CODES
REM
        CE1="" :CE2="" :CE3=""
        FOR I = 1 TO NF :CE1=CE1+EMSMAT(I) :NEXT I
        FOR I = 1 TO NS :CE2=CE2+FS(I) :NEXT I
        IF NNSS > 0 THEN
          FOR I = 1 TO NNSS :CE3=CE3+CNSS(I) :NEXT I
        IF NNSS = 0 THEN CE3="          "
```

```
        PRINT #3, CE1 :PRINT #3,NS :PRINT #3, CE2
        PRINT #3, NNSS :PRINT #3, CE3
        IF EMSQR="1" THEN GOTO 60
REM
        GOSUB 4140  :REM SUBROUTINE LINEAR COMBINATIONS
REM
        GOSUB 4790  :REM SUBROUTINE UNCORRECTED SUMS OF SQUARES
REM
        GOSUB 5880  :REM SUBROUTINE ANOVA TABLE
REM
60    NEXT R
REM
      IF EMSQR="1" THEN GOTO 250
      IF NADF < TOTDF THEN GOTO 100 ELSE GOTO 200
100     RES=1
        IF COV = "1" THEN PRINT #1,"RESIDUAL" :_
          PRINT #1,TOTDF-NADF :PRINT #1,0
        K=1
        FOR I = 1 TO NV
          PRINT #5,USING FOR4; I; "RESIDUAL"; TOTDF-NADF;_
            TOTSS(I,I)-ASS(I,I); (TOTSS(I,I)-ASS(I,I))/(TOTDF-NADF)
          IF COV = "1" THEN GOTO 110 ELSE GOTO 120
110         FOR J = I TO NV
              PRINT #1,TOTSS(I,J)-ASS(I,J);
              IF K MOD 5 = 0 THEN PRINT #1,""
              K=K+1
            NEXT J
120     NEXT I
        IF COV = "1" THEN GOTO 130 ELSE GOTO 200
130     IF (K-1) MOD 5 <> 0 THEN PRINT #1,""
REM
200   FOR7="  VAR ## = \                    \"
      PRINT #5,"" :PRINT #5,"NOTE:"
      FOR I = 1 TO NV :PRINT #5,USING FOR7; I; VAR(I)  :NEXT I
```

```
      CLOSE #5 :CLOSE #1 :CLOSE #6 :CLOSE #2
250   CLOSE #3
REM
      GOSUB 6120  :REM SUBROUTINE EXPECTED MEAN SQUARES
REM
      IF EMSQR="1" THEN PRINT "" :FT="" :FT=FLN :FT=FT+"EMS.OUT" :
      PRINT "EXPECTED MEAN SQUARES ARE ON FILE" TAB(35) FT :GOTO 300
REM
      IF COV = "1" THEN
      GOSUB 7000  :REM SUBROUTINE CORRECTED SS/SP MATRICES BLOCK
REM
      IF COV = "1" THEN
      GOSUB 9000  :REM SUBROUTINE COVARIANCE BLOCK
REM
      PRINT ""
      PRINT "" :PRINT "FILES CREATED                    DESCRIPTION"
      PRINT "-------------      ------------------------------------"
      FT="" :FT=FLN :FT=FT+"AOV.OUT"
      PRINT FT TAB(22) "CONTAINS THE ANOVA TABLE"
      FT="" :FT=FLN :FT=FT+"AVG.OUT"
      PRINT FT TAB(22) "CONTAINS OVERALL AND CELL MEANS"
      FT="" :FT=FLN :FT=FT+"TOT.OUT"
      PRINT FT TAB(22) "CONTAINS TOTALS USED"
      FT="" :FT=FLN :FT=FT+"USS.OUT"
      PRINT FT TAB(22) "CONTAINS UNCORRECTED SUMS-OF-SQUARES"
      FT="" :FT=FLN :FT=FT+"EMS.OUT"
      PRINT FT TAB(22) "CONTAINS EXPECTED MEAN SQUARES"
      FT="" :FT=FLN :FT=FT+"COP.DAT"
      PRINT FT TAB(22) "CONTAINS COPY OF THE DATA FILE"
      IF COV = "1" THEN GOTO 260 ELSE GOTO 290
260   FT="" :FT=FLN :FT=FT+"COV.OUT"
      PRINT FT TAB(22) "CONTAINS COVARIANCE ANALYSIS RESULTS"
      FT="" :FT=FLN :FT=FT+"CSS.OUT"
      PRINT FT TAB(22) "CONTIANS CORRECTED SS/SP MATRICES"
```

```
      FT="" :FT=FLN :FT=FT+"REG.OUT"
      PRINT FT TAB(22) "CONTAINS REGRESSION COEFFICIENTS"
290   PRINT "-------------      ------------------------------------"
      PRINT ""
      PRINT "NOTE:  SEE DMAOV.BAS USER'S GUIDE FOR EACH FILE'S FORMAT"
REM
300   END
REM*************************************************************************
REM  SUBROUTINE:  INPUT PARAMETER FILE
REM*************************************************************************
1200  FOR I=1 TO 7 :READ COMND(I) :NEXT I
      DATA "TITLE","REVAR","MODEL","FIXED","RANDM","DATFN","EMSQR"
      FOR I=8 TO 12 :READ COMND(I) :NEXT I
      DATA "COVAR","INDEP","DEPEN","ERRLN","SORCE"
      PRINT "ENTER DISK DRIVE AND JOB PARAMETER FILE NAME"
      INPUT "SEPARATE WITH A COLON (e.g. B:FILENAME.EXT)";CPFN
REM
      OPEN "I",#1,CPFN
      K=1 :WHILE ((MID$(CPFN,K,1) <> ".") AND (K < 20)) :K=K+1 :WEND
      IF K > 5 THEN FLN=MID$(CPFN,1,5)
      IF K <= 5 THEN FLN=MID$(CPFN,1,K-1)
REM
      NV=0 :NT=0 :NDMC=0 :KNT=0 :EMSQR="0" :RES=0 :COV="0" :NCM=0
REM
      PRINT " " :PRINT "PARAMETER FILE STRUCTURE " :PRINT " "
1210  IF EOF(1) THEN GOTO 1600
      LINE INPUT #1,CLINE
      PRINT CLINE
      KNT=KNT+1
      FOR I=1 TO 13
        IF I = 13 THEN_
          PRINT "Error In Parameter File Specification"_
                " See DMAOV.BAS Users Guide" :CLOSE #1 :STOP
        IF MID$(CLINE,1,5)=COMND(I) THEN_
```

