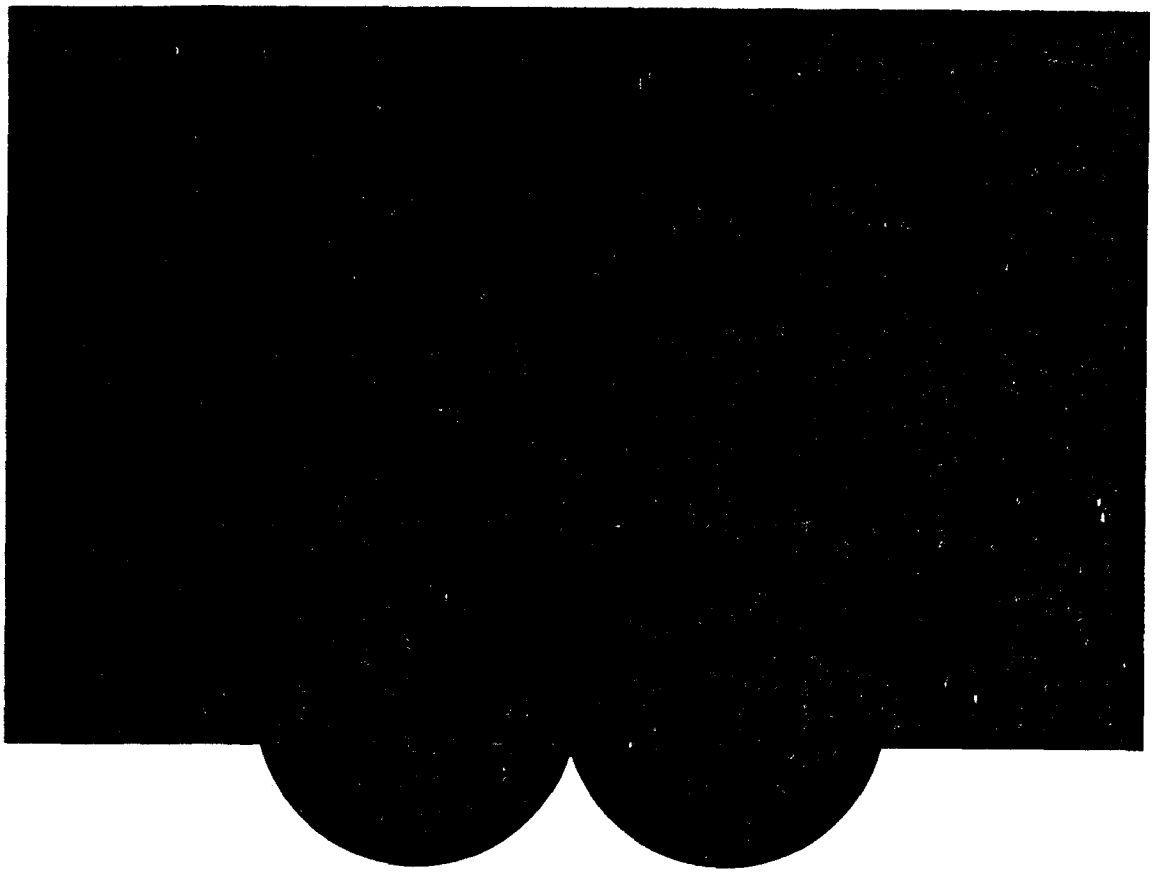

A COMPREHENSIVE, MATRIX FREE ALGORITHM FOR ANALYSIS OF VARIANCE
PART I

by

William J. Hemmerle
University of Rhode Island

Institute of Statistics Mimeograph Series No. 1210
February 1979

NORTH CAROLINA STATE UNIVERSITY
Raleigh, North Carolina

A COMPREHENSIVE, MATRIX FREE ALGORITHM FOR ANALYSIS OF VARIANCE

WILLIAM J. HEMMERLE
University of Rhode Island

A storage efficient algorithm is developed for the analysis of balanced data and unbalanced data, including data with missing cells, in an analysis of variance situation. The algorithm may be used for either estimation or hypothesis testing related to an algebraically specified statistical model. Balanced analysis of variance operators are applied iteratively, when necessary, to obtain exact statistical results for unbalanced data. As a consequence, the algorithm has the ability to process large analysis of variance problems involving unbalanced data on small computers.

Key Words and Phrases: linear models, analysis of variance, unbalanced data, missing cells, estimation, hypothesis testing, storage efficient algorithm, iterative improvement, balanced data operators, algebraic model specification, rank computations, G-inverse solution.
CR Categories: 5.5, 5.14

1. INTRODUCTION

Some of the work to be described is reminiscent of AARDVARK [14], an analysis of variance package, at Iowa State University in the early sixties. This program was used for most of the large number of analyses processed at Iowa State at that time; however, the program was definitely not transportable, being written in a mix of assembler and FORTRAN for a non-standard IBM 7074. Subsequent versions, lacking some of its initial capabilities but adding others, were prepared for other machines including the IBM 360 (see for example [11]). One distinctive feature of the program, that greatly aided in its useability, was algebraic specification of the statistical model by the user [13], [18]. This facility has now become common in statistical packages.