```
          ON I GOTO 1220,1230,1240,1250,1250,1260,1270,1280,_
                    1290,1300,1310,1320
      NEXT I
REM
REM   TITLE
1220  NT=NT+1
      CTITLE(NT)=MID$(CLINE,7,74)
      GOTO 1210
REM
REM   REVAR
1230  NV=NV+1
      CLOSE #1
      OPEN "I",#1,CPFN
      FOR I=1 TO KNT :LINE INPUT #1,CLINE :NEXT I
      PT1=1 :L=6 :WHILE (MID$(CLINE,L,1) = " ") :L=L+1 :WEND
1231  FOR I=L TO 200
        IF ((MID$(CLINE,I,1) = ",") OR (MID$(CLINE,I,1) = ";")) THEN_
          GOTO 1232
      NEXT I
1232  IF PT1 = 1 THEN VAR(NV) = MID$(CLINE,L,I-L) :_
        L=I+1 :PT1=PT1+1 :GOTO 1231
      IF (PT1 = 2) AND (MID$(CLINE,I,1) = ",") THEN_
        CBEG = MID$(CLINE,L,I-L) :NBEG(NV)=VAL(CBEG) :_
        L=I+1 :PT1=PT1+1 :GOTO 1231
      IF (PT1 = 2) AND (MID$(CLINE,I,1) = ";") THEN_
        GOTO 1233
      IF PT1 = 3 THEN CWID = MID$(CLINE,L,I-L) :NWID(NV)=VAL(CWID) :_
        L=I+1 :PT1=PT1+1 :GOTO 1231
      IF PT1 = 4 THEN CDEC = MID$(CLINE,L,I-L) :NDEC(NV)=VAL(CDEC) :_
        GOTO 1233
1233  IF NV=1 THEN GOTO 1234 ELSE GOTO 1210
1234    NF=0
        FOR I=1 TO 10
          IF MID$(VAR(NV),I,1)="(" THEN GOTO 1236 ELSE GOTO 1239
```

```
1236          J=I+1
              FOR K=1 TO 30
                IF MID$(VAR(NV),J,1)<>")" THEN_
                  VARS(K)=MID$(VAR(NV),J,1) :J=J+1 :NF=NF+1
                IF MID$(VAR(NV),J,1)=")" THEN GOTO 1210
              NEXT K
1239        NEXT I
REM
REM     MODEL
1240    J=0
        K=6 :WHILE(MID$(CLINE,K,1) = " ") :K=K+1 :WEND
        L=K
1241    IF ((MID$(CLINE,L,1) = "+") OR (MID$(CLINE,L,1) = " ")) THEN_
           GOTO 1243 ELSE GOTO 1242
1242      IF MID$(CLINE,L,1) = ";" THEN NDMC=NDMC+1 :_
             DMC(NDMC)=MID$(CLINE,K,L-K) :J=1 :GOTO 1244
          L=L+1 :IF L = 255 THEN PRINT "Error with model command" :_
                   Print "Missing semicolon" :END
          GOTO 1241
1243      NDMC=NDMC+1
          DMC(NDMC)=MID$(CLINE,K,L-K)
          K=L+1
          WHILE((MID$(CLINE,K,1) = " ") OR (MID$(CLINE,K,1) = "+")) :_
          K=K+1 :WEND
          L=K
1244      IF (DMC(NDMC)="RESIDUAL") THEN NDMC=NDMC-1 :RES=1
          IF (DMC(NDMC)="residual") THEN NDMC=NDMC-1 :RES=1
          IF (DMC(NDMC)="Residual") THEN NDMC=NDMC-1 :RES=1
          IF J = 1 THEN GOTO 1210
        GOTO 1241
REM
REM     FIXED or RANDM (Must Follow REVAR in parameter file)
1250    CLOSE #1
        OPEN "I",#1,CPFN
```

```
      FOR I=1 TO KNT :LINE INPUT #1,CLINE :NEXT I
      I=6 :WHILE (MID$(CLINE,I,1) = " ") :I=I+1 :WEND
      FC=MID$(CLINE,I,1)
      FOR I = 6 TO 80
        IF MID$(CLINE,I,1) = ";" THEN GOTO 1254
        IF MID$(CLINE,I,1) = "," THEN K=I+1 :GOTO 1252
      NEXT I
1252  FOR I = K TO 80
        IF MID$(CLINE,I,1) = ";" THEN L=I :GOTO 1253
      NEXT I
1253  DC=MID$(CLINE,K,L-K)
      NLL=VAL(DC)
1254  FOR I=1 TO NF
        IF FC=VARS(I) THEN NFL(I)=NLL :K=I :GOTO 1256
      NEXT I
1256  IF MID$(CLINE,1,5)="FIXED" THEN CROF(K)="0" :GOTO 1210
      IF MID$(CLINE,1,5)="RANDM" THEN CROF(K)="1" :GOTO 1210
REM
REM   DATFN
1260  L=6 :WHILE(MID$(CLINE,L,1) = " ") :L=L+1 :WEND
      FOR I=L TO 80
        IF MID$(CLINE,I,1)=";" THEN
          DFLN=MID$(CLINE,L,I-L) :GOTO 1210
        IF I = 80 THEN PRINT "Error with DATFN command" :_
          PRINT "Missing semicolon" :END
      NEXT I
REM
REM   EMSQR = EXPECTED MEAN SQUARES ONLY
1270  EMSQR="1" :GOTO 1210
REM
REM   COVARIANCE AND NUMBER OF MODELS
1280  COV="1"
      FT="" :FT=FLN :FT=FT+"MOD.TMP" :OPEN "O",#2,FT
      FT="" :FT=FLN :FT=FT+"SRC.TMP" :OPEN "O",#3,FT
```

```
      K=6 :WHILE(MID$(CLINE,K,1) = " ") :K=K+1 :WEND
      FOR I = K TO 80
        IF MID$(CLINE,I,1) = ";" THEN L=I :GOTO 1282
        IF I = 80 THEN PRINT "Error with COVAR command" :_
          PRINT "Missing semicolon" :END
      NEXT I
1282  DC=MID$(CLINE,K,L-K)
      NMODEL=VAL(DC) :PRINT #2,NMODEL  :GOTO 1210
REM
REM   INDEP = NO. OF INDEPENDENT VARIABLES AND LOCATION (MUST FOLLOW COVAR
1290  K=6 :WHILE(MID$(CLINE,K,1) = " ") :K=K+1 :WEND
      NX=0
1291  FOR I = K TO 80
        IF (MID$(CLINE,I,1) = ";") OR (MID$(CLINE,I,1) = ",") THEN_
          L=I :GOTO 1292
        IF I = 80 THEN PRINT "Error with INDEP command" :_
          PRINT "Missing semicolon" :END
      NEXT I
1292  DC=MID$(CLINE,K,L-K)
      NX=NX+1 :IDX(NX)=VAL(DC) :K=L+1
      IF MID$(CLINE,L,1) = "," THEN GOTO 1291
      IF MID$(CLINE,L,1) = ";" THEN GOTO 1210
REM
REM   DEPEN = NO. OF DEPENDENT VARIABLES AND LOCATION (MUST FOLLOW DEPEN)
1300  K=6 :WHILE(MID$(CLINE,K,1) = " ") :K=K+1 :WEND
      NY=0
1301  FOR I = K TO 80
        IF (MID$(CLINE,I,1) = ";") OR (MID$(CLINE,I,1) = ",")  THEN_
          L=I :GOTO 1302
        IF I = 80 THEN PRINT "Error with DEPEN command" :_
          PRINT "Missing semicolon" :END
      NEXT I
1302  DC=MID$(CLINE,K,L-K)
      NY=NY+1 :IDY(NY)=VAL(DC) :K=L+1
```