The initial AARDVARK relied principally on balanced analysis of variance algorithms and approximate statistical methods were applied to treat unbalanced data. An iterative A.O.V. algorithm was latter developed by Hemmerle [10], which permits using balanced analysis of variance algorithms to obtain exact statistical results for unbalanced data. This algorithm along with subsequent work will be exploited in our development of a comprehensive analysis of variance algorithm for balanced or unbalanced data.

This research was partially supported by the National Science Foundation under Grants DCR74-15331 and MCS74-15331 A01 and by the Department of Statistics, N. C. State University during a period of sabbatical leave.
Author's address: Department of Computer Science and Experimental Statistics, University of Rhode Island, Kingston, RI 02881.

Balanced analysis of variance algorithms frequently represent a level of elegance of logic that is difficult to surpass algorithmically. One fine example is the algorithm due to H. O. Harley [2] which obtains a complete factorial decomposition. Another is Wilkinson's recursive algorithm [23]. Unfortunately, these algorithms are also frequently difficult to explain using straightforward notation; they are likely to involve matrices with complicated structures implicitly rather than explicitly. Balanced analysis of variance algorithms normally do not require the coding of indicator variables, either externally or internally by the program. They possess the virtue of efficiency, both with respect to the number of operations performed and the amount of storage required.

Data on the other hand is frequently unbalanced and is likely to have missing cells. Both theoretically and computationally, the order of complexity of analysis of variance problems increases down the following categories:

1) balanced data;
2) unbalanced data with no missing cells;
3) unbalanced data with missing cells.

In some cases, the unbalanced data arises from a balanced design in which some of the observations are missing as a result of the experiment. Computationally, there are a number of missing data algorithms, notably Healy and Westmacott's iterative algorithm [4] and Rubin's non-iterative algorithm [16], which essentially fill in the missing values for use with balanced algorithms. One could also use dummy covariates for each missing cell and then compute a balanced analysis of covariance. These approaches have the disadvantage that, in general, they require balancing all of the data (including the missing observations) to have equal cell frequencies.

Almost all algorithms which seek to process unbalanced data with any degree of generality in an analysis of variance situation, including those in statistical packages, are matrix oriented least squares or regression algorithms. These involve the creation of indicator variables, to cast the analysis of variance model as a regression model, and the formation of the design matrix X_0 or the coefficients matrix of the normal equations $X_0'X_0$. The normal equations are then solved using either orthogonalization or elimination methods. In some cases, SAS Procedure GLM [1] for example, an explicit G-inverse is obtained for $X_0'X_0$. One problem with the matrix oriented approach is that the dimensions of X_0 or $X_0'X_0$ can be overwhelming for a moderate to large analysis of variance problem. This approach is also extremely inefficient when the data is balanced.

2. FEATURES AND DESIGN OBJECTIVES

In what follows, we will describe a global algorithm for the analysis of variance with the following features:

1) Balanced data, unbalanced data, and unbalanced data with missing cells are all processed by the algorithm. This is accomplished without losing the operational efficiencies obtainable from balanced data and without applying approximate statistical methods to the unbalanced data;

2) The algorithm is very general with respect to the kind of problem it can handle. Specifically, it bases its calculations upon an algebraically specified analysis of variance model of the type discussed in Searle [19]. This includes models with crossed factors, nested factors and interactions between factors. This general model, along with the facility to handle missing cells also includes such designs as incomplete blocks, lattices, and Latin squares;

3) Very large problems may be processed using a relatively small amount of computer storage. With one minor exception, no matrices are stored and no explicit matrix operations are performed. In particular, neither X_0 or $X_0' X_0$ are stored or computed. An exact G-inverse solution to the normal equations is obtained without ever computing a G-inverse. The rank of the design matrix X_0 is obtained from the pattern of missing cells without explicit operations on X_0;

4) The algorithm provides a heuristic optimum of maximizing analysis of variance capabilities while minimizing lines of source code. Since a principle attribute of the algorithm is its limited demands for array storage, an effort was made to also constrain program storage through a judicious selection of functional capabilities and the production of tight code. Consequently, the algorithm should be well-suited for use on small computers or for inclusion in interactive systems with storage restraints;

5) The user is given reasonable facilities in specifying his analysis of variance problem algebraically and various options alpha-numerically in an essentially format free manner;

6) The algorithm is well-suited to interactive usage and for inclusion in interactive systems. It was developed using CALL/OS at URI and TSO at N. C. State. It may, of course, also be used as a batch program. A mode parameter, interactive or batch, determines the nature of prompting, I/O, and error returns depending upon the mode selected;

7) The program has been written in ANS (basic) FORTRAN for portability. The program has been verified by the Bell Laboratories PFORT verifier [17] and will run on any computer supported with a FORTRAN compiler.

Along with the effort to constrain program storage, steps were also taken to reduce the number of lines of source code. Lines of source (including COMMENT statements) can be a limiting factor with respect to file storage in interactive systems. Most of the following measures served to reduce program storage as well as lines of source code: limiting the number of specification statements, WRITE statements, FORMAT statements, and unnecessary CONTINUE statements; limiting the amount of constant alphanumeric information in FORMAT statements; using selective, concise COMMENT statements; accepting trivial operational inefficiencies which reduce lines of source code; avoiding subroutinization not contributing to operational efficiency or storage reduction; and using single character identifiers when possible. The portability objective essentially limits one to storage of a single alphanumeric character per alphanumeric constant or array location. Rather than use additional code to interpret multiple character identifiers, for such things as specified options, we limit these to one character.

As a consequence of these measures, the complete program for the global algorithm consists of 836 FORTRAN statements exclusive of COMMENT statements. The object code produced by the WATFIV compiler was contained in 36,656 bytes. In addition, the program was written in such a manner that several of the functional capabilities may easily be deleted, if necessary, to relax program storage.

3. CONSTITUENT ALGORITHMS

We have called the algorithm under discussion a global algorithm since it encompasses the use of several independent algorithms. The principal independent algorithms, which when interlaced or interwoven become the global algorithm,are the following:

3.1 The Iterative A.O.V. Algorithm

This algorithm applies expectation and residual operators (E/R operators) for balanced data iteratively to obtain exact results for unbalanced data. In other words, stated simply, the algorithm computes an analysis of variance for unbalanced data by successive computation of balanced analyses. Since the algorithm uses balanced E/R operators to obtain its results, there is no need to create indicator variables or form the design matrix X_0 or $X_0'X_0$. The algorithm operates upon a vector of cell sums and a vector of cell frequencies rather than upon a (potentially large) matrix so that array storage requirements are minimal. Missing cells are handled routinely and convergence of the algorithm is guaranteed. A useful monotonic property serves to limit iteration when testing hypotheses. A balanced analysis of variance is a special case requiring only one iteration.

The basic algorithm [10] is very general and applies to any analysis of variance; however one must know or determine the balanced E/R operators. It is in this context that one can apply much of the algorithmic lore associated with balanced analyses. An initial implementation using a relatively comprehensive class of balanced E/R operators was discussed by Hemmerle and Piette [9]. Many of the features of the global algorithm were included in this implementation; however certain unnecessary rank restrictions were placed upon the design matrix X_0. The latter restrictions were subsequently removed by Hemmerle [7] who showed that, with proper parameter selection, the iterative A.O.V. algorithm would obtain a G-inverse solution to the normal equations for any design matrix. An improved iterative approximation, which halves the number of iterations in testing hypotheses, is also given in [7] and is used here.

An iterative analysis of covariance algorithm was also developed by Hemmerle [8] as an extension of the Herative A.O.V. algorithm. Although this covariance algorithm adapts well to the logic of the global algorithm and is array storage efficient, covariance capabilities were excluded here principally because of the additional program storage requirements; however, this algorithm is available as a PFORT verified FORTRAN program [15].

3.2 An Iterative Rank Algorithm

With unbalanced data including missing cells, the ranks of the design matrix and restricted design matrix must be computed to determine degrees of freedom for F statistics. Algorithms have been included (section 3.5) to determine rank non-iteratively from the pattern of missing cells for a given model when possible. When this cannot be accomplished, a recent adaptation of the iterative A.O.V. algorithm is used to compute rank in one of two ways depending upon the number of missing cells. Either way will produce monotonically increasing convergence to the integer rank value. For a small number of missing cells we use some matrix storage (the exception cited earlier) and obtain monotonically increasing quadratic convergence. Both methods will be outlined in a subsequent section. A complete discussion is given in [6].

3.3 A Balanced Factorial Decomposition Algorithm

The technique for computing a balanced analysis of variance based upon a factorial decomposition of linear combinations of classification means dates back to H. O. Hartley [2].

We use the algorithm described in Hemmerle [12] (pp. 180-185) to obtain this decomposition. A one to one correspondence is maintained between the distinct arrays in this decomposition and the locations of an E/R list which we shall discuss shortly.

3.4 A Pooling Algorithm

In order to construct the E/R operators needed for the iterative A.O.V. algorithms, we must "pool" (add together in the proper manner) distinct arrays in the factorial decomposition as dictated by the particular analysis of variance model. The complete factorial decomposition will be stored in a single linear, one-dimensional, array and we must be able to map one of the distinct arrays within this decomposition into another. We use essentially the same algorithm described in Schlater and Hemmerle [18] to do the mapping and pooling; however, we pool arrays of the decomposition rather than classification means as do the latter.

3.5 Algorithms to Recognize Balance and Re-structure Data

A balanced analysis of variance is a special case for the iterative A.O.V. algorithm. When properly executed, final results will be obtained on the first iteration. Furthermore, iterative rank computations are unnecessary. With unbalanced data, there are also frequent situations in which some of the results may be obtained non-iteratively. An example would be a full factorial model; the rank of the design matrix, the sum of squares for error (SSE), the sum of squares for regression (SSR), and a G-inverse solution to the normal equations may all be obtained from a balanced analysis of variance on cell means [10]. Also, when testing an hypothesis involving unbalanced data, the data structure for the model restricted by the hypothesis frequently has fewer dimensions than it has for the full model. A simple restructuring of the data may facilitate obtaining results without iterating. Essentially, this involves creating a surrogate vector of cell frequencies based upon reduced dimensions for the same data; however, duplicate entries are made in this surrogate vector to maintain a proper one to one correspondence with the original vector of cell sums.

Several algorithms have been incorporated into the global algorithm to recognize balance and restructure data, when appropriate, to avoid unnecessary iteration. The related theory and a description of the algorithms will be found in [5].