```
      IF MID$(CLINE,L,1) = "," THEN GOTO 1301
      IF MID$(CLINE,L,1) = ";" THEN GOTO 1303
1303  PRINT #2,NX;NY
      FOR I = 1 TO NX :PRINT #2,IDX(I); :NEXT I
      FOR I = 1 TO NY :PRINT #2,IDY(I); :NEXT I
      PRINT #2,"" :GOTO 1210
REM
REM   ERRLN = ERROR LINE FOR COVARIANCE (MUST PRECEDE SORCE(S) GROUPS)
1310  NCM=NCM+1
      K=6 :WHILE(MID$(CLINE,K,1) = " ") :K=K+1 :WEND
      PRINT #3,1
      FOR I = K TO 80
        IF MID$(CLINE,I,1) = ";" THEN L=I :GOTO 1312
        IF I = 80 THEN PRINT "Error with ERRLN command" :_
          PRINT "Missing semicolon" :END
      NEXT I
1312  PRINT #3,MID$(CLINE,K,L-K)
      PRINT #3,-1 :GOTO 1210
REM
REM   SORCE = ADJUSTED SOURCES OF VARIATION IN COVARIANCE ANALYSIS
1320  NCM=NCM+1
      K=6 :WHILE(MID$(CLINE,K,1) = " ") :K=K+1 :WEND :J=0
      FOR I = K TO 80
        IF(MID$(CLINE,I,1) = "+") OR (MID$(CLINE,I,1) = ";") THEN_
          J=J+1
      NEXT I
      PRINT #3,J
      L=K
1322  IF((MID$(CLINE,L,1) = "+") OR (MID$(CLINE,L,1) = " ")) THEN_
        GOTO 1326
1323  IF MID$(CLINE,L,1) = ";"  THEN_
        GOTO 1326 ELSE GOTO 1324
1324    L=L+1
        IF L = 255 THEN_
```

```
            PRINT "Error with SORCE command, missing semicolon" :END
          GOTO 1322
1326  PRINT #3,MID$(CLINE,K,L-K)
      IF MID$(CLINE,L,1) = ";" THEN GOTO 1328
      K=L+1
      WHILE((MID$(CLINE,K,1)=" ")OR(MID$(CLINE,K,1)="+")) :K=K+1 :WEND
      L=K :GOTO 1322
1328  PRINT #3,1 :GOTO 1210
REM
1600  CLOSE #1
      IF COV = "1" THEN :CLOSE #3 :CLOSE #2
      PRINT " " :LINE INPUT "CONTINUE ? (Y OR N) ";CANS
      IF (CANS <> "Y") AND (CANS <> "y") THEN STOP
      RETURN
REM*************************************************************************
REM SUBROUTINE:  DATA INPUT
REM*************************************************************************
4300  NOBS=1
      FOR I=1 TO NF
        NOBS=NOBS*NFL(I)
      NEXT I
      NDF(1)=NOBS
      NDF(2)=1
REM
      OPEN "I",#5,DFLN
      FT="" :FT=FLN :FT=FT+"COP.DAT" :OPEN "O",#4,FT
REM
      FOR I=1 TO NV
        S(I)=0.0
        FOR J = I TO NV :A(I,J)=0.0 :NEXT J
      NEXT I
      PRINT " " :PRINT "FIRST FIVE OBSERVATIONS ARE " :PRINT " "
      FOR K= 1 TO NOBS
        LINE INPUT #5,CSTRG
```

```
        FOR I=1 TO NV
          CH=MID$(CSTRG,NBEG(I),NWID(I))
          ID=INSTR(CH,".")
          X(I)=CDBL(VAL(CH))
          IF ID > 0 THEN GOTO 4305
          IF NDEC(I) > 0 THEN
            FOR M=1 TO NDEC(I) :X(I)=X(I)/10.0 :NEXT M
          IF NDEC(I) < 0 THEN
            FOR M=1 TO NDEC(I) :X(I)=X(I)*10.0 :NEXT M
4305      IF K <= 5 THEN PRINT X(I);
          IF (K <= 5) AND (I MOD 10 = 0) THEN PRINT " "
          PRINT #4,X(I);
          IF I MOD 10 = 0 THEN PRINT #4," "
          S(I)=S(I)+X(I)
          IF COV="0" THEN
            A(I,I)=A(I,I)+X(I)*X(I)
          IF COV="1" THEN
            FOR J = 1 TO I :A(J,I)=A(J,I)+X(I)*X(J) :NEXT J
        NEXT I
        IF (K <= 5) THEN PRINT " "
        IF NV MOD 10 <> 0 THEN PRINT #4," "
      NEXT K
      PRINT " "
      LINE INPUT "CONTINUE ? (Y OR N) ";CANS
      IF (CANS <> "Y") AND (CANS <> "y") THEN STOP
REM
      FT="" :FT=FLN :FT=FT+"TOT.OUT" :OPEN "O",#6,FT
REM
      PRINT #6,NDF(2);NDF(1)
REM
      FT="" :FT=FLN :FT=FT+"USS.OUT" :OPEN "O",#7,FT
REM
      G="11111111111111111111"
      H="00000000000000000000"
```

```
      PRINT #2,"AVERAGES FOR" TAB(14) MID$(G,1,NF)
      PRINT #2,"------------"
      PRINT #2,"DIV = " TAB(7) NOBS :PRINT #2,"LEVEL = OVERALL"
      FOR I=1 TO NV
        PRINT #2,S(I)/NOBS; :IF I MOD 5 = 0 THEN PRINT #2,""
        PRINT #6,S(I); :IF I MOD 5 = 0 THEN PRINT #6,""
      NEXT I
      PRINT #2,"" :PRINT #6,""
      PRINT #7,MID$(G,1,NF)
      PRINT #7,NDF(1)
      K=1
      FOR I = 1 TO NV
        IF COV="0" THEN GOTO 4310 ELSE GOTO 4320
4310      PRINT #7,A(I,I);
          IF K MOD 5 = 0 THEN PRINT #7,""
          K=K+1 :GOTO 4330
4320    FOR J = I TO NV
          PRINT #7,A(I,J);
          IF K MOD 5 = 0 THEN PRINT #7,""
          K=K+1
        NEXT J
4330  NEXT I
      IF (K-1) MOD 5 <> 0 THEN PRINT #7,""
      PRINT #7,MID$(H,1,NF)
      PRINT #7,NDF(2)
      K=1
      FOR I = 1 TO NV
        IF COV="0" THEN GOTO 4340 ELSE GOTO 4350
4340      PRINT #7,(S(I)*S(I))/NOBS;
          IF K MOD 5 = 0 THEN PRINT #7,""
          K=K+1 :GOTO 4360
4350    FOR J = I TO NV
          PRINT #7,(S(I)*S(J))/NOBS;
          IF K MOD 5 = 0 THEN PRINT #7,""
```