3.6 An E/R List Construction Algorithm

Numeric calculations in the global algorithm are driven by what we have called an E/R list. If there are n factors in the full analysis of variance model, then the E/R list will have 2^n entries corresponding to the 2^n terms (mean, main effects, and interactions) in a full factorial model. The E/R list is constructed by scanning or parsing an algebraic model statement. Numeric entries are made in the list which uniquely describe the model. Following the procedure suggested by Hemmerle [13], decreasing powers of two are assigned as numerical values to factor symbols and their associated subscripts, with unity for the last factor and its associated subscript.

The sum of the numerical values of the factor symbols (plus one) is computed for each term in the model. For crossed factors or interactions between crossed factors, this sum is entered in the corresponding location of the E/R list. For nested factors or interactions involving nested factors, multiple entries are made in the E/R list. The sum is entered into each location of the E/R list which corresponds to an array in the factorial decomposition which must be pooled (see [12] p. 174-176). The algorithm determines these locations from the numerical values of the factor symbols and subscripts in the model term. Examples will be given later in the sequel.

An hypothesis statement indicates those terms in the full model which should be deleted in forming the reduced model. As this statement

is parsed, the E/R list entries for the model terms in the statement are made negative. The R operator is then always formed by pooling the arrays in the factorial decomposition associated with zero or negative E/R list entries.

4. STATISTICAL FOUNDATIONS AND COMPUTATIONS

Analysis of variance models tend to create confusion because of the fact that, in the form that they are usually written, these models are over-parametrized. For example, the two-way classification with interaction is usually written as

$$y_{ijk} = \mu + a_i + b_j + ab_{ij} + e_{ijk} \qquad (4.1)$$

$$i=1,\ldots,I,\ j=1,\ldots,J,\ k=1,\ldots,n_{ij}$$

With e_{ijk} distributed $N(o,\sigma^2)$ and σ^2 unknown. None of the parameters μ, a_i, b_j, or ab_{ij} in (4.1) are estimable (for any i or j) even though the data may be balanced. That is, although the normal equations for (4.1) are consistent, there is no unique solution for these equations. We may obtain a unique solution by imposing constraints or side conditions upon the solution. These conditions have no effect upon what is estimable from the data for the model; however, they do have a bearing upon the hypothesis that is tested by applying the same applicable conditions to a reduced model.

Hocking and Speed [3] initially demonstrated the confusion surrounding this issue by exhibiting some of the unusual hypotheses tested by some of the procedures recommended in accepted statistical texts. Speed and Hocking [21] discuss the resulting difference in the application of Searle's R( ) notation [19] to the initial, overparametrized model as opposed to a reparametrized model obtained by imposing conditions on the

original model. They refer to the latter as R*( ) or procedure 2. Speed et al. [22] survey the hypotheses being tested by current computational methods and computer programs. A significant by-product of this work is that, when using procedure 2, the unweighted summation constraints (e.g. $\sum_i a_i = \sum_j b_j = \sum_i ab_{ij} = \sum_j ab_{ij} = 0$) yield the most likely hypotheses of interest, the classical Yates hypotheses [24].

The global algorithm uses procedure 2 with the unweighted summation conditions. In matrix form, the full reparametrized model is given by

$$y = X_0\beta + e \tag{4.2}$$

where: y is a vector of N observations.

X_0 is an $N \times p$ matrix of -1,0,1 indicator variables consistent with the unweighted summation conditions;

β is a vector of p parameters; and

the elements of e are distributed $N(0,\sigma^2)$.

Note that the p parameters in β will be linear combinations of the parameters appearing in the over parametrized model. As an example, consider the two-way classification without interaction. We have that

$$\beta' = (\mu + a_\cdot + b_\cdot,\ a_1 - a_\cdot, \ldots a_{I-1} - a_\cdot,\ b_1 - b_\cdot, \ldots,\ b_{J-1} - b_\cdot) \tag{4.3}$$

with the dot notation denoting a mean. If X_0 has full rank, then all of the elements of β are linearly independent estimable functions. A fact that sometimes causes confusion is that $a_I - a_\cdot$ and $b_J - b_\cdot$ are also estimable in the above example; however, they are not linearly independent of the elements in β. We mention this fact here since the global algorithm yields a G-inverse solution for the initial overparametrized model as

well as a G-inverse solution for (4.2). These solutions are the same except for the inclusion of additional non-linear independent elements in the former solution.

In order to consider hypothesis testing, we partition the design matrix X_0 for the full model as $[X_{01}|X_{02}]$, where X_{01} is N x k, and write the model (4.2) as

$$y = X_{01}\beta_1 + X_{02}\beta_2 + e \ . \tag{4.4}$$

The global algorithm will attempt to test the hypothesis

$$H_0: \beta_2 = 0 \tag{4.5}$$

by fitting the reduced model

$$y = X_{01}\beta_1 + e \ . \tag{4.6}$$

The hypothesis statement specifies the model terms which are to be deleted from the full model in forming the reduced model (4.6). For balanced data or unbalanced data with no missing cells, these are the same hypotheses tested, with F statistics, in a standard balanced analysis of variance table. A rule is given in [6], to determine whether or not the equivalent balanced data hypothesis is testable for unbalanced data with missing cells. For a model with all factors crossed, this rule is simply: the equivalent balanced data hypothesis is testable if and only if the rank of the full model minus the rank of the reduced model equals the degrees of freedom one would use for the hypothesis with balanced data. It is the author's opinion that when the equivalent balanced data hypothesis is not testable, then those hypotheses that are testable are usually very difficult to interpret in a meaningful way.

Some designs which are considered to be balanced designs will require iteration because the residual operator used applies to a more general

model. An example of this is a Latin square which the algorithm treats as a missing cell problem. Iteration will be necessary to obtain SSE for the full model; however, as a result of data restructuring, iteration is not required in testing the relevant hypotheses.

The following statistics may be obtained from the algorithm, most of them on an optional basis:

1) Cell sums, frequencies, and means

2) Classification sums, frequencies, and means

3) Rank (X_0)

4) SSE (full model) = $y'[I - X_0(X_0'X_0)^{-}X_0']y$ (invariant)

5) SSR (full model) = $y'X_0(X'X_0)^{-}X_0'y$

6) A G-inverse solution to $X_0'X_0\hat{\beta} = X_0'y$; $\hat{\beta} = (X_0'X_0)^{-}X_0'y$

7) Estimates of expected cell means = $X_0\hat{\beta}$ (invariant)

8) Rank (X_{01})

9) SSE (reduced model) = $y'[I - X_{01}(X_{01}'X_{01})^{-}X_{01}']y$

10) SSR (reduced model) = $y'X_{01}(X_{01}'X_{01})^{-}X_{01}'y$

11) $$F = \frac{(SSR(full) - SSR(reduced)))/(Rank(X_0) - Rank(X_{01}))}{SSE(full)/(N - Rank(X_0))}$$

12) Probability values; $Prob\{F|H_0\}$

We indicated earlier that a G-inverse solution is obtained for the over-parametrized model as well. It should be clear that the rank of X_0 is the same as the rank of the design matrix of the overparametrized model formed using 0, 1 indicator variables.

5. LOGICAL COMPONENTS

The program for the global algorithm consists of a main program and eight subroutines. The principal function of these components is described below.

MAINA processes the factor, level statement; computes cell sums, cell frequencies, and $y'y$ as it reads the data; processes the options statement and contains much of the code to execute the A (cell means) and C (classification means) options.

SCAN processes the model and hypothesis statement to construct (or modify) the E/R list; computes parameters needed in restructuring data; computes the degrees of freedom applicable to data with no missing cells.

IGET is used by the main program and subroutine SCAN to sequentially retrieve characters (other than blank, plus, or comma) from the input buffer.

PART1 restructures the data (cell frequencies) when appropriate; checks for balance and alternative non-iterative computations; computes rank non-iteratively if possible or iteratively otherwise when the R (rank) option is specified.

PART2 is the principal numeric computational component; with the use of the remaining four subroutines, it computes SSE, SSR, estimates of expected cell means, a G-inverse solution to the normal equations, F statistics, and probability values.

STEP performs one basic step of the iterative A.O.V. algorithm [10] using the improved approximation for SSR given in [7]. This basic step consists of the following sub-steps:

1) $A \leftarrow (Y - D \cdot V)/c$

2) $V \leftarrow V + A$

3) $B \leftarrow B + A$

4) $A \leftarrow R[A]$

5) $V \leftarrow V - A$

6) $S \leftarrow 2 \cdot Y'V - V'DV$

where:

Y is the vector of cell sums, D is the diagonal matrix of cell frequencies stored as a vector, and c is an algorithm constant set to assure convergence and, in hypothesis testing, to assure monotonicity of the approximation S to SSR;

A, B, and V are work vectors of size n_c with n_c being the number of cells. An additional work vector large enough to contain the factorial decomposition cited earlier is also required; however the vector A will occupy the first n_c locations of this additional vector;

R[A] is the residual operator applied to the vector A. Sub-step 4) above consists of a factorial decomposition of the vector A followed by pooling the appropriate arrays of the decomposition back into A.

The vectors B and V must be initially set to zero. Upon completion, perhaps after many steps, estimates of the expected cell means will be contained in V, and a G-inverse solution is obtained by applying the E operator to the vector B.

Sub-step 1) above is modified slightly for iterative rank computations and for restructured data. In computing rank, this takes the form

$$A \leftarrow (Y - \Delta V)$$

where Δ is a diagonal matrix with unit diagonals for filled cells and zero diagonals for missing cells. Operations with Δ are handled implicitly and this matrix is never stored. In performing non-iterative calculations (a single step) on restructured cell frequencies, sub-step 1) is modified to

$$A \leftarrow (D2^{-} \cdot Y - V)$$

where $D2^{-}$ is a diagonal G-inverse of the diagonal matrix of restructured cell frequencies [5].

Sub-step 6) above must also be modified in computing rank iteratively with Δ replacing D. Furthermore, in rank computations, the vector Y contains a dummy unit vector; cell sums have been temporarily stored elsewhere in a vacant vector.

DECOMP obtains a factorial decomposition of a given vector; determines the classification frequencies needed in PART1 to restructure data; computes classification means for the C option in MAINA.

POOL either moves the secondary array into the primary array, duplicating entries where needed, or it pools the secondary array into the primary array by addition.

LABEL calculates the array of coefficients for the array map needed in pooling; produces output labels for classification means and for the G-inverse solution.

Of these logical components, all of PART1 (162 statements) may be deleted to further reduce program storage without making PART2 inoperable; however, this would make the program less efficient due to unnecessary

iteration and full rank would be assumed by default for X_0 or X_{01}. Almost all of SCAN (155 statements) can also be deleted provided the user constructs and inputs his own E/R list.