```
          K=K+1
        NEXT J
4360  NEXT I
      IF (K-1) MOD 5 <> 0 THEN PRINT #7,""
REM
      CLOSE #5 :CLOSE #7
REM
      RETURN
REM*************************************************************************
REM SUBROUTINE:  COUNT SUBSCRIPTS
REM*************************************************************************
2460  NS=0 :M=1 :NNSS=0
      FOR I = 1 TO 21
        IF I = 21 THEN PRINT "Error with model design" :STOP
        IF MID$(DMC(R),I,1) = "(" THEN J=I+1 :GOTO 2462
      NEXT I
2462  FOR K = 1 TO 21
        IF (MID$(DMC(R),J,1) <> ")") AND (MID$(DMC(R),J,1) <> "/") THEN_
          FS(K)=MID$(DMC(R),J,1) :J=J+1 :NS=NS+1
        IF MID$(DMC(R),J,1)="/" THEN GOTO 2464
        IF MID$(DMC(R),J,1)=")" THEN GOTO 2466
        IF K=21 THEN PRINT "Error with model design" :STOP
      NEXT K
2464  FOR L = J+1 TO 21
        IF MID$(DMC(R),L,1) <> ")" THEN_
          CNSS(M)=MID$(DMC(R),L,1) :NNSS=NNSS+1 :M=M+1
        IF MID$(DMC(R),L,1) = ")" THEN GOTO 2468
        IF L=21 THEN PRINT "Error with model design" :STOP
      NEXT L
2466  CNSS(1)="0" :NNSS=0 :GOTO 2468
2468  RETURN
REM*************************************************************************
REM SUBROUTINE:  BINARY ARITHMETIC
REM*************************************************************************
```

```
2810  FOR I = 1 TO 2^NS :NBINSUM(I)=0 :NEXT I
      FOR J = 1 TO NS :Q(J)=0 :NBINCOD(1,J)=Q(J) :NEXT J
      RR=2
2812  Q(1)=Q(1)+1 :N=1
2814  IF Q(N) = 2 THEN GOTO 2816 ELSE GOTO 2818
2816    Q(N+1)=Q(N+1)+1 :Q(N)=0
        IF (Q(N+1)=2) AND ((N+1) <> NS) THEN_
          N=N+1 :GOTO 2814
        IF (Q(N+1)=2) AND ((N+1) = NS) THEN_
          GOTO 2820
2818  FOR I = 1 TO NS
        NBINCOD(RR,I)=Q(I)
        NBINSUM(RR)=NBINSUM(RR)+Q(I)
      NEXT I
      RR=RR+1
      IF NS=1 GOTO 2820
      GOTO 2812
2820  RETURN
REM*************************************************************************
REM SUBROUTINE:  QUICK SORT
REM*************************************************************************
REM   THIS SUBROUTINE WAS REWRITTEN IN MBASIC FROM FORTRAN QKSORT (HURST)
REM   WITH MODIFICATIONS TO SUIT THIS PROGRAMS NEEDS
3140  N=2^NS :LV(1)=1 :RUV(1)=N :P=1
3145  IF P < 1 THEN GOTO 3161
3147  IF (RUV(P)-LV(P)) >= 1 THEN GOTO 3149
      P=P-1 :GOTO 3145
3149  LP=LV(P)-1 :RUP=RUV(P) :NY=NBINSUM(RUP)
      FOR I = 1 TO NS :Q(I)=NBINCOD(RUP,I) :NEXT I
3151  IF (RUP-LP) < 2 THEN GOTO 3157
      LP=LP+1
      IF NBINSUM(LP) <= NY THEN GOTO 3151
      NBINSUM(RUP)=NBINSUM(LP)
      FOR I = 1 TO NS :NBINCOD(RUP,I)=NBINCOD(LP,I) :NEXT I
```

```
3153  IF (RUP-LP) < 2 THEN GOTO 3155
      RUP=RUP-1
      IF NBINSUM(RUP) >= NY THEN GOTO 3153
      NBINSUM(LP)=NBINSUM(RUP)
      FOR I = 1 TO NS :NBINCOD(LP,I)=NBINCOD(RUP,I) :NEXT I
      GOTO 3151
3155  RUP=RUP-1
3157  NBINSUM(RUP)=NY
      FOR I = 1 TO NS :NBINCOD(RUP,I)=Q(I) :NEXT I
      IF (RUP-LV(P)) < (RUV(P)-RUP) THEN GOTO 3159
      LV(P+1)=RUP+1 :RUV(P+1)=RUV(P) :RUV(P)=RUP-1 :P=P+1
      GOTO 3147
3159  LV(P+1)=LV(P) :RUV(P+1)=RUP-1 :LV(P)=RUP+1 :P=P+1
      GOTO 3147
3161  RETURN
REM*************************************************************************
REM SUBROUTINE:  COMBINE SUBSCRIPT BINARY CODES
REM*************************************************************************
3650  FOR I = 1 TO NF
        EMSMAT(I)="#"
        FOR J = 1 TO 2^NS :NBINCOD(J,I)=0 :NEXT J
      NEXT I
      J=1 :L=1
      FOR I = 1 TO NF
        IF FS(L)=VARS(J) THEN GOTO 3652 ELSE GOTO 3654
3652      FOR K = 1 TO 2^NS
            NBINCOD(K,J)=NSBNCD(K,L)
          NEXT K
          EMSMAT(J)=CROF(J) :J=J+1 :L=L+1
          GOTO 3656
3654      FOR K = 1 TO 2^NS
            NBINCOD(K,I)=0
          NEXT K
          EMSMAT(J)=VARS(J) :J=J+1
```

```
          GOTO 3656
3656  NEXT I
      J=1 :L=1
      FOR I = 1 TO NF
        IF NNSS = 0 THEN GOTO 3670
        IF CNSS(L) = VARS(J) THEN GOTO 3658 ELSE GOTO 3660
3658      FOR K = 1 TO 2^NS
            NBINCOD(K,I)=1
          NEXT K
          EMSMAT(J)="1" :L=L+1 :J=J+1
          GOTO 3662
3660      J=J+1 :GOTO 3662
3662  NEXT I
3670
REM   FOR I = 1 TO 2^NS
REM     FOR J = 1 TO NF :PRINT NBINCOD(I,J);   :NEXT J
REM     PRINT " "
REM   NEXT I
      RETURN
REM*************************************************************************
REM SUBROUTINE:  LINEAR COMBINATIONS
REM*************************************************************************
4140  LC(1)=1 :PT1=1
      FOR I = 2 TO 2^NS
        IF NSBNSUM(I)=NSBNSUM(I-1) THEN_
          LC(I)=LC(I-1)*1   :PT1=I
        IF NSBNSUM(I) <> NSBNSUM(I-1) THEN_
          LC(I)=LC(PT1)*(-1)
      NEXT I
REM   FOR I = 1 TO 2^NS :PRINT LC(I) :NEXT I
      RETURN
REM*************************************************************************
REM SUBROUTINE: UNCORRECTED SUMS OF SQUARES
REM*************************************************************************
```