6. ARRAY STORAGE

We previously emphasized the storage efficiency of the algorithm for moderate to large problems. Fundamentally, this efficiency is attributable to the use of this iterative A.O.V. algorithm; however, array storage economy was also stressed in constructing the global algorithm. The following is a brief description of all of the arrays included:

LSTFI is the array discussed in [12] and [18] which is used in formation and subsequent manipulation of the arrays in the factorial decomposition; this is an array of size 2^n where n is the number of factors.

LER is the E/R list which is also of size 2^n.

LE is an alphanumeric array for factor symbols of size n.

LS is an alphanumeric array of associated subscripts of size n.

LV is an array of the numerical values assigned to the factor symbols and their associated subscripts; it also has size n.

LLIM contains the number of levels for each factor and is of size n.

LT is a temporary work array of size n.

LP is also a work array; however, its size is 10 - the maximum number of factors handled by the program.

LD is an alphanumeric array of size 10 containing the digits 0 - 9; this array is used to convert alphanumeric input, such as the number of levels for a factor, to numeric.

LO is an alphanumeric array of size 10 containing the single character option identifiers.

IA is an alphanumeric input buffer - a card image - which is used in processing the factor, levels, options, model, and hypothesis statements and also for storage of a variable FORMAT statement for input data. (The model and hypothesis statements have continuation facilities.)

Q is the only two-dimensional array. It is used exclusively for rank computations and its size, in relation to the number of missing cells, determines the rank algorithm applied. If the dimension of Q were fixed at 10 x 10, then the quadratically convergent rank algorithm would be used only if there were no more than 10 missing cells.

QT is a vector whose size is the same as one of the dimensions of Q; it is used in conjunction with Q in computing rank.

W is a linear array which is used for all of the numeric computations. This array logically consists of 6 contiguous vectors. The implicit names of these vectors coincide with those used in describing a step of the iterative A.O.V. algorithm; they are ordered in storage as

Y, D, D2, V, B, A.

The first 5 of these vectors have size n_c where n_c is the number of cells. The 6th vector is the size of the factorial decomposition for the problem. For example, with two factors we have $n_c = I \cdot J$ while the vector A has

size (I+1)(J+1). Ordinarily, the array W would be allocated all of the remaining computer storage; the implicit variable dimensioning of W maximizes the number of analysis of variance problems that can be processed within a given amount of storage.

7. THE FACTOR, LEVELS STATEMENT

The factor, levels statement creates entries for the arrays LE, LS, LV, and LLIM. The array LSTFI is then formed from LLIM. Factor symbols, subscript symbols, and factor levels are included in the statement; the number of factors and their numerical values are determined. As an example of a factor, levels statement, consider an analysis involving 3 factors with levels 5, 10, and 15. If we decide to name the factors R, C, T with subscripts I, J, K then the statement would be written (typed/punched) on one line as:

F(R,C,T),L(I(5),J(10),K(15))

All of the commas above are cosmetic as are blanks and pluses; the same statement could be written as

F(RCT)L(I(5)J(10)K(15))

8. OPTIONS AND THE OPTIONS STATEMENT

A list of option identifiers and default values is given in Table 1. The first 3 identifiers include an argument enclosed within a parentheses which is used to modify a parameter. The remaining 7 identifiers trigger on/off switches; with the exception of the A and C option, these switches change their status each time the option is specified.

TABLE 1

Options

Identifier	Function of Option	Default
S(d)	Significant digits in results	d = 5
T($\ell \times 10^4$)	Test level	ℓ = .05
I(m)	Iteration maximum	m = 100
R	Rank computations (iterative)	off
V	Estimates of expected cell means	off
G	G-inverse solution to normal equations	off
P	Probability values for F statistics	off
Z	Intermediate output	off
A	Cell sums, frequencies, and means	off*
C	Classification sums, frequencies, and means	off*

The S option sets the relative iterative tolerance for SSR by forming the test constant $.05 \times 10^{-d} \times (y'y)$. Iteration ceases when two successive approximations to SSR differ in magnitude by less than this test constant. This approach in practice seems to provide at least the requested accuracy in results obtained iteratively. A specification of $d = 0$ results in the skipping of all calculations in PART2 and thus provides an option for independent rank calculations.

The T option sets the probability level at which F tests will be conducted. When iterative calculations are necessary for hypothesis testing, the constant c of the iterative A.O.V. algorithm [10] is selected such that the sequence of approximations to the F statistic will be monotonically decreasing. The approximation given in [20] is used to compute the probability level for the F approximation. Whenever this value falls below the level specified, iteration ceases due to lack of significance. In the event that this value stays above the level specified, iteration continues until two successive F approximations differ by less than .005. When the P option is specified, iteration is continued for nonsignificant F's as well in order to also obtain final probability levels for these statistics.

The R option must be in on status to compute rank <u>iteratively</u>. Non-iterative rank computations in PART1 are not affected by this option. The terminal iterative approximation to rank need not be highly accurate since the approximations converge monotonically to an integer; furthermore, absolute rather than relative cut-off tolerances may be used. Although the pre-set values for these tolerances may be modified, they have been tested to obtain the correct rank with minimal iteration.