```
REM    SOME PARTS OF THIS SUBROUTINE WAS EXTRACTED FROM FCT.BAS
REM    DEVELOPED BY DR. REX HURST, UTAH STATE UNIVERSITY
4790   FT="" :FT=FLN :FT=FT+"USS.OUT" :OPEN "I",#7,FT
       FT="" :FT=FLN :FT=FT+"USS.COP" :OPEN "O",#8,FT
REM
       FOR M = 1 TO 2^NS
         CBT1=""
         FOR I = 1 TO NF
           CBTA(I)=STR$(NBINCOD(M,I))
           CBT1=CBT1+MID$(CBTA(I),2,1)
         NEXT I
REM      PRINT CBT1
         FOR I = 1 TO 257
           IF EOF (7) THEN GOTO 4795
           INPUT #7,CBT2 :PRINT #8,CBT2
           INPUT #7,NDFT :PRINT #8,NDFT
           L=1
           FOR J = 1 TO NV
             IF COV = "0" THEN GOTO 4791 ELSE GOTO 4792
4791           INPUT #7,A(J,J)
               PRINT #8,A(J,J);
               IF L MOD 5 = 0 THEN PRINT #8,""
               L=L+1 :GOTO 4793
4792         FOR K = J TO NV
               INPUT #7,A(J,K)
               PRINT #8,A(J,K);
               IF L MOD 5 = 0 THEN PRINT #8,""
               L=L+1
             NEXT K
4793       NEXT J
           IF (L-1) MOD 5 <> 0 THEN PRINT #8,""
           IF CBT1=CBT2 THEN GOTO 5762
         NEXT I
4795     JNF=0 :JLEN=1
```

```
        FOR I = 1 TO NV
          FOR J = I TO NV :A(I,J)=0.0 :NEXT J
        NEXT I
        FOR I = 1 TO NF
          IF NBINCOD(M,I) = 1 THEN JNF=JNF+1 :_
            JLEN=JLEN*NFL(I) :JNL(JNF)=NFL(I)
        NEXT I
        FOR I = 1 TO JLEN*NV :TOT(I)=0.0 :NEXT I
        NDIV=NOBS/JLEN :NDF(M+2)=JLEN :JJNX(JNF)=1
        IF JNF > 1 THEN GOTO 4798 ELSE GOTO 4799
4798      FOR I = 1 TO JNF-1
            J=JNF-I :JJNX(J)=JJNX(J+1)*JNL(J+1)
          NEXT I
4799    J=0
        FOR I = 1 TO NF
          IF NBINCOD(M,I)=0 THEN JNX(I)=0 ELSE_
            J=J+1 :JNX(I)=JJNX(J)
          IX(I)=1
        NEXT I
        CLOSE #4
        FT="" :FT=FLN :FT=FT+"COP.DAT" :OPEN "I",#4,FT
        FOR L = 1 TO NOBS
          FOR I = 1 TO NV :INPUT #4,X(I) :NEXT I
          NLOC=1
          FOR I = 1 TO NF
            NLOC=NLOC+(IX(I)-1)*JNX(I)
          NEXT I
          N1=((NLOC-1)*NV)+1
          N2=(N1+NV)-1
          I=1
          FOR J = N1 TO N2
            TOT(J)=TOT(J)+X(I) :I=I+1
          NEXT J
          IX(NF)=IX(NF)+1
```

```
          IF IX(NF) <= NFL(NF) THEN GOTO 4807
          ISW=0 :II=2
          WHILE ((II <= NF) AND (ISW=0))
            I=NF+1-II
            IX(I+1)=1
            IX(I)=IX(I)+1
            IF IX(I) <= NFL(I) THEN ISW=1
            II=II+1
          WEND
4807    NEXT L
        PRINT #2,"" :PRINT #2,"AVERAGES FOR" TAB(14) CBT1
        PRINT #2,"------------"
        PRINT #2,"DIV = " TAB(7) NDIV
        PRINT #6,JLEN;NDIV
        KK=1
        FOR I = 1 TO JLEN
          PRINT #2,"LEVEL ";
          L=I
          FOR MM = 1 TO JNF-1
            PRINT #2,((L-1)\JJNX(MM)+1);
            L=((L-1) MOD JJNX(MM))+1
          NEXT MM
          PRINT #2,L
          FOR J = 1 TO NV
            TTOT(J)=TOT(KK)
            PRINT #2,TOT(KK)/NDIV; :IF J MOD 5 = 0 THEN PRINT #2,""
            PRINT #6,TOT(KK); :IF J MOD 5 = 0 THEN PRINT #6,""
            KK=KK+1
          NEXT J
          PRINT #2,"" :PRINT #6,""
          FOR J = 1 TO NV
            IF COV="0" THEN GOTO 4810 ELSE GOTO 4812
4810          A(J,J)=A(J,J)+TTOT(J)*TTOT(J)/NDIV
              GOTO 4814
```

```
4812          FOR K = J TO NV
                A(J,K)=A(J,K)+TTOT(J)*TTOT(K)/NDIV
              NEXT K
4814        NEXT J
          NEXT I
          PRINT #8,CBT1
          PRINT #8,NDF(M+2)
          K=1
          FOR I = 1 TO NV
            IF COV = "0" THEN GOTO 4820 ELSE GOTO 4822
4820          PRINT #8,A(I,I);
              IF K MOD 5 = 0 THEN PRINT #8,""
              K=K+1 :GOTO 4824
4822        FOR J = I TO NV
              PRINT #8, A(I,J);
              IF K MOD 5 = 0 THEN PRINT #8,""
              K=K+1
            NEXT J
4824      NEXT I
          IF (K-1) MOD 5 <> 0 THEN PRINT #8,""
          CLOSE #7
          FT1="" :FT1=FLN :FT1=FT1+"USS.OUT" :KILL FT1
          CLOSE #8 :FT="" :FT=FLN :FT=FT+"USS.COP"
          NAME FT AS FT1
          FT="" :FT=FLN :FT=FT+"USS.OUT" :OPEN "I",#7,FT
          FT="" :FT=FLN :FT=FT+"USS.COP" :OPEN "O",#8,FT
          GOTO 5764
5762      CLOSE #7
          FT="" :FT=FLN :FT=FT+"USS.OUT" :OPEN "I",#7,FT
          CLOSE #8
          FT="" :FT=FLN :FT=FT+"USS.COP" :KILL FT
          FT="" :FT=FLN :FT=FT+"USS.COP" :OPEN "O",#8,FT
5764    NEXT M
        CLOSE #4 :CLOSE #7 :CLOSE #8
```

```
      FT="" :FT=FLN :FT=FT+"USS.COF" :KILL FT
      RETURN
REM*************************************************************************
REM SUBROUTINE:  ANOVA TABLE
REM*************************************************************************
5880  FOR I = 1 TO NV
        FOR J = I TO NV
           A(I,J)=0.0 :SS(I,J)=0.0
        NEXT J
      NEXT I
      NDGF=0
      FT="" :FT=FLN :FT=FT+"USS.OUT" :OPEN "I",#7,FT
      FOR M = 1 TO 2^NS
        CBT1=""
        FOR I = 1 TO NF
          CBTA(I)=STR$(NBINCOD(M,I))
          CBT1=CBT1+MID$(CBTA(I),2,1)
        NEXT I
5882    INPUT #7,CBT2
        INPUT #7,NNDF
        FOR J = 1 TO NV
          IF COV = "0" THEN_
            INPUT #7,A(J,J)
          IF COV = "1" THEN_
            FOR K = J TO NV:INPUT #7,A(J,K) :NEXT K
        NEXT J
        IF CBT1 = CBT2 THEN GOTO 5883 ELSE GOTO 5884
5883      FOR L = 1 TO NV
            IF COV = "0" THEN_
              SS(L,L)=SS(L,L)+A(L,L)*LC(M)
            IF COV = "1" THEN_
              FOR K = L TO NV:SS(L,K)=SS(L,K)+A(L,K)*LC(M) :NEXT K
          NEXT L
          NDGF=NDGF+NNDF*LC(M)
```