The iterative rank method best suited to a relatively small number of missing cells powers a matrix whose order is the number of missing cells. The initial matrix is formed non-iteratively. Successive powers, 1, 2, 4, 8, ..., of this matrix are formed in the array Q. The trace of the matrix in Q converges quadratically and monotonically to the rank of X_0. In this case, we cease iteration when successive traces differ by less than .1. The rank is taken as the next integer. The method best suited to a large number of missing cells applies the iterative A.O.V. algorithm to dummy unit vectors for each filled cell with D replaced by Δ. The sum of the SSR's obtained for each unit vector will equal the rank of X_0. In this case we cease iteration when two successive approximations to SSR for a given unit vector differ by less than $.1/n_f$ where n_f is the number of filled cells. This method will be expensive time-wise when n_f is large.

When the Z option is on, the following intermediate output is produced when applicable: the current E/R list; the successive traces of the matrix in array Q or the cummulative sum of the SSR's for unit vectors in computing rank; the approximations to SSR for the full model; the approximations to SSR for the reduced model of the hypothesis.

Although the A and C options may appear to be somewhat mundane, these options require no additional array storage and they require little additional code; furthermore, it is likely that they would be requested as part of an analysis of variance.

An option statement is written on one line as a string of selected identifiers enclosed in parentheses and preceded by an O. The identifiers may be in any order. For example, with the R and Z options in off status, to specify 8 significant digits of accuracy, iterative rank computations,

and intermediate output, the options statement may be written as

O(R,S(8),Z)

where the commas are again cosmetic.

9. MODEL AND HYPOTHESIS STATEMENTS

Examples of the model and hypothesis statements for 3 factors are given in Table 2 along with the E/R list entries that would be produced for these statements. These examples assume that factor symbols A,B,C and subscript symbols I,J,K were used in the factor, levels statement. The analysis of variance model, as it is usually written, appears above each model statement in Table 2. Notice that the variate $y_{ijk\ell}$ and the error term $e_{ijk\ell}$ do not appear in the model statement. The additional subscript ℓ is also implicit; that is, the number of observations in the (i,j,k) cell depends upon the data. The rules for constructing the model statement basically parallel those used in [13] and [18]: the associated subscript of a factor must always appear along with the factor symbol in a model term; if a factor is nested, it must be nested within a factor or factors appearing to the left of it in the factor, levels statement; the ordering of the associated subscripts in the factor, levels statement must correspond with the ordering of the data. As indicated earlier, pluses, blanks, and commas serve only to improve the appearance of these statements; these characters are ignored in parsing. Should either the model or hypothesis statement require more than one line (or card), continuation to the next line is indicated by placing a slash (/) after the last model term in the current line. For example, the 4th model statement in Table 2 could be written as

```
M + A(I) + B(J) + AB(LJ)/
  + C(LJK)
```

TABLE 2

Model and Hypothesis Statements

Statements

$(y_{ijk\ell} = \mu + a_i + b_j + c_k + e_{ijk\ell})$

M + A(I) + B(J) + C(K)

H B(J)

ABC	AB	AC	A	BC	B	C	M
0	0	0	5	0	3	2	1
0	0	0	5	0	-3	2	1

$(y_{ijk\ell} = \mu + a_i + b_{ij} + c_{ijk} + e_{ijk\ell})$

M + A(I) + B(IJ) + C(IJK)

H C(IJK)

ABC	AB	AC	A	BC	B	C	M
2	3	2	5	2	3	2	1
-2	3	-2	5	-2	3	-2	1

$(y_{ijk\ell} = \mu + a_i + b_{ij} + c_k + ac_{ik} + e_{ijk\ell})$

M + A(I) + B(IJ) + C(K) + AC(IK)

H AC(IK)

ABC	AB	AC	A	BC	B	C	M
0	3	6	5	0	3	2	1
0	3	-6	5	0	3	2	1

$(y_{ijk\ell} = \mu + a_i + b_j + ab_{ij} + c_{ijk} + e_{ijk\ell})$

M + A(I) + B(J) + AB(IJ) + C(IJK)

H AB(IJ), C(IJK)

ABC	AB	AC	A	BC	B	C	M
2	7	2	5	2	3	2	1
-2	-7	-2	5	-2	3	-2	1

As a convenience, a full factorial model may be specified as M*, an asterisk following M. Hypothesis statements are then written normally using the same symbols appearing in the factor, levels statement. The E/R list construction algorithm includes the logic necessary to detect invalid models or hypotheses incorrectly specified. During interactive use, corrections can be made when an error is detected.

Any number of models may be applied experimentally to a given set of data. This includes models with fewer factors than appear in the factor, levels statement; the data will automatically be restructured for such models.

10. FLOW OF CONTROL

Flow of control proceeds in the following manner:

1) factor, levels statement
2) variable FORMAT statement
3) data
4) blank line (card)
5) options statement/model statement/hypothesis statement/ blank line/E

The data is prepared with one observation per line and cell indicators preceding the observation; the variable FORMAT statement must reflect this order. Cell indicators are the factor levels for the observation. There is no indicator for within cell replication and no entries are made for missing cells. A check on lexicographical ordering of indicators is included; however, this check is easily deleted if inconvenient.

Once the data has been entered, control is centered upon 5). Any number of options, model, or hypothesis statements may be entered in any order (except that a model statement must precede any hypothesis statements). Entering a blank line returns control to 1) to accept a factor levels

statement for a new data set. Entering the character E, for end, will terminate processing.