```
        GOTO 5886
5884    GOTO 5882
5886    CLOSE #7
        FT="" :FT=FLN :FT=FT+"USS.OUT" :OPEN "I",#7,FT
      NEXT M
      NADF=NADF+NDGF :K=1
      IF COV = "1" THEN
        PRINT #1,DMC(R) :PRINT #1,NDGF :PRINT #1,NBC :NBC=NBC+1
      FOR I = 1 TO NV
        IF COV = "0" THEN
          ASS(I,I)=ASS(I,I)+SS(I,I)
        IF COV = "1" THEN GOTO 5888 ELSE GOTO 5890
5888      FOR J = I TO NV
            ASS(I,J)=ASS(I,J)+SS(I,J)
            PRINT #1,SS(I,J);
            IF K MOD 5 = 0 THEN PRINT #1,""
            K=K+1
          NEXT J
5890    PRINT #5,USING FOR4; I; DMC(R); NDGF; SS(I,I); SS(I,I)/NDGF
      NEXT I
      IF COV = "1" THEN GOTO 5891 ELSE GOTO 5892
5891    IF (K-1) MOD 5 <>.0 THEN PRINT #1,""
5892  CLOSE #7
      RETURN
REM*************************************************************************
REM SUBROUTINE:  EXPECTED MEAN SQUARES
REM*************************************************************************
6120  FT="" :FT=FLN :FT=FT+"EMS.MAT" :OPEN "I",#3,FT
      PRINT ""
      PRINT "PROCESSING EXPECTED MEAN SQUARES" TAB(35) "-- PLEASE WAIT --"
      FT="" :FT=FLN :FT=FT+"EMS.OUT" :OPEN "O",#4,FT
      IF NT > 0 THEN
        FOR I = 1 TO NT :PRINT #4,CTITLE(I) :NEXT I :PRINT #4," "
      PRINT #4,TAB(4) "SOURCE" TAB(31) "EXPECTED VALUES OF MEAN SQUARES"
```

```
      PRINT #4,TAB(4) "------" TAB(31) "------------------------------"
      FOR I = 1 TO NF :CFAC=CFAC+VARS(I) :NEXT I
      PT1=1
6121  FOR I = 1 TO PT1
        CE1="" :NSUB=0 :CSUB="" :NNSS=0 :CE3=""
        IF EOF(3) GOTO 6199
        INPUT #3,CE1 :INPUT #3,NSUB :INPUT #3,CSUB
        INPUT #3,NNSS :INPUT #3,CE3
      NEXT I
      CLOSE #3
      FT="" :FT=FLN :FT=FT+"EMS.MAT" :OPEN "I",#3,FT
      M=1 :CTSUB="####################"
      FOR I = 1 TO NF
        IF MID$(CSUB,M,1)=MID$(CFAC,I,1) THEN GOTO 6122 ELSE GOTO 6124
6122      MID$(CTSUB,I,1)=MID$(CSUB,M,1)
          M=M+1  :GOTO 6126
6124      MID$(CTSUB,I,1)="#"
6126  NEXT I
      EMS=""  :L=1  :EMS=EMS+"*"  :L=L+1
      FOR I = 1 TO NDMC
        M=1 :COMP1="####################"
        COEF="" :NS=0 :FS="" :NNSS=0 :CNSS=""
        INPUT #3,COEF :INPUT #3,NS :INPUT #3,FS
        INPUT #3,NNSS :INPUT #3,CNSS
        FOR J = 1 TO NF
          IF MID$(FS,M,1)=MID$(CFAC,J,1) THEN
            MID$(COMP1,J,1)=MID$(FS,M,1) :M=M+1 ELSE MID$(COMP1,J,1)="%"
        NEXT J
        IF NNSS > 0 THEN GOTO 6128 ELSE GOTO 6130
6128    M=1
        FOR J = 1 TO NF
          IF MID$(CNSS,M,1)=MID$(CFAC,J,1) THEN
            MID$(COMP1,J,1)=MID$(CNSS,M,1) :M=M+1
        NEXT J
```

```
6130    COMP2="####################"
        FOR J = 1 TO NSUB
          FOR K = 1 TO NF
            IF MID$(CSUB,J,1)=MID$(CFAC,K,1) THEN_
              MID$(COMP2,J,1)=MID$(COMP1,K,1)
          NEXT K
        NEXT J
        IF MID$(COMP2,1,NSUB)=MID$(CSUB,1,NSUB) THEN_
          NFLAG=1 ELSE NFLAG=0
        IF NFLAG = 1 THEN GOTO 6132 ELSE GOTO 6140
6132      FOR J = 1 TO NF
            IF MID$(CTSUB,J,1) <> MID$(CFAC,J,1) THEN GOTO 6134 ELSE_
              GOTO 6136
6134          FOR Q = 1 TO NF
                IF ((MID$(COEF,Q,1)="0") AND_
                   (MID$(CTSUB,Q,1)<>MID$(CFAC,Q,1))) THEN GOTO 6140
              NEXT Q
              IF MID$(COEF,J,1)="1" THEN GOTO 6136
              EMS=EMS+MID$(COEF,J,1)
              L=L+1
6136      NEXT J
          EMS=EMS+"(V)"
          L=L+3
          EMS=EMS+FS
           L=L+NS
           IF NNSS > 0 THEN GOTO 6137 ELSE GOTO 6138
6137        EMS=EMS+"/"
            L=L+1
            EMS=EMS+CNSS
            L=L+NNSS
6138      EMS=EMS+"+" :L=L+1
6140  NEXT I
      NC1=L-2 :FEMS=""
      IF RES = 1 THEN FEMS="(V)RES + "
```

```
6146  FOR I = L-2 TO 1 STEP -1
        NC4=I
        IF (MID$(EMS,I,1)="+") OR (I=1) THEN GOTO 6148 ELSE GOTO 6150
6148      FEMS=FEMS+MID$(EMS,NC4+1,NC1-NC4)
          IF I > 1 THEN FEMS=FEMS+" + "
          NC1=NC4-1
6150  NEXT I
      PRINT #4,DMC(PT1) TAB(21) FEMS
      FEMS=""
      CLOSE #3
      FT="".:FT=FLN :FT=FT+"EMS.MAT" :OPEN "I",#3,FT
      PT1=PT1+1
      GOTO 6121
6199  IF RES = 1 THEN PRINT #4,"RESIDUAL" TAB(21) "(V)RES"
      CLOSE #3
      FT="" :FT=FLN :FT=FT+"EMS.MAT" :KILL FT
      CLOSE #4
      RETURN
REM*************************************************************************
REM SUBROUTINE:  CORRECTED SS/SP MATRICES BLOCK
REM*************************************************************************
7000  PRINT ""
      PRINT "PROCESSING CORRECTED SS/SP MATRIX" TAB(35) "-- PLEASE WAIT --"
      FT="" :FT=FLN :FT=FT+"CSS.OUT" :OPEN "O",#5,FT
      FT="" :FT=FLN :FT=FT+"SRC.TMP" :OPEN "I",#3,FT
      FT="" :FT=FLN :FT=FT+"CSS.TMP" :OPEN "I",#1,FT
7010  IF EOF(3) THEN GOTO 7040
      INPUT #3,NS
      NTDF=0 :CNAME=""
      FOR I = 1 TO NV
        FOR J = I TO NV :A(I,J)=0.0 :AT(I,J)=0.0 :NEXT J
      NEXT I
      FOR I = 1 TO NS
        LINE INPUT #3,CNM1
```

```
7020    IF EOF(1) THEN_
          PRINT "ERROR WITH COVARIANCE PARAMETER FILE SPECIFICATIONS" :END
        LINE INPUT #1,CNM2
        INPUT #1,NDF
        INPUT #1,IDLINE
        IF I = 1 THEN ID=IDLINE
        FOR J = 1 TO NV
          FOR K = J TO NV :INPUT #1,A(J,K) :NEXT K
        NEXT J
        IF CNM1 = CNM2 THEN GOTO 7030 ELSE GOTO 7020
7030    FOR J = 1 TO NV
          FOR K = J TO NV
            AT(J,K)=AT(J,K)+A(J,K)
          NEXT K
        NEXT J
        NTDF=NTDF+NDF
        IF I = 1 THEN CNAME=CNAME+CNM1 ELSE_
          CNAME=CNAME+" + " :CNAME=CNAME+CNM1
        CLOSE #1 :FT="" :FT=FLN :FT=FT+"CSS.TMP" :OPEN "I",#1,FT
      NEXT I
      INPUT #3,MSIGN
      PRINT #5,CNAME
      IF MSIGN < 0 THEN ID=0
      PRINT #5,MSIGN;NTDF;ID
      FOR I = 1 TO NV
        FOR J = I TO NV
          PRINT #5,AT(I,J);
          IF J MOD 5 = 0 THEN PRINT #5,""
        NEXT J
        IF (NV-I+1) MOD 5 <> 0 THEN PRINT #5,""
      NEXT I
      CLOSE #1
      FT="" :FT=FLN :FT=FT+"CSS.TMP" :OPEN "I",#1,FT
      GOTO 7010
```

```
7040  CLOSE #3 :FT="" :FT=FLN :FT=FT+"SRC.TMP" :KILL FT
      CLOSE #1 :FT="" :FT=FLN :FT=FT+"CSS.TMP" :KILL FT
      CLOSE #5
      RETURN
REM*************************************************************************
REM SUBROUTINE:  COVARIANCE BLOCK
REM*************************************************************************
REM  MOST PARTS OF THIS SUBROUTINE WAS EXTRACTED FROM Dr. REX HURST
REM  PROGRAM--FCT.BAS
9000 PRINT ""
     PRINT "PROCESSING COVARIANCE BLOCK" TAB(35)  "-- PLEASE WAIT --"
     FT="" :FT=FLN :FT=FT+"MOD.TMP" :OPEN "I",#2,FT
     FT="" :FT=FLN :FT=FT+"CSS.OUT" :OPEN "I",#5,FT
     FT="" :FT=FLN :FT=FT+"COV.OUT" :OPEN "O",#3,FT
     FT="" :FT=FLN :FT=FT+"TOT.OUT" :OPEN "I",#6,FT
     FT="" :FT=FLN :FT=FT+"REG.OUT" :OPEN "O",#4,FT
     IF NT > 0 THEN
       FOR I = 1 TO NT :PRINT #3,CTITLE(I) :NEXT I :PRINT #3,""
     PRINT #3, TAB(12) "ANALYSIS OF COVARIANCE RESULTS"
     PRINT #3, TAB(12) "------------------------------"
     INPUT #2,NMODEL
     NM=1
9022 IF EOF(2) THEN GOTO 9035
      PRINT #3,"" :PRINT #3,TAB(23) "MODEL " TAB(29) NM
      PRINT #3,TAB(21) "-----------" :PRINT #3,""
      INPUT #2,NX,NY
        NSV=NX+NY
        FOR K = 1 TO NSV
          INPUT #2,ID(K)
        NEXT K
        CLOSE #5
        FT="" :FT=FLN :FT=FT+"CSS.OUT" :OPEN "I",#5,FT
        N=1
9024    LINE INPUT #5,CMNAME
```

```
            INPUT #5,MSIGN,NDFLINE,IDTOT
            L=1
            FOR I = 1 TO NV
              NSWI=0
              K=1
              WHILE ((NSWI = 0) AND (K <= NSV))
                IF (I = ID(K)) THEN II=K :NSWI=1
                K=K+1
              WEND
              FOR J = I TO NV
                INPUT #5,TOT(L)
                IF NSWI = 0 THEN GOTO 9030
                NSWJ=0
                K=1
                WHILE ((NSWJ = 0) AND (K <= NSV))
                  IF (J = ID(K)) THEN JJ=K :NSWJ=1
                  K=K+1
                WEND
                IF NSWJ = 0 THEN GOTO 9030
                IF (II < JJ) THEN A(II,JJ)=TOT(L) ELSE A(JJ,II)=TOT(L)
9030            L=L+1
              NEXT J
            NEXT I
            N1=1
            N2=NX
            N3=NX+1
            NY=NY
            FOR I = 1 TO NX
              FOR J = NX+1 TO NSV
                A(J+1,I)=A(I,J)
              NEXT J
            NEXT I
            GOSUB 6000
            IF (MSIGN < 0) THEN GOSUB 7500 ELSE GOSUB 8000
```

```
        N=N+1
      IF (N <= NCM) THEN GOTO 9024
      NM=NM+1
    IF (NM <= NMODEL) THEN GOTO 9022
    CLOSE #2 :FT="" :FT=FLN :FT=FT+"MOD.TMP" :KILL FT
    CLOSE #3
9035  RETURN
REM*************************************************************************
REM SUBROUTINE DMATIV
REM*************************************************************************
REM   THIS SUBROUTINE WAS EXTRACTED FROM Dr. REX HURST PROGRAM--FCT.BAS
6000  TDET=1.0
    TEST=1.0E-10
    NK=N3+NY-1
    FOR L = N1 TO N2
      IF ABS(A(L,L)) < TEST THEN_
        TDET=0.0 :TRECIP=0.0 :_
      ELSE_
        TDET=TDET*A(L,L) :TRECIP=1.0/A(L,L)
      FOR I = N1 TO N2
        IF I < L THEN TR=A(I,L)*TRECIP
        IF I > L THEN TR=A(L,I)*TRECIP
        IF I = L THEN TR=0.0
        FOR J = I TO N2
          IF I < L THEN A(I,J)=A(I,J)-TR*A(J,L)
          IF J > L THEN A(I,J)=A(I,J)-TR*A(L,J)
        NEXT J
        IF NY > 0 THEN_
          FOR J = N3 TO NK :_
            A(I,J)=A(I,J)-TR*A(L,J) :_
          NEXT J
        IF I < L THEN A(I,L)=TR
        IF I > L THEN A(L,I)=TR
        IF I = L THEN A(L,L)=-TRECIP
```