11. SUMMARY

We have described a comprehensive algorithm for analysis of variance which has the facility to handle large problems involving unbalanced data on small computers. At the same time, the computational efficiencies common to balanced data are realized. Array storage economy is achieved by virtue of the fact that the algorithm performs vector operations as opposed to matrix operations. Neither the design matrix of indicator variables X_0 nor the matrix $X_0'X_0$ is ever formed or stored. Ancillary computations and output amenities have been deliberately limited to conserve program storage; however, the facility for algebraic problem specification has been included. For some data sets and models requiring iteration, the algorithm may converge slowly so that it will be expensive time-wise to achieve a high degree of accuracy; however, performance studies of the iterative A.O.V. algorithm given in [7] and [9] indicate that the algorithm is likely to outperform matrix oriented algorithms time-wise on large problems with unbalanced data. It should always outperform these algorithms time-wise when the data is balanced.

ACKNOWLEDGEMENTS

The author wishes to acknowledge three former research assistants; Donald Piette, Clifford Pelletier, and James Petrarea. Their work in implementing the iterative A.O.V. algorithm and extensions of this algorithm to covariance and multivariate analysis contributed to the ultimate design of the algorithm presented here. The author also appreciates the efforts of Karen Hemmerle Hodge related to verifying the transportability of the program.

REFERENCES

1. BARR, A. J., GOODNIGHT, J. H., SALL, J. P., and HELWIG, J. T. A user's guide to SAS 76. SAS Institute Inc., Raleigh, N. C., 1976.
2. HARTLEY, H. O. A plan for programming analysis of variance for general purpose computers. Biometrics 12 (1956), 110-122.
3. HOCKING, R. R. and SPEED, F. M. A full rank analysis of some linear model problems. JASA 70 (1975), 706-712.
4. HEALY, M. and WESTMACOTT, M. Missing values in experiments analysed on automatic computers. Applied Statistics 5 (1956), 203-206.
5. HEMMERLE, W. J. Recognizing balance with unbalanced data. Communications in Statistics (to appear).
6. HEMMERLE, W. J. Balanced hypotheses and unbalanced data. TR 78-134, Dept. of Comp. Sci. and Exp. Stat., U. of Rhode Island, Kingston, R.I., 1978.
7. HEMMERLE, W. J. Extensions and improvements of recent linear model algorithms. Proc. ASA Statistical Computing Section 1976, 73-82.
8. HEMMERLE, W. J. Iterative nonorthogonal analysis of covariance. JASA 71 (1976), 195-199.
9. HEMMERLE, W. J. and PIETTE, D. F. An iterative algorithm for non-orthogonal analysis of variance. Proc. Computer Science and Statistics: 8th Annual Symposium on the Interface 1975, 191-195.
10. HEMMERLE, W. J. Nonorthogonal analysis of variance using iterative improvement and balanced residuals. JASA 69 (1974), 772-778.
11. HEMMERLE, W. J. and CARNEY, E. J. An algorithm for multivariate analysis of covariance (implemented in AARDVARK). In Statistical Computation, R. C. Milton and J. A. Nelder, Eds., Academic Press, NY, 1969.

12. HEMMERLE, W. J. Statistical Computations on a Digital Computer. Blaisdell, Waltham, Mass., 1967.

13. HEMMERLE, W. J. Algebraic specification of statistical models for analysis of variance computations. JACM 11 (1964), 234-239.

14. HEMMERLE, W. J., CARNEY, E. J., JOHNSON, A. F., MENSING, R. W., and SMITH, J. AARDVARK, a compiler-monitor system for analysis of variance. Reference Manual, Iowa State Statistical Laboratory, Ames, IA, 1963.

15. PELLETIER, C. L. A user's guide to ITNOAC, an implementation of Hemmerle's iterative algorithm for nonorthogonal analysis of covariance. Dept. of Comp. Sci. and Exp. Stat., U. of Rhode Island, Kingston, R. I., 1976.

16. RUBIN, D. B. A non-iterative algorithm for least squares estimation of missing values in any analysis of variance design. Applied Statistics 21 (1972), 136-141.

17. RYDER, B. G. and HALL, A. D. The FORTRAN verifier; User's guide. Computing Sci. Tech. Rep. 12, Bell Telephone Laboratories, Murray Hill, N. J., 1973, Rev. 1975.

18. SCHLATER, J. E., and HEMMERLE, W. J. Statistical computations based upon algebraically specified models. CACM 9 (1966), 865-869.

19. SEARLE, S. R. Linear Models. John Wiley and Sons, NY, 1967.

20. SMILLIE, K. W., and ANSTEY, T. H. A note on the calculation of probabilities in an F-distribution. CACM 7 (1964), 725.

21. SPEED, F. M. and HOCKING, R. R. The use of the R( ) notation with unbalanced data. The American Statistician 30 (1976), 30-33.

22. SPEED, F. M., HOCKING, R. R., and HACKNEY, O. P. Methods of analysis of linear models with unbalanced data. JASA 73 (1978), 105-112.

23. WILKINSON, G. N. A general recursive procedure for analysis of variance. Biometrica 57 (1970), 19-46.

24. YATES, F. The analysis of multiple classifications with unequal numbers in different classes. JASA 29 (1934), 52-66.