```
        NEXT I
        IF NY > 0 THEN_
          FOR J = N3 TO NK :_
            A(L,J)=A(L,J)*TRECIP :_
          NEXT J
      NEXT L
      FOR I = N1 TO N2
        FOR J = I TO N2
          A(I,J)=-A(I,J)
        NEXT J
      NEXT I
      RETURN
REM*************************************************************************
REM SUBROUTINE ERROR CALCULATIONS
REM*************************************************************************
REM   THIS SUBROUTINE WAS EXTRACTED FROM Dr.REX HURST PROGRAM--FCT.BAS
7500  CLOSE #4
      FT="" :FT=FLN :FT=FT+"REG.OUT" :OPEN "O",#4,FT
      FOR J = NX+1 TO NSV
        FOR I = 1 TO NX
          PRINT #4,A(I,J)
        NEXT I
      NEXT J
      NERRDF=NDFLINE-NX
REM   PRINT AND SAVE INVERSE IN A'
      PRINT #3,"" :PRINT #3,"INVERSE MATRIX" :PRINT #3,"--------------"
      FOR I = 1 TO NX
        PRINT #3,ID(I);
        FOR J = I TO NX
          PRINT #3,A(I,J);
          IF ((J-I+1) MOD 5 = 0) THEN PRINT #3,"" :PRINT #3,"       ";
          A(J+1,I)=A(I,J)
        NEXT J
        IF (NX-I+1) MOD 5 <> 0 THEN PRINT #3,""
```

```
        NEXT I
REM     PRINT REGRESSION COEFFICIENTS
        PRINT #3,"" :PRINT #3,"REGRESSION COEFFICIENTS"
        PRINT #3,"-----------------------"
        FOR I = 1 TO NX
          PRINT #3,ID(I);
          FOR J = NX+1 TO NSV
            PRINT #3,A(I,J);
            IF ((J-NX) MOD 5 = 0) THEN PRINT #3,"" :PRINT #3,"      ";
          NEXT J
          PRINT #3,""
        NEXT I
REM     PRINT AND STORE SSSP DUE TO REG
        PRINT #3,"" :PRINT #3,"SSSP/MSMP DUE TO REGRESSION, DF= ";NX
        PRINT #3,"---------------------------"
        FOR J = NX+1 TO NSV
          FOR I = J+1 TO NSV+1
            A(I,J)=0.0
            FOR K = 1 TO NX
              TEMP=A(I,K)*A(K,J)
              A(I,J)=A(I,J)+TEMP
            NEXT K
            PRINT #3,ID(J-NX); ID(I-NX-1); A(I,J); A(I,J)/NX
            A(I,J)=A(J,I-1)-A(I,J)
          NEXT I
        NEXT J
        PRINT #3,"" :PRINT #3,"RESIDUAL SSSP/MSMP, DF= ";NERRDF
        PRINT #3,"------------------"
        FOR J = NX+1 TO NSV
          FOR I = J+1 TO NSV+1
            PRINT #3,ID(J-NX); ID(I-NX-1); A(I,J); A(I,J)/NERRDF
          NEXT I
        NEXT J
        RETURN
```

```
REM*************************************************************************
REM SUBROUTINE TREATMENT PLUS ERROR CALCULATIONS
REM*************************************************************************
REM    THIS SUBROUTINE WAS EXTRACTED FROM DR.REX HURST PROGRAM--FCT.BAS
8000   NTRTDF=NDFLINE-NX-NERRDF
       PRINT #3,"" :PRINT #3,CMNAME
       PRINT #3,"----------------------------------------"
       PRINT #3,"TREATMENT(ADJ) SSSP/MSMP, DF= ";NTRTDF
       FOR J = NX+1 TO NSV
         FOR I = J+1 TO NSV+1
           T1=0.0
           FOR K = 1 TO NX
             TEMP=A(I,K)*A(K,J)
             T1=T1+TEMP
           NEXT K
           T2=A(J,I-1)-T1-A(I,J)
           PRINT #3,ID(J-NX); ID(I-NX-1); T2; T2/NTRTDF
         NEXT I
       NEXT J
REM    COMPUTE ADJUSTED TREATMENT DEVIATIONS
       CLOSE #4
       FT="" :FT=FLN :FT=FT+"REG.OUT" :OPEN "I",#4,FT
       FOR J = NX+1 TO NSV
         FOR I = 1 TO NX
           INPUT #4,A(I,J)
         NEXT I
       NEXT J
       CLOSE #4
       CLOSE #6
       FT="" :FT=FLN :FT=FT+"TOT.OUT" :OPEN "I",#6,FT
       INPUT #6,NTOT,NDIV
       FOR I = 1 TO NV
         INPUT #6,TOT(I)
         TOT(I)=TOT(I)/NDI
```

```
      NEXT I
      FOR I = 1 TO NSV
        X(I)=TOT(ID(I))
      NEXT I
      IF (IDTOT <= 3) THEN GOTO 8035
      FOR K = 3 TO IDTOT-1
        INPUT #6,NTOT,NDIV
        FOR I = 1 TO NTOT
          FOR J = 1 TO NV
            INPUT #6,TOT(J)
          NEXT J
        NEXT I
      NEXT K
8035  PRINT #3,""
      PRINT #3,"TREATMENT DEVIATIONS, ADJ DEVIATIONS, STD ERROR
      INPUT #6,NTOT,NDIV
      FOR1="### #### ### #.########^^^^ #.########^^^^ #.########^^^^"
      FOR M = 1 TO NTOT
        FOR I = 1 TO NV
          INPUT #6,TOT(I)
          TOT(I)=TOT(I)/NDIV
        NEXT I
        FOR I = 1 TO NSV
          S(I)=TOT(ID(I))-X(I)
        NEXT I
        T1=0.0
        FOR I = 1 TO NX
          FOR J = I TO NX
            IF (I = J) THEN T1=T1+S(I)*S(I)*A(I+1,I) ELSE_
              T1=T1+2.0*S(I)*S(J)*A(J+1,I)
          NEXT J
        NEXT I
        FOR J = 1 TO NY
          T2=0.0
```

```
            FOR I = 1 TO NX
              T2=T2+A(I,J+NX)*S(I)
            NEXT I
            TEMP=A(NX+1+J,NX+J)/NERRDF
            TEMP=SQR(TEMP*(1.0/NDIV+T1))
            PRINT #3,USING FOR1; M; ID(J+NX); NDIV; S(NX+J);_
              (S(NX+J)-T2); TEMP
          NEXT J
        NEXT M
        RETURN
REM*************************************************************************
